# Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation

Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das

MIT Lincoln Laboratory
244 Wood Street, Lexington, MA 02173-9108, USA
rpl@sst.ll.mit.edu
jhaines@sst.ll.mit.edu

**Abstract.** Eight sites participated in the second DARPA off-line intrusion detection evaluation in 1999. Three weeks of training and two weeks of test data were generated on a test bed that emulates a small government site. More than 200 instances of 58 attack types were launched against victim UNIX and Windows NT hosts. False alarm rates were low (less than 10 per day). Best detection was provided by network-based systems for old probe and old denial-of-service (DoS) attacks and by host-based systems for Solaris user-to-root (U2R) attacks. Best overall performance would have been provided by a combined system that used both host- and network-based intrusion detection. Detection accuracy was poor for previously unseen new, stealthy, and Windows NT attacks. Ten of the 58 attack types were completely missed by all systems. Systems missed attacks because protocols and TCP services were not analyzed at all or to the depth required, because signatures for old attacks did not generalize to new attacks, and because auditing was not available on all hosts.

## 1 Introduction

Computer attacks launched over the Internet are capable of inflicting heavy damage due to increased reliance on network services and worldwide connectivity. It is difficult to prevent attacks by security policies, firewalls, or other mechanisms. System and application software always contains unknown weaknesses or bugs, and complex often unforeseen interactions between software components and/or network protocols are continually exploited by attackers. Intrusion detection systems are designed to detect attacks that inevitably occur despite security precautions.

Discussions of alternate approaches to intrusion detection are available in [1,6,16]. Some approaches detect attacks in real time and can be used to monitor and possibly stop an attack in progress. Others provide after-the-fact forensic information about attacks and can help repair damage, understand the attack mechanism, and reduce the possibility of future attacks of the same type. More advanced intrusion detection systems detect never-before-seen, new, attacks, while the more typical systems detect previously seen, known attacks.

The widespread deployment and high cost of both commercial and government-developed intrusion detection systems has led to an interest in evaluating these systems. Evaluations that focus on algorithm performance are essential for ongoing research. They can contribute to rapid research progress by focusing efforts on difficult technical areas, they can produce common shared corpora or data bases which can be used to benchmark performance levels, and they make it easier for new researchers to enter a field and explore alternate approaches. A review of past intrusion detection evaluations is provided in [11].

The most comprehensive evaluations of intrusion detection systems performed to date were supported by DARPA in 1998 and 1999 [3,11,12]. These evaluations included research intrusion detection systems and attacks against UNIX, Windows NT, and Cisco Routers. They also used a relatively simple network architecture and background traffic designed to be similar to traffic on one Air Force base. The most recent 1999 evaluation included many novel aspects [11]. Both detection and false alarm rates were carefully measured for more than 18 systems. More than 56 attack types included stealthy and novel new attacks were used to measure detection rates and weeks of background traffic were used to measure false alarm rates. In addition, a unique intrusion detection corpus was created that includes weeks of background traffic and hundreds of labeled and documented attacks. This corpus has been widely distributed and is being used as a benchmark for evaluating and developing intrusion detection systems. Both 1998 and 1999 DARPA evaluations included two components. An off-line component produced labeled benchmark corpora that were used simultaneously at many sites to develop and evaluate intrusion detection systems [11,12]. The complementary real-time component [3] assessed only systems that had real-time implementations using fewer attacks and hours instead of weeks of background traffic. The remainder of this paper focuses on the off-line component of the 1999 evaluation. It provides a summary of this research effort, discusses details concerning the motivation and design of background traffic and stealthy attacks, and discusses an analytic approach that can be used to predict whether an intrusion detection system will miss a particular new attack. This paper complements [11], which provides further background and summary results for the 1999 off-line evaluation. Detailed descriptions of attacks in the 1999 evaluation are available in [2,8,9,13,14]. Further details and downloadable corpora are available at [14].

## 2 Overview of the 1999 Evaluation

The 1999 off-line evaluation included three weeks of training data with background traffic and labeled attacks to develop and tune intrusion detection systems and two weeks of test data with background traffic and unlabeled attacks. Techniques originally developed during the 1998 evaluation [12] were extended to more fully analyze system behavior and cover more attack types. Figure 1 shows the isolated test bed network used to generate background traffic and attacks. Scripting techniques, which extend the approaches used in [19], generate live background traffic similar to that which flows between the inside of one Air

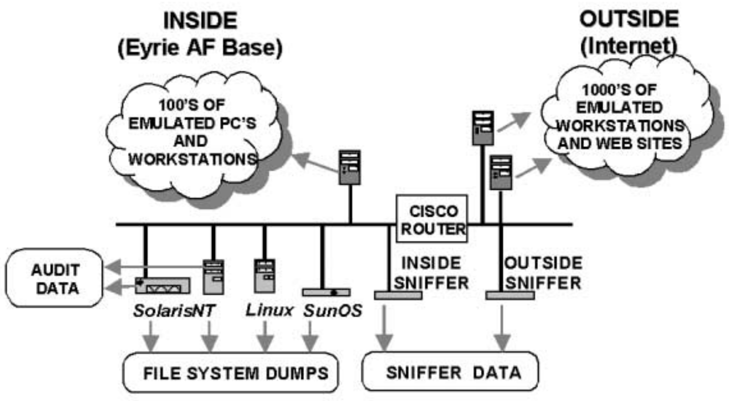


**Fig. 1.** Block diagram of 1999 test bed

Force base and the outside Internet. This approach was selected for the evaluation because hosts can be attacked without degrading operational Air Force systems and because corpora containing background traffic and attacks can be widely distributed without security or privacy concerns. A rich variety of background traffic that looks as if it were initiated by hundreds of users on thousands of hosts is generated in the test bed. The left side of Figure 1 represents the inside of the fictional Eyrie Air Force base created for the evaluations and the right side represents the outside Internet. Automated attacks were launched against four inside UNIX and Windows NT victim machines (Linux 2.0.27, SunOS 4.1.4, Sun Solaris 2.5.1, Windows NT 4.0) and a Cisco 2514 router. More than 200 instances of 58 different attacks were embedded in three weeks of training data and two weeks of test data. Inside and outside machines labeled sniffer in Figure 1 run a program named *tcpdump* [10] to capture all packets transmitted over the attached network segments. This program was customized to open a new output data file after the current active output file size exceeds 1 Gbytes. The status line printed when *tcpdump* was terminated each day never indicated that any packets were dropped. Data collected to evaluate intrusion detection systems include this network sniffing data, Solaris Basic Security Module (BSM) audit data collected from the Solaris host, Windows NT audit event logs collected from the Windows NT host, nightly listings of all files on the four victim machines, and nightly dumps of security-related files on all victim machines.

New features in the 1999 off-line evaluation include the Windows NT victim machine and associated attacks and audit data. These were added due to increased reliance on Windows NT systems by the military. Inside attacks, inside sniffer data to detect these attacks, and stealthy attacks were also added due the dangers posed by inside attacks and an emphasis on sophisticated attackers who can carefully craft attacks to look like normal traffic. In addition, an analysis of

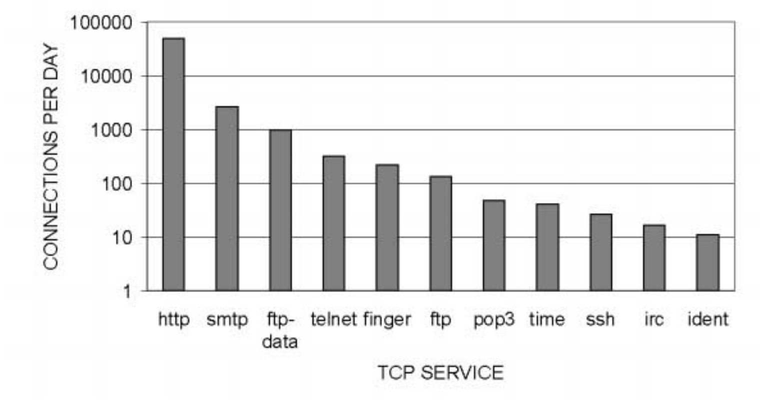


**Fig. 2.** Average connections per day for dominant TCP services

misses and high-scoring false alarms was performed for each system to determine why systems miss specific attacks.

The 1999 evaluation was designed primarily to measure the ability of systems to detect new attacks without first training on instances of these attacks. The previous 1998 evaluation had demonstrated that systems could not detect new attacks well. The new 1999 evaluation was designed to evaluate enhanced systems which can detect new attacks and to analyze why systems miss new attacks. Many new attacks were thus developed and only examples of a few of these were provided in training data.

## 3 Test Bed Network and Background Traffic

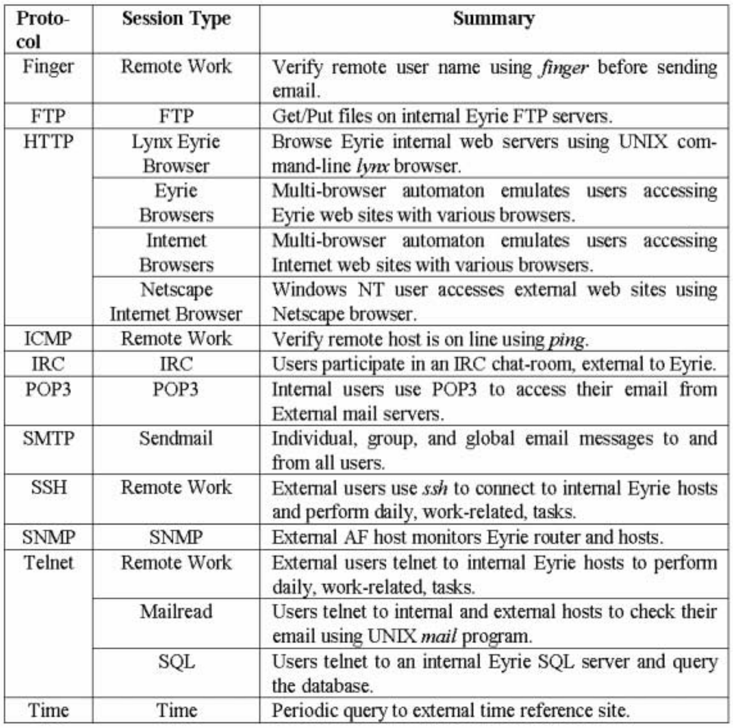
The test bed architecture shown in Figure 1 is a basic foundation that is becoming more complex as the evaluations progress. It was designed to simplify network administration and to support attack and background traffic generation and also instrumentation required to collect input data required by intrusion detection systems. This flat network architecture is not representative of an Air Force base. It is a minimal network designed to support intrusion detection systems that desired to participate in 1998 and 1999, attack types of interest, and most of the network traffic types seen across many Air Force bases. Future evaluations may include more complex networks including firewalls and other protective devices.

Background traffic was generated in the test bed for a variety of reasons. This traffic made it possible to measure baseline false alarm rates of evaluated intrusion detection systems and to deter the development of limited non-robust intrusion detection systems that simply trigger when a particular traffic type occurs. It also led to reasonably high data rates and a fairly rich set of traffic types that exercise traffic handling and analysis capabilities of network analysis and intrusion detection tools tested with evaluation corpora. Finally, the synthesized nature of the traffic allows widespread and relatively unrestricted access to the evaluation corpora. False alarm rates measured with the evaluation corpus may not represent operational false alarm rates at any location. As noted in [17], network traffic varies widely with location and time. This implies that it may be difficult to predict the false alarm rates at operational sites from false alarm rates measured during any evaluation because traffic characteristics, including details that affect false alarm rates, are likely to differ widely from those used in the evaluation. The approach taken in the test bed was to generate realistic traffic that is roughly similar to traffic measured on one Air Force base in early 1998. In addition, details of this traffic (e.g. the frequency of occurrence of words in mail, telnet sessions, and FTP file transfers) were designed to produce false alarm rates similar to operational rates obtained in 1998 using the Air Force ASIM intrusion detection system (ASIM is similar to the Network Security Monitor described in [5]). False alarm rates measured with this traffic can be used to benchmark or compare intrusion detection systems on reference evaluation background traffic corpora. They may not, however, be representative of false alarm rates on operational data. Supplementary measurements using restricted-access data are necessary to determine operational false alarm rates. Traffic characteristics of test bed background traffic that were similar to characteristics of measured Air Force base traffic include the following:

1. The overall traffic level in connections per day.
2. The number of connections per day for the dominant TCP services.
3. The identity of many web sites that are visited from internal users.
4. The average time-of-day variation of traffic as measured in 15-minute intervals.
5. The general purpose of telnet sessions.
6. The frequency of usage of UNIX commands in telnet sessions.
7. The use of the UNIX *time* command to obtain an accurate remote time reference.
8. The frequency of occurrence of ASIM keywords in telnet sessions, mail messages, and files downloaded using FTP.
9. The frequency of occurrence of users mistyping their passwords.
10. Inclusion of an SQL database server that starts up automatically after a user telnets to remote server.

Custom software automata in the test bed simulate hundreds of programmers, secretaries, managers, and other types of users running common UNIX and Windows NT application programs. Automata interact with high-level user application programs such as *Netscape*, *lynx*, *mail*, *ftp*, *telnet*, *ssh*, *irc*, and *ping* or they implement clients for network services such as HTTP, SMTP, and POP3. Low-level TCP/IP protocol interactions are handled by kernel software and are not simulated. The average number of background-traffic bytes transmitted per day between the inside and outside of this test bed is roughly 411 Mbytes per day, with most of the traffic concentrated between 8:00 AM and 6:00 PM. The dominant protocols are TCP (384 Mbytes), UDP (26 Mbytes), and ICMP (98

**Table 1.** Major types of network services and automaton session types generated to create background traffic in the test bed



| Proto-col | Session Type | Summary |
|---|---|---|
| Finger | Remote Work | Verify remote user name using *finger* before sending email. |
| FTP | FTP | Get/Put files on internal Eyrie FTP servers. |
| HTTP | Lynx Eyrie Browser | Browse Eyrie internal web servers using UNIX command-line *lynx* browser. |
| | Eyrie Browsers | Multi-browser automaton emulates users accessing Eyrie web sites with various browsers. |
| | Internet Browsers | Multi-browser automaton emulates users accessing Internet web sites with various browsers. |
| | Netscape Internet Browser | Windows NT user accesses external web sites using Netscape browser. |
| ICMP | Remote Work | Verify remote host is on line using *ping*. |
| IRC | IRC | Users participate in an IRC chat-room, external to Eyrie. |
| POP3 | POP3 | Internal users use POP3 to access their email from External mail servers. |
| SMTP | Sendmail | Individual, group, and global email messages to and from all users. |
| SSH | Remote Work | External users use *ssh* to connect to internal Eyrie hosts and perform daily, work-related, tasks. |
| SNMP | SNMP | External AF host monitors Eyrie router and hosts. |
| Telnet | Remote Work | External users telnet to internal Eyrie hosts to perform daily, work-related, tasks. |
| | Mailread | Users telnet to internal and external hosts to check their email using UNIX *mail* program. |
| | SQL | Users telnet to an internal Eyrie SQL server and query the database. |
| Time | Time | Periodic query to external time reference site. |

Kbytes). These traffic rates are low compared to current rates at some large commercial and academic sites. They are representative of 1998 Air Force data and they also lead to sniffed data file sizes that can still be transported over the Internet without practical difficulties. Figure 2 shows the average number of connections per day for the most common TCP services. As can be seen, web traffic dominates but many other types of traffic are generated which use a variety of services.

Table 1 shows the many types of user sessions generated by automata and the types of network traffic these sessions create. As can be seen, user automata send and receive mail, browse web sites, send and receive files using the FTP protocol, use *telnet* and *ssh* to log into remote computers and perform work, monitor the router remotely using SNMP, and perform other tasks. For example, Table 1 shows that four different automata are used to generate HTTP traffic. The *lynx* command-line browser is used during telnet and console sessions to access internal Eyrie web sites, a multi-browser automaton which emulates many types of browsers including Netscape Navigator and Internet Explorer is used to browse both internal and external web sites, and a JavaScript browser that runs

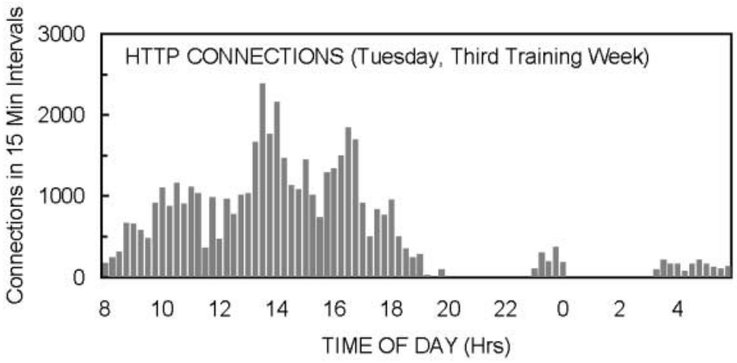


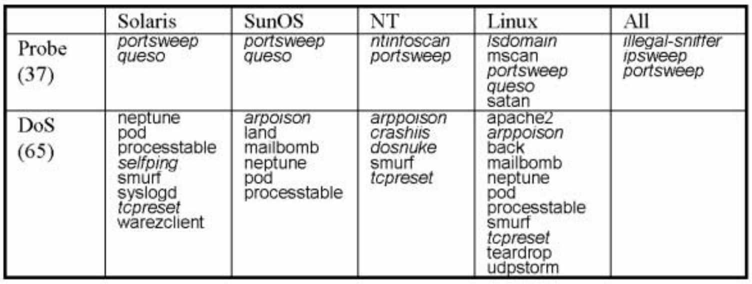
**Fig. 3.** Number of HTTP connections measured in 15 minute intervals generated by the four types of web automaton during Tuesday of the third week of training

inside Netscape Navigator browses external web sites from the internal Windows NT host.

Table 1 also shows that three automata are used to generate telnet sessions. First, remote programmers, secretaries, and administrators connect into internal Eyrie machines to work throughout the day using telnet or SSH. Characteristics of these work sessions including the frequency of occurrence of different UNIX commands issued, files accessed, the number of sessions per day, and the start time and duration of sessions are assigned probabilistically depending on the user type. A second telnet automaton simulates users who telnet to hosts to read and respond to mail using the UNIX *mail* program. The final telnet automaton simulates users who access an SQL database on an internal database machine. This machine automatically opens an SQL database server program, instead of a shell, after successful telnet logins. In addition to automatic traffic, the test bed allows human actors to generate background traffic and attacks when the traffic or attack is too complex to automate. For example, human actors performed attacks that included remote X-Windows Netscape browser displays.

Traffic varies over each simulation day to produce roughly the same average overall traffic rates in 15-minute intervals as measured in one week of operational Air Force traffic. Figure 3 shows the number of HTTP connections generated by the four browsing automata from Table 1 in one day of test bed traffic. Start times of browsing sessions are chosen using a Poisson process model with a time-dependent rate parameter and times between browsing actions within a session also have independent exponential distributions. Each browsing session accesses from 1 to 50 web pages. The model of human typing provided in expect is used for typing responses in telnet and other sessions where users normally provide responses from a keyboard. As can be seen in Figure 3, traffic rates are highest during the middle of the 8:00 AM to 6:00 PM workday and low after these hours. These plots vary with time in a similar manner for telnet and other session types, except the maximum number of sessions is scaled down.

**Table 2.** Probe and Denial of Service (DoS) attacks



| | Solaris | SunOS | NT | Linux | All |
|---|---|---|---|---|---|
| Probe (37) | *portsweep*<br>*queso* | *portsweep*<br>*queso* | *ntinfoscan*<br>*portsweep* | *lsdomain*<br>mscan<br>*portsweep*<br>*queso*<br>satan | *illegal-sniffer*<br>*ipsweep*<br>*portsweep* |
| DoS (65) | neptune<br>pod<br>processtable<br>*selfping*<br>smurf<br>syslogd<br>*tcpreset*<br>warezclient | *arpoison*<br>land<br>mailbomb<br>neptune<br>pod<br>processtable | *arppoison*<br>*crashiis*<br>*dosnuke*<br>smurf<br>*tcpreset* | apache2<br>*arppoison*<br>back<br>mailbomb<br>neptune<br>pod<br>processtable<br>smurf<br>*tcpreset*<br>teardrop<br>udpstorm | |

## 4 Attacks

Twelve new Windows NT attacks were added in 1999 along with stealthy versions of many 1998 attacks, new inside console-based attacks, and six new UNIX attacks. The 56 different attack types shown in Tables 2 and 3 were used in the evaluation. Attacks in normal font in these tables are old attacks from 1998 executed in the clear (114 instances). Attacks in italics are new attacks developed for 1999 (62 instances), or stealthy versions of attacks used in 1998 (35 instances). Details on attacks including further references and information on implementations are available in [2,8,9,13,14]. Five major attack categories and the attack victims are shown in Tables 2 and 3. Primary victims listed along the top of these tables are the four inside victim hosts, shown in the gray box of Figure 1, and the Cisco router. In addition, some probes query all machines in a given range of IP addresses as indicated by the column labeled "all" in Table 2.

The upper row of Table 2 lists probe or scan attacks. These attacks automatically scan a network of computers or a DNS server to find valid IP addresses (ipsweep, lsdomain, mscan), active ports (portsweep, mscan), host operating system types (queso, mscan), and known vulnerabilities (satan). All of these probes except two (mscan and satan) are either new in 1999 (e.g. ntinfoscan, queso, illegalsniffer) or are stealthy versions of 1999 probes (e.g. portsweep, ipsweep). Probes are considered stealthy if they issue ten or fewer connections or packets or if they wait longer than 59 seconds between successive network transmissions. The new "illegal-sniffer" attack is different from the other probes. During this attack, a Linux sniffer machine is installed on the inside network running the *tcpdump* program in a manner that creates many DNS queries from this new and illegal IP address.

The second row of Table 2 contains denial of service (DoS) attacks designed to disrupt a host or network service. New 1999 DoS attacks crash the Solaris operating system (selfping), actively terminate all TCP connections to a specific host (tcpreset), corrupt ARP cache entries for a victim not in others caches (arppoison), crash the Microsoft Windows NT web server (crashiis), and crash Windows NT (dosnuke).

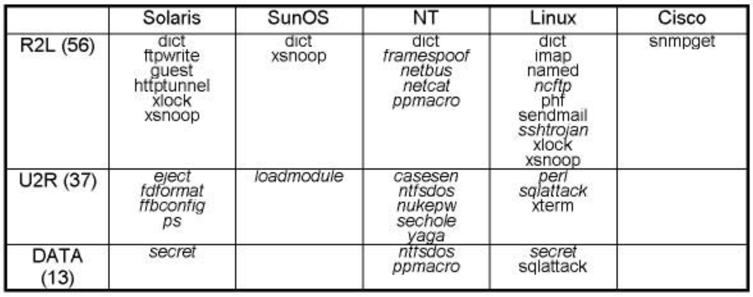
The first row of Table 3 contains Remote to Local (R2L) attacks. In these attacks, an attacker who does not have an account on a victim machine gains local access to the machine (e.g. guest, dict), exfiltrates files from the machine (e.g. ppmacro), or modifies data in transit to the machine (e.g. framespoof). New 1999 R2L attacks include an NT PowerPoint macro attack (ppmacro), a man-in-the middle web browser attack (framespoof), an NT trojan-installed remote-administration tool (netbus), a Linux trojan SSH server (sshtrojan), and a version of a Linux FTP file access-utility with a bug that allows remote commands to run on a local machine (ncftp).

The second row of Table 3 contains user to root (U2R) attacks where a local user on a machine is able to obtain privileges normally reserved for the UNIX super user or the Windows NT administrator. All five NT U2R attacks are new this year and all other attacks except one (xterm) are versions of 1998 UNIX U2R attacks that were redesigned to be stealthy to network-based intrusion detection systems evaluated in 1998. These stealthy attacks are described below. The bottom row in Table 3 contains Data attacks. The goal of a Data attack is to exfiltrate special files, which the security policy specifies should remain on the victim hosts. These include "secret" attacks where a user who is allowed to access the special files exfiltrates them via common applications such as mail or FTP, and other attacks where privilege to access the special files is obtained using a U2R attack (ntfsdos, sqlattack). Note that an attack could be labeled as both a U2R and a Data attack if one of the U2R attacks was used to obtain access to the special files. The "Data" category thus specifies the goal of an attack rather than the attack mechanism.

## 4.1 Stealthy U2R Attacks

UNIX U2R attacks were made stealthy to network-based intrusion detection systems using a variety of techniques designed to hide attack-specific keywords from network-based sniffers [2,13]. Most stealthy U2R attacks included the components shown by the five columns in Figure 4. Attack scripts were first encoded, transported to the victim machine, and then decoded and executed. Actions such as altering or accessing secret or security-related files were performed and the attacker then removed files created for the attack and restored original permissions of altered or accessed files to clean up. The dark filled in actions in Figure 4 show one particular stealthy attack. In this attack, the clear-text attack script is encoded by "character stuffing" where extra unique characters (e.g. "AA") are added after every original character, the attack script is transported to the victim machine using FTP, the attack script is decoded using vi (not shown, but implicit), attack execution is hidden by generating screens full of chaff text directed to the standard output from a background process, and the attacker changes file permission on a secret file, displays the file, and then restores file permissions back to original settings and erases the attack script. As can be seen from Figure 4, there are many other possible variants of stealthy attacks. Five approaches were used to encode/decode and transport attack scripts and

**Table 3.** Remote to Local (R2L), User to Root (U2R), and Data attacks



| | Solaris | SunOS | NT | Linux | Cisco |
|---|---|---|---|---|---|
| R2L (56) | dict<br>ftpwrite<br>guest<br>httptunnel<br>xlock<br>xsnoop | dict<br>xsnoop | dict<br>*framespoof*<br>*netbus*<br>*netcat*<br>*ppmacro* | dict<br>imap<br>named<br>*ncftp*<br>phf<br>sendmail<br>*sshtrojan*<br>xlock<br>xsnoop | snmpget |
| U2R (37) | *eject*<br>*fdformat*<br>*ffbconfig*<br>*ps* | *loadmodule* | *casesen*<br>*ntfsdos*<br>*nukepw*<br>*sechole*<br>*yaga* | *perl*<br>*sqlattack*<br>xterm | |
| DATA (13) | *secret* | | *ntfsdos*<br>*ppmacro* | *secret*<br>sqlattack | |

to execute these scripts. The encode action "Octal Characters" refers to encoding binary files using the C *printf* octal backslash notation and then decoding the binary file using the *tcsh* builtin *echo* command. The execute action "Shell Variables" refers to encoding shell commands using shell variables to obscure the commands that are issued. The execute action "Delay Execution" refers to using *cron* or *at* to execute scripts at a later time after the session that created the attack script and "Multiple Sessions" refers to downloading, decoding, and running the attack script over multiple sessions. Further details and examples of other actions are available in [2,13].

Stealthy techniques that rely on packet fragmentation and other forms of packet manipulation [18] were implemented as part of the 1999 evaluation. Time constraints and the variety of victim operating systems used precluded extensive experimentation with these approaches. Preliminary exploratory results are provided in [2].

## 5 Participants and Scoring

Eight research groups participated in the evaluation using a variety of approaches to intrusion detection. Papers by these groups describing high-performing systems are provided in [4,7,15,20,21,22,23]. One requirement for participation in the evaluation was the submission of a detailed system description that was used for scoring and analysis. System descriptions described the types of attacks the system was designed to detect, data sources used, features extracted, and whether optional attack identification information was provided as an output. Most systems used network sniffer data to detect Probe and DoS attacks against all systems [7,15,21,23] or BSM Solaris host audit data to detect Solaris R2L and U2R attacks [4,15,23]. Two systems produced a combined output from both network sniffer data and host audit data [15,23]. A few systems used network sniffer data to detect R2L and U2R attacks against the UNIX victims [15,23]. One system used NT audit data to detect U2R and R2L attacks against the Windows NT victim [20] and two systems used BSM audit data to detect Data attacks against the Solaris victim [15,23]. A final system used information from a nightly

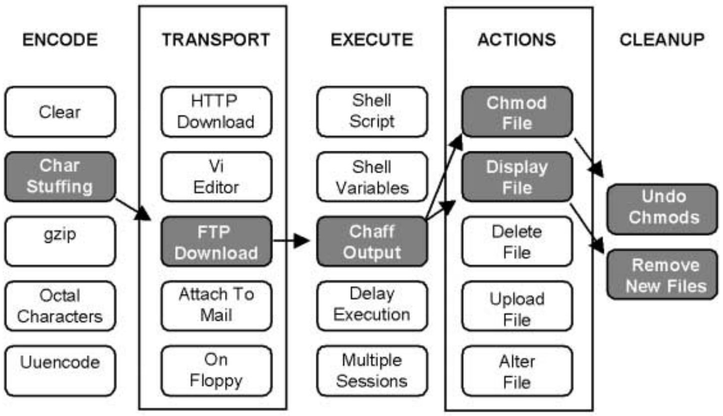


**Fig. 4.** Possible paths to generate stealthy user to root (U2R) attacks. Each attack requires selection of one or more of the alternate approaches shown in each column

file system scan to detect R2L, U2R, and Data attacks against the Solaris victim [22]. The software program that performs this scan was the only custom auditing tool used in the evaluation. A variety of approaches were employed including expert systems that use rules or signatures to detect attacks, anomaly detectors, pattern classifiers, recurrent neural networks, data mining techniques, and a reasoning system that performs a forensic analysis of the Solaris file system.

Three weeks of training data, composed of two weeks of background traffic with no attacks and one week of background traffic with a few attacks, were provided to participants from mid May to mid July 1999 to support system tuning and training. Only five weekdays of traffic were provided for each week. Locations of attacks in the training data were clearly labeled. Two weeks of unlabeled test data were provided from late September to the middle of October. Participants downloaded this data from a web site, processed it through their intrusion detection systems, and generated putative hits or alerts at the output of their intrusion detection systems. Lists of alerts were due back by early October. In addition, participants could optionally return more extensive identification lists for each attack.

A simplified approach was used in 1999 to label attacks and score alerts and new scoring procedures were added to analyze the optional identification lists. In 1998, every network TCP/IP connection, UDP packet, and ICMP packet was labeled, and participants determined which connections and packets corresponded to attacks. Although this approach pre-specifies all potential attack packets and thus simplifies scoring and analysis, it can make submitting alerts difficult because aligning alerts with the network connections and packets that generate alerts is often complex. In addition, this approach cannot be used with inside attacks that generate no network traffic. In 1999, a new simplified approach was adopted. Each alert only had to indicate the date, time, victim IP address, and score for each putative attack detection. An alert could also optionally indicate the attack category. This was used to assign false alarms to attack categories. Putative detections returned by participants were counted as true "hits" or true detections if the time of any alert occurred during the time of any attack segment and the alert was for the correct victim IP address. Alerts that occur outside all attack segments were counted as "misses" or false alarms. Attack segments generally correspond to the duration of all network packets and connections generated by an attack and to time intervals when attack processes are running on a victim host. To account for small timing inconsistencies across hosts, an extra 60 seconds leeway was typically allowed for alerts before and after the end of each attack segment. The analysis of each system only included attacks which that system was designed to detect, as specified in the system description. Systems werent penalized for missing attacks they were not designed to detect and false alarms that occurred during segments of out-of-spec attacks were ignored.

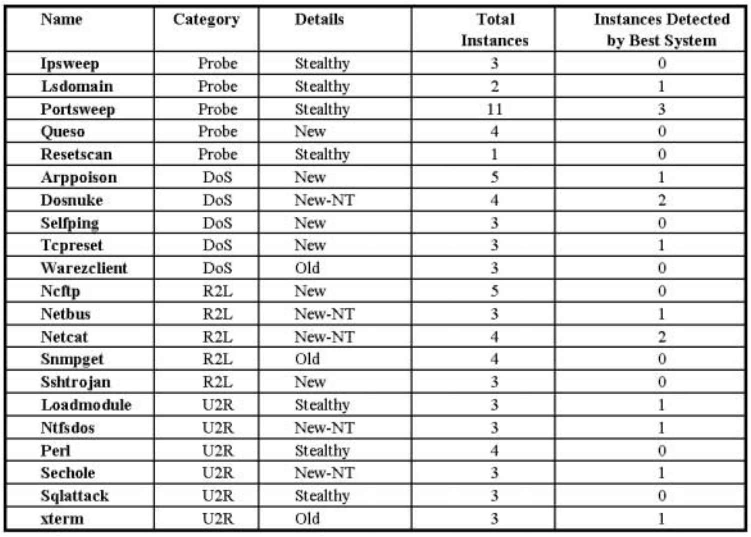
The score produced by a system was required to be a number that increases as the certainty of an attack at the specified time increases. All participants returned numbers ranging between zero and one, and many participants produced binary outputs (0s and 1s only). If alerts occurred in multiple attack segments of one attack, then the score assigned to that attack for further analysis was the highest score in all the alerts. Some participants returned optional identification information for attacks. This included the attack category, the name for old attacks selected from a list of provided names, and the attack source and destination IP addresses, start time, duration, and the ports/services used. This information was analyzed separately from the alert lists used for detection scoring. Results in this paper focus on detection results derived from the required alert lists. Information on identification results is provided in [11].

Attack labels were used to designate attack segments in the training data and also to score lists of alerts returned by participants. Attack labels were provided using list files similar to those used in 1998, except a separate list file was provided for each attack specifying all segments of that attack. Entries in these list files include the date, start time, duration, a unique attack identifier, the attack name, source and destination ports and IP addresses, the protocol, and details concerning the attack. Details include indications that the attack is clear or stealthy, old or new, inside or outside, the victim machine type, and whether traces of the attack occur in each of the different data types that were collected. Attack list files are available at [14].

## 6 Results

An initial analysis was performed to determine how well all systems taken together detect attacks regardless of false alarm rates. The best system was selected for each attack as the system that detects the most instances of that attack. The detection rate for these best systems provides a rough upper bound on compos-

**Table 4.** Poorly detected attacks where the best system for each attack detects half or fewer of the attack instances



| Name | Category | Details | Total Instances | Instances Detected by Best System |
|---|---|---|---|---|
| **Ipsweep** | Probe | Stealthy | 3 | 0 |
| **Lsdomain** | Probe | Stealthy | 2 | 1 |
| **Portsweep** | Probe | Stealthy | 11 | 3 |
| **Queso** | Probe | New | 4 | 0 |
| **Resetscan** | Probe | Stealthy | 1 | 0 |
| **Arppoison** | DoS | New | 5 | 1 |
| **Dosnuke** | DoS | New-NT | 4 | 2 |
| **Selfping** | DoS | New | 3 | 0 |
| **Tcpreset** | DoS | New | 3 | 1 |
| **Warezclient** | DoS | Old | 3 | 0 |
| **Ncftp** | R2L | New | 5 | 0 |
| **Netbus** | R2L | New-NT | 3 | 1 |
| **Netcat** | R2L | New-NT | 4 | 2 |
| **Snmpget** | R2L | Old | 4 | 0 |
| **Sshtrojan** | R2L | New | 3 | 0 |
| **Loadmodule** | U2R | Stealthy | 3 | 1 |
| **Ntfsdos** | U2R | New-NT | 3 | 1 |
| **Perl** | U2R | Stealthy | 4 | 0 |
| **Sechole** | U2R | New-NT | 3 | 1 |
| **Sqlattack** | U2R | Stealthy | 3 | 0 |
| **xterm** | U2R | Old | 3 | 1 |

ite system performance. Thirty seven of the 58 attack types were detected well by this composite system, but many stealthy and new attacks were always or frequently missed. Poorly detected attacks for which half or more of the attack instances were not detected by the best system are listed in Table 4. This table lists the attack name, the attack category, details concerning whether the attack is old, new, or stealthy, the total number of instances for this attack, and the number of instances detected by the system which detected this attack best. Table 4 contains 21 attack types and is dominated by new attacks and attacks designed to be stealthy to 1998 network-based intrusion detection systems. All instances of 10 of the attack types in Table 4 were totally missed by all systems. These results suggest that the new systems developed for the 1999 evaluation still are not detecting new attacks well and that stealthy probes and U2R attacks can avoid detection by network-based systems.

Further analyses evaluated system performance at false alarm rates in a specified range. The detection rate of each system at different false alarm rates can be determined by lowering a threshold from above 1.0 to below 0.0, counting the detections with scores above the threshold as hits, and counting the number of alerts above the threshold that do not detect attacks as false alarms. This results in one or more operating points for each system which trade off false alarm rate against detection rate. It was found that almost all systems, except some anomaly detection systems, achieved their maximum detection accuracy at or below 10 false alarms per day on the 1999 corpus. These low false alarm rates were presumably due to the low overall traffic volume, the relative stationarity of the traffic, and the ability to tune systems to reduce false alarms on three weeks of training data. In the remaining presentation, the detection rate reported for each system is the highest detection rate achieved at or below 10 false alarms per day on the two weeks of test data.

Table 5 shows average detection rates at 10 false alarms per day for each attack category and victim type. This table provides overall results and does not separately analyze old, new, and stealthy attacks. The upper number in a cell, surrounded by dashes, is the number of attack instances in that cell and the other entries provide the percent correct detections for all systems with detection rates above 40% in that cell. A cell contains only the number of instances if no system detected more than 40% of the instances. Only one entry is filled for the bottom row because only probe attacks were against all the victim machines and the SunOS/Data cell is empty because there were no Data attacks against the SunOS victim. High-performance systems listed in Table 5 include rule-based expert systems that use network sniffing data and/or Solaris BSM audit data (Expert-1 through Expert-3 [15,23,21]), a data mining system that uses network sniffing data (Dmine [7]), a pattern classification approach that uses network sniffing data (Pclassify), an anomaly detection system which uses recurrent neural networks to analyze system call sequences in Solaris BSM audit data (Anomaly [4]), and a reasoning system which performs a nightly forensic analysis of the Solaris file system (Forensics [22]).

No one approach or system provides the best performance across all categories. The best performance is provided for probe and denial of service attacks for systems that use network sniffer data and for U2R and Data attacks against the Solaris victim for systems that use BSM audit data. Detection rates for U2R and Data attacks are generally poor for SunOS and Linux victims where extensive audit data is not available. Detection rates for R2L, U2R, and Data attacks are poor for Windows NT, which was included in the evaluation for the first time this year.

Figure 5 shows the performance of the best intrusion detection system in each attack category at a false alarm rate of 10 false alarms per day. The left chart compares the percentage of attack instances detected for old-clear and new attacks and the right chart compares performance for old-clear and stealthy attacks. The numbers in parentheses on the horizontal axis below the attack category indicate the number of instances of attacks of different types. For example, in Figure 3A, there were 49 oldclear and 15 new denial-of-service attacks. Figure 3A demonstrates that detection of new attacks was much worse than detection of old-clear attacks across all attack categories, and especially for DoS, R2L, and U2R attacks. The average detection rate for old-clear attacks was 72% and this dropped to 19% for new attacks. Figure 3B demonstrates that stealthy probes and U2R attacks were much more difficult to detect for network-based intrusion detection systems that used sniffing data. User-to-root attacks against the Solaris victim, however, were accurately detected by host-based intrusion detection systems that used BSM audit data.

Attacks are detected best when they produce a consistent "signature," trace, or sequence of events in tcpdump data or in audit data that is different from

**Table 5.** Percent attack instances detected for systems with a detection rate above 40% in each cell and at false alarm rates below 10 false alarms per day

| | DoS | Probe | R2L | U2R | Data |
|---|---|---|---|---|---|
| Solaris | -19- Expert-1: 63% Expert-2: 53% | -5- Expert-2: 60% Expert-3: 50% | -12- Expert-1: 50% Forensics: 50% | -11- Expert-1: 100% Expert-2: 100% Anomaly: 100% Forensics: 73% | -6- Expert-2: 100% Forensics: 83% |
| NT | -16- Expert-1: 69% Expert-2: 69% | -5- Expert-1: 80% Expert-2: 60% | -12- | -13- | -5- |
| SunOS | -8- DMine: 88% Expert-1: 63% Expert-2: 50% | -5- PClassify: 60% | -3- Expert-2: 67% | -3- | |
| Linux | -19- Expert-1: 84% DMine: 74% Expert-2: 68% | -8- Expert-3: 60% DMine: 50% | -25- Expert-2: 64% Expert-1: 44% | -10- | -4- |
| All | | -11- Expert-1: 46% | | | |

sequences produced for normal traffic. A detailed analysis by participants demonstrated that attacks were missed for a variety of reasons. Systems which relied on rules or signatures missed new attacks because signatures did not exist for these attacks, and because existing signatures did not generalize to variants of old attacks, or to new and stealthy attacks. For example ncftp a ls_domain attacks were visible in tcpdump data, but were missed because no rules existed to detect these attacks. Stealthy probes were missed because hard thresholds in rules were set to issue an alert only for more rapid probes, even though slow probes often provided as much information to attackers. These thresholds could be changed to detect stealthy probes at the expense of generating more false alarms. Stealthy U2R attacks were missed by network-based systems because attack actions were hidden in sniffer data and rules generated for clear versions of these attacks no longer applied. Many of the Windows NT attacks were missed due to lack of experience with Windows NT audit data and attacks. A detailed analysis of the Windows NT attacks [9] indicated that all but two of these attacks (ppmacro, framespoof) can be detected from the 1999 NT audit data using attack-specific signatures which generate far fewer than 10 false alarms per day.

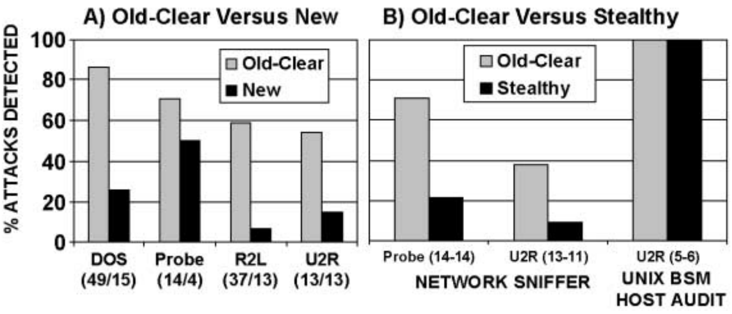


**Fig. 5.** Comparison of detection accuracy at 10 false alarms per day for (A) Old-Clear versus New attacks and (B) Old-Clear versus stealthy attacks

## 7 Predicting when New Attacks will Be Detected

Many network sniffer-based intrusion detection systems missed attacks because particular protocols or services were not monitored or because services were not analyzed to the required depth. This is illustrated in Figure 6. The horizontal axis in this figure shows the protocols and services that were used for many of the probe and DoS attacks and the vertical axis shows the depth of analysis required to reliably determine the action performed by an attack. Attacks near the top of Figure 6 require only lowlevel analysis of single or multiple packet headers. Attacks near the bottom of Figure 6 require understanding of the protocol used to extract the connection content and highlevel analysis of the content to determine the action performed. Well-known attacks can be detected at lower levels than shown when the attack produces a signature or trace at a lower level that is unique from background traffic. This approach is used in most signature-based intrusion detection systems. Determining the intended action of a new attack, however, requires the depth of analysis shown.

Attack names surrounded by white ovals in Figure 6 were detected well, while attacks surrounded by dark ovals were not. For example, many systems missed the "ARP Poison" attack on the bottom left because the ARP protocol was not monitored or because the attackers duplicate responses to arp-who-has requests were not detected. Many systems also missed the Illegal Sniffer and LS_DOMAIN attacks on the left middle because the DNS service was not monitored or because DNS traffic was not analyzed to determine either when an "ls" command is successfully answered by a DNS server or when a DNS request is sent from a new IP address. Many systems also missed the "SELF-PING" attack because telnet sessions were not reconstructed and commands issued in telnet sessions were not analyzed. Many of the attacks that were detected well required simpler high-level analysis of packet headers. For example, the "LAND" attack includes a UDP packet with the same source and destination IP address and the "TEARDROP" attack includes a mis-fragmented UDP packet. Other attacks that were detected well required sequential analysis of multiple packets

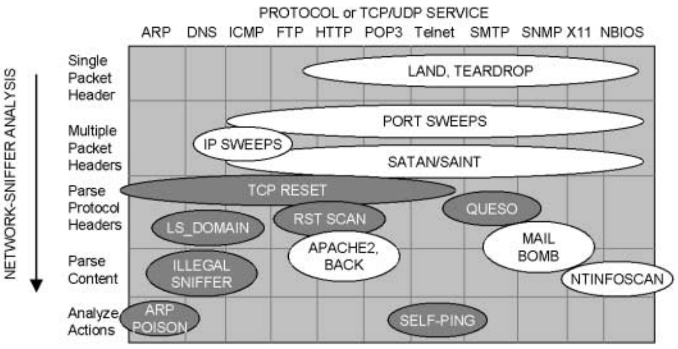


**Fig. 6.** Probe and DoS attacks displayed to show the services and protocols used and the maximum depth of analysis of network traffic required to reliably detect attacks. Attacks in white ovals were detected well by network-based systems, attacks in dark ovals were not

or deeper analysis of a particular protocol. For example, the "SATAN" and "NTINFOSCAN" attacks include a large variety of connections that occur in a short time interval, as do non-stealthy "IP SWEEPS" and "PORT SWEEPS".

The attack analysis shown in Figure 6 illustrates how two pieces of information are required to predict whether a new attack will be missed by network-based systems. Evidence of the attack or knowledge of where the attack manifests itself in data sources and also knowledge of input data and features used by the intrusion detection system are required. This analysis can be extended to other types of attacks and to host-based systems by analyzing the evidence an attack leaves on the victim host in audit records, log files, file system access times, and other locations. The general rule is that attacks will be missed if no evidence of the attack is available in data analyzed by the intrusion detection system or if necessary features are not extracted from this data. This may occur for many reasons. The required host-based data may not be available, network sensors may be in the wrong location to record attack trace components, required protocols or services may not be analyzed, a new attack may require a novel type of feature extraction which is not yet included, or a stealthy attack may leave no traces in information analyzed. If traces of an attack are processed by an intrusion detection system, then the attack may or may not be detected. Performance depends on the overlap with normal input features and details of the intrusion detection system. The analysis described above requires attack trace information and detailed intrusion detection system descriptions. It can be used as a preliminary analysis to determine which attacks an intrusion detection system may detect and can reduce the necessity of expensive experimentation. Network attack traces and system descriptions are available on the Lincoln Laboratory web site and included as part of the 1999 DARPA Intrusion Detection Evaluation corpus [14]. The traces list all network packets generated by each attack instance.

## 8 Discussion

The DARPA 1999 off-line intrusion detection evaluation successfully evaluated 18 intrusion detection systems from 8 sites using more than 200 instances of 58 attack types embedded in three weeks of training data and two weeks of test data. Attacks were launched against UNIX and Windows NT hosts and a Cisco router. Best detection was provided by network-based systems for old probe and old denial of service attacks and by host-based systems for Solaris user-to-root attacks launched either remotely or from the local console. A number of sites developed systems that detect known old attacks by searching for signatures in network sniffer data or Solaris BSM audit data using expert systems or rules. These systems detect old attacks well when they match known signatures, but miss many new UNIX attacks, Windows NT attacks, and stealthy attacks. Promising capabilities were provided by Solaris host-based systems which detected console-based and remote-stealthy U2R attacks, by anomaly detection systems which could detect some U2R and DoS attacks without requiring signatures, and by a host-based system that could detect Solaris U2R and R2L attacks without using audit information but by performing a forensic analysis of the Solaris file system.

A major result of the 1998 and 1999 evaluations is that current research intrusion detection systems miss many new and stealthy attacks. Despite the focus in 1999 on developing approaches to detect new attacks, all systems evaluated in 1999 completely missed 10 out of 58 attack types and, even after combining output alerts from all systems, 23 attack types were detected poorly (half or fewer instances of an attack type detected). Detailed analyses of individual systems indicated that attacks were missed for many reasons. Input data sources that contained evidence of attacks were sometimes not analyzed or they werent analyzed to the required depth and rules, thresholds, or signatures created for old attacks often did not generalize to new attacks. This result is relatively independent of evaluation details because it depends only on attack traces and an analysis of why attacks were missed and how systems operate. An analysis of why attacks were missed suggested an analytic approach that can be used to predict whether an intrusion detection system will miss a particular new attack. It requires detailed attack traces and system descriptions to determine whether components of attack traces are contained in the inputs to an intrusion detection system and whether necessary features are extracted from these inputs. This analytic approach may be useful for designing future evaluations and reducing the need for experimentation.

False alarm rate results of the 1999 evaluation should be interpreted within the context of the test bed and background traffic used. The evaluation used a simple network topology, a non-restrictive security policy, a limited number of victim machines and intrusion detection systems, stationary and low-volume background traffic, lenient scoring, and extensive instrumentation to provide inputs to intrusion detection systems. Most systems had low false alarm rates (well below 10 false alarms per day). As noted above, these low rates may be caused by the use of relatively low volume background traffic with a time varying, but relatively fixed proportion of different traffic types and by the availability of training data to tune or train systems.

Extensions to the current evaluation are planned to verify false alarm rates using operational network traffic and a small number of high-performing systems. Operational measurements will also be made to update traffic statistics and traffic generators used in the test bed. Further evaluations are also required to explore performance with commercial and updated research intrusion detection systems, with more complex network topologies, with a wider range of attacks, and with more complex background traffic. In addition, other approaches to making attacks stealthy should be explored including low-level packet modifications (e.g. [18]) and attacks which remove evidence from Windows NT and Solaris BSM audit records and other system audit logs before terminating.

Comprehensive evaluations of DARPA research systems have now been performed in 1998 and 1999. These evaluations take time and effort on the part of the evaluators and the participants. They have provided benchmark measurements that do not now need to be repeated again until system developers are able to implement many desired improvements. The current planned short-term focus in 2000 is to provide assistance to intrusion detection system developers to advance their systems and not to evaluate performance. System development can be expedited by providing descriptions, traces, and labeled examples of many new attacks, by developing threat and attack models, and by carefully evaluating COTS systems to determine where to focus research efforts.

A number of research directions are suggested by 1999 results. First, researchers should focus on anomaly detection and other approaches that have the potential of detecting new attacks. Second, techniques should be developed to process Windows NT audit data. Third, host-based systems shouldnt rely exclusively on C2-level audit data such as Solaris BSM data or NT audit data. Instead other types of host and network input features should also be explored. These could be provided by new system auditing software, by firewall or router audit logs, by SNMP queries, by software wrappers, by commercial intrusion detection system components, by forensic analysis of file-system changes as in [22], or by application-specific auditing. Fourth, research efforts should not overlap but should provide missing functionality. Finally, a greater breadth of analysis is required including a wider range of protocols and services.

## Acknowledgements

This work was sponsored by the Department of Defense Advanced Research Projects Agency under Air Force Contract F19628-95-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force. We would like to thank Sami Saydjari for supporting this effort. Many involved participants made this evaluation possible including Dick Kemmerer, Giovanni Vigna, Mabri Tyson, Phil Porras, Anup Ghosh, R. C. Sekar, and NingNing Wu. We would also like to thank Terry Champion and Steve Durst from AFRL for providing Linux kernel modifications that make one host simulate many IP addresses. Finally, we would like to thank others who contributed including Marc Zissman, Rob Cunningham, Seth Webster, Kris Kendall, Raj Basu, Jesse Rabek, and Simson Garfinkel.

## References

1. E. G. Amoroso, Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response, Intrusion.Net Books, 1999. 162
2. K. Das, The Development of Stealthy Attacks to Evaluate Intrusion Detection Systems, S. M. Thesis, MIT Department of Electrical Engineering and Computer Science, June 2000. 163, 169, 170, 171
3. R. Durst, Terrence Champion, Brian Witten, Eric Miller and Luigi Spagnuolo, Testing and evaluating computer intrusion detection systems, Communications of the ACM, 42, 1999, 53-61. 163
4. A. K. Ghosh and A. Schwartzbard, A Study in Using Neural Networks for Anomaly and Misuse Detection, in Proceedings of the USENIX Security Symposium, August 23-26, 1999, Washington, D.C, http://www.rstcorp.com/˜anup. 171, 175
5. T. Heberlein, T., Network Security Monitor (NSM) - Final Report, U. C. Davis: February 1995, http://seclab.cs.ucdavis.edu/papers/NSM-final.pdf 166
6. K. Jackson, Intrusion Detection System (IDS) Product Survey, Los Alamos National Laboratory, Report LA-UR-99-3883, 1999. 162
7. S. Jajodia, D. Barbara, B. Speegle, and N. Wu, Audit Data Analysis and Mining (ADAM), project described in http://www.isse.gmu.edu/˜dbarbara/adam.html, April, 2000. 171, 175
8. K. Kendall, A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems, S. M. Thesis, MIT Department of Electrical Engineering and Computer Science, June 1999. 163, 169
9. J. Korba, Windows NT Attacks for the Evaluation of Intrusion Detection Systems, S. M. Thesis, MIT Department of Electrical Engineering and Computer Science, June 2000. 163, 169, 176
10. Lawrence Berkeley National Laboratory Network Research Group provides tcpdump at http://www-nrg.ee.lbl.gov. 164
11. R. P. Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das, The 1999 DARPA Off-Line Intrusion Detection Evaluation, Computer Networks, In Press, 2000. 163, 173
12. R. 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, IEEE Press, January 2000. 163
13. R. P. Lippmann and R. K. Cunningham, Guide to Creating Stealthy Attacks for the 1999 DARPA Off-Line Intrusion Detection Evaluation, MIT Lincoln Laboratory Project Report IDDE-1, June 1999. 163, 169, 170, 171
14. MIT Lincoln Laboratory, A public web site http://www.ll.mit.edu/IST/ ideval/index.html, contains limited information on the 1998 and 1999 evaluations. Follow instructions on this web site or send email to the authors (rpl or jhaines@sst.ll.mit.edu) to obtain access to a password-protected site with more complete information on these evaluations and results. Software scripts to execute attacks are not provided on these or other web sites. 163, 169, 173, 178

15. 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, 73-80, http://www.sdl.sri.com/emerald/index.html. 171, 175
16. S. Northcutt, Network Intrusion Detection; An Analysis Handbook, New Riders Publishing, Indianapolis, 1999. 162
17. V. 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. 166
18. T. H. Ptacek and T. N. Newsham, Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, Secure Networks, Inc. Report, January 1998. 171, 180
19. N. Puketza, M. Chung, R. A. Olsson, and B. Mukherjee, A Software Platform for Testing Intrusion Detection Systems, IEEE Software, September/October, 1997, 43-51. 163
20. A. Schwartzbard and A. K. Ghosh, A Study in the Feasibility of Performing Host-based Anomaly Detection on Windows NT, in Proceedings of the 2nd Recent Advances in Intrusion Detection (RAID 1999) Workshop, West Lafayette, IN, September 7-9, 1999. 171
21. R. Sekar and P. Uppuluri, Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications, in Proceedings 8th Usenix Security Symposium, Washington DC, Aug. 1999, http://rcs-sgi.cs.iastate.edu/sekar/abs/usenixsec99.htm. 171, 175
22. M. Tyson, P. Berry, N. Williams, D. Moran, D. Blei, DERBI: Diagnosis, Explanation and Recovery from computer Break-Ins, project described in http://www.ai.sri.com/~derbi/, April. 2000. 171, 172, 175, 180
23. G. Vigna, S. T. Eckmann, and R. A. Kemmerer, The STAT Tool Suite, in Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), IEEE Press, January 2000. 171, 175