Network Radar:

STTR Phase I
Final Report

L. Todd Heberlein

Net Squared, Inc.
4324 Vista Way
Davis, CA 95616

1 Introduction

This is the final report for the Network Radar project, Phase I, performed by Net Squared, Inc. for the United States Air Force's Rome Labs under an STTR contract. The Network Radar project encompasses a broad range of network monitoring technologies which together provide a comprehensive surveillance capability over an organization's computer networks. While the primary purpose of the Phase I contract was to establish the feasibility of the technology, we have also completed several early prototype network monitors.

Section two of the report presents an overview of network monitoring. We provide some historical perspective to establish a frame of reference for the current design of many existing network monitors. Next, we provide several advantages and disadvantages to network monitoring in general. Understanding both the strengths and weaknesses of network monitoring is critical to the design of an overarching security architecture.

Section three presents the key technology and research performed under the contract. The Network Monitoring Toolkit (NMT), the heart of the Network Radar effort, is introduced. We follow by covering two network monitors constructed with the NMT: Network Audit Trail (NAT) and Tracker. In the final part of this section we present the Non-cooperative Service Recognition (NCSR) technology. Since the NCSR technology was the original focus of the Network Radar Phase I proposal, we cover this material in a little more depth.

Section four consists of UNIX manual pages, or "man pages", for the various programs delivered for Phase I of this contract.

Finally, section five presents actual transcripts of the delivered programs being used. These transcripts, along with the man pages in section four, should provide the reader with a fairly good understanding of the delivered software.

2 Network Monitoring

The first network-based intrusion detection system came about as a direct result of Cliff Stoll’s "wiley hacker". One site hit by this hacker was Lawrence Livermore National Laboratory (LLNL), and very soon afterwards the Department of Energy (DoE) and LLNL was funding UC Davis to develop the Network Security Monitor (NSM).

While network-based monitoring provides definite advantages, it also has drawbacks. Any overarching security architecture should consider the advantages and disadvantages when investing and deploying network monitors, the Network Radar components included. This section presents several of these advantages and disadvantages.

2.1 Network Monitoring Advantages

2.1.1 The Network Explosion

As the NSM began to be deployed and known within the intrusion detection community, many criticized it as a system for only detecting "outsiders". Indeed, it was an outsider’s attack which motivated the initial development of the NSM, and the NSM was commonly deployed on the "border" between the Internet and a site’s internal network. However, in the late 1980s and early 1990s, many within the intrusion detection community believed that the "insider" – the employee or consultant with legitimate access to the system – was the real threat.

However, despite the widely held belief that the insider was the real threat, there was virtually no concrete evidence that this was true. Certainly, studies prior to 1980 would have shown the predominance of computer misuse was done by insiders, but this was well before internal networks were connected to large public networks (e.g., the Internet). The simple fact was that an outsider had virtually no access to the computer systems of concern. By the time the NSM was being deployed, the rules had changed.

Lawrence Livermore National Laboratory (LLNL) began funding UC Davis to develop the NSM in the summer of 1988. According to NW.COM, 33,000 hosts were connected to the Internet in July of 1988. By January of 1997, 16,146,000 hosts were connected to the Internet – nearly a 500 fold increase in eight and a half years. Meanwhile, the number of employees at LLNL has remained fairly constants over the same period of time. So while the amount of threat posed by insiders vs. outsiders may have been debatable in 1988, the simple fact is that for LLNL the growth in potential outsiders has greatly outpaced the growth of potential insiders.

2.1.2 Protocol standards

Network monitoring, and in particular the Network Security Monitor (NSM), has benefited by the standardization of network protocols. Unlike operating systems and their proprietary audit trails, network protocols are well documented, standard across many operating systems, and evolve very slowly. Within a week of the NSM being operational, it detected attacks into a VMS system as well as several different types of UNIX systems. In many instances, the computer systems attacked couldn't even produce an audit trail that a host-based intrusion system could use. Furthermore, because the IP, TCP, and UDP protocols today are the same as they were when the NSM was first developed, the same code and algorithms used by the NSM to detect attacks in 1990 can still detect attacks against currently shipping systems. Meanwhile, operating systems have evolved rapidly over the same period of time, and their audit trail formats have changed many times.

2.1.3 No load on end-systems

One of the fundamental concerns of users is the amount of impact an intrusion detection system will have on their existing computer environment. Additional CPU time must be dedicated to the generation and analysis of the audit trails, local disk storage may be needed to archive the audit records, and if the records are stored or analyzed off-host, network bandwidth must be used to ship the audit records to a remote system. Furthermore, the installation and management of audit trails in a large heterogeneous network can be difficult.

Network-based monitors avoid many of these problems. Indeed, a network monitor can be installed in a network without any visible change to the network or end-systems being monitored.

2.1.4 Invisible to attackers

One of the concerns of security administrators and intrusion detection developers is that an attacker might simply turn off the intrusion detection system. While ad-hoc solutions such as a "heart beat" have been deployed in many IDS systems, the simple fact remains that once an attacker has seized control of a host, all information (or lack of it) is unreliable.

Network-based monitors benefit from the fact that computer controls the network. Thus, even if an attacker seizes control of a host, communication in and out of that host will still be visible to the network monitor. In fact, there is virtually no way an attacker can even know if a network-based monitor is observing his activities.

2.1.5 Easier to administer

Since a single network-based intrusion detection system can monitor potentially thousands of computers from a single strategic location, it can be orders of magnitude easier to install and manage than host-based intrusion detection systems. Firewalls have become popular for essentially the same reason–it is easier to secure one network location than secure all the end systems at a site.

2.1.6 Content available

One very powerful, albeit controversial, capability network-based monitors have is their ability to analyze content of a session. That is, the contents of a file, e-mail message, or password used can be analyzed. In contrast, host-based monitors will only know that a file has been created or changed, an email message has been sent, and that a password has been entered. This capability can be used to:

- Determine if a downloaded file is actually a password file that is about to be run through a password file
- Identify sensitive information being shipped off-site by e-mail
- Quickly determine if a password is guessable by a password cracker

2.1.7 Presentation clearer

In many instances, network-based monitors can present computer misuse and attacks in a format that is much easier to understand than host-based monitors can. For example, for attacks using remote login facilities (e.g., Telnet or Rlogin) VCR-like playback tools can be used to recreate exactly what an attacker saw on his screen, including timing information and contents of edited files. This is possible because complete screen control and display information can be captured from a session. A host-based monitor has no way of capturing or recreating this information. Also, interactions with an application can be captured (e.g., access to a Sendmail server). Since very few applications generate detailed audit records of their own, host-based monitors only have access to the application's calls to the operating system, resulting in a possible significant loss in related context.

2.1.8 Attacks only visible from network

Many attacks do not generate audit records and are therefore only visible via network monitoring. For example, the CERT advisory CA-91:21 refers to an attack against the Network File System (NFS) which does not generate any information that can be used by a host-based monitor to detect the attack. In the attack, a client host creates what it hopes is a legitimate file handle on a remote NFS server. If the file handle is not legitimate, the NFS server sends back a "Stale NFS handle" error, and the client tries again. Once the client has correctly guessed one handle, it can use it to eventually access much of the file system on the remote server. In the attack, no audit records are generated (even for failed accesses), no access control tables are checked, and no new processes are created. The attack is invisible to any host based monitor, but it is easy to detect from a network-based monitor.

Even the SYN-flooding attacks are not guaranteed to generate audit records since the attack only creates partially open connections. There are also variations on network sweeps (where an attack tries to determine which server ports on which machines have network servers) which can be used to avoid detection by host-based monitors. TCP session hijacking can also be virtually impossible to detect from host-based monitors but is often very easy to detect with network-based monitors.

2.2 Disadvantages of network monitoring

While network-based monitors have many advantages, they also have several disadvantages and limitations. Any plan to deploy and rely on network-based monitors must consider these issues, especially since these disadvantages may become a greater issue with changes in technology and network usage. This section discuses several of those limitations.

2.2.1 Encryption

Of all the factors limiting network monitors' capabilities and effectiveness, perhaps the greatest is encryption. While encryption (as well as cryptographic mechanisms) is widely perceived as the solution for many, if not most, network security problems, it also poses serious challenges to network intrusion detection systems, incident handling tools, and possibly to firewalls as well.

To date, Net Squared researchers have not had the greatest record in predicting the wide spread adoption of encryption in network communication. As early as 1988, when initially developing what would become the Network Security Monitor (NSM), we realized that virtually all traffic, including passwords, crossed the network as plain text. We assumed that once this became common knowledge, encryption would quickly take hold. Indeed, it did eventually become common knowledge in 1993, and password grabbing "sniffers" became widespread. However, instead of encryption being adopted, one-time passwords were perceived as the solution to the sniffer problem. So while an attacker could still sniff someone's connection and password, the attacker could not use that information to gain access to the remote machine. Because of this, network security monitors continued to be successful even with advent of sniffers.

Yet the pressures continue to mount for adoption of encryption in network communication. This pressure is now largely driven not by the security community but by the commercial community. With the advent of the World Wide Web, the Internet is becoming a mechanism for commerce, and that has created an economic interest in creating secure network communication. Industry and independent standards efforts which bring encryption to network communication include:

- Intel's Common Data Security Architecture (CDSA)
- HP's Enterprise Security Framework
- Netscape's Internet Application Framework, which supports Secure Sockets Layer (SSL) and Secure Electronic Transactions (SET) protocols.
- Microsoft's Windows NT operating system (with Service Pack 2) includes support for SSL.
- VISA and Mastercard's Secure Electronic Transactions (SET)
- Secure Sockets Layer (SSL); originally developed by Netscape but available for independent development.
- SSH Communications Security's SSH protocol (similar in concept to SSL)
- IPv6 and IP encapsulation

With the economic motivation for encryption, and its ease-of-use provided by new protocols, we believe encryption will be on the rise. Indeed, in a recent sample of TCP/IP connections visible on UC Davis' backbone network (130,574 connections), more encrypted WWW connections were observed than either FTP or Gopher connections.

Encryption will effect network monitors in at least two ways. First, protocols such as SSH and SSL typically encrypt packets above the TCP layer. Thus, network monitors will still be able to detect and track individual connections as well as make an educated guess as to their service types (based on port numbers). However, content analysis (e.g., string matching), arguably the greatest strength of most existing network monitors, will be rendered useless. Second, protocols such as IPv6 and IP encapsulation will even prevent the identification of individual connections because the transport layer (e.g., TCP or UDP) port information will be invisible to the network monitors.

2.2.2 Loss of data

To date, network monitors have always been prone to the occasional "packet drop". That is, because of a full buffer, noise, or another problem, the network monitor is unable to capture and process a network packet. A dropped packet can seriously impair a network monitor’s ability to match strings in a connection (once again, this is a very powerful feature in existing network monitors). Experiments at UC Davis have shown that an attacker can increase the number of packets a network monitor drops by either increasing the load on the monitor or increasing the number of packets on the network. An attacker can exploit this weakness to reduce the monitor's probability of capturing a real attack.

The loss of data can be further complicated by some forms of encryption techniques. Even if all encrypted network traffic used fixed keys (that is, not generated at session time) which were made available to the network monitor, decryption can be hampered by loss of data. For example, if the encrypted data is C = <c_2, c_2, ..., c_k>, and segment c_i is dropped, then it may be impossible to retrieve any of the original data. Similarly, if cipher block chaining (CBC) is used, loss of the encrypted data segment c_i will, at the minimum, cause the loss of the corresponding data segments m_i and m_{i+1}, and may possibly result in the inability to retrieve all data segments $m_k, k \geq i$.

Some groups claim their network monitors avoid packet loss by coupling their monitors to a router. However, these coupling often consists of an isolated Ethernet segment to which the router dumps packets and from which the monitor can sniff the packets. Other configurations may include the router wrapping the original packet in a UDP packet and forwarding it to the network monitor. However, without additional hardware or software protocols verifying packet receipt, these mechanisms do not provide any additional guarantees against packet loss than the traditional network sniffers.

2.2.3 Loss of context

While the monitor may detect an event (e.g., match a string in the data), the loss of context in which that event occurred limits the monitor's ability to appropriately judge the seriousness of the event. For example, if a user logs in across a monitored network and reads a security newsgroup, the network monitor may detect many suspicious strings in that user's connections. The network monitor is unable to determine that the context in which the strings are matched is the reading of a newsgroup, so the connection is inappropriately given a high warning value.

In general, because network monitors are unable to determine a user's actions (e.g., reading mail, reading news, editing a report, or entering malicious commands), they are unable to fully understand the significance of the data they are observing.

2.2.4 Insider problem

If a user logs in at a workstation console or directly connected terminal, no network traffic will be generated. Without network traffic, a network-based monitor obviously cannot detect any traffic. Also, Ethernet hubs and switches have greatly compartmentalized internal network traffic. In order to get complete coverage of internal traffic, network-based monitors may have to be deployed on virtually every network hub, thus greatly increasing the equipment cost and management effort needed to effectively deploy network monitors.

2.2.5 Voluminous data

With the increased use of distributed computing, client-server solutions (e.g., WWW), and communication by computers (e.g., e-mail, Internet phones, and video conferencing), the amount of traffic on networks is increasing dramatically. This growth in traffic puts two strains on network-based monitors. First, network-based monitors must process increasing amounts of traffic. This processing is limited by available CPU, bus, and memory speeds. Today, 100 MBit network are standard, and even faster Ethernet speeds as well as ATM will continue to push bandwidth higher. Second, archiving data (e.g., packets) for future analysis require much greater storage capacity, and archiving *all* packets on today's high speed, heavily used networks is virtually impossible.

3 Network Radar: Phase I Concepts and Deliverables

This section presents material delivered as part of Network Radar Phase I contract. Because of the size of the total software delivered, only a small portion is discussed here. For additional information on the various programs delivered in phase I, see section 4: Manual Pages. Also, additional information can be found with the software delivered as part of phase I.

Section 3.1 presents the Network Monitoring Toolkit, the heart of the Network Radar effort. Section 3.2 presents the Network Audit Trail, a program able to create a rich audit trail of network activity. Section 3.3 presents Tracker, an IP-Watcher cloned designed to aid incident support teams. Finally, section 3.4 presents the algorithms and experimental results for the Non-cooperative Service Recognition (NCSR) technology.

All figures for this section are presented at the end of the section.

3.1 Network Monitoring Toolkit

The key technology behind Net Squared's network monitors is the Network Monitoring Toolkit (NMT). The NMT consists of a family of software objects and a framework for assembling these objects. This approach mirrors several other software development efforts including University of Arizona's X-kernel project and NeXT Software's Application Toolkit.

The NMT software suite is almost entirely developed in C++. While, in theory, any programming language could be used to build the NMT, C++'s language constructs, wide availability on many platforms, and its fast run-time characteristics made it a logical choice. However, on occasion, we have used other programming languages where appropriate. For example, an early version running under NEXTSTEP, the basis for Apple's next-generation operating system, used Objective-C for the user interface components, and the current NMT software uses Lawrence Berkeley Laboratory's packet capture library, libpcap, which is written in C.

NMT supports two major object classes: Layers and Streams. The major difference between a Layer and a Stream object is that a Layer object is created at run-time and exists throughout the lifetime of the program, whereas a Stream object is created when an appropriate network session is detected and is destroyed when that network session ends.

Examples of Layer objects include EthernetLayer, IpLayer, and TcpLayer. Examples of Streams include TelnetStream, LoginStream, FtpStream, and StringStream.

Figure 1A shows a sample NMT Layer architecture. In this example, there are five Layer objects: an Ethernet layer, two IP layers, a UDP layer, and a TCP layer. Each object would be created at run-time, assembled in this configuration, and exist throughout the lifetime of the program. In this example, we use two IP layers, the second one is used to support IP encapsulation, Both IP layers can send data to UDP and TCP layers.

Figure 1B shows three potential stream stacks. A stream stack is a series of Stream objects, stacked one on top of another, which process the data from a single network session (e.g., a TCP/IP connection)[1]. The first stack might be used to analyze a Telnet connection; the second stack might be used to analyze an HTTP, or WWW, connection; and the third stack might be used to processes secure HTTP, HTTP running over the Secure Sockets Layer (SSL)[2] protocol. These stacks would be created and destroyed as network sessions are created and destroyed. Each stream stack object can create an audit record, and all the records for a single stream class (e.g., Login class) will be maintained in a single audit file.

Figure 2A shows a sample set of objects and their relationships which might be present in a typical monitor during operations. Three layer objects (Ethernet, IP, and TCP) and a Tap (PcapTap) are created when the monitor is started and run throughout the life of the program. Three network sessions are also being monitored: a Telnet connection, a WWW connection, and an Rlogin connection. The program audit_log(1)[3] is an example of such a monitor.

Figure 2B shows an example monitor used to extract and present a single network session from a session file (a file containing packets for a single session). The only new objects in this monitor (from the one shown in Figure 2A) are a File stream object and a FileTap object. The File stream object would be used to present the data from the connection (e.g., print to a screen or write to a file). The FileTap reads the packets archives in the session file. The program print_session(1) uses this basic architecture.

Figure 2C shows an example thumbprint monitor. The key new object in this example is the Thumbprint stream object. Since the actual calculations of a thumbprint is

[1]Note that the TCP protocol has been broken in two: part of a packet's data is processed by a TCP Layer object and part of the data is processed by a TCP Stream object.
[2]If the SSL is using fixed keys (public or private) which is available to the network monitor, complete HTTP analysis can be performed. However, if keys are computed on the fly (e.g., Diffie-Hellman), only limited analysis can be performed.
[3]See the mannual, or "man pages", section of this report.

relatively straight forward, the Thumbprint object design and implementation is relatively simple. While Phase I did include the implementation of a thumbprint monitor, we plan to deliver one as part of Phase II.

3.2 Network Audit Trail (NAT)

The Network Audit Trail (NAT) monitor exploits the structure in network protocols to create a content-rich audit trail of network activity. For example, for Rlogin connections, we can identify the user name on the client machine, the first user name tried on the server, the terminal type, and the terminal window size. We can also determine additional information such as additional login names tried, passwords tried, and string matches.

This audit trail can be browsed and searched by supporting tools we will deliver as part of this contract. For example, if you discover a Trojan login program which allows an attacker to login by simply supplying the password "BAA97-11", you can search through your audit logs for every time the string "BAA97-11" was used as a password. It is important to note that unlike existing monitors (NSM, NID, ASIM, NetRanger, etc.), you did not have to previously direct your monitor to look for the string "BAA97-11" – you can search for this string "after the fact". Furthermore, other network monitors would match every instance of the string "BAA97-11", which may be quite common on your network. NAT can search for the string only in the context of it being used for a password.

A prototype of NAT is currently running on an DEC Alpha workstation monitoring a 100 MBit Ethernet backbone. A version for Solaris has also been delivered to Rome Labs as part of the Phase I contract..

We have also developed some prototype browsing tools as part of our Phase I STTR contract with Rome Labs. For example, Figure 3A shows UCD Computer Science's network server ports which were accessed via the Internet for one hour. As can be seen, port 80, the typical WWW port, had the most connections. Ports 25, 21, and 23 (Sendmail, FTP, and Telnet respectively) were also accessed. However, ports 8001, 1024, 8080, and 1234, were also quite popular. None of these servers were installed by the system administrators. Eventually we will integrate the Non-cooperative Service Recognition (NCSR) technology to determine the server type running at each one of these ports. Figure 3B shows us "zooming in", discovering the actual hosts which housed these server ports and the number of connections to each one of them.

Figure 4 shows another view of the audit trail. Figure 4A shows a high-level view of five connections (0 through 4). Figure 4B shows us "zooming in" on connection 0. The connection was an FTP connection; the attempted user names were "spinkb" and "heberlei". The user tried to retrieve a file called "Rome", upload a file called "sniffer", and create a directory called "..." (three dots).

3.3 Tracker-Incident Support Tool

Tracker is a network monitor tailored to support incident handling. In particular, if you are already aware that a system has been compromised, you can use Tracker to monitor all connections in and out of a small number of hosts; tap into each connection; send messages back to the attacker's screen; and kill the attacker's TCP/IP connections. A prototype has been developed and delivered to Rome Labs as part of our Phase I contract. The fifth figure at the end of this section is a screen shot of the Tracker monitor. (Because of the size of the image, the "Figure" label does not fit on the same page as the image, so it has been omitted). For more details, see the manual page tracker(1).

3.4 Non-cooperative Service Recognition

Non-cooperative Service Recognition (NCSR) technology is designed to determine the service type of a network session. For example, a TCP/IP connection to port 25 is typically a Sendmail connection, but nothing prevents someone from installing a login server or a WWW server at port 25. Because of the unusual nature of the problem that this technology is designed to address, we reproduce a section from the original proposal describing the problem. If you are familiar with the proposal, you can safely skip section 3.4.1.

3.4.1 Threats to Control and Integrity

For a security administrator to have control of his network and to maintain a high level assurance of its integrity, he must know what services are operating within his domain, and he must know they are operating properly. In this section we present a number of anecdotal stories and examples demonstrating how easy it is for an innocent user or malicious attacker to destroy the integrity of a network. These examples are just a few of the threats that Network Radar and the NCSR technology will address.

3.4.1.1 New Server Vulnerability

Because of various research needs, all UC Davis Computer Security Laboratory employees are given root privileges on the lab machines. Taking advantage of this privilege, one student, wanting to be able exchange relatively large files with his friends, installed a

File Transfer Protocol (FTP) server which accepted anonymous logins[4]. Furthermore, since he wanted to allow his friends to upload files, one of the directories was set world writeable. Unfortunately, a hacker discovered this, and soon hackers from around the world began using the security lab server to exchange illegal copied of commercial software. The hackers tried to cover their tracks by using unusual file names which would not appear when normally looking at the contents of the directory. The security lab only discovered the site after being notified by a system administrator from another organization. Beyond being embarrassed, the security lab was concerned about the possible legal liabilities of having pirated software on their FTP server.

3.4.1.2 Rogue Servers

Following the hacker FTP site incident, the UC Davis Computer Security Laboratory tightened its own security. All unnecessary network services were removed from hosts, and the ones which were left were wrapped with TCP wrappers. The wrappers' audit logs were continuously displayed on one of the faculty's machine. Several months later, a lab employee performed an audit of all network traffic in and out of the lab, and despite all the precautions taken, he discovered numerous connections from around the world were being made into Security Lab machines. After further investigation he discovered that an alumni had installed several Internet Relay Chat (IRC) servers on the Security Lab machines. Since any user, even without special privileges, can install these servers, preventing this from happening is almost impossible. The only solution is either continuous monitoring of every process on every machine or monitoring every connection into the Security Lab.

3.4.1.3 Invisibility to Network Monitors

The USAF's Countermeasures Engineering Team, the USAF's ASIM project, DISA's ASSIST team, and others in the military have used the NSM[5] to both detect and track attacks into and against their sites. The NSM reads packets from the network, saves them to disk, and analyzes the packets for potential attacks. Because an average network can generate an enormous amount of traffic, only a small portion of this traffic is actually saved. The choice of traffic saved is usually based on which services are common for attackers. For example, Telnet, Rlogin, and Sendmail[6] are commonly exploited by intruders, so the NSM is often

---

[4] The user name "anonymous" is used, and any password is accepted.

[5]The Network Security Monitor (NSM) continues to be developed by Lawrence Livermore National Laboratory under the name Network Intruder Detector (NID).

[6] Telnet is a standard network protocol for terminal logins; Rlogin is a UNIX variant of Telnet; Sendmail is a standard network protocol for exchanging e-mail. Most UNIX systems are shipped with servers for each of these protocols.

configured to save packets to/from ports 23, 513, and 25, the ports these services usually reside on[7]. However, with some simple steps, an attacker can establish a rogue server, rendering his connections invisible to the NSM.

For example adding the two lines in Example 1 to /etc/services and /etc/inetd.conf respectively will install a Telnet server at port 1066. When an attacker uses this Telnet server, his connection will be invisible to most NSM monitors. To connect to this "secret Telnet" server, an attacker simply needs to type

```
% telnet hostname 1066
```

```
sec_telnet  stream  tcp  nowait  root  /usr/etc/telnetd  sec_telnet
sec_telnet  1066/tcp
```

Example 1

In this example, the attacker needs to establish root access on the target machine and modify two configuration files. Also, the attack still uses the Telnet daemon which creates some minor log files, potentially revealing the attackers presence. However, a simple login server can be constructed with about twenty lines of code which can be installed by any users at any port. This simple server will also bypass any authentication checks and will not generate any additional log messages.

3.4.1.4 Tunneling Through Firewalls

As a result of the perceived threats from attacks by outsiders, many organizations are installing firewalls. One of the most popular classes of firewalls is the filtering firewall[8], which forwards only certain types of network traffic. Because of the recognized importance of electronic mail for conducting business, most organizations configure their filtering firewalls to allow Sendmail traffic through. However, in reality they are not allowing Sendmail through but packets to or from port 25, the port number where Sendmail servers are usually located.

Unfortunately, any network server can be installed at port 25. For example, just as in the case of the secret Telnet example, the two lines in Example 2 added to /etc/services and /etc/inetd.conf will install the Telnet server at port 25[9]. To connect to this Telnet server,

[7] By convention, well known Internet services are located at well known ports on each host. For example, Telnet servers are installed on port 23, Rlogin on port 513, and Sendmail on port 25.

[8] A filtering firewall allows or disallows a packet through based on certain patterns within the packet headers. Typically the firewall is configured to block all packets except those with certain values in their source and destination ports (ports are fields within the TCP and UDP headers). These port values are associated with well known services such as Sendmail (port 25) or WWW traffic (port 80).

[9]This assumes that a Sendmail daemon isn’t already installed on the system.

masquerading as a Sendmail connection and tunneling through the firewall, an attacker simply needs to type:

```
% telnet hostname 25
```

```
masq_telnet  stream  tcp  nowait  root  /usr/etc/telnetd  masq_telnet
masq_telnet  25/tcp
```

Example 2

3.4.1.5 Schizophrenic Servers

One approach to dealing with the new and rogue servers is to perform exhaustive scans of every port on every system in a network on a regular basis. Unfortunately, these scans take time (not to mention network bandwidth), so transient server may go unnoticed. Furthermore, this approach is ineffective with masquerading servers. To combat masquerading servers, the scan would have to try to establish a dialogue with each server in order to confirm its type (e.g., is this really a Sendmail server established at port 25, or is this really a finger daemon installed at port 79). However, an intruder can easily defeat even this approach with a schizophrenic server.

A schizophrenic server is one which behaves as one type of server for certain requests and another type of server for another request. For example, the single line in Example 3 added to a TCP wrapper configuration file can be used to establish a schizophrenic finger server. In this particular case, connections to port 79, the normal finger daemon port, will appear exactly like a finger server for every single host except hack.com. When a user from hack.com connects to port 79, he receives the Telnet server.

```
in.fingerd: hack.com: twist exec in.telnetd
```

Example 3

3.4.1.6 Broken Server

Another threat is simply a broken server. In part of the Internet Worm’s attack, vulnerabilities were exploited in the finger daemon and Sendmail daemon[10] essentially providing the worm with a login connection to the remote host. In all likelihood, there will be future cases of broken servers, where the server provides more capability than was originally intended. Active scans of a network will not be able to reveal a heretofore unknown vulnerability in a network service.

[10]The hole in the Sendmail daemon could be viewed by some as a feature. This feature is removed from most but not all systems today.

3.4.1.7 Threat Summary

Control and integrity of a network cannot be assured without careful monitoring and analysis of *all* activity on the network. Introduction of new network services by legitimate users can introduce new vulnerabilities into the network. Introduction of new services by malicious users can allow the users to hide their activity from network monitors such as the NSM or to circumvent filtering firewalls.

3.4.2 NCSR: Design and Algorithms

Non-cooperative Service Recognition (NCSR) is designed to identify these and other threats to the control and integrity of a network. To detect these threats, the NCSR-based monitor must be able to detect, track, and identify every single session on the network. Objects from the NMT, and the Network Audit Trail (NAT) software in particular, already provide the ability to detect and track network sessions. The missing piece is the technology to identify the service type of a network session.

While a unified NCSR network monitor is not part of the Phase I deliverables, all the key technologies have been prototyped and delivered in various programs. These programs are discussed in Section 4 (Manual pages), and they are demonstrated in example 4 of Section 5 (Example Output). The key programs are:

audit_log	– This is the Network Audit Trail program. audit_log identifies, tracks, and collects statistical information for each network session. This information is saved in a TCP audit log.
class_extractor	– This program parses the TCP audit log, extracting and printing statistical information associated with sessions to a preselected port. For example, information regarding network sessions to port 23 (the typical Telnet server port) can be printed.
max_extractor	– This program analyzes data created by class_extractor, pulling out information to be used to "normalize" each session's statistical information. Statistical information is normalized to optimize the behavior of the classifier used to actually identify the service type of a session.
class_normalize	– This program transforms the data generated by class_extractor, using a vector created by max_extractor to normalize the statistical information.
train_test	– This program splits the normalized data for each session type into two disjoint sets. One set will be used to train the classifier, and the other set will be used to test the classifier.
net4	– This program actually contains the core NCSR technology, a supervised classifier. The "supervised" term comes from the fact that

we initially supply the classifier with a set of data points (the train file set created by the train_test program) and tell it what type of network service each data point belongs to. The classifier is then evaluated with the data in the test data set.

net5 – This program is essentially the same as net4; however, additional information about the classifier's performance is provided. This information should be useful for enhancing the classifier's overall performance.

The programs net4 and net5 could use any of a number of supervised classification algorithms can be used. For the Phase I proof-of-concept, we chose a well known neural network developed by Rumelhart, Hinton, and Williams[11], and described in the text Elements of Artificial Neural Networks[12]. The heart of the algorithm is shown in Figures 6, 7, and 8, and we provide a brief description in the next few paragraphs.

Figure 6 shows the calculation for the output value for a neural node. The same calculations is performed for hidden nodes and output nodes (input nodes simply pass the input vector into the first hidden layer). The calculation simply takes the inner product of the input vector (either the output from the previous layer or the input vector representing the data point) with the weight vector, and this value is passed through the sigmoid function, creating a value between zero and one. Ideally, if this was an output node, a value of 1 would indicate that the input vector was from the class represented by this output node, and a value of 0 would indicate that the input vector belonged to another class.

Figure 7 shows the backpropagation calculation for output nodes. These calculations update the node's weight vector, and store a representation of this node's "local" contribution to the output error, $\delta_{j,r}$. This latter value is needed when updating the hidden layer's nodes. The primary difference between the original Rumelhart algorithm is that we normalize the input vector before calculating the delta W values.

Figure 8 shows the backpropagation calculation for hidden nodes. These calculations are nearly identical to those for the output nodes (see Figure 7) except the initial calculation of the "local" contribution to the output error. This calculation of this value is more complex because the actual error is masked by the output layer (and any hidden layer after this one).

There have been a number of enhancements to this algorithm; however, for this initial proof-of-concept implementation, we remained very close to the original algorithm. We hope to enhance our NCSR capability by applying various enhancements made to the

[11]Rumelhart, Hinton, and Williams. "Learning internal representations by error propagation". *Parallel Distributed Processing*, 1, 1986.
[12]Mehrotra, Mohan, Ranka. *Elements of Artificial Neural Networks.* The MIT Press. 1997.

algorithm over the past eleven years. However, despite using this early implementation and a very simple statistical representation of a network session, we achieved extremely promising results.

For our experiments with the NCSR technology, we used the following statistical values to represent a session:

- number of packets sent by the client
- number of packets sent by the server
- number of bytes sent by the client
- number of bytes sent by the server
- ratio of bytes per packet sent by the client
- ratio of bytes per packet sent by the server
- duration of session (in seconds and micro-seconds)

Because the ratio of bytes per packet can be computed from the number of bytes and the number of packets, only five unique statistics are used in our Phase I experiments. We had hoped to use a much greater number and variety of statistics, but the six month time limitation of Phase I prevented us from trying additional experiments. In Phase II, we hope to try additional statistical measurements.

Despite the limited statistics used for our experiments, we were still able to get extremely good experimental results. Figure 9 shows the misclassification rates of five of the top six network services on the test network[13], WWW, Sendmail, Authentication[14], POP (Post Office Protocol)[15], and Telnet. The aggregate misclassification rate was 3.9% for a single network session; in other words, any given network session has a 96.1% probability of being classified correctly.

While we were very happy with a correct classification rate of 96.1% per connection on our very first prototype, the numbers are even more impressive when trying to classify a server. If we assume for a moment that the server is not a schizophrenic server (see section on threats), we can attempt to classify a server by classifying the observed network sessions into that server. For example, assume we examine five sessions into a given server, we classify each session, and then we "take a vote" as to the type of server these sessions are

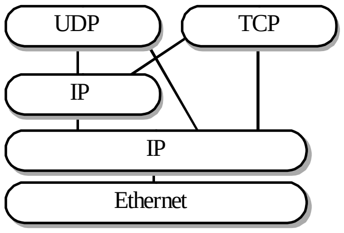
[13]The test network was one of UC Davis' 100 MBit Ethernet backbones. One service (port 6667) was not chosen since we were unsure of its type.
[14] Authentication is a protocol to allow a host receiving a connection to ask the client host the user's name who is responsible for the connection.
[15] POP is used to connect a mail client (e.g., Internet Explorer on a PC) to a mail server (e.g., a UNIX or NT server).

connecting to. The probability that the majority of those five sessions are correctly classified (at least 3 of 5) can be calculated using the binomial probability distribution:

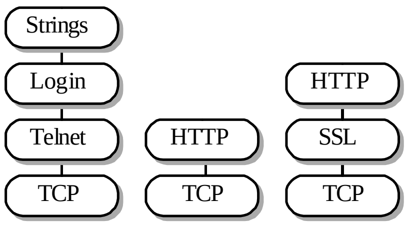
$$P(x) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

and calculating P(3)+P(4)+P(5), the probability that three or more of the network sessions were classified correctly. Performing this calculation gives us a probability of 99.9% that we can correctly classify a server with just five network sessions to that server!

Manual pages for the tools used in these experiments are in Section 4, Manual Pages, and a transcript showing how these tools were used to produce these experimental results is in Section 5, Example Output.

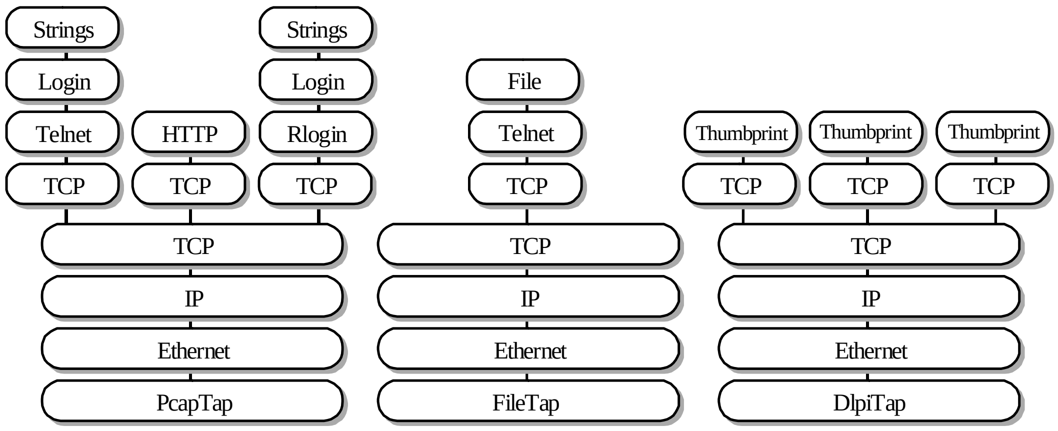


A



B

Figure 1



A **B** **C**

Figure 2

```
SERVER PORT    80: 1892 connections
SERVER PORT  8001: 106 connections
SERVER PORT    25: 79 connections
SERVER PORT  1024: 56 connections
SERVER PORT    21: 24 connections
SERVER PORT    23: 20 connections
SERVER PORT  8080: 13 connections
SERVER PORT  1234: 10 connections
```

A

```
----------------------------------------------
SERVER PORT    80: 1892 connections
    SERVER ADDRESS   128.120.55.68: 1358 connections
    SERVER ADDRESS   128.120.56.77: 372 connections
    SERVER ADDRESS  128.120.56.107: 69 connections
    SERVER ADDRESS   128.120.56.87: 39 connections
    SERVER ADDRESS   128.120.56.11: 37 connections
    SERVER ADDRESS  128.120.56.223: 14 connections
    SERVER ADDRESS  128.120.56.117: 2 connections
    SERVER ADDRESS   128.120.56.90: 1 connections
----------------------------------------------
SERVER PORT  8001: 106 connections
    SERVER ADDRESS  128.120.56.186: 106 connections
----------------------------------------------
SERVER PORT  1024: 56 connections
    SERVER ADDRESS  128.120.56.104: 56 connections
----------------------------------------------
```

B

Figure 3

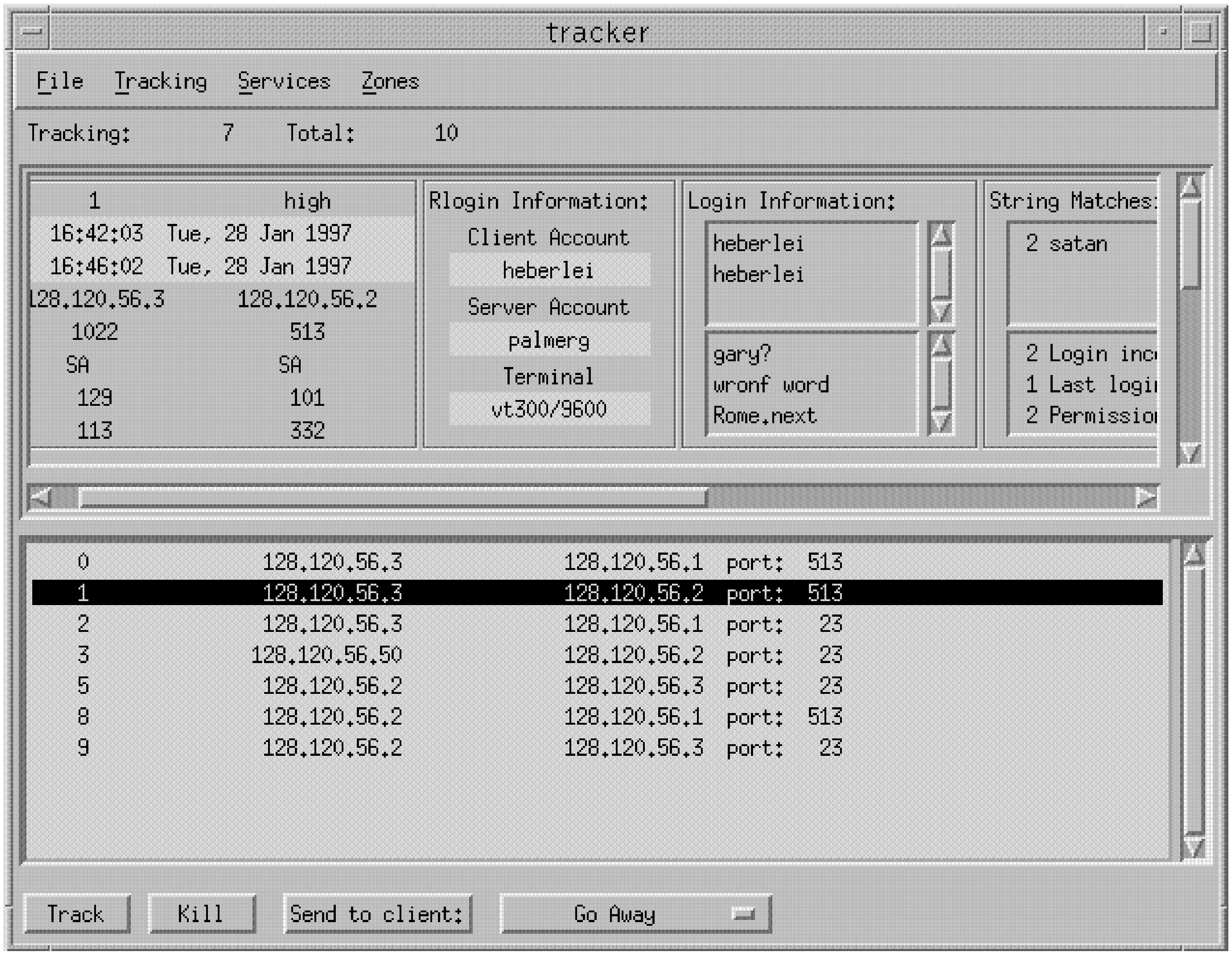
```
1     128.120.56.1  -->     128.120.56.3   1026
2     128.120.56.1  -->     128.120.56.3   1028
0     128.120.56.3  -->     128.120.56.1   21
4     128.120.56.4  -->     128.120.56.1   23
3     128.120.56.3  -->     128.120.56.1   513
```

A

```
-------------------------------------------------------------------
   0     128.120.56.3  -->     128.120.56.1  21
from: 16:07:17 ( 1/26/1997)  to: 16:09:59 ( 1/26/1997),  162.298887 secs
client flags: SAF     server_flags: SAF
        ---- FTP ------------------------------------------
        USER: spinkb
              heberlei
        PASS: Rome.sun
        RETR: Rome
        STOR: sniffer
        CWD:  Secret
              ...
        MKD:  ...
        FAILURES: 3
-------------------------------------------------------------------
```

B

Figure 4



```
Tracking:        7     Total:       10

   1                  high
 16:42:03  Tue, 28 Jan 1997
 16:46:02  Tue, 28 Jan 1997
128.120.56.3       128.120.56.2
    1022                513
    SA                  SA
     129                101
     113                332

Rlogin Information:
    Client Account
       heberlei
    Server Account
       palmerg
       Terminal
     vt300/9600

Login Information:
heberlei
heberlei

gary?
wronf word
Rome.next

String Matches:
 2 satan

 2 Login inc
 1 Last logi
 2 Permissio

  0        128.120.56.3        128.120.56.1  port:  513
  1        128.120.56.3        128.120.56.2  port:  513
  2        128.120.56.3        128.120.56.1  port:   23
  3       128.120.56.50        128.120.56.2  port:   23
  5        128.120.56.2        128.120.56.3  port:   23
  8        128.120.56.2        128.120.56.1  port:  513
  9        128.120.56.2        128.120.56.3  port:   23
```

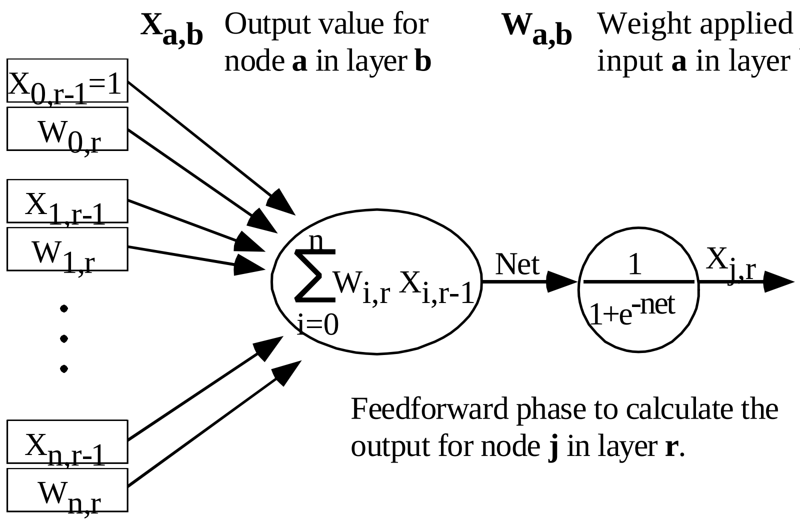

$X_{a,b}$ Output value for node **a** in layer **b**
$W_{a,b}$ Weight applied to input **a** in layer **b**
$X_{0,r-1}=1$
$W_{0,r}$
$X_{1,r-1}$
$W_{1,r}$
$X_{n,r-1}$
$W_{n,r}$
$\sum_{i=0}^{n} W_{i,r} X_{i,r-1}$
Net
$\frac{1}{1+e^{-net}}$
$X_{j,r}$
Feedforward phase to calculate the output for node **j** in layer **r**.

Figure 6

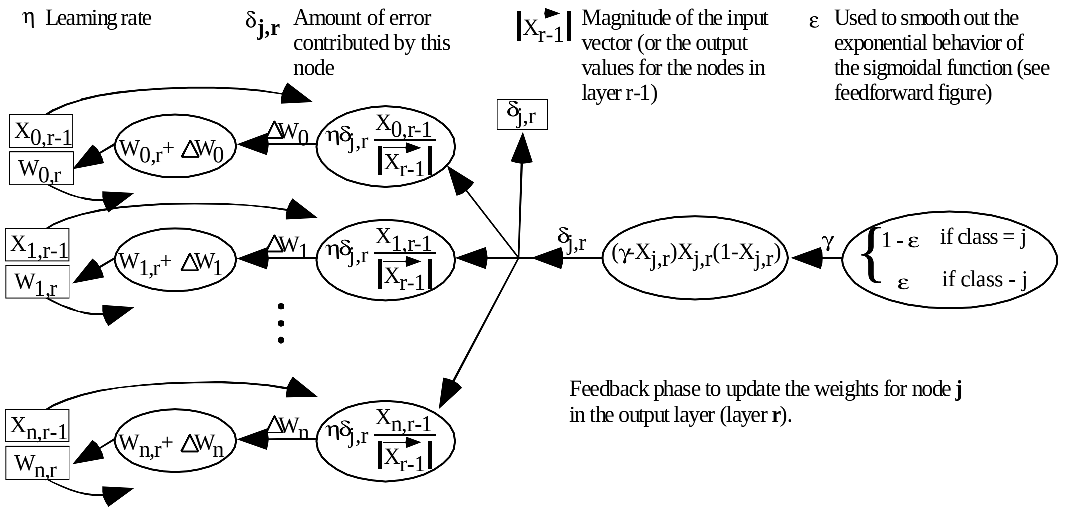

η Learning rate
$\delta_{j,r}$ Amount of error contributed by this node
$|\vec{X}_{r-1}|$ Magnitude of the input vector (or the output values for the nodes in layer r-1)
ε Used to smooth out the exponential behavior of the sigmoidal function (see feedforward figure)
$\delta_{j,r}$
$X_{0,r-1}$
$W_{0,r}$
$W_{0,r}+\Delta W_0$
ΔW_0
$\eta\delta_{j,r}\frac{X_{0,r-1}}{|\vec{X}_{r-1}|}$
$X_{1,r-1}$
$W_{1,r}$
$W_{1,r}+\Delta W_1$
ΔW_1
$\eta\delta_{j,r}\frac{X_{1,r-1}}{|\vec{X}_{r-1}|}$
$X_{n,r-1}$
$W_{n,r}$
$W_{n,r}+\Delta W_n$
ΔW_n
$\eta\delta_{j,r}\frac{X_{n,r-1}}{|\vec{X}_{r-1}|}$
$\delta_{j,r}$
$(\gamma-X_{j,r})X_{j,r}(1-X_{j,r})$
γ
$\begin{cases}1-\varepsilon & \text{if class = j}\\ \varepsilon & \text{if class - j}\end{cases}$
Feedback phase to update the weights for node **j** in the output layer (layer **r**).

Figure 7

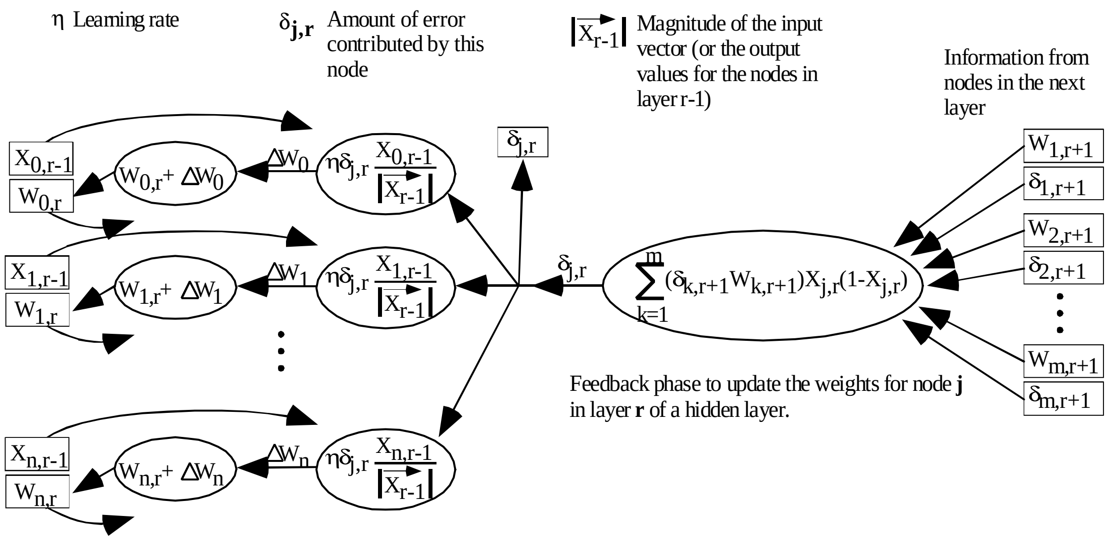

η Learning rate
δj,r Amount of error contributed by this node
|Xr-1| Magnitude of the input vector (or the output values for the nodes in layer r-1)
Information from nodes in the next layer
X0,r-1
W0,r
W0,r+ ΔW0
ΔW0
ηδj,r X0,r-1/|Xr-1|
X1,r-1
W1,r
W1,r+ ΔW1
ΔW1
ηδj,r X1,r-1/|Xr-1|
Xn,r-1
Wn,r
Wn,r+ ΔWn
ΔWn
ηδj,r Xn,r-1/|Xr-1|
δj,r
δj,r
Σ k=1 to m (δk,r+1Wk,r+1)Xj,r(1-Xj,r)
W1,r+1
δ1,r+1
W2,r+1
δ2,r+1
Wm,r+1
δm,r+1
Feedback phase to update the weights for node j in layer r of a hidden layer.

Figure 8

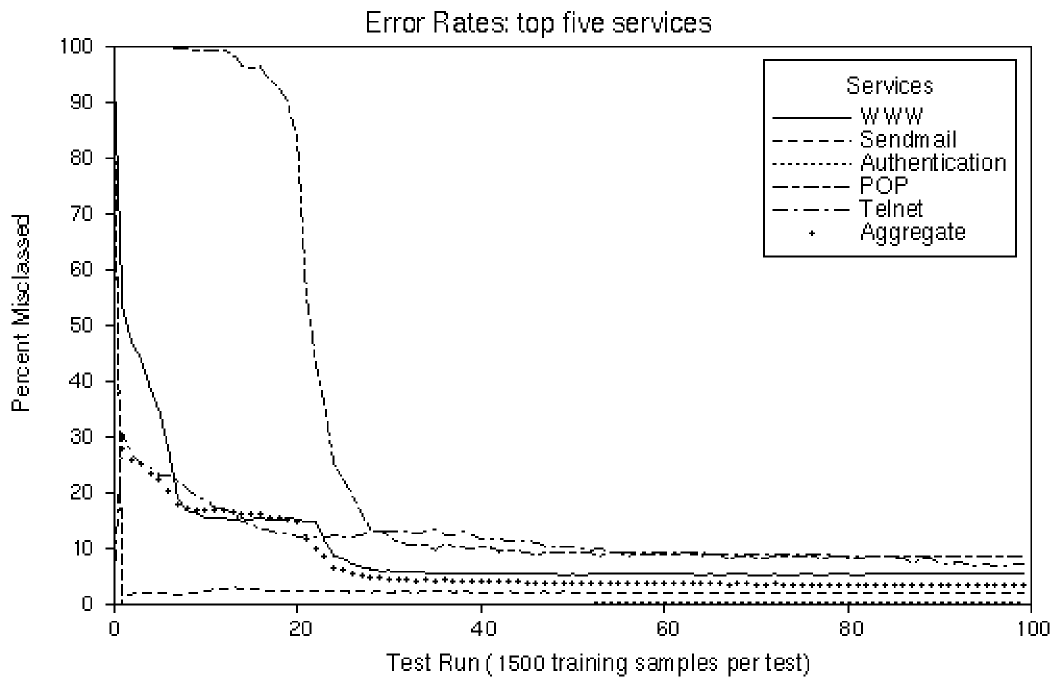


Figure 9

4 UNIX Manual Pages

This section presents manual pages (often referred to as "man pages") for the programs delivered as part of Phase I of the Network Radar STTR effort. These manual pages are also available in an on-line form for UNIX computer systems.

NAME

audit_log – Network Audit Tool

SYNOPSIS

audit_log -tap (PcapTap | FileTap | TcpdumpTap | SnoopTap) [-i <dev-name>] [-a <pkt-file>] [-m <minutes>] [-ports <port-file>] [-pat <pattern-file>] [-serv-filt <file>]

DESCRIPTION

audit_log is a tool to monitor live network traffic or analyze archived packets. *audit_log* identifies network sessions (e.g., TCP/IP connections), tracks the sessions, and performs additional analysis. The additional analysis is performed by specific stream modules (discussed below), each of which generates its own audit trail. These "parallel" audit trails can be reassembled by log_reader(1) to provide a comprehensive, integrated network audit trail.

Currently, *audit_log* supports the TcpStream, RloginStream, TelnetStream, LoginString, RshellStream, and StringStream stream modules. These Stream objects are described below in the STREAM MODULES section.

OPTIONS

-tap (PcapTap | FileTap | TcpdumpTap | SnoopTap)

specifies the type of network tap to use. Each tap type is used to read packets from a different data source. PcapTap is used to monitor live packets from the network. FileTap is used to read NMT packet files. The NMT packet file format is the native format used by the Network Monitoring Toolkit (NMT) suite of tools. TcpdumpTap is used to read packets archived by the tcpdump program. TcpdumpTap can also be used to read live network traffic. This is performed by running tcpdump, writing the packets to standard out, and then piping the results to *audit_log*. Examples are provided below. Finally, SnoopTap is used to read packets archived by the Solaris snoop(1M) program.

-i <dev-name>

directs *audit_log* to use the given file name or network device for input. For example, if you have chosen FileTap, -i my_file will direct *audit_log* to read in the packets from the file my_file. If you are using tcpdump and piping the data to *audit_log*, you can use -i - to direct *audit_log* to read packets from standard-in. Examples are provided below. Eventually this option will also be used to select a specific interface when running on hosts connected to multiple networks.

-a <pkt-file>

directs *audit_log* to archive packets to the given file name. The service filter file can be used to determine which type of packets to be saved. The packets are saved in NMT native packet file format. When reading these packets with an NMT tool supporting multiple Tap types, use the FileTap.

-m <minutes>
specifies how often the audit log files (and packet archive file) should be changed. This option only makes sense when monitoring live network traffic.

-ports <port-file>
specifies a file containing well known server ports. When *audit_log* misses the SYN or SYN-ACK packets of a TCP/IP connection (either because the packets were dropped or the connection was already running when *audit_log* was started), the port numbers in <port-file> will be used to try to identify which host is the client and which is the server.

-pat <pattern-file>
specifies that the strings in the <pattern-file> will be searched for in the connections. Currently, only simple string matching is performed via the Knuth-Morris-Pratt algorithm. This is the same algorithm originally implemented in NSM, ASIM, and NID.

-serv-filt <file>
directs *audit_log* to use the filter file to determine which packets are to be processed, ignored, and/or saved. Packets which are to be saved are written to the file specified by the -a <pkt-file> option.

STREAM MODULES

TcpStream

TcpStream performs the primary analysis of TCP/IP connections. A variety of statistics are collected. These statistics will be used by the NCSR software to identify the service class (e.g., WWW server or Sendmail server) of the connection. TcpStream also identifies duplicate data (from the rebroadcast of data) and missed data (due to dropped packets).

RloginStream

RloginStream identifies the username on the client which initiated the connection. The client’s terminal type and the initial username tried on the server are also recorded. The actual user name used on the server may be different if the user generates a "Login incorrect" and then enters a new user name at a subsequent "login:" prompt. This information is removed from the data stream before passing data to higher level streams.

TelnetStream

TelnetStream records a number of facts exchanged in TELNET negotiations. Facts can (but not necessarily) include information about terminal type, terminal speed, and window size. All TELNET negotiation is identified and removed from the data stream before passing data to higher level streams.

RshellStream

RshellStream identifies the user name on the client performing the remote shell. The user name used on the server, the requested command, and the error port are also recorded. A remote shell connection uses a secondary connection, identified by the error port, to exchange error messages. This information is removed from the data stream before passing data to higher level streams.

LoginStream

LoginStream tries to identify the information the user provides at "login:" and "Password:" prompts. Unfortunately, there is no standard protocol for prompting users for their login name and password, so this stream module is prone to errors.

StringStream

StringStream matches strings in the data stream. The set of strings matched are determined by the file in the -pat <pattern-file> option. StringStream uses the same Knuth-Morris-Pratt algorithm used in original NSM, ASIM, and NID network monitors.

FtpStream

FtpStream records the number (up to 256) of user account names tried, passwords sent, files retrieved, files stored, directories accesses, and directories created. Currently, the success or failure of these actions are not determined. Furthermore, up to the first 10 user account names, passwords, files retrieved, files stored, directories accessed, and files retrieved are recorded. Finally, the number of failed actions (up to 256) are recorded; however, the reason for the failed action is currently not recorded.

FILES

The current working directory should contain the following files:

.FtpClientFSM
.FtpServerFSM
.TelnetFSM
.TelnetSubnegFSM

Optional supporting file include a ports file containing well known server ports; a pattern-file containing a set of strings to match; and a service filter file containing rules to determine which packets should be ignored, processed, and/or saved.

CAVEATS

When using *audit_log* to collect live packets via PcapTap, or when using tcpdump to collect packets from the network, you must be running as root.

USAGE

```
# audit_log -tap PcapTap

# audit_log -tap PcapTap -pat patterns -port ports

# audit_log -tap PcapTap -pat patterns -port ports -a nmt.pkts -serv-filt filter
# audit_log -tap SnoopTap -i snoop_file -pat patterns

# audit_log -tap TcpdumpTap -i dump_file -pat patterns

# tcpdump -s 5000 -w - | audit_log -tap TcpdumpTap -i -

# tcpdump -s 5000 -w dump.pkts
# audit_log -tap TcpdumpTap -i dump.pkts

# snoop -s 5000 -o snoop.pkts
# audit_log -tap SnoopTap -i snoop.pks
```

SEE ALSO

class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

class_extractor – Extract records of a specific class.

SYNOPSIS

class_extractor (-tcp <tcp-log>)+ -server <port> [-c <count>]

DESCRIPTION

class_extractor extracts information about individual sessions destined to servers installed on the specified <port>. This program is part of a series of programs used to train and test a classifier for Non-Cooperative Service Recognition (NCSR). For each matching session, the following information is printed to standard out:

- number of packets sent by the client
- number of packets sent by the server
- number of bytes sent by the client
- number of bytes sent by the server
- ratio bytes/packet sent by the client
- ratio bytes/packet sent by the server
- duration of session (in seconds)

Since the ratio of bytes per packet can be constructed with the number of bytes sent and the number of packets sent, we only have five independent variables for each session. These values will be printed as number strings printed to standard out, with one line for each session.

A "matching session" is a session matching the following constraints:

- confidence = 2 (we are positive that we know which host is the client and which is the server)
- server port = <port>
- at least one packet was sent by the client AND server
- at least one byte was sent by either the client OR server

These constraints ensure that we are using at least relatively normal connections. That is, both the client and the server generated packets, and at least some data was transmitted across the connection. A degenerate connection (one which does not match these constraints) associated with one service would be virtually impossible to distinguish from a degenerate connection associated with another service.

OPTIONS

-tcp <tcp-log>

specifies a TCP log file or a directory containing TCP log files. As indicated by the '+', this command line argument has to be applied one or more times. The log files will be processed in the order provided. If <tcp-log> is a directory containing TCP log files, the files will be processed in alphabetical order.

-server <port>
specifies the server port of interest. Only connections to the given <port> will be matched.

-c <count>
specifies the maximum number of connections to match. The default value is 100.

USAGE

```
% class_extractor -tcp data3 -server 80 -c 5000 > 80.sessions

% class_extractor -tcp TCP.19970321.08 -server 23 > telnet.sesions
```

SEE ALSO

audit_log(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

class_normalizer – normalize each session vector.

SYNOPSIS

```
class_normalizer -f <file-name> -n <normalizer-file>
```

DESCRIPTION

class_normalizer reads in a file of session statistics and prints out their "normalized" values. A file, referenced with the -n option, contains a vector of normalizing statistics. The normalized value for each statistic is defined as follows:

normalized_value = value/normalizer

A normalizing vector can be generated by the program max_extractor(1). For example, the value “average+(3*standard deviation)”, the last line printed by max_extractor, makes a good normalizing vector.

The normalized values (printed as a vector of real numbers, one vector per line) are printed to standard out. Typically you will redirect the output to a file.

OPTIONS

-f <file-name>

specifies a file of session statistics of the form created by class_extractor(1).

-n <normalizer-file>

specifies the file containing a vector of normalizing values. The file should consist of a single line of numbers with the same format as the lines generated by max_extractor.

USAGE

```
% class_normalizer -f 80.sessions -n norm_vec > 80.norm
```

SEE ALSO

audit_log(1), class_extractor(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

log_reader – Read audit_log's Audit Trails

SYNOPSIS

log_reader (-tcp <tcp-log>)+ [-rlogin <rlogin-log>]* [-telnet <telnet-log>] [-rshell <rshell-log>]* [-login <login-log>]* [-ftp <ftp-log>]* [-patlog <pattern-log>]* [-pat <pattern-file>] [-detail]

DESCRIPTION

log_reader provides a simple text display of the audit trails generated by tools such as audit_log(1) and tracker(1). These tools generate parallel audit logs stored in binary formats, and *log_reader* integrates these parallel logs into a single audit trail designed for human consumption.

The key command line argument to *log_reader* is -tcp <tcp-log>. The <tcp-log> can be either a single TCP audit log file or a directory containing TCP log files (and only TCP log files). Every observed TCP/IP connection generates a TCP audit log record, and these audit records are used to identify and "pair-up" audit records generated by the other stream modules. Therefore, at least one <tcp-log> files needs to be included in the command line.

The audit logs will be searched in the order that they are provided on the command line. If the supplied <*-log> name is a directory of log files, each file in the directory will be read one after another in alphabetical order.

OPTIONS

-tcp <tcp-log>

specifies a TCP log file or a directory containing TCP log files. As indicated by the '+', this command line argument has to be applied one or more times. The log files will be processed in the order provided. If <tcp-log> is a directory containing TCP log files, the files will be processed in alphabetical order.

-rlogin <rlogin-log>

specifies an Rlogin log file or a directory containing Rlogin log files. The '*' indicates that this option can be applied zero, one, or many times. If <rlogin-log> is a directory containing Rlogin log files, the files will be processed in alphabetical order.

-telnet <telnet-log>

specifies a Telnet log file or a directory containing Telnet log files. The '*' indicates that this option can be applied zero, one, or many times. If <telnet-log> is a directory containing Telnet log files, the files will be processed in alphabetical order.

-rshell <rshell-log>

specifies an Rshell log file or a directory containing Rshell log files. The '*' indicates that this option can be applied zero, one, or many times. If <rshell-log> is a directory containing Rshell log files, the files will be processed in alphabetical order.

-login <login-log>

specifies a Login log file or a directory containing Login log files. The '*' indicates that this option can be applied zero, one, or many times. If <login-log> is a directory containing Login log files, the files will be processed in alphabetical order.

-ftp <ftp-log>

specifies an FTP log file or a directory containing FTP log files. The '*' indicates that this option can be applied zero, one, or many times. If <ftp-log> is a directory containing FTP log files, the files will be processes in alphabetical order.

-patlog <pattern-log>

specifies a Pattern log file or a directory containing Pattern log files created by the StringStream stream module. The '*' indicates that this option can be applied zero, one, or many times. If <pattern-log> is a directory containing Pattern log files, the files will be processed in alphabetical order. NOTE: the actual strings matched in a session are not kept in these log files, so if you use this option, you need to use the -pat <pattern-file> option as well.

-pat <pattern-file>

specifies the file containing the original patterns looked for in the network sessions. *log_reader* will use this information with the audit records in Pattern log files to determine which strings were actually matched in each session.

-detail

directs *log_reader* to print full details about each session; otherwise, only a one line summary is printed for each session. If you are going to use any of the addition audit log sources (Rlogin, Login, Rshell, or Pattern log files), you should use this option.

FILES

If you use the -patlog option, you must supply the same pattern file, via the -pat <pattern-file> option, which was used by audit_log(1) to generate these log files.

USAGE

```
% log_reader -tcp tcp_dir

% log_reader -tcp TCP.961222.13

% log_reader -tcp TCP.961222.13 -tcp TCP.961222.14 -tcp TCP.961222.15
% log_reader -tcp TCP.961222.13 -login Login.961222.13 -detail
```

```
% log_reader -tcp tcp_dir -rlogin rlogin_dir -rshell rshell_dir -login login_dir -patlog
pat_dir -pat strings -detail
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

max_extractor – Print statistical information from the files.

SYNOPSIS

max_extractor (-f <session-file>)+

DESCRIPTION

max_extractor processes multiple session files created by class_extractor(1) and prints out the following information about each of the variables:

- maximum value
- average value
- standard deviation
- average + 3 standard deviations

According to Chebyshev's Rule, at least 8/9 of all measurements will fall within three standard deviations of the mean (average). The “average + 3 standard deviations” line is often used as the normalizing vector used in class_normalizer(1).

OPTIONS

-f <session-file>

specifies a session file created by class_extractor(1). As indicated by the '+', this command line argument has to be applied one or more times.

USAGE

```
% max_extractor -f 80.sessions -f 25.sessions
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

net4 – build and test a neural network classifier

SYNOPSIS

net4 (-c <train-file> <test-file>)+ [-s <seed>] [-l <rate>] [-e <rate>] [-h <hidden>] [-test <tests>] [-train <num>]

DESCRIPTION

net4 reads in a series of training and testing data sets, builds a neural network classifier, and then repeatedly trains and tests the classifier's ability to correctly classify network sessions.

net4 prints a line of data for each round of experiments. The first number indicates the test round, which starts at zero. The number of training sets given to the neural network at a particular stage is the test round number multiplied by the number training samples per round. The number of training samples per round is set by the -train option; the default value is 100.

The last number is the percentage of test sessions for all classes which were misclassified. The numbers between the first and last are the percent of test sessions misclassified for each service class. The command line below is an example run:

```
% net4 -c 80.norm.train 80.norm.test -c 25.norm.train 25.norm.test
```

where 80.norm.train is training sessions for HTTP (WWW), and 80.norm.test is the testing sessions for HTTP (see train_test(1)); 25.norm.train and 25.norm.test are the training and testing sets for Sendmail.

A line from the output is:

```
20   0.2    3.2    1.7
```

These numbers indicate that 20 training rounds have taken place, 0.2% of HTTP sessions were misclassified; 3.2% of Sendmail sessions were misclassified; and 1.7% of all the testing sessions were misclassified.

OPTIONS

-c <train-file> <test-file>

specifies training and testing file sets for a particular service class. This option should be applied at least twice (trying to classify a single service doesn't make sense). While the training and testing files could be the same file, a better analysis of the classifier uses disjoint training and testing data sets. See train_test(1) for creating disjoint training and testing data sets.

-s <seed>

specifies the seed used to initialize the random number generator.

-l <rate>

specifies the learning rate for the neural network classifier. A high learning rate will accelerate the learning of the neural network, but it may also lead to oscillations as the error rate becomes smaller. The default value is 0.05.

-e <rate>

specifies the epsilon value used when determining the error values for the output nodes. Ideally, the output for output node two for a input session belonging to class two should be 1, and all other output nodes should be 0. However, the values <1 - e> and <e> are often used in place of 1 and 0, where e is the epsilon value. The use of the epsilon value is based on some mathematical behavior of the neural network. The default value is 0.05.

-h <hidden>

specifies the number of nodes in the hidden layer. The default value is 5.

-tests <tests>

specifies the number of training/testing rounds to perform. The default value is 100.

-train <num>

specifies the number of training sets to be used for each training round. If three services are being compared, a single training set consists of three sessions: one for each service type. The default value is 100.

USAGE

```
% net4 -c 80.norm.train 80.norm.test -c 25.norm.train 25.norm.test

% net4 -c 80.norm.train 80.norm.test -c 25.norm.train 25.norm.test -train 250 -l 0.01 -
s 100

% net4 -c 80.norm.train 80.norm.test -c 25.norm.train 25.norm.test -c 113.norm.train
113.norm.test -c 110.norm.train 110.norm.test -c 23.norm.train 23.norm.test -l 0.01 -
train 1500 -s 500 -h 2
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

net5 – build and test a neural network classifier

SYNOPSIS

net5 (-c <train-file> <test-file>)+ [-s <seed>] [-l <rate>] [-e <rate>] [-h <hidden>] [-test <tests>] [-train <num>]

DESCRIPTION

net5 is essentially the same program as net4(1) except intermediate error information is not printed and additional error information is printed at the end.

net5 first prints two lines representing the misclassification rates before training the neural network and after training the neural network. The first value of each line represents the percent of test samples from the first service class which were misclassified. Similarly, the second value represents the percent of test samples from the second service class which were misclassified, and so on. The final number is the percent of sessions misclassed for all services.

Following the first two lines, classification statistics for each service class are provided. In particular, the following information is displayed:

- The number of test sessions which were calculated to be in each class. A total of ten columns are printed, numbered from zero to nine. No samples should be classified in class 0. If five service types are given to the classifier (via the -c option), all sample sessions should be calculated to belong to classes in the range from one to five.
- The number of test sessions which were calculated to be in each class when the test sessions were correctly classified. In essence, this simply states the number of sessions correctly classified.
- The number of test sessions which were calculated to be in each class when the sessions were incorrectly classified. In other words, when a session is misclassified, what class is it calculated to be in. This could be helpful when trying to find new statistics to gather which might help reduce the number of misclassified sessions. For example, if the vast majority of misclassified sessions belonging to class X are classified as class Y, then you should try to find a statistic which discriminates between X and Y.
- The number of test sessions with the "confidence value" in the given range. The first column represents the number of sessions which had a confidence value from 0.0 to 1.0. The second column represents the confidence from 0.1+ to 0.2. The last column represents the confidence from 0.9+ to 1.0. The "confidence value" is simply the output value for the node with the highest output (that is, the node which determines the class the session belongs to).

- The number of sample sessions with the "confidence value" in the given range when the sample was correctly classified. We would expect that in general, correctly classified sessions should have a high confidence value.
- The number of sample sessions with the "confidence value" in the given range when the sample was incorrectly classified. We would expect that in general, samples misclassified will have a lower confidence value than samples which were correctly classified.

OPTIONS

-c <train-file> <test-file>
specifies training and testing file sets for a particular service class. This option should be applied at least twice (trying to classify a single service doesn't make sense). While the training and testing files could be the same file, a better analysis of the classifier uses disjoint training and testing data sets. See train_test(1) for creating disjoint training and testing data sets.

-s <seed>
specifies the seed used to initialize the random number generator.

-l <rate>
specifies the learning rate for the neural network classifier. A high learning rate will accelerate the learning of the neural network, but it may also lead to oscillations as the error rate becomes smaller. The default value is 0.05.

-e <rate>
specifies the epsilon value used when determining the error values for the output nodes. Ideally, the output for output node two for a input session belonging to class two should be 1, and all other output nodes should be 0. However, the values <1 - e> and <e> are often used in place of 1 and 0, where e is the epsilon value. The use of the epsilon value is based on some mathematical behavior of the neural network. The default value is 0.05.

-h <hidden>
specifies the number of nodes in the hidden layer. The default value is 5.

-tests <tests>
specifies the number of training/testing rounds to perform. The default value is 100.

-train <num>
specifies the number of training sets to be used for each training round. If three services are being compared, a single training set consists of three sessions: one for each service type. The default value is 100.

USAGE

```
% net5 -c 80.norm.train 80.norm.test -c 25.norm.train 25.norm.test
% net5 -c 80.norm.train 80.norm.test -c 25.norm.train 25.norm.test -train 250 -l 0.01 -s 100
```

```
% net5 -c 80.norm.train 80.norm.test -c 25.norm.train 25.norm.test -c 113.norm.train 113.norm.test -c 110.norm.train 110.norm.test -c 23.norm.train 23.norm.test -l 0.01 -train 1500 -s 500 -h 2
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

print_pkts – print packet records.

SYNOPSIS

print_pkts -input <pkt-file> [-output <txt-file>] [-ether <indent-val>] [-ip <indent-val>] [-tcp <indent-val>] [-data oct | dec | hex] [-tcpcheck] [-tcpbug]

DESCRIPTION

print_pkts prints the packets from the input file. The Ethernet, IP, and TCP headers, and the TCP payload, or data, are all displayed. The TCP payload data is displayed in a column, the first field of which is the hexadecimal value for each byte, and the second field is the ASCII character (if printable).

print_pkts' main purpose is very low-level analysis of network activity. For example, you can observe the exact behavior of a server under a SYN-flooding attack.

OPTIONS

-input <pkt-file>

specifies the input packet file. Only a single packet file can be specified, and it has to be in the native NMT packet file format. transfer(1) can be used to translate from one packet file format to the NMT packet file format.

-output <txt-file>

directs *print_pkts* to write the output to the file <txt-file>.

-ether <indent-val>

specifies the number of spaces that Ethernet information will be indented. The default value is 0 (flush with the left margin).

-ip <indent-val>

specifies the number of spaces that IP information will be indented. The default value is 16.

-tcp <indent-val>

specifies the number of spaces that TCP information will be indented. The default value is 32.

-data oct|dec|hex

specifies the format used when printing the TCP payload data. By default, the byte values are printed in hexadecimal format.

-tcpcheck

directs *print_pkts* to print each stage during the calculation of the TCP checksum. This option has been used to identify a strange behavior on some Solaris systems. In particular, a packet sent to the Sun performing the monitoring will have its payload modified in at least one condition -- in particular, a carriage return will be modified to a new line. This results in a checksum which is off by the value 0x0300.

-tcpbug

directs *print_pkts* to allow TCP packets with a checksum that is off by the value 0x0300 to continue being processed. (see the -tcpcheck option)

LIMITATIONS

print_pkts can only print packet file stored in NMT's native packet file format. If the original packets are stored in another format, the program transfer(1) can be used to create an NMT packet file. All packets must be Ethernet packets. Any higher layer protocol on top of Ethernet is acceptable (e.g., ARP, RARP, and UDP), but only Ethernet, IP, and TCP information will be printed.

USAGE

```
% print_pkts -input 8.session

% print_pkts -input 8.session | more

% print_pkts -input 8.session -output 8.print

% print_pkts -input 3.session -ip 0 -tcp 0 -data dec
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_session(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

print_session – print the data in a session

SYNOPSIS

print_session -input <pkt-file> [-out <file-name>] [-out2 <root-name>] [-rlogin] [-telnet [<state-file> <state-file>]] [-client <state-file>] [-server <state-file>] [-tcpbug]

DESCRIPTION

print_session prints the data in the network session. *print_session* provides two main capabilities. First, it can generate transcript-like files similar to those created by the UNIX script(1) command or the NSM/ASIM/NID transcript program. These transcript-like files can be used to determine what commands were used and what data was accessed by a user using login services such as rlogin or telnet. Second, *print_session* can present a detailed stream of the bytes transmitted by both the client and server. These results can be very useful when analyzing different protocols (e.g., to build or tune an NMT Stream module) or understanding a new network-based attack.

print_session's default behavior is the second one – presenting a stream of data transmitted by the client and server. The data from a session is printed in two major columns. The left column contains the data transmitted by the client, and the right column contains the data transmitted by the server. Each major column consists of four sub-columns. The first three sub-columns are a byte's value printed in octal, hexadecimal, and decimal formats. The fourth sub-column prints the ASCII character code of the byte (if it is printable).

print_session displays the value of the data in multiple formats because different types of analysis require that the data be printed in different formats. For example, the TELNET protocol (as well as the various TELNET subnegotiation protocols), as defined in the various RFCs, use decimal values (0-255) when defining their protocols. However, ASCII tables are usually printed in octal and/or hexadecimal formats (see ascii(5)). Finally, the ASCII character code, or "letter", is printed for reading normal data entered by the user or displayed on the user's terminal.

print_session is largely controlled by finite state machine (FSM) configuration files. For example, the default behavior comes from using the FSM state file named .StreamFSM. This file must be present in the current working directory. *print_session*'s behavior can be altered by using different configuration files. For example, by using the FSM state files .nlFSM and .cr-nlFSM, *print_session* will generate transcript-like output.

When using the -telnet option, *print_session* will also use FSM configuration files to control the processing of TELNET protocol information. Telnet uses two FSM files: one for controlling Telnet’s primary state machine and one for controlling Telnet’s subnegotiation state machine. The default FSM files used are .TelnetFSM and .TelnetSubnegFSM. While you are able to change these configuration files, it is not recommended.

When using the -telnet or -rlogin arguments, the TELNET and/or RLOGIN protocol will be stripped from the data stream before printing the data. For example, TELNET negotiation data will not be printed in the transcript files. This is important if you want to present what the user actually types (as opposed to including the data the telnet program actually sent).

OPTIONS

-input <pkt-file>

specifies the input packet file. Only a single packet file can be used, and it has to be in the native NMT packet file format. Furthermore, the <pkt-file> should only contain packets from a single network session. Typically, this file is created with the session(1) program.

-out <file-name>

specifies an output file to which both the client and server data will be written. This option is usually used when printing data streams (as opposed to creating transcript-like output). This option should not be used with the -out2 option.

-out2 <root-name>

specifies the root name for the files to which the client and server data will be written. The client data will be written to the file <root-name>.client, and the server data will be written to the file <root-name>.server. For example, if <root-name> is "session1", the two output files will be "session1.client" and "session1.server". This option is usually used when creating transcript-like output. This option should not be used with the -out option.

-telnet [<state-file> <state-file>]

directs *print_session* to remove the data associated with the TELNET protocol. For example, if the session is a telnet session, there is often a number of bytes at the beginning of the connection which was neither sent by the user nor login daemon. The default FSM state files used are .TelnetFSM and .TelnetSubnegFSM. Both of these files must be present in the current working directory. You can specify alternative FSM configuration files, but it is not recommended.

-rlogin

directs *print_session* to remove the data associated with the RLOGIN protocol. In particular, the user names used on the client and server as well as the terminal type are stripped from the data. This is data which was neither sent by the user or the login daemon on the server.

-client <state-file>

directs *print_session* to use the FSM state file <state-file> to control the printing of information from the client. The default state file .StreamFSM prints each byte from the client in octal, hexadecimal, and decimal formats. Also, if the byte is a printable ASCII character, the "letter" is printed. The FSM state file .FileClientFSM can be used to create a transcript-like output of the keystrokes from the client. In particular, the file .FileClientFSM is used for clients which generate a single new line byte when the return key is pressed. You should probably use this option in conjunction with the -server and -out2 options.

-server <state-file>

directs *print_session* to use the FSM state file <state-file> to control the printing of information from the server. The default state file .StreamFSM prints each byte from the server in octal, hexadecimal, and decimal formats. Also, if the byte is a printable ASCII character, the "letter" is printed. The FSM state file .FileServerFSM can be used to create a transcript-like output of the data generated by the server. In particular, the file .FileServerFSM is used for servers which generate both carriage return and new line bytes every time a new line is created. You should probably use this option in conjunction with the -client and -out2 options.

-tcpbug

directs *print_session* to allow TCP packets with a checksum that is off by the value 0x0300 to continue being processed. This option has been provided because of a strange behavior observed on some Solaris systems. In particular, a packet sent to the Sun performing the monitoring will have its payload modified in at least one condition -- a carriage return byte will be modified to a new line byte. This results in a checksum which is off by the value 0x0300.

FILES

The current working directory should contain the following files:

.TelnetFSM
.TelnetSubnegFSM
.StreamFSM

Furthermore, to support transcript-like output, you should have the following additional files in your directory.

.nlFSM
.cr-nlFSM

LIMITATIONS

print_session has a number of limitations. First, the input file needs to be a native NMT packet file. Second, the file is expected to contain only packets from a single session. Both of these first two limitations are not a problem if the packet file was created with the session(1) program. Third, if the session file does not start with a SYN or SYN-ACK packet, the client and server identities may be reversed. Fourth, several files are expected to be in your current working directory (see FILES above).

USAGE

```
% print_session -input 8.session

% print_session -input 8.session | more
% print_session -input 8.session -out 8.print
```

```
% print_session -input 8.session -out2 8.print
```

```
% print_session -input 8.session -out2 8.script -telnet -client .nlFSM -server .cr-nlFSM
```

```
% print_session -input 9.session -out2 9.script -rlogin -client .nlFSM -server .cr-lnFSM -tcpbug
```

```
% print_session -input 3.session -out2 3.script -client .cr-lnFSM -server .cr-lnFSM
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

session – extract a session's packets.

SYNOPSIS

session -tcp <tcp-log> -input <pkt-file> -output <pkt-file> -id <session-ID> [-tcpbug]

DESCRIPTION

session extracts a network session's packets from the input packet file containing [possibly] many sessions and saves these packets to the output file. This output file can then be processed by other programs such as print_session(1). The session is identified by the <session-ID> and the <tcp-log>.

Currently, *session* has a number of limitations. First, only a single TCP log file can be specified (as opposed to a set of TCP log files or a directory of TCP log files). Second, the input packet file is expected to be in the standard NMT packet file format. If your packet archive is in another format, you can use transfer(1) to translate the packets into the native NMT packet file format.

Third, all the arguments listed except -tcpbug must be provided.

OPTIONS

-tcp <tcp-log>
specifies the TCP log file which contains the audit records of interest. As has been mentioned, only a single TCP log file can be specified.

-input <pkt-file>
specifies the input packet file. Only a single packet file can be specified, and it has to be in the native NMT packet file format. transfer(1) can be used to translate packets in another format to the native NMT file format.

-output <pkt-file>
specifies the file to which the session's packets will be saved. The file format is NMT native packet file format. This file can then be processed by other tools such as print_session(1) and print_pkts(1).

-id <session-ID>
specifies the ID of the session in the TCP log file you want to extract. *session* will find the session audit record in the TCP log file and then extract the packets which match that particular session from the input file. log_reader(1) will print the IDs of the sessions contained in the <tcp-log> file.

-tcpbug
directs print_pkts to allow TCP packets with a checksum that is off by the value 0x0300 to continue being processed. This option has been provided because of a strange behavior observed on some Solaris systems. In particular, a packet sent to the Sun performing the monitoring will have its payload modified in at least one condition -- a carriage return byte will be

modified to a new line byte. This results in a checksum which is off by the value 0x0300.

USAGE

```
% session -tcp TCP.961222.13 -input pkts.file -output 8.session -id 8

% session -tcp TCP.961222.13 -input pkts.file -output 8.session -id 8 -tcpbug
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), summary(1), tracker(1), train_test(1), transfer(1)

NAME

summary – statistical summary of connection logs.

SYNOPSIS

summary (-tcp <tcp-log>)+ (-sort <sort-class>)+ [-nosort]

DESCRIPTION

summary creates a statistical summary of TCP session logs. The specific statistics generated will be determined by choice and order of the sort classes. *summary*'s default behavior is to sort the statistics such that the most frequently matched items will appear first.

For example, if the only sort-value used is server-port, *summary* will identify all server port values used (e.g., 80, 23, and 25), and the number of sessions to each server port value is counted. Once all the sessions are processed, the server-port values are sorted based on the number of sessions to each one. Finally, the server port values and their associated counts are presented with the server port values with the highest counts presented first.

Multiple sort values can be used to reveal interesting behavior. For example, if server-net, server-port, and server-addr are used, you can identify the most popular destination networks for connections, the most popular server port values in each of those networks, and finally, the most popular hosts for each server port value.

OPTIONS

-tcp <tcp-log>

specifies a TCP log file or a directory containing TCP log files. As indicated by the '+', this command line argument has to be applied one or more times. The log files will be processed in the order provided. If <tcp-log> is a directory containing TCP log files, the files will be processed in alphabetical order.

-sort <sort-class>

specifies a <sort-class> on which to sort. The available sort classes are server-net, server-addr, server-port, client-net, and client-addr. These classes are described below. As indicated by the '+', this command line argument has to be applied one or more times. The sort classes will be applied in the order in which they appear on the command line. While technically you can apply a specific <sort-class> more than once, it would not make sense.

-nosort

directs *summary* not to sort the information based on the number of sessions. For example, if you sort on server-net, then information about the network 128.115.0.0 will be presented before network 128.120.0.0 no matter how many sessions were destined for each network.

SORT CLASSES

server-net

Sort on the network address of the server. An Internet address can be broken into two parts: a network part and a local part. For example, the Internet address 128.120.56.1 has a network address of 128.120.0.0. Sorting on the server-net will identify all the network addresses for all servers observed.

server-addr

Sort on the entire Internet address of the server (as opposed to just the network portion of the address). Sorting on server-addr will identify all hosts running a servers. Note: a single host can run several servers (e.g., WWW, Sendmail, and telnet).

server-port

Sort on the server's port. Generally, the server port identifies the type of server it is. For example, ports 80, 25, and 23 usually indicate WWW, Sendmail, and telnet servers respectively. However, as will become apparent when using this program, a substantial number of servers run an unlisted ports.

client-net

Sort on the client's network address (see server-net for a description of network addresses). This can be useful to find what type of sites tend to connect into your site.

client-addr

Sort on the client's Internet address.

USAGE

```
% summary -tcp tcp-dir -sort server-net -server-port

% summary -tcp TCP.961229.16 -tcp TCP.961229.17 -sort server-port
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), tracker(1), train_test(1), transfer(1)

NAME

tracker – X-windows Incident Handling Support Tool

SYNOPSIS

tracker -tap (PcapTap | TcpdumpTap) [-i <dev-name>] [-a <pkt-file> [-m <minutes>] [-ports <ports-file>] [-pat <pattern-file>] [-serv-filt <svc-filter-file>] [-tcpbug]

DESCRIPTION

tracker is designed to identify and track a limited number of connections. For example, if you know of a system or small set of systems which have been broken into, you can use *tracker* to watch certain types of connections (e.g., telnet and rlogin) into and out of those hosts.

tracker has two windows. The main window supports a status line, a details area, a list of connections being tracked, and a row of control buttons. The status line displays the number of connections currently being tracked and the total number of connections observed for this run of the program. The details area displays additional information about a selected connection. The details displayed will depend on the type of connection (e.g., rlogin, telnet, ftp, etc.).

The list of connections displays the currently tracked connections. Each connection is represented by a single line consisting of the connection's ID, client IP address, server IP address, and server port (which usually indicates the service type). When you click on a line in the list, details about the connection will be displayed in the details area.

Finally, the control area supports the *track*, *kill* and *send to client:* buttons and a pop-up menu. When you press *track*, the currently selected connection will be tracked. That is, input and output will be displayed in the Tracker window. When you press *kill*, the currently selected connection will be killed. This is performed by sending reset (RST) packets to the client and server. When you press *send to client:*, the message in the pop-up menu will be sent to the client of the currently selected connection. You may want to send the message to a user's screen prior to killing their connection.

The pop-up menu lets you select between three messages which will be sent to the user's screen (assuming the connection is a login-type connection). The messages are “Go Away”, “Security Violation”, and “Network Error.” All three messages will clear the screen. The first two messages are centered on the screen (for a standard 24 rows X 80 columns screen), and the Network Error message is displayed in the lower left hand corner. Once the message has been sent (by pressing the *Send to client:* button) the user may lose control over the connection and the connection will be marked as "hijacked".

The second window is a display window – it displays the contents from the client (typically key strokes for login connections) and server (typically output displayed to the attacker's screen for login connections). The current display window is tailored for login-type (telnet, rlogin, rshell) services.

OPTIONS

-tap (PcapTap | TcpdumpTap)
specifies the type of network tap to use. The only taps currently used to support live monitoring are PcapTap and TcpdumpTap. When using the TcpdumpTap, you have to run tcpdump, write the data to stdout, and then direct the TcpdumpTap to read from stdin. Examples are provided below.

-i <dev-name>
specifies that the provided input device be used. For example, if you are running a dual homed host (one with two Ethernet interfaces, each running on a different network), you specify which network device you want to use (e.g., ie0 or ie1). When running with TcpdumpTap, the <dev-name> should be –, directing the tap to read from stdin.

-a <pkt-file>
directs *tracker* to save/archive packets to the file <pkt-file>. The service filter file can be used to determine which packets are to be saved.

-m <minutes>
directs *tracker* to start new audit_log files every <minutes> minutes.

-ports <port-file>
directs *tracker* to use the well known port values in <port-file> to help identify which host is the client and which is the server. Normally, the SYN or SYN-ACK packets are used to properly identify which host is the server, but if *tracker* misses both of these packets (which can happen if the connection was already established before *tracker* was started or both packets are simply missed by the monitoring host), *tracker* will check to see if one of the ports in the connection matches the "well known" ports in <port-file>. If so, then *tracker* assumes that is the server.

-pat <pattern-file>
directs *tracker* to search for the strings in the <pattern-file>. Matches on the patterns will be displayed in the details area when a connection is selected.

-serv-filt <svc-filter-file>
directs *tracker* to use the filter file to determine which packets are to be processed, ignored, and/or saved. Packets which are to be saved are written to the file specified by the -a <pkt-file> option.

-tcpbug
directs *tracker* to allow TCP packets with a checksum that is off by the value 0x0300 to continue being processed. This option has been provided because of a strange behavior observed on some Solaris systems. In particular, a

packet sent to the Sun performing the monitoring will have its payload modified in at least one condition – a carriage return byte will be modified to a new line byte. This results in a checksum which is off by the value 0x0300.

CAVEAT

tracker must be run as root (or someone must set it SUID root) when running with PcapTap. If tcpdump is used to capture packet, tcpdump must be run as root or SUID root.

USAGE

```
# tracker -tap PcapTap

# tracker -tap PcapTap -pat patterns -port ports

# tracker -tap PcapTap -pat patterns -serv-filt filter

# tcpdump -s 5000 -w - | tracker -tap TcpdumpTap -i -
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), train_test(1), transfer(1)

NAME

train_test – create training and testing data sets

SYNOPSIS

```
train_test -f <file-name> [-p <percent>] [-s <seed>]
```

DESCRIPTION

train_test reads in a file of session statistics and creates two disjoint files of session statistics, one used for training a classifier and one for testing the classifier. A session vector will go into the train file with probability <percent> and into the test file with probability (1.0 - <percent>). The default value for <percent> is 66%, but this can be changed on the command line.

The training and testing files created are <file-name>.train and <file-name>.test, respectively.

train_test is designed to operate on files created by class_extractor(1) or class_normalizer(1). However, the program actually operates on a line at a time, so *train_test* can be used to separate any text file into two files.

OPTIONS

-f <file-name>
specifies the file of session statistics of the form created by class_extractor(1) or class_normalizer(1).

-p <percent>
specifies the probability for which the session vectors will go into the train file. In other words, roughly <percent> of the connections in <file-name> will go into <file-name>.train. The default value is 66%

-s <seed>
specifies the seed used to initialize the random number generator.

USAGE

```
% train_test -f 80.norm

% train_test -f 23.norm -p 50 -s 100

% train_test -f 23.norm -p 50 -s 200
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), transfer(1)

NAME

transfer – Transfer Packet Archive File to NMT Packet Format

SYNOPSIS

transfer -tap (PcapTap | FileTap | TcpdumpTap | SnoopTap) [-i <dev-name>] [-a <pkt-file>] [-c <count>]

DESCRIPTION

transfer translates packets in a file archive (either Tcpdump, Snoop, or NMT File format archives) to NMT's standard packet archive format. In addition to translating packets from an archive, *transfer* can actually be used to grab packets off the network, but this is not its intended purpose. This function is needed because some tools including session(1), print_session(1), and print_pkts(1) assume their input files are in the standard NMT packet format.

OPTIONS

-tap (PcapTap | FileTap | TcpdumpTap | SnoopTap)

specifies the type of network tap to use. TcpdumpTap and SnoopTap are used to translate tcpdump and snoop archive files respectively. TcpdumpTap can also be used to read packets directly from the network. FileTap reads standard NMT packet archives. PcapTap reads packets off the network.

-i <dev-name>

specifies the input device to be used. As designed, <dev-name> should be a file containing network packets, but you can use it to read packets directly from the network.

-a <pkt-file>

specifies the file to which packets are to be archived/saved. While this is an option, it would not make a lot of sense to run *transfer* without it.

-c <count>

directs *transfer* to transfer at most <count> packets into the archive.

USAGE

```
% transfer -tap SnoopTap -i pkts.snoop -a pkts.nmt

% transfer -tap TcpdumpTap -i pkts.tcpdump -a pkts.nmt

% transfer -tap FileTap -i pkts.nmt -a 10pkt.nmt -c 10

% transfer -tap PcapTap -a 20pkt.nmt -c 20

% tcpdump -s 5000 -w - | audit_log -tap TcpdumpTap -i - -c 20
```

SEE ALSO

audit_log(1), class_extractor(1), class_normalizer(1), log_reader(1), max_extractor(1), net4(1), net5(1), print_pkts(1), print_session(1), session(1), summary(1), tracker(1), train_test(1)

5 Example Transcripts

This section presents transcripts showing the use of many of the of the programs delivered as part of the Phase I STTR Network Radar effort.

This section demonstrates the processing of Sendmail data. In particular, the tools audit_log(1), log_reader(1), transfer(1), session(1), print_session(1) are demonstrated. Text entered by the user is bolded. In order to reduce the amount of text presented, some information has been deleted as indicated by the label **[deleted text]**. Comments are presented in greyed text.

We begin by examining the contents of the current working directory. The directory contains a number of configuration files, two packet files (exp4.snoop and exp4.tcpdump), and the typescript file (the file recording this session).

```
yoda% ls -aC
.                  .TelnetFSM         exp4.snoop         typescript
..                 .TelnetSubnegFSM   exp4.tcpdump
.StreamFSM         dot.FileNL.CR-FSM  patterns
```

audit_log processes the raw packets in the file exp4.snoop, creating several parallel audit files (see the information following the subsequent "ls" command). audit_log can process several packet file formats as well as packets directly from the network.

```
yoda% audit_log -tap SnoopTap -i exp4.snoop -pat patterns
setting tap to SnoopTap
link type: 4
SnoopTap::tapRead: couldn’t read header

yoda% ls -lt
total 58
-rwxr-xr-x   1 heberlei staff          0 Jan 28 17:51 Login.970128.17
-rwxr-xr-x   1 heberlei staff          0 Jan 28 17:51 Pattern.970128.17
-rwxr-xr-x   1 heberlei staff          0 Jan 28 17:51 Rlogin.970128.17
-rwxr-xr-x   1 heberlei staff          0 Jan 28 17:51 Rshell.970128.17
-rwxr-xr-x   1 heberlei staff        632 Jan 28 17:51 TCP.970128.17
-rwxr-xr-x   1 heberlei staff          0 Jan 28 17:51 Telnet.19970128.17
-rw-r--r--   1 heberlei staff         43 Jan 28 17:50 typescript
-rw-r--r--   1 heberlei staff         65 Jan 27 21:33 dot.FileNL.CR-FSM
-rw-r--r--   1 heberlei staff         83 Jan 27 21:11 patterns
-rw-r--r--   1 heberlei staff      11722 Jan 26 16:44 exp4.tcpdump
-rw-r--r--   1 root     other      12540 Jan 26 15:33 exp4.snoop
```

log_reader processes the various parallel audit files created by audit_log and presents the information in various levels of detail. As used here, log_reader displays the least amount of information. In this example, four network sessions are displayed, each to a server residing at port 25–the port usually reserved for Sendmail.

```
yoda% log_reader -tcp TCP.970128.17
   0     128.120.56.2  -->     128.120.56.1   25
   1     128.120.56.2  -->     128.120.56.1   25
   3    128.120.56.50  -->     128.120.56.1   25
   2     128.120.56.2  -->     128.120.56.1   25
```

transfer is used to translate packets from one file format to the native Network Monitoring Toolkit (NMT) format. Once the data is translated into this format, the data can be processed with additional NMT tools. In this case, the new data is saved as exp4.nmt. Following the use of transfer, the session command is used to extract the data for a single network session, the session with ID=1. This new file, 1.session, is in the NMT packet file format.

```
yoda% transfer -tap SnoopTap -i exp4.snoop -a exp4.nmt
link type: 4
SnoopTap::tapRead: couldn’t read header
Done set at location 1
113 packets transferred

yoda% session -tcp TCP.970128.17 -input exp4.nmt -output 1.session -id 1
Done!
no more packets
```

print_session displays a session’s data in several ways. The first use shown below is the default behavior. Two main columns of data are displayed. The first column displays the data transmitted by the client, and the second column displays the data transmitted by the server. As described in print_session’s manual page, each byte of data is displayed in four different formats: octal, hexadecimal, decimal, and the ASCII character code. This default behavior is excellent for understanding application protocols and client/server interactions. As can be seen by this example, sessions presented in this method can be voluminous. In fact, some data has actually been deleted.

```
yoda% print_session -input 1.session
                                        [ 62]  [ 32]  [ 50]  2
                                        [ 62]  [ 32]  [ 50]  2
                                        [ 60]  [ 30]  [ 48]  0
                                        [ 40]  [ 20]  [ 32]
                                        [171]  [ 79]  [121]  y
                                        [157]  [ 6f]  [111]  o
                                        [144]  [ 64]  [100]  d
                                        [141]  [ 61]  [ 97]  a
                                        [ 56]  [ 2e]  [ 46]  .
                                        [ 40]  [ 20]  [ 32]
                                        [123]  [ 53]  [ 83]  S
                                        [145]  [ 65]  [101]  e
                                        [156]  [ 6e]  [110]  n
                                        [144]  [ 64]  [100]  d
                                        [155]  [ 6d]  [109]  m
                                        [141]  [ 61]  [ 97]  a
                                        [151]  [ 69]  [105]  i
                                        [154]  [ 6c]  [108]  l
                                        [ 40]  [ 20]  [ 32]
                                        [123]  [ 53]  [ 83]  S
                                        [115]  [ 4d]  [ 77]  M
                                        [111]  [ 49]  [ 73]  I
                                        [ 55]  [ 2d]  [ 45]  -
                                        [ 70]  [ 38]  [ 56]  8
                                        [ 56]  [ 2e]  [ 46]  .
                                        [ 66]  [ 36]  [ 54]  6
                                        [ 57]  [ 2f]  [ 47]  /
                                        [123]  [ 53]  [ 83]  S
                                        [115]  [ 4d]  [ 77]  M
                                        [111]  [ 49]  [ 73]  I
```

```
                 [ 55]  [ 2d]  [ 45]  -
                 [123]  [ 53]  [ 83]  S
                 [126]  [ 56]  [ 86]  V
                 [122]  [ 52]  [ 82]  R
                 [ 64]  [ 34]  [ 52]  4
                 [ 40]  [ 20]  [ 32]
                 [162]  [ 72]  [114]  r
                 [145]  [ 65]  [101]  e
                 [141]  [ 61]  [ 97]  a
                 [144]  [ 64]  [100]  d
                 [171]  [ 79]  [121]  y
                 [ 40]  [ 20]  [ 32]
                 [141]  [ 61]  [ 97]  a
                 [164]  [ 74]  [116]  t
                 [ 40]  [ 20]  [ 32]
                 [123]  [ 53]  [ 83]  S
                 [165]  [ 75]  [117]  u
                 [156]  [ 6e]  [110]  n
                 [ 54]  [ 2c]  [ 44]  ,
                 [ 40]  [ 20]  [ 32]
                 [ 62]  [ 32]  [ 50]  2
                 [ 66]  [ 36]  [ 54]  6
                 [ 40]  [ 20]  [ 32]
                 [112]  [ 4a]  [ 74]  J
                 [141]  [ 61]  [ 97]  a
                 [156]  [ 6e]  [110]  n
                 [ 40]  [ 20]  [ 32]
                 [ 61]  [ 31]  [ 49]  1
                 [ 71]  [ 39]  [ 57]  9
                 [ 71]  [ 39]  [ 57]  9
                 [ 67]  [ 37]  [ 55]  7
                 [ 40]  [ 20]  [ 32]
                 [ 61]  [ 31]  [ 49]  1
                 [ 65]  [ 35]  [ 53]  5
                 [ 72]  [ 3a]  [ 58]  :
                 [ 63]  [ 33]  [ 51]  3
                 [ 62]  [ 32]  [ 50]  2
                 [ 72]  [ 3a]  [ 58]  :
                 [ 62]  [ 32]  [ 50]  2
                 [ 65]  [ 35]  [ 53]  5
                 [ 40]  [ 20]  [ 32]
                 [ 55]  [ 2d]  [ 45]  -
                 [ 60]  [ 30]  [ 48]  0
                 [ 70]  [ 38]  [ 56]  8
                 [ 60]  [ 30]  [ 48]  0
                 [ 60]  [ 30]  [ 48]  0
                 [ 15]  [  d]  [ 13]
                 [ 12]  [  a]  [ 10]
[110]  [ 48]  [ 72]  H
[105]  [ 45]  [ 69]  E
[114]  [ 4c]  [ 76]  L
[117]  [ 4f]  [ 79]  O
[ 40]  [ 20]  [ 32]
[157]  [ 6f]  [111]  o
```

```
[142]  [ 62]  [ 98]  b
[151]  [ 69]  [105]  i
[ 55]  [ 2d]  [ 45]  -
[167]  [ 77]  [119]  w
[141]  [ 61]  [ 97]  a
[156]  [ 6e]  [110]  n
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
                                        [ 62]  [ 32]  [ 50]  2
                                        [ 65]  [ 35]  [ 53]  5
                                        [ 60]  [ 30]  [ 48]  0
                                        [ 40]  [ 20]  [ 32]
                                        [171]  [ 79]  [121]  y
                                        [157]  [ 6f]  [111]  o
                                        [144]  [ 64]  [100]  d
                                        [141]  [ 61]  [ 97]  a
                                        [ 56]  [ 2e]  [ 46]  .
                                        [ 40]  [ 20]  [ 32]
                                        [110]  [ 48]  [ 72]  H
                                        [145]  [ 65]  [101]  e
                                        [154]  [ 6c]  [108]  l
                                        [154]  [ 6c]  [108]  l
                                        [157]  [ 6f]  [111]  o
                                        [ 40]  [ 20]  [ 32]
                                        [157]  [ 6f]  [111]  o
                                        [142]  [ 62]  [ 98]  b
                                        [151]  [ 69]  [105]  i
                                        [ 55]  [ 2d]  [ 45]  -
                                        [167]  [ 77]  [119]  w
                                        [141]  [ 61]  [ 97]  a
                                        [156]  [ 6e]  [110]  n
                                        [ 40]  [ 20]  [ 32]
                                        [133]  [ 5b]  [ 91]  [
                                        [ 61]  [ 31]  [ 49]  1
                                        [ 62]  [ 32]  [ 50]  2
                                        [ 70]  [ 38]  [ 56]  8
                                        [ 56]  [ 2e]  [ 46]  .
                                        [ 61]  [ 31]  [ 49]  1
                                        [ 62]  [ 32]  [ 50]  2
                                        [ 60]  [ 30]  [ 48]  0
                                        [ 56]  [ 2e]  [ 46]  .
                                        [ 65]  [ 35]  [ 53]  5
                                        [ 66]  [ 36]  [ 54]  6
                                        [ 56]  [ 2e]  [ 46]  .
                                        [ 62]  [ 32]  [ 50]  2
                                        [135]  [ 5d]  [ 93]  ]
                                        [ 54]  [ 2c]  [ 44]  ,
                                        [ 40]  [ 20]  [ 32]
                                        [160]  [ 70]  [112]  p
                                        [154]  [ 6c]  [108]  l
                                        [145]  [ 65]  [101]  e
                                        [141]  [ 61]  [ 97]  a
                                        [163]  [ 73]  [115]  s
                                        [145]  [ 65]  [101]  e
                                        [144]  [ 64]  [100]  d
                                        [ 40]  [ 20]  [ 32]
                                        [164]  [ 74]  [116]  t
                                        [157]  [ 6f]  [111]  o
                                        [ 40]  [ 20]  [ 32]
                                        [155]  [ 6d]  [109]  m
                                        [145]  [ 65]  [101]  e
                                        [145]  [ 65]  [101]  e
                                        [164]  [ 74]  [116]  t
                                        [ 40]  [ 20]  [ 32]
                                        [171]  [ 79]  [121]  y
                                        [157]  [ 6f]  [111]  o
                                        [165]  [ 75]  [117]  u
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
[115]  [ 4d]  [ 77]  M
[101]  [ 41]  [ 65]  A
[111]  [ 49]  [ 73]  I
[114]  [ 4c]  [ 76]  L
[ 40]  [ 20]  [ 32]
[106]  [ 46]  [ 70]  F
[162]  [ 72]  [114]  r
[157]  [ 6f]  [111]  o
[155]  [ 6d]  [109]  m
[ 72]  [ 3a]  [ 58]  :
[ 74]  [ 3c]  [ 60]  <
[150]  [ 68]  [104]  h
[145]  [ 65]  [101]  e
[142]  [ 62]  [ 98]  b
[145]  [ 65]  [101]  e
[162]  [ 72]  [114]  r
[154]  [ 6c]  [108]  l
[145]  [ 65]  [101]  e
[151]  [ 69]  [105]  i
[100]  [ 40]  [ 64]  @
[157]  [ 6f]  [111]  o
[142]  [ 62]  [ 98]  b
[151]  [ 69]  [105]  i
[ 55]  [ 2d]  [ 45]  -
[167]  [ 77]  [119]  w
[141]  [ 61]  [ 97]  a
[156]  [ 6e]  [110]  n
[ 76]  [ 3e]  [ 62]  >
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
                                        [ 62]  [ 32]  [ 50]  2
                                        [ 65]  [ 35]  [ 53]  5
                                        [ 60]  [ 30]  [ 48]  0
                                        [ 40]  [ 20]  [ 32]
                                        [ 74]  [ 3c]  [ 60]  <
                                        [150]  [ 68]  [104]  h
                                        [145]  [ 65]  [101]  e
                                        [142]  [ 62]  [ 98]  b
                                        [145]  [ 65]  [101]  e
```

```
                    [162]  [ 72]  [114]  r
                    [154]  [ 6c]  [108]  l
                    [145]  [ 65]  [101]  e
                    [151]  [ 69]  [105]  i
                    [100]  [ 40]  [ 64]  @
                    [157]  [ 6f]  [111]  o
                    [142]  [ 62]  [ 98]  b
                    [151]  [ 69]  [105]  i
                    [ 55]  [ 2d]  [ 45]  -
                    [167]  [ 77]  [119]  w
                    [141]  [ 61]  [ 97]  a
                    [156]  [ 6e]  [110]  n
                    [ 76]  [ 3e]  [ 62]  >
                    [ 56]  [ 2e]  [ 46]  .
                    [ 56]  [ 2e]  [ 46]  .
                    [ 56]  [ 2e]  [ 46]  .
                    [ 40]  [ 20]  [ 32]
                    [123]  [ 53]  [ 83]  S
                    [145]  [ 65]  [101]  e
                    [156]  [ 6e]  [110]  n
                    [144]  [ 64]  [100]  d
                    [145]  [ 65]  [101]  e
                    [162]  [ 72]  [114]  r
                    [ 40]  [ 20]  [ 32]
                    [157]  [ 6f]  [111]  o
                    [153]  [ 6b]  [107]  k
                    [ 15]  [  d]  [ 13]
                    [ 12]  [  a]  [ 10]
[122]  [ 52]  [ 82]  R
[103]  [ 43]  [ 67]  C
[120]  [ 50]  [ 80]  P
[124]  [ 54]  [ 84]  T
[ 40]  [ 20]  [ 32]
[124]  [ 54]  [ 84]  T
[157]  [ 6f]  [111]  o
[ 72]  [ 3a]  [ 58]  :
[ 74]  [ 3c]  [ 60]  <
[162]  [ 72]  [114]  r
[157]  [ 6f]  [111]  o
[157]  [ 6f]  [111]  o
[164]  [ 74]  [116]  t
[100]  [ 40]  [ 64]  @
[171]  [ 79]  [121]  y
[157]  [ 6f]  [111]  o
[144]  [ 64]  [100]  d
[141]  [ 61]  [ 97]  a
[ 76]  [ 3e]  [ 62]  >
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
                    [ 62]  [ 32]  [ 50]  2
                    [ 65]  [ 35]  [ 53]  5
                    [ 60]  [ 30]  [ 48]  0
                    [ 40]  [ 20]  [ 32]
                    [ 74]  [ 3c]  [ 60]  <
```

```
                                   [162]  [ 72]  [114]  r
                                   [157]  [ 6f]  [111]  o
                                   [157]  [ 6f]  [111]  o
                                   [164]  [ 74]  [116]  t
                                   [100]  [ 40]  [ 64]  @
                                   [171]  [ 79]  [121]  y
                                   [157]  [ 6f]  [111]  o
                                   [144]  [ 64]  [100]  d
                                   [141]  [ 61]  [ 97]  a
                                   [ 76]  [ 3e]  [ 62]  >
                                   [ 56]  [ 2e]  [ 46]  .
                                   [ 56]  [ 2e]  [ 46]  .
                                   [ 56]  [ 2e]  [ 46]  .
                                   [ 40]  [ 20]  [ 32]
                                   [122]  [ 52]  [ 82]  R
                                   [145]  [ 65]  [101]  e
                                   [143]  [ 63]  [ 99]  c
                                   [151]  [ 69]  [105]  i
                                   [160]  [ 70]  [112]  p
                                   [151]  [ 69]  [105]  i
                                   [145]  [ 65]  [101]  e
                                   [156]  [ 6e]  [110]  n
                                   [164]  [ 74]  [116]  t
                                   [ 40]  [ 20]  [ 32]
                                   [157]  [ 6f]  [111]  o
                                   [153]  [ 6b]  [107]  k
                                   [ 15]  [  d]  [ 13]
                                   [ 12]  [  a]  [ 10]
```

[deleted text]

```
[104]  [ 44]  [ 68]  D
[101]  [ 41]  [ 65]  A
[124]  [ 54]  [ 84]  T
[101]  [ 41]  [ 65]  A
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
                                   [ 63]  [ 33]  [ 51]  3
                                   [ 65]  [ 35]  [ 53]  5
                                   [ 64]  [ 34]  [ 52]  4
                                   [ 40]  [ 20]  [ 32]
                                   [105]  [ 45]  [ 69]  E
                                   [156]  [ 6e]  [110]  n
                                   [164]  [ 74]  [116]  t
                                   [145]  [ 65]  [101]  e
                                   [162]  [ 72]  [114]  r
                                   [ 40]  [ 20]  [ 32]
                                   [155]  [ 6d]  [109]  m
                                   [141]  [ 61]  [ 97]  a
                                   [151]  [ 69]  [105]  i
                                   [154]  [ 6c]  [108]  l
                                   [ 54]  [ 2c]  [ 44]  ,
                                   [ 40]  [ 20]  [ 32]
                                   [145]  [ 65]  [101]  e
                                   [156]  [ 6e]  [110]  n
                                   [144]  [ 64]  [100]  d
```

```
                                  [ 40]  [ 20]  [ 32]
                                  [167]  [ 77]  [119]  w
                                  [151]  [ 69]  [105]  i
                                  [164]  [ 74]  [116]  t
                                  [150]  [ 68]  [104]  h
                                  [ 40]  [ 20]  [ 32]
                                  [ 42]  [ 22]  [ 34]  "
                                  [ 56]  [ 2e]  [ 46]  .
                                  [ 42]  [ 22]  [ 34]  "
                                  [ 40]  [ 20]  [ 32]
                                  [157]  [ 6f]  [111]  o
                                  [156]  [ 6e]  [110]  n
                                  [ 40]  [ 20]  [ 32]
                                  [141]  [ 61]  [ 97]  a
                                  [ 40]  [ 20]  [ 32]
                                  [154]  [ 6c]  [108]  l
                                  [151]  [ 69]  [105]  i
                                  [156]  [ 6e]  [110]  n
                                  [145]  [ 65]  [101]  e
                                  [ 40]  [ 20]  [ 32]
                                  [142]  [ 62]  [ 98]  b
                                  [171]  [ 79]  [121]  y
                                  [ 40]  [ 20]  [ 32]
                                  [151]  [ 69]  [105]  i
                                  [164]  [ 74]  [116]  t
                                  [163]  [ 73]  [115]  s
                                  [145]  [ 65]  [101]  e
                                  [154]  [ 6c]  [108]  l
                                  [146]  [ 66]  [102]  f
                                  [ 15]  [  d]  [ 13]
                                  [ 12]  [  a]  [ 10]
[122]  [ 52]  [ 82]  R
[145]  [ 65]  [101]  e
[143]  [ 63]  [ 99]  c
[145]  [ 65]  [101]  e
[151]  [ 69]  [105]  i
[166]  [ 76]  [118]  v
[145]  [ 65]  [101]  e
[144]  [ 64]  [100]  d
[ 72]  [ 3a]  [ 58]  :
[ 40]  [ 20]  [ 32]
[142]  [ 62]  [ 98]  b
[171]  [ 79]  [121]  y
[ 40]  [ 20]  [ 32]
[157]  [ 6f]  [111]  o
[142]  [ 62]  [ 98]  b
[151]  [ 69]  [105]  i
[ 55]  [ 2d]  [ 45]  -
[167]  [ 77]  [119]  w
[141]  [ 61]  [ 97]  a
[156]  [ 6e]  [110]  n
```

[deleted text]

```
[110]  [ 48]  [ 72]  H
[145]  [ 65]  [101]  e
```

```
[154]  [ 6c]  [108]  l
[154]  [ 6c]  [108]  l
[157]  [ 6f]  [111]  o
[ 40]  [ 20]  [ 32]
[141]  [ 61]  [ 97]  a
[147]  [ 67]  [103]  g
[141]  [ 61]  [ 97]  a
[151]  [ 69]  [105]  i
[156]  [ 6e]  [110]  n
[ 54]  [ 2c]  [ 44]  ,
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[124]  [ 54]  [ 84]  T
[157]  [ 6f]  [111]  o
[144]  [ 64]  [100]  d
[144]  [ 64]  [100]  d
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[ 56]  [ 2e]  [ 46]  .
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
                                  [ 62]  [ 32]  [ 50]  2
                                  [ 65]  [ 35]  [ 53]  5
                                  [ 60]  [ 30]  [ 48]  0
                                  [ 40]  [ 20]  [ 32]
                                  [120]  [ 50]  [ 80]  P
                                  [101]  [ 41]  [ 65]  A
                                  [101]  [ 41]  [ 65]  A
                                  [ 60]  [ 30]  [ 48]  0
                                  [ 60]  [ 30]  [ 48]  0
                                  [ 64]  [ 34]  [ 52]  4
                                  [ 70]  [ 38]  [ 56]  8
                                  [ 63]  [ 33]  [ 51]  3
                                  [ 40]  [ 20]  [ 32]
                                  [115]  [ 4d]  [ 77]  M
                                  [145]  [ 65]  [101]  e
                                  [163]  [ 73]  [115]  s
                                  [163]  [ 73]  [115]  s
                                  [141]  [ 61]  [ 97]  a
                                  [147]  [ 67]  [103]  g
                                  [145]  [ 65]  [101]  e
                                  [ 40]  [ 20]  [ 32]
                                  [141]  [ 61]  [ 97]  a
                                  [143]  [ 63]  [ 99]  c
                                  [143]  [ 63]  [ 99]  c
                                  [145]  [ 65]  [101]  e
                                  [160]  [ 70]  [112]  p
                                  [164]  [ 74]  [116]  t
                                  [145]  [ 65]  [101]  e
                                  [144]  [ 64]  [100]  d
                                  [ 40]  [ 20]  [ 32]
                                  [146]  [ 66]  [102]  f
```

```
                                                     [157]  [ 6f]  [111]  o
                                                     [162]  [ 72]  [114]  r
                                                     [ 40]  [ 20]  [ 32]
                                                     [144]  [ 64]  [100]  d
                                                     [145]  [ 65]  [101]  e
                                                     [154]  [ 6c]  [108]  l
                                                     [151]  [ 69]  [105]  i
                                                     [166]  [ 76]  [118]  v
                                                     [145]  [ 65]  [101]  e
                                                     [162]  [ 72]  [114]  r
                                                     [171]  [ 79]  [121]  y
                                                     [ 15]  [  d]  [ 13]
                                                     [ 12]  [  a]  [ 10]
[121]  [ 51]  [ 81]  Q
[125]  [ 55]  [ 85]  U
[111]  [ 49]  [ 73]  I
[124]  [ 54]  [ 84]  T
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
                                                     [ 62]  [ 32]  [ 50]  2
                                                     [ 62]  [ 32]  [ 50]  2
                                                     [ 61]  [ 31]  [ 49]  1
                                                     [ 40]  [ 20]  [ 32]
                                                     [171]  [ 79]  [121]  y
                                                     [157]  [ 6f]  [111]  o
                                                     [144]  [ 64]  [100]  d
                                                     [141]  [ 61]  [ 97]  a
                                                     [ 56]  [ 2e]  [ 46]  .
                                                     [ 40]  [ 20]  [ 32]
                                                     [143]  [ 63]  [ 99]  c
                                                     [154]  [ 6c]  [108]  l
                                                     [157]  [ 6f]  [111]  o
                                                     [163]  [ 73]  [115]  s
                                                     [151]  [ 69]  [105]  i
                                                     [156]  [ 6e]  [110]  n
                                                     [147]  [ 67]  [103]  g
                                                     [ 40]  [ 20]  [ 32]
                                                     [143]  [ 63]  [ 99]  c
                                                     [157]  [ 6f]  [111]  o
                                                     [156]  [ 6e]  [110]  n
                                                     [156]  [ 6e]  [110]  n
                                                     [145]  [ 65]  [101]  e
                                                     [143]  [ 63]  [ 99]  c
                                                     [164]  [ 74]  [116]  t
                                                     [151]  [ 69]  [105]  i
                                                     [157]  [ 6f]  [111]  o
                                                     [156]  [ 6e]  [110]  n
                                                     [ 15]  [  d]  [ 13]
                                                     [ 12]  [  a]  [ 10]
```

As described in print_session’s manual page, print_session’s interpretation and presentation of a session’s data is controlled by finite state machines (FSMs). Below, print_session processes the same NMT packet file, but it uses the FSM files dot.FileNL.CR-FSM and the -

out2 directive to generate two compact files representing the data generated by the client and server. Following the print_session command, the contents of the server file and the client file are displayed.

```
yoda% print_session -input 1.session -out2 1.script -client dot.FileNL.CR-FSM -
server dot.FileNL.CR-FSM

yoda% cat 1.script.server
220 yoda. Sendmail SMI-8.6/SMI-SVR4 ready at Sun, 26 Jan 1997 15:32:25 -0800
250 yoda. Hello obi-wan [128.120.56.2], pleased to meet you
250 <heberlei@obi-wan>... Sender ok
250 <root@yoda>... Recipient ok
250 <heberlei@yoda>... Recipient ok
250 <aheberle@yoda>... Recipient ok
354 Enter mail, end with "." on a line by itself
250 PAA00483 Message accepted for delivery
221 yoda. closing connection

yoda% cat 1.script.client
HELO obi-wan
MAIL From:<heberlei@obi-wan>
RCPT To:<root@yoda>
RCPT To:<heberlei@yoda>
RCPT To:<aheberle@yoda>
DATA
Received: by obi-wan (NX5.67e/NX3.0S)
.id AA00234; Sun, 26 Jan 97 14:18:22 -0800
Date: Sun, 26 Jan 97 14:18:22 -0800
From: Louis Todd Heberlein <heberlei@obi-wan>
Message-Id: <9701262218.AA00234@obi-wan>
To: aheberle@yoda, heberlei@yoda, root@yoda
Subject: Multi-recipient

This is an e-mail message with multiple recipients

Hello again,

Todd
.
QUIT
```

This section demonstrates the processing of telnet data. In particular, the tools audit_log(1), log_reader(1), transfer(1), session(1), and print_session(1) are demonstrated. Text entered by the user is bolded. In order to reduce the amount of text presented, some information has been deleted as indicated by the label **[deleted text]**. Comments are presented in greyed text.

We begin by examining the current working directory. The directory contains a number of configuration files, two packet files (exp3.snoop and exp3.tcpdump), and the typescript (the file recording this session).

```
yoda% ls -aC
.                  .TelnetFSM         dot.FileNL.CR-FSM  patterns
..                 .TelnetSubnegFSM   exp3.snoop         typescript
.StreamFSM         dot.FileNL-FSM     exp3.tcpdump
```

audit_log processes the raw packets in the file exp3.snoop, creating several parallel audit files (see the information following the subsequent "ls" command). audit_log can process several packet file formats as well as reading packets directly from the network.

```
yoda% audit_log -tap SnoopTap -i exp3.snoop -pat patterns
setting tap to SnoopTap
link type: 4
SnoopTap::tapRead: couldn’t read header

yoda% ls -lt
total 448
-rwxr-xr-x   1 heberlei staff        124 Jan 27 21:56 Login.970127.21
-rwxr-xr-x   1 heberlei staff        150 Jan 27 21:56 Pattern.970127.21
-rwxr-xr-x   1 heberlei staff          0 Jan 27 21:56 Rlogin.970127.21
-rwxr-xr-x   1 heberlei staff          0 Jan 27 21:56 Rshell.970127.21
-rwxr-xr-x   1 heberlei staff        474 Jan 27 21:56 TCP.970127.21
-rwxr-xr-x   1 heberlei staff         45 Jan 27 21:56 Telnet.19970127.21
-rw-r--r--   1 heberlei staff         43 Jan 27 21:54 typescript
-rw-r--r--   1 heberlei staff         35 Jan 27 20:43 dot.FileNL-FSM
-rw-r--r--   1 heberlei staff         65 Jan 27 20:43 dot.FileNL.CR-FSM
-rw-r--r--   1 heberlei staff         83 Jan 27 20:30 patterns
-rw-r--r--   1 heberlei staff      97366 Jan 26 16:53 exp3.tcpdump
-rw-r--r--   1 root     other     107180 Jan 26 15:20 exp3.snoop
```

log_reader processes the various parallel audit files created by audit_log and presents the information in various levels of detail. Initially, we show the most concise presentation; each session is displayed in a single line. Shown below are three sessions, each to a server residing at port 23–the port usually reserved for telnet.

```
yoda% log_reader -tcp TCP.970127.21
   1    128.120.56.50  -->     128.120.56.2   23
   0    128.120.56.50  -->     128.120.56.1   23
   2    128.120.56.50  -->     128.120.56.3   23
```

We follow with a more detailed presentation of each network session. In this example, TELNET, LOGIN, and STRINGS information is presented for each session; however, the information remains fairly terse.

```
yoda% log_reader -tcp TCP.970127.21 -telnet Telnet.19970127.21 -login
Login.970127.21 -patlog Pattern.970127.21 -pat patterns
   1    128.120.56.50  -->     128.120.56.2   23
        TELNET: terminal: VT220  window:  24 rows X 80 cols
        LOGIN:  root,
        STRINGS: 1 client matches,  5 server matches
   0    128.120.56.50  -->     128.120.56.1   23
        TELNET: terminal: VT220  window:  24 rows X 80 cols
        LOGIN:  spinkb, palmerg, heberlei,
        STRINGS: 2 client matches,  9 server matches
   2    128.120.56.50  -->     128.120.56.3   23
        TELNET: terminal: VT220  window:  24 rows X 80 cols
        LOGIN:  sys, heberlei,
        STRINGS: 3 client matches,  9 server matches
```

Finally, we present the same three sessions with the greatest amount of detail available. Specific information from each parallel audit file is presented full detail. For example, the TELNET audit file shows that for the first session (session ID=1), the terminal type was a VT200 terminal with 24 rows and 80 columns.

```
yoda% log_reader -tcp TCP.970127.21 -telnet Telnet.19970127.21 -login
Login.970127.21 -patlog Pattern.970127.21 -pat patterns -detail
-------------------------------------------------------------------
   1    128.120.56.50  -->     128.120.56.2  23
from: 15:18:39 ( 1/26/1997)  to: 15:19:24 ( 1/26/1997),  45.405028 secs
client flags: SAF     server_flags: SAF
        ---- TELNET ----------------------------------------
        Terminal type: VT220
        Terminal size: 24 rows X 80 columns
        ---- LOGIN -----------------------------------------
        login count: 1,  password count: 0
        Login: root
        ---- STRING MATCHES --------------------------------
        From Client: passwd = 1
        From Server: Login incorrect = 1
        From Server: Last login: = 1
        From Server: Permission denied = 1
        From Server: passwd = 2
-------------------------------------------------------------------
   0    128.120.56.50  -->     128.120.56.1  23
from: 15:17:19 ( 1/26/1997)  to: 15:20:19 ( 1/26/1997),  180.768157 secs
client flags: SAF     server_flags: SAF
        ---- TELNET ----------------------------------------
        Terminal type: VT220
        Terminal size: 24 rows X 80 columns
        ---- LOGIN -----------------------------------------
        login count: 3,  password count: 3
        Login: spinkb
        Login: palmerg
        Login: heberlei
        Password: I don't know
        Password: Star Trek
        Password: Rome.sun
        ---- STRING MATCHES --------------------------------
        From Client: passwd = 1
        From Client: satan = 1
        From Server: Login incorrect = 2
        From Server: Last login: = 1
        From Server: Permission denied = 2
        From Server: daemon: = 1
        From Server: passwd = 1
        From Server: satan = 2
-------------------------------------------------------------------
   2    128.120.56.50  -->     128.120.56.3  23
from: 15:19:33 ( 1/26/1997)  to: 15:20:08 ( 1/26/1997),  34.843961 secs
client flags: SAF     server_flags: SAF
        ---- TELNET ----------------------------------------
        Terminal type: VT220
        Terminal size: 24 rows X 80 columns
        ---- LOGIN ----------------------------------------
        login count: 2,  password count: 1
        Login: sys
        Login: heberlei
        Password: sys
        ---- STRING MATCHES ---------------------------------
        From Client: passwd = 2
        From Client: satan = 1
        From Server: Login incorrect = 1
        From Server: Last login: = 1
        From Server: Permission denied = 1
        From Server: daemon: = 1
        From Server: passwd = 3
        From Server: satan = 2
```

transfer is used to translate packets from one file format to the native Network Monitoring Toolkit (NMT) format. The new file, exp3.nmt, will be used by other NMT tools. In particular, we use the session command to extract the packets from a single network session (the session ID=0) and save them to another NMT packet file, 0.session.

```
yoda% transfer -tap SnoopTap -i exp3.snoop -a exp3.nmt
link type: 4
SnoopTap::tapRead: couldn’t read header
Done set at location 1
1211 packets transferred

yoda% session -tcp TCP.970127.21 -input exp3.nmt -output 0.session -id 0
```

print_session displays a session’s data in several ways. The first use shown below is the default behavior. Two main columns of data are displayed. The first column displays the data transmitted by the client, and the second column displays the data transmitted by the server. As described in print_session’s manual page, each byte of data is displayed in four different formats: octal, hexadecimal, decimal, and the ASCII character code. This default behavior is excellent for understanding application protocols and client/server interactions. As can be seen by this example, sessions presented in this method can be voluminous. In fact, some data has actually been deleted.

0.session is a telnet connection, and there is a substantial amount of tenet negotiations data exchanged between the client and server. It is important to note that this negotiation data is neither sent by nor presented to the user. This is critical when attempting to understand or analyze a particular connection. Furthermore, telnet negotiations can provide additional information which can be useful for identifying the user and/or recreating what the user saw his or her terminal (e.g., the terminal type and window size).

```
yoda% print_session -input 0.session
[377]  [ ff]  [255]
[375]  [ fd]  [253]
[  1]  [  1]  [  1]
[377]  [ ff]  [255]
[375]  [ fd]  [253]
[  3]  [  3]  [  3]
[377]  [ ff]  [255]
[374]  [ fc]  [252]
[ 43]  [ 23]  [ 35]  #
                                                   [377]  [ ff]  [255]
                                                   [375]  [ fd]  [253]
                                                   [ 30]  [ 18]  [ 24]
                                                   [377]  [ ff]  [255]
                                                   [375]  [ fd]  [253]
                                                   [ 37]  [ 1f]  [ 31]
                                                   [377]  [ ff]  [255]
                                                   [375]  [ fd]  [253]
                                                   [ 43]  [ 23]  [ 35]  #
                                                   [377]  [ ff]  [255]
                                                   [375]  [ fd]  [253]
                                                   [ 47]  [ 27]  [ 39]  '
                                                   [377]  [ ff]  [255]
                                                   [375]  [ fd]  [253]
                                                   [ 44]  [ 24]  [ 36]  $
                                                   [377]  [ ff]  [255]
                                                   [373]  [ fb]  [251]
                                                   [  1]  [  1]  [  1]
                                                   [377]  [ ff]  [255]
                                                   [373]  [ fb]  [251]
                                                   [  3]  [  3]  [  3]
                                                   [377]  [ ff]  [255]
                                                   [376]  [ fe]  [254]
                                                   [ 43]  [ 23]  [ 35]  #
[377]  [ ff]  [255]
[373]  [ fb]  [251]
[ 30]  [ 18]  [ 24]
[377]  [ ff]  [255]
[373]  [ fb]  [251]
[ 37]  [ 1f]  [ 31]
[377]  [ ff]  [255]
[372]  [ fa]  [250]
[ 37]  [ 1f]  [ 31]
[  0]  [  0]  [  0]
[120]  [ 50]  [ 80]  P
[  0]  [  0]  [  0]
[ 30]  [ 18]  [ 24]
[377]  [ ff]  [255]
[360]  [ f0]  [240]
[377]  [ ff]  [255]
[374]  [ fc]  [252]
[ 43]  [ 23]  [ 35]  #
[377]  [ ff]  [255]
[374]  [ fc]  [252]
[ 47]  [ 27]  [ 39]  ’
[377]  [ ff]  [255]
[374]  [ fc]  [252]
[ 44]  [ 24]  [ 36]  $
[377]  [ ff]  [255]
[375]  [ fd]  [253]
[  1]  [  1]  [  1]
                                         [377]  [ ff]  [255]
                                         [376]  [ fe]  [254]
                                         [ 47]  [ 27]  [ 39]  ’
                                         [377]  [ ff]  [255]
                                         [376]  [ fe]  [254]
                                         [ 44]  [ 24]  [ 36]  $
                                         [377]  [ ff]  [255]
                                         [372]  [ fa]  [250]
                                         [ 30]  [ 18]  [ 24]
                                         [  1]  [  1]  [  1]
                                         [377]  [ ff]  [255]
                                         [360]  [ f0]  [240]
[377]  [ ff]  [255]
[372]  [ fa]  [250]
[ 30]  [ 18]  [ 24]
[  0]  [  0]  [  0]
[126]  [ 56]  [ 86]  V
[124]  [ 54]  [ 84]  T
[ 62]  [ 32]  [ 50]  2
[ 62]  [ 32]  [ 50]  2
[ 60]  [ 30]  [ 48]  0
[377]  [ ff]  [255]
[360]  [ f0]  [240]
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
                                         [125]  [ 55]  [ 85]  U
                                         [116]  [ 4e]  [ 78]  N
                                         [111]  [ 49]  [ 73]  I
                                         [130]  [ 58]  [ 88]  X
                                         [ 50]  [ 28]  [ 40]  (
                                         [162]  [ 72]  [114]  r
                                         [ 51]  [ 29]  [ 41]  )
                                         [ 40]  [ 20]  [ 32]
                                         [123]  [ 53]  [ 83]  S
                                         [171]  [ 79]  [121]  y
                                         [163]  [ 73]  [115]  s
                                         [164]  [ 74]  [116]  t
```

```
                                        [145]  [ 65]  [101]  e
                                        [155]  [ 6d]  [109]  m
                                        [ 40]  [ 20]  [ 32]
                                        [126]  [ 56]  [ 86]  V
                                        [ 40]  [ 20]  [ 32]
                                        [122]  [ 52]  [ 82]  R
                                        [145]  [ 65]  [101]  e
                                        [154]  [ 6c]  [108]  l
                                        [145]  [ 65]  [101]  e
                                        [141]  [ 61]  [ 97]  a
                                        [163]  [ 73]  [115]  s
                                        [145]  [ 65]  [101]  e
                                        [ 40]  [ 20]  [ 32]
                                        [ 64]  [ 34]  [ 52]  4
                                        [ 56]  [ 2e]  [ 46]  .
                                        [ 60]  [ 30]  [ 48]  0
                                        [ 40]  [ 20]  [ 32]
                                        [ 50]  [ 28]  [ 40]  (
                                        [171]  [ 79]  [121]  y
                                        [157]  [ 6f]  [111]  o
                                        [144]  [ 64]  [100]  d
                                        [141]  [ 61]  [ 97]  a
                                        [ 51]  [ 29]  [ 41]  )
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
                                        [ 15]  [  d]  [ 13]
                                        [  0]  [  0]  [  0]
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
                                        [ 15]  [  d]  [ 13]
                                        [  0]  [  0]  [  0]
                                        [377]  [ ff]  [255]
                                        [375]  [ fd]  [253]
                                        [  1]  [  1]  [  1]
[377]  [ ff]  [255]
[374]  [ fc]  [252]
[  1]  [  1]  [  1]
                                        [377]  [ ff]  [255]
                                        [376]  [ fe]  [254]
                                        [  1]  [  1]  [  1]
                                        [154]  [ 6c]  [108]  l
                                        [157]  [ 6f]  [111]  o
                                        [147]  [ 67]  [103]  g
                                        [151]  [ 69]  [105]  i
                                        [156]  [ 6e]  [110]  n
                                        [ 72]  [ 3a]  [ 58]  :
                                        [ 40]  [ 20]  [ 32]
[163]  [ 73]  [115]  s
                                        [163]  [ 73]  [115]  s
[160]  [ 70]  [112]  p
                                        [160]  [ 70]  [112]  p
[151]  [ 69]  [105]  i
                                        [151]  [ 69]  [105]  i
[156]  [ 6e]  [110]  n
```

```
                                        [156]  [ 6e]  [110]  n
[153]  [ 6b]  [107]  k
                                        [153]  [ 6b]  [107]  k
[142]  [ 62]  [ 98]  b
                                        [142]  [ 62]  [ 98]  b
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
                                        [120]  [ 50]  [ 80]  P
                                        [141]  [ 61]  [ 97]  a
                                        [163]  [ 73]  [115]  s
                                        [163]  [ 73]  [115]  s
                                        [167]  [ 77]  [119]  w
                                        [157]  [ 6f]  [111]  o
                                        [162]  [ 72]  [114]  r
                                        [144]  [ 64]  [100]  d
                                        [ 72]  [ 3a]  [ 58]  :
                                        [ 40]  [ 20]  [ 32]
[111]  [ 49]  [ 73]  I
[ 40]  [ 20]  [ 32]
[144]  [ 64]  [100]  d
[157]  [ 6f]  [111]  o
[156]  [ 6e]  [110]  n
[ 47]  [ 27]  [ 39]  ’
[164]  [ 74]  [116]  t
[ 40]  [ 20]  [ 32]
[153]  [ 6b]  [107]  k
[156]  [ 6e]  [110]  n
[157]  [ 6f]  [111]  o
[167]  [ 77]  [119]  w
[ 12]  [  a]  [ 10]
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
                                        [114]  [ 4c]  [ 76]  L
                                        [157]  [ 6f]  [111]  o
                                        [147]  [ 67]  [103]  g
                                        [151]  [ 69]  [105]  i
                                        [156]  [ 6e]  [110]  n
                                        [ 40]  [ 20]  [ 32]
                                        [151]  [ 69]  [105]  i
                                        [156]  [ 6e]  [110]  n
                                        [143]  [ 63]  [ 99]  c
                                        [157]  [ 6f]  [111]  o
                                        [162]  [ 72]  [114]  r
                                        [162]  [ 72]  [114]  r
                                        [145]  [ 65]  [101]  e
                                        [143]  [ 63]  [ 99]  c
                                        [164]  [ 74]  [116]  t
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
                                        [154]  [ 6c]  [108]  l
                                        [157]  [ 6f]  [111]  o
                                        [147]  [ 67]  [103]  g
                                        [151]  [ 69]  [105]  i
```

```
                                    [156]  [ 6e]  [110]  n
                                    [ 72]  [ 3a]  [ 58]  :
                                    [ 40]  [ 20]  [ 32]
```

[deleted text]

```
[150]  [ 68]  [104]  h
                                    [150]  [ 68]  [104]  h
[145]  [ 65]  [101]  e
                                    [145]  [ 65]  [101]  e
[142]  [ 62]  [ 98]  b
                                    [142]  [ 62]  [ 98]  b
[145]  [ 65]  [101]  e
                                    [145]  [ 65]  [101]  e
[162]  [ 72]  [114]  r
                                    [162]  [ 72]  [114]  r
[154]  [ 6c]  [108]  l
                                    [154]  [ 6c]  [108]  l
[145]  [ 65]  [101]  e
                                    [145]  [ 65]  [101]  e
[151]  [ 69]  [105]  i
                                    [151]  [ 69]  [105]  i
                                    [ 15]  [  d]  [ 13]
                                    [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
                                    [120]  [ 50]  [ 80]  P
                                    [141]  [ 61]  [ 97]  a
                                    [163]  [ 73]  [115]  s
                                    [163]  [ 73]  [115]  s
                                    [167]  [ 77]  [119]  w
                                    [157]  [ 6f]  [111]  o
                                    [162]  [ 72]  [114]  r
                                    [144]  [ 64]  [100]  d
                                    [ 72]  [ 3a]  [ 58]  :
                                    [ 40]  [ 20]  [ 32]
[122]  [ 52]  [ 82]  R
[157]  [ 6f]  [111]  o
[155]  [ 6d]  [109]  m
[145]  [ 65]  [101]  e
[ 56]  [ 2e]  [ 46]  .
[163]  [ 73]  [115]  s
[165]  [ 75]  [117]  u
[156]  [ 6e]  [110]  n
[ 12]  [  a]  [ 10]
                                    [ 15]  [  d]  [ 13]
                                    [ 12]  [  a]  [ 10]
                                    [114]  [ 4c]  [ 76]  L
                                    [141]  [ 61]  [ 97]  a
                                    [163]  [ 73]  [115]  s
                                    [164]  [ 74]  [116]  t
                                    [ 40]  [ 20]  [ 32]
                                    [154]  [ 6c]  [108]  l
                                    [157]  [ 6f]  [111]  o
                                    [147]  [ 67]  [103]  g
                                    [151]  [ 69]  [105]  i
                                    [156]  [ 6e]  [110]  n
```

```
[ 72]  [ 3a]  [ 58]  :
[ 40]  [ 20]  [ 32]   
[127]  [ 57]  [ 87]  W
[145]  [ 65]  [101]  e
[144]  [ 64]  [100]  d
[ 40]  [ 20]  [ 32]   
[112]  [ 4a]  [ 74]  J
[141]  [ 61]  [ 97]  a
[156]  [ 6e]  [110]  n
[ 40]  [ 20]  [ 32]   
[ 62]  [ 32]  [ 50]  2
[ 62]  [ 32]  [ 50]  2
[ 40]  [ 20]  [ 32]   
[ 61]  [ 31]  [ 49]  1
[ 60]  [ 30]  [ 48]  0
[ 72]  [ 3a]  [ 58]  :
[ 64]  [ 34]  [ 52]  4
[ 61]  [ 31]  [ 49]  1
[ 72]  [ 3a]  [ 58]  :
[ 63]  [ 33]  [ 51]  3
[ 64]  [ 34]  [ 52]  4
[ 40]  [ 20]  [ 32]   
[146]  [ 66]  [102]  f
[162]  [ 72]  [114]  r
[157]  [ 6f]  [111]  o
[155]  [ 6d]  [109]  m
[ 40]  [ 20]  [ 32]   
[143]  [ 63]  [ 99]  c
[ 63]  [ 33]  [ 51]  3
[160]  [ 70]  [112]  p
[ 60]  [ 30]  [ 48]  0
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[123]  [ 53]  [ 83]  S
[165]  [ 75]  [117]  u
[156]  [ 6e]  [110]  n
[ 40]  [ 20]  [ 32]   
[115]  [ 4d]  [ 77]  M
[151]  [ 69]  [105]  i
[143]  [ 63]  [ 99]  c
[162]  [ 72]  [114]  r
[157]  [ 6f]  [111]  o
[163]  [ 73]  [115]  s
[171]  [ 79]  [121]  y
[163]  [ 73]  [115]  s
[164]  [ 74]  [116]  t
[145]  [ 65]  [101]  e
[155]  [ 6d]  [109]  m
[163]  [ 73]  [115]  s
[ 40]  [ 20]  [ 32]   
[111]  [ 49]  [ 73]  I
[156]  [ 6e]  [110]  n
[143]  [ 63]  [ 99]  c
[ 56]  [ 2e]  [ 46]  .
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[123]  [ 53]  [ 83]  S
[165]  [ 75]  [117]  u
[156]  [ 6e]  [110]  n
[117]  [ 4f]  [ 79]  O
[123]  [ 53]  [ 83]  S
[ 40]  [ 20]  [ 32]
[ 65]  [ 35]  [ 53]  5
[ 56]  [ 2e]  [ 46]  .
[ 65]  [ 35]  [ 53]  5
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[107]  [ 47]  [ 71]  G
[145]  [ 65]  [101]  e
[156]  [ 6e]  [110]  n
[145]  [ 65]  [101]  e
[162]  [ 72]  [114]  r
[151]  [ 69]  [105]  i
[143]  [ 63]  [ 99]  c
[ 40]  [ 20]  [ 32]
[116]  [ 4e]  [ 78]  N
[157]  [ 6f]  [111]  o
[166]  [ 76]  [118]  v
[145]  [ 65]  [101]  e
[155]  [ 6d]  [109]  m
[142]  [ 62]  [ 98]  b
[145]  [ 65]  [101]  e
[162]  [ 72]  [114]  r
[ 40]  [ 20]  [ 32]
[ 61]  [ 31]  [ 49]  1
[ 71]  [ 39]  [ 57]  9
[ 71]  [ 39]  [ 57]  9
[ 65]  [ 35]  [ 53]  5
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[131]  [ 59]  [ 89]  Y
[157]  [ 6f]  [111]  o
[165]  [ 75]  [117]  u
[ 40]  [ 20]  [ 32]
[150]  [ 68]  [104]  h
[141]  [ 61]  [ 97]  a
[166]  [ 76]  [118]  v
[145]  [ 65]  [101]  e
[ 40]  [ 20]  [ 32]
[155]  [ 6d]  [109]  m
[141]  [ 61]  [ 97]  a
[151]  [ 69]  [105]  i
```

```
                                         [154]  [ 6c]  [108]  l
                                         [ 56]  [ 2e]  [ 46]  .
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
                                         [171]  [ 79]  [121]  y
                                         [157]  [ 6f]  [111]  o
                                         [144]  [ 64]  [100]  d
                                         [141]  [ 61]  [ 97]  a
                                         [ 45]  [ 25]  [ 37]  %
                                         [ 40]  [ 20]  [ 32]
[143]  [ 63]  [ 99]  c
                                         [143]  [ 63]  [ 99]  c
[141]  [ 61]  [ 97]  a
                                         [141]  [ 61]  [ 97]  a
[164]  [ 74]  [116]  t
                                         [164]  [ 74]  [116]  t
[ 40]  [ 20]  [ 32]
                                         [ 40]  [ 20]  [ 32]
[ 57]  [ 2f]  [ 47]  /
                                         [ 57]  [ 2f]  [ 47]  /
[145]  [ 65]  [101]  e
                                         [145]  [ 65]  [101]  e
[164]  [ 74]  [116]  t
                                         [164]  [ 74]  [116]  t
[143]  [ 63]  [ 99]  c
                                         [143]  [ 63]  [ 99]  c
[ 57]  [ 2f]  [ 47]  /
                                         [ 57]  [ 2f]  [ 47]  /
[160]  [ 70]  [112]  p
                                         [160]  [ 70]  [112]  p
[141]  [ 61]  [ 97]  a
                                         [141]  [ 61]  [ 97]  a
[163]  [ 73]  [115]  s
                                         [163]  [ 73]  [115]  s
[163]  [ 73]  [115]  s
                                         [163]  [ 73]  [115]  s
[167]  [ 77]  [119]  w
                                         [167]  [ 77]  [119]  w
[144]  [ 64]  [100]  d
                                         [144]  [ 64]  [100]  d
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
                                         [162]  [ 72]  [114]  r
                                         [157]  [ 6f]  [111]  o
                                         [157]  [ 6f]  [111]  o
                                         [164]  [ 74]  [116]  t
                                         [ 72]  [ 3a]  [ 58]  :
                                         [170]  [ 78]  [120]  x
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 60]  [ 30]  [ 48]  0
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 61]  [ 31]  [ 49]  1
                                         [ 72]  [ 3a]  [ 58]  :
```

```
                                         [123]  [ 53]  [ 83]  S
                                         [165]  [ 75]  [117]  u
                                         [160]  [ 70]  [112]  p
                                         [145]  [ 65]  [101]  e
                                         [162]  [ 72]  [114]  r
                                         [ 55]  [ 2d]  [ 45]  -
                                         [125]  [ 55]  [ 85]  U
                                         [163]  [ 73]  [115]  s
                                         [145]  [ 65]  [101]  e
                                         [162]  [ 72]  [114]  r
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 57]  [ 2f]  [ 47]  /
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 57]  [ 2f]  [ 47]  /
                                         [163]  [ 73]  [115]  s
                                         [142]  [ 62]  [ 98]  b
                                         [151]  [ 69]  [105]  i
                                         [156]  [ 6e]  [110]  n
                                         [ 57]  [ 2f]  [ 47]  /
                                         [163]  [ 73]  [115]  s
                                         [150]  [ 68]  [104]  h
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
```

[deleted text]

```
[143]  [ 63]  [ 99]  c
                                         [143]  [ 63]  [ 99]  c
[154]  [ 6c]  [108]  l
                                         [154]  [ 6c]  [108]  l
[145]  [ 65]  [101]  e
                                         [145]  [ 65]  [101]  e
[141]  [ 61]  [ 97]  a
                                         [141]  [ 61]  [ 97]  a
[162]  [ 72]  [114]  r
                                         [162]  [ 72]  [114]  r
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
                                         [ 33]  [ 1b]  [ 27]
                                         [133]  [ 5b]  [ 91]  [
                                         [110]  [ 48]  [ 72]  H
                                         [ 33]  [ 1b]  [ 27]
                                         [133]  [ 5b]  [ 91]  [
                                         [112]  [ 4a]  [ 74]  J
                                         [171]  [ 79]  [121]  y
                                         [157]  [ 6f]  [111]  o
                                         [144]  [ 64]  [100]  d
                                         [141]  [ 61]  [ 97]  a
                                         [ 45]  [ 25]  [ 37]  %
                                         [ 40]  [ 20]  [ 32]
[154]  [ 6c]  [108]  l
                                         [154]  [ 6c]  [108]  l
[157]  [ 6f]  [111]  o
                                         [157]  [ 6f]  [111]  o
[147]  [ 67]  [103]  g
```

```
                                        [147]  [ 67]  [103]  g
[157]  [ 6f]  [111]  o
                                        [157]  [ 6f]  [111]  o
[165]  [ 75]  [117]  u
                                        [165]  [ 75]  [117]  u
[164]  [ 74]  [116]  t
                                        [164]  [ 74]  [116]  t
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
```

Below we apply print_session to the same data set, 0.session, but we use the -telnet option. This option inserts the TelnetStream object in the protocol stack, and the TelnetStream is responsible to for the analysis and removal of telnet negotiations from the data stream. Thus, the data shown below more closely resembles what the user transmits or sees.

```
yoda% print_session -input 0.session -telnet
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
                                        [125]  [ 55]  [ 85]  U
                                        [116]  [ 4e]  [ 78]  N
                                        [111]  [ 49]  [ 73]  I
                                        [130]  [ 58]  [ 88]  X
                                        [ 50]  [ 28]  [ 40]  (
                                        [162]  [ 72]  [114]  r
                                        [ 51]  [ 29]  [ 41]  )
                                        [ 40]  [ 20]  [ 32]
                                        [123]  [ 53]  [ 83]  S
                                        [171]  [ 79]  [121]  y
                                        [163]  [ 73]  [115]  s
                                        [164]  [ 74]  [116]  t
                                        [145]  [ 65]  [101]  e
                                        [155]  [ 6d]  [109]  m
                                        [ 40]  [ 20]  [ 32]
                                        [126]  [ 56]  [ 86]  V
                                        [ 40]  [ 20]  [ 32]
                                        [122]  [ 52]  [ 82]  R
                                        [145]  [ 65]  [101]  e
                                        [154]  [ 6c]  [108]  l
                                        [145]  [ 65]  [101]  e
                                        [141]  [ 61]  [ 97]  a
                                        [163]  [ 73]  [115]  s
                                        [145]  [ 65]  [101]  e
                                        [ 40]  [ 20]  [ 32]
                                        [ 64]  [ 34]  [ 52]  4
                                        [ 56]  [ 2e]  [ 46]  .
                                        [ 60]  [ 30]  [ 48]  0
                                        [ 40]  [ 20]  [ 32]
                                        [ 50]  [ 28]  [ 40]  (
                                        [171]  [ 79]  [121]  y
                                        [157]  [ 6f]  [111]  o
                                        [144]  [ 64]  [100]  d
```

```
                                         [141]  [ 61]  [ 97]  a
                                         [ 51]  [ 29]  [ 41]  )
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
                                         [ 15]  [  d]  [ 13]
                                         [  0]  [  0]  [  0]
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
                                         [154]  [ 6c]  [108]  l
                                         [157]  [ 6f]  [111]  o
                                         [147]  [ 67]  [103]  g
                                         [151]  [ 69]  [105]  i
                                         [156]  [ 6e]  [110]  n
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 40]  [ 20]  [ 32]
[163]  [ 73]  [115]  s
                                         [163]  [ 73]  [115]  s
[160]  [ 70]  [112]  p
                                         [160]  [ 70]  [112]  p
[151]  [ 69]  [105]  i
                                         [151]  [ 69]  [105]  i
[156]  [ 6e]  [110]  n
                                         [156]  [ 6e]  [110]  n
[153]  [ 6b]  [107]  k
                                         [153]  [ 6b]  [107]  k
[142]  [ 62]  [ 98]  b
                                         [142]  [ 62]  [ 98]  b
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
                                         [120]  [ 50]  [ 80]  P
                                         [141]  [ 61]  [ 97]  a
                                         [163]  [ 73]  [115]  s
                                         [163]  [ 73]  [115]  s
                                         [167]  [ 77]  [119]  w
                                         [157]  [ 6f]  [111]  o
                                         [162]  [ 72]  [114]  r
                                         [144]  [ 64]  [100]  d
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 40]  [ 20]  [ 32]
[111]  [ 49]  [ 73]  I
[ 40]  [ 20]  [ 32]
[144]  [ 64]  [100]  d
[157]  [ 6f]  [111]  o
[156]  [ 6e]  [110]  n
[ 47]  [ 27]  [ 39]  ’
[164]  [ 74]  [116]  t
[ 40]  [ 20]  [ 32]
[153]  [ 6b]  [107]  k
[156]  [ 6e]  [110]  n
[157]  [ 6f]  [111]  o
[167]  [ 77]  [119]  w
[ 12]  [  a]  [ 10]
                                         [ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[114]  [ 4c]  [ 76]  L
[157]  [ 6f]  [111]  o
[147]  [ 67]  [103]  g
[151]  [ 69]  [105]  i
[156]  [ 6e]  [110]  n
[ 40]  [ 20]  [ 32]
[151]  [ 69]  [105]  i
[156]  [ 6e]  [110]  n
[143]  [ 63]  [ 99]  c
[157]  [ 6f]  [111]  o
[162]  [ 72]  [114]  r
[162]  [ 72]  [114]  r
[145]  [ 65]  [101]  e
[143]  [ 63]  [ 99]  c
[164]  [ 74]  [116]  t
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[154]  [ 6c]  [108]  l
[157]  [ 6f]  [111]  o
[147]  [ 67]  [103]  g
[151]  [ 69]  [105]  i
[156]  [ 6e]  [110]  n
[ 72]  [ 3a]  [ 58]  :
[ 40]  [ 20]  [ 32]
```

[deleted text]

```
[150]  [ 68]  [104]  h

[145]  [ 65]  [101]  e

[142]  [ 62]  [ 98]  b

[145]  [ 65]  [101]  e

[162]  [ 72]  [114]  r

[154]  [ 6c]  [108]  l

[145]  [ 65]  [101]  e

[151]  [ 69]  [105]  i



[ 12]  [  a]  [ 10]
```

```
[150]  [ 68]  [104]  h

[145]  [ 65]  [101]  e

[142]  [ 62]  [ 98]  b

[145]  [ 65]  [101]  e

[162]  [ 72]  [114]  r

[154]  [ 6c]  [108]  l

[145]  [ 65]  [101]  e

[151]  [ 69]  [105]  i
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]

[120]  [ 50]  [ 80]  P
[141]  [ 61]  [ 97]  a
[163]  [ 73]  [115]  s
[163]  [ 73]  [115]  s
[167]  [ 77]  [119]  w
[157]  [ 6f]  [111]  o
[162]  [ 72]  [114]  r
[144]  [ 64]  [100]  d
[ 72]  [ 3a]  [ 58]  :
```

```
                                         [ 40]  [ 20]  [ 32]
[122]  [ 52]  [ 82]  R
[157]  [ 6f]  [111]  o
[155]  [ 6d]  [109]  m
[145]  [ 65]  [101]  e
[ 56]  [ 2e]  [ 46]  .
[163]  [ 73]  [115]  s
[165]  [ 75]  [117]  u
[156]  [ 6e]  [110]  n
[ 12]  [  a]  [ 10]
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
                                         [114]  [ 4c]  [ 76]  L
                                         [141]  [ 61]  [ 97]  a
                                         [163]  [ 73]  [115]  s
                                         [164]  [ 74]  [116]  t
                                         [ 40]  [ 20]  [ 32]
                                         [154]  [ 6c]  [108]  l
                                         [157]  [ 6f]  [111]  o
                                         [147]  [ 67]  [103]  g
                                         [151]  [ 69]  [105]  i
                                         [156]  [ 6e]  [110]  n
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 40]  [ 20]  [ 32]
                                         [127]  [ 57]  [ 87]  W
                                         [145]  [ 65]  [101]  e
                                         [144]  [ 64]  [100]  d
                                         [ 40]  [ 20]  [ 32]
                                         [112]  [ 4a]  [ 74]  J
                                         [141]  [ 61]  [ 97]  a
                                         [156]  [ 6e]  [110]  n
                                         [ 40]  [ 20]  [ 32]
                                         [ 62]  [ 32]  [ 50]  2
                                         [ 62]  [ 32]  [ 50]  2
                                         [ 40]  [ 20]  [ 32]
                                         [ 61]  [ 31]  [ 49]  1
                                         [ 60]  [ 30]  [ 48]  0
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 64]  [ 34]  [ 52]  4
                                         [ 61]  [ 31]  [ 49]  1
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 63]  [ 33]  [ 51]  3
                                         [ 64]  [ 34]  [ 52]  4
                                         [ 40]  [ 20]  [ 32]
                                         [146]  [ 66]  [102]  f
                                         [162]  [ 72]  [114]  r
                                         [157]  [ 6f]  [111]  o
                                         [155]  [ 6d]  [109]  m
                                         [ 40]  [ 20]  [ 32]
                                         [143]  [ 63]  [ 99]  c
                                         [ 63]  [ 33]  [ 51]  3
                                         [160]  [ 70]  [112]  p
                                         [ 60]  [ 30]  [ 48]  0
                                         [ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[123]  [ 53]  [ 83]  S
[165]  [ 75]  [117]  u
[156]  [ 6e]  [110]  n
[ 40]  [ 20]  [ 32]
[115]  [ 4d]  [ 77]  M
[151]  [ 69]  [105]  i
[143]  [ 63]  [ 99]  c
[162]  [ 72]  [114]  r
[157]  [ 6f]  [111]  o
[163]  [ 73]  [115]  s
[171]  [ 79]  [121]  y
[163]  [ 73]  [115]  s
[164]  [ 74]  [116]  t
[145]  [ 65]  [101]  e
[155]  [ 6d]  [109]  m
[163]  [ 73]  [115]  s
[ 40]  [ 20]  [ 32]
[111]  [ 49]  [ 73]  I
[156]  [ 6e]  [110]  n
[143]  [ 63]  [ 99]  c
[ 56]  [ 2e]  [ 46]  .
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[123]  [ 53]  [ 83]  S
[165]  [ 75]  [117]  u
[156]  [ 6e]  [110]  n
[117]  [ 4f]  [ 79]  O
[123]  [ 53]  [ 83]  S
[ 40]  [ 20]  [ 32]
[ 65]  [ 35]  [ 53]  5
[ 56]  [ 2e]  [ 46]  .
[ 65]  [ 35]  [ 53]  5
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[ 40]  [ 20]  [ 32]
[107]  [ 47]  [ 71]  G
[145]  [ 65]  [101]  e
[156]  [ 6e]  [110]  n
[145]  [ 65]  [101]  e
[162]  [ 72]  [114]  r
[151]  [ 69]  [105]  i
[143]  [ 63]  [ 99]  c
[ 40]  [ 20]  [ 32]
[116]  [ 4e]  [ 78]  N
[157]  [ 6f]  [111]  o
[166]  [ 76]  [118]  v
[145]  [ 65]  [101]  e
[155]  [ 6d]  [109]  m
```

```
[142]  [ 62]  [ 98]  b
[145]  [ 65]  [101]  e
[162]  [ 72]  [114]  r
[ 40]  [ 20]  [ 32]
[ 61]  [ 31]  [ 49]  1
[ 71]  [ 39]  [ 57]  9
[ 71]  [ 39]  [ 57]  9
[ 65]  [ 35]  [ 53]  5
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[131]  [ 59]  [ 89]  Y
[157]  [ 6f]  [111]  o
[165]  [ 75]  [117]  u
[ 40]  [ 20]  [ 32]
[150]  [ 68]  [104]  h
[141]  [ 61]  [ 97]  a
[166]  [ 76]  [118]  v
[145]  [ 65]  [101]  e
[ 40]  [ 20]  [ 32]
[155]  [ 6d]  [109]  m
[141]  [ 61]  [ 97]  a
[151]  [ 69]  [105]  i
[154]  [ 6c]  [108]  l
[ 56]  [ 2e]  [ 46]  .
[ 15]  [  d]  [ 13]
[ 12]  [  a]  [ 10]
[171]  [ 79]  [121]  y
[157]  [ 6f]  [111]  o
[144]  [ 64]  [100]  d
[141]  [ 61]  [ 97]  a
[ 45]  [ 25]  [ 37]  %
[ 40]  [ 20]  [ 32]

[143]  [ 63]  [ 99]  c

[141]  [ 61]  [ 97]  a

[164]  [ 74]  [116]  t

[ 40]  [ 20]  [ 32]

[ 57]  [ 2f]  [ 47]  /

[145]  [ 65]  [101]  e

[164]  [ 74]  [116]  t

[143]  [ 63]  [ 99]  c

[ 57]  [ 2f]  [ 47]  /

[160]  [ 70]  [112]  p

[141]  [ 61]  [ 97]  a
```

```
[143]  [ 63]  [ 99]  c

[141]  [ 61]  [ 97]  a

[164]  [ 74]  [116]  t

[ 40]  [ 20]  [ 32]

[ 57]  [ 2f]  [ 47]  /

[145]  [ 65]  [101]  e

[164]  [ 74]  [116]  t

[143]  [ 63]  [ 99]  c

[ 57]  [ 2f]  [ 47]  /

[160]  [ 70]  [112]  p

[141]  [ 61]  [ 97]  a
```

```
[163]  [ 73]  [115]  s
                                         [163]  [ 73]  [115]  s
[163]  [ 73]  [115]  s
                                         [163]  [ 73]  [115]  s
[167]  [ 77]  [119]  w
                                         [167]  [ 77]  [119]  w
[144]  [ 64]  [100]  d
                                         [144]  [ 64]  [100]  d
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
                                         [162]  [ 72]  [114]  r
                                         [157]  [ 6f]  [111]  o
                                         [157]  [ 6f]  [111]  o
                                         [164]  [ 74]  [116]  t
                                         [ 72]  [ 3a]  [ 58]  :
                                         [170]  [ 78]  [120]  x
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 60]  [ 30]  [ 48]  0
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 61]  [ 31]  [ 49]  1
                                         [ 72]  [ 3a]  [ 58]  :
                                         [123]  [ 53]  [ 83]  S
                                         [165]  [ 75]  [117]  u
                                         [160]  [ 70]  [112]  p
                                         [145]  [ 65]  [101]  e
                                         [162]  [ 72]  [114]  r
                                         [ 55]  [ 2d]  [ 45]  -
                                         [125]  [ 55]  [ 85]  U
                                         [163]  [ 73]  [115]  s
                                         [145]  [ 65]  [101]  e
                                         [162]  [ 72]  [114]  r
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 57]  [ 2f]  [ 47]  /
                                         [ 72]  [ 3a]  [ 58]  :
                                         [ 57]  [ 2f]  [ 47]  /
                                         [163]  [ 73]  [115]  s
                                         [142]  [ 62]  [ 98]  b
                                         [151]  [ 69]  [105]  i
                                         [156]  [ 6e]  [110]  n
                                         [ 57]  [ 2f]  [ 47]  /
                                         [163]  [ 73]  [115]  s
                                         [150]  [ 68]  [104]  h
                                         [ 15]  [  d]  [ 13]
                                         [ 12]  [  a]  [ 10]
```

[deleted text]

```
[143]  [ 63]  [ 99]  c
                                         [143]  [ 63]  [ 99]  c
[154]  [ 6c]  [108]  l
                                         [154]  [ 6c]  [108]  l
[145]  [ 65]  [101]  e
                                         [145]  [ 65]  [101]  e
[141]  [ 61]  [ 97]  a
                                         [141]  [ 61]  [ 97]  a
```

```
[162]  [ 72]  [114]  r
                                        [162]  [ 72]  [114]  r
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
                                        [ 33]  [ 1b]  [ 27]
                                        [133]  [ 5b]  [ 91]  [
                                        [110]  [ 48]  [ 72]  H
                                        [ 33]  [ 1b]  [ 27]
                                        [133]  [ 5b]  [ 91]  [
                                        [112]  [ 4a]  [ 74]  J
                                        [171]  [ 79]  [121]  y
                                        [157]  [ 6f]  [111]  o
                                        [144]  [ 64]  [100]  d
                                        [141]  [ 61]  [ 97]  a
                                        [ 45]  [ 25]  [ 37]  %
                                        [ 40]  [ 20]  [ 32]
[154]  [ 6c]  [108]  l
                                        [154]  [ 6c]  [108]  l
[157]  [ 6f]  [111]  o
                                        [157]  [ 6f]  [111]  o
[147]  [ 67]  [103]  g
                                        [147]  [ 67]  [103]  g
[157]  [ 6f]  [111]  o
                                        [157]  [ 6f]  [111]  o
[165]  [ 75]  [117]  u
                                        [165]  [ 75]  [117]  u
[164]  [ 74]  [116]  t
                                        [164]  [ 74]  [116]  t
                                        [ 15]  [  d]  [ 13]
                                        [ 12]  [  a]  [ 10]
[ 12]  [  a]  [ 10]
```

We apply print_session two 0.session a third time, but this time we use the -out2, -client, and -server options. -out2 directs print_session to create the file 0.script.client and 0.script.server to which the client and server data will be written respectively. The -client and -server options direct print_session to use the FSM file dot.FileNL.CR-FSM to drive the processing of the data file. This FSM file will create transcript-like output similar to the UNIX script utility or the NSM/ASIM transcript tool. In this particular run, the -telnet option is still used, so telnet negotiations will be removed from the data stream.

```
yoda% print_session -input 0.session -telnet -out2 0.script -client dot.FileNL-FSM
-server dot.FileNL.CR-FSM
```

Below is the contents of the file 0.script.client. This file contains the keystrokes sent by the user. Account names, passwords, and commands are all visible. Unprintable characters (e.g., delete and/or backspace) are displayed as a single period, ’.’.

```
yoda% cat 0.script.client
spinkb
I don’t know
palmerg
Star Trek
heberlei
```

```
Rome.sun
cat /etc/passwd
find satn.an
find . -name sta...satan -print
clear
logout
```

Below is the contents of the file 0.script.server. This file contains the data transmitted by the server to the user's terminal. Generally, this is what the user sees on their screen; however, some data is designed to modify the display and can be difficult to interpret in this format. As an example, see the data following the "clear" command near the end of the file. Unprintable characters are displayed as a single period, '.'.

```
yoda% cat 0.script.server


UNIX(r) System V Release 4.0 (yoda)

login: spinkb
Password:
Login incorrect
login: palmerg
Password:
Login incorrect
login: heberlei
Password:
Last login: Wed Jan 22 10:41:34 from c3p0
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995
You have mail.
yoda% cat /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
smtp:x:0:0:Mail Daemon User:/:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
nobody:x:60001:60001:Nobody:/:
noaccess:x:60002:60002:No Access User:/:
heberlei:x:101:10::/export/home/heberlei:/bin/csh
aheberle:x:100:10:Great Looking:/export/home/aheberle:/bin/csh
www:x:102:10::/export/home/www:/bin/sh
pepper:x:104:10::/export/home/pepper:/bin/csh
yoda% find satn. .an
find: cannot open satan: No such file or directory
yoda% find . -name sta. .. .. .satan -print
find: cannot read dir ./.dt-old/sessions/current.old: Permission denied
find: cannot read dir ./.dtold/sessions/current.old: Permission denied
yoda% clear
.[H.[Jyoda% logout
```

We repeat the process but without the -telnet option; the output is saved to the files 0.script2.client and 0.script.server. Without the TelnetStream object, the telnet negotiations will be written to the two files.

```
yoda% print_session -input 0.session -out2 0.script2 -client dot.FileNL-FSM -server
dot.FileNL.CR-FSM
```

The contents of 0.script2.client are shown below. The information is similar to 0.script.client except the telnet negotiations data can be seen on the first line.

```
yoda% cat 0.script2.client
........#.........P.....#..'..$......VT220.....spinkb
I don't know
palmerg
Star Trek
heberlei
Rome.sun
cat /etc/passwd
find satn.an
find . -name sta...satan -print
clear
logout
```

Finally, the contents of 0.script2.server are shown below. The information is similar to 0.script.server except the telnet negotiations data can be seen on the first line.

```
yoda% cat 0.script2.server
........#..'..$........#..'..$......

UNIX(r) System V Release 4.0 (yoda)

......login: spinkb
Password:
Login incorrect
login: palmerg
Password:
Login incorrect
login: heberlei
Password:
Last login: Wed Jan 22 10:41:34 from c3p0
Sun Microsystems Inc.   SunOS 5.5       Generic November 1995
You have mail.
yoda% cat /etc/passwd
root:x:0:1:Super-User:/:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
smtp:x:0:0:Mail Daemon User:/:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
listen:x:37:4:Network Admin:/usr/net/nls:
```

```
nobody:x:60001:60001:Nobody:/:
noaccess:x:60002:60002:No Access User:/:
heberlei:x:101:10::/export/home/heberlei:/bin/csh
aheberle:x:100:10:Great Looking:/export/home/aheberle:/bin/csh
www:x:102:10::/export/home/www:/bin/sh
pepper:x:104:10::/export/home/pepper:/bin/csh
yoda% find satn. .an
find: cannot open satan: No such file or directory
yoda% find . -name sta. .. .. .satan -print
find: cannot read dir ./.dt-old/sessions/current.old: Permission denied
find: cannot read dir ./.dtold/sessions/current.old: Permission denied
yoda% clear
.[H.[Jyoda% logout
```

This section demonstrates how the summary tool can be used to examine large scale behaviors within a network. Text entered by the user is bolded. In order to reduce the amount of text presented, some information has been deleted as indicated by the label **[deleted text]**. Comments are presented in greyed text.

We begin by examining the contents of the current working directory. The directory contains only two files: TCP.audit_logs and typescript. TCP.audit_logs is an audit file previously created by audit_log(1). typescript is the file recording this session.

```
yoda% ls
TCP.audit_logs  typescript
```

summary(1) summarizes the connections in a TCP audit log, and it is particularly useful when looking for large scale behavior and trends within network audit trails. In this first example, the connections are sorted by server network. Each Internet address can be divided into two parts: a network part and a local part. For example, the host 128.120.56.1 belongs to the class B network 128.120, and it has a local part 56.1. (Typically, a class B network such as 128.120 uses subnet masks to further divide their local network.) Thus, for each connection, the network portion of the server is identified, and number of connections to servers in that network are counted.

In this example, 38,728 of the 13,0574 connections, roughly 30%, of connections are destined to the class B network 128.120.0.0. This class B network belongs to UC Davis, and it was from a UC Davis host that these statistics were gathered. The second most popular network is the class C network 205.218.156.0 with 2.3% of the connections.

```
yoda% summary -tcp TCP.audit_logs -sort server-net

130574 streams read in
SERVER NETWORK:     128.120.0.0: 38728 connections
SERVER NETWORK:   205.218.156.0: 2995 connections
SERVER NETWORK:    204.71.177.0: 2610 connections
SERVER NETWORK:    207.12.240.0: 1696 connections
SERVER NETWORK:    198.95.251.0: 1440 connections
SERVER NETWORK:     207.77.90.0: 1239 connections
SERVER NETWORK:     152.163.0.0: 1126 connections
SERVER NETWORK:      198.3.98.0: 1075 connections
SERVER NETWORK:     198.81.11.0: 1044 connections
SERVER NETWORK:     204.123.2.0: 1003 connections
SERVER NETWORK:    204.201.32.0: 982 connections
SERVER NETWORK:   192.216.191.0: 899 connections
SERVER NETWORK:   205.228.184.0: 758 connections
SERVER NETWORK:    204.162.96.0: 749 connections
SERVER NETWORK:    207.68.156.0: 727 connections
SERVER NETWORK:      152.79.0.0: 700 connections
SERVER NETWORK:      208.1.57.0: 691 connections
SERVER NETWORK:      152.52.0.0: 657 connections
SERVER NETWORK:   208.134.241.0: 648 connections
SERVER NETWORK:    204.146.46.0: 628 connections
SERVER NETWORK:    199.95.208.0: 586 connections
SERVER NETWORK:    204.71.242.0: 579 connections
SERVER NETWORK:   205.217.106.0: 555 connections
SERVER NETWORK:      130.91.0.0: 553 connections
SERVER NETWORK:    204.212.36.0: 552 connections
SERVER NETWORK:   205.216.163.0: 552 connections
```
[deleted text]
```
SERVER NETWORK:    208.202.14.0: 1 connections
SERVER NETWORK:   208.205.142.0: 1 connections
SERVER NETWORK:   208.206.222.0: 1 connections
SERVER NETWORK:   208.206.180.0: 1 connections
SERVER NETWORK:      209.24.8.0: 1 connections
SERVER NETWORK:      209.12.7.0: 1 connections
```

Below, we repeat the sorting on the server's network address, but we also sort on the server's full address as well. Once again, we see the most popular network destination is the class B network 128.120.0.0. However, we also see that within 128.120.0.0, the most popular server host is 128.120.8.183. 128.120.8.183 receives 2282 connections, or 5.9% of the connections into the network 128.120.0.0. We also see that in the second most popular network, 205.218.156.0, the most popular host is 205.218.156.52 with 32% of connections to that network.

```
yoda% summary -tcp TCP.audit_logs -sort server-net -sort server-addr

130574 streams read in
-----------------------------------------------
SERVER NETWORK:     128.120.0.0: 38728 connections
    SERVER ADDRESS   128.120.8.183: 2282 connections
    SERVER ADDRESS   128.120.55.68: 2211 connections
    SERVER ADDRESS   128.120.8.181: 1807 connections
    SERVER ADDRESS 128.120.100.181: 1788 connections
    SERVER ADDRESS 128.120.226.141: 1761 connections
    SERVER ADDRESS   128.120.65.22: 1525 connections
    SERVER ADDRESS 128.120.101.102: 1380 connections
    SERVER ADDRESS 128.120.101.103: 1296 connections
    SERVER ADDRESS  128.120.54.142: 1287 connections
```
[deleted text]
```
    SERVER ADDRESS  128.120.254.86: 1 connections
    SERVER ADDRESS  128.120.254.95: 1 connections
    SERVER ADDRESS  128.120.254.96: 1 connections
-----------------------------------------------
SERVER NETWORK:   205.218.156.0: 2995 connections
    SERVER ADDRESS  205.218.156.52: 971 connections
    SERVER ADDRESS  205.218.156.47: 963 connections
    SERVER ADDRESS  205.218.156.48: 457 connections
    SERVER ADDRESS  205.218.156.56: 372 connections
    SERVER ADDRESS  205.218.156.41: 195 connections
    SERVER ADDRESS  205.218.156.46: 25 connections
    SERVER ADDRESS  205.218.156.39: 10 connections
    SERVER ADDRESS  205.218.156.34: 1 connections
    SERVER ADDRESS  205.218.156.54: 1 connections
-----------------------------------------------
SERVER NETWORK:    204.71.177.0: 2610 connections
    SERVER ADDRESS   204.71.177.71: 458 connections
    SERVER ADDRESS   204.71.177.73: 407 connections
    SERVER ADDRESS   204.71.177.66: 300 connections
    SERVER ADDRESS   204.71.177.74: 224 connections
```

[deleted text]

We sort again on the server network and server address, but now we also sort by server port. The server port often indicates the type of service being accessed. Below, the network address 128.120.0.0 has 38,728 connections; 2282 of these connections are destined to the address 128.120.8.183; and 2273 connections of these connections are to port 113, the authentication service.

The second most popular address in the network 128.120.0.0 is 128.120.56.68, and the most popular server port on this host is 80, the WWW server port. For the third most popular host, 128.120.8.181, the most popular server port is 25, the Sendmail port.

From this quick view, we see that for some reason, many hosts are requesting authentication services from host 128.120.8.183, and hosts 128.120.55.68 and 128.120.8.181 host popular web and Sendmail servers for the network.

```
yoda% summary -tcp TCP.audit_logs -sort server-net -sort server-addr -sort server-
port

130574 streams read in
-----------------------------------------------
SERVER NETWORK:     128.120.0.0: 38728 connections
    -----------------------------------------------
    SERVER ADDRESS   128.120.8.183: 2282 connections
        SERVER PORT   113: 2273 connections
        SERVER PORT 42776: 1 connections
        SERVER PORT   540: 1 connections
        SERVER PORT 42881: 1 connections
        SERVER PORT 45447: 1 connections
        SERVER PORT 43138: 1 connections
        SERVER PORT 46033: 1 connections
        SERVER PORT 47081: 1 connections
        SERVER PORT 46710: 1 connections
        SERVER PORT 46411: 1 connections
    -----------------------------------------------
    SERVER ADDRESS   128.120.55.68: 2211 connections
        SERVER PORT    80: 2209 connections
        SERVER PORT  8080: 1 connections
        SERVER PORT  8000: 1 connections
    -----------------------------------------------
    SERVER ADDRESS   128.120.8.181: 1807 connections
        SERVER PORT    25: 1787 connections
        SERVER PORT   113: 16 connections
        SERVER PORT 50749: 1 connections
        SERVER PORT 36512: 1 connections
        SERVER PORT 50870: 1 connections
        SERVER PORT 52617: 1 connections
    -----------------------------------------------
    SERVER ADDRESS 128.120.100.181: 1788 connections
        SERVER PORT    25: 1786 connections
        SERVER PORT 34440: 1 connections
        SERVER PORT 54848: 1 connections
    ----------------------------------------------
    SERVER ADDRESS 128.120.226.141: 1761 connections
        SERVER PORT    80: 1734 connections
        SERVER PORT    70: 25 connections
        SERVER PORT    13: 2 connections
    ----------------------------------------------
    SERVER ADDRESS   128.120.65.22: 1525 connections
        SERVER PORT    80: 1525 connections
    ----------------------------------------------
    SERVER ADDRESS 128.120.101.102: 1380 connections
        SERVER PORT  6667: 1277 connections
        SERVER PORT  6669: 50 connections
        SERVER PORT  6668: 25 connections
        SERVER PORT  6666: 23 connections
        SERVER PORT    80: 3 connections
        SERVER PORT  6665: 2 connections
    ----------------------------------------------
```

[deleted text]

```
----------------------------------------------
SERVER NETWORK:   205.218.156.0: 2995 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.52: 971 connections
        SERVER PORT    80: 971 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.47: 963 connections
        SERVER PORT    80: 963 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.48: 457 connections
        SERVER PORT    80: 457 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.56: 372 connections
        SERVER PORT    80: 372 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.41: 195 connections
        SERVER PORT  6499: 195 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.46: 25 connections
        SERVER PORT    80: 25 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.39: 10 connections
        SERVER PORT    80: 10 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.34: 1 connections
        SERVER PORT    80: 1 connections
    ----------------------------------------------
    SERVER ADDRESS  205.218.156.54: 1 connections
        SERVER PORT    25: 1 connections
----------------------------------------------
SERVER NETWORK:    204.71.177.0: 2610 connections
    ----------------------------------------------
    SERVER ADDRESS   204.71.177.71: 458 connections
        SERVER PORT    80: 458 connections
    ----------------------------------------------
    SERVER ADDRESS   204.71.177.73: 407 connections
        SERVER PORT    80: 407 connections
    ----------------------------------------------
```

[deleted text]

In the next series of examples, we begin sorting based on the server port. As can be seen below, the most popular server port is 80, the WWW server port, with 71% of the connections. The next three most popular services are Sendmail, authentication, POP.

We also notice numerous connections associated with unknown server ports, including ports 6667, 7000, 8000, 8001, and 6499. In general, any user can establish a network server at ports above 1023, so detecting these servers and identifying their types is extremely important and may be impossible to do by looking at host configuration files (since normal users may be starting these servers without using a host's typical configuration files). This is role of Network Radar's Non-cooperative Service Recognition (NCSR) technology.

Finally, we see that port 443 has 354 connections (0.3% of the total number of connections). 443 is the port used for encrypted WWW traffic, and while 0.3% of the connections may seem very small, it is still larger than the number of FTP connections (335 at port 21), and gopher connections (182 at port 70). Since network monitors are limited in their ability to analyze encrypted sessions, the DOD should pay attention to this trend as they develop their future security architectures.

```
yoda% summary -tcp TCP.audit_logs -sort server-port

130574 streams read in
SERVER PORT    80: 92605 connections
SERVER PORT    25: 15142 connections
SERVER PORT   113: 10739 connections
SERVER PORT   110: 1913 connections
SERVER PORT  6667: 1774 connections
SERVER PORT    23: 1312 connections
SERVER PORT    79: 601 connections
SERVER PORT  7000: 480 connections
SERVER PORT   119: 456 connections
SERVER PORT   443: 354 connections
SERVER PORT    21: 335 connections
SERVER PORT  8000: 208 connections
SERVER PORT  8001: 198 connections
SERVER PORT  6499: 195 connections
SERVER PORT    70: 182 connections
SERVER PORT  1521: 178 connections
SERVER PORT  6669: 141 connections
SERVER PORT  8080: 140 connections
SERVER PORT  2443: 126 connections
SERVER PORT  6668: 122 connections
SERVER PORT  1024: 111 connections
SERVER PORT  4380: 104 connections
SERVER PORT   666: 69 connections
```

[deleted text]

```
SERVER PORT 63060: 1 connections
SERVER PORT 61900: 1 connections
SERVER PORT 62117: 1 connections
```

We repeat the sort on the server port again, but we also request the most popular addresses for each server port. For example, server port 80 (WWW) is the most popular server, and 128.120.55.68 is the most popular WWW server. The next most popular WWW hosts are 128.120.226.141, 128.120.65.22, and 128.120.54.142. The most popular WWW server outside UC Davis's network is 198.95.251.63 (Netscape).

Following WWW servers, we see the most popular services are sendmail and authentication.

```
yoda% summary -tcp TCP.audit_logs -sort server-port -sort server-addr

130574 streams read in
----------------------------------------------
SERVER PORT    80: 92605 connections
    SERVER ADDRESS   128.120.55.68: 2209 connections
    SERVER ADDRESS 128.120.226.141: 1734 connections
    SERVER ADDRESS   128.120.65.22: 1525 connections
    SERVER ADDRESS  128.120.54.142: 1286 connections
    SERVER ADDRESS   198.95.251.63: 1278 connections
    SERVER ADDRESS   207.12.240.66: 1136 connections
    SERVER ADDRESS  205.218.156.52: 971 connections
```

[deleted text]

```
----------------------------------------------
SERVER PORT    25: 15142 connections
    SERVER ADDRESS   128.120.8.181: 1787 connections
    SERVER ADDRESS 128.120.100.181: 1786 connections
    SERVER ADDRESS   128.120.2.152: 360 connections
    SERVER ADDRESS 128.120.101.103: 123 connections
    SERVER ADDRESS 128.120.101.184: 117 connections
    SERVER ADDRESS   128.120.8.184: 116 connections
    SERVER ADDRESS   128.120.2.184: 112 connections
    SERVER ADDRESS  131.229.64.156: 109 connections
    SERVER ADDRESS  128.120.56.188: 108 connections
    SERVER ADDRESS    128.120.57.1: 107 connections
    SERVER ADDRESS 128.120.100.184: 101 connections
    SERVER ADDRESS   129.219.42.13: 95 connections
```

[deleted text]

```
----------------------------------------------
SERVER PORT   113: 10739 connections
    SERVER ADDRESS   128.120.8.183: 2273 connections
    SERVER ADDRESS   204.212.36.59: 551 connections
    SERVER ADDRESS   128.120.54.57: 474 connections
    SERVER ADDRESS 128.120.253.120: 460 connections
    SERVER ADDRESS    198.95.254.5: 288 connections
    SERVER ADDRESS   128.120.2.152: 225 connections
    SERVER ADDRESS   128.120.8.150: 98 connections
```

[deleted text]

In the next example, sessions are sorted first on the client address followed by the server port. The results will be the number of connections started by a client as well as the type of service requested. As shown, the host 128.120.8.183 generated the most connections at 6027. It should be noted that this audit set represents only a single hour of traffic, so the host was generating 100 connections every minute. However, we also see that virtually all of these connections were to sendmail servers (port 25). This may indicate that this host is probing sendmail servers on thousands of machines, or the host may be a very busy mail hub. The answer to this will become a little clearer in the next example.

The next most popular host is 128.120.8.181, with authentication requests accounting for 1770 of its 1847 connections. Previously, when we sorted by server port and server address, this host was also the most popular sendmail server with 1787 mail connections, nearly the exact same number of authentication requests. It appears that 128.120.8.181 is generating authentication requests for each incoming sendmail connection. We also see the same behavior with the third most popular client, 128.120.100.181.

A little further down we see the host 128.120.220.22 generating 1347 connections to WWW servers. Once again, we might interpret this as an attempt to probe for holes in WWW servers. It might also be a web crawler indexing pages or a very prolific user accessing numerous pages. Further analysis is needed to distinguish between these possibilities.

```
yoda% summary -tcp TCP.audit_logs -sort client-addr -sort server-port

130574 streams read in
-----------------------------------------------
CLIENT ADDRESS   128.120.8.183: 6027 connections
    SERVER PORT    25: 6022 connections
    SERVER PORT 44168: 5 connections
-----------------------------------------------
CLIENT ADDRESS   128.120.8.181: 1847 connections
    SERVER PORT   113: 1770 connections
    SERVER PORT    25: 69 connections
    SERVER PORT 61723: 4 connections
    SERVER PORT 53636: 4 connections
-----------------------------------------------
CLIENT ADDRESS 128.120.100.181: 1775 connections
    SERVER PORT   113: 1769 connections
    SERVER PORT 48963: 4 connections
    SERVER PORT  1044: 1 connections
    SERVER PORT 46729: 1 connections
-----------------------------------------------
CLIENT ADDRESS 128.120.101.102: 1373 connections
    SERVER PORT   113: 1365 connections
    SERVER PORT  1068: 8 connections
-----------------------------------------------
CLIENT ADDRESS  128.120.220.22: 1347 connections
    SERVER PORT    80: 1347 connections
-----------------------------------------------
CLIENT ADDRESS   128.120.2.152: 1289 connections
    SERVER PORT    25: 920 connections
    SERVER PORT   113: 362 connections
    SERVER PORT    53: 7 connections
----------------------------------------------
CLIENT ADDRESS 128.120.253.120: 854 connections
    SERVER PORT    23: 414 connections
    SERVER PORT   110: 334 connections
    SERVER PORT    25: 106 connections
----------------------------------------------
```
[deleted text]

Finally, we repeat the client address and server port sort, but we add a third level, sorting on the server address. For example, the client 128.120.8.183 generated the most connections, 6027, with 6022 of those to sendmail server ports. Of those 6022 sendmail connections, 109 went to the server 131.229.64.156, 66 went to 137.99.202.4, etc. If the client 128.120.8.183 was scanning a site, we would expect it to initiate only a single connection to each server, perhaps try known vulnerabilities, and then move to the next host. However, since the client is initiating many connections to the same server, it appears more likely that this is simply a case of a very busy mail server.

What appears more interesting is the fact that host 128.120.253.120 initiated 414 telnet connections, and virtually all the connections are destined to just seven different servers. One possible reason for this behavior is that 128.120.252.120 is a terminal/modem server which uses a load balancing techniques such as round-robin for requests to a single server name. Familiarity with the actual site's operation or additional NMT tools could be used to get a clearer picture for the reasons behind the observed behavior.

```
yoda% summary -tcp TCP.audit_logs -sort client-addr -sort server-port -sort server-
addr

130574 streams read in
----------------------------------------------
CLIENT ADDRESS   128.120.8.183: 6027 connections
    ----------------------------------------------
    SERVER PORT    25: 6022 connections
        SERVER ADDRESS  131.229.64.156: 109 connections
        SERVER ADDRESS    137.99.202.4: 66 connections
        SERVER ADDRESS   141.166.92.70: 66 connections
        SERVER ADDRESS   128.192.90.81: 65 connections
        SERVER ADDRESS    198.81.11.79: 23 connections
        SERVER ADDRESS    198.81.11.53: 22 connections
        SERVER ADDRESS   204.127.131.3: 21 connections
        SERVER ADDRESS   204.95.110.80: 20 connections
```
[deleted text]
```
    ----------------------------------------------
    SERVER PORT 44168: 5 connections
        SERVER ADDRESS    128.8.10.200: 5 connections
----------------------------------------------
CLIENT ADDRESS   128.120.8.181: 1847 connections
    ----------------------------------------------
    SERVER PORT   113: 1770 connections
        SERVER ADDRESS   192.102.249.3: 26 connections
        SERVER ADDRESS   199.172.62.20: 26 connections
        SERVER ADDRESS     152.52.2.29: 25 connections
        SERVER ADDRESS     208.6.87.15: 20 connections
```
[deleted text]
```
    -----------------------------------------------
    SERVER PORT    25: 69 connections
        SERVER ADDRESS   168.150.253.1: 3 connections
        SERVER ADDRESS  138.26.154.170: 2 connections
        SERVER ADDRESS   129.214.1.100: 2 connections
        SERVER ADDRESS   128.32.224.55: 2 connections
        SERVER ADDRESS 129.214.165.110: 2 connections
        SERVER ADDRESS   134.114.96.14: 2 connections
```
[deleted text]
```
    -----------------------------------------------
    SERVER PORT 61723: 4 connections
        SERVER ADDRESS  130.207.166.56: 4 connections
    -----------------------------------------------
    SERVER PORT 53636: 4 connections
        SERVER ADDRESS     134.192.1.5: 4 connections
-----------------------------------------------
CLIENT ADDRESS 128.120.100.181: 1775 connections
    -----------------------------------------------
    SERVER PORT   113: 1769 connections
        SERVER ADDRESS    206.241.12.2: 50 connections
        SERVER ADDRESS   199.172.62.20: 27 connections
        SERVER ADDRESS   192.102.249.3: 25 connections
        SERVER ADDRESS     208.6.87.15: 21 connections
        SERVER ADDRESS   128.32.155.12: 19 connections
```
[deleted text]
```
-----------------------------------------------
CLIENT ADDRESS 128.120.253.120: 854 connections
    -----------------------------------------------
    SERVER PORT    23: 414 connections
        SERVER ADDRESS   128.120.2.150: 67 connections
        SERVER ADDRESS   128.120.2.149: 58 connections
        SERVER ADDRESS   128.120.8.151: 55 connections
        SERVER ADDRESS   128.120.2.148: 52 connections
        SERVER ADDRESS   128.120.2.167: 51 connections
        SERVER ADDRESS   128.120.8.150: 25 connections
        SERVER ADDRESS   128.120.8.167: 20 connections
        SERVER ADDRESS    128.120.2.11: 6 connections
```
[deleted text]

This section demonstrates the tools used to prototype Network Radar's Non-cooperative Service Recognition (NCSR) technology. In particular, the tools class_extractor(1), max_extractor(1), class_normalizer(1), train_test(1), net4(1), and net(5). Text entered by the user is bolded. In order to reduce the amount of text presented, some information has been deleted as indicated by the label **[deleted text]**. Comments are presented in greyed text.

class_extractor searches a TCP audit trail, identifying records matching the requested service (indicated by the -server option), and printing statistics about each session. The statistics for sessions, each displayed on a single line, consists of

- number of packets sent by the client.
- number of packets sent by the server
- number of bytes sent by the client
- number of bytes sent by the server
- average number of bytes per packet sent by client.
- average number of bytes per packet sent by server
- duration of session (in seconds and microseconds)

In the example below, the first twenty sessions to server port 80 (traditionally the WWW server port) are printed.

```
yoda% class_extractor -tcp TCP.audit_logs -server 80 -c 20
      3        1      309         0    103.0000      0.0000       0.409920
      3        2      362         0    120.6667      0.0000       0.233264
     10       11      298     11303     29.8000   1027.5454       2.334368
      4        3      237       726     59.2500    242.0000       1.738432
      3        1      310         0    103.3333      0.0000       0.258240
      4        3      229       727     57.2500    242.3333       1.728672
      9        6      394      4096     43.7778    682.6667       6.443605
      3        1      171         0     57.0000      0.0000       0.092720
      3        1       34         0     11.3333      0.0000       0.154208
      3        2      231       189     77.0000     94.5000       1.643760
      3        2      220       177     73.3333     88.5000       0.183488
      7       14      309      5656     44.1429    404.0000       1.959984
      8        6      215      1224     26.8750    204.0000      89.033498
      5        4      230       375     46.0000     93.7500       1.145424
      6        5       62       512     10.3333    102.4000       3.805668
     10        6      374      5373     37.4000    895.5000     350.118482
      5        4      229       400     45.8000    100.0000       1.256688
      7        6      229      4121     32.7143    686.8333       4.121881
      6        2      370       798     61.6667    399.0000     349.954514
     11        6      735      5442     66.8182    907.0000     353.286235
```

class_extractor is used again to extract up to 5000 sessions associated with WWW, sendmail, authentication, usenet, finger, and telnet. The output for each set is redirected to a file (e.g., 80.sessions for WWW sessions). Note, not all services had 5000 sessions in the TCP audit files.

```
yoda% class_extractor -tcp TCP.audit_logs -server 80 -c 5000 > 80.sessions
yoda% class_extractor -tcp TCP.audit_logs -server 25 -c 5000 > 25.sessions
yoda% class_extractor -tcp TCP.audit_logs -server 113 -c 5000 > 113.sessions
yoda% class_extractor -tcp TCP.audit_logs -server 110 -c 5000 > 110.sessions
```

```
yoda% class_extractor -tcp TCP.audit_logs -server 79 -c 5000 > 79.sessions
yoda% class_extractor -tcp TCP.audit_logs -server 23 -c 5000 > 23.sessions
```

max_extractor reads in several session files (created by class_extractor), and prints summary information about the session statistics generated by class_extractor. For each statistic (e.g., number of packets sent by the client and server), the following information is calculated and displayed:

- maximum value
- average value
- standard deviation
- average plus three standard deviations

According to Chebyshev's rule, at least 8/9 of all measurements should fall within three standard deviations from the mean (average). We use the rule since we cannot assume a normal (gaussian) distribution of the statistics. This last value will be used as a normalizing vector in the next step.

As shown below, the maximum number of packets sent by the client is 124,386, and the maximum number of packets sent by the server is 99,788. The average number of packets sent by the client is 92, and the average plus three standard deviations for the number of packets sent by the client is 1401.5673.

```
yoda% max_extractor -f 80.sessions -f 25.sessions -f 113.sessions -f 110.sessions -
f 79.sessions -f 23.sessions

    124386      99788     760731    6125987  1293.0609  1400.8484 104781.5703

        92         78       1102       7296    51.3217   130.3511   106.9943

  436.5224   444.5388   269.0869   289.6433    81.6172   235.3299  1056.2272

 1401.5673  1411.6164  1909.2608  8164.9300   296.1732   836.3406  3275.6760
```

The last line printed by max_extractor is saved to the file norm_vec to be used as a normalizing vector. Next, we use class_normalizer to apply the normalizing vector against each session in a file. The results are printed to the screen, but in the examples below, we have directed the output to a file. The command is repeated for each session type.

We "normalize" each vector so we can treat each class of statistics similarly. Otherwise, we might run into the case where some value may range from zero to ten, while another would range from zero to one million. Large differences like this may have negative effects on the various classification algorithms.

```
yoda% class_normalizer -f 80.sessions -n norm_vec > 80.norm
yoda% class_normalizer -f 25.sessions -n norm_vec > 25.norm
yoda% class_normalizer -f 113.sessions -n norm_vec > 113.norm
yoda% class_normalizer -f 110.sessions -n norm_vec > 110.norm
yoda% class_normalizer -f 79.sessions -n norm_vec > 79.norm
yoda% class_normalizer -f 23.sessions -n norm_vec > 23.norm
```

train_test is used to split a session file into two disjoint files. One file will be used to test the classifier, and the other file will be used train the classifier. A random number generator is used to determine whether a session will go into the training file or the testing file. With probability 0.66 the session will be placed in the training file; in other words, roughly 66% of the sessions will go into the training file.

In the example below, train_test is run on the file 80.norm (the normalized WWW sessions). train_test creates the files 80.norm.train and 80.norm.test, the training and testing files respectively. Following the train_test command, we use the UNIX utility wc(1) to examine the number of lines (the first value printed) for the various WWW session files we have created. As can be seen, the original session file and normalized session files each have 5000 lines in them (each line represents a single session), and the training and testing files have 3297 and 1703 lines respectively (roughly 65.9% and 34.0%).

```
yoda% train_test -f 80.norm
yoda% wc 80.sessions 80.norm 80.norm.train 80.norm.test
    5000   35000  375000 80.sessions
    5000   35000  385000 80.norm
    3297   23079  253869 80.norm.train
    1703   11921  131131 80.norm.test
   15000  105000 1145000 total
```

We repeat the process on each of the normalized session files.

```
yoda% train_test -f 25.norm
yoda% train_test -f 113.norm
yoda% train_test -f 110.norm
yoda% train_test -f 79.norm
yoda% train_test -f 23.norm
```

`net4 is a classifier program based on a simple backpropagation neural network.` The classifier takes a series of training and testing file sets for each service type and repeatedly trains and tests the neural network. In the example below, three services are presented to the network: telnet (port 23), WWW (port 80), and sendmail (port 25).

The output of the first line (without any training) shows that 0.0% of telnet were misclassified, while 100.0% of the WWW and sendmail were misclassified. The misclassification rate for all services is 88.3% Following 99 rounds of training, the misclassification rates for telnet, WWW, and sendmail are 1.3%, 4.0%, and 1.7% respectively. The aggregate error rate is 2.7%

```
yoda% net4 -c 23.norm.train 23.norm.test -c 80.norm.train 80.norm.test -c
25.norm.train 25.norm.test
Finished reading in training and testing files
     0   0.0  100.0  100.0   88.3
     1   0.0  100.0   69.7   74.9
     2   0.3   82.2    3.5   37.9
     3   0.1   96.8    2.0   43.6
     4   0.0   99.3    1.9   44.7
     5   0.0   98.4    1.8   44.2
     6   0.2   97.1    1.7   43.7
     7   0.2   89.1    1.7   40.1
     8   0.2   79.0    1.7   35.7
     9   0.2   70.5    1.7   31.9
    10   3.4   55.2    1.8   25.5
    11   2.9   48.3    1.8   22.4
    12   1.8   41.2    1.8   19.2
    13   1.3   33.9    1.8   15.9
    14   2.9   26.2    1.8   12.7
    15   1.9   22.8    2.7   11.5
    16   2.2   20.6    3.2   10.8
    17   2.9   19.4    2.7   10.1
    18   7.0   16.2    2.8    9.2
    19   3.9   16.8    3.0    9.2
    20   2.9   17.4    3.1    9.4
    21   2.4   17.2    2.9    9.2
    22   2.4   17.5    2.3    9.0
    23   2.4   16.3    2.8    8.7
    24   1.4   15.9    2.7    8.4
    25   1.4   14.6    4.4    8.5
    26   1.3   13.9    5.0    8.5
    27   1.3   13.0    4.5    7.9
    28   0.9   13.5    3.7    7.7
    29   0.8   14.0    4.4    8.2
    30   0.6   13.7    3.4    7.6
    31   0.3   15.1    3.5    8.3
    32   0.3   15.8    3.0    8.4
    33   0.5   14.3    2.2    7.3
    34   0.3   13.1    2.8    7.0
    35   0.3   12.3    2.8    6.7
    36   0.5   11.8    2.7    6.5
    37   0.6   10.9    2.6    6.0
    38   0.7   10.5    2.4    5.8
    39   0.7   10.1    2.2    5.5
    40   0.6   10.3    2.4    5.7
    41   0.3   10.9    2.7    6.1
    42   0.5   10.1    2.2    5.5
    43   0.3   10.7    2.2    5.7
    44   0.7    8.7    2.2    4.9
    45   0.6    9.4    2.0    5.1
    46   0.8    8.7    2.0    4.8
    47   0.8    8.0    1.9    4.5
    48   0.5    9.6    2.1    5.2
    49   0.6    9.1    2.7    5.3
    50   0.6    8.2    2.4    4.8
    51   0.7    7.6    2.0    4.3
    52   0.8    6.5    2.0    3.9
    53   0.8    5.8    2.0    3.5
    54   0.8    6.4    1.9    3.8
    55   0.9    5.9    1.9    3.5
    56   0.8    6.2    1.9    3.7
    57   0.8    6.7    1.9    3.9
    58   0.7    6.9    1.9    4.0
    59   0.7    6.6    2.2    4.0
    60   0.7    6.1    2.0    3.7
```

```
61   0.8    5.6    2.2    3.5
62   0.8    5.6    2.7    3.7
63   0.8    5.0    2.0    3.2
64   0.8    6.1    1.9    3.7
65   0.5    6.9    1.9    4.0
66   0.7    6.9    1.8    3.9
67   0.6    6.7    1.9    3.8
68   0.8    6.2    1.9    3.7
69   0.8    6.1    1.9    3.6
70   0.8    4.9    1.9    3.1
71   0.8    4.8    1.9    3.1
72   1.4    3.9    1.9    2.7
73   1.3    4.0    1.9    2.7
74   0.9    4.6    1.9    3.0
75   0.8    5.1    1.9    3.2
76   0.7    5.6    1.9    3.4
77   0.8    5.2    1.9    3.2
78   0.8    5.4    1.9    3.3
79   0.9    4.2    1.9    2.8
80   1.1    3.9    1.9    2.7
81   1.1    4.1    1.9    2.8
82   0.8    4.5    1.9    2.9
83   0.8    4.7    1.9    3.0
84   0.8    4.3    1.9    2.8
85   0.9    4.2    1.9    2.8
86   1.0    4.1    1.9    2.8
87   1.0    4.0    1.9    2.7
88   1.4    3.3    1.8    2.4
89   1.4    3.9    1.9    2.7
90   1.4    3.9    1.8    2.6
91   0.9    4.6    1.8    2.9
92   0.8    4.9    1.9    3.1
93   0.9    4.7    1.8    3.0
94   1.0    4.0    1.9    2.7
95   1.0    3.9    2.1    2.8
96   1.1    3.9    1.9    2.7
97   1.4    3.5    1.9    2.6
98   1.1    4.0    1.9    2.7
99   1.3    4.0    1.7    2.7
```

net5 uses the same algorithm as net4, but it is used to examine details of the classification behavior following all the training rounds. telnet, WWW, and sendmail sessions are submitted to net5. The first two lines printed are essentially the same as the first and last lines printed for net4, namely, the misclassification rates for each service and the aggregate misclassification rate. As shown below, following the 99 training rounds, the misclassification rates for telnet, WWW, and sendmail are 1.3%, 4.0%, and 1.7% respectively. The aggregate misclassification rate is 2.7%.

net5's real value are on the subsequent lines. Detailed classification information is presented for each class. For class 1 (the telnet sessions) the first line shows 862 were classified as belonging to class 1 (telnet), 7 were classified as class 2 (WWW), and 4 were classified as class 3 (WWW). The second line shows the classification distribution for sessions correctly classified, and the third line shows the classification distribution for sessions incorrectly classified.

Continuing with class 1 (telnet), the fourth line shows the confidence value distribution for the classification. When a session is classified, not only does the neural network indicate the class it thinks the session belongs in, it also generates a score between 0 and 1. This value can be roughly interpreted as the confidence value for the classification. In this example, 531 sessions classified had a confidence value between 0.9 and 1.0; 170 sessions classified had a confidence value between 0.8 and 0.9; and 4 sessions had a confidence value of 0.4 to 0.5.

The fifth and sixth lines show the confidence values for sessions which were correctly classified and incorrectly classified respectively. As can be seen, of the eleven sessions misclassified, two had confidence values between 0.8 and 9.0, and none had a confidence between 0.9 and 1.0.

```
yoda% net5 -c 23.norm.train 23.norm.test -c 80.norm.train 80.norm.test -c
25.norm.train 25.norm.test
Finished reading in training and testing files
  0.0  100.0  100.0   88.3
  1.3    4.0    1.7    2.7
For class: 1
     0   862     7     4     0     0     0     0     0     0
     0   862     0     0     0     0     0     0     0     0
     0     0     7     4     0     0     0     0     0     0
     0     0     0     1     4     2    20   145   170   531
     0     0     0     0     1     1    18   143   168   531
     0     0     0     1     3     1     2     2     2     0
For class: 2
     0   122  3165    10     0     0     0     0     0     0
     0     0  3165     0     0     0     0     0     0     0
     0   122     0    10     0     0     0     0     0     0
     0     0     1    24   113   109   136   326  1430  1158
     0     0     1    17    89    93   122   302  1417  1124
     0     0     0     7    24    16    14    24    13    34
For class: 3
     0    37    18  3242     0     0     0     0     0     0
     0     0     0  3242     0     0     0     0     0     0
     0    37    18     0     0     0     0     0     0     0
     0     0     6    11    20    27    47    83   255  2848
     0     0     1     2     4    17    36    81   255  2846
     0     0     5     9    16    10    11     2     0     2
```