Windows NT Attacks for the Evaluation of Intrusion Detection Systems*

by

Jonathan Korba

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Author.…………………………………………………………………………………...
Department of Electrical Engineering and Computer Science,
May 22, 2000

Certified by………………………………………………………………………………
Richard Lippmann
Senior Scientist, MIT Lincoln Laboratory
Thesis Supervisor

Accepted by……………………………………………………………………………...
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

*This work was sponsored by the Department of Defense Advanced Research Projects Agency under Air Force Contract F19628-95-C-002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force.

Windows NT Attacks for the Evaluation of Intrusion Detection Systems

by
Jonathan Korba

Submitted to the Department of Electrical Engineering and Computer Science

May 22, 2000

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

The 1999 DARPA Off-Line Intrusion Detection Evaluation provided a standard corpus for evaluating intrusion detection systems. It improved on the 1998 evaluation by providing training data containing no attacks to train anomaly detection systems, scoring systems on attack identification in addition to attack detection, simplifying scoring and verification procedures, providing a written security policy, and performing more detailed analysis of missed detections and false alarms. It also introduced more stealthy attacks, insider attacks, and attacks against the Windows NT operating system.

The focus of this thesis is the integration of Windows NT systems, background traffic, and attacks into the 1999 evaluation. Three Windows NT systems were added to the original test bed network: a victim machine, an outside attacker machine, and an insider attacker machine. The victim machine is a server with 92 user accounts, telnet, FTP, email, and web services, and security auditing. UNIX scripts from the 1998 evaluation were modified to create Windows NT background traffic. In addition, web traffic originating from the server was automated by developing a Javascript program called AutoBrowser.

A realistic and relatively comprehensive set of 12 Windows NT attacks was developed for the 1999 evaluation. The set includes denial-of-service attacks, remote-to-local attacks, user-to-root attacks, probe attacks, insider attacks, console-based attacks, a man-in-the-middle attack, and an attack using macro code in a Microsoft application. Signatures in network traffic and Windows NT host data were analyzed for each attack. A PERL program called NTAD (ntaudit-detect.pl) was developed to evaluate the detectability of the Windows NT attacks in audit log data. NTAD successfully used the attack signatures to detect attack instances in Windows NT audit logs collected during the evaluation.

Thesis Supervisor: Richard Lippmann
Title: Senior Scientist, MIT Lincoln Laboratory

Acknowledgments

I would like to thank my thesis advisor, Richard Lippmann, for supporting my thesis efforts and sharing his knowledge. I would like to thank David Fried and Raj Basu for taking the time to revise and edit my thesis. I would also like to thank Rich, Dave, Raj, Robert Cunningham, Joshua Haines, Kristopher Kendall, Seth Webster, Jesse Rabek, Kumar Das, and the rest of the intrusion detection group for always being friendly, fun, and helpful. Finally, I thank my parents, whose constant love and support give me confidence in all aspects of life.

Contents

List of Figures

Chapter 1

Introduction

1.1 DARPA Intrusion Detection System Evaluations

Widespread use of networked computers has made computer security a serious issue. Every networked computer, to varying degrees, is vulnerable to malicious computer attacks that can result in a range of security violations, such as, unauthorized user access to a system or the disruption of system services. Traditionally, computer security approaches have focused on preventing such attacks from occurring through the use of firewalls and security policies. However, for most systems, complete attack prevention is not realistically attainable due to system complexity, configuration and administration errors, and abuse by authorized users. For this reason, attack detection has been an important aspect of recent computer security efforts [27].

Systems designed to detect computer attacks are called intrusion detection systems. They monitor computers and networks for attacks that are inevitable, despite security precautions. If an attack is discovered, intrusion detection systems can alert an administrator, defend against the attack, or provide forensic information that may help prevent future attacks. Intrusion detection systems are not all equal in capabilities or reliability. A particular system may only detect a specific subset of possible attacks. In addition, it may have a different level of detection accuracy or a different false alarm rate than other systems. Results from intrusion detection system evaluations allow users to make informed decisions on what system to use, and are extremely important for guiding research. The importance of evaluating intrusion detection systems has prompted the development of the DARPA Off-Line Intrusion Detection Evaluations. A primary goal of these evaluations is to generate standard evaluation corpora that can be used off-line by many sites to evaluate a wide variety of intrusion detection systems.

1.2 The 1998 Evaluation

The 1998 DARPA off-line intrusion detection evaluation was the first annual evaluation under DARPA ITO and Air Force Research Laboratory sponsorships. It produced the first standard corpus for evaluating computer intrusion detection systems. Six different intrusion detection systems were evaluated. Seven weeks of training data with labeled attacks were produced for system development, followed by two weeks of test data with unlabeled attacks used for a blind evaluation.

A test bed of computers used to produce the data emulated 100's of users interacting on 1000's of hosts. Along with realistic background traffic, there were over 300 instances of 38 different attacks against three UNIX victim machines (SunOS, Solaris, and Linux operating systems). The test data included novel attacks created specifically for the evaluation, recent new attacks, and attacks in the training data. Details of the 1998 evaluation can be found in [10], [11], and [13].

The results of the evaluation were analyzed by plotting attack detection rates versus false alarm rates using receiver operating characteristic curves (ROCs). Many intrusion detection systems were able to detect the attacks used in the training data attacks with high accuracy (63% to 93%) and few false alarms (10 per day). However, systems did not perform well with new and novel attacks. The top three systems missed all of the novel attacks and approximately half of the new attacks. An analysis of the results revealed that participating systems could reliably detect known attacks if the systems were tuned using those attacks from the training data. However, many systems did not reliably detect dangerous new attacks, especially when the attack mechanism or TCP/IP service differed from attacks used for system training [13].

The 1998 evaluation was successful in providing an unbiased, realistic, and comprehensive evaluation of a diverse set of intrusion detection systems. More than 80 sites have downloaded all or part of the 1998 corpus from the MIT Lincoln Laboratory web site [11]. This indicates the extensive interest in obtaining training and test corpora for the development and evaluation of intrusion detection systems. Those who participated in the 1998 evaluation made several suggestions for improvements. These suggestions included, providing training data containing no attacks to train anomaly detection systems, scoring systems on attack identification in addition to attack detection, simplifying scoring and verification procedures, providing a written security policy, and performing more detailed analysis of attack misses and false alarms. Almost all of the suggestions were incorporated in the 1999 evaluation. In addition, the 1999 attack set was extended to include more stealthy attacks [6], insider attacks, and attacks against the Windows NT operating system. The 1998 data set only contained attacks against Sun, Solaris, and Linux operating systems from attack machines outside of the victim network.

1.3 Windows NT Attacks for the 1999 DARPA Evaluation

This thesis describes the development and analysis of attacks against the Windows NT operating system for the 1999 DARPA evaluation. It is important that intrusion detection systems are capable of detecting attacks against the Windows NT operating system because of its growing importance in government and commercial environments. For this reason, it was decided that the 1999 evaluation should test intrusion detection systems with both UNIX and Windows NT attacks. This decision required several modifications to the 1998 test bed, including the addition of Windows NT computers, background traffic representing a Windows NT environment, and most importantly, attacks against the Windows NT operating system.

1.4 Outline of the Thesis

This thesis is organized as follows. Chapter 2 provides background information about the 1998 evaluation test bed network, traffic generation, and input data for intrusion detection systems. Chapter 3 details how the test bed was modified to integrate Windows NT machines, how the machines were configured, and the type of Windows NT traffic generated in the test bed.

Chapter 4 describes how Windows NT attacks were developed and analyzed for the 1999 evaluation. Chapter 5 defines the final set of Windows NT attacks. It also gives an overview of an attack taxonomy that guided the selection of the attacks. Chapters 6 through 9 classify and document each Windows NT attack used in the 1999 evaluation. For each attack, there is a description, along with directions for execution, verification, and cleanup in the test bed network. There is also a description of how each attack may be detected in network traffic and audit logs.

Chapter 10 discusses post-evaluation work performed with Windows NT audit logs. Audit logs from the evaluation were analyzed to test the host-based detectability of Windows NT attacks, and test the validity of predicted audit log attack signatures. The goal of the analysis was to make it easier for developers to extend existing systems to detect Windows NT attacks.

Finally, Chapter 11 summarizes the results of 1999 DARPA Intrusion Detection Evaluation with regards to the Windows NT attack set. Suggestions are presented for future work in upcoming evaluations.

Chapter 2

Background

2.1 The 1998 Evaluation Test Bed Network

A conceptual view of the original test bed network used in the 1998 DARPA Intrusion Detection Evaluation is shown in Figure 2-1. This test bed generates traffic similar to

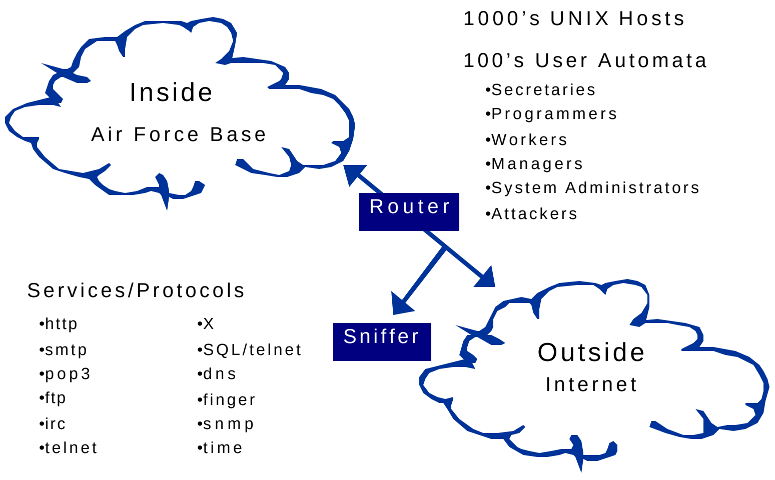


Figure 2-1: Conceptual View of the Original 1998 Evaluation Test Bed

that seen between a small Air Force Base network and the Internet. Custom software emulates 100's of users using UNIX applications and common network services. The network traffic produced by these users includes sending and receiving email, using FTP to send and receive files, accessing other computers via telnet sessions, sending and receiving IRC messages, and browsing web pages. Custom software also makes it possible for a small number of physical hosts to appear as if they are 1000's of hosts with different IP addresses. In this original 1998 test bed, all of the hosts are UNIX machines and all attacks originate from outside of the Air Force Base. A sniffer positioned outside of the base collects all network traffic originating from the Internet, including all of the attacks.

Figure 2-2 shows a more detailed view of the test bed network. The Air Force Base network contains four machines that are the victims of the attacks. The operating

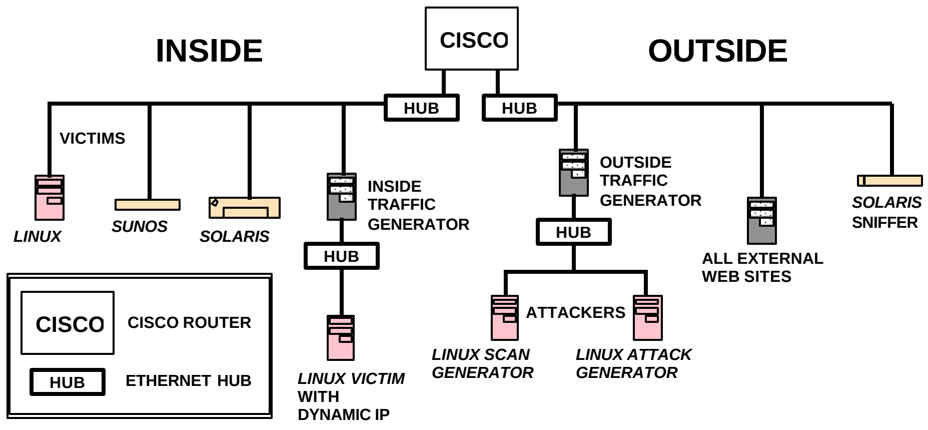


Figure 2-2: Detailed Diagram of the 1998 Evaluation Test Bed Network Topology.

systems of the machines are SunOS 4.1.4, Solaris 2.5.1, Linux 4.2 and Linux 5.0. The Linux 5.0 victim has the ability to dynamically change IP addresses. The remaining computer in the inside network is a traffic generator which emulates all other inside IP addresses.

The outside network, representing the rest of the Internet, contains two Linux machines for launching manual attacks that cannot be easily automated. The remaining three machines are a traffic generator, a sniffer, and a web server. The traffic generator emulates hundreds of outside workstations. It generates background traffic originating outside of the Air Force Base network and launches all automated attacks. The sniffer records all network traffic destined for the Air Force Base, including all of the attacks. The external web server mimics thousands of Internet web sites. The inside and outside traffic generators and the outside web server are equipped with modified operating systems which allow them to emulate "virtual" machines with different IP addresses [10].

2.2 Traffic Generation

Custom software automates most of the background traffic and attack traffic in the test bed. The software was designed to provide automatic, reproducible, and robust traffic generation. To achieve these design goals, the Expect scripting language was chosen, as suggested in [28]. It allows the creation of sessions that emulate users typing at computer keyboards.

The Expect traffic generator automatically launches specially formatted "exs" scripts for each attack instance [10]. If an error occurs when generating or collecting the traffic, the same "exs" scripts can be easily rerun. The "exs" scripts are also used to automate most of the background traffic and attack traffic for the test bed. Sessions that cannot be automated using "exs" scripts are manually executed. Examples of such manual traffic include traffic created by GUI interaction, such as X Windows.

2.3 Input Data for Intrusion Detection Systems

There are many sources of information that an intrusion detection system can utilize for attack detection. Some systems, called network-based intrusion detection systems, rely on information collected by sniffing network traffic. Other systems, called host-based intrusion detection systems, use data gathered from and specific to an individual host computer. There are some systems that utilize both sources of information.

The 1998 DARPA evaluation collected the information necessary to satisfy the inputs for all of the participating intrusion detection systems. A program called tcpdump [12], running on the Solaris sniffer, recorded the network traffic in the test bed. In addition, the participants were provided with various types of host data. Sun Basic Security Module (BSM) audit data was collected from the Solaris victim machine and nightly file dumps were provided from all three UNIX victim machines. After all of the data was collected from the test bed for the 1998 evaluation, it was written to CD-ROMs and posted on a web site for the participants to download [11].

Chapter 3

Windows NT in the 1999 Test Bed Network

Many steps were necessary to integrate Windows NT into the 1999 evaluation. Machines were connected to the inside and outside networks and configured for the test bed, Windows NT services and applications were installed, and host security auditing was configured. Figure 3-1 shows the updated 1999 test bed network. Machines that were not present in the 1998 test bed are labeled with underlined text. New machines unrelated to Windows NT include a Linux machine for insider attack generation and an insider sniffer to collect traffic generated by insider attacks.

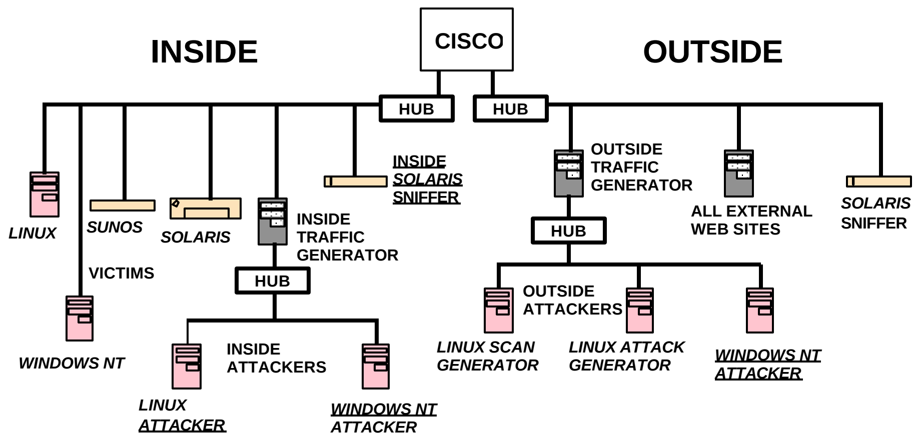


Figure 3-1: Detailed Diagram of the 1999 Evaluation Test Bed Network Topology with Underlined Text Indicating Machines that did not Exist in the 1998 Evaluation.

3.1 Machines

Three Windows NT machines were added to the test bed. One Windows NT victim machine is in the inside network. In addition, there are two Windows NT attack machines: an inside attacker and an outside attacker. All attacks, including Windows NT attacks, originate from one of the dedicated Windows NT or Linux attack machines. This convention makes it possible to separately sniff and collect network traffic specific to each attack. The collected network attack sniffer data can be used to verify the success of attacks and for analysis of attack signatures.

3.2 Configurations and Software

Windows NT Domain Server 4.0 (Build 1381) is installed on the Windows NT victim machine. It is the primary domain server (PDS) for the Eyrie Air Force Base Windows NT network. The inside Windows NT attacker machine is setup with Windows NT Workstation 4.0, as a workstation in the victim machine's domain. The outside Windows NT attacker is a stand-alone workstation also setup with Windows NT Workstation 4.0. All of the Windows NT machines' operating systems include installations of Service Pack 1. No additional Service Packs were installed.

3.2.1 Services and Applications

Several services are installed on the Windows NT victim machine. Included in the operating system is IIS (Internet Information Server) version 2.0, which provides FTP, Gopher, and web services. Figure 3-2 presents the IIS settings for the 1999 evaluation. The FTP and web services allow anonymous connections and all connections are logged. To ensure that normal background traffic connections do not overload the services, the maximum number of simultaneous connections for each service is set at a level high enough to accommodate the number of connections generated in the background traffic.

	FTP	**Web**	**Gopher** Not used in 1999 evaluation
Port Number	21	80	N/A
Max Simultaneous Connections	1,000	100,000	N/A
Root Directory	C:\InetPub\ftproot	C:\InetPub\wwwroot	N/A
Anonymous Access	Yes	Yes	N/A
Sessions Logged to	C:\Winnt\System32\Logfiles\	C:\Winnt\System32\Logfiles\	N/A

Figure 3-2: IIS Settings for the Windows NT Victim Server.

The Resource Kit for Windows NT 4.0 is installed, separate from the operating system. It includes various utilities and services, one of which is a mail server, called MailSrv, used in the evaluation. The MailSrv program has a bug that can cause SMTP connections to hang and eventually consume 100 percent of the available CPU cycles on a Windows NT machine [23]. Unfortunately, the bug was not discovered in the evaluation until several days of the evaluation had already been completed. The more reliable Microsoft Exchange Mail Server could not be used because it requires Service Pack 3. To remedy the situation for the 1999 evaluation, MailSrv was stopped and restarted whenever the machine's CPU utilization reached 100%. This happened about once or twice a day.

In addition to MailSrv, the Resource Kit includes the following UNIX commands, which are interpreted by the operating system, via the Windows NT POSIX subsystem:

- cat
- chmod
- chown
- cp
- find
- grep
- ln
- ls
- mkdir
- mv
- rm
- rmdir
- sh
- touch
- vi
- wc

The Resource Kit also provides a telnet service, telnetd.exe. However, the program is a beta version [16]. When it was tested in the evaluation test bed, it was very unreliable and crashed frequently. Therefore, a third party telnet service was chosen. To provide reliable telnet capabilities, Ataman TCP Remote Logon Service (version 2.4) is installed [1] on the Windows NT victim machine.

The common software applications, Netscape 4.0.8 and Microsoft Office 97, are installed on all of the Windows NT machines in the evaluation. In addition, some Windows NT attacks developed for the 1999 evaluation required additional common software programs to be installed so that the attacks could be properly executed in the test bed environment. Such software programs include a compression utility, WinZip 7.0 [43], and a utility to gather web server statistics, Wusage 6.0 [2].

3.2.2 User and Group Accounts

The Windows NT victim machine stores 92 user accounts in its user account database. Of those 92, 89 accounts are normal users. Their accounts never expire and their passwords never expire. Their privileges allow telnet access and FTP access to the system, but do not allow local logins. Each user can remotely access the machine via telnet any day of the week, at any time. The remaining three of the 92 user accounts are the following, and exist by default:

- *Administrator:* This root account allows remote and local logins and full control of system software.
- *Guest:* This default account, setup by the operating system, allows limited anonymous access to system resources.
- *IUSR_<machine name>:* This default account, setup by IIS, provides web access for anonymous Internet users.

Windows NT also supports group accounts. All members of a group account inherit the privileges of the group. Figure 3-3 presents the user group accounts for the Windows NT victim machine, a brief description of each group, and the users who are group members [34].

Group	Description	Users
Account Operators	Members can administer domain user and group accounts	Administrator
Administrators	Members can fully administer the computer/domain	Administrator
Backup Operators	Members can bypass file security to back up files	Administrator
Domain Admins	Designated administrators of the domain	Administrator
Domain Guests	Users granted guest access to the domain	Administrator, Guest
Domain Users	All domain users	All users
EAFB_Users	Ordinary users of the Air Force Base network	All users except for Guest
Guests	Users granted guest access to the computer/domain	All users
Print Operators	Designated administrators of domain printers	Administrator
Secret	Users granted access to Air Force secret files in d:\home\secret	Administrator and four ordinary users chosen at random
Replicator	Supports file replication in a domain	Administrator
Server Operators	Designated administrators of domain servers	Administrator
Users	Ordinary users	All users except for Guest

Figure 3-3: User Groups for the Windows NT Victim Server.

3.2.3 Security Auditing

The Windows NT event logging service maintains three event logs: a system log, an application log, and a security log. The system log primarily records device failures and I/O errors, and the starting and stopping of services. The application log records application defined messages, such as failure to allocate memory or failure to access a system object. The security log is the repository for all Windows NT security audit information. Windows NT security auditing is built-in to the event logging service and satisfies the requirements for the C2 security evaluation class [17]. These requirements are:

- The system has the ability to record all security-related events that occur on the system in the form of audit records.
- The system provides a way for the audit records to be reviewed by the system administrators.
- The auditing software and logs must be protected by the operating system from unauthorized access and modification, and access must be limited to authorized system administrators.
- A mechanism must exist that allows the selection of security events to be audited.
- The system must be able to audit individual users.

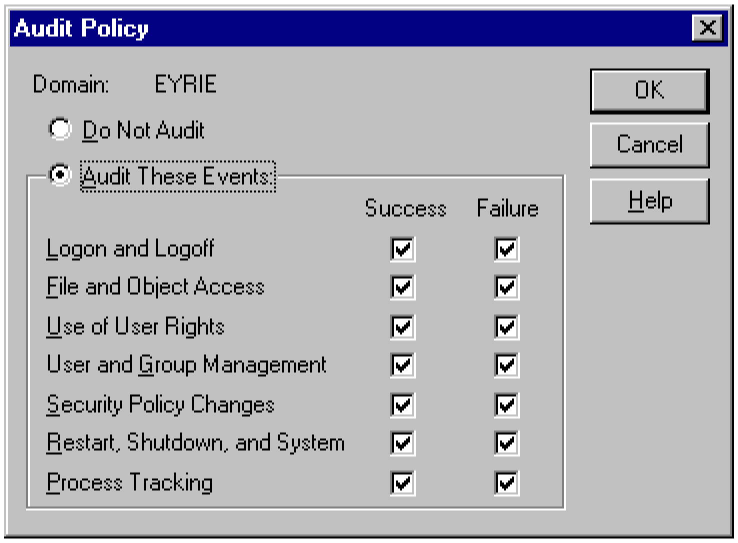
The Windows NT User Manager audit policy window, shown in Figure 3-4, is used to select which types of security events are audited. Full auditing for all user accounts is enabled in the Windows NT victim machine's User Manager for the 1999 evaluation. In addition, auditing of base objects is enabled with the following value added to the

Registry key,

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\AuditBase-Objects:

```
    Name: AuditBaseObjects
    Type: REG_DWORD
    Value: 1
```

Everything is viewed as an object by the Windows NT operating system (files, drives, memory, etc.) By enabling base object auditing, low-level activities, such as memory requests by a process, are recorded by the logging service. However, to audit access of specific files, printer use, and Registry access, settings must be adjusted in the Windows NT Explorer, Control Panels, and Registry Editor. These aspects of Windows NT



Domain: EYRIE

( ) Do Not Audit

(●) Audit These Events:

	Success	Failure
Logon and Logoff	✓	✓
File and Object Access	✓	✓
Use of User Rights	✓	✓
User and Group Management	✓	✓
Security Policy Changes	✓	✓
Restart, Shutdown, and System	✓	✓
Process Tracking	✓	✓

Figure 3-4: System Auditing Policy in the Windows NT User Manager.

auditing were not selected for the 1999 evaluation.

Log settings are adjusted to allow very large log files, approximately 200 megabytes. This ensures that the log files do not fill up and begin to overwrite earlier log entries. In addition, it is specified that audit logs should not be automatically cleared. Only the Administrator account has the ability to manually clear the audit logs.

It is important to note that Windows NT auditing is different and may not be as powerful as the Basic Security Module (BSM) auditing used by the Solaris victim machines in the 1998 and 1999 evaluations [35]. For example, BSM records all system calls and their arguments; Windows NT auditing does not. Therefore, developers who wish to extend their UNIX host-based intrusion detection systems to detect Windows NT attacks may not be able to reuse their detection strategies. Attack detection that relies on tracing system calls or analyzing their arguments cannot be implemented using Windows NT auditing.

3.3 Background Traffic

3.3.1 General Background Traffic

Most of the Windows NT background traffic is created using the regenerator software developed in the 1998 evaluation for UNIX machines [10]. "Exs" scripts are used to automatically emulate telnet, FTP, and web, and email connections to the Windows NT victim machine. These scripts are modified versions of the "exs" scripts used for UNIX connections in the 1998 evaluation. The telnet scripts are modified to include a set of Windows NT commands: dir, del, net, etc. In addition, the Windows NT POSIX subsystem accepts and translates basic UNIX commands during telnet sessions (Section 3.2.1 lists these commands).

The mail scripts are modified to properly close mail connections to the Windows NT Resource Kit MailSrv program. The reason for the modifications is a bug in the MailSrv program [23]. MailSrv does not recognize the key combination, [CR].[CR] (carriage return, period, carriage return), used by the original "exs" scripts to terminate a SMTP connection. As a result, the connections remain open and eventually take up 100% of the CPU cycles of the Windows NT victim machine. Most of the new "exs" scripts are modified so that they close SMTP connections with the key combination, [CR][LF].[CR][LF] (carriage return, line feed, period, carriage return, line feed). These modified scripts properly close SMTP connection to MailSrv. However, not all mail scripts were properly modified, so once or twice a day during the 1999 evaluation, MailSrv consumed 100% of the CPU cycles with hung SMTP connections, and needed to be restarted.

No special modifications are necessary for the FTP and web scripts. Some background traffic actions, such as creating Microsoft Office documents and opening email attachments, are performed manually because they cannot be easily automated with "exs" scripts.

3.3.2 AutoBrowser Web Traffic

Web browsing from the Windows NT victim machine is automated via a Javascript program, called AutoBrowser, written for the 1999 evaluation. This script emulates browsing activity by a human user. The AutoBrowser source code includes a list of 4140 web URLs extracted from the web server in the outside test bed network. The AutoBrowser program is in the Startup Group of the Windows NT victim machine so that it automatically executes at the beginning of each day of the evaluation, via the Netscape web browser. The program emulates human-user web browsing by alternating periods of idle time and browsing time. There is more idle time during the beginning and end of the day and more browsing time during the middle of the day. During browsing periods, the program randomly visits URLs in the list. The frequency at which pages are visited also depends on the time of day. Pages are browsed with greater frequency during the middle of the day than they are during the beginning and end of the day.

All time delays between user actions are determined by an exponential distribution, computed in Javascript using the following function:

```
delay = -round(mean * ln(rand[0,1])).
```

Studies have shown that this function roughly approximates the delay between TCP connections initiated by a human-user [26]. When the function is used to calculate the delay between page visits while browsing, the mean is set to 30 seconds. When the function is used to calculate the lengths of idle times between browsing sessions, the mean depends on the time of day. The mean is set to one hour for most of the day, except for the middle of the afternoon (11AM – 1PM), when the mean is set to 30 minutes, and after 4PM, when the mean is set to 2 hours. In addition, the maximum number of pages visited during a browsing session varies throughout the day, as shown in the following table:

Time of Day	8AM-9AM	9AM-11AM	11AM-1PM	1PM-3PM	3PM-4PM	4PM-
Maximum # of Pages Visited per Browsing Session	8	15	30	15	8	3

The graph in Figure 3-5 plots the web connections initiated by the AutoBrowser during week five, day one, of the 1999 evaluation. The x-axis (time of day) is divided into five-minute blocks. The height of each bar in the y-axis represents the number of web connections in each five-minute block. The graph clearly shows that the

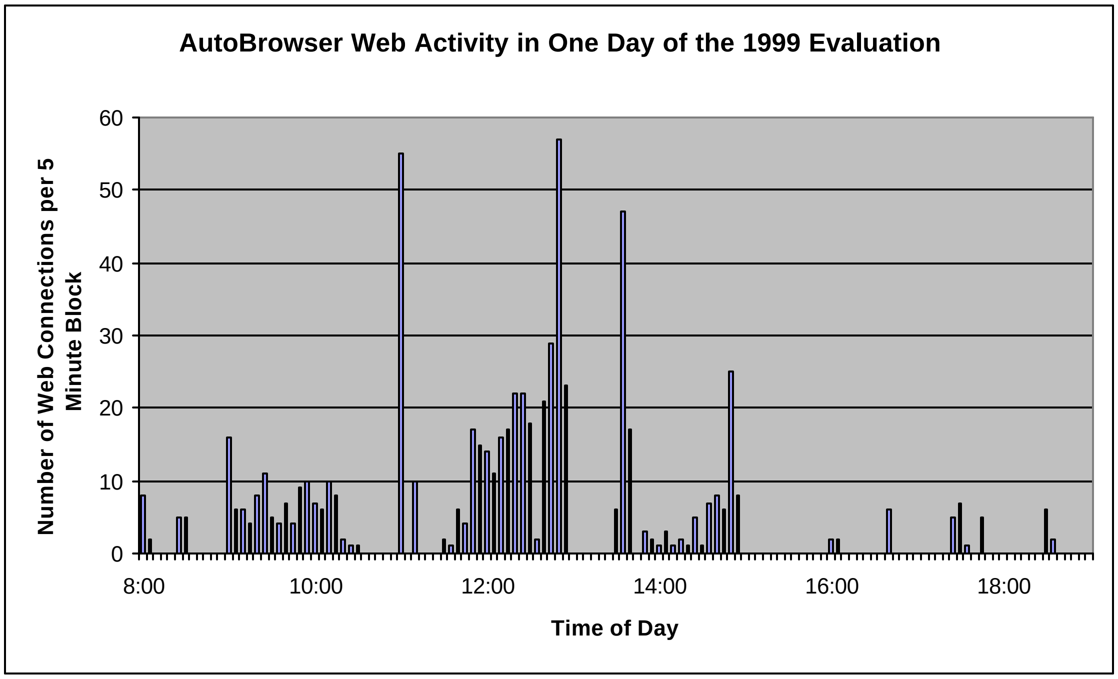
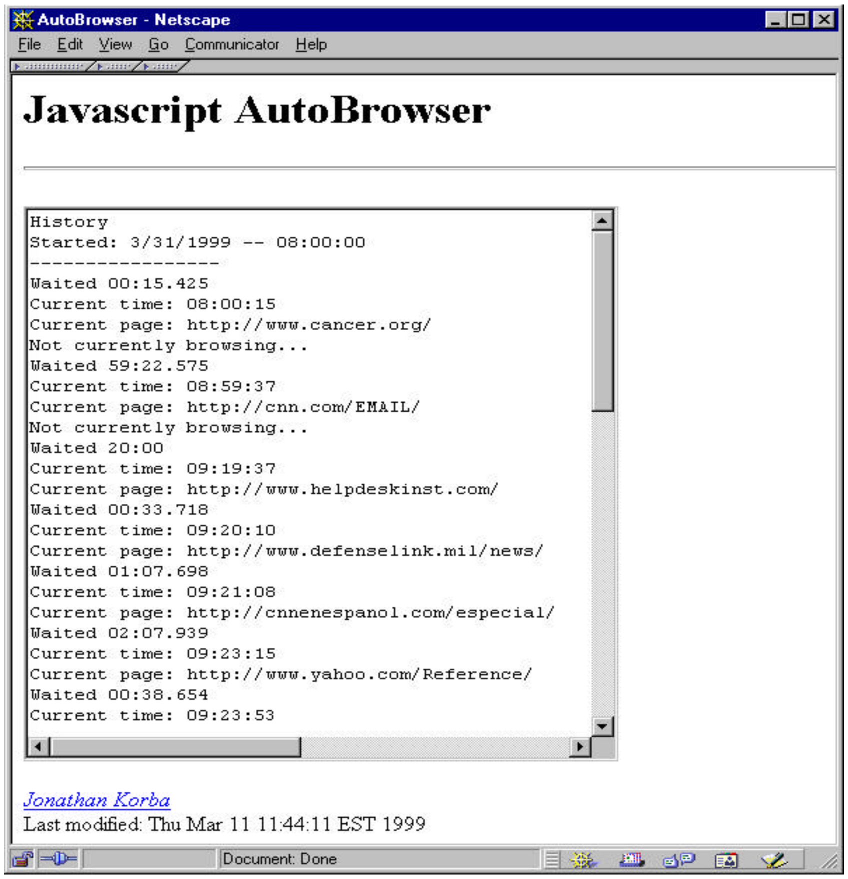


Figure 3-5: Graph of AutoBrowser Activity in One Day of the 1999 Evaluation.

AutoBrowser visits more pages during the middle of the day then during the beginning or end of the day.

To monitor web connections from the Windows NT victim machine during the evaluation, program activity is displayed in a browser window text box. The browser window containing the text box is separate from the window loading the browsed pages. Information in the text box includes durations of idle times and the URLs and access times of web page visits. An example of AutoBrowser activity recorded in the text box is shown in Figure 3-6.



```
History
Started: 3/31/1999 -- 08:00:00
------------------
Waited 00:15.425
Current time: 08:00:15
Current page: http://www.cancer.org/
Not currently browsing...
Waited 59:22.575
Current time: 08:59:37
Current page: http://cnn.com/EMAIL/
Not currently browsing...
Waited 20:00
Current time: 09:19:37
Current page: http://www.helpdeskinst.com/
Waited 00:33.718
Current time: 09:20:10
Current page: http://www.defenselink.mil/news/
Waited 01:07.698
Current time: 09:21:08
Current page: http://cnnenespanol.com/especial/
Waited 02:07.939
Current time: 09:23:15
Current page: http://www.yahoo.com/Reference/
Waited 00:38.654
Current time: 09:23:53
```

Jonathan Korba
Last modified: Thu Mar 11 11:44:11 EST 1999

Figure 3-6: AutoBrowser Activity Recorded in a Text Box.

3.4 Windows NT Input Data for Intrusion Detection Systems

The 1999 evaluation provided long listings of directory trees and full dumps of two directories, C:\WINNT\system32\LogFiles and C:\WINNT\system32\config, from the Windows NT victim machine. These files are posted at http://ideval.mit.edu/1999_index.html [11] for download by participants of the evaluation. The following sections give more details about the types of information contained in the data.

3.4.1 Long Listings of Directory Trees

Long listings of directory trees were collected at the end of each day of the evaluation. They were created with the Resource Kit POSIX command, "C:\ntreskit\posix\find / -ls\", which lists all files on the hard drive with the following information (in order from left to right):

- File index number
- File size in 512-byte blocks
- Permissions
- Number of hard links
- Owner name
- Group name
- Size in bytes
- Modification timestamp
- Filename

Figure 3-7 shows a small portion of a directory listing.

```
     5    17 drwx---rwx    1 Administ Administ     8192 Mar  4 04:32 /
105462 138121 -rwx---rwx    1 Administ Domain U 70717440 Mar  4 02:00 /www.tar
    17    53 drwxrwxrwx    1 Administ Administ    26624 Mar  3 06:10 /WINNT
  1079     2 -rwxrwxr-x    1 Administ NETWORK       707 Oct 13  1996 /WINNT/_DEFAULT.PIF
  1748    19 -rwxrwxr-x    1 Administ NETWORK      9522 Oct 14  1996 /WINNT/Zapotec.bmp
  1749    17 -rwxrwxrwx    1 Administ NETWORK      8312 Oct 14  1996 /WINNT/Zapotec 16.bmp
  2804     4 -rwxrwxrwx    1 Administ Domain U     1782 Mar  3 05:00 /WINNT/winzip32.ini
  3541     1 drwxrwxrwx    1 Administ Domain U        0 Feb 23 08:27 /WINNT/Winnt_mailspool
  1031   606 -rwx---r-x    1 Administ Administ   310032 Oct 13  1996 /WINNT/winhlp32.exe
  1083   501 -rwxrwxr-x    1 Administ NETWORK    256192 Oct 13  1996 /WINNT/WINHELP.EXE
  1082     1 -rwxrwxr-x    1 Administ NETWORK         3 Oct 13  1996 /WINNT/WINFILE.INI
  3591     1 -rwxrwxrwx    1 Administ Domain U      120 Feb 24 08:51 /WINNT/Winchat.ini
  1081     1 -rwxrwxrwx    1 Administ NETWORK       217 Mar  3 05:00 /WINNT/WIN.INI
  1032    44 -rwx---r-x    1 Administ Administ    22288 Oct 13  1996 /WINNT/welcome.exe
```

Figure 3-7: Portion of a Long Directory Listing of a Windows NT Disk.

3.4.2 Logfiles Directory Dump

All of the files located in C:\WINNT\system32\LogFiles were collected and distributed for each day of the evaluation. This directory contains a log file for each day that records all access to the IIS (i.e. web server and FTP connections). The filenames are in the format, inYYMMDD.log, based on the date when they were created. A sample log file is shown in Figure 3-8. Each line in the file contains the following information from left to right: client IP address, client username, date, time, service, host server name, server IP address, elapsed time in seconds, bytes received, bytes sent, service status code, Windows NT status code, name of operation, target of operation.

```
172.16.112.105, -, 3/31/99, 9:06:50, W3SVC, HUME, 172.16.112.100, 214188, 278, 18652, 200, 0, GET, /html/index.html
172.16.112.105, -, 3/31/99, 9:07:29, W3SVC, HUME, 172.16.112.100, 16704, 325, 11853, 200, 0, GET, /icons/worldmap.jpg
172.16.112.105, -, 3/31/99, 9:07:29, W3SVC, HUME, 172.16.112.100, 16844, 280, 1276, 200, 0, GET, /html/welcome.html
172.16.112.105, -, 3/31/99, 9:07:44, W3SVC, HUME, 172.16.112.100, 11703, 324, 622, 200, 0, GET, /html/ban.html
172.16.112.105, -, 3/31/99, 9:07:46, W3SVC, HUME, 172.16.112.100, 1484, 332, 111, 404, 2, GET, /html/assignments.html
172.16.112.105, -, 3/31/99, 9:08:06, W3SVC, HUME, 172.16.112.100, 125, 328, 445, 200, 0, GET, /html/trouble.html
172.16.112.105, -, 3/31/99, 9:08:09, W3SVC, HUME, 172.16.112.100, 3140, 278, 18652, 200, 0, GET, /html/index.html
172.16.112.105, -, 3/31/99, 9:08:10, W3SVC, HUME, 172.16.112.100, 1016, 325, 636, 200, 0, GET, /html/code.html
```

Figure 3-8: Sample Log File Produced by Windows NT Web Server Accesses.

3.4.3 Config Directory Dump

All of the files located in C:\WINNT\system32\config were collected and distributed for each day of the evaluation. This directory includes the file that stores the user database (SAM – Security Accounts Manager), files containing Windows NT Registry data (default, system, software, security), and the Windows NT event logs (AppEvent.Evt, SecEvent.Evt, SysEvent.Evt). The SAM file and Registry files are collected by executing the Resource Kit backup program, C:\ntreskit\regback.exe.

The SAM file contains an encrypted list of all user accounts and passwords. The Registry data files can be viewed by executing the Windows NT Registry editor, regedt32.exe, and opening the files with the "Load Hive" menu command. The Windows NT event logs can be viewed by using the Windows NT Event Viewer.

Chapter 4

Developing Windows NT Attacks

Several stages of work were involved for each Windows NT attack included in the 1999 evaluation. Each attack required development, analysis, and documentation. The process sometimes required modifications to the test bed environment, such as adding new software or creating new background traffic. The following list outlines the steps that were taken in developing Windows NT attacks for the evaluation:

1) Research or invent the attack.
2) Modify the attack to work in the test bed.
3) Analyze attack signatures in Windows NT audit logs and in network data.
4) Attempt to make the attack stealthy.
5) If necessary, design background traffic to make attack traffic seem less anomalous.
6) Automate the execution of the attack or define a procedure for manual execution.
7) Define a procedure to verify attack success.
8) Define a procedure to cleanup after the attack.
9) Document the attack.

4.1 Attack Research and Development

Many of the Windows NT attacks were obtained from public sources on the Internet. Web sites maintained by organizations, such as NTBugtraq [24], CERT [5], NTSecurity.net [22], ISS [9], Rootshell [29], Whitehats.com [41], and Insecure.org [8], post announcements concerning recent vulnerabilities and attacks against the Windows

NT operating system. They also archive information about older attacks. Sometimes, they provide source code that exploits known vulnerabilities and also instructions on how to execute attacks. However, even with instructions and source code, it frequently took a significant amount of work to get an attack to function properly for the evaluation. In addition, not all of the Windows NT attacks in the 1999 evaluation were derived from the Internet sources. Some of the attacks were developed specifically for the evaluation in order to test intrusion detection system performance with never-before-seen attacks.

After each attack for the evaluation was researched and downloaded from the Internet, or invented based on known Windows NT vulnerabilities, it was deployed in the test bed to ensure that it could successfully and reliably execute in the test bed environment. Some attacks required new software to be installed in the test bed. For example, the Netcat attack sent a WinZip self-extracting executable as an email attachment to the victim. In order to make it possible for the victim to unzip the file, WinZip 7.0 was installed on the Windows NT victim machine [43].

4.2 Determining Attack Signatures

Once the attack could successfully and reliably execute in the test bed environment, steps were taken to make the attack less detectable. Network traffic and Windows NT audit logs were collected and analyzed to determine what detectable signatures were left by the attack.

Tcpdump [12] was used to filter network traffic collected by the sniffer machines. The program allowed packet filtering by features such as, source address, destination address, and port number. Packet filtering made it easy to isolate the traffic created by each attack instance for analysis.

Net Tracker [40] also proved to be a useful program for analyzing attack signatures in network traffic. Net Tracker takes, as input, a tcpdump file and reassembles the data into transcripts. The transcripts are ASCII text records of what occurred during each TCP session.

To determine host-based attack signatures, Windows NT audit logs were analyzed. No filtering software was available for audit logs so the following procedure was defined to isolate the events that were logged for each attack:

1) Make sure no background traffic is running in the test bed.
2) Clear all of the audit logs on the Windows NT victim machine.
3) Launch the attack.
4) Save the audit logs.

By using this procedure, most of the events in the saved audit logs were logged as a result of the attack.

Once the signatures were defined for an attack, attempts were made to make the attack less obvious. The source code and method of execution of some attacks were modified to make the attack more stealthy. For example, the original probe attack, NTInfoscan (downloaded from the Internet), established an anonymous FTP connection to the victim machine with the password "NTInfoScan@security.check." Any intrusion detection system searching for this string will detect all NTInfoScan attacks launched using the original executable. In an effort to make the attack less detectable, the executable was modified to provide an inconspicuous anonymous FTP password, "guestaccnt@compuserve.com." Additional background traffic was also generated to make some attack actions seem less anomalous. For example, traffic generated by the AutoBrowser program masks attacks that require the victim to access web pages.

If possible, the attack was embedded in Expect and “exs” scripts so it could automatically execute in the test bed. Console attacks and attacks requiring web browsing or the opening of email attachments could not be automated. The next step was to clearly define a procedure for verifying that the attack was successful. Verification usually involves inspecting the network traffic for attack signatures.

Some of the attacks require cleanup actions before another instance of the attack can occur. Attackers and/or victim administrators can perform cleanup actions. An attacker cleans up after an attack to make detection more difficult, while an administrator cleans up to repair and re-secure the victim machine. Cleanup actions include erasing attack files, killing a process, restarting a service, or rebooting the victim machine. A powerful cleanup action that may be performed by an attacker is deleting or altering audit log data that resulted from the attack. However, as specified in the design of the 1999 evaluation, audit logs were never altered or deleted during the evaluation days. Finally, documentation was drafted to include all of the above-mentioned characteristics and procedures for each attack.

4.3 Extended Auditing

As stated in Section 3.2.3, full system auditing and base object auditing were enabled in the 1999 evaluation, but individual files and Registry keys were not audited. A Windows NT system with a different auditing policy may yield different attack signatures in the security log, or none at all. It would be useful to know all possible audit log attack signatures. To achieve this goal, a separate experiment was performed after the 1999 evaluation was completed. Each Windows NT attack was launched against the Windows NT victim machine with maximum auditing enabled (i.e. audit settings used in the 1999 evaluation plus auditing of all files and Registry keys). The data generated in the experiment was used to document, for each attack, an audit log attack signature that was as complete as possible.

Chapters six through nine of this thesis document the attacks used in the 1999 evaluation. For each attack there is a section called "Host Data for the 1999 Evaluation" that details ways in which the attack may be detected in host data generated and distributed in the evaluation. If the file and Registry auditing experiment yielded additional signatures for an attack, these signatures are noted in a separate section, called "Extended Host Data." This section also includes any attack signatures that may occur in other types of host data that were not provided in the 1999 evaluation, such as log files for individual applications or real-time file system monitoring.

Chapter 5

Assembling a Windows NT Attack Set

The Windows NT attacks in the 1999 evaluation were chosen such that, collectively, they form a realistic and relatively comprehensive set of Windows NT attacks. An attack taxonomy, originally presented in [39] and used in the 1998 evaluation [10], provided a methodology for classifying Windows NT attacks. The selection of Windows NT attacks for the 1999 evaluation was guided in part by the taxonomy. In addition, the attacks were selected so as to include both network and console based attacks, a man-in-the-middle attack, and an attack using code in a Microsoft application macro.

5.1 Overview of an Attack Taxonomy

For a given attack, the user begins with a specific level of privileges and either executes a method of transition to obtain privileges at higher level, and/or performs some action. The taxonomy provides a way to classify attacks by defining a set of privilege levels, possible methods of transition, and a set of actions. One-character strings are used to represent the privilege levels, methods of transition, and actions. A classification is assigned to each attack by assembling the one-character strings to form multi-character strings.

Possible levels of privilege include remote network access (R), user access (U), root or super-user access (S), and physical access to the host (P). A set of possible methods of transition between levels of privilege is listed below. Each method is also represented by a one-character string.

m) Masquerading: In some cases it is possible to fool a system into giving access by misrepresenting oneself. Examples of masquerading include using a stolen username/password or sending a TCP packet with a forged source address.

a) Abuse of Feature: There are legitimate actions that one can perform, or is even expected to perform, that when taken to the extreme can lead to system failure. Example include filling up a disk partition with user files or starting hundreds of telnet connections to a host to fill its process table.

b) Implementation Bug: A bug in a trusted program might allow an attack to proceed. Specific examples include buffer overflows and race conditions.

c) System Misconfiguration: An attacker can exploit errors in security policy configuration that allows the attacker to operate at a higher level of privilege than intended.

s) Social Engineering: An attacker may be able to coerce a human operator of a computer system into giving the attacker access.

A set of possible actions that an attacker can perform is shown in Figure 5-1.

The following classifications of example attacks demonstrate the application of the taxonomy. If a user with remote network access (R), exploits a bug in the web server (B) to temporarily deny service (Deny), the attack classification label is "R-b-

Category	*Specific Type*	*Description*
Probe	Probe(Machines)	Determine types and numbers of machines on a network
	Probe(Services)	Determine the services a particular system supports
	Probe(Users)	Determine the names or other information about users with accounts on a given system
Deny	Deny(Temporary)	Temporary Denial-of-Service with automatic recovery
	Deny(Administrative)	Denial of Service requiring administrative intervention
	Deny(Permanent)	Permanent alteration of a system such that a particular service is no longer available
Intercept	Intercept(Files)	Intercept files on a system
	Intercept(Network)	Intercept traffic on a network
	Intercept(Keystrokes)	Intercept keystrokes pressed by a user
Alter	Alter(Data)	Alteration of stored data or data in transit
	Alter(Communication)	Alteration of data in transit
	Alter(Intrusion-Traces)	Removal of hint of an intrusion, such as entries in log files
Use	Use(Recreational)	Use of the system for enjoyment, such as playing games or bragging on IRC
	Use(Intrusion-Related)	Use of the system as a staging area/entry point for future attacks

Figure 5-1: Summary of Possible Types of Actions.

Deny(Temporary)." If a user with an local account (U), runs a program to decrypt the password file (Use), the classification is "U-Use(Intrusion)." If a user with remote network access (R) obtains root access (S) by tricking another user (s), and then uses the new privileges to modify files (Alter), the classification is "U-s-S-Alter(Files)."

5.2 Windows NT Attack Set

Figure 5-2 lists the 12 Windows NT attacks developed for the 1999 DARPA Intrusion Detection Evaluation. The four attack categories represent groupings of the possible attack types listed in the taxonomy. These four groups are: Denial-of-Service (R-?-Deny), Remote-to-User (R-?-U), local-User-to-Super-user (U-?-S), and Probes (R-?-Probe). The following four chapters present a description of each attack category and document the individual Windows NT attacks in each category. The documentation includes descriptions of the attacks, procedures for executing, verifying, and cleaning up after the attack, and attack signatures detectable in network traffic and Windows NT host data.

Attack Category	Attack Name
Denial-Of-Service (R-Deny)	CrashIIS DoSNuke
Remote-to-User (Remote to Local) (R-?-U,S)	Framespoofer Netbus NetCat PPMacro
User-to-Super-user (User-to-Root) (U,P-?-S)	AnyPW CaseSen NTFSDOS SecHole Yaga
Probes (R-Probe)	NTInfoScan

Figure 5-2: Windows NT Attacks Developed for the 1999 DARPA Intrusion Detection Evaluation.

Chapter 6

Denial-of-Service Attacks

A denial-of-service attack prevents users from accessing the resources or services of a victim machine or network of machines. An attacker can accomplish a denial-of-service through a range of destructive actions, such as, disabling a network service, consuming large amounts of network bandwidth or CPU cycles, or completely crashing a machine. Common methods used in denial-of-service attacks include sending a specially constructed packet to a port on a victim machine, or using many packets to sustain high utilization of network or computer resources. Some of the denial-of-service attacks used in the 1998 evaluation were also used to attack the Windows NT victim in the 1999 evaluation, namely, Neptune and Smurf. These attacks are fully documented in [10]. In addition, two denial-of-service attacks, CrashIIS and DoSNuke, were developed to specifically target the Windows NT victim machine in the 1999 evaluation. CrashIIS disables the Windows NT web server and DoSNuke crashes a Windows NT victim machine. The following sections describe both attacks in detail.

6.1 CrashIIS R-b-Deny(Administrative)

Description

CrashIIS is a denial-of-service attack against the Windows NT IIS web server. The attacker sends a malformed GET request via telnet to port 80 on the Windows NT victim machine. Due to a bug in IIS, the command "GET ../.." crashes the web server and sometimes crashes the FTP and Gopher daemons as well, because they are part of IIS [22].

Test Bed Details

Execution: The attack is fully automated by wrapping an “exs” script around the Expect script, crashiis.exp. From an inside or outside UNIX attacker machine, crashiis.exp telnets to port 80 on the Windows NT victim and sends the command "GET ../..". Running "crashiis.exp <victim IP>" will crash the victim's web server (and possibly the FTP and Gopher servers as well).

Verification: After the attack has successfully completed, the IIS web server on the victim will be terminated. This can be verified on the victim machine by observing that the process, inetinfo.exe, is not longer in the Task Manager processes list. Attack success can be verified from a remote machine by typing the command "telnet <victim IP> 80" (it should no longer connect) or by using a browser to access a page on the victim web server (it should not load the page).

Cleanup: An administrator must manually restart the victim's web server via the Microsoft Internet Service Manager. Usually the FTP and Gopher services need to be restarted as well.

Detection

Network traffic: The malformed GET command string, “GET ../..” can be detected in network traffic. However, the collected traffic must be processed first, because pieces of the text string may have been sent in separate TCP packets due to the telnet protocol or packet fragmentation in the network. Net Tracker [40] (a UNIX program) takes, as input, a dump file generated by the tcpdump program. It reassembles the network traffic, and outputs the results in individual transcript files for each TCP connection. The attack occurred if a transcript file reveals the malformed GET command sent from an attacker machine to port 80 of the victim machine, as shown in Figure 6-1. The first line in the transcript specifies the source and destination of the connection. The victim IP address and port number in the figure are shown in boldface text. The second line indicates the date and time when the connection began with a SYN packet. The third line reveals the malformed GET request and the fourth line indicates the end of the connection. The fourth line of the connection would have ended with a letter "F" if the connection closed with a FIN packet. However, the connection ends abnormally because IIS crashes. Net Tracker never detects a FIN packet, so it labels the end of the transcript with the letter "C," which stands for "Continued."

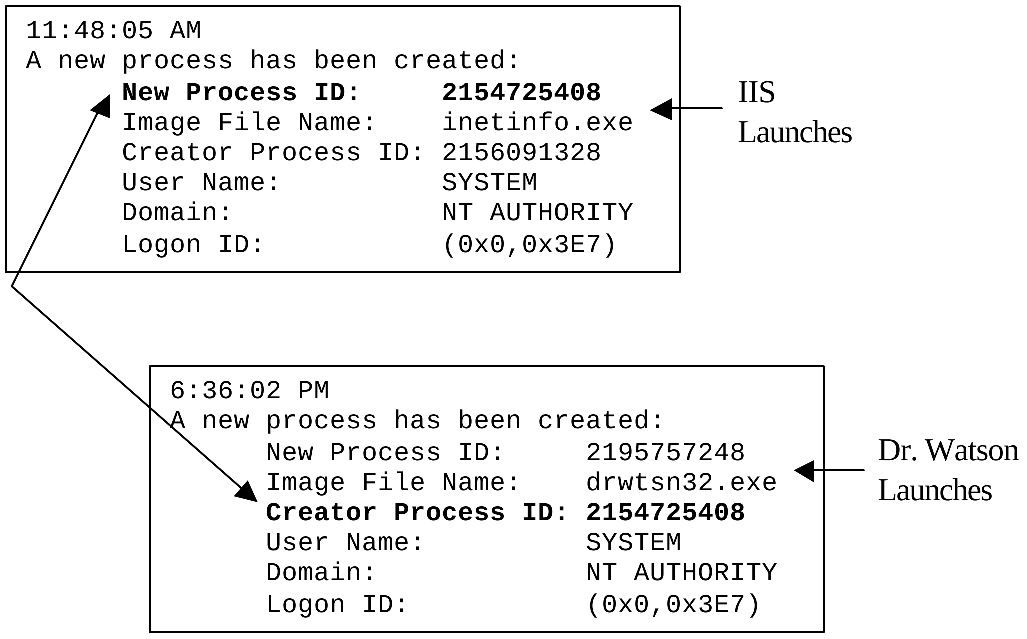
```
202.72.1.77:8756=>172.16.112.100:80        (Attack machine to port 80 of victim machine)
04/05/1999 22:36:11 S                      (Start of connection – SYN packet)
GET ../..                                  (Malformed GET command)
04/05/1999 22:36:18 C                      (End of connection – no FIN packet)
```

Figure 6-1: CrashIIS Malformed 'GET' Request Revealed in Session Transcript.

All queries to the web server will fail until the administrator of the victim machine restarts the service. These failed connections can be used to detect the effects of the attack.

Host Data from the 1999 Evaluation: When the IIS service is turned on, a process called inetinfo.exe is created and recorded in the security log. When IIS crashes, the default debugger for application errors, Dr. Watson, is launched and recorded in the security log. The CrashIIS attack can be detected in the security log by matching the

Creator Process ID number of the drwtsn32.exe process (Dr. Watson) with the Process ID number of inetinfo.exe (IIS) as shown in Figure 6-2.



```
11:48:05 AM
A new process has been created:
      New Process ID:     2154725408
      Image File Name:    inetinfo.exe
      Creator Process ID: 2156091328
      User Name:          SYSTEM
      Domain:             NT AUTHORITY
      Logon ID:           (0x0,0x3E7)
```

```
6:36:02 PM
A new process has been created:
      New Process ID:     2195757248
      Image File Name:    drwtsn32.exe
      Creator Process ID: 2154725408
      User Name:          SYSTEM
      Domain:             NT AUTHORITY
      Logon ID:           (0x0,0x3E7)
```

Figure 6-2: Dr. Watson Program Launches when IIS Crashes.

Extended Host Data: When a CrashIIS attack occurs, the Dr. Watson log file, C:\WINNT\user.dmp, on the Windows NT victim machine (not provided in the 1999 evaluation) will reveal that the IIS crashed. The log file will indicate that an error occurred in an application called "exe\inetinfo.dbg." Figure 6-3 shows a portion of a Dr. Watson log file after a CrashIIS attack, with the application name in boldface. The log entry also notes the date and time and the type of error that occurred.

```
Microsoft (R) Windows NT (TM) Version 4.00 DrWtsn32
Copyright (C) 1985-1996 Microsoft Corp. All rights reserved.

Application exception occurred:
        App: exe\inetinfo.dbg (pid=161)
        When: 3/31/1999 @ 18:36:9.906
        Exception number: c0000005 (access violation)
```

Figure 6-3: A Portion of the Information in the Dr. Watson Log File after IIS Crashes.

6.2 DoSNuke R-b-Deny(Administrative)

Description

DoSNuke is a Denial-of-Service attack that sends Out Of Band data (MSG_OOB) to port 139 (NetBIOS), crashing the Windows NT victim machine. A NetBIOS connection is established, followed by a series of packets sent with the MSG_OOB flag set. Due to a bug in the operating system, Windows NT with Service Pack 1 panics and the result is the "blue screen of death." Windows NT 4.0 with Service Pack 4 or greater is not vulnerable to the attack [21] [19].

Test Bed Details

Execution: The attack is prepared for execution on a Windows NT machine by opening the PERL script, dosnuke.pl, for editing, and setting the time of day to launch the attack. Then dosnuke.pl is executed or a shortcut to it is placed in the Windows NT Startup group for automated execution. The script takes no arguments (always targets the IP address of the Windows NT victim machine). Dosnuke.pl launches dosnuke.exe, which establishes a NetBIOS connection to the victim machine, and then sends five packets with the MSG_OOB flag set. Only one packet is necessary to crash the victim machine, but five are sent in case packets are lost.

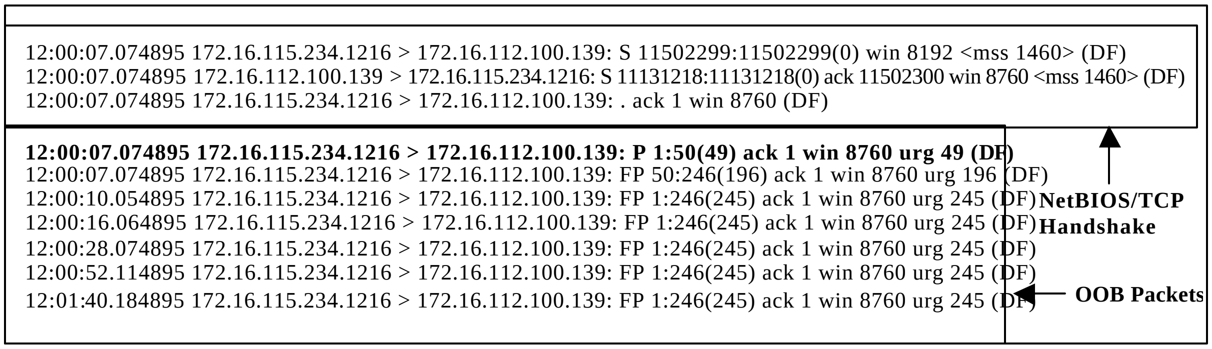
Verification: After successful completion of the attack, the victim machine will crash and display a "bluescreen of death." The success of the attack can be remotely verified by pinging the IP address of the victim machine. If the ping times out, then the attack succeeded.

Cleanup: An administrator must manually reboot the victim machine.

Detection

Network traffic: Figure 6-4 shows the network traffic created by the attack, displayed by the tcpdump program. A three-way handshake, between the attacker machine and the victim machine, establishes the TCP connection to the NetBIOS port of the victim (port 139). The packets following the handshake are marked with the TCP "urg" because TCP marks Out of Band packets as urgent. The attack can be detected by searching the network data for a NetBIOS handshake followed by a series of NetBIOS packets with the "urg" flag. The bold line in Figure 6-4 indicates the packet that crashes the machine. The following packet contains the data that could not fit in the first packet. The rest of the "urg" packets are packets resent by the TCP protocol because no acknowledgement is received from the victim machine (the victim machine is disabled). Tcpdump can be used to search for "urg" packets by executing the command:

"tcpdump –nr <network traffic dump file> 'tcp[13] & 1 != 0'"



```
12:00:07.074895 172.16.115.234.1216 > 172.16.112.100.139: S 11502299:11502299(0) win 8192 <mss 1460> (DF)
12:00:07.074895 172.16.112.100.139 > 172.16.115.234.1216: S 11131218:11131218(0) ack 11502300 win 8760 <mss 1460> (DF)
12:00:07.074895 172.16.115.234.1216 > 172.16.112.100.139: . ack 1 win 8760 (DF)

12:00:07.074895 172.16.115.234.1216 > 172.16.112.100.139: P 1:50(49) ack 1 win 8760 urg 49 (DF)
12:00:07.074895 172.16.115.234.1216 > 172.16.112.100.139: FP 50:246(196) ack 1 win 8760 urg 196 (DF)
12:00:10.054895 172.16.115.234.1216 > 172.16.112.100.139: FP 1:246(245) ack 1 win 8760 urg 245 (DF)
12:00:16.064895 172.16.115.234.1216 > 172.16.112.100.139: FP 1:246(245) ack 1 win 8760 urg 245 (DF)
12:00:28.074895 172.16.115.234.1216 > 172.16.112.100.139: FP 1:246(245) ack 1 win 8760 urg 245 (DF)
12:00:52.114895 172.16.115.234.1216 > 172.16.112.100.139: FP 1:246(245) ack 1 win 8760 urg 245 (DF)
12:01:40.184895 172.16.115.234.1216 > 172.16.112.100.139: FP 1:246(245) ack 1 win 8760 urg 245 (DF)
```

Figure 6-4: A DoSNuke Signature in Network Traffic.

The original attack downloaded from the Internet, transmitted the string "Hey, I can't help getting these nasty VXD errors!" to the victim. The attack was modified to send a blank string instead. Other versions of the attack may still send the string, which can be used in detecting the attack.

Host Data from the 1999 Evaluation: The victim's security audit log will indicate a hard reboot after the system is restarted by an administrator. The reboot will be a hard reboot (turning the machine off and then back on again) and not a soft reboot (CTRL-ALT-DEL), because a bluescreen system crash cannot be soft rebooted. A soft reboot audit signature is a "SeShutdownPrivilege" Privilege Use Event followed by an event stating, "Windows NT is starting up." A hard reboot audit signature can be detected because it does not include the "SeShutdownPrivilege" event.

A hard reboot can be used to detect but not identify the DoSNuke attack, because other attacks may also result in hard reboots (NTFSDOS, AnyPW, etc.). In addition, a hard reboot may occur in the absence of an attack (power outages, system halts, etc).

Chapter 7

Remote-to-User Attacks

A remote-to-user attack results in an attacker on a remote host obtaining unauthorized access to another computer system. An attacker who does not have an account can gain local access to the victim computer by sending packets over the network from a remote computer. The attacker may exploit a vulnerability in the victim computer or network, or use social engineering to trick an authorized user into opening a backdoor.

One remote-to-user attack, Dictionary [10], developed in the 1998 evaluation was used to attack the Windows NT victim machine in the 1999 evaluation. In addition, four Windows NT new remote-to-user attacks were developed for the 1999 evaluation: Framespoofer, NetBus, NetCat, and PPMacro. Framespoofer exploits a bug in the Netscape browser. NetBus and NetCat use trojan programs to establish back doors on the victim system. PPMacro inserts malicious macro code in a PowerPoint presentation. The following sections give detailed descriptions of the four attacks.

7.1 Framespoofer R-m-Alter(Data)

Description

The Framespoofer attack is a type of man-in-the-middle attack. It tricks the victim user into believing he or she is viewing a web page with frames on a trusted web site. In actuality, the page's main body frame is replaced with a frame created by the attacker.

The attacker presents false information in the “spoofed” frame, in an attempt to manipulate the victim user’s actions.

In the version of the attack used in the 1999 evaluation, the attacker sends a forged email, directing the victim to a web page that displays security procedures for Air Force Base computer networks. The page resides on a computer controlled by the attacker and contains what looks like a link to a page with security procedures specific to the local Eyrie Air Force Base. When the victim user clicks on the “link,” it runs a Javascript function, which brings up the trusted web site and then inserts a malicious web page, with misleading information, into the main frame. The URL displayed in the browser remains unchanged. The misleading information for this version of the attack instructs the victim to disable the local intrusion detection system on specific days. Versions of Netscape after version 4.0.8 are not vulnerable to this attack [42].

Test Bed Details

Execution: Sending the email is automated by wrapping an “exs” script around a PERL script, sendmail.pl, written for the evaluation. Sendmail.pl takes as an argument a preformatted mail message. From a UNIX attacker, the command “sendmail.pl mail.txt

ted, where mail.txt is a Javascript email message with instructions for the victim. The mail can also be sent manually from a Windows NT attacker machine. A template of the Javascript mail message is shown in Figure 7-1. The victim must manually receive the mail and click on the links.

```
<script language="javascript">

<!--
function loadchild() {
 Wtarg=window.open("[TRUSTED SITE’S PAGE WITH FRAMES]
 setTimeout("Wtarg.frames[1].location=
[ATTACK PAGE WITH MISLEADING INFORMATION]",”[# MSEC BEFORE           );
}
// -->
</script>

<body>
[TEXT INSTRUCTIONS FOR THE VICTIM]
<a href="#" onclick="loadchild()">
[URL OF TRUSTED SITE’S PAGE WITH FRAMES]</a>
</body>
```

Figure 7-1: Javascript Email for the Framespoofer Attack.

Verification: The security procedures page for the local Air Force Base will display a page and then, a few seconds later, the main frame will switch to the frame created by the attacker.

Cleanup: The browser cache on the victim machine must be cleared after executing the attack. Otherwise, the browser will load a cached page during the next execution of the attack, and no web traffic will be generated on the network.

Detection

Network Traffic: The attack can be detected in the network traffic by using Net Tracker to reassemble the web connections. Net Tracker will output transcripts of HTTP connections that occur during the attack. The connections will be from the victim machine to port 80 of the attacker machine. The attack can be detected by carefully examining the first web connection for the Javascript code shown in Figure 7-1. Variable names may vary in different versions of the attack. However, the Javascript keywords: “javascript,” “window.open,” “frames[1].location,” and “onclick” will appear in all versions of this attack. A keyword intrusion detection system can use these strings to detect the attack.

Host Data from the 1999 Evaluation: Audit logs for the 1999 evaluation reveal nothing about the attack. Auditing additional files and Registry keys does not aid in detecting the attack.

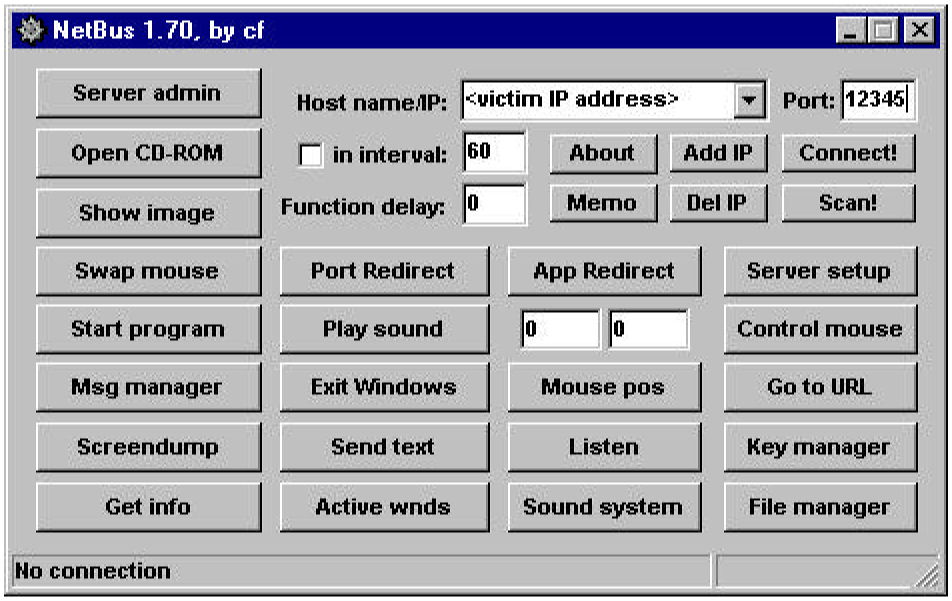
7.2 Netbus R-s-U

Description

The attacker uses a trojan program to install and run the Netbus server, version 1.7, on the victim machine. Once the Netbus server is running, it acts as a backdoor. The attacker can then remotely access the machine using the Netbus client [18].

The attacker sends an email with an executable attachment (a game called whackamole). When the victim executes the "whackamole" attachment, it launches the Netbus server (explore.exe), which is placed in C:\WINNT, and then launches the "whackamole" game. The user plays the game, not realizing that the Netbus server was installed. The attack also edits the Windows NT Registry so the Netbus server restarts at every login. This is accomplished by adding explore.exe to the "HKEY_LOCAL_ MACHINE/Software/Microsoft/Windows/Current Version/Run" Registry key.

The attacker can use the Netbus client program, shown in Figure 7-2, to manipulate files on the victim machine, download screen dumps, move the mouse pointer, etc. The attacker's access privileges are identical to the user currently logged on to the victim machine. If an administrator is using the victim, the attacker will have full administrator privileges. Through use of the "Scan!" button, the Netbus client can also be used as a probe attack to scan IP addresses for NetBus servers.



Host name/IP: <victim IP address>
Port: 12345
in interval: 60
Function delay: 0
0 0
No connection

Figure 7-2: The NetBus Client GUI.

Test Bed Details

Execution: Sending the email with the "whackamole" attachment is automated by wrapping an "exs" script around a PERL script, sendmail.pl, written for the evaluation. Sendmail.pl takes as an argument a preformatted mail message. On a UNIX attacker, the command "sendmail.pl netbus.txt <attacker@computer>" is executed, where netbus.txt is an email text message containing the "whackamole" executable attachment. The mail can also be sent manually from a Windows NT attacker machine.

The second stage of the attack is manually utilizing the backdoor. After the victim has executed the email attachment, a Windows NT attack machine is used to execute the NetBus client and connect to port 12345 of the victim machine.

Verification: After the attack has completed, the victim machine should be remotely accessible via the Netbus client running on a Windows NT attacker machine. The success of the attack can be verified in collected network traffic by the using the attack detection methods described below, in the section on Network Traffic.

Cleanup: The attacker clicks the "Server admin" button on the NetBus client and chooses "Remove server." The Registry key is removed and the server process, explore.exe, is terminated. However, the explore.exe file is not deleted from the victim's file system. For full cleanup, a victim user, usually the Administrator, must delete C:\WINNT\explore.exe.

Detection

Network Traffic: Two TCP connections are established when the NetBus server is accessed by an attacker using the NetBus client. The attacker client sends commands via a connection to port 12345 of the victim machine. The victim server transmits data in response via a connection to port 12346 of the victim machine. The attack can be detected by using the following tcpdump command to search the network traffic for connections to port 12345 or port 12346 of the victim machine:

"tcpdump –nr <network traffic dump file> port 12345 or port 12345 and host <victim IP address>"

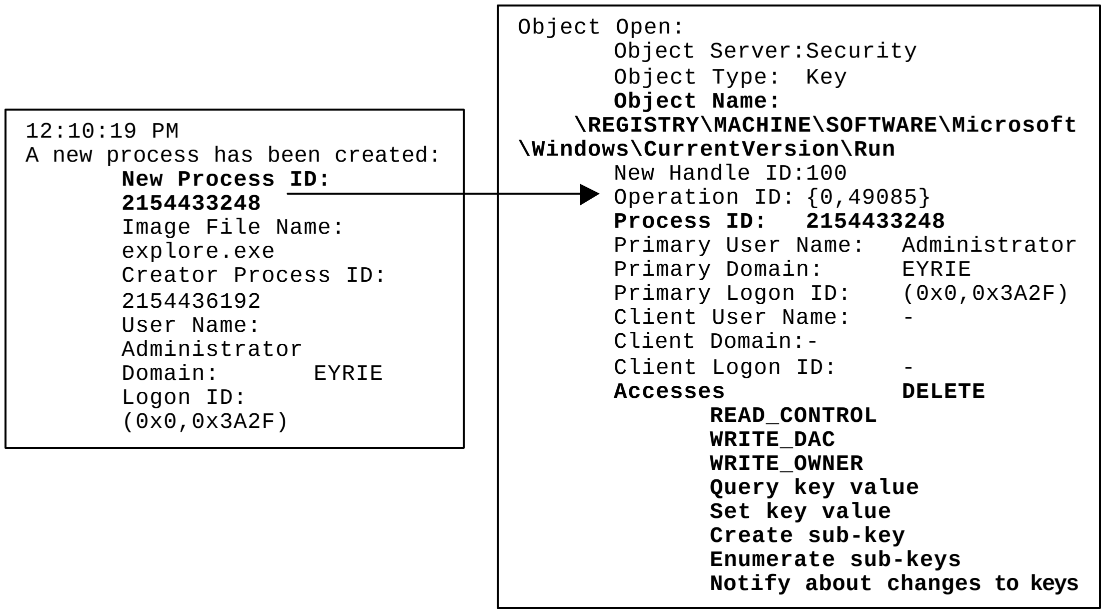
Net Tracker can be used to reassemble the network traffic into transcript files. When the attacker uses the Netbus client to access the victim, it creates network traffic that is easy to identify in the transcript files. The word "Netbus" will appear and all of the commands are in plaintext. The format of a NetBus command is: the name of the command, followed by a semicolon, followed by the arguments separated by semicolons. Figure 7-3 shows some of the strings that may appear in the Net Tracker transcript files after an instance of the Netbus attack is launched. A string-matching intrusion detection system could use these strings to detect NetBus attacks.

```
NetBus 1.6                                    Attacker Connects to Server
GetInfo
Info;Program Path: C:\TEMP\
~WZS0400.TMP\explore.exe|
Restart persistent: Yes|Login                 GetInfo Command
ID: Administrator|Clients
connected to this host: 1
CaptureScreen
CaptureReady;0                                CaptureScreen Command
CaptureReady;1;242654
StartApp;c:\winnt\system32\Calc.exe           Calc.exe executed
RemoveServer;1                                RemoveServer Command
```

Figure 7-3: Strings Revealed in Network Traffic After a NetBus Attack.

Host Data from the 1999 Evaluation: Explore.exe is the most commonly used filename for the Netbus attack. The Windows NT security log will show that explore.exe was launched when the attachment was executed.

Extended Host Data: If the "HKEY_LOCAL_MACHINE/Software/Microsoft/Windows/Current Version/Run" Registry key is audited (not audited in the 1999 evaluation), then an audit log record will indicate that the key is accessed with write privileges when explore.exe is added to it. The attack can be detected by matching the process ID of explore.exe to the process ID that opens the Registry key. Figure 7-4 shows the audit record indicating the process ID of explore.exe and the record generated when the Registry key is accessed. The process IDs, Registry key name, and access privileges are in boldface text.



```
12:10:19 PM
A new process has been created:
       New Process ID:
       2154433248
       Image File Name:
       explore.exe
       Creator Process ID:
       2154436192
       User Name:
       Administrator
       Domain:       EYRIE
       Logon ID:
       (0x0,0x3A2F)
```

```
Object Open:
       Object Server:Security
       Object Type:  Key
       Object Name:
    \REGISTRY\MACHINE\SOFTWARE\Microsoft
\Windows\CurrentVersion\Run
       New Handle ID:100
       Operation ID: {0,49085}
       Process ID:   2154433248
       Primary User Name:   Administrator
       Primary Domain:      EYRIE
       Primary Logon ID:    (0x0,0x3A2F)
       Client User Name:    -
       Client Domain:-
       Client Logon ID:     -
       Accesses             DELETE
              READ_CONTROL
              WRITE_DAC
              WRITE_OWNER
              Query key value
              Set key value
              Create sub-key
              Enumerate sub-keys
              Notify about changes to keys
```

Figure 7-4: Audit Records Show Registry Key Write Access by the Netbus Process.

L –d –p 53 –t –e cmd.exe," runs every time a user logs on to the machine. Then winlog.bat deletes all unnecessary attack files. The y2ktest folder and its contents, and C:\WINNT\system32\winlog.exe are what remain.

The attacker later uses the command "nc v <victim IP> <port>" on a remote machine (UNIX or NT with the nc program) to access the victim without a username or password.

v <victim IP> <port>" is executed to connect to the victim machine.

The files included in the self-extracting WinZip file are called winlog.bat, winlog.exe, and winlog.txt. When the WinZip file is executed, it tells the user that it puts a total of seven files into C:\y2ktest. The attack files are moved or deleted, resulting in only four files in the directory. To avoid this inconsistency, the attack batch makes three copies of one of the y2ktest files and renames them, check1, check2, and check3.

The attack modifies the Registry but does not run Netcat (winlog) right away. The backdoor does not take affect until the victim user logs out and logs in again, activating the Registry key. This makes the attack stealthier because the setup of the attack is split into two steps. NetCat can use any port, but if it uses port 23, all telnet sessions to the victim will be unauthenticated (i.e. the user will not be prompted for a username or password.)

Verification: After the attack has completed, the victim machine should be remotely accessible without authentication via the command "nc v <victim IP address> <port>." The success of the attack can also be verified by checking the victim machine's Task Manager process list for the winlog.exe process.

Cleanup: An administrator uses the Registry Editor to delete the winlog.exe command from the Registry key, deletes C:\WINNT\system32\winlog.exe, and removes winlog.exe from the process table via the Task Manager.

Detection

Network Traffic: The Net Tracker program can be used to generate a transcript of the connection from the attacker machine to port 53 of the victim machine. Figure 7-5 compares a transcript of a NetCat attack to a transcript of a normal telnet session. The NetCat session appears similar to a telnet session. However, the attack can be detected by noting that the connection is not authenticated (no request for an account name or password) and that the connection is to port 53 of the victim machine instead of telnet port 23. Figure 7-5 indicates these differences in bold text.

TRANSCRIPT OF NETCAT CONNECTION

```
206.48.44.18:1229=>172.16.112.100:53
03/31/1999 16:11:08 S
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.
C:\WINNT\Profiles\Administrator\Desktop>dir
 Volume in drive C has no label.
 Volume Serial Number is 4816-2A08
 Directory of C:\WINNT\Profiles\Administrator\Desktop
03/29/99  11:53a        <DIR>          .
03/29/99  11:53a        <DIR>          ..
02/08/99  09:32a        <DIR>          My Briefcase
03/29/99  07:33a                   430 RealPlayer.lnk
03/09/99  08:03a                   361 WinAt.lnk
03/17/99  10:20a                   434 WinZip.lnk
               6 File(s)          1,225 bytes
                          2,193,192,448 bytes free
C:\WINNT\Profiles\Administrator\Desktop>path
PATH=C:\Perl\bin;C:\WINNT\system32;C:\WINNT;C:
TRESKIT;C:\NTRESKIT\Perl
C:\WINNT\Profiles\Administrator\Desktop>vol
 Volume in drive C has no label.
 Volume Serial Number is 4816-2A08
C:\WINNT\Profiles\Administrator\Desktop>
03/31/1999 16:11:40 F
```

TRANSCRIPT OF NORMAL TELNET CONNECTION

```
135.8.60.182:5203=>172.16.112.100:23
03/30/1999 19:44:56 S
This copy of the Ataman TCP Remote Logon Services is
registered as licensed to:
Eyrie Air Force Base
Welcome to Eyrie Air Force Base
"Mundus Vult Decipi"
***** WARNING *****
This is an unsecured, declassified, publically
accessible, network computer cluster.
Account Name: orionc
Password:
Microsoft(R) Windows NT(TM)(C) Copyright 1985-1996
Microsoft Corp
d:\home>ver
Windows NT Version 4.0
d:\home>vol
Volume in drive D has no label
Volume Serial Number is B4F8-0D40
d:\home>exft
The name specified is not recognized as aninternal or
external command, operable program or batch file.
d:\home>exit
03/30/1999 19:54:10 F
```

Figure 7-5: NetCat Transcript Differs from Normal Telnet Session.

Host Data from the 1999 Evaluation: The security audit log will contain events indicating the execution of REGEDIT (the trojan edits the Registry), later followed by the execution of winlog.exe (the backdoor is setup).

Extended Host Data: If the “HKEY_LOCAL_MACHINE/Software/Microsoft/Windows/CurrentVersion/Run” Registry key is audited (not audited in the 1999 evaluation), then an audit log record will indicate that the key is accessed with full read and write privileges. The attack can be detected by looking for this audit log record, shown in Figure 7-6 with the key name and privileges in bold text.

```
Object Open:
        Object Server:      Security
        Object Type:        Key
        Object Name:
       \REGISTRY\MACHINE\SOFTWARE\Microsoft\
Windows\CurrentVersion\Run
        New Handle ID:      84
        Operation ID:       {0,32669}
        Process ID:         2154688544
        Primary User Name:  Administrator
        Primary Domain:     EYRIE
        Primary Logon ID:   (0x0,0x2565)
        Client User Name:   -
        Client Domain:      -
        Client Logon ID:    -
        Accesses            DELETE
                READ_CONTROL
                WRITE_DAC
                WRITE_OWNER
                Query key value
                Set key value
                Create sub-key
                Enumerate sub-keys
                Notify about changes to keys
                Create Link

        Privileges          -
```

Figure 7-6: Audit Record Shows Modification of the “Run” Registry Key.

7.4 PPMacro R-s-U

Description

This PPMacro attack uses a trojan PowerPoint macro to access secret files on the victim machine. This attack is based on a particular scenario, where the victim user periodically receives a PowerPoint template from a trusted outside source, via an email attachment. The victim opens the template and runs a built-in macro that inserts a graph displaying web statistics. The victim then saves the presentation and posts it on the web.

The attacker, who has knowledge of this scenario and a copy of the template, writes a fake email to the victim and attaches the template with additional attack code appended to the macro. This attack code reads a secret file from the victim machine (from d:\home\secret\) and inserts it as small font, white text in the background of the master slide of the presentation. When the presentation is posted on the web, the attacker downloads it and examines the PowerPoint file to reveal the text of the secret file. The macro also stores a counter variable in the victim machine's Registry, so that each time the victim user runs the macro, a different file from the secret directory is inserted into the presentation. The counter value is stored in HKEY_CURRENT_USER\Software\VB and VBA Program Seetings\webstats\info\idx.

Test Bed Details

Execution: Sending the email with the PowerPoint template attachment is automated by wrapping an "exs" script around a PERL script, sendmail.pl, written for the evaluation. Sendmail.pl takes as an argument a preformatted mail message. From a UNIX attacker, the command, "sendmail.pl ppatack.txt <attacker@computer>," is executed, where ppatack.txt is an email text message containing the template attachment. The mail can also be sent manually from a Windows NT attacker machine.

The victim must then execute a program, called Wusage [2], which gathers web server statistics and generates graphs. The victim renames one of the graphs in C:\WINNT\reports to graph.gif, opens the PowerPoint template, executes the embedded macro, and posts the PowerPoint file on the web server by saving it in C:\inetpub\wwwroot. The attacker later uses a browser to download the PowerPoint file.

Verification: After the attack has completed, the attacker should be able to view the secret file by downloading the PowerPoint file from the web. Net Tracker can be used to create a transcript file for the HTTP session. The attack is successful if the transcript file contains the text of the secret file.

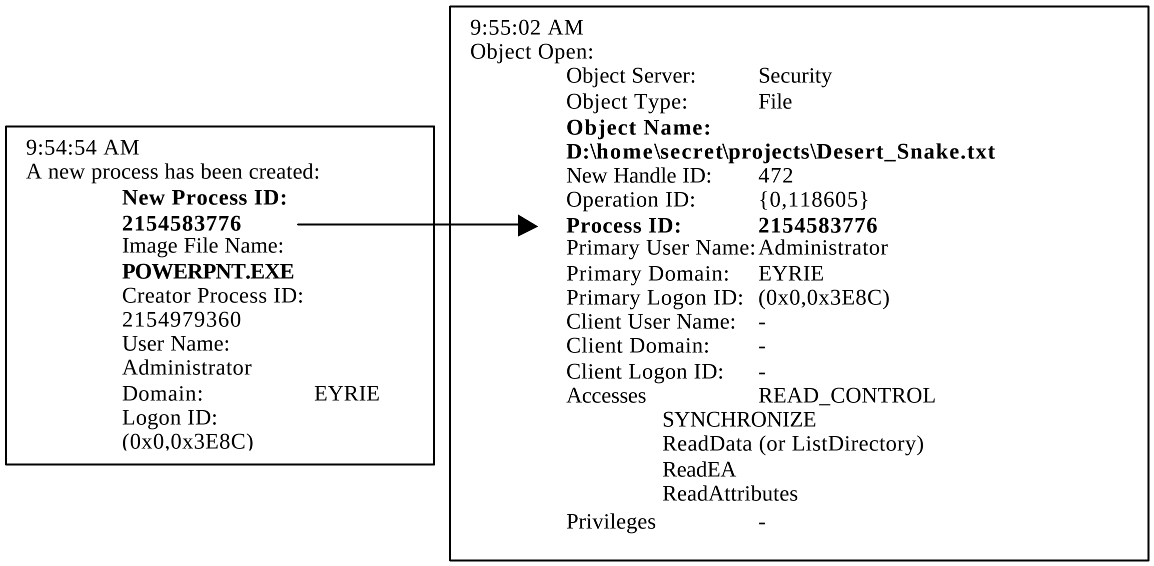
Cleanup: The Administrator should delete the PowerPoint template file and the Registry key from the victim's Registry.

Detection

Network Traffic: The attack can be detected by using Net Tracker to reassemble the HTTP session into a transcript file, and searching the file for the text of the secret file. However, the attacker can modify the macro to encrypt the secret file, thereby making the attack stealthier.

Host Data from the 1999 Evaluation: Auditing for the 1999 evaluation reveals nothing about the attack.

Extended Host Data: If the secret files, "D:\home\secret\*," are audited (not audited in the 1999 evaluation), then an audit log record will indicate that a secret file is accessed by the Powerpoint application. As shown in Figure 7-7, the attack can be detected by matching the Powerpoint process ID to the process ID that accesses the secret file.



```
9:54:54 AM
A new process has been created:
          New Process ID:
          2154583776
          Image File Name:
          POWERPNT.EXE
          Creator Process ID:
          2154979360
          User Name:
          Administrator
          Domain:              EYRIE
          Logon ID:
          (0x0,0x3E8C)
```

```
9:55:02 AM
Object Open:
          Object Server:       Security
          Object Type:         File
          Object Name:
          D:\home\secret\projects\Desert_Snake.txt
          New Handle ID:       472
          Operation ID:        {0,118605}
          Process ID:          2154583776
          Primary User Name:   Administrator
          Primary Domain:      EYRIE
          Primary Logon ID:    (0x0,0x3E8C)
          Client User Name:    -
          Client Domain:       -
          Client Logon ID:     -
          Accesses             READ_CONTROL
                    SYNCHRONIZE
                    ReadData (or ListDirectory)
                    ReadEA
                    ReadAttributes
          Privileges           -
```

Figure 7-7: Audit Records Show Secret File Access by the PowerPoint Program.

Chapter 8

User-to-Root Attacks

A user-to-root attack is used by an attacker to gain unauthorized administrator privileges on a machine. The attacker, who initially has an account with user level privileges, can exploit a vulnerability in the system and obtain root access. Five Windows NT user-to-root attacks were developed for the 1999 evaluation: AnyPW, CaseSen, SecHole, NTFSDOS, and Yaga. AnyPW and NTFSDOS are console-based attacks requiring physical access to the machine. CaseSen, SecHole, and Yaga exploit bugs in the operating system via FTP and telnet sessions. The following sections provide detailed descriptions of all five attacks.

8.1 AnyPW U-b-S

Description

AnyPW is a console user-to-root attack that allows the attacker to logon to the system without a password. A bootable floppy disk is used to modify the Windows NT MSV1_0 authentication package so that a valid username can login with any password string. Logins via telnet will also work with any password.

Test Bed Details

Execution: The attacker inserts, into the victim machine, a bootable floppy disk containing the attack, and reboots the machine. A hexadecimal number appears in the upper left of the screen. The numbers slowly increment as the attack searches for the signature of the MSV1_0 authentication package. When an asterisk appears beside the number, the package has been modified. The attacker removes the diskette and reboots the machine. Later, the attacker telnets to the victim machine as Administrator and enters any password to logon.

Note: If the attacker physically logs on to the machine with a random password string and then locks the machine, only the password that was used to logon can unlock the machine.

Verification: Any password will be accepted with a valid username.

Cleanup: The administrator of the victim machines must replace the file C:\WINNT\system32\msv1_0.dll with an uncorrupted copy.

Detection

Network Traffic: The attack cannot be detected in network traffic. Even if the attacker remotely accesses the victim machine with an incorrect password, there is no way to determine if the password is indeed incorrect.

Host Data from the 1999 Evaluation: The victim's security audit log will indicate a reboot after the system is restarted by the attacker. Most likely, the attacker had to hard reboot the machine (physically press the reset button or power cycle the machine) because he or she did not have a password to login or unlock the machine. A soft reboot audit signature is a “SeShutdownPrivilege” Privilege Use Event followed by an event stating, “Windows NT is starting up.” A hard reboot audit signature can be detected because it does not include the “SeShutdownPrivilege” event.

A hard reboot can be used to detect but not identify the AnyPW attack, because other attacks may also result in hard reboots (DoSNuke, NTFSDOS, etc.). In addition, a hard reboot may occur in the absence of an attack (power outages, system halts, etc).

Extended Host Data: The AnyPW attack can be detected by using a software tool to monitor modifications of the file, C:\WINNT\system32\msv1_0.dll. TripWire is an example of such a tool [37].

8.2 CaseSen U-b-S

Description

The CaseSen attack exploits the case sensitivity of the Windows NT object directory. All users have write permissions to the "\??" object directory. These are the default permissions so that users are able to map network drives or alias directories to new drive letters. Each drive has an entry in the "\??" object directory. Each entry is actually a symbolic link which points to the device associated with that drive. For example, the symbolic link, "\??\C:," points to the device, "\Device\HardDisk0\Partition1." It is possible to create another version of this symbolic link, "\??\c:," using lower case "c." By doing this, all the requests to drive "C" get routed through the new symbolic link. For example, copying the contents of symbolic link, "\??\D:," to the new symbolic link, "\??\c:," and then executing the "dir" command on drive "C" will display the directory listing for drive "D." By exploiting this feature, it is possible to trick the operating system into running an attack executable with the privileges of a system executable [4].

The CaseSen attack uploads to the victim three files via FTP: soundedt.exe, editwavs.exe, and psxss.exe. The files are uploaded to C:\inetpub\ftproot. The attack then telnets to the victim and executes soundedt.exe. Soundedt.exe copies editwavs.exe and psxss.exe to C:\inetpub\ftproot\WINNT\system32. It also copies all the POSIX subsystem binaries and required DLLs (except PSXSS.EXE) from C:\WINNT\system32 to C:\inetpub\ftproot\WINNT\system32. Then soundedt.exe creates a new object in the object directory, labeled "\??\c." It links to C:\inetpub\ftproot and supercedes "\??\C," which links to drive "C." Soundedt.exe starts a POSIX application by executing "POSIX

/c editwavs.exe." The Windows NT Session Manager (smss.exe) activates the POSIX subsystem, which loads the attack copy of psxss.ese. Psxss.exe inherits the security context privileges of smss.exe and adds the current user to the Administrators user group [22].

Test Bed Details

Execution: There are two stages to the attack: a setup and a break-in. The setup stage adds the attacker username to the victim machine's Administrator group. The break-in stage connects to the victim machine with the new administrator privileges. Both stages are fully automated by wrapping "exs" scripts around the Expect scripts case_s.exp and case_b.exp. Case_s.exp uploads the attack files, telnets to the victim, and launches the attack. It also deletes the three attack files after they have been used. Case_b.exp (the break-in script) telnets to the victim with the new administrator privileges, executes some generic commands ("dir", "ver", etc.), and cleans up by removing the user from the Administrators group and deleting files generated by the attack.

To prepare for the attack, the attacker places the three attack files in "/home/<user>" of a UNIX attacker machine, where <user> is the username of the attacker. The attacker executes case_s.exp by typing "case_s.exp <victim IP> <user> <password>." Later, the attacker executes "case_b.exp <victim IP> <user>

Verification: After case_s.exp is executed, the username specified in the command line of the attack should appear in the Administrators group of the victim machine (check the User Manager). After case_b.runs, the username should no longer be in the Administrators group.

The collected network traffic data can also be used to verify the attack. Net Tracker can be used to create a transcript of the telnet sessions. The transcript of the break-in telnet session should contain the line "command completed successfully." This indicates that the command to remove the user from the Administrators group was successful, which implies that the entire attack was successful.

Cleanup: No manual cleanup is necessary. The setup script deletes the three attack files. The break-in script removes the user from the Administrators group and deletes the directory, C:\inetpub\ftproot\WINNT\, which is created during the attack setup.

The attack results in some system instability. Usually the victim machine must be rebooted before the attack can be launched a second time. The attack cannot be launched more than two twice without rebooting the victim.

Detection

Network Traffic: The attack can be detected by using Net Tracker to create transcripts of the FTP and telnet sessions. Searching the FTP transcript for the strings "psxss.exe,"
xe," will reveal the transfer of the three attack files in this version of the attack. Searching the telnet transcript for the string "soundedt.exe" will reveal the execution of that file. However, editwavs.exe and soundedt.exe were names chosen specifically for the simulation. The original version of the attack, available on the Internet [22], uses the filenames, dummyapp.exe and besysadm.exe, respectively. The filename, psxss.exe, cannot be changed in different instances of the attack.

Host Data from the 1999 Evaluation: The security log shows the execution of the files posix.exe and psxss.exe, whose filenames will not differ in other versions of the attack. In addition, a log entry will state that a username is added to the Administrators group by

"NT AUTHORITY/SYSTEM." This is because the username is added via an application (very uncommon). Normally, the Administrator would use the User Manager program, Usrmgr, to add the user to a group. The corresponding log entry would indicate that the user was added by "Administrator," not "NT AUTHORITY/SYSTEM."

Figure 8-1 compares two audit log entries. The first entry was created when a CaseSen attack added a user to the Administrators group of the victim machine. The second entry was created when the Administrator of the victim machine added the same user to the Administrators group via the User Manager. The Caller User Names in the two audit log entries differ as indicated by the boldface text.

```
CASESEN ATTACK ADDS USER TO ADMINISTRATORS GROUP

Local Group Member Added:
          Member:  S-1-5-21-742865521-1025978620-313593124-1040
          Target Account Name:          Administrators
          Target Domain:                Builtin
          Target Account ID:            S-1-5-32-544
          Caller User Name:             SYSTEM
          Caller Domain:                NT AUTHORITY
          Caller Logon ID:              (0x0,0x3E7)
          Privileges:-
```

```
ADMINISTRATOR ADDS USER TO ADMINISTRATORS GROUP

Local Group Member Added:
          Member:  S-1-5-21-742865521-1025978620-313593124-1040
          Target Account Name:          Administrators
          Target Domain:                Builtin
          Target Account ID:            S-1-5-32-544
          Caller User Name:             Administrator
          Caller Domain:                EYRIE
          Caller Logon ID:              (0x0,0x3AAB)
          Privileges:-
```

Figure 8-1: A User is Added to the Administrators Group by SYSTEM in a Casesen Attack.

8.3 NTFSDOS U-b-S

Description

This console-based attack reboots the system from a floppy disk containing the program, NTFSDOS.EXE. This executable is able to recognize and mount NTFS drives. It makes them appear indistinguishable from standard FAT drives, giving the attacker the ability to read and copy files that would otherwise be protected by Windows NTFS security. The attacker does not need to be an authorized user of the victim machine. However, the attack is considered to be a user-to-root attack because physical access to the machine is required to initiate the attack [36].

Test Bed Details

Execution: The attack is completely manual. The attacker inserts the diskette (a bootable floppy disk containing the ntfsdos.exe program) into the "A" drive of the victim machine, and pushes the reset button on the CPU. After the system reboots, the attacker types "ntfsdos" at the DOS prompt. He or she then changes directories to "C:\secret," copies the secret files to the diskette, removes the diskette, and reboots the machine.

Verification: The secret files will be stored on the diskette.

Cleanup: No cleanup is necessary.

Detection

Network Traffic: The attack does not create network traffic.

Host Data from the 1999 Evaluation: The victim's security audit log will indicate a reboot after the system is restarted by the attacker. Most likely, the attacker had to hard reboot the machine (physically press the reset button or power cycle the machine) because he or she did not have a password to login or unlock the machine. A soft reboot audit signature is a “SeShutdownPrivilege” Privilege Use Event followed by an event stating, “Windows NT is starting up.” A hard reboot audit signature can be detected because it does not include the “SeShutdownPrivilege” event.

A hard reboot can be used to detect but not identify the NTFSDOS attack, because other attacks may also result in hard reboots (DoSNuke, AnyPW, etc.). In addition, a hard reboot may occur in the absence of an attack (power outages, system halts, etc).

8.4 SecHole U-b-S

Description

The attacker, a malicious user, establishes an FTP connection to the victim and uploads the files test.exe and testfile.dll (filenames were chosen to be stealthy). The attacker then telnets to the victim and executes test.exe. As a result, the attacker is added to the Administrators group.

Test.exe locates the memory address of a particular API function (OpenProcess) and modifies the instructions at that address. This is possible because the function is in the user space of test.exe. The function is modified so that it returns a success response, instead of a failure response, when it is asked to open a process to which it does not have privileges. Test.exe then calls DebugActiveProcess with the RPCSS system process (Remote Procedure Call Service) as an argument. Before granting test.exe debug access to the RPCSS process, DebugActiveProcess calls OpenProcess to check for privileges. The request is successful because of the modifications made to OpenProcess. Once test.exe has debug access to the RPCSS process, it exploits the system process privileges to add the attacker username to the local Administrators group [32][33]. The user later telnets to the victim machine with administrator privileges.

Test Bed Details

Execution: There are two stages to the attack: a setup and a break-in. The setup stage adds the user to the victim machine's Administrator group. The break-in stage connects to the victim machine with the new administrator privileges. Both stages are fully automated by wrapping "exs" scripts around the Expect scripts sec_s.exp and sec_b.exp.

Sec_s.exp uploads the attack files, telnets to the victim, and executes the attack. It deletes the two attack files after they have been used. Sec_b.exp (the break-in script) telnets to the victim with the new Administrator privileges, executes some generic commands, and cleans up by removing the user from the Administrators user group and deleting files generated by the attack.

The attacker prepares the attack by placing test.exe and testfile.dll in /home/<user> of an UNIX attacker machine, where <user> is the username of the attacker. Sec_s.exp is executed by typing “sec_s.exp <victim IP> <user> <password>.” Later, the attacker executes "sec_b.exp <victim IP> <user> <password>,” to connect to the machine with administrator privileges.

Verification: After sec_s.exp is launched, the username specified in the command line of the attack should appear in the Administrators user group on the victim machine (check the User Manager). After sec_b.exp executes, the username should no longer be in the Administrators user group.

The collected network traffic data can also be used to verify the attack. Net Tracker can be used to create a transcript of the telnet sessions. The transcript of the break-in telnet session should contain the line "command completed successfully." This indicates that the command to remove the user from the Administrators group was successful, which implies that the entire attack was successful.

Cleanup: The setup script deletes the two attack files. The break-in script removes the user from the Administrators group. The attack may result in system instability. It is unlikely, but the victim system may lock up after the attack. If this happens, the victim user must reboot the machine. The attack still succeeds.

Detection

Network Traffic: The attack can be detected by using Net Tracker to create transcripts of the FTP and telnet sessions. Searching the FTP transcript for the strings “test.exe” and “testfile.dll,” will reveal the transfer of the two attack files in this version of the attack. Searching the telnet transcript for the string “test.exe” will reveal the execution of that file. However, filenames were chosen specifically for the simulation. The original version of the attack uses the filenames, sechole.exe and admindll.exe.

Host Data from the 1999 Evaluation: After a SecHole attack, the Windows NT security log will contain a log entry indicating the execution of the file, test.exe (sechole.exe). In addition, a log entry will show that a username was added to the Administrators user group by “NT AUTHORITY/SYSTEM.” This is because the username is added via an application (very uncommon). Normally, the Administrator would use the User Manager program, Usrmgr, to add the user to a group. The corresponding bg entry would indicate that the user was added by “Administrator,” not “NT AUTHORITY/SYSTEM.”

8.5 Yaga U-b-S

Description

The Yaga attack (Yet Another Get Admin) edits the victim's Registry so that the next time a service crash occurs on the victim machine, the attacker is added to the Domain Admins group. To setup the attack, the attacker must upload to the victim machine a file with Registry key information and then use it to edit the Registry. This is accomplished via a telnet session. The Registry key originally contains a value indicating that the Dr. Watson debugger program (drwtsn32.exe) should execute when an application error occurs (e.g. a service crashes). The Yaga attack modifies the key value so that the drwtsn32.exe command is replaced with a command that adds the attacker username to the Domain Admins user group. Once the setup is complete, the attacker uses a denial-of-service attack, CrashIIS, to remotely crash a service on the victim machine. As a result, the attacker username is added to the Domain Admins user group.

Test Bed Details

Execution: The attack is fully automated by wrapping a "exs" scripts around the Expect scripts, yaga_s.exp and yaga_b.exp. The Expect setup script, yaga_s.exp, establishes a telnet connection with the Windows NT victim computer. It uses the "cat" command to create the file, "entry," with Registry key information and then edits the Registry key, HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDe-bug," so that the Dr. Watson command, "drwtsn32 –p %ld –e %ld -g," is replaced with the command, "net group "Doman Admins" <attacker username> /ADD." The attack then executes the CrashIIS attack to crash the IIS web server. As a result, the Registry key is accessed, the net.exe command is executed, and the attacker username is added to the Domain Admins group. The web server remains disabled.

The break-in expect script, yaga_b.exp, telnets to the victim machine with the new Domain Admin permissions, executes some generic commands, and cleans up by removing the user from the Domain Admins group and restoring the original Registry key.

Verification: After yaga_s is launched, its success can be verified by accessing the User Manager on the victim machine to verify that the attacker username is in the Domain Admins group. After yaga_b executes, the username should no longer be in the Domain Admins group.

The collected network data can also be used to verify the attack. Examining transcripts created by Net Tracker will reveal the line "command completed successfully." This indicates that the command to remove the user from the Domain Admins group was successful, which implies that the entire attack was successful.

Cleanup: The break-in script removes the attacker username from the Domain Admins group and restores the original AeDebug Registry key. The Administrator must manually restart the IIS service(s).

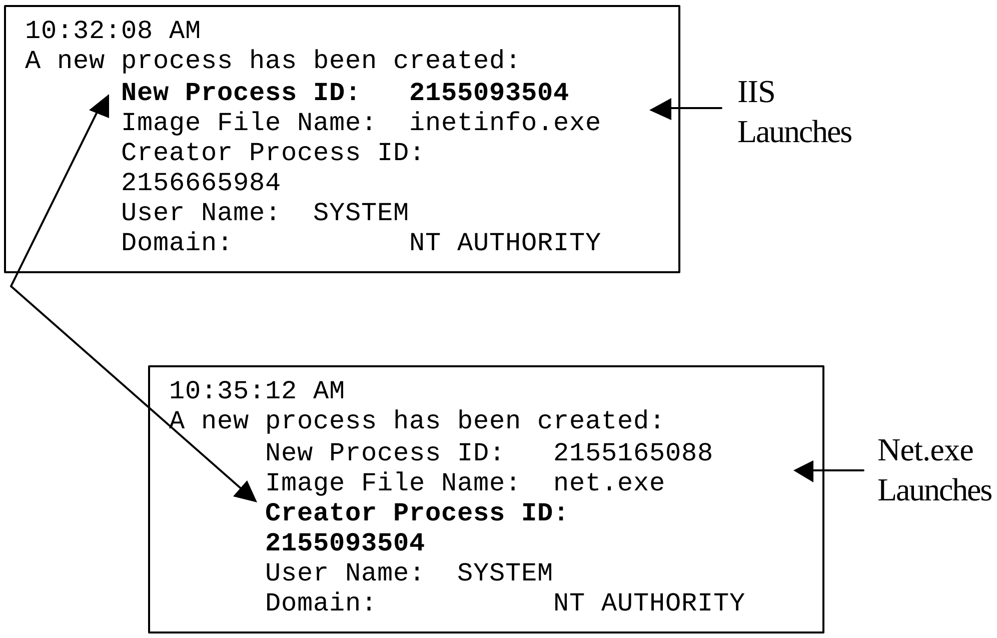
Detection

Network Traffic: Net Tracker can be used to reassemble the collected network traffic in TCP transcripts. These transcripts can be examined for attack keywords. The creation of the file, “entry,” containing the Registry information, is done with the “cat” command. As a result, the TCP transcripts will reveal the text strings of the file, shown in Figure 8-2. The text strings can be used by a string-matching intrusion detection system to detect the attack. A good string to search for is “Aedebug.” In addition, the transcripts will show that regedit.exe was run by the attacker.

```
REGEDIT4
 [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\AeDebug]
"UserDebuggerHotKey"=dword:00000000"
"Debugger"="net group \"Domain Admins\" <attacker username /ADD"
"Auto"="1"
```

Figure 8-2: The Yaga Attack Creates a Text File Containing Registry Information.

Host Data from the 1999 Evaluation: Similar to the way a CrashIIS attack can be detected, a Yaga attack can be detected in the Windows NT security log by observing that a command is executed by the inetinfo.exe service when the service crashes. In this case, the process ID of the inetinfo.exe process will match the process ID that launches the net.exe command, as shown in Figure 8-3.



```
10:32:08 AM
A new process has been created:
      New Process ID:    2155093504
      Image File Name:   inetinfo.exe
      Creator Process ID:
      2156665984
      User Name:  SYSTEM
      Domain:           NT AUTHORITY
```

```
10:35:12 AM
A new process has been created:
      New Process ID:    2155165088
      Image File Name:   net.exe
      Creator Process ID:
      2155093504
      User Name:  SYSTEM
      Domain:           NT AUTHORITY
```

Figure 8-3: Net.exe Program Launches when IIS Crashes in a Yaga Attack.

In addition, an entry in the security log will indicate that the attacker username was added to the Administrators group by "NT AUTHORITY/SYSTEM." This is because the user is added via an application (very uncommon). Normally, the Administrator would use the User Manager program, Usrmgr, to add the user to a group. The corresponding log entry would indicate that the user was added by "Administrator,"

Extended Host Data: If the "HKEY_LOCAL_MACHINE/Software/Microsoft/Windows NT/CurrentVersion/AeDebug" Registry key is audited (not audited in the 1999 evaluation), then an audit log record will indicate that the key is accessed with privileges to modify key values. The attack can be detected by looking for this audit log record log, shown in Figure 8-4 with the key name and "set key value" privilege in bold text.

```
Object Open:
            Object Server:      Security
            Object Type:        Key
            Object Name:
           \REGISTRY\MACHINE\SOFTWARE\Microsoft\
Windows NT\CurrentVersion\AeDebug
            New Handle ID:      16
            Operation ID:       {0,38791}
            Process ID:         2154622176
            Primary User Name:  alie
            Primary Domain:     EYRIE
            Primary Logon ID:   (0x0,0x960D)
            Client User Name:   -
            Client Domain:      -
            Client Logon ID:    -
            Accesses            READ_CONTROL
                        Query key value
                        Set key value
                        Create sub-key
                        Enumerate sub-keys
                        Notify about changes to keys

            Privileges          -
```

Figure 8-4: Audit Record Shows Modification of the "AeDebug" Registry Key.

Chapter 9

Probes

A probe or scan attack is used by an attacker as a reconnaissance tool. A probe may search a network for valid IP addresses, scan a single computer for active ports, or gather information about a computer's configuration, operating system, or TCP services. Information obtained with a probe attack may reveal vulnerabilities that the attacker can exploit in later attacks. Some probe attacks developed in the 1998 evaluation were also used against the Windows NT victim in the 1999 evaluation, namely, Ipsweep and Nmap (portscan). These attacks are fully documented in [10]. In addition, one Windows NT probe attack, NTInfoScan, was developed for the 1999 evaluation.

9.1 NTInfoScan R-a-Probe(Services/Known Vulnerabilities)

Description

NTInfoScan (Version 4.2.1) is a security scanner tool that administrators can used to test Windows NT systems for security holes. However, an attacker can use the same tool to scan a Windows NT victim machine and obtain share information, the names of all the users, services running, and vulnerabilities in the system configuration. The results are saved in an HTML file, named <victim>.html, where <victim> is the IP address of the victim machine [25].

Test Bed Details

Execution: To execute the attack, a PERL script, ntis.pl, runs on a Windows NT attacker machine. The attack is prepared by editing the first line of ntis.pl with the time of day the attack will run. Ntis.pl is then executed or placed it in the Startup group for automated execution. Ntis.pl takes no input arguments and automatically scans the Windows NT victim machine. The attack may take up to 20 min. to complete.

Ntis performs many tests of the victim machine. It attempts anonymous FTP interaction, tests for many vulnerabilities in the web server, and gathers information about users and shared directories via NetBIOS connections. Ntis temporarily hangs during the web services portion of the attack when it executes a particular server-side CGI application (newdsn.exe). There is a timeout of 15min, after which the attack will continue executing.

Verification: If the attack succeeds, there will be a file on the attacker machine, named <victim>.html, where <victim> is the IP address of the victim machine. In addition, the "last modified" date must agree with the date and time when the attack was launched, and the file must be opened to verify that data was collected by the scan.

Cleanup: No cleanup is required.

Detection

Network Traffic: Net Tracker can be used to reconstruct the FTP and HTTP connections that occur during the attack. Figure 9-1 shows the text of the generated transcripts. The attack can be detected with a keyword matching intrusion detection system by searching for the text strings shown in boldface. The FTP transcript shows the attacker logging into the victim machine and attempting to upload a file called, "ntis-ftp.txt." The HTTP transcripts show the attacker testing for vulnerabilities in the victim system by attempting several "GET" requests to the victim web server.

FTP CONNECTION	HTTP CONNECTIONS
206.48.44.18:1256=>172.16.112.100:21 user anonymous **pass guestaccnt@compuserve.com** port 199,199,199,199,0,80 port 199,199,199,199,10,10 cwd /c **stor ntis-ftp.txt** quit 500 Invalid PORT Command. 500 Invalid PORT Command. 250 CWD command successful. 150 Opening ASCII mode data connection for ntis-ftp.txt. 425 Can't open data connection.	206.48.44.18:1256=>172.16.112.100:80 **GET /*.idc HTTP/1.0** HTTP/1.0 400 Bad Request **GET /cgi-bin/ HTTP/1.0** HTTP/1.0 403 Access Forbidden **GET /scripts/ HTTP/1.0** HTTP/1.0 403 Access Forbidden **GET /cgi-bin/perl.exe?-v HTTP/1.0** HTTP/1.0 403 Access Forbidden **GET /scripts/perl.exe?-v HTTP/1.0** HTTP/1.0 403 Access Forbidden **GET /scripts/tools/newdsn.exe HTTP/1.0** HTTP/1.0 502 Gateway Error Server: Microsoft-IIS/2.0 Content-Type: text/html <head><title>CGI Application Timeout</title></head> <body><h1>CGI Timeout</h1> The specified CGI application exceeded the allowed time for processing. The server has deleted the process.</body> **GET /_vti_bin/fpcount.exe?Page=default.htm|Image=3|Digits=15** HTTP/1.0 HTTP/1.0 403 Access Forbidden **GET /scripts/*%0a.pl HTTP/1.0** HTTP/1.0 403 Access Forbidden **GET /samples/search/queryhit.htm HTTP/1.0** HTTP/1.0 404 Object Not Found
HTTP CONNECTION	
206.48.44.18:1256=>172.16.112.100:80 HEAD / HTTP/1.0 HTTP/1.0 200 OK Server: Microsoft-IIS/2.0 Date: Thu, 08 Apr 1999 15:27:33 GMT Content-Type: text/html Accept-Ranges: bytes Last-Modified: Wed, 03 Mar 1999 16:40:55 GMT Content-Length: 1513	

Figure 9-1: Transcripts of FTP and HTTP Connections from an NTInfoScan Attack.

In this version of the attack, the attacker uses the password, "guestaccnt@compuserve.com" to establish an anonymous FTP connection to the victim machine. The original version of NTInfoScan uses the password, "NTInfoScan@security.check."

Host Data from the 1999 Evaluation: The NTInfoscan attack can be detected by searching for a particular series of events in the 1999 evaluation Windows NT audit logs. When the attack connects to the victim machine to collect user account information, there will be an individual audit log entry created for each access of the Security Account Manager (SAM). The number of accesses will be equal to the number of user accounts on the system. Figure 9-2 shows one of the 92 audit log entries created during an

NTInfoScan attack against the Windows NT victim machine. SYSTEM accesses the Security Account Manager with read-only privileges to gather user account information.

```
Object Open:
            Object Server:      Security Account Manager
            Object Type:        SAM_USER
            Object Name:        DOMAINS\Account\Users\0000040E
            New Handle ID:      1509008
            Operation ID:       {0,32531}
            Process ID:         2156644800
            Primary User Name:SYSTEM
            Primary Domain:     NT AUTHORITY
            Primary Logon ID:   (0x0,0x3E7)
            Client User Name:
            Client Domain:
            Client Logon ID:    (0x0,0x25D8)
            Accesses            READ_CONTROL
                        ReadGeneralInformation
                        ReadPreferences
                        ReadLogon
                        ReadAccount
                        ListGroups

            Privileges          -
```

Figure 9-2: One Access of the Security Account Manager by an NTInfoScan Attack.

The NTInfoScan attack can also be detected in the IIS log file. The same keywords that are revealed in network traffic transcripts will be revealed in the IIS log file. Figure 9-3 shows the IIS log file entries generated by an NTInfoScan attack with keywords in boldface text.

```
206.48.44.18, anonymous, 4/1/99, 7:59:59, MSFTPSVC, HUME, -, 0, 15, 0, 0, 0, [1]  USER , anonymous, -,
206.48.44.18, guestaccnt@compuserve.com, 4/1/99, 7:59:59, MSFTPSVC, HUME, -, 469, 31, 0, 0, 0, [1]  PASS ,
guestaccnt@compuserve.com, -,
206.48.44.18, guestaccnt@compuserve.com, 4/1/99, 8:00:01,MSFTPSVC,HUME, -, 1625, 78, 0, 0, 10061, [1] created , ntis-ftp.txt, -,
206.48.44.18, guestaccnt@compuserve.com, 4/1/99, 8:00:59, MSFTPSVC, HUME, -, 0, 5, 0, 0, 0, [1]  QUIT , -, -,
206.48.44.18, -, 4/1/99, 8:00:59, W3SVC, HUME, 172.16.112.100, 63, 17, 198, 200, 0, HEAD, /Default.htm, -,
206.48.44.18, -, 4/1/99, 8:00:59, W3SVC, HUME, 172.16.112.100, 0, 21, 101, 400, 123, GET, /*.idc, -,
206.48.44.18, -, 4/1/99, 8:00:59, W3SVC, HUME, 172.16.112.100, 15, 24, 111, 403, 5, GET, /cgi-bin/, -,
206.48.44.18, -, 4/1/99, 8:00:59, W3SVC, HUME, 172.16.112.100, 0, 24, 273, 403, 5, GET, /scripts/, -,
206.48.44.18, -, 4/1/99, 8:00:59, W3SVC, HUME, 172.16.112.100, 0, 35, 273, 403, 5, GET, /scripts/perl.exe, -v,
206.48.44.18, -, 4/1/99, 8:15:59, W3SVC, HUME, 172.16.112.100, 900094, 40, 275, 502, 0, GET, /scripts/tools/newdsn.exe, -,
206.48.44.18, -, 4/1/99, 8:15:59, W3SVC, HUME, 172.16.112.100, 0, 31, 273, 403, 5, GET, /scripts/*.pl, -,
206.48.44.18, -, 4/1/99, 8:15:59, W3SVC, HUME, 172.16.112.100, 32, 43, 111, 404, 3, GET, /samples/search/queryhit.htm, -,
```

Figure 9-3: IIS Log Entries Recorded During a NTInfoScan Attack.

Chapter 10

Detectability of Attacks

This chapter presents an experiment to determine the detectability of the new 1999 evaluation Windows NT attacks in Windows NT audit logs. The detection information included in chapter six through chapter nine is assembled into attack signatures for each of the 12 attacks. These signatures are coded into a PERL script program called NT Audit Detect (NTAD), developed specifically for this experiment. NTAD uses Windows NT audit logs, from the 1999 evaluation test data, as input data. Detection and false alarm results are presented.

10.1 Motivation and Goal

The motivation for this experiment is to promote more research and development of intrusion detection systems that utilize Windows NT audit data. Windows NT hosts are essential components in many computing environments. Despite their growing importance, researchers are only beginning to develop intrusion detection systems that use Windows NT audit data. Only one participant in the 1999 DARPA Intrusion Detection Evaluation submitted a system that could detect attacks against Windows NT hosts using Windows NT audit data [7].

The goal of the experiment is to present the detectability of the 1999 Windows NT attacks in audit data and provide information that will make it easier for researchers to extend their existing systems to process Windows NT audit data and begin detecting Windows NT attacks.

10.2 Testing Audit Log Signatures

To test the validity of the signatures described in chapters six through nine, a PERL program called NTaudit-detect.pl (NTAD) was written (full source code in Appendix A). NTAD uses the signatures defined in the attack documentation to scan for the new 1999 Windows NT attacks in audit log data. It processes comma-separated text versions of the audit logs. These are created by using the Windows NT Event Viewer to save the original event logs as comma-delimited text files.

Figure 10-1 shows one function of the NTAD program. This function detects CrashIIS attacks in a Windows NT security event log. Line six begins the loop that searches through the event log, one line at a time. Lines seven through 11 look for the process ID of the IIS process (inetinfo.exe). Lines 12 through 17 look for the Dr. Watson process (drwtsn32.exe), and checks to see if its Creator Process ID matches the process ID of inetinfo.exe. If the ID’s match, lines 18 through 25 parse the date and time from the event log and print an alert indicating that a CrashIIS attack was detected (See Section 6-1 for CrashIIS documentation). There are similar functions in the NTAD code for all of the Windows NT attacks developed for the 1999 evaluation (Appendix A contains the full source code).

The results of running NTAD on the 1999 test data are shown in Figure 10-2. Note that this is not an official set of results and that the results are overly optimistic because the same attack generation tools were used twice, both to create test data and to develop signatures.

```
sub detect_crashiis {
 # drwtsn.exe started by the inetinfo.exe process will indicate a CrashIIS attack
  print "Looking for CrashIIS attacks...\n";
  # save previous line for process ID of inetinfo
  $prevline = "";
  while (<EVENTLOG>) {
    if (($_ =~ "inetinfo.exe") && ($prevline =~ "New Process")) {
      print "Discovered inetinfo.exe \n";
      # get process ID for IIS
      @fields = split (" ", $prevline);
      $processID = $fields[3];}
    if ($_ =~ "drwtsn32.exe") {
      print "Discovered drwtsn32.exe \n";
      # skip 1 line to look for the creator process ID
      $_ = <EVENTLOG>;
      # compare with inetinfo ID
      if ($_ =~ $processID) {
        # skip down to get date and time
        while (!($_ =~ "HUME")) {
          $_ = <EVENTLOG>;}
        @fields = split /,/, $_;
        $date = $fields[0];
        $time = $fields[1];
        print "CrashIIS attack detected!!:\n";
        print "on $date at $time.\n";}
      else {print "\n"};}
    $prevline = $_;}}
```

Figure 10-1: Function in NTAD Detects CrashIIS Attacks in an Audit Log.

Column one indicates the week and day when the attack instance occurred. Week one through week three were training data weeks, so week four translates to week one of the test data and week five translates to week two of the test data. Days numbered one through five represent the days Monday through Friday respectively. Column two lists the time of day for each attack, in the form HH:MM:SS. Columns three and four indicate the name and type of each attack. Attack instances labeled “CrashIIS-Yaga” indicate that the CrashIIS attack was launched as part of the Yaga attack (See documentation of the Yaga attack in Section 8.5). The fifth column of the table contains a “1” if NTAD issued an alert for the attack and a "0" if it did not. The sixth and final column displays, for the given attack, the total number of false alarms generated by NTAD in the two weeks of test data. For example, a "1" in this column for a CrashIIS attacks indicates that NTAD only generated one false alarm when searching the test data for CrashIIS attacks.

As indicated in the table, the audit logs for day five of week five were not collected properly and therefore could not be used. In addition, the audit logs from some days of the evaluation were cleared in the beginning of the day, but after the Windows NT victim machine booted up. As a result, the initiation of the IIS process (inetinfo.exe) in the beginning of the day was not audited. NTAD was unable to detect CrashIIS attacks, because the IIS process ID could not be matched with the process ID of the drwtsn32.exe process (See CrashIIS documentation in Section 6.1). The CrashIIS attacks marked with asterisks (*CrashIIS) are the attacks that NTAD was unable to detect. The attacks are still labeled as detected because they would have been detected if the audit logs were not cleared after the Windows NT victim machine was booted.

As can be seen, 26 of the 29 (90%) attacks that occurred during periods where Windows NT audit records were available were detected and only 1 false alarm was generated. This good result, and the relatively simple nature of the signatures, demonstrates that the Windows NT audit records collected in the 1999 evaluation contain much useful information concerning the 1999 Windows NT attacks. This information, however, needs to be supplemented to detect attacks, such as PPMacro and other attacks where information on file and Registry access is important. An audit policy that audits the Registry key and files modified by the PPMacro program would make it possible to detect the attack in audit data (see documentation of PPMacro in Section 7.4).

Week-Day	Time	Name	Category	Detected	Total False Alarms
All	All	All Attacks	All	26 of 29	1
4-1	09:00:00	NTFSDOS	U2R	1	0
4-1	15:50:48	Yaga	U2R	1	0
4-1	16:13:08	*CrashIIS-Yaga	DoS	1	1
4-2	12:00:00	NTFSDOS	U2R	1	0
4-2	14:32:28	SecHole	U2R	1	0
4-2	21:04:10	*CrashIIS	DoS	1	1
4-3	10:00:00	PPMacro	R2L	0	0
4-3	11:00:00	NetCat	R2L	1	0
4-4	08:00:00	NTInfoScan	Probe	1	0
4-4	08:30:00	NetBus	R2L	1	0
4-4	11:00:00	DoSNuke	DoS	1	0
4-4	12:05:00	PPMacro	R2L	0	0
4-5	12:01:46	NetBus	R2L	1	0
4-5	16:50:09	SecHole	U2R	1	0
5-1	11:45:00	DoSNuke	DoS	1	0
5-1	18:36:23	CrashIIS	DoS	1	1
5-1	19:47:15	DoSNuke	DoS	1	0
5-2	08:53:57	CaseSen	U2R	1	0
5-2	13:50:03	CrashIIS	DoS	1	1
5-2	14:02:07	PPMacro	R2L	0	0
5-2	20:56:05	DoSNuke	DoS	1	0
5-3	09:48:00	NetBus	R2L	1	0
5-3	11:05:00	NetCat	R2L	1	0
5-4	09:12:00	CaseSen	U2R	1	0
5-4	10:21:02	NTFSDOS	U2R	1	0
5-4	11:04:16	NTInfoScan	Probe	1	0
5-4	11:50:00	Yaga	U2R	1	0
5-4	11:57:01	*CrashIIS-Yaga	DoS	1	1
5-4	16:03:41	SecHole	U2R	1	0
5-4	18:30:02	NTInfoScan	Probe	1	0
Audit logs for Week 5, Day 5 were not collected properly:					
5-5	08:14:18	CrashIIS	DoS	-	-
5-5	08:55:14	NetCat	R2L	-	-
5-5	10:06:00	AnyPW	U2R	-	-
5-5	11:08:00	Framespoofer	R2L	-	-
5-5	12:44:00	Yaga	U2R	-	-
5-5	12:51:12	CrashIIS-Yaga	DoS	-	-
5-5	12:58:30	CrashIIS	DoS	-	-
5-5	20:49:25	Casesen	U2R	-	-

Figure 10-2: Detection Results of the ntaudit-detect.pl Script (NTAD) for the New 1999 Windows NT Attacks.

Chapter 11

Results and Future Work

Overall, the 1999 DARPA Off-Line Intrusion Detection Evaluation was a success and a major improvement over the 1998 evaluation. It provided training data containing no attacks for training anomaly detection systems. Systems were scored on attack identification in addition to attack detection. Scoring and verification procedures were simplified, a written security policy was provided, and a more detailed analysis of attack misses and false alarms was performed. In addition, the 1999 attack set was extended to include more stealthy attacks, insider attacks, and attacks against the Windows NT operating system. This chapter summarizes the results of the 1999 evaluation regarding Windows NT attacks and presents suggestions related to Windows NT for future evaluations.

11.1 Windows NT Results of the 1999 Evaluation

The results of the 1999 evaluation [11] were analyzed to determine how well the best systems performed in detecting Windows NT attacks. Systems that were designed to detect denial-of-service and probe attacks against the Windows NT victim machine performed well. The top two systems in this category are UCSB [38] and Emerald Expert [20]. Systems that were designed to detect remote-to-local and user-to-root attacks against Windows NT performed poorly. In fact, only one participant, RST [7], designed systems to detect these types of attacks.

The Windows NT attack detection results for the 1999 evaluation can be found in [11]. Figure 11-1 shows the detections results for the two systems best at detecting probe and denial-of-service attacks. There were a total of 16 instances of Windows NT denial of service attacks and 8 instances of Windows NT probe attacks. These numbers include new attacks developed for the 1999 evaluation and old attacks that were developed in the 1998 evaluation. The detection results shown in the table are relative to a maximum of 10 false alarms per day for each system. The highest scoring system was the Emerald Expert system, which detected 69% of the Windows NT denial-of-service attacks (11 of 16) and 63% of the Windows NT probe attacks (5 of 8). One reason why these systems did so well is that many of the attacks were not new to them. Six of the eight probe attacks, instances of NTInfoScan, Ipsweep and Portsweep, and nine of the 16 denial-of-service attacks, instances of CrashIIS and Smurf, were attacks seen in the 1999 training data.

RST was the only participant that designed systems that detect Windows NT remote-to-local and user-to-root attacks. The RST system that was best at detecting these attacks in the 1999 evaluation was RST State-Tester [7]. However, this system detected fewer than 20% of the Windows NT remote-to-local and user-to-root attacks. This result may not reflect the performance that can be achieved by the state-tester approach. This approach uses Windows NT audit log data to detect attacks. It examines audit logs for sequences of records that are anomalous for known processes. The state-tester approach works well in detecting UNIX remote-to-local and user-to-root attacks, because many of

Week-Day	Time	Name	Category	UCSB	Emerald Expert
All	All	All DoS	DoS	69% (11/16)	69% (11/16)
All	All	All Probe	Probe	38% (3/8)	63% (5/8)
4-1	12:22:22	Portsweep	Probe	0	0
4-1	16:13:08	CrashIIS	DoS	1	1
4-2	21:04:10	CrashIIS	DoS	1	1
4-3	14:45:47	Smurf	DoS	1	1
4-3	16:43:34	Portsweep	Probe	0	0
4-4	08:00:59	NTInfoScan	Probe	1	1
4-4	11:00:00	DoSNuke	DoS	0	1
4-5	19:25:23	Ipsweep	Probe	0	1
5-1	11:45:00	DoSNuke	DoS	0	0
5-1	13:30:19	ArpPoison	DoS	0	0
5-1	18:36:23	CrashIIS	DoS	1	1
5-1	19:47:15	DoSNuke	DoS	1	0
5-2	13:50:03	CrashIIS	DoS	1	1
5-2	20:56:05	DoSNuke	DoS	0	1
5-4	11:04:16	NTInfoScan	Probe	1	1
5-4	11:57:01	CrashIIS	DoS	1	1
5-4	17:01:32	ResetScan	Probe	0	0
5-4	18:30:02	NTInfoScan	Probe	1	1
5-4	22:51:31	ArpPoison	Dos	1	0
5-5	08:14:18	CrashIIS	DoS	1	1
5-5	08:55:50	InsideSniffer	Probe	0	1
5-5	10:20:00	TCPReset	DoS	0	0
5-5	12:51:12	CrashIIS	DoS	1	1
5-5	12:58:30	CrashIIS	DoS	1	1

Figure 11-1: Detection Results for Probe and Denial-of-Service Attacks with a Maximum of 10 False Alarms per Day.

those attacks misuse existing programs, thereby creating anomalous BSM log records. However, many of the 1999 Windows NT remote-to-local and user-to-root attacks, such as NetBus and NetCat, do not misuse existing programs. Instead, these attacks introduce new malicious code. In addition, BSM auditing information differs from Windows NT auditing information. BSM auditing records system calls while Windows NT auditing records higher-level information, such as object access. Finally, this was the first year that RST designed a system to detect Windows NT attacks in audit log data.

Despite the poor detection results for Windows NT remote-to-local and user-to-root attacks, it is evident by the experiment performed in Chapter 10 that the attacks are detectable in Windows NT audit logs. The experiment in Chapter 10 validated the usefulness of the attack signatures documented in Chapters 6 through 9. These signatures provide a good set of features that could be used to develop host-based signature-based Windows NT intrusion detection systems. In future DARPA evaluations, the 1999 Windows NT test data will be available for training. This data, combined with other Windows NT data, can be used to develop improved Windows NT intrusion detection systems.

11.2 Windows NT Suggestions for Future Evaluations

This section provides suggestions regarding Windows NT in future evaluations. These suggestions span four aspects of Windows NT in the evaluation: hardware and software, distributed host data, traffic automation, and the attack set.

11.2.1 Hardware and Software

The following is a list of suggestions related to Windows NT hardware and software for future evaluations:

- Use Microsoft Exchange Server as the mail server for the Windows NT victim machine.
- Add additional Windows NT victim machines with more up-to-date Service Packs to the test bed.

Microsoft Exchange Server [15] is the recommended Windows NT mail server for future evaluations because it is the most popular Windows NT mail server in business and military environments. It is more realistic to use Microsoft Exchange Server than the

Resource Kit mail server (Mailsrv), which has been announced as faulty and unsupported by Microsoft [23]. At least one Windows NT victim machine must be equipped with Service Pack 3 to be capable of running Microsoft Exchange Server. Introducing machines with more recent Service Packs will also make the evaluation more realistic.

11.2.2 Distributed Host Data

The following is a list of suggestions related to Windows NT distributed host data for future evaluations:

- Generate and distribute audit logs with a more extensive security auditing policy.
- Distribute other log files.

Several attack signatures were listed in the "Extended Host Data" sections of Chapters six through ten. If the data indicated in these sections is provided in future evaluations, participating systems that utilize host data will have a better chance of detecting Windows NT attacks. For this reason, a more extensive Windows NT auditing policy should be adopted in future evaluations. Such an audit policy should audit important Registry keys and important files on the system. However, too much auditing can significantly affect system performance. Experiments should be conducted to determine an audit policy that provides the most useful information without severely affecting system performance. Other Windows NT files mentioned in the "Extended Host Data" sections contain attack signatures, and should also be distributed in future evaluations. An example of such a file is the Dr. Watson log file, C:\WINNT\user.dmp.

11.2.3 Traffic Automation

The following is a list of suggestions related to Windows NT traffic automation for future evaluations:

- Samba automation
- Other automation (macros)

In the 1999 evaluation, Windows NT attacks were executed manually when attack actions, such as executing an email attachment or visiting a specific web page, were necessary. If these types of actions and other Windows NT actions were automated, it would be simpler and less time consuming to deploy Windows NT attacks in the evaluation test bed. Automation possibilities that should be explored for future evaluations include Samba [31], which allows UNIX machines to control Windows NT machines, and other types of automation, such as Windows NT macros.

11.2.4 The Attack Set

The following is a list of suggestions related to the Windows NT attack set for future evaluations:

- Buffer overflow attacks.
- More probe attacks.
- More attacks requiring the execution of Visual Basic and ActiveX email attachments of various types.

The Windows NT attack set must be updated and extended in each successive evaluation, to remain realistic and relatively comprehensive. Attack types that were lacking in the 1999 Windows NT attack set, and that should be considered for future attack sets include, buffer overflow attacks and more probe attacks. In addition, Windows NT attacks that require the victim to execute Visual Basic and ActiveX email attachments are currently popular. Future evaluations should include more of these types of attacks to create realistic attack sets.

Appendix A

Source Code for NTAD (ntaudit-detect.pl)

```
#!/usr/local/bin/perl
#
# NTAD - NTAUDIT-DETECT.PL
# Jonathan Korba - Last Updated 5/18/2000
#
# This program demonstrates the detectability of NT attacks
# in the NT audit data gathered from the victim NT server (HUME) in the
# 1999 DARPA Off-Line Intrusion Detection Evaluation.
# Detection is signature based.
#
# Input parameters for this program are the name of the audit log
# text file to scan for attacks, and the type of attack(s) to detect.
# The audit log text file must be created by opening an audit log
# in NT EventViewer, ordering it from oldest record to newest record,
# and then saving it as a comma-delimited text file.
#

sub usage {
    print "\nUsage:\n";
    print "ntaudit-detect.pl <audit log text file> <attack(s) to detect>\n";
    print "\nPossible attack(s) to detect:\n";
    print " casesen (U2R)\n";
    print " crashiis (DoS)\n";
    print " hardboot (Hard Reboot - Could indicate DoSNuke, AnyPW, NTFSDOS, etc.)\n";
    print " netbus (R2L)\n";
    print " netcat (R2L)\n";
    print " ntis (NTInfoScan - Probe)\n";
    print " sechole (U2R)\n";
    print " yaga (U2R)\n";
    print " all (All of the above)\n\n";
    print "\nUndetectable with 1999 Auditing Policy:\n";
    print " FrameSpoofer\n";
    print " PPMacro\n\n";
}

sub detect_casesen {
# A good signature for the CaseSen Attack is:
# POSIX.EXE executes, PSXSS.EXE executes,
# then a user added to Admin group by SYSTEM
    $posix = 0;
    $psxss = 0;
    print "Looking for CaseSen Attacks...\n";
    while (<EVENTLOG>) {
       if (($_ =~ "POSIX.EXE") && ($posix == 0)) {
           print "Discovered execution of POSIX.EXE ";
           for ($x = 0; $x < 2; $x++) {
              # skip 2 lines to look for the User
              $_ = <EVENTLOG>;}
           if (($_ =~ "User\ Name\:") && !($_ =~ "Administrator")) {
              $posix = 1;
              print ": not run by Administrator"; }
           print "\n";
       }
       if (($_ =~ "PSXSS.EXE") && ($posix == 1) && ($psxss == 0)) {
           print "Discovered execution of PSXSS.EXE\n";
           $psxss = 1;
       }
       if (($_ =~ "Group\ Member\ Added") && ($psxss == 1)) {
           print "Discovered Group Member Added";
           # get date and time
```

```
            @fields = split /,/, $_;
            $date = $fields[0];
            $time = $fields[1];
            for ($x = 0; $x < 5; $x++) {
               # skip 5 lines to look for the User
               $_ = <EVENTLOG>;}
            if (($_ =~ "Caller\ User\ Name\:") && ($_ =~ "SYSTEM")) {
               print ": by SYSTEM\n";
               # CaseSen has been detected!!!
               print "CaseSen Detected at $date $time\n";
               # Reset variables because there may be more casesens
               $posix = 0;
               $psxss = 0;}
        }
    }
}

sub detect_crashiis {
# drwtsn.exe started by the inetinfo.exe process will indicate a CrashIIS attack
    print "Looking for CrashIIS attacks...\n";
    # save previous line for process ID of inetinfo
    $prevline = "";
    while (<EVENTLOG>) {
       if (($_ =~ "inetinfo.exe") && ($prevline =~ "New Process")) {
           print "Discovered inetinfo.exe\n";
           # get process ID for IIS
           @fields = split (" ", $prevline);
           $processID = $fields[3];
       }
       if ($_ =~ "drwtsn32.exe") {
           print "Discovered drwtsn32.exe\n";
           # skip 1 line to look for the creator process ID
           $_ = <EVENTLOG>;
           # compare with inetinfo ID
           if ($_ =~ $processID) {
              # skip down to get date and time
              while (!($_ =~ "HUME")) {
                  $_ = <EVENTLOG>;}
              @fields = split /,/, $_;
              $date = $fields[0];
              $time = $fields[1];
              print "CrashIIS attack detected!!:\n";
              print "on $date at $time.\n";}
            else {print "\n"};
       }
       $prevline = $_;
    }
}

sub detect_hardboot {
# if a "Windows NT is starting up" System Event
#  is not preceded by "SeShutdownPrivilege" Privilege Use Event
# then a hard reboot occurred
# Possible attacks: DoSNuke, WinNuke, AnyPW, NTFSDOS
    print "Looking for Hard Reboots...\n";
    $privilege = 0;  # flag indicating SeShutdownPrivelege Event
    while (<EVENTLOG>) {
       if ($_ =~ "SeShutdownPrivilege") {
           $privilege = 1;}
       if ($_ =~ "Windows NT is starting up.") {
           if ($privilege == 1) {
              print "Detected soft reboot.\n";}
           else {
              # skip down to get date and time
              while (!($_ =~ "HUME")) {
                  $_ = <EVENTLOG>;}
              @fields = split /,/, $_;
                $date = $fields[0];
                $time = $fields[1];
              print "Detected hard reboot!!:\n";
              print "on $date at $time. (Possible attacks: DoSNuke, AnyPW, NTFSDOS)\n";}
            $privilege = 0;  # Reset variable because there may be more reboots
        }
    }
}

sub detect_netbus {
# the execution of a file called explore.exe
# is an indicator of the NetBus attack
# Note: If Netbus uses a different file name it will not be detected by this program
    print "Looking for NetBus Attacks...\n";
    while (<EVENTLOG>) {
       if ($_ =~ "explore.exe") {
           print "Discovered execution of explore.exe (common name for NetBus)\n";
           # skip down to get date and time
           while (!($_ =~ "HUME")) {
              $_ = <EVENTLOG>;}
           @fields = split /,/, $_;
           $date = $fields[0];
           $time = $fields[1];
           print "NetBus attack detected!!:\n";
           print "on $date at $time.\n";}
    }
}

sub detect_netcat {
# A good signature for netcat is: REGEDIT.EXE executes,
#   and later winlog.exe executes (common name for netcat trojan)
# Note: A netcat attack which uses a name other than winlog.exe will not be detected
    print "Looking for NetCat Attacks...\n";
    $reg = 0;  # flag will be set to 1 if REGEDIT.EXE is run
    while (<EVENTLOG>) {
       if ($_ =~ "REGEDIT.EXE") {
           print "Discovered REGEDIT.EXE\n";
           $reg = 1;
       }
       if (($_ =~ "winlog.exe") && ($reg == 1)) {
           # skip down to get date and time
           while (!($_ =~ "HUME")) {
              $_ = <EVENTLOG>;}
           @fields = split /,/, $_;
           $date = $fields[0];
           $time = $fields[1];
           print "Detected Netcat Attack!!:\n";
           print "on $date at $time.\n";
           $reg = 0;  # Reset variable because there may be more netcat attacks
       }
    }
}

sub detect_ntis {
# Successful Logon IUSR via Advapi + newdsn.exe executed by SYSTEM => web scan
# Successful Logon via KSecDD + multiple SAM_USER accessed by SYSTEM => netbios scan
    print "Looking for NTIS attacks...\n";
    $wlogon = 0;  # web scan login
    $nlogon = 0;  # netbios scan login
    $iuser = 0;   # IUSR login
    $readusr = 0; # num times user database was read (at least 50 for netbios scan)
    while (<EVENTLOG>) {
       if ($_ =~ "Successful Logon") {
           $_ = <EVENTLOG>;
           if ($_ =~ "IUSR") {
              $iuser = 1;}
           for ($x = 0; $x < 4; $x++) {
                # skip 4 lines to get Logon Process
                $_ = <EVENTLOG>;}
           if ($_ =~ "KSecDD") {
              print "Detected logon via KSecDD.\n";
              $nlogon = 1;}
           if (($_ =~ "Advapi") && ($iuser == 1)) {
              print "Detected IUSR logon using Advapi.\n";
              $iuser = 0;
              $wlogon = 1;}}
       if (($_ =~ "newdsn.exe") && ($wlogon == 1)) {
           # skip down to get date and time
           while (!($_ =~ "HUME")) {
              $_ = <EVENTLOG>;}
           @fields = split /,/, $_;
           $date = $fields[0];
           $time = $fields[1];
           print "Detected NTIS Web Scan!!!:\n";
           print "on $date at $time.\n";
           # reset variables and look for more scans
           $wlogon = 0;
       }
       if (($_ =~ "SAM_USER") && ($nlogon == 1)) {
           for ($x = 0; $x < 5; $x++) {
              # skip 5 lines to look for the User
                $_ = <EVENTLOG>;}
           if (($_ =~ "Primary\ User\ Name\:") && ($_ =~ "SYSTEM")) {
              $readusr += 1;}
           if ($readusr == 50) {
              # skip down to get date and time
              while (!($_ =~ "HUME")) {
                  $_ = <EVENTLOG>;}
              @fields = split /,/, $_;
                $date = $fields[0];
                $time = $fields[1];
              print "Detected NTIS NetBios Scan!!!:\n";
              print "on $date at $time.\n";
              # reset variables and look for more scans
              $nlogon = 0;
              $readusr = 0;}
       }
    }
}

sub detect_sechole {
# A good signature for the SecHole Attack is:
#  a user added to Admin group by SYSTEM
# Note: Could also indicate a different attack (e.g. casesen, yaga)
    print "Looking for SecHole attacks...\n";
    while (<EVENTLOG>) {
       if ($_ =~ "Group\ Member\ Added") {
           print "Discovered Group Member Added";
            # get date and time
           @fields = split /,/, $_;
           $date = $fields[0];
           $time = $fields[1];
           for ($x = 0; $x < 5; $x++) {
                # skip 5 lines to look for the User
              $_ = <EVENTLOG>;}
           if (($_ =~ "Caller\ User\ Name\:") && ($_ =~ "SYSTEM")) {
              print ": by SYSTEM\n";
                # Possible Sechole has been detected!!!
              print "Possible Sechole Detected at $date $time\n";}
           else {print "\n";}
       }
    }
}

sub detect_yaga {
# A good signature for the Yaga Attack is:
# 1) CAT.EXE runs (not necessary)
# 2) REGEDIT.EXE run by a user other than Administrator
# 3) net.exe command run by SYSTEM (not a user)
# 4) Group Member Added by SYSTEM (not Administrator)
#      (this last one happens with CaseSen and SecHole as well)
    $cat = 0; # flag set to one if CAT.EXE runs
    $reg = 0; # flag set to one if REGEDIT.EXE runs
    $net = 0; # flag set to one if net.exe runs
    print "Looking for Yaga Attacks...\n";
    while (<EVENTLOG>) {
       if ($_ =~ "CAT\.EXE") {
           print "Discovered CAT.EXE\n";
           $cat = 1;}
       if (($_ =~ "REGEDIT\.EXE") && ($reg != 1)) {
           print "Discovered REGEDIT.EXE";
           for ($x = 0; $x < 2; $x++) {
              # skip 2 lines to look for the User
              $_ = <EVENTLOG>;}
           if (($_ =~ "User\ Name\:") && !($_ =~ "Administrator")) {
              $reg = 1;
              print ": not run by Administrator"; }
           print "\n";
           next;}
       if (($_ =~ "net\.exe") && ($reg == 1)) {
           print "Discovered net.exe";
           for ($x = 0; $x < 2; $x++) {
              # skip 2 lines to look for the User
              $_ = <EVENTLOG>;}
           if (($_ =~ "User\ Name\:") && ($_ =~ "SYSTEM")) {
              $net = 1;
              print ": run by SYSTEM";}
           print "\n";
           next;}
       if (($_ =~ "Group\ Member\ Added") && ($net == 1)) {
           print "Discovered Group Member Added";
           # get date and time
           @fields = split /,/, $_;
           $date = $fields[0];
           $time = $fields[1];
           for ($x = 0; $x < 5; $x++) {
              # skip 5 lines to look for the User
              $_ = <EVENTLOG>;}
           if (($_ =~ "Caller\ User\ Name\:") && ($_ =~ "SYSTEM")) {
              print ": by SYSTEM\n";
              # Yaga has been detected!!!
              print "Yaga Attack Detected at $date $time\n";
              # Reset cause there may be more yagas
              $cat = 0;
              $reg = 0;
              $net = 0;
           }
           else {print "\n";}
           next;}
    }
}

if ($#ARGV != 1) {
    # requires exactly 2 args
    usage;}
else {
    # open security event log textfile
    open(EVENTLOG,"<$ARGV[0]") ||
       die "Cannot open Event Log File $!";
    $attack = $ARGV[1];
    if ($attack eq "casesen") {
       detect_casesen;}
    elsif ($attack eq "crashiis") {
       detect_crashiis;}
    elsif ($attack eq "hardboot") {
       detect_hardboot;}
    elsif ($attack eq "netbus") {
       detect_netbus;}
    elsif ($attack eq "netcat") {
       detect_netcat;}
    elsif ($attack eq "ntis") {
       detect_ntis;}
    elsif ($attack eq "sechole") {
       detect_sechole;}
    elsif ($attack eq "yaga") {
       detect_yaga;}
    elsif ($attack eq "all") {
       detect_casesen;
       detect_crashiis;
       detect_hardboot;
       detect_netbus;
       detect_netcat;
       detect_ntis;
       detect_sechole;
       detect_yaga;}
    else {usage;}}

exit;
```

References

[1] Ataman Software web site, http://www.ataman.com/.

[2] Boutell.com web site, http://www.boutell.com/wusage/.

[3] Bugtraq Archives (e-mail regarding Apache vulnerability). http://www.geek-girl.com/bugtraq/1998_3/0442.html/. August 7, 1998.

[4] "Case Sensitivity Vulnerabiliy," NT Security News, http://www.ntsecurity.net/scripts/loader.asp?iD=/security/casesensitive.htm.

[5] Computer Emergency Response Team Website. http://www.cert.org/.

[6] Kumar Das, "Attack Development for Intrusion Detection Evaluation," M.Eng. Thesis, MIT Department of Electrical Engineering and Computer Science, June 2000.

[7] A. K. Ghosh, A. Schwatzbard and M. Shatz, "Learning Program Behavior Profiles for Intrusion Detection," in Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California, April 1999, http://www.rstcorp.com/~anup/..

[8] Insecure.org. http://www.insecure.org/.

[9] Internet Security Systems X-Force. http://www.iss.net/.

[10] Kris Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," M.Eng. Thesis, MIT Department of Electrical Engineering and Computer Science, June 1999.

[11] Lincoln Laboratory ID Evaluation Website, MIT, http://www.ll.mit.edu/IST/ideval/index.html/, 2000, contains information on the 1998 and 1999 evaluations. Follow instructions on this web site or send email to the authors (rpl or jwh@sst.ll.mit.edu) to obtain access to a password protected site with complete up-to-date information on these evaluations and results.

[12] Lawrence Berkeley National Laboratory, Network Research Group Homepage. http://www.nrg.ee.lbl.gov/. May 1999.

[13] Richard P. Lippmann, David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman, "Evaluating Intrusion Detection Systems: the 1998 DARPA Off-Line Intrusion Detection Evaluation," in Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), Vol. 2 (2000).

[14] Richard P. Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, Kumar Das, "The 1999 DARPA Off-Line Intrusion Detection Evaluation," submitted to Proceedings of the 3rd International Workshop on Recent Advances in Intrusion Detection (RAID 2000).

[15] "Microsoft Exchange Server Site," Microsoft BackOffice, http://www.microsoft.com/exchange/.

[16] *Microsoft Windows NT Server Resource Kit*, Version 4.0, Microsoft Press, One Microsoft Way, Redmond, Washington, 98052, October 1996.

[17] James D. Murray. *Windows NT Event Logging*. O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol CA, 95472, September 1998.

[18] NetBus web site, http://www.netbus.com.

[19] Net Security web site, http://www.net-security.sk/bugs/NT/oob.html.

[20] P. Neumann and P. Porras, "Experience with EMERALD to DATE", in Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, California, April 1999, pp. 73-80, http://www.sdl.sri.com/emerald/index.html/.

[21] Next Step Software web site, http://nssoft.hypermart.net/.

[22] NT Security News, http://www.ntsecurity.net/.

[23] "NT 4.0 Resource Kit Utilities Corrections and Comments," Microsoft Product Support Services, http://support.microsoft.com/support/kb/articles/Q159/5/64.asp.

[24] NTBugTraq web site, http://www.ntbugtraq.com/.

[25] NTInfoScan Home Page, http://www.infowar.co.uk/mnemonix/ntinfoscan.htm/.

[26] Vern Paxson, "Empirically-Derived Analytic Models of Wide-Area TCP Connections", IEEE/ACM Transactions on Networking, Vol. 2, No. 4, August, 1994, ftp://ftp.ee.lbl.gov/papers/WAN-TCP-models.ps.Z..

[27] Nicholas Puketza, Mandy Chung, Ronald Olsson, and Biswanath Mukherjee. "A Software Platform for Testing Intrusion Detection Systems," IEEE Software, September/October, 1997.

[28] Nicholas Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee, Ronald Olsson. "A Methodology for Testing Intrusion Detection Systems." Technical report, University of California, Davis, Department of Computer Science, Davis, CA 95616, September 1995.

[29] Rootshell Website. http://www.rootshell.com/, 1999.

[30] Ko, C., M. Ruschitzka, and K. Levitt. "Execution Monitoring of Security-Critical Programs in a Distributed System: A Specifications-Based Approach," In Proceedings 1997 IEEE Symposium on Security and Privacy, pp. 134-144, Oakland, CA: IEEE Computer Society Press.

[31] Samba web site, http://www.samba.org/.

[32] "Sechole Lets Non-administrative Users Gain Debug Level Access to a System Process," Microsoft Product Support Services, http://support.microsoft.com/support/kb/articles/Q190/2/88.ASP?LN=EN-US&SD=gn&FR=0/.

[33] "Security Hole #1," Cybermedia Software Private Limited, http://www.cybermedia.co.in/cspl21/nt_security/Sechole.htm.

[34] Tom Sheldon. *Windows NT Security Handbook*. Osborne McGraw-Hill, 2600 Tenth Street, Berkeley CA, 94710, 1997.

[35] Sun Microsystems, Solaris Security Website. http://www.sun.com/solaris/2.6/ds-security.html. May 1999.

[36] Sysinternals web site, http://www.sysinternals.com/.

[37] Tripwire web site, http://www.tripwire.com/.

[38] G. Vigna and R. Kemmerer, "NetSTAT: A network-based intrusion detection approach", in Proceedings of the 14th Annual Computer Security Applications Conference, Scottsdale, Arizona, December 1998, http://www.cs.ucsb.edu/~kemm/netstat.html/.

[39] Daniel Weber, "A Taxonomy of Computer Intrusions," M.Eng Thesis, MIT Department of Electrical Engineering and Computer Science, June 1998.

[40] Seth Webster. "The Development and Analysis of Intrusion Detection Algorithms.," Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, 02139, 1998.

[41] "Whitehats Max Vision Network Security and Penetration Testing," http://www.whitehats.com/.

[42] "Windows spoofing security bug," http://www.whitehats.com/browsers/b14/b14.html.

[43] Winzip web site, http://www.winzip.com/.