Network Security Monitor

Final Report

L. Todd Heberlein

This final report is prepared at the request of Lawrence Livermore National Laboratory (LLNL) and the University of California, Davis (UCD). All material presented in this report, as well as all associated code, have been twice delivered to LLNL in electronic form, the first time in February of 1995. No work has been done on this project since the February 1995 delivery.

README FILES

This section presents two of the most important README files included with the Network Security Monitor (NSM) software distribution. The first README file presents an overview of the NSM and its software distribution. The second README file presents a history of the changes to the NSM over the years, including the most recent changes.

NSM Overview

The NSM is not a program but a suite of tools to search for intrusive activity occurring over a network. The tools can be roughly broken down into three catagories: data capture tools, data analysis tools, and support tools.

Data capture tools save network traffic to disk for later analysis. In addition to capturing data, DIDS_lan_mon and X_nsm_kernel also perform on-the-fly analysis. Two of the capture programs, etherdump and network_capture, are inclused for historical purposes; if you are just installing the NSM tools, I would recommend not using these tools.

Data analysis tools are the core of the NSM suite; these are the tools which actually detect and support analyses of intrusive activity. With the exception of the GUI_nsm, these are post-mortem tools to investigate data already saved to disk. In addition to analyzing data collected by one of the NSM's data capture tools, these post-mortem tools can also analyze data collected by TCPdump.

Support tools manipulate existing data to support further analysis and enable the other NSM tools. tcpdump_conv will convert data saved by the tcpdump program into data which can be analyzed by the NSM tools.

The tools, their catagories, and the platforms on which they run are presented below...

```
Data capture tools:
===================
    DIDS_lan_mon  (part of DIDS pkg)            SunOS 4.x
    X_nsm_kernel  (used w/ GUI_nsm)             SunOS 4.x
    capture                                     SunOS 4.x
    etherdump           (old, not supported)    SunOS 4.x
    network_capture     (old, not supported)    SunOS 4.x

Data analysis tools:
====================
    analyze                         SunOS 4.x    NeXTSTEP 3.0
    packet_print                    SunOS 4.x    NeXTSTEP 3.0
    playback                        SunOS 4.x    NeXTSTEP 3.0
    previewer                       SunOS 4.x    NeXTSTEP 3.0
    report                          SunOS 4.x    NeXTSTEP 3.0
    transcript                      SunOS 4.x    NeXTSTEP 3.0
    GUI_nsm       (need X windows)  SunOS 4.x

Support tools:
====================
    run_install                     SunOS 4.x    NeXTSTEP 3.0
    tcpdump_conv                    SunOS 4.x    NeXTSTEP 3.0
    stream                          SunOS 4.x    NeXTSTEP 3.0
    top_con                         SunOS 4.x    NeXTSTEP 3.0
    warn_sort                       SunOS 4.x    NeXTSTEP 3.0
```

As mentioned previously, network traffic can be captured by the program tcpdump and analyzed with the NSM tools. Below are the hardware and operating systems on which tcpdump currently runs (taken from the tcpdump-2.2.1 README file):

machine	os	packet filter
hp300	4.3BSD Tahoe/Reno	bpf
sparc	SunOS 4.x	bpf, nit
sun3	SunOS 3.5, SunOS 4.x	bpf, nit
Decstation	Ultrix 4.0 (and higher)	packetfilter
IBM RT	4.3BSD	enet
386/486	4.3BSD netII	bpf

Although we have only had access to a tcpdump on a SPARCstation, we do believe that data files from the other machines should work as well. Run the tcpdump program with the snaplength equal to or greater than your network's maximum transmission unit (mtu); "-s 1550" should work in most cases. Also, use the -w option to save the data to a file. For example,

```
% tcpdump -s 1550 -w tcpdump.data host athena.mit.edu
```

will save all traffic from the host athena.mit.edu to the data file tcpdump.data. This data can then be converted to an NSM data file with the tcpdump_conv program.

Network Security Monitor (NSM) V 0.8, 25 June 93

This is the main directory for the Network Security Monitor (NSM). The NSM is a set of tools designed to help a security officer detect and analyze intrusive behavior over a network.

Currently the NSM tools only work on Sun computers running SunOS 4.1.x and NeXTstations running NeXTSTEP 3.0 (I have not tried 3.1 yet).

DIRECTORY DESCRIPTIONS

analysis: The main directory in which most analysis will be performed.

bin: The directory holding a collection of programs which make up the suite of the NSM tools. The directory should already contain the tools compiled for a SPARC computer.

doc: The directory holding the documentation for the NSM tools. Currently, only the manual pages ("man" pages) and PostScript.

tmp: This is a "scratch" directory used for storing the network data files. This directory is specified in the config.file in the NSM/analysis directory - a new data directory can be changed by changing the config.file

DIFFERENCES FROM VERSION 0.3

- New tools: stream, packet_print, playback, and previewer have been added to the suite of NSM tools.
- A slightly modified version of Tim Tessin's etherdump program is included in the sute of tools. Currently I do not have a man page; however, the usage is similar to that of etherfind.
- Although network_capture and the version of etherdump provided in this package still generate files in the format logYYMMDD.HH, the other analysis tools ignore the file names; they determine times covered by the files by looking at the times of the network packets themselves. This solves two problems: analyzing data collected in a different time zone, and analyzing data created by Tim Tessin's orginal etherdump program.
- A bug generating transcript file for remote shells has been fixed by including a "-n" option. See the man page for transcript.
- Code reduction. Much of the code has been rewritten, and common code has been extracted and placed in the directory src/Common. This has resulted in a much smaller package.
- On line documentation. See above.

CHANGES FROM V 0.6 TO V 0.6b:

- top_con now accepts the same options that transcript does. A small bug when changing the permissions on the output_file (making it executable) has been fixed.
- analyze has extra error checking.
- the NSM doesn't barf on directories or compressed files (files ending ".Z") in the data directory.
- manual pages have been updated, and the the file structure for documentation now has NSM/doc/man/man1. This allows you to set your man path to include the NSM man pages (e.g., ~heberlei/NSM/doc/man).

  catman has been run on the manual pages and the results placed in NSM/doc/man/cat1. These can be easily viewed with "more."
- Optimize flags have been turned on in the Makefiles.
- A draft NSM document (written in Microsoft Word 4.00 for the Macintosh) has been updated and added in NSM/doc/nsm.sit.hqx. The file has been archived and binhexed with StuffIt Classic 1.6.

CHANGES FROM V 0.6b TO V 0.6c:

- A bug in analyze, which caused problems when sometimes analyzing data which crossed monthly boundaries, has been fixed.
- A bug in transcript was fixed to print the connection file name and the connection index properly in the transcript header.
- A new program called capture2 has been added. See the man page for additional information.

CHANGES FROM V 0.6c TO V 0.6d (UNOFFICIAL):

- capture2 has been replace with capture3. From the user's point of view there is no difference; however, significant changes were made to the design and structure of the code. These changes were designed to facilitate code reuse in other modules
- LAN_kernel was added to the source code directory. This is the code used by both the LAN monitor portion of DIDS and an X-window based NSM. This merging of the two projects is should reduce the code maintenace problem and speed up my work.
- GUI directory has been added to the source code directory. This code is the X-windows code John Fisher developed to interface with the LAN kernel.
- GUI_xterm was added to the source code directory. This is a slightly modified version of the xterm source code. The code is used by the NSM's X based GUI to create user monitors (read "wire tap").

CHANGES FROM V 0.6d (UNOFFICIAL) TO V 0.6e (UNOFFICIAL):

- analyze has been changed so that, when LLNL is defined (see the Makefile), an existing connection log file will NOT be overwritten. Instead, a connections.log.n file is created where 'n' is the lowest index file possible. For example, if the file connections.log already exists when analyze is run, the connection log file will be "connections.log.1". If analyze is run yet again, the file "connections.log.2" will be created.

CHANGES FROM V 0.6e (UNOFFICIAL) TO V 0.6f (UNOFFICIAL):

- Major portions of the code directories have been restructured. Mainly, code which was almost identical was placed in the "Shared_source" directory. This code is shared between several programs; however, unlike the code in the Common directory, the code cannot be compiled once for all the NSM tools. In each of the directories which need access to the shared code, symbolic links are made to these files. These changes will reduce the total code size, and, hopefully make the maintenance easier.
- transcript now prints the internet names, if possible, of the source and destinion hosts in the transcript header.

24 Aug 92: CHANGES FROM V 0.6f (UNOFFICIAL) TO V 0.7

- When compiled with the LLNL option, the previewer tool prints the connection index on both the first AND the second line of a connection. This allows awk programs (or grep) to print the connection index when searching for access by/to certain hosts.
- When previewer tries to print a connection by an unknown service, it now prints the source and destination ports of the connection after indicating that the service is "unknown"
- The capture tool now takes advantage of the DB file exceptions.file. This file allows the user to capture all traffic specified by the address_filter.file and service_filter.file EXCEPT for traffic matching that in the exceptions.file. See the man pages for exception.file (exceptions.file(5)) and capture (capture(1)) for more detail.
- A bug has been fixed when the NSM tools attempt to process some malformed packets. For example, if the TCP header indicates that the packet is longer than that reported by the IP header, the packet is considered malformed and is discarded. This is a very very rare event.
- Several new manual pages (man pages) have been added.
- The NSM must be registered to a particular machine. If not, the NSM tools will not run. In order to register your copy of the NSM, execute the program run_install from the analysis directory. The program will present you with an ID which you must give to your NSM distributor. Your NSM distributor will then give you a password to install on your machine.
- If you are evaluating the NSM tools on a test basis. The NSM tools will not work properly beyond the test expiration date. If you want a permanent release (and all future updates), please contact your NSM distributor for a new release.

21 Sep 92: CHANGES FROM V 0.7 TO V 0.7a

- A bug in capture (which was introduced in v6.f) that would cause the program to stop running after about 1.5 days has been fixed.
- The output from analyze can now be redirected to a user specified file by using the -o command line option (see analyze(1)). For example, the user can create a connection log file named test.log by:
- analyze -o test.log YY MM DD HH num_of_hours

  The next release of analyze will try to get rid of the ugly date format arguments currently required.
- A new tooled called "report" has been added. It allows the user to view the connections in a connection log file in a number of ways. Please see report(1) for more information.

22 Oct 92: CHANGES FROM V 0.7a TO V 0.7b

- A new tool, tcpdump_conv, has been added. tcpdump_conv converts a data file created by the program TCPdump into a file format which can be read in by the NSM tools. Usage is:

  ```
  % tcpdump_conv < tcpdump_file > nsm_data_file
  ```

  The major advantage of being compatible with TCPdump is that TCPdump has been ported to a variety of platforms (HP300, IBM RT, DECststion, 386/486 running 4.3BSD net II, and of course Sun-3s and SPARCstations).

  When running TCPdump, use a snapp length (-s option I believe greater than the Maximum Transmission Unit (MTU) of your local network. This will guarantee that not network packets will be cut in half (TCPdump does not save the entire packet under normal conditions). A snapp length of 1550 will probably work fine on most Ethernets.
- Much of the code is now compiled statically, so the NSM does not require the same libraries on the remote machines.

27 Oct 92: CHANGES FROM V 0.7b TO V 0.7c (UNOFFICIAL)

- Some changes have been made to analyze to allow it to specify the processing of a single network data file. That is, the awkward date format normally used by anlyze is not needed to process a single data file.

  For now, the old analyze tool remains, and the new analyze tool is named analyze2 (see analyze2(1)). When I become comfortable that everything is working properly with analyze2, it will be renamed analyze, and the old tool will be removed.
- The man page for the report tool has been updated. If the NOT symbol, '!' is specified in a match, the character must be preceeded with the escape character '\'. This is now reflected in the documentation.

25 June 93: CHANGES FROM V 0.7c (UNOFFICIAL) TO V 0.8

- The primary change with this release is the NeXTSTEP support for many of the NSM analysis tools. The following tools can now be run on a NeXT: analyze, packet_print, playback, previewer, report, transcript, run_install, tcpdump_conv, stream, top_con, and warn_sort. All tools run exactly the same on both platforms.

  To date, we have only been able to test these tools on a NeXTstation running NeXTSTEP 3.0; we have not had an opportunity to test under NeXTSTEP 3.1 or NeXTSTEP on Intel machines.

  The data collected by the NSM capture tools on a SPARCstation can be analyzed on either platform. Similarly, data collected by tcpdump on a SPARCstation can be converted to NSM data files on either platform. We have not tested data collected by tcpdump on other platforms; however, we believe this should work as well. Please let us know if you find out.

- The old analyze has been discontinued, and analyze2 has been renamed "analyze". Since analyze2 was never part of an official release, most users only need to know that analyze arguments have changed.

  To use the updated analyze program like the previous version, add the argument "-date" before the input date. For example,

  ```
  OLD: % analyze 93 6 10 6 24
  NEW: % analyze -date 93 6 10 6 24
  ```

  The new analyze also supports the analysis of a single data file. For example, to analyze the single data file "special.data" in the directory ../tmp, use:

  ```
  % analyze -i ../tmp/special.data
  ```

  See the man page analyze(1) for more information.

- Bob Palasek has been named as the key distributor. When installing the NSM, execute the run_install program to get your special ID number. Give this number to Bob Palasek (number and address provided in the run_install program).

24 February 95: CHANGES FROM V 0.8 TO V 0.9

- Transcript has been enhanced in three major ways. First, the TCP sequence numbers are used to recognize missing and duplicate data. The missing data can be replaced with a "place holder" character. The default is the letter 'X', but it can be changed. For example, if an intruder types "rlogin", but you miss the second byte, transcript will print out "rXogin" (as opposed to the earlier transcript output of "rogin"). Also, if an intruder types "guest", but the 'g' gets transmitted twice, we will still only see "guest" (as opposed to the earlier transcript output of "gguest").

  The second enhancement is support for the parsing and filtering of telnet negotiation protocol. When a telnet client initiates a connection with a telnet server, the client and server exchange several messages (called negotiations) to determine such things as terminal type, window size, and terminal speed. Previously these showed up as squiggly characters and curly braces at the beginning of the transcript file. Now they are removed

(any information discovered from the negotiations is included at the bottom of the transcript file.

The third enhancement is support for the NFS sessions. A transcript of a UDP-based NFS session will present a sequence of rows, each row associated with a request (and possibly reply results) between the client and the server. The row consists of the user's UID, his host's name, the program (always NFS), the procedure name (e.g., RFS_WRITE), the file name on which the operation is to be performed (if possible), and the results (Ok, not enough permissions, not owner, etc.).

The transcript man page has been updated to reflect these changes.

- o Analyze has been enhanced to analyze UDP-based sessions with the portmapper and NFS daemons. General RPC patterns can be detected including the the use of specific program and procedure (e.g., the portmapper program and its CallIt procedure) and the 16-bit UID attack. Also, NFS patterns can be detected including access to key file names and error conditions.

  The strings.file man page has been updated to reflect the new patterns which can be searched.

- Some minor bugs were patched.

BUG REPORTS

Please send bug reports to:

`heberlei@cs.ucdavis.edu`

or

Todd Heberlein
Department of Computer Science
University of California
Davis, Ca. 95616

I would like thank LLNL, the USAF, and Haystack Laboratories, Inc. for their support and direction.

UNIX Manual Pages
User Commands

This section the presents manual pages (often referred to as “man pages”) for the various programs delivered as part of the NSM software distribution. These manual pages are also available in an on-line form for UNIX computer systems.

NAME

analyze – NSM network analyzer (SunOS 4.1.1)

SYNOPSIS

analyze [-d <dir>] [-o <log_file>] (-i <data_file> | -date <yy> <mm> <dd> <hh> <duration>)

analyze identifies individual network connections and assigns warning values to each connection. analyze reads in network packets from a data file created by capture(1), network_capture(1), or etherdump(1). It can also analyze data files created by TCPdump by translating the TCPdump data file with tcpdump_conv(1). The directory containing these data files is specified in the configuration file config.file; however, another data directory can be specified with the "-d dir" option.

analyze creates a file called connections.log (the default name) containing the list of the identified connections, and the DB files profile.file and con_count.file are updated. The output file can be modified with the -o option.

analyze differs from the original analyze program by either accepting a data file name (with the -i option), or accepting the traditional date format. However, the date format must now be preceeded with the -date flag.

The traditional date format requires five arguments. The first four, yy mm dd hh, specify the hour for which you want to analyze data, and the argument "duration" specifies the total number of hours you would like to analyze. yy is the year specified as the number of years since 1900. mm is the month (Jan = 1, Dec = 12). dd is the day of the month. And hh is the.

CAVEATS

analyze normally overwrites any existing connections.log file. However, if the -DLLNL flag is set in the CFLAGS at compile time, analyze writes to another file of the form connections.log.#, where '#' is the lowest integer (starting at 1) for which another file by that name does not exist. For example, the first run of the program will produce connections.log, the second run will produce connections.log.1, the next connections.log.2, and so on.

Running more than one analyze job simultaneosly will result in an incorrect profile.file. For example if you run one analyze job saving the output to out1.log and run a second job saving the output to out2.log, whichever analyze job finished last will wipe out the changes to profile.file that the job which finished first made.

OPTIONS

-d <dir>

use the directory "dir" as the source of the network packet data files. The default data directory is listed in config.file

-o <log_file>

use the name <log_file> instead of connections.log as the output of analyze. Warning, even when compiled with the LLNL option, if the -o command line option is used, analyze will overwrite any file with the same name as <log_file>.

-i <data_file>

process the network data file called "data_file".

-date <yy> <mm> <dd> <hh> <duration>

starts processing packets beginning after the time specified by the year, yy, month, mm, day of the month, mm, and hour, hh. Processing ends when <duration> hours of network packets have been processed.

USAGE

analyze must be started in a directory containing a configuration file, config.file, and a DB subdirectory containing the required database files.

To start analyze, enter the command and the required arguments. For example, to process 24 hours worth of data starting on Dec 18, 1991 at 6 AM. enter:

```
% analyze  -date 91  12  18  6  24
```

If you are executing the code from the "analysis" directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/analyze  -date 91  12  18  6  24
```

If you have another directory containing the the network packet data files, for example ../tmp2, you can use the -d option:

```
% ../bin/analyze -d ../tmp2  -date 91  12  18  6  24
```

To analyze a single data file, special.data, in the directory ../tmp, use the -i option:

```
% analyze -i ../tmp/special.data
```

FILES

The following files must be in the current working directory:

```
config.file
DB/con_count.file
DB/host.file
DB/profile.file
DB/strings.file
DB/tcp.file
DB/udp.file
```

analyze generates the file, in the current working directory,

```
connections.log
```

MAKEFILE

To make analyze, just type "make" at the command line in the NSM/src/Analyze directory. It defaults to "make all," and places the executable program "analyze" in the NSM/src/Analyze directory.

"make install" will make analyze and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

SEE ALSO

etherdump, network_capture, packet_print, playback, stream, top_con, transcript, warn_sort

NAME

capture – another NSM packet grabber (SunOS 4.1.1)

SYNOPSIS

capture [-verbose] [-no_stats] [-debug] [-no_checks] [-i <interface>] [-svc svc_file] [-addr addr_file]

DESCRIPTION

capture is yet another NSM tool to extract network packets off the ethernet. capture provides a finer level of control over which traffic to capture from the network than network_capture, and its usage is much easier than etherdump.

capture reads in two files from the DB directory controlling the filtering of packets: service_filter.file and address_filter.file. service_filter.file specifies which services a user wishes to capture; only packets associated with the services listed in this file will be captured. address_filter.file specifies the set of hosts you are interested in protecting. Only packets between one of the "protected" hosts and an "unprotected" host will be captured.

The network traffic will be stored in files as specified by the config.file.

CAVEATS

Since capture places the ethernet controller in promiscuous mode, root privilege is required to execute it.

capture, as with most of the NSM tools, must be started in the NSM/analysis directory.

The program places various statistical information in the file stats.log.

The format for the service filter file is the same as the format used in the file /etc/services - except comment lines (beginning with #) are currently not accepted. This format has the service name followed by the port/protocol. Any further information on the line is ignored.

The format for the address filter file is one class A net, class B net, class C net, host internet address, or host internet name per line. After the first address/name on the line, all other text until the end-of-line is reached is considered comments. To specify a class A, B, or C network, enter the network address terminated by a period. For example, "128.", "128.120.", and "128.120.56." represent a class A, B, and C network respectfully.

OPTIONS

-verbose

prints extra information associated with the internal workings to the screen.

-no_stats
prevents the printing of statistical information to the stats file.

-debug
used mostly to help debug the code.

-no_checks
prevents the program from performing checksums on the network traffic. This can speed up the data capture (therefore reducing the probability of missing a packet), but it introduces a chance for bad data to slip into the data files.

-i <interface>
requests a specific ethernet device to use (e.g., "ie0" for the Intel Ethernet device, and "le0" for the Lance Ethernet device). Without this option, the program asks the operating system which device to use.

-svc <svc_file>
requests a service filter file other than the default file. svc_file is the alternate file.

-addr <addr_file>
requests an address filter file other than the default file. addr_file is the alternate file.

USAGE

To start capture, simply enter the command:

```
% capture
```

If you are executing the program from the NSM/analysis directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/capture
```

FILES

The following files must be in the current working directory:

```
config.file
DB/
```

capture generates, as specified in config.file, network packet data file of the form

```
logYYMMDD.HH
```

MAKEFILE

To make capture, just type "make" at the command line in the NSM/src/Capture directory. It defaults to "make all," and places the executable program "capture" in the NSM/src/Capture directory.

"make install" will make capture and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

The program does not check for available disk space, so it can fill up the disk. Since the program runs with root privilege, it can fill the disk to 111%.

SEE ALSO

analyze, etherdump, network_capture, packet_print, playback, stream, top_con, transcript, warn_sort

NAME

DIDS_lan_mon – LAN monitor for the Distributed Intrusion Detection System (DIDS) (SunOS 4.1.1)

SYNOPSIS

DIDS_lan_mon [-verbose] [-no_stats] [-debug] [-no_checks] [-save] [-no_agent] [-i device]

DESCRIPTION

DIDS_lan_mon is the program which provides network information to the Distributed Intrusion Destion System (DIDS). It captures the Ethernet traffic, identifies new connections, identifies connection closings, matches strings in the connection data, and provides live monitoring of intruders.

DIDS_lan_mon determines which network traffic to filter for based on the settings specified in the DB file sets.file. This file uses a rather cryptic format created for DIDS to describe which hosts and services to monitor as well as which strings to search for (see sets.file(1) for a description of this format). Future releases will probably allow the current filtering method or the method provided for the program capture to be used.

CAVEATS

Because the program places the Ethernet controller in promiscuous mode, the user needs to run it as root.

This program is usually run in conjunction with an agent which passes data and commands between the DIDS_lan_mon and the rest of DIDS.

OPTIONS

-verbose

prints extra information associated with the internal workings to the screen.

-no_stats

prevents the printing of statistical information to the stats file.

-debug

used mostly to help debug the code.

-no_checks

prevents the program from performing checksums on the network traffic. This can speed up the data capture (therefore reducing the probability of missing a packet), but it introduces a chance for bad data to slip into the data files.

-i <interface>

requests a specific ethernet device to use (e.g., "ie0" for the Intel Ethernet device, and "le0" for the Lance Ethernet device). Without this option, the program asks the operating system which device to use.

-save

requests that all network traffic be saved to data files for further analysis.

-no_agent

informs the program that the program is to run in stand-alone mode (that is, without the DIDS' agent.

USAGE

To start DIDS_lan_mon simply enter the command:

```
% DIDS_lan_mon
```

If you are executing the program from the NSM/analysis directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/DIDS_lan_mon
```

FILES

The following files must be in the current working directory:

```
config.file
DB/
```

With the -save flag, DIDS_lan_mon generates, as specified in config.file, network packet data files of the form

```
logYYMMDD.HH
```

MAKEFILE

To make DIDS_lan_mon just type "make" at the command line in the NSM/src/LAN_kernel directory. It defaults to "make all," and places the executable programs "X_nsm_kernel" and "DIDS_lan_mon" in the NSM/src/LAN_kernel directory.

"make install" will make both programs and place copies in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

The program does not check for available disk space, so it can fill up the disk. Since the program runs with root privilege, it can fill the disk to 111%.

SEE ALSO

analyze, etherdump, network_capture, packet_print, playback, stream, top_con, transcript, warn_sort

NAME

network_capture – NSM packet grabber (SunOS 4.1.1)

SYNOPSIS

network_capture

DESCRIPTION

network_capture extracts the network packets off the ethernet, filters for certain IP packets, and saves the filtered packets in data files.

network_capture reads the configuration file config.file to determine the filter parameters, the directory and root file name for the data files, and the time between file name switches. The file config.file must be in the current working directory.

CAVEATS

Since network_capture places the ethernet controller in promiscuous mode, root privilege is required to execute it.

USAGE

network_capture needs to be started in a directory containing the configuration file: config.file.

To start network_capture, simply enter the command:

```
% network_capture
```

If you are executing the program from the NSM/analysis directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/network_capture
```

To start a network_capture program and place it in the background, enter the command with an ampersand following it:

```
% network_capture &
```

FILES

The following files must be in the current working directory:

```
config.file
```

network_capture generates, as specified in config.file, network packet data file of the form

```
logYYMMDD.HH
```

MAKEFILE

To make network_capture, just type "make" at the command line in the NSM/src/Network_capture directory. It defaults to "make all," and places the executable program "network_capture" in the NSM/src/Network_capture directory.

"make install" will make network_capture and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

The program does not check for available disk space, so it can fill up the disk. Since the program runs with root privilege, it can fill the disk to 111%.

SEE ALSO

analyze, etherdump, packet_print, playback, stream, top_con, transcript, warn_sort

NAME

packet_print – NSM packet printer (SunOS 4.1.1)

SYNOPSIS

packet_print [-break] [-hex | oct] [-s] [-t] [-e] [-ip] [-tcp | udp] data_stream

DESCRIPTION

packet_print reads in a network packet data file, data_stream, and prints information about the packet to the screen. The various options determine how much information about each packet is to be displayed.

Although packet_print can print the packets of a data file created with network_capture or etherdump, it was designed to display a filtered stream of data packets created by the program stream.

Without any options, packet_print displays only the data portion of each packet - one byte per line. The byte in printed as an ASCII letter if printable, and the decimal value is printed as well. For example:

```
h  104
e  101
l  108
l  108
o  111
```

OPTIONS

-break

prints a dotted line, "---------" between bytes separated by packets. That is, packet boundaries are shown. The packet boundaries are automatically printed if the -t, -t, -ip, -tcp, or -udp falgs are used.

-hex

prints the bytes in hexadecimal as opposed to decimal. This option cannot be used with the -oct option.

-oct

prints the bytes in octal as opposed to decimal. This option cannot be used with the -hex flag.

-s

prints the bytes with the 8th bit masked out, and then if the 8th bit was on, the word "signed" is printed following the byte. This 8th bit is often used as a checksum for data being processed by modems.

-t

will print the time stamp for each packet. The format is given in seconds and micro-seconds.

-e
prints the ethernet header information out.

-ip
prints out the internet header out.

-tcp
prints the TCP header out.

-udp
prints the UDP header out.

USAGE

To run packet_print, simply enter the command:

```
% packet_print data_stream
```

where data_stream is a network packet data file. Typically packet_print will generate a large number of lines, so it would be wise to redirect the output to a file or pipe the output through "more".

FILES

The data_stream is a network packet data file created with programs stream, network_capture, or etherdump.

MAKEFILE

To make packet_print, just type "make" at the command line in the NSM/src/Packet_print directory. It defaults to "make all," and places the executable program "packet_print" in the NSM/src/Packet_print directory.

"make install" will make packet_print and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

SEE ALSO

etherdump, network_capture, stream

NAME

playback – NSM stream player (SunOS 4.1.1)

SYNOPSIS

playback [-t max_wait] [-ff fast_forward] data_stream

DESCRIPTION

playback is used to play a recorded network connection directly back to the screen as if the events were "live." The screen/window in which playback is run will interpret all screen control commands; therefore, allowing the viewing of intruders using visual editors, the UNIX talk program, and other programs which use screen control commands. The required input, data_stream, is a network packet data file consisting of packets for a single stream of a connection.

Linear printouts, such as those created by the transcript program, are not very useful when analyzing an intruder using programs which rely heavily on screen control (e.g., vi, emacs, and talk). This program is used to analyze such activity.

If the playback is run in a screen/window with the same (or very similar) terminal type (i.e., vt100 and xterm) as that used by the intruder, you will be able to see exactly what the intruder saw on his/her screen.

Timestamps are used, so the display timing is the same as the original activity. The typing speed, pauses, etc. will be displayed at the same rate as the original activity. By using the options, this timing can be modified.

The required option data_stream is usually created with the "stream" program. stream creates two files: *.stream.dest and *.stream.init. The file *.stream.dest is the data displayed on the intruders screen, and therefore, it should be the file viewed by the playback program.

OPTIONS

-t <max_wait>

This option specifies the maximum amount of time you want the program to wait before printing the next packet data. The actual intruder may have long delays (from minutes to hours) which can be filtered out with this option. The value max_wait is a float representing the maximum amount of time in seconds you want to wait. Therefore, "-t 1" would state that the program should wait at most one second before displaying the data in the next packet. "-t 1.5" has a maximum wait of one and a half seconds.

-f <fast_forward>

This option specifies a different speed that you want the data to be played back in. The value fast_forward is a real number representing the playback time ratio. For example, the "-ff 2" option would fast forward through the

data at twice normal speed. On the otherhand, "-ff 0.5" would play through the data at half the normal speed.

USAGE

To run playback on a network packet data file called con.98.stream.dest, simply enter the command:

```
% playback con.98.stream.dest
```

If you would like to play the program back at 150% of normal speed, and you want a maximum wait time of one second, enter:

```
% playback -t 1 -ff 1.5 con.98.stream.dest
```

FILES

The data_stream is a network packet data file created with the program stream.

MAKEFILE

To make playback, just type "make" at the command line in the NSM/src/Playback directory. It defaults to "make all," and places the executable program "playback" in the NSM/src/Playback directory.

"make install" will make playback and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

It is may be possibe that certain screen control commands (i.e. ctrl-s) displayed by playback can freeze your screen/window.

The screen control commands may leave your window in a strange state. For example, only half the window may scroll. Commands such as "reset" and "tset" can rset your screen/window to its proper state.

SEE ALSO

etherdump, network_capture, stream, transcript

NAME

previewer – NSM connection viewer (SunOS 4.1.1)

SYNOPSIS

previewer [-n num] connection_file

DESCRIPTION

previewer previewer takes as input a connection log file and generates a human readable print out of the connections in the connection_file.

OPTIONS

-n <num>

prints only the first num connections in the connection file.

USAGE

To run previewer, simply enter the command with the required argument. For example, to view the connection log file warn91-12-20, enter the command:

```
% previewer warn91-12-20
```

If you are executing the code from the "analysis" directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/previewer warn91-12-20
```

To view only the top ten connections, use the -n option:

```
% ../bin/previewer -n 10 warn91-12-20
```

For example:

```
denali# ../bin/previewer -n 2 warn92-2-2
1382  src:   128.120.56.188               toadflax.cs.ucdavis.edu
      dst:   134.139.1.21                        beach.csulb.edu
      svc:  --  telnet --
      start: Sun-Feb-02-21:41:20-1992   stop: Sun-Feb-02-21:41:35-1992
     warning value: 8.097
      strings from source computer:
      strings from destination computer:
          login: root   1
          Login incorrect   2
--------------------------------------------------
410   src:   36.14.0.150                     shasta.Stanford.EDU
      dst:   128.120.56.133                cayenne.cs.ucdavis.edu
      svc:  --  telnet --
      start: Sun-Feb-02-11:35:56-1992   stop: Sun-Feb-02-11:44:08-1992
     warning value: 7.441
      strings from source computer:
      strings from destination computer:
          Permission denied  28
          Login incorrect   1
--------------------------------------------------
```

In the above example, two connections are shown: connections 1382 and 410. The first connection is a telnet from toadflax to beach. It begins at 21:41:20 (9:41pm and 20 seconds) and stops fifteen seconds later. The string "login: root" was matched

once, and the string "Login incorrect" was matched twice. This would indicate that someone tried to login as root, and, because of the "Login incorrect" and the short connection time, the attempt was unsuccessful.

MAKEFILE

To make previewer just type "make" at the command line in the NSM/src/Previewer directory. It defaults to "make all," and places the executable program "previewer" in the NSM/src/Previewer directory.

"make install" will make previewer and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

SEE ALSO

analyze, etherdump, network_capture, packet_print, playback, stream, top_con, transcript, warn_sort

NAME

report – another NSM reporting tool (SunOS 4.1.1)

SYNOPSIS

report [-index] [-detailed] [-no_svc_break] [-n <num>] [-sort <method>] [-d <direction>] [-match <string>] [-match_file <file>] connection_log_file

DESCRIPTION

report is yet another NSM tool to report about information in a connection log file. Connections can be grouped by service (the default), sorted by time, warning value, or host address, displayed tersely (only connection index, warning value and host names), or displayed in detail (the same format used by previewer). In addition, the user can choose only to view certain connections which match some condition set by the user (e.g., only international connections).

With the proper options, report can generate the same output as previewer (see previewer(1)). Furthermore, since connections can be sorted by warning value, the connection log file does not have to be preprocessed by warn_sort (see warn_sort(1)). To generate the same output as previewer, use the following options:

```
% report -detailed -no_svc_break connection_file
```

When report generates a terse connection description, the format is:

```
<index> <warning_value> <src host> <dst host>
```

When report generates a detailed connection description (by using the -detailed option), the format is:

```
<index>     src: <src host address> <src host name>
            dst: <dst host address> <dst host name>
            svc: <service name>
            start: <time>                stop: <time>
            warning value: <number>
            strings from source computer:
                  <list of stings and counts>
            strings from destination computer:
                  <list of stings and counts>
```

A connection consists of two streams of data: one from the source host (which started the connection) to the destination host, and one from the destination host back to the source host. You can think of them as an input stream and an output stream (I/O). Thus, for telnets, remote shells, and remote logins, strings from the source host are strings the user typed in. Meanwhile, strings from the destination host are strings which were sent back to the intruder to be displaye on his/her screen.

CAVEATS

Since report has to find the internet name for every host observed on the network, this program may take several minutes to complete.

For many login shells, the exclamation mark, '!', must be preceeded with an escape character like the backslash, '\\'.

The file DB/address_filter.file must be present. This file allows report to determine which host of a connection is from the protected domain and which is outside the domain.

OPTIONS

-index

prints the index number on both of the first two lines of a connection when viewed in the detailed mode.

-detailed

prints more information about the connection (see above).

-no_svc_break

does not group connections by service. This can be used if you want to see the top 'n' most suspicious connections no matter what service they are associated with.

-n <num>

prints only <num> connections for a single service. For example, if <num> is 10, at most, only 10 connections will be printed for each network service. If the services are not grouped (the -no_svc_break option), then only the first <num> connections of the entire file are printed.

-sort <method>

sorts the connection (within a service) by either the starting time (time), the warning value (warn), or the address value of the first host (addr). The default value is warn.

-d <direction>

allows you to only see the connections which are going into your site from the outside (in), going from your site to the outside world (out), or connections in both directions (both). The default is both.

-match <string>

looks for the string <string> in both the host addresses and names. The matching can be positive (just a string), in which case only connections in which the <string> is a substring of one of the hosts involved will be printed. The match can be negative (by prefixing the string with the character '!'), in which case connections in which both hosts have the substring in their addresses (or names) will NOT be printed.

The format for <string> is as follows:

```
[!][^]substring[$]
```

If '!' is used, then the substring is treated as a suppression string. That is, if the string is a substring of both hosts, then the connection will not be printed. For example, if you do not want to see any connections between two UC Davis machines, you can use the substring "!ucdavis.edu"

If '^' is used, then the substring must align with the beginning of the host's address or name. For example, if you want to match all hosts which have the internet address 128.120.*.*, then you can simply use the substring "^128.120".

If '$' is used, then the substring must align with the ending of the host's address or name. For example, if you want to match all hosts from Finland, then you can use the substring ".fi$".

Multiple matches can be specified, but each one must be preceded with the -match. The match is NOT case sensitive, so the substring ".edu" will match the host "JAWS.UCDAVIS.EDU".

-match_file <file>

directs report to use the strings in the specified file as match parameters. This is useful if you need to match/suppress several strings, or do the same match everytime. For example, if you want to only look at connections between your site and a site outside the country, you can specify a file which contains:

```
!.gov$
!.mil$
!.com$
!.edu$
!.org$
!.net$
```

If you want to also view connections between your site and MIT as well, you could add the string:

```
.mit.edu$
```

Multilple match files can be specified.

USAGE

To start report, enter the command followed by the connection log file. For example,

```
% report connections.log
```

If you are executing the program from the NSM/analysis directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/report connections.log
```

Since the output can be quite long, you may want to pipe the result through more or redirect it to a file.

FILES

The following files must be in the current working directory:

```
DB/address_filter.file
```

MAKEFILE

To make report, just type "make" at the command line in the NSM/src/Report directory. It defaults to "make all," and places the executable program "report" in the NSM/src/Report directory.

"make install" will make report and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

SEE ALSO

previewer(1), analyze(1), warn_sort(1)

NAME

run_install – NSM password installation program (SunOS 4.1.1)

SYNOPSIS

run_install

DESCRIPTION

run_install must be run before any of the other NSM tools will work properly. run_install will place a password file, .passwd, in your local directory. Once this file is in place, all other tools will be able to execute.

To use run_install, change directories into the NSM/analysis directory (You should execute all your NSM tools from this directory). Start run_install by typing

```
% ../bin/run_install
```

or if you have the NSM/bin directory in your path, simply type

```
% run_install
```

You will be provided with a location to write or call for a key as well as a special ID number. Contact one of the key distributors and supply them with your "special ID." The key distributor will then provide you with a two number password. Enter these two numbers, with a space between them, at the prompt.

If the password is entered correctly, run_install will report that your password has been successfully installed. If a number has been mistyped (by either you or the key distributor), you will be given two more chances to enter the correct number.

CAVEATS

run_install creates a password which is valid for only a single machine. If you wish to use the NSM tool from other machines, you should (1) have a separate analysis directory for each CPU (all other directories can be shared), and (2) execute run_install on each CPU from the appropriate analysis directory.

NAME

stream – NSM stream generator (SunOS 4.1.1)

SYNOPSIS

stream [-d dir] connection_log index

DESCRIPTION

stream is used to generate two network packet data files: one representing the stream from the initiating host, and one representing the stream from the destination host. These two files can then be displayed with the packet_print and playback programs.

connection_log is the name of the local connection file, created by analyze, containing the connection of interest. index is the number of the connection of interest.

OPTIONS

-d <dir>

use the directory "dir" as the source of the network packet data files. The default data directory is listed in config.file.

USAGE

stream must be started in a directory containing a configuration file, config.file.

To start stream, enter the command and the required arguments. For example, to generate stream files for the 38th connection in the file connection.log, enter:

```
% stream connection.log 38
```

If you are executing the code from the "analysis" directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/stream connection.log 38
```

If you have another directory containing the the network packet data files, for example ../tmp2, you can use the -d option:

```
% ../bin/stream -d ../tmp2 connection.log 38
```

FILES

The config.file must be in the current working directory.

stream generates two files of the form

```
(1) connection_log.index.stream.init
(2) connection_log.index.stream.dest.
```

MAKEFILE

To make stream, just type "make" at the command line in the NSM/src/Stream directory. It defaults to "make all," and places the executable program "stream" in the NSM/src/Stream directory.

"make install" will make stream and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

SEE ALSO

etherdump, network_capture, playback, transcript

NAME

top_con – NSM transcript shell script generator (SunOS 4.1.1)

SYNOPSIS

top_con [-d dir] [-n] connection_file output_file connection_num

DESCRIPTION

top_con generates a shell script file to execute "transcript" for a number of connections. connection_file is a (usually sorted) connection log file; output_file will be the file name of the resulting shell script; and connection_num is the number of connections for which you want to generate transcripts.

top_con simply generates a "transcript" command for each of the first connection_num connections in the connection_file. The shell script, output_file, must be executed from the command line (or within another shell script) to generate the actual transcript files.

OPTIONS

-n

This option is passed to each call of transcript.

In transcript, it generates a carriage return when the "new line" byte is seen. This option is often used when generating transcript files for a "remote shell" connection.

-d <dir>

This option is passed to each call of transcript. In transcript, the directory "dir" is used as the source of the network packet data files. The default data directory is listed in config.file.

USAGE

To run top_con, simply enter the command with the required arguments. For example, to generate a shell script for the top ten most intrusive looking connections in the sorted connection log file named warn91-12-20, enter the command:

```
% top_con warn91-12-20 trans.generator 10
```

If you are executing the code from the NSM/analysis directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/top_con warn91-12-20 trans.generator 10
```

The result will be a shell script file named trans.generator. To execute this shell script, simply enter its name at the command line:

```
% trans.generator
```

MAKEFILE

To make top_con, just type "make" at the command line in the NSM/src/Top_con directory. It defaults to "make all," and places the executable program "top_con" in the NSM/src/Top_con directory.

"make install" will make top_con and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

SEE ALSO

analyze, etherdump, network_capture, packet_print, playback, stream, transcript, warn_sort

NAME

transcript – NSM transcript printer (SunOS 4.1.1)

SYNOPSIS

transcript [-n] [-d dir] [-no_sub] [-sub <char>] [-no_filter] [-errors] [-p_depth <depth>] [-name_width <width>] connection_file index

DESCRIPTION

transcript generates script files (similar to those generated with the UNIX script command) for the two streams (input and output) of a connection. The file names for the two script files generated for a connection have the formats:

```
con_file.index.dest
con_file.index.init
```

where con_file is the name of the connection file passed to the program, and index is the number of the connection in the connections file for which the script files are generated.

The one exception is for transcripts for UDP-based NFS sessions. The results from the server (which would be the "dest" stream) are not directly displayed; this information will be used in the display of the clients data (the "init" stream). The two major results from a server is the file handle for a requested file name and the error results for a requested operation. The returned file handle will be used to display future operation requests--the requests are done on the file handle, we map the file handle to a name. And the error results are printed in the "init" file so it can be easily understood which operation the error is associated with.

OPTIONS

-n

generates a carriage return when the "new line" byte is seen. This option is often used when generating transcript files for a "remote shell" connection.

-d <dir>

use the directory "dir" as the source of the network packet data files. The default data directory is listed in config.file.

-no_sub

Currently, missing bytes in a TCP/IP connections are substituted with the letter 'X'. This option will prevent any substitution from taking place.

-sub <char>

Currently, missing bytes in a TCP/IP connections are substituted with the letter 'X'. This option will change the substitute character to the supplied character. (e.g., "-sub Y" will use 'Y' as the new substitute letter.

-no_filter

Turn of the telnet protocol parser/filter. When performing a transcript on a telnet connection, the default mode is to remove the data sent between the client and server (this information is called the negotiations); the information is displayed at the end of the transcript.

-errors

For transcripts of NFS sessions, add the error code associated with each request.

-p_depth <depth>

For transcript of NFS sessions, change the number of parents displayed for a file name. For example, if an operation is performed on a file called .rhosts, a p_depth of two would add two parents perhaps "home" and "heberlei" resulting in home/heberlei/.rhosts being displayed.

-name_width <width>

For transcript of NFS sessions, change the minimum amount of characters reserved for the name of file being operated on. The default widht is 18. By providing a fixed width for file names, we can get the error codes (which follow the file names) to line up in a nice column.

USAGE

transcript must be started in a directory containing a configuration file, config.file.

To start transcript, enter the command and the required arguments. For example, to generate script files for the 38th connection in the connection file connection.log, enter:

```
% transcript connection.log 38
```

If you are executing the code from the "analysis" directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/transcript connection.log 38
```

If you have another directory containing the the network packet data files, for example ../tmp2, you can use the -d option:

```
% ../bin/transcript -d ../tmp2 connection.log 38
```

FILES

The config.file must be in the current working directory.

MAKEFILE

To make transcript, just type "make" at the command line in the NSM/src/Transcript directory. It defaults to "make all," and places the executable program "transcript" in the NSM/src/Transcript directory.

"make install" will make transcript and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

A number of problems can occur when performing transcripts on NFS sessions. First, all operations are performed on file handles, 32 bytes of opaque data, not names. We try to map the file handle to a name, but we may not alway be successful. The result will be no name written at the end of the operation. Second, error results for an NFS option may not always be displayed. This occurs if we miss the associated packet. TCP-based RPC sessions (e.g., for portmapper or NFS) are currently not supported.

SEE ALSO

etherdump, network_capture, stream, playback.

NAME

warn_sort – NSM connection log sorter (SunOS 4.1.1)

SYNOPSIS

warn_sort src_file dst_file

DESCRIPTION

warn_sort takes as input a connection log file and generates a second connection log file containing the connection records sorted by warning value. The connections with the greatest warning values will be at the front of the new connection log file.

USAGE

To run warn_sort, simply enter the command with the required arguments. For example, to sort the file connections.log and place the sorted connection records in a file called warn91-12-20, enter the command:

```
% warn_sort connections.log warn91-12-20
```

If you are executing the code from the NSM/analysis directory and the NSM/bin directory is not in your PATH, enter:

```
% ../bin/warn_sort connections.log warn91-12-20
```

MAKEFILE

To make warn_sort, just type "make" at the command line in the NSM/src/Warn_sort directory. It defaults to "make all," and places the executable program "warn_sort" in the NSM/src/Warn_sort directory.

"make install" will make warn_sort and place a copy in NSM/bin.

"make clean" will remove the executable files and *.o files.

Typing "make install" from the NSM/src directory will make all NSM executables and install them in NSM/bin.

BUGS

SEE ALSO

analyze, etherdump, network_capture, packet_print, playback, stream, top_con, transcript

UNIX Manual Pages
File Formats

This section the presents manual pages (often referred to as “man pages”) for the various file formats used in the NSM software distribution. These manual pages are also available in an on-line form for UNIX computer systems.

NAME

address_filter.file – NSM address filter for capture (SunOS 4.1.1)

DESCRIPTION

The file address_filter.file, used by the program capture (see capture(1)), defines the set of host addresses you are interested in protecting. Any traffic flowing between one of the protected hosts and an unprotected host via a specified service (see service_filter.file(2)) will be captured for later analysis.

This file should be in the NSM directory NSM/analysis/DB.

Unfortunately, at this time, no comments are allowed in this file.

The file contains a list off address which can take one of six different formats. First, the address can be a local host name (e.g., toadflax). Second, the address can specify the full internet name (e.g., toadflax.cs.ucdavis.edu). Third, the addess can specify the full internet address (e.g., 128.120.56.188). Fourth, the address can be a class A address. This address must end with a period (e.g., 128.). All hosts with addresses 128.*.*.* will be considered protected. Fifth, the address can be a class B address. Once again, this address must end with a period (e.g., 128.120.). All hosts with the address 128.120.*.* will be considered protected. And sixth, the address can be a class C address ending in a period (e.g., 128.120.56.). All hosts with the address 128.120.56.* will be considered protected.

As an example, supposed you want to protect three sensitive hosts named lime, lemon, and grape. Then the address_filter.file would look like:

```
lime
lemon
grape
```

Now suppose you want to protect your entire department, which is has its own class C network (128.120.56.*), and two computers from another department called salt.other.UCL.edu and pepper.other.UCL.edu. The address_filter.file could then look like:

```
128.120.56.
salt.other.UCL.edu
pepper.other.UCL.edu
```

NAME

config.file – NSM configuration file (SunOS 4.1.1)

DESCRIPTION

The file config.file contains information which most of the NSM tools need in order to operate. Early in the NSM development, this was the only resource file used by the NSM tools. To this day, most of the information in this file is used to specify the filter for the original network traffic capture program: network_capture. Most NSM tools use only the last two lines of config.file.

The file config.file must be in the current working directory when one of the NSM tools is started. The default location of this file is the NSM/analysis directory.

The format of the file is:

```
comment     num_of_class_B_nets
[comment    address]*
comment     num_of_dial_in_ports
[comment    address]*
comment     file_name
comment     number
```

Each comment has to be a single string containing no white spaces (returns, spaces, or tabs). The lines enclosed in brackets and followed by a star indicate that the line may occur zero, one, or more times depending on the number specified on the line before.

The first line specifies the number of class B networks at the local site (see network_capture(1) for more information). For example, a large organization may have two class B networks to protect, so they will want to specify the number two here. An example line is:

```
#num_of_local_class_B_nets    2
```

The text #num_of_local_class_B_nets is the comment field, and the number 2 is the value.

For the number of class B networks specified, there must be a line specifying each of these addresses. Each line has the the format of a comment followed by an address. The address is of the Internet format A.B.C.D, where A, B, C, and D are numbers. Because we are specifying class B addresses, the last two numbers will be zeros. An example line is:

```
#class_B_network  128.120.0.0
```

After the the class B networks have been specified, the number of dial-in ports need to be specified. An example line is:

```
#num_of_dial-in_ports   3
```

For the number of dial-in ports specified, there must be a line specifying the Internet address of each dial-in port. An example line is:

```
#dial-in_port           128.120.57.201
```

The next line specifies the directory where the network traffic data files should be saved and their root file name. The line format in once again a comment followed by the value. The value in this case actually contains two pieces of information: the directory name, and the root file name. An example line is:

```
#data_file_name         ../tmp/log
```

The value ../tmp/log specifies that the directory where the data files will be kept is ../tmp. Furthermore, the files will be of the format of logYYMMDDHH; where YYMMDDHH specify the time (year, month, day, and hour) that the data file was created. For example, if one of the network traffic capture programs is started at 1:00pm on March 21, 1992, the first data file created in the directory ../tmp will be called log92032113.

Finally, the last line of the file specifies how often a new data file should be created. There are two main reasons to switch data files on a regular basis: 1) if a single file becomes two large, moving it (e.g. via magnetic media) may become difficult if the single file won't fit on the media, and 2) for one of the NSM tools to find data associated with a particular connection, that tool must begin at the beginning of a file an move sequentally through the file; therefore, limiting the size of the files makes for faster searches. The value field specifies the number of minutes between file times. An example line is:

```
#minutes_between_files  60
```

An example config.file is:

```
#num_of_local_class_B_nets    2
#class_B_network              128.120.0.0
#class_B_network              128.121.0.0
#num_of_dial-in_ports         3
#dial-in_port                 128.120.57.201
#dial-in_port                 128.120.57.202
#dial-in_port                 128.120.57.203
#data_file_name               ../tmp/log
#minutes_between_file         60
```

Since most users will use the program capture as opposed to the older program network_capture, most of the information in config.file, with the exception of the last two lines, will not be used. Therefore, the default config.file shipped with the NSM is

```
#num_of_local_class_B_nets    0
#num_of_dial-in_ports         0
#data_file_name               ../tmp/log
#minutes_between_file         60
```

NAME

exceptions.file – NSM exceptions for network traffic capture (SunOS 4.1.1)

DESCRIPTION

The file exceptions.file is used by the program capture (see capture(1)) to allow certain traffic between protected and unprotected hosts to pass through without being captured.

The following scenario helps explain when and how to use the exceptions.file.

Suppose a single host (called mercury) is used to back-up your protected hosts as well as a number of unprotected hosts. Furthermore, the back-up procedure uses the remote shell service which you are concerned about and have defined in your address_filter.file (see address_filter.file(5). Now if mercury is defined as a protected host, all the back-ups via remote shells to unprotected hosts will be captured by the program capture and will quickly fill up disk space. Similarly, if mercury is defined as an unprotected host, all back-ups of the protected hosts will be captured.

The solution is to say, "I want to capture all remote shell traffic between protected and unprotected hosts EXCEPT when one of those hosts is mercury." To implement this requirement, we use the exception file.

The the exception file is a list of lines of the format:

```
host1 host2 protocol    port
```

The field host1 is the name (local internet name (e.g., toadflax), full internet name (e.g., toadflax.cs.ucdavis.edu), or internet address (e.g., 128.120.56.188)) of the host we are interested in.

The field host2 is the name of a second host. The exception will be for traffic between host1 and host2. The field host2 can be a wild card, '*', and the exception will be traffic bewteen host1 and all other hosts. [note: as of the writing of this man page, only wild cards are accepted in the host2 field]

The next two fields specify the network service. The protocol field indicated whether the exception is for a TCP/IP service (in which case the value "TCP" is used) or a UDP/IP service (in which case the value "UDP" is used). The port field is the port number used by the service. For example, the remote shell service uses port number 514.

Therefore, to solve our dilemma, we add the line:

```
mercury * TCP 514
```

NAME

service_filter.file – NSM service filter (SunOS 4.1.1)

DESCRIPTION

The file service_filter.file, used by the program capture (see capture(1)), defines the set of network services you are concerned about and want to analyze. Any traffic flowing between one of the protected hosts and an unprotected host (see address_filter.file(5)) via one of these specified service will be captured for later analysis.

This file should be in the NSM directory NSM/analysis/DB.

Unfortunately, at this time, comments are only allowed at the end of a line.

The file contains a list of network services in a similar format as that of the file /etc/services:

```
service-name      port/protocol     [comment]
```

Currently, only the protocols TCP/IP and UDP/IP are recognized by the NSM tools. To specify a TCP/IP service, the protocol field should be the string "tcp". To specify a UDP/IP service, the protocol field should be the string "udp".

For example, suppose you want to protect your hosts agains the network services telnet, rlogin, FTP, and TFTP. Your address_filter.file would then look like:

```
telnet          23/tcp
login           513/tcp   # this is the UNIX rlogin
ftp             21/tcp    # only control info, no data
tftp            69/udp
```

NAME

sets.file – NSM sets file for DIDS_lan_mon and X_nsm_kernel (SunOS 4.1.1)

DESCRIPTION

The file sets.file, used by the programs DIDS_lan_mon and X_nsm_kernel, specifies what network activity to monitor by way of SET commands.

The LAN monitor reads in an initial list of SET commands from the file DB/sets.file at start-up. The SET commands (at least from a user setting up a sets.file point of view) falls into three catagories: binding names to objects, setting which objects (by using their name) to monitor, and setting strings to monitored in the connections.

```
BINDING NAMES TO OBJECTS
      binding a name to an Internet address
      binding a service name to a protocol and
        port number
SETTING OBJECTS TO MONITOR
      turning the monitor on a host
      turning the monitor on a service
SETTING STRINGS TO MONITOR
      setting strings to be searched for in the network
        connections
```

The following is a summary of the syntax for these operations:

```
SET NAME ON HOST <host-name> ADDR <inet-addr>
SET NAME ON SERVICE <name> PROTOCOL <proto> PORT <port>
SET HOST ON NAME <host-name>
SET SERVICE ON NAME <service-name> PROTOCOL <proto>
SET STRING <string> ON  SERVICE <name> PROTOCOL <proto> DIR <dir>
```

Where:

```
<inet-addr> is the standard internet address format A.B.C.D
<proto> is the transport protocol used by the service (TCP or
UDP)
<name> is name of service (e.g., telnet)
<string> a quotes string.  (e.g., "Login incorrect")
<dir> is a stream direction in a connection (INPUT ot OUTPUT)
```

`SET NAME ON HOST <host-name> ADDR <inet-addr>`

Binds the name <host-name> to the internet address <inet-addr>. Whenever future SET commands are sent regarding this host, only the host name is used. The reason this command is done is because people prefer host names, but the LAN monitor uses internet addresses.

`SET NAME ON SERVICE <name> PROTOCOL <proto> PORT <port>`

Binds the (<name>,<protocol>) pair to the port number <port>. For example, the service "telnet" is usually a TCP protocol which uses the reserved port number 23. However, telnet can also be implemented over UDP, so the name "telnet" can be associated with netork services over two different protocols.

`SET HOST ON NAME <host-name>`
Starts monitoring the host refered to by <host-name>. <host-name> must be already bound to an Internet address in a previous SET command.

`SET SERVICE ON NAME <service-name> PROTOCOL <proto>`
Starts monitoring the service and protocol referenced by <service-name> and <proto>. The name and protocol must already be bound to a port by a previous SET command.

`SET STRING <string> ON SERVICE <name> PROTOCOL <proto> DIR <dir>`
Looks for the string <string> in the network connections (defined by the <name> and <proto> in a previous SETS command), in the <dir> data stream.

EXAMPLE

The following is a sets.file for monitoring two hosts (ararat and erebus), two network services (telnet and rlogin), and looking for three strings ("daemon", "Login incorrect", and "Last login:") in the output data streams (what the user/intruder sees on their screen) of the two network services

```
SET NAME ON HOST ararat ADDR 128.120.56.193
SET NAME ON HOST erebus ADDR 128.120.56.195
SET NAME ON SERVICE telnet PROTOCOL TCP PORT 23
SET NAME ON SERVICE rlogin PROTOCOL TCP PORT 513
SET HOST ON NAME ararat
SET HOST ON NAME erebus
SET SERVICE ON NAME telnet PROTOCOL TCP
SET SERVICE ON NAME rlogin PROTOCOL TCP
SET STRING "Login incorrect" ON SERVICE rlogin PROTOCOL TCP DIR OUTPUT
SET STRING "Login incorrect" ON SERVICE telnet PROTOCOL TCP DIR OUTPUT
SET STRING "Last login:" ON SERVICE rlogin PROTOCOL TCP DIR OUTPUT
SET STRING "Last login:" ON SERVICE telnet PROTOCOL TCP DIR OUTPUT
SET STRING "daemon:" ON SERVICE rlogin PROTOCOL TCP DIR OUTPUT
SET STRING "daemon:" ON SERVICE telnet PROTOCOL TCP DIR OUTPUT
```

NAME

strings.file – NSM strings file for analyze (SunOS 4.1.1)

DESCRIPTION

The file strings.file, used by the program analyze (see analyze(1)), defines the set of strings we want to look for in the recorded network traffic. Each string will be searched for in both the input and output streams of all services. However, if the string is prefixed with "NFS: " or "RPC: ", then the rest of the line specifies a pattern specific to the NFS and RPC analysis procedures. Matches of the NFS or RPC patterns will be reported just like a normal string match.

The RPC patterns are:

```
RPC: <program> <procedure>
RPC: 16_bit_attack
```

The "RPC: <program> <procedure>" will count the number of time the remote procedure call matching the program <program> and the procedure <procedure> were seen. The <program> and <procedure> can be either a number or the recognized names are provided below.

The "RPC: 16_bit_attack" will detect when a UID is used in the request which is a non-zero UID, but the lower 16 bits are set to zero. This attack works on systems which use only 16-bits to recognize the UID field in its file system.

The NFS patterns are:

```
NFS: <filename>
NFS: STATUS <nfs_reply_stat>
```

The "NFS: <filename>" will count the number of times a file handle to a file with the name <filename> was requested. This file name can also be the name of a directory.

The "NFS: STATUS <nfs_reply_stat>" will count the number of time the error code <nfs_reply_stat> was returned as the result of a file operation. <nfs_reply_stat> can be either a number or a name. The acceptable names are provided below.

This file should be in the NSM directory NSM/analysis/DB.

The file consists of a list of lines. Each lines specifies a string to look for. DO NOT leave any empty lines at the beginning, middle, or end of the file.

An example strings.file is:

```
login: guest
Login incorrect
daemon:
passwd
```

```
login: root
Permission denied
CWD ~ROOT
.rhosts
hosts.equiv
RPC: 16_bit_attack
RPC: NFS RFS_WRITE
RPC: PORTMAPPER PMAPPROC_CALLIT
NFS: .rhosts
NFS: STATUS NFSERR_STALE
PROGRAM NAMES
```

For the pattern "RPC: <program> <procedure>", acceptable program names include:

```
NFS
PORTMAPPER
```

PROCEDURE NAMES

For the pattern "RPC: <program> <procedure>", acceptable procedure names include:

```
RFS_NULL
RFS_GETATTR
RFS_SETATTR
RFS_ROOT
RFS_LOOKUP
RFS_READLINK
RFS_READ
RFS_WRITECACHE
RFS_WRITE
RFS_CREATE
RFS_REMOVE
RFS_RENAME
RFS_LINK
RFS_SYMLINK
RFS_MKDIR
RFS_RMDIR
RFS_READDIR
RFS_STATFS
RFS_NPROC
PMAPPROC_NULL
PMAPPROC_SET
PMAPPROC_UNSET
PMAPPROC_GETPORT
PMAPPROC_DUMP
PMAPPROC_CALLIT
```

STATUS NAMES

For the pattern "NFS: STATUS <nfs_reply_stat>", acceptable status names include:

```
NFSERR_PERM
NFSERR_NOENT
NFSERR_IO
```

```
NFSERR_NXIO
NFSERR_ACCES
NFSERR_EXIST
NFSERR_NODEV
NFSERR_NOTDIR
NFSERR_ISDIR
NFSERR_FBIG
NFSERR_NOSPC
NFSERR_ROFS
NFSERR_NAMETOOLONG
NFSERR_NOTEMPTY
NFSERR_DQUOT
NFSERR_STALE
```