NORTH ATLANTIC TREATY ORGANISATION

RESEARCH AND TECHNOLOGY ORGANISATION

AC/323(IST-041)TP/27

www.rta.nato.int

RTO MEETING PROCEEDINGS **MP-IST-041**

Adaptive Defence in Unclassified Networks

(La défense adaptative pour les réseaux non classifiés)

Papers presented at the RTO Information Systems Technology Panel (IST) Symposium held in Toulouse, France, 19-20 April 2004.

Published November 2004

NORTH ATLANTIC TREATY ORGANISATION

RESEARCH AND TECHNOLOGY ORGANISATION

AC/323(IST-041)TP/27

www.rta.nato.int

RTO MEETING PROCEEDINGS **MP-IST-041**

Adaptive Defence in Unclassified Networks

(La défense adaptative pour les réseaux non classifiés)

Papers presented at the RTO Information Systems Technology Panel (IST) Symposium held in Toulouse, France, 19-20 April 2004.

The Research and Technology Organisation (RTO) of NATO

RTO is the single focus in NATO for Defence Research and Technology activities. Its mission is to conduct and promote co-operative research and information exchange. The objective is to support the development and effective use of national defence research and technology and to meet the military needs of the Alliance, to maintain a technological lead, and to provide advice to NATO and national decision makers. The RTO performs its mission with the support of an extensive network of national experts. It also ensures effective co-ordination with other NATO bodies involved in R&T activities.

RTO reports both to the Military Committee of NATO and to the Conference of National Armament Directors. It comprises a Research and Technology Board (RTB) as the highest level of national representation and the Research and Technology Agency (RTA), a dedicated staff with its headquarters in Neuilly, near Paris, France. In order to facilitate contacts with the military users and other NATO activities, a small part of the RTA staff is located in NATO Headquarters in Brussels. The Brussels staff also co-ordinates RTO's co-operation with nations in Middle and Eastern Europe, to which RTO attaches particular importance especially as working together in the field of research is one of the more promising areas of co-operation.

The total spectrum of R&T activities is covered by the following 7 bodies:

- AVT Applied Vehicle Technology Panel
- HFM Human Factors and Medicine Panel
- IST Information Systems Technology Panel
- NMSG NATO Modelling and Simulation Group
- SAS Studies, Analysis and Simulation Panel
- SCI Systems Concepts and Integration Panel
- SET Sensors and Electronics Technology Panel

These bodies are made up of national representatives as well as generally recognised 'world class' scientists. They also provide a communication link to military users and other NATO bodies. RTO's scientific and technological work is carried out by Technical Teams, created for specific activities and with a specific duration. Such Technical Teams can organise workshops, symposia, field trials, lecture series and training courses. An important function of these Technical Teams is to ensure the continuity of the expert networks.

RTO builds upon earlier co-operation in defence research and technology as set-up under the Advisory Group for Aerospace Research and Development (AGARD) and the Defence Research Group (DRG). AGARD and the DRG share common roots in that they were both established at the initiative of Dr Theodore von Kármán, a leading aerospace scientist, who early on recognised the importance of scientific support for the Allied Armed Forces. RTO is capitalising on these common roots in order to provide the Alliance and the NATO nations with a strong scientific and technological basis that will guarantee a solid base for the future.

Published November 2004

ISBN 92-837-0039-2

Adaptive Defence in Unclassified Networks

(RTO-MP-IST-041)

Executive Summary

Classified military networks are designed to maintain functionality under active attack. Their robustness comes from careful design, controlled isolation and purposely limiting functionality. Unclassified networked systems represent the vast majority of military information system and are very important for NATO by providing services such as email, logistics support, unclassified file transport, and access to the vast amount of information on the World Wide Web. Similarly, governments and the military operate web sites and other services that are of critical importance to the civil community, particularly in times of crisis. For military and government users ready access to external services can be pivotal to maintenance of efficient operations. Techniques employed to protect classified information systems for many applications are too limiting and costly to apply to unclassified applications where the benefits of reasonably open access greatly outweigh the potential risks. However, such systems still need protection, but not from loss of information so much as from denial of service and the potential for corruption of information. Balancing protection and capability is a complex problem. The goal of dynamically balancing access while limiting real damage is in many respects a more complex problem than absolute protection of more limited core services. This symposium addressed NATO interests and issues in operating unclassified computing systems and networks while maintaining desired levels of assurance.

The symposium provided an interdisciplinary forum for research scientists, military experts, and system engineers to present the state of the art of research and technology in the protection of unclassified computer systems and networks. The broad scope of topics was discussed by an excellent keynote speech and 21 interesting and well appreciated papers which were presented in 7 technical sessions to an audience of approximately 100 participants. The symposium opened with a keynote address by Professor Richard Kemmerer of the University of California. His address, entitled “Designing a Web of Highly Configurable Intrusion Sensors”, described a framework for the development of intrusion detection systems that overcomes the limitations of more traditional approaches that are often developed in an ad hoc manner for certain types of domains and are hard to configure, extend, and control remotely. This theme was carried over into a session on Intrusion Detection and Response that looked at issues of developing cooperative intrusion detection components in dynamic coalition environments, intrusion tolerance, and reactions. A paper from this session on selecting appropriate countermeasures was awarded the “Best Paper Award”.

A session on Coalition Networks looked at various methods for securing unclassified coalition networks, including software-engineering based approaches, Virtual LANs, and the application of Virtual Machines. Other sessions focused on specific aspects of this problem. For example, a session on Honeypots looked at the use of honeypot technology to gather information about attack processes that can be found on the Internet and at ways that honeypot systems can emulate database servers, and a session on Servers and Viruses looked at improved ways for deploying antivirus technology, ways to protect public servers, and the effects of outsourcing government web sites. A session on Network Technology looked at issues of vulnerability assessment, passive network discovery, and network monitoring, while a session on securing networks looked at ways to use commercial messaging software securely, ways to enforce security policies in heterogeneous environments, technology to use public networks without being subject to traffic analysis, and ways to enforce security policies on secure socket layer connections. A final session addressed the problem of safely using COTS by breaking up system monocultures and by building trusted paths within COTS components.

La défense adaptative pour les réseaux non classifiés

(RTO-MP-IST-041)

Synthèse

Les réseaux militaires classifiés sont conçus de façon à conserver leur intégrité opérationnelle, même en cas d'attaque directe. Leur robustesse est le résultat d'une conception soignée, d'une isolation contrôlée et d'un nombre limité de fonctionnalités. Les systèmes en réseaux non classifiés représentent la grande majorité des systèmes d'information militaires et sont très importants pour l'OTAN, puisqu'ils fournissent des services tels que le courrier électronique, le soutien logistique, la transmission de fichiers non classifiés et l'accès à l'énorme quantité d'informations disponibles sur la toile mondiale. De la même façon, les gouvernements et les militaires exploitent des sites Web ainsi que d'autres services qui sont d'une importance vitale pour les civils, en particulier en temps de crise. Pour ces utilisateurs, l'accès direct à des services externes peut-être déterminant pour le maintien d'opérations efficaces. Pour bon nombre d'applications non classifiées, les techniques utilisées en vue de la protection des systèmes d'information classifiés sont trop contraignantes et trop coûteuses, car, dans ces cas, les avantages que comporte un accès relativement libre l'emportent largement sur les risques éventuels. Cependant, ces systèmes ont encore besoin de protection, non pas contre la perte d'informations, mais plutôt contre le refus de services et la possibilité d'altération des données. Concilier protection et capacité est un problème complexe. A bien des égards, l'objectif qui consiste à concilier ces éléments de façon dynamique, tout en limitant les dommages réels, est un problème plus complexe que celui de la protection intégrale de certains services essentiels plus restreints. Ce symposium a fait le point sur les intérêts de l'OTAN, ainsi que sur les problèmes qu'elle rencontre pour assurer l'exploitation de systèmes et de réseaux informatiques non classifiés dans des conditions de sécurité acceptables.

Ce symposium a servi de forum interdisciplinaire, permettant aux chercheurs scientifiques, aux spécialistes militaires, et aux ingénieurs systèmes de présenter l'état actuel de la recherche et de la technologie dans le domaine de la protection des systèmes et des réseaux informatiques non classifiés. L'ensemble des sujets a été abordé dans un excellent discours d'ouverture, ainsi que dans 21 communications intéressantes et bien accueillies, présentées au cours de 7 sessions techniques devant une assistance d'une centaine de participants. Le symposium a débuté par un discours d'ouverture prononcé par le Professeur Kemmerer, de l'Université de Californie. Dans sa présentation, « La conception d'un réseau de capteurs d'intrus facilement configurable », il a fourni la description d'un cadre pour le développement de systèmes de détection d'intrus qui s'affranchit des limitations imposées par les systèmes plus classiques qui sont souvent développés pour la circonstance au profit de certains types de domaines et qui sont difficiles à configurer, à améliorer et à télécommander. Ce thème a été poursuivi lors d'une session sur la détection et la réponse aux intrus, couvrant le développement en coopération de dispositifs de détection d'intrus en environnement dynamique de coalition, la tolérance aux intrus et les réactions. L'une des présentations de cette session a été désignée « meilleure communication du symposium ».

La session sur les réseaux de coalition a permis d'examiner différentes méthodes de sécurisation des réseaux de coalition non classifiés, y compris des approches basées sur le génie logiciel, les LAN virtuels et la mise en œuvre de machines virtuelles. D'autres sessions ont privilégié des aspects spécifiques de ce problème. Par exemple, une session sur les technologies de pointe a examiné leur capacité de rassembler des informations sur les mécanismes d'attaque affichés sur l'Internet, ainsi que leurs possibilités d'émulation de serveurs de bases de données. Une autre session sur les serveurs et les virus a examiné les nouvelles applications des technologies antivirus, la protection des serveurs grand public et les conséquences de la sous-traitance des sites web gouvernementaux. Une session sur les technologies des réseaux a permis d'étudier des questions concernant l'évaluation de la vulnérabilité, la découverte des réseaux passifs, et le contrôle des réseaux. Une autre session, sur la sécurisation des réseaux, a examiné la sécurisation des logiciels de messagerie du commerce, la mise en application de politiques de sécurité en environnement hétérogène, les technologies permettant d'exploiter les réseaux publics sans être soumis aux analyses de trafic et la mise en application de politiques de sécurité concernant les connexions sécurisées au niveau de la couche « socket ». La session finale a examiné une approche du problème de la mise en œuvre de systèmes COTS en toute sécurité, qui consiste à décomposer les monocultures inhérentes aux systèmes pour ensuite créer des chemins fiables au sein des composants COTS.

Table of Contents

SESSION III – HONEYPOTS
Chairman: Dr. G. EIZENBERG, FR

SESSION IV – SERVERS AND VIRUSES
Chairman: Mr. P. CHAUVE, FR

SESSION V – NETWORK TECHNOLOGY
Chairman: Dr. J. LEFEBVRE, CA

SESSION VI – SECURING NETWORKS
Chairman: Mr. G. HALLINGSTAD, NO

SESSION VII – DEALING WITH COTS
Chairman: Dr. R. SHUMAKER, US

Information Systems Technology Panel

ACKNOWLEDGEMENTS/REMERCIEMENTS

The IST Panel wishes to express its thanks to the French RTB members to RTA for the invitation to hold this Symposium in Toulouse and for the facilities and personnel which made the Symposium possible.

Le Panel IST tient à remercier les membres français du RTB auprès de la RTA de leur invitation à tenir cette réunion à Toulouse, ainsi que pour les installations et le personnel mis à sa disposition.

Introduction

Yves CORREC
DGA/DCE/CELAR
BP7, 35998 Rennes Armées
France

yves.correc@dga.defense.gouv.fr

There is little doubt that information systems have gradually become the electronic brain and nerves of our modern (best possible??) world, thanks to the so-called silicon revolution of the last half century.

While making a biological analogy, one must bear in mind that these systems can be disrupted by computer attacks, just the same way our neurocortical functions can be impaired by the use of very small quantities of the right (neurotoxic) stuff. On the path of evolution, computer systems are currently mutating from the fortress model to the living being model: At dawn of computer history, expensive systems were dedicated to critical high value functions, and their protection was quite simple. Tomorrow, pervasive computing will extend our capabilities up to shockingly high levels: our brand-new computer-aided washing machine, or emergent Network Centric Warfare concepts are faint early beginnings. But security will become a correspondingly trickier concern, and rely on complex mechanisms. Yet an evolution of this magnitude must go through intermediate stages, with smaller and smaller interconnected fortresses… A fairly good view of the present stage can be found in the IATF (Information Assurance Technical Framework) document, issued by the NSA.

What can be said about the threat? The publicized threat is on line hacking of computers, with strong underlying assumptions of connectivity and restricted targets… The actual threat is unfortunately related to more physical assets (political, economical, military) and their potential value for the foe, through the whole spectrum of available means. In this present chapter of mankind history, we are just transposing usual human behaviour in a new space. It will require an evolution of security concepts and tools to keep it civilized.

IATF information infrastructure model connects local computing environments through enclave boundaries and networks. Some of them are classified, and accordingly protected. The others are not. Classified networks are supposedly designed to support noble functions, and carry confidential information (a paper legacy). Hard regulatory constraints often result in hardened more or less dedicated systems. But we cannot be unaware of some discretionary aspects of homologation and classification processes. Unclassified networks on the other hand deal with everything else, with a strong need for availability and integrity of the services. COTS software and hardware are widely used in both cases (with some precautionary measures –GOTS- for classified networks) for evident economical reasons.

The existence of various (and secure of course) gateways between all our networks must then be acknowledged, to conclude they are in fact part of a weakly segmented technical continuum. This blurred boundary between classified and unclassified networks calls for a common technological framework, including the full set of security mechanisms (protection, deception, IDS, monitoring & analysis, etc). In this respect, unclassified networks may be considered as a testbed for advanced technologies.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Let us now think of the security engineering process, as set out in IATF. A key component is effectiveness assessment, which unfolds in operational effectiveness (required functionalities) and security matters (risk analysis). These concepts are unfortunately too often opposed (security versus capability). The question can be settled if we are able to quantify security, but this very problem has long been some sort of a security Holy Grail…

So we cope now with the strong current trend towards ubiquituous computing, and a basically ill-posed problem, by using a nice combination of technology and organization, which we grant more empirical faith than theoretical proofs. Possible improvements may come from a more systematic use of models, allowing for a better understanding of systems behaviour in an operational environment. We give a simple example of a layered model accounting for a better definition of infowar concepts (cyberwar analysis grid) and subsequent works (deriving attack paths from the so-called foe's hopscotch).

Introduction

Professor Ann Miller
Department of Electrical and Computer Engineering
University of Missouri – Rolla
Rolla, Missouri 65409-0040
USA

milleran@umr.edu

This paper was received as a PowerPoint presentation without supporting text.

Click here to view PowerPoint presentation; Press Esc to exit

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Designing and Implementing a Family of Intrusion Detection Systems

Richard A. Kemmerer
Reliable Software Group
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
USA

kemm@cs.ucsb.edu

Intrusion detection systems (IDSs) analyze information about the activities performed in a computer system or network, looking for evidence of malicious behavior. Attacks against a system manifest themselves in terms of events. These events can be of a different nature and level of granularity. For example, they may be represented by network packets, operating system calls, audit records produced by the operating system auditing facilities, or log messages produced by applications. The goal of intrusion detection systems is to analyze one or more event streams and identify manifestations of attacks.

The intrusion detection community has developed a number of different tools that perform intrusion detection in particular domains (e.g., hosts or networks), in specific environments (e.g., Windows NT or Solaris), and at different levels of abstraction (e.g., kernel-level tools and alert correlation systems). These tools suffer from two main limitations: they are developed *ad hoc* for certain types of domains and/or environments, and they are difficult to configure, extend, and control remotely.

In the specific case of signature-based intrusion detection systems the sensors are equipped with a number of attack models that are matched against a stream of incoming events. The attack models are described using an *ad hoc*, domain-specific language (e.g., N-code, which is the language used by the Network Flight Recorder intrusion detection system). Therefore, performing intrusion detection in a new environment requires the development of both a new system and a new attack modeling language. As intrusion detection is applied to new and previously unforeseen domains, this approach results in increased development effort.

Today's network are not only heterogeneous, but also dynamic. Therefore, intrusion detection systems need to support mechanisms to dynamically change their configuration as the security state of the protected system evolves. Most existing intrusion detection systems are initialized with a set of signatures at startup time. Updating the signature set requires stopping the IDS, adding new signatures, and then restarting execution. Some of these systems provide a way to enable/disable some of the available signatures, but few systems allow for the dynamic inclusion of new signatures at execution time. In addition, the *ad hoc* nature of existing IDSs does not allow one to dynamically configure a running sensor so that a new event stream can be used as input for the security analysis.

Another limitation of existing IDSs is the relatively static configuration of responses. Normally it is possible to choose only from a specific subset of possible responses. In addition, to our knowledge, none of the systems allows one to associate a response with *intermediate* steps of an attack. This is a severe limitation, especially in the case of distributed attacks carried out over a long time span.

Finally, the configuration of existing IDSs is mostly performed manually and at a very low level. This task is particularly error-prone, especially if the intrusion detection systems are deployed across a very heterogeneous environment and with very different configurations.

This talk describes a framework for the development of intrusion detection systems, called STAT, that overcomes these limitations. The STAT framework includes a domain-independent attack modeling language and a domain-independent event processing analysis engine. The framework can be extended in a well-defined way to match new domains, new event sources, and new responses. The resulting set of applications is a software family whose members share a number of features, including dynamic reconfigurability and a fine-grained control over a wide range of characteristics. The main advantage of this approach is the limited development effort and the increased reuse that result from using an object-oriented framework and a component-based approach.

STAT is both unique and novel. First, STAT is the only known framework-based approach to the development of intrusion detection systems. Second, even though the use of frameworks to develop families of systems is a well-known approach, the STAT framework is novel in the fact that the framework extension process includes, as a by-product, the generation of an attack modeling language closely tailored to the target environment. This talk focuses primarily on the STAT framework.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Securing the Interaction between Unclassified Military Networks and Other Systems

Richard Hicks MA MBCS CEng
QinetiQ
Malvern Technology Centre
St Andrews Road
Malvern
Worcestershire
WR14 3PS
UNITED KINGDOM

rmhicks@QinetiQ.com

ABSTRACT

Unclassified networked systems represent a significant proportion of military information systems and are very important for NATO and are vital in order to enable NATO to perform its military functions. This paper gives sample threats that must be countered. However, there are other security requirements, including the ability to share information in a controlled manner, to have it always available, wherever the users are and whenever they need it, but only to the correct people. This very need to manage the sharing of information with a wide variety of users makes it harder to manage the security of an unclassified network than a classified network. It is necessary to balance the requirements to share with the need to protect. It is not just the computers that need protection — the communications need protection too; this should be done holistically. The document finishes by giving some potential solutions to computer security problems.

1.0 INTRODUCTION

The theme of the symposium is to discuss methods by which Unclassified computer networks can be used to perform military functions. Unclassified networked systems represent a significant proportion of military information systems and are very important for NATO in providing services such as email, logistics support, unclassified file transport and access to the vast amount of information on the World Wide Web. Such services are vital in order to enable NATO to perform its military functions. NATO forces need to be able to deploy rapidly and inter-connect to allies, partners and civil organisations, both governmental and non-governmental. In peace-keeping scenarios, NATO forces need to be able to connect to opposing factions to facilitate and to mediate peace-keeping operations. Furthermore, NATO needs to be able to connect these unclassified networks to its own classified networks. Indeed, NATO has a major business need for all these networks to **appear** to the **users** to be seamlessly interconnected to facilitate the flow of information to all NATO staff who have need for the information, whilst, at the same time, ensuring that sensitive information does not leak to inappropriate destinations.

[1] QinetiQ grants NATO the right to make copies of this document and to publish it without seeking permission from QinetiQ.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

2 BALANCING THE REQUIREMENTS

However, there are other security requirements, including the ability to share information in a controlled manner, to have it always available, wherever the users are and whenever they need it, but only to the correct people. Indeed, it is this very need to manage the sharing of information with a wide variety of users that makes it, in many ways, harder to manage the security of an unclassified network, (Figure 1). Indeed, it can be argued that the security requirements of an unclassified network used for military purposes are much harder to solve than for a TOP SECRET network. In a TOP SECRET network, the concept of "lock it up tight" is often much more acceptable than on an unclassified network. On unclassified networks, fast-responses, any-to-any communications requirements make security much harder to enforce.

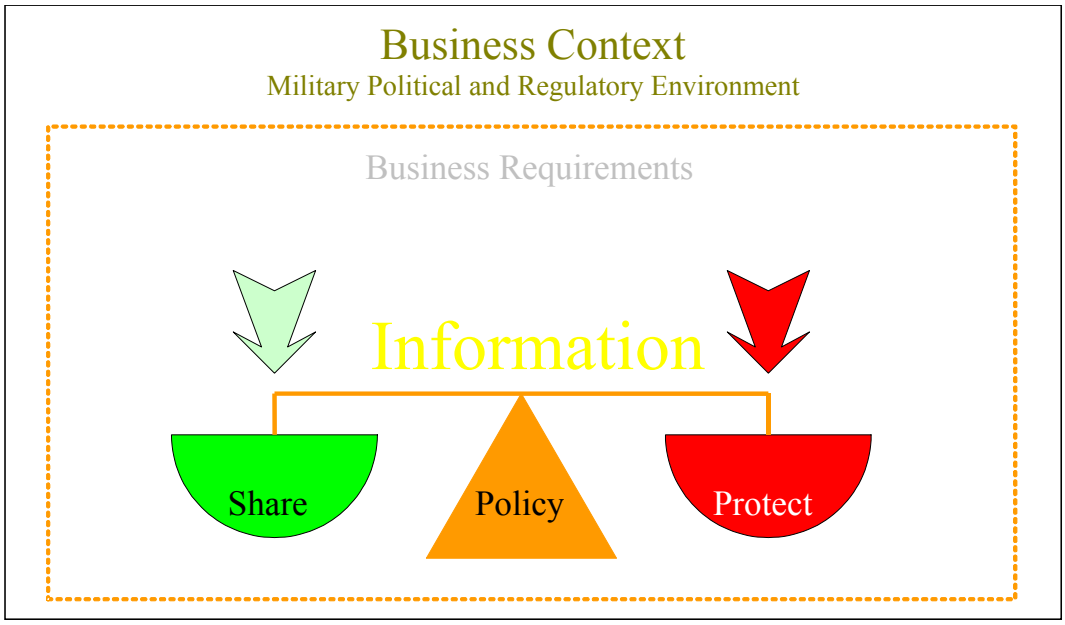


Figure 1: Share versus protect

Security needs can be many, and for a company's CEO, QinetiQ would suggest the top two security priorities could be expressed as:

- What does the company have to do so that I do not go to jail?
- What does the company have to do in order not to go bankrupt?

For the military commander, they may be expressed as:

- What do I have to do to best attack the enemy within the terms of engagement?
- What do I have to do in order to stop him attacking me?

In both cases, there is a need to share information — if intelligence know something, but decide that it is too secret to let the planners know it, or the planners deny vital information to the actual war-fighter, they may well have caused a self-inflicted denial-of-service. However, it is always risky to share information, whether with a dubious ally-for-the-time-being, or somebody with the same nationality and high clearance in the same business area. Policy may also affect things. As an example, a plausible policy could be to require a nuclear-powered submarine commander to comply with security policies, but give him the discretion to break any of these rules if necessary in order to prevent the submarine being sunk or captured — except that is for any rule that is designed to prevent the nuclear reactor exploding

The high-level policy rules have to be interpreted at a lower level. For many years, QinetiQ has been advising the UK MoD and commercial organisations on the best ways of connecting disparate computer networks with only partly compatible security systems to optimally meet the business and security objectives. Quoting from one QinetiQ document that has been endorsed by the UK MoD

a. If the presence of any software on a computer is of significant use to an attacker, and there is no significant business requirement foreseen for the software, the software shall not be present on the computer;

b. If the granting of an access right on a network or computer is of significant use to an attacker, and there is no significant business requirement foreseen for the granting of the access right, that access right shall not be granted;

c. As an objective, for an attack to succeed, it shall be necessary to breach the basic file permissions, two configuration barriers, registry access permissions if running Microsoft software, exploit a bug or Trojan functionality in vital operating system security software or Trojan functionality in a driver.

The first of these principles says in essence, "do not give attackers software tools to attack you with unless you have to". Only too often computers are configured with out-of-the-box insecurity. Too many system managers also say, "Yes, I would like to make my system secure, but the badly-written application software that I have demands that many trapdoors are left open", only of course they have to phrase this in more tactful language. Only too often, security is considered too late and system owners have to make a risk judgement between meeting a business need with insecure software, or protecting other business needs by denying the use of badly written software.

The second principle is essentially the same, but discusses access rights instead of the presence of software.

The third one takes a risk management approach. File permissions are simple, frequently used and obviously vital, so are likely to provide trustworthy defence mechanisms. Microsoft registry access rights, if correctly configured, are also likely to be secure for the same reasons. However, software items that can be configured are less likely to be secure as there are more options, with potentially untested combinations, some of which may be insecure. Hence, as a target, it should be necessary to breach two configuration barriers for an attack to succeed. Ideally, all security-relevant software is free of bugs and Trojan functionality. However, good design techniques, code reviews and testing cannot eliminate these entirely and they need to be appropriately complemented by defence in depth.

3.0 SAMPLE ATTACKS

By way of introduction to the problems of meeting these requirements, and to give a technical rationale for the suggested security measures, this paper gives some examples of how real-life operational networks have been broken by making use of documented features of security systems. In this paper, information regarding the identity of the computer systems that have been discussed has been intentionally omitted from this unclassified document.

3.1 Poor database interface design

A good secure system will usually require a person to authenticate before granting access. The rationale for this is that if a person is required to authenticate themselves to a computer system, the knowledge that unauthorised actions can be attributed to the person making them may well **deter** them from making the action in the first place. However, for this to be realistic, the person must be convinced that their actions can be attributed. There is also a de facto requirement that the security logs are sufficiently protected so that legal action can be taken if required, only too often this is not the case.

One example concerned a utility company. Its customers would phone up to challenge a bill, ask for waivers or for debt forgiveness. The utility company employed staff to listen to its customer's reasons, and to modify the customer's bill following guidelines supplied by the utility company. The company was aware of the fraud risk that its employees may offer to decrease customer's bills in return for a bribe, thereby causing a major loss of revenue. To counteract this, they insisted that their staff could only access the billing database via a front-end that required them to authenticate and which in addition securely logged all cases where a customer's bill was decreased. However, the security was applied to the front-end, not the server itself. It was discovered that if a user started up a session and then unplugged their terminal briefly, the user application failed. However, the connection into the database was restored, allowing the user to make uncontrolled, unsupervised, un-logged modifications to the database. The cure should be obvious, protect the object itself, not the user interface.

3.2 Insecure trust chains

Here is a further generic example. The users were required to enter a password which was hashed in the client computer using a special secure algorithm that was passed to an authentication server together with the claimed user name. If the user was authorised, a message was sent to the database server instructing it that the user was to be treated as a trusted user and giving the types of permitted access (Read-only user, Read/write user, database administrator etc). As these messages went out on a broadcast LAN, all that an attacker had to do was to look for an authentication package going to the database server from somebody else, store it, wait for them to log-off (as evidenced eg by no more traffic from the client) and then to replay the authentication instruction.

It is believed that an attacker could also have replayed the hashed password message at the authentication server. This was not tested as the attack was marginally harder and it made an entry in the security log. (Note too that adding a bespoke government password hash gained little as the attacker did not want the password but the **hashed** password.)

Once again, it was necessary to protect the asset, not the user-interface. QinetiQ was faced with a similar problem; in this case making a secure connection between a Server running a secured Microsoft operating system and Trusted Solaris. QinetiQ solved the problem that their security models were incompatible by writing a small bespoke add-in to both systems which used a symmetric key to authenticate the transactions between the two trust domains.

3.3 Bluetooth Technology

3.3.1 Description of Bluetooth

Bluetooth is a generic, short range radio connection that is used to transfer data over the air. The radio link operates in the 2.4GHz Industrial Scientific Medicine (ISM) band.

Packets are transferred over the air using a frequency-hopping scheme. The frequency that the packets are sent on changes in a pseudo-random fashion. Bluetooth devices must know the frequency hop sequence in order to communicate with each other. Frequency hopping reduces the probability of interference from other devices using the same frequency.

All Bluetooth devices have the ability to be either a master or a slave. The device initiating a connection becomes the master. Communication only occurs between the master and slaves. Slaves are not able to communicate directly with each other; they need to communicate via the master.

A network of Bluetooth devices is called a 'piconet'. Each piconet can have only one master and up to seven slave devices. Devices can operate in multiple piconets called a 'scatternet'.

To find other Bluetooth devices in range, a device sends out an 'inquiry' packet. All devices that are in range and 'discoverable' will reply with a Frequency Hopping Synchronisation (FHS) packet which contains information such as its Bluetooth device address (BD_ADDR) and the type of device it is eg a phone. The FHS packet contains all the information needed to try and connect to the device. However, devices that are 'non-discoverable' will never reply to an inquiry. Making a Bluetooth device 'non-discoverable' is considered good security practice because it hides the presence of the device when an attacker tries to find devices using the inquiry mechanism.

3.3.2 RedFang

RedFang is a free Linux tool to find Bluetooth devices that are in non-discoverable mode by "brute forcing" the Bluetooth device address (BD_ADDR). The original version was written by @stake Inc. It takes advantage of a function in the Bluetooth standard called 'Remote_Name_Request' which allows a Bluetooth device to obtain the user-friendly name of another Bluetooth device. If a connection does not exist between the two devices, a temporary link layer connection is established in order to obtain the name of the remote device. This temporary connection can always be created with devices that adhere to the Bluetooth standard, even when full Bluetooth security is applied to the device.

RedFang passes a BD_ADDR and timeout value into the 'read_remote_name()' function and waits for a response to see if a remote Bluetooth device with the BD_ADDR specified is within range. If there is such a device, it will send its user-friendly name to RedFang computer. If there is no response, RedFang will continue on to the next BD_ADDR. All Bluetooth compliant devices will respond to RedFang even if the device is in non-discoverable mode. Hence RedFang can find all Bluetooth devices that are within radio range.

In October 2003, RedFang version 2.5 was released, which was developed in collaboration with QinetiQ. RedFang 1.00 is only able to search for devices made by TDK. It requires the user to change to the source code if they were to search for non-TDK Bluetooth devices. However, RedFang 2.5 is vastly improved and allows the user to input the BD_ADDR search space and is multi-threaded which allows a theoretical maximum of 127 Bluetooth USB devices to search simultaneously for Bluetooth devices.

3.3.3 Practical Feasibility of RedFang

To search an entire manufacturer range of Bluetooth addresses, RedFang needs to search 166 (ie 16777216) addresses. Assuming a timeout of 20 seconds for each address to ensure maximum reliability, it would take (166 * 20) seconds to search the entire address space. This is approximately 10.7 years.

It should be noted that RedFang has a theoretical maximum of being able to use 127 devices in parallel to search. Assuming the same timeout of 20 seconds and excluding the effects of radio interference, the entire search space could be searched in approximately 30.6 days. If you factor in the effects of radio interference, this search time would be increased dramatically because of collisions from devices sending packets on the same frequency.

@stake claims that RedFang can search an entire manufacturer address space in approximately 90 minutes using just 8 USB devices working in parallel. However, this setting will not yield reliable results. The longer the timeout RedFang uses, the more reliable the results will be, but the search time will be longer.

Due to the short range and frequency hopping of Bluetooth technology, an attacker may have to be fairly close to the target system to use RedFang effectively.

Cures to Bluetooth are harder, as this is essentially a brute-force type of attack. It is suggested that the best cure is to keep the BD_ADDRs secret, to allocate them randomly and ideally, to automatically change them frequently. Some would say there is a better cure, ie not to use Bluetooth, but that causes a 100% self-inflicted Denial of Service. One has to balance the business need to communicate using Bluetooth against the threat to confidentiality and authenticity if Bluetooth is used.

3.4 Remote Denial of Service

Remote denial of service has obvious attractions to an attacker. Perhaps the best-known example of this is SYN flooding, which is not discussed here it is now well-known. However, there are many other methods.

Wireless attacks are attractive as they permit attackers to perform an attack logically from within the physically protected perimeter, but safely outside the physically area. Wireless waves do not stop at barbed-wire fences, a point that is too often overlooked.

As an example, QinetiQ was studying a 802.11 network for a customer. We masqueraded as an access point and instructed a valid client to dis-associate. It promptly tried to re-associate. We could then have repeated the disassociate instruction until the client thought that the network was permanently faulty and gave up trying to connect. We could then have moved the attack to the next client. After killing all the clients on one access point we could then change frequency to kill the next access point. If we had used an illegal high-gain directional antenna we could have killed all the access points on that site from a single off-site location. Eventually, the clients tried to reconnect, but as this happens so slowly, the attacker can keep multiple clients killed using only a single attack computer.

Alternatively, we could have waited for an encrypted DHCP request. QinetiQ can easily spot these by observing the protocol flow. From this, we can use a known plain text attack to recover the key stream. This, coupled with a knowledge of the CRC used, would allow us to construct arbitrary attack traffic of the same message length or shorter. Unfortunately from the attackers point of view, apart from performing a Denial of Service attack against a DHCP server, there is little that can be done with such short messages. However, we constructed a PING message with a length of 1 byte longer than the known key stream. To do this we had to postulate the trailing key stream byte. We had a one in 256 chance of getting this correct. If we got a response, encrypted by a key we did not know, we knew we had guessed the correct key stream byte. If not, we repeated until we got a response. Each time we got a key stream byte correct, we kept repeating the attack, lengthening the key stream until we could construct an arbitrary attack message of useful length. This attack typically took us only a few seconds.

From this point, we could have launched a standard SYN flooding attack, but now with the advantage that we were untraceably outside the secure perimeter, and with no incriminating wire link to the people being attacked. Other attacks were also possible but are out of scope of this unclassified document.

4 NON-COMPUTER IT SECURITY REQUIREMENTS

Computer security can only cover some aspects of the total security requirements. As an example, in one valuable military asset there were two main networks, one at SECRET, the other at RESTRICTED. Most staff were only cleared to see the RESTRICTED traffic. In this environment, it

was vital to be able to reconfigure the system at very short notice to meet new requirements. There were also extremely severe space requirements which required the SECRET servers to share the same racks and some networking components as the RESTRICTED computers. There was a very real risk that, in the heat of battle, the two networks might be crossed-connected. This would imperil both the local networks and the networks connected to the military asset. Various solutions were discussed such as:

- Having a detector connected to both networks that would identify all computing assets on each network and produce an alarm if it spotted that any computing asset had been inappropriately moved from one network to the other.
- Having a firewall on the outgoing links from the RESTRICTED network to detect the presence of SECRET information. Two sub-options were possible, barring the SECRET traffic leaving over the RESTRICTED link or merely raising an alarm. The first would better protect the confidentiality of the information, but could potentially needlessly endanger the military asset. Allowing the SECRET information to leave on the RESTRICTED network was not as bad as it seemed at first sight as most or all of the recipients would in fact be cleared to see SECRET information. This method could potentially also catch security breaches due to end-users copying information incorrectly from one network to the other using eg floppy disks.
- Doing nothing, on the basis that the probability of an enemy being able to exploit the potential security breach was low. For a real breach, several things had to happen, the network administrator had to make the mistake, the enemy had to detect the mistake and to take advantage of the error whilst the information was still useful.

The following example shows how real-life network management can be a vital part of information security. Consider the following network (Figure 2), where high-grade cryptos are used to inter-connect the two areas.

Figure 2: A secure environment…

Arguably, the system is secure because of the use of high-grade cryptos. However, looking more carefully at the system depicted in Figure 3, it can be seen that the system may be insecure as there is a Traffic Flow Security bypass channel. Rogue software operating within the secure areas can modulate the amount of traffic passing through the crypto and set up wide-bandwidth links that bypass the security nominally provided by the crypto. Indeed, QinetiQ has produced two demonstrators of how to get useful information un-encrypted though quality cryptos.

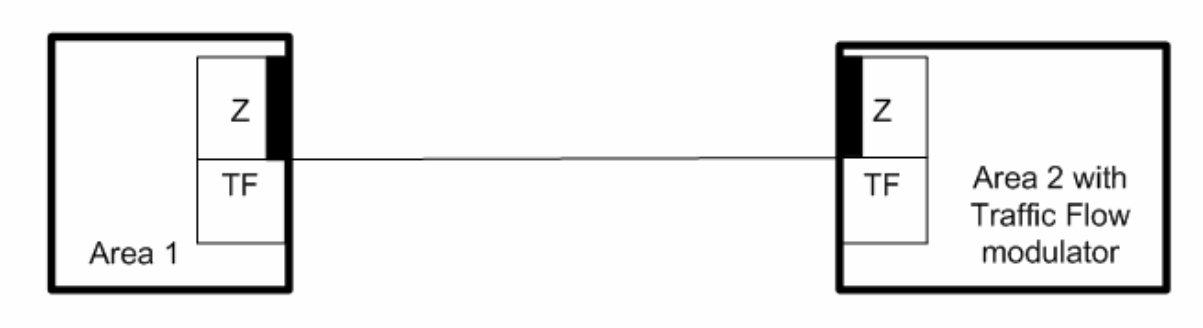


Figure 3: … which is actually insecure

QinetiQ did not breach the security of the cryptos, as, explicitly, they were never claimed to be secure against the methods QinetiQ adopted. All it did was to demonstrate that to achieve a "secure" system you have to consider both **computer** security and **network** security.

5 COMPUTER SECURITY CURES

So, how do you make things secure?

Well, a simple method is to deny access to unwanted ports. This is not just barring ports, but barring where the traffic can come from too. As an example, QinetiQ was advising on a system where a switching function was being performed by commercial hardware. This hardware was configured via web browser interface, protected by a fixed password. No mechanisms were available to prevent brute-force password attacks. Potentially many hundreds of these devices had, for network-management reasons, to share the same password and the devices had to permit web browsing traffic to traverse it.

However, the customer had very little effective control as to who could generate web traffic on some of the ports. The hardware device was not useless from the security point of view; it could perform port blocking. All QinetiQ had to do then was to only allow web browsing on standard logical ports to enter the hardware from untrusted users, and move the management interface to a non-standard port. Thus, the only source of an attack on this hardware that could reach the management interface was from a clamped-down computer (or a mis-connection by a network engineer).

Wireless connections are problematic too. The information owners in one system did not want the built-in wireless links being used, (or abused to information in and out of their secure systems bypassing the firewalls). QinetiQ has discovered a way with reasonable confidence to stop the information paths being mis-configured by users who wish to bypass security, or by software masquerading on behalf of careful users.

The Computer Incident Advisory Capability (CIAC) report in bulletin http://www.ciac.org/ciac/bulletins/m-005.shtml that says "sensitive or private information could inadvertently be sent to Microsoft. Some simple testing of the feature found document information in one message out of three." For one customer, QinetiQ recommended minor changes that actually improved the user experience (Is this a case where security has a negative cost?) preventing potentially sensitive information leaking out via this mechanism.

Using NTFS, attackers can add "Alternative File Streams". This is attractive to attackers as sensitive information can be added covertly to an innocuous file. What makes this particularly serious is that all the normal tests on a file, such as checking its size, do not display anything untoward. As an example,

a 1 Mbyte secret file can be attached[2] to a 20 byte insensitive file. The DIR command, Windows Explorer/Properties and similar will show that the file size is 20 bytes, not 1,000,020 bytes. However, anybody who knows the name of the Alternative File Stream can easily get the secret information. A public example of this was a rogue webmaster who attached pornographic information to some files in his company's web site. It was completely invisible to all except those who knew it was there. A government example could be a traitor who published sensitive information on a server for access by remote enemies of the UK.

6 PROTECT, DETECT AND REACT

The techniques discussed above are only one of the three major security techniques, that is Protect, Detect and React. QinetiQ has developed a range of Intrusion Detection Systems (IDS). The techniques employed in these systems can be used to spot intruders before they actually make an attack, eg whilst they are performing initial probes of the system before mounting an attack. With the increasing use of wireless, QinetiQ has also developed a tool that undetectably monitors a wireless network, looking for attackers who are trying to break-in. IDS tools have the advantage that they are passive and therefore undetectable to the attackers — they can never know if they have been spotted. In many ways they are an alternative to honey-pot sites, which have the disadvantage that their use can cause their owner to run foul of entrapment legislation.

The third part of the security techniques depend very much on the philosophy of the system owner and their objectives. Potential reactions available to the military can include:

- Emailing the system owner;
- Blacklisting the source IP address;
- Performing offensive Electronic Warfare;
- Use of a honey pot site to deceive the attacker;
- Sending back plausible but misleading information;
- Accepting the attack for a time, information-mining it to discover what the enemy wants, and then using that information to launch a counter-offensive;
- Physical attacks against the attacker.

However, such techniques are out of scope of this unclassified paper.

More importantly, for any system, it is unwise to decide on the fly what to do when you are attacked. Thinking things though first before you procure the defensive hardware is usually better.

In summary, the threats to the security of using and interconnecting unclassified and classified networks are very real. However, by careful management these risks can be reduced to an acceptable level.

[2] As a simplified example generate a short text file named "`safe.txt`" Then use the command "`echo dangerous.txt > safe.txt:hide`" Using the type command, the dangerous text cannot be seen. The command "`more < safe.txt:hide`" will display the hidden text.

Dynamic Virtual LANs for Adaptive Network Security

Diego Merani, Alessandro Berni, Michel Leonard
NATO Undersea Research Centre
Viale S. Bartolomeo 400
19138 La Spezia
Italy

netcentric@saclantc.nato.int

SUMMARY

The NATO Undersea Research Centre (formerly SACLANTCEN), the research establishment of the Allied Command Transformation (ACT) strongly relies on Network Centric technologies and capabilities to improve the effectiveness of its scientific research. This requires architectures for the interconnection and data sharing that are flexible, scalable, and built on open standards, to ensure transparent interoperability between shore laboratories (both NATO and national) and assets located at sea (research vessels, buoys, autonomous vehicles, sensors and acquisition systems), all connected using a wide range of communications media (e.g. SATCOM, wireless ad-hoc networks, acoustical undersea communications). In addition to that, to fulfil its mission, the Centre has an extensive cooperation program with scientists and researchers, consultants and contractors, civil and military personnel coming, for a limited time period, from several NATO nations. It is a common requirement for them to be temporary connected to the Intranet and to the external Internet: this requirement presents important issues about the security within the internal network; access to networking resources must be controlled while preserving the relative "openness" of a research centre. This paper presents some of the concepts and architectures developed to control the access to network resources and to react to internal attacks.

1.0 INTRODUCTION

The NATO Undersea Research Centre, located in La Spezia, Italy, is dedicated to fulfilling NATO's Operational Requirements in undersea warfare science and technology. Its Scientific Programme of Work, currently organized along three main thrust areas (Antisubmarine Warfare, Mine Countermeasures, and Rapid Environmental Assessment) has resulted during the past 40 years in several scientific and technical contributions that are now part of the set of standard capabilities of all NATO navies.

The execution of the Scientific Programme of Work is performed by an interdisciplinary team that covers different disciplines, such as acoustics, oceanography, ocean engineering, real-time processing, and signal processing. Over the past years, a continuously increasing focus has been put on Network-Centric technologies and capabilities, which have emerged as essential tools to enable and improve the effectiveness of its scientific research

The development of Network-Enabled capabilities in support of undersea research requires architectures for the interconnection and data sharing that are flexible, scalable, and built on open standards. This is essential to ensure transparent interoperability between shore laboratories (both NATO and national) and assets located at sea (research vessels, buoys, autonomous vehicles, sensors and acquisition systems). Also, a wide range of communications media needs to be supported (e.g. SATCOM, wireless at-hoc networks, acoustical undersea communications).

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

The efforts in the development of Network-Enabled concepts are specifically oriented towards the definition of new generations of scientific instruments. The network will be used to improve the transmission of data from sensors to processors, increasing the capabilities of individual instruments. The resulting increase in efficiency will be larger than the sum of the individual instruments efficiencies.

1.1 The need for adaptive network security

The unclassified network of the Centre is connected to the Internet, and provides the standard services that are requested to a modern enterprise network: office automation, e-mail, Internet access and workgroup file sharing.

The Centre has an extensive cooperation program with scientists and researchers, consultants and contractors, civilian and military personnel visiting the Centre for periods of limited duration. It is a common requirement for them to be connected to the Internet, to use e-mail, download files. They also need to be able to exchange files with Centre staff and print their documents. For these reasons, the Centre has to provide connection to his network or part of it.

The impact of having external people working in collaboration with Centre staff is more evident because the internal infrastructure relies on Windows 2000 Active Directory standardization. Each new computer is automatically installed by Windows 2000 RIS (Remote Installation Server): this brings a high level of standardization in the operating system and software installed. Before connecting a new computer to the network, it is also checked against a checklist, to maintain the highest level of standardization and quality of service.

Visitors that come with their own laptop brings some configuration management issues: IP address and routing need to be reconfigured, and the same has to be done with network cards and printers. Also, when the visitors require participating in a workgroup with Centre staff members, file sharing and mutual authentication need to be tuned.

Having external computers connecting to the network introduces important threats: viruses or Trojan horses can be already active on the host; misconfigurations or previously installed software can unpredictably interact with the network; also, malicious user behaviour cannot be excluded (download of malicious applications from the Internet, unauthorized attempt to access Centre resources, DoS, eavesdropping etc.).

Centre policies for visitors connecting the network are able to mitigate, partially, the risks associated with viruses or malicious code. However, a wide range of threats comes and spreads using the network infrastructure as a vehicle. Also, identifying and remove the origin of an internal attack often involves a time lapse that can be used by the malicious agent (a virus or an attacker) to gain the control or spread across the network.

In this perspective, it is critical to have an infrastructure that is able to self-reconfigure and isolate portions of the network, when they are recognized as the source of an attack: therefore the network has to be deployed so that the impact of an attack can be quickly confined with the minimum impact to the availability of network facilities to the other users.

2.0 REQUIREMENTS AND TECHNICAL APPROACH

Undersea research, regardless of the specialty area being considered, requires the provision of network-enabled capabilities to a wide range of shared systems, ranging from acquisition systems and experimental sonars, to command and control information systems.

Those systems are in most cases experimental prototypes that are built in the Centre laboratories ashore, while the actual testing at sea is conducted onboard the two NATO Research Vessels, NRV Alliance and NRV Leonardo.

The computing facilities are rarely shared in between workgroups, this meaning that network utilization is well defined and confined inside subnetworks; on the other hand, a high rate of computer mobility leads to additional issues, like the automatic reconfiguration of a computer when moved from one network to the other (i.e. from the Centre to the Ship).

Scientific departments often use “non standard” computers. Those systems are not centrally managed and not automatically reconfigured when moved. This adds a level of complexity, because misconfigurations or human errors coming from those systems could interfere with the MIS or other computing facilities, thus reducing the QoS perceived by other users.

2.1 Technical approach

The operational environment of the NATO Undersea Research Centre is characterized by a centralized network management, which covers all locations where work is being conducted. The network is almost fully switched, being the access layer switches cascaded with a fiber optic link (trunk). Remote locations (the research vessels) are connected with the main network through a VSAT satellite link.

The network has been divided into logical subnets, corresponding to various workgroups. Each subnet is assigned a reserved address space. Usually, the address space assigned to a workgroup includes four class C networks, thus allowing further subnetting for special project-related cells inside a workgroup[1].

This architectural approach allows a high Quality of Service, limiting broadcast traffic, and confining undesired protocols within the boundaries of the subnet. In addition, the use of subnets increases security, allowing the discrimination of traffic at gateways, or limitations of access to resources or servers on a need-to-know basis.

The use of a plain subnetting scheme does not necessarily match with the physical positioning of workplaces in our building (this is also a problem for a large number of enterprises). Departments are often spread throughout a building, thus creating physically non-contiguous subnets. In addition, users and computers are often moved when reassigned from one project to another. This situation induces an overhead of re-configuration in a standard network.

The use of VLANs (Virtual Local Area Networks) provides a solution to problems associated with mobility and physical positioning of computers, by dynamically assigning the hosts to their subnet regardless of their physical position.

2.2 Static VLAN vs Dynamic VLAN

Two different approaches can be used when assigning computers to VLANs: static and dynamic. Normally, both methods are used, following the implementation schema suggested by Cisco Systems.

Static VLAN assignment means that each port of the switch is (statically) associated to a certain VLAN: a computer participates in the VLAN according to the port to which is connected. This approach is useful when the network is relatively stable and computers are rarely moved from one socket to another: once the switches are properly configured, the configuration overhead is low. In such an environment, security can be further improved by securing the MAC addresses on switch ports, thus limiting the number of MAC addresses allowed on a specific switch port. In addition, to get further selectivity and security, it is possible to manually assign MAC addresses to a switch port.

[1] See “Architectures for Network Centric Operations in Undersea Research [3]”

The configuration of static VLANs requires a perfect knowledge of network physical topology, since the network administrator needs to specify the VLAN number for every port of the access switches. On existing networks, this could create a relevant initial configuration overhead. In any case, switch ports must be reconfigured whenever a computer changes position or VLAN.

The dynamic VLAN approach on the other hand is more complex, but it does not require any reconfiguration when computers change their physical position in the network (for example, when they are moved from one office to another office). The VLAN assignment is made on the basis of physical (MAC) address of the connecting computer.

A database is maintained centrally on the VLAN Management Policy Server (VMPS); this database includes MAC addresses, and their association to the corresponding VLAN. When a computer is moved to a switch port configured as "dynamic port", the switch observes the MAC address of the connecting computer. A query is then sent to the VMPS server, using the queried MAC address as keyword, obtaining the VLAN assignment in return. The port is then inserted in the VLAN specified in the configuration file. All these operations are performed in few seconds, in a manner transparent to the user.

2.3 VLAN Management Policy Server[2]

Using the VMPS, it's possible to assign switch ports to VLAN dynamically, on the basis of the source MAC address of the device connected to the port. When a computer is moved from one switch in the network to another, it receives the new port to the proper VLAN for that host dynamically. The VMPS opens a User Datagram Protocol (UDP) socket to communicate and listen to client requests. When the VMPS server receives a valid request from a client, it searches its database for a MAC address-to-VLAN mapping, as illustrated in the following figure.

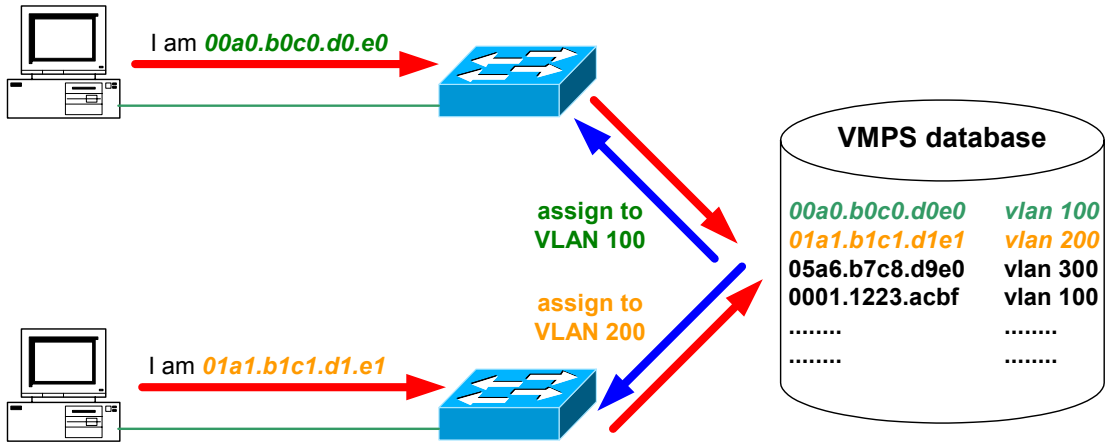


Figure 1 – VMPS initialization dialogue

If the assigned VLAN is restricted to a group of ports, VMPS verifies the requesting port against this group. If the VLAN is allowed on the port, the VLAN name is returned to the client. If the VLAN is not allowed on the port and VMPS is in open mode, the host receives an "access denied" response. If VMPS is in *secure mod*e, the port is shut down.

[2] Cisco Systems document no. 78-14904-01 [4]

If a VLAN in the database does not match the current VLAN on the port and active hosts are on the port, VMPS sends an "access denied" or a "port shutdown" response based on the VMPS secure mode.

It is possible to configure a fallback VLAN name. When a device whose MAC address is not listed in the database, VMPS inserts the client in the fallback VLAN. If a fallback VLAN is not configured, and the MAC address is not listed in the database, VMPS sends an *"access denied"* or a *"port shutdown"* depending whether the VMPS is in open mode or in secure mode. If VMPS is in secure mode, it sends a response.

It is possible to make an explicit entry in the configuration table to deny access to specific MAC addresses for security reasons: this is done by specifying the --NONE-- keyword as the VLAN name. In this case, VMPS sends an "access denied" or "port shutdown" response.

2.4 VMPS and configuration management

The Centre's network, as previously specified, is based on Windows 2000 Active Directory. Windows 2000 Active Directory DHCP server, configured in "static mode", maintains the IP addressing. Static DHCP rely on the MAC address to assign the IP address. With this assumption, each computer on the network needs to have a reservation inside the DHCP server in order to get the IP address; moreover, the IP address assigned to a computer will be always the same. Administering the DHCP server requires manual insertion of the MAC address and manual assignment of a free IP address.

To summarize, these three different databases, DHCP, DNS and VLAN, have to be maintained to describe the configuration of the network. The data intersection scheme is given in table 1. Another database, the "Administrative Property Database", has to be maintained to track administrative properties of the asset, since none of the operational databases is detailed enough for this purpose.

	RECORDS INSIDE EACH TABLE					
DHCP table	**IP**	**MAC**		**COMPUTER NAME**	**USER/ CONTACT**	
DNS table	**IP**			**COMPUTER NAME**		
VLAN table		**MAC**	**VLAN NAME**			
ADMIN.VE PROPERTY table		**MAC**			**USER/ CONTACT**	**ASSET No.**

Table 1

The VMPS configuration file is a text file based on a simple and effective syntax. It contains a brief set of common commands specifying the VMPS domain, ports groups and the policy to be used for unknown MAC addresses (shutdown or fallback VLAN).

The main section of the configuration file is a list of MAC addresses and VLAN names. This simplicity is also a weakness if seen from a network management perspective: inserting, modifying or deleting an item is not a user-friendly task, because the operations have to be performed after manual inspection of the MAC address.

To consolidate the configuration management issues, a new application has been written to generate the VMPS configuration file as a function of the other databases. The application, written in Java, stores all data associated with each computer owned by the Centre and offers a user-friendly graphical interface.

A planned extension of the software will also allow the definition of DHCP and DNS parameters, thus it will be possible to configure the participation of a new computer to the network, and also modifications or deletions, from a single management console.

3.0 ATTACK AND RESPONSE

3.1 Network Intrusion Detection

Network intrusion detection is assured by Cisco Secure Intrusion Detection Appliances. Each appliance "sniffs" the traffic looking for malicious or unauthorized traffic. Updated signatures are loaded on the IDS on a weekly basis, to compare the observed network traffic against those signatures. The system is able to detect atomal attacks (those identified by the single packet) and also composite attacks (those that requires the de-encapsulation of data flow).

The real challenge associated with a network IDS in a VLAN switched network is the actual positioning of the appliance. The success of IDS detection relies on the ability to snoop the traffic through the sensor network card: therefore the effectiveness is directly related to the traffic flow that can be physically observed.

A simple corporate network can be divided into at least three zones: the main access layer, the server farm, and the WAN/Internet zone. Normally the three zones are implemented using separate subnetsand routing is required between them.

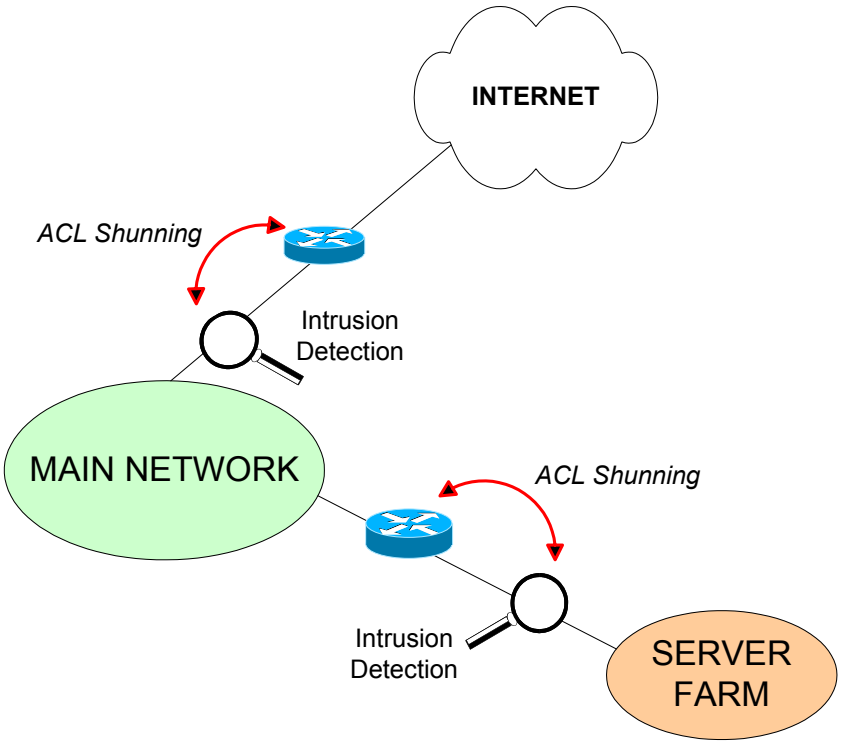


Figure 2 – IDS deployment

Good practice is to position the IDS sensor just above the links between two different zones: in the case depicted in picture 2 this means one IDS behind the firewall and another behind the server farm router. The sensors deployed following this pattern can detect attacks directed to the server farm or to the Internet.

Not only a sensor is able to detect malicious traffic, but also to send alert messages and take corrective actions. The IDS can shun the traffic using routers' ACL. Giving the proper coordinates to the IDS system, he can shun the traffic coming from IP address of an attacker, also blocking the subnet to which the attacker belong.

The shunning can to stop at the perimeter router attacks coming from the Internet. Therefore, relying on the IP address, the configuration has to be fine tuned to avoid self Denial of Service: the source address could be spoofed and the shunning action could block a legitimate user or network. For this reason, a literature exists that it's against the use of shunning on Internet routers.

Shunning, instead, can be very useful if used in the intranet. Since attacks need to bypass defined boundaries to be effective, a segmented network can be modified in real-time or near to real time to isolate and confine malicious activities or code.

Far from discussing all the possible targets and methods to conduct a network attack, it is enough to consider that an insider attacker (malicious user or code) will, with all likelihood, target the Internet or the server farm. The Internet is the preferred target of mass-mailer worms, for example; but also malicious software may also perform other Internet activities, such as connections to pre-defined remote sites. Nevertheless, a user who wants to launch an attack to a corporate network will target the server farm, and possibly will use the Internet to download tools or send leaked information.

With these assumptions, the possibility of isolating attackers both from the server farm and from the Internet when an attack is detected would be a powerful countermeasure.

3.2 Access Control Lists and Shunning

It's common to consider a Local Network as an open space with no traffic shaping or authentication gates. This is however a violation of the good practice of *need-to-know* policy access. In our case, for example, there is no need for scientific workgroups to have permanent access to administrative computers; on the same basis, there is no need for having a permanent flow of traffic between groups, unless they are involved in a joint research or project.

This profile can be well translated into access-lists between subnets. File sharing is the main threat today, as viruses "jump" across network shares; indeed the NETBIOS protocol, used for Windows file sharing, can be blocked at the gateway between those groups that do not have the requirement to use it.

Traffic flows from the main network to the server farm have also to be profiled, since not all users need to connect to all servers.

A structure in which all the traffic is filtered and parsed by special-purpose systems, brings a higher level of security. Anyway, the evaluation of risk assessment and the need-to-know policy can unpredictably change when an attacker enters the network. Would this be a malicious user or a viral code, the need-to-know associated to the host of threat should decrease to zero.

Response to attack is a process that can be divided into logical steps:

- evaluation and correlation
- automated action
- incident analysis
- managed action
- resolution

During the evaluation and correlation of an IDS event, the IDS engine decides the alarm level of the detected event. The level of a detected alarm is, usually, automatically classified on the basis of the signature file that constitute the IDS knowledge base. Anyway, the alarm level classification ca be tuned to fit the needs of the environment to which it is applied. When the alarm is critical (obviously, the thresholds of the classification can be configured), an action can be triggered. Cisco IDS rely on two reactions: TCP resets and ACL shunning.

TCP reset acts on the TCP session that originated the alarm. The TCP reset sent by the IDS appliance is able to drop the session in which the malicious activity is detected.

IDS shunning blocks the host or network originating the attack for a configurable amount of time. Shunning is performed through "on-fly" reconfiguration of a router or firewall, which has to be identified during the initial setup of the IDS. The blocking device gets modified from the IDS, which inject ACL statements to shun the attack.

This powerful feature can therefore create self Denial Of Service attacks, if not properly configured: in particular, attacks coming from the external world can have a spoofed address as a source address, thus causing a legitimate IP address to be blocked; nevertheless, the attacker IP address could be used by a remote firewall to make Network Address Translation: blocking the address could result in preventing all the network behind the NATted address to reach the site protected by the shunner.

IDS shunning can better be used internally to isolate the subnets in which a suspicious activity is detected. It acts on the gateway that protect the server farm to limit the access to servers from the VLAN in which a suspicious signature is detected. This action can be performed real time and in an automated fashion.

The incident analysis is a survey to determine the effects of an attack. For example, if the attack is the result of a virus infection, the survey can estimate the entity of the virus infection; further actions can be issued, if needed, to isolate the attack sources by VLAN ACL (that can block traffic within the VLAN) or banning the MAC addresses of the attackers from the network.

This last stage of incident handling is required since the current IDS versions cannot shun connections at layer 2, leaving the possibility of connections within the subnet. In addition to that, the escalation of counter-attack actions should not be conducted automatically beyond a certain threshold to avoid the risk that false positives create a chain reaction and bring self-DoS.

4.0 CONCLUSIONS

The solution proposed in this paper applies to small-medium business enterprises, where the use of a Catalyst series 5000 or above (that supports IDS onboard, and is able to snoop inside VLANs) is not a cost-effective choice.

The segmentation of the network, combined with intrusion detection performed on the local traffic, gives the opportunity of implementing a simple, heuristic mechanism of automatic reaction to anomalies or attacks, nevertheless maintaining a level of service for the portions of the network that are not interested or involved in the attack.

The improvement to the overall security posture can be implemented adapting a network architecture based centrally managed virtual LANs. VLANs provide not only better quality of service to different workgroups, but also allow the quick isolation of network users and segments, which appear to be subject to an attach, thus limiting the impact to the whole of the network.

The examples given are based on the Cisco Systems Safe framework, but the concept can be applied to other environments when the necessary functions are supported in the network hardware and software.

5.0 REFERENCES

[1] Leonard, M., Berni, A., Merani, D., "Architectures for Network Centric Operations in Undersea Research", presented at the RTO SCI-137 Symposium on "Architecture for Network-Centric Operations", held in Athens, Greece, 20 to 22 October 2003

[2] Leonard, M., Berni, A., Merani, D., "INSC applications for undersea research", presented at the Interoperable Networks for Secure Communications (INSC) symposium, held at NC3A, The Hague, The Netherlands, 4 to 6 November 2003

[3] Berni, A., Leonard, M., "Network Architecture for real-time contact sharing in multistatic sonar operations", SM-367 (NATO RESTRICTED), NATO Undersea Research Centre, La Spezia, 2003

[4] Cisco Systems, "Configuring dynamic VLAN membership with VMPS", Software Configuration Guide for Catalyst 4000 family, document no. 78-14904-01

Virtual Machine Applicability to Dynamic Coalitions

Lauren B. Eisenberg Davis and David V. Heinbuch
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road
Laurel, MD 20723
USA

Lauren.Davis@jhuapl.edu, David.Heinbuch@jhuapl.edu

ABSTRACT

The Johns Hopkins University Applied Physics Laboratory has investigated the suitability of virtual machine technology for use in dynamic coalition networks. The remote creation and teardown of dynamic coalition networks among partners with different degrees of trustworthiness is a very desirable capability and poses a difficult challenge to implement. There are many issues to be addressed in developing such a coalition capability, especially considering the high degree of security that is required. Configuring, distributing, and managing virtual machines from a central location may provide a secure means to quickly deploy, maintain, and take down private networks among coalition members.

JHU/APL has also investigated operational requirements that, together, encompass a variety of possible coalition network applications, such as networking among allied partners for military operations, dynamic collaboration of civilian partners, or coordinated large-scale system testing among multiple international organizations. For any particular application, only a subset of the requirements might apply.

This paper describes the system requirements for a Dynamic Coalition System, the applicability of virtual machine technology to the problem, and the additional technologies that would be necessary to fulfill the unmet requirements.

1.0 INTRODUCTION

Current and future reliance on coalitions and multi-national operations is complicated by the ability to share information electronically with coalition partners, many of whom are not members of traditional or long-standing alliances. The need to apply and deny access to some kind of coalition system in an ever-changing world situation is a challenge that must be met, addressing the need for availability of common information and tight security. The creation and termination of dynamic coalition networks among partners with different degrees of trustworthiness is a desirable capability and poses difficult challenges. Many issues need to be addressed in developing such a coalition capability, especially considering the high degree of security and integrity required. Configuring and managing virtual machines (VMs) from a central location may provide a secure means to quickly deploy, maintain, and destroy private networks among coalition members.

This paper discusses dynamic coalition networks, their security and how VM technology can address Dynamic Coalition (DC) implementation. The use of VM technology in securely creating Coalition Networks (CNs), and how such networks can be unfailingly shut down when no longer required, is explored.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

2.0 OVERVIEW

This paper proposes a framework and outlines technical issues associated with forming and protecting CNs using VM technology. After a brief overview of VM technology, we describe the concept of a Dynamic Coalition System (DCS), consisting of a Coalition Management Server (CMS) and multiple CNs that may operate at different levels of trust. The DCS provides for DC creation, via a VM paradigm, that will:

- Secure communications between acknowledged members of a coalition whose membership can change dynamically.
- Deny access to non-members and former members.
- Support rapid setup, termination, and reconfiguration.

The paper describes the desired DCS capability, operational concepts, and general system and security requirements. The requirements combine concepts developed at The Johns Hopkins University Applied Physics Laboratory (JHU/APL) with constraints raised in the Department of Defense (DoD) Goal Security Architecture (DGSA) **[1]**, with respect to issues discussed in Joint Vision 2010 **[2]** and Joint Vision 2020 **[3]**. The validity of these concepts has been demonstrated in a system prototype; the prototype DCS will be described in another paper.

3.0 DEFINITIONS

The following terms and definitions will be used throughout this document.

Coalition	A group of separate entities that must coordinate efforts and communication to achieve shared goals; examples are (1) a group of countries all working toward a common goal or (2) a partnership of autonomous, diverse, geographically distant members.
Coalition Network	A subset of the coalition members, all at the same level of trust, all sharing one set of information. A coalition may require more than one CN. A system of CNs will support different levels of trust in a hierarchical fashion.
Information	May include analysis and conclusions drawn from pools of raw data.
Information Overlays	Automatic way to provide data or information to more than one CN. The information would reside only in one place, but could be accessed by members of multiple CNs. This can help with database synchronization.
Membership Revocation	Returning in total the state of a member to that prior to its membership.
Vulnerability	A security exposure or weakness in a system that can be exploited to undermine the integrity, availability, or confidentiality of the information or resources of the system.
Incident	An actual exploitation of a vulnerability.

4.0 ROLES

The DCS will allow for three roles within its framework:

- Coalition Management Server (CMS) – sets up and takes down CNs in response to authorized requests of network owners. The initial assumption is that NATO controls the CMS.
- Coalition Network Owner (CNO) – determines membership and policy for the CN; the CNO is normally, but not necessarily always, a coalition network member.
- Coalition Network Member (CNM) – participant in the CN.

Figure 1 illustrates a simple example of a DC network depicting CNs whose members are either of different trust levels or are supporting different missions. The example involves three coalition or alliance groupings associated with NATO:

- The lead NATO countries of the International Security Assistance Force (ISAF). The ISAF is the international peacekeeping mission in Afghanistan **[4]**, and was put under the command of the several NATO members at the time of its deployment in January 2002.
- The Northern Light NATO member participants. Northern Light 2003 is "a NATO live, joint and combined exercise [which took] place between 15 and 26 September 2003 in the Irish Sea, on the West Coast of Scotland and Brittany." **[5]**
- The Northern Light partner participants (non-NATO).

The CMS serves three CNs whose CNOs and CNMs are illustrated in Table 1. The countries enclosed in parentheses are members Northern Light, but are not depicted in Figure 1 due to space constraints. Note that Germany, the Netherlands, and the United Kingdom (UK) are participants in multiple CNs. UK is the CNO of both Northern Light CNs, and a CNM of the ISAF lead nations CN; the Netherlands and Germany are CNMs in both the Northern Light NATO member participant CN and the ISAF lead nations CN. The data shared is not necessarily at the same trust level for all CNs.

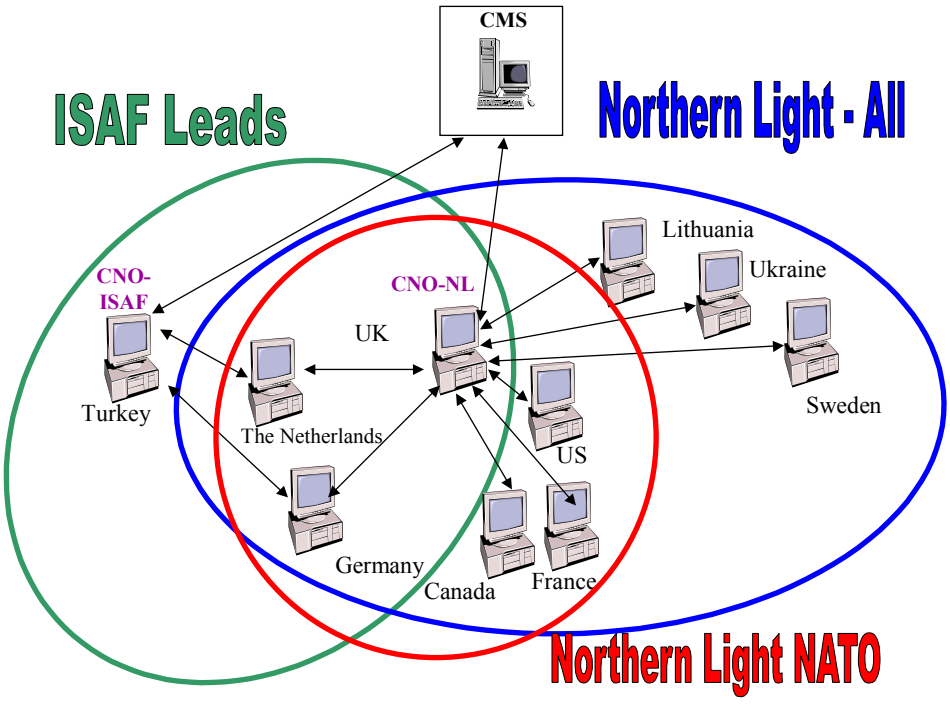


Figure 1 Example Coalition Network

Table 1 Coalition Network Example

Coalition Network	Description	CNO	CNM
CN#1	Lead NATO countries of the ISAF	Turkey	The Netherlands UK Germany
CN#2	Northern Light NATO member participants	UK	France US The Netherlands Germany Canada (Belgium) (Denmark) (Italy) (Norway) (Poland) (Spain)

Coalition Network	Description	CNO	CNM
CN#3	Northern Light NATO plus partner participants (non-NATO)	UK	Lithuania Ukraine Sweden France US The Netherlands Germany Canada (Belgium) (Denmark) (Italy) (Norway) (Poland) (Spain)

5.0 REQUIREMENTS

Before assessing the ability of VM technology to support DC networking, it is first necessary to determine all requirements that must be met to implement such a capability. DC networking is a high interest area that is being researched by a variety of organizations. Therefore, JHU/APL surveyed these efforts, as well as a variety of operational policy documents to determine a fairly comprehensive set of high-level technical requirements.

The following requirements were derived in part from paper reviews and research into other application types, including Defense Advanced Research Projects Agency (DARPA) DC projects and the DGSA. These requirements are for the full DCS, including aspects that may not be covered directly by VM technology, and fall into the following categories: Security Policy Creation, Initial Set Up and Termination, Coalition Network Membership Management, Effective Information Sharing, Data Separation, Data Protection, Communications Protection, Specialized Services Management, Interoperability Management, and Compromise Mitigation.

The requirements together encompass a variety of possible coalition network applications, such as networking among allied partners for military operations, dynamic collaboration of intelligence community partners, or coordinated large-scale system testing among multiple NATO members and task forces. For any particular application, only a subset of the requirements might apply.

Security policy creation requirements encompass combining security policies from each of the participants to form a cohesive policy, agreement on level of sensitivity, and control of the information shared between participants. Agreement on the level of sensitivity poses two interesting problems: labeling terminology, and the relative importance of the information to the different nations involved in the coalition.

Initial setup and termination requirements involve the technological logistics of several groups or enclaves joined together to achieve a specific purpose. A coalition may require more than one CN (Northern Light, for example, has a NATO member component, and a component of participating nations that are not NATO members), or an entity may belong to more than one coalition and require connection to several CNs (Germany, the Netherlands, and the UK, for example, are members of both ISAF and Northern Light NATO). The CNs need to be both established and separated, and a mechanism provided for establishing initial membership to CNs. To participate in DCS, each participant must have a system capable of supporting VM technology, prior to the initial setup. Rapid remote setup may be required for specific coalition situations. Each CN to which a user belongs shall be separated on that user's system. Termination of a CN entails

removing all members. In some cases, rapid remote termination may also be required. Note that any cryptographic products must implement algorithms releasable according to multi-national requirements of the participating nations.

Coalition network membership management requirements provide centralized management of the CNs, and encompasses admission of additional members; termination of members, both voluntary withdrawal and involuntary severance; and authentication of participants. For example, the original lead nations for ISAF were UK, Turkey, the Netherlands and Germany; however, currently, the lead nations are Germany, the Netherlands and Canada. Membership changes may need to be rapidly administered. New members may need to be admitted rapidly to support the coalition mission; membership revocation must be instantaneous, and revoked members must revert completely to the state of non-members. DCS must establish a mechanism for authenticating requests to the CN and control network access accordingly.

Effective information sharing requirements encompass technology to support CN communication, timely interaction with other CN partners, sharing of communication resources, and sharing of data and information among members. Dissimilar electronic communications systems must be networked transparently. No assumptions can be made about the sophistication of the communications suites of the member nations; consequently, the DCS must be able to operate with a minimum communications suite. Provisions must be made for both information push and information pull, if a CN allows data to be placed in a shared area. Furthermore, information must be revocable, if necessary.

Data separation requirements include establishing multiple trust levels (e.g., coexistence of varying sensitivities of information on the same information system so a single user can participate in multiple CNs), labeling information as well as regrading to a different level of trust, transfer of information outside the CN, and management of derived information.

Data protection requirements encompass strict mechanisms for ensuring that non-members cannot gain access to data shared between CN members, including an approved key management system to comply with international policy concerning protection of keying material and a strong identification and authentication system, including biometrics technology to ensure that non-members cannot gain access. This must include both data at rest and data in transit, and address unclassified but sensitive information as well as classified information.

Communications protection requirements encompass establishment of trust over the network and protection of network resources. Knowledge of the existence of a CN must be protected in sensitive operations, so that non-member entities do not know it exists. Outside observers must not be able to determine CN traffic flow characteristics. Protection must be provided to enforce protection of network resources, availability of the channel, integrity of the channel. Remote user environments must provide equivalent protection.

Specialized services management requirements encompass integrating specialized capabilities into a common operating scheme, to include integration of voice, imagery, and data; distribution of compatible software suites; chat capabilities; and white-boarding capabilities.

Interoperability management requirements include: addressing the wide diversity of hardware and software that each participant may wish to utilize; common understanding of terminology; and adaptation of commercial products, standards and technologies, as well as connectivity via common carrier communications systems, into the DCS. DCS must be platform independent.

Compromise mitigation requirements encompass prevention, detection, and reaction to security vulnerabilities and incidents, in order to protect information from hostile entities on the network. Systems should be patched, and patches tested on isolated systems. OS patching, or patching of shared application suites should be CN-wide, and configuration of independent member systems should be as standardized as possible, with permissions set as tightly as possible. COTS and GOTS components should be selected with security features as standard elements.

6.0 VM TECHNOLOGY AND DYNAMIC COALITIONS

Virtual machine technology allows a user to run multiple operating systems at the same time on the same PC. A virtual machine is one system image in a computer that supports multiple system images. Each virtual machine consists of an operating system and associated applications. The multiple system images on a physical machine may either all run the same operating system or different ones.

There are several different models of virtual machines used in the field of computer science today. The model most relevant to the dynamic coalitions problem stems from IBM's original work in the 1960's. IBM's model effectively partitions a computer system into several copies of itself with those copies having some portion of the total resources of the system. Like all virtual machines the IBM model provides a mapping of functionality from the virtual machine to the real hardware. In the IBM model, most of the virtual machine's instructions can be mapped directly to real hardware. However, to preserve the security of the virtual machines, a special set of instructions is trapped by a monitor function. This special set of instructions contains any instruction that would allow a virtual machine to affect another virtual machine. After these instructions are trapped, the virtual machine system then tries to emulate the desired effect for the virtual machine. The instruction trapping and emulation preserves the appearance that functions running in the virtual machine are running on standard hardware but prevents interference with other virtual machines. The important aspect of the IBM model for the dynamic coalitions problem is the non-interference or separation of the virtual machines. While the direct execution of portions of the instruction set is desirable from a performance perspective, it is not necessary.

DCS consists of a collection of CNs – networks of user machines capable of supporting VMs, at least one CMS, and a special INE VM. Each VM would be used to host a single CN for a single user, to separate data and communications by level of trust. VM technology provides the framework for the paradigm proposed for DC networking. The DCS would include VM technology and an inline network encryptor (INE), plus a means of incorporating a VM for every CN to which a participant would belong.

The ability to provide multiple VMs on a single workstation in a secure manner can play an important role in solutions to the dynamic coalition network problem. For example, multiple VMs can be electronically configured for each CN member, allowing access to multiple CNs on a single machine. Figure 2 illustrates a sample CN VM set up. Each CN VM will use an INE VM to connect to a CN.

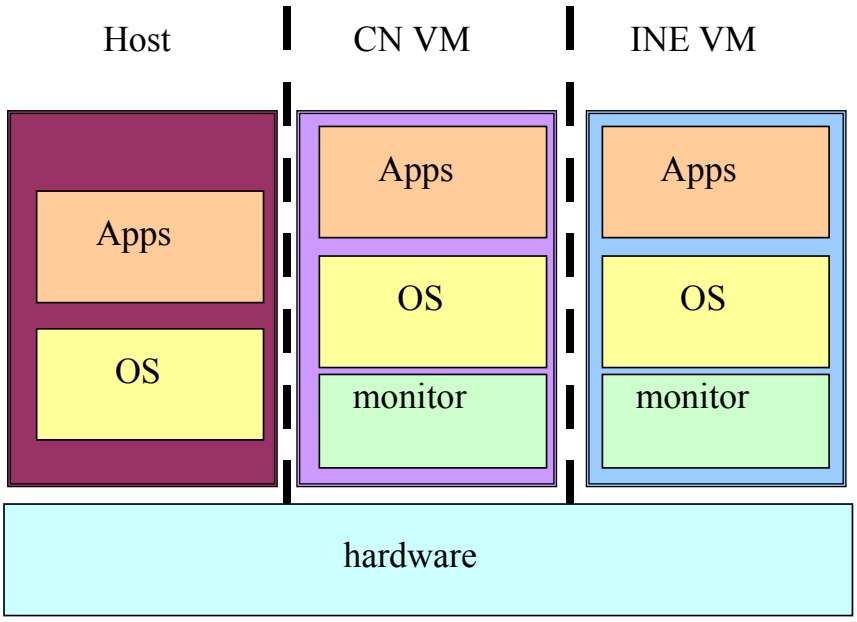


Figure 2 Coalition Network Virtual Machine Set Up

Figure 3 illustrates a possible architecture with three CNMs connected to a CMS. In this architecture all CN traffic passes through virtualized VPN routers on the CMS. This example utilizes the sample coalition setup described in Figure 1 and Table 1; both the Sweden and Germany CNMs are participating in the Northern Light – All (NL-ALL) CN. The Germany CNM also participates in the ISAF CN, along with the Turkey CNM. In addition the Germany CNM participates in the Northern Light – NATO (NL-NATO) CN along with other CNMs not shown in the figure. Each CNM requires a management network for communication with services on the CMS to manage its membership in the various communities of interest (COIs). For instance, a CNM might request membership in a COI and be notified of its acceptance via this management network.

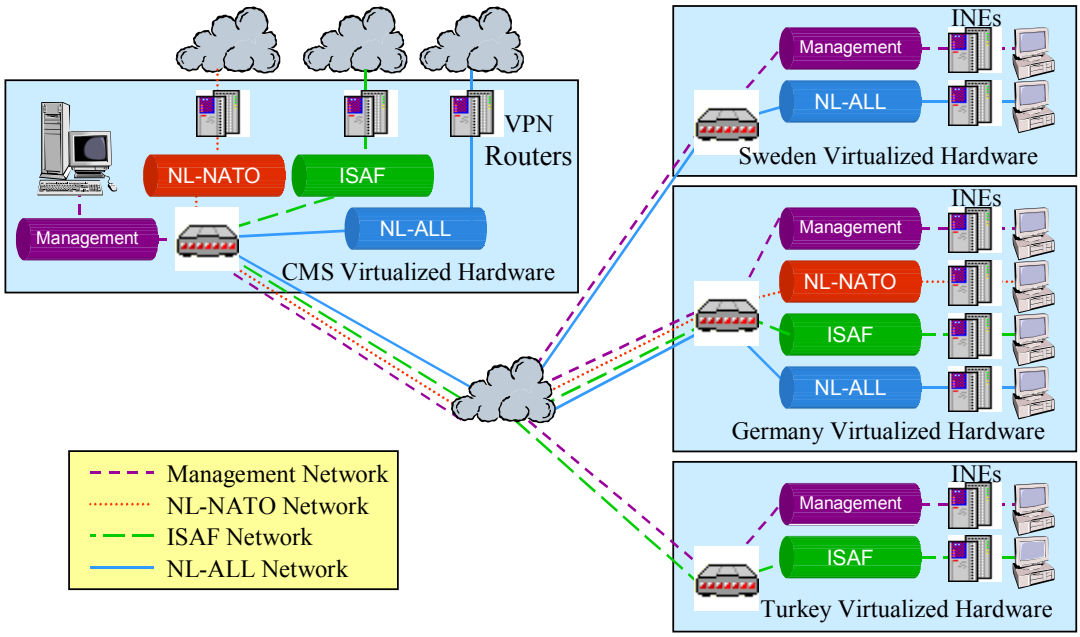


Figure 3 Example Coalition Network Virtualized Architecture

7.0 VM TECHNOLOGY APPLICABILITY

7.1 Security Policy Creation

VM technology does not provide any security policy and would rely on the host machine OS security policy. This has very little effect on the activities that take place inside the VM OS. The host OS security policy would primarily control the separation of the VMs and only control the VM's access to hardware, host file systems, and networks. Consequently, VM technology cannot address most security policy creation requirements for DCS.

For coalition operations, it is necessary to combine security policies of members to form a single cohesive policy that encompasses requirements of all participants, to create a merging of multiple security policies in use by member countries or organizations. The challenges may include reconciliation of policies written in different policy languages, if amalgamation is to be automated within DCS. This would dictate a need to establish a standard language for policy expression.

Because VM technology is unrelated to policy reconciliation, the assumption is that the policy combination will take place independent of VM technology and the CN policy is presented to the CN as a single cohesive unit. Consequently, new members will have to accept the existing policy; otherwise, manual adjustment of the CN policy will be necessary. Future work on DC may include technologies to combine policies.

Agreement on what a particular security level means is largely handled by the definition of CN, where data is equally available and approved for distribution to all members of a CN, who by definition are at an equal trust level. However, given data that is tagged by security level, separation and filtering can be securely enabled via a High Assurance Guard (HAG) in a VM.

7.2 Initial Setup and Termination

VM technology inherently supports joining coalition members on networks and allowing access to more than one network from a single computer. This allows coalition members to participate with different groupings of members or at different trust levels on separate networks.

A further goal is to allow networks of coalition members to be established and dismantled rapidly. The setup includes membership definition and communication capability. The CNO provides an approved membership list to the CMS for management. The challenges of this requirement include the logistics of secure, remote CN and the speed of CN setup with policy and membership in place. To handle the establishment and separation of CNs, a method to establish configuration settings for a standardized VM must be developed. This raises two important issues: (1) how the networks are set up and secured to all the different potential coalition members and (2) how the proper use of the VM or configuration for a standardized VM is assured. Actual time for rapid remote setup will depend on what logistics are agreed upon for distribution of Virtual Machine capability and configuration.

To start, all members must be possession of a machine that can support VM technology configured with dynamic CN capability. VM technology can be distributed in one of two ways: either pre-installed machines may be delivered with VM technology installed, or a CD containing VM technology software can be delivered for installation on a machine at each DCS site. Deploying among non-NATO allies[1] may be a critical problem. The technology for use in multi-national force (MNF) coalitions must be releasable. Available technologies exist to encrypt parts of the kernel, which can only be decrypted with a key provided by the CMS at runtime. It may be possible to meet this requirement via such concepts as a tamper-resistant box, but additional research would be required. Releasability of cryptographic algorithms is beyond the scope of this paper; however, any INE provided would have to meet releasability restrictions of the owner-nation or NATO group. Legal export restrictions for public key infrastructure (PKI) and encryption algorithms must be addressed for CNs with parties from multiple countries.

Initially, a secure network connection needs to be established with the coalition members. Each CNO submits a list of approved members to the CMS. The technical challenges include connecting members to the appropriate CN(s) only.

Rapid remote termination is supported by further VM technology development because the CMS or CNO will be able to turn off a VM once membership is terminated. When a CN is terminated, residual data storage becomes the responsibility of the CNO.

[1] Coalition partners with whom long standing alliances, joint policies, and protocols for data exchange have not been established.

7.3 Coalition Network Member Management

Participants must be configured with the correct VMs. While the central management may be controlled by a NATO entity, the control of a specific CN may be passed to the CNO. The CNO can then request from the CMS that access be given to or revoked from other participants. DCS will need a CNO that can be granted ownership of a particular CN and the ability to control its membership through interaction with the CMS. User registry is not provided directly by VM technology. VM technology does not provide user authentication. Although members are organizations or nations, actions performed in the context of the coalition need accountability to a particular person.

Member identity protection can only be partially provided via VM technology. The use of separate protected networks or encrypted tunnels should make the work required for an adversary to determine the real endpoints and content of the communications sufficiently high. It is assumed that with enough resources and/or time, an adversary could determine the real endpoints or content of the communications.

Using multiple VMs enables users to participate in more than one CN at a time by simultaneously using multiple VMs. CN membership revocation can either be developed or implemented with COTS products. DCS must be able to handle both voluntary withdrawal and involuntary severance. One approach may be to deny access keys for VPN tunnels of given networks. DCS must also consider automatic reconfiguration of the network as membership changes. Reconfiguration of the CN could be done by requiring all members to re-key.

7.4 Effective Information Sharing

By virtue of all CNMs using compatible VM technology, DCS provides transparency because all end-users are employing the same system. The establishment of a minimum communications suite carries the assumption of a minimum hardware standard, for example an X-86 personal computer (PC) with an Internet Protocol (IP) connection, to support VM technology.

The dynamic sharing of communication resources can be accomplished using VPN tunneling, which is supported by VM technology. Compatible VM technology can provide standardization of capabilities and environment. Timely interaction between CNMs can all be handled by the use of standard IP networking tools via network management of channel availability, load balancing, adaptive routing, and relief of traffic congestion. VM technology is not providing anything extra, but allows for IP quality of service provisions from IP networking tools.

Communications between the management server and clients may need to be handled separately. These communications could reveal information regarding which coalition participants are members of which networks. Because that kind of information could be very sensitive, some method of hiding that information or an out-of-band system needs to be developed. VPN tunneling may not be sufficient protection in all cases; either further protection of communication endpoint identities needs to be developed or separate physical networks will need to be used.

By standardizing CNs with compatible VM technologies, problems with process and procedure compatibility should be minimized from a technological point of view. Using common hardware and OS platform reduces the possibility of incompatibilities between coalition members, and allows for use of COTS products

Separation of information and users can be provided by either separate physical networks or an INE. VM technology can support information sharing. The complexity and type of information shared, processed, or

disseminated is limited to that which can be supported on those platforms. The CMS can manage multiple CNs at one time. VM technology will support data and information sharing if members use compatible application suites. VM-specific issues involve issuing a message to all selected potential participants – "You have been invited to participate in…" – which must be made understandable to the recipients, particularly if there is no de-facto standard language. CNMs of the same CN who do not share a common language are not precluded from employing technology to promote mutual understanding, but such technology is not directly provided by the VM.

Support of information push and pull is directly supported with common application suites that can be incorporated in each CN.

CNOs of all CNs involved in an information overlay must agree to share the overlay data. It will be difficult to implement information overlays using VM technology because it has no guards.

VM technology cannot directly revoke information that has been transferred to the CNM's machine, even if that information resides on the VM . Note that if information is pushed and/or pulled onto a CNMs machine, it can't necessarily be revoked when membership is revoked. The CNM's decryption capability can be disabled via a watchdog timer, rendering the information useless.

7.5 Data Separation

VM technology can support separation of multiple trust levels on the same system. Information and user activities will be separated by having each CN on a different VM within the user's system, and there is no mechanism for the information to cross VMs.

VM technology does not provide for proper labeling of classified information on the system, but does not preclude it. The user interface to each VM can be labeled with its trust level. The OS in each VM may be able to support labeling of the data or files within that VM. Data owners need to be responsible for information labeling or a COTS/Government off-the-shelf (GOTS) product would need to be incorporated.

The requirement for the data owner's permission to transfer data to other CN's must be handled via Concept of Operations (CONOPS) or use of third-party software. Data cannot be transferred between CNs except by the member(s) belonging to multiple CNs, and it can only be their own data.

VM technology cannot directly provide for re-grading; it must be developed from scratch or implemented via a COTS re-grader. The derived data (data correlation) restriction occurs when CNs share limited information and then use that information in analysis with other data that is not coalition-wide. Provisions must be made to keep any resulting analysis out of the CN. Derived data restrictions cannot be supported directly via VM technology; it must be a process handled in the CONOPS.

7.6 Data Protection

Data at rest can be protected by the host machine OS security policy and file system encryption. Data in transit can be protected with VPN via the INE, or by physically protected networks.

Standards required for DCS are: (a) security protocols, (b) authentication information, (c) key management and distribution, and (d) voice communications. The INE can support IPsec and probably many other tunneling protocols. A PKI system will be needed to handle VPN key distribution. Voice communications might be provided and protected by using Voice over Internet Protocol (VOIP) applications with the INE.

The host authentication system could include a biometrics authentication device. The VPN keying system and the host authentication together should provide a strong identification and authentication system for the DCS.

Protection of unclassified but sensitive information can be provided by limiting the information to the CN on which it is distributed.

Secure display implementations minimally will include screen lock provided via the OS. Not all OSs have a screen lock. For example, Windows NT/2000/XP do provide a screen lock capability; MS DOS does not. Unix/Linux variants require X Windows for the screen lock capability. However, the host machine OS could provide a host-based screen lock, if needed.

7.7 Communications Protection

An important aspect of protecting communications will be ensuring member confidentiality. A method for configuration of VMs, setup of CNs, and normal interaction on those networks should be designed so that CN participation is only known to the participants of that network and the server. It is always possible for an outside observer to determine that communications are taking place between two endpoints. The use of separated protected networks or encrypted tunnels should help increase the work required for the outside observer to determine this while still concealing the true endpoints and communications content. It would be impossible to develop a solution to provide full protection. One possibility is to produce constant artificial traffic at all times.

The host machine OS security policy can only provide protection for virtualized network resources running on VMs, such as filtering routers and INEs. Any real physical resources would fall outside of its scope.

VM technology can use separate protected networks for each CN. To deploy these networks dynamically, VPN tunneling or some other VPN technology will need to be used. Communication of management clients and server should be designed to provide availability and integrity. This will protect against denial of service, preventing CN member changes from being propagated to members in a timely manner and preventing members from communicating effectively. Aside from the existing protection offered through VPNs, this could be further enhanced by the addition of options like a biometric authentication system for the host machine OS.

The method of configuring VM technology systems to end-users will help ensure that the remote user environments provide equivalent protection. Configuration is dictated by standard distribution. Physical security protection cannot be controlled, but if preinstalled systems are shipped out and hard drive encryption is used, the security provided by a Virtual Machine Technology system will be assured. With a distribution method involving the installation of a Virtual Machine Technology system with CDs at the end user's location, the certainty that equivalent security is being provided is not as high.

7.8 Specialized Services Management

VM technology can already support a large number of specialized services with existing COTS products on the supported platforms. VMs can be preconfigured to include the compatible software suites. The specific services can be fulfilled to the extent that any application running on the supported the chosen VM OSs and virtual hardware can be supported by VM technology under DCS. However, any applications falling outside this suite will either not be supported or need to be developed for VM technology applicability.

7.9 Interoperability Management

Interoperability will be provided for those platforms supported by compatible VM technology. CNMs may span a wide diversity of hardware and software systems and varying levels of technology.

The use of common carrier communication systems will require certification and accreditation (C&A) of an INE VM to declare that the protection is sufficient. VMs are COTS products. Additional technologies may also include COTS components. A dynamic coalition solution using VM technology cannot guarantee that there is a version of the selected technology that can run on every platform.

7.10 Compromise Mitigation

All OS vulnerabilities are open to exploitation. The VM OS vulnerabilities are only exposed to whatever applications and services the user connects to the VM OS. If everything is tunneled, the VM OS is only exposed to other CN members. If the tunnels run on open networks, any INE vulnerabilities[2] that might surface would be exposed to the outside world.

An intrusion detection system (IDS) could be added to the system to provide for detection of security incidents, but one is not within the scope of VM technology. Audit logs can be maintained as well.

Procedural methods for mitigating compromise should be handled via a CONOPS. It may be possible to incorporate an auto-patcher as a COTS add-on, but that is not recommended by JHU/APL because it removes the necessary decision-making control over patch advisability. Because the security policy used on the host machine OS does not have any real control over the interactions internal to the VM OS, it cannot provide or enforce security policy for the CN. New components used for dynamic coalitions should be designed to further enhance these features.

VMs will provide the standard configuration necessary for dynamic coalition implementation, and users should be prevented from changing the specified configuration due to the host OS security policy. Security mechanisms for protection against hostile entities would have to be developed. Assessment of vulnerabilities of chosen or candidate technologies and COTS components is necessary. The host machine OS must enforce prevention of members changing the VM configurations. Key management is not provided by VM technology and would have to be addressed by third-party products.

Certain security features are provided by VM technology, such as isolation between VMs, which helps to ensure integrity and confidentiality. The issue of security features (e.g., access management) must be considered for additional technologies beyond VM technology. New VMs or VM patches could be distributed CN-wide when vulnerabilities are found in the existing VM configurations.

Non-persistent disk features should be used to provide the capability to revert to a clean state in a VM if the disk is corrupted. Incident recognition and a system for executing possible responses would need to be developed to implement a capability to support a documented recovery process for VMs; otherwise, it remains a policy issue.

Note that it may be possible for the VM can become corrupted if it crashed or the computer was turned off while the VM was running. In those cases, procedures would have to be instituted to recover from any corruption that might occur. Host machine OS corruption is possible with some file systems in the event that the disks are not cleanly shutdown. Using journaling file systems might eliminate the potential for corruption.

[2] None known at this time.

8.0 ARCHITECTURE OPTIONS

This section discusses ways to implement the DCS concept with VM technology. To implement this concept it is necessary to allow the creation and, possibly, removal of VMs, as well as allow a specific VM or host process to communicate with a management server to manage these VMs.

CN configuration can be centralized or decentralized. Table 2 summarizes the advantages and disadvantages of both configuration options, and the details are discussed in Section Table 2.

Table 2 Dynamic Coalition System Architecture Configuration Options

Configuration	Advantages	Disadvantages
Centralized	Membership revocation easier – remaining CNMs do not need to be re-keyed CN membership information more easily hidden – use network address translation (NAT) to obfuscate addresses at CMS	Central network services create a possible bottleneck (processing – encryption/ decryption, network bandwidth) Central network servers are an additional single point of failure (all CNs cease operating)
Decentralized	Possibly higher total throughput Normal CN operation does not rely on central resources	Membership revocation more difficult and takes longer – requires re-keying all CNMs Hiding CN membership is more difficult and less efficient – all possible CNMs must communicate random data constantly amongst themselves CNO loses control of membership if the CMS fails

8.1 Dynamic Coalition Network Concepts

Using virtual machine technology as part of the solution to the dynamic coalitions problem will allow multiple CNs to be accessed from a single end-user system. One of the hard problems is managing the creation of these CNs. The CMS controls the initial construction of CNs and manages interaction with the CNO by handling either the keying of VPN routers (for centralized control) or keying all CNMs (for decentralized control) to get them synchronized for the CN. A CN is created based on interaction between the CNO and the server. The CNO supplies the CMS with an initial list of participants. The CMS establishes the CN. The CNO is then given authority over the network and may add or remove members as needed.

Using a centralized network routing architecture, all CN communications pass through the VPN router responsible for that CN, and the CMS passes the membership information to that VPN router. The CMS can update this membership information on the VPN router as needed, based on communications with the CNO. With a centralized network system, membership revocation is easier than with a decentralized network system because a VPN server simply denies that member access. Another advantage of the centralized network architecture is that CN membership information can be more easily hidden using a form of NAT to hide which CNMs are talking to which servers. A disadvantage of the centralized network architecture is that the central network services pose a potential bottleneck and a single point of failure. The central networks services are a potential bottleneck for cryptographic processing and for network bandwidth. If the central network servers are unavailable, the CNs can no longer operate.

A decentralized network would allow the CN communications to travel only to the intended destinations and are not routed through a central VPN router. In the decentralized network, the CMS must pass out the proper key information to establish tunnels between CN members. The use of a decentralized network makes it difficult, if not impractical, to conceal the CN membership information from those outside the CN.

One advantage with a decentralized network architecture is that the cryptographic processing is distributed among all of the CNMs and there are no central network bottlenecks. Another advantage is that even if the CMS is unavailable, the CNs may continue to operate. A disadvantage with a decentralized network architecture is that membership revocation is more difficult and may take longer than with a centralized network system because all the CNMs need to re-key for the new membership list. Hiding CN membership information is also more difficult with a decentralized network architecture because all of the possible CNMs would need to constantly transmit random data amongst themselves to mask the real communications amongst actual CNMs.

9.0 CONCLUSION

VM technology can be used to meet many Dynamic Coalition requirements. Specifically, VM technology supports establishment and separation of coalition networks, membership in multiple coalition networks, transparent networking, multiple trust levels, protection of data at rest and in transit, management of specialized services, and interoperability. Further development would be necessary to address remote distribution of coalition technology, termination of a coalition network, addition and termination of participants, protection of communications, and compromise mitigation. Additional technologies or policies would be needed to address creation and management of joint security policies, authentication, access control, and protection of derived information.

A dynamic coalitions solution employing VM technology will solve many issues paramount to rapid remote coalition operations. Additional development and incorporation of third party technologies can provide the remaining functionality.

10.0 REFERENCES

[1] Department of Defense Technical Architecture Framework for Information Management, “Volume 6: Department of Defense (DoD) Goal Security Architecture, Version 3.0,” 30 April 1996

[2] Chairman of the Joint Chiefs of Staff, “Joint Vision 2010,” 1996

[3] Director for Strategic Plans and Policies, “Joint Vision 2020,” 2000

[4] NATO Update webpage, “Same name, same banner, same mission a NATO enhances ISAF role,” http://www.nato.int/docu/update/2003/04-april/e0416a.htm

[5] NATO International Military Staff webpage, “Exercise Northern Light 2003: NATO exercises a joint combined Task Force in Northern Europe,” http://www.nato.int/ims/2003/p030903e.htm

Components for Cooperative Intrusion Detection in Dynamic Coalition Environments

Marko Jahnke, Michael Bussmann, Sven Henkel
Research Establishment for Applied Science (FGAN)
Research Institute for Communication, Information Processing and Ergonomics (FKIE)
Neuenahrer Str. 20, 53347 Wachtberg
Germany

{jahnke|henkel|bussmann}@fgan.de

Jens Tölle
University of Bonn
Institute for Computer Science, Dept. IV
Römerstr. 164, 53117 Bonn
Germany

toelle@cs.bonn.edu

ABSTRACT

We present a prototype of an Intrusion Warning System for combining event message flows of multiple domain-specific security tools in order to determine anomalies for early warning and response. Unlike other approaches for cooperating Intrusion Detection Systems (IDS), we suggest a modified star shape architecture for distributing attack information and feed back warning messages. We assume that there are almost no known properties, neither of the underlying information providing local security tools nor of their local security policies. Such heterogeneous environments are typical for dynamic coalitions like NATO. We extended a well-known hierarchical distributed IDS architecture to provide Meta IDS services with feedback to the local access points. The extensions include three major items: Early Anomaly Warning - A graph clustering based anomaly detector for the event messages is used as an adaptive early warning module for largely scaled activities, e.g. internet worms. Information Sanitizing - Event messages are anonymized when leaving the local domain, according to a domain-specific information sharing policy. Message Aggregation - Additional filters for data reduction and application of predefined correlation rules make the data flow feasible

1.0 INTRODUCTION

Since many networks in coalition environments (CE) - like NATO - are connected with each other using the public internet to transfer unclassified data - or even classified information, using appropriate security mechanisms - it is possible to save enormous amounts of costs for dedicated connections. Thus, those networks are exposed to many different threats.

One of the biggest threats are spreading internet worms. Obviously it is easier to detect largely scaled security related activities when having access to large security attack information (i.e. event messages or audit records) from different locations within a computer network (IDS, firewalls, virus scanners etc.). But combining different attack information sources becomes even more important when aiming at detecting coordinated attacks against large numbers of target systems, where there is a common strategy behind all activities.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

So, one major goal from the CE security analyst's perspective is to combine all attack information in order to gain more input for this task. We define an Intrusion Warning System (*IWS*) for coalition environments as a system that gathers attack information from all available sources and generates warning messages about unusual system behaviour. There are different architectures to process coalition-wide audit data, such as *distributed cooperating IDS* and *centralized Meta IDS* which gain and process data from local security tools. This paper describes an architecture which meets general requirements of heterogeneous dynamic CEs. Additionally it discusses algorithms used for event message processing and anomaly detection and the status of their implementation.
The rest of this paper is structured as follows: Section 2 points out the essential issues for designing and building an IWS for CEs. Section 3 describes the overall architecture of our prototype system. Then, sections 5, 4 and 6 describe new architecture components which are necessary to solve the described issues. In section 7 we discuss the initial implementation and testing results. Section 8 outlines some of the related work that already has been done and points out the differences to our approach. Section 9 concludes on past activities and describes possible future directions.

2.0 ISSUES AND APPROACHES FOR A CE IWS

Before it is possible to design and build a robust and effective IWS for CE scenarios, several issues are to be solved. Some of these issues have been identified in a NATO/RTO working group report on IDS [1]. For our specific needs, we extend these issues to the following list of items:

- *Efficient Analyzer Algorithms*
  To get a benefit from collecting information distributed in the CE, we primarily need algorithms which help us to detect largely-scaled activities. On one hand, we need correlation techniques to detect coordinated activities. On the other, we also need anomaly detectors to act as an early warning system to feed the local security staffs with appropriate information.
- *Information Sharing Policies (IShPs)*
  One of the most important issues to be solved in order to realise coalition-wide exchange of attack information are differences between IShPs, since there may be information within IDS messages which are not to be exposed to even the closest coalition partners. An example for the information sanitizing is the anonymization of IP addresses, if a domain wants to keep its network topology secret. Another application would be hiding of information about utilized security products which is often contained within event messages. To solve this problem, we need flexible mechanisms for *attack information sanitizing*.
- *Security Policies*
  Also the security policies which are to be enforced by local IDS may differ in various ways (e.g. in one domain, port scans are treated as a potential attack, whereas in another domain, they are completely neglected). This leads to alternating pools of messages with different priorities, levels of trust etc.. To handle this, either the system can use mechanisms for “Message Normalization” to homogenize incoming data or use analyzer techniques, which work independently of these properties.
- *Architecture*
  There are different ways to implement an architecture for cooperative intrusion detection. In earlier approaches, such as described in a generic paper of Frincke et al. [5], the term “Cooperative IDS” was interpreted as establishing direct communication links between the IDS nodes (or, in general, local event message collectors) in the affected domains. For dynamic CEs, this model has several disadvantages, and thus it is not applicable:

- *Traffic Overhead*
  A fully connected network of n domains would require $n \cdot (n-1)$ communication links between the IDS nodes. This leads obviously to a lot more network traffic than necessary.
- *Separate Information Sanitizing Policies*
  Since information about attacks is transferred to each other domain separately, a separate policy for sanitizing information has to be created and maintained, in accordance to the established trust levels.
- *Continuous Reconfiguration*
  Since in a dynamic CE new domains may be integrated and the trust levels between the existing ones may change, continuous reconfiguration efforts are necessary for up to $n \cdot (n-1)$ information flows.
- *CE Command Structures*
  Since the conventional command structures in CEs like NATO are organized in a hierarchical way, individually communicating IDS nodes are contradicting.

An alternative for a peer-to-peer structure is a star shape architecture with bi-directional links from the single domains to the centralized event message processing unit. But it also has to be mentioned that using only one central unit is a single point of failure, and therefore fallback mechanisms have to be integrated in the architecture.

- *Data Format and Protocols*
  To use event messages, which have been collected by different security tools from different vendors, we need a commonly agreed data format and message transport protocol. Fortunately, the Intrusion Detection Working Group of the IETF (IDWG) has released proposals for this, such as the XML-based IDMEF format [3] and the IDXP profile [4] for the BEEP protocol [12]. Many vendors of commercial systems claimed to support these formats in order to be interoperable with other systems.
- *Secure and Reliable Communication*
  To deliver event messages in a secure manner, we need according communication channels. Thus, mechanisms for information integrity, confidentiality and authentication are necessary. These services can be provided by adding an additional crypto layer to the communication channels, such as SSL/TLS as a part of IDXP/BEEP, which itself provides a reliable message transport on top of TCP.

Although goals like a *Coordinated Intrusion Response* are very important for reacting on distributed attacks, they are beyond the scope of this paper.

3.0 GENERAL ARCHITECTURE

According to the objections to a peer-to-peer structure of message flows in section 2, we suggest a different architecture: a centralized flow of attack information with central information processing capabilities, and additional feed back links to the local IDS nodes for early warning information, as depicted in fig. 1.

This architecture is based on our IDS infrastructure framework which provides generic pluggable components. One or more central units (each called a Meta IDS Console) is deployed for processing all messages received from different domains. These component contain information storage capabilities as well as message filtering and processing modules. GUIs are deployed for offline message inspection and for real-time display of incoming messages. The central analyzing component is a message anomaly detector, which is described in section 4. To avoid a single point of failure, additional consoles are acting as a simple fallback solution for the primary console.

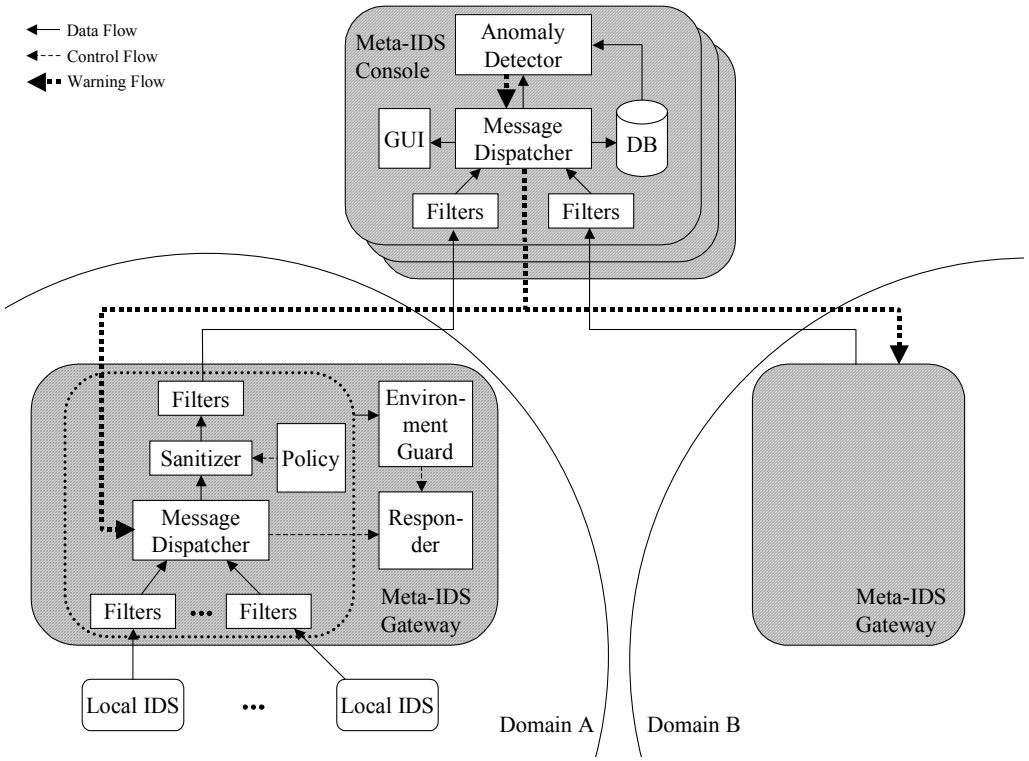


Figure 1: Architectural components of the meta IDS

Consoles receive event messages from several Meta IDS Gateways (GWs) which collect all messages occurring within a specific domain and distributes warning information back to the GWs. Thus, our GWs provide a suitable bi-directional interface to the domains; section 5 provides more information on our GW concept. Since the communication behaviour of GWs is similar to IDS agents, all previously developed infrastructure related countermeasures against Denial-of-Service attacks can be applied (e.g. component redundandencies, message overload protection, process environment integrity checks and process observation, see [6]) .

In our system, the IDMEF recommendation [3] of the IETF IDWG is used as a common data model for representing attack information as event messages. Accordingly, messages are encoded as XML documents and transmitted over network links via the IDXP/BEEP protocol [4].

4.0 ANOMALY DETECTION IN THE EVENT MESSAGE DATA MODEL

The structure of the entire system allows the usage of a method for detecting unusual activities. The event messages generated by the systems located in each domain are sent via the gateways to the meta IDS. The collected event messages are the base for an approach for detecting abnormal system behaviour. Due to the idea of the entire system, it is difficult to make assumptions on the type, the quality, or the frequency of event messages. The basic idea for surveying the current system state and for the detection of deviations (anomaly detection) is the continuous monitoring of the arriving event messages. The method used here was originally developed for monitoring and detecting network traffic anomalies (see [13]). In this application context, the method works as follows:

The typical structures of network traffic are quite stable. Fundamental changes are unusual. This allows to gather the traffic in regular intervals (this can be done using monitoring devices or traffic sniffer, in

switched networks it is necessary to use a monitoring port of the switch) and stored as a traffic matrix. This matrix can be seen as a graph $G = (V, K)$. Nodes (vertices) $v_i \in V$ of graph G represent communicating devices while edges $K_{i,j} \in K$ represent the communication between node v_i and v_j. The edges are weighted according to the intensity of the communication during the measurement period.

Those graphs can be partitioned into subgraphs using graph clustering algorithms. The clustering of the graph represents the typical communication structure of the monitored network. *Clustering* means finding a mapping of each node to one out of a set of several clusters. This exclusive classification is also called partitioning of the object set. Each object of an object set is assigned to exactly one cluster. In a more formal description, clustering is the partitioning of a graph $G = (V,K)$ into n cluster

- $C_i \subseteq V, C_i \neq \emptyset, 0 \leq i \leq n-1$ with
- $C_0 \cup C_1 \cup ... \cup C_{n-1} = V$ and
- $\forall 0 \leq i, j \leq n-1, i \neq j : C_i \cap C_j = \emptyset$

Sudden variations of these structures are regarded as anomaly. To discover these kind of variation, metrics are needed. Those metrics have to rate similarities (or dissimilarities) of consecutive clusterings $\mathcal{R}_t$ and $\mathcal{R}_{t+1}$.

This method used in the area of traffic structures needs some adaptations for the event message model presented in this section. Most of the event messages arriving at the meta IDS are suitable for building a graph. All the event messages containing detailed information on the destination (the attacked system) and the origin (assumed originator) belong to this category. Such a message generates a new edge k in the set of edges K of our message graph G. The edge k connects the nodes representing the origin v_{origin} and the destination $v_{destination}$.

Depending on the level of detail of the event messages, some gateways may just indicate the affected system but not specify the (assumed) originator. Depending on the configuration of the underlying security tools and meta IDS gateways, this information may not be available. In order to consider these event messages in the event message graph, a *pseudo node* representing these event message types may be assigned to the domains. An edge between such a pseudo node and a node representing an affected system represents an event message not containing an originator address.

The graph partitioned with graph clustering algorithms describes the typical structure of the incoming event messages. Variations from the typical message structure are seen as an anomaly. Those anomalies are reported as warning messages. The comparison measures used for anomaly detection in network traffic structures can be used in this domain as well.

When using anomaly detection methods, a major challenge is the sensitivity to false alarms (*false positives*) or unreported serious events (*false negatives*). This holds for the methods used here as well. A careful selection of parameters according to the usage scenario is necessary to keep false positive and false negative rates low. This method is currently being integrated into the overall prototype system, as described in section 7.

5.0 POLICY-CONFIGURABLE EVENT MESSAGE GATEWAYS

To provide an event message sanitizing service, domain-specific gateways (GWs) are needed. Their basic structure is outlined in the lower left part of figure 1. They perform the following tasks:

- Collecting all local event messages and passing them to the central meta IDS, independently of the local security tools that generate attack information.
- Inspecting message flow for sensitive information and filter (sanitize) messages according to the current IShP (see sect. 2).

- Accessing warning information from the meta IDS and providing it to the local security staffs and their supporting systems.
- Optionally perform filter operations on messages, such as normalizing message contents and adjusting priority values.

GWs are functionally controlled by the meta IDS, but configuring the IShP is a matter of the local domain and the responsible security staffs. Since IShPs are one of the most important issues concerning IDS in CEs, a more formal specification of the information sanitizing process within the GW is useful. It can be expressed as an event matching and transformation problem:

Given the local IShP, it should be possible to generate a set of conditional transformation rules in the shape of $R = (\{ E_0^M \}, \varnothing, E^T)$ as described in appendix A. But unfortunately, a useful information sanitizing process - as needed for our GWs - cannot solely rely on static text substitution. For example, when replacing IP addresses with fixed values, all information about network topology is lost. Obviously, this may hamper the intrusion detection process, especially when using traffic-related anomaly detection. Thus, we need a more flexible way of defining transformation rules: *submatching references*.
Let $s(E_0^M)$ be a set of qualified subexpressions in the matching template E_0^M, and $s(E_0^M, e)$ the set of substrings of a given event *e* which actually match the subexpressions in E_0^M. Now, if we extend the transformation template to

$$E^T : \Sigma \times s(E_0^M) \times P \rightarrow P(\Sigma),$$

we have the possibility to construct new events, based on submatchings of the old events. An example for this is the 'substitute' command of the standard Unix `sed(1)` tool (e.g. the command

```
s/192\.22\.([0-9]{1,3}\.[0-9]{1,3})/191\.72\.\1/
```

leads to the substitution of the 2-byte-prefix of matching IP addresses).
To extend the possibilities for the transformation rules even more, it is obviously useful to involve current system parameters like the time of the last matching, the current system time or the IP address of the gateway host as parameter set *P*.

6.0 EVENT MESSAGE AGGREGATION MODULES

In our approach, two kinds of message aggregation should help us in making high amounts of messages feasible: *redundancy filtering* and application of predefined rules for sets of correlated events (*combination detection*). Filter modules for event aggregation can be applied at different places in the architecture, as shown in figure 1. Like the message sanitizing techniques, as described in the previous section, we can reduce it to an event pattern matching and transformation problem.

6.1 Redundancy Filtering

In order to reduce the number of event messages, it is obviously necessary to avoid redundancies. Several messages with a “similar” content shall be merged into one single message, that represents the merged ones sufficiently. To specify the merging process of the aggregated messages, we can again utilize transformation templates E^M as described in section 5.

The definition of event “similarities” must be configurable, since it strongly depends on domain specific parameters, like deployed security tools and the security policy that has to be enforced. Therefore, we extend event matching templates E^M accordingly. Since we do not only need absolute (initial) matchings, but even *relative matchings* (i.e. depending on previous ones, defining the difference), we have to enhance

the matching process by introducing a second matching template E_1^M that refers to submatchings of the initial matching template E_0^M. Thus, for the set matching templates $E^M = \{E_0^M, E_1^M\}$, we have

$$E_0^M : \Sigma \to \mathrm{B}, \qquad E_1^M : \Sigma \times s(E_0^M) \to \mathrm{B}$$

Example: If all messages with identical classifications and source/target IP addresses are to be filtered, and if the message creation time is within a time range of one second,

- E_0^M has to contain qualified subexpressions for the message creation time and for source and target address to be able to refer to them later,
- E_1^M must contain a conditional expression for the values of source and target address, which are identical to the according values of the event *e* that matches the initial matching template E_0^M (referred as $e \dashv E_0^M$), and
- E_1^M must contain a conditional expression for the values of creation time, which are not more than 1 second away from the creation time of the initial matching event.

In contrast to (stateless) conditional transformations, where a separate matching of all templates $E^M{}_i$ is required for a transformation, we need two buckets

$$B_i(t) = \{e \mid e \dashv E_i^M\}, i = 0,1 \qquad \text{with} \qquad 0 \le |B_0(t)| \le 1, |B_1(t)| \ge 0 \forall t \in T$$

for previously matched events, at least to determine which template is currently to be matched next.

Additionally, we need the storage time t_{B_0} of the first event which has been stored in bucket B_0 to be able to indicate the end of the aggregation phase after a maximum storage time Δt_{B_0} has been exceeded. The transformation function does not only depend on the current event *e*, but also on the previously stored ones, i.e. we define

$$E^T : P(\Sigma) \times P(\Sigma) \times s(E_0^M) \times P \to P(\Sigma)$$

and for the transformation function f_R a redundandency filtering process algorithm that does not involve external parameters. Therefore it is straightforward:

Input: $R=(\{E_0^M, E_1^M\}, \varnothing, E^T)$, Δt_{B_0} and a sequence of events $e(t)$

Output: A sequence of event sets $\{e\}$, where similar events are summarized.

Algorithm:

$B_0(0) := \varnothing$; $B_1(0) := \varnothing$; $t_B := 0$ $t := 1$

```
while true do
      read e(t)
      if t_B0 ≠ 0 and t − t_B0 ≥ Δt_B then
            output E^T(B_0(t), B_1(t), s(E_0^M))
            B_0(t) := ∅ ; B_1(t) := ∅
            t_B0 := 0
      fi
      if e(t) ⊣ E_0^M and e(t) ⊣ E_1^M then
            output {e(t)}
```

```
        fi
        else if |B_0| ≠ 0 and e(t) ⊣ E_0^M
        then
                B_0(t) := {e(t)}
                t_B := t
        fi
        else if |B_0| > 0 and e(t) ⊣ E_1^M
        then
                B_1(t) := B_1(t−1) ∪ {e(t)}
        fi
        t := t+1
done
```

Note that this algorithm does not require that similar events are coming in sequentially, without having different messages in between them. Additionally, it is either possible to use multiple buckets $B_{i,j}(t)$ in parallel, but the number of buckets must be limited in order to avoid overload situations and DoS attacks. It is also possible to modify the later output event *e*' each time a matching occurs, instead of calculating *e*' = $E^T(B_0,B_1,s(E_0^M))$ after the end of the aggregation phase. Since then only one bucket for one event is needed, a lot of memory is saved.

6. 2 Combination Detectors

We define *event combinations* as correlated sets of event messages. Detectors for these combinations apply predefined rules for the events and their correlation. Note that our approach does not include the generation of correlation rules between event messages, but once they are specified (e. g. by a procedure as described by Julisch [7]), we are able to apply them on the message flow in a very flexible manner. Combination detectors are a generalized case of the redundancy filters as described above, since instead of having one absolute and one relative matching, we need a set of relative matching templates $E^M = \{E_1^M, ..., E_n^M\}$ to describe an event combination. Every relative matching may not only refer to the submatchings of the initial matching E_0^M, but also of the

$$E_i^M : \Sigma \times \prod_{j=0, j\neq i}^{n} s(E_j^M) \rightarrow \mathrm{B}, \forall i = 1, ..., n$$

We assume multiple buckets $B_i, 0 \leq i < n$ for the different matching templates. Thus it shall be possible to refer to all of the stored events in the buckets by extending E^T to:

$$E^T : \prod_{i=0}^{n} P(\Sigma) \times \prod_{j=0}^{n} s(E_j^M) \times P \rightarrow P(\Sigma)$$

With this approach, we can model different correlations which are important to detecting security relevant situations. To see how this works, we look at correlations between two matching templates E_0^M and E_1^M:

- *Classification Correlation*:
  Defining expressions in E_0^M and E_1^M which describe a subset of possible event classifications (e.g. enumerations or ranges of attack database IDs).

- *Temporal Correlation*:
  Defining a subexpression in E_0^M containing the time value and an arithmetic expression in E_1^M that defines the correlation (e.g. a maximum detection time difference).
- *Location Correlation*:
  Defining a subexpression in E_0^M containing the address value and an arithmetic expression in E_1^M that defines the correlation (e.g. belonging to the same subnet).

Of course, arbitrary combinations of all of the above may be defined to specify more complex correlations, e.g. *More than 100 TCP packages from address A to address B on port 22, containing potential NoOp instructions and one packet from B to A from port 22, containing text "uid=0(root)" which comes right after the last packet of the earlier type* which might be a serious indication for a well-known attack against the SSH login daemon.

7.0 INITIAL RESULTS

The architecture is implemented as a prototype system, which is called "MSIDI" (Meta SIDI), which is based on our earlier IDS prototype "SIDI" (Survivable Intrusion Detection Infrastructure, [6]), using its C++ class framework. Using instances of SIDI for providing event messages to the meta IDS and additionally developed gateways, all necessary infrastructure and management components are implemented, as well as the majority of the message filtering and processing features mentioned within this paper.

7.1 Anomaly Detector

Our approach for detecting anomalies in event message flows has been implemented, currently only looking at the source/target address properties of messages. At present, the detector is being integrated in the overall prototype system.

The evaluation of the detection approach is currently done using several data sources. One of those data sources is a selection of more than 400 Mbytes of real event messages representing the original data traffic of one of our test configurations. An additional test system is a system used for tests with artificially generated event messages mixed with real event messages. This test system can be used with input data, e.g. generated by a system simulating worm spreading. This simulator simulates the spreading of real worms (Code Red v2, Code Red II,...) or worms using modified or new spreading methods.

7.2 Information Sanitizing

Our approaches for information sanitizing, for redundancy filtering and for event combination detection presented in sections [5] and [6] are all reduced to the conditional event message transformation (see appendix A). A common way of performing complex transformations on XML formatted data is the application of XSLT Stylesheet [14]. Unfortunately, this recommendation does not include REs, and no suitable implementations of according extensions are available at the moment. Thus, for our first meta IDS prototype, we implemented an XML processor which recognizes POSIX.1 REs in XSLT templates, expands Boolean/arithmetic expressions for additional conditions and applies the matching and transformation on IDMEF messages.

An equivalent to the IP address prefix substitution example from sect. 5 is the following *transformation sheet* (integrated specification of matching and transformation template):

```
<address>
192\.22\.([0-9]{1,3})$v1$\.([0-9]{1,3})$v2$
  <condition>($v2<255)</condition>
  <transform>
```

```
    <xsl:copy>
     191.72.<xsl:value-of select="$X"/>.<xsl:value-of select="$Y"/>
    </xsl:copy>
  </transform>
</address>
```

The matchings of the last two bytes of the IP address `192.22.X.Y` are further on referred as variables `$X` and `$Y`. The additional condition for performing the transformation is that `$Y` is not the traditional broadcast suffix 255.

Using these basic transformation techniques, we have implemented the information sanitizing functionalities of the event message gateways as described in section 5. We are utilizing the GNOME libxml2 and libxslt libraries for XML processing to create highly configurable C++ filter modules which can be easily plugged into our infrastructure components.

To evaluate the efficiency of our filtering approaches, we created a number of transformation sheets and counted the number of messages per time frame which our sanitizing gateway is able to transform.
Using a PIII/1GHz PC with 256MB RAM, running Debian GNU/Linux, the message gateway is able to handle the amounts of message transformations, as shown in figure 2 (messages supplied by one domain IDS, no compiler optimizations). Our test results confirmed the assumption, that the way of specifying matching and transformation templates has a big impact on the filtering performance. Wherever applicable, different matching templates should be condensed into one. In this case, it is not necessary to analyze incoming messages up to their ends before the next matching template is applied.

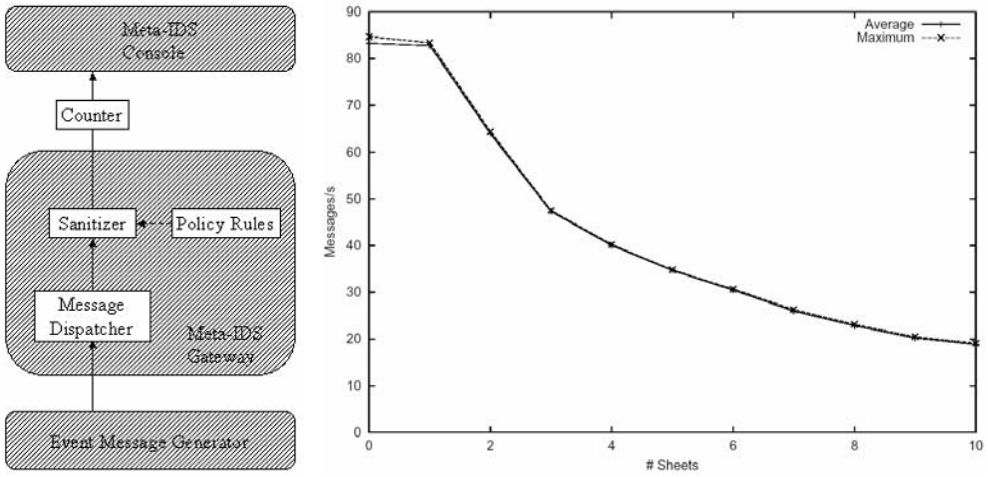


Figure 2: Scenario and results of the throughput tests for the information sanitizing gateway.

8.0 RELATED WORK

Cooperative IDS have been examined in many publications:

An early generic cooperative intrusion detection paper of Frincke et al. [5], identified many of the cooperation issues, such as policy conflicts, information sharing and filtering. In these premises and in the context of allowing different local audit tools to be used within the affected domains, the paper can be considered a basis for our work. A major difference to our work is the described architecture, which relies on direct communication links between the distributed audit data collectors.

The approach of Kim et al. [8] has several architectural analogies to our work. Unfortunately their approach relies on the fact that the suggested architectural components are installed all over the place which is to be monitored. This is not suitable for dynamic CEs, since every single domain has its own principles for selecting products and vendors.

The cooperating software entities, described in [9] as well as in [15], have similar concepts to our meta IDS gateway, but again, the architecture is on a peer-to-peer basis.

A lot of papers describe approaches for detecting correlations between alerts (e.g. [2], [10], [11]). Since our system only provides mechanisms for detecting predefined correlations of events, off-line correlation techniques are necessary to define the according detection rules. Due to our very general approach in specifying such rules, it should be feasible to import externally generated correlation rules.

9.0 SUMMARY AND FUTURE ACTIVITIES

This paper has presented an architecture to act as a cooperative intrusion warning system for dynamic heterogeneous coalition environments. Due to its structure, it meets several requirements for dynamic CEs, which cannot be met by other approaches relying on directly communicating domain-specific units.

The most important extensions are a central message processing unit where an anomaly-based detection module is included for providing early warning information about largely-scaled attacks, and integrated modules for information sanitizing and data reduction. For a prototypical implementation, we extended a previously developed distributed IDS infrastructure.

The anomaly detector module is based on a graph clustering technique which has been successfully deployed for IP traffic analysis. All event message processing tasks have been reduced to problems of Conditional Pattern Matching and Transformation (CPMT), which we have implemented as extended XML/XSLT Stylesheet processors. Thus, we are able to filter the message flow in a very flexible manner.

Currently, we are focussing on

- exploring the impact of other information contained in event messages on the graph-based anomaly detection approach,
- distinguishing the different reasons for detecting anomalies and extracting useful information for warning messages, and
- the implementation of the redundandency filter and the combination detector by extending the CPM techniques.

ACKNOWLEDGEMENTS

Parts of this work have been sponsored by the German Federal Office for Information Management and Information Technology of the Bundeswehr (Bundeswehr IT-Office). The authors would like to thank Michael Schmeing (FGAN/FKIE) for valuable comments on the topic.

References

[1] R. Coolen and H.A.M. Luiijf.
Intrusion Detection Systems - Generics and State of the Art.
Technical report, NATO/RTO Information Systems Panel, Task Group on Information Assurance (TGIA, IST-008), 2001.

[2] F. Cuppens and A. Miege.
Alert Correlation in a Cooperative Intrusion Detection Framework.
In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 202-215, Oakland, CA, May 2002. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.

[3] D. Curry and H. Debar.
Intrusion Detection Message Exchange Format - Data Model and Extensible Markup Language (XML) Document Type Definition.
IETF Internet Draft draft-ietf-idwg-idmef-xml-10.txt, January 2003.
IETF IDWG.

[4] B. Feinstein, G. Matthews, and J. White.
The Intrusion Detection Exchange Protocol.
IETF Internet Draft draft-ietf-idwg-beep-idxp-07.txt, October 2002.
IETF IDWG.

[5] D. A. Frincke, D. Tobin, J. C. McConnell, J. Marconi, and D. Polla.
A Framework for Cooperative Intrusion Detection.
In *Proc. 21st NIST-NCSC National Information Systems Security Conference*, pages 361-373, 1998.

[6] M. Jahnke.
An Open and Secure Infrastructure for Distributed Intrusion Detection Sensors.
In *Proceedings of the NATO Regional Conference on Communication and Information Systems (RCMCIS'02), Zegrze, Poland*, October 2002.

[7] K. Julisch.
Mining Alarm Clusters to Improve Alarm Handling Efficiency.
In *Proceedings of the NATO/RTO IST Workshop on Inforensics and Incident Response, The Hague, The Netherlands*, October 2002.

[8] Byoung-Koo Kim, Jong-Su Jang, and Tai M. Chung.
Design of Network Security Control System for Cooperative Intrusion Detection.
Lecture Notes in Computer Science, 2344:389-, 2002.

[9] G. Koutepas, F. Stamatelopoulos, V. Hatzigiannakis, and B. Maglaris.
An Adaptable Inter-Domain Infrastructure Against DoS Attacks.
In *Proc. of the International Conferences on Advances in Infrastructures for e-Business, e-Education, e-Science, e-Medicine on the Internet (SSGRR 2003)*, L'Acquila, Italy, January 2003.

[10] Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducassé.
M2D2: A formal data model for IDS alert correlation.
volume 2516, page 115, 2002.

[11] P. Ning, Y. Cui, and D. Reeves.
Constructing attack scenarios through correlation intrusion alerts.
Technical Report TR-2002-12, Department of Computer Science, North Carolina State University, August 14 2002.

[12] M. Rose.
RFC 3080: The Blocks Extensible Exchange Protocol Core.
http://www.ietf.org/rfc/rfc3080.txt, March 2001.

[13] J. Tölle and C. de Waal.
A Simple Traffic Model Using Graph Clustering For Anomaly Detection.
Proc. of Applied Simulation and Modelling (ASM) Crete, Greece, June 2002.

[14] W3C.
W3C Recommendation 16: XSL Transformations (XSLT) Version 1.0.
http://www.w3.org/, 1999.

[15] Q. Zhang and R. Janakiraman.
Indra: A Distributed Approach to Network Intrusion Detection and Prevention.
Technical report, University of Washington, St. Louis, 2003.

A NOTATION

Since many of the information flow problems in distributed IDS can be modelled as variations of conditional pattern transformations for event message, it is useful to describe the problem in a more formal fashion:

Let Σ be the set of event messages or *events*. An *event matching template* E is an expression that expands to a set of events i.e. $Expand(E) \in P(\Sigma)$. An event e *matches* a template E (noted as $e \dashv E$), if and only if $e \in Expand(E)$. The *matching function* of E is intuitively given as

$$E : \Sigma \to IB, \quad E(e) = \begin{cases} true, & \text{if } e \in Expand(E) \\ false, & \text{if } e \notin Expand(E) \end{cases}$$

Let $P = \{(p_1, \ldots p_m)\}$ be a set of arbitrary given time-variant parameter values. A *conditional transformation rule* R is a triple

$$R = (E^M, P, E^T)$$

with a set of *matching templates*

$$E^M = \{E_i^M \mid E_i^M : \Sigma \to IB, i = 0, \ldots, n\}$$

and a *conditional transformation template*

$$E^T : \Sigma \times P \to P(\Sigma).$$

Let T be the time interval of interest, $e(t) \in \Sigma \forall t \in T$ a sequence of events and $p(t) \in P$ a sequence of parameter values. The *conditional transformation function* of the rule R is

$$f_R : \Sigma \times T \to P(\Sigma) \text{ with}$$

$$f_R(e(t), t) = \begin{cases} E^T(e(t), p(t)), & \text{if } e(t) \dashv E_i^M, \forall i = 0, \ldots, n \\ \{e(t)\}, & \text{otherwise} \end{cases}$$

i.e., depending on the matches of the templates E_i^M to the input event $e(t)$ at time t, zero or more output events are generated by the filter. Note, that this mechanism is stateless, i.e. no input values are stored within the filter.

As a simple example: If event messages, encoded in the IDMEF format (see [3]), shall be transformed in other messages, which have a modified target address value, the according transformation rule is

$$R = (\{E_0^M\}, \varnothing, E^T)$$

i.e. the transformation depends only on the matching of the input event e onto E_0^M, otherwise e is not modified. Now we can define E^M as an XML/IDMEF formatted message with regular expressions (REs) as:

```
<Alert>
 <Target >
  <Node category="dns">
   <Address category="ipv4-net-mask">
    <address>
     192\.22\.[0-9]{1,3}\.[0-9]{1,3}
    </address>
    <netmask>
     255\.255\.255\.255
    </netmask>
   </Address>
  </Node>
 </Target>
 <Classification origin="bugtraqid">
  <name>124</name>
 </Classification>
</Alert>
```

Then E^T could be constructed as

```
<Alert>
 <Target>
  <Node category="dns">
   <Address category="ipv4-net-mask">
    <address>
     191.72.1.1
    </address>
   </Address>
  </Node>
 </Target>
</Alert>
```

If $e(t) \dashv E^M$ (i.e. messages concerning the "teardrop" attack against nodes with an address prefix of "192.22.") arrives at the filter, a new message e' is generated, which contains all elements of $e(t)$, with the exception of the target address, which will be constantly 191.72.1.1, and the old message is discarded.

Intrusion Tolerance for Unclassified Networked Systems

Yves Deswarte & David Powell
LAAS-CNRS
7 avenue du Colonel Roche
31077 Toulouse cedex 4
France

{Yves.Deswarte, David.Powell}@laas.fr

ABSTRACT

Information such as security advisories, emergency recommendations, e-government information, etc., is unclassified, but its availability and integrity may be vital. Such data are intended to be made widely available and thus need to be accessible through open networks such as the Internet. The systems distributing this kind of information are usually built from COTS hardware and software, since their functions do not require specific software or hardware development. Openness and use of COTS make these systems very vulnerable, and traditional security means are insufficient to achieve the required availability and integrity. In that case, fault tolerance can be viewed as a complementary, valuable technique to cope with possible intrusions, as well as accidental failures of system components.

This paper presents the techniques of intrusion tolerance, and describe some recent experimental architectures, developed by the European project MAFTIA and the DARPA project DIT.

1 INTRODUCTION

Many systems that store, process or distribute unclassified but vital information, are connected to open networks, such as the Internet, in order to interact easily with other systems, or to give a wide public access to important information: security advisories, emergency recommendations, e-government information, etc.

Even though confidentiality is not critical for these systems since their information is unclassified, integrity and availability might be vital, in particular in emergency or crisis situations. On the other hand, most of these systems use COTS hardware and software, for economic reasons, but also because these systems support classic widely-deployed applications, which do not require specific software or hardware development.

These two characteristics, open network connection and COTS hardware and software, make these systems very vulnerable to attacks, while at the same time attacks are becoming ever more frequent on the Internet. Facing this evolution, the usual security techniques are insufficient and fault-tolerance techniques are increasingly worthwhile. Nevertheless, intrusions are different from accidental faults and specific fault-tolerance techniques must be designed to cope with their peculiarities.

In particular, the fault independence assumptions, which can be justified for the kind of accidental faults that are addressed by most fault tolerance techniques, are not valid for deliberate attacks. This means that if one kind of intrusion can be successful on a part of the system, the same kind of intrusion will be successful on other similar parts. Consequently, hardware and software diversification are essential.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

The first section of this paper discusses the limits of conventional security techniques. Then the principles and techniques of intrusion tolerance are presented in Section 3. Sections 4 and 5 are dedicated to some recent experimental architectures, respectively developed by the European project MAFTIA and the DARPA project DIT.

2 SHORTCOMINGS OF CONVENTIONAL SECURITY TECHNIQUES

Computer and communication security relies mostly on user authentication and authorization, i.e. control of access rights. Authentication is necessary to identify each user with sufficient confidence, in order to assign him adequate privileges and to make him responsible and liable for his actions. Authorization aims to allow the user to perform only legitimate actions. As much as possible, authorization should obey the *least privilege principle*: at any time, a user can only perform the actions needed to achieve the task duly assigned to him. Authorization is implemented through protection mechanisms, which aim to detect and block any attempt by a user to exceed his privileges. Security officers can then detect such attempts and initiate legal actions, which in turn constitute deterrence against further attempts. Authentication, authorization, detection, retaliation and deterrence constitute the weapons of security defenders.

Unfortunately, these weapons are of little efficiency in the context considered in this paper:

- Unclassified information systems are often supposed to be accessed by a wide public, making strong authentication infeasible.
- COTS operating systems and application software contain many design flaws that can be exploited by attackers to circumvent protection mechanisms; when software companies develop and distribute patches to correct such flaws, relatively few system administrators apply the patches either because this would require more time or competence than available, or because the patches may disable certain features needed by other legitimate software.
- Most Internet protocols were designed thirty years ago, at a time when computing and communication resources were expensive and unreliable, and when intrusions were unlikely; so communication availability was the primary objective. Many facilities developed for that purpose can be diverted by malicious agents to perform denial of service attacks (e.g., by SYN flooding) or to multiply their efficiency (e.g., by smurfing), to by-pass protection mechanisms such as firewalls (e.g., by source routing), to hide their tracks (e.g., by IP address spoofing), etc.
- Due to harsh competition, most Internet Service Providers and telecommunication operators do not implement ingress filtering and trace-back facilities, which would help to locate and identify attackers.

3 INTRUSION TOLERANCE TECHNIQUES

3.1 Fault tolerance

Fault tolerance [1] is a technique that has proven to be efficient to implement computing systems able to provide a correct service despite accidental phenomena such as environment perturbations (external faults), failures of hardware components (internal physical faults), or even design faults such as software bugs.

According to the dependability terminology [2], *faults* are causes of errors, errors are abnormal parts of the computing system state, and *failures* happen when errors propagate through the system-to-user interface, i.e., when the service provided by the system is incorrect. When faults are accidental and sufficiently rare, they can be tolerated. To do so, errors must be *detected* before they lead to failure, and then corrected or *recovered*: this is the role of *error handling*. It is also necessary to diagnose the underlying faults (i.e., to

identify and locate the faulty components), so as to be able to isolate them, and then replace or repair them, and finally to re-establish the system in its nominal configuration: fault diagnosis, isolation, repair and reconfiguration together constitute *fault handling*.

To detect errors, two main classes of techniques can be used. The first class is made of likelihood checks, which consist in observing the system state, in particular certain values or events, and verifying their likelihood. This usually imposes only a small hardware or software overhead (*redundancy*). Among hardware likelihood checks, let us note that most microprocessors detect non-existing or unauthorized instructions and commands, non-existing addresses and unauthorized access modes, and that watchdogs can detect excessive execution durations. Software likelihood tests can be inserted into programs to check the values of certain variables, or the instants or sequences of certain events (*defensive programming*). Some error detecting codes can also be viewed as likelihood checks.

The other main class of error detection techniques consists in comparing several executions, carried out either sequentially on the same hardware, or on different hardware units. This requires more redundancy than the first class of error detection techniques, but it also assumes that a single fault would not produce the same effect (i.e., identical errors) on the different executions. If only internal physical faults are considered, the same computation can be run on identical hardware units, since it is very unlikely that each hardware unit would suffer an identical internal fault at the same execution instant to produce the same error. On the contrary, design faults would produce the same errors if the same process is run on identical hardware units, and thus the comparison of the executions would not detect discrepancies. In that case, it is necessary to diversify the underlying execution support (hardware and/or software), so that a single design fault would affect only one execution, or at least would affect differently the different executions [3].

To correct errors, one approach it to take the system back to a state that it had occupied prior to the detection of errors, i.e., to carry out rollback recovery. To be able to do that, it is necessary to have created and saved copies of the system state, known as recovery points or *checkpoints*. Another error correction technique is called *forward recovery*, which consists of replacing the erroneous system state by a new, healthy state, and then continuing execution. This is possible, for example, in certain real-time control systems in which the system can be re-initialized and input data re-read from sensors before continuing execution. Finally, a third technique consists in “masking” errors; This is possible when there is enough redundant state information for a correct state to be built from the erroneous state, e.g., by a majority vote on three (or more) executions.

In most cases, the efficacy of fault tolerance techniques relies on the fact that faults are rare phenomena that occur at random points in time. It is thus possible, for example in a triple modular redundant architecture, to suppose that is unlikely for a second unit to fail while a failed unit is being repaired. This hypothesis is unfortunately not valid when intrusions are considered. An attacker that succeeds in penetrating one system can pursue his attack on that system, and also simultaneously attack other similar systems.

3.2 Vulnerability, attack, intrusion

An intrusion occurs when an attack is able to successfully exploit a vulnerability (a design or configuration fault, in the terminology of dependability) [2]. The intrusion may be considered as an internal fault, which can cause errors that may provoke a system security failure, i.e., a violation of the system’s security policy. As discussed earlier, it would be illusory to imagine that attacks over the Internet can be prevented. Similarly, it is impossible to eliminate all possible vulnerabilities. For example, the very fact that a system is connected to the Internet is in itself a vulnerability, but what use would a Web server be if it were not connected to the net?

It is therefore of interest to *tolerate intrusions*, i.e., to arrange things such that an intrusion in one part of the system has no consequence on its overall security. To do that, we can use techniques developed in the traditional field of fault tolerance. However, there are two main problems:

- It should be made very difficult for the same type of attack to succeed in different parts of the system. This means that each "part" of the system must be sufficiently protected in its own right (so that there are no trivial attacks), and should ideally be diversified.
- An intrusion into a part of the system should not allow the attacker to obtain confidential data. This is especially important in that redundancy, which is necessary for fault tolerance, may result in more alternative targets for hackers to attack.

If these problems can be solved, we can apply to intrusions the techniques that have been developed for traditional fault tolerance: error handling (detection and recovery) and fault handling (diagnosis, isolation, repair, reconfiguration). In the context of intrusions, specific detection techniques have been developed. These have been named "intrusion detection" techniques, but it should be noted that they do not directly detect intrusions, but only their effects, i.e., the errors due to intrusions (or even due to attacks which did not successfully cause intrusions).

The so-called intrusion detection techniques may be divided into two categories: anomaly detection and misuse detection (see Figure 1). Anomaly detection consists in comparing the observed activity (for example, of a given user) with a reference "normal activity" (for the considered user). Any deviation between the two activities raises an alert. Conversely, misuse detection consists in comparing the observed activity with a reference defining known attack scenarios. Both types of detection techniques are characterized by their proportions of false alarms (known as false positives) and of undetected intrusive activities (known as false negatives). In the case of anomaly detection, one can generally adjust the "threshold" or, by analogy with radar systems, the "gain" of the detector to choose a point of operation that offers the best compromise between the proportions of false positives and false negatives. On the other hand, misuse detection techniques have the advantage of identifying specific attacks, with few false positives. However, they only enable the detection of known attack symptoms. In both cases, it should be noted that detection is based on likelihood checks.

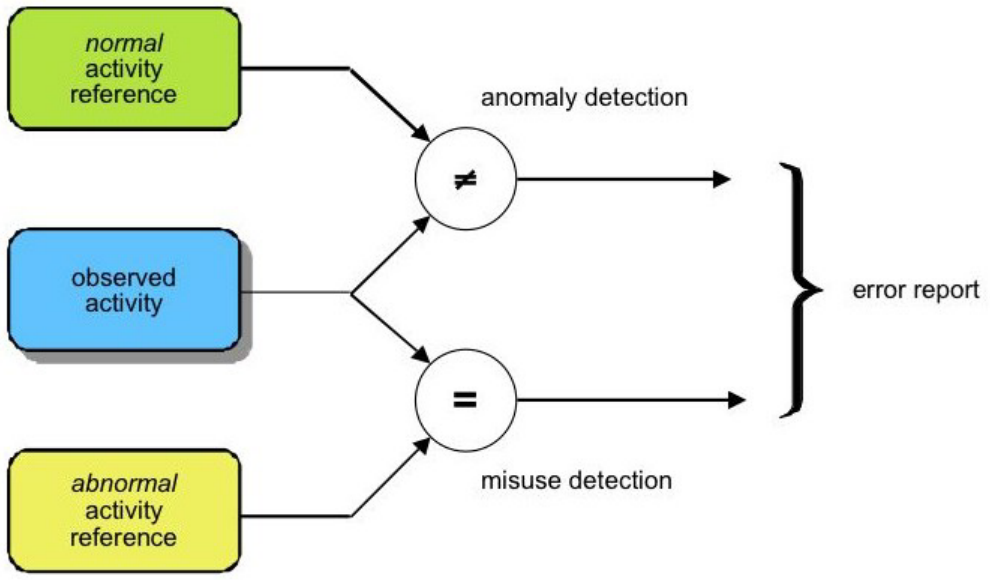


Figure 1: Intrusion detection paradigms

To correct the damage caused by the intrusion, one may, like in traditional fault tolerance, carry out backward recovery (if one has taken the precaution of maintaining up-to-date backups) or forward recovery (if one can rebuild a healthy state), but it is often easier and more efficient to mask errors, using some form of active (or modular) redundancy.

3.3 Fragmentation, redundancy, scattering

Several years ago, we developed an error masking technique, called "fragmentation, redundancy and scattering, or FRS", aimed at protecting sensitive data and computations [4]. This technique exploits distribution of a computing system to ensure that intrusion into part of the system cannot compromise the confidentiality, integrity and availability of the system. Fragmentation consists of splitting the sensitive data into fragments such that a single isolated fragment does not contain any significant information (confidentiality). The fragments are then replicated so that the modification or the destruction of fragment replicas does not impede the reconstruction of correct data (integrity and availability). Finally, scattering aims to ensure that an intrusion only gives access to isolated fragments. Scattering may be: *topological*, by using different data storage sites or by transmitting data over independent communication channels, or *temporal*, by transmitting fragments in a random order and possibly adding false padding fragments. Scattering can also be applied to privileges, by requiring the cooperation of several persons with different privileges in order to carry out some critical operation (separation of duty).

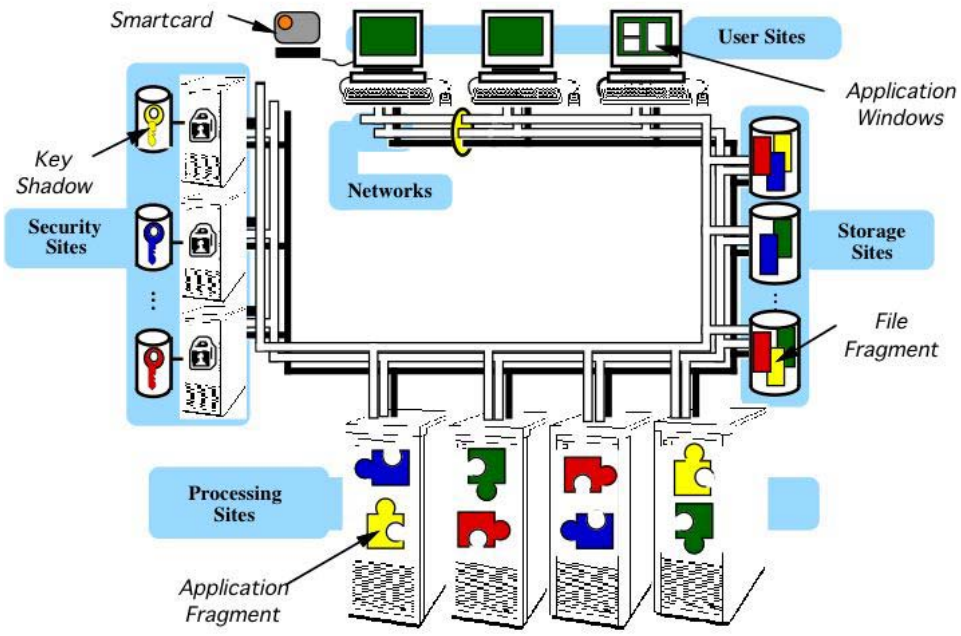


Figure 2 — Fragmentation-replication-scattering in Delta-4

The FRS technique was originally developed in the Delta-4 project [5] for file storage, security management and data processing (see Figure 2). For file storage, fragmentation is carried out using simple cryptographic techniques and fragment naming employs a secret key one-way function. The fragments are sent over the network in a random order, which means that one of the hardest tasks for an intruder would be to sort all the fragments into the right order before being able to carry out cryptanalysis. For security management, the principle resides in the distribution of the authentication and authorization functions between a set of sites administrated by different people so that failure of a few sites or misfeasance by a

small number of administrators do not endanger the security functions. On these sites, non-sensitive data is replicated, whereas secret data is fragmented using threshold cryptographic functions. Finally, for data processing, two data types are considered: a) numerical and logical data, whose semantics are defined by the application, (b) contextual data (e.g., character strings) that is subjected only to simple operations (input, display, concatenation, etc.). In this scheme, contextual data is ciphered and deciphered only on a user site during input and display. In contrast, context data is subjected to successively finer fragmentation until the fragments do not contain any significant information. This is achieved using an object-oriented decomposition method.

3.4 The MAFTIA project

The techniques developed in Delta-4 are well adapted to predominately homogeneous applications that are distributed over a LAN. However, they are not directly transposable to Internet, especially when the concerned applications involve mutually suspicious companies or organizations. In this case, it is no longer possible to manage security in a homogeneous way.

The European project MAFTIA was directly aimed at the development of intrusion-tolerant Internet applications [6]. Protocols and middleware were developed to facilitate the management of fault-tolerant group communications (including tolerance of Byzantine faults), possibly with real-time, confidentiality and/or integrity constraints [7, 8, 9]. In particular, the developed protocols and middleware enabled the implementation of trusted third parties or TTPs (e.g., a certification authority) that tolerate intrusions (including administrator misfeasance) [10]. Particular attention was paid to intrusion detection techniques distributed over Internet, since intrusion detection not only contributes to intrusion tolerance, but is itself an attractive target for attack. It is thus necessary to organize the intrusion detection mechanisms in such a way as to make them intrusion-tolerant [11]. Furthermore, the project developed an authorization scheme for applications involving mutually suspicious organizations [12]. An authorization server, implemented as an intrusion-tolerant TTP, checks whether each multiparty transaction is authorized. If that is so, the server generates the authorization proofs that are necessary for the execution of each component of the transaction (invocations on elementary objects). On each of the sites participating in the authorization scheme, a reference monitor, implemented on a JavaCard, checks that each method invocation is accompanied by a valid authorization proof. The scheme is intrusion-tolerant in the sense that the corruption of a participating site does not allow the intruder to obtain any additional privileges regarding objects residing on other sites [12].

3.5 The DIT project

In cooperation with SRI International, we are participating in the development of the DIT (Dependable Intrusion Tolerance) architecture [13]. The objective is to be able to build Web servers that continue to provide correct service in the presence of attacks. For this type of application, confidentiality is not essential, but integrity and availability must be ensured, even if the system is under attack from competent attackers. It is thus essential that a successful attack on one component of the system should not facilitate attacks on other components. The architecture design is thus centered on a diversification approach.

The architecture is composed of a pool of ordinary Web servers, using as much diversification as possible at the hardware level (Sparc, Pentium, PowerPC, etc.), the operating system level (Solaris, Microsoft Windows, Linux, MacOS, etc.) and Web application software level (Apache, IIS, Enterprise Server, Openview Server, etc.) (see Figure 3). Only the content of the Web pages is identical on each server. There are sufficient application servers at a given redundancy level (see below) to ensure an adequate response time for the nominal request rate. The servers are isolated from the Internet by *proxies*, which are implemented by purpose-built software executed on diversified hardware. Requests from the Internet, filtered by a firewall, are taken into account by one of the proxies acting as a *leader*. The leader distributes the requests to multiple Web servers and checks the corresponding responses before returning them to the

request initiator. The back-up proxies monitor the behavior of the leader by observing the firewall/proxy and proxy/server networks. If they detect a failure of the leader, they elect a new leader from among themselves. The proxies also process alarms from intrusion detection sensors placed on the Web servers and on both networks.

Depending on the current level of alert, the leader sends each request to one server (simplex mode), two servers (duplex mode), three servers (triplex mode) or to all available servers. Each server prepares its response and then computes an MD5 cryptographic checksum of this response and send it to the leader. In simplex mode, the server also sends its response to the leader, which recomputes the checksum and compares it to the one sent by the server. In duplex mode, the leader compares the two checksums from the servers and, if they concur, requests one the responses, which is verified by recomputing the checksum. In triplex or all-available modes, the checksums are subjected to a majority vote, and the response is requested from of the majority servers.

The alert level is defined as either a function of recent alarms triggered by the intrusion detection mechanisms or other error detection mechanisms (result cross-checking, integrity tests, etc.), or by information sent by external sources (CERTs, other trusted centers, etc). The redundancy level is raised towards a more severe mode (higher redundancy level) as soon as alarms are received, but reverts to a less severe mode (lower redundancy level) when failed components have been diagnosed and repaired, and when the alarm rate has decreased. This adaptation of the redundancy level is thus tightly related to the detection, diagnosis, reconfiguration and repair mechanisms. In the case of read-only data servers, such as passive Web servers, repair involves just a simple re-initialization of the server from a back-up (an authenticated copy on read-only storage).

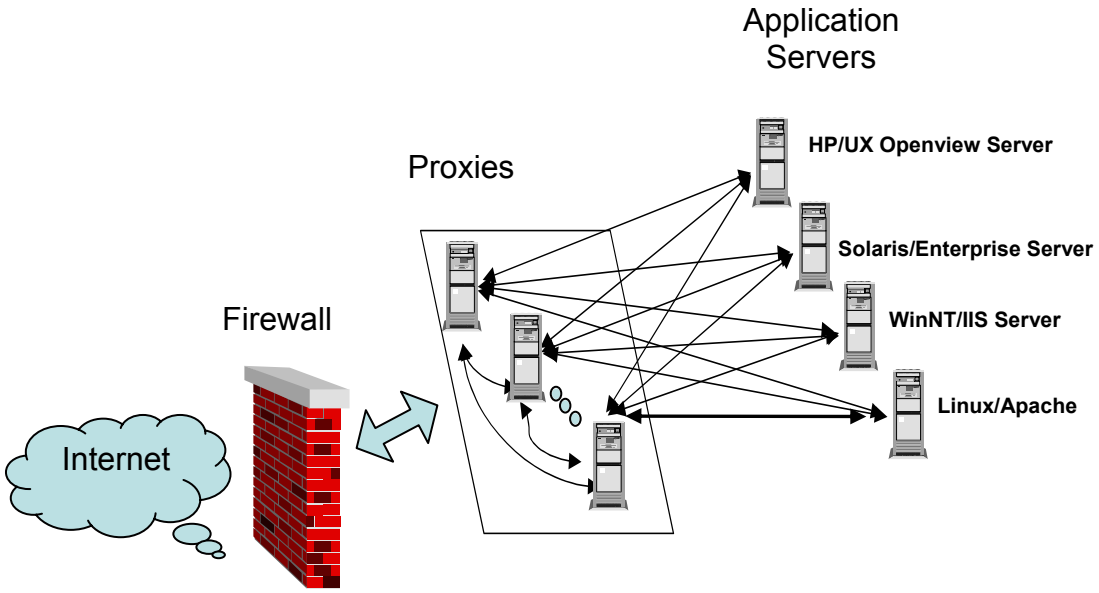


Figure 3: DIT architecture

Diversification renders the task of the attacker as difficult as possible: when an attacker sends a Web page request (the only means for him to access the application servers), he does not know towards which servers his request will be forwarded and thus which hardware or software will process it. Even if he were able to design an attack that would be effective on all server types (except maybe for denial-of-service attacks, which are easy to detect), it would be very difficult to cause redundant servers (in duplex mode and above) to reply in exactly the same incorrect way.

4 CONCLUSION

Given the current rate of attacks on Internet, and the large number of vulnerabilities in contemporary computing systems, intrusion tolerance appears to be a promising technique for implemented more secure applications, particularly with diversified hardware and software platforms. There is of course a price to pay, since it is expensive to support multiple heterogeneous systems. However, this is probably the price that must be paid for security in an open, and therefore, uncertain world.

5 BIBLIOGRAPHY

[1] J. Arlat, Y. Crouzet, Y. Deswarte, J.-C. Laprie, D. Powell, P. David, J.-L. Dega, C. Rabéjac, H. Schindler, J.-F. Roucailles, "Fault Tolerant Computing", in *Encyclopedia of Electrical and Electronic Engineering*, John G. Webster, Ed., Wiley-Interscience, Volume 7, 1999, pp.285-313.

[2] A. Avizienis, J.-C. Laprie, B. Randell, "Fundamental Concepts of Dependability", in *Proc. 3rd Information Survivability Workshop*, IEEE CS Press, 24-26 October 2000, Boston, MA, USA, pp. 7-12.

[3] Y. Deswarte, K. Kanoun, J.-C. Laprie, "Diversity against accidental and deliberate faults", in *Computer Security, Dependability and Assurance*: *From needs to Solutions*, P. Amman, B.H. Barnes, S. Jajodia & E.H. Sibley Eds., IEEE Computer Society Press, 1999, pp. 171-182.

[4] Yves Deswarte, Laurent Blain, Jean-Charles Fabre, "Intrusion Tolerance in Distributed Systems", *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, Oakland (USA), 20-22 May 1991, pp. 110-121.

[5] D. Powell, Ed., *Delta-4: a Generic Architecture for Dependable Distributed Computing,* in Research Reports ESPRIT series, Springer-Verlag, 1991, ISBN 3-540-54985-4, 484 pages.

[6] D. Powell, A. Adelsbasch, C. Cachin, S. Creese, M. Dacier, Y. Deswarte, T. McCutcheon, N. Neves, B. Pfitzmann, B. Randell, R. Stroud, P. Veríssimo, M. Waidner, "MAFTIA (Malicious- and Accidental-Fault Tolerance for Internet Applications)", *Sup. of the Proceedings of the 2001 International Conference on Dependable Systems and Networks (DSN2001)*, Göteborg (Sweden), 1-4 July 2001, pp. D-32-D-35.

[7] Michael Backes and Christian Cachin, "Reliable Broadcast in a Computational Hybrid Model with Byzantine Faults, Crashes, and Recoveries", in *Proc. Intl. Conference on Dependable Systems and Networks (DSN-2003)*, IEEE Computer Society Press, Washington, CA, USA, June 2003, pp. 37-46.

[8] M. Correia and L. C. Lung and N. F. Neves, P. Veríssimo, "Efficient Byzantine-Resilient Reliable Multicast on a Hybrid Failure Model", in *Proceedings of the 21th IEEE Symposium on Reliable Distributed Systems (SRDS 21)*, IEEE Computer Society Press, October 2002, Suita, Japan, pp. 2-11.

[9] M. Correia, P. Veríssimo, Nuno F. Neves, "The Design of a COTS Real-Time Distributed Security Kernel", in *Proceedings of the Fourth European Dependable Computing Conference* (*EDCC 4*), October 2002, Toulouse, France, pp. 234-252.

[10] Christian Cachin, "Distributing trust on the Internet", in *Proc. Intl. Conference on Dependable Systems and Networks (DSN-2001)*, IEEE Computer Society Press, Göteborg (Sweden), 1-4 July 2001, pp. 183-192.

[11] H. Debar and A. Wespi, "Aggregation and Correlation of Intrusion-Detection Alerts", in *Proceedings of Recent Advances in Intrusion Detection (RAID 2001)*, LNCS 2212, Springer, 2001, W. Lee, L. Mé, A. Wespi (eds.), pp. 85-103.

[12] Yves Deswarte, Noreddine Abghour, Vincent Nicomette, David Powell, "An Intrusion-Tolerant Authorization Scheme for Internet Applications", in *Sup. of the Proceedings of the 2002 International Conference on Dependable Systems and Networks (DSN2002)*, Washington, D.C. (USA), 23-26 June 2002, pp. C-1.1 — C-1.6.

[13] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte, B. Dutertre, J. Levy, H. Saïdi, V. Stavridou, T. Uribe, "An Adaptative Intrusion-Tolerant Server Architecture", *Proceedings of the 10th International Workshop on Security Protocols*, Cambridge (UK), April 2002, to appear in Springer LNCS Series.

Selecting Appropriate Counter-Measures in an Intrusion Detection Framework

F. Cuppens, S. Combault, T. Sans
GET/ENST-Bretagne
2, rue de la Châtaigneraie
35576 Cesson Sévigné
France

{frederic.cuppens, sylvain.combault, thierry.sans}@enst-bretagne.fr

Abstract

Since current computer infrastructures are increasingly vulnerable to malicious activities, intrusion detection is necessary but unfortunately not sufficient. We need to design effective response techniques to circumvent intrusions when they are detected. Our approach is based on a library that implements different types of counter-measures. The idea is to design a decision support tool to help the administrator to choose, in this library, the appropriate counter-measure when a given intrusion occurs. For this purpose, we formally define the notion of anti-correlation which is used to determine the counter-measures that are effective to stop the intrusion. Finally, we present a platform of intrusion detection, called DIAMS, that implements the response mechanisms presented in this paper.

Keywords : Intrusion detection, IDMEF, response, counter-measures, correlation, anti-correlation.

1 Introduction

Current systems that compose distributed computer infrastructures are increasingly vulnerable to intrusions and malicious activities. Several approaches have been suggested to detect such intrusions [1, 10, 12]. However, it is generally considered that current intrusion detection systems produce very large volume of alerts, including true alerts but also many false positives (see [4, 8] for instance). This is why recent research work attempts to understand and model intrusion strategies to provide a more global and precise diagnostic of the intrusion [5, 9]. These approaches are interesting and represent a step in the right direction but detecting intrusion is not sufficient. It is also necessary to develop automated defenses capable of appropriate responses to counter intrusions when they occur.

Several response strategies are possible including launching counter measures against the intruder to prevent his or her malicious activity to proceed or acting on the target system to stop the intrusion and recover in a safe state. Direct responses against the intruder is a complex problem that includes several technical difficulties (in particular, it is necessary to precisely identify the origin of the intrusion) and legal and ethic complications (directly acting on the intruder is generally viewed as illegal activities). In this paper, we shall not consider this type of response and actually focus on responses that consist in acting on the target system.

When an intrusion occurs, the appropriate response on the target system generally depends on the type of intrusion being performed. For instance, the responses will not be the same in the case of a denial of service (DOS) attack or a user to root (U2R) attack. Thus, our approach is based on a library of responses that contains different types of possible counter-measures which may be launched to stop intrusions. The problem addressed in this paper is to choose the appropriate counter-measure when a given intrusion occurs. This may represent a complex task for the administrator to make such a choice and some support might be useful to help the administrator. It is also necessary to fix the parameters of the response. For instance when the response consists in closing a given connection, the IP addresses of the source and target must be appropriately fixed before launching the response.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

In this paper, we suggest an approach to define decision mechanisms to help the administrator to choose, in the response library, the counter-measure candidates when an intrusion occurs and to present them to the administrator with the appropriate parameters to circumvent the intrusion. Once the administrator selects a counter-measure, this counter-measure with the appropriate parameters is automatically executed to stop the intrusion in the target system.

Our approach is based on a logical formalization of both attacks and counter-measures. This formalism is used to derive, from the attack description (especially the effects of an attack on the target system), one or several counter-measures that may circumvent the attack. For this purpose, we define the notion of *anti-correlation*. This notion is used to determine the counter-measures that will have a negative effect on the attack and therefore will enable the administrator to stop this attack.

The remainder of this paper is organized as follows. Section 2 presents the notion of response and suggests several types of response. In section 3, we present our formalism to model attacks and counter-measures. Our formalism is based on LAMBDA, a language suggested in [6] to model attacks. In this section, we also suggest using LAMBDA to model counter-measures. Section 4 recalls the definition of correlation [5] and introduces the notion of anti-correlation. In section 5, we show how to use anti-correlation to determine relevant counter-measures (1) to act on the objective of an intrusion or (2) to cut an ongoing attack scenario by acting on a given step of this scenario. We also present how our approach provides means to parameterize the selected counter-measures. Section 6 gives an example to illustrate the approach and presents DIAMS, the platform of intrusion detection that includes the response mechanisms we have presented in this paper. Finally, section 7 concludes the paper and suggests several possible extensions to our approach.

2 Response mechanism and counter-measure

Even though some improvements were made recently, current Intrusion Detection Systems (IDS) propose few response mechanisms in addition to alerts and reports. There is only a small variety of response techniques and the decision criteria that are used to activate the response remain often simplistic. Moreover, in a context of exploitation, security administrators generally balk at using the most interesting responses like automatic reconfiguration of firewalls or routers. This is due to lack of confidence in the capabilities of the IDS to take the right decision. Administrators also fear of not controlling the consequences of the automation of counter-measures. Lastly, the objective of most responses consists in stopping an ongoing attack. More elaborate responses that are effective to automatically correct the detected vulnerabilities, remain marginal.

In [7], the following taxonomy of counter-measures was suggested:

- Information: Specific action that raises an alert for the security administrator. This action can be launched after detecting an intrusion of sufficient severity.
- Deterrence: Action performed against the intruder so that he or she will be willing to stop his or her malicious activity. For instance, a message sent to the intruder to notify that his or her malicious action were detected is the simplest (but not always effective) form of deterrence.
- Correction: Action to modify the system state to correct an identified vulnerability or a system miss-configuration with respect to the security policy. For instance, installing a patch is a form of correction.
- Compensation: Action performed to block the attack but without correction. The system is still vulnerable but the response prevent the intruder from performing his or her intrusion. For instance, it is possible to stop a vulnerable service, or close the connection between the intruder and the target (using a TCP-Reset), or reconfigure a firewall to block the attack source.

We accept this taxonomy but we make a difference between actions that change the state of the system to be protected and other actions that not cause such a change. In the remainder of this paper, we shall actually consider actions that change the system state, that is correction and compensation. Since information and deterrence have respectively an effect on the security administrator or the intruder, they are not included in our analysis.

To avoid confusion, we make a distinction between the notions of response and counter-measure. In the following, we call *counter-measure* any action used as a compensation or a correction. Therefore, a counter-measure changes the system state so that the intrusion is stopped. We define *response* as the decision mechanism used to choose the adequate counter-measure when an intrusion is detected.

Our approach of response mechanism is integrated in the recognition process of the intruder's intentions presented in [3]. When an intrusion scenario is identified, we can anticipate on the objective that the intruder attempt to achieve and on the future attack that he or she will perform to achieve it. Thus, a response is an action that modifies the system state to prevent the intruder to achieve his or her goal. As explained in the following section, a goal can be an intrusion objective or a future attack.

3 Modelling intrusion and counter-measure

In this section, we present our formalism, based on LAMBDA [6], to model both intrusions and counter-measures.

3.1 Modelling attack and intrusion objective in LAMBDA

LAMBDA is the acronym for LAnguage to Model a dataBase for Detection of Attacks. It is used to provide a logical description of an attack. This description is generic, in the sense that it does not include elements specific to a particular intrusion detection process.

A LAMBDA description of an attack is composed of several attributes:

- **pre-condition** defines the state of the system required for the success of the attack.
- **post-condition** defines the state of the system after the success of the attack.
- **detection** is a description of the expected alert corresponding to the detection of the attack.
- **verification** specifies the conditions to verify the success of the attack[1].

We define an intrusion objective as a specific system state [3]. This state is characteristic of a violation of the security policy. A LAMBDA description of an intrusion objective is composed by only one attribute:

- **state** defines the state of the system that corresponds to a security policy violation.

3.2 How to use LAMBDA

LAMBDA is used to describe possible violations of security policy (intrusion objective) and possible actions (attack) an intruder can perform on a system to achieve an intrusion objective. This database of LAMBDA descriptions is used to recognize an intrusion process and predict the intention of the intruder.

Let us present an example of intrusion modelled with LAMBDA. First an intruder scans port 139. If it is open, he concludes that the Operating System is Windows and uses the NetBios service. The intruder can then execute a Winnuke attack on this target system that will cause a denial of service. Figures 1 and 2 respectively give the description in LAMBDA of the Scan and Winnuke attacks performed by an *Agent* on a given *Host*.

attack	scan-port(Agent,Host,Port)	
pre:	open(Host,Port)	– Port is open on Host
detection:	classification(Alert,'TCP-Scan')	– the alert classification is 'TCP-Scan'
	$\wedge$ source(Alert,Agent)	– the source in alert is Agent
	$\wedge$ target(Alert,Target)	– the target in alert is Host
	$\wedge$ target_service_port(Alert,Port)	– the scanned port is Port
post:	knows(Agent,open(Host,Port))	– Agent knows that Port is open
verification:	true	– always true

Figure 1: *Scan* Attack performed by an *Agent* on a given *Host*

There is a violation of security policy when a web server goes down. This is represented by the intrusion objective presented in figure 3.

[1]The alerts launched by IDS generally provide evidence of the occurrence of some malicious events but are not sufficient to conclude that these events will actually cause some damage to the target system. This depends on the system state when the malicious events occur. This is why a LAMBDA description also includes a verification attribute that provides conditions to be checked to conclude that the attack is a success.

attack	winnuke(Agent,Host,Service)	
pre:	use_os(Host,windows)	– OS on Host is Windows
	$\wedge$ use_service(Host,'Netbios')	– Host uses Netbios service
	$\wedge$ open(Host,139)	– Port 139 is open on Host
detection:	classification(Alert,'Winnuke')	– alert classification is winnuke
	$\wedge$ source(Alert,Agent)	– source in alert is Agent
	$\wedge$ target(Alert,Host)	– target in alert is Host
post:	deny_of_service(Host)	– deny of service on Host
verification:	unreachable(Host)	– Host does not reply

Figure 2: *Winnuke* Attack performed by an *Agent* on a given *Host*

objective	webserver_failure(Host)	
State:	deny_of_service(Host)	– deny of service on Host
	$\wedge$ server(Host,http)	– Host is an http server

Figure 3: Intrusion objective: Denial of service on a web server

As we can see in these examples, each LAMBDA description uses several variables (corresponding to terms starting with an upper case letter). When an alert can be associated with a LAMBDA description through the *detection* attribute, then we can unify variables with values. We call *attack occurrence* a LAMBDA description where variables have been unified with values.

3.3 Using LAMBDA to model counter-measure

We suggest adopting the same formalism to model counter-measure. Thus, a counter-measure have similar attributes as an attack. The main difference is that the *detection* attribute associated with an attack is replaced by the attribute *action*. This leads to the following model for counter-measures:

- **pre-condition** defines the system state required for the success of the counter-measure.
- **post-condition** defines the system state after applying the counter-measure.
- **action** defines the actions necessary to perform the counter-measure.
- **verification** specifies the conditions to verify the success of the counter-measure.

Figure 4 provides an example of counter-measure specified in this model. It consists in closing a connection between a given *Source* and a given *Target* connected through a *Port-Source* and a *Port-Target*.

counter-measure	close-remote-access(Source,Target)	
pre:	remote-access(Source,Target)	– Source has a remote access to Target
action:	TCP-Reset(Source,Target)	– A TCP-Reset closes the connection
post:	not(remote-access(Source,Target))	– Connection closed between Source and Target
verification:	not(TCP-Connection(Source,Port-Source,Target,Port-Target))	– Verify that the connection is closed on Target

Figure 4: Counter-measure: Closing a TCP connection

As for the attacks and objectives, the approach is to use this formalism to specify a library of possible counter-measures that apply to the system to block an intrusion. We shall now define a response mechanism to select the adequate counter-measures for a detected scenario. This mechanism is based on a principle called *anti-correlation* which is close to the *correlation* principle suggested in [5]. These two principles are formally presented in the following section.

4 Correlation and anti-correlation

Our response mechanism is based on recognizing the intruder's intentions. Using LAMBDA, [5] shows how to correlate detected attacks to identify a scenario. Section 4.1 recalls the definition for the correlation principle. It is then possible to extrapolate this scenario to predict future attacks that the intruder will probably perform and the objective that he attempts to achieve. When several possible scenarios are extrapolated, [2] suggests an approach to define an order of preference between these scenarios to select the most likely ones.

To design the response process, we suggest a second notion, called *anti-correlation* that is formally defined in section 4.3 and then used in section 5 to select the counter-measure candidates in the response process.

4.1 Correlation

Our approach of correlation is based on the unification principle [13] on predicates[2]. Let a and b be two LAMBDA descriptions of attacks. $post_a$ is the set of literals of *post-condition*[3] of a and pre_b is the set of literals of *pre-condition* of b.

Direct correlation: a and b are directly correlated if the following condition is satisfied:

$\exists\ E_a$ and E_b such that
$(E_a \in post_a \wedge E_b \in post_b)$ or $(not(E_a) \in post_a \wedge not(E_b) \in pre_b)$
and E_a and E_b are unifiable through a most global unifier θ.

[5] also defines the notion of *Knowledge gathering correlation*, a variation of the above definition of correlation that is useful to integrate, in the detection process, preliminary steps the intruder performs to collect data on the target system. This notion is defined as follows:

Knowledge gathering correlation: a and b are knowledge gathering correlated if the following condition is satisfied:

$\exists\ E_a$ and E_b such that
$(knows(Agent, E_a) \in post_a \wedge E_b \in post_b)$
or $(knows(Agent, not(E_a)) \in post_a \wedge not(E_b) \in pre_b)$
and E_a and E_b are unifiable through a most global unifier θ.

As an example, there is a knowledge gathering correlation between the Scan attack (see figure 1) and the Winnuke attack (see figure 2) through the predicate *open* and the unifier that matches variable *Host* in both attack definitions and variable *Port* in the Scan attack to constant 139. This means that an intruder who knows that port 139 is open on a given host, can then perform a Winnuke attack on this host.

Correlation unifier: denoted Ξ_{ab}, is the set of all possible unifiers[4] to correlate $post_a$ and pre_b.

With the same approach, it is possible to define correlation between an attack and an intrusion objective. In this case, we have simply to replace *pre-condition* by *state* in the previous definition.

4.2 How to use correlation

Once attacks and intrusion objectives are specified in LAMBDA, we can generate all correlation unifiers between each pair of attacks (respectively between an attack and an intrusion objective). When two attack occurrences are detected, if some unifier in the unifier set is identified, we can then say that these attack occurrences are correlated in the same intrusion scenario.

Using this approach, it is possible to build a correlation graph. Figure 5 presents such a correlation graph where nodes are LAMBDA descriptions and edges are correlation unifiers.

When first steps of a given intrusion scenario are identified, we can, with the same mechanisms, predict possible continuations of this scenario. We can generate hypothesis about future attacks and the

[2] as used in PROLOG
[3] *post-condition* is represented in its conjunctive form
[4] Unifiers of direct or knowledge gathering correlation

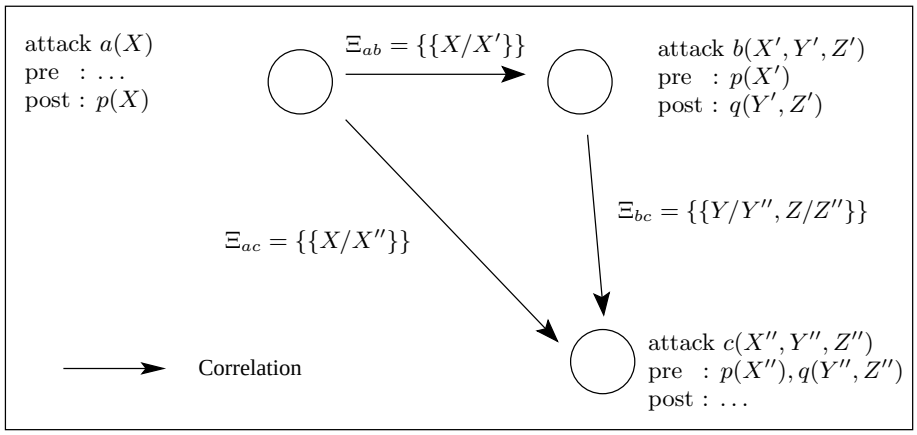


Figure 5: Correlation graph example

intrusion objectives the intruder attempts to achieve. We shall call *virtual attack* an attack predicted by this process of intention recognition. A virtual attack becomes effective once its occurrence is detected.

Thus, it is sometimes possible to anticipate on the actions performed by the intruder and develop a specific counter-measure in response. This means that our approach may be used to launch a counter-measure not only after a given intrusion objective is achieved by the intruder but also when the beginning of a given scenario is detected. In this latter case, the counter-measure will be used to prevent continuations of this starting scenario.

We shall now see how to define and use the anti-correlation principle to elaborate the counter-measure.

4.3 Anti-correlation

Let a and b be respectively LAMBDA descriptions of a counter-measure and an attack. $post_a$ is the set of literals of *post-condition* of a and pre_b is the set of literals of *pre-condition* of b.

Anti-correlation: a and b are anti-correlated if the following condition is satisfied:

$\exists\ E_a$ and E_b such that
$(E_a \in post_a \wedge not(E_b) \in post_b)$ or $(not(E_a) \in post_a \wedge E_b \in pre_b)$
and E_a and E_b are unifiable through a most global unifier θ.

Anti-correlation unifier: denoted Ψ_{ab}, is the set of all unifiers θ possible to anti-correlate $post_a$ and pre_b.

Using the same approach, it is possible to define anti-correlation between a counter-measure and an intrusion objective. We have simply to replace *pre-condition* by *state* in the previous definition.

In the following section 5, we show how to use the anti-correlation notion to design a response mechanism to an intrusion scenario. In particular, figures 6 and 7 provide examples of anti-correlation and how to use it in a response mechanism.

5 Using anti-correlation for response

When a scenario is identified, the correlation process provides a graph of attack occurrences, virtual attacks and intrusion objective. A counter-measure will apply to invalidate future attacks or invalidate intrusion objective. Thus, we have two response mechanisms, one that applies against virtual attacks and the other on an intrusion objective.

5.1 Response to an intrusion objective

In this case, response aims at updating the system state to invalidate the intrusion objective in an intrusion scenario.

Let o be an intrusion objective. To invalidate this intrusion objective, we must find a LAMBDA definition r of a counter-measure such that $\Psi_{ro} \neq \emptyset$. Then, it is possible to parameterize this counter-measure candidate with the unifier of correlation Ψ_{ro}.

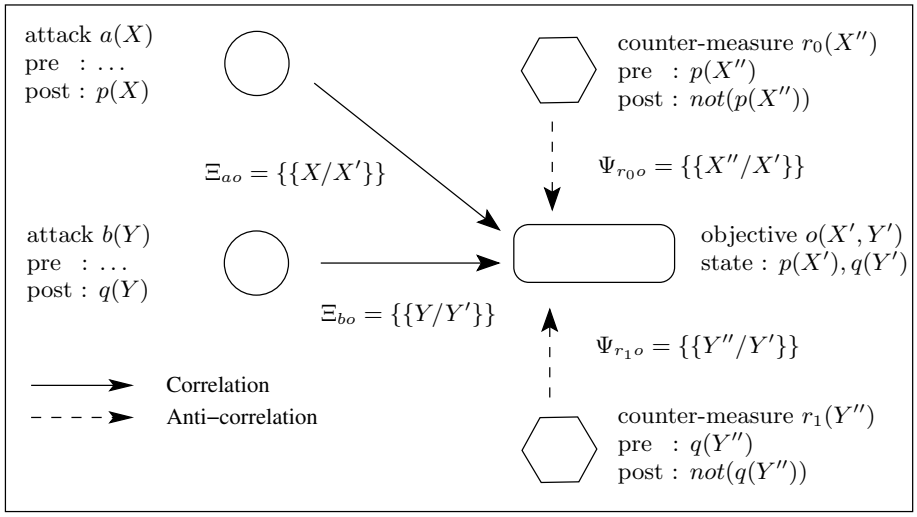


Figure 6: correlation graph with direct response on objective

In figure 6, we assume that two occurrences of attack are detected: an occurrence of a with argument $X = x$ and an occurrence of b with argument $Y = y$. The correlation process diagnoses that these two attacks are correlated with a given intrusion objective o and this objective is achieved. The response process finds two counter-measure candidates: (1) counter-measure r_0 with parameter $X'' = X'$ to invalidates condition p (and thus objective o) and (2) counter-measure r_1 with parameter $Y'' = Y'$ that invalidates condition q (and thus also objective o). In our approach, these two counter-measures are suggested to the administrator who can select one of them (or both). The *verification* field of the selected counter-measure is then evaluated to check if the counter-measure was executed successfully. If this is the case, we can reevaluate the state condition of the intrusion objective to false.

5.2 Response to an ongoing scenario

It is possible that a counter-measure may not apply directly to an intrusion objective if one of these conditions holds:

- There is not any counter-measure in the response library which may apply to invalidate the intrusion objective.
- The counter-measure does not apply to the system state because the pre-condition of this counter-measure is evaluated to false.
- All counter-measure candidates were launched without success.

In this case, a possible solution is to modify the system state to invalidate one attack in a sequence of virtual attacks.

Let $a_1 ... a_n$ be a sequence of virtual attacks and o an intrusion objective such that for every $i \in [1, n-1]$, a_i is correlated with a_{i+1} and a_n is correlated with o.

To block this sequence of attacks, we must find a valid LAMBDA counter-measure r such that r is anti-correlated with one of the attacks a_k ($k \in [1, n]$).

For instance, let us assume, in figure 7, that we detect an occurrence of a. The recognizing intention process identifies that the intruder may perform b after a to achieve the objective o. In this case, the response process can find a counter-measure r to invalidate the *pre-condition* of b. This will prevent performance of attack b and invalidate this scenario.

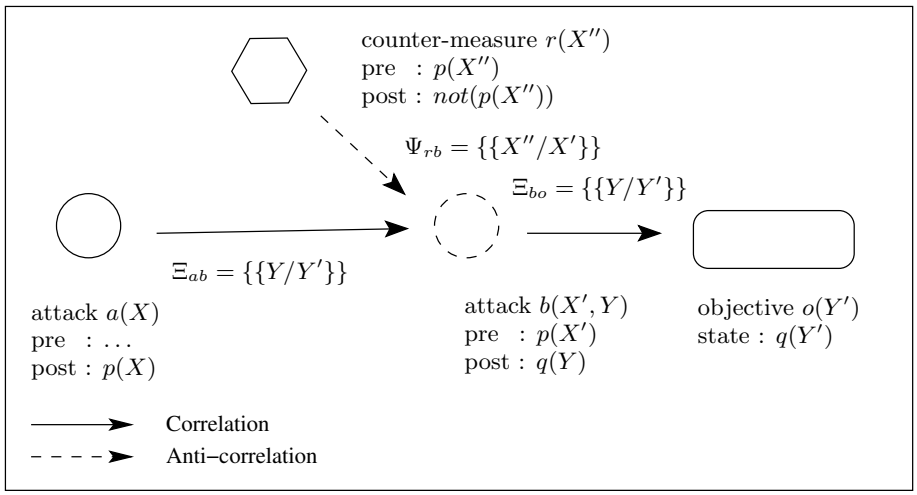


Figure 7: Correlation graph with response on a sequence of virtual attacks

6 Example and experimentation

This section presents DIAMS, a platform of intrusion detection we have developed and implemented in Java. In particular, this platform implements the response mechanism suggested in this paper.

The global architecture of DIAMS is presented in figure 8. DIAMS first collects the syslog alerts raised by different IDS and translates them in the IDMEF (Intrusion Detection Message Exchange Format). Then, these alerts are sent by the router to an alert database (managed by PostgresSQL) and to CRIM. CRIM [5] analyzes these alerts to correlate them using the approach presented in section 4.1 and generates a diagnosis of the detected intrusions. This diagnosis is transmitted to the response mechanism that provides the administrator with a set of counter-measure candidates using the approach based on anti-correlation defined in section 5. The administrator can then select one or several counter-measures. Finally, a module called DIAMS-Action automatically executes the script corresponding to these selected counter-measures.

The CRIM and response modules are currently implemented in Prolog. To illustrate this implementation, let us consider the attack scenario that corresponds to the correlation graph presented in figure 9. This intrusion exploits a system miss-configuration of UNIX export partitions. Through a user partition, the intruder can increase his privilege to get a remote user access on the target host.

With the $rpcinfo$ attack, A (the intruder) collects information about RPC service (Remote Procedure Call) on H (Host). The attack pre-condition specifies that the intruder must have a remote access on host H and this host must have RPC service running. The post-condition specifies that the intruder knows that RPC is running on H.

With the *showmount* attack, an intruder A collects information about export partitions on host H. The pre-condition specifies that H has an export partition P. Then, post-condition specifies that the intruder knows it.

With the *mount* attack, A mounts the partition P from H to his/her local host. The post-condition specifies that the partition is mounted by the intruder in his or her local host.

With the $rhost\text{-}modification$ attack, A modifies the .rhost file whose owner is U in the partition P. If a user account U is hosted by the partition, then the user can get a user access on this account.

With the *login* attack, A is connected to H through the user account U.

The intrusion objective *illegal-user-access* specifies that the intruder bypasses the password verification and obtains a illegal user access.

When alerts corresponding to different steps of this scenario are raised, CRIM applies the correlation mechanism recalled in this paper to recognize the global scenario.

Once this diagnosis is obtained, the response module is used to select a counter-measure in the counter-measure library. Faced to this intrusion scenario, the response module actually selects two possible counter-measures. On one hand, a counter-measure can apply directly on the intrusion objective. This counter-measure, called *kill-login*, kills the login process. On the other hand, we can react before

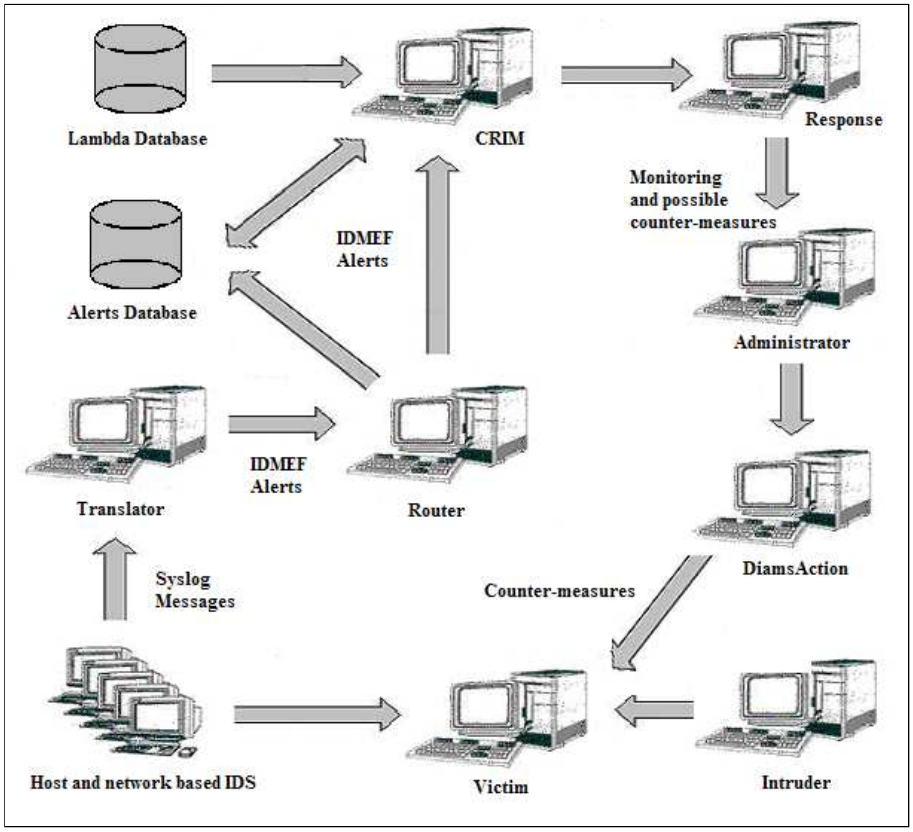


Figure 8: Diams framework

the intrusion objective is achieved. The counter-measure, called *close-connection*, closes the connection[5] between the host and the intruder.

These two counter-measures are presented to the administrator who can select one of them. After a counter-measure is selected, the corresponding script is executed by DIAMS-Action. DIAMS will then send an alert to CRIM to tell that a given counter-measure was launched. Finally, CRIM can apply the *verification* field of the counter-measure to check that that the counter-measure is effective.

7 Conclusion

In this paper, we have presented a global approach to select and apply response mechanisms when an intrusion occurs. This approach is based on a logical representation in LAMBDA of both intrusions and counter-measures. This is used to build libraries of intrusions and counter-measures. The library of counter-measure is organized into a taxonomy that takes its inspiration from [7]. The notion of anti-correlation is then used to select relevant responses to a given intrusion in order to help the administrator to decide which appropriate counter-measures may be launched. This mechanism is integrated in DIAMS, a platform of intrusion detection that collects and analyzes alerts generated by various intrusion detection systems. DIAMS is a research prototype implemented in Java whereas the response module is implemented in Prolog. These two modules can interact via JPL (Java Prolog Interface).

Up to now, we only use this approach to provide a support to the administrator who takes the final decision to choose and launch a given response. This is a prudent strategy but it introduces an overhead that is sometimes incompatible with real time response. This is why we are currently analyzing situations where it would be possible to *automatically* decide to launch the response.

Notice that a possible response consists in reconfiguring the security policy to prevent a new occurrence of a given intrusion. However, as suggested in [11], dynamic changes of the security policy may cause failure of some software components. This is why [11] suggests the notion of security agility, a

[5] by sending a TCP-Reset

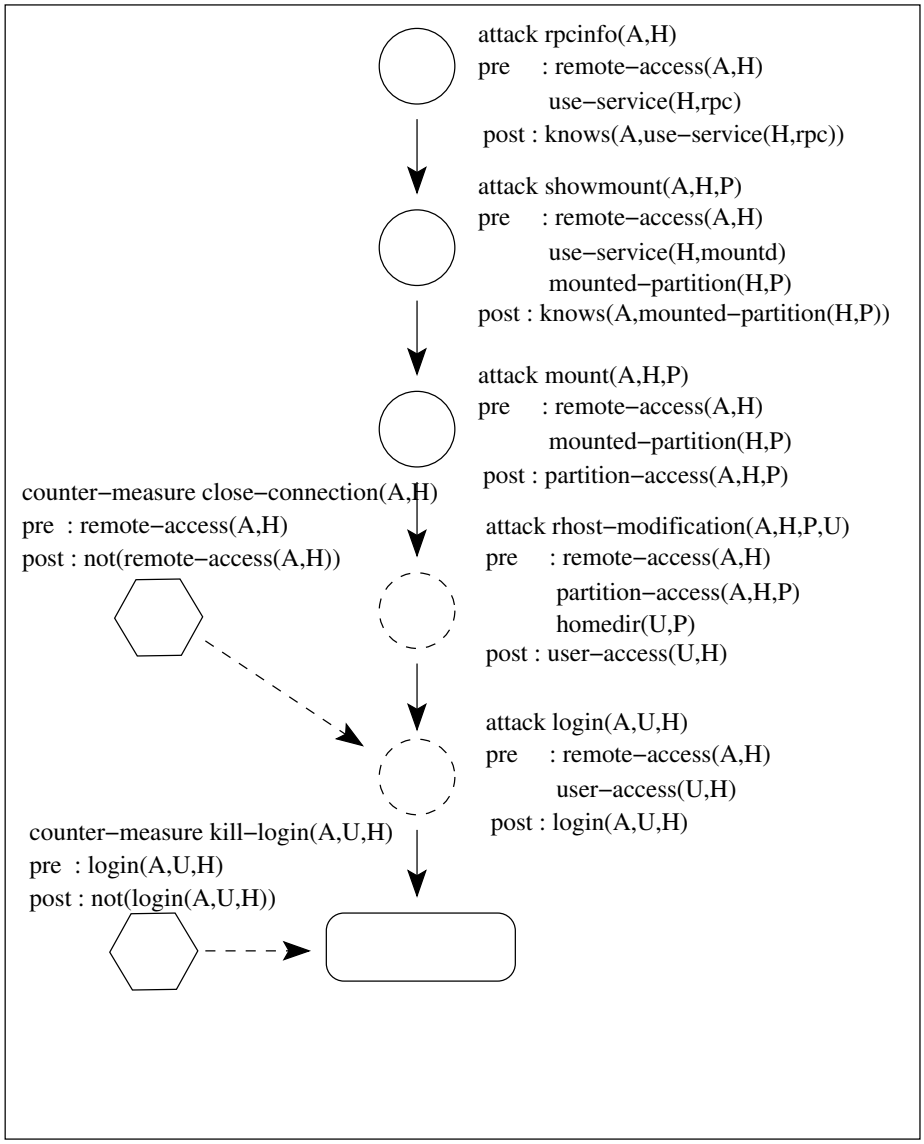


Figure 9: Illegal remote user access

strategy to provide software components with adaptability to security policy changes. Security agility might be nicely included into the intrusion detection and response framework suggested in this paper. This represents a possible extension of our work.

When using anti-correlation, several responses may be selected. In this case, it would be interesting to rank these different responses and a possible ranking criteria would be to evaluate the effectiveness of the responses to stop the attack. For this purpose, we plan to extend the response formalism with temporal logic to include the fact that a given response will stop an intrusion *until* another additional event occurs. More difficult is performing an action that will cause this additional event, more effective is the response. This also represents further work that remains to be done.

References

[1] R. Bace. *Intrusion Detection.* McMillan, 2000.

[2] S. Benferhat, F. Autrel, and F. Cuppens. Enhanced correlation in a intrusion detection process. In *Mathematical Methods, Models and Architecture for Computer Network Security (MMM-ACNS 2003)*, St Petersburg, Russia, September 2003.

[3] F. Cuppens, F. Autrel, A. Miège, and S. Benferhat. Recognizing malicious intention in an intrusion detection process. In *Second International Conference on Hybrid Intelligent Systems*, Santiago, Chili, December 2002. Special session: "Hybrid Intelligent Systems for Intrusion Detection".

[4] Frédéric Cuppens. Managing alert in a cooperative intrusion detection environnement. In *17th Annual Computer Security Applications Conference (ACSAC)*, New-Orleans, LA, December 2001.

[5] Frédéric Cuppens and Alexandre Miège. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 2002.

[6] Frédéric Cuppens and Rodolphe Ortalo. Lambda: A language to model a database for detection of attacks. In H. Debar, L. Mé, and S. F. Wu, editors, *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, number 1907 in LNCS, pages 197–216, October 2000.

[7] Sylvain Gombault and Mamadou Diop. Response function. In *1st Symposium on Real Time Intrusion Detection (NATO)*, Lisbon, Portugal, May 2002.

[8] Klaus Julisch. Mining alarm clusters to improve alarm handling efficiency. In *17th Annual Computer Security Applications Conference (ACSAC)*, New-Orleans, LA, December 2001.

[9] Peng Ning and Dingbang Xu. Learning attack strategies from intrusion alerts. In *10th ACM Conference on Computer and Communication Security*, Washington, DC, 2003.

[10] S. Northcutt and J. Novak. *Network Intrusion Detection*. Paperback, 1999.

[11] Mike Petkac and Lee Badger. Security agility in response to intrusion detection. In *16th Annual Computer Security Applications Conference (ACSAC)*, New-Orleans, LA, 2000.

[12] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of LISA'99: 13th Systems Administration Conference*, Seattle, Washington, USA, November 7-12 1999.

[13] H.C.M De Swart. *Mathematics, Language, Computer Science and Philosophy*, volume 2. Peter Lang, 1994.

IRA
Intrusion - Réaction - Appâts

Louis Derathe
THALES Communications
Strategy & Advanced Systems
System & Architecture
160, Blvd de Valmy, BP 82
92704 Colombes

Louis.derathe@fr.thalesgroup.com

RÉSUMÉ

La problématique classique de la protection en bordure des systèmes d'information interconnectés évolue aujourd'hui vers la mise en œuvre du principe de « défense en profondeur » ; la protection des systèmes d'information repose alors sur une succession de couches de protection, renforcées par des outils de veille, d'alarme et de réaction.

Dans une vision intégrée d'un système d'information, la protection de bordure doit ainsi être soutenue par des dispositifs intervenant aussi en interne au SI : systèmes de contrôle comme les IDS[1], système d'alarme et de réaction automatique, systèmes de dérivation des attaques.

De nombreux outils sont disponibles sur le marché ou le Web, mais le problème essentiel reste la mise en cohérence de ces dispositifs.

[1] Intrusion Detection system

Communication présentée lors du symposium RTO IST sur "La défense adaptative pour les réseaux non classifiés", organisé à Toulouse, France, du 19 au 20 avril 2004, et publié dansRTO-MP-IST-041.

Plan

Liste des Figures

Glossaire

Alerte	Suite à une alarme, avertissement pouvant être de diverses formes : e-mail, trappe SNMP, bip sonore…
Analyse Forensic ou analyse post-mortem	Branche complète de l'informatique, chargée de la récupération des données sur des supports corrompus ou effacé.
Attaque par buffer overflow (ou Débordement de Tampon)	Des vulnérabilités par débordement de tampon apparaissent quand les développeurs utilisent de mauvaises méthodes de codage pour effectuer une fonction dans un programme. Certaines fonctions, notamment en C, ne contrôlent pas la taille de leurs arguments. Ceci crée une faille que certains attaquants sont capables d'exploiter afin d'obtenir un accès sur le système.
Attaque par Déni de Service (DoS pour Denial of Service)	Attaque visant la disponibilité d'un système (réseau, services…).
Cheval de Troie	Programme qui se cache dans d'autres logiciels et qui permet de pénétrer les systèmes informatiques.
Classification	Système de codification indiquant le degré de confidentialité.
Cracker	Personne qui casse les codes d'accès ou les clefs de sécurité des logiciels.
Cryptogramme	Texte chiffré.
Cryptographie	Relatif aux moyens de chiffrement qui permettent de rendre illisible une information à toute personne ne partageant pas le secret pour la déchiffrer.
Cryptographie Asymétrique	Chaque entité possède deux clés : une clé publique de chiffrement qui permet de chiffrer les messages qui lui sont destinés, et une clé privée qui lui permet, à elle seule, de lire les message qui ont été chiffrés avec sa clé publique.
Cryptographie Symétrique	Chaque entité possède la même clé qui permet de chiffrer et de dé chiffrer une information.
Cypherpunk	Hacker qui s'intéresse au cryptage des données informatiques (fichiers, mots de passe, etc).
Evènement	Action sur un système d'information pouvant faire l'objet d'un enregistrement.
Faille	Brèche dans la sécurité d'un système informatique laissée par les concepteurs et la maintenance technique.
Faux-négatif (false negative)	Action malicieuse non détectée par le système de détection d'intrusion.
Faux-positif (false positive)	Action non malicieuse détectée comme étant dangereuse par le système de détection d'intrusion.
FIC, *File integrity checker*, vérificateur d'intégrité de fichier	Outil permettant la surveillance d'un système vis-à-vis de toute modification de ses fichiers. Il surveille l'état de ceux-ci et peut parfois restaurer l'état initial par récupération sur une copie protégée.
Firewall (pare-feu)	« Dispositif informatique qui filtre les flux d'information entre un réseau interne à un organisme et un réseau externe en vue de neutraliser les tentatives de pénétration en provenance de l'extérieur et de maîtriser les accès vers l'intérieur » (*Journal Officiel de la République française*, 16 mars 1999, « Vocabulaire de l'informatique et de l'internet »).Voir garde barrière.
Garde Barrière	Dispositif logiciel ou matériel effectuant un filtrage statique ou dynamique sur les données transitant entre deux réseaux ou plus.
Habilitation de personnel	Reconnaissance de la capacité d'une personne à accomplir en sécurité les tâches fixées
Hacker	Voir pirate informatique.
Homologation	Autorisation d'utiliser, dans un but précisé ou dans des conditions prévues, un produit, un processus ou un service.
HoneyNet	Ensemble d'éléments réels, à but non productif, hautement surveillé et mis en place pour leurrer et observer les agissements d'attaquants.

HoneyPot de production	Honeypot dont le but est d'abaisser les risques pour une société en étant adjoint au système de production.
Honeypot de recherche	Honeypot élaboré dans le but d'acquérir de l'information sur la communauté des attaquants.
HoneyPot ou « pot de miel »	Sens large : ressource dont le but est d'être testée, attaquée ou compromise ; Sens restreint : système connecté à un réseau de production dont le but est de leurrer un attaquant ; Dans le cadre d'un honeynet : machine réelle au sein d'un honeynet
IDS, *Intrusion Detection System*, système de détection d'intrusion	Système qui a pour objectif de déceler une tentative d'intrusion sur l'équipement ou le système à protéger.
Intrusion	Toute utilisation d'un système informatique à des fins autres que celles prévues, généralement dues à l'obtention de privilèges de façon illégitime.
Leurre Informatique	Technique utilisée par les experts pour confondre un pirate. Cela consiste à placer sur un système des fichiers comportant de fausses données qui captiveront l'attention du hacker et qui le maintiendront plus longtemps connecté jusqu'à ce qu'il soit repéré
Backdoor	Voir faille.
Pirate Informatique	Expert en informatique et programmation dont le passe temps est d'explorer les limites des systèmes et des réseaux
Signature d'attaque	Motif d'une attaque (utilisées par les N-IDS).
Sniffing (Ecoute)	Récupération illicite d'informations circulant sur un réseau.
Spoofing	Voir mascarade.
WAREZ	Site qui collecte de nombreuses informations destinées aux hackers.

Chapitre 1 - Introduction

THALES Communication réalise de nombreux systèmes et équipements dans le domaine de la Défense et dispose d'un réel savoir-faire dans les domaines SSI et VAR[2], aussi bien sur les aspects organisationnels que techniques.

THALES Communication est, pour cette raison, à même de proposer dans le cadre de ce symposium une participation sur les sujets technologiques suivants :

- La détection d'intrus, le contrôle et les technologies de réaction
- Les technologies «d'appât» (ressources informatiques conçues pour attirer les pirates et servir d'alarme)

Dans le cadre de la détection d'intrus, THALES Communication présente au chapitre « Tests d'Intrusions dans les Réseaux Informatiques » et au chapitre « Intrusion et Génération de rapports d'Audits » les aspects suivants :

- Inventaire et descriptif des techniques les plus courantes des pirates informatiques dans les domaines des intrusions (interception, balayage, écoute, piégeage, ingénierie sociale, fouille, utilisation et mise en place de canaux cachés, déguisement, mystification, rejeu, déni de service, virus et chevaux de Troie, etc.)
- Inventaire des différents outils (network based IDS, host based IDS, file integrity checkers, analyseurs de logs) et classement selon principes de détection, comportement après détection, sources et fonctionnement (temps réel ou analyse périodique).

Dans le cadre des technologies « d'appât », THALES Communication présente au chapitre « Appâts (honeyPot & Honeynet) » les aspects suivants :

- Définition du concept général de Honey-Pot / Honey-Net, apports de ces technologies et objectifs (protection, étude des profils d'attaques, désinformations)
- Description des aspects d'intégration de ces technologies dans le cadre d'un système d'information (Organisation et personnel, aspects juridiques (notion de piège et responsabilités), problèmes liés aux rebonds et à la remontée d'anonymat, détermination du contenu de ces « appâts », aspects SSI de ces environnements, liens avec systèmes VAR)
- Description des relations entre les technologies IDS et « appât »
- Description des différentes mises en œuvre (Émulation par scripts, Émulation de services et d'environnements, Création d'environnements)

Dans le cadre des protections en bordure, THALES Communication présente au chapitre « passerelle inter-réseaux de sensibilités différentes » et au chapitre « Dissimulation et fuite d'information (Canaux Cachés) » les aspects suivants :

- Concepts techniques et organisationnels des passerelles inter-niveaux
- Architecture de référence d'une passerelle
- Principales techniques de dissimulation d'information

[2] Veille Alerte Réponse

Nota : Les quelques éléments présentés dans ce document ne couvrent pas de façon exhaustive tous les aspects étudiés ou pris en compte, mais devraient permettre aux spécialistes d'appréhender le périmètre de l'étude.

Chapitre 2 - Tests d'Intrusions dans les Réseaux Informatiques

2.1 Généralités

Le projet « Tests d'Intrusions dans les Réseaux Informatiques (TIRI) [1] », réalisé durant le premier semestre 2000, avait pour buts, d'une part de réaliser un état de l'art sur les différentes techniques d'attaques des systèmes d'information, et d'autre part de définir et de mettre en œuvre une plate-forme de tests d'intrusions et de sensibilisation.

Les techniques présentées ont été regroupées en deux catégories : les attaques physiques et les attaques logiques. On parle d'attaque physique dès lors que l'attaque concerne le médium de communication du système d'information. Il peut par exemple s'agir d'interceptions, de brouillages, de balayages ou d'écoutes. Les attaques logiques consistent le plus souvent à exploiter des défauts de configuration ou de conception dans les réseaux ou dans les logiciels. Les types d'attaques logiques les plus connus sont les fouilles de données, les canaux cachés, les usurpations d'identités, les rejeux de séquences, les chevaux de Troie, les dénis de service, les fuites d'informations, les virus, les vers et la cryptanalyse.

Le projet s'est particulièrement attaché à la méthodologie utilisée par les attaquants pour s'introduire frauduleusement dans les systèmes d'information. Dans la grande majorité des cas, il est possible de distinguer quatre phases distinctes dans une intrusion. La première est une phase de recherche d'anonymat qui consiste pour l'attaquant à s'assurer que l'attaque qu'il compte mener lui procurera un très haut degré d'anonymat. La deuxième phase est une phase de recherche d'informations qui consiste le plus souvent à réaliser une cartographie complète des services du réseau ou de la machine cible. Cette étape permet à l'attaquant de sélectionner l'attaque la plus adaptée qu'il pourra alors lancer lors de la phase d'infiltration. Enfin une attaque se termine souvent par une phase d'effacement des traces qu'elle a laissées.

L'intérêt d'étudier les méthodologies d'intrusions est de définir les moyens de protection conséquents. Les techniques de protection sont aussi variées que les techniques d'attaques. Tout comme il existe des grands principes pour les attaques, il existe également des grands principes pour les protections : principe du moindre privilège, principe de la défense en profondeur, principe du maillon faible… De la même façon, tout comme il existe des outils pour automatiser les attaques, il existe des outils pour automatiser les protections : pare-feu, outils de chiffrement, scanners… Les procédures et les outils à mettre en oeuvre pour sécuriser un site doivent être clairement définies dans un document appelé « politique de sécurité ».

2.2 Quelques Eléments constitutifs de l'étude

Les pirates informatiques utilisent différentes techniques pour obtenir la cartographie d'un réseau. Selon les outils utilisés, les traces seront plus ou moins visibles dans les fichiers de journalisation ; aussi, l'attaquant va-t-il essayer le plus souvent de passer tout simplement par l'entrée principale et si possible de façon normale (il essaiera d'obtenir un mot de passe), sinon, il utilisera les failles ou trappes de certains logiciels.

2.2.1 Quelques Exemples d'Attaques Physiques

Interception L'attaquant cherche à intercepter un signal électromagnétique et à l'interpréter. L'interception peut porter sur des signaux hyperfréquences, ou hertziens, émis, rayonnés, ou conduits. L'agresseur se mettra ainsi à la recherche des émissions satellites, et radio, mais aussi des signaux parasites émis par les SI, principalement par les terminaux, les câbles et les éléments conducteurs entourant les SI.

Brouillage Utilisée en télécommunication, cette technique rend le SI inopérant. C'est une attaque de haut niveau, car elle nécessite des moyens importants, qui se détectent facilement.

Balayage « scanning »Le balayage consiste à envoyer au SI un ensemble d'informations de natures diverses afin de déterminer celles qui suscitent une réponse positive. L'attaquant pourra aisément automatiser cette tâche, et déduire, par exemple, les numéros téléphoniques qui permettent d'accéder à un système, le type du système, et pourquoi pas, le nom de certains utilisateurs ainsi que leur mot de passe.

Écoute « sniffing » L'écoute consiste à se placer sur un réseau informatique ou de télécommunication et à analyser et à sauvegarder les informations qui transitent. De nombreux appareils du commerce, généralement conçus pour l'administration réseau, facilitent les analyses et permettent notamment d'interpréter en temps réel les trames qui circulent sur un réseau informatique.

Piégeage L'agresseur tentera d'introduire des fonctions cachées dans le SI, en principe en phase de conception, de fabrication, de transport ou de maintenance.

2.2.2 Quelques Exemples d'Attaques Logiques

Ingénierie sociale L'ingénierie sociale consiste à obtenir des informations sur un système d'information en exploitant les faiblesses des utilisateurs. On peut, par exemple, envoyer un mél dont l'expéditeur est supposé être l'administrateur et demander au destinataire de changer son mot de passe, voire de lui en communiquer un nouveau pour la maintenance du réseau.

Fouille La fouille informatique, par analogie avec la fouille physique, consiste à étudier méthodiquement l'ensemble des fichiers et des variables d'un SI pour en retirer des données de valeur. Cette recherche systématique d'informations est en général grandement facilitée par la mauvaise gestion des protections classiques qu'il est possible d'attribuer à un fichier.

Craquage de mots de passe Une fois le mot de passe crypté récupéré, il est nécessaire de le retrouver sous sa forme originale. Deux approches d'attaques sont envisageables. La première consiste à s'appuyer d'un dictionnaire, à crypter systématiquement ses éléments et à vérifier si le résultat coïncide avec le mot de passe capturé. La seconde méthode, appelée méthode par force brute, essaye toutes les combinaisons possibles de caractères jusqu'à obtenir le bon résultat.

Déguisement Il s'agit d'une attaque informatique qui consiste à se faire passer pour quelqu'un d'autre, et obtenir les privilèges de celui ou celle dont on usurpe l'identité.

Mystification L'attaquant va simuler le comportement d'une machine pour tromper un utilisateur légitime et s'emparer de son identifiant et de son mot de passe.

Rejeu Le rejeu est une variante du déguisement qui permet à un attaquant de pénétrer dans un SI en envoyant une séquence de connexion effectuée par un utilisateur légitime et préalablement enregistrée à son insu.

Substitution Ce type d'attaque est réalisable sur un réseau ou sur un système d'information comportant des terminaux distants. L'agresseur écoute une ligne et intercepte la demande de déconnexion d'un utilisateur travaillant sur une machine distante. Il peut alors se substituer à ce dernier et continuer une session normale sans que le système note un changement d'utilisateur.

Saturation : déni de service (Dos ou Ddos) Cette attaque contre la disponibilité consiste à remplir une zone de stockage ou un canal de communication jusqu'à ce qu'on ne puisse plus l'utiliser.

Cheval de Troie Un cheval de Troie est un programme qui se cache dans un autre programme, apparemment inoffensif, qui, dès que ce dernier est lancé, s'exécute de façon clandestine.

Trappe & faille Une trappe est un point d'entrée dans un système informatique auquel les mesures de sécurité normales ne sont pas appliquées. Une trappe peut être par exemple, délibérément mis en place par les programmeurs ou les personnes chargées de la maintenance, pour des raisons de simplicité et de rapidité d'action (réalisation de tests par exemple), et qu'ils ont oublié de supprimer.

Bombe Une bombe est un programme destructif en attente d'un événement spécifique déterminé par le programmeur pour se déclencher.

Virus Un virus est un programme parasitant les machines et capable de se reproduire. Il est en mesure d'infecter d'autres programmes, qui à leur tour se conduiront comme le virus père, et peuvent endommager les systèmes.

Ver Un ver est un programme capable de se propager de réseau en réseau et de les rendre indisponibles.

Asynchronisme Ce type d'attaque évoluée exploite le fonctionnement asynchrone de certaines parties ou commandes du système d'exploitation. Les requêtes concernant de nombreux périphériques sont mises en file d'attente dans l'ordre des priorités puis traitées séquentiellement. Des tâches sont ainsi endormies puis réveillées lorsque les précédentes requêtes sont satisfaites. A chaque fois qu'une tâche ou qu'un programme est ainsi endormi, son contexte d'exécution est sauvegardé pour être restitué en l'état lors du réveil. Ces sauvegardes de contexte contiennent donc des informations propres à l'état du système et un attaquant averti peut les modifier afin de contourner les mesures de sécurité.

Cryptanalyse La cryptanalyse est une discipline de la cryptographie dont le but est de retrouver le message clair à partir du message chiffré sans détenir tous les éléments (clés, algorithme).

2.3 Environnement Technique étudié

Selon des scénarios mettant en œuvre le déroulement classique d'une attaque (Anonymat / Recherche d'informations / Infiltration / Effacement des traces), la plate-forme de démonstration a permis de montrer les mode d'actions d'outils comme ceux présentés ci-dessous.

Bombes emails

- ***Kaboom*** programme pour bombarder une boîte aux lettres
- ***Anonymail*** permet de créer des méls sous une fausse identité
- ***Homicide*** utilitaire qui permet de bombarder une personne en dissimulant l'adresse source des messages
- ***Avalanche*** utilitaire qui permet de bombarder un compte de messagerie, et permet également d'inscrire le compte cible à différentes listes de diffusion

Craquage de mots de passe

- ***NTUCrack*** utilitaire qui permet de craquer les mots de passe Unix
- ***John The Ripper*** un des programmes les plus célèbres pour craquer les mots de passe de type Unix.
- ***L0phtCrack*** l'utilitaire le plus célèbre pour craquer, entre autres, les mots de passe de Windows NT

Déni de service

- ***Winnuke*** provoque l'arrêt de Windows 95 et de Windows NT 3.51 et Windows NT4 (si le service Pack est inférieur à 4 car Microsoft a depuis corrigé la faille exploitée)
- ***Killwin*** pour Linux paralyse en quelques secondes une machine NT 4.0 pack 6 en envoyant des messages de dépassement de bande (MSG_OOB)
- ***Bitchslap*** utilitaire réalisé par des hackers des groupes SIN et technophoria permet d'envoyer un message de dépassement de bande (MSG_OOB) vers le port 139 d'un poste Windows
- ***Portfuck***

Sniffeurs

- ***Ethereal*** fonctionne aussi bien sous Windows que sous Linux
- ***Analyser***

Scanners

- ***Ultrascan*** logiciel de balayage de ports. Il peut scanner une plage d'adresses IP et une plage de ports
- ***Nmap*** logiciel de balayage est assez sophistiqué car il peut, par exemple, effectuer des balayages par demi-connexions

Chevaux de Troie

- Netbus

Chapitre 3 - Intrusion et Génération de Rapports d'Audit

3.1 Généralités

Le projet « Tests d'Intrusion et Génération de Rapports d'Audit (TIGRA) [4] » réalisé durant le premier semestre 2002 s'est intéressé aux systèmes de détection d'intrusions communément dénommés « IDS ». La première phase du projet a consisté à réaliser un état de l'art sur les IDS. L'objet de la deuxième phase fut de mettre en œuvre un IDS au sein d'une plate-forme de tests afin d'en présenter les différents aspects.

Les trois fonctionnalités principales des IDS sont la détection des attaques, la réaction aux attaques et l'analyse des attaques. Certains IDS sont capables de réagir aux attaques en les bloquant (d'un point de vue fonctionnel, ces IDS qui fonctionnent en mode coupure, se rapprochent des pare-feux) mais dans la majorité des cas, un IDS est capable de journaliser les tentatives d'attaques, d'évaluer leur niveau de gravité et de déclencher des alertes appropriées.

La détection des attaques consiste le plus souvent à détecter les signatures des attaques mais des méthodes d'analyse comportementale commencent à apparaître (C-IDS). La réaction aux attaques peut se faire soit de manière active (blocage de l'attaque ou dérivation) soit de manière passive (journalisation et génération d'alertes uniquement). Les sources de données des IDS sont variées : il peut s'agir d'audits systèmes (systèmes de fichiers), d'audits applicatifs (contrôle des débordements de mémoire) ou encore des flux de données transitant sur le réseau.

Les deux types d'IDS les plus couramment rencontrés sont les « Network Based IDS » (N-IDS) et les « Host Based IDS » (H-IDS). Les premiers sont installés au niveau des goulets d'étranglement (bordures) des réseaux pour analyser les flux entrant et sortant, tandis que les seconds sont positionnés sur les hôtes, et sont chargés d'analyser des sources de données plus variées (systèmes de fichiers, mémoire vive, processus …).

Les IDS présentent quelques limites. Lorsqu'ils sont mal paramétrés, ils peuvent générer des quantités importantes de « faux-négatif » et de « faux-positifs ». Enfin, d'un point de vue pratique, les IDS peinent à analyser des débits importants, et peuvent occasionner des pertes de qualité de service. De plus, il est possible pour un attaquant de générer un grand nombre d'attaques dont il sait qu'elles sont très coûteuses en ressources pour l'IDS, et ainsi provoquer une rupture de service de l'IDS.

3.2 Quelques Eléments constitutifs de l'étude

3.2.1 Fonctions utiles des IDS en matière de SSI

1 / Détection des attaques :

- ***détection*** précoce de l'attaque afin de limiter sa propagation à un nombre restreint d'hôtes
- ***journalisation*** des actions entreprises par l'attaquant
  - afin d'analyser les moyens mis en oeuvre et comprendre la méthode employée pour adapter la protection actuelle
  - avec gradation des enregistrements selon le niveau de gravité

2 / Réaction aux attaques :

- ***clôture de la connexion*** (Session Snipping)
- ***reconfiguration dynamique des règles du pare-feu***

- ***ou autre selon*** des ***scénarios d'attaques***

3 / Avertissement local et / ou distant vers le Responsable SSI

4 / Analyse

3.2.2 Modes d'action des IDS

Trois modes d'actions sont caractéristiques des outils de détection d'intrusion :

- ***mode furtif*** correspond à un mode où l'IDS n'offre aucune interaction directe avec le système (ainsi un paquet qui le traverse conserve son adresse MAC)
- ***mode coupure*** : toutes les trames sont contrôlées avant de pouvoir passer (comme dans le cas d'un pare-feu)
- ***mode écoute***, sur un brin du réseau où il est possible d'enregistrer tout ou partie du trafic, il collecte toute l'information transitant sur le brin (il n'agit pas en mode coupure : fonction d'alerte en cas d'attaque, mais pas de rejet de la trame « suspecte »).

3.2.3 Classement des IDS

Il s'effectue selon :

- ***le principe de détection*** comportementale (C-IDS) et par scénarios (essentiellement basé sur signature)
- ***le comportement après détection*** suivant s'il est actif (réaction de riposte à l'attaque) ou passif (journalisation uniquement)
- ***les sources de données*** pouvant provenir soit d'un audit système (exemple : modification du système de fichiers…), soit applicatif (contrôle des débordement de mémoire) ou encore de paquets transitant sur le réseau (N-IDS)
- ***la fréquence d'utilisation*** selon si elle s'effectue en temps réel ou par analyse périodique

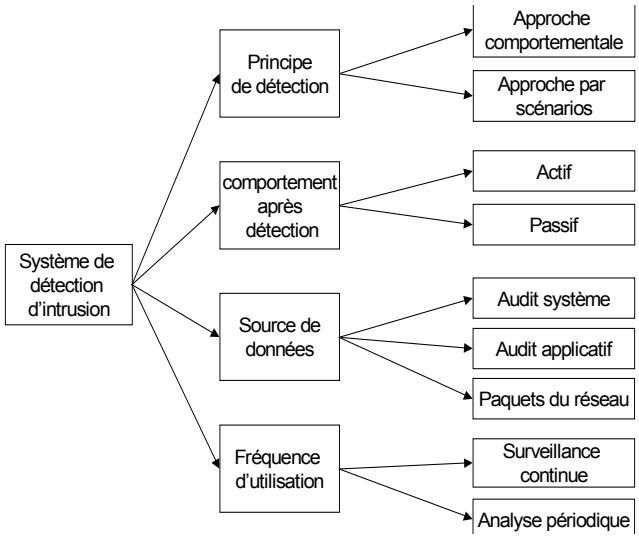


Figure 1 : principes de fonctionnement des systèmes de détection d'intrusion

3.2.4 Les différents types d'IDS

Les N-IDS (Network Based IDS):

- ils détectent les actions malicieuses au niveau du réseau (capture des trames et comparaison à une base d'actions intrusives appelées *signatures d'attaque)*
- Les signatures se basent parfois sur la norme CVE (*Common Vulnerability Exposure*) qui décrit des attaques connues
- Contrairement aux H-IDS, ils ne gèrent pas les attaques opérées sur un terminal (modifications fichiers…)
- Ex : ***Snort, BlackIce, Cisco Secure IDS ou NetRanger, Shadow, RealSecure***

Les H-IDS (Host Based IDS):

- Ils surveillent l'état d'un hôte :
  - contrôle de la gestion de la mémoire (afin d'éviter des dépassements de mémoire : *buffer overflow*)
  - contrôle de la journalisation
  - contrôle des droits des utilisateurs
- Ex : ***Swatch, RealSecure OS Sensor, Intruder Alert***

Les File Integrity Checkers:

- Ils utilisent des outils cryptographiques afin d'établir une base de signatures des données à protéger (*motifs de scellement*). Ceci leur permet de déceler des modifications non autorisées de la table de fichiers
- Ex : ***Tripwire, AIDE, Intact*** ou ***Integrit***

Les autres types d'IDS :

- **Honeypots** (logiciel *Deception Toolkit*) et les **Honeynets** émulent des services dans le but de leurrer un attaquant et de le retarder
- Les **Hybrid IDS (***CyberCop Monitor ou CentraxICE***)** qui regroupent les fonctionnalités d'un N-IDS et d'un H-IDS sur un seul hôte. Très utiles dans le cas d'une architecture commutée où un N-IDS traditionnel ne peut opérer
- Les **Network Node IDS** (NN-IDS) chargé de la supervision d'un groupe de machines
- les **C-IDS** (pour *Comportemental IDS*). Ils surveillent et tentent d'analyser le comportement des utilisateurs internes et externes au réseau :
  - **Détection par anomalie** : elle se base sur l'adage « *tout ce qui n'est pas normal est dangereux* ». Cette démarche, conduit, par exemple, à journaliser tous les accès aux « services exotiques » (i.e. ports non associés à un service connu par exemple)
  - **Détection par mauvaise utilisation** : « *tout ce qui n'est pas dangereux est normal* »
- ***Spade*** (pour *Statistical Packet Anomaly Detection Engine*), ce module effectue un recensement statistique des données sur le réseau local et établit ainsi ce qui correspond à un « fonctionnement normal » ; schématiquement, toutes les données rares sont suspectes.

3.3 Environnement Technique étudié

Une plate-forme de test a permis d'apprécier les fonctionnalités types d'un IDS : ***Snort***[3] compilé avec les options de réponses actives (module « flexresp » permettant la clôture de « sessions malicieuses ») et d'envoi de trappes SNMP en cas d'alerte.

3.3.1 Les attaques mises en oeuvre :

Détection des balayages de ports (« scan de ports »)

- ***Balayage de connexion TCP*** : simulation d'une demande de connexion
- ***Balayage TCP/SYN*** méthode consistant à envoyer un paquet marqué du drapeau SYN sur les ports de l'hôte à sonder. Si le système renvoie un SYN/ACK pour un service donné c'est que le port est à l'écoute
- ***Balayage TCP/FIN*** : balayage basé sur le type de réponse renvoyée en cas d'envoi d'un drapeau FIN à 1
- ***Balayage Xmas Tree*** : envoi à la cible d'une trame avec une suite de zéros ainsi que les drapeaux Urgent (URG), Push (PSH), et FIN à 1
- ***Balayage NULL*** : la trame a tous ces drapeaux à 0 et une séquence de 0

Attaque par validation d'entrée *Unicode* : *Unicode* a pour objectif de former un jeu de caractères unique pour tous les langages existants. L'origine de la faille se trouve dans les logiciels associés à Unicode comme le serveur Web Microsoft IIS

Vulnérabilité test-cgi Cette vulnérabilité permet de recenser l'ensemble des fichiers du serveur attaqué

Vulnérabilité du script *codebrws.asp* permet d'éditer n'importe quel fichier contenu sur le poste de la victime

Remontée de répertoires

Attaque par débordement de tampon (attaque iishack sur le serveur Web Microsoft IIS 4.0) un utilisateur tente de placer plus de données dans un tampon que ce qu'il ne peut en recevoir

Remontée d'informations vers l'attaquant correspond en général à des flux non autorisés par la politique de sécurité de l'entreprise

3.3.2 Mise en évidence des limites de fonctionnement des IDS actuels

quatre limites ont été testées au niveau de la plate-forme :

Augmentation de la fenêtre de temps, test est effectué avec ***nmap***, avec options « Paranoid » (balayage port toutes les 5 minutes) et « Polite » (balayage port toutes les 15 secondes)

Modification de la signature de l'attaque

Attaque DoS, tests effectués avec un « stresseur d'IDS » (***Stick***) :

- ***attaque portant sur le niveau IP*** : falsification de l'adresse IP (l'adresse IP source est égale à l'adresse IP de destination)

[3] www.snort.org

- ***attaque portant sur le niveau TCP*** : tentative de connexion sur un service fermé, ici telnet (port 23 TCP)
- ***attaque portant sur les niveaux supérieurs*** : attaque lancée sur une trame HTTP

Attaque répartie, attaque réalisée par plusieurs assaillants

Chapitre 4 - Appâts (HoneyPot & Honeynet)

4.1 Généralités

Le projet « HoneyPot & Honeynet (HP&HN) [3] » réalisé durant le premier semestre 2002 s'est intéressé aux principe du « pot de miel » : mise en œuvre de dispositifs destinés à leurrer les attaquants pour les écarter des cibles sensibles et pour mieux étudier leurs comportements. L'objectif du projet était d'une part de réaliser un état de l'art sur les HoneyPot & Honeynet et d'autre part de réaliser une plate-forme de démonstration.

Les HoneyPot & Honeynet ont l'avantage de générer des données de connexions pertinentes avec en général un bruit faible. Contrairement aux systèmes de détection d'intrusions, ils ne sont pas submergés par des flux réseaux et offrent donc en général une bonne disponibilité. Les deux principaux inconvénients liés aux HoneyPot & Honeynet sont d'une part la perte de ressources matérielles et d'autre part, s'ils sont mal maîtrisés, la possibilité pour un attaquant de se servir de ces dispositifs comme un rebond pour mener ses attaques vers d'autres systèmes.

Parmi les HoneyPot & Honeynet on peut distinguer ceux de production et ceux de recherche. Les HoneyPot & Honeynet de production sont destinés à leurrer les attaquants pour les écarter des zones sensibles du site en leur proposant des défis plus simples. On parle alors souvent de «honeypots ». Les pots de miels de recherche sont destinés à étudier le comportement des attaquants pour mieux comprendre leurs méthodes et leurs stratégies. Les « honeynets » par exemple recréent aux mieux les conditions d'un réseau d'entreprise en mettant souvent plusieurs machines et matériels réseau à la disposition de l'attaquant.

L'aspect le plus délicat des HoneyPot & Honeynet est sans doute le paramétrage de leur niveau de sécurité. En deçà du seuil dit de la « limite d'écoute », les informations qu'ils génèrent sont polluées par les attaques à faible niveau de compétences qui sont parfaitement connues des responsables systèmes et qui n'apportent aucune information supplémentaire sur le comportement des attaquants. Au-delà du seuil dit de la « limite de risque acceptée », le HoneyPot & Honeynet devient fortement compromis et peut mettre en danger le reste du système d'information (Cf. plus loin les aspects « gradation de la difficulté d'accès).

D'un point de vue pratique les honeypots sont souvent basés sur des scripts qui émulent des services et des environnements ainsi que sur des environnements dits de « prison ». Les architectures des honeynets sont plus variées et complexes et mettent en œuvre des mécanismes de contrôle de données et de capture d'informations. Les mécanismes de contrôle de données ont pour but de maîtriser les flux des attaquants pour les empêcher d'attaquer d'autres systèmes. Les mécanismes de capture d'informations servent à archiver de façon sûre les actions menées par les attaquants pour ensuite pouvoir procéder à une analyse de leurs comportements.

4.2 Quelques Eléments constitutifs de l'étude

4.2.1 Périmètre réel des HoneyPots et HoneyNets

Le concept de HoneyPot et HoneyNet dépasse largement celui des outils censés les concrétiser, la « cartographie »[4] présentée ci-dessous propose un périmètre plus juste des aspects à prendre en compte :

[4]. Cette réflexion peut s'inscrire dans le cadre plus global de l'étude d'une « Cartographie de la Maîtrise de l'Information », HoneyPot (et HoneyNet) représentant alors un « composant » mis à disposition.

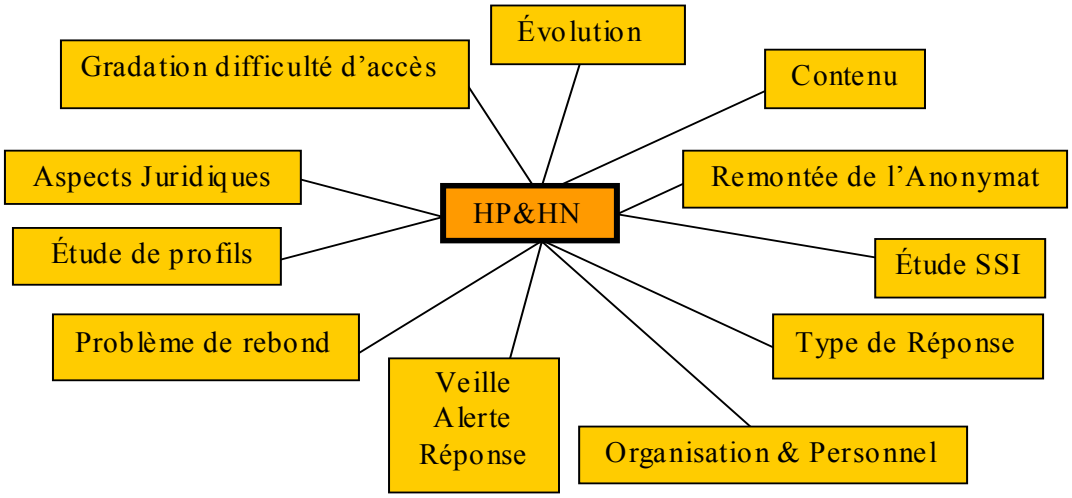


Figure 2 : Les aspects des HoneyPots et HoneyNets.

Organisation et personnel : compétence nécessaires (profils), processus de sélection (conservation du secret)

Aspects juridiques : notion de piège et éléments de preuve, responsabilité de l'organisme

problèmes de rebond : risques judiciaires, information des sites concernés (amont, aval)

Remontée de l'anonymat : droits et outils adaptés

Contenu : composition de la partie réelle et de la partie désinformation

Gradation de la difficulté d'accès : modulation du niveau de sécurité de l'HoneyNet (étape **(1)**), pour sélection d'une catégorie d'attaquants particulière (étape **(2)**)

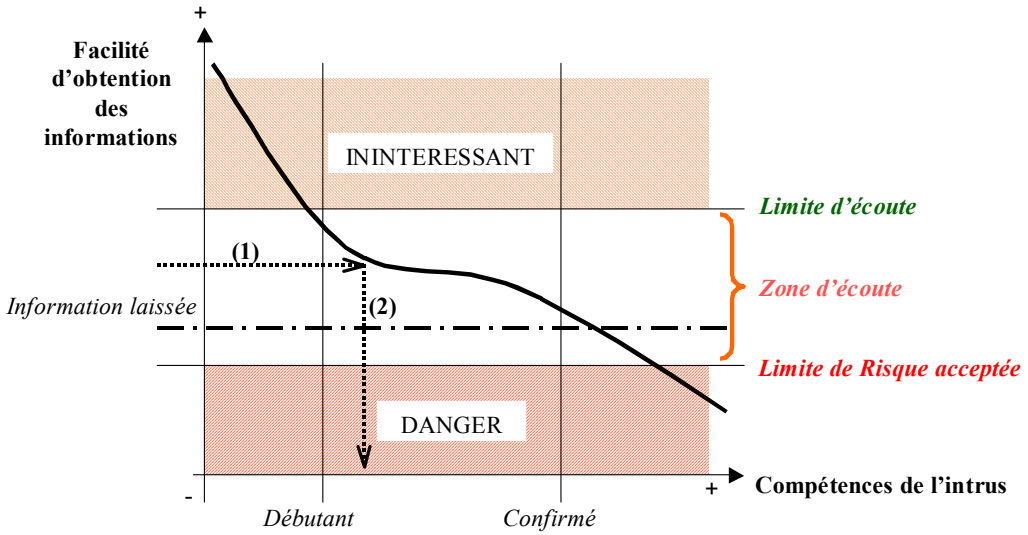


Figure 3 : Sécurisation de l'information et compétences de l'intrus.

Évolution : en fonction des « modes », de l'étude des attaques et de l'évolution des stratégies d'attaque

Étude de la Sécurité des Systèmes d'Information (SSI) : test de moyens de protection ou copie conforme de la politique SSI interne

Type de réponse : coupure de connexion, envoi d'un message d'avertissement, riposte

Veille, Alerte, Réponse (V.A.R.) : liens avec système VAR de l'organisme

Étude du profil : analyse du profil des attaquants, liens avec systèmes de protection

4.2.2 Le HoneyPot

Le honeypot est, à l'heure actuelle, un produit aux fonctionnalités plus ou moins avancées et présentant surtout des degrés d'interaction avec l'attaquant différents.

- **Émulation par scripts** : un ensemble de scripts émule un certain nombre de vulnérabilités connues. (ex version ancienne d'un serveur de messagerie *Simple Mail Transfer Protocol* (SMTP) à même de fournir un fichier de mots de passe comportant de fausses empreintes) ; l'intérêt d'un tel outil réside dans le leurre et dans la déception[5]
- **Émulation de services et d'environnements :** produits, libres ou commerciaux, permettant de recréer un discours logique avec l'attaquant par réplication de la pile IP.
- **Création d'environnements dits « prisons » :** exécute une image, appelée « prison », d'un système d'exploitation au sein d'un autre ; Cette image est surveillée et ne déborde pas vers le système hôte, elle propose un environnement complet à l'attaquant.

4.2.3 Architecture et modèle des HoneyNets

Les HoneyNets constituent des systèmes réels, rien n'est émulé ; leurs architectures reposent sur deux aspects critiques, le contrôle des données, et la capture des informations.

La figure suivante montre ce que pourrait être un honeynet. Trois réseaux séparés par un pare-feu :

- l'Internet, réseau d'où surviennent les attaques
- le honeynet, ensemble de machines dont le but est d'être testé, attaqué ou compromis (HP = honeypots)
- le réseau d'administration, réseau de confiance où seront recueillies les données du honeynet et permettra l'administration de celui-ci.

[5] entendre ici l'acquisition de données sur lesquelles l'attaquant dépensera du temps pour un profit nul

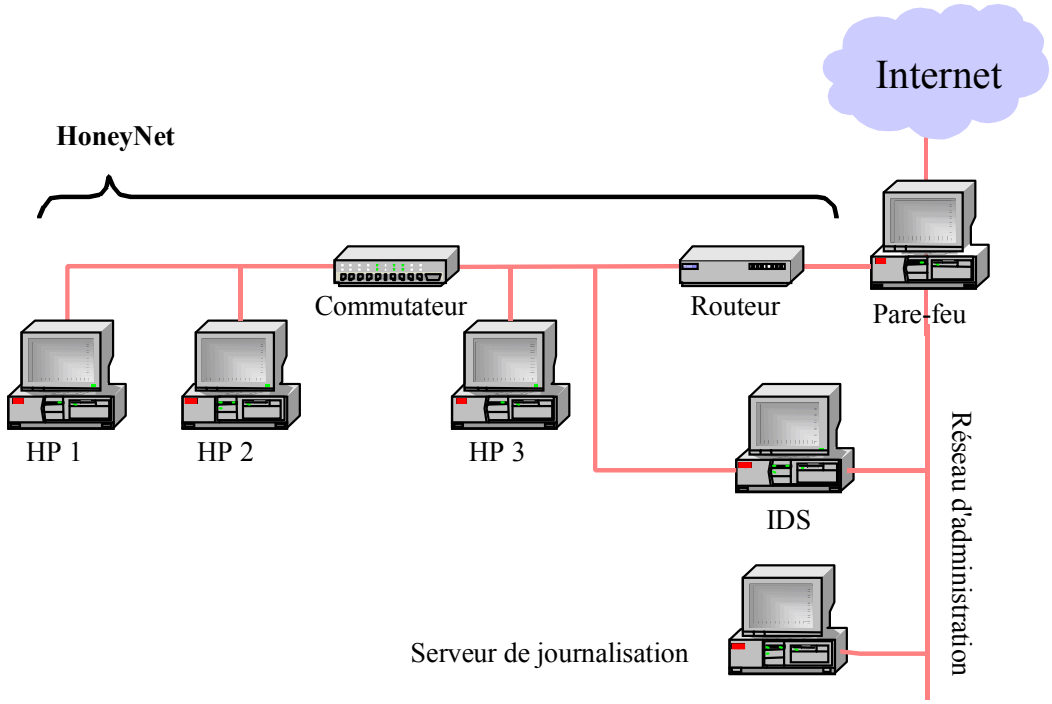


Figure 4 : Architecture d'un HoneyNet.

Contrôle des données en provenance de l'Internet : il ne sera pas utile de tout observer ou laisser passer, il faudra limiter le trafic en provenance de l'Internet

Contrôle des données en provenance des honeypots : éviter les rebonds, permettre la capture de données manipulées

4.3 Environnement Technique étudié

honeypots.

- ***Deception ToolKit*** de Fred COHEN, programme d'émulation par scripts, est un produit libre du monde Linux.
- ***Back Officer Friendly*** de NFR Security, est libre de téléchargement pour un usage personnel, prévient l'administrateur de toute tentative de prise de contrôle de la machine par « *Back Orifice* »[6], donne à l'attaquant de fausses informations, tout en enregistrant les adresses de provenance de l'intrus et ses différentes manipulations.(contient aussi des programmes simulant des services, comme transfert de fichiers (FTP), web (HTTP), ou courrier (SMTP))
- ***Honeyd*** est un démon Linux qui crée des hôtes virtuels, il permet de simuler un réseau complet répondant aux fonctionnalité TCP/IP (réponses PING et TraceRoute par exemple).
- ***Recourse ManTrap*** produit de la catégorie des systèmes de détection d'intrusion, il crée des environnements prisons surveillés pour repérer, contenir et suivre toute attaque. Cet outil permet aussi de gérer les réponses à mettre en œuvre : suivant le degré de l'attaque, il pourra alerter,

[6]. « *Back Orifice* » est un programme répandu. Il permet, par l'installation d'un serveur sur la machine Windows convoitée, de donnée un accès complet à l'attaquant exécutant le client correspondant. Ceci lui permettra de revenir, en utilisant cette porte dérobée, comme bon lui semble.

enregistrer le déroulement de l'attaque, terminer la session, ou exécuter toute autre action personnalisée.

- ***Specter*** émule différents systèmes, des plus répandus, comme Windows 98, NT, 2000, MacOs, Linux et autres UNIX comme Solaris, NeXTStep, Tru64, Irix, AIX.

Pour les honeynets

- Contrôle d'accès : ***CheckPoint FireWall-1*** est un pare-feu très répandu ; en plus des fonctionnalités primaires, il permet un enregistrement dans des journaux, nécessaires pour l'étude post-attaque, le suivi des connexions actives, ou la surveillance de la sécurité des exécutables Java ou ActiveX.
- Couche réseaux : ***IDS SNORT*** système de détection d'intrusion pour réseau (NIDS) ; léger et aisé à mettre en place
- Couche Hors-ligne : ***TripWire,*** outil libre pour le monde Linux, vérifie l'intégrité du système ; ***The coroner's Toolkit*** boîte à outils pour l'analyse post-mortem du système

Chapitre 5 - Passerelle inter-réseaux de sensibilités différentes

5.1 Généralités

La sécurité des échanges entre systèmes d'information est un problème d'actualité d'autant plus critique que l'on assiste de nos jours à une dématérialisation de plus en plus généralisée de l'information et des services associés. Cette dématérialisation s'opère bien évidemment dans des secteurs publics sensibles tels que le secteur bancaire, avec les transactions en ligne ou le secteur de l'assurance maladie, avec la création de la carte électronique de l'assuré, mais le secteur de la Défense ne fait pas exception à la règle puisqu'il connaît une demande croissante en la matière : le cadre opérationnel tactique implique de plus en plus d'échanges et de partage d'informations sensibles (intervention dans le cadre de coalitions).

Malheureusement, la recrudescence et la diversité des attaques (déni de service, compromission…) tendent à fragiliser les réseaux de communication actuels et l'on doit donc recourir à des solutions d'échanges sécurisés de l'information.

Or, à l'heure actuelle, les diverses réglementations ne préconisent pas (loin s'en faut) l'échange directe d'information entre deux systèmes de sensibilité différente, elle impose parfois même une rupture physique afin de prévenir une éventuelle fuite d'information ou se prémunir d'éventuelles attaques externes. Cette rupture physique empêche l'automatisation de l'échange des données, et nuit alors aux performances du service offert par le système.

C'est donc afin de pallier cette limitation que THALES a étudié, dans le cadre du programme « Système d'Echange Sécurisé pour un Accès Multi niveaux Expérimental (SESAME) [5] » la possibilité de créer une passerelle dite « multi niveaux » sécurisée qui réponde aux contraintes fortes liées à l'interconnexion de ce type de systèmes.

Cette passerelle met donc en oeuvre un mécanisme d'échange d'information sécurisé entre des systèmes de sensibilité différente reposant principalement sur une notion de SAS cloisonnant les deux systèmes ; ce SAS devant être automatisé autant que possible afin limiter le besoin d'intervention humaine.

5.2 Quelques Eléments constitutifs de l'étude

5.2.1 Rappel sur la réglementation

Afin de réaliser des systèmes sécurisés et homologués pour une utilisation dans cadre classifié de défense, toutes les exigences de sécurité, d'un point de vue technique et organisationnel, sont traduites au sein de textes de loi (dispositions législatives et réglementaires[7]) ; elles doivent être prises en compte en particulier lors de l'interconnexion de systèmes d'information.

5.2.2 Etat de l'art

Interconnecter physiquement deux réseaux de sensibilité différente est à l'heure actuelle souvent interdit, la seule solution sûre et généralement autorisée pour un échange d'information reste à ce jour la rupture physique entre les deux systèmes (mesure organisationnelle) : la transmission d'information s'effectue alors par support externe (par exemple par disquette ou cédérom) détenu par une personne habilitée (au niveau de sensibilité le plus élevé mis en jeu dans le cadre de cet échange).

[7] *Rappel* : dans le cadre de l'OTAN, les deux textes de référence sont « Security within the North Atlantic Treaty Organisation (NATO) - C-M(2002)49 du 17 juin 2002" pour ce qui concerne l'homologation des SI, et « Directive sur les aspects techniques et la mise en oeuvre de l'INFOSEC pour l'interconnexion des systèmes d'information et de communication (SIC) - AC/322-D/0030-REV1 » pour ce qui concerne les interconnexions.

5.2.3 Concepts techniques et organisationnels

Une architecture d'échange sécurisée devra, donc, mettre en œuvre les éléments constitutifs suivants :

- Un « SAS automatisé » : un système d'échange par lequel transite l'information sans qu'il y ait jamais de lien continu établi entre système émetteur et système récepteur. Le SAS répond donc au besoin de coupure physique et rend le système étanche aux attaques extérieures.
- Un système de marquage ou de « labellisation » : permettant de caractériser la nature de l'information en associant à celle-ci une description formelle plus ou moins détaillée de son contenu et de sa sensibilité.
- Une autorité « Valideur » : autorité en charge de décider si le caractère d'une information donnée autorise une transmission vers le système de sensibilité inférieure
- Un système « Infrastructures à clé publique, ou PKI (Public Key Infrastructure) » : Infrastructure qui permet d'établir une relation de confiance dans le cadre d'une politique de sécurité. Une telle infrastructure est composée de plusieurs éléments :
  - en premier lieu une autorité de certification qui est chargée de générer les certificats, en associant l'identité d'une personne ou d'un système à une signature numérique ;
  - le second élément est l'autorité d'enregistrement qui capture et identifie l'identité des utilisateurs et soumet les demande de certificats à l'autorité de certification ;
  - le troisième composant d'une PKI est le système de distribution des clés.
- Un système de « Pare-feu » : système permettant de protéger un ordinateur des intrusions provenant du réseau.

5.2.4 Une Passerelle de type hybride conséquente

La différence de sensibilité des flux montant et descendant nous mène à différencier le traitement à réaliser pour chaque sens d'échange de données.

Ainsi, on utilisera une architecture de type diode simple au niveau du flux montant (ce flux étant généralement autorisé). Par contre, le sens descendant étant plus sensible, on a recours à une architecture plus contraignante, associant une diode à un SAS. Le principal avantage de cette architecture étant lié à l'apport de notion de coupure physique entre les deux systèmes grâce au sas modélisé par les 3 pare-feux.

Passerelle de type hybride

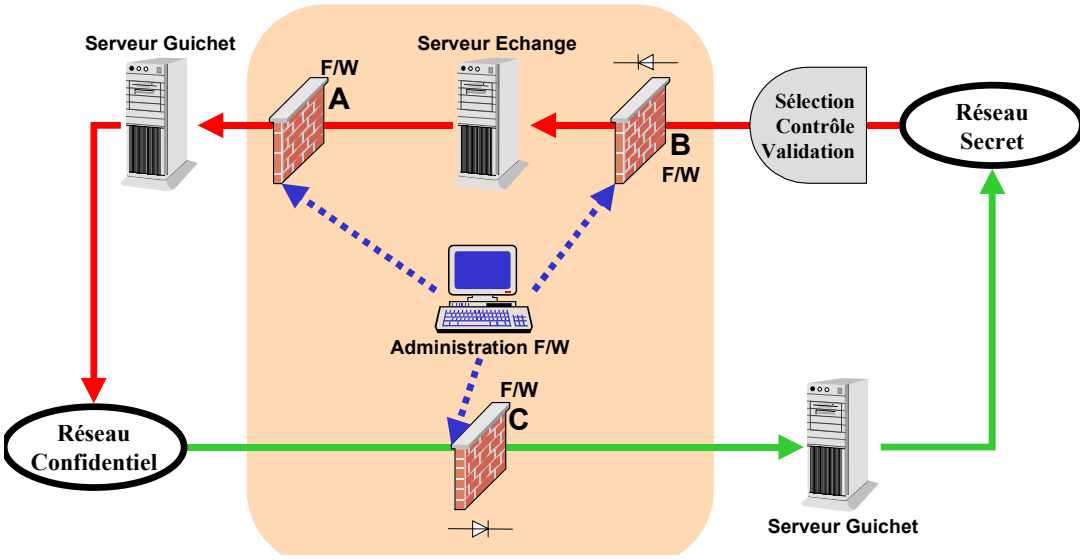


Figure 5 : Passerelle de type hybride

Description :

- Cette modélisation apporte une optimisation des ressources utilisées pour le transfert des informations tout en essayant de conserver la notion de coupure physique. En effet, les deux pare-feu ne sont jamais passants en même temps et sont ouverts et fermés en alternance. Ainsi on garantit bien que l'on a jamais un lien constant entre les deux systèmes d'information
- La passerelle, pour les échanges dans le sens descendant, s'apparente à une DMZ classique. En effet, le serveur d'échange sert de proxy et est protégé par un pare-feu A. Le pare-feu B supplémentaire fonctionne en mode « diode » et permet d'assurer un échange unidirectionnel (et protège le réseau haut des attaques).

5.3 Environnement Technique étudié

protocole de transfert	***FTP*** (File Transfert Protocol)
développements	***Java*** sur plate-forme LINUX (avec le logiciel I.D.E. ***Jbuilder*** pour la réalisation de l'IHM)
structuration des messages échangés	***XML***
protection par pare-feu*x*	***iptables*** de LINUX avec règles de filtrage réalisées grâce au logiciel ***Firewall Builder***

Chapitre 6 - Dissimulation et fuite d'information (Canaux Cachés)

6.1 Généralités

« L'étude sur la dissimulation et la fuite d'information par l'intermédiaire de canaux cachés (EFFICACE) [6] » avait pour objet la réalisation d'un état de l'art sur ces notions. Un panel des techniques permettant ces actions, comme la stéganographie et l'encapsulation protocolaire, ont été ainsi étduiées, des solutions préventives ont été préconisées et mises en application afin de limiter le champ d'action d'un éventuel attaquant et par conséquent la violation de la politique de sécurité mise en oeuvre.

Ce projet s'inscrit dans la continuité logique des études présentées plus haut. Elle a été menée sur la plate-forme Tests d'Intrusions dans les Réseaux Informatiques (TIRI) et fait suite aux études sur l'intrusion et Génération de Rapports d'Audit (TIGRA) et les Honeypot (HP/HN).

Cette étude avait pour objectif d'étudier les techniques et expérimentations actuelles consacrées à la dissimulation et la fuite d'information au sens large, et orientée particulièrement sur les problèmes de sécurité suscités pour les réseaux sensibles ; elle avait pour ambition de permettre l'identification de ce type de vulnérabilités, et dans la mesure du possible d'y associer des contre-mesures.

Cette étude partait du principe que les failles de sécurité, dans le cadre d'un échange d'information, ne se situent plus à partir de la couche réseau[8] (niveau 2 du modèle OSI) mais à partir de la couche application (niveau 7 du modèle OSI) ; elle s'est déroulée en parallèle de l'étude destinée à sécuriser le transfert de données entre deux réseaux de sensibilités différentes (Cf. Passerelle inter-réseaux) [9].

Dans le cadre de ces échanges d'information, EFFICACE nous a amenés à distinguer deux notions :

- l*a dissimulation* qui désigne le fait de rendre invisible une donnée. Typiquement, il peut s'agir d'une donnée cachée au sein d'une autre, au sein d'un système de fichier, d'un processus tournant en tâche de fond et non visible pour un utilisateur ;
- l*a fuite* qui fait référence à un transfert d'information non désiré notamment par l'usage de canaux cachés. Une illustration de ce concept est la récupération d'informations sensibles présentes au sein d'un réseau local sécurisé depuis un poste situé sur l'Internet par l'utilisation de failles protocolaire.

Cette étude est partie des considérations suivantes :

- toute information est susceptible de contenir de l'information cachée
- la vérification de la présence d'une donnée dissimulée n'est pas l'objet de cette étude (même si cela a été réalisé à titre indicatif)
- un opérateur peut à tout moment vérifier les données échangées sur un réseau.

Outre ces principes, des limites ont été inclues à notre étude car chaque technique évoquée au sein de ce document pourrait faire l'objet d'une étude à par entière :

- cet état de l'art porte uniquement sur les réseaux de type IPv4 et failles liées aux protocoles TCP/IP,

[8] Il existe déjà , en effet, un certains nombre de solutions, dont l'équipement FOX de THALES, qui permettent de réduire considérablement les risques de vulnérabilités liées aux couches réseaux.

[9] L'objectif de SESAME est de se prémunir contre tout transfert d'information violant la politique de sécurité en vigueur pour les réseaux sensibles.

- l'aspect mathématique n'a pas été abordé.

6.2 Quelques Eléments constitutifs de l'étude

6.2.1 Les techniques employées dans le processus de transfert et de fuite de l'information

Les faiblesses des éléments constitutifs d'un réseau

- **Les IDS** détection des vulnérabilités connues uniquement et aucune protection sur les transactions sécurisées SSL
- **Les pare-feux** agissent généralement exclusivement sur le type des communications mais pas sur leur contenu

Les faiblesses protocolaire

- **Exploitation de l'en-tête TCP/IP** construction de faux paquets en utilisant certains champs des en-têtes IP et TCP
- **Exploitation de champs d'en-tête inexploités** Ex : TCP et IGMP
- **Exploitation de la zone de bourrage**
- **L'encapsulation protocolaire** ex :encapsuler des requêtes dans des paquets ICMP dont les champs optionnel sont enregistrés et non vérifié (route empruntée par la trame, temps de parcours)

Le tunneling permet d'utiliser des protocoles non autorisés à travers un pare-feu

Les troyens remplissent des fonctions non désirées, inconnues de l'utilisateur

Les spywares programmes conçu dans le but de collecter des données personnelles sur ses utilisateurs. Ces données sont généralement envoyées à son concepteur

6.2.2 La stéganographie

Quelques exemples de techniques de stéganographie étudiées.

Substitution d'information et espaces réservés

- **Substitution des bits de poids faible** remplacement des bits de poids faible de codage d'un pixel d'une image (JPEG, GIF) par d'autres bits d'information
- **Permutations pseudo-aléatoires** incorporation d'informations dans les propriétés statistiques de la luminance des pixels (ex : *Outguess*)
- **Les bits de redondance** écrasement de bits redondants par d'autres bits d'information
- **Réorganisation de la palette de couleurs** réorganisation obtenue par la modification des bits de poids faibles du codage des pixels
- **Dissimulation dans des fichiers binaires**
- **Exploitation d'espaces inutilisés au sein d'un ordinateur**
- **Les système de fichiers stéganographique** (les systèmes de fichiers stéganographiques eux-mêmes qui offrent deux niveaux logiques de stockage d'information, et les espaces interstitiels, utilisation de l'espace de stockage alloué à un fichier et non utilisé)

- **Exploitation du système de fichiers NTFS** exploitation des flux de fichiers, ou ADS

Les techniques de transformée

- **Transformation en Cosinus Discret** cette méthode exploite le fait que les valeurs calculées par l'utilisation de cette transformée ne sont pas précises, ainsi la répétition de tels calculs conduit à une limitation de la précision et à l'introduction d'erreurs
- **Stéganographie appliquée à un fichier audio** la dissimulation d'information dans les plages de fréquences inaudible à l'oreille humaine

L'approche statistique méthode utilisée afin de leurrer les outils de stéganalyse

Encodage d'information dans un texte texte dans le texte, mots codes ou synonymes, ajout d'espaces à la fin des phrases ou espacement entre les mots

Techniques appliquées aux documents propriétaires

- **Adobe** utilisation de l'option du menu Document/Recadrer pour cacher les marges
- **MS-Office** informations « propriété », inclusion automatique de données (INCLUDTEXT), utilisation éditeur hexadecimal (*UltraEdi*t)

MS OUTLOOK et MS EXCHANGE exploitation du fichier winmail.dat

6.2.3 Le Watermarking : état de l'art

Le watermarking diffère de la stéganographie par le fait que l'on se limite souvent à dissimuler très peu d'information (très souvent un seul bit) dans l'image hôte et a pour objectif de démontrer l'intégrité du document ou encore d'en protéger les droits d'auteur.

Les techniques de marquage

- **Modification des bits de poids faible** les premiers algorithmes allaient inscrire la marque dans les bits de poids faible de la luminance de l'image
- **Technique du "Patchwork"** répétition un grand nombre de fois du même bit pour qu'une étude statistique donne le bit marqué
- **Algorithme de Koch et Zhao** marquage d'un bit sur les moyennes fréquences correspondant à un calcul de transformée en cosinus discrète
- **Watermarking par étalement de spectre** envoi d'un message sur un grand spectre de fréquences de telle manière que, à toute fréquence, la puissance du signal émis soit inférieure au bruit

Les traitements malveillants

- **L'attaque par mosaïque** effacement de signature par morcellement d'image
- **Stirmark** banc de test permettant d'apprécier la robustesse d'un schéma de tatouage d'image appliquant principalement des distorsions géometriques

6.3 Environnement Technique étudié

6.3.1 Quelques outils stéganographiques

Médium de type texte

- **TextHide** *Syst. Unix, Windows* dissimule des données par l'intermédiaire d'un dictionnaire de synonymes
- **StegParty** *Syst. Unix* utilise la ponctuation
- **Snow** *Syst. Unix* utilise les tabulations et les espaces de fin de ligne
- **NiceText** *Syst. Unix* utilise un dictionnaire de synonymes
- **FFEncode** *Syst. DOS* dissimulation d'information par l'intermédiaire de caractères nuls utilisant un code de type morse

Médium de type image

- **Invisible Secrets** *Syst. Unix, Windows* dissimulation d'un message dans de nombreux types de fichier (JPEG, PNG, BMP, HTML et WAV).
- **Gifshuffle** *Syst. Unix, Windows* permutation des couleurs dans la palette
- **JPegX** *Windows* que JPG
- **BMP Secrets** *Windows* que BMP
- **Digital Picture Envelope** *Windows*
- **CryptArkan** *Windows* dissimule des données dans une image BMP ou un fichier audio de type WAV
- **Cameleon** *Windows* chiffrement AES 256 bits pour des données cachées
- **JSteg Shell** *Syst. Unix, Windows*
- **Outguess** *Windows* seul programme qui reste indétectable par ***Stegdetect 0.2***

Médium de type fichier audio

- **WeavWav** *Freeware, Windows* dissimulation d'information dans un fichier de type WAV
- **Stego-Lame** *Freeware, Windows* différents formats (MP3, Ogg Vorbis, etc.)
- **MP3Stego** *Freeware, Windows, Syst. Unix* peut également servir comme outil de marquage de fichier MP3

Médium de type fichier compressé

- **GZSteg** *Freeware, DOS* dissimule de l'information au sein d'un fichier de type GZip

Médium de type document propriétaire

- **FactMiner** permet d'extraire un grand nombre d'informations sur l'auteur du document, le produit d'édition utilisé, la langue, etc. Ce logiciel est appliquable aux documents de types texte, RTF, HTML, SGML, XML, PDF, PostScript

6.3.2 Quelques Exemple de troyens

ACK Cmd permet d'établir une communication entre un attaquant et un serveur par l'envoie de segments ACK ("Ack Tunneling").

Gatslag combinaison entre un tunnel et un troyen, il exploite les faiblesses d'Internet Explorer

SETIRI version améliorée du troyen ***Gatslag*** il ne contient aucune commande exécutable susceptible d'être bloquée par un pare-feu

Chapitre 7 - Conclusion

Tous ces aspects restent à conforter par les réalisations et études internes relatives aux :

- Systèmes et équipements utilisables dans un cadre OTAN (ex : chiffrement IP agréé OTAN TCE621, chiffreur Echinops (agrément OTAN en cours), système de contrôle d'accès au postes Minicita)
- Equipements et architectures techniques de protection de bordure des système opérationnels de coalition (sécurisation des WAN de théâtre, interconnexion de systèmes multi-niveaux de sécurité, passerelles, etc.)
- Outils et études dans le cadre de la maîtrise de l'information et de la gestion de l'environnement psychologique (PSYOPS)
- Etudes VAR

Références

[1] projet **TIRI** (Tests d'Intrusions dans les Réseaux Informatiques), réalisé par *Sébastien Breton* premier semestre 2000

[2] projet **CCSI** (Configuration Centralisée des Systèmes d'Information) réalisé par *Vincent Bechet* septembre 2001

[3] projet **HP&HN** (HoneyPot & Honeynet) réalisé par *Franck Crepel* premier semestre 2002

[4] projet **TIGRA** (Tests d'Intrusion et Génération de Rapports d'Audit) réalisé par *Patricia Cabanel* premier semestre 2002

[5] projet **SESAME** (Système d'EchangeSécurisé pour un Accès Multi-niveau Expérimental) réalisé par *Renaud SCHAUBER* et *Arnaud KEMP* octobre 2003

[6] projet **EFFICACE** (Elaboration Furtive de Fuite d'Information CAChés atemporels d'Evasion) réalisé par *Sébastien POLLET* Septembre 2003

[7] Projet **THONIC** (synthèse des autres travaux) réalisée par *Vincent Ribaillier* octobre 2002

Attack Processes Found on the Internet

Marc Dacier, Fabien Pouget
Eurecom
2229, route des Crêtes; BP 193
06904 Sophia-Antipolis Cedex; France

{dacier, pouget}@eurecom.fr

Hervé Debar
France Télécom R&D
42, rue des Coutures; BP 6243
14066 Caen Cedex 4; France

herve.debar@francetelecom.com

ABSTRACT

In this paper, we show that simple, cheap and easily deployable honeypots can help to get a better understanding of the attack processes that machines in unclassified networks are facing. Acquiring this knowledge is a prerequisite for the sound design and implementation of efficient intrusion tolerant systems. We propose some in depth analyses carried out on data gathered during a 10 months period by several honeypots. We highlight the need for a well defined set up of honeypots, replicated in many diverse locations. Such an environment would enable the scientific community to answer the remaining open issues described here after.

1.0 INTRODUCTION

Recently, several papers have explained how so-called "Internet telescopes" can be used to get a better understanding of worms propagations ([44, 45]). As a follow up to large scale DdoS attacks, the scientific community has shown a growing interest for a pragmatic analysis of real data streams to identify the various attack processes threatening Internet users. For instance, three papers where devoted to these problems during the last SIGCOMM conference [60], and four other published at INFOCOMM 2003 [34].

In this paper, we propose to use simple honeynets instead of large "telescopes" to gather data. Based on our experimental set up we show that despite orders of magnitude of differences in the amount of collected data this approach is more efficient. Indeed, the limited, yet richer, amount of data offers a convenient way to carry out some systematic analysis that leads to interesting findings.

This paper is a follow up to an earlier publication [18] in which first results obtained over a four months period had been proposed. In this new publication, we review the conclusions drawn in [18] thanks to six more months of data and, more importantly, propose new findings. We discuss the need for a distributed infrastructure in order to confirm some of our conjectures and to answer some of the remaining open questions.

The paper is organized as follows. Section 2 proposes a general introduction to the notion of honeypots. Section 3 offers a survey of existing work. Section 4 describes the set up we have used for our experiments. Section 5 reviews the results proposed in [18] and offers new material. Section 6 concludes the paper.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

2.0 HONEYPOTS: INTRODUCTION

2.1 Definitions

Honeypots, honeytokens and honeynets have been used for more than fifteen years in computing systems even if the use of this terminology is recent. In the late 80's, Clifford Stoll [67] had the idea of placing 'interesting' data in appropriate places to lure hackers. In the 90's Cheswik implemented and deployed a real "honeypot" [12]. Bellovin discussed the very same year the advantages and problems related to its usage [6]. In 98, Grundschober and Dacier ([26, 27]) introduced the notion of "sniffer detector" (see also [1]), one of the various forms of what is called today a "honeytoken".

Lance Spitzner has the merit of having been the first one to try to define these concepts by introducing the terms "honeypot", "honeytoken" and "honeynet". Yet, we feel uncomfortable with his definitions because they describe how to use honeypots instead of defining what they really are. However, due to the lack of any better definition, the community of honeypot users/developers keeps using the following ones, taken from [65][1]:

- *Definition 1: «A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.» [65]*
- *Definition 2: «[…] a honeytoken is a honeypot that is not a computer. Instead it is some type of digital entity. » [65]*

In order to provide a more rigorous definition of what honeypots are, we propose to take advantage of well defined concepts introduced by the dependability community. To that end, we reuse the definitions proposed by several contributors within the European MAFTIA project. In [53, page 32], the authors introduce the notions of attack, vulnerability and intrusions as follows:

- *Definition 3: «An **attack** is a malicious interaction fault, through which an attacker aims to deliberately violate one or more security properties ; an intrusion attempt»*
- *Definition 4: «A **vulnerability** is a fault created during development of the system, or during operation, that could be exploited to create an intrusion»*
- *Definition 5: «An **intrusion** is a malicious, externally-induced fault resulting from an attack that has been successful in exploiting a vulnerability»*

Based on these definitions, we derive the following new one:

- *Definition 6: A **honeypot** consists in an environment where vulnerabilities have been deliberately introduced in order to observe attacks and intrusions.*

2.2 Existing platforms

During the last 2 years, many different implementations of the concept of honeypots have been proposed. Some attempts have been made to classify them (see for instance [15, 29, 64]). For the sake of completeness, we offer a brief overview of the existing solutions in Appendix A. We report the interested reader to [52] for a more detailed presentation of these tools as well as for a discussion of the classification issues.

Besides the large number of solutions, it is worth noting that some work has also been done to propose software architectures, such as the GENI and GENII Honeynet [32], to create more sophisticated

[1] It is worth noting that a) this definition is different from the one given by the same author in his book [64], ii) this new definition has been discussed at length on the honeypot mailing list [30] but no final consensus has been reached among the participants.

environments by integrating various pieces of hardware and software. More recently, a strong trend has emerged in favor of the use of so-called virtual honeynets where a complete network is emulated by a single machine using tools such as User Mode Linux [31, 71], VMware [59, 72] or VSERVER [73].

3.0 STATE OF THE ART

3.1 Introduction

As indicated by Appendix A, most of the effort spent by honeypots developers has been devoted to implementation issues. Besides this, we identify in Section 3.2 three main trends in the usage of data collected by honeypots. In Section 3.3, we also consider the vast body of knowledge that has been accumulated by the people studying networking performance issues.

3.2 Research on data collected by honeypots

3.2.1 Post mortem analyses

In this case, honeypots machines are expected to be fully compromised by hackers [50]. Once this stage is reached, the machine is halted and analyzed by means of tools such as the « coroner toolkit » [25] or its successor, the «Sleuth kit» [9]. The main purpose of these analyses is to discover new software tools used by hackers and not yet found in the wild.

3.2.2 Identification of new threats

It has been claimed that honeypots could also be used as early warning systems. They could be designed to quickly identify new types of threats. However, as far as we can tell, no published work has investigated this problem in some depth. Some anecdotic report exists, such as the ones done with wireless honeypots [16, 39, 75] that aim at showing the risk of leaving wireless networks unattended [74].

3.2.3 A statistical data gathering tool

The rapid evolution of existing platforms might be the reason for the surprising lack of publication of data collected by honeypots over a long period of time. The most visible project, the honeynet project has published a first document in 2001 [30] but, since then, seems to have focused on implementations issues. The Irish team appears to be the only member of the Honeynet Research Alliance to offer such data on its web site [28] but this concerns their sole environment and it does not provide any kind of analysis. A noteworthy exception in the field of statistical data analysis can be found in [20] where the authors analyze, thanks to their honeypots, the propagation of the NIMDA worm. Thus, as of today, only one team seems to have investigated the possibility of using data from honeypots to model attack processes.

3.3 Research in network monitoring

As early as 1993, Bellovin [7] has shown the interest of studying real packets passing on the networks. He showed the existence of anomalous behaviors, packets that did not indicate an attempted break-in but that, nevertheless, were worthy of attention. The museum of broken packets [78] offers a survey of such weird packets. There is now a conference [37] (previously a workshop [35, 36]) where results of work dedicated to the analysis of real data streams are presented. Security issues were not really considered by that community in the past. The focus was more on performance, quality of service and optimization issues. The rise of denial of service attacks has changed things. In 2001, Moore et al. published the first quantitative analysis of these phenomena [46], followed by others [5, 44, 63]. During the last SIGCOMM conference [60], three papers were dedicated to that problem. Four others were published at the last

INFOCOM [37]. Two things are important to mention at this point: i) these studies focus on Denial of Service attacks, which represent only the tip of the iceberg in terms of attack processes to look at, ii) besides [11] none of these teams has had access to data collected by honeypots.

4.0 TESTBED DESCRIPTION

Our experimental set up consists in a virtual network built on top of VMware [72] to which three virtual machines, or guests in the VMware terminology, called mach0, mach1 and mach2 are connected. The VMware commercial product enables us to configure them according to our specific needs. mach0 is a Windows98 workstation, mach1 is a Windows NT Server and mach2 is a Linux Redhat 7.3 server. The three virtual guests are built on non-persistent disks [72]: changes are lost when virtual machines are powered off or reset. In other words, rebooting a compromised machine is a convenient and simple backward error recovery mechanism. The three machines are attached to a virtual Ethernet switch[2]. ARP spoofing is implemented so that they can be reached from the outside world. A fourth virtual machine is created to collect data in the virtual network. It is also attached to the virtual switch and tcpdump is used as a packet gatherer [69]. This machine and the VMware host station are configured to be invisible from the outside world. Both Mach1 and Mach2 run an ftp server; in addition, Mach1 also provides a static web server. Logs are collected daily and transferred to an independent and safe place where they are enriched with some external data, as discussed below, and inserted into a database.

5.0 RESULTS

5.1 Introduction

First and foremost, we introduce two definitions which will be used in the following:

- *Attack Source:* it defines an IP address that targets our honeypot environment within one day. This time constraint is arbitrary and based on our observation only: attacks are all limited to short time periods (no more than 1 minute). Thus, if the same IP address is sending packets to one of our honeypots on the 13th of March and then on the 28th of March, we consider that they come from two distinct *attack sources.*
- *Ports sequences:* attack sources send packets to specific ports of the honeypots. A *Ports Sequence* defines the specific order according to which ports have been targeted on a given honeypot. For instance, if source A sends requests on port 80, and then on ports 8080 and 1080, the associated *ports sequence* will be {80; 8080; 1080}.

In the following, we report the results collected from March 1 until December 31, 2003. During that 10 months period, we have observed a total number of 3028316 incoming packets, sent by 18075 different sources. The attack sources rate is quite constant over the months as illustrated in Figure 1 where we have represented the number of observed sources per month. The slight increase observed in November and in December is due to the worm MBlaster, as explained in Section 5.3.2.

[2] A switch in the Vmware jargon but it actually behaves as a hub.

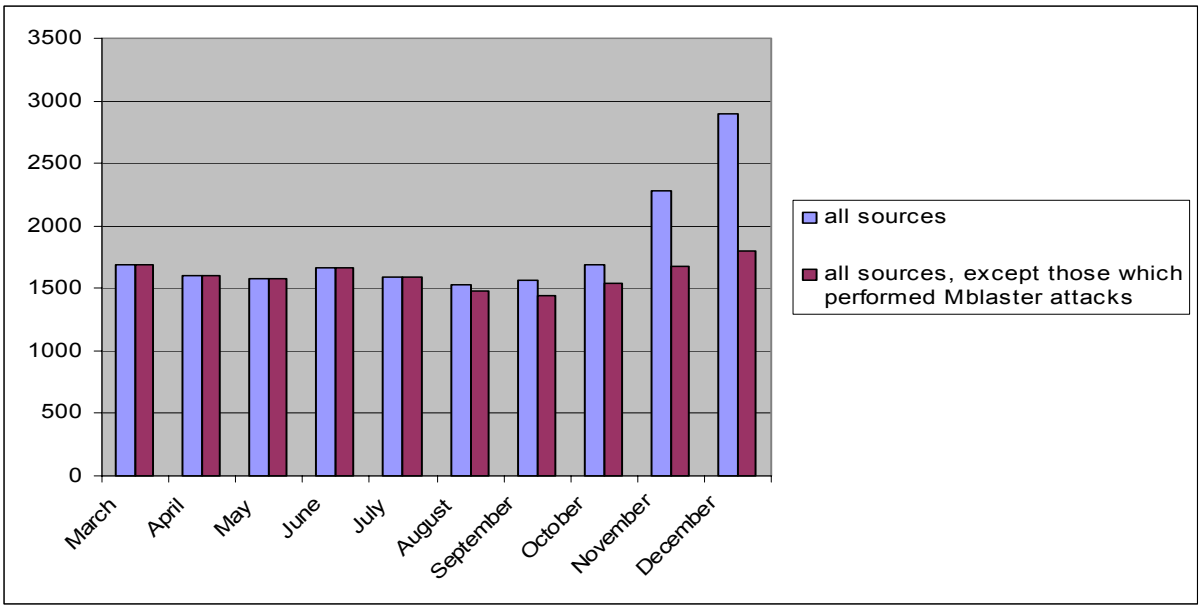


Figure 1: Number of attack sources per month.

We observe some changes with the results presented in [18]. First, the vast majority of the observed packets are TCP ones (84.7%). Others are ICMP (1.6%) and UDP packets (13.7%). In [18], almost all observed packets were TCP ones (97.9%). The difference is due to the increase of attacks to the NetBios Name Service (more precisely attacks to UDP port 137 associated with attacks on TCP port 139) at the end of 2003 (see Section 5.3.1).

However, we confirm that attacks are directed to a very specific number of ports, 188 in total. In 69% of the cases, an attacking source has sent requests to the three honeypots in a very short period of time (typically in a few seconds). In only 5.5%, attack sources contact only two out of the three machines. However, and this is something important we discuss later, in 25.5% they have focused on only one of the three honeypots. Interestingly enough also, we noted in [18] that no IP address did seem to come back: none had been observed during more than one day. This property is confirmed with a few exceptions though: 81 IP addresses have been seen many times (one of them has been observed in 14 different days) from August to December. They all seem to be taking part to the same attack process described in Section 5.3: they periodically test a known vulnerability on ports 135 (Microsoft RPC end-point mapper) and 139 (NetBIOS File and Print Sharing).

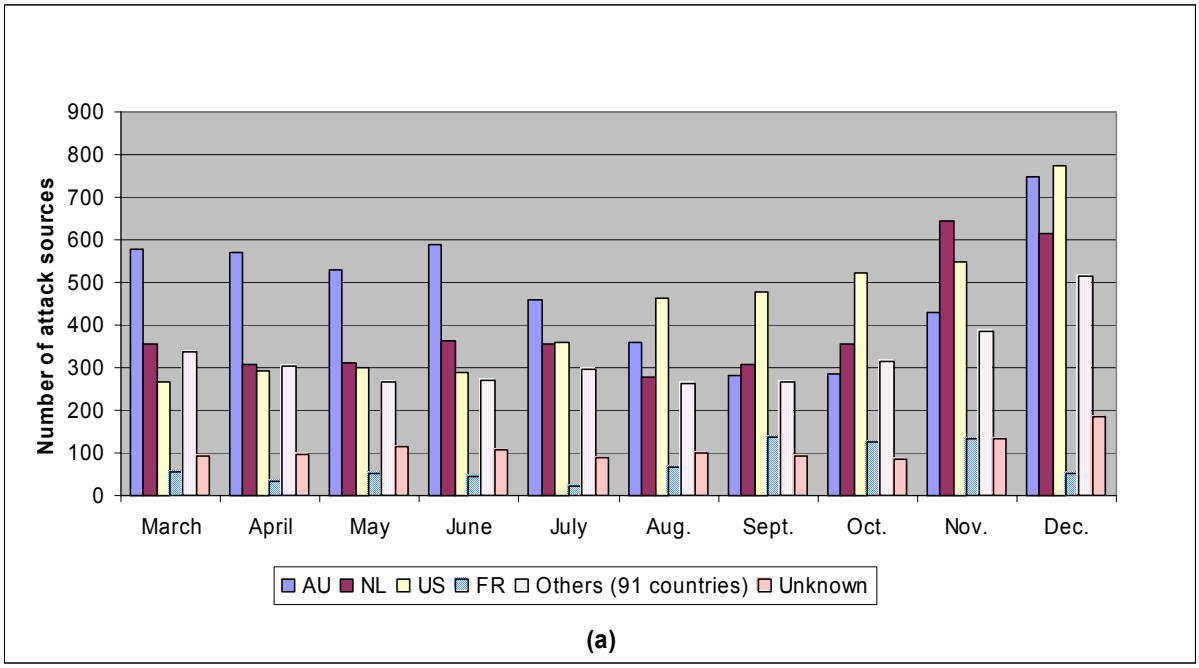
There are many things that could be said with the data we have collected so far, but in the context of this paper, we want to show two things. First, it is indeed possible to learn something about the attack processes and the threats we are facing. The data highlights the existence of some stable processes that could and should be modeled. Second, the observations we have made show the need for designing a more global set up, to answer questions that are left open.

Our results are divided into three main categories that characterize i) the attacking machines ii) the attacked machines iii) the attacked ports.

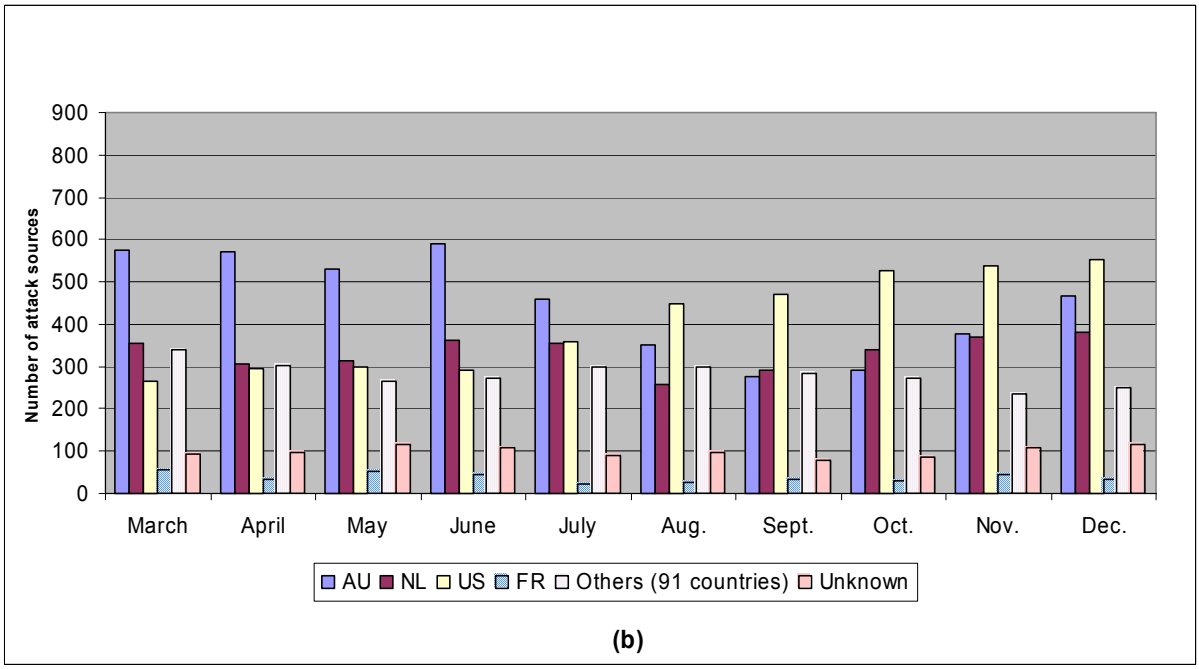
5.2 Information on attacking machines

5.2.1 Geographical location

Geographical location tells us where attacks are coming from. It is obtained thanks to the Netgeo utility [48], developed in the context of the CAIDA project[3]. This is a database and a collection of sophisticated perl scripts that map IP addresses and AS numbers to geographical locations. The result of running the Netgeo script on our logs is represented in Figures 2 a, b and c.



(a)



(b)

[3] It seems that Netgeo is hedging some results and favors the Netherlands as European default country in some undetermined cases. We intend to use MaxMind GeoIP, a commercial utility very similar to NetGeo in order to validate this observation.[42].

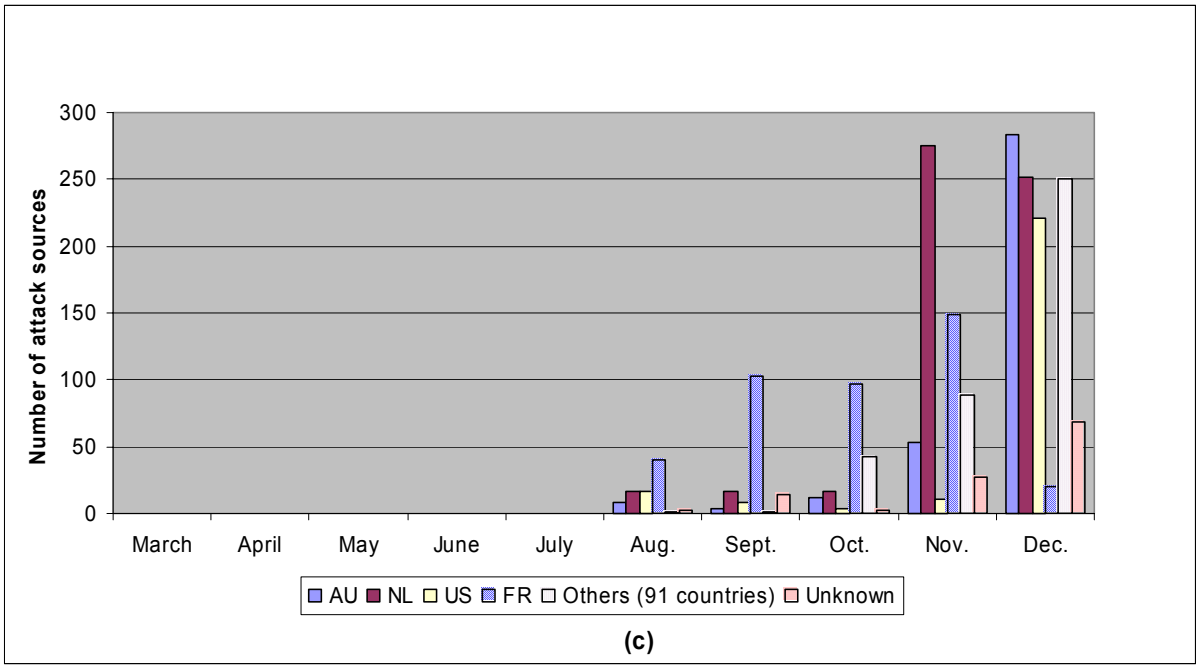


Figure 2 a) b) c): % of IP Sources per geographical location over 10 months: Figure a) for all sources – Figure b) for all sources but those associated to variants of MBlaster – Figure c) for sources associated with MBlaster only.

Figure 2 a) represents the number of attack sources from different geographical locations per month. Surprisingly enough, 71% of the attacks originate from only three countries: Australia (26.6%), the USA (22.7%) and the Netherlands (22%), whereas more than 90 other countries are also represented. These three countries are definitely not the usual suspects that would be quoted by security experts [23]. Furthermore, we would like to point out the increasing part of attacks coming from France and Netherlands since September. The main reason lies in the increasing MBlaster traffic. Figure 2 b) is similar to Figure 2 a), where we have removed sources associated to MBlaster (all variants). Figure 2c) instead, represents only those sources associated to MBlaster. These figures confirm that the majority of the MBlaster traffic that we observe is coming from France and the Netherlands. However, it does not totally explain the decrease of attacks from Australia. A closer look at the data indicates that this phenomenon is due to the slow down of the traffic to ports 80 (CodeRed/Nimda/Nachi variants [13]) and 445 (Win32.Randon [76]). Last but not least, the increase of US attacks cannot be linked to any specific ports sequence. It still requires a deeper analysis at this stage. Finally, we are really surprised by the regularity of these phenomena over the months. It seems to indicate the existence of some very stable processes, but more data from other honeypots are necessary to check if this phenomenon is, or not, a local artifact.

5.2.2 Operating System of the attacker

We have used two utilities, respectively Disco ([2, 21]) and p0f ([55]) to passively fingerprint TCP packets[4]. They both have similar characteristics but give slightly different results: all non-determined OSs from Disco are classified as Windows by p0f. Our results are presented in Figure 3. With no real surprise, we find out that the majority of attacks originate from Windows machines.

[4] Active fingerprinting techniques such as Nmap [51], Quezo [56], or Xprobe [77] have not been considered to minimize the risk of alerting the attacker of our investigations.

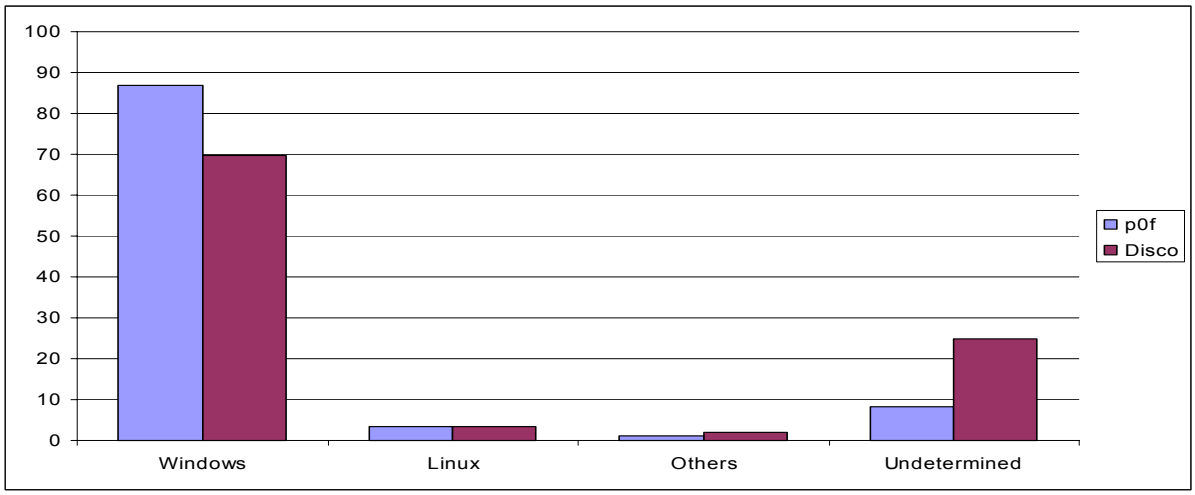


Figure 3: % of diff. IP per OS type with Disco and p0f

5.2.3 Timing of the attacks

We have shown so far that the majority of the attackers were coming from three countries and that this phenomenon was constant over the ten month observation period. Moreover, passive fingerprinting confirms that attacks originate mostly from Windows machines. Additionally, we explain in Section 5.3.1 that attack sequences are limited and very repetitive. Thus, it seems to confirm the classical assumption which claims that the surge in attacks is due to compromised personal computers running automated robots. To validate this claim, we have represented in Figure 4 the percentage of IP addresses observed per hour per country (time is the local time of the attacking source) For instance, point A in that Figure indicates that 4.5% of the attacks coming from Australia occurred between 4 a.m. and 5 a.m. (Australian local time). We have not represented all attacks from the USA, but, instead, only those coming from Virginia and California. These two states account for almost 70% of the American observed attack sources. Each of them is within a single time zone.

There are three important things that are worth being noted in this Figure:

- Attacks are launched on 24x7 basis. This confirms the idea that robots are attacking continuously
- All curves indicate a slight increase of the attack during the late afternoon and in the evening. This is very clear for the attacks originating from France. This indicates that, on top of the attacks running during the whole day, there is another set of attacks that is somehow linked with the activity of human beings.
- The curves of the attacks originating from Virginia and California have a very weird, but similar, shape. They follow the same trends described here after but they are less regular. More interestingly, they seem to move from high to low values at the same rate. So far, we have no good explanation for this unexpected similarity between these curves. It might just be a coincidence or, at the contrary, the expression of a specific attack process running at well defined time intervals.

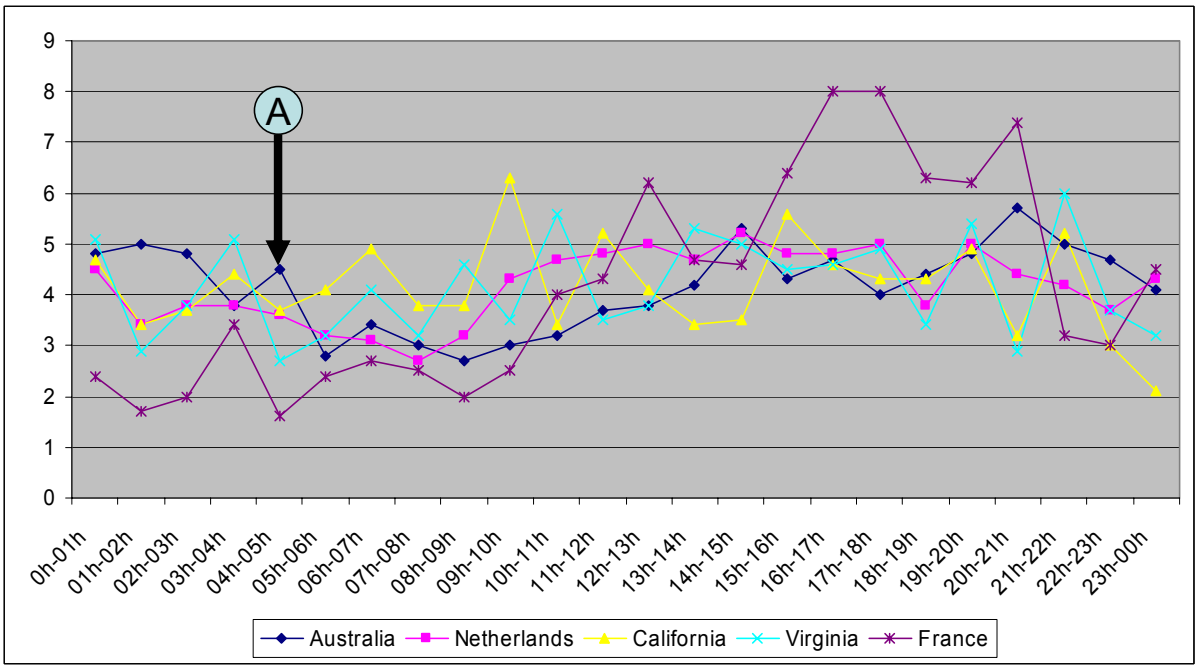


Figure 4: % of IP sources geography distribution per hour and per location

5.3 Details on targeted ports

5.3.1 Generalities

188 different ports have been probed. Each attacking machine probes one or more targeted ports following a given *ports sequence* (see 5.1). The number of different observed sequences is limited to 470 distinct sequences. Furthermore, we note that i) each sequence is often limited to one port and ii) a given set of ports almost always uniquely identifies a *ports sequence*, as we have very rarely observed two sequences differing only by the order of ports being probed. Of course, two attackers can scan same ports for different purposes. A closer inspection of the data payload indicates that the same *ports sequences* often also correspond to similar sequences of packets: in transport level (flags fields) and in the application layer (similar requests/data). In other words, each sequence could potentially be linked to a small number of available attack tools/robots. Some examples are described in Section 5.4, concerning sequences {135, 4444} and {554}.

Table 1 represents, per month, the top 8 *ports sequences* performed by the attack sources observed during that month. For instance, one can see that in March, 45.5% of the attack sources have sent packets to the sole port 445, while 2.7% have scanned ports 80, 57 and 21 in that precise order. We notice that there is a slow evolution over time. These sequences characterize the activity of around 75% of the attack sources every month. Even if in every month we observe the same set of traditional scanned ports (i.e. 21, 80, 135, 139, 445, and 1433), there are a few other cases, such as ports 443, 554, 4444, 17300 and 27374 that are only present in some specific months. These ports correspond to:

- port 443: https, http protocol over TLS/SSL
- port 554: Real-time streaming Protocol
- port 4444: Kerberos authentication (see Section 5.3.2)
- port 17300: Kuang2 Trojan
- port 27374: Ramen linux Worm - SubSeven Trojan

March	April	May	June
445 (45.5%) 80 (15.9%) 1433 (8.8%) 139 (5.8%) 21 (3.5%) 135 (2.9%) 80, 57, 21 (2.7%) 443 (2.1%) Others (12.8%)	445 (43.5%) 80 (15.4%) 1433 (7.6%) 139 (7.5%) 21 (3.3%) 135 (2.2%) 80, 57, 21 (1.4%) 443 (1.4%) Others (17.7%)	445 (40.1%) 80 (15%) 1433 (8.8%) 139 (7.1%) 21 (3.3%) 135 (3.2%) 80, 57, 21 (2.2%) 139, 445 (2%) Others (18.3%)	445 (30.6%) 80 (15%) 139 (9.9%) 1433 (8.2%) 445,139 (4.4%) 139, 445 (3.7%) 21 (3.3%) 135 (3.2%) Others (21.7%)
July	**August**	**September**	**October**
445 (29.5%) 80 (20.5%) 1433 (9.6%) 139 (8.1%) 21 (3.1%) 139, 445 (2.9%) 135 (2.5%) 443 (1.6%) Others (22.2%)	445 (23.6%) 80 (17%) 139 (11%) 1433 (9.5%) 135 (5.8%) 139, 445 (4.1%) 17300 (3.4%) 21 (3.2%) Others (22.4%)	80 (15.8%) 1433 (12.6%) 445 (12.6%) 139 (11.6%) 135 (7.8%) 554 (5.1%) 139,445 (4.2%) 21 (4.2%) 135, 4444 (3.8%) Others (22.3%)	80 (15.6%) 1433 (11.9%) 139 (10.6%) 135 (9.8%) 445 (8.4%) 27374 (6%) 139,445 (5.2%) 135, 4444 (5.2%) 21 (4.4%) Others (22.9%)
November	**December**		
135 (28.9%) 135, 4444 (13.6%) 80 (9.2%) 1433 (8.7%) 139 (8%) 445 (6.4%) 21 (3.5%) 554 (2.9%) Others (20.8%)	135 (28.4%) 135, 4444 (17.3%) 80 (9.4%) 1433 (8.5%) 139 (8.4%) 445 (5.4%) 139, 445 (2.6%) 21 (2.1%) Others	**Others ≈** 185 distinct sequences each month	

Table 1: % of *ports sequences* per month

Port 17300 was mainly scanned during June, July and August, while scanning phases on ports 554 and 4444 have been virulent since September only. This phenomenon is illustrated in figures 5 and 6 and explained in Section 5.3.2 and 5.3.3.

Moreover, a deeper look at attacks on port 139 reveals that there is an increasing number of attacks to TCP port 139 associated with UDP port 137 (in terms of packet numbers). They are due to an increasing number of scans to netbios services. We are convinced they are linked to the recent published vulnerabilities on those two ports: one from March 2003 (CERT CA-2003-08), and one from September 2003 (CERT CA-2003-23) [10].

To conclude, we were expecting to find peaks of activities against specific ports as a result of the publication of a new attack tool is published in underground mailing lists. As explained here above, this is the case. However, we note from Table 1 that the phenomena can be lost in the noise and take some time to become very visible. Figure 5 shows the evolution of the sequence {135, 4444} over the 10 months. There is a similar evolution with the simple sequence {554}, as illustrated in Figure 6. Both sequences are discussed in the two following sections.

5.3.2 Worm observation

This sequence {135, 4444} can be attributed to the proliferation of the Blaster worm.

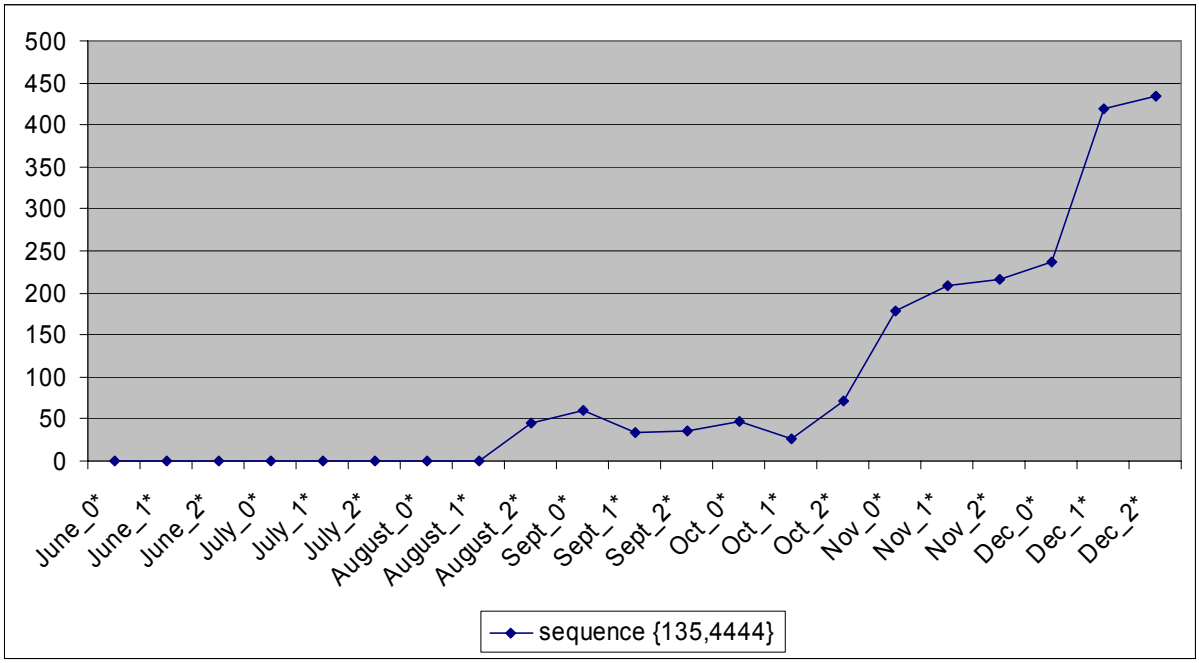


Figure 5: Number of sources scanning ports sequence {135, 4444}

This, by now, famous worm, is named W32.Blaster worm but also "msblast.exe", "Lovscan", "Poza", "Exploit-DcomRpc", etc. TCP port 4444 is normally used for Kerberos authentication and oracle9i communication. A host infected with the W32.Blaster worm opens a command shell on this port, allowing the machine to be remotely controlled. This worm exploits a vulnerability previously disclosed by Microsoft, details of which can be found at [43]. Figure 5 indicates that this worm has been launched on the early days of August 2003 [68]. In our environment, this sequence has only been observed on Mach1. Other machines were never scanned on port 4444, but on port 135 only. This is due to the fact that the worm first scans ports 135 and never goes further if the port is closed. Mach0 is a windows 98 workstation where port 135 is closed. Mach3 is a Linux RedHat server that has by default its port 135 closed. In a more general way, the major part of observations which were published about MBlaster are easily checked and validated in our environment [22, 68]. Finally, we want to point out the gap that exists between public statements on some attacks on one side and concrete network activity on the other side. For instance, Symantec has downgraded the MBlaster worm threat on the 8th of October 2003, *due to a decreased rate of submissions* [68]. However, Figure 5 clearly shows that the epidemic is still expending.

5.3.3 Scanning tools

There is no well-identified worm that produces traffic on port 554. However, a new RTSP scanner has been released on August 2003: this utility, freely available at [57] looks for "Real Server" vulnerabilities (version 7, 8 and 9 on Windows and Linux) to obtain remote shell with root privileges. The tool was publicly released on the 28th of August, while the first scan appeared on the 26th of the same month. This indicates that it has been tested in the wild by its authors before being published.

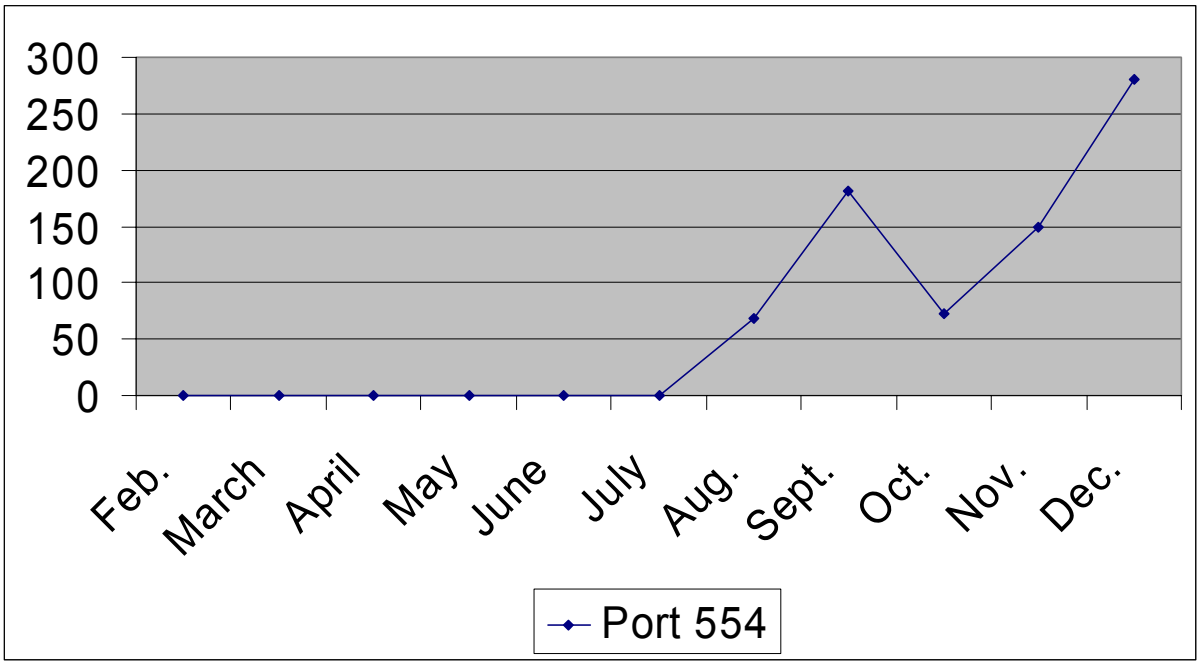


Figure 6: Number of sources scanning ports sequence {554}

5.4 Analysis of targeted machines

5.4.1 Targeted machines: the attacks distribution

We distinguish two kinds of attacking sources: those that have sent packets to all our honeypots, and the others that have only been observed by a subset of our honeypots. We focus in this part on the first set (called '*attacks of type I'* in the following). The next section is dedicated to the other set. First and foremost, whenever a source attacks our three honeypots, we observe that the ports sequence is always the same, and so is the honeypots sequence, namely mach0, mach1 and then mach2. We have never observed any other honeypots sequence during the 10 month period. Furthermore, port sequences are also very limited. To make things shorter, the three honeypots are targeted similarly, independently of their Operating System or offered services. A deeper look at the packets shows that most of sequences have several common characteristic patterns. For instance almost all requests on port 80 observed from July to December 2003 consist in the following request: 'GET /scripts/nsiislog.dll' (98% of http requests). This corresponds to a Microsoft Remote Buffer overflow vulnerability published in June 25th 2003 (CVE CAN-2003-0349) [17]. In the same way, we note that a very few different commands have been issued after a successful login to our ftp server. Those given in Table 2 represent 99% of the ftp connection attempts. Other observed attempts (1% of all ftp attempts) are either hand-written ones (assumption based on the inter-request time interval) or dictionary attacks.

USER	PASS	Other commands
anonymous anonymous@ftp.adobe.com anonymous@ftp.microsoft.com:21 anonymous@ftp.microsoft.com	_gpuser@home.com abc@126.com guest@here.com me@here.com ano@ano.com	TYPE I PORT 80, 180, 24, 202, 7, _ RETR /products/mediaplayer/unix/netshow_su

Table 2: Frequent observed FTP commands

Searches on the underground sites of blackhats communities provide the name of associated tools (Grim's Ping public ftp scanning tool, Roadkil's FTP Probe, and so on). Some specific Unix attacks are applied against all machines, no matter what OS they run. Other tools are issuing ftp commands while login connections have failed. Thus, many characteristics show that we are facing simplistic robots and that their number is rather limited. People using them do not even take the time to change the tool fingerprinting characteristics.

5.4.2 Attacks focusing on one target only

These attacks happen less frequently (25.5%) than attacks of type I. We distinguish two types of attacks: those which are similar to the ones observed on all machines (type I), and those which seem specific to one machine only (type II). Type I attacks are similar to those described in Section 5.4.1 but the difference lies in the scanning phase. In Section 5.4.1, all machines are scanned sequentially while we observe here a different scanning method. Either one or two machines are scanned. Thus, the involved tools have different scanning options or versions but they can be recognized as belonging to type I because of the ports sequence. On the other hand, when looking at attack sources sending new sequences against a single of our honeypots, two different situations coexist:

- The address of the honeypot has been spoofed and is used in an ongoing DDoS attack against a third party.
- A focused attack is being tried on one of our honeypot.

In the first case, the observed packets are so-called "backscatter" packets. They are responses to packets from Denial of Service attacks, in which our IP addresses have been used. These attacks are well-analyzed by Moore et al. in [46]. Figure 7 summarizes the various types of responses (column 'response from victim') that can be sent "against" our honeypots. These packets hit a large variety of ports that are traditionally unused, such as 27374 (TCP RST), 11224 (TCP SYN ACK), 9026 (RST ACK), etc...

Packet sent	Response from victim
TCP SYN (to open port)	TCP SYN/ACK
TCP SYN (to closed port)	TCP RST (ACK)
TCP ACK	TCP RST (ACK)
TCP DATA	TCP RST (ACK)
TCP RST	no response
TCP NULL	TCP RST (ACK)
ICMP ECHO Request	ICMP Echo Reply
ICMP TS Request	ICMP TS Reply
UDP pkt (to open port)	protocol dependent
UDP pkt (to closed port)	ICMP Port Unreach
...	...

Figure 7: Some victim responses to flooding attacks [46]

Type II attacks are quite well identifiable: they correspond to 36% of the sources that send packets to only one of our honeypots. We are more concerned about the other category: attacks which target specific services on our machines. In these situations, the machines have never been seen doing a port scan. On the contrary, and much to our surprise, they are always, systematically, sending requests to ports that are open on the machine they are communicating with. Statistically speaking, it is very unlikely to see this phenomenon. It shows that those machines benefit from the results of the scans performed by other machines. In order to highlight this problem, we have opened four new ports on Mach2 on mid-October: one is a well-known Microsoft service, named MS SQL (port 1433), and others are old Trojans (ports 8998, 28934 and 54321). The results we obtain for port sequence {1433} are given in Figure 8. It represents the number of sources that targeted only port 1433 on that sole machine. None of these IP addresses had been observed before. Other ports give similar results.

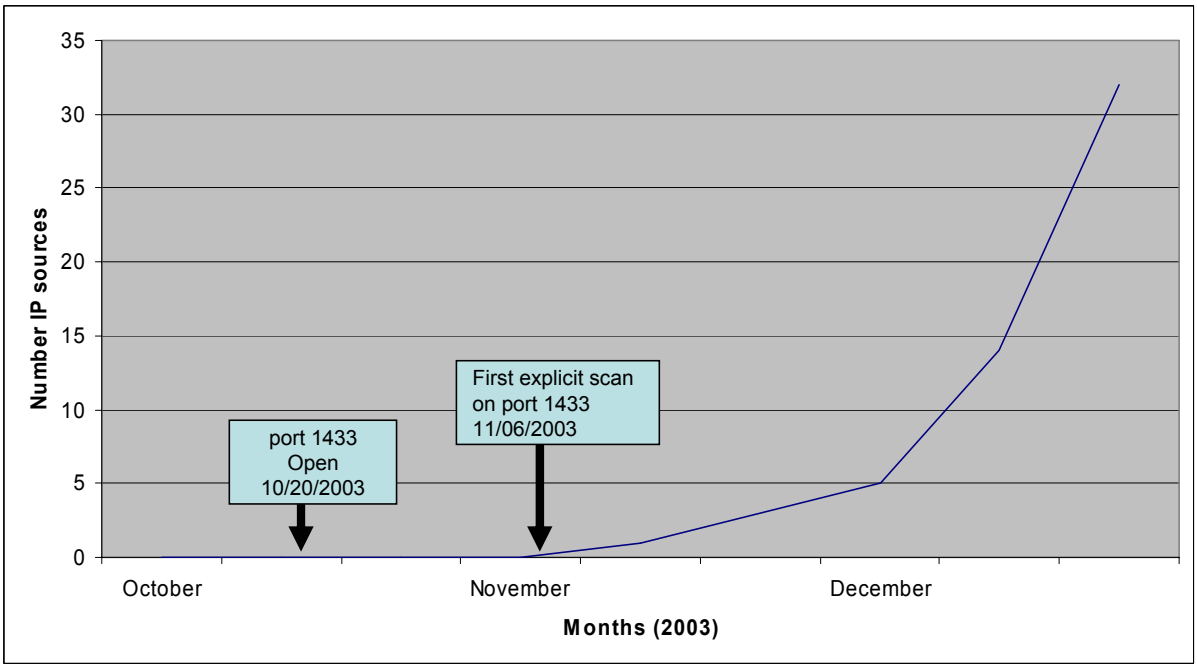


Figure 8: Number of IP sources targeting mach2 only and on port 1433

Figure 8 reveals that it takes 17 days for the first precise attack to happen. Also, it is worth pointing out that port 1433 has been opened on a Linux machine whereas an attack against the service running on that port only exists for Windows machines. Thus, this indicates that scanning machines provide some basic information regarding the opened ports but fail in fingerprinting the OS of the machine they have probed.

6.0 CONCLUSION

In this paper, we have presented data obtained by means of honeypots being attacked over a period of ten months. We have shown the interest a honeypot environment can bring to the network analyst. Due to space limitations, we did not detail to much our analyses but, instead, have focused on the variety of information which can be obtained this way.

We have described the information gathered regarding the attackers and their attack processes. Some results are quite surprising, such as the regularity of the attacking behaviors. We postulate that this information is complementary to what can be obtained through 'Internet telescopes' and vulnerability lists [44, 58].

Our current results have opened issues for further research. Some questions have been left unanswered. The unexpected behavior of machines knowing our environment setup without having probed it and the increase of attacks coming from the USA are two examples of issues that must be clarified. In order to compare local results, new dynamic environments must be designed, not only to find out how long it takes for new information to be collected and shared, but also to figure out if we are facing one or several populations of collaborating attackers.

7.0 BIBLIOGRAPHY

[1] H. AbdelallahElhadj, H. M. Khelalfa and H. M. Kortebi, "An experimental sniffer detector: SnifferWall", *SEcurité des Communications sur Internet Workshop (SECI'02)*, Tunisia, Sept. 2002. http://www.lsv.ens-cachan.fr/~goubault/SECI-02/Final/actes-seci02/pdf/008-Abdelallahelhadj.pdf

[2] O. Arkin, "*ICMP Usage in Scanning - The complete Know-How*", The Sys-Security Group. Version 3.0. June 2001. http://www.sys-security.com

[3] *Back Officer Friendly* home page: http://www.nfr.com/resource/backOfficer.php

[4] *Bait N Switch Honeypot* home page: http://violating.us/projects/baitnswitch/

[5] P. Barford, J. Kline, D. Plonka, A. Ron, "A Signal Analysis of Network Traffic Anomalies", *Internet Measurement Workshop 2002*.; www.icir.org/vern/imw-2002/imw2002-papers/173.pdf

[6] S. Bellovin, "There Be Dragons", *Proc. of the Third Usenix Security Symposium*, Baltimore MD. Sept. 1992. Available on line: http://www.research.att.com/~smb/papers/dragon.pdf

[7] S. M. Bellovin, "Packets Found on an Internet", *Computer Communications Review* 23:3, pp. 26-31, July 1993. Available on line: http://www.research.att.com/~smb/papers/packets.pdf

[8] *Bigeye* home page: http://violating.us/projects/bigeye/

[9] *The sleuth kit V1.62* (previously known as TASK), Brian Carrier, 2003, www.sleuthkit.org/

[10] CERT @Vulnerabilities advisories home page: http://www.cert.org/advisories/

[11] Z. Chen, L. Gao, K. Kwiat, "Modeling the Spread of Active Worms", *IEEE- INFOCOM 2003*, April 2003, San Francisco. Available on line: http://www.ieee-infocom.org/2003/papers/46_03.PDF

[12] B. Cheswick, "An evening with Berferd in which a cracker is lured, endured and studied", *Proc Winter USENIX Conference*, San Francisco, Jan 20, 1992.

[13] Some CodeRed statistics available at: http://kropf.net/coderedstats.html

[14] *Deception Tool Kit*, DTK, Fred Cohen & Associates. http://www.all.net/dtk/dtk.html

[15] F. Cohen, D. Lambert, C. Preston, N. Berry, C. Stewart and E. Thomas, "*A Framework for Deception*", Tech. Report, July 2001, http://all.net/journal/deception/Framework/Framework.html

[16] P. Cracknell, "*The wireless honeypot project: A brief look at how wireless networks are used and misused in the City of London*", RSA Security UK Limited (RSA SUL), CISSP, Tech. Report. http://www.rsasecurity.com/worldwide/downloads/honeypot_report2003.pdf

[17] Common Vulnerabilities and Exposures (CVE) home page: http://www.cve.mitre.org/

[18] M. Dacier, F. Pouget and H. Debar, "Honeypots: Practical Means to Validate Malicious Fault Assumptions", *Proc. Of the 10th Pacific Ream Dependable Computing Conference (PRDC04)*, February 2004.

[19] *Symantec Decoy Server*, enterprisesecurity.symantec.com/products/products.cfm?ProductID=157

[20] H. Debar and D. Lefranc, "Observations on the Internet traffic reaching broadband-connected users", *EICAR Conference*, Copenhagen, Mai 2003.

[21] *The Disco tool* home page: http:// www.altmode.com/disco

[22] eEye Digital Security Research home page: http://www.eeye.com/html/Research/

[23] J. Evers, "*Experts: Most Code Red attacks coming from Asia*", IDG News Service, 2001. Available on line: www.computerworld.com/securitytopics/security/story/0,10801,62730,00.html

[24] FakeAP, par Black Alchemy, home page: http://www.blackalchemy.to/project/fakeap/

[25] Coroner toolkit, D. Farmer, W. Venema, home page: http://www.porcupine.org/forensics/tct.html

[26] S. Grundschober, "*Sniffer Detector Report*", ,Internship Report from IBM Zurich for the Eurecom Institute , June 1998, 50 pages, ref. Eurecom : CE-98/IBM/GRUN - Document number: 1914. Available on line: http://www.eurecom.fr/~nsteam/Papers/grundschober98.ps

[27] S. Grundschober, M. Dacier, "Design and Implementation of a Sniffer Detector", *Recent Advances on Intrusion Detection Workshop (RAID98)*, 1998. www.raid-symposium.org/raid98/

[28] Irish Honeynet Alliance members. Collected data available on line: www.honeynet.ie/results.htm

[29] *Honeypot mailing list*, available at: www.securityfocus.com/popups/forums/honeypots/

[30] "*Know Your Enemy: Statistics Analyzing the past ... predicting the future*", Honeynet Project, July 2001. Available on line: http://project.honeynet.org/papers/stats

[31] "*Know Your Enemy: Learning with User-Mode Linux Building Virtual Honeynets using UML*", Honeynet Project, December 2002. Available on line: http://www.honeynet.org/papers/uml/

[32] "*Know Your Enemy: GenII Honeynets Easier to deploy, harder to detect, safer to maintain*", by the honeynet Project members, June 2003. Available on line: http://project.honeynet.org/papers/gen2/

[33] Honeyweb download page, http://www.var-log.com/files/

[34] *IEEE INFOCOM, April 2003*, San Francisco, www.ieee-infocom.org/2003/technical_programs.htm

[35] *Internet Management Workshop 2001*, home page: http://www.icir.org/vern/imw-2001/

[36] *Internet Management Workshop 2002*, home page: http://www.icir.org/vern/imw-2002/

[37] *Internet Management Conference 2003*, home page: http://www.icir.org/vern/imc-2003/

[38] *Internet Storm Center*, home page: http://isc.incidents.org/

[39] E. Jacksch, "*Tenebris Wireless Honeypot Project: Assessing the threat against wireless access points.1.0*", CISSP, Tenebris Technologies Inc, 2002. www.tenebris.ca/docs/TWHP20021119.pdf

[40] KFSensor, by Keyfocus, home page http://www.keyfocus.net/kfsensor/

[41] Labrea Tarpit Project, http://labrea.sourceforge.net/

[42] MaxMind GeoIP utility home page: http://www.maxmind.com/app/home

[43] Microsoft vulnerability announcement: http://www.microsoft.com/technet/security/bulletin/MS03-026.asp

[44] D. Moore, C. Shannon, K. Claffy, "*Code-Red: a case study on the spread and victims of an Internet worm*", Internet Measurement Workshop 2002, www.icir.org/vern/imw-2002/imw2002-papers/209.ps.gz

[45] D. Moore, G. Voelker et S. Savage. "*Inferring Internet Denial-of-Service Activity*", 2001 USENIX Sec. Symp. www.caida.org/outreach/papers/2001/BackScatter/usenixsecurity01.pdf

[46] *Netbait* home page http://www.netbaitinc.com/

[47] *NetFacade intrusion detection service,* Verizon utility homepage: www22.verizon.com/fns/netsec/fns_netsecurity_netfacade.html

[48] *Netgeo Utility*, available online at http://netgeo.caida.org/perl/netgeo.cgi

[49] *Specter 7.0*, by Netsec, home page: http://www.netsec.ch/

[50] A. Neville, "*IDS Logs in Forensics Investigations: An Analysis of a Compromised Honeypot*", March 2003. Available on line: http://www.securityfocus.com/infocus/1676

[51] "The Art of Port Scanning". *Phrack Magazine* Volume 7, 1997, article 11 (www.nmap.org)

[52] F. Pouget, M. Dacier, H. Debar. "*Honeypots: a comparative survey*", Eurecom Report, RR-03-81, July 2003.

[53] "*Conceptual Model and Architecture of MAFTIA"*, D. Powell et R. Stroud (Editors), MAFTIA Project (IST-1999-11583), Deliverable D21, January 2003; available on line at www.maftia.org.

[54] *Honeyd* Home page, Niels Provos, http://www.citi.umich.edu/u/provos/honeyd/

[55] p0f passive fingerprinting tool home page: http://lcamtuf.coredump.cx/p0f-beta.tgz

[56] *The Quezo tool*, home page: http://www.apostols.org/projectz/queso/

[57] RTSP Scanner available at: http://iperl.homelinux.org/haxor/scanner.pl]

[58] The SANS Security Institute, home page: http://www.sans.org

[59] K. Seifried, "*Honeypotting with VMware – basics*". www.seifried.org/security/ids/20020107-honeypot-vmware-basics.html

[60] *SIGCOMM 2003 Conference*, August 2003, http://www.acm.org/sigcomm/sigcomm2003/

[61] *Single Honeypot*, home page http://sourceforge.net/projects/single-honeypot/

[62] *Smoke Detector*, product home page http://palisadesys.com/products/smokedetector/index.shtml

[63] D. Song, R. Malan and R. Stone, "*a global snapshot of internet worm activity*", November 2001. Technical Report, http://research.arbor.net/downloads/snapshot_worm_activity.pdf.

[64] L. Spitzner, "*Honeypots: Tracking Hackers*", Addislon-Wesley, ISBN from-321-10895-7, 2002.

[65] L. Spitzner, "*Honeytokens: The Other Honeypot*", 2003. www.securityfocus.com/infocus/1713

[66] L. Spitzner, "*Specter: a Commercial Honeypot Solution for Windows*", 2003, http://www.securityfocus.com/infocus/1683

[67] C. Stoll, "Stalking the Wiley Hacker", *Communications of the ACM*, Vol. 31 No 5. May 1988.

[68] Symantec MBlaster worm updates, available at : http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.c.worm.html

[69] *Tcpdump* home page: http://www.tcpdump.org/

[70] *Tiny Honeypot* home page: http://www.alpinista.org/thp/

[71] *User Mode Linux*, UML, home page: http://user-mode-linux.sourceforge.net/

[72] *VMWARE*, User's manual. Version 3.1, home page: http://www.vmware.com

[73] *VSERVER*, home page: http://freshmeat.net/projects/vserver/

[74] "*Warchalking: Collaboratively creating a hobo-language for free wireless networking*", www.warchalking.org/

[75] *WISE*, Wireless Information Security Experiment, http://www.incident-response.org/WISE.htm

[76] W32.randon virus information, available at: http://www.viruslibrary.com/virusinfo/Worm.Win32.Randon.htm

[77] *The Xprobe project*, version 0.0.1 July 13, 2001, http://sourceforge.net/projects/xprobe

[78] "*The museum of broken packet*", M. Zalewski, http://lcamtuf.coredump.cx/mobp/

8.0 ANNEX A

	Interaction Level	Freeware	Emulated Services	Emulated OS	Host OS	Regularly maintained
Bait N Switch Honeypot [4]	Medium	Yes	Malicious traffic is redirected towards non critical targets	None	Linux (on the switch)	Yes
BigEye [8]	Medium	Yes	2 (ftp, http)		Unix	
BOF [3]	Low	No	7 (telnet, ftp, smtp, http, pop3, imap2)		Win 32, Unix	Apparently yes
Decoy Server [19]	High	No	No limit	several	Windows (9x, 2000, NT) Solaris	yes
DTK [14]	Low to Medium	Yes	No limit		Unix	Apparently no
FakeAp [24]	Low	Yes	802.11b	none	Linux	yes
HoneyD [54]	Medium	Yes	No limit	No limit	Unix	yes
HoneyWeb [33]	Medium	Yes	1 (web server)	none	Win32, Unix	Apparently no
KFSensor [40]	Medium	No	No limit	none	Win32	yes
Labrea [41]	Low to Medium	Yes	None	none	Win32s, Linux	no
Netbait [46]	Medium	No	No limit	several		yes
NetFacade [47]	High	No	13	8	Solaris	yes
Single Honeypot [61]	Low	Yes	Smtp, pop3		Linux, Freebsd, Solaris	no
Smoke Detector [62]	High	No	22	9	Windows 2000	
Specter [49], [66]	High	No	14	13	Windows NT, 2000, XP	yes
Tiny HoneyPot [70]	Medium	No			Linux	

Development of Honeypot System Emulating Functions of Database Server

Antanas Čenys[a], Darius Rainys[a,b], Lukas Radvilavičius[c] & Andrej Bielko[b]
[a]Information System Laboratory
Semiconductor Physics Institute
A.Goštauto 11, Vilnius Lithuania
[b]UAB "BlueBridge"
J.Jasinskio 16, Vilnius, Lithuania
[c]Vilnius Gediminas Technical University
Saulėtekio 11, Vilnius Lithuania

cenys@uj.pfi.lt/darius@mokslas.lt/nso@xxx.lt/andrej.bielko@bluebridge.lt

SUMMARY

Possibilities to develop the honeypot type intrusion detection systems (IDS) for databases are discussed. Two types of concept honeypot systems are suggested. Network level system is based on the emulation of the database connections and is aimed to detect intruders searching for database servers and attempting to read basic database listeners information. Honeypot type database level IDS module is aimed to react to enquiries of database tables not used by real applications but loaded to the database to attract intruders.

1.0 INTRODUCTION

Importance of network security is rapidly growing all around the world and Lithuania is not an exception. Public awareness, however, is mainly concentrated on the damage related with the spread viruses and worms, or loss of productivity due to spam. Up to now it were no highly publicized incidents of other type in Lithuania. Public organizations responsible for the information security and computer security community are, however, well aware that situation is not that simple. Reliable data statistical data on illegal activities in computer network is very important both for the analysis and as a tool to raise public awareness.

Intrusion detection systems (IDS) are the main tool to detect illegal activities in the network. Most of attacks to computer networks are based on known vulnerabilities and methods. As a result malicious software can be detected from known "signatures" and illegal intrusion can detected recording untypical or suspicious activities in the system. In our days, however, hackers improve their tools, methods and skills very fast. Nowadays skilled hacker can compromise the system without owner even noticing that, since all log files or even server itself can be modified. As a result IDS should adapt to the new threats very fast. One of very promising methods receiving wide attention recently is deployment of the traps to intruders named honeypots. The main idea of the method was introduced at the beginning of 90's [Stoll 1990, Cheswick 1991]. The honeypot is a service or a system in the network without any real use. According to L. Spitzner a *honeypot is an information system resource whose value lies in unauthorized of illicit use of that recourse* [Spitzner 2003]. The mains goal of honeypot systems is to detect the illegal activities and to gather information about intruder's methods, tools, and habits remaining invisible as well as to protect organization's productive servers. According to these goals two types of honeypot systems can be distinguished: *production* honeypots and *research* honeypots. The purpose of production honeypot systems is to minimize the danger of an intrusion into organization's servers. The research honeypot systems are used for the information gathering.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Honeypot technology is very well suited for the security situation monitoring due to very simple analysis of collected data and absence of false alarms. To test the advantages of the honeypot technology we have deployed research purpose high-interaction honeypot in the external segment of the local network. Using the system we were monitoring security situation in the computer network during two months. Significant number of scans, probes and attempts to compromise the system was detected. Most of the attempts were simple scans, so called "script kiddies", trying to hack the system using automated tools. Rate of attacks was more or less flat during a day as well as during a week. One of the most popular attempts was to implement "*code red*" virus and other worms. Collected statistical data presented elsewhere [Rainys 2003] allows the conclusion that most of attacks come from viruses, scanning activities viruses, and low level hackers. Serious and really dangerous attacks occurred not very often.
High interaction honeypots, in the external segment is a very useful to detect and to analyze the external threats. Most of security incidents including most dangerous ones, however, arise from the inside, i.e. organisation's full time employees, temporary workers as well as the workers from the services providers. Moreover according to the expert of the network security Bruce Schneier the only one incident deserving definitely to be defined as a cyber-terrorism attack in Australia was coming from the person with the deep inside knowledge [Schneier 2000]. High interaction honeypots inside the local area network can provide very interesting information about the local activity but such deployment adds an additional risk. Most of the network and servers IDS's are also oriented to detect the security incidents originating from the external network. They are also mainly oriented towards standard TCP/IP services: e-mail, WWW, FTP servers, SSH and other distant connection tools. Most valuable classified information, however, usually is stored in the databases of the organization and security of the database server is crucial.

2.0 ACTIVE TOOLS FOR DATABASE SECURITY

Access control in the database level is first and very important line of defense. Very often databases design and programming faults leads to the situation when the user without special authorization can get access to the confidential information which should not be accessible for this particular user. Most of the databases operate using client/server scheme. Applications residing on the remote servers connect to the database server using standard network protocols as TCP/IP, IPX/SPX, etc and special database server's protocol for connections functioning at the higher level. Often not only organization's applications servers but also workstations use direct connections to the database server. As a result it is very complicated or even impossible to ensure network segmentation minimizing possibilities to connect to the database server without limiting functional capabilities of the whole system. Due to this the measures to ensure database security should take into account also threats originating from local as well as external networks.

Network segmentation, network traffic filtration, restrictions of database tables access are passive tools of security. These tools often fail to detect unauthorized access attempts early. To achieve the higher level of security enabling to detect intrusion at the initial stage and to identify the intruder the active security tools such as database IDS and honeypots are to be developed and deployed. Two types of the active database security tools can be introduced depending on the level at which they are operating:

- Honeypot system emulating database server functions at the network level.

- Honeypot type database level IDS module. The aim of such module is to detect any suspicious activities of any user and to respond by calling specifically defined external procedures.

The two types of database IDS systems are aimed at detection of the different type of intruders. In the first case of network level database IDS the intruder would be detected during the search for database servers in the organization's network and attempts to read basic database listeners information such as database server version, databases names, etc.

Database level IDS module is aimed to detect the intruders who already have user name and password for access to the database. The module is aimed to react to enquires of database tables not used by real applications but loaded to the database to attract intruders.

In both cases realization of the active database security tools should include the following subsystems:

- Analysis subsystem. This is subsystem responsible for the detection of any illegal activities in the database.

- Reaction subsystem. This subsystem is responsible for all actions undertaken after the detection of such activities. The action can include reporting to the network operating or central IDS system via SNMP traps, etc, or to security administrator via e-mail, pager, etc.

Below we analyze the simplest ways to develop the active type tools to be used with the one of the most widespread database systems namely Oracle database.

3.0 DATABASE EMULATION IN NETWORK LEVEL

An effective tool to monitor illegal activities would be a honeypot type system emulating database functions at the network level. A system of this type is oriented for monitoring of the external attacks. One possibility to develop such system is to create artificial Oracle database “Listener” enabling detection of all unauthorized activities and defining IP addresses and user names used in these connection attempts. As a response the system administrator can be informed or further connections from particular IP addresses can be denied automatically.

The system should include two subsystems:

- Subsystem imitating connections to database

- Subsystem responsible for monitoring and response.

The first subsystem is the most important and can be created using various means. The direct way is to write the software module emulating real connections according to particular database protocol. However this way can be very time and recourses consuming if the protocol is complicated or proprietary and/or encoded. A simple way around this problem is to use freely available software tools developed for protocol testing and emulation. In this case emulation is not completely corresponding to the original protocol, but it can be sufficient if full scale dialog with the database is not needed and the purpose is only to detect an attempt of connection.

As an example we have exploited the software package SPIKE. Using this package it is possible to create communication rules emulating real connections. The corresponding example script can be of the following form:

```
s_block_start("firstpacket");
s_binary_block_size_halfword_bigendian_variable("firstpacket");
s_int_variable(0x0000,5);
s_binary("01 00 00 00 01 38 01 2c 00 00 08 00");
s_int_variable(0x7fff,5);
s_binary(" 86 0e 00 00 01 00 ");
s_binary_block_size_halfword_bigendian_variable("connectpacket");
s_binary(" 00 3a 00 00");
```

```
s_int_variable(0x0200,5);
s_binary("01 01 00 00 00 00 00 00  00 00 00 00 0b 2c 00 00 ");
s_binary(" 3b fb 00 00 00 00 00 00  00 00");
s_block_start("connectpacket");
s_string_variable("");
s_string("(");
s_string_variable("DESCRIPTION");
s_string_variable("=");
s_string("(");
s_string_variable("CONNECT_DATA");
s_string_variable("=");
s_string("(");
s_string_variable("SID");
s_string_variable("=");
s_string("*");
s_string(")");
s_string_variable("");
……..
```

Having script of this or more sophisticated type the SPIKE component “generic_listen_tep” is to be exploited to create a tool listening for the particular port and acting according to the rules established by the script.

The second subsystems can be realized using simple sniffers running on the same machine. Software packages tepdump, ettercap or others can be exploited for monitoring and recording. The standard tools are to be used for collection and analysis of the recorded data. The simplest way is to inform the system administration via e-mail about all connection attempts. Automated response tools also can be easily created.

4.0 HONEYPOT TYPE IDS MODULE IN DATABASE LEVEL

The simplest case of the honeypot type security tools for databases are honeytokens described by Lance Spitzner [Spitzner 2003b]. Bogus information of any type can be loaded into the database just to attract the attention of the intruders. Another type of the trap for intruders can be table without any real information but with attracting name. The aim of the honeypot type IDS module in the database level is to create a system reacting in real time to any attempt to access such table or tables. The simple way to detect access attempts is to use sniffers with select settings like in the case of network level system. However, in the case of encoded connections this way is impossible and it is necessary to use internal capabilities of the particular database. The realization in this case depends significantly on the particular database and can vary from version to version. As an example below we analyze schematically the possibility to create such system for Oracle database server shown in Fig. 1.

In the Oracle database audit is turned on for writing to the database by adding the following line to the file *init.ora*:

Audit_trail = db

Every time a user attempts anything in the database with the enabled audit the Oracle kernel checks to find out if an audit record should be created or updated (in the case or a session record) and generates the

record in a table owned by the SYS user called AUD$. This table is, by default, located in the SYSTEM tablespace. Writing to this table can cause problems with potential denial of service attacks. If the SYSTEM tablespace fills up, the database will hang.

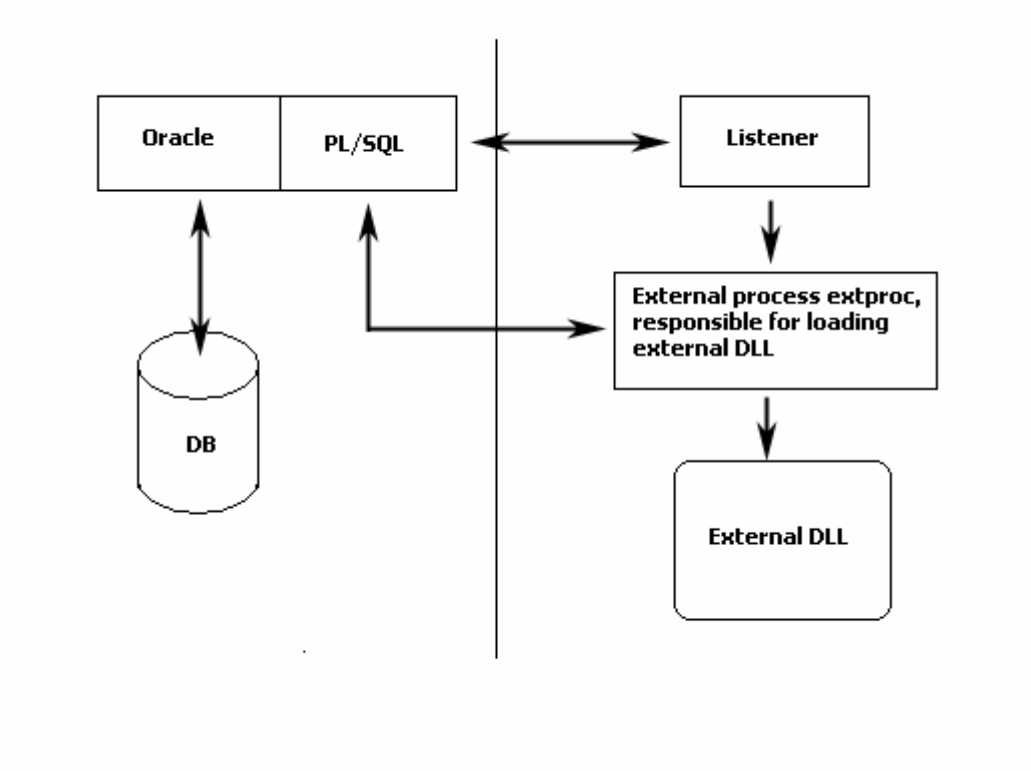


Figure 1: Scheme of the Oracle database server's subsystems

The following command will activate an audit of the access attempts for the table honeypot_table_name:

SQL> audit select on honeypot_table_name;
Audit succeeded.

Audit records can be checked in a different ways: by selecting from SYS.AUD$ where the raw audit trail is recorded or by selecting from dba_audit_trail giving a DBA view of the raw audit trail. So to realize the analysis subsystem for the honeypot type database IDS system PL/SQL code should be written checking the SYS.AUD$ table and analyzing it for the attempts to read data from the honeypot table in the database.

In the recent version 9i of the Oracle database there are possibilities for the fine grained auditing. In this case it is possible to realize honeypot type database IDS system without checking accesses to the honeypot table, but only that providing values of the particular columns from the table:

DBMS_FGA.ADD_POLICY(
object_schema => 'cs',
object_name => 'account',
policy_name => 'chk_account',

audit_condition => 'dept = "CS" ',
audit_column => 'honeypot_column');

Either of the following SQL statements will cause the database to log an audit event record:

SELECT name, honeypot_column FROM cs.account WHERE dept = 'CS';

or

*SELECT * FROM cs.account;*

In order to realize reaction subsystem, it is possible to use an event handler to generate an alert of an administrator that a honeypot system is activated, or to create external procedure saved in DLL file and registered PL/SQL. During the operation PL/SQL loads dynamically DLL-library and calls in external procedures as PL/SQL subprograms.

5.0 CONCLUSIONS

It is demonstrated that honeypot type IDS systems for enhancement of the database security can be developed using freely available software tools. Two suggested concept systems are based on the basic honeypot technology principle to exploit information system recourse without any real value except for attracting and detecting illegal activities. For the databases with large number of records and users it can be a very effective way for early intrusion detection. It should be pointed out, however, that honeypot type tools having many advantages such as simplicity of the concept, ability to detect local threats could not replace other security tools.

6.0 REFERENCES

[Cheswick 1991] B.Cheswick, “An evening with Berferd in which a Cracker is Lured, Endured, and Studied”, http://www.tracking-hackers.com/papers/berferd.pdf , 1991.

[Rainys 2003] D. Rainys, A. Bielko, A. Čenys, **“**Enhancing IDS - Honeypot Systems”, 4th Conference on Informatics and Information Technology, Macedonia: 2003 in press.

[Schneier 2000] B. Schneier, J. Wiley, “Secrets And Lies: Digital Security in a Networked World”, John Wiley & Sons, 2000.

[Spitzner 2003a] L. Spitzner, “Honeypots: Definitions and Value of Honeypots”, http://www.tracking-hackers.com/papers/honeypots.html, 2003.

[Spitzner 2003b] L. Spitzner, “Honeytokens: The Other Honeypot”, http://www.securityfocus.com/infocus/1713, 2003.

[Stoll 1990] C. Stoll, “Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage”, Pocket Books, New York, 1990.

Automated Anti-Virus Deployment

Michel Leonard, Alessandro Berni, Diego Merani
NATO Undersea Research Centre
Viale S. Bartolomeo 400
19138 La Spezia
Italy

netcentric@saclantc.nato.int

SUMMARY

NATO Undersea Research Centre (SACLANTCEN), the research establishment of the Allied Command Transformation (ACT) strongly relies on Network Centric technologies and capabilities to improve the effectiveness of its scientific research. This requires architectures for the interconnection and data sharing that are flexible, scalable, and built on open standards, to ensure transparent interoperability between shore laboratories (both NATO and national) and assets located at sea (research vessels, buoys, autonomous vehicles, sensors and acquisition systems), all connected using a wide range of communications media (e.g. SATCOM, wireless at-hoc networks, acoustical undersea communications).

In addition, to fulfil its mission, SACLANTCEN has an extensive cooperation program with scientists and researchers, consultants and contractors, civil and military personnel coming from several NATO nations. It is a common requirement for them to be connected to the Intranet and to the external Internet.

During at-sea experiments, and whenever a joint research or partnership requires it, computers coming from several external (national and non- national) networks need to be connected to our network, and be able to exchange and share data sets with staff members' computers. External collaborators and visitors also need to keep in contact with their home laboratories or institutes, using the Internet to exchange e-mails or files.

Staff members make large use of Internet to perform their daily duties. The use of Internet is encouraged to distribute information to enhance SACLANTCEN business and visibility; the unclassified network, for this reason, is heavily used to exchange data with external world through Mail, HTTP, FTP transfers. Also the use of Internet is necessary to download security patches for CIS systems, and allows the automatic update of anti-virus signatures.

1.0 THE THREATS

The number of computer viruses in the world follows an exponential growth. From two dozen in 1989, 6000 in 1995, it reaches more than 85.000+ viruses in the wild today. According to tabloid press, 50 percent of all emails could contain a virus in 2014.

In 2003 alone, virus attacks cost global businesses an estimated 55 billion dollars in damages. This represents a substantial increase compared to the roughly 25 and 13 billion recorded respectively in 2002 and 2001.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

The term “malicious code” is used to refer to all code produced with malicious intent, including viruses, Trojan Horses, and worms. Viruses are accounted for the vast majority of malicious codes. Therefore, the term “virus” and “malicious code” will be used interchangeably in this document.

Viruses will not go away anytime soon. A recent CSI/FBI survey indicated that about 98% of respondents have implemented anti-virus software as a security measure. Nevertheless, every day the destructiveness of viruses reach new heights and highlight how primary methods of dealing with viruses today simply aren’t working well enough.

Laptops, wireless devices, file sharing and user behaviour contribute to an ever-expanding set of infection points, leaving companies scrambling to secure their networks. However, history has shown that proactive companies, following a handful of essential security practices fight better against malicious code exploits. These controls include protections such as anti-virus tools, virus walls, specific configuration for routers, email clients, email servers, Web browsers, and security applications that are generally easy to implement, require infrequent updates, and go unnoticed by the average user because of their transparency.

The NATO Undersea Research Centre (NURC) records more than three million high-risk rated attempt of attacks each year. Firewalls are constantly under attack. At least half a dozen of perspicacious and motivated hackers are conscientiously scanning for possible vulnerabilities. However, the biggest threats and fears are concentrated in a few viruses among the 50.000 knocking at the Center doors every year.

Since 2001, the Centre has taken a proactive and adaptive stance in securing its networks. The anti-virus strategy is no longer deployed as stand-alone applications. Rather, it is a layered defence system deployed with other components like host or network-based intrusion detection, global and personal firewalls, logical network security, forensic analyser, etc. By melding these disparate technologies, heterogeneous mechanisms are combined to effectively protect the Centre against new and unknown threats. Even though a few viruses have occasionally entered the NURC network, overall the strategy outlined in this article is until now a success: no payload has ever been successfully activated. The various threats hitting are still transparent to the users’ community.

2.0 THE VULNERABILITIES

Vulnerability is a weakness in processes, administration or technology that can be exploited to compromise IT security. The Vulnerability of our computing architecture has increased with the convergence of several technical factors. The homogeneity of computing hardware, operating systems, application-software and communication platforms is a major enabler for computer viruses, worms and Trojan horses. Together with the increasing connectivity and growing amount of today’s communication systems, it permits viruses to spread faster, and to a larger target audience. The latter are easily built using omnipresent application-specific (e.g. macro languages) or high-level programming languages. Finally, the migration of the PC from the corporation to the home, and the further adoption of home networking have lowered the bar for virus development.

During the last twenty-five years, computer viruses have evolved from a single computer program capable of local infection to complex software worms able to shutdown and damage entire networks. The evolution combines all new technologies. Today, “blended” or “convergence” threats use network viruses, spam as a vehicle, and social engineering as a lure.

Blended threats typically use multiple mechanisms to spread, combining traditional hacker techniques to find operating system or software vulnerabilities with virus-like behaviour to spread further and cause

damages. Blended threats include hacker-like behaviour to automatically probe for and exploit system vulnerabilities. Once the vulnerability is found, the worm can remotely infect the computer and data accessed. It can also be used as a proxy machine for mounting further attacks or even distribute spam emails. Once inside the internal network, it will spread like wildfire.

In the future, viruses will be able to carry multiple intelligent payloads residing only in RAM and firing four or five different actions on a target. A single virus will be capable of entering a network, paralyse or cripple the network infrastructure and its equipment, perform intelligence by monitoring the traffic during the recovery, and "a la carte" alter the confidentiality, integrity or availability of the target architecture.

Social and cultural aspects have also had an impact on present and future vulnerabilities. The Internet is carrying the seeds of a new culture characterized by the universal and instantaneous access to information. Malicious source codes, virus generator kit and vulnerabilities are accessible at anytime, by anybody and from anywhere in the world. With very limited knowledge, it takes only a few minutes to create variants of dangerous viruses and infect thousands of computers. The human factor is the weakest link in the security chain of an organization. Viruses target this vulnerability by using social engineering, which is the art and science of getting people to comply with specific wishes. Apart from security awareness, there is no effective way to protect against such techniques. Indeed, no matter what security measures are implemented, there will always be the possibility of influencing the "human factor" by social or cultural events. The "I love you" virus is one of the most famous examples.

The dependence of defence systems on external or commercial companies creates indirect vulnerabilities. Even if heuristic features (rule-based techniques allowing to discover unknown viruses) give good results (more than 50% detection for unknown binary and more than 80% for unknown macro virus), the detection of a new virus solely depends on the goodwill of anti-virus companies. For example, some companies do not include new viruses in the detection list until a threat level trigger is reached. Repeated apparitions of new virus variants in a very short time often weaken detection and deletion engine updates. For example, on a particular brand, recent events of various MyDoom virus versions have led to CPU Denial of Service. With the exception of the Eicar signature, there is no way to assess the efficiency of an anti-virus. Like for any security system, there is nearly a forced blind trust.

While building or updating security architecture, it will be important to understand the cocktail of techniques that will be used by a virus' creator. "Day zero attacks" (attacks that occur before countermeasures are available) are expected to increase to reach 30 percent in 2006 compared to 15 percent in 2003. A proactive strategy must, therefore, be able to prevent, confine and cure the systems of any virus.

3.0 THE RISKS

The risk to an organization like NATO is the product of threats and vulnerabilities. Risks associated to classified information are obvious. Comparatively, risks to NATO unclassified information and networks are initially less obvious. Threats comes across a large spectrum of diverse activities ranging from non-NATO state sponsored information warfare and espionage, commercial intelligence, investigative journalists, single subject pressure groups (e.g. alter-mondialist, peace organization, whales defenders, anarchists, etc), malicious code writers, hackers in quest of glory and misuses of the organization bandwidth or network equipment for various purposes. Correct setups for servers and firewalls forbidding unnecessary inbound or outbound traffic efficiently reduce most of the threats.

As is the case for commercial entities, the greatest damage to NATO unclassified networks comes from the virus and worm's fraternity. Malicious codes are a direct threat to the core information security

objectives (confidentiality, integrity and availability) of any organization's information assets. Damages can be measured in downtime and costs to recover the situation. The associated costs can be compared to any research facility of a high technology commercial company. However, damages caused by a Trojan horse are difficult to assess.

Generally in NATO, unclassified networks are mostly used to communicate low value information with the external world through the Internet by email, web servers and file transfer. The role of the unclassified network connected for NATO Research Centre is somewhat larger. Unclassified information represents a high percentage all information processed. Internet is used to share information with other research laboratories around the world, perform joint researches or experiments. The content of these networks is of high value for the future of the organization, at least in terms of intellectual property.

The aggregation principle may increase the value of the information stored on unclassified network and thus, the risk. Taken separately, bits of information are unclassified. However, combined together, conceivably it can reach a top-secret level. For example, the target strength of an unidentified submarine is unclassified. However, the value of the information is raised an elevated classification level if sonar frequencies and the class of the submarine can be found somewhere else and correlated.

The beauty of the aggregation principle lies in its silent threat and the power of its consequences. Information can be obtained from independent sources placing the information on unclassified networks or Internet in good faith. This is likely to happen in a scientific environment. "Publish or die" is the motto of most researchers. Indeed, peer revision is a classical measurement of the value of scientific works. A typical unfortunate scenario would include four individuals. A coordinator writing a coordination brief describing the experiment with a new submarine, a researcher publishing its result on the target strength, other publishing results on the sonar technology used and a hacker using a Rootkit Trojan Horse. Taken separately each piece of information collected could, in all good faith, be unclassified high value; together the level is classified. Even if detected, it is nearly impossible to correctly assess the damage if the information is dispersed on different computers.

Even if a high percentage of the information processed by NATO is unclassified, it does not mean that the information is not used to plan, prepare and/or conduct military research and exercises. Although NATO regulations prohibit the direct connection of classified systems to the Internet, there is no policy concerning how and where to treat unclassified information prior to its injection into operational systems, e.g. a tactical decision aid system. By silently altering the integrity of data sets temporary processed or stored on unclassified networks, whether voluntary or involuntary, malicious codes could have the potential to place future military operations at risk.

4.0 THE CONTEXT OF THE NATO UNDERSEA RESEARCH CENTRE.

The NATO Undersea Research Centre, located in La Spezia, Italy, is dedicated to fulfilling NATO's Operational Requirements in undersea warfare science and technology. Its Scientific Programme of Work, currently organized along three main thrust areas (Anti-Submarine Warfare, Mine Countermeasures, and Rapid Environmental Assessment) has resulted during the past 40 years in several scientific and technical contributions that are now part of the set of standard capabilities of all NATO navies.

An interdisciplinary team that covers different disciplines, such as acoustics, oceanography, ocean engineering, real-time processing, and signal processing, performs the execution of the Scientific Programme of Work. Over the past years, a continuously increasing focus has been put on Network-Centric technologies and capabilities, which have emerged as essential tools to enable and improve the effectiveness of its scientific research.

The development of Network-Enabled capabilities in support of undersea research requires architectures for the interconnection and data sharing that are flexible, scalable, and built on open standards. This is essential to ensure transparent interoperability between shore laboratories (both NATO and national) and assets located at sea (research vessels, buoys, autonomous vehicles, sensors and acquisition systems). Also, a wide range of communications media needs to be supported (e.g. SATCOM, wireless at-hoc networks, acoustical undersea communications).

The efforts in the development of Network-Enabled concepts are specifically oriented towards the definition of new generations of scientific instruments. The network will be used to improve the transmission of data from sensors to processors, increasing the capabilities of individual instruments. The resulting increase in efficiency will be larger than the sum of the individual instruments efficiencies.

5.0 THE NEED FOR ADAPTIVE NETWORK SECURITY

The unclassified network of the Centre is connected to the Internet, and provides the standard services that are requested to a modern enterprise network: office automation, e-mail, Internet access and workgroup file sharing.

The Centre has an extensive cooperation program with scientists and researchers, consultants and contractors, civilian and military personnel visiting the Centre for periods of limited duration. To enable efficient collaboration, various services are offered on demand to visitors: access to the Internet, file exchange, printing facilities and use of e-mail. These services are provided on land and at sea on both NATO only owned ships (NRV Alliance and CRV Leonardo). The Centre provides connection to the Internet through its own network or part of it as shown in figure 1. The internal network infrastructure has been designed to prototype the development of network-enhanced capabilities.

Guest computers connected to the internal network introduce additional threats. Either downloaded or already active on guest hosts, malicious codes represent a wide range of threats able to use the network infrastructure as an infection vehicle. For example, in a Microsoft-based network, a virus spreading through shared directories can infect all systems in a few minutes. Policies are able to mitigate, partially, the risks. However, when prevention fails it is needed to detect and confine immediately the danger. Identifying and removing the origin of an internal attack will always be challenge.

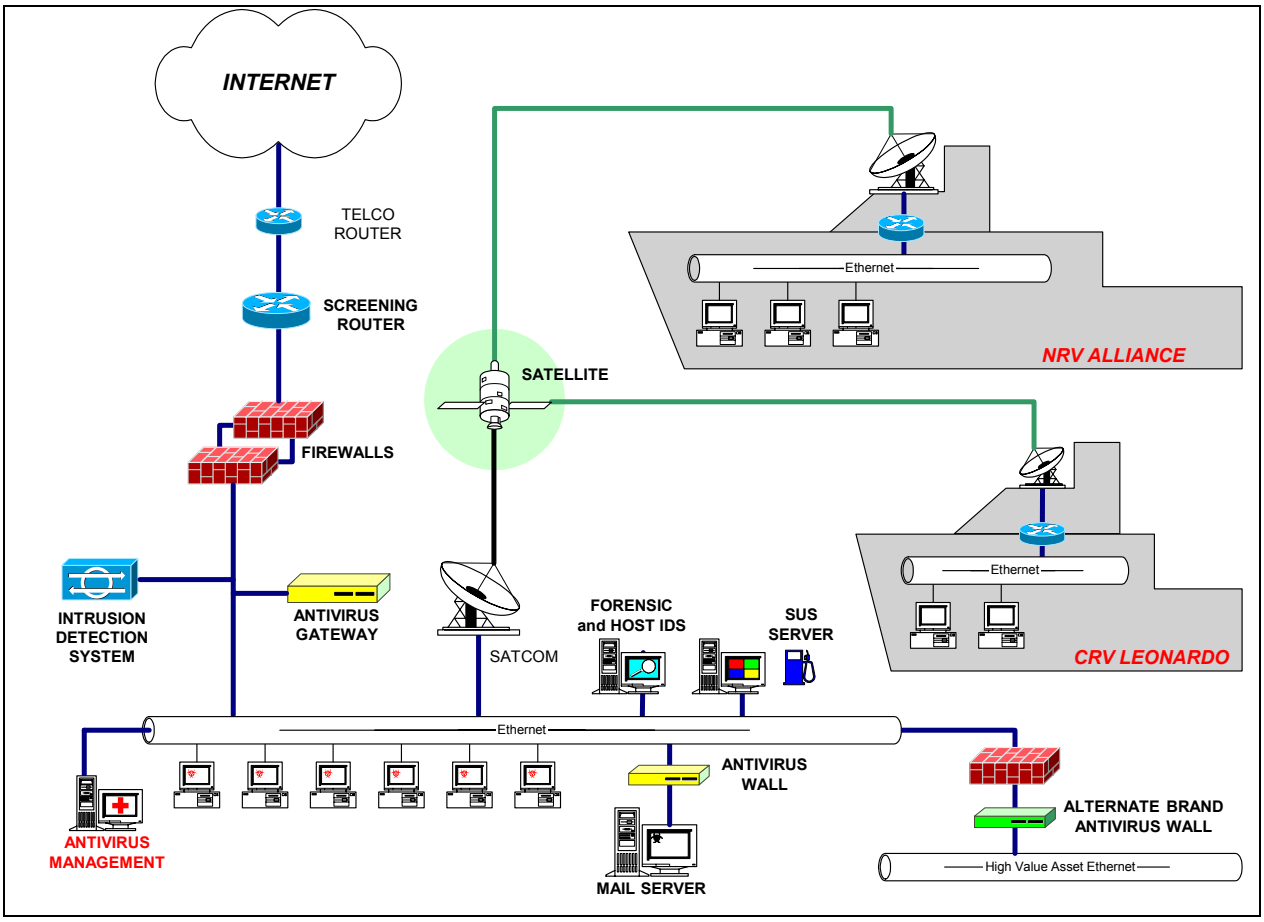


Figure 1 - Standard NURC Unclassified network structure

Malicious code writers have achieved a state of the art where their most complex creations take days or weeks to analyse and develop countermeasures for. However, the average computer virus can be processed in minutes. Virus technologies will continue to co-evolve with the virus technology. The war is in the leap time between the treat and the cure. Unfortunately, threats will always have a step ahead.

The NURC network architecture is modified in a substantial way roughly 30 times in a year to fulfil the requirements of scientific experiments. At each time, complex setups involving satellite communications between multiple ships, inside guest networks, asymmetric networks increase the risk of vulnerabilities from the inside or the outside world. There is a need for continuous vulnerability management. At any time, exposures must be discovered, prioritised, shielded and mitigated. Security controls must be established and enforced. Once root causes are eliminated, Science must go on.

6.0 NURC STRATEGY

The sooner an infection is detected, the lower the cost of eradicating it and the lower the residual costs due to damage and data loss. The most effective enterprise security strategy is preventing attacks by selecting,

developing, deploying and maintaining systems that eliminate or shield vulnerabilities. The centre efforts on prevention reflect clearly this statement. However, in a scientific environment the sinews of the war is to find an acceptable compromise between scientific and security requirements.

6.1 Standard Anti-virus Technologies

Standard Anti-virus technologies are the core of the Centre's automated Anti-virus deployment systems. Its purpose is basically to protect all assets from known malicious attacks by multiple protections in series. It is also designed to keep up with new sets of malicious code risks created by the pervasive adoption and use of web service and active content.

Malicious code writers have achieved a state of the art where their most complex creations take days or weeks to analyse and develop cure for. During the last years, repeated updates in very short times have led several times to defective scanning engines leaving networks without any protection. The Centre's layered defence approach is a direct consequence of these incidents. Each protection is a self-sufficient layer able to fully protect a network. To reach a target, a virus must pass through at least three different protections belonging to at least two different vendors. Beyond redundancy, it turns the competition between the vendors to good account to optimise response times in terms of signature availability, signature delivery and systems reliability.

The Centre's automated Anti-virus deployment is mainly composed of virus-walls (also called anti-virus gateways) and local anti-virus systems.

Anti-virus software is installed on every computer connected to a network. It protects any server or desktop against viruses, vandals, worms, Trojans and other types of malware. Signature-based scanning, heuristic analysis, generic detection, generic decryption and behavioural analysis are available and enforced each time a file is read or written. In combination, these methodologies detect not only all known viruses in the wild, but also malicious Java, activeX and Java Applets, as well as polymorphic viruses, worms, Trojan Horses, Zombie Codes and other security threats.

A central management software solution enables security administrators to manage and enforce anti-virus policies transparently. Virus definition databases are transparently and automatically updated one or two times per day with minimum bandwidth use. It can be configured to enforce updates or upgrades to selectively to one or all of users in order rapidly to stop an outbreak. The standard Centre policy calls for a full scan of all desktops every day.

A Virus-wall or virus-gateway is a dedicated appliance installed at every entry (gateway) of a network. It scans email and Internet content before it reaches the network. Upon live receipt of signatures, it delivers instant gateway security on SMTP, HTTP, FTP, and POP3 traffic. Heuristic searches and generic protection are activated on both inbound and outbound traffic. Once a virus-wall is updated, local anti-virus software are updated with minimum bandwidth and disturbance for users. When a network entry defines the boundary between two internal networks, for example between a low value network and a high value network, another vendor product is used. In addition, for sensible servers, like email servers, are protected with a virus-wall mounted in pass-through (transparent) mode.

The layered defence approach protects gateways, e-mail servers and computers. An inbound virus needs to defeat first the network virus gateway, second pass through the e-mail server virus-wall and third beat all local anti-virus software. Early in March 2004, the first two levels of protection have failed due to vendor's mistakes. Fortunately, the third level protection reacted correctly. For traditional viruses spreading by email, the layered defence system works also very well for virus entering via a media. It first

needs to beat the local anti-virus, second defeat the server virus wall and third pass through the virus gateway to spread to the Internet or an internal high value network.

New viruses are able to bypass gateway level web and email virus protection spreading through other channels. Firewalls are adjusted to reduce additional threats. They are used to secure these ports and minimize the number of ports opened. Any inbound or outbound port outside the scope of a virus-wall is strictly controlled. Authorization to use the port is granted on a case-by-case basis. If granted, it is only for a limited period and additional protection measures are taken to reduce the potential threats. Several additional protections are implemented on firewalls or screening routers (e.g. time-based services) but those fall outside the scope of this article and its classification.

Anti-Spam Appliances detect spam (unsolicited e-mails) at the gateway, preventing it from consuming valuable network resources. Spam is stopped cold through advanced, rules-based scanning and scoring plus multiple levels of intelligent spam detection. Although Anti-Spam Appliances contribute little to the global anti-virus strategy, they reduce potential threats related to malicious codes using Spam as a spreading vehicle.

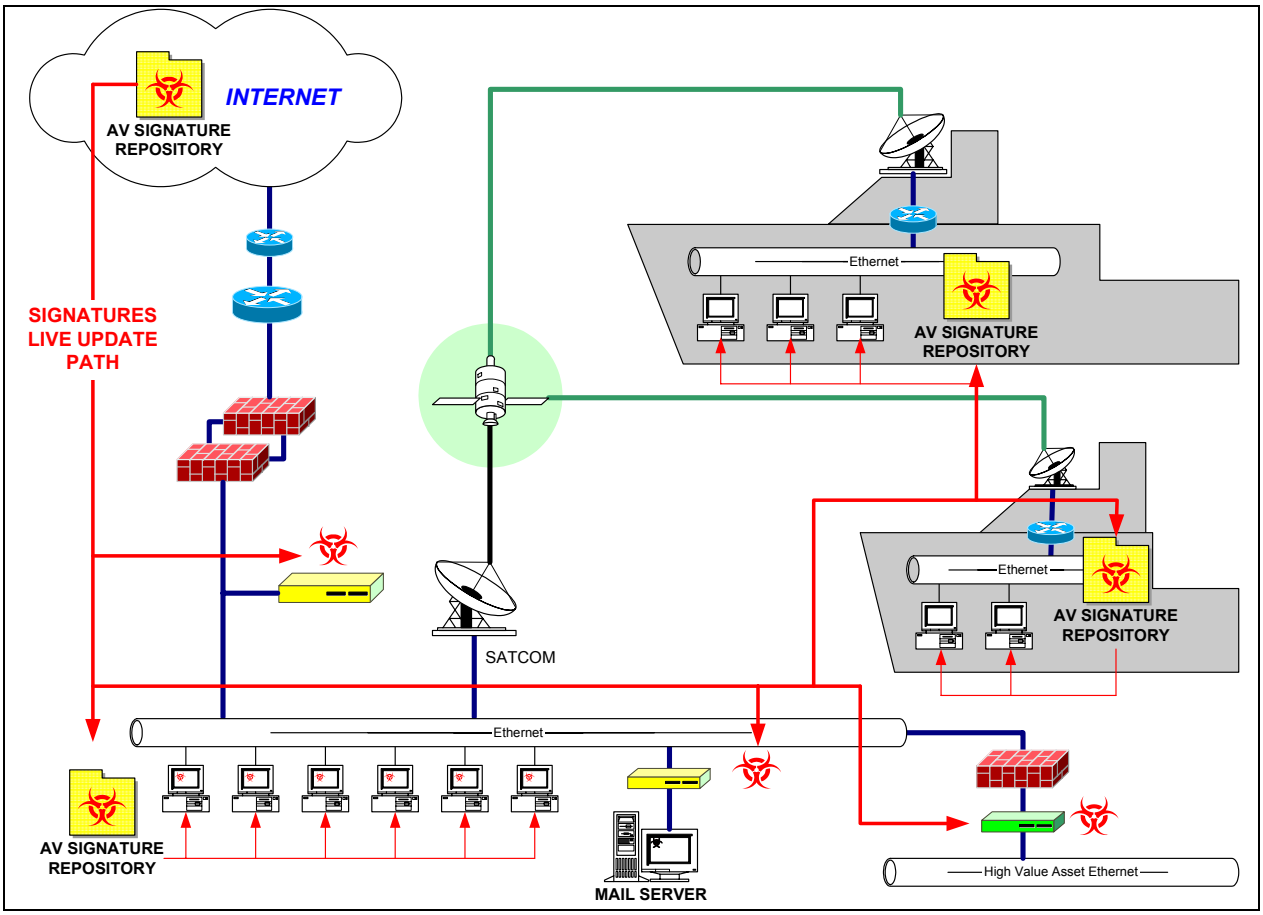


Figure 2 – Automated Anti-Virus deployment (layered defence system).

One of the major weaknesses of this layered defence approach resides in the usage and potential threats created by laptops. Nothing prevents a user with an infected laptop to connect to a network and spread a

virus like a tsunami. However, Network Admission Control is the next technology to be implemented in the Centre (e.g. Cisco Network Admission Control – CNAC). This interesting and a long awaited technology is now emerging slowly on the market. It should automatically limit network access from unprotected assets. It is the link between the anti-virus technology and the network technology. Prior to any connection to a network, the security level of a computer will be systematically assessed and accepted by the network security management system.

In the mean time, a strict security policy is enforced. A policy is as necessary to good information security program as a solid foundation is to a house. Policies, standards, and guidelines form the foundation of any information security program. Recently, all Centre's Communication and Information Systems (CIS) policies have been updated to take into account the technological evolution of information technologies and its threats. A Security Authority must approve every change on networks or systems. Firewalls, anti-virus gateway, Intrusion Detection System (IDS) are mandatory on any entry point of a network. An up-to-date anti-virus is mandatory on every computer. Failure to comply with these regulations can have severe consequences. Security inspections are conducted by Centre's security officers on regularly and randomly bases.

6.2 Complementary Technologies

The previous section showed how the centre strategy is able to block known inbound viruses and known email spreading viruses infecting an internal network. What about blended threats entering the network by other means? The potential number and frequency of malicious-software signature will increase exponentially with the emergence of active content, web services (XML, SOAP, .NET), and peer-to-peer technologies. Together with the increasing complexity, it will contribute to double the number of "day zero" attacks. How to prevent disasters caused by a "day zero" attack? Complementary technologies must be used to better prevent, detect and confine unknown threats.

Prevention against unknown threats is targeted to reduce vulnerabilities. Several technologies can reduce in a substantial ways the vulnerabilities and thus the threats for the centre.

The Centre vulnerability management strategy includes active vulnerability and compliance monitoring, as well as well-defined response processes. Each change in the environment could introduce a new vulnerability, and new external threats are constantly emerging. Therefore, the Centre performs frequent vulnerability scanning to minimize the number of unknown exposures both on firewalls and on desktop platforms.

Most viruses are today targeted to Microsoft applications and operating systems. Therefore, every Microsoft operated computer is updated every night with the latest patches by means of Service Update Server (SUS) receiving live updates directly from Microsoft. In addition, Microsoft security policies are used to thwart malicious codes on high value system by locking down all PC to only allow trusted software to be executed.

Heuristic analysis is only able to discover more than 50 percent of unknown binaries and 80 percent of unknown Macro viruses. Several technologies are needed to confine and detect the rest of the threats. These are particularly required to track potential one-of-a-kind malicious codes targeted to alter the confidentiality or the integrity of NATO information.

Virtual Local Area Networks (VLANs) are used to confine malicious codes or malicious network activities once detected. VLAN's were initially formed to group related users regardless of the physical connections of their hosts to the network. Although switch designers had something other than security in

mind, VLANs make it possible to isolate traffic that share the same switch, or even group of switches. This technique is called partition and is used in the Centre to enforce need to know policies. Each computer must be authorized to enter in a VLAN. In case of problems, a complete VLAN or a single computer can be isolated in a few seconds manually or automatically.

Several Network based Intrusion Detection System (N-IDS) monitors all traffic passing on sensible segments where a detector/sensor is installed, reacting to suspicious anomaly or signature based activity. As well as alerting to an attack, the N-IDS can automatically defend against them. This is achieved by reconfiguring switches/routers to reject from the objectionable sources (shunning) or by crafting some packet to reset the connection. If needed a N-IDS is able to shun traffic using routers' Access Control Lists to isolate malicious activities. Next generation of IDS will be able to shun on virtual interfaces, thus increasing the effectiveness of the actions also inside a VLAN enabled environment. N-IDS Signatures are updated through live updates.

A central Host based Intrusion Detection System (H-IDS) collects all audit trails recorded on each computer. It monitors all event logs for suspicious activities. The H-IDS is the best placed to detect malicious code emerging from the inside of the network as well as those who have infiltrated the network by evading all other methods of detection. The H-IDS includes signature analysis across multiple events and/or time. It also incorporates heuristics analysis to detect unknown malicious behaviours.

Working with IDS is a complex task. Indeed, IDS systems require a substantial amount of attention, training and fine tunings to reduce false (positive) alarms. Therefore, IDS systems are mostly used as passive systems. The centre H-IDS is almost always used as a passive system; it detects a potential security breach, logs the information and signals an alert. The N-IDS is mostly used as a passive system. However, when required it is used as a reactive system that responds to suspicious activities by shunning switches to block network traffic from the suspected malicious source. H-IDS and N-IDS are valuable tools to detect Trojan Horses. For example, the N-IDS is able to detect covert activities like back tunnelling activities.

Last but not least, the most complex tool of the NURC security program is a very complete and powerful forensic analyser. It can be thought of as a pumped-up protocol analyser combined with sophisticated analysis and data-visualization tools. It gathers data about a network, its structure, its traffic and its users by analysing raw network packets. Raw packets are assembled and organized into a knowledgebase and render events into visual representation. The program can combine logs from firewalls, routers, or intrusion-detection systems with saved session information for comprehensive analysis of network activity and enables the examination of real-time suspicious activities. The program contains a collection of advanced tools for creating and examining image representation of network traffic through advanced visualization of logical, physical or topological view of a network. This software is a major asset to detect possible one-of-a-kind malicious code and especially intelligence related Trojan Horses. Unfortunately, this strategy falls outside the scope of this article and its classification.

7.0 CONCLUSION

Since 2001, the NATO Undersea Research Centre has taken a proactive and adaptive stance in securing its networks. The strategy is constantly reviewed and adapted to fight the development of increasingly powerful, more complex computer virus threats. By continuously anticipating the next step in the co-evolution of the virus technology and incorporating countermeasure, this adaptive approach has been, until now, a true success. However, the war is far from being over . .

Acknowledgements

The authors would like thank Mr Jim Obal (SACT INFOSEC) and COW3 David Olson (NURC) for their valuable comments.

8.0 BIBLIOGRAPHY

[1] "Commentary", M. Nicolett, Gartner Group Research Note COM-20-7278 – 3 September 2003.

[2] "Management update: Increase security in Desktop Computing Through Diversity", R. Wagner and J. Pescatore, Article IGG-10152003-03, 15 October 2003.

[3] "Microsoft offers MyDoom Reward", Andrew Stein, posted in money.cnn.com, 30 January 2004

[4] "The Global Threat To information Technology Security", posted in Itsecurity.com, 14 April 2003.

[5] "The Evolving Virus Threat", Carey Nachenberg, Symantec Corporation, posted in Symantec.com.

[6] "Blended threats-How to combat them" – White paper – posted in datafellows.com , F-secure Corporation, December 2003.

[7] "Intrusion Detection Terminologies (part one)", Andy Cuff, posted in securityfocus.com, 3 September 2003

[8] "Intrusion Detection Terminologies (part two)", Andy Cuff, posted in securityfocus.com, 24 September 2003

[9] "Dynamic Virtual LANs for adaptive network security", Diego Merani, Alessandro Berni, Michel Leonard, *in* RTO IST-041/RSY-013 Symposium on "Adaptive Defence in Unclassified Networks", Toulouse, France, April, 2004

[10] Berni A., Leonard M., "Antisubmarine Warfare wireless network for real time data fusion", Proceedings of NATO Regional Conference on Military Communications and Information Systems 2001, Zegrze, Poland, 2001

[11] Berni A., Leonard M., "Antisubmarine Warfare wireless network for real time data fusion", RTO Meeting Proceedings RTO-MP-065 - AC/323(IST-023)TP/12 , Military Communications, Warsaw, Poland, 2001

[12] Leonard, M., Berni, A., Merani, D., "Architectures for Network Centric operations in undersea research" RTO SCI-137 Symposium on "Architecture for Network-Centric Operations", Athens, Greece, 2003

Practical Protection for Public Servers

Joe Spagnolo
NRNS Incorporated
4043 Carling Avenue
Suite 106
Ottawa, Ontario K2K 2A3
Canada

Joe.Spagnolo@nrns.ca

ABSTRACT

The protection of public servers presents a challenge due to their high level of exposure. We present a practical approach to protecting public servers based on experience within a defence research and development network. Although our defence-in-depth approach has proven effective in protecting public servers, we believe the protection posture can be further improved. We outline the areas in which these improvements can be made, and discuss areas such as logging, intrusion detection, event correlation and automated response that we have not yet fully addressed in practice.

INTRODUCTION

The research and development (R&D) agency of the Canadian Department of National Defence, Defence R&D Canada (DRDC), operates an unclassified network called the DREnet. The DREnet supports both R&D initiatives as well as some of the agency's business activities. The DREnet was the first Canadian network connected to the Internet's predecessor – the Arpanet. Although the network is only used to process unclassified information, the protection of the network has always been a top priority for the agency. Early versions of the proprietary DREnet router systems included a packet filtering capability based on Internet Protocol (IP) addresses. After an incident in 1993 in which files were copied covertly from a private DREnet server by Internet based hackers, it became evident that filtering above the network layer was required. The next evolutionary step was to filter well-known protocols by their port numbers since the source of an attack could not be predicted. DRDC adopted strict firewall-like filtering on its border routers in 1993 and then in 1997 deployed commercial firewalls, which were followed later by virtual private network (VPN) encryption and intrusion detection technology.

Although the DREnet would not be considered a large network in this day and age, the DREnet does offer all the challenges associated with the operation of an Internet-connected network. More specifically, the DREnet offers public services that are susceptible to network based attacks since conceivably the associated servers need to communicate directly with any Internet client regardless of the time of day or the physical location of the user. In addition to perimeter protection, DREnet public servers utilize several host-based protection mechanisms and are offered a number of network-based protection mechanisms to shield them from intruders.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

THE CHALLENGE

An organization that connects its unclassified internal network or "Intranet" to public networks such as the Internet exposes its internal systems to numerous threats and risks. Public servers present an even greater challenge since the user of the service initiates the communication at a time and from a location of his choosing. Since the information served by these public servers is unclassified and readily available to the public, the concern is not normally the information's confidentiality but rather its availability and integrity. As such, the server requires adequate protection to minimize the risk of compromise and the server must be monitored to detect unauthorized and malicious activity.

A typical protection posture usually begins with perimeter protection at all external connection points, with firewalls as the systems of choice for protecting an organization's Intranet from unwanted intruders. Firewalls alone, however, do not provide sufficient protection for public servers since firewall policies can be circumvented and firewall inspection engines often do not detect malicious activity that occurs within what appears to be the normal course of the client/server dialogue. As such, the firewall will not prevent a server compromise if the server is executing vulnerable software accessible through permitted traffic flows.

Depending on the intended purpose of an unclassified network, its network operations centre might only be staffed during local business hours (8 hours per day / 5 days per week - 8/5 management), with no on-call support. Public servers however should provide service around the clock 365 days per year in order to satisfy client requests. This presents additional challenges since the public servers would then operate without supervision more than 75% of the time. An attack against a public server that occurs during silent hours would likely only be detected the following workday, which in the case of a holiday weekend could be several days later. If a public server is compromised, it can be used as a launch point to attack other public servers or even internal systems within the Intranet. Information theft is now a possibility since a compromised public server can be used to steal sensitive information from an internal system.

The challenge, then, is to create a defence-in-depth approach to protecting public servers from unauthorized access. Protection can be afforded by network security devices such as firewalls and filtering routers, by properly configured host operating systems, as well as by correctly configured and tested server software. A strategic combination of these protection mechanisms will mitigate the risk of a server compromise and increase the information's availability and integrity. Monitoring and the ability to respond are equally as important as the protection mechanisms since it is imperative that an intrusion be detected and dealt with as soon as possible. If a compromise occurs during the silent hours, a system that permits the infrastructure to react by disabling the public server, containing the intruder, or performing some other type of predefined action would be beneficial.

Technology alone cannot protect a network against attack. The technology must be applied in a sensible fashion by knowledgeable and skilled personnel who are able to assess the latest threats, recognize attacks and adapt the protection posture in order to mitigate the risk of server compromise.

THE DE-MILITARIZED ZONE

Ideally, any public server that communicates directly with Internet clients should reside in an isolated enclave known as a De-Militarized Zone (DMZ). A DMZ provides isolation between the public network and the organization's Intranet. A DMZ can be deployed in many configurations such as in the common architectures shown in Figure 1. An *Attached DMZ* is connected to a third network interface on the firewall, while an *In-Line DMZ* resides between two firewalls. An *Internal DMZ* is not located with the organization's publicly

facing connections but rather resides within the protected Intranet. Like the *Attached DMZ* and the *In-Line DMZ*, a firewall provides isolation between the *Internal DMZ* and the Intranet.

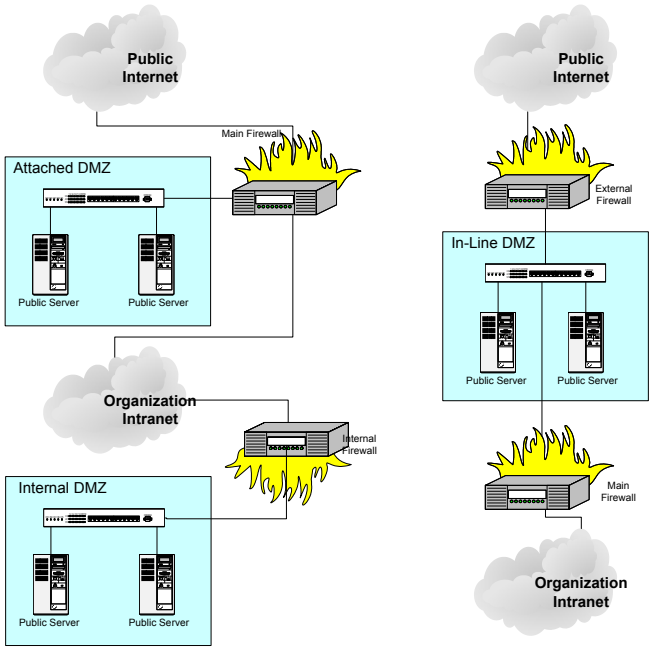


Figure 1 - DMZ Configurations

POLICY

A Security policy dictates the rules and regulations for establishing a public server; i.e. how public servers interact with public networks such as the Internet. A typical policy for the establishment of a public server may include the following statements:

- A public server must be established on dedicated hardware.
- A public server must reside within a DMZ segregated from the public network as well as the Intranet by a firewall.
- Only the communication absolutely required to provide the public service must be permitted by the firewall.
- A public server must execute software free of known vulnerabilities.
- All system and network services that are not absolutely required to provide the public service must be disabled on the public server.
- A public server must be scanned before it is deployed to ensure it is not executing software with known vulnerabilities.
- A public server must be periodically scanned to ensure it is not susceptible to newly discovered vulnerabilities.

- A public server must be monitored to detect attacks as well as compromises.
- A public server must be immediately disabled if it is suspected of compromise.

BASE SYSTEM CONFIGURATION AND VALIDATION

A most important aspect of a public server's security is its own configuration. If the server is comprised of properly configured software that is free of known vulnerabilities, the chance of successful compromise by an attacker is greatly reduced.

It is imperative that the underlying operating system be secured as much as possible. Relevant operating system patches must be applied to the system on a continual basis as made available by vendors. We do not recommend however that all operating system patches should be blindly applied. Some patches do not affect security, but may influence the operation of the application. As the system administrators, we require a good understanding of the system as well as the application in order to determine if patches are relevant or not.

Only the system and network services absolutely required to provide the intended service should remain enabled. The default configuration of both Unix and Windows based systems is likely to include insecure services. File and print sharing services, such as Network File System (NFS) on Unix systems and Microsoft Networking on Windows systems, must be completely disabled since these services are not typically required by public servers. Numerous other services also need to be assessed to determine whether they are required or not. If not required they should be disabled and, if possible, removed from the system. Respectable security conscious organizations have published numerous documents describing how to secure both Unix and Windows based systems [1] [2].

Of equal importance is the application software that implements the intended service. The server software must be free of known vulnerabilities and must be properly configured. For instance, a web server should not permit clients to list and navigate its file system. One must never assume that the default configuration is sensible. Instead, all configurable settings should be reviewed and set to achieve the most secure configuration possible while still meeting the defined business requirements.

Once the server configuration is complete, the public server system must be scanned to identify open ports, review the configuration, and uncover vulnerabilities. The scans must be repeated each time the server software is modified, when the server's configuration is altered, or when a new vulnerability is discovered. From time-to-time, scans must also be done to confirm the server's configuration and account for configuration drift.

PROTECTION MECHANISMS

We employ a defence-in-depth approach to heighten the protection offered to a DREnet public server. This section identifies a number of mechanisms for protecting public servers. While most of the mechanisms can be implemented on both Unix and Microsoft Windows based platforms, some are only available on Unix based platforms.

The sample security policy presented in this paper requires that the public server reside within a DMZ segregated from the public network as well as from the Intranet by a firewall, with the firewall providing the first line of defence. A firewall may also offer protection against certain denial of service attacks that are designed to deplete public server resources through the initiation of partially opened transport layer

connections. A firewall can ensure that connections to public servers are fully established and that data-less connections are subsequently terminated.

Only traffic absolutely required to provide an intended service should be permitted to pass through the firewall. This not only includes traffic between Internet based clients and the public server, but may also include traffic between the public server and a back-end server such as a database server located within the Intranet. All other traffic to/from the public server should be denied and may be logged depending upon its potential and assessed impact. This is known as ingress and egress filtering.

Many firewalls on the market today are based on stateful inspection technology, which offers better performance than application proxy based firewalls. It can be argued however, that stateful inspection firewalls do not provide adequate protection since they do not fully interpret the application layer dialogue and therefore are not able to detect malicious activity at the application layer. Leading stateful inspection firewall vendors have recognized this deficiency and improved their products to monitor the application layer dialogue better and eliminate malicious intent. The latest Checkpoint firewall software with Application Intelligence [3] is currently being looked at to determine its effectiveness in discovering and eradicating malicious traffic. Application layer dialogue can also be monitored with in-line security devices from vendors such as E-Safe [4] and Radware [5] that can detect and purge malicious activity.

A network-based firewall cannot prevent a public server in the DMZ from attacking another public server that also resides in the same DMZ. Virtual local area network (VLAN) mechanisms can segregate public servers located within the same DMZ. Simple port based VLANs configured in a Local Area Network (LAN) switch can prevent communication between public servers that share a DMZ segment. Our configuration includes a distinct VLAN configured for each public server, with each VLAN consisting of only the public server and the firewall. This simple configuration forbids all communication between servers connected to the same LAN switch and therefore prevents the use of a compromised server to attack another server. As with all DREnet firewalls we maintain configuration control of all LAN switches used to implement DMZs regardless of their physical location. If VLANs are incorporated into the protection posture, the associated LAN switches must be configured in a secure manner to prevent tampering with the VLAN configuration.

Each public server should include a host-based firewall to control how it communicates on the network. We employ the packet filtering and firewall capabilities available on most Unix based systems to place tight controls on the server's communication. The host-based firewall policies, which effectively mimic the policies configured in the network-based firewall, protect the public server from attacks launched from other servers co-located within the same DMZ in the event that the segregation provided by the configured VLANs fails. The host-based firewall policies also protect the public server in the event an attacker is able to circumvent the main firewall. On one occasion, Internet intruders discovered a poorly configured Formmail.pl module on a DREnet web server and attempted to send unsolicited bulk email from the web server. The host-based firewall on the server blocked the outgoing Simple Mail Transfer Protocol (SMTP) connections since it was not traffic associated with the delivery of web services. It is interesting to note the effectiveness of the layered protection in this case: the same SMTP connections would have been blocked by the network-based firewall if the host-based firewall had not intervened.

Whenever possible, we do not permit processes that provide network services to execute with system level privileges. If a process with system level privileges is compromised, the attacker inherits the system level privileges, which provide unrestricted access to the system. Unfortunately, most operating systems do not permit non-privileged processes to bind to privileged transport ports [6] in the 1-1023 range, making privileged access necessary to run some applications. Some software supports the circumvention of this

restriction by permitting the server process to accept incoming connection requests on privileged ports, but then passing the "open" socket to a child process that executes with restricted user level privileges. If a process with restricted user level privileges is compromised, the attacker only achieves restricted access to the system. To limit the damage that an intruder can inflict, we grant server processes read-only access to system files and directories as well as to server configuration and content files. For instance, the *httpd* process that provides public web services executes as a non-privileged user named *httpd.* If the *httpd* user is granted read-only access to the web content files, the intruder is not able to modify the content and deface the web site.

Most remotely exploitable vulnerabilities are associated with buffer overflow conditions introduced by careless programmers. Stack-smashing attacks [7] attempt to exploit buffer overflow vulnerabilities by corrupting the processor's execution stack and invoking the attacker supplied code. When supported by the processor (e.g. Sun Sparc), we enable an operating system feature that prevents execution of code on the processor stack. This operating system feature, although not 100% fool-proof [8], foils most buffer overflow related vulnerabilities since it causes the process to incur a segmentation violation before the process can be hijacked by the attacker. Other software based approaches, such as IBM's Stack-Smashing Protector (also known as ProPolice) [9], insert code at compile time that protects applications against stack-smashing attacks. IBM provides instructions on how to build complete Linux and FreeBSD systems with protection against stack-smashing, while OpenBSD now includes the protection directly within its software distribution. Commercial offerings that provide similar stack-smashing protection include the Immunix Secure Linux Operating System [10]. Microsoft is also incorporating code to guard against stack-smashing attacks in its compiler [11] [12] and hopefully stack-smashing protection will form part of future releases of Microsoft operating system software. Whether implemented in hardware or software, stack-smashing protection can not only prevent the exploitation of newly discovered buffer overflow vulnerabilities but it can also alert the system administrator that the server is under attack.

Executing the server process as a non-privileged user can limit the damage an intruder can inflict in the event the server process is compromised. Although the intruder’s inherited access rights may limit her actions, the intruder still possesses a complete view of the file system. Since a non-privileged local user can gain system level privileges through a locally exploitable vulnerability, the intruder’s view of the file system should be minimized. We create a virtual environment for the server process that appears like the real file system but in reality is simply a small subset of the file system. Unix systems include commands such as *chroot* or *jail* that implement a virtual sandbox environment that can be used to imprison the server process as well as the intruder if the server process is compromised. The use of a sandbox is especially attractive if the server process must execute with privileged rights. When the sandbox is created, only the files that are absolutely required to execute the server process are replicated in the sandbox environment. These typically include server binary and configuration files, a small number of required system libraries and a small number of required system configuration files. Consider an exploit for a buffer overflow condition that overflows the server’s buffer with code that attempts to execute a command shell (/bin/sh). If the /bin/sh executable file is not present in the sandbox environment, the exploit fails.

Figure 2 illustrates a simple sandbox environment. The Real File System contains system startup files such as */etc/rc.conf*, system binaries such as */bin/sh* and shared system libraries such as */usr/lib/libc.so* and */usr/lib/libm.so*. The Virtual Sandbox only contains the required system files, system binaries and system libraries in addition to the server binary */bin/server*.

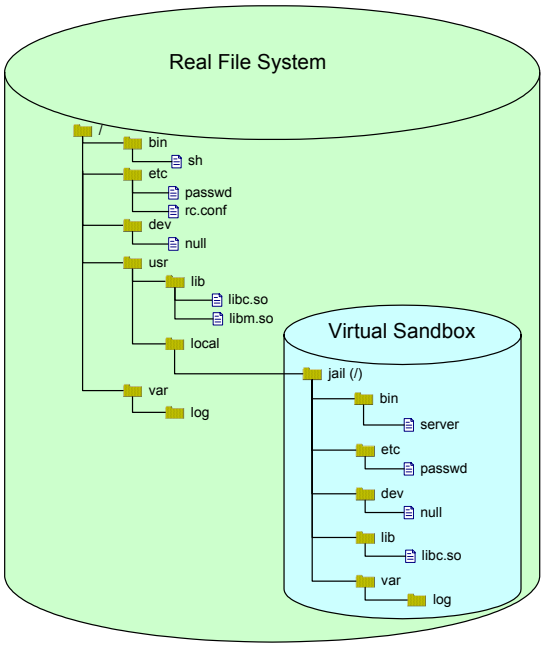


Figure 2 - The Virtual Sandbox

Imprisoning a server process in a sandbox environment mitigates the risk of compromise and reduces the damage the attacker can inflict if the server process is compromised. If the attacker inherits system level privileges and the attacker possesses detailed knowledge of the compromised system, it may still be possible for the attacker to alter the system configuration from within the sandbox environment. We enable kernel level security available on certain Unix variants that would restrict access to certain system operations regardless of the user's privilege level. Examples of system operations that would be restricted include:

- The ability to modify system files whose "immutable" flag is set.
- The ability to load and unload kernel modules.
- The ability to open raw disk devices and kernel memory devices for writing.
- The ability to alter packet filter and firewall rules.

Since the kernel security level cannot be lowered at run-time, the attacker must alter the contents of a system startup file and force the system to reboot. Recall however that an intruder trapped within a sandbox environment does not have access to the real system configuration files such as the system startup file.

REPLICATION OF CONTENT

The security mechanisms discussed in this paper provide protection against server compromise and restrict the attacker's activities in the event the server process is compromised. As an added precaution, a public server should not possess the master copy of the information it serves. If the public server possesses only a secondary copy of the content, the attacker is not able to alter the master copy of the information. We are contemplating the use of file/content synchronization software to eliminate the risk of corruption to the main copy of the content. For instance, a web server's content should be produced and maintained on a separate content server

that resides within the Intranet. Ideally, the content should be pushed from the content server to the public server. In the event a compromise occurs and the integrity of the information served is in question, it will then be possible to restore altered information such as defaced web pages. The Microsoft Content Deployment Service (CDS) [13] facilitates replication for Microsoft web environments; the open source *rsync* software [14] can be used to synchronize web content in Unix web environments; while products such as the RepliWeb Deployment Suite (RDS) [15] may be better suited for heterogeneous environments.

The content server should initiate the replication process. The number of connections initiated by public servers to Intranet destinations must be kept at an absolute minimum. It is imperative that the public server confirms the identity of the initiator using some form of authentication. As discussed in the "Protection Mechanisms" section of this paper, the public server process must be granted read-only access to content files to prevent tampering by intruders. As a result, the replication service on the public server must have the ability to set the ownership and access rights on the content files accordingly.

MONITORING AND INTRUSION DETECTION

Intrusion Detection System (IDS) technology monitors network activity in hopes of identifying attacks and intrusions. Network based Intrusion Detection System (NIDS) sensors are deployed in strategic locations within the network infrastructure to monitor passively network traffic to detect attacks and intrusions. Host based IDS (HIDS) solutions are located on the server itself and monitor system activity.

A NIDS sensor should be deployed within each DMZ to monitor the network activity of public servers. Unlike a NIDS sensor assigned to monitor all traffic entering or leaving the Intranet, a DMZ NIDS sensor can be precisely tuned according to the services provided by the public servers. A NIDS specifically tailored to understand the server's intended use can recognize anomalies in the server's network interaction. For example, if the purpose of a public server is to provide web services, then it should only receive HTTP traffic on TCP port 80 and respond to initiated TCP sessions established via this port. That being the case, the NIDS should be configured to raise an alarm if it detects the public server attempting to communicate on ports other than TCP port 80. The presence of rogue traffic would likely be the result of either a faulty firewall configuration or an incorrect server configuration, or else intrusion attempts and compromises.

A HIDS checks for anomalies in system activity such as unauthorized processes and network services as well as altered system files. Some even check for kernel level instructions that appear malicious. Tripwire [17] in particular is very effective at establishing a system baseline for use in detecting modifications to system files. Once the HIDS establishes a baseline for system files, any attempt by an intruder to create or modify a file is detected and reported by the HIDS. If a service is implemented within a sandbox environment, the HIDS should employ finer-grain monitoring of the sandbox environment. An attacker who gains system level privileges on a public server may attempt to disable the HIDS before the HIDS detects his presence. If the intruder is imprisoned within a sandbox environment, the intruder is unable to access HIDS configuration and baseline files that reside outside the sandbox environment.

If our protection posture includes an automated content update or replication mechanism, we must synchronize its activities with that of the HIDS. Each time the replication mechanism updates the server content, the HIDS must re-establish its baseline. Otherwise, the HIDS will incorrectly interpret newly updated files as a server compromise. Ideally, the replication mechanism should stop the service while it updates the server content and only restart the service after the HIDS has re-established its baseline. This may be difficult to achieve since the HIDS and content replication mechanism usually are not available as a single integrated product or solution.

LOGGING AND CORRELATION

Ideally, all security devices and public servers should log all events to a common repository where event correlation processing can occur. This common log server should be configured in a secure manner using the same protection mechanisms offered to public servers. Intruders generally want to eliminate any evidence of their covert activities. When the logs are stored on a separate dedicated log server, the intruder is forced to compromise an additional system in order to conceal his presence. In any event, the integrity of the log data needs to be preserved and any tampering of the log data needs to be easily detected. *Syslog* replacements for Unix such as *Syslog-ng* [18] and *Modular Syslog* [19] employ cryptography to protect the integrity of the log data.

Hundreds of thousands of individual events may not seem suspicious when examined by the naked eye, but once the information is normalized and correlated, patterns may be discovered which cause attackers and intruders to be revealed. Currently, simple event correlation using locally developed scripts is often employed to identify the top sources and targets of suspicious activity. Although useful in a small environment, such locally developed scripts may not possess the sophistication to detect the activities of skilled attackers whose inconspicuous methods are hidden within hundreds of megabytes of log data. In addition, these scripts generally tend to scale rather poorly as the enterprise grows. Networks might benefit from utilizing enterprise security management products such as Intellitactics' Network Security Manager [21]. The event correlation and data mining capabilities of such products can be used to discover and analyze attack patterns across the entire enterprise. Unfortunately, enterprise class products typically demand an enterprise level price tag, which may not be easily justifiable in a lot of environments. Organizations should not underestimate the effort and associated cost required to deploy an enterprise security management solution. Although the organization should achieve immediate benefit from an enterprise security management solution, the software must be specifically tailored to reflect the local network environment. Event correlation in a network context is still an immature discipline and the subject of continued research.

STANDBY SERVERS

Once a server is compromised, it must be taken offline and carefully examined to determine the cause of the breach. This process can take hours, if not days, resulting in denial of service. We could restore the server from a recent backup, but the restored files likely would contain the same vulnerability that the intruder previously exploited to gain unauthorized access to the server.

If a prolonged service outage for a public service is not acceptable, a standby server built on a different platform could be activated to assume the duties of the compromised server. Consider the scenario where we establish public web servers on both a Microsoft Internet Information Server (IIS) platform and an Apache server on a Unix platform with both servers serving identical content. The primary public server actively satisfies client requests, while the standby server remains offline – either disconnected from the network or connected to a disabled LAN switch port. If we suspect the primary public server of compromise, we would disconnect it from the network immediately and we would activate the standby server to assume its duties. Of course, the standby server's content may not be up-to-date initially and would have to be synchronized with that of the content server. The compromised server could then be properly examined to determine the cause of the breach and subsequently rebuilt and fully patched so that it may resume its duties.

To date, the availability requirements for DREnet servers have not warranted the cost associated with the establishment and maintenance of separate standby servers.

AUTOMATED RESPONSE

For networks that are afforded only 8/5 management, it might be possible for an intruder to control a compromised server for several days before the breach is discovered. Since, in general, a compromised server should be disabled as soon as possible, we require an automated response mechanism that can detect a server compromise and disable the server during silent hours. Monitoring and intrusion detection technologies may detect when a public server is under attack or a public server is compromised. The response technique could be as simple as reconfiguring the DMZ LAN switch to disable the port that interconnects the comprised server to the DMZ upon receipt of a high severity alert. If a standby server is available, the same response technique could be used to activate the standby server so it can immediately assume the duties of the compromised server.

When designing automated response procedures, the response technique is usually not particularly difficult to implement. Rather the difficulty lies in deciding what should trigger an automated response. Relying on NIDS alerts alone may produce false positives and cause erratic behaviour from the automated response system. A combination of critical NIDS and HIDS alerts may provide sufficient evidence that a server is compromised, especially if the HIDS reports unauthorized processes, altered files or newly created files. As previously discussed in the "Logging and Correlation" section, events from all security devices and public servers should be logged to a common repository where the enterprise security management system can perform event correlation. As a result, the enterprise security management system is best equipped to determine if an automated response should be initiated. Although enterprise security management products can be configured to "react" based on a predefined set of conditions, they require high-quality information in order to identify effectively a successful server compromise. Without a high-degree of certainty, the result of the automated response system could be an unexpected self-imposed denial of service.

SUMMARY

Public servers present a security challenge since the user of the service initiates the communication at a time and from a location of his choosing. Well managed networks currently employ a defence in-depth approach to protect public servers from unauthorized access and this paper outlines additional measures that could be employed, particularly with respect to logging, intrusion detection and automated response. Figure 3 illustrates the target architecture for protecting and monitoring public servers on such a network. Its key features include:

Protection Mechanisms

1. A public server is established on dedicated hardware. It does not share a system with other services, especially services considered important to the organization’s day-to-day operations. The fewer services running on a system the easier it is to manage from an operational and security standpoint.

2. The underlying operating system is secured as much as possible. All security related operating system patches are applied to the system and only the system and network services absolutely required to provide the intended service remain enabled.

3. Before deployment, a public server system is scanned to identify open ports and to uncover any known vulnerabilities. The scans are repeated each time the server software is modified, when the server’s configuration is altered, and when new vulnerabilities are discovered. Scheduled scans are also done from time-to-time to confirm the server’s configuration.

4. Each public server, regardless of its physical location, resides within a DMZ segregated by a network-based firewall.

5. Only traffic absolutely required to provide the public service is permitted by the network-based firewall. All other traffic is dropped and depending upon severity is logged.

6. VLANs provide isolation between public servers that share the same DMZ. The DMZ switch prevents one public server from communicating with another public server.

7. A host-based firewall on the public server provides further protection from servers that share the same DMZ as well as protection from all other sources.

8. Public servers include mechanisms to guard against stack-smashing attacks.

9. Public server processes do not execute with system level privileges in order to minimize the damage that an intruder can inflict in the event the server is compromised.

10. The public server processes do not possess the ability to alter configuration and content files. A public server process is only granted read-only access to these files.

11. Whenever practical, public server processes reside in a sandbox environment to complicate attacker attempts to compromise the server and to minimize the damage that an intruder can inflict in the event the server is compromised.

12. The public servers employ kernel level security to protect critical system resources.

13. The public servers only possess a copy of the content, which is pushed to the public server from a content server that resides in the Intranet.

Monitoring

14. NIDS are configured to monitor the network activity of the public server within the DMZ. The NIDS configuration is specifically tailored to understand the server's normal network behaviour so it can recognize anomalies in the server's network interactions

15. A HIDS is active on each public server to monitor system activity such as unauthorized processes and network services as well as altered system files.

16. Events from all security devices and public servers should be logged to a common repository where event correlation can take place.

Standby Servers

17. To meet high-availability requirements, a standby server built on a platform different from the primary server is deployed to assume the primary server's duties in the event the primary server is compromised.

Automated Response

18. Automated response procedures are employed to disable a compromised public server and to activate the standby server if available. The enterprise security management system is best equipped to determine if an automated response should be initiated.

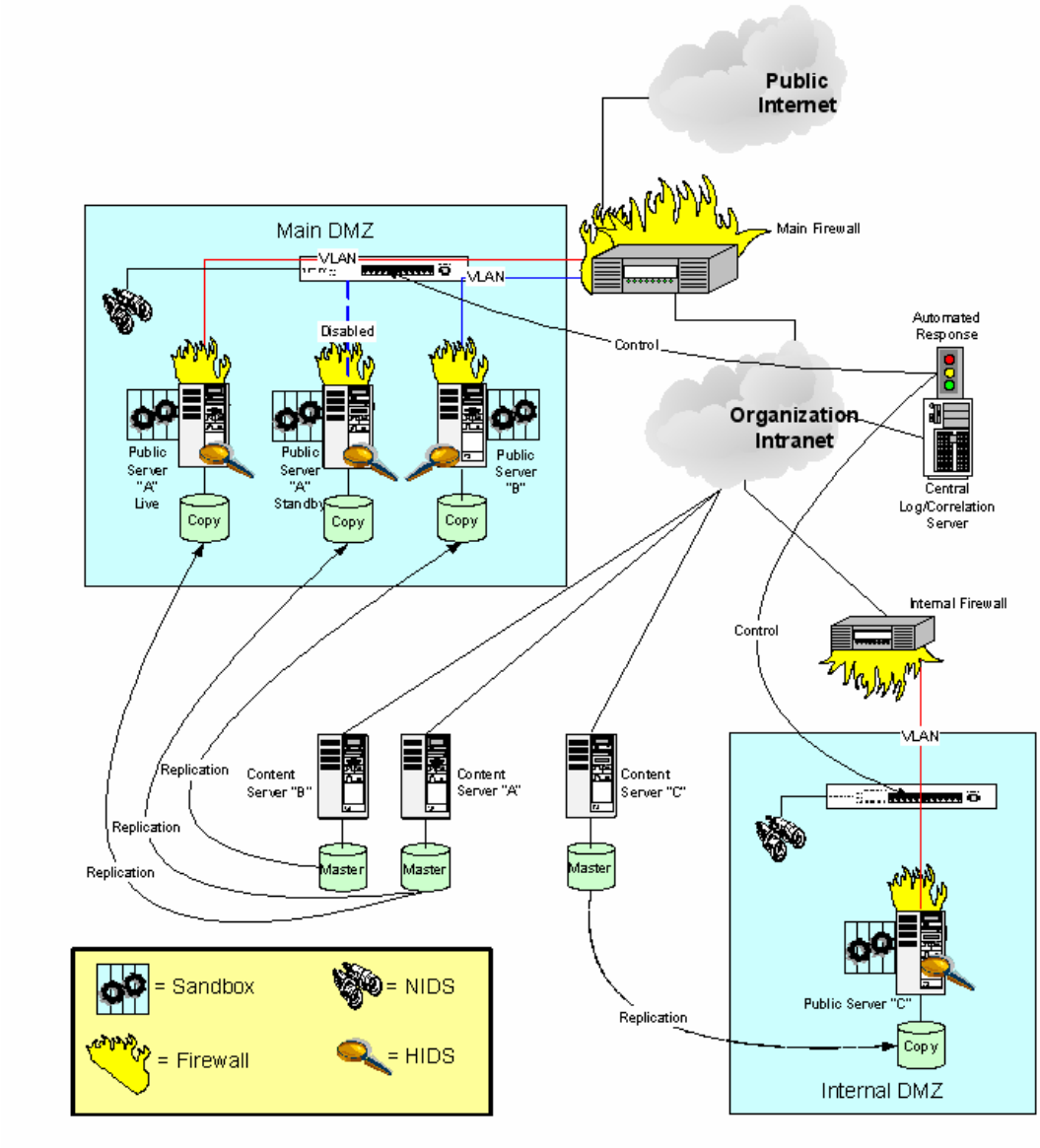


Figure 3 - Target Architecture

CONCLUSION

Much experience has been amassed within the DREnet in offering protection to public servers. An effective protection posture must start with the secure configuration and subsequent validation of the base system. If only the absolute minimum required system and network services are available on the system and the associated software is free of known vulnerabilities, the likelihood of a successful compromise is greatly reduced. When layered protection mechanisms are also implemented, the security of the public server is further enhanced. Most of the protection mechanisms described within this paper are relatively easy to implement. The strength of the resulting protection posture is directly attributable to the manner and depth in which the security mechanisms are applied.

It is difficult to determine how much protection is sufficient without the ability to monitor public servers. Our current monitoring and intrusion detection capabilities could still be strengthened since they cannot easily detect compromises and do not provide good visibility into the health and operation of the server. One should deploy NIDS sensors to monitor DMZ segments and deploy HIDS on each public server. A shortcoming of normal monitoring is that log information is collected and stored in numerous locations. Individual scripts can be used to achieve basic event correlation, but sophisticated event correlation requires the deployment of enterprise security management products. However, commercial enterprise security management solutions tend to be quite expensive and this cost must be weighed against their expected benefits.

An automated response system can disable a server if it believes the server to be compromised. A good automated response system is particularly important in unclassified environments where the servers can operate in unmanned environments up to 75% of the time. We believe we can develop the response technique without much difficulty by reconfiguring a DMZ LAN switch to disable the port associated with a compromised server. The challenge is in deciding what should trigger an automated response – a task best suited for the enterprise security management system since it possesses a broad view of the entire network. In the absence of an enterprise security management solution, however, the HIDS alone may provide sufficient evidence of a successful compromise since it possesses a detailed view of the server itself. The design and implementation of an automated response system requires a great deal of thought since it could result in an unexpected self-imposed denial of service.

We believe that a layered protection posture is an absolute requirement for protecting public servers. Although the DREnet only implements a subset of the target architecture presented in this paper, we already struggle to ensure configuration consistency across the various heterogeneous network devices, security devices and server systems. When a new service is deployed or an existing service is altered or retired, numerous devices must be independently reconfigured using the device's own management tool or interface. Without configuration tool support, this process can be error prone and could result in a weakened protection posture or improper server operation. This issue can be addressed by adopting a single vendor homogeneous solution or by the availability of a configuration audit tool that is able to operate with network and security products from different vendors.

ACKNOWLEDGEMENTS

This paper is based in large part on experiences gained and lessons learned through managing the DREnet over a number of years on behalf of Defence R&D Canada. The author would like to thank Vincent Taylor, Joanne Treurniet and Steve Zeber of Defence R&D Canada as well as Christopher James French of NRNS Incorporated for their advice and support during the writing of this paper.

REFERENCES

[1] SANS School Store, Step-by-Step Guides, https://store.sans.org/store_category.php?category=stepxstep

[2] National Security Agency, Security Recommendation Guides, http://www.nsa.gov/snac/

[3] VPN-1/FireWall-1 Gateways , http://www.checkpoint.com/products/protect/firewall-1.html

[4] Anti-virus, Anti-spam, Content Filtering Security , http://www.esafe.com/esafe/default.asp

[5] Defence Pro, First Strike Security, http://www.radware.com/content/products/dp/Default.asp

[6] IANA Assigned Port Numbers, http://www.iana.org/assignments/port-numbers

[7] Smashing The Stack For Fun And Profit, http://www.insecure.org/stf/smashstack.txt

[8] Defeating Solaris/SPARC Non-Executable Stack Protection, http://www.dtmf.com.ar/texts/non-exec-stack-sol.html

[9] GCC Extension for Protecting Applications From Stack-Smashing Attacks, http://www.trl.ibm.com/projects/security/ssp/

[10] StackGuard, http://www.immunix.org/stackguard.html

[11] Overview of Windows XP SP2 Security Technologies, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwxp/html/securityinxpsp2.asp

[12] Compiler Security Checks In Depth, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vstechart/html/vctchcompilersecuritychecksindepth.asp

[13] Synchronization Fundamentals, http://www.microsoft.com/technet/prodtechnol/acs/proddocs/ac2k/acjgma_sycover.asp

[14] Welcome to the rsync Web Pages, http://samba.anu.edu.au/rsync/index.html

[15] RepliWeb File Replication and Content Distribution Software, http://www.repliweb.com

[16] The Open Source Network Intrusion Detection System, http://www.snort.org

[17] Home of the Tripwire Open Source Project, http://www.tripwire.org

[18] Syslog-ng, http://www.balabit.com/products/syslog_ng/

[19] CORE WISDOM – Secure Logging, http://www1.corest.com/products/corewisdom/CW01.php

[20] SILICON DEFENSE SnortSnarf Snort Alert Browser, http://www.silicondefense.com/software/snortsnarf/

[21] Intellitactics, Inc. - Enterprise Security Management Software, http://www.intellitactics.com

GLOSSARY OF TERMS

DMZ (De-Militarized Zone)	A DMZ provides isolation between the public network and the organization's Intranet.
Enterprise Security Management	Enterprise security management products include event correlation and data mining capabilities that facilitate the discovery and analysis of attack patterns across the entire enterprise.
HIDS (Host Intrusion Detection System)	A HIDS is co-located on the server itself and monitors system activity such as unauthorized processes and network services as well as altered system files.
Intranet	An internal controlled network that is typically protected from the public uncontrolled network (i.e. the Internet) with the use of a firewall.
NIDS (Network Intrusion Detection System)	NIDS sensors are deployed in strategic locations within the network infrastructure to monitor network traffic passively in order to detect attacks and intrusions.
Stack-Smashing Attack	Stack-smashing attacks attempt to exploit buffer overflow vulnerabilities by corrupting the processor's execution stack and invoking the attacker supplied code.
Virtual Sandbox	A Virtual Sandbox provides a controlled operating environment for public server processes and is used to imprison the server process as well as the intruder if the server process is compromised.
VLAN (Virtual Local Area Network)	VLANs provide link layer isolation on local area networks. Network nodes can only communicate with members of the same VLAN.

Contracting out Governmental Web Services

(Externalisation de l'hébergement de sites web gouvernementaux)

Laurent ROGER
DGA/DCE/CELAR
BP7419 35174 Bruz
France

laurent.roger@dga.defense.gouv.fr

ABSTRACT

Contracting out governmental web services

This paper describes the out contracting process of governemental web services focused on the analysis of provider's security measures.

This analysis relies on CELAR (French MoD – Procurement Agency) savoir faire. Input, output, tools and process improvements are described.

The results of the assessments conducted during the past 3 years are pushed into System Security Engineering-Capability Maturity Model. A new concept is proposed ,based on this model : the adaptative confidence profile. Lessons learned are detailed in conclusion.

Externalisation de l'hébergement de sites web gouvernementaux

L'exposé porte sur la démarche d'externalisation de l'hébergement de sites web gouvernementaux en particulier l'examen des dispositions de sécurité des hébergeurs.

L'analyse de ces dispositions est réalisée suivant un savoir-faire maîtrisé par le CELAR (Ministère de la Défense - Délégation Générale pour l'Armement - Centre d'Electronique de l'Armement) depuis 1998. Les éléments clés de ce savoir-faire sont décrits : entrées, sorties, outils et amélioration du processus.

L'évaluation des résultats pratiques obtenus depuis 3 ans est effectuée par rapport aux modèles de maturité SSE/CMM (System Security Engineering-Capability Maturity Model): présentation du modèle SSE/CMM, grille d'analyse pour l'hébergement (profil de confiance dynamique), retour d'expérience.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

1.0 CONTRACTING INTERNET SERVICES FOR MOD

French Ministry of Defense identified early Internet both as a threat for his information systems and an opportunity for his institutional communication.

The first project was in 1998 the www.defense.gouv.fr web site. Upgrades of this site and other web sites project are now available on Internet : research (www.recherche.dga.defense.gouv.fr) , on line procurement (www.achats.defense.gouv.fr) , armament portal (www.ixarm.com), etc …

Use of internet services is defined by Ministry of Defense directives [1][2][3]. Directives advise the project manager to use CELAR expertise for security aspects.

Basic requirements for those projects are :

- **domain naming** : usually root domain is gouv.fr, exceptions are handled by a committee
- institutional communication requires **integrity** of incoming data (news, publishing time) and output data (web pages). Public image of MoD must be preserved.
- Web sites must be available anywhere, anytime. Stopping for short period of maintenance might be accepted but overall **availability** is a major concern.
- **Imputability** : MoD wants to be sure that unidentified person can't produce information on the site.

2.0 CELAR ISO9001 PROCESS

CELAR is ISO9001 since 1998.

The technical process , aimed to "assist project manager for their internet services project" , was introduced into our quality system in 2001.

1.1 Process input

It is required to meet the project manager to exchange : explanation on applicable laws and directives, project documentation, project timeline, outcontracting requirements etc …

Internet Service Provider ISP's assessment is based on questionnaire (that can be sent within the procurement process) and on site visit for final selectionned ISP. Data collected with these imputs are used to produce the outputs.

1.2 Process output

Expertise on project documentation is the first job : missing requirements are added, questions related to information security : supplier organization, project management, existing infrastructures or previous projects.

Expertise on system architecture : the solution proposed by the supplier is reviewed to reveal architecture weaknesses or vulnerabilities.

Expertise on ISP « maturity » : with the questionnaire and on site visit, this maturity is evaluated. An action plan is proposed both for ISP and project manager. Indeed, not only the supplier can improve his process, organization or technical solution, but the project manager has some tasks to complete in order to meet the MoD requirements previously listed.

1.3 Process tools

Models of reports are used to minimize the delivery delay. The questionnaire is a short check-list about the following topics :

- security policy : level of formalization and use : steering committee, training, responsibility …
- organization : description of jobs involved and responsible for security
- procedures : description, how are they diffused, known and verified
- physical security : description
- networks : availability, remote access
- backup
- security survey : subjects, who, how
- security configuration : who, how, relevance, coherence, test and validation
- audit : who specifies and uses internal audit logs, warning procedure, external assessment, previous alerts management.

1.4 Process improvement

Written in 2001, this process was updated in 2003 : a new model of reports was added.

3.0 SYSTEM SECURITY ENGINEERING-CAPABILITY MATURITY MODEL

Reader is invited to read [4] for complete explanation on SSE-CMM.

Please note that no appraisal compliant with SSE-CMM have been done for the following paragraphs, it's just an exercice ☺.

We will only study in this paper this model as a "basis for security engineering evaluation organizations to establish organizational capability-based confidences".
For the purpose of contracting internet services, there are three actors in this process : the project manager, the ISP and the MoD expert.

The three main area of the security engineering process are : engineering, risk and assurance process. The three actors are involved in these 3 area depending on the process area studied.

A capability level from 1 to 5 is determined for each process area :

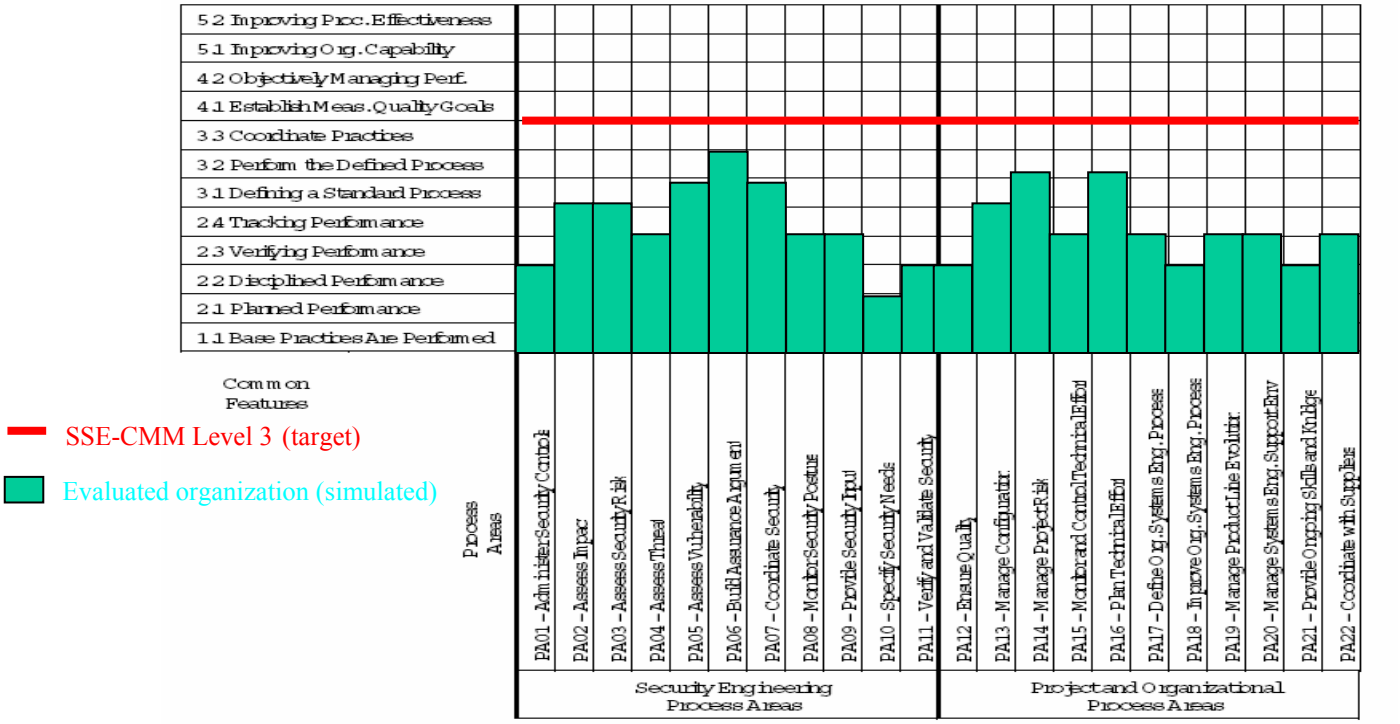


Figure 1: capability level (simulation)

In this simulated case, we see that level 3 is not reached, level 2 neither. If we try to measure the effort to reach level 3 by using the following metrics : 1 point for 1 step, we find 91 points. This metrics is not good enough because effort is not the same along process area and level steps but it's enough for our study.

Action plan to reach level 3 would be conducted for each of the three actors : let's say 70 points for the ISP, 15 for project manager and 6 points for MoD expert.

4.0 EXTENSION TO SSE-CMM : ADAPTATIVE CONFIDENCE PROFILE

This model can be improved by 2 ways for our purpose :

- ISP don't need to reach a full SSE-CMM level to match our needs (full compliance costs time and money)
- the level of assurance depends on the system and the environment (it might be modified by AWR – Alert Warning Response - levels for example)

We propose the use of an « adaptative confidence profile »

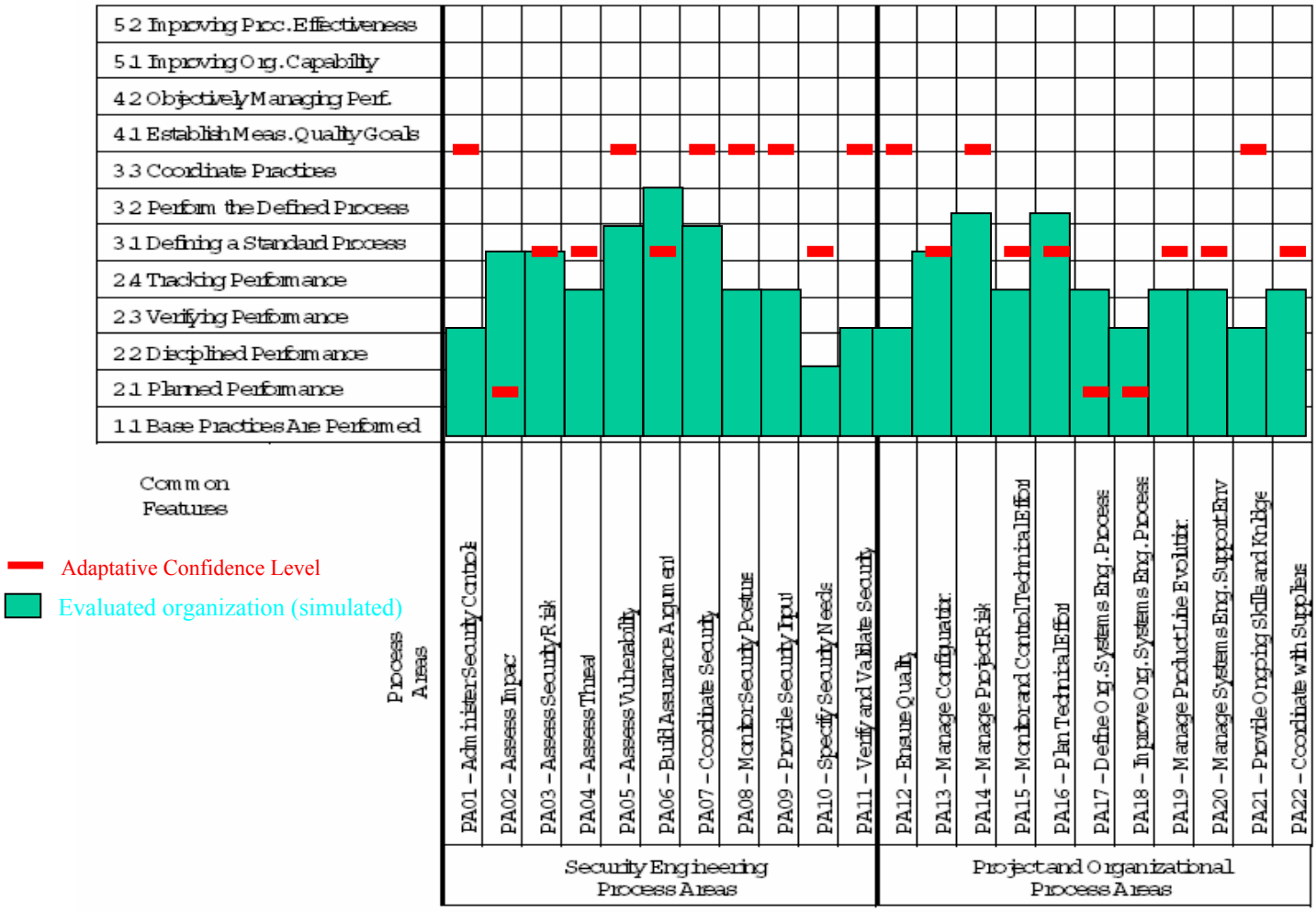


Figure 2: adaptative confidence level (simulation)

In this simulated case, we see that our confidence level is sometimes not reached, sometimes exceeded. If we try to measure the effort to reach our confidence level by using the previous metrics : 1 point for 1 step, we find 42 points. We can also see that it exceeds our needs by 12 points.

Action plan to reach our confidence level would be conducted for each of the three actors : let's say 21 points for the ISP, 15 for project manager and 6 points for MoD expert.

Let's comment this, if MoD expert and project manager probably had the same amount of work , the benefit would be first for the ISP who would divide by 2,5 the amount of work, but the major benefit would be for the project cost : the less time we spend, the more money we earn for the same level of confidence. The exceeding levels should be studied to reduce cost too.

The main difficulty is the definition of the confidence profile but another advantage is the ability to match this with AWR levels. For example, to prepare all levels of warnings but only spend money during high level of warning, and reduce cost of ownership during low level of warning.

5.0 RESULTS [1998-2003]

- First period allow to construct and simplify our process
- Second period (until now) dedicated to improve this process
- Divide time and charge of expert by 2.5 between 1998 and 2003.
- ISP improved their security during this period : this is demonstrated by ISP that have been evaluated at least twice

6.0 LESSONS LEARNED

- security label for ISP (ISO12207, IS17799) is not enough : some ISP have such a label but the perimetrer is not always the same required by our projects, another analysis should be done to analyse differences between these bests pratices.
- People and organizations are major risk factors.
- Project manager is the « key » for success
- Adaptative confidence profil is useful for
  - the expert (assessment time)
  - the project manager (adaptative confidence)
  - the evaluated organization (money)

[1] Instruction n°1829/DEF/CAB/CM/3 relative à la charte de nommage Internet du ministère de la défense : http://www.defense.gouv.fr/creasite/txt_instruction1829.htm

[2] Instruction n°1830/DEF/CAB/CM/3 relative à la mise en œuvre de services en lignes ou de sites Internet par les états majors, directions et services du ministère de la défense : http://www.defense.gouv.fr/creasite/txt_instruction1830.htm

[3] Instruction ministérielle n°8192/DEF/CAB/CM3 relative aux modalités d'accès et à l'utilisation d'Internet au sein du ministère. http://www.bo.sga.defense.gouv.fr/visualisation.aspx?JOB=03PP31&PAGE=5182

[4] System Security Engineering-Capability Maturity Model - Model Description Document version 2.0 April 1999 http://www.sse-cmm.org/model/ssecmmv2final.pdf

Network Vulnerability Assessment: A Multi-Layer Approach to Adaptivity

Professor Ann Miller and Professor Kelvin T. Erickson
Department of Electrical and Computer Engineering
University of Missouri – Rolla
Rolla, Missouri 65409-0040
USA

milleran@umr.edu kte@umr.edu

ABSTRACT

Adaptivity requires at least frequent, and ideally real-time, updates and also requires the ability to analyze, respond and reconfigure. Such network management flexibility requires several types of information and capabilities, which include response of the network to node failure, reachability of nodes, and boundary control. The first issue determines how the network will respond to a "problem", which might be caused by an internal defect of the node or by a failure caused by external faults through the environment, accidental user error, or malicious input. In this event, network analysis on reachability can provide input for routing around the disabled node in order to maintain connectivity for as much of the network as possible. Furthermore, up-front reachability analysis also helps to identify potential access which is unwanted and thus aids in intrusion avoidance. The problem is exacerbated when multiple nodes fail, e.g., distributed denial of service. In this case, boundary control is important in order to isolate affected subnets, to keep these nodes from affecting additional nodes, and to provide limited services until the full system can be restored.

This paper presents a multi-layer analysis of a small (30 node) factory automation laboratory, used in Supervisory Control and Data Acquisition (SCADA) applications. Our approach views the network and applications as a system of systems. Network vulnerability is assessed at several layers; results and recommendations are discussed. Finally, considerations for extension and scalability are presented.

1.0 INTRODUCTION

NATO and its member Nations are engaged in Transformation; a significant portion of this involves the shift from platform-centric operations to network-centric or network-enabled operations. In civilian systems as well as in the military arena, computer-based systems are increasingly linked with other systems to form large-scale systems of systems. These networked systems provide opportunities for increased robustness, for example, load balancing and load sharing in electric utilities. These networked systems also pose the risk of cascading failures, with blackouts across utility boundaries as a classic example.

For the military, network-centric warfare includes the capability of sensor-to-shooter operations. In civilian applications as well, for example, factory automation, networks of sensors are connected to larger networks, typically through some form of gateway or access point. There is a wide spectrum in capability and type of sensors. Many provide Supervisory, Control, and Data Acquisition (SCADA) capability. As we move to large-scale systems of networked sensors, there is the need to understand the robustness and the vulnerability of these networks when interconnected with C2 and other critical systems.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Communication connections, especially network connections, are the main security threat to a factory automation system. At the very least, one must connect to a controller in order to program it, usually through serial interface. Most security threats are through a peer-to-peer network where the controllers, personal computers, and communication gateways are connected. The security threats to a factory automation system can be categorized as (1) inside the system by unauthorized users; or (2) outside the system by unauthorized or malicious users. Both types of threats are considered in this paper. All factory automation systems are vulnerable to inside threats. An inside threat is often from an inexperienced employee that tries to access a controller for which s/he is not authorized. Less often, a disgruntled employee can also be a security threat.

This paper first explains the approach and study test bed. Next, the vulnerabilities at the application layer and command packet layer are assessed. Finally, recommendations for further study are presented.

2.0 APPROACH AND STUDY TESTBED

Given the importance of understanding the impact of netted sensors within network-centric operations, the approach in this study was to understand a small network of sensors within a larger application and to assess the vulnerability of the entire system through multiple layers of application, network, and device. While the particular application is that of factory automation, the network topologies examined are similar to those in many other environments, including military communications. And while the application software is specific to SCADA network, it exemplifies the myriad problems encountered with Commercial Off-the-Shelf (COTS) products, particularly those which run in a Microsoft Windows environment controlling sensors consisting of Programmable Logic Controllers (PLCs).

The testbed for this analysis is the Factory Automation Laboratory within the Electrical and Computer Engineering Department at the University of Missouri – Rolla. The laboratory contains a SCADA network with over thirty PLCs from various vendors and multiple communication networks, as shown in Figure 1. The laboratory has PLCs from Rockwell Automation (ControlLogix, PLC-5, and SLC-500), Modicon (Quantum, Momentum, 984), GE Fanuc (VersaMax, Micro 90-30) and Siemens (S7, TI505 series, Moore APACS). In addition to an Ethernet connection, the laboratory supports Control Net, Data Highway+ (Allen-Bradley), Modbus+ (Modicon), and Series Ninety Protocol (GE Fanuc). In addition, eight workstations in the laboratory support the programming of all the PLCs in the laboratory and provide the human-machine interface (HMI) programming software. The workstations all support the InTouch (Wonderware), RSView (Rockwell Automation), and Genesis (Iconics) HMI packages and can connect to any of the PLCs in the laboratory using the network. The Laboratory contains a few wireless nodes, including both 802.11 and Bluetooth nodes for comparison.

This study examined several network layers. At the application layer, the laboratory supports two control system families, Invensys/WonderWare and Modicon Quantum NOE771-10/FactoryCast, across the network. A system to control a pick-and-place robot arm has been configured that will use one of the PLCs for control; this will be the application for which we examine the relative security of each control system family. Each system family was implemented to allow write access to the PLC on the plant floor.

The SCADA Laboratory network facilities are also accessible from outside the laboratory. The application PLC program is controlled by Wonderware Suite Voyager web server, which can be remotely accessed to control the robot arm.

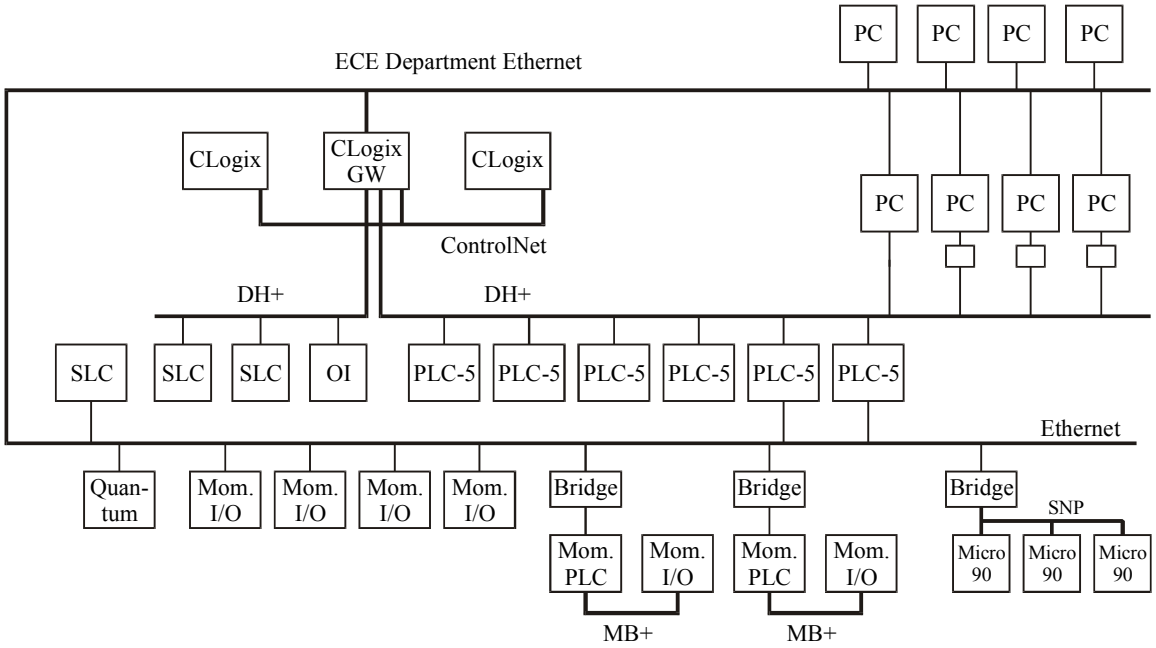


Figure 1: SCADA Network

3.0 APPLICATION LAYER

Although the integrity flaws differ, vulnerabilities exist in both control system products. Schneider Electric's NOE771 communication module in conjunction with FactoryCast software uses only passwords to authenticate the user's identity. These passwords are not encrypted or hidden. The SuiteVoyager system by Invensys/Wonderware uses Microsoft NT authentication, rendering it vulnerable to a myriad of attacks. The software itself works in conjunction with Internet Information Services (IIS) and MS SQL Server, both of which have numerous publicly documented vulnerabilities. Details on each control system family follow.

3.1 Assessment of the Invensys A^2 Control System

The Invensys A^2 control system uses Microsoft.NET technology to integrate the many components. The software family relies on Microsoft NT security to safeguard the availability of readable and writeable web pages in Suite Voyage. Implementation of any of the known attacks to NT vulnerabilities would disable the SuiteVoyager(SV) system. However, since SV requires that WonderWare's InTouch must be in runtime mode on a local machine to access the physical system through Suite Voyager, floor operations would not be interrupted unless the attack was targeted at that specific machine.

Susceptibility to External Attacks. Due to the integration of Microsoft software in running Suite Voyager, the system is sensitive to any type of system attack exploiting known vulnerabilities which have not been patched. The key to unauthorized access in Suite Voyager is an administrative login. The antivirus software vendor Sophos has identified a worm called W32/Sluter-A that searches for administrative passwords. This worm searches IP addresses for open ports then checks C$ and Admin$ for administrative passwords. There is also a remote access Trojan virus that allows access to Windows NT/XP/2000 machines. Successful implementation of the Trojan gives the hacker the ability to gather system information, scan the network, and change SQL Server activities.

Susceptibility to Internal Attacks. Internal tampering is very easy by anyone with administrative rights on the server machine. With admininstrative access, one could alter the user profile for any user in the MS Server 2000 system. Since SuiteVoyager relies on the server for user information, any changes would prevent the user from entering the system. A simple scenario illustrates this concept. An administrative user leaves, with the terminal still logged in. While the administrator is away from the terminal, the attacker takes a seat. The attacker opens the administrative tools and enters the Users folder. Passwords can be changed from here by simply right clicking and entering the new password. The server does not ask for a confirmation of the old password before changing to the newly entered password. Figures 2, 3, and 4 show this sequence of action.

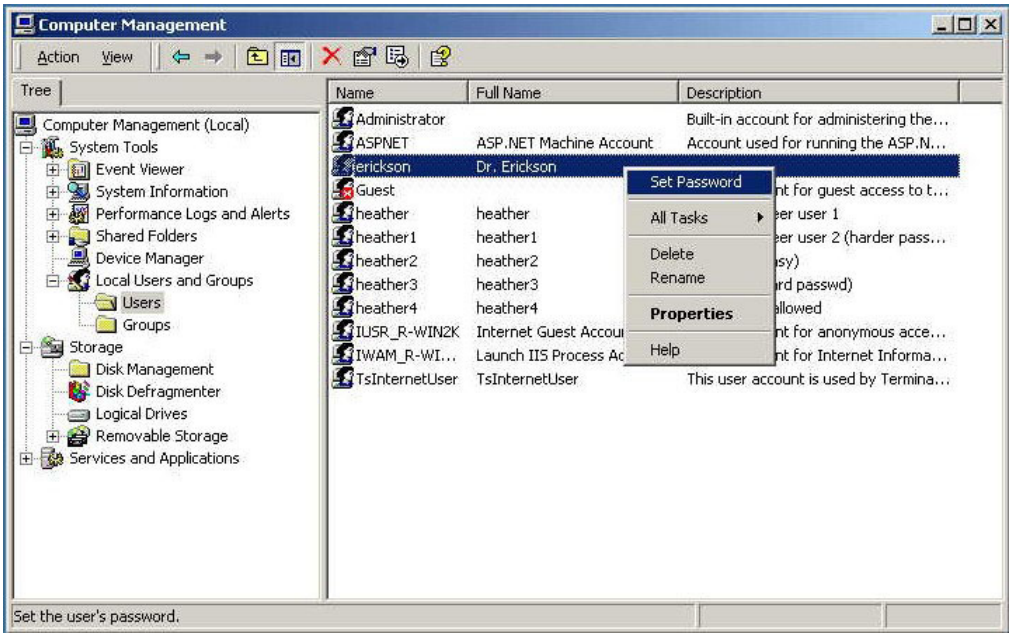


Figure 2: Start of Administrative Rights Takeover

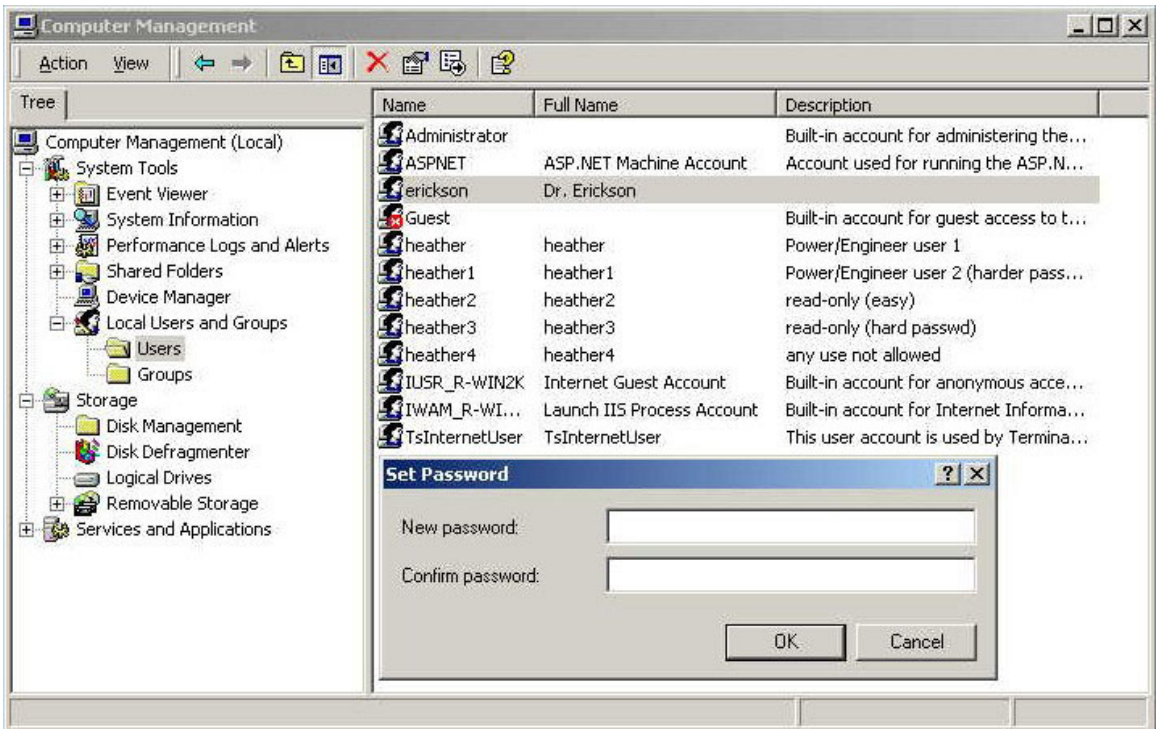


Figure 3: Change of Admin Password without Confirmation of Old Password

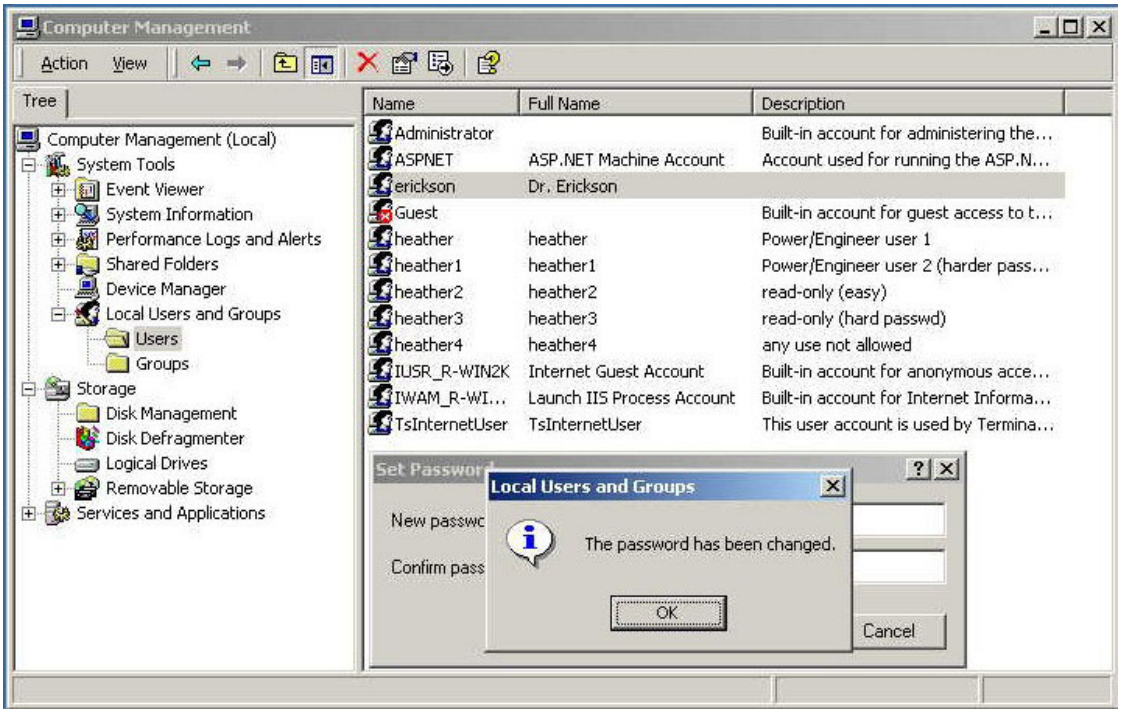


Figure 4: Successful Admin Password Takeover

With administrative rights, the intruder has the ability to change a user's status as well as changing remote login/ authentication options, the user group, and the lifetime of the account password.

Susceptibility to Buffer Overflow Attacks. The complete integration of Microsoft software within the Invensys system makes SuiteVoyager vulnerable to this type of attack. Reported buffer overflow vulnerabilities include: (i) Windows 2000 operating system file, ntdll.dll used to react with the system's kernel. (ii) other critically rated advisories relate to a buffer overflow vulnerabilities in MS SQL Server; and (iii) MS IIS also has buffer overflow vulnerabilities: (1) crashing IIS Server by sending overlong SebDAV, PROPFIND, or SEARCH requests; (2) exploitation of buffer overflow in the Server Side Include File handling code; (3) a cross-site scripting bug; and (4) a denial of service flaw that requires the hacker to upload files to the server.

3.2 Assessment of the Schneider Automation Control System.

Attacks on this system were focused on interrupting the webserver module or gaining access to the system. The module and FactoryCast software rely on password protection only to deter intruders. One major flaw that relates to both the external and internal threat is the possibility of someone copying the FactoryCast software. By using the same destination file, an outsider could overwrite all of the parameters set by the original engineer. All that is needed is the IP address of the NOE module.

Susceptibility to External Attack. The external attack was simulated by retrieving the password file of the NOE module through the file transport protocol (ftp). Read permission userID and password are not encrypted and can be found in a file named userlist.dat in the main directory of the webserver. Write permission is controlled through a password only and is also not encrypted. This "write" password resides at the address …rdt/password.rde. The FTP password file is the only encrypted file in the system. The file is in the following form: USERID:encrypted password. Files such as these can be deciphered using common cracking programs.

Susceptibility to Internal Attack. This system appears very friendly to internal attacks. Anyone with access to the FactoryCast program (even outside the organization) can change the write access password by transferring a new security file to the module. The external attacker also has this capability. However, the read access file cannot be changed unless the module is powered down and restarted. One quirk in the system is that one only needs read access to change the ftp access password. Also, the internal attacker has the ability to copy the FactoryCast program and re-implement it to control the same system with different parameters. The only safeguard against such actions is the time/date matching that the module performs when downloading changes to the program. This time/date matching requires that the source program and the namespace be transferred on the same day and close to the same time. This helps protect changes to the variables, but does nothing to protect the aforementioned password flaws.

Susceptibility to Buffer Overflow Attack. A successful attack was generated by using the write-back privileges on the data page of the website. When using the data editor to provide a setpoint for a feedback proportional-integral-derivative (PID) control system, the operator could easily enter a value that is outside the range of the controller.

4.0 COMMAND PACKET LAYER

To examine the vulnerabilities at the command packet layer, the behaviour of three different Allen-Bradley PLC Ethernet interfaces were analyzed: (1) PLC-5/20E processor; (2) SLC-5/05 processor; and (3) ControlLogix ENET interface. Both the PLC-5/20E and SLC-5/05 processors have built-in mechanisms for handling a DoS attack.

The PLC-5/20E and SLC-5/05 processors have a built-in web server, so Denial of Service (DoS) attacks were tried by flooding port 80 with packets. When presented with high levels of Ethernet packets, the communication port is temporarily closed [1]. Specifically, if the port receives more than 16 Ethernet frames within 10 ms, the processor disables receive interrupts for 6 ms. After the 6 ms interval, receive interrupts are re-enabled and sets the limit to 8 frames within 10 ms. Only when the processor receives no more than 8 frames within 10 ms does it increase the limit to 16 frames. The processor also tracks the number of these network "storms." Flooding the port resulted in successful Denial of Service attacks. No solutions were found to prevent the DoS attack. It should be noted that these attacks were generated by simply flooding the PLCs with data and did not require any significant level of expertise.

For the ControlLogix ENET module, the response to a DoS attack was different. It gave no response to a small amount of data , but as the amount of data sent to it and the speed of transmission were increased it started responding by sending arbitrary data. This returned data was not decoded.

For the PLC-5/20E and the SLC-5/05, their behaviour was tested when an operator interface was communicating with the processor and not through the web server, which has limited capability. A small batch control application program was running in the PLC and a workstation was providing the operator interface with a Invensys/Wonderware InTouch run-time screen. When this screen is displayed, the InTouch communication server periodically (every second) polls the PLC for changes in the status of objects on the screen and sends operator commands to the PLC. A second workstation was set up with a client program to capture the data packets between the InTouch application and the PLC. The IP addresses in the captured data packets were verified with the IP addresses of the host machines and the PLCs. Then the hex dump was analyzed to obtain the source and destination port numbers. This analysis indicated that the PLC-5 and SLC-5 communicated on port 2222. The client program was modified to connect to the PLC-5 or SLC-5 at the port 2222. The VB client successfully connected to the PLCs/SLC and some raw data were sent from the client. After receiving 26 bytes of data, the PLCs/SLC (server) closed the port connection to the client.

5.0 CONCLUSIONS AND EXTENSIONS

At the application layer, this study examined two commonly used commercial control system families, Invensys/WonderWare and Modicon Quantum NOE771-10/FactoryCast. A system to control a pick-and-place robot arm was configured that used one of the PLCs for control; this was the application for which we examined the relative security of each control system family. Each system family was implemented to allow write access to the PLC on the plant floor. Although the integrity flaws differed, vulnerabilities were found in both control system products. Schneider Electric's NOE771 communication module in conjunction with FactoryCast software uses only passwords to authenticate the user's identity. These passwords are not encrypted or hidden. The SuiteVoyager system by Invensys/Wonderware uses Microsoft NT authentication, rendering it vulnerable to a myriad of attacks. The software itself works in conjunction with Internet Information Services (IIS) and MS SQL Server, both of which have numerous documented vulnerabilities.

Both products are susceptible to external attacks, including buffer overflow attacks. Both products are susceptible to internal attacks. It should be noted that these attacks do not require any significant level of expertise on the part of the attacker. This illustrates the need to carefully evaluate COTS products for security as well as functionality prior to vendor selection.

Buffer overflows were also generated at the command packet level, resulting in successful Denial of Service attacks. No solutions were found to prevent the DoS attack. Again, this attack did not require a significant level of expertise. However, a fail-safe feature in the PLCs shut down the PLCs to avoid further corruption or attack.

There are several possible avenues for future work. Possible extensions include:

1. Expansion of the buffer overflow attack to flood the PLCs so that when the time-out is completed and the PLC opens the communication port the attack continues.
2. Analysis of the **CIP** (Control and Information Protocol), **Ethernet /IP** (Ethernet /Industrial Protocol) protocol formats so that data meaningful to the PLC-5 and SLC-5 PLCs can be transmitted.
3. Step 2 would then allow design of attacks beyond Denial of Service, specifically to take control of the PLCs remotely.
4. Analysis of the data returned by the ControlLogix ENET module to the client program. This would allow additional attack generation.
5. Detailed modeling and reachability analysis of the SCADA network using a Network Analyzer. There are two types of analyses which we think would be of interest in both commercial and military environments:
   (a) Analysis of just the IP portion of the network, to determine which hosts can connect to the gateway services, and thus who may be able to use the control functions. This analysis will be of interest if there are access control mechanisms, such as firewalls or access control lists in routers, in place in the IP network.

   (b) Some SCADA networks support devices that forward messages between different networks (e.g. between DeviceNet and Data Highway, or between two ControlNets). These devices are analogous to routers in an IP network. The analysis could be applied to the control networks themselves, if a model for the addressing system and the way these routers work can be added to the existing IP model. If the functionality of the IP gateways can similarly be modeled, then the analyzer could be used to determine which devices on a SCADA network could be controlled from which places on the IP network. This analysis would superior to the IP-only analysis, if there is significant functionality in the gateway and control message routing software. In that case, there might be, for example, public read access to some SCADA devices and more controlled access to others.

In conclusion, this study has assessed the vulnerabilities of two of the most commonly used COTS software packages for networks of SCADA sensors consisting of PLCs. It has also demonstrated the susceptibility of such sensor networks to DoS attacks at the command packet level. The need for adaptivity measures, such as boundary control based on reachability, is clearly identified.

ACKNOWLEDGEMENT

This work was partially funded by the Los Alamos National Lab, subcontract 64314-001-03. Their support is gratefully acknowledged.

REFERENCE

[1] Allen-Bradley, *Product Release Notes: Ethernet PLC-5 Programmable Controllers*, pub. 1785-RN003D-EN-P, 2002.

Passive Network Discovery for Real Time Situation Awareness

Annie De Montigny-Leboeuf, Frédéric Massicotte
Communication Research Centre Canada
3701 Carling Avenue
Ottawa, ON, K2H 8S2
Canada

annie.demontigny@crc.ca, frederic.massicotte@crc.ca

ABSTRACT

Network security analysts are confronted with numerous ambiguities when interpreting alerts produced by security devices. Even with the increased accuracy of these tools, analysts still have to sort through a tremendous number of potential security events in order to maintain the desired level of assurance. This paper describes how passive network discovery and persistent monitoring can provide significant contextual information valuable to network security professionals responsible for protecting the network. Techniques discussed include the capability to discover active nodes, their operating systems, the role they carry out, their system uptime, the services they offer, the protocols they support, and their IP network configuration. An attractive feature of this approach is that it focuses on mechanisms that do not rely on access to user data. While this is rarely a concern for the intruder, it can be of the utmost importance to the security analyst. One of the main interests in using a passive approach is that the information gathering process has no impact on the bandwidth or on the monitored assets. This is in contrast with active scanning techniques that are often noisy and intrusive. Passive techniques can be used at all times, allowing near real-time awareness of the security posture of ever-changing networks, and thus helping network administrators remain in control and anticipate upcoming security problems. A network monitoring prototype has been developed to test the techniques described in this paper.

1. INTRODUCTION

Traditional network security devices such as Intrusion Detection Systems (IDS), firewalls, and security scanners operate independently of one another, with virtually no knowledge of the network assets they are defending. This lack of information results in numerous ambiguities when interpreting alerts and making decisions on adequate responses. Passive network discovery and persistent monitoring can provide significant contextual information regarding the components to be protected. This can help address the problem of false positive alarms. The immediate availability of critical and relevant information can limit the number of alerts to investigate, thus help save precious resources. As networks are constantly evolving, it is particularly desirable to detect changes such as the addition of hosts or services, changes regarding protocol usage or operating system versions. New active elements attached to the network can be discovered and profiled passively as they produce traffic.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Passive techniques rely on sensors to monitor packets flowing on the network and inspect packet content (headers or data encapsulated in packets). In contrast with an active approach, the passive information gathering process has no impact on the bandwidth[1] or on the monitored assets. Passive monitoring can therefore be used at all times with no risk of causing service disruptions. On the other hand, it usually takes more time to profile assets passively than actively. Moreover, passive methods will discover network services only if they are in use. For example, an idle but vulnerable sendmail service could be running on a computer and remain undetected. By stimulating responses and behaviours, active tools perform better at uncovering idle points of weakness. This of course assumes that vulnerabilities to look for are known in advance. Because active tools can be disruptive, even the most vigilant organisations tend to use them sparingly.

To benefit from the strength of both technologies, passive and active approaches can be used in conjunction to maintain a complete picture of the network.

The Network Security Research Group at the Communication Research Centre (CRC)[2] has been developing a passive tool to build and maintain an inventory of network components. It is primarily designed to operate on Local Area Networks. This prototype complements an in-house Network Mapping Tool that actively scans the network to discover its topology and available information about the network components [1].

The strength of the passive approach adopted here is in combining outcomes of techniques defined for different purposes in order to produce and maintain a comprehensive up-to-date description of network resources. An attractive feature is that it does not rely on access to user data. Moreover, the techniques used are not vendor-dependent and are therefore applicable to heterogeneous networks.

It is assumed when describing the techniques that the proper sensor coverage is available. Network devices such as switches, routers and firewalls will limit any one sensor's view of the entire network. The locations and number of sensors required may vary greatly from one network to another based on the topology and the coverage level desired. The sensor deployment and monitoring strategy along with the issues around the distributed aspect of such systems are not within the scope of this paper.

The prototype is being tested using Switched Port Analyzer (SPAN) ports[3] on different switches. The information gathering techniques are not however restricted to this deployment strategy. Many of the profiling techniques require a minimal number of packets and do not assume that the sensor is able to see all of a host's communications. This approach allows flexibility when determining the monitoring coverage scenario.

Passive network profiling is an emerging technology. Given the unique strengths that a passive approach provides, a number of companies are starting to include passive network discovery solutions into their product lines. Real-Time Network Awareness (RNA) by Sourcefire [2] and Network Vulnerability Observer (NeVO) by Tenable[3] are examples of emerging network security products based on passive profiling techniques.

[1] Depending on the sensor deployment strategy, the sensors may have the capability to communicate among themselves or with a management station to exchange information. While the gathering process consumes no bandwidth, the communication among the sensorss will consume part of the bandwidth, unless the network architecture design includes a separate network dedicated to sensors and management communications.

[2] An agency of Industry Canada, Government of Canada.

[3] A switch SPAN port is setup on a switch to copy all packets traveling through the switch to this port. The term SPAN is a Cisco term; other vendors may refer to this functionality using the term "port mirroring".

The remainder of the paper is laid out as follows.

- Section 2 discusses how the information passively gathered can provide significant contextual information and help security analyst maintaining situation awareness.
- Section 3 describes some of the underlying techniques derived during the development of the prototype. The mechanisms discussed provide the capability for passively discovering active nodes, their operating systems, the roles they carry out, their system uptime, the services they offer, the protocols they support, and their IP configuration. The section ends with an example illustrating the accumulation of information by monitoring normal traffic.
- The paper concludes after a brief discussion in Section 4 of concurrent related work being done by the Security Research Group at CRC.

2. BENEFITS OF THE APPROACH

2.1. Immediate Availability of Critical Information

To improve the security posture of the network, many organisations prohibit the presence of peer-to-peer applications (e.g. Napsters, Gnutella, Kazaa) and instant messaging applications such as ICQ. Most also impose restrictions on the configuration of computers that connect to the network. This can be for example related to the choice of the Operating System (OS).

In environments where desktop and server locked down solutions are not viable, profiling the network assets based on their Operating System (OS) versions, the network services they provide and the protocols in use is therefore necessary to ensure and enforce compliance with the organisation's security policy.

Vulnerability scanners or network auto-discovery tools typically conduct this sort of security assessment. Such tools however can only provide a snapshot of the security posture at a particular point in time. The continual availability of accurate descriptions of network assets is important to network security professionals responsible for identifying and fixing vulnerable systems before they are compromised, making the network more robust and resilient to cyber attacks. When attacks do occur or when a known virus is spreading, the information can be used to ensure that human resources are not wasted chasing down false positives.

As an example, consider the case of a Samba exploit released last spring that affected Linux, Solaris, FreeBSD, NetBSD and OpenBSD running Samba 2.2.x [4]. The exploit forced the compromised computer to return a root shell, which allows an attacker full access to the victim's computer. The attack has a distinctive signature based on port numbers and data content and can thereby be detected by traditional IDS. The IDS are however likely to raise false alarms if the attack is randomly targeting several computers. Many of the targeted systems may not be vulnerable to this attack. An alert generated for a host running Windows for example should get a low priority for this particular alarm. The network administrator could then focus attention towards vulnerable targeted systems.

2.2. Transparency of Passive Techniques

While active security scanners may provide an excellent picture of a network's security posture, they may generate a lot of traffic in order to identify vulnerabilities in hosts connected to the network. Not only are they noisy, they can sometime crash a system with packets that deviate from what the target is configured to

handle. Because of these pitfalls, organisations tend to use active scanners with care. In some environments, this leaves the security analyst troubleshooting problems based on outdated network profiles.

Passive technology may therefore be particularly desirable between intermittent security assessments. In contrast with active techniques, passive discovery mechanisms do not consume bandwidth and do not disrupt network assets.

2.3. Access to application data

Some passive techniques as proposed in [5] are based on clear text information that can be retrieved from decoded packets. An example of this is the content of the HTTP User-Agent field, which states with more or less precision the Operating System (OS) running on the client requesting a web page. For instance "Linux 2.4.18-14 i686" indicates a Linux kernel 2.4.18-14, which is known to be the basic kernel version distributed with Redhat 8.0. Similarly, the X-Mailer field in headers of E-mails can also reveal the OS of the sender. Telnet and FTP server banner grabbing are other examples of explicit information gathering.

While information obtained in this manner can be quite precise[4], these techniques rely on the availability of the data at the application layer. This limits the applicability of the techniques. For example, the application layer may be encrypted, as it is the case with the Secure Shell (SSH) and the Secure Sockets Layer (SSL) protocols. Another limitation of the applicability of an application layer based approach is when the application layer of every packet is ripped off at the time of capture. Depending on the organisation's policy, this may be done for privacy issues. Limiting the capture length of packet is also a common practice when storage capacity is in place to allow for post analysis of traffic traces.

The approach adopted here attempts to derive information rather than grabbing it explicitly. The analysis is confined to headers at the data-link, network, and transport layers. This approach can lead to highly accurate results, while not depending on access to any sensitive user data. It also remains viable whether the application layer is encrypted or not.

3. PASSIVE NETWORK DISCOVERY

This section describes how some useful information can be derived from observing the traffic activity of hosts connected to a network. The passive techniques adopted go beyond individual packet analysis. Stimulus and response packets are identified, paired, and analysed together to allow for more accuracy. Our method also allows for analysing samples of packets transmitted by a computer (typically to observe how certain header fields evolve).

3.1. Monitoring Mechanisms

Packets are monitored based on the three categories of tests described below.

[4] Masquerading can however be relatively easy at the applications layer using third party tools or simply by modifying the properties. Fields like HTTP User-Agent and X-Mailer are not even mandatory. The content of such fields can be overwritten without affecting the communication. In the same train of thoughts, "Banners" are systematically rewritten nowadays by system administrators. Obfuscation and masquerading at the network and transport layers are also feasible, but care must be taken not to break connectivity.

3.1.1. Singleton

Tests in this category are conducted on a single packet (a singleton) emitted by a host. The *Singleton* tests typically look for default values of header fields to gain some knowledge regarding the sender of the packet. The general algorithm to perform a *Singleton* test is as follows:

i. Monitor traffic, listening for packets satisfying a specified filter;

ii. Derive the information once a packet is seen.

3.1.2. Sample

This second category of test requires monitoring a sample of packets generated by a given host. A *Sample* test typically analyses how a certain field evolves as packets are being transmitted. The general algorithm in this case is:

i. Monitor traffic, listening for packets satisfying a specified filter;

ii. Hold in memory monitored packets by Source address until a sample is complete.

iii. Derive the information from the sample.

3.1.3. Stimulus-Response

This category of test consists in listening for pairs of packets coming in opposite directions. Each pair consists of a "stimulus" and a "response". Analysing the two packets together (as opposed to separately) can lead to better accuracy when deriving conclusions. Some information in a response can only be interpreted if the response is examined in the context of the stimulus. Similarly, a stimulus seen with no response also allows better insight as it may indicate that the target remains silent to a certain stimulus. While matching stimulus and response is easily done with our approach, it seems to be overlooked by current available passive tools. They typically perform a test of type *Singleton* whether the packet is a stimulus or a response. The general algorithm of a *Stimulus-Response* test is as follows:

i. Monitor traffic, listening for packets satisfying either the stimulus or the response filter;

ii. If the packet is a stimulus, hold it in memory by Destination address and go back to monitoring. If the packet is a response, search allocated memory for the corresponding stimulus and then go to step iii;

iii. Infer the information based on the stimulus-response pair.

In some situations, it may be appropriate to combine two or more of the above categories.

3.2. Passive Capabilities and Underlying Techniques

Of all the information that can be retrieved using a passive approach, the following are of particular interest for providing contextual, real-time situation awareness:

- the capability to discover active nodes,
- system uptime (i.e. time elapsed since last reboot),

- operating systems,
- role carried out (e.g. workstation, server, switch, router),
- services offered (e.g. web server, DNS server, DHCP server),
- protocols supported,
- IP network configuration.

Basic lightweight mechanisms to gather such information are introduced in the next sections.

3.2.1. Host Discovery

The host discovery process attempts to identify every active host connected to the network. An active element can be discovered as it produces traffic (of any kind). In particular, the Address Resolution Protocol (ARP) can be used to identify IPv4 hosts that are directly attached to the Local Area Network (LAN) being monitored. The ARP protocol is confined to the broadcasting domain and is used to map an IP address to the hardware address. ARP requests are broadcast each time a host needs to obtain the hardware address associated with an IP address. The machine matching that IP address sends back its hardware address. Because the ARP requests are broadcast, a sensor located anywhere on the LAN may monitor these requests to build a list of active IP addresses attached to the LAN (along with their corresponding hardware addresses).

Passive monitoring of networking activity helps maintain constant awareness of resource inventory and of introduction of new hosts on the networks. Additionally, it may also be useful to detect system reboots since they may coincide with system configuration changes. Frequent reboots can also be the symptoms of virus-like infection as it was the case recently with the MS Blaster worm [6]. Monitoring such events can help identify malfunctioning systems, and minimize disruption of services.

There are passive methods that can be used to detect reboot events. First, several systems show a particular transmitting behaviour when they connect to a network (and thus if they reboot on that network). When the network interface is being brought up, some hosts broadcast a series of ARP requests asking for their own IP addresses. Such requests are known as “gratuitous ARP” packets and are sent in an attempt to determine if some other host already uses the IP address in question. Gratuitous ARP packets are easily detected, as the ARP header fields *Sender IP Address* and *Target IP Address* are identical. Sensing a lot of gratuitous ARPs coming from the same host in a short period of time may indicate a problem.

Secondly, it is possible to determine when a system last rebooted by monitoring the TCP packets having the TCP Timestamp option set. This option is supported by most OSes. The Timestamp Value (TSval) field contains the current value of a (virtual) clock called the "timestamp clock". As pointed in [7], the TSval is tied to the system uptime for several OSes (e.g. some versions of OpenBSD, FreeBSD, NetBSD, Solaris, MacOS, and Linux). This means that each time the OS reboots, the TSval field is reset to 0. By observing how this value increments with time for a few packets, one can determine the update rate of their timestamp clock. The time elapsed since the last reboot can then be computed from the value of the last TSval seen. This technique for discovering passively system uptime constitutes a typical example of a *Sample* test described in section 3.1.2.

3.2.2. Operating System Identification

The ability to remotely identify a target operating system (OS) and version is a definite asset when trying to identify vulnerabilities. For this reason, there has been a lot of effort from the hacker community to develop

tools for OS identification, also referred to as OS fingerprinting. OS identification capability can also be valuable to a network administrator as it can provide information about potential vulnerable hosts connected to the network they are protecting.

OS fingerprinting methods are often based on the principle that differences exist among OS families and versions regarding the implementation of the networking protocol stack. This is especially true when it comes to handling non-standard packets. To detect these differences, several active tools send a sequence of carefully crafted packets (stimuli) to the targets and analyse the resulting behaviour of the targets [8][9][10]. Other tools examine the differences in the way operating systems retransmit unanswered packets [11][12]. These differences can be in terms of delay (between each retransmission) or in terms of number of attempts before giving up. Direct queries seeking OS information can also be made through the Simple Network Management Protocol (SNMP) and NetBIOS [1].

OS fingerprinting can also be conducted passively. Two popular tools, *p0f* [13] and *ettercap* [14] analyse values of selected TCP header fields to deduce the OS. While *p0f* restricts itself to SYN packets, *ettercap* also listens for SYN/ACK packets (a SYN/ACK packet is the response to a SYN packet when a TCP connection is initiated). The fingerprint of *ettercap* based on the SYN/ACK packet is less reliable because some of the fields are influenced by those of the SYN packet. For instance, if a SYN doesn't request the use of a certain TCP option, the SYN/ACK will not advertise the option even if the host supports it.

Other passive techniques as proposed in [5] consist of looking at the application layer, seeking special strings that could identify the operating system. Telnet and FTP banners for instance often state in clear text the OS that the server is running. Some applications also involve option negotiation prior to exchanging any data, and because applications are often platform dependent, this can sometimes be used in OS fingerprinting [5]. These ideas can also be applied to passively retrieve explicit OS information contained in certain SNMP and NetBIOS packets. In all cases, access to the payload of packets is required.

The network security research group at CRC has developed novel techniques for passive operating system discovery. Some of them were inspired by active tools and adapted to be conducted passively on regular traffic. Over a dozen tests have been developed to analyse headers of packets seen on a network. The tests are conducted on various types of protocol headers: ARP, IP, ICMP, UDP and TCP. A database containing the fingerprints of over 200 versions of operating systems has been built.

One of the tests is based on a *Stimulus-Response* (see section 3.1.3) type of test in which SYN and SYN/ACK packets are being paired. While a *Singleton* test is appropriate to examine the SYN packet (and thus gain OS information concerning the initiator of the connection), a *Stimulus-Response* examination is more appropriate to gain precision on the OS of the responder (the sender of the SYN/ACK).

To give another example, one of the tests examines ARP requests. It can be observed that the content of the *Target Hardware address* field varies with operating systems. Because the target hardware address is unknown initially, the field carries no meaning in the request. Most OSes initialise it with 0x000000000000, some others fill it with 0xffffffffffff. More over, the implementers of some versions of FreeBSD forgot to initialise the field and so it contains allocated memory garbage. The prototype manages the information coming from all the tests as traffic is being produced to infer a more precise description of the operating system version.

3.2.3. Protocol Discovery

Keeping track of the protocols in use on a network provides the ability to monitor networking activities. Moreover, protocol discovery can help identify the role of a host, as discussed in a subsequent section. At the time of writing, the protocols of interest for the type of network the prototype monitors are those carried on top of the data link layer (e.g. IPv4, IPv6, IPX, CDP, STP, AppleTalk, Netware, NetBIOS, etc.) and those on top of IP (e.g. IGMP, OSPF, TCP, UDP, ICMP, etc.).

Protocols above IP can be passively identified by examining the IP header of each packet. The protocol number appears in the 8-bit integer *Protocol* field (IPv4), or *Next header* field (IPv6) in the IP headers of each packet. Examples of values are 0x02 for the Internet Group Management Protocol (IGMP), 0x29 for IPv6 over IPv4, 0x59 for Open Shortest Path First (OSPF). The list of protocol numbers is available from the Internet Assigned Numbers Authority (IANA) website [17]. The passive detection method therefore assumes that if a sender uses the protocol, then it must support it.

Protocols over the data link layer are identified in a similar manner but the fields of interest depend on the frame format. If the frame format is Ethernet II for instance, the network protocol appears in the 16-bit integer *Type* field of the Ethernet header. If the frame format is Ethernet 802.3, the *Type* field of the Ethernet II format is replaced with two 8-bit Service Access Point (SAP) fields found in the Logical Link Control (LLC) upper layer. A variance of the 802.3 frame format includes a Sub Network Access Protocol (SNAP) header on top of the LLC header. In this frame format, the *Type* field of the SNAP header is the same as the Type field of the Ethernet II frame. A list of assigned Ethernet codes and LLC Service Access Point identifiers can be found in [18]. Examples of protocols that can be monitored in this manner are IP, AppleTalk, Spanning Tree Protocol, Cisco Discovery Protocol, and Internetwork Packet eXchange.

As a limitation, protocols in use by a host may be passively discovered provided that the host makes use of them. Moreover, because the protocol discovery mechanism described above is based on a *Singleton* type of test, the technique can be sensitive to false positives due to packet crafting. Note that as a rule of thumb, if a host appears to support a large number of different protocols over IP, it can be assumed that this host is running a protocol scanner tool such as *nmap*. This is because active protocol scanning techniques over IP consist in sending a massive amount of crafted IP datagrams to the target, each having a different *Protocol* field value.

3.2.4. Service Discovery

Determining what services a machine provides can help network administrators identify prohibited or potentially vulnerable network applications. A TCP or UDP port number can often be mapped to a specific network service [15], particularly for well-known services. For instance UDP port 53 is usually associated with the Domain Name System (DNS). Therefore detecting which TCP/UDP ports are opened (listening) can reveal what networked services are likely hosted on a workstation or server. A number of active techniques have been developed for surveying the ports on which a machine is listening. The basic active technique to determine whether or not a TCP port is open consists in initiating a connection to that port with a TCP SYN packet. A successful connection indicates that the port is open. Because UDP is a connectionless protocol, the technique is a little different and consists in sending a UDP packet to the target port and wait to see whether an ICMP Port-Unreachable message is returned. If such a message is returned, it can be concluded that the UDP port is not open. Tools such as *nmap* [8] offer different port scanning methods to provide the user with the possibility of hiding its activity or defeating firewall rules. A port scan however can generate a significant amount of traffic since it requires sending as many packets to a target as there are port numbers to scan. In fact, of all the active gathering techniques that accumulate information about a host, the port scan is probably the one that generates the most packet traffic.

Fortunately, listening ports can also be identified passively by monitoring communications on the network. A service can thereby be detected, provided a client makes a request for it and succeeds.

TCP open ports are passively identified by monitoring connection set-ups. When a SYN/ACK is issued in response to a SYN, the source port of the TCP header field of the SYN/ACK packet can be assumed open. There are exceptions to this simple rule of thumb. One example of a situation that breaks the rule is with FTP transfer. When a client uses FTP to transfer one file, there are two concurrent TCP sessions: a communication channel and a data channel. When the data channel is being established, the source port of the SYN/ACK packet is ephemeral. The port will go back to the closed state once the connection terminates. There are other examples in which the source port of a SYN/ACK packet is not necessarily tied to a network service. Consider for instance a TCP wrapper configured to respond on behalf of a turned off service [16]. In most cases however, simple mechanisms can be implemented to rule out whether or not such a port is tied to a service. This is especially true if the program has the capability to match packets belonging to the same communication.

Passively discovering UDP ports is a little more difficult because there is no synchronization mechanism in the UDP protocol (i.e. no SYN-like packet). It can be assumed however that a UDP port number less than 1024 and transmitting packets is opened. Note that while this technique cannot discover open ports higher than 1023, communications that involve ports in the ephemeral range in both directions can be flagged and both ports can be identified as potentially hosting network services.

3.2.5. Switch and Router Discovery

Switches and routers are considered critical assets since they compose the infrastructure of the communication network. When problems occur, they typically get higher priority service responses compared to other nodes.

Security analysts monitoring a network are typically aware of the network and hardware addresses of routers and switches. Having the capability to discover these devices passively is still very useful in a dynamic environment as this can help identify newly added devices, whether they were preauthorized or not. While any unauthorised host must be detected, those trying to impersonate core elements are potentially more threatening. They typically attempt to disrupt services or redirect traffic to an attacker's system.

One way of recognising switches and routers is by monitoring traffic that typically comes from these devices. There are specific protocols that routers and switches use to communicate with their peers. The switches operate at layer 2, forwarding the traffic based on the hardware addresses of the hosts. They are rather transparent to other nodes; they themselves have little contact with their peers. Typically, they communicate with their neighbours when determining the spanning tree, which identifies the unique forwarding path for each switch. This is done using the Spanning Tree Protocol (STP) [19]. Monitoring this traffic will help identify the addresses of the switches.

The routers, which forward packets from subnets to subnets, operate based on network addresses. When configured to perform dynamic routing, they communicate with their peers through particular protocols. To name a few, there is the Open Shortest Path First (OSPF) routing protocol which is carried over IP (protocol 0x59), and the Routing Information Protocol (RIP) carried in UDP port 520. There are also certain ICMP messages that should only be seen as coming from a router (or any host acting as a router). This is the case for ICMP Host and Network Unreachable Errors, ICMP Host and Network Destination administratively prohibited Errors, ICMP fragmentation needed but DF bit set, ICMP Router advertisement, ICMP TTL-exceeded messages, or ICMP redirect messages.

Another technique to recognize routers passively consists in keeping track of associations between hardware and IP addresses. The source hardware address contained in the data-link layer of packets with a non-local source IP address can be recognised as the hardware address of a router. The IP address of the router can then be retrieved from the look-up table built from monitoring ARP requests. This of course can only be done if the range of IP addresses of local hosts is known. If not, identifying which hardware address has multiple IP addresses can help identify routers. Actually, even in cases where the IP address range is known, one should always keep an eye on hardware addresses associated with multiple IP addresses as this can very well indicate an ARP cache poisoning[5] situation with a man in the middle acting as a router (with IP Forwarding capabilities).

3.2.6. IP Host Configuration

This section describes ways to determine the netmask and the designated gateways each IP address is configured with. This provides information regarding a network's lay out and can help identify misconfigured systems.

The netmask identifies the number of bits in an IP address corresponding to the subnet number. If the netmask of a host is entered such that it incorrectly specifies a subnet range larger than what the subnet range really is, the host may not be able to communicate with some systems located on other subnets. This is because the sender would attempt to communicate with the remote systems directly. If on the opposite the netmask incorrectly defines a subnet range too small, then the host will send packets intended for local systems to the router. In this case the packets will reach their destination because the router will send them to the intended system. The router solicitation in this scenario is however unnecessary and introduces delay and additional traffic.

A technique that can help identify the netmask of a host is illustrated here with an example. Suppose a Class B IPv4 network 128.1.0.0 is further subdivided into subnets. Suppose one of these subnets has a range of IP addresses between 128.1.64.0 and 128.1.127.255. The first IP address in a subnet range is known as the Network Address and the last one is the Subnet Directed Broadcast Address.

The netmask of IP addresses that belong to this subnet is 255.255.192.0 (or 11111111.11111111.11000000.00000000 in binary), which indicates that the subnet number is 18 bits long (i.e. the first 18 bits of any two IP addresses in this subnet are identical).

Suppose the subnet break down scheme for that network is unknown and that a communication from IP 128.1.65.3 on that network is observed. ARP traffic can be monitored to determine with more or less precision what the netmask of an IPv4 host might be. The idea is to keep track of the "highest" and "lowest" IP addresses with which a host has communicated using ARP (i.e. IP addresses the host tried to resolve). One can then do a bitwise comparison between the two to find the common prefix. To illustrate this, suppose the "lowest" and "highest" IP addresses this host has communicated with through ARP are 128.1.65.3[6] and 128.1.95.66 respectively. In binary, these addresses are 10000000.00000001.01000001.00000011 and 10000000.00000001.01011111.01000010 respectively. The bits that are underlined are the contiguous identical bits. The estimated netmask is therefore 11111111.11111111.11100000.00000000 (i.e. 255.255.224.0, which over estimates[7] the true value 255.255.192.0). The estimate will obviously get more

[5] Cache poisoning refers to the act of updating a target computer's cache with misleading forged entries.

[6] Since this is the host's own IP address, this means the IP address this host has communicated with (using ARP) all had IP addresses greater than his.

[7] An estimated value greater than what the netmask really is means that the network range is underestimated.

precise as the lower and upper limits get closer to the true bounds delimited by the netmask (128.1.64.0 and 128.1.127.255 in the example above). The technique is accurate if the hosts in a subnet are widely dispersed in the subnet's address space. It will lead to an upper bound otherwise.

On Microsoft-centric networks the hosts will regularly transmit NetBIOS name service (netbios-ns) packets to their subnet broadcast address. For the example above, this means that the host 128.1.65.3 would send on a regular basis netbios-ns packet to the subnet broadcast address 128.1.127.255. This simplifies the task of determining the netmask since the highest bound of the subnet address range is known for sure. More generally, IP datagrams for which the destination hardware address of the data-link layer is the broadcast address can be monitored to discover IP addresses delimiting the subnet range.

Once the netmask is known, the routers to which a host is configured to forward packets can be determined. Any IP datagram this host will send to non-local IP addresses will be directed to the hardware address of a designated gateway. This gateway is either a default gateway or a gateway specified through a static route in the host's network configuration. Once the hardware address of the router is known, its IP address can then be retrieved from the look-up table built from monitoring ARP requests.

The derived IP configuration of local hosts can be compared. The actual netmask of the subnet can be elected and misconfigured hosts can be identified.

3.3. Pulling It Together

The strength of the approach is in combining the techniques to accumulate information. Tests are performed separately and the outcomes are combined to provide a coherent description of a network component. The prototype can retrieve, with a minimal number of monitored packets, key information about computers located on the network. As an example, much information can be retrieved from the three-way handshake of a client/server TCP connection. Suppose that the client does not know the server's hardware address and that both systems are located on the same LAN. Furthermore, suppose that a sensor can monitor the traffic on at least one of the two segments on which the hosts are located. When the client initiates the connection, it sends an ARP request packet to the server. From this packet, the client is known to be active. Hints about its operating system family and IP configuration are also collected. The server responds with an ARP reply. At this point, the server is known to be active and information about some parts of its IP configuration is gathered. The client then sends a TCP SYN packet to the server. By combining the information from the TCP SYN packet with the previously monitored ARP request, a more precise description of the host operating system family and version can be drawn. The server responds with a TCP SYN-ACK packet to the client. At this point, the operating system family group of the server can be inferred based on the combination of the SYN and the SYN-ACK packet. The server's response also determines that the server offers a service on the port requested by the client. The client then sends a TCP ACK packet to the server. By combining information from this packet with the SYN packet, it may also be possible to infer the system uptime of the client. More information will be inferred from these two computers throughout the remainder of the session.

4. COMPLEMENTARY WORK

The Network Security Research Group is concurrently developing correlation functionalities to integrate the information obtained from network profiling techniques with events produced by traditional firewalls and IDS devices, and with information contained in vulnerability databases. This correlation can help reduce the burden on network security analyst furthermore by filtering and processing part of the information

automatically. A prototype for correlating such information is under development to investigate the feasibility and reliability of this approach. The team is also examining the possibility of using formal methods to assess the threat coverage provided by a given sensor deployment strategy. This will help identify optimum placements of the sensors.

5. CONCLUDING DISCUSSIONS

This paper has described the importance of passive techniques to provide security analysts the context required to make informed decisions.

Host discovery can help detect unauthorized nodes connected to the network. Detecting system reboots can help identify problematic systems. Examining IP configurations can help determine network layout and identify misconfigured hosts. Identification of operating systems, services, and protocols can reveal security vulnerabilities and non-compliance with policies. Information about the role carried by a host can help identify critical assets or malicious behaviours that can potentially disrupt normal operation of the network.

The passive approach avoids affecting the network operations and yet can provide a comprehensive picture of the network's security posture. It allows constant awareness of ever-changing networks and helps network administrators remain in control and anticipate security problems. While the mechanisms described primarily focus on profiling IPv4 hosts, most can easily be adapted to support the IPv6 generation.

It is reasonable to believe that organizations will benefit from using passive monitoring techniques. The additional knowledge they provide can leverage existing investment in other security products. In particular, they allow network administrators to better interpret intrusion detection events by providing significant contextual information.

REFERENCES

[1] F. Massicotte, T. Whalen, .C. Bilodeau, "Network Mapping Tool for Real-Time Security Analysis". In Proceeding of NATO/RTO Information Systems Technology Panel Symposium on Real Time Intrusion Detection, Estoril, Portugal, May 2002.

[2] RNA, Real-Time Network Awareness by Sourcefire. Product information can be found at http://www.sourcefire.com/products/rna.html

[3] NeVO version 1.0, Network Vulnerability Observer by Tenable. Product information can be found at http://www.tenablesecurity.com/nevo.html

[4] SecurityFocus Bugtraq, [DDI-1013] Buffer Overflow in Samba allows remote root compromise, April 7, 2003. Security Advisory available at http://www.securityfocus.com/archive/1/317615/2003-04-04/2003-04-10/0

[5] J. Nazario, "Passive System fingerprinting using Network Client Applications", November 27, 2000. Available: http://www.crimelabs.net/docs/passive.pdf

[6] SecurityFocus news, Blaster's Microsoft Attack Fizzles, August 15 2003, available at http://www.securityfocus.com/news/6736

[7] B. McDanel, Beyond Security Ltd, "TCP Timestamping - Obtaining System Uptime Remotely", March 2001. Article can be found at the SecuriTeam.com web site http://www.securiteam.com/securitynews/5NP0C153PI.html

[8] F. Yarochkin, "Remote OS detection via TCP/IP Stack FingerPrinting", October 1998. Document and nmap program available at www.insecure.org/nmap

[9] O. Arkin, F. Yarochkin, "X remote ICMP based OS fingerprinting techniques", August 2001. Document and xprobe program available at http://www.sys-security.com/html/projects/X.html

[10] Foundstone Inc., FoundScan Engine. Product information available from: http://www.foundstone.com/

[11] F. Veysset, O. Courtay, O. Heen, New Tool and Technique for remote Operating System Fingerprinting, April 2002. This document and the *ring* program were available originally at www.intranode.com/pdf/techno/. Any reference to *ring* has now disappeared from this site; however, a new version of ring (Cron-Os) can be obtained from http://cron-os.tuxfamily.org/

[12] "concept", OS Detection with ARP, Napalm e-zine, Issue 6, July 2000. Article can be found at http://www.iwar.org.uk/hackers/resources/napalm-mag/napalm6.htm. induce-arp, the tool using these techniques can be downloaded from several sites, in particular from http://www.packetstormsecurity.org/ UNIX/misc/induce-arp.tgz.

[13] p0f program, a passive OS fingerprinting tool by Michal Zalewski. The tool can be downloaded from http://lcamtuf.coredump.cx/p0f.shtml

[14] Ettercap program, a multipurpose sniffer/interceptor/ logger for switched LAN. It supports several active and passive features for network and host analysis. The tool can be downloaded from http://ettercap.sourceforge.net/.

[15] The Internet Assigned Numbers Authority (IANA) assigns and maintains a list of well-known port numbers for TCP and UDP. This list is available at http://www.iana.org/assignments/port-numbers.

[16] Lance Spitzner, Intrusion Detection, Knowing when someone is knocking on your door. Document available at http://www.spitzner.net/ids.html.

[17] The Internet Assigned Numbers Authority (IANA) houses a list of Protocols numbers over IP. This list is available at http://www.iana.org/assignments/protocol-numbers.

[18] Ethernet Type codes and Logical Link Control addresses are issued by the IEEE. Several websites maintain list of assigned identifiers; in particular, both LLC and Ethernet numbers can be found at http://www.wildpackets.com/compendium/REF/L1-Refrn.html

[19] A. S. Tanenbaum, Computer Networks, third edition, Prentice Hall, Upper Saddle River, New Jersey, 1996.

Monitoring of Network Topology Dynamics

Dr. Vladimir Gudkov / Dr. Joseph E. Johnson
Department of Physics and Astronomy
University of South Carolina,
Columbia, SC 29208
USA
Email: gudkov@sc.edu / jjohnson@sc.edu

Mr. Rajesh Madamanchi
Department of Computer Science and Engineering
University of South Carolina,
Columbia, SC 29208
USA
Email: madamanc@engr.sc.edu

Mr. James L. Sidoran
Air Force Research Laboratory
Defensive Information Warfare
Rome, NY 13441-4505
USA
Email: James.Sidoran@rl.af.mil

ABSTRACT:

We present software for deriving innovative metrics describing dominant parts of the internal structure of large networks. The algorithm is sufficiently fast for the network metrics to support real time monitoring of network dynamics. The network connections (connectivity matrix) are mathematically constructed by capturing the appropriate header parameters of selected internet/network traffic. Our metrics are in part derived from a network cluster decomposition that is based upon a physical model analogy for the network that is very rapidly evolved revealing cluster structures. Certain of the metrics consist in part of Renyi (generalized Shannon) entropy measures on the resulting network clusters and subclusters. This evolution depends upon the connectivity matrix and revels many qualitative features of the network.

1.0 INTRODUCTION

Network systems have very complex structures and temporal behavior because of their intensive nonlinear and convoluted information dynamics. Therefore, recent advances in the mathematical physics of complex systems, as well as computational biology, for the modeling and simulation of complex systems could provide the framework and scale level needed for the development of a quantitative foundation for cyberspace systems.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Existing approaches for the study of network information traffic usually include the study of the dependence of network stability in terms of network complexity and topology (see, for example [1,2] and references therein); signature-based analysis techniques; and statistical analysis and modeling of network traffic (see, for example [3-6]). Recently, methods have been proposed to study both spatial traffic flows [7], and correlation functions of irregular sequences of numbers occurring in the operation of computer networks [8].

In this paper we describe the developed network monitoring technique based on new approaches [9-12]: one for reconstruction of network topology, using a physics analogy of the motion of particles in a liquid medium in a multi-dimensional space, and another for rapid monitoring of topology dynamics, using generalized entropies.

2.0 ALGORITHM DESCRIPTION

For network cluster decomposition we use an algorithm based on an analogue physical model which is dynamically evolved. The detailed algorithm description is given in papers [9-11]. Here we recall the main features of the algorithm. To describe the connectivity of a network with N nodes we use the connectivity matrix C ($n \times n$), refereed to in graph theory as the adjacency matrix. We allow matrix elements C_{ij} to be only 0 or 1 - for disconnected and connected nodes i and j, correspondingly. It should be noted that if two matrices differ only by the labeling of the vertices, effecting the permutation of rows (columns), they represent the same network.

The number of C matrices representing the same network is equal to $2^{\frac{n(n-1)}{2}}$ and is very large already for moderate size of network. We are looking for the unique matrix C which has block-diagonal structure representing the active dynamically connected groups of node on the given network. To solve this problem avoiding large number ($n!$) of operations in combinatorial approach, we use a completely symmetric and unbiased initial configuration which does not depend on the numbering: place the n nodes of the network at the n vertices of a symmetric simplex inscribed inside the unit sphere in $n-1$ dimensions. All vertices are equidistant from the origin and from each other.

The distance between any pair of vertices of the simplex i.e. between any pair of the nodes is, therefore:

$$| \vec{r}_i - \vec{r}_j | = \sqrt{\frac{2n}{n-1}} \qquad \text{for all } i \neq j \quad i, j = 1...n, \tag{1}$$

We next consider the nodes as massive point-like particles and endow our system with some dynamics introducing an attractive force between points corresponding to nodes which are connected in the initial network of interest.

Thus we postulate linear forces

$$\vec{F}_{ij}(\vec{r}_i, \vec{r}_j) = g(C_{ij}) \frac{(\vec{r}_i - \vec{r}_j)}{| \vec{r}_i - \vec{r}_j |}, \tag{2}$$

where the $g(C_{ij})$ is the intensity of interactions as a function of the value of matrix element C_{ij}. To be the force attracting the point mass i to the point mass j, in the direction of $\vec{r}_i - \vec{r}_j$. To retain the initial symmetry and avoid any biasing we take the same force law for all pairs. Then the only way information about the specific graph of interest is communicated to our dynamical n body system is via the overall strengths $g(C_{ij})$ of the forces. In the presented program the default value of $g(C_{ij})$ is equal to zero if $C_{ij} = 0$.

Next let our point move according to first order "Aristotelian Dynamics"":

$$\mu_i \frac{d\vec{r}_i}{dt} = \vec{F}_i = \sum_j \vec{F}_{ij}, \tag{3}$$

which corresponds to particle motion in a very viscose liquid (this let us use more simple differential equations of first order instead of second order equations for standard Newtonian dynamics). To preserve the initial symmetry we take all viscosities μ_i=1.

With only attractive forces present our n point system eventually collapses towards the origin. A collapse of all n points happening before the vertices belonging to "clusters in the network"" have separately concentrated in different regions defeats our goal of identifying the latter clusters.
To avoid the radial collapse we constrain $\vec{r}_i$, to be at all times on the unit sphere.
While the above avoids the radial collapse, the residual tangential forces can still initiate a collapse at some point on the unit sphere. After a sufficient time (or sufficiently many steps in evaluation of our dynamic system) has elapsed so that any point moved on average an appreciable distance away from its initial location geometrical clusters of points tend to form. The points in each geometrical cluster correspond to the original vertices in a cluster of the network which these points represent. (We recall the definition of a cluster in the graph/network as a subset nodes with a higher number of connections between them than the average number of connections with ``external'' nodes, which are not in the cluster.)

Then, calculating the mutual distances between nodes, one can separate them into groups of the clusters. Therefore, scaling the parameter of the "critical" distance one can re-define clusters based on the intensity of connections, and to resolve sub-clusters of the clusters. It should be noted that definition of the function $g(C_{ij})$ gives the principal for a cluster definition: intensity of connections, e-mail exchange, etc.

For monitoring rapid changes of the network topology we calculate mutual entropies of the network (see for details refs.[10,12]). To do this we redefine the connectivity matrix in terms of probabilities of the connectivity in such way that each matrix element represents the probability that two nodes are connected to each other. Thus we normalize the connectivity matrix C so that

$$\sum_{i,j=1}^{n} C_{ij} = 1. \tag{4}$$

Then the sum over all columns $P_i = \sum_j^n C_{ij}$ can be considered as the probability of the connectivity for the node i, and the Shannon entropy

$$H(row) = -\sum_{j=1}^{n} P_i \log P_i \tag{5}$$

is a measure of the uncertainty of the connections for a given network. In the same way one can define the entropy for "inversed" connections: $H(column)$.(Due to symmetry of the connectivity matrix in our case, $H(row) = H(column)$.) The amount of mutual information (or negative entropy) gained via the given connectivity of the network is

$$\begin{aligned} I(C) &= H(row) + H(column) - H(column \mid row) \\ &= \sum_{i,j}^{n} C_{ij} \log(C_{ij} / P_i P_j), \end{aligned} \tag{6}$$

where

$$H(column \mid row) = -\sum_{i,j}^{n} C_{ij} \log(C_{ij}). \tag{7}$$

It should be noted that $I(C)$ does not depend on the vertex relabeling, and, as a consequence, this is a permutation invariant measure for the connectivity matrix. If the mutual entropies for two connectivity matrices are different, they represent different topological structures of network. This entropy is already sufficient to distinguish even between graphs that are normally cospectral.

The obvious extension of this definition of mutual Shannon entropy (information) could be used for calculations of mutual Rényi entropy . For example, based on definition of Rényi entropy [13], the expressions in eqs.(5) and (6) are
transformed into

$$H_q(row) = -\frac{1}{1-q} \log \sum_{j=1}^{n} P_i^q \tag{8}$$

and

$$H_q(column \mid row) = -\frac{1}{1-q} \sum_{i,j}^{n} \log(C^q{}_{ij}), \tag{9}$$

giving mutual information Rényi $I_q(C)$ for the given matrix C.

Using the Rényi information, one can not only distinguish between different network topologies on the base of the connectivity matrixes but extract information about network topology, such as number of clusters, cluster's dimensionalities etc. Moreover, by monitoring appropriate functions of mutual information, one can observe in real time a change in topology of the given network including a cluster formation, disappearance or appearance of group connections, change of the connection "styles", and other features.
For example, the difference between mutual Shannon information and Rényi information of kind 2 ($q = 2$) displays a sharp dependence of the size of the formed cluster (Figure 1).

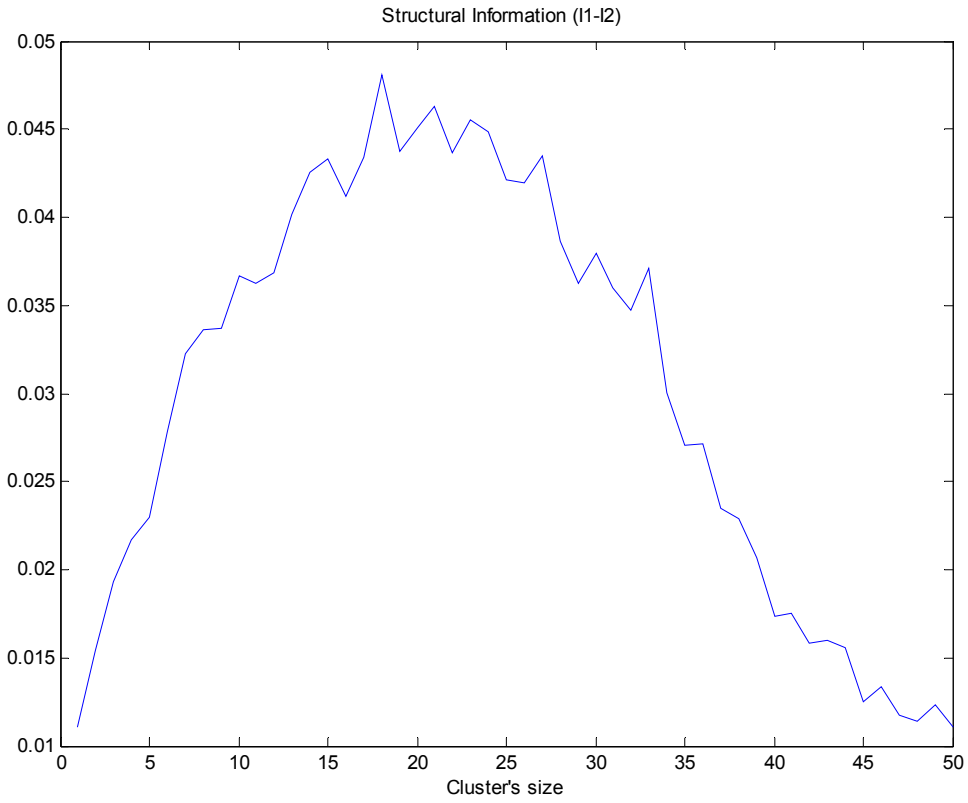


Figure 1: The difference between mutual Shannon information and Rényi information of kind 2.

This gives a unique opportunity to monitor dynamical behavior of the network in real time. It should be noted that different entropies are sensitive to different patterns of network topology (such a size of clusters, number of clusters, fractional dimensionality, etc), therefore many important properties of network can be extracted using suggested methods.

3.0 PROGRAM DESCRIPTION

Based on the described algorithms we have developed the real time network system "Ipcluster" for dynamical reconstruction and monitoring network communications. The program contains three modules: capture and connectivity matrix construction, analysis, and visualization.

The capture module is a program which puts the network card of the computer in promiscuous mode, captures the network packets and dumps the packets along with the specified headers parameters to a file (as a back-up). Then builds a connectivity matrix for all currently active nodes based on their IP addresses.

The analysis module applies the topology reconstruction (cluster identification) algorithm and calculates a set of Rényi entropies for the obtained connectivity matrix. In the current program the complexity of algorithms are and for the topology reconstruction and entropy calculations, correspondingly. Therefore,

we can separate the time intervals for topology analysis and entropy monitoring, since the last one can be done in much less time.

The visualization module (see the typical snap short on Figure 2) presents both results of topology reconstruction and entropy on separate windows.

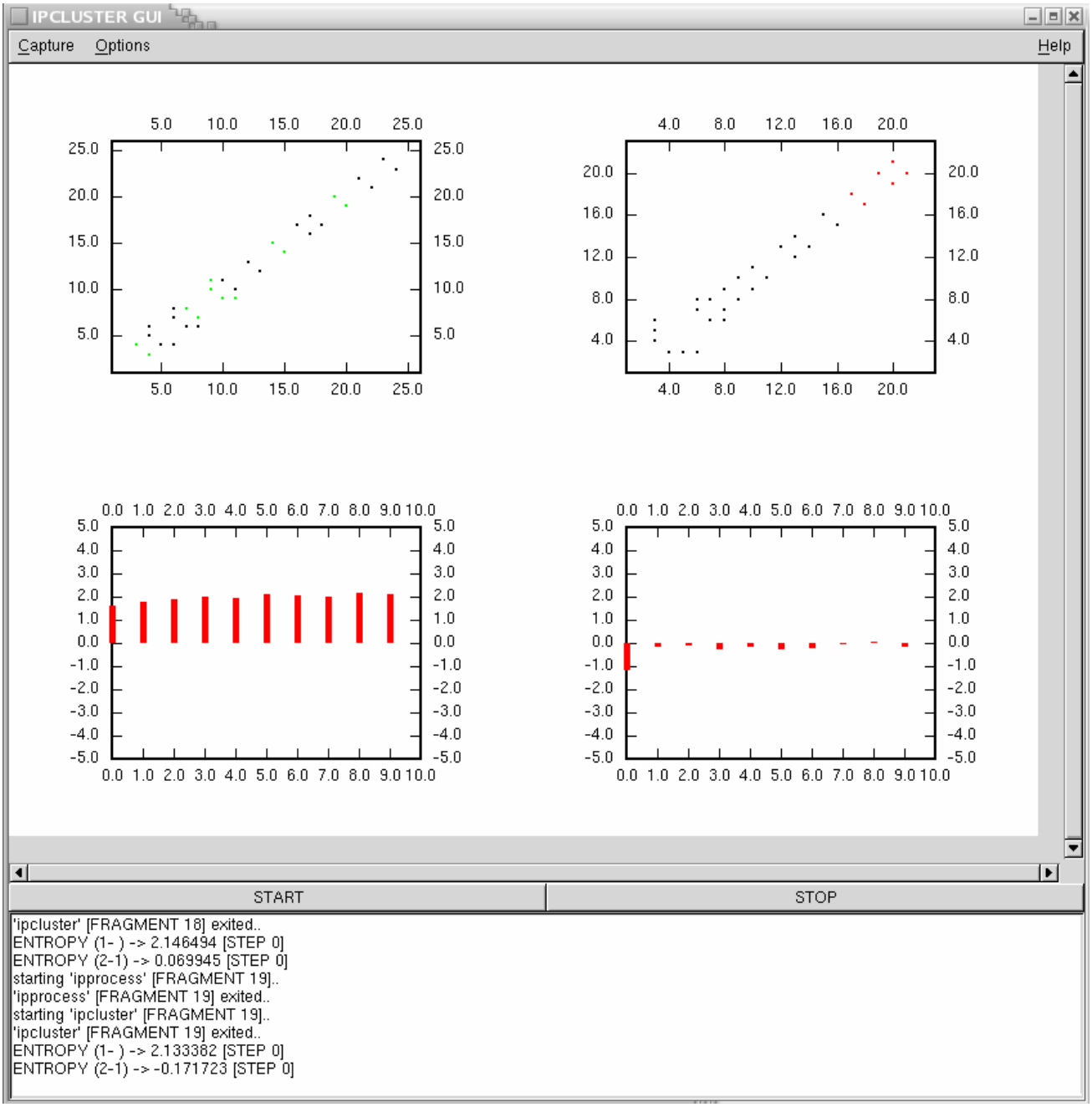


Figure 2: Visualization Module Snapshot

The two upper windows represent the reconstructed cluster structure of the network within the given time delay Δt. The cluster representation in each window uses different colors for continuously active connections during current Δt period; for connections which existed for the previous time interval, but disappeared for the current Δt; and for connections that just appeared for the current Δt period. Therefore, one can monitor topological changes on the network with the interval Δt.
Two lower windows present the histograms for chosen combinations of generalized mutual entropies as functions of time. It gives the opportunity to monitor particular topological structures in the network for which the given function of entropies is sensitive most. The upper windows are refreshed with the interval Δt, but the lower ones plot continuous histograms over time.

It should be noted that the visualization module dynamically analyze the reconstructed cluster structure for each Δt interval. By the end of each interval, it applies the reconstruction algorithm on the captured packets and displays the reconstructed topological structure. During the process of reconstruction, another thread to capture traffic packets for the next step is running. Therefore we do not lose any network traffic.

The current version of the program lets to choose a variety of options for definition of different actions as a connection, as frequency of connection, port of connection, protocol etc. Also we can vary time of monitoring and frequency of analysis, as well as different sets of mutual information to be visualized.

4.0 CONCLUSIONS

The presented program for real time network topology monitoring provides extremely fast method and confirms readability and efficiency of the approach developed in papers [9-13]. It may be used for real time monitoring of large networks and the internet. The presented algorithms may be also applied for dynamical monitoring and analysis of different kinds networks, for example communication networks, social networks etc. It provides quantitative method to define connected groups (clusters) one large networks with the ability to extract topology (structure and sub-structure of clusters) of the given network in real time with elements of visualization

BIBLIOGRAPHY

1. A. Reka, J. Hawoong and B. Albert-Laszlo, “Error and Attack Tolerance of Complex Networks”, Nature, Vol. 406, pp. 378-381, 2000.
2. S. H . Strogatz, “Exploring Complex Networks”, Nature, Vol. 410, pp. 268-276, 2000.
3. L. Deri L. and S. Suin, “Practical Network Security: Experiences with ntop”, Computer Networks, Vol.34, pp. 873-880, 2000.
4. P. A. Porras and A. Valdes, ``Live Traffic Analysis of TCP/IP Gateways", Internet, Society Symposium on Network and Distributed System Security, SanDiego, California March 11-13, 1998.
5. J. B. D. Cabrera, B. Ravichandram and R. K. Mehra, “Statistical Traffic Modeling for Network Intrusion Detection", Proceedings of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE, 2000.
6. T. Huisinga, R. Barlovic, W. Knospe, A. Schadschneider, and M. Schreckenberg, “A Microscopic Model for Packet Transport in the Internet”, arXiv:cond-mat/0102516, 2001.

7. N. G. Duffield and Grossglauser,"Trajectory Sampling for Direct Traffic Obsevation", MIEEE/ACM Transactions on Networking, Vol. 9, No 3, pp. 280-292, 2001.
8. M. Ayedemir, L. Bottomley, M. Coffin, C. Jeffries, P. Kiessler, K. Kumar, W. Ligon, J. Marin, A. Nilsson, J. McGovern, A. Rindos, K. Vu, S. Woolet, A. Zaglow, K. Zhu, "Two Tools for Network Traffic Analysis", Computer Networks, Vol. 36, pp.169-179, 2001.
9. V. Gudkov, J. E. Johnson and S. Nussinov, "Approaches to Network Classification", arXiv: cond-mat/0209111 (2002); submitted to Phys. Rev. E.
10. V. Gudkov and S. Nussinov, "Graph equivalence and characterization via a continuous evolution of a physical analog ", arXiv: cond-mat/0209112 (2002).
11. V. Gudkov, S. Nussinov and Z. Nussinov, "A Novel Approach Applied to the Largest Clique Problem", arXiv: cond-mat/0209419 (2002).
12. V. Gudkov and J. E. Johnson, "Applications of Generalized Entropies to Network Analysis", talk at the "Networks 2003", Santa Fe, NM (2003).
13. A. Rényi, "Probability Theory", North-Holland Pub. Co. – Amsterdam, London & American Elsevier Pub. Co., Inc.-New York (1970).

Information Exchange between Resilient and High-Threat Networks: Techniques for Threat Mitigation

Tim Dean and Graham Wyatt
QinetiQ – Trusted Information Management
Woodward Building B006,
St Andrews Road, Malvern,
WR14 3PS
United Kingdom

tbdean@qinetiq.com / gwyatt@qinetiq.com

SUMMARY

High resilience military networks frequently have requirements for exchange of information with networks of low assurance, including networks of unknown threat such as the Internet. Traditionally, the approach to solving this problem is an air-gap between the two domains, with information exchanged between them on floppy disk. This approach is both time-consuming and potentially risky. This paper proposes alternative techniques to enable assured, two-way, information flow between high resilience networks and other networks of unknown threat. The techniques include conventional and novel technologies designed to control and constrain information formats, manage the environment between domains with network level controls, and provide assured, user-instigated, release sanctions.

1.0 INTRODUCTION

Access to the Internet for e-mail and web access is now essential for military and government users. There are other, innumerable, unclassified or low classification systems with which military and government networks must also communicate, such as the Red Cross, the Red Crescent, police forces, other civil agencies, other government departments, and broad coalition networks.

There is inevitably a requirement to transfer information between such low classification systems to higher classification military systems that require higher assurance and a greater degree of resilience, such as command and control systems, logistics systems and intelligence networks. In such a context, Internet-connected systems are inevitably considered a high threat. Strict controls must be placed at the boundaries between these systems to prevent both the introduction of malicious content into the high system and the leakage of high data to the low system. *Note on terminology: from here on we refer to the two kinds of network as 'High' and 'Low Assurance'. Lower assurance arises due to the higher (or unknown) threat level, and in general is likely to lead to lower levels of resilience in applications.*

Historically, the security separation problem has been solved by total electronic separation between the low and high networks, sometimes referred to as an "air gap". But experience has shown that this is not always practical, as the low classification information may have high value in the high system – for example, weather data, situational awareness information, open-source intelligence, collaborative planning, and information from civil agencies.

There is a common requirement to enable a secure connection between these high and low networks in such a way that information can be exchanged without posing a threat to the resilience of the high network.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

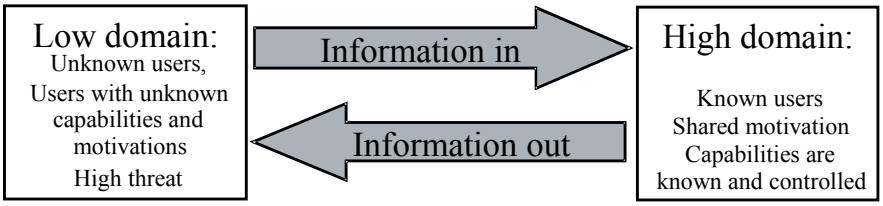


Figure 1 – Exchanging information between low and high domains

As a consequence, floppy disks, USB memory sticks and CD-RWs are often used to bridge the gap. But these techniques raise a number of risks and are inadequate to deal with modern threats that use multiple propagation mechanisms to gain access to networks. This paper will examine the threats of connecting low and high assurance domains, discuss some traditional methods of mitigating the risks posed by the threats and outline novel technical solutions that enable information exchange in particularly high-threat scenarios.

2.0 THE TECHNICAL THREAT

Some of the dangers posed to a high assurance domain are reported widely and frequently. For example: attacks against Internet facing web servers to extract client credit card details; e-mail arriving from the Internet carrying viruses that infect a business' Intranet; or more recently, the emergence of 'phishing' where the attacker masquerades as an on-line retailer or bank, sending e-mails to customers inviting them to click on a web hyper-link under the pretext of performing a necessary administrative task. The attacker, who owns the web site, can then harvest the usernames, passwords and credit card details of the victims and use these credentials to gain access to the high assurance system.

Attacks can be categorised in many ways, using terms such as:

- Viruses – malicious code that replicates itself to other host programs; areas of memory; disk boot sectors; or macro capable documents. Can also execute a malicious payload.
- Worms – malicious code that makes copies of itself and can exploit program vulnerabilities to propagate or can include the propagation mechanism within the code.
- Trojan Horses – a program that does not replicate itself but can damage the host computer or use the host to launch further attacks, often under the direct control of the attacker.
- Other kinds of attacks include: back doors, rootkits, BIOS and Microcode malware, social engineering attacks, and buffer overflows.

Anyone who reads the technical press will be familiar with examples of the above attacks and may well have been subjected to some of them. However the attacks that get reported are generally the opportunistic type that become global phenomena, such as variants of the Sobig or LoveBug viruses. What is less reported and less understood are the direct, targeted attacks. It is difficult to know if their apparent lack of frequency is due to their rarity, or the unwillingness of organisations to discuss such attacks publicly, or, more worryingly, their success (i.e. the attack succeeds and is so carefully disguised, it is never discovered).

The scattergun approach taken by opportunistic attackers is a time-consuming and troublesome nuisance to system administrators who must secure their Internet facing networks from the attacks. When these networks are connected to an affiliated high domain with high-resilience requirements, the high domain administrators must deal with the same threats posed by the opportunistic attackers, as well as potentially more devastating targeted attacks launched by skilled, motivated and highly funded attackers.

It is well known that the majority of attacks on the Internet are from relatively unskilled attackers making use of publicly available tools and code that exploit known vulnerabilities, for which manufacturers' patches are commonly available. A properly resourced attacker could discover new vulnerabilities, develop new exploits, and attack a network using these novel methods that would avoid detection by commercial intrusion detection and filtering systems. Such systems generally detect only known recognised patterns, or signatures, of attack.

A successful attack will compromise one or more elements of the information security trinity: i.e. information confidentiality, integrity and availability.

- Confidentiality – Information within the high domain should remain within the high domain unless its release to a lower domain is authorised and appropriate. For example, personal information relating to the clients of a bank must remain confidential and not be leaked to the Internet, either accidentally or deliberately.
- Integrity – Information within the high domain should remain uncorrupted. An integrity attack might lead, for example, to a message 'Credit Joe Bloggs £3000.00' being changed to 'Debit Joe Bloggs £3000.00.'
- Availability – Information services within the high domain must remain available. A well-known availability attack is the Distributed Denial of Service attack that has affected many Internet facing companies such as on-line banks and retailers. The servers of these companies are bombarded with bogus requests from thousands of computers infected with trojans controlled by the attacker. Valid user requests are unable to reach the server due to the overwhelming quantity of bogus traffic.

Analysing some recent, successful and well-publicised attacks reveals a new trend for *combination malware*, or to put it another way, malicious code that:

- Has combined characteristics of virus and worm for propagation, using mobile code (active content) that usually requires some manner of user interaction, as well as an element involving network attack that can happen automatically.
- Can embed itself into a system as a Trojan horse.
- Can add a back door allowing a two-way communication channel back to 'base' (or more likely, a web site or IRC chat room in the attacker's control) where commands or updates to the virus code can be posted.

An interesting point about such code is the diverse range of attack vectors being used for propagation. Methods include direct communication using TCP/IP, application channels such as SMTP, and corrupted application data, all being used in combination. Also of note is the fact that this kind of attack can propagate using floppy disks and CDs, therefore crossing what are perceived as "air gaps" to threaten the resilience of critical networks.

The recent Sobig.F (August 2003) attack exhibited many such characteristics. It used several propagation techniques, including spreading via e-mail attachments and network shares, and it included an embedded SMTP engine. The payload was an URL downloader which, in effect, meant that the payload was infinitely variable, being dependent upon the imagination and talent of the attacker to create their preferred attack and post the code to the web site from which the virus was pre-programmed to seek the download. The most obvious effect of the Sobig.F attack was denial of service through network flooding and e-mail system overload.

Furthermore, the number of malware attacks reported is increasing at an alarming rate. For example, CERT reports the following figures:

- 21,756 viruses reported in 2000
- 114,855 viruses reported in 2003 (to October)[1]

The propagation speed is also increasing as the number of attack vectors has increased, and as the time between the announcement of a vulnerability and the associated virus release has narrowed, sometimes to a few hours.

Yet, as previously discussed, these public statistics largely overlook the issue of targeted attacks, the prevalence and effects of which are almost completely unknown. Unfortunately, the threat of targeted attacks is the key concern to military networks.

3.0 SECURING THE BOUNDARY BETWEEN HIGH AND LOW ASSURANCE DOMAINS: COMBINING TRADITIONAL AND NOVEL COUNTER-MEASURES

As part of the research on behalf of the United Kingdom Ministry of Defence, QinetiQ has been examining how traditional and novel technologies can be employed to enable a controlled, bi-directional exchange of information between high and low domains.

We define three categories of techniques that can be used to control information flow:

- Format control
- Environment control
- User control and release sanctions

3.1 Format Control

Data format control techniques involve three related processes: (i) using 'safe' formats, (ii) format conversion - where data is transformed from one format to another format that obeys a different format 'grammar' and 'syntax' and (iii) checking that the formatting rules have been obeyed.

3.1.1 'Safe' Formats

The dangers of complex data formats are well understood and documented. For example, HTML used in web pages and e-mail can contain active content described with a scripting language that, by default, is processed and rendered by the client machine. Many attacks exploit vulnerabilities in the script interpreter or the host application to run malicious and damaging program code[3]. Similarly, complex formats used to describe documents created with a word processor can contain macros that again perform malicious actions. Word processor formats also allow for 'hidden content', where multiple versions of the document are embedded within the document description, containing data previously deleted by the user and not visible on the screen, but still encoded within the file[2].

Formats that ban or tightly constrain active content are safer and are in general to be preferred. One example is earlier versions of PDF (PostScript Distribution Format), which tightly constrained the use of scripting, banning dangerous functions such as general file access. However, these cannot be regarded as entirely safe since any complex language allows for the possibility of malformed data structures; this can produce unexpected behaviour in end systems, such as denial of service or buffer overflow attacks, which in effect convert a passive data structure into active code.

PDF suffers from such problems, and there have been some documented examples of such abuse[4]. Complex protocol specification languages such as ASN.1 (Abstract Syntax Notation One) also suffer similar problems. Projects such as Oulu University's Protos toolkit[5] have shown just how vulnerable ASN.1-based protocols can be to attack[6].

An approach to solving these problems is to write document viewers and protocol implementations that are carefully crafted and exhaustively tested. In general, industry is increasingly recognising the need for this, but complex languages cause the data-space requiring testing to be huge, or unbounded. Therefore, the assurance of such applications is limited, and something further is required in high assurance environments. A 'safe', or at least safer, format should be incapable of such (albeit unintended) subterfuge, using simple data structures with a well-defined syntax and grammar with a finite space.

Two obvious examples of simple formats are ASCII text and (certain kinds of) Bitmap images. Their simplicity aids another part of format control, the format checking. But before that happens, the data formats must be converted. This is discussed in the next section.

3.1.2 Format Conversion

The simple idea behind format conversion is that by transforming the representation of data from one form to another, possibly with several iterations, any malware capabilities contained in the original information format would be lost, especially if one of the transformation processes introduces an unpredictable, random element to the conversion process.

There are two stages at which format conversion might be used in a high assurance scenario explored in this paper.

First, many complex formats can be converted into ASCII text or Bitmap images. Every time we use the 'Print Screen' function in Windows (Press Alt key and Print Screen key together) we convert whatever is on the screen into a Bitmap image (which is then copied to the Windows clipboard). To take one example, if we take a Bitmap image of a Word document, all the informational content remains (we can still read the document), however all the hidden content encoded in the complex Word format is removed, along with the associated threat posed by that hidden content.

There are a number of commercial programs that can be used to take Bitmap images of complex documents (in HTML, Word, Excel, Adobe Acrobat, and other formats) in a more sophisticated manner than a simple 'Print Screen' request. For example, these 'screen-scrape' programs can open a Word document so that the entire document is converted to a Bitmap image, not just the single page visible on the screen.

In addition, some of the screen-scrape programs incorporate a 'text-scrape' function where the text within a Word or Adobe formatted document can be converted to ASCII text, again removing any hidden content (though at this stage any text-scraped scripting code, for example, may be preserved).

It is worth differentiating between the effect of a screen and text scrape. The text scrape takes a document and translates the complex encoding used by the word processor to represent letters of the alphabet and translates that into simple ASCII encoding. However both encodings are used to represent letters of the alphabet. Converting a word-processed document into an image profoundly alters the encoding; while the word processor is encoding letters of the alphabet, the image is encoding the colour and brightness of pixels in the image. It is only in the viewer's brain that those encodings are translated back into text.

The second time a format conversion might be performed is when releasing Bitmap images, just after the format checking is completed. At this stage, the Bitmap image could be transformed into a JPEG image.

The transformation process, using a lossy compression algorithm like JPEG, would mean that while the information content would remain the same (the JPEG and Bitmap images would appear almost identical to the viewer), the data format would undergo a transformation.

This second transformation would make it even less likely that an opportunistic attacker's hidden or malicious content would be preserved, particularly given it may be irretrievably lost thanks to the lossy compression. Compression is of course doubly desirable given that bandwidth restrictions would make the exchange of large Bitmap files difficult to scale to a large user community.

To defeat the targeted attacker it would also be possible to introduce some random alterations to the Bitmap image prior to the (optional) JPEG conversion; for example, by making small, random modifications to the brightness values of some, or all, of the pixels in the Bitmap, and even to the image dimensions by adding a randomly sized border to the image. The changes would be near undetectable to the viewer of the final image, but make it very difficult for an expert attacker to predict the exact output of the conversion and so design an attack to exploit a hypothetical vulnerability in the recipient's image viewer.

Crucially, the initial screen or text scrape conversion process does *not* need to be trusted or assured. What *must* be assured is the optional process to randomise Bitmaps and the format checking stage, described below.

3.1.3 Format Checking

There are an increasing number of products that are designed to check complex formats such as HTML, XML, Word, or Excel for hidden content, malicious macros and the like. Such checkers are extremely useful and valuable in scenarios where this richness of information transfer is permissible. They can use simple heuristics such as ensuring that Word documents have not been "fast saved". But simple heuristics are not adequate to address the kind of threats discussed above.

For these situations, much more constrained data formats are required. If only two simple information formats such as ASCII text and Bitmap images are allowed to be exchanged between the two domains, the scale of the problem is much reduced.

By only permitting ASCII text messages and the simplest variety of bitmap images to pass between domains there is a significant reduction in the complexity of the content checkers. For example, an ASCII text checker would consist of one very simple function that would eliminate from the message any characters that were not from a recognised alpha-numeric list or from a small subset of permitted punctuation markings. The punctuation could even be replaced by words, as in old-fashioned telegrams (STOP for fullstop). All other content would raise a warning and be removed from the message before being passed on. Therefore if the message contained Javascript, the opening and closing brackets ('<' '>') used to denote a 'tag' would be removed, rendering the Javascript incomprehensible to the receiving application. An optional, simple ASCII text search could warn of the presence of blacklisted words.

A Bitmap checker would be simpler still, as long as the simplest kind of raw Bitmap format is used exclusively. These simple bitmaps are a rudimentary image format to parse, containing a definition of the dimension of the image (numbers of pixels across and high), followed by the colour parameters for each of the pixels (defined in terms of the strength, from 0 to 255, of the Red, Green and Blue constituent colours). The Bitmap format checker would perform a simple grammar and syntax check of each image so that each file was known to conform to these Bitmap encoding rules.

3.2 Environment Control

The techniques of format control discussed above must be carefully managed and protected to ensure that they cannot be subverted. Here we specifically refer to the environment at the interface between the high and low assurance domains. Traditional techniques to control the interface environment include firewalls, one-way data diodes (where information can only flow in one direction, for example from a low to high domain), and the use of De-Militarised Zones, generally bounded by two firewalls or data diodes (or both).

To construct such a controlled environment, QinetiQ has used commercial off-the-shelf components, some of which have formal (EAL) assurance rating. Some of the components are QinetiQ products, such as the one-way data diode (SyBard/Diode)[1] and the SWIPSY® firewall toolkit. The SWIPSY toolkit is an E3 (equivalent to EAL4) evaluated product that allows additional code to be added to its security 'compartments' without affecting the evaluation status of the toolkit itself. SWIPSY has security properties that assure network and process separation: processes communicating with one network can not communicate directly with the other network other than by 'trusted mover agents' that in turn force data to be passed to the format and content checkers.

Environment controls are insufficient individually to control the new generation of malware, as the controls typically defend at the network level. However, when combined with application level 'checker' software and the novel techniques discussed in this paper, they are vital in forcing the data through the checking processes. A number of other mechanisms must also be used if high assurance is required. These include assured user sanction, and potentially, machine virtualisation and these are discussed in the following subsections.

3.3 User Sanction and Control

The assured intervention of a human user is critical for the release of data from a high to low domain for two reasons. First, if every exchange of information from high to low is governed by user sanction, the process can be meaningfully audited, making the users accountable for their actions. Secondly, a properly implemented and assured user sanction mechanism would prevent any high machine infected by a back-door attack from communicating with the low domain by any other route than the user sanctioned channel. With this restriction on information flow in place, as well as the format checking and format conversion, it becomes a significantly challenging task for the back door program to communicate with the low domain attacker without arousing the suspicions of the sanctioning user.

The software used to implement the user sanction process should be simple and trusted. First, the data should be presented to the user for approval and release. This data should be confined to simple formats such as ASCII text or Bitmap images as previously discussed. Second, the user could, optionally, instruct the software to digitally sign the data to prevent modifications between the user's desktop and the domain boundary. The user must be able to view everything on the screen that they are about to sign. This implies that a trusted viewer is used, i.e. an assured computer that the user can trust will display the complete contents of the message and that this viewer and signer can not be subverted by any attack.

It is worth mentioning that the idea of a trusted signing device is a recognised requirement in some civil applications. The EU Digital Signature directive[7] recognises the need for a Secure Signature Creation Device (SSCD), and some EU member states have enshrined this requirement in their legislation. In addition, industry seems to be following a similar trend: the Trusted Computing Group's (TCG)[8] Trusted Platform Module (TPM)[9] specification includes "Data Attestation" facilities for signing data structures. The most recently published documents describing Microsoft's Next Generation Computing Base (formerly Palladium) detail aspirations for both trusted viewers and trusted signing mechanisms, as part of the trusted part of the operating system[10].

[1] **Part of the QinetiQ SyBard::Suite®. SyBard::Suite and SWIPSY are registered trademarks of QinetiQ Ltd.**

4.0 HIGH ASSURANCE PROTOTYPE – SENDING INFORMATION FROM HIGH TO LOW DOMAINS

As part of its research, QinetiQ has experimented extensively with prototype implementations of the above techniques. The prototype mechanism for releasing information from high to low domains is described in the figure and text below.

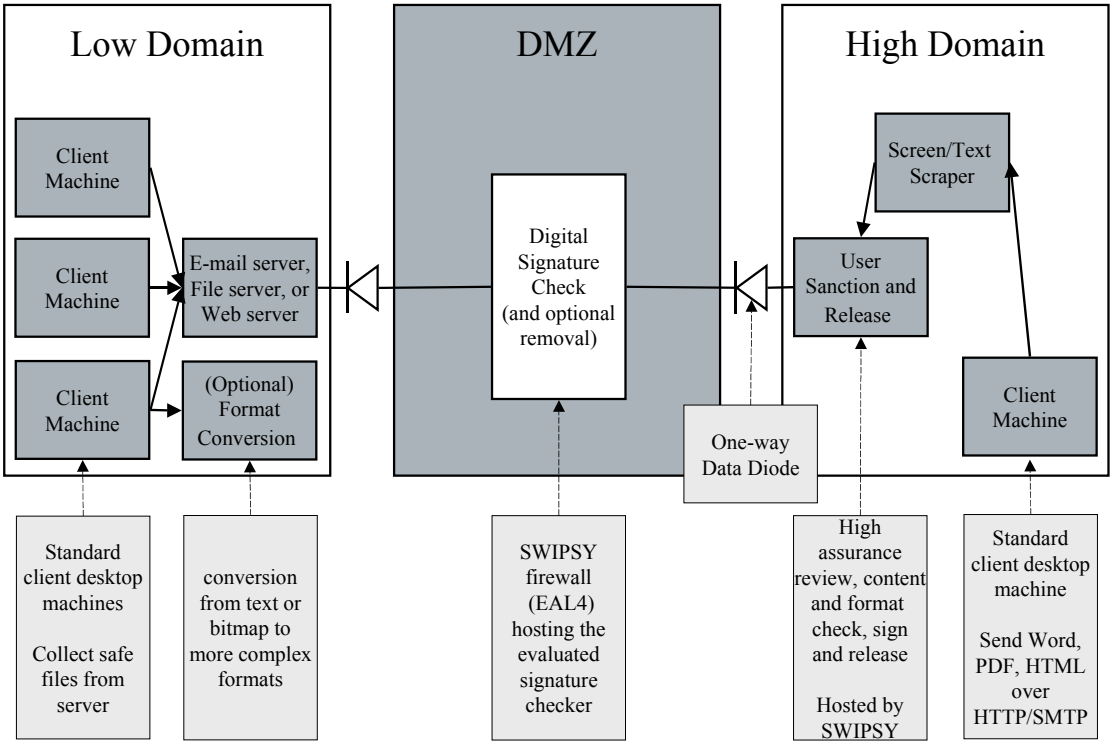


Figure 2 – The High Assurance prototype mechanism for High to Low information release

The steps in the release process are as follows:

- Format Conversion: The user connects to a high domain web server that can perform the screen-scrape or text-scrape, converting the complex file (e.g. a Word document) into a Bitmap image or ASCII text document
- The converted file is routed, via e-mail, to the user sanction release mechanism
- The trusted user sanction device is built upon the Trusted Solaris operating system (EAL4 rating) modified with the SWIPSY toolkit (EAL4 rating also). Upon this system a simple console-based viewing program has been written and installed
- The console viewer strips the MIME encoding surrounding the message body (or attachment in the case of a Bitmap image), performs the format check, and presents the ASCII text or Bitmap to the user
- The user can then choose to stop or release the message. If the release option is chosen, the message is digitally signed and packaged in an S/MIME envelope.
- The digital signature is applied using an assured cryptographic library (currently the CESG Cryptserve algorithm suite - EAL4) and, in this early prototype, a full S/MIME toolkit. Future versions of the system will employ ‘cut-down’ program code to create the S/MIME envelope. One

slight modification has already been made to the S/MIME standard: The inclusion of a copy of the "To:", "From:" and other header fields in the message body in order to counteract the subtle S/MIME weakness where header information is unsigned. However, this does not affect its correct reception and checking in a standard S/MIME client.

- The signed message is sent to the DMZ and to another SWIPSY firewall (this one operating without human intervention). At the DMZ firewall the digital signature is verified and (optionally) removed before it is forwarded to the low domain SMTP server.

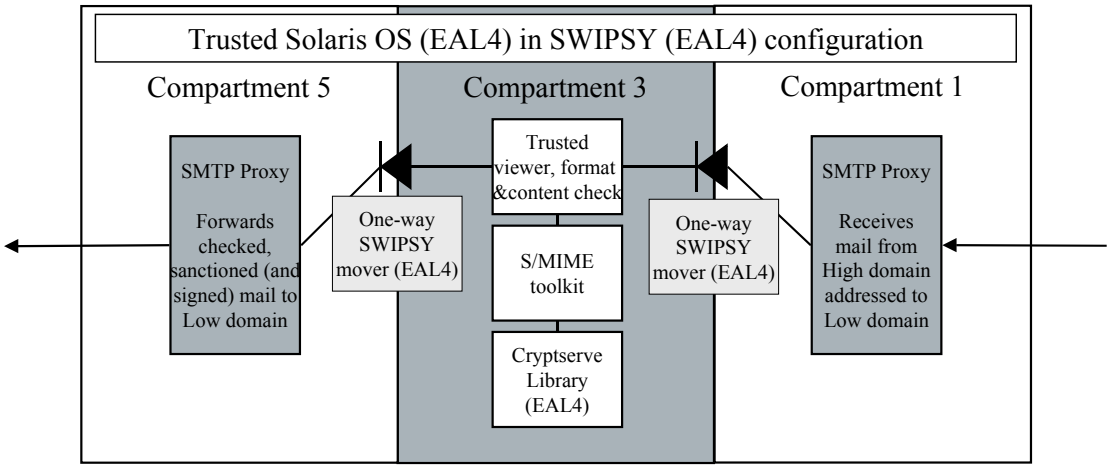


Figure 3 – Detail of the User Sanction and Release Mechanism

5.0 HIGH ASSURANCE PROTOTYPE – SENDING INFORMATION FROM LOW TO HIGH DOMAINS

The route from a low to high domain does not necessarily require a user sanction (although there may be good security reasons why such intervention may be required in some circumstances). This simpler process can be fully automated and is detailed in the proposed chain of events and diagram below:

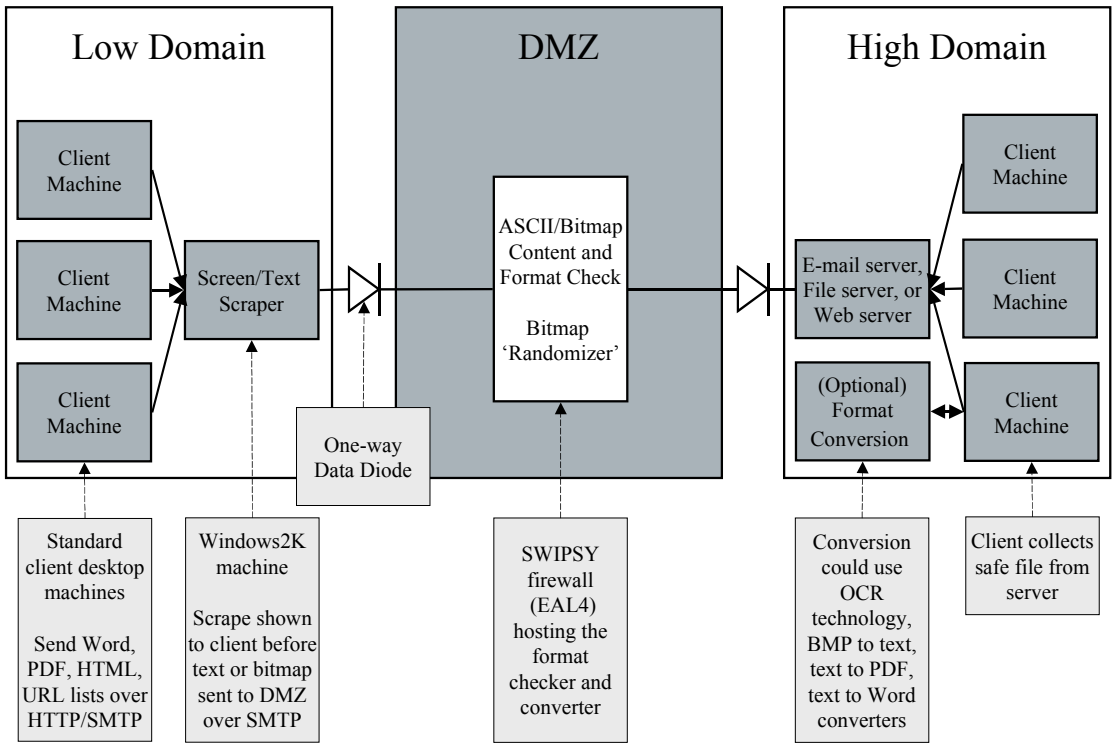


Figure 4 – Sending information from low to high domains

- Format Conversion: The user connects to a low domain web server that can perform the screen-scrape or text-scrape, converting the complex file (e.g. a Word document) into a Bitmap image or ASCII text document
- The scraped file is sent through the one-way data diode to the trusted format checker hosted by a SWIPSY machine in the DMZ. The checker performs a content check on the ASCII or Bitmap file as before, and if successful…
- The Bitmap file is 'randomised' and optionally transformed into a JPEG file
- The transport protocol encodings that surrounded the file are stripped and reformed on the SWIPSY machine (e.g. MIME encoding and SMTP headers). Alternatively, protocols like FTP can be used that require no transport 'envelope' around the file.
- The checked file is passed to the high domain through a second one-way data diode
- The file is delivered to the high domain server from which the client can collect the 'safe' file
- The file *could* be transformed into the original document format at this stage, for example into a Word file. This is a potential problem though as the original 'malware' may be reconstructed as part of the process

It can be seen that the data diodes illustrated in the three diagrams above are pointed in different directions, creating a two-way flow of data between high and low that might appear to make the data diodes redundant. However, the high and low domains illustrated in the diagrams are generic labels and not meant to imply that the low domain is the same low domain in both diagrams. Rather, the scenarios described in this paper are assuming the high domain has a multiplicity of connections to lower domains.

6.0 FUTURE DEVELOPMENTS AND ALTERNATIVE DESIGNS

6.1 Transforming Complex Files to XML

This paper has explored a scenario requiring particularly stringent constraints on the information transfer between domains. Other systems may require a 'richer' exchange of information, involving more complex file types and data encodings than simple ASCII text and Bitmap images. Examples include PDF files and database updates. For these kinds of files, QinetiQ is developing techniques for transforming complex file types into a single, standard XML representation. The complex (but uniform) XML would then be subjected to content checking on a trusted platform, and then transformed from the XML back to the original encoding (Word again) or to a new encoding, such as PDF format.

6.2 Trusted Viewing and Signing on the Client Desktop with Virtual Machines

Machine Virtualisation allows a second complete operating system to be installed on a user's machine and for that second operating system to run concurrently as a second 'virtual' machine with the host operating system. Virtual machine software such as VMWare allows many such virtual machines to operate simultaneously.

There have been some interesting developments recently using virtual machine technology to create secure multi-system desktops where the two virtual machines are separated by assured mechanisms (for example, the NetTop approach developed by the NSA[11]).

It would be advantageous if the trusted user sanction mechanism could be built into a separate virtual machine on the client's desktop. This could then involve an assured and highly locked down Operating System that is specifically for this one purpose. Such a development would add flexibility and convenience, allowing users to release documents from their own desktops, either to replace the trusted 'domain' signer described in this paper, or in addition to their sanction, creating an effective two-man rule release mechanism.

7.0 CONCLUSIONS

This paper has proposed a combination of techniques to allow a potentially assured, two-way exchange of information between high and low assurance domains. The scenario has assumed that the stringent requirements for separation of the two domains could only be achieved by an air-gap, given current commercially available solutions.

What has been proposed in the paper is intended to offer *greater* security than an air-gap, principally because, like winning the lottery, an air-gap is an attractive idea in theory but difficult to achieve in practice. Pragmatic users faced with an air-gap between security domains will often reach for a floppy disk or USB memory stick rather than re-key the entire document from the high machine to the low.

The combination of format controls, environment controls, and user controlled release sanctions outlined in this paper have been designed to offer a secure solution to the urgent and unavoidable business requirement to share information.

8.0 REFERENCES

[1] CERT Co-ordination Statistics 1988-2003, http://www.cert.org/stats/cert_stats.html

[2] Simon Wiseman, QinetiQ White Paper, “Documents with Hidden Surprises”, http://www.qinetiq.com/homepage/services/information/white_paper0/documents.html

[3] George Guninski, Security Research Web Page "Internet Explorer Security", http://www.guninski.com/browsers.html

[4] Common Vulnerabilities and Exposures, “Buffer Overflow in Adobe Acrobat…”, http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2000-0713

[5] Univeristy of Oulu, Security Testing of Protocol Implementations, http://www.ee.oulu.fi/research/ouspg/protos/

[6] Kevin Poulson, The Register, "Brits pound OpenSSL bugs" http://www.theregister.co.uk/content/archive/33148.html

[7] Directive 1999/93/EC of the european Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. Official Journal L 013, 19/01/2000 p. 0012 – 0020. http://europa.eu.int/ISPO/ecommerce/legal/digital.html

[8] Trusted Computing Group Homepage, https://www.trustedcomputinggroup.org/home

[9] Trusted Computing Group, Trusted Platform Module, "Design Principles", https://www.trustedcomputinggroup.org/downloads/tpmwg-mainrev62_Part1_Design_Principles.pdf

[10] Microsoft, Next-Generation Secure Computing Base Homepage, http://www.microsoft.com/resources/ngscb/default.mspx

[11] National Security Agency, NetTop Technology Profile Fact Sheet, http://www.nsa.gov/programs/tech/factshts/20030103-3.htm

Tempering Network Stacks

Dr.-Ing. Stephen D. Wolthusen
Fraunhofer-IGD
Security Technology Department
Fraunhoferstr. 5
64283 Darmstadt
Germany

wolt@igd.fhg.de

SUMMARY

This paper summarizes existing and describes ongoing work on securit ypolicy definition and particularly enforcement in heterogeneous distributed systems. Based on a formal model of operating systems and interactions among networked nodes in a distributed system axiomatizing relations among and abstractions in distributed systems, arbitrary security policies can be defined over the same model; automated reasoning techniques can be used to dynamically derive the compliance of operations with all applicable security policies. A key component for enforcing such security policies in operating system network stacks is described along with instrumentation techniques for the Microsoft Windows NT family of operating systems.

1.0 INTRODUCTION

Information assurance in distributed, heterogeneous systems frequently requires that formal and informal security policies be enforced by technical means. The expressiveness required by security models and, more generally, policies [21], however, frequently exceed the capabilities of the mechanisms available in currently deployed networking components and general purpose operating systems.

Extant (deployed) systems, particularly for network security policies are generally limited to simple access control lists and in some cases to elementary heuristics in the scope of their proactive security mechanisms both in case of operating system capabilities and add-on components.

Similar limitations also exist in the capabilities of the controls themselves since existing controls are typically limited in their design to enforce simple access control mechanisms, e.g.~in the form of basic or stateful packet filtering mechanisms integrated into operating systems.

A number of threats increasingly necessitate improvement of security policy mechanisms, controls, and the assurance provided by such controls even in nominally secured networks. Topologically oriented security mechanisms such as perimeter firewalls increasingly undermined through the use of a variety of wireless network interfaces including IEEE 802.11x, Bluetooth, and even IrDA/FIR frequently found on standard workstations that support ad hoc networking among peer nodes.

Threats inherent in this include denial of service, eavesdropping, and active penetrations, but more importantly also represent a vector through which malicious code can be inserted into a network to cause arbitrary damage. Frequently, such access is provided deliberately by end users, e.g.~when establishing a piconet or point-to-point network interface connections for sharing materials written using popular office

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

productivity applications with extensive macro capabilities and therefore also vulnerability surfaces for malware.

Moreover, even network traffic passing through perimeter security controls is increasingly opaque to such perimeter firewalls and network intrusion detection mechanisms. Reasons for this include the use of encrypted end-to-end channels or message formats that partially blind perimeter network security mechanisms nominally capable of scanning network traffic, but also the proliferation of protocols that are explicitly designed to circumvent network security controls such as the SOAP protocol [4,14] and additional layered protocols.

These observations lead to a number of desiderata for improving network stack security. Since information required for reaching decisions regarding conformity of network traffic and operations is increasingly available only at the end nodes themselves and end nodes may also be directly exposed to hostile traffic, network security controls must be integrated directly into the end nodes themselves [3].

2.0 SECURITY ARCHITECTURE BACKGROUND

General security policies within organizations, typically created only in informal prose, must be mapped onto available security models and ultimately security controls, potentially losing accuracy at each of these steps and also potentially incomplete because of limitations of the layer mapped onto in each step. Such mapping errors can be detrimental both in omitting controls that policies call for and in imposing overly restrictive controls that limit capabilities and effectiveness of the information system.

Moreover, demonstrating the correspondence of each mapping (e.g. document handling guidance onto technical access controls) is a resource-intensive effort and similarly prone to errors and oversights as the original mapping.

In modeling individual nodes (i.e. operating systems and the resources controlled by these systems) and interactions among nodes at a level of abstraction sufficient to capture operational semantics across multiple general-purpose operating systems through formal concept analysis using formal logic, bijective mappings onto specific instances of operating systems can be considered interpretations of such formal theories.

Arbitrary security policies can then be formulated within the same formal theory, interpreted as either permitted or required operations. Automated deduction mechanisms can therefore be used to derive additional statements and instances of the model.

By including abstraction relations over entities and operations within the axiomatization of the underlying system, the reasoning can, moreover, occur over multiple abstraction layers, such as deriving the permissibility of an read operation accessing an individual block on a fixed disk by a given process based on abstractions tracing these entities and operations onto e.g. personnel roles and documents; this can be achieved by mapping an operation onto the formal model in the form of a well-formed formula, a proof of the validity of such a hypothesis (derived e.g. via term rewriting or automated deduction mechanisms) is then considered permission to perform the operation.

The axiomatization, based on embedding algebraic (e.g. lattice) structures within the formal theory provides critical efficiency gains not only in a priori providing proof structuring aids but also in permitting the re-use of decisions. By embedding lattices over both entity and operation types and over entity identities, policy decisions can be reached quickly by avoiding resolution to ground termns and re-used by simple rewriting in later decisions. Such derived (proven) formulae can be considered part of the policy set with legitimate operations being described by the Lindenbaum operator over all policies with each such statement being assigned a lifetime providing implicit pruning and dynamism [31].

In the underlying architecture, there exist a number of nodes called external reference monitors (ERM) which are repositories for one or more security policies, each presumably derived from a security model. The system on which an ERM resides is called a Policy Controller Node (PCN). The other component of the framework consists of a number of nodes which are subject to the policies of one or more ERM [27,31].

The policies obtained from ERM are enforced through externally controlled reference monitors (ECRM) and its enforcement modules (EM); a system configured with a combination of ECRM and EMs is called a Policy Enforcing Node (PEN). As implied by the term reference monitor, each operation of the controlled nodes is mediated by the ECRM and may only proceed if it is found to be in compliance with all applicable policies. Applicable policies (and hence the ERMs to be consulted) are determined from the identity of subjects and objects involved which are uniquely identified by the conjunction of a subject identity and a subject type constant.

3.0 NETWORK ENFORCEMENT MECHANISMS

In addition to other system components such as device interfaces [32], file systems [28], and process management required for ensuring the completeness property for reference monitors, network interfaces constitute one of the minimum required controls for security policy enforcement.

The network enforcement module must satisfy a number of functional requirements, namely to control all in- and outbound data packets and circuit operations in such a way that data flows are presented to the host operating system only after having been validated; this is particularly relevant for operating systems where the network stack may not be capable of properly handling malformed data flows.

Moreover, the network enforcement module must provide transparent data object labeling to permit the identification of higher level entities in case of data flows among nodes within the security architecture.

To establish secure in-band communication with PCN nodes that may not be possible because of state space restrictions in using the host operating system network stack, the enforcement module must also provide a fully separated cryptographically secured communication channel.

This paper describes one such ongoing enforcement module implementation for the Microsoft Windows NT family[1] of operating systems.

However, the following sections concentrate mainly on the instrumentation mechanisms and omit more advanced object identification and security protocol elements.

3.1 Windows NT Family Network Protocol Stacks

Unlike the other components such as file system handling, the networking mechanisms provided by the Microsoft Windows NT operating system family do not share a common abstraction for all supported types of network communication.

Therefore, in addition to multiple environmental subsystems providing different access mechanisms to network communication subsystems, there exist several conceptually different networking application programming interfaces, namely

- WinSock
- Named Pipes

[1] including the 3.*x*, 4.0, 2000, XP, and 2003 versions at the time writing.

- Mailslots
- Remote Procedure Call
- NetBIOS
- Telephony

Other services such as DCOM [8] or the .NET framework [26,18] may be layered on top of these interfaces; while some of these interfaces have their own security and encryption mechanisms (such as RPC), others rely on the connection being assumed as secure and simply enforce access controls (e.g. named pipes and mail slots which are implemented as file systems and can use the access control mechanisms for file systems, see [23,24]).

Of these mechanisms, the telephony interfaces (TAPI) are in an unique class based on the mechanism used by user level programs to communicate with kernel-level components.

The TAPI user level component (`TAPISRV.DLL`) provides access to a number of TAPI service providers (TSP); while most of these map to networking subsystems discussed later in this section, this also includes direct access to device drivers for modem devices (which can themselves be used to establish arbitrary network connections including interfacing to other network protocols).

This particular component therefore requires specific enforcement mechanism support (e.g. in the form of device-level enforcement for modem-type devices) to avoid the introduction of unenforceable information flow paths.

In the general case, the network architecture of the Microsoft Windows NT family consists of a number of layers, depicted in figure 1.

At the lowest level is the physical device. Access to individual devices is regulated by the hardware abstraction layer (HAL). Network device drivers are generally realized as NDIS (Network Driver Interface Specification) modules consisting of the generic NDIS library and the device-specific NDIS miniport drivers; the library fully encapsulates the miniport drivers.

Accessing the NDIS library is the TDI (Transport Driver Interface) mechanism. This itself consists of transports (or protocol drivers), supporting the various transport mechanisms such as NetBEUI (NetBIOS Extended User Interface) and TCP/IP, and TDI clients which provide services for sockets and NetBIOS calls. None of these modules can be called directly from applications since they are protected kernel mode interfaces.

Upper-level APIs (application programming interfaces) such as NetBIOS and Windows Sockets are subsequently implemented at the user level and must use the aforementioned interface layers.

The Windows Sockets API (or WinSock) is modeled after the original BSD Unix sockets API [20] and has undergone significant revisions under various platforms before arriving in its current form [1,2]. It is available for both the NT-based and DOS-based operating systems from Microsoft Corporation.

The Windows Sockets API is itself composed of several modules. From an application's perspective the sockets API consists of the exposed API DLL; this DLL (dynamically linked library) communicates with the SPI (Service Provider Interface) layer.

This layer is controlled by the transport service provider DLL which in turn calls on a number of transport helper DLLs and namespace helper DLLs to perform its operations. Moreover, the transport service provider DLL forwards the thus generated calls to the System Support Library DLL that represents the interface to the abovementioned kernel components.

Since the Microsoft Windows NT design is predicated on a file system model and represents sockets as file handles, a translation mechanism is required. This service is performed by an Ancillary Function Driver (AFD).

Of particular interest in this is the ability to stack several of the transport helper DLLs so as to provide additional services at each level (there is no layering mechanism for namespace helper DLLs). WinSock here distinguishes between "base protocols" and "layered protocols". The former are protocols capable of performing actual communication with a remote endpoint, the latter must rely on base protocols for actual communication and only provide added value.

Provided that all elements of such a stack are conforming to the interface specifications set forth in [1,2], it is possible to implement several stacked layers of such layered protocols.

At an abstraction level below the user level API mechanisms, the protocol driver layer accepts requests from API-level mechanisms and translates these into respective network protocol elements. The number and type of protocol drivers are variable among nodes and may include but are not limited to TCP/IP, NetBEUI, IPX/SPX (provided in a single protocol driver instance), and AppleTalk. Typically, each protocol driver supports all protocols of a protocol family (e.g. IP, ARP, RARP, ICMP, IGMP, UDP, and TCP in case of the TCP/IP protocol driver, `TCPIP.SYS`).

All protocol drivers communicate with API-level components (as well as other components such as the previously noted Windows Sockets ancillary function driver) using part of the TDI; which is specified in the form of IRP classes.

For connection-oriented protocols, `TdiDispatchCreate` creates a file object (also referred to as an address object) by through the use of an `IRP_MJ_CREATE` IRP which represents the node-local connection endpoint. This subsequently must be associated with an opened file object representing an address, referred to as an connection object.

Depending on the initiating node, subsequent IRP messages must then transition the connection object into listening or connecting state, which is then transitioned into an accepting state on the part of the listening node, which occurs using the IRP creation mechanisms `TdiDispatchDeviceControl`, `TdiDispatchFastDeviceControl`, and `TdiDispatchInternalDeviceControl`, respectively.

After a connection object has been discarded, `TdiDispatchClose` is used to discard the address object after `TdiDispatchCleanup` has ensured that no pending IRP messages exist for the address object; connectionless protocols omit the listening and connecting phase of this control flow.

TDI also permits the use of callback mechanisms and the intermediate caching of network protocol data units for efficiency purposes; this requires the registration of events with TDI client interfaces. Typically, this results in TDI clients generating `TDI_SEND` IRP messages and reacting to `TDI_EVENT_CHAINED_RECEIVE` and the `TDI_EVENT_RECEIVE*` family of IRP messages.

For communication with the device drivers controlling the network interface adapters, the TDI protocol drivers communicate by way of a library encapsulating device-specific properties.

This library, NDIS, provides a procedural interface for the TDI as well as for the actual device drivers (miniport drivers), which communicate to the remainder of the operating system only through the NDIS library.

Internally, however, the Microsoft Windows NT implementation of NDIS itself uses IRP-based messaging for control flow. The NDIS library provides services for both connectionless (e.g. IP) and connection-oriented (e.g. ATM) protocols as well as a number of other services [24].

NDIS also provides several other security-relevant services that need to be addressed, such as the ability to forward datagrams from one network interface to another without processing by the remaining operating system network protocol stack or the offloading of certain network processing (specifically TCP/IP-related operations) to the network interface device and hence the NDIS level.

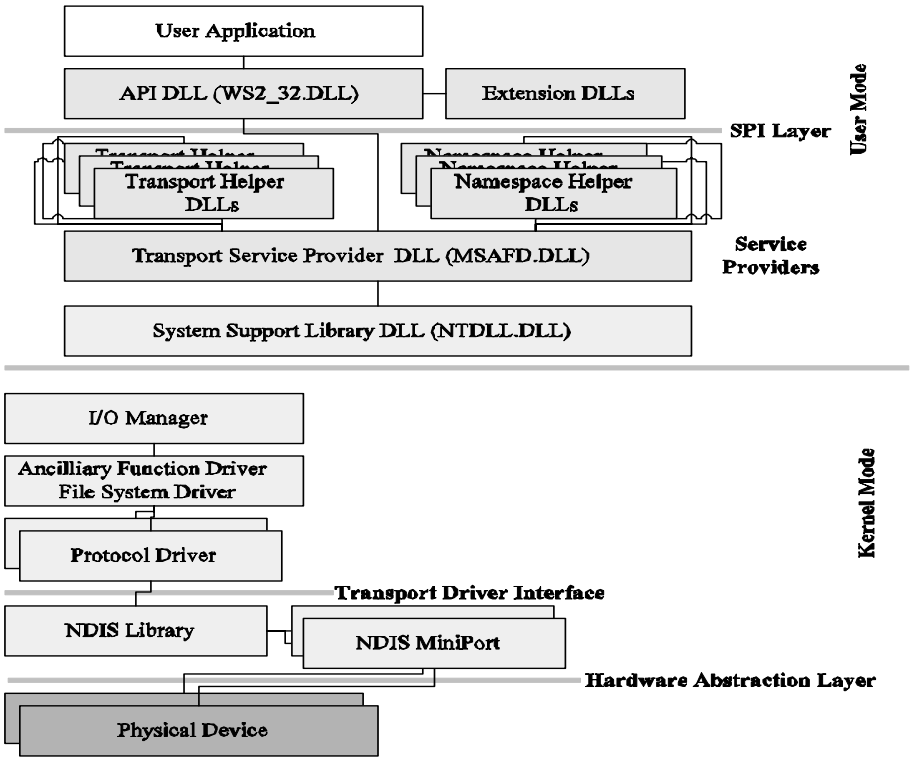


Figure 1: Components involved in networking mechanisms in the Microsoft Windows NT operating system family.

3.2 Protocol Stack Integration

The provision of the semantics appropriate for a network enforcement layer requires the insertion of instrumentation at least at two of the protocol layers described in the preceding section, namely at the NDIS and TDI layers.

3.2.1 NDIS Instrumentation

With the exception of parts of TAPI discussed above, all network traffic within the Microsoft Windows NT family is transmitted by way of NDIS devices, regardless of the API and protocol used; it is also possible for a user level process to directly communicate with the NDIS layer (again, TAPI is an example of this behavior).

It is therefore imperative for the provision of the required interpretation semantics to intercept and instrument the processing at the NDIS level. For this purpose, several implementation alternatives exist, two of which are of sufficient generality for the purposes discussed here.

One possible approach is the use of an NDIS Intermediate Driver, which permits the interpositioning of code between miniport drivers and the remainder of the NDIS library. While appealing and providing a well-defined interface for interposition, this approach does not provide the most general mechanism since NDISWAN miniport drivers are not supported in the NDIS revision (version 5.0 and 5.1, respectively) used by Microsoft Windows 2000, XP, and 2003.

This would require the mandatory use of backwards-compatible NDIS version 4.0 mechanisms, which for obvious reasons is highly undesirable given the improvements and features added in NDIS version 5.*x*.

The alternative to intermediate drivers providing the most general coverage of mechanisms supported is in the manual interception of control flows destined for and within NDIS.

Since it is possible that the configuration of both protocols and network interfaces may change at any time during runtime (e.g. through the addition of an ad-hoc network interface), a general mechanism is required that supports not only bootstrapping mechanisms but also provides monitoring and dynamic interception of such configuration changes. For this purpose, an NDIS layer enforcement sub-module can be loaded and started prior to the initialization of the network subsystem.

Since the NDIS architecture differs significantly in initialization and particular I/O flow from normal device I/O unter the Windows NT platform, however, the interception cannot be effected by registering with the I/O manager and redirecting the flow of IRP messages, but must occur directly by redirecting function entry points to the enforcement sub-module itself and subsequent transfer of control flow to the NDIS library once the required operations have been performed on the part of the enforcement sub-module.

To ensure that policies can be enforced uniformly, all network interfaces on a node must be intercepted and brought under the control of the enforcement sub-module. This occurs by intercepting the NDIS functions `OpenAdapter` and `CloseAdapter` and tracking the activation and deactivation of any (virtual) network interface; the actual interception mechanism relies on modifying the addresses contained in the export table of the module providing the NDIS library upon loading of the NDIS module.

Similarly, to be able to track information and control flows — particularly for callback mechanisms — the enforcement sub-module must retain information on which protocol drivers are registered with (and hence may access) the NDIS layer. This is accomplished by intercepting the `NdisRegisterProtocol` functions for registration and, correspondingly for unloading and deregistration, the `NdisMRegisterUnloadHandler` and `NdisDeregisterProtocol` functions.

The information thus obtained permits the correlation of information and subsequent coordination with instrumentation provided by enforcement sub-modules at the protocol driver level discussed in the following section.

While NDIS is the proper location to capture all control and data flows pertaining to network traffic and therefore also to perform protocol-specific operations, the information available at the level of the NDIS library (and hence the interception mechanism) are severely limited. At the level of the NDIS library, it is not directly possible to identify the subject (i.e. process) a data flow is associated with since the data flow from a process directed towards NDIS is translated into IRP messages at the kernel level, thus obliterating the information on the subject.

Conversely, data flows directed towards subjects are not associated with processes directly, but only with protocol drivers. It is therefore necessary (as described in the following section) to correlate information regarding the subject association with a data flow by coordinating the information available at the NDIS level with information from higher abstraction levels to permit the employment of security controls available only at the lower NDIS layer.

A similar problem exists with regard to the payload of the individual data flows processed by NDIS. At the NDIS level, only data already processed into protocol data units (PDU) are presented, and NDIS is expected to operate opaquely on the data provided in either direction. In this case, the information as to which protocol is associated with a PDU is obtained indirectly through the information gathered on registration of protocols.

A list of known protocol drivers must be maintained (this can occur through known identifying characteristics within the protocol driver or indirectly through the file system enforcement mechanism providing an unique fingerprint for a given protocol driver by way of the ECRM), and protocol-specific operations must be invoked on the PDU based on the information thus obtained.

However, since PDU may be constrained either by the respective protocol or by the network interface, it is not always possible to transform PDU in place. Instead, a given PDU (regardless of inbound or outbound processing) may result in several PDU after processing by the protocol-specific enforcement sub-module and, moreover, the protocol-specific enforcement sub-module can withhold the processed PDU (and hence process additional PDU from the same data flow) prior to emitting one or more PDU for further processing by the NDIS layer.

The necessary information for identifying subjects, objects, and operations are transmitted by the TDI sub-module as discussed in the following section. Individually, the instrumentation provided by the NDIS layer enforcement sub-module can monitor the activation and deactivation of protocols and adapters as well as monitor in- and outbound data flows, including the elimination of inbound traffic as well as outbound[2].

Another operation that can be performed by the NDIS layer without interoperation with other sub-layers is data flow normalization, i.e. providing well-defined temporal characteristics for all or selected data streams such as inter-PDU time intervals. This, however, requires potentially large buffers in the absence of flow control mechanisms that can be applied transparently to the communicating parties proper.

3.2.2 TDI Instrumentation

While the enforcement mechanism proper is located at the NDIS level as described in the preceding section, the implementation of the Microsoft Windows NT operating system family necessitates the addition of a further enforcement sub-module at the protocol driver level.

The need for this additional sub-module stems from the lack of information regarding the association of subjects (and potentially of operations) as well as of objects of higher abstraction levels from which a given object or PDU is derived at the NDIS layer.

However, as noted before, there are potentially multiple protocol drivers active within a given node, each of which requiring specific actions for deriving the requisite information for reaching policy decisions by the ECRM in conjunction with other sub-module information. For the purposes of this dissertation, the discussion concentrates without loss of generality on the TCP/IP protocol driver.

[2] The `NdisCancelSendPackets` and `NdisGeneratePartialCancelId` mechanisms are, while not strictly necessary for this purpose, supported only from NDIS 5.1 onward.

Interception of the protocol driver occurs analogous to the mechanism described for the NDIS library in the preceding section; entry points are dynamically redirected on initialization of the protocol driver and forwarded after processing. As with the NDIS layer, this facilitates dynamic addition and removal of protocol drivers at runtime provided that the proper enforcement sub-module for a given TDI protocol driver is available.

The main operation performed at the TDI enforcement sub-module is the collection of information regarding subjects, objects, and data flows (the latter information is available implicitly through the observation of calls to the TDI); subject (i.e. process information that can be correlated with other subject information at the ECRM) information is implicitly available through the calling mechanism. In case of an outbound data flow, the information thus gathered must be made available to the NDIS sub-module to permit proper processing.

While it would be conceivable to transmit this information out of band or to store it at the ECRM itself, both possible alternatives would require not only considerable storage, but also imply complex storage management since the processing order is not necessarily the same for data flows at the TDI and NDIS layer, and special cases such as canceled data flows would need to be taken into account to avoid stale storage.

To avoid these problems as well as performance issues arising from extraneous communication between sub-modules (typically in the form of IOCTL messages that require considerable processing overhead), data flows can be annotated in-band with the requisite information. The NDIS sub-module can extract this information[3] and continue processing as described in the preceding section.

Similarly, inbound data flows can be reverse-associated with the information regarding subjects, objects, and data flows. This requires one instance of communication between the NDIS and TDI sub-modules for each flow (in the worst case of connectionless protocols, this is once for each PDU, although heuristics and information from other sub-modules not discussed here can be established to identify virtual flows based on addressing information in the more general case of connectionless protocols).

4.0 DEVELOPMENTAL ASSURANCE ASPECTS

Overall assurance achievable by the security architecture discussed here is, to a first approximation, limited by the lowest level of assurance of any component.

The set of components first and foremost also includes the host operating system in case of retrofits of security mechanisms as described in this paper since defects therein can potentially compromise or bypass any additional security controls. However, since the requirement for using systems with such limited overall assurance exists — primarily because of application program availability — it is imperative to provide sufficient levels of developmental assurance within said confines.

This can be achieved by including the system to be instrumented in the process of formally modeling the enforcement module. By not relying (solely) on documentation but rather performing reverse engineering and evaluation of the network stack components and capturing both expected and observed behavior in the model (primarily using the Z notation [25,15]), certain types of flaws based on incorrect assumptions or documentation can be avoided or, if subsequent evaluation results in assumptions becoming invalidated, new information can be incorporated into the model rigorously.

[3] There exists a mechanism for this purpose in the `NDIS_PACKET_STACK` structure introduced in NDIS 5.1; prior NDIS versions require the allocation of a new, larger packet for the integration of this data.

5.0 DISCUSSION AND RELATED WORK

The mechanisms described here represent part of a larger security architecture that touches upon a number of fields; the discussion here is restricted to network security policy mechanisms and implementation strategies.

Based on observations on the use of mobile devices, remote access mechanisms, and the performance requirements for choke point firewalls resulting from increasing network performance, Bellovin proposed the migration of firewall enforcement to the nodes to be protected while retaining a central policy definition for network access control [3]. Instead of relying on topological information for obtaining statements on the identity of an entity, Bellovin proposed to use the mechanisms for the use of public key infrastructures inherent in IPSec which also addressed the issue of support for virtual private networks found in traditional firewalls [17], although this was done earlier by Chitturi [7] in 1998 within the Fluke project context [10].

Another approach to distributed firewalling, also derived from concepts introduced by Bellovin was pursued by Payne and Markham at SCC. Payne and Markham realized the embedding of the firewalling mechanisms (EFW) on a COTS network interface card with cryptographic and limited processing capabilities [19,22]. A similar mechanism for EFWs was also developed by Friedman and Nagle at Carnegie Mellon University [11,12].

Policy-based network security has been the subject of intensive research; Burns *et al.* describe a network security policy architecture based on security models at moderate abstraction levels [5]; for an earlier survey of such mechanisms see [6].

The problem of insufficient instrumentation for security purposes has also been addressed by other researchers; Keromytis describes a data flow tagging architecture similar to the one described here for the OpenBSD environment, these are also used in OpenBSD to record buffer- (packet-) specific information such as security data that is not retained in normal data and control flows [16]; more specialized interposition mechanisms include, among others, work on Exokernels at MIT by Kaashoek *et al.* [9] and SLIC by Ghormley *et al.* [13].

6.0 CONCLUSIONS AND FUTURE WORK

This paper has described an instrumentation mechanism for enforcing flexible and dynamic security policies imposed by a distributed security policy mechanism and a specific implementation thereof for retrofitting an instrumentation mechanism onto a COTS (commercial off the shelf) operating system without access to source code. While the modeling of the host operating system under such conditions requires significant resources, it also ensures that major discrepancies between intended (documented) and actual behavior are discovered.

Uses of the network policy mechanisms include dynamic distributed firewalling [27] and intrusion detection and response [29,30] as described in earlier papers.

Ongoing extensions to the instrumentation include the implementation of an in-line (*bump-in-the-stack*) IPSec mechanism for the Internet protocol family and further refinements of the mirror network stack mechanism itself.

In addition, measurements and tuning are a major focus of ongoing work as the latency imposed by the mirror stack and policy enforcement can become pronounced for TCP/IP connections when used rapid circuit establishment and teardown occur over high speed network interfaces.

7.0 REFERENCES

[1] ANDERSEN, D. B. Windows Sockets 2 Application Provider Interface. Techn. Rep. Intel Corp., May 1997, Version 2.2.1

[2] ANDERSEN, D. B. Windows Sockets 2 Service Provider Interface. Techn. Rep. Intel Corp., May 1997, Version 2.2.1

[3] BELLOVIN, S. M. Distributed Firewalls. *;login. The USENIX Association Newsletter 19*, Special issue on security (Nov. 1999), 39-47.

[4] BOX, D., EHNEBUSKE, D., KAKIVAYA, G., LAYMAN, A., MENDELSOHN, N.,NIELSEN, H.F., THATTE, S., AND WINER, D. Simple Object Access Protocol (SOAP) 1.1. Techn. Rep. W3C, May 2000. Status: W3C Note.

[5] BURNS, J. CHENG, A., GURUNG, P., RAJAGOPALAN, S., RAO, P., ROSENBLUTH, D. SURENDRAN, A., AND MARTIN, JR. D. M. Automatic Management of Network Security Policy. In *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX II)* (Anaheim, CA, USA, June 2001), IEEE Press, pp.1012-1026.

[6] CARNEY, M., AND LOE, B. A Comparison of Methods for Implementing Adaptive Security Policies. In *Proceedings of the 7th USENIX Security Symposium* (San Antonio, TX, USA, Jan. 1998), USENIX, pp.1-14

[7] CHITTURI, A. Implementing Mandatory Network Security in a Policy-flexible System. Master's thesis, University of Utah Department of Computer Science, Salt Lake City, UT, USA, June 1998.

[8] EDDON, G., AND EDDON, H. *Inside Distributed COM*. Microsoft Press, Redmond, WA, USA, 1998.

[9] ENGLER, D.R., AND KAASHOEK, M.F., AND O'TOOLE, JR., J. Exokernel: An Operating System Architecture for Application-Level Resource Management. *ACM Operating Systems Reviewd 29,5* (Dec. 1995), 251-266. Proceedings of the 15th Symposium on Operating Systems Principles (15th SOSP '95).

[10] FORD, B, HIBLER, M., LEPREAU, J., MCGRATH, R., AND TULLMANN, P. Interface and Execution Models in the Fluke Kernel. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI '99)* (New Orleans, LA, USA, Feb. 1999), ACM Press, pp. 101–116. Published as an ACM Operating Systems Review Special Issue.

[11] FRIEDMAN, D., AND NAGLE, D. F. Building Scalable Firewalls with Intelligent Network Interface Cards. Tech. Rep. CMU-CS-00-173, Carnegie Mellon University School of Computer Science, Pittsburgh, PA, USA, Dec. 2000.

[12] GANGER, G. R., AND NAGLE, D. F. Enabling Dynamic Security Management of Networked Systems via Device-Embedded Security. Tech. Rep. CMU-CS-00-174, Carnegie Mellon University School of Computer Science, Pittsburgh, PA, USA, Dec. 2000.

[13] GHORMLEY, D. P., RODRIGUES, S. H., PETROU, D., AND ANDERSON, T. E. Interposition as an Operating System Extension Mechanism. Tech. Rep. CSD-96-920, University of California at Berkeley, Berkeley, CA, USA, Apr. 1997.

[14] GUDGIN, M., HADLEY, M., MOREAU, J.-J., AND NIELSEN, H. F. Simple Object Access Protocol (SOAP) 1.2, Dec. 2001. W3C Working Draft, consists of three parts.

[15] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION AND INTERNATIONAL ELECTROTECHNICAL COMMITTEE. *Information Technology — Z Formal Specification Notation — Syntax, Type System and Semantics. International Standard 13568*, 2002.

[16] KEROMYTIS, A. D. Tagging Data in the Network Stack: mbuf tags. In *Proceedings of BSDCon '03* (San Mateo, CA, USA, Sept. 2003), USENIX, pp. 125–132.

[17] KEROMYTIS, A. D., AND WRIGHT, J. L. Transparent Network Security Policy Enforcement. In *Proceedings of the 2000 USENIX Annual Technical Conference* (San Diego, CA, USA, June 2000), USENIX, pp. 215–226.

[18] LAMACCHIA, B., LANGE, S., LYONS, M., MARTIN, R., AND PRICE, K. *.NET Framework Security*. Addison-Wesley, Reading, MA, USA, 2002.

[19] MARKHAM, T., AND PAYNE, C. Security at the Network Edge: A Distributed Firewall Architecture. In *Proceedings of the DARPA Information Survivability Conference (DISCEX II)* (Anaheim, CA, USA, June 2001), pp. 18–27.

[20] MCKUSICK, M. K., BOSTIC, K., KARELS, M. J., AND QUARTERMAN, J. S. *The Design and Implementation of the 4.4 BSD Operating System*. Addison Wesley, Reading, MA, USA, 1996.

[21] MCLEAN, J. Security Models. In *Encyclopaedia of Software Engineering*, J. J. Marciniak, Ed. John Wiley & Sons, Inc., New York, NY, USA, 1994, pp. 1136–1145.

[22] PAYNE, C., AND MARKHAM, T. Architecture and Applications for a Distributed Firewall. In *Proceedings 17th Annual Computer Security Applications Conference (ACSAC'01)* (New Orleans, LA, USA, Dec. 2001), IEEE Computer Society Press, pp. 329–336.

[23] SOLOMON, D. A., AND CUSTER, H. *Inside Microsoft Windows NT*, 2nd ed. Microsoft Press, Redmond, WA, USA, 1998.

[24] SOLOMON, D. A., AND RUSSINOVICH, M. E. *Inside Microsoft Windows 2000*, 3rd ed. Microsoft Press, Redmond, WA, USA, 2000.

[25] SPIVEY, J. M. *The Z Notation: A Reference Manual*, 2nd ed. Prentice Hall International Series in Computer Science. Prentice Hall, London, UK, 1992.

[26] THAI, T., AND LAM, H. Q. *.NET Framework Essentials*. O'Reilly & Associates, Sebastopol, CA, USA, 2002.

[27] WOLTHUSEN, S. Layered Multipoint Network Defense and Security Policy Enforcement. In *Proceedings from the Second Annual IEEE SMC Information Assurance Workshop*, United States Military Academy (West Point, NY, USA, June 2001), IEEE Press, pp. 100–108.

[28] WOLTHUSEN, S. Security Policy Enforcement at the File System Level in the Windows NT Operating System Family. In *Proceedings 17th Annual Computer Security Applications Conference (ACSAC'01)* (New Orleans, LA, USA, Dec. 2001), IEEE Press, pp. 55–63.

[29] WOLTHUSEN, S. Distributed Intrusion Detection for Policy-Controlled Heterogeneous Environments. In *Proceedings from the Third Annual IEEE SMC Information Assurance Workshop*, United States Military Academy (West Point, NY, USA, June 2002), IEEE Press, pp. 255–262.

[30] WOLTHUSEN, S. Embedding Policy-Controlled ID Sensors within Host Operating System Security Enforcement Components for Real Time Monitoring. In *Proceedings of the NATO RTO Symposium on Real Time Intrusion Detection Symposium (RTID)* (Estoril, Portugal, May 2002), NATO Research and Technology Organization. Publication RTO-MP-101.

[31] WOLTHUSEN, S. *A Model-Independent Security Architecture for Distributed Heterogeneous Systems*. Logos Verlag, Berlin, Germany, 2003.

[32] WOLTHUSEN, S. Goalkeeper: Close-In Interface Protection. In *Proceedings 19th Annual Computer Security Applications Conference (ACSAC'03)* (Las Vegas, NV, USA, Dec. 2003), IEEE Press, pp. 334– 341.

Resisting Traffic Analysis on Unclassified Networks*

Roger Dingledine
The Free Haven Project
arma@freehaven.net

Nick Mathewson
The Free Haven Project
nickm@freehaven.net

Catherine Meadows
Naval Research Laboratory
meadows@itd.nrl.navy.mil

Paul Syverson
Naval Research Laboratory
syverson@itd.nrl.navy.mil

Abstract

While the need for data and message confidentiality is well known, the need to protect against traffic analysis on networks, including unclassified networks, is less widely recognized. Tor is a circuit-based low-latency anonymous communication service that resists traffic analysis. This second-generation Onion Routing system adds to the first-generation design with perfect forward secrecy, congestion control, directory servers, integrity checking, variable exit policies, and a practical design for rendezvous points. Tor works on the real-world Internet, requires no special privileges or kernel modifications, requires little synchronization or coordination between nodes, and provides a reasonable tradeoff between anonymity, usability, and efficiency.

1 Introduction

It is well known that encryption hides the content of communication but does nothing to hide who is communicating with whom. Indeed, Whit Diffie, an inventor of public-key cryptography, has noted that traffic analysis, not cryptanalysis, is the backbone of signals intelligence [9]. The military has many reasons to communicate over open networks without revealing its communications partners. Communicating in this way assists intelligence gathering from open Internet sources, rapid formation of dynamic coalitions without an existing shared private infrastructure between members, and private communication with vendors to help conceal procurement patterns. Finally, it is sometimes not the communicants that are sensitive but their location: a server whose physical or logical location is known may be vulnerable to physical attack and denial of service.

* This work supported by DARPA and ONR.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Onion Routing is an overlay network concept for making anonymous connections resistant to eavesdropping and traffic analysis. It permits low-latency TCP-based communication such as web traffic, secure shell remote login, and instant messaging. The current design and implementation, Tor, improves on the original [13, 16, 18, 19] by providing perfect forward secrecy (see Section 2), interfacing to unmodified applications via SOCKS, multiplexing application connections on Onion Routing circuits, adding congestion control adding integrity checking, and including a rendezvous points design that protects the responder of a connection in addition to the initiator.

Onion Routing may be used anywhere traffic analysis is a concern. Because Onion Routing is an overlay network, it can exist on top of public networks such as the Internet without any modification to the underlying routing structure or protocols. In addition to protecting data confidentiality and integrity, the Onion Routing protocol hides the endpoint of each transmission. An intelligence analyst surfing a web site through Onion Routing is hidden both from that web site and from the Onion Routing network itself. On the other hand, Onion Routing separates anonymity of the communication from that of the data stream. That is, a procurement officer can place orders with a vendor and completely authenticate himself to the vendor while still hiding the communication from any observers—including compromised Onion Routing network components. Onion Routing can also be used to provide location hidden servers with better protection and yet less redundancy than standard approaches to distributed denial of service. In this paper we provide a brief overview of the Tor design. More detailed description is given in [10]. As we describe the system design, we will note how Onion Routing can be used to protect military communications in the above described settings.

1.1 Related Work

We give here a broad description of prior work; for a more complete list of references and comparisons, see [10].

Modern anonymity systems date to Chaum's **Mix-Net** design [5]. Chaum proposed hiding the correspondence between sender and recipient by wrapping messages in layers of public-key cryptography, and relaying them through a path composed of "mixes." Each mix in turn decrypts, delays, and re-orders messages, before relaying them toward their destinations.

Subsequent relay-based anonymity designs have diverged in two main directions. Some have tried to maximize anonymity at the cost of introducing comparatively large and variable latencies. Because of this decision, these *high-latency* networks resist strong global adversaries, but introduce too much lag for interactive tasks like web browsing, Internet chat, or SSH connections.

Tor belongs to the second category: *low-latency* designs that try to anonymize interactive network traffic. These systems handle a variety of bidirectional protocols. They also provide more convenient mail delivery than the high-latency anonymous email networks, because the remote mail server provides explicit and timely delivery confirmation. But because these designs typically involve many packets that must be delivered quickly, it is difficult for them to prevent an attacker who can eavesdrop both ends of the communication from correlating the timing and volume of traffic entering

the anonymity network with traffic leaving it. These protocols are also vulnerable to active attacks in which an adversary introduces timing patterns into traffic entering the network and looks for correlated patterns among exiting traffic. Although some work has been done to frustrate these attacks, most designs protect primarily against traffic analysis rather than traffic confirmation (cf. Section 2.1).

The simplest low-latency designs are single-hop proxies such as the Anonymizer [2], wherein a single trusted server strips the data's origin before relaying it. More complex are distributed-trust, circuit-based anonymizing systems. In these designs, a user establishes one or more medium-term bidirectional end-to-end circuits, and tunnels data in fixed-size cells. Establishing circuits is computationally expensive and typically requires public-key cryptography, whereas relaying cells is comparatively inexpensive and typically requires only symmetric encryption. Because a circuit crosses several servers, and each server only knows the adjacent servers in the circuit, no single server can link a user to her communication partners.

There are many other circuit-based designs, that make a variety of design choices; we again refer the reader to [10] for more information.

2 Design goals and assumptions

Goals

Like other low-latency anonymity designs, Tor seeks to frustrate attackers from linking communication partners, or from linking multiple communications to or from a single user. Within this main goal, however, several considerations have directed Tor's evolution.

Diversity: If all onion routers were operated by the defense department or ministry of a single nation and all users of the network were DoD users, then traffic patterns of individuals, enclaves, and commands can be protected from hostile observers, whether external or internal. However, any traffic emerging from the Onion Routing network to the Internet would still be recognized as coming from the DoD, since the network would only carry DoD traffic. Therefore, it is necessary that the Onion Routing network carry traffic of a broader class of users. Similarly, having onion routers run by diverse entities, including nonmilitary entities and entities from different countries, will help broaden and enlarge the class of users who will trust that system insiders will not monitor their traffic. This will provide both a greater diversity and greater volume of cover traffic. Unlike confidentiality, a single entity cannot achieve anonymity without collaboration, no matter how strong the technology.

Deployability: The design must be deployed and used in the real world. Thus it must not be expensive to run (for example, by requiring more bandwidth than onion router operators are willing to provide); must not place a heavy liability burden on operators (for example, by allowing attackers to implicate onion routers in illegal activities); and must not be difficult or expensive to implement (for example, by requiring kernel patches, or separate proxies for every protocol). We also cannot require non-anonymous parties (such as websites) to run our software.

Usability: A hard-to-use system has fewer users—and because anonymity systems hide users among users, a system with fewer users provides less anonymity. Usability

is thus not only a convenience: it is a security requirement [1, 3]. Tor should therefore not require modifying applications; should not introduce prohibitive delays; and should require users to make as few configuration decisions as possible. Finally, Tor should be easily implemented on all common platforms; we cannot require users to change their operating system in order to be anonymous. (The current Tor implementation runs on Windows and assorted Unix clones including Linux, FreeBSD, and MacOS X.)

Flexibility: The protocol must be flexible and well-specified, so Tor can serve as a test-bed for future research. Many of the open problems in low-latency anonymity networks, such as generating dummy traffic or preventing Sybil attacks (where one entity masquerades as many) [11], may be solvable independently from the issues solved by Tor. Hopefully future systems will not need to reinvent Tor's design. (But note that while a flexible design benefits researchers, there is a danger that differing choices of extensions will make users distinguishable. Experiments should be run on a separate network.)

Simple design: The protocol's design and security parameters must be well-understood. Additional features impose implementation and complexity costs; adding unproven techniques to the design threatens deployability, readability, and ease of security analysis. Tor aims to deploy a simple and stable system that integrates the best accepted approaches to protecting anonymity.

Non-goals

In favoring simple, deployable designs, we have explicitly deferred several possible goals, either because they are solved elsewhere, or because they are not yet solved.

Not peer-to-peer: Tarzan and MorphMix aim to scale to completely decentralized peer-to-peer environments with thousands of short-lived servers, many of which may be controlled by an adversary. This approach is appealing, but still has many open problems, such as greater effects of Sybil attacks and of greater network dynamics [12, 17].

Not secure against end-to-end attacks: We do not claim that Tor provides a definitive solution to end-to-end attacks, such as correlating the timing of connections opening or correlating when users are on the system with when certain traffic is observed (also known as intersection attacks). Some approaches may help, for example, accessing the network only through your own onion router; see [10] for more discussion.

No protocol normalization: Tor does not provide *protocol normalization* like Privoxy [15] or the Anonymizer [2]. In other words, Tor anonymizes the channel, but not the data or applications that pass over it. This means that Tor in itself will not hide, for example, a web surfer from being identified by the data or application protocol information observed at a visited web site. If anonymization from the responder is desired for complex and variable protocols like HTTP, Tor must be layered with a filtering proxy such as Privoxy to hide differences between clients, and expunge protocol features that leak identity. Note that by this separation Tor can also provide services that are anonymous to the network yet authenticated to the responder, like SSH. So, for example, road warriors can make authenticated connections to their home systems without revealing this to anyone including the local network access point. Similarly, Tor does not currently integrate tunneling for non-stream-based protocols like UDP;

this too must be provided by an external service.

Not steganographic: Tor does not try to conceal who is connected to the network from someone in a position to observe that connection.

2.1 Threat Model

A global passive adversary is the most commonly assumed threat when analyzing theoretical anonymity designs. But like all practical low-latency systems, Tor does not protect against such a strong adversary. Instead, we assume an adversary who can observe some fraction of network traffic; who can generate, modify, delete, or delay traffic; who can operate onion routers of its own; and who can compromise some fraction of the onion routers.

In low-latency anonymity systems that use layered encryption, the adversary's typical goal is to observe both the initiator and the responder. By observing both ends, passive attackers can confirm a suspicion that Alice is talking to Bob if the timing and volume patterns of the traffic on the connection are distinct enough; active attackers can induce timing signatures on the traffic to force distinct patterns. Rather than focusing on these *traffic confirmation* attacks, we aim to prevent *traffic analysis* attacks, where the adversary uses traffic patterns to learn which points in the network he should attack.

Our adversary might try to link an initiator Alice with her communication partners, or try to build a profile of Alice's behavior. He might mount passive attacks by observing the network edges and correlating traffic entering and leaving the network—by relationships in packet timing, volume, or externally visible user-selected options. The adversary can also mount active attacks by compromising routers or keys; by replaying traffic; by selectively denying service to trustworthy routers to move users to compromised routers, or denying service to users to see if traffic elsewhere in the network stops; or by introducing patterns into traffic that can later be detected. The adversary might subvert the directory servers to give users differing views of network state. Additionally, he can try to decrease the network's reliability by attacking nodes or by performing antisocial activities from reliable servers and trying to get them taken down; making the network unreliable flushes users to other less anonymous systems, where they may be easier to attack.

3 Highlights of the Tor Design

The Tor network is an overlay network; each onion router (OR) runs as a normal user-level process without any special privileges. Each onion router maintains a long-term TLS [8] connection to every other onion router. Using TLS conceals the data on the connection with perfect forward secrecy (see below), and prevents an attacker from modifying data on the wire or impersonating an OR. Each user runs local software called an onion proxy (OP) to fetch directories, establish circuits across the network, and handle connections from user applications. These onion proxies accept TCP streams and multiplex them across the circuits. The onion router on the other side of the circuit connects to the destinations of the TCP streams and relays data.

Traffic passes along these connections in fixed-size cells. Each cell is 512 bytes, and consists of a header and a payload. The header includes a circuit identifier (circID) that specifies which circuit the cell refers to (many circuits can be multiplexed over each TLS connection), and a command to describe what to do with the cell's payload. (Circuit identifiers are connection-specific: each single circuit has a different circID on each OP/OR or OR/OR connection it traverses.) Based on their command, cells are either *control* cells, which are always interpreted by the node that receives them, or *relay* cells, which carry end-to-end stream data.

Relay cells have an additional header (the relay header) after the cell header, containing a stream identifier (many streams can be multiplexed over a circuit); an end-to-end checksum for integrity checking; the length of the relay payload; and a relay command. The entire contents of the relay header and the relay cell payload are encrypted or decrypted together as the relay cell moves along the circuit, using the 128-bit AES cipher in counter mode to generate a cipher stream.

In Tor, just as each connection can be shared by many circuits, each circuit can be shared by many application-level TCP streams. To avoid delays, users construct circuits preemptively. To limit linkability among their streams, users' OPs build a new circuit periodically if the previous one has been used, and expire old used circuits that no longer have any open streams. OPs consider making a new circuit once a minute: thus even heavy users spend negligible time building circuits, but a limited number of requests can be linked to each other through a given exit node. Also, because circuits are built in the background, OPs can recover from failed circuit creation without delaying streams (which would harm user experience).

The full Tor design paper [10] describes the Onion Routing protocol in detail; we highlight a few of its properties here:

- **Perfect forward secrecy:** Onion Routing was originally vulnerable to a single hostile node recording traffic and later compromising successive nodes in the circuit and forcing them to decrypt it. Rather than using a single multiply encrypted data structure (an *onion*) to lay each circuit, Tor now uses an incremental or *telescoping* path-building design, where the initiator negotiates session keys with each successive hop in the circuit. Once these keys are deleted, subsequently compromised nodes cannot decrypt old traffic. As a side benefit, onion replay detection is no longer necessary, and the process of building circuits is more reliable, since the initiator knows when a hop fails and can then try extending to a new node.

- **Leaky-pipe circuit topology:** Through in-band signaling within the circuit, Tor initiators can direct traffic to nodes partway down the circuit. This novel approach allows traffic to exit the circuit from the middle—possibly frustrating traffic shape and volume attacks based on observing the end of the circuit. (It also allows for long-range padding if future research shows this to be worthwhile.)

- **End-to-end integrity checking:** The original Onion Routing design did no integrity checking on data. Any node on the circuit could change the contents of data cells as they passed by—for example, to alter a connection request so

it would connect to a different webserver, or to 'tag' encrypted traffic and look for corresponding corrupted traffic at the network edges [7]. Tor hampers these attacks by checking data integrity before it leaves the network.

- **Improved robustness to failed nodes:** A failed node in the old design meant that circuit building failed, but thanks to Tor's step-by-step circuit building, users notice failed nodes while building circuits and route around them. Additionally, liveness information from directories allows users to avoid unreliable nodes in the first place.
- **Congestion control:** Even with bandwidth rate limiting, we still need to worry about congestion, either accidental or intentional. If enough users choose the same OR-to-OR connection for their circuits, that connection can become saturated. For example, an attacker could send a large file through the Tor network to a webserver he runs, and then refuse to read any of the bytes at the webserver end of the circuit. Without some congestion control mechanism, these bottlenecks can propagate back through the entire network. We don't need to reimplement full TCP windows (with sequence numbers, the ability to drop cells when we're full and retransmit later, and so on), because TCP already guarantees in-order delivery of each cell. Tor provides both circuit and stream level throttling.

4 Other design decisions

4.1 Resource management and denial-of-service

Providing Tor as a public service creates many opportunities for denial-of-service attacks against the network. While flow control and rate limiting prevent users from consuming more bandwidth than routers are willing to provide, opportunities remain for users to consume more network resources than their fair share, or to render the network unusable for others. We discuss some of these in [10].

4.2 Exit policies and abuse

Exit abuse is a serious barrier to wide-scale Tor deployment. Anonymity presents would-be vandals and abusers with an opportunity to hide the origins of their activities. Attackers can harm the Tor network by implicating exit servers for their abuse. Also, applications that commonly use IP-based authentication (such as institutional mail or webservers) can be fooled by the fact that anonymous connections appear to originate at the exit OR.

We stress that Tor does not enable any new class of abuse. Spammers and other attackers already have access to thousands of misconfigured systems worldwide, and the Tor network is far from the easiest way to launch antisocial or illegal attacks. But because the onion routers can easily be mistaken for the originators of the abuse, and the volunteers who run them may not want to deal with the hassle of repeatedly explaining anonymity networks, we must block or limit the abuse that travels through the Tor network.

To mitigate abuse issues, in Tor, each onion router's *exit policy* describes to which external addresses and ports the router will connect. This is described further in [10].

Finally, we note that exit abuse must not be dismissed as a peripheral issue: when a system's public image suffers, it can reduce the number and diversity of that system's users, and thereby reduce the anonymity of the system itself. Like usability, public perception is a security parameter. Sadly, preventing abuse of open exit nodes is an unsolved problem, and will probably remain an arms race for the foreseeable future. The abuse problems faced by Princeton's CoDeeN project [14] give us a glimpse of likely issues.

4.3 Directory Servers

First-generation Onion Routing designs [4, 16] used in-band network status updates: each router flooded a signed statement to its neighbors, which propagated it onward. But anonymizing networks have different security goals than typical link-state routing protocols. For example, delays (accidental or intentional) that can cause different parts of the network to have different views of link-state and topology are not only inconvenient: they give attackers an opportunity to exploit differences in client knowledge, by observing induced differences in client behavior. We also worry about attacks to deceive a client about the router membership list, topology, or current network state. Such *partitioning attacks* on client knowledge help an adversary to efficiently deploy resources against a target [7].

Tor uses a small group of redundant, well-known onion routers to track changes in network topology and node state, including keys and exit policies. Each such *directory server* acts as an HTTP server, so participants can fetch current network state and router lists, and so other ORs can upload state information. Onion routers periodically publish signed statements of their state to each directory server. The directory servers combine this state information with their own views of network liveness, and generate a signed description (a *directory*) of the entire network state. Client software is preloaded with a list of the directory servers and their keys, to bootstrap each client's view of the network. More details are provided in [10].

Using directory servers is simpler and more flexible than flooding. Flooding is expensive, and complicates the analysis when we start experimenting with non-clique network topologies. Signed directories can be cached by other onion routers, so directory servers are not a performance bottleneck when we have many users, and do not aid traffic analysis by forcing clients to periodically announce their existence to any central point.

5 Rendezvous points and hidden services

Rendezvous points are a building block for *location-hidden services* (also known as *responder anonymity*) in the Tor network. Location-hidden services allow Bob to offer a TCP service, such as a webserver, without revealing its IP address. This type of anonymity protects against distributed DoS attacks: attackers are forced to attack the onion routing network as a whole rather than just Bob's IP address.

Our design for location-hidden servers has the following goals. **Access-controlled:** Bob needs a way to filter incoming requests, so an attacker cannot flood Bob simply by making many connections to him. **Robust:** Bob should be able to maintain a long-term pseudonymous identity even in the presence of router failure. Bob's service must not be tied to a single OR, and Bob must be able to tie his service to new ORs. **Smear-resistant:** A social attacker who offers an illegal or disreputable location-hidden service should not be able to "frame" a rendezvous router by making observers believe the router created that service. **Application-transparent:** Although we require users to run special software to access location-hidden servers, we must not require them to modify their applications.

We provide location-hiding for Bob by allowing him to advertise several onion routers (his *introduction points*) as contact points. He may do this on any robust efficient key-value lookup system with authenticated updates, such as a distributed hash table (DHT) like CFS [6][1] Alice, the client, chooses an OR as her *rendezvous point*. She connects to one of Bob's introduction points, informs him of her rendezvous point, and then waits for him to connect to the rendezvous point. This extra level of indirection helps Bob's introduction points avoid problems associated with serving unpopular files directly (for example, if Bob serves material that the introduction point's community finds objectionable, or if Bob's service tends to get attacked by network vandals). The extra level of indirection also allows Bob to respond to some requests and ignore others.

5.1 Integration with user applications

Bob configures his onion proxy to know the local IP address and port of his service, a strategy for authorizing clients, and a public key. Bob publishes the public key, an expiration time ("not valid after"), and the current introduction points for his service into the DHT, indexed by the hash of the public key. Bob's webserver is unmodified, and doesn't even know that it's hidden behind the Tor network.

Alice's applications also work unchanged—her client interface remains a SOCKS proxy. We encode all of the necessary information into the fully qualified domain name Alice uses when establishing her connection. Location-hidden services use a virtual top level domain called `.onion`: thus hostnames take the form `x.y.onion` where `x` is the authorization cookie, and `y` encodes the hash of the public key. Alice's onion proxy examines addresses; if they're destined for a hidden server, it decodes the key and starts the rendezvous as described above.

6 Future Directions

Tor brings together many innovations into a unified deployable system. The next immediate steps include:

Scalability: Tor's emphasis on deployability and design simplicity has led us to adopt a clique topology, semi-centralized directories, and a full-network-visibility model

[1]Rather than rely on an external infrastructure, the Onion Routing network can run the DHT itself. At first, we can run a simple lookup system on the directory servers.

for client knowledge. These properties will not scale past a few hundred servers. The Tor design paper [10] describes some promising approaches, but more deployment experience will be helpful in learning the relative importance of these bottlenecks.

Bandwidth classes: This paper assumes that all ORs have good bandwidth and latency. We should instead adopt the MorphMix model, where nodes advertise their bandwidth level (DSL, T1, T3), and Alice avoids bottlenecks by choosing nodes that match or exceed her bandwidth. In this way DSL users can usefully join the Tor network.

Incentives: Volunteers who run nodes are rewarded with potentially better anonymity, and those who value the notoriety can be rewarded with publicity [1]. More nodes means increased scalability, and more users can mean more anonymity. We need to continue examining the incentive structures for participating in Tor.

Padding Cover traffic: Currently Tor omits padding for cover traffic—its costs in performance and bandwidth are clear but its security benefits are not well understood. We must pursue more research on link-level cover traffic and long-range cover traffic to determine whether some simple padding method offers provable protection against our chosen adversary.

Caching at exit nodes: Perhaps each exit node should run a caching web proxy, to improve anonymity for cached pages (Alice's request never leaves the Tor network), to improve speed, and to reduce bandwidth cost. On the other hand, forward security is weakened because caches constitute a record of retrieved files. We must find the right balance between usability and security.

Better directory distribution: Clients currently download a description of the entire network every 15 minutes. As the state grows larger and clients more numerous, we may need a solution in which clients receive incremental updates to directory state. More generally, we must find more scalable yet practical ways to distribute up-to-date snapshots of network status without introducing new attacks.

Implement location-hidden services: The design in Section 5 has not yet been implemented. While doing so we are likely to encounter additional issues that must be resolved, both in terms of usability and anonymity.

Further specification review: Although we have a public byte-level specification for the Tor protocols, it needs extensive external review. We hope that as Tor is more widely deployed, more people will examine its specification.

Multisystem interoperability: We are currently working with the designer of MorphMix to unify the specification and implementation of the common elements of our two systems. So far, this seems to be relatively straightforward. Interoperability will allow testing and direct comparison of the two designs for trust and scalability.

Wider-scale deployment: The original goal of Tor was to gain experience in deploying an anonymizing overlay network, and learn from having actual users. As of writing there is a distributed network of roughly a dozen nodes. We are now at a point in design and development where we can start deploying a wider network. Once we have many actual users, we will doubtlessly be better able to evaluate some of our design decisions, including our robustness/latency tradeoffs, our performance tradeoffs (including cell size), our abuse-prevention mechanisms, and our overall usability.

References

[1] Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the economics of anonymity. In Rebecca N. Wright, editor, *Financial Cryptography*. Springer-Verlag, LNCS 2742, 2003.

[2] The Anonymizer. `http://anonymizer.com/`.

[3] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In Ira S. Moskowitz, editor, *Information Hiding (IH 2001)*, pages 245–257. Springer-Verlag, LNCS 2137, 2001.

[4] Philippe Boucher, Adam Shostack, and Ian Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., December 2000.

[5] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[6] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada, October 2001.

[7] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *2003 IEEE Symposium on Security and Privacy*, pages 2–15. IEEE CS, May 2003.

[8] T. Dierks and C. Allen. The TLS Protocol — Version 1.0. IETF RFC 2246, January 1999. `http://www.rfc-editor.org/rfc/rfc2246.txt`.

[9] Whitfield Diffie and Susan Landau. *Privacy On the Line: The Politics of Wiretapping and Encryption*. MIT Press, 1998.

[10] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Submitted for publication. Available at `http://freehaven.net/tor/`.

[11] John Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.

[12] Michael J. Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.

[13] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding routing information. In R. Anderson, editor, *Information Hiding, First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.

[14] Vivek S. Pai, Limin Wang, KyoungSoo Park, Ruoming Pang, and Larry Peterson. The Dark Side of the Web: An Open Proxy's View. Submitted to HotNets-II. `http://codeen.cs.princeton.edu/`.

[15] Privoxy. `http://www.privoxy.org/`.

[16] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[17] Marc Rennhard and Bernhard Plattner. Practical anonymity for the masses with morphmix. In Ari Juels, editor, *Financial Cryptography*. Springer-Verlag, LNCS (forthcoming), 2004.

[18] Paul Syverson, Michael Reed, and David Goldschlag. Onion Routing access configurations. In *DARPA Information Survivability Conference and Exposition (DISCEX 2000)*, volume 1, pages 34–40. IEEE CS Press, 2000.

[19] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Workshop on Design Issue in Anonymity and Unobservability*, pages 96–114. Springer-Verlag, LNCS 2009, July 2000.

A Policy Based Approach to Securing Egress Secure Socket Layer Connections on Local Area Networks

Joseph Mathews, James Rowell, David Nadwodny
US Naval Research Lab, ITT Industries
Center for High Assurance Computing Systems
4555 Overlook Ave SW
Washington, DC 20375
USA

mathews@itd.nrl.navy.mil / rowell@itd.nrl.navy.mil

ABSTRACT

Common network environments allow users a wide variety of protocols and applications to accomplish their job functions as well as day-to-day communications. One such example is the Secure Sockets Layer (SSL) protocol. SSL provides client and server authentication, data confidentiality and data integrity. SSL has been successfully employed in conjunction with a number of legacy protocols in order to ensure additional security. While many of these services are a requirement to complete basic mission-critical tasks, they can be manipulated in order to produce network activities that would normally be prohibited. SSL can be used to tunnel other applications or protocols and can therefore hide traffic and activity that would normally never be allowed out of a network. Traffic utilizing SSL is encrypted and cannot be screened by traditional methods of network defence for unauthorized activities. There is an increasing need to monitor and regulate all traffic in networked environments. Due to the confidentiality provided, SSL traffic offers a unique challenge to these requirements. We explore a policy-based interception solution that allows additional controls to be placed on egress SSL traffic. This solution will provide the ability to detect and prevent SSL misuse.

1.0 INTRODUCTION

In the early 1990s the Internet was growing while preparing to handle commercial applications. There was a clear need to provide secure Hyper Text Transfer Protocol (HTTP) connections. To fill this void, Netscape™ developed the Secure Sockets Layer (SSL) protocol. SSL resides between the transport layer and the higher layers of the Open Systems Interconnect (OSI) protocol stack. It allows for server authentication to a client and vice versa while providing an end-to-end secure channel between two nodes. While SSL was originally designed for HTTP, due to its non-application centric design, it has been used to secure many other cleartext legacy protocols and services. The wide spread acceptance of SSL has resulted in an abundance of egress encrypted traffic on Local Area Networks (LAN).

The SSL protocol is composed of two layers. The SSL Handshake Protocol enables the client and server to authenticate one another and negotiate encryption algorithms and cryptographic keys prior to transmission of application layer data. The SSL Record Protocol resides on top of the Transmission Control Protocol (TCP) layer and cryptographically encapsulates data from higher level application layer protocols. The encapsulated data is then transmitted over a network as the payload of an SSL packet.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

Since virtually any network application can be tunnelled over SSL, an opportunity is created for a user to send traffic out of a network that would normally not be permitted. This could include connections to unauthorized web sites, transference of sensitive data, and connections to Internet Relay Chat (IRC) servers. To make matters worse, common network security tools like application layer proxies and Network Intrusion Detection Systems (NIDS) lack the ability to effectively prevent or even detect this sort of activity. In order to mitigate this threat, the traffic must be put in its cleartext form allowing for a determination to be made of whether or not it is acceptable for transmission to other networks. This can be accomplished using what is traditionally known as a Man-In-the-Middle (MITM) technique and applying a rules engine to SSL traffic for analysis.

2.0 SSL TUNNELLING

SSL is compatible with any network protocol that runs over the TCP layer of the OSI stack. Therefore, it is possible to set up an SSL tunnel and transmit a wide array of data over that tunnel. This enables any user capable of sending outbound SSL traffic to send any type of data out of the network.

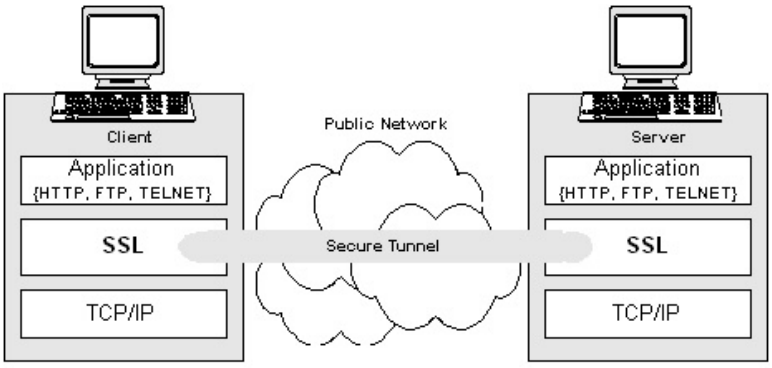


Figure 1: SSL Operation

As depicted in Figure 1, once an SSL tunnel is established between two hosts, any sort of application layer protocol may be encrypted and transmitted over an insecure medium such as a public network. Because the data is encrypted, it is not possible to determine what sort of data is being transmitted.

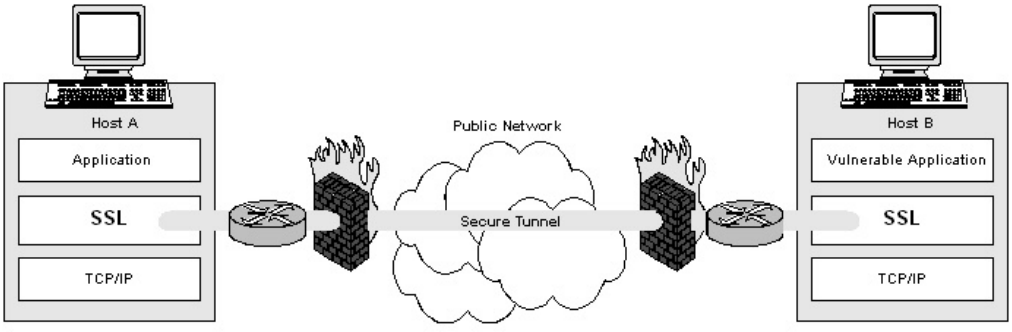


Figure 2: SSL Tunnelling

As depicted in Figure 2, Host A has the ability to exploit vulnerabilities in Host B (and vice versa) without network defences at either end of the path detecting the malicious activity due to the secure tunnel in which the packets travel. At this point either host could be attacked or even compromised without being detected. As shown, SSL has the potential to severely degrade the integrity of network perimeter defences that are deployed. It is also important to note that SSL tunnels can be established to numerous ports on multiple hosts.

3.0 TRADITIONAL MEASURES OF PREVENTION

SSL traffic is largely unmanageable using the suite of available network security tools and devices. At the most exhaustive level, payload analysis of every packet on the wire yields gibberish when dealing with encrypted traffic. Smarter sensors are limited to monitoring the state of the connection itself while implicitly trusting the encrypted payloads once an SSL connection has been successfully negotiated. To illustrate, we examine two of the most popular methods of network defences: firewalls and intrusion detection systems.

3.1 Firewalls

Standard packet filters and stateful inspection firewalls have no feature suitable to defend against SSL misuse. Application layer proxies, often integrated into firewalls, have the ability to compare traffic against the protocol specification for a particular application, but this is ineffective for SSL because traffic is encrypted after the initial handshake is made, preventing the firewall from seeing what application data is actually going out over a port. They are limited to monitoring the details of the connection itself, such as the encryption algorithm utilized, rather than the payload.

3.2 Network Intrusion Detection Systems

Network Intrusion Detection Systems will listen to traffic traversing a network in an effort to identify malicious traffic. These are effective tools for monitoring malicious activity such as network reconnaissance scanning and denial-of-service (DOS) attacks. Sensors may employ multiple methodologies to detect attacks such as signature analysis, protocol decoding, or anomaly detection. Once an attack is detected, the engine will provide a variety of options to notify, alert, or log with respect to the event at hand.

NIDS are equally ineffective at detecting SSL misuse. Despite the multiple detection methods they may utilize, they lack the ability to analyze encrypted payloads for attack signatures or traffic anomalies rendering them useless for SSL management. Other methods of intrusion detection exist, such as monitoring the state of connections between hosts, but are not as effective.

4.0 POLICY BASED INTERCEPTION

By creating a proxy point in a network for all outbound SSL connections and intercepting the SSL handshake, it is possible to decrypt SSL traffic, analyze it, and determine whether it is permitted to leave the network enclave. There are two tasks for such a system: interception of the SSL traffic, and a policy based analysis of the traffic in its cleartext form.

4.1 SSL Interception

The interception method for SSL is relatively well known. A form of the Man-in-the-Middle (MITM) technique, the method involves intercepting the outgoing SSL handshake from a client to server, forging the server's reply back to the client, and then forwarding the traffic along to the actual destination. This enables the SSL traffic leaving the network to be seen in cleartext while passing through a proxy, prior to being forwarded to its destination. Figure 3 depicts an SSL connection leaving the client and getting through the proxy as the steps listed below illustrate.

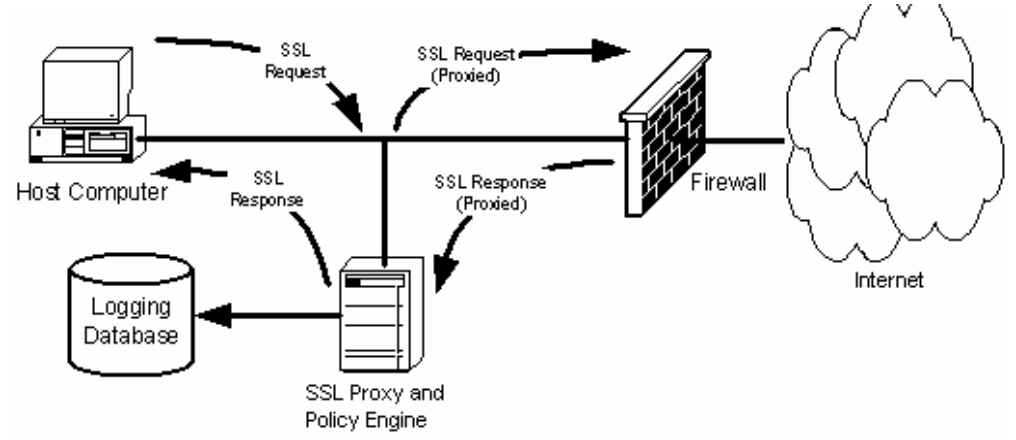


Figure 3: SSL Interception

- The client will make a request for an SSL connection with an SSL handshake.
- Traffic is routed to the SSL proxy by dynamically routing traffic based on its destination port.
- The proxy masquerades as the end point and completes the SSL handshake with the server.
- The proxy masquerades the server's certificate and presents it to the client. Depending on the application service requested the client would receive a warning concerning the server's certificate. This can be eliminated by adding the proxy's certificate to the client's trusted Certificate Authority (CA) list.
- Once the masqueraded certificate is accepted, the proxy completes the SSL handshake with the client's original destination.
- After the proxy and the client's destination complete the handshake process, the proxy is receiving encrypted traffic from the client, which is then passed through the proxy in cleartext and encrypted again before being forwarded to the client's actual destination.

4.2 Policy Engine

Figure 4 illustrates the operation of the SSL proxy. Incoming packets are decrypted and examined in cleartext form by the policy engine using the rules list as a reference for legitimate sites and activities. Legitimate SSL activity is encrypted again and forwarded to the original location while non-compliant data is prevented from leaving the network and an alert is written to an event database. Security administrators may view this database through a console in order to better understand the types of malicious activity and invalid uses of SSL occurring on their network.

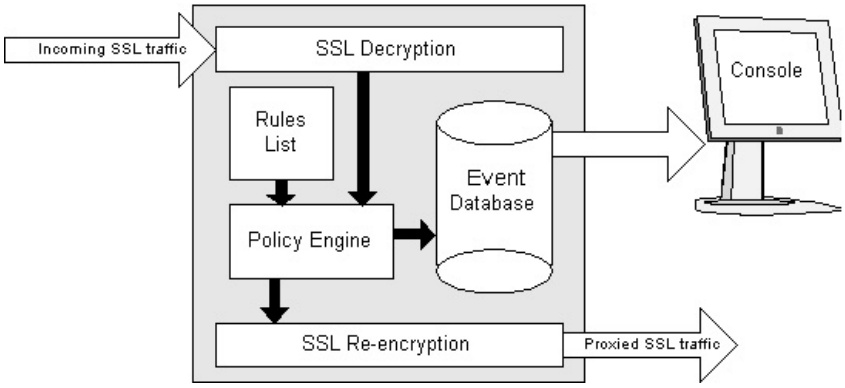


Figure 4: Policy Engine

The policy engine's rules list for a protocol should contain logic similar to that of an application proxy, checking for the correct application layer protocol running over SSL. It may also be tailored to accommodate the particular environment in which the system is employed. For example, the rules list should include a list of acceptable end points for SSL connections which can be left open (outgoing SSL connections are allowed to any host), or in an extreme case statically set to include only a few outbound destinations. Certain hosts on the network may want to be implicitly trusted and not monitored at all. All outgoing SSL traffic may be archived. Or perhaps only traffic from particular hosts or to particular destinations may be archived. It may be desirable in certain circumstances to only log the connection instance. A powerful application of this policy based engine would be to archive all egress SSL traffic while forwarding the traffic in its cleartext form to an IDS sensor which could then apply its own method of analysis. This will prevent unauthorized application tunnelling and aid the mitigation of attempted exploits.

5.0 CONCLUSION

The confidentiality provided by SSL is essential for many network connections, but the nature of encrypted traffic and the inability to effectively control and monitor egress SSL traffic present serious network security risks. It is possible to mitigate many of the risks associated with allowing egress SSL traffic by leveraging a traffic interception technique while applying the operation of a policy engine.

6.0 REFERENCES

[1] McClure, Stuart. Hacking Exposed: Network Security Secrets and Solutions. McGraw-Hill Osborne Media, Feb 2003.

[2] Northcutt, Stephen. Inside Network Perimeter Security. New Riders, Jul 2003.

[3] Transport Layer Security Working Group. The SSL Protocol: Version 3.0. INTERNET-DRAFT, Netscape Communications. Nov 1996. Available: http://wp.netscape.com/eng/ssl3/draft302.txt

Dealing with System Monocultures

Angelos Keromytis
Computer Science Departmet
Columbia University
New York, NY, USA
angelos@cs.columbia.edu

Vassilis Prevelakis
Computer Science Department
Drexel University
Philadelphia, PA, USA
vp@cs.drexel.edu

ABSTRACT

Software systems often share common vulnerabilities that allow a single attack to compromise large numbers of machines (write once, exploit everywhere). Borrowing from biology, several researchers have proposed the introduction of artificial diversity in systems as a means for countering this phenomenon. The introduced differences affect the way code is constructed or executed, but retain the functionality of the original system. In this way, systems that exhibit the same functionality have unique characteristics that protect them from common mode attacks. Over the years, several such have been proposed. We examine some of the most significant techniques and draw conclusions on how they can be used to harden systems against attacks.

1. INTRODUCTION

The recent widespread disruptions of systems across the Internet underlined the inherent weakness of an infrastructure that relies on large numbers of effectively identical systems. Common elements in these systems include the operating system, the system architecture (e.g., Intel Pentium), particular applications (e.g., email, Word processing software), and the internal network architecture. Common-mode attacks occur when an attacker exploits vulnerabilities in one of these common elements to strike large numbers of victim machines. If each of these systems were different, then the attacker would have to customize their technique to the peculiarities of each system, thus reducing the scope of the attack and the rate of its spread.

However, running different systems in a network creates its own set of problems involving configuration, management and certification of each new platform. In certain cases, running such multi-platform environments can decrease the overall security of the network [1]. The premise of this paper is that by introducing randomness in existing systems we can vary their behavior sufficiently to prevent common mode attacks. Thus, our systems are similar enough to ease administration, but sufficiently different to resist common mode attacks.

Randomization can be introduced in various parts of a system. Areas include the configuration of the network infrastructure so that remote attackers cannot target a specific host or service (e.g., the White House site or the Microsoft software update server), the implementation of specific protocols (e.g., changing some aspects of the TCP/IP engine to reduce the risk of fingerprinting), or even the processor architecture to guard against foreign code injections attacks. In this paper, we describe various randomization techniques and examine how they can be used to strengthen the security of systems.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

2. CLASSIFICATION

The diversification techniques that have been proposed over the years can be broadly classified into three categories: those that modify the structure of the system, those that modify the execution environment, and those that affect the system behavior. For example, systems such as StackGuard insert code that verifies the integrity of the stack every time the code returns from a subroutine call, whereas the Instruction-Set Randomization technique changes the instruction set of the processor so that unauthorized code will not run successfully.

2.1 Modifying the Structure

Structure modification techniques insert special code that performs sanity or consistency checks at various points in the execution of the program.

Perhaps the best-known of these techniques is StackGuard [2], a system that protects against buffer overflows. This is a patch to the popular gcc compiler that inserts a canary word right before the return address in a function's activation record on the stack (Figure 1). The canary is checked just before the function returns, and execution is halted if it is not the correct value, which would be the case if a stack-smashing attack had overwritten it. This protects against simple stack-based attacks, although some attacks were demonstrated against the original approach [3], which has since been amended to address the problem.

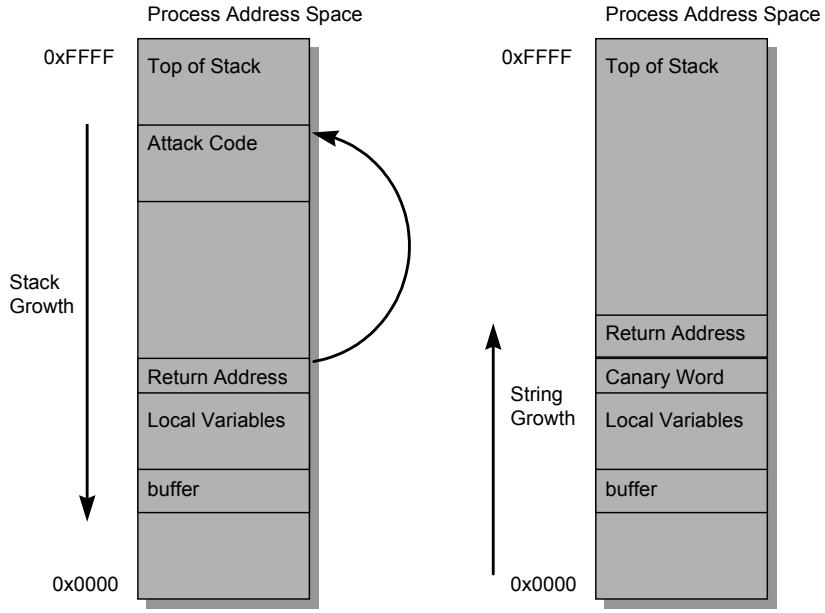


Figure 1: Stack buffer overflow (left) and StackGuard-modified stack frame (right).

Stack Guard is one of many similar systems such as MemGuard [2], FormatGuard [4], ProPolice [5], etc. Generally, these approaches have three limitations. First, the performance implications (at least for some of them) are non-trivial. Second, they do not seem to offer sufficient protection against stack-smashing attacks on their own, as shown in [3, 6] (although work-arounds exist against some of the attacks). Finally, they do not protect against other types of code-injection attacks, such as heap overflows [7]. For the purposes of our

discussion, however, these techniques have the problem that they make deterministic changes to the code, and thus cannot protect against monoculture threats.

A system that is more applicable to this discussion is PointGuard [8] which encrypts all pointers while they reside in memory and decrypts them only before they are loaded to a CPU register. This is implemented as an extension to the gcc compiler, which injects the necessary instructions at compilation time, allowing a pure-software implementation of the scheme. Another approach, address obfuscation [9], randomizes the absolute locations of all code and data, as well as the distances between different data items. Several transformations are used, such as randomizing the base addresses of memory regions (stack, heap, dynamically-linked libraries, routines, static data, etc.), permuting the order of variables/routines, and introducing random gaps between objects (e.g., randomly pad stack frames or malloc'ed regions). Although very effective against jump-into-libc attacks, it is less so against other common attacks, due to the fact that the amount of possible randomization is relatively small. However, address obfuscation can protect against attacks that aim to corrupt variables or other data.

A persistent concern in employing techniques such as the ones described above, is to maintain the efficiency of the application. In other words, the overheads associated with the use of these mechanisms must be minimized. Naturally, this discourages the use of more exhaustive and hence more expensive techniques. If, however, we can identify the parts of the code where a bug has a higher probability of resulting in a security vulnerability, we can reserve the use of the more expensive mechanisms to these sensitive regions.

Tools developed under the DARPA funded CHATS/CoSAK project facilitate the identification of such regions. This work is based on the assumption that a small percentage of functions near a source of input (such as file I/O), called *Inputs*, are the most likely to contain a security vulnerability [17]. The original hypothesis was confirmed by reviewing large numbers of bugs that have been posted in security forums such as the CERT. These reports also include the patches that correct the bugs, thus identifying the code that was responsible for the vulnerability (called *Targets*). The analysis of the existing systems revealed that Targets tend to be located "near" Inputs (where "near" is defined as a number of function calls). With this information, new systems can be analyzed by the CoSAK tools. The way they work is by examining the source code of a computer system in order to identify the Inputs. Then a call graph of the entire system is generated and the code appearing within a set number of function calls from the Inputs is pinpointed. Special mechanisms (e.g. code emulation, execution under a virtual environment, or limitations on privileges) can be activated when the flow of control strays into the sensitive regions.

2.2 Modifying the Environment

Systems do not exist in isolation but they need to interact with their environment (be it the processor architecture, the operating system, the network topology, etc.). To see how randomization techniques can be used to influence the execution environment let us look a bit closer at the problem of code-injection attacks.

Code-injection attacks attempt to deposit executable code (typically machine code, but there are cases where intermediate or interpreted code has been used) within the address space of the victim process, and then pass control to this code. These attacks can only succeed if the injected code is compatible with the execution environment. For example, injecting x86 machine code to a process running on a SUN/SPARC system may crash the process (either by causing the CPU to execute an illegal op-code, or through an illegal memory reference), but will not cause a security breach. Notice that in this example, there may well exist sequences of bytes that will crash on neither processor.

The instruction randomization technique [11] leverages this observation by creating an execution environment that is unique to the running process, so that the attacker does not know the "language" used and hence cannot "speak" to the machine. This is achieved by applying a reversible transformation between the processor and main memory. Effectively, new instruction sets are created for each process executing within the same system. Code-injection attacks against this system are unlikely to succeed as the attacker cannot guess the transformation that has been applied to the currently executing process. Of course, if the attackers had access to the machine and the randomized binaries through other means, they could easily mount a dictionary or known-plaintext attack against the transformation and thus "learn the language". However, we are primarily concerned with attacks against remote services (e.g., http, dhcp, DNS, and so on). Vulnerabilities in this type of server allow external attacks (i.e., attacks that do not require a local account on the target system), and thus enable large-scale (automated) exploitation. Protecting against internal users is a much more difficult problem, which we do not address in this work.

The power of the technique can be demonstrated by its applicability to other settings, such as SQL injection attacks. Such attacks target databases that are accessible through a web front-end, and take advantage of flaws in the input validation logic of Web components such as CGI scripts. The concept of instruction randomization has been applied to that setting, to create instances of the SQL language that are unpredictable to the attacker. Preliminary results indicate that the mechanism imposes negligible performance overhead to query processing, and can be easily retrofitted to existing systems. The same technique can easily be applied to any interpreted-language setting that is susceptible to code injection attacks.

In a different context, randomization of a system's environment has been used to combat network-based denial of service (DoS) attacks. The Secure Overlay Services (SOS) [18] approach addresses the problem of securing communication on top of today's existing IP infrastructure from DoS attacks, where the communication is between a predetermined location and users, located anywhere in the wide-area network, who have authorization to communicate with that location. The scheme was later extended to support unknown users, by using Graphic Turing Tests to discriminate between zombie machines and real humans [19].

In a nutshell, the portion of the network immediately surrounding the target (location to be protected) aggressively filters and blocks all incoming packets whose source addresses are not "approved". The small set of source addresses that are "approved" at any particular time is kept secret so that attackers cannot use them to pass through the filter. These addresses are picked from among those within a distributed set of nodes throughout the wide area network, that form a secure overlay: any transmissions that wish to traverse the overlay must first be validated at entry points of the overlay. Once inside the overlay, the traffic is tunneled securely for several hops along the overlay to the "approved" (and secret from attackers) locations, which can then forward the validated traffic through the filtering routers to the target. The two main principles behind this design are: (i) elimination of communication "pinch" points, which constitute attractive DoS targets, via a combination of filtering and overlay routing to obscure the identities of the sites whose traffic is permitted to pass through the filter, and (ii) the ability to recover from random or induced failures within the forwarding infrastructure or among the overlay nodes.

The overlays are secure with high probability, given attackers who have a large but finite set of resources to perform the attacks. The attackers also know the IP addresses of the nodes that participate in the overlay and of the target that is to be protected, as well as the details of the operation of protocols used to perform the forwarding. However, the assumption is that the attacker does not have unobstructed access to the network core. That is, the model allows for the attacker to take over an arbitrary (but finite) number of hosts, but only a small number of core routers. It is more difficult (but not impossible) to take control of a router than an end-host or server, due to the limited number of potentially exploitable services offered by the former. While

routers offer very attractive targets to hackers, there have been very few confirmed cases where take-over attacks have been successful. Finally, SOS assumes that the attacker cannot acquire sufficient resources to severely disrupt large portions of the backbone itself (i.e., such that all paths to the target are congested).

Under these assumptions, by periodically selecting a new "approved" overlay node at random, a site can allow only authorized clients to communicate with it. An attacker must either amass enough resources to subvert the infrastructure itself, or attempt to guess the identity of the current approved node. Effectively, SOS allows the creation and use of an arbitrary number of virtual topologies over the real network (which can, perhaps perversely, viewed as a monoculture), which only legitimate users can use. The performance impact of doing so is studied in [19]. To summarize, end-to-end latency is increased by a factor of 2, while remaining impervious to the effects of a DoS attack.

2.3 Modifying the Behavior

Computer systems are to a large extent deterministic and this can be used as a means of identification (fingerprinting), or, worse, as means of subverting a system by anticipating its response to various events.

Fingerprinting is a technique that allows remote attackers to gather enough information about a system so that they can determine its type and software configuration (version of operating system, applications etc.) [14]. This information can then be used to determine what vulnerabilities may be present in that configuration and thus better plan an attack.

Having a system with predictable behavior can have devastating consequences for its security. The most celebrated example is the attack that exploited easy to guess TCP/IP packet sequence numbers [15]. By being able to guess the sequence number of a TCP connection with a remote system, we can construct and transmit replies to packets that we never receive (perhaps because a firewall prevents the remote system from talking to us, or because we use a spoofed source address in our packets).

More recently, a denial of service attack based on the TCP retransmission time-out [16], allowed an attacker to periodically send bursts of packets to the victim host, forcing the TCP subsystem on the victim host to repeatedly time-out causing near-zero throughput. In this case as well, by changing the behavior of the TCP implementation (randomizing the retransmission time-out), the attack can be mitigated.

The general Internet philosophy of "being conservative in what you send and liberal in what you accept" (RFC1341), while enhancing interoperability, sometimes creates vulnerabilities by allowing greater ambiguity in what a networked application may accept. Especially in the case of the Internet Protocols these minor variations have been used as the basis of attacks (e.g. the overlapping fragment attacks and the small packet attacks of the early 90s), and more recently as a means to facilitate fingerprinting.

OpenBSD's packet filter, pf (4), includes a "scrub" function that normalizes and defragments incoming packets. This allows applications and hosts on the internal network some form of protection against hand-crafted packets designed to trigger vulnerabilities. Another approach is to apply a similar technique to outgoing packets in order to hide identifying features of the IP stack implementation [20]. A key part of the process of the obfuscation process is protection against time-dependent probes. Different TCP implementations have variations in their time-out counters, congestion avoidance algorithms, etc. By monitoring the response of the host under inspection to simulated packet loss, the timing probe can determine the version of the TCP implementation and by extension that of the OS. Also the use of various techniques for rate limiting ICMP messages by the victim system, can provide hints to the attacker. The effectiveness of such

probes can be reduced, by homogenizing the rate of ICMP traffic going through the system that connects the trusted network to the outside world, or by introducing random delays to ICMP replies.

3. SUMMARY AND CONCLUDING REMARKS

The commoditization of computer systems has dramatically lowered the cost of ownership of large collections of computers. It is thus no longer economically feasible to have one-off configurations for individual computers or networks, which, in turn, leads to monocultures, vulnerable to common-mode attacks. There is a lively debate going on as to the effects of a diverse computing environment on security. One camp claims that diversity is not required as it distracts from the task of producing a single secure configuration that can then be widely deployed, thus spreading the development and security administration costs to a large number of machines. The other camp claims that by standardizing the interfaces between subsystems, multiple implementations can be deployed, thus reducing the risk of a single problem affecting all the deployed systems. Our view is that both sides are fundamentally wrong. Having potentially huge numbers of identically configured hosts invites disaster: no amount of effort can secure large software systems that have not been built with security in mind. Even in cases where formal methods have been used in the design, field upgrades and maintenance can weaken the security posture. On the other hand, attempting to introduce diversity through the development of different software systems is not viable. Designing, developing and maintaining a system is so expensive that once we have a working version we tend to use it widely. Even in critical systems such as avionics, the same software is used on multiple hardware platforms (creating redundancy only at the hardware level). The failure of the inaugural flight of the Ariane 5 launcher due to a software bug crashing both navigation computers is proof that having the same software running on redundant hardware does not provide true redundancy.

Our intention has been to demonstrate that the *effects* of diversity can be introduced through automated means. The techniques described in this paper allow the introduction of small but critical variations to the these off-the-shelf systems. While randomization is by no means the silver bullet that will solve the problem of generic software, or system exploits (these can only begin to be addressed if we abandon the current ad hoc design and development techniques) they do provide an effective method for mitigating attacks and exposing the bugs that make such attacks possible.

4. REFERENCES

1. Prevelakis, V.: A secure station for network monitoring and control. In: Proceedings of the 8th USENIX Security Symposium. (1999)

2. Cowan, C., Pu, C., Maier, D., Hinton, H., Walpole, J., Bakke, P., Beattie, S., Grier, A., Wagle, P., Zhang, Q.: Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks. In: Proceedings of the 7th USENIX Security Symposium. (1998)

3. Bulba, Kil3r: Bypassing StackGuard and StackShield. Phrack 5 (2000)

4. Cowan, C., Barringer, M., Beattie, S., Kroah-Hartman, G.: FormatGuard: Automatic Protection From printf Format String Vulnerabilities. In: Proceedings of the 10th USENIX Security Symposium. (2001) 191-199

5. Etoh, J.: GCC extension for protecting applications from stack-smashing attacks. http://www.trl.ibm.com/projects/security/ssp/ (2000)

6. Wilander, J., Kamkar, M.: A Comparison of Publicly Available Tools for Dynamic Intrusion Prevention. In: Proceedings of the Symposium on Network and Distributed Systems Security (SNDSS). (2003) 123-130

7. M. Conover and w00w00 Security Team: w00w00 on heap overflows. http://www.w00w00.org/files/articles/heaptut.txt (1999)

8. Cowan, C., Beattie, S., Johansen, J., Wagle, P.: PointGuard: Protecting Pointers From Buffer Overflow Vulnerabilities. In: Proceedings of the 12th USENIX Security Symposium. (2003) 91-104

9. Bhatkar, S., DuVarney, D.C., Sekar, R.: Address Obfuscation: an Efficient Approach to Combat a Broad Range of Memory Error Exploits. In: Proceedings of the 12th USENIX Security Symposium. (2003) 105-120

10. DaCosta, D., Dahn, C., Mancoridis, S., Prevelakis, V.: Characterizing the Security Vulnerability Likelihood of Software Functions . In: Proceedings of the 2003 International Conference on Software Maintenance (ICSMy03). (2003) 61-72

11. Kc, G.S., Keromytis, A.D., Prevelakis, V.: Countering Code-Injection Attacks With Instruction-Set Randomization. In: Proceedings of the ACM Computer and Communications Security (CCS) Conference. (2003)

12. Barrantes, G., Ackley, D., Palmer, T., Zovi, D.D., Forrest, S., Stefanovic, D.: Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks. In: Proceedings of the ACM Computer and Communications Security (CCS) Conference. (2003)

13. Keromytis, A.D., Misra, V., Rubenstein, D.: Secure overlay services. In: Proceedings of the ACM SIGCOMM Conference. (2002) 61-72

14. Smart, M., Malan, R., Jahanian, F.: Defeating TCP/IP Stack Fingerprinting. In: Proceedings of the 9th USENIX Security Symposium. (2000) 229-240

15. Inc, S.N.: A simple TCP spoofing attack. http://niels.xtdnet.nl/papers/secnet-spoof.txt (1997)

16. Yang, G.: Low-rate denial-of-service (DoS) attacks to TCP. http://www.cs.ucla.edu/yangg/research/research.htm (2003)

17. DaCosta, D., Dahn, C., Mancoridis, S., Prevelakis, V.: Characterizing the Security Vulnerability Likelihood of Software Functions . In: Proceedings of the 2003 International Conference on Software Maintenance (ICSMy03). (2003) 61-72.

18. Keromytis, A.D., Misra, V., Rubenstein, D.: Secure overlay services. In: Proceedings of the ACM SIGCOMM Conference. (2002) 61-72.

19. Morein, W.G., Stavrou, A., Cook, D.L., Keromytis, A.D., Misra, V., Rubenstein, D.: Using Graphic Turing Tests to Counter Automated DDoS Attacks Against Web Servers. In: Proceedings of the 10th ACM International Conference on Computer and Communications Security (CCS). (2003) 8-19.

20. Smart, M., Malan, R., Jahanian, F.: Defeating TCP/IP Stack Fingerprinting. In: Proceedings of the 9th USENIX Security Symposium. (2000) 229-240

Building a Trusted Path for Applications Using COTS Components

Hanno Langweg
Norwegian Information Security Laboratory – NISlab[1]
Department of Computer Science and Media Technology, Gjøvik University College
P.O. Box 191, 2802 Gjøvik
Norway

hanno.langweg@hig.no

ABSTRACT

Client computers are often a weak link in a technical network infrastructure. Increasing the security of client systems and applications against malicious software attacks increases the security of the network as a whole.

Our work solves integrity and authenticity of input, confidentiality, integrity and authenticity of output.

We present components to integrate a trusted path into an application to directly communicate with a user at a personal computer. This allows security sensitive parts of applications to continue operating while being attacked with malicious software in an event-driven system. Our approach uses widely employed COTS software – DirectX – and can be varied in design and implementation, hence making it more difficult to defeat with generic attack tools.

1.0 INTRODUCTION

Client computers are often a weak link in a technical network infrastructure. Confidentiality and integrity of connections between nodes in a network can be secured employing strong cryptography. However, this does not help against attacks by malicious software. Trojan horse programs, i.e., programs with additional hidden, often malicious, functions, are more and more popular forms of attack. These assail the endpoints of secured transactions. A vulnerable interface is the interaction of an application with the physically present user.

Examples for a direct user interaction are the creation of electronic signatures, online voting, financial transaction processing, communication software etc. The concept for this is not new and is called a trusted path. A trusted path exists between the physically present user and the operating system, e.g., when invoking the logon process.

In the Clark/Wilson access control model (cf. Clark et al. 1987), integrity is protected by transformation procedures that can not be bypassed. Applications manipulating data in existing systems could be seen as these procedures. If they are not able to tell apart another program from a user, untrusted processes could manipulate data that is assumed to be protected by a special access path.

Current event-driven systems, especially the Microsoft Windows operating system, do provide little to distinguish between users and other processes. Input can be simulated, output can not be authenticated and can be captured by other processes.

[1] This work was partly completed while the author was working at Department of Computer Science III, University of Bonn, Germany

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

In this paper we deal with integrity and authenticity of input, confidentiality, integrity and authenticity of output. We use existing COTS (commercial off the shelf) software, originally deployed as an interface for game developers, to assemble a trusted path. It is offered as a set of components to application developers.

This paper is organized as follows. We discuss previous and related work and give an overview of the standard Microsoft Windows input and output model. We then present components obtaining input and output by different means. This is followed by a discussion on retrofitting existing applications with a trusted path. An examination of the security of our approach concludes the presentation.

It may be important to note that we focus on architectural vulnerabilities of platforms and applications. We do not cover vulnerabilities that stem from flaws in an implementation.

2.0 PREVIOUS AND RELATED WORK

User interface security has always been an issue. In the Orange Book (1985) a Trusted path was required to establish a secure communication between the user and the operating system. It is there defined as follows: "Trusted Path – A mechanism by which a person at a terminal can communicate directly with the Trusted Computing Base. This mechanism can only be activated by the person or the Trusted Computing Base and cannot be imitated by untrusted software."

Wiseman et al. (1988) propose a user interface for the SMITE system to prevent Trojan horses from tampering with an application's output. Rooker (1993) questions a focus on the operating system. In his view applications should play a more active role in enforcing security. Operating systems should provide object-oriented support for trusted user interfaces and security embedded in applications.

In the Microsoft Windows operating system, applications typically receive information about user actions by way of messages. Since these messages can be sent by malicious applications as well, this turns out to be a convenient vector of attack.[2] It is a vulnerability by design. In Microsoft's view, all applications assigned to the same desktop are treated equally. If a program needs an undisturbed interface, it should be assigned a separate desktop. This approach is pursued by Balfanz (2001). However, managing separate desktops can be cumbersome for developers. So most of today's software that interacts with a user sitting at the machine runs in a single desktop shared by benign and malign programs.

This problem is encountered by local security applications such as virus scanners, personal fire walls etc. Schmid et al. (2002) point out their dilemma when notifying the user about a security event. The user is notified about the presence of a possibly malicious application that could hide that notification instantaneously. Xenitellis (2002a, 2002b) discusses Windows messages in event-driven systems in general and laments a lack of authentication. He proposes a rigorous filtering of messages that could be harmful to an application or separating applications from each other, thereby reducing co-operation among them. The straight-forward alternative, outlined by Spalka et al. (2002), would be to add an authenticated origin to messages. It would require changes in the decade-old messaging system and is hence unlikely to be adopted by the manufacturer of the operating system. In the X-Windows system, a radical approach is pursued, allowing to disable conveyance of all messages placed by the *SendEvents* function (cf. Bråthen 1998). There may be occasions, like computer-based training, in which remote control of another application or parts of it is desired. Only a fraction of all applications expose an interface by which they can be explicitly automated. Consequently, simulating user input is a quick and convenient way for small helper applications.

Paget (2002) and Howard (2002) remind us that messages can be sent between processes running in different security contexts. Says Howard: 'In the Windows user interface, the desktop is the security

[2] Cult of the Dead Cow (2003). *Back Orifice 2000.* http://bo2k.sourceforge.net

boundary, and any application running on the interactive desktop can interact with any window on the interactive desktop, even if that window is invisible. This is true regardless of the security context of the application that creates the window and the security context of the application.'

Carlisle et al. (2001) investigate some improvements for dialog-based security. Application output should be defended against hiding. Actions should be delayed so that a user could interfere when a program is controlled by simulated input. Scripting and automatic collection of information about the user interface should be restricted. Langweg (2002) uses the DirectX interface to distinguish between key strokes and simulated Windows messages. Output is orchestrated by DirectX instead of the co-operative Windows GDI. Ye et al. (2002) advocate a modified web browser to convey meta-information to the user about which browser windows can be trusted.

3.0 ARCHITECTURAL OVERVIEW

3.1 Windows Input Model

Microsoft Windows uses an internal messaging model to control Windows applications. Messages are generated whenever an event occurs. For example, when a user presses a key on the keyboard and releases it or moves the mouse, a message is generated by the operating system. The message is then placed in the message queue for the appropriate thread. An application checks its message queue to retrieve messages.[3]

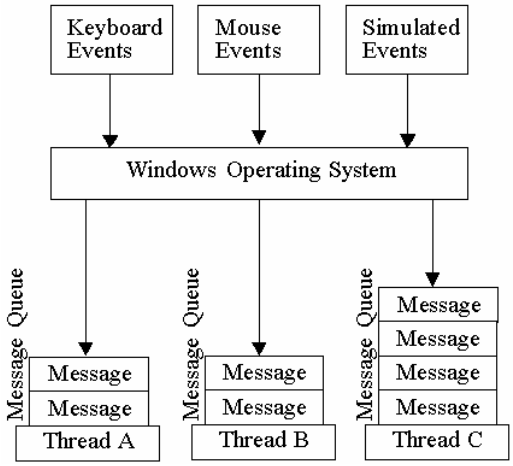


Figure 1: Input processing on a Windows desktop

The system passes all input for an application to the various windows in the application. Each window has a function, called a window procedure, that the system calls whenever it has input for the window. The window procedure processes the input and returns control to the system. All aspects of a window's appearance and behaviour depend on the window procedure's response to these messages.

In the model, it is not possible to distinguish between messages placed in the queue by the operating system and messages placed by another application. To make it even worse, ordinary programs can synthesize input by help of the SendInput API function (keybd_event, mouse_event prior to NT4 SP3). This synthesized input is processed by the operating system into messages for an application. This was originally intended to assist users in operating an application by different input facilities other than the

[3] Microsoft (1998). *Microsoft Windows Architecture for Developers Training Kit.*

standard keyboard and mouse, e.g., assistive technology for users with disabilities. It is also a convenient tool for malicious programs.

3.2 DirectX

Microsoft DirectX is a group of technologies designed by Microsoft to make Microsoft Windows-based computers an ideal platform for running and displaying applications such as games. Built directly into Windows operating systems, DirectX is an integral part of Windows 98, Windows Me, and Windows 2000/XP.

DirectX gives software developers a consistent set of APIs that provides them with improved access to hardware. These APIs control what are called "low-level functions," including graphics memory management and rendering, and support for input devices such as joysticks, keyboards, and mice. The low-level functions are grouped into components that make up DirectX: Microsoft Direct3D, Microsoft DirectDraw, Microsoft DirectInput, to name just a few.[4] In this paper we are concerned with DirectInput and DirectDraw.

DirectInput retrieves information before it is distilled by the operating system to Windows messages. Hence, input synthesized by placing a forged message in a program's message queue is ignored.

DirectDraw allows to access the display hardware in exclusive mode, keeping other programs from distorting the information presented to the user.

In the sketch it is shown how an application actually transfers its output to the screen. Without DirectX it uses the GDI (Graphical Device Interface) and the DDI (Display Driver Interface). With DirectX, namely its DirectDraw part, the DDI is bypassed in favour of the HAL (Hardware Abstraction Layer). If there is no direct hardware support, the HEL (Hardware Emulation Layer) is used instead.[5]

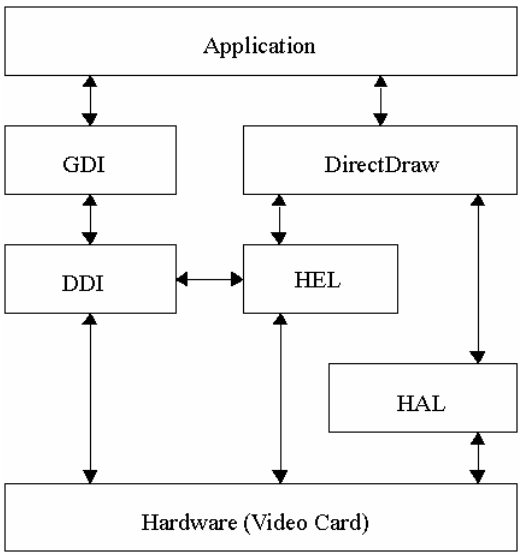


Figure 2: Output processing on a Windows desktop

[4] Microsoft (2000). 'Microsoft DirectX Overview'. *Microsoft Developer Network Library.*

[5] Microsoft (2001). 'DirectDraw Architecture, System Integration'. *Microsoft Developer Network Library.*

4.0 TRUSTED PATH FOR APPLICATIONS

A couple of applications need to communicate directly with the user. This includes situations where it is eminent to get input from the user being physically present at the machine, or to ensure that the user sees data undisturbed by other applications.

Examples are creation of electronic signatures, online voting, financial transactions, communication software etc. The concept for this is the trusted path. A trusted path exists between the user and the operating system, e.g., when invoking the logon process.

For an implementation of a trusted path for security-aware software we build on Langweg's (2002) work. In this approach, DirectX is used to determine whether input was initiated by a device or by a simulated message. Integrity and authenticity of user input is achieved that way.

4.1 Input Components

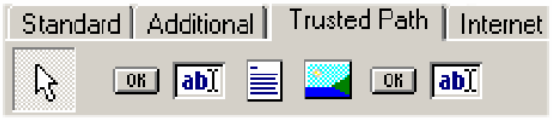


Figure 3: Components palette for input and output

We provide components for application developers. They are shown in the component palette above and comprise two kinds of buttons, two kinds of edit boxes, a memo box, and components for output that will be discussed in a later section. The components are implemented using Borland Delphi.

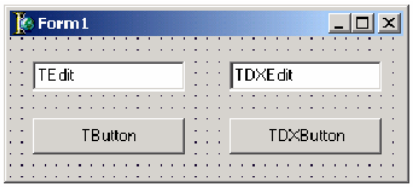


Figure 4: Form showing both standard and trusted path components

These components can be used like the standard Windows controls. In the case of input, we have three different components, an edit control, a memo box, and a button. They offer almost the same functionality as the original controls. Since they inherit from the original TEdit, TMemo and TButton classes, they can replace these components in existing projects.

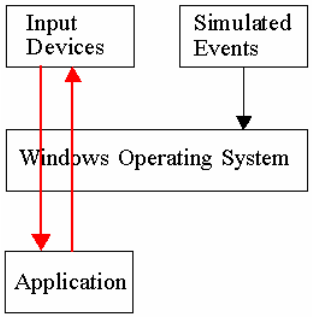


Figure 5: Input processing using DirectX

The new edit control, TDXEdit, does not rely on Windows messages being accurate. It confirms if input can be verified via the DirectInput interface, originally added for game development.

We have two different implementations. In the first, we observe the message queue for input messages. As soon as an input message arrives, we check the DirectInput keyboard state. If a key press or release can be confirmed, we process the input message; otherwise it is discarded. In situations where there is high load on the system, the user may have to type less fast than usual. This did not present a practical problem in tests with most users. However, sometimes genuine input was discarded when it should not have been.

In our second implementation, we process input in a separate thread. This thread observes the DirectInput device either by polling or by a call back function. When input is detected at the DirectInput interface, a message is composed and posted to the message queue. This message is specified as, e.g., WM_DX_KEYDOWN. Its parameters contain an index and a pointer to the input data. The parameters are encrypted using AES to defend against other processes composing a similar message. In the message processing loop, the WM_DX_KEYDOWN parameters are decrypted, and the keyboard input data retrieved from memory. Then a conventional WM_KEYDOWN message is constructed internally and processed by the default method. All other WM_KEYDOWN messages that arrive directly and not as a WM_DX_KEYDOWN are discarded because they could have been manipulated. This holds true for WM_KEYDOWN, WM_KEYUP, WM_CHAR, WM_PASTE. WM_SETTEXT and WM_GETTEXT are also processed only when they originate from inside the process.

We also have a TEdit variant that works with the API function GetAsyncKeyState instead of DirectInput. Its implementation is almost identical to the version presented above.

Our TDXMemo component works similar to the TDXEdit but offers multiple lines for user input.

The TDXButton component offers a button that can not be clicked by a simulated message. The implementation is simpler since only WM_LBUTTONDOWN, WM_LBUTTONUP, WM_LBUTTONDBLCLK, WM_MOUSEMOVE messages have to be observed. We also have two variants as described above, the first validating messages when they arrive, the second generating messages based on DirectInput and discarding all others.

AES was chosen with respect to high speed in software. The overhead of encrypting and decrypting the input messages is remarkably low. On our 850 MHz PIII time spent on message processing increases by less than two percent in the case of message validation. In the second implementation we observe that message processing takes almost twice as long as in the unencrypted case.

4.2 SendInput

The problem of simulated Windows messages solved, input can in principle still be fabricated. The operating system allows processes running in the same desktop to simulate input by the Win32 API function SendInput.

To prohibit other (possibly malicious) programs to call SendInput and synthesize the user's key strokes we can inject control code into running processes. This code resides in a dynamic link library (DLL) which is activated when USER32.DLL is loaded. Since SendInput is a USER32 function, loading of our DLL is assured. It is necessary to add our DLL to the key `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows\APPInit_DLLs` in the Windows registry.[6] This requires administrative privileges during installation.

The DLL modifies the Import Address Table of the supervised program and redirects all calls to SendInput to its own version of that function. Calls can then be blocked or forwarded to the original SendInput function when blocking is not required. The same effect could be achieved by employing a COTS sandbox software for programs running on the computer that monitors and restricts the use of certain API calls including SendInput.

It may be possible to distinguish users and untrustworthy programs by observing their input behaviour, e.g., programs simulating input much faster than an ordinary user could type. This rather falls in the field of biometrics (cf. e.g. Bergadano et al. 2002).

We have found another way to tackle the SendInput problem. We call the first method 'Fast Hook Renewal'. Our input components install a system-wide low-level keyboard hook to catch all input before it is processed by applications. Here we check the LLKHF_INJECTED flag for injected input. If it is set, we discard that input so that it does not reach our secure components. Since other (and malicious) applications can employ this method as well to promote their agenda, we renew our hook after a short period, i.e., some milliseconds, to get to the beginning of the hook chain. This method works quite reliably, albeit not 100% of the time.

Our second approach is modifying the desktop's ACL (access control list). It is possible to enable system-wide hooks for certain accounts only. Hence, ordinary applications could be denied using system-wide hooks while our protected application could use them. A desktop's ACL has to be modified before a (possibly malicious) process is started. So, the canonical point to do this is the creation of the desktop which is the responsibility of the GINA (Graphical Identification and Authentication) library. This cannot be circumvented by an application. However, it requires replacing either the standard GINA of the operating system or that of a third party vendor, e.g., with a smart card or biometric authentication. Currently, creating a chain of GINA libraries does not add to overall system stability.

Eliminating faked messages 100% of the time and discarding injected SendInput data most of the time is still an advantage. It can be done easily be changing some components and leaving the rest of a program untouched. Messages allow an attacker to exactly specify which parts of the user interface should receive input. Achieving the same with SendInput is harder to do. In lack of an appropriate metric, it is not known how much harder exactly, though. Attacks by SendInput can also be countered by introducing delays in the input process (cf. Carlisle et al. 2001). Attacks are not prevented then, but made detectable by the user (or by the application if it suspects the user typing so fast that input can only come from a malicious program). Changing the user interface slightly from time to time also makes it harder for an attacker to use SendInput as a tool.

[6] Microsoft (2000) 'Working with the APPInit_DLLs Registry Value'. *Microsoft Knowledge Base Q197571.*

4.3 Output Components

We provide components for application developers. They are shown in the component palette in fig. 3 and comprise a couple of input components and three output components, one of which is shown as a picture. The components are implemented using Borland Delphi.

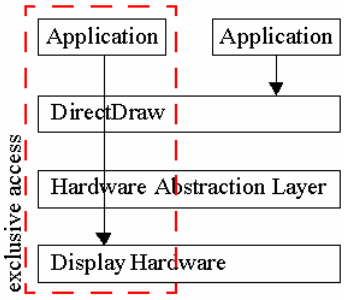


Figure 6: Output processing using DirectX

We have two goals as regards output components. On the one hand, integrity and authenticity of the output has to be ensured. On the other, we want to combine a secure display with the user being able to respond along a trusted path.

Hence, we provide three components. Two display the content of a window on a secure surface. The third offers a standardized limited functionality for user input in combination with a secure surface to draw on.

The first output component, DXFormShow, is used in conjunction with a form's show method. We activate DirectDraw, acquire the screen exclusively, clear the screen, paint it black and draw the form's content on the centre of the screen. As shown in previous work, this ensures a display that can not be manipulated by other processes. In addition, it provides confidentiality of the output. An application developer needs to drop the component on a form and replace calls to a form's show method. The rest of the application's code, including the code used to draw the form, can be left unchanged. However, this only ensures that a form is drawn on a secure surface. If the form contains sophisticated input controls these can not be used by the user. So, we recommend this component only for displaying, e.g., details of a financial transaction or data to be signed when no modification is essential.

Figure 7: Component DXEnhancedMessageBox with simple user interface and application hologram

Our second component used for output, DXMessageBox, is a replacement for the standard Win32 API function MessageBox. A developer provides a caption and text to display in the message box, and the buttons that are shown to the user. We then switch to a secure surface as described above and show the message box. The buttons employed are TDXButtons that we use for trusted input. The user then has a choice of which button to click to give the application a response to the output.

In a variant of the message box we offer an enhanced message box, DXEnhancedMessageBox. In addition to displaying text and offering buttons, the box allows one TDXEdit control to be used. The user may there enter commands securely in text mode.

4.4 Window Personalization

Authenticity of the output is done by window personalization. This concept is described by Tygar et al. (1996). At installation time the user adopts a picture that is displayed each time our application invokes the trusted display mode. Other applications do not have access to the picture and the user can thus determine whether or not to accept the output.

The NTFS file system of Windows NT/2000/XP only allows to specify access rights differentiating users. We put a service on top that identifies processes that request the secret picture. This service runs under a separate account. Access to files containing a picture identifying an application is restricted to the account of the service. Hence, confidentiality and integrity of this picture is protected. We call it an 'application hologram' (e.g., the teddy bear in the example in fig. 7).

The protocol for communicating with our service contains four steps.

- An application opens a named pipe to the service requesting access to the secret graphic. The request contains a handle of a window of the requesting process.
- The service checks which executable module belongs to the window identified by the handle. If the executable is not in the service's white list of benign processes, the protocol stops and an entry is added to the security log. The check could also involve whether the executable file has been tampered with, but this lies outside the scope of our tool. If the service decides that the window belongs to a benign process, it sends a random 32 bit value (a nonce) wrapped in a special message to the window.
- The receiving window checks whether it requested access to the secret graphic. If it decides to proceed, it sends the received value to the service via the named pipe opened in the first step.
- The service checks if the received value matches the value sent in the current session. If they match, the service opens the file containing the application hologram and sends it to the application via the named pipe.

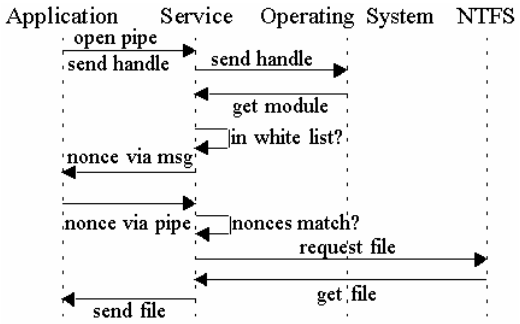


Figure 8: Communication diagram for retrieving an application hologram

At the end of the protocol run the application hologram has been transferred to the secure application that will display it on a confidentiality preserving secure surface. Hence, only trusted applications have access to the graphic, and the user can trust applications that show the graphic.

Using a named pipe ensures that the file is sent only to a benign application that opened the pipe. Sending a nonce via standard Windows messaging and associating it with the pipe session ensures that only an application present in the white list can request a hologram. Messages are put in the message queue of the respective application only.

As a precaution we use modified desktop ACLs as described earlier to disallow global hooks for other applications. The Win32 API otherwise offers to install a hook to retrieve all messages globally. Even then, an attack would have to be tailored to this specific protocol.

The same effect could also be achieved by using a file system filter driver identifying requests not only by user account but also by process.

4.5 Microsoft Windows Interface Changes

In the Microsoft Windows XP successor, code-named Longhorn, changes are anticipated as regards how applications use the desktop for displaying data. The Direct3D part of DirectX is expected to be used to render output. From a programmer's perspective, however, the new desktop composition engine will replace GDI and GDI+, but the interfaces will be similar. The desktop is still shared among applications. Hence, there may still be a need to acquire the display in exclusive mode to ensure a trusted output. We are not aware of changes to the input model.

Longhorn is currently expected to ship by 2005, so even if some problems with trusted output are solved with the new release, the need for practical solutions exists today. With the components we have presented here change can be applied to applications easily to heighten security.

NGSCB/Palladium (cf. England et al. 2003), Microsoft's attempt to add a separate secure kernel to Windows, also intends to provide a trusted path from an application to a user. However, this requires programming a new application since NGSCB works with a different application programming interface compared with today's Win32 API. In addition, NGSCB is not yet available.

5.0 RETROFITTING EXISTING APPLICATIONS

Today applications that would profit from a trusted path to the user exist. These applications can not be rewritten completely.

Edit and button controls in security sensitive dialog windows can be replaced by the DirectX-enhanced controls that we have presented in the preceding section. This replacement can take place at design/build time when a maintenance update or a new version of the application is produced. Since the controls are compatible with the standard controls, there is no need to change the source code depending on these input controls.

Replacing output requires some change to an application's source code. It may not be desired to switch to full-screen exclusive mode every time a message box is shown. Hence, only the security sensitive parts of the application have to be touched where displaying information to the user is important to be trusted.

If every message box can be replaced by our new component, i.e., if there are few, then another approach can be taken even at run time. A dynamic link library (DLL) could be injected into the process' address space. This DLL alters the address table for imported functions. Since our DXMessageBox function has the same signature as the Win32 API MessageBox function, all it takes is a change of the pointer to the function. However, changing function pointers from the outside without knowing the source code of the application may become a stability problem. We therefore favour minor changes in source code during build time.

We use version 8 of the DirectInput interface and version 7 of the DirectDraw interface. We have not investigated if it would be possible to revert to earlier versions of DirectX, namely versions 3 or 5. Hence we do not know whether our components would work with the earlier Windows NT 4 operating systems.

6.0 DISCUSSION OF THE SECURITY OF THE APPROACH

The security of our implementation of a trusted path for an application relies on the integrity of the operating system and resistance against system-wide global hooks.

DirectX is a part of the Windows 2000/XP operating system. As such, the operating system is responsible for the integrity of the DirectX modules. In addition, existing integrity protection tools could be used.

One of our implementation variants sends messages using the Windows messaging system. These messages are encrypted using AES, prohibiting other processes from inserting fabricated messages undetected in the stream.

To preclude processes from simulating input via the SendInput API, we offer multiple ways. One approach is to block use of the SendInput function by adding code to another process' memory and modifying the import address table. Simulated input can also be detected and dismissed by employing low level hooks that we use in our fast hook renewal method. Other processes have to be cut off from installing system-wide hooks. This is either done by fast hook renewal or by modifying the access control list of the desktop.

A hologram service is used to authenticate and authorize access of applications to a secret picture, called a hologram. Our protocol for communication with the service authenticates the application by a handle and a message. Again, system-wide hooks have to be denied other processes. An implementation variant could obviate a separate service, placing the functionality in a file system filter driver.

As stated earlier, our focus lies on protection against architectural vulnerabilities. If there are flaws owing to errors in the implementation of the platform, they have to be covered by other means.

Variants in the implementation, which we partly offer, help to increase resistance against generic attack tools. We offer three variants of our input components' implementation, and three variants as regards protection against simulated input, two of which can also be used for the hologram service.

7.0 CONCLUSION

Trojan horse programs, i.e., programs with additional hidden, often malicious, functions, are more and more popular forms of attack. Applications that execute in an insecure environment should have control over their communication with the user.

Our work solves integrity and authenticity of input, confidentiality, integrity and authenticity of output. Confidentiality of user input is not discussed and remains to be scrutinized.

We have in detail explored components for establishing a trusted path for an application in an event-driven system. These components can be adopted by developers to reinforce existing applications or to build new security sensitive applications.

We have presented a creative way of exploiting existing COTS components – DirectX – to directly access input and output devices. Compared with dedicated hardware or operating system replacements our solution gives tractable and cost-effective means to incorporate better protection into programs on desktop computers.

Different implementations of the components provide different levels of security and different points of attack, increasing resistance against generic attack tools. Current malicious software that threatens integrity of desktop user interaction often relies on weaknesses in the standard messaging system. These weaknesses are abated by our approach.

Further research should include metrics to measure the added security by employing COTS components and varying implementations. Probably additional standard user interface components like combo boxes or tool bars could be implemented using our approach to offer application developers more choices. It remains to be examined whether components could be built that also work with older versions of the Windows operating system family, i.e., NT4. It might also be of interest to explore if other application programming interfaces, e.g., the Qt framework or OpenGL, could be used in the same way as DirectX.

8.0 REFERENCES

[1] Balfanz, D. (2001). *Access Control for Ad-hoc Collaboration.* PhD thesis, Princeton University.

[2] Bergadano, F., Gunetti, D., and Picardi, C. (2002). 'User Authentication through Keystroke Dynamics'. ACM Transactions on Information and System Security 5.4(2002):367-397.

[3] Bråthen, R. (1998). 'Crash Course in X Windows Security'. *GridLock* 1(1998):1. http://www.hackphreak.org/gridlock/issues/issue.1/xwin.html

[4] Carlisle, M.C. and Studer, S.D. (2001). 'Reinforcing Dialog-Based Security'. *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security.* Pp. 24-29.

[5] CERT Coordination Center (1999). *CERT Advisory CA-99-02-Trojan-Horses*. http://www.cert.org/advisories/CA-1999-02.html

[6] Clark, D.D. and Wilson, D.R. (1987). 'A Comparison of Commercial and Military Computer Security Policies'. *Proceedings of 1987 IEEE Symposium on Security and Privacy.* Pp. 184-194.

[7] Cult of the Dead Cow (2003). *Back Orifice 2000*. http://bo2k.sourceforge.net

[8] Delphi-Jedi Project (2003). *DirectX headers and samples.* http://www.delphi-jedi.org

[9] Department of Defense (1985). *DoD 5200.28-STD Department of Defense Trusted Computer System Evaluation Criteria.* ('Orange Book')

[10] England, P., Lampson, B., Manferdelli, J., Peinado, M., and Willman, B. (2003). 'A Trusted Open Platform'. *Computer* 36.7(2003):55-62.

[11] Howard, M. (2002). *Tackling Two Obscure Security Issues.* http://msdn.microsoft.com/library/default.asp?url=/library/ en-us/dncode/html/secure08192002.asp

[12] Langweg, H. (2002). 'With Gaming Technology towards Secure User Interfaces'. *Proceedings of Annual Computer Security Applications Conference 2002.* Pp. 44-50.

[13] Microsoft (1998). *Microsoft Windows Architecture for Developers Training Kit.*

[14] Microsoft (2003). *Microsoft Developer Network Library.*

[15] Paget, C. (2002). 'Exploiting design flaws in the Win32 API for privilege escalation. Or... Shatter Attacks – How to break Windows'. http://www.google.com/search?q=cache:security.tombom.co.uk/shatter.html

[16] Rooker, T. (1993). 'Application Level Security Using an Object-Oriented Graphical User Interface'. *Proceedings of the 1992-1993 Workshop on New Security Paradigms.* Pp. 105-108.

[17] Schmid, M., Hill, F., and Ghosh, A.K. (2002). 'Protecting Data from Malicious Software'. *Proceedings of Annual Computer Security Applications Conference 2002.* Pp. 199-208.

[18] Spalka, A., Cremers, A.B., and Langweg, H. (2001). 'The Fairy Tale of »What You See Is What You Sign«. Trojan Horse Attacks on Software for Digital Signatures'. *Proceedings of IFIP Working Conference on Security and Control of IT in Society-II.* Pp. 75-86.

[19] Spalka, A., and Langweg, H. (2002). 'Notes on Program-Orientated Access Control'. *Proceedings of First International Workshop on Trust and Privacy in Digital Business – TrustBus.* Pp. 451-455.

[20] Thurrott, P. (2003). 'The Road To Windows 'Longhorn' Part Two'. Paul Thurrott's SuperSite for Windows. http://www.winsupersite.com/showcase/ longhorn_preview_2003.asp

[21] Tygar, J.D., and Whitten, A. (1996). 'WWW Electronic Commerce and Java Trojan Horses'. *Proceedings of the Second USENIX Workshop on Electronic Commerce.*

[22] Wiseman, S., Terry, P., Wood, A., and Harrold, C. (1988). 'The Trusted Path between SMITE and the User'. *Proceedings of 1988 IEEE Symposium on Security and Privacy.* Pp. 147-155.

[23] Xenitellis, S. (2002a). 'Security vulnerabilities in event-driven systems'. *Proceedings of IFIP SEC'2002.* Pp. 147-160

[24] Xenitellis, S. (2002b). 'A New Avenue of Attack: Event-driven System Vulnerabilities'. *Proceedings of European Conference on Information Warfare and Security, MCIL.* Pp. 177-185.

[25] Ye, E. and Smith, S. (2002). 'Trusted Paths for Browsers'. *Proceedings of 11th USENIX Security Symposium.* Pp. 263-279.

REPORT DOCUMENTATION PAGE

1. Recipient's Reference	2. Originator's References	3. Further Reference	4. Security Classification of Document
	RTO-MP-IST-041 AC/323(IST-041)TP/27	ISBN 92-837-0039-2	UNCLASSIFIED/ UNLIMITED

5. Originator

Research and Technology Organisation
North Atlantic Treaty Organisation
BP 25, F-92201 Neuilly-sur-Seine Cedex, France

6. Title

Adaptive Defence in Unclassified Networks

7. Presented at/Sponsored by

The RTO Information Systems Technology Panel (IST) Symposium held in Toulouse, France, 19-20 April 2004.

8. Author(s)/Editor(s)	9. Date
Multiple	November 2004
10. Author's/Editor's Address	**11. Pages**
Multiple	288 (text) 497 (slides)

12. Distribution Statement

There are no restrictions on the distribution of this document. Information about the availability of this and other RTO unclassified publications is given on the back cover.

13. Keywords/Descriptors

Communications networks
Computer information security
Computer networks
Computer security
Denial of service attacks
Design
Electronic security
IDS (Intrusion Detection Systems)
Information assurance
Information security
Information systems
Integrated systems
International cooperation
Intrusion detectors
Secure communication
Surveillance
Systems engineering
Threat evaluation
Vulnerability

14. Abstract

This volume contains 21 papers, presented at the Symposium of the Information Systems Technology Panel (IST) held in Toulouse, France, from 19th to 20th April 2004.

The papers were presented in seven sessions covering the following headings: Coalition Networks; Intrusion Detection and Response; Honeypots; Servers and Viruses, Network Technology; Securing Networks; Dealing with COTS.

NORTH ATLANTIC TREATY ORGANISATION

RESEARCH AND TECHNOLOGY ORGANISATION

BP 25
F-92201 NEUILLY-SUR-SEINE CEDEX • FRANCE
Télécopie 0(1)55.61.22.99 • E-mail mailbox@rta.nato.int

DIFFUSION DES PUBLICATIONS

RTO NON CLASSIFIEES

Les publications de l'AGARD et de la RTO peuvent parfois être obtenues auprès des centres nationaux de distribution indiqués ci-dessous. Si vous souhaitez recevoir toutes les publications de la RTO, ou simplement celles qui concernent certains Panels, vous pouvez demander d'être inclus soit à titre personnel, soit au nom de votre organisation, sur la liste d'envoi.

Les publications de la RTO et de l'AGARD sont également en vente auprès des agences de vente indiquées ci-dessous.

Les demandes de documents RTO ou AGARD doivent comporter la dénomination « RTO » ou « AGARD » selon le cas, suivi du numéro de série. Des informations analogues, telles que le titre est la date de publication sont souhaitables.

Si vous souhaitez recevoir une notification électronique de la disponibilité des rapports de la RTO au fur et à mesure de leur publication, vous pouvez consulter notre site Web (www.rta.nato.int) et vous abonner à ce service.

CENTRES DE DIFFUSION NATIONAUX

ALLEMAGNE
Streitkräfteamt / Abteilung III
Fachinformationszentrum der
Bundeswehr (FIZBw)
Friedrich-Ebert-Allee 34, D-53113 Bonn

BELGIQUE
Etat-Major de la Défense
Département d'Etat-Major Stratégie
ACOS-STRAT – Coord. RTO
Quartier Reine Elisabeth
Rue d'Evère, B-1140 Bruxelles

CANADA
DSIGRD2
Bibliothécaire des ressources du savoir
R et D pour la défense Canada
Ministère de la Défense nationale
305, rue Rideau, 9e étage
Ottawa, Ontario K1A 0K2

DANEMARK
Danish Defence Research Establishment
Ryvangs Allé 1, P.O. Box 2715
DK-2100 Copenhagen Ø

ESPAGNE
SDG TECEN / DGAM
C/ Arturo Soria 289
Madrid 28033

ETATS-UNIS
NASA Center for AeroSpace
Information (CASI)
Parkway Center, 7121 Standard Drive
Hanover, MD 21076-1320

FRANCE
O.N.E.R.A. (ISP)
29, Avenue de la Division Leclerc
BP 72, 92322 Châtillon Cedex

GRECE (Correspondant)
Defence Industry & Research
General Directorate, Research Directorate
Fakinos Base Camp, S.T.G. 1020
Holargos, Athens

HONGRIE
Department for Scientific Analysis
Institute of Military Technology
Ministry of Defence
H-1525 Budapest P O Box 26

ISLANDE
Director of Aviation
c/o Flugrad
Reykjavik

ITALIE
Centro di Documentazione
Tecnico-Scientifica della Difesa
Via XX Settembre 123
00187 Roma

LUXEMBOURG
Voir Belgique

NORVEGE
Norwegian Defence Research Establishment
Attn: Biblioteket
P.O. Box 25, NO-2007 Kjeller

PAYS-BAS
Royal Netherlands Military
Academy Library
P.O. Box 90.002
4800 PA Breda

POLOGNE
Armament Policy Department
218 Niepodleglosci Av.
00-911 Warsaw

PORTUGAL
Estado Maior da Força Aérea
SDFA – Centro de Documentação
Alfragide
P-2720 Amadora

REPUBLIQUE TCHEQUE
LOM PRAHA s. p.
o. z. VTÚLaPVO
Mladoboleslavská 944
PO Box 18
197 21 Praha 9

ROYAUME-UNI
Dstl Knowledge Services
Information Centre, Building 247
Dstl Porton Down
Salisbury
Wiltshire SP4 0JQ

TURQUIE
Milli Savunma Bakanlığı (MSB)
ARGE ve Teknoloji Dairesi Başkanlığı
06650 Bakanliklar – Ankara

AGENCES DE VENTE

NASA Center for AeroSpace Information (CASI)
Parkway Center, 7121 Standard Drive
Hanover, MD 21076-1320
ETATS-UNIS

The British Library Document Supply Centre
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
ROYAUME-UNI

Canada Institute for Scientific and Technical Information (CISTI)
National Research Council
Acquisitions, Montreal Road, Building M-55
Ottawa K1A 0S2, CANADA

Les demandes de documents RTO ou AGARD doivent comporter la dénomination « RTO » ou « AGARD » selon le cas, suivie du numéro de série (par exemple AGARD-AG-315). Des informations analogues, telles que le titre et la date de publication sont souhaitables. Des références bibliographiques complètes ainsi que des résumés des publications RTO et AGARD figurent dans les journaux suivants :

Scientific and Technical Aerospace Reports (STAR)
STAR peut être consulté en ligne au localisateur de ressources uniformes (URL) suivant:
http://www.sti.nasa.gov/Pubs/star/Star.html
STAR est édité par CASI dans le cadre du programme NASA d'information scientifique et technique (STI)
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
ETATS-UNIS

Government Reports Announcements & Index (GRA&I)
publié par le National Technical Information Service
Springfield
Virginia 2216
ETATS-UNIS
(accessible également en mode interactif dans la base de données bibliographiques en ligne du NTIS, et sur CD-ROM)

NORTH ATLANTIC TREATY ORGANISATION

RESEARCH AND TECHNOLOGY ORGANISATION

BP 25
F-92201 NEUILLY-SUR-SEINE CEDEX • FRANCE
Télécopie 0(1)55.61.22.99 • E-mail mailbox@rta.nato.int

DISTRIBUTION OF UNCLASSIFIED RTO PUBLICATIONS

AGARD & RTO publications are sometimes available from the National Distribution Centres listed below. If you wish to receive all RTO reports, or just those relating to one or more specific RTO Panels, they may be willing to include you (or your Organisation) in their distribution.

RTO and AGARD reports may also be purchased from the Sales Agencies listed below.

Requests for RTO or AGARD documents should include the word 'RTO' or 'AGARD', as appropriate, followed by the serial number. Collateral information such as title and publication date is desirable.

If you wish to receive electronic notification of RTO reports as they are published, please visit our website (www.rta.nato.int) from where you can register for this service.

NATIONAL DISTRIBUTION CENTRES

BELGIUM
Etat-Major de la Défense
Département d'Etat-Major Stratégie
ACOS-STRAT – Coord. RTO
Quartier Reine Elisabeth
Rue d'Evère
B-1140 Bruxelles

CANADA
DRDKIM2
Knowledge Resources Librarian
Defence R&D Canada
Department of National Defence
305 Rideau Street
9th Floor
Ottawa, Ontario K1A 0K2

CZECH REPUBLIC
LOM PRAHA s. p.
o. z. VTÚLaPVO
Mladoboleslavská 944
PO Box 18
197 21 Praha 9

DENMARK
Danish Defence Research Establishment
Ryvangs Allé 1
P.O. Box 2715
DK-2100 Copenhagen Ø

FRANCE
O.N.E.R.A. (ISP)
29, Avenue de la Division Leclerc
BP 72
92322 Châtillon Cedex

GERMANY
Streitkräfteamt / Abteilung III
Fachinformationszentrum der Bundeswehr (FIZBw)
Friedrich-Ebert-Allee 34
D-53113 Bonn

GREECE (Point of Contact)
Defence Industry & Research General Directorate, Research Directorate
Fakinos Base Camp, S.T.G. 1020
Holargos, Athens

HUNGARY
Department for Scientific Analysis
Institute of Military Technology
Ministry of Defence
H-1525 Budapest P O Box 26

ICELAND
Director of Aviation
c/o Flugrad, Reykjavik

ITALY
Centro di Documentazione Tecnico-Scientifica della Difesa
Via XX Settembre 123
00187 Roma

LUXEMBOURG
See Belgium

NETHERLANDS
Royal Netherlands Military Academy Library
P.O. Box 90.002
4800 PA Breda

NORWAY
Norwegian Defence Research Establishment
Attn: Biblioteket
P.O. Box 25, NO-2007 Kjeller

POLAND
Armament Policy Department
218 Niepodleglosci Av.
00-911 Warsaw

PORTUGAL
Estado Maior da Força Aérea
SDFA – Centro de Documentação
Alfragide, P-2720 Amadora

SPAIN
SDG TECEN / DGAM
C/ Arturo Soria 289
Madrid 28033

TURKEY
Milli Savunma Bakanlığı (MSB)
ARGE ve Teknoloji Dairesi Başkanlığı
06650 Bakanliklar – Ankara

UNITED KINGDOM
Dstl Knowledge Services
Information Centre, Building 247
Dstl Porton Down
Salisbury, Wiltshire SP4 0JQ

UNITED STATES
NASA Center for AeroSpace Information (CASI)
Parkway Center, 7121 Standard Drive
Hanover, MD 21076-1320

SALES AGENCIES

NASA Center for AeroSpace Information (CASI)
Parkway Center
7121 Standard Drive
Hanover, MD 21076-1320
UNITED STATES

The British Library Document Supply Centre
Boston Spa, Wetherby
West Yorkshire LS23 7BQ
UNITED KINGDOM

Canada Institute for Scientific and Technical Information (CISTI)
National Research Council
Acquisitions
Montreal Road, Building M-55
Ottawa K1A 0S2, CANADA

Requests for RTO or AGARD documents should include the word 'RTO' or 'AGARD', as appropriate, followed by the serial number (for example AGARD-AG-315). Collateral information such as title and publication date is desirable. Full bibliographical references and abstracts of RTO and AGARD publications are given in the following journals:

Scientific and Technical Aerospace Reports (STAR)
STAR is available on-line at the following uniform resource locator:
http://www.sti.nasa.gov/Pubs/star/Star.html
STAR is published by CASI for the NASA Scientific and Technical Information (STI) Program
STI Program Office, MS 157A
NASA Langley Research Center
Hampton, Virginia 23681-0001
UNITED STATES

Government Reports Announcements & Index (GRA&I)
published by the National Technical Information Service
Springfield
Virginia 2216
UNITED STATES
(also available online in the NTIS Bibliographic Database or on CD-ROM)

ISBN 92-837-0039-2