Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

An Overview of Passive Information Gathering Techniques for Network Security

J. Treurniet

Defence R&D Canada – Ottawa
TECHNICAL MEMORANDUM
DRDC Ottawa TM 2004-073
May 2004

Canada

An Overview of Passive Information Gathering Techniques for Network Security

J. Treurniet

Defence R&D Canada – Ottawa

Technical Memorandum

DRDC Ottawa TM 2004-073

May 2004

Abstract

Network awareness is a crucial aspect of network security, and is usually achieved through the use of active scanning. This method periodically introduces a large amount of traffic on the network, using up bandwidth. Much of the information obtained through active methods may also be obtained by passively listening to traffic. This document explains how passive scanning methods can aid in achieving network awareness without introducing unnecessary traffic, and documents a proof-of-concept tool to show how the methods presented might be implemented.

Résumé

La surveillance, qui est un aspect très important de la sécurité de réseau, se fait habituellement au moyen d'une analyse active. Toutefois, cette méthode augmente périodiquement le volume de trafic sur le réseau, utilisant ainsi toute la bande passante. Vous pouvez surveiller le trafic au moyen de méthodes passives pour obtenir la plupart des renseignements obtenus par l'entremise de méthodes actives. Le présent document explique la façon dont les méthodes d'analyse passive peuvent vous aider à surveiller le réseau sans augmenter inutilement le trafic. Il donne également de l'information sur un outil de validation de principes expliquant comment mettre en place les méthodes présentées.

This page intentionally left blank.

Acknowledgements

The author would like to acknowledge Annie De Montigny-Leboeuf of the Communications Research Centre for her contribution of comprehensive sets of rules for determining the operating system of a host based on SYN and SYN-ACK packets. Thanks also to Vincent Taylor of DRDC Ottawa for suggesting the use of distributed sensors.

Executive summary

Background

Effective network security requires situational awareness of the composition of and the activities taking place on the network. The network manager needs to have a real-time picture to verify that the network policy is not being violated by any user or misconfiguration.

To obtain real-time awareness, one may choose to periodically scan the network. Such "active" techniques have two disadvantages: First, they introduce traffic onto the network. The bandwidth consumed by active scanning can be considerable for larger networks. Second, there may be instances where the scanning misses an activity, for example, when a specific port must be probed with a specific protocol.

These drawbacks can be addressed by using "passive" techniques. No traffic is introduced on the network when using these techniques. A sniffer is strategically placed on the network and the traffic is examined as it passes by. The behaviour of the traffic can be compared to an established policy for deviations. The technique can also be used to identify what information about the network may be used by attackers because it is being leaked to the Internet.

Principal results

A proof-of-concept tool was created to analyse TCP, UDP and ICMP traffic. The tool was built using the Network Traffic Analysis Toolbox, a suite of functions created at DRDC to load and analyse network traffic. Traffic collected in 1999 for use in DARPA's Intrusion Detection Evaluation was analysed by the tool and the results were examined. Good agreement was found between the test program results and the documented network attributes.

Significance of results and future Work

The success of the tool shows that passive discovery is useful in securing a network. Future work includes examination of protocols other than TCP, UDP and ICMP, including routing protocols and non-routable protocols. Real-time active discovery methods can be combined with passive methods to provide a complete network picture with minimal disturbance.

J. Treurniet; 2004; An Overview of Passive Information Gathering Techniques for Network Security; DRDC Ottawa TM 2004-073; Defence R&D Canada – Ottawa.

Sommaire

Renseignements généraux

Pour assurer la sécurité du réseau de façon efficace, le gestionnaire de réseau doit porter une attention particulière à la nature des activités et aux circonstances dans lesquelles elles y sont pratiquées. Il doit, à cet effet, avoir une vue d'ensemble du réseau, en temps réel, pour s'assurer que tous les utilisateurs respectent la politique du réseau et n'en changent pas la configuration.

Pour surveiller le réseau en temps réel, vous pouvez l'analyser périodiquement. Une telle technique, dite « active », présente toutefois deux inconvénients. Elle accroît, d'une part, le volume de trafic sur le réseau ; la bande passante peut, notamment, être grandement sollicitée lors de l'analyse active de grands réseaux. D'autre part, il peut arriver qu'une activité ne soit pas analysée, par exemple, lorsqu'un port doit être testé au moyen d'un protocole spécifique.

Vous pouvez éviter ces inconvénients en utilisant des techniques dites « passives », qui n'augmentent pas le trafic sur le réseau. Un programme renifleur est alors placé à un endroit stratégique sur le réseau pour en examiner le volume de trafic. Vous pouvez ainsi repérer les activités qui ne respectent pas la politique de réseau établie. Cette technique vous permet aussi d'identifier les renseignements sur le réseau qui ont été diffusés sur Internet, en raison de fuites, et qui peuvent être utilisés par des pirates informatiques.

Résultats principaux

Un outil de validation de principes, servant à analyser le trafic TCP, UDP et ICMP, a été conçu au moyen de la suite d'outils pour l'analyse de trafic de réseau, qui a été créée à RDDC pour charger et analyser le volume de trafic sur le réseau. Les données sur le trafic de 1999, qui ont été traitées par le programme Intrusion Detection Evaluation de DARPA, ont été analysées par l'outil, et les résultats ont été examinés. Les résultats obtenus par le programme d'essai reflètent bien les attributs de réseau documentés.

Importance des résultats et travaux futurs

Le rendement positif de cet outil démontre que les méthodes de découverte passives sont utiles pour sécuriser un réseau. Les tâches à venir consisteront, notamment, à examiner des protocoles autres que TCP, UDP et ICMP, y compris des protocoles de routage et des protocoles non routables. Vous pouvez combiner des méthodes de découverte actives, en temps réel, avec des méthodes passives pour obtenir une vue d'ensemble complète du réseau, sans trop perturber le trafic.

J. Treurniet; 2004; An Overview of Passive Information Gathering Techniques for Network Security; DRDC Ottawa TM 2004-073; R & D pour la défense Canada – Ottawa.

This page intentionally left blank.

Table of contents

List of figures

List of tables

This page intentionally left blank.

1 Introduction

To ensure the security of a network effectively, the network manager must be continuously informed of the status and composition of the network, as well as the activities that are taking place on it. Real-time awareness is the ultimate goal. Traditionally, "active" discovery methods are used periodically to attain some degree of near-real-time awareness.

Active discovery methods are methods that introduce traffic onto the network, such as ICMP ping, SNMP queries, and TCP SYN portscans. There are two drawbacks to using these techniques. The first, common to any active sensing technique, is counter-detection. The second drawback is the potential introduction of large amounts of traffic onto the network, which can, at peak times, be a burden on the network. For example, a simple ping sweep on a class B network can introduce up to 3.6 megabytes of traffic per sweep. If one were to also scan for all 65535 ports using TCP SYN packets, this number skyrockets to over 170 gigabytes per sweep.

To address these drawbacks, "passive" techniques can be integrated into network discovery algorithms. Passive techniques strictly listen to traffic and do not introduce traffic onto the network. A strategically placed network sniffer can collect the passing traffic for further processing to yield the desired information.

An additional advantage of passive techniques are "windfalls", when unusual network activity is found unexpectedly. In comparison, active methods do not give the complete picture of the activities on the network because they must look for a particular activity in order to find it.

Active methods used to discover one's own network are also known to be employed by attackers in their attempts to obtain information about the network. It stands to reason, then, that more experienced attackers would also employ passive methods to obtain network information. It is important to include threats due to information leakage when assessing the security of the network. Passive techniques have been in use in both defensive and offensive approaches for years [1, 2], but have only appeared recently in commercial products [3, 4].

1.1 Information Requirements for Network Situational Awareness

The information required to attain network situational awareness is being studied at DRDC, see for example [5]. For the purposes of this report, a preliminary list of information requirements will be used. The information required can be roughly separated into two categories: configuration and status. The former includes information about the devices on the networks, such as IP address, operating system, applications installed/in use, usernames and passwords. A subset of this category is related to network policy. The latter category includes such characteristics as link utilization and system utilization, and device availability. All of this information contributes to an assessment of the overall security of the network. Configuration information allows one to evaluate policy adherence and unauthorized use.

Status information may indicate the presence of worms or activity that is not authorized as defined by the policy, such as Internet radio.

In addition to the status and configuration categories, any other information that could be leaked for use by an attacker is also of interest. This includes internal e-mail addresses, external user names and external passwords (accounts on servers not on the network). Leakage of information of this type could aid a clever attacker. As well, to make an effective network map without having prior knowledge of the network, it is useful to determine the most likely "depth" (number of router hops from the point of sniffing) of each host. We include depth as "device logical location" to show the picture that an attacker can obtain.

The above properties are listed in Table 1 with a summary of how the information might be obtained passively, if possible. A $\surd$ indicates that the property can usually be found passively, $\sim$ indicates that it can sometimes be found, and $\times$ indicates that it cannot be found passively. Section 2 will explain the methods in detail.

Table 1: *Properties of a network, and whether they can be discovered passively. Note that this table assumes that the device is active and that a sniffer is placed such that the necessary packets can be read.*

Property	Passive Discovery Method	
Device IP address	$\surd$	Existence
Device MAC address	$\surd$	ARP, DHCP, if sniffer inside
Device Hostname	$\sim$ $\surd$	DNS, application headers, if sniffer outside NetBIOS, ARP, DHCP, if sniffer inside
Device OS and version	$\surd$	Fingerprinting, application headers
Device OS patch level	$\times$	
Applications running	$\surd$	Application headers
Usernames	$\sim$	Application content
Passwords	$\sim$	Application content
Device type (*e.g.* client, server,...)	$\surd$	Port and protocol usage, ICMP
Services running	$\surd$	Port and protocol usage
Device operational status	$\sim$	Activity levels and ICMP
Device system utilization (CPU, memory, disk)	$\times$	
Application status	$\sim$	Outgoing activity level of device on application port
Link operational status	$\sim$	Activity levels and ICMP
Link utilization	$\sim$	Activity levels
Device physical location	$\times$	
Device logical location	$\sim$	Hop depth based on TTL

1.2 Sniffer Placement

The location of network sniffers is determined by the type of network under observation. Most large networks are composed of subnets with routers or switches at their boundaries. Traffic that does not or cannot pass through these gateways may be of interest for real-time network awareness.

One potential configuration is to have a network sniffer placed within each subnet, each processing the data and reporting back to a central location that coordinates the data. Additionally, a sniffer could be placed outside of the network entry point to observe what information can be obtained by an attacker on the Internet.

At this time, a test program has been implemented to obtain as much information as possible from traces collected on a simulated network. The sniffer was positioned on a segment just inside the border gateway. Since only TCP, UDP and ICMP protocols have been implemented in this proof-of-concept tool, it is limited in the traits it can identify. It does not identify those traits that can only be found through non-routable protocols such as *e.g.* hardware address, host name, shared directories, account names.

2 Passive Methodologies

Table 1 shows a list of network/device attributes that can be obtained passively. In the sections that follow, we describe the methods in more detail and how they are implemented in the test program.

A packet is defined as incoming or outgoing relative to the address space of the network under surveillance. “Incoming” means it is coming into the network under surveillance and “outgoing” means it is going out of the network under surveillance.

2.1 Existence

The very fact that a packet exists means an IP address can be obtained. When sniffing on an external segment, every outgoing packet contains an IP address. One difficulty arises on networks that use Network Address Translation (NAT), because the IP address is reported as the same for all hosts behind the proxy.

2.2 Domain Name System

The Domain Name System (DNS) provides name resolution, *i.e.* translates Internet names to IP addresses. Some hosts (mainly servers) are listed as entries in a DNS server accessible from the Internet. IP addresses and hostnames can be extracted from the packets exchanged when a DNS request is made. The DNS message format is shown in Figure 1. The answer resource records can give the IP address for the queried hostname, and/or aliases by which the host is known. Conversely, a response can give a hostname for a given IP.

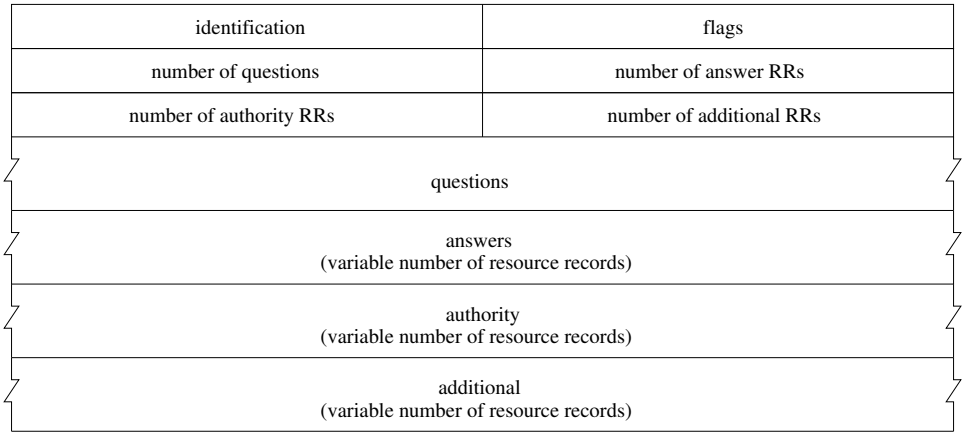

Figure 1: *A DNS message header, recreated from [6].*

In the test program, DNS replies of type 1(A), 5(CNAME) and 12(PTR) are examined to relate IP addresses to hostnames; only the *answer* resource records are examined.

The UDP protocol is generally used for DNS queries and responses, but TCP can be used for long packets. TCP is also used for DNS zone transfers. A zone transfer is a periodic update of the resource records for hosts within a zone between the primary and secondary DNS servers for a zone. It may occur as a transfer of the complete master file, or as an "incremental zone transfer", where only elements that have changed since the last poll of the primary server are transferred. If a complete zone transfer can be sniffed, the listener obtains all resource records for every host in the zone, giving hostname/IP resolution for all hosts. DNS zone transfer analysis is not implemented in the test program.

2.3 Application Payload

Application headers contain a lot of information. The technique of identifying application/device traits through reading application headers is known as "banner grabbing". Application names can be detected by parsing for strings in the content of the connection. Most security-conscious administrators will, however, modify the default string to remove such information.

Usernames can often be found in the content of packets using telnet, rlogin, POP, FTP, or other network-based applications. These can be found through sniffing on segments and when sniffing externally. Passwords are sent in the clear in some networked applications, such as telnet, FTP, and POP mail. It is also useful to record usernames and passwords used on external hosts since people often re-use passwords.

The following sections describe how application payloads were examined in the test program, and what information they gave.

2.3.1 File Transfer Protocol

A File Transfer Protocol (FTP) greeting begins with the control code “220”. The default for an FTP greeting generally follows the format:

```
220 $hostname $server ready
```

The following are examples of FTP server greetings:

```
220 $hostname FTP server (Version wu-2.6.1(1) $date_of_compile)
220 $hostname Microsoft FTP Service (Version 4.0)
220 glitch002's FTP Svr by G6 FTP Server
220 ProFTPD 1.2.0 Server
220 Server WAR-FTPD1.6.5
220 $hostname NcFTPd Server
220 Serv-U FTP-Server
```

In the test program, we searched for the above patterns of known FTP servers within any string beginning with 220. In these strings, we isolated the server type and version.

For internal usernames and passwords, we looked for the strings “USER” and “PASS” in packets destined to an internal server, and stored the string that followed each. For external usernames and passwords, we searched for the same strings in packets destined for an external server.

2.3.2 HyperText Transfer Protocol

In an HTTP session, the format of the greeting is as follows:

```
HTTP/1.0 200 OK <CRLF> Date: $date <CRLF> Server: $application
```

The same applies to HTTP/1.1. The `$application` variable contains the type and version of the server, *e.g.* `Microsoft-IIS/2.0` or `Apache/1.2.6`. In the test program, we searched for the string “server:” in any string beginning with 200 to get the application string.

2.3.3 Post Office Protocol

A Post Office Protocol (POP) session puts the greeting in the first positive reply (positive replies start with the string “+OK”), for example,

```
+OK $host $application server ready
```

where the `$host` variable contains the server hostname and the `$application` variable contains the type and version of the server. There can be a significant amount of variability in the format of the greeting; hence in the test program, we searched for the strings "server" or "service" in lines beginning with "+OK" and stored the entire line.

For internal usernames and passwords, we looked for the strings "USER" and "PASS" in packets destined to an internal server. For external usernames and passwords, we searched for the same strings in packets destined for an external server.

2.3.4 Network News Transfer Protocol

The application name in a Network News Transfer Protocol (NNTP) response follows "200". In the data collected internally, the format of a response from a news server was typically

```
200 $hostname $application $date ready
```

To account for any variability in the format of the greeting, the entire greeting was collected by the test program.

2.3.5 Simple Mail Transfer Protocol

A Simple Mail Transfer Protocol (SMTP) server will begin it's greeting with "220" if the response is contained in one packet. If the greeting runs over into the next packet, the line will begin with "220-". Examples are:

```
220 ESMTP Server (Microsoft Exchange Internet Mail Service 5.
220 $host ESMTP $sendmail_version; $date
220 $host ESMTP mailer ready at $date
```

In the test program, if the greeting contained the string "sendmail" or "Microsoft", we parsed it as shown in the examples. If the greeting contained the string "mailer ready at", we stored nothing because it contained no mailer-related information. Otherwise, we stored the entire greeting. Determination of the hostname from `$host` in the greeting was not implemented in the test program. However, we implemented a variation on this host discovery technique by taking advantage of the fact that the hostname of the SMTP client is sometimes given with the client's greeting, following "HELO" or "EHLO" in the packet payload:

```
$internal_host_IP:31113 -> $external_host_IP:25
Content: EHLO $internal_hostname
```

For both internal clients and internal servers, internal e-mail addresses can be identified:

- A client sends a packet with payload containing the string “MAIL FROM”. The string following contains the internal e-mail address of the sender enclosed by “<” and “>”.
- Incoming mail destined to an internal user has the string “RCPT TO” in the incoming payload. The e-mail address to which it is destined follows this string, enclosed by “<” and “>”.

The test program implemented this technique to discover e-mail addresses.

2.3.6 Telnet

The operating system can be identified in telnet sessions through the message displayed in the banner. In the test program, this technique was implemented by concatenating the payloads of all packets leaving the server. Using a set of unique OS “families” derived from the fingerprinting rules (see Section 2.4), we searched for the matching entry and identified the location in the string of the start of the OS name. We then located the end of the word after it (the OS version) and stored both.

To obtain usernames and passwords from a telnet session, the payloads of all packets for the session were concatenated into a single string and then searched for the keywords “login:” and “password:”. This technique can be applied for both internal and external telnet servers. The test program relies on the presence of these keywords for the determination of user names and passwords. There are, however, some servers that do not utilize these keywords; for example, the Ataman telnet server does not echo “login:” as a prompt.

2.3.7 Identification

The Identification protocol (RFC 1413) is used to provide a verification of users with port pairs. The format of a response typically follows the format

```
<port1>, <port2> : USERID : <OS identifier> : <user identifier>
```

By identifying packet payloads with three colons and storing the last two entries, one can obtain the operating system and the username. The OS identifier is a string as defined in the “Official System Names” [7]. If the OS identifier is specified as “OTHER”, the user identifier may contain a username or an unformatted string that contains an encrypted audit token or other non-user ID information such as the description field from the */etc/passwd* file. In the test program, Identification responses were parsed by colons to produce user and OS strings.

2.3.8 Other Applications

For all other protocols, the best that can be done is to search for applications in the banners by locating the strings “server” or “service” in the payload and storing the entire line associated with it.

2.4 OS Fingerprinting

The operating system of a device can sometimes be identified through fingerprinting based on the behaviour of various fields in an IP packet [8–10]. The most common method is to examine the headers of TCP packets for OS signatures. A signature is a set of packet features that together define a fingerprint for an operating system.

Fingerprinting programs and signatures are available on the Internet: *p0f v1* [11] and *siphon* [12] are the two open-source programs, each with its own signature set, that were applied in the test program. While the *p0f* signatures are intended for use with SYN packets only, the *siphon* signatures appear to be applicable to all TCP packets. Both use TCP/IP header information to form its signatures, however *p0f* also includes TCP options. *p0f v2* has recently been released and includes signatures for SYN-ACK packets and ICMP packets.

Signature sets for TCP SYN and SYN-ACK packets were contributed by the Communications Research Centre (CRC), where each signature was rigorously tested [13]. To apply the SYN-ACK packet signatures, the instigating SYN packet must accompany the SYN-ACK packet.

In the test program, the *siphon*, *p0f v1* and CRC SYN rules were used on TCP SYN packets, and the CRC SYN-ACK rules were used on TCP SYN-ACK packets. In each case, the OS family and version were stored along with the rule definition. For example, the *p0f* signatures are defined by the features: window size, TTL, maximum segment size, DF bit set, window scaling, sackOK option, NOP option, and packet size. The second entry for the *p0f* rules, valid for Linux 2.0.35–2.0.38, is defined in the test program as an array of these features, as follows:

```
p0f_os(2,1).family = {'Linux'};
p0f_os(2,1).version = {'2.0.35'};
p0f_os(2,2).family = {'Linux'};
p0f_os(2,2).version = {'2.0.36'};
p0f_os(2,3).family = {'Linux'};
p0f_os(2,3).version = {'2.0.37'};
p0f_os(2,4).family = {'Linux'};
p0f_os(2,4).version = {'2.0.38'};
p0f_rule(2,:) = [512 64 1460 0 0 0 0 44];
```

Each OS family and version was concatenated to form a single string, and a matrix was formed with one row for each test and one column for each unique operating system name. Upon completion, the results for all tests were output. These results could be combined to determine the most probable OS family and version, using a method that is loosely based on the one used in *Xprobe v2.0* [14]. Each test is given a weight based on the "reliability" of the test – the accuracy of the open-source signatures is somewhat dubious since the signatures can be submitted by anyone and are not rigorously tested by the program authors.

To obtain a relative probability that the j^{th} OS is correct, we define $N(OS_j)$ as the weighted sum of the fingerprinting test results:

$$N(OS_j) = \sum_{k=1}^{n_{\text{test}}} \delta(k)\text{trust}(k) \tag{1}$$

where n_{test} is the number of OS tests, trust(k) is an arbitrary number assigned to reflect the level of trust in the test, and

$$\delta(k) = \begin{cases} 1, & \text{if a signature match was found to test } k \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

In the test program we used trust=1 for the CRC rules, trust=0.5 for *p0f v1* and trust=0.25 for *siphon*. The relative probability P is then

$$P(\text{OS}_j) = \frac{N(\text{OS}_j)}{\sum_k N(\text{OS}_k)}. \tag{3}$$

This method of applying multiple rules to determine the most likely OS allows us to add in new OS fingerprint rules from other sources. ICMP signatures can also be used to determine OS [14–17]. The CRC is also generating rules for TCP reset packets. TCP retransmission timeouts can also be exploited to identify operating systems [18]. Note that fingerprint evasion techniques [19] can limit the usefulness of strict OS fingerprinting [14]. The "fuzzy" technique used in *Xprobe* is worth investigating in further tool development.

2.5 Port and Protocol Usage

A host's port and protocol usage suggests the services used or being provided by the host. It also indicates the role, either client or server, the host is taking with respect to each service. Client/server differentiation is determined in the test program by SYN/SYN-ACK packets. An outgoing SYN-ACK packet indicates that the host is providing a TCP service. An outgoing SYN packet indicates that the host is acting as a client.

UDP services are not as clear-cut to determine without examining packet content, but one can infer the presence of a UDP service. In the test program, if an incoming UDP packet is responded to on the same IP/port pair a UDP service is assumed to be in use.

Device behaviour can signify a breach in security. For example, policy may not allow services to run on desktop machines. Client behaviour on a server or router may signify a policy violation.

2.6 ICMP Messages

The behaviour of a host defines its purpose on the network. A host may behave as a client, accessing the other hosts on the Internet, or a server, accepting connections from the Internet

(Section 2.5), or it may behave as a gateway or filtering device. The test program determines gateway and filtering device behaviour based on ICMP messages, as shown in Table 2.

A host can be flagged as non-operational if an *ICMP destination unreachable – host unreachable* packet is encountered. More ambiguously, a host may be non-operational if it has been inactive for a specified length of time. Perhaps this is a good example of where active methods would be quite useful; if a host has been inactive for a predetermined time, a ping could determine its status.

ICMP source quench messages indicate that a host is overloaded and effectively down. Combined with activity levels this gives an indication of link status.

Table 2: *ICMP messages relevant to identifying device behaviour.*

Type	Code	Meaning	Comes from...
3	0	Network unreachable	Gateway [20]
	1	Host unreachable	Gateway [20]
	4	Fragmentation needed	Gateway [20] [21]
	5	Source route failed	Gateway [20]
	6	Destination network unknown	Gateway [22]
	7	Destination host unknown	Gateway [22]
	8	Source host isolated (obsolete)	Gateway[1] [22]
	9	Destination network admin prohibited	Encryption device [23]
	10	Destination host admin prohibited	Encryption device [23]
	11	Network unreachable for TOS	Gateway [23]
	12	Host unreachable for TOS	Gateway [23]
	13	Communication administratively prohibited by filtering	Filtering gateway [23]
	14	Host precedence violation	Gateway [23]
	15	Precedence cutoff in effect	Gateway [23]
4	0	Source quench	Gateway or host[2] [20]
5	0/1/2/3	Redirect	Gateway [20]
8	0	Echo request	Client (not gateway [23])
9	0	Router advertisement	Gateway [24]
10	0	Router solicitation	Gateway [24]
11	0	TTL=0 during transit	Gateway [20]
	1	TTL=0 during reassembly	Host [20]
12	0/2	Parameter problem/bad length	Anyone [20]
	1	Required option is missing (military security code)	Client or server [22]
15	0	Information request (obsolete)	Anyone [20] Diskless workstations [22]
17	0	Address mask request	Diskless workstation [6]
18	0	Address mask reply	Gateway or host configured as an authoritative agent for address masks [25] [22]
30	0	Traceroute packet successfully forwarded	Gateway [26]
	1	Traceroute packet discarded	Gateway [26]
40	1/2/3/4/5	Security failures	Encryption devices

2.7 Activity level

The activity level of all hosts is important to network managers in identifying bandwidth abusers, and possibly misconfigurations. In the test program, the number of packets per unit time is taken as the basic measure of activity level.

2.8 Hop Depth

An attacker could use a packet's Time-To-Live (TTL) to get some idea of the topology of the network. The depth of a host is found by looking at the TTL field in the IP header of the packet. All operating systems have a default TTL that is assigned to a packet when it is sent out. Every time the packet passes through a router, the TTL field is decremented by 1. Assuming that the initial TTL can be determined, then the hop depth can be calculated.

2.8.1 Determining the initial TTL

The default initial TTL values are usually one of $\{16, 30, 32, 60, 64, 128, 255\}$ [27]. We assume here that the initial TTL value hasn't been modified from the default for the operating system being used. If we are sniffing close to the main gateway of a network, we assume that there are not more than 16 hops inside the network, so we choose the most likely initial TTL as being the first in the set that is greater than the current TTL.

2.8.2 Problems

We assume that the default TTL has not been manually changed and that there are not more than 16 hops within the network, but for some very large networks this is not the case. As well, this does not work for initial TTL values of 30 or 60 because we are really assuming no more than 4 hops, which isn't the case for large networks. In the test program, initial TTL values of 30 and 60 were not used as they are less common.

In our initial tests, some hosts appeared to be at multiple depths. This was caused by the inclusion of reset packets in the depth calculations. Investigation of the Linux 2.2 source code [28] for its TCP implementation confirmed that this implementation, at least, does not use the OS default TTL. Instead, it returns a reset with initial TTL value equal to the TTL of the packet to which the reset is responding. Because of this unreliability, reset packets were not used for depth calculations in the later tests. If multiple TTL values do now appear, it would suggest that there are multiple paths for that particular host.

2.9 Other Information Sources

The hostname may give away the function of the host. For example, many firewalls have "fw" in their hostname, and many mail servers have "mail" in their hostnames. The test program identifies these strings in hostnames.

2.10 Internal Protocols

As discussed in Section 1.2, internal protocols can be sniffed on separate physical segments and the results can be collected at a central location. On internal segments, the IP address and MAC address can be found in ARP and DHCP packets and the Ethernet and IP headers, and the hostname can be found in NetBIOS packets.

These protocols were not implemented in the test program as the distributed sniffer architecture is beyond the scope of the proof-of-concept tool. They are discussed briefly in the following sections.

2.10.1 Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) [29], taken in request/reply pairs, gives the host name corresponding to an IP address. DHCP is built on, but replaces, the BOOTP protocol. DHCP can be used through a router: when a request is received by the router, it is relayed to the real DHCP server.

2.10.2 Address Resolution Protocol

The Address Resolution Protocol (ARP) provides a mapping between the addressing system used by the protocol and the addressing system used by the hardware. For example, Ethernet uses 48-bit addresses (MAC addresses) and TCP/IP uses 32-bit addresses. The ARP request is broadcast, and the ARP reply, containing the mapping information, is directed back to the requester. In the example below, A.B.C.139 broadcasts a request for the Ethernet address of A.B.C.1. A.B.C.1 replies directly to A.B.C.139 with its Ethernet address.

```
13:34:03.593134 arp who-has A.B.C.1 tell A.B.C.139
13:34:03.593795 arp reply A.B.C.1 is-at 0:4:80:7d:c:0
```

2.10.3 NetBIOS

Port 137 UDP is reserved for the NetBIOS name service. The packets used in this service reveal usernames for a given IP address, and the type of account. UDP packets on port 138 are used for the NetBIOS datagram service. Such packets include information giving the IP, hostname, and type of host. The data portion of the packet may contain the physical location or other relevant information. TCP port 139 is reserved for the NetBIOS session service, such as when a shared directory is accessed. Packets involved in these sessions can reveal shared directory names and operating systems.

3 Implementation and Test Results

We chose to test the implementation by using data collected on a simulated network by DARPA for the 1999 Intrusion Detection Evaluation [30]. For that year, data was collected

on a simulated network at both internal and external locations. Three weeks of training data were provided: the first and third weeks of the training data do not contain any attacks, while the second week of the training data contains attacks. We selected training data sniffed external to the network for the Monday of the first week. A network diagram was provided (re-printed in Figure 2), as well as documentation of hostnames, operating systems, and the function of some hosts [31]. To allow for the existence of a large number of hosts on the network without actually having them there, internal workstations were simulated by a custom kernel modification to a single host's Linux 2.0.32 kernel (*hobbes*).

Appendix A shows the test program flow chart in Figure A.1, broken up into the TCP, UDP and ICMP protocols in Figures A.2, A.3 and A.4, respectively. In Appendix B, Tables B.1, B.2 and B.3 show the results obtained by the test program for the data provided by the outside sniffer, along with the expected results for the data. Table B.1 contains the results for the real internal hosts on the network. Table B.2 contains the results for the real external hosts on the network. Table B.3 contains the results for the simulated internal hosts.

The test program identified hosts and interfaces to routers and hubs. The MAC addresses for the real and simulated internal hosts were identical and differed for the real external hosts, as expected since the data was sniffed on an external segment. Any packet seen by the sniffer will have traversed a router and had its Ethernet address modified to that of the router.

For the real internal hosts, 5 out of 6 hostnames were resolved. Every host was used as a client at some time over the period, except for the router and hub. The router interface showed as hop depth of 0, and all others but *zeno* showed a hop depth of 1, as expected, with *zeno* having a hop depth of 5. Investigation showed a TTL of 59 at the sniffer, suggesting that the initial TTL for *zeno* is 60, and not 64 as was expected. Of the 6 real internal hosts, 4 OS determinations were of the correct family but not the correct version and 2 were incorrect. One of these was because there was no rule for the OS in question. For the other, there was no TCP traffic on which to determine the OS. The OS determinations with the correct family in general matched signatures from more than one test. The correct operating system version was listed as a possibility, but not as the most probable. Investigation shows that the signatures contributed by CRC are more often correct and the *p0f* and *siphon* signatures skewed the results.

Of the 5 external hosts, excluding the hub, one hostname was resolved and only one OS was found. This is because there was no TCP traffic from which to determine the OS. Note that the traffic generated by the hub consisted of ARP and echo requests. Although not implemented in the test program, these requests should classify it as a client. All of the external hosts were used as a client at some time over the period. The hop depth was found to be 0 for all external hosts, except for a hop depth of 1 for *aesop*. For this host, 2609 packets showed a TTL of 64 and 727 packets showed a TTL of 63. A lot of the TTL=63 packets are replays of identical packets with a TTL of 64, and all of the TTL=63 packets are destined to simulated external hosts. This suggests the possibility of a routing loop caused by the simulation of external hosts. Note that *aesop* did not list IRC in the documentation as

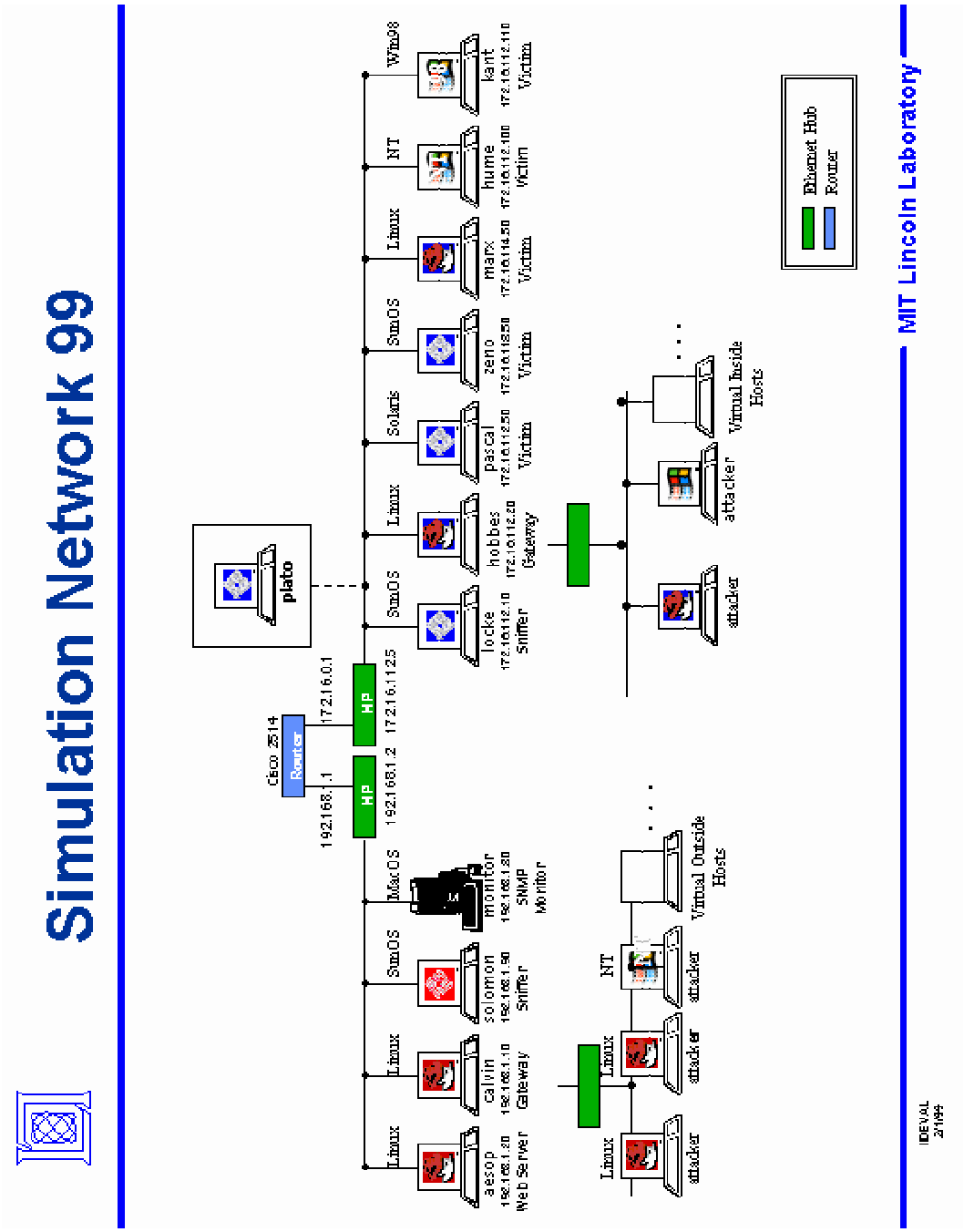

Figure 2: *Network topology of the simulated network used to create the 1999 DARPA ID Evaluation data [32].*

one of its services. This is an example of how passive methods can determine unexpected activity.

Twenty internal hosts were simulated by *hobbes*. As such, the MAC address and hop depth are identical for all of these hosts. The OS was found to be Linux 2.0.36 for all simulated internal hosts, corresponding to the OS found for *hobbes* by the test program. All server hostnames were resolved and every simulated internal host was found to have behaved as a client. Ten hosts offered services, and good agreement was found between the documentation and the test program results. It is estimated that 39 users existed on the simulated network due to the 39 e-mail addresses repeated on multiple machines. Multiple telnet username/password pairs were sniffed from the traffic.

4 Discussion and Conclusions

This work has shown that passive network discovery techniques can aid in network security. Whether for situational awareness of current status and activities or for auditing information leakage, such techniques can improve on the level of awareness provided by active discovery methods.

Passive OS identification signatures using TCP reset packets and ICMP packets could be implemented to identify operating system types more certainly. As seen in the results of the DARPA99 data, the operating system determined to be the best match might not be the true match due to the variation in accuracy of the signatures applied. Increasing the number of tests is expected to improve the results, and a rigorous testing of the open-source signatures would lend some insight to their viability.

Network Address Translation (NAT) can minimize information leakage. One host acts as a proxy for all of the hosts behind it. Although work has been done to attempt to estimate the number of hosts behind a proxy [33], the characteristics of the hosts behind the proxy cannot be seen. Traffic collected outside of the host performing NAT for the network can be processed to determine what information is being leaked.

The test program did not perform analysis of packets involving routing protocols, which could yield information regarding the structure of the network. This may need to be investigated in future work. As well, to apply passive methods to network management, internal protocols may need to be investigated further.

The test program was not configured to assess the success or failure of a login attempt for the Telnet protocol, which resulted in some false username/password pairs in the analysis of the DARPA99 data. This should be considered in future work. Future work should also include finding ways to isolate usernames and passwords without relying on the presence of keywords. Similarly, methods should be investigated to identify server banners in application headers without using keywords.

Passive techniques require time to enable a picture to be built up. They aren't good for

"snapshots" because they rely on the network to reveal itself through network activity. Combining active techniques with passive techniques addresses this issue while retaining the advantage of reducing network traffic and having a current picture of network activities. Passive techniques may be employed continuously, with active scans taking place at periodic intervals. An investigation into the optimal means of combining the two methods would be beneficial.

The time interval for which the results of a passive analysis are valid depends on the nature of the network under investigation. Strategic networks, which tend not to change very much over time, should have a longer time interval in which the analysis is valid. Tactical and ad-hoc networks, however, can change dramatically in relatively short time periods and therefore the time interval for the validity of the analysis is shorter.

References

1. So, H. (Jan. 2002). How can passive techniques be used to audit and discover network vulnerability? (Online). http://www.sans.org/resources/idfaq/passive_vuln.php (6 Nov 2003).

2. Long, J. (Apr. 2000). Passive Aggression - A New Paradigm for Information Gathering (IG) (Online). http://johnny.ihackstuff.com/security/passive_ig.doc (6 Nov 2003).

3. Sourcefire Inc. (June 2003). Real-Time Network Awareness (Online). http://www.sourcefire.com (9 July 2003).

4. Deraison, R., Gula, R., and Hayton, T. (Aug. 2003). Passive Vulnerability Scanning Introduction to NeVO (Online). Tenable Network Security. http://www.tenablesecurity.com/docs/passive_scanning_tenable.pdf (10 Oct 2003).

5. Gül, B. (2002). Network Situational Awareness Information Requirement Report. (CAC Project # 510-2929-03). Consulting and Audit Canada (for DRDC).

6. Stevens, W. R. (1994). TCP/IP Illustrated, Volume 1: The Protocols, Indianapolis, IN: Addison Wesley.

7. Internet Assigned Numbers Authority (IANA) (Apr. 2002). OPERATING SYSTEM NAMES (Online). http://www.iana.org/assignments/operating-system-names (30 Oct 2003).

8. Miller, T. (2002). Passive OS Fingerprinting: Details and Techniques (Online). The SANS Institute. http://www.incidents.org/papers/OSfingerprinting.php (1 Oct 2002).

9. Max Vision (2001). Passive Host Fingerprinting (Online). Whitehats Network Security Resource. http://www.whitehats.com/library/passive (1 Oct 2002).

10. Honeynet Project (Mar. 2002). Know Your Enemy: Passive Fingerprinting (Online). http://project.honeynet.org/papers/finger (1 Oct 2002).

11. The Evil Twin (2003). freshmeat.net: p0f 2.0.2 (Online). http://freshmeat.net/releases/136918 (30 Oct 2003).

12. Subterrain Security Group (2000). The Siphon Project: The Passive Network Mapping Tool (Online). http://siphon.datanerds.net (30 Oct 2003).

13. De Montigny-Leboeuf, A. (2002). Private communication.

14. Arkin, O. and Yarochkin, F. (Aug. 2002). Xprobe v2.0: A "Fuzzy" Approach to Remote Active Operating System Fingerprinting (Online). The Sys-Security Group. http://www.sys-security.com/archive/papers/xprobe2.pdf (30 Oct 2003).

15. Arkin, O. and Yarochkin, F. (2001). ICMP Based Remote OS TCP/IP Stack Fingerprinting Techniques. *Phrack*, Vol. 11.

16. Arkin, O. (Sept. 2000). ICMP Usage in Scanning (Online). The Sys-Security Group. http://www.sys-security.com/archive/papers/ICMP_Scanning_v2.01.pdf (30 Oct 2003).

17. Simple Nomad (July 2001). icmpid.c (Online). Nomad Mobile Research Centre. http://www.nmrc.org/project/nmrc/icmpid.c (6 Nov 2003).

18. Veysset, F., Courtay, O., and Heen, O. (Apr. 2002). New Tool And Technique For Remote Operating System Fingerprinting (Online). Intranode Research. www.intranode.com/fr/doc/ring-full-paper.pdf (30 Oct 2003).

19. Beck, R. (2001). Passive-Aggressive Resistance: OS Fingerprint Evasion. In *Linux Journal*, 89.

20. Postel, J. (1981). RFC 792: Internet Control Message Protocol. (Online). http://www.rfc-editor.org/rfc/rfc792.txt.

21. Mogul, J. and Deering, S. (1990). RFC 1191: Path MTU Discovery. (Online). http://www.rfc-editor.org/rfc/rfc1191.txt.

22. R. Braden, ed. (1989). RFC 1122: Requirements for Internet Hosts – Communication Layers. (Online). http://www.rfc-editor.org/rfc/rfc1122.txt.

23. Baker, F. (1995). RFC 1812: Requirements for IP Version 4 Routers. (Online). http://www.rfc-editor.org/rfc/rfc1812.txt.

24. Deering, S. (1991). RFC 1256: ICMP Router Discovery Messages. (Online). http://www.rfc-editor.org/rfc/rfc1256.txt.

25. Mogul, J. and Postel, J. (1985). RFC 950: Internet Standard Subnetting Procedure. (Online). http://www.rfc-editor.org/rfc/rfc950.txt.

26. Malkin, G. (1993). RFC 1393: Traceroute Using an IP Option. (Online). http://www.rfc-editor.org/rfc/rfc1393.txt.

27. Swiss Academic & Research Network (Apr. 1999). Default TTL Values in TCP/IP (Online). http://etna.switch.ch/docs/ttl_default.htm (2 Oct 2002).

28. kernel.org (Nov. 2003). The Linux Kernel Archives (Online). The kernel.org Organization, Inc.. http://www.kernel.org (20 Nov 2003).

29. Droms, R. (1993). RFC 1541: Dynamic Host Configuration Protocol. (Online). http://www.rfc-editor.org/rfc/rfc1541.txt.

30. Zissman, M. (2001). Simulation Network 99 (Online). MIT Lincoln Laboratory. http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html (18 Nov 2003).

31. Haines, J. W., Lippmann, R. P., Fried, D. J., Zissman, M. A., Tran, E., and Boswell, S. B. (2001). 1999 DARPA Intrusion Detection System Evaluation: Design and Procedures. (Technical Report 1062). MIT Lincoln Laboratory.

32. Laboratory, MIT Lincoln (Feb. 1999). Simulation Network 99 (Online). http://www.ll.mit.edu/IST/ideval/docs/1999/Network_Topology.gif (18 Nov 2003).

33. Bellovin, S. (2002). A Technique for Counting NATted Hosts. In *Proceedings of the Second Internet Measurement Workshop*, pp. 267–272. Marseilles, France: ACM SIGCOMM.

Annex A
Test Program Flow Charts

This section contains diagrams of the test program flow. The main body of the program is shown in Figure A.1. The protocol-dependent parts of the program are shown in Figures A.2, A.3 and A.4, for TCP, UDP and ICMP, respectively.

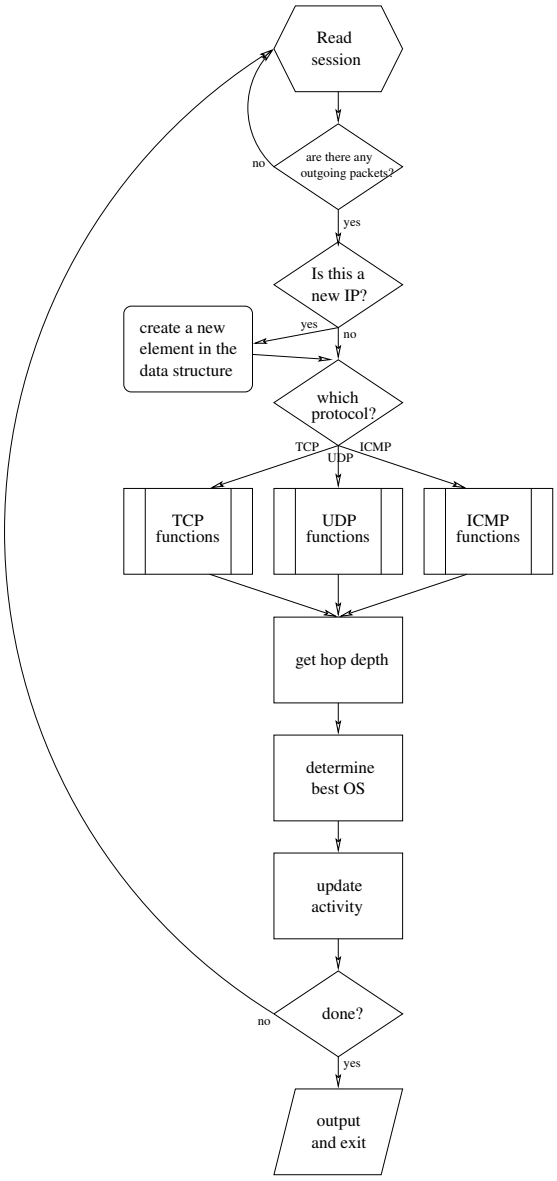

Figure A.1: *Flow chart of the main body of the program.*

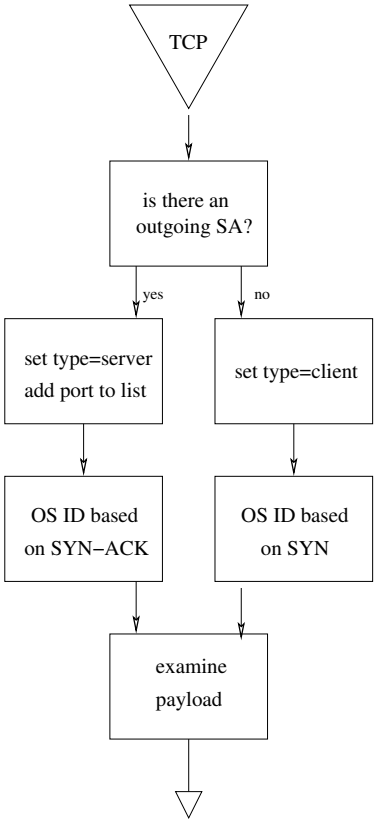

Figure A.2: *Flow chart for the TCP-specific section of the program.*

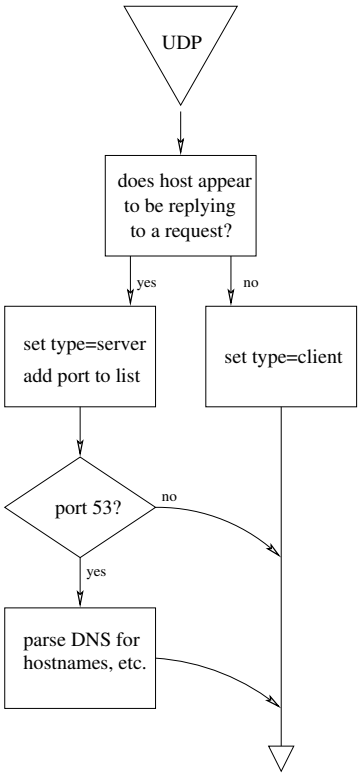

Figure A.3: *Flow chart of the UDP-specific section of the program.*

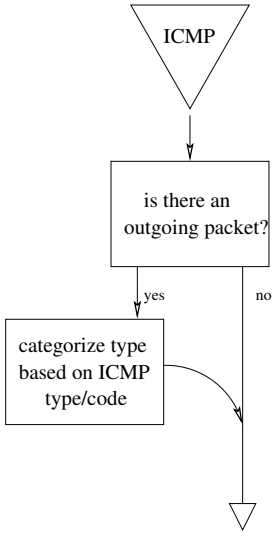

Figure A.4: *Flow chart of the ICMP-specific section of the program.*

Annex B
Results for the DARPA99 Training Data

Table B.1: Results for the DARPA99 training data, real internal hosts. The relative probability of each operating system (OS) is shown in brackets after the OS name.

IP	Parameter	Real value	Test program value
172. 16. 0. 1	Hostname		
	OS		Unknown
	MAC		0:10:7b:38:46:32
	Type	Cisco 2514 router, internal interface	udpserver
	Open ports		161
	Hop depth		0
172. 16.112. 5	Hostname	None	
	OS		Unknown
	MAC		0:10:7b:38:46:32
	Type	HP EtherTwist Hub	udpserver
	Open ports		161
	Hop depth		1
172. 16.112. 10	Hostname	locke.eyrie.af.mil	
	OS	Solaris 2.6	Unknown
	MAC		0:10:7b:38:46:32
	Type	inside sniffer	client, udpserver
	Open ports		123
	Hop depth		1
172. 16.112. 20	Hostname	hobbes.eyrie.af.mil	hobbes.eyrie.af.mil
	OS	Linux 2.0.32	Linux2.0.36 (0.38) Linux2.0.32 (0.31) Linux2.0.35 (0.08) Linux2.0.37 (0.08) Linux2.0.38 (0.08) Linux2.0.34 (0.08)
	MAC		0:10:7b:38:46:32
	Type	internal IP simulator, primary DNS	udpserver, client
	Open ports		53
	Applications	Bind 4.9.6	
	Hop depth		1
	Other Information		2 e-mails addresses found @beta.banana.edu and @zeno.eyrie.af.mil
172. 16.112. 50	Hostname	pascal.eyrie.af.mil	pascal.eyrie.af.mil
	OS	Solaris 2.5.1	Solaris2.6 (0.24)

Table B.1: (continued)

IP	Parameter	Real value	Test program value
			Solaris7 (0.20) Solaris2.5.1 (0.20) Linux2.0.36 (0.12) Linux2.0.32 (0.10) Solaris2.7 (0.05) Linux2.0.35 (0.02) Linux2.0.37 (0.02) Linux2.0.38 (0.02) Linux2.0.34 (0.02)
	MAC		0:10:7b:38:46:32
	Type		tcpserver, client
	Open ports		23 25 22 79
	Applications	telnet, smtp, ssh, ftp, finger	Sendmail SMI-8.6/SMI-SVR4
	Hop depth		1
	Other Information		OS from Telnet: System V R, 33 Telnet login/password pairs, 33 e-mail accounts found, 3 FTP login/password pairs found
172. 16.112.100	Hostname	hume.eyrie.af.mil	hume.eyrie.af.mil
	OS	Windows NT 4 sp1	WindowsNT 4 (0.17) WindowsNT 4 sp3 (0.17) WindowsNT 4 sp4 (0.17) WindowsNT 4 sp6 (0.17) Windows98 (0.13) Windows98 SE (0.09) WindowsNT (0.04) Windows95 (0.04)
	MAC		0:10:7b:38:46:32
	Type		udpserver, client, tcpserver
	Open ports		161 21 25 80 23
	Applications	IIS 2.0 (FTP, gopher, web), MailSrv, ataman telnet	Microsoft FTP Service (Version 2.0), SMTP Server for hume ready, Microsoft-IIS/2.0
	Hop depth		1
	Other Information		90 e-mail addresses found, e-mail address was found to be the password for the FTP server, 58 additional addresses found, 5 telnet passwords found
172. 16.113. 50	Hostname	zeno.eyrie.af.mil	zeno.eyrie.af.mil

Table B.1: (continued)

IP	Parameter	Real value	Test program value
	OS	SunOS 4.1.4	Linux2.0.36 (0.33) Linux2.0.32 (0.27) SCOUnixWare 2.1.2 (0.13) Linux2.0.35 (0.07) Linux2.0.37 (0.07) Linux2.0.38 (0.07) Linux2.0.34 (0.07)
	MAC		0:10:7b:38:46:32
	Type		tcpserver, client
	Open ports		25 23 79
	Applications	sendmail, ftp, telnet, finger	Sendmail 4.1/SMI-4.1
	Hop depth		5
	Other Information		OS from Telnet: SunOS UNIX, 6 FTP username/password pairs found, 13 telnet username/password pairs found, 18 e-mail addresses found
172. 16.114. 50	Hostname	marx.eyrie.af.mil	marx.eyrie.af.mil
	OS	Linux 2.0.30	Linux2.0.36 (0.38) Linux2.0.32 (0.31) Linux2.0.35 (0.08) Linux2.0.37 (0.08) Linux2.0.38 (0.08) Linux2.0.34 (0.08)
	MAC		0:10:7b:38:46:32
	Type		client, tcpserver
	Open ports		25 22 80 23 79
	Applications	Apache 1.1.3, sshd 1.2.25, snmpd, telnet, ftp, finger	Sendmail 8.8.5/8.8.5 Apache/1.1.3
	Hop depth		1
	Other Information		OS from Telnet: Linux release 4, 7 telnet username/password pairs found, 16 e-mail addresses found

Table B.2: Results for the DARPA99 training data, real external hosts. The relative probability of each operating system (OS) is shown in brackets after the OS name.

IP	Parameter	Real value	Test program value
192.168. 1. 1	Hostname	loud.world.net	
	OS		Unknown
	MAC		0:10:7b:38:46:32
	Type	Cisco 2514 Router, external interface	client
	Hop depth		0
192.168. 1. 2	Hostname	None	
	OS		Unknown
	MAC		8:0:9:61:aa:c9
	Type	HP EtherTwist Hub	
	Hop depth		0
192.168. 1. 10	Hostname	calvin.world.net	
	OS	Linux	Unknown
	MAC		0:c0:4f:a3:58:23
	Type	outside gateway, external IP simulator	client, udpserver
	Open ports		53 123
	Applications		
	Hop depth		0
	Other Information		
192.168. 1. 20	Hostname	aesop.world.net	aesop.world.net
	OS	Linux	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	MAC		0:60:97:de:54:36
	Type	outside web server	udpserver, client, tcpserver
	Open ports		53 6667
	Applications		Internet Relay Network version u2.10.04
	Hop depth		1
	Other Information		
192.168. 1. 30	Hostname	monitor.af.mil	
	OS	MacOS	Unknown
	MAC		8:0:7:ce:5:21
	Type		client
	Hop depth		0
192.168. 1. 90	Hostname	solomon.world.net	

Table B.2: (continued)

IP	Parameter	Real value	Test program value
	OS	SunOS	Unknown
	MAC		8:0:20:11:34:bf
	Type		client
	Hop depth		0

Table B.3: Results for the DARPA99 data, simulated internal hosts. The relative probability of each operating system (OS) is shown in brackets after the OS name. The MAC address is 0:10:7b:38:46:32 for all hosts, corresponding to the external interface of the router, and the hop depth is 1.

IP	Parameter	Real value	Test program value
172. 16.112.149	Hostname	eagle.eyrie.af.mil	eagle.eyrie.af.mil
	OS	Linux 2.0.32	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		tcpserver, client
	Open ports		25 79 23
	Applications		Sendmail 8.8.7/8.8.7
	Other Information		1 external telnet username/password pair found @eagle.eyire.af.mil, 39 e-mail addresses found
172. 16.112.194	Hostname	falcon.eyrie.af.mil	falcon.eyrie.af.mil
	OS	Solaris 2.5.1	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		tcpserver, client
	Open ports		23 25 79
	Applications		Sendmail SMI-8.6/SMI-SVR4
	Other Information		11 telnet username/password pairs found, 1 external telnet username/password pair found, 39 e-mail addresses found
172. 16.112.207	Hostname	robin.eyrie.af.mil	robin.eyrie.af.mil
	OS	SunOS 4.1.4	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client, tcpserver
	Open ports		25 79 23
	Applications		Sendmail 4.1/SMI-4.1

Table B.3: (continued)

IP	Parameter	Real value	Test program value
	Other Information		37 e-mail addresses found @robin.eyrie.af.mil, 2 external telnet username/password pairs found
172. 16.113. 84	Hostname	duck.eyrie.af.mil	duck.eyrie.af.mil
	OS	SunOS 4.1.4	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		tcpserver, client
	Open ports		113 25 79
	Applications		Sendmail 4.1/SMI-4.1
	Other Information		3 Ident users found, 39 e-mail addresses found (@duck.eyrie.af.mil), 2 external telnet username/password pairs
172. 16.113.105	Hostname	swallow.eyrie.af.mil	swallow.eyrie.af.mil
	OS	Linux 2.0.32	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client, tcpserver
	Open ports		25 79 113
	Applications		Sendmail 8.8.7/8.8.7
	Other Information		39 e-mail addresses found @swallow.eyrie.af.mil, 4 external telnet username/password pairs found, 1 user found through ident
172. 16.113.204	Hostname	goose.eyrie.af.mil	goose.eyrie.af.mil
	OS	Solaris 2.5.1	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		tcpserver, client
	Open ports		113 25 79 23

Table B.3: (continued)

IP	Parameter	Real value	Test program value
	Applications		Sendmail SMI-8.6/SMI-SVR4
	Other Information		39 e-mail addresses found @goose.eyrie.af.mil, 2 Ident users found, 1 External Telnet username/password pair found, 1 telnet password found
172. 16.114.148	Hostname	crow.eyrie.af.mil	crow.eyrie.af.mil
	OS	Linux 2.0.32	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client, tcpserver
	Open ports		21 25 79
	Applications		wu-2.4.2-academ[BETA-15](1) Sendmail 8.8.7/8.8.7
	Other Information		anonymous FTP server with e-mail address as password, 39 e-mail addresses found
172. 16.114.168	Hostname	finch.eyrie.af.mil	finch.eyrie.af.mil
	OS	SunOS 4.1.4	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client, tcpserver
	Open ports		25 79 113
	Applications		Sendmail 4.1/SMI-4.1
	Other Information		39 e-mail accounts found, 2 external telnet username/password pairs, 1 user found from Ident
172. 16.114.169	Hostname	swan.eyrie.af.mil	swan.eyrie.af.mil
	OS	Solaris 2.5.1	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		tcpserver, client

Table B.3: (continued)

IP	Parameter	Real value	Test program value
	Open ports		25 79 23
	Applications		Sendmail SMI-8.6/SMI-SVR4
	Other Information		2 external telnet username/password pairs, 1 telnet username/password pair, 39 e-mail addresses found
172. 16.114.207	Hostname	pigeon.eyrie.af.mil	pigeon.eyrie.af.mil
	OS	Linux 2.0.32	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client, tcpserver
	Open ports		25 79
	Applications		Sendmail 8.8.7/8.8.7
	Other Information		3 external telnet username/password pairs found,38 e-mail addresses found
172. 16.115. 5	Hostname	pc1.eyrie.af.mil	
	OS	Windows 95	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found
172. 16.115. 87	Hostname	pc2.eyrie.af.mil	pc2.eyrie.af.mil.
	OS	Windows 95	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found
172. 16.115.234	Hostname	pc0.eyrie.af.mil	

Table B.3: (continued)

IP	Parameter	Real value	Test program value
	OS	Windows NT sp1	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		
172. 16.116. 44	Hostname	pc5.eyrie.af.mil	
	OS	Windows 3.1	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found
172. 16.116.194	Hostname	pc3.eyrie.af.mil	
	OS	Windows 95	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found
172. 16.116.201	Hostname	pc4.eyrie.af.mil	
	OS		Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found
172. 16.117. 52	Hostname	pc7.eyrie.af.mil	

Table B.3: (continued)

IP	Parameter	Real value	Test program value
	OS	Windows 3.1	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found
172. 16.117.103	Hostname	pc9.eyrie.af.mil	
	OS	MacOS	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found
172. 16.117.111	Hostname	pc8.eyrie.af.mil	
	OS	MacOS	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found
172. 16.117.132	Hostname	pc6.eyrie.af.mil	
	OS	Windows 3.1	Linux2.0.36 (0.43) Linux2.0.32 (0.38) Linux2.0.35 (0.05) Linux2.0.37 (0.05) Linux2.0.38 (0.05) Linux2.0.34 (0.05)
	Type		client
	Other Information		1 external POP username/password pair found

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence R&D Canada – Ottawa
3701 Carling Ave., Ottawa, ON K1A 0Z4

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

An Overview of Passive Information Gathering Techniques for Network Security (U)

4. AUTHORS (Last name, first name, middle initial)

Treurniet, Joanne R.

5. DATE OF PUBLICATION (month and year of publication of document)

May 2004

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

44

6b. NO. OF REFS (total cited in document)

33

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

15bf28

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DRDC Ottawa TM 2004-073

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(x) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Full unlimited announcement

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

Network awareness is a crucial aspect of network security, and is usually achieved through the use of active scanning. This method periodically introduces a large amount of traffic on the network, using up bandwidth. Much of the information obtained through active methods may also be obtained by passively listening to traffic. This document explains how passive scanning methods can aid in achieving network awareness without introducing unnecessary traffic, and documents a proof-of-concept tool to show how the methods presented might be implemented.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

network discovery, passive network discovery

Defence R&D Canada

Canada's leader in defence
and national security R&D

R & D pour la défense Canada

Chef de file au Canada en R & D
pour la défense et la sécurité nationale

www.drdc-rddc.gc.ca