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.

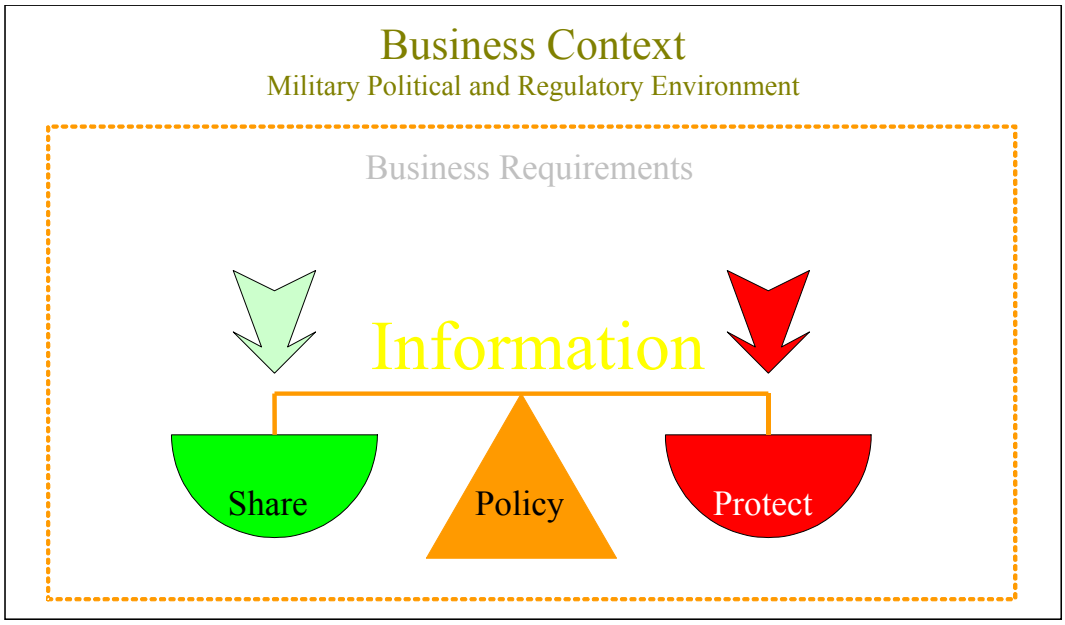

Figure 1: Share versus protect

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.

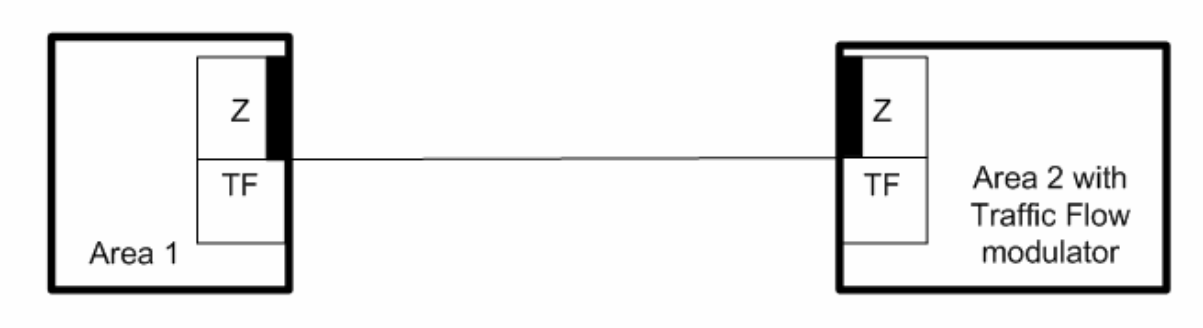

Figure 3: … which is actually insecure

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.