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.