Appendix A

Network Security Monitor

a brief description

6 June 1991

L. Todd Heberlein

Installation

To install NSM from the tar file, enter the command

tar xvf nsm.tar

If the NSM is distributed on tape, enter the command

tar xv

Either of the commands will produce a new directory called *NSM* in your local directory. Under this directory are four more directories: *analysis*, *bin*, *src*, and *tmp*. If the computer you are using is a Sun-4 architecture, the binary programs should be fine. However, if you are running on a Sun-3, or if you simply want to recompile the programs, you will have to enter each of the program directories under *NSM/src* and remake the programs. To remake a program for a Sun-4, type the command

make

If you want to run on a Sun-3 architecture, modify the *makefile* by changing the -DSUN4 option in the CFLAGS to the -DSUN3 option.

Note: the program *network_capture* will not run on the computer unless the Networking Tools and Programs software installation option was installed (do a man on *etherfind* for more information). Basically, if *etherfind* works on your computer, so will *network_capture*.

The configuration file, *NSM/analysis/config.file*, will have to be modified to reflect your site's internet addresses and dail-up ports. For more information on this file, see the section File Descriptions.

File Descriptions

This section discusses the files used by the NSM. At the end of the appendix is a figure providing a diagram of the file structure.

NSM

(type-directory): *NSM* is the root directory for the NSM tools. Underneath the *NSM* are four directories called *analysis*, *bin*, *src*, and *tmp*.

NSM/analysis

(type-directory): *analysis* is the directory from which the user will do all of his or her work. This directory contains the data files which direct the the various programs of the NSM to do their work. Results from analysis (e.g.. connection files, profile files, and transcript files) are stored here as well.

NSM/bin

(type-directory): *bin* is the directory for all the executable programs for the NSM. The directory contains the programs *analyze*, *network_capture*, *top_con*, *transcript*, and *warn_sort*. This directory is static and should not change unless one of the programs is recompiled.

NSM/src

(type-directory): *src* is the directory for the source code for all the executable programs in the *NSM/bin directory*. The source code for each executable program is in its own directory. The directory names are the same as the executables: *Analyze*, *Network_capture*, *Top_con*, *Transcript*, and *Warn_sort*.

NSM/tmp

(type-directory): *tmp* is the directory for the network traffic data files. These log files will be of the format *filename*YYMMDD.HH; where YY is the year, MM is the month, DD is the day of the month, and HH is the hour of the day. The choice for *filename* can be changed in the *config file*.

NSM/analysis/con_count file

(type-text data file): *con_count file* contains a single number representing the total number of connections analyzed by the *analyze* program. This number will be initially set to zero. If, for example, analyze was run on three days of which 4000, 4010, and 4020 connections were analyzed , then *con_count file* would contain the number 4000 after the first day, 8010 after the second day, and 12030 after the third day.

Example:

12030

NSM/analysis/config file

(type-text data file): *config file* is the most important data file in this directory, and almost every program of the NSM uses it. *config file* contains the information describing the local site's internet addresses (the class B networks for the site), the local dial-up ports, the place to store the data files (e.g.. NSM/tmp), and how often to switch data files (typically it is on every hour).

Example:

```
#num_of_local_class_B_nets      1
#class_B_net                    128.120.0.0
#num_of_local_gateways          4
#local_gateway                  128.120.2.251
#local_gateway                  128.120.2.253
#local_gateway                  128.120.2.254
#engr-dnet1                     128.120.59.29
#output_root_file_name          ../tmp/log
```

```
#minutes_between_files          20
```

Each line contains two pieces of information: a comment and a value. The comment has to be a single "word" (I read it in with a single scanf(...%s...)). The values in this file can be read as follows:

#num_of_local_class_B_nets = 1 - This number indicates how many class B networks are at the local site.

#class_B_net = 128.120.0.0 - There will be n lines of this form, where the number n is given in the previous line (in this case, 1). This value gives the internet address of the local class B network. In this case it is 128.120.*.*. Place zeros, where the stars are. Any packet which has a source or destination address will be considered a foreign packet.

#num_of_local_gateways = 4 - This number indicates how many local gateways or dial-ups there are. Any packet from/to these dial-up addresses is considered to a foreign packet.

#local_gateway = 128.120.2.251 - This is an internet number of a dial up

#local_gateway = 128.120.2.253 - This is an internet number of a dial up

#local_gateway = 128.120.2.254 - This is an internet number of a dial up

#engr-dnet1 = 128.120.59.29 - This is an internet number of a dial up

#output_root_file_name = ../tmp/log - This is the directory where the network traffic will be stored (../tmp) and the root file name for the data files (log).

#minutes_between_files = 20 - This indicates that the file name where traffic is stored will be changed every twenty minutes. It will be aligned on the hour.

NSM/analysis/connections.file

(type-text data file): *connections.file* is a log file of all the connections observed by the last run of the program *analyze*. The first line of the file specifies the number of strings searched for by *analyze*. The following lines are the strings searched for. After the strings are the actual logs of connections ordered by when they were closed (or terminated). Each log is on a single line, but each line typically wraps around three line on a normal 80 column display.

Example:

```
7
login: guest
Login incorrect
daemon:
passwd
login: root
Permission denied
CWD ~ROOT
 218 267389 8.944 5.778 10.000 10.000   128.120.2.251   128.120.57.60   6 25858
    23    telnet   Mon-Jun-03-18:12:03-1991   Mon-Jun-03-18:12:38-1991   35
    51     40     34     144 0-rec-1 1-rec-2
 199 267370 8.944 5.778 10.000 10.000   128.120.2.251   128.120.57.14   6 10498
    23    telnet   Mon-Jun-03-18:10:09-1991   Mon-Jun-03-18:10:36-1991   27
   126     93     71     278 0-rec-1 1-rec-5
 119 267290 8.944 5.778 10.000 10.000   128.120.2.251   128.120.57.67   6 11020
    23    telnet   Mon-Jun-03-17:59:48-1991   Mon-Jun-03-18:00:22-1991   34
    109     81     70     243 0-rec-3 1-rec-3
```

Although this file represents a connection file which has actually been sorted by waring value, the information in this data file is exactly the same as the original connections.file

The first line of the file contains a number indicating the number of strings which were searched for while processing this log file. In this case, seven strings were used. Following the number are the actual strings used. These strings are indexed from 0 to (n - 1) (in this case, 0 to 6). Therefore the string

login: guest

has the index number 0, and the string

CWD ~ROOT

has the index number 6.

Following the strings are the actual connection logs. The first connection log, although it is one line, is wrapped across three different text lines. The information in this first record can be interpreted as follows:

connection index for this particular processing = 218
connection index for all time = 267389
composite warning value = 8.944
attack model warning value = 5.778
anomaly detection warning value = 10.000
expert system warning value = 10.000
address of the host initiating the connection = 128.120.2.251
address of the host receiving the connection = 128.120.57.60
protocol of service (6 - TCP, 17 - UDP) = 6
port used by initiating host = 25858
port used by receiving host = 23
service name = telnet
start time of the connection = Mon-Jun-03-18:12:03-1991
ending time of the connection = Mon-Jun-03-18:12:38-1991
duration of the connection in seconds = 35
number of packets initiator sent = 51
number of bytes initiator sent = 40
number of packets receiver sent = 34
number of bytes receiver sent = 144
string match: (0-rec-1) [This states that the string index 0, "login: guest," was sent by the receiving host (rec -> receiving host, init -> initiator host), and it was matched 1 time]
string match: (1-rec-2) [This states that the string index 1, "Login incorrect," was sent by the receiving host 2 times]

NSM/analysis/hostfile

(type-text data file): *hostfile* contains a list of internet addresses and security level numbers. The data file is usually empty; however, it is possible to specify the security levels (between 0 and 10) of individual hosts. A telnet from a low security machine to a high security machine will be considered more suspicious than a telnet from a high security machine to a low security machine. This is used by the program *analyze* to calculate the attack model portion of the warning value. This file can be empty. Any local host not seen in this file will be given the default security level of 3, and any foreign host not seen in this file will be given the default security level of 1.

Example:

```
128.120.57.1      5
128.120.57.117    8
128.120.57.118    8
128.120.57.119    9
128.120.57.120    5
128.120.57.121    5
```

```
128.120.57.122      5
128.120.57.130      5
128.120.57.131      8
128.120.57.132      8
```

In this example the host 128.120.57.1 is assigned a security level of 5, and the host with the internet address 128.120.57.117 is assigned a security level of 8.

NSM/analysis/profile.file

(type-text data file): *profile.file* contains a record of observed past network activity. It is used by the program *analyze* to calculate the anomalousness of an observed connection. *profile.file* will initially be empty.

Example:

```
128.115.1.1  128.120.57.1  6  25  0  32
130.86.71.1  128.120.57.1  6  79  0  1
130.86.71.2  128.120.57.20  6  513  1  255
128.228.1.2  128.120.57.20  6  25  1  255
128.118.56.2  128.120.57.20  6  25  0  111
137.39.1.2  128.120.57.20  6  25  0  48
130.86.71.2  128.120.57.20  6  514  1  136
129.245.1.2  128.120.57.20  6  25  0  33
129.10.1.2  128.120.57.20  6  25  0  67
```

The first line of the profile represents a data path from the host 128.115.1.1 to 128.120.57.1 by the TCP/IP protocol (6 = TCP, 17 = UDP) and the service port 25 (electronic mail). The 0 indicates that there were no connections on this data path on the most recent day. The last number, 32, is translated to a bit list (00100000). A one indicates that a connection was observed on this data path during that day. The most recent day is represented by the most significant bit (left most), and the furthest day remembered is represented by the least significant bit (right most)

NSM/analysis/strings.file

(type-text data file): *strings.file* is perhaps the most useful file for detecting intrusions. *strings.file* contains a list of strings that the program *analyze* will search for in the network connections. For example "login: guest" can be searched for in the network connections. If it is matched, it usually implies that someone tried to login as guest. A number of stings are searched for (e.g.. "Login incorrect"), and the results of this search is used by *analyze* to calculate the expert system component of the warning value; however, the search for other strings is done by simply placing a new string at the end of the list.

Example:

```
login: guest
Login incorrect
daemon:
passwd
login: root
Permission denied
CWD ~ROOT
```

To find the results of a string match, examine the content of the file *connections.file*. For example to find all occurrences of the string "Permission denied," string index 5, simply grep for a occurrences of the strings "5-rec" and "5-rec." For example:

egrep "5-recl5-init" connections.file

NSM/analysis/tcp.file

(type-text data file): *tcp.file* contains a list of known tcp service ports and names. Also associated with each tcp service are two numbers representing the service's capability and authentication requirement (each number has a value between 0 and 10). For example, telnet has strong capabilities and strong authentication requirements (a password). The capability and authentication requirement values are used by *analyze* to determine the expert system component of the warning value.

Example:

7	echo	1	1
9	discard	1	1
11	systat	1	1
13	daytime	1	1
15	netstat	1	1
20	ftp-data	7	7
21	ftp	7	3
23	telnet	10	3
25	smtp	4	3
37	time	1	1
43	whois	1	1
53	domain	1	1
101	hostnames	1	1
111	sunrpc	8	3
77	rje	1	1
79	finger	4	1

In this example, the first service occupies port number 7, its name is "echo," its capabilities are listed at a strength of one, and its authentication is also listed at a strength of one.

NSM/analysis/udp.file

(type-text data file): *udp.file* contains a list of known udp service ports and names. Also associated with each udp service are two numbers representing the service's capability and authentication requirement (each number has a value between 0 and 10). For example, tftp has strong capabilities, but it has very weak authentication requirements (none). The capability and authentication requirement values are used by *analyze* to determine the expert system component of the warning value.

Example:

53	domain	1	1
111	sunrpc	7	7
69	tftp	8	1
123	ntp	1	1
512	biff	1	1
513	who	1	1
514	syslog	1	1
517	talk	1	1

In this example, the first service occupies port number 53, its name is "domain," it has a capability strength of one, and it also has a authentication strength of one.

NSM/bin/analyze

(type-binary program): *analyze* is the most important of the NSM programs; *analyze* is the program which actually detects and analyzes the connections in the network

traffic. This program requires as input the starting year, month, day, and hour of interest in network traffic data files and the total number of hours to search for data. The output of this program is the data files *connections.file*, *profile.file*, and *con_count.file*. The previous *connections.file*, *profile.file*, and *con_count.file* will be overwritten. In addition to the input mentioned above, *analyze* uses the following data files: *con_count.file*, *config.file*, *host.file*, *profile.file*, *strings.file*, *tcp.file*, and *udp.file*.

Example:

../bin/analyze 91 6 3 0 24

This command will analyze the data starting at 1991, June 3rd, and the 0th hour (midnight), and it will analyze 24 hours worth of data.

NSM/bin/network_capture

(type-binary program): *network_capture* pulls the network traffic off the network and places the packets into the data files in the directory *NSM/tmp*. Only a specific portion of the network traffic is recorded-the traffic between local hosts and hosts off site and the traffic to and from local dial-ups. Currently no FTP data is stored. The program has no input. The output are the log files in the *tmp* directory. *network_capture* uses the information in *config.file* to determine which traffic to capture and where to store it.

Example:

../bin/network_capture

That is it.

NSM/bin/top_con

(type-binary program): *top_con* takes a file of connection logs and creates a script file which when executed will create transcript files for the first *n* connections. The format is "*top_con input_file output_file n*." *input_file* is the name of the connection log file (a file such as *connections.file*); *output_file* is the name of the new script file; *n* is the number of connections for which transcripts will be created.

Example:

../bin/top_con warn91-6-3 top-10 10

This will take a connection file called warn91-6-3 (same format as connections.file) as input, and it will create a shell script file called top-10 which will include the commands to generate transcript files of the first 10 connections in the file warn91-6-3. To execute this shell script, simply type "top-10."

NSM/bin/transcript

(type-binary program): *transcript* takes a connection file and a connection number as input, and it creates two transcript files (one for the input stream and one for the output stream) for the that connection. The names for these transcript files will be *connection_file_name.number*.init and *connection_file_name.number*.dest where *connection_file_name* is the name of the connection file supplied as input and *number* is the connection number. The file ending in init is the data sent by the host initializing the connection, and the file ending in dest is the data sent by the host to which the connection was made.

Example:

../bin/transcript warn91-6-3 199

This will take the connection file named warn91-6-3 and look for connection index number 199 in it. When it finds this particular connection log, it will generate two transcript files (warn91-6-4.199.init and warn91-6-3.199.dest) representing the

data flowing from the host which initialized the connection and the data from the destination host.

NSM/bin/warn_sort
(type-binary program): *warn_sort* takes as input an unsorted connection file and outputs a sorted (by warning value) connection file.

Example:

../bin/warn_sort connections.file warn91-6-3

This example will take the connection log file called connections.file and generate a connection log file sorted by warning value called warn91-6-3.

NSM/src/Analyze
(type-directory): *Analyze* is the directory for all the source code for the program *analyze*. To re-make the program *analyze*, simply type the command "make" from inside the *Analyze* directory.

NSM/src/Network_capture
(type-directory): *Network_capture* is the directory for all the source code for the program *network_capture*. To re-make the program *network_capture*, simply type the command "make" from inside the *Network_capture* directory.

NSM/src/Top_con
(type-directory): *Top_con* is the directory for all the source code for the program *top_con*. To re-make the program *top_con*, simply type the command "make" from inside the *Top_con* directory.

NSM/src/Transcript
(type-directory): *Transcript* is the directory for all the source code for the program *transcript*. To re-make the program *transcript*, simply type the command "make" from inside the *Transcript* directory.

NSM/src/Warn_sort
(type-directory): *Warn_sort* is the directory for all the source code for the program *warn_sort*. To re-make the program *warn_sort*, simply type the command "make" from inside the *Warn_sort* directory.

NSM/tmp/logYYMMDD.HH
(type-binary data file): logYYMMDD.HH is the general form for the binary data files for the network traffic. For example, log910530.00 is the network traffic for 1991 May 30 and the 0^{th} hour (e.g., midnight until 1:00 am). The name "log" can be changed by modifying the data file *analysis/config.file*.

Basic Operations

This section describes the simple operations needed to keep a moderate handle on network security at a particular site. I assume that a single person (or perhaps a small group of people) will be in charge with examining the network activity regularly for possibly intrusive data.

The first step is to start the collection of data. Assuming the config.file is set up correctly (see **Installation** and **File Descriptions**: analysis/config.file), the collection of data is started simply by typing the following command (as root) from the analysis directory:

../bin/network_capture

This command will start the collection of data. It should run continuously, and if desired, this process can be placed in the background. The CPU overhead of this process is barely noticeable; however, the disk usage can be expensive. Keep an eye on the disk space. Finally, I strongly recommend that the data be kept on a local hard disk; shipping the data across the network to another disk will reduce your network bandwidth.

The second step is to analyze the data. The following list of commands, which can be put into a shell script (e.g.. my_shell_script), are all that is needed to produce transcript files for the days ten most intrusive looking connections:

```
../bin/analyze 91 6 3 0 24
../bin/warn_sort connections.file warning.list
../bin/top_con warning.list top-10 10
top-10
```

The execution of these commands will produce ten pairs of transcript file of the form warning.list.###.init and warning.list.###.dest (### will be the connection number). A security officer need only verify whether these connections are legitimate or not by examining them with any word processor.

To modify the date to process new data, simply modify the first command. For example, to process the data the next day simply change the first line to:

../bin/analyze 91 6 4 0 24

The following two files are actual transcript files generated by the above method for a particular connection. The first file is the data sent by the destination host back to the hacker. The second file is the data sent by the hacker to the target computer.

First file:

```
TRANSCRIPT

For connection file:   warn91-6-3
and connection index: 218

Initiating host:   128.120.2.251
Destination host: 128.120.57.60
Service:          telnet

Start time:  Mon-Jun-03-18:12:03-1991
End time:    Mon-Jun-03-18:12:38-1991

Warning level: 8.944
words matched from initiating host:
words matched from destination host:
       Login incorrect    2
       login: guest    1
```

```
Data from destination host
--------------------------------------------
}}{{~}

SunOS UNIX (surya)


{~login: guest
Password:
Login incorrect
login: uucp
Password:
Login incorrect
```

Second file:

```
TRANSCRIPT

For connection file: warn91-6-3
and connection index: 218

Initiating host: 128.120.2.251
Destination host: 128.120.57.60
Service:         telnet

Start time: Mon-Jun-03-18:12:03-1991
End time:   Mon-Jun-03-18:12:38-1991

Warning level: 8.944
words matched from initiating host:
words matched from destination host:
        Login incorrect  2
        login: guest  1

Data from initiation host
-------------------------------------------
{}}{{guest
guest
uucp
```

In summary, the following operations are needed:

../bin/network_capture

my_shell_script

The dates in the shell script, my_shell_ script, will have to be changed, but that is the only modification which is needed. Since network_capture needs to be only started once, the only program which needs to be executed on a daily basis is my_shell_ script.

Advanced Operations

This section describes a few extra operations a security officer may wish to perform to provide extra security. First, if intrusions are suspected from a particular host, a security officer may want to find all connections from or to that host. This can be accomplished by simply grep-ing for the host's internet address in the connection file. This will produce all the connection records associated with that host. For example:

grep 141.225.1.2 connections.file

may produce the following results:

```
2340 267290 8.944 5.778 10.000 10.000   141.225.1.2   128.120.57.120   6 11020
    23       telnet    Mon-Jun-03-17:59:48-1991    Mon-Jun-03-18:00:22-1991      34
    109      81        70       243 0-rec-3 1-rec-3
2345 267295 8.944 5.778 10.000 10.000   141.225.1.2   128.120.57.121   6 1235
    23       telnet    Mon-Jun-03-18:01:00-1991    Mon-Jun-03-18:02:22-1991      82
    218      165       140      612 0-rec-1
```

This result indicates that there were two connections from 141.225.1.2: connection 2340 and connection 2345. The first connection matched the string "login: guest" three times and the string "Login incorrect" three times. The second connection matched the string "login: guest" once. To generate the transcript reports for these two connections, simply type:

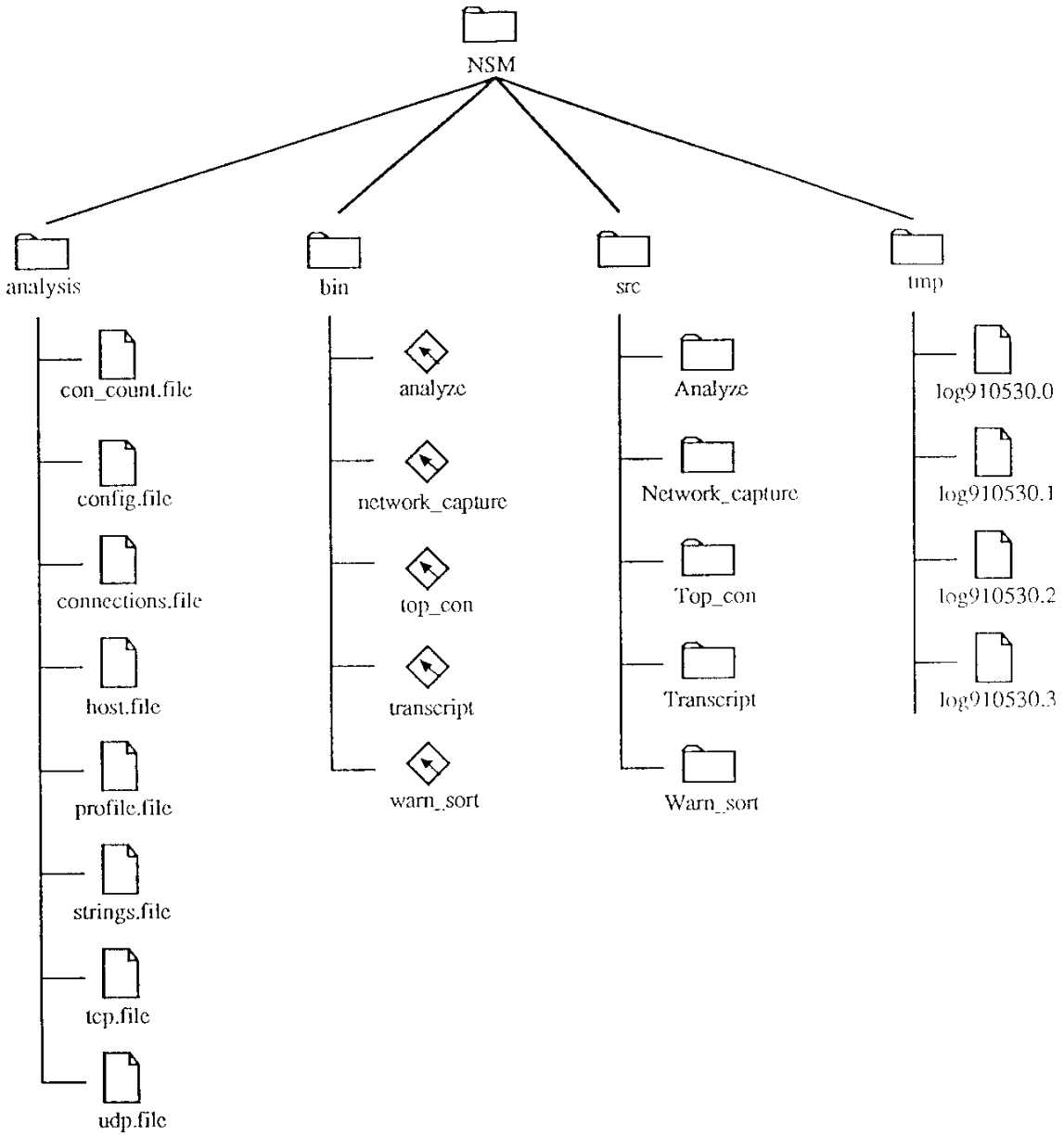
../bin/transcript connections.file 2340
../bin/transcript connections.file 2345

Another operation which might be useful is to search for a particular key word or string. For example, suppose the string "CLASSIFIED" is contained in all classified documents. To search for such a string in all network traffic, Add the string to the end of the *strings.file*, and then process the data with *analyze*. If this string is the 8th string in the file, all occurrences of it can be searched for with the command

egrep "7-init|7-rec" connections.file

(note that the strings are counted from 0 to (n-1). Thus the eigth string is actually indexed as string 7). If this string was matched in any of the connections, the connection records will be printed. To retrieve the actual transcript of the connections, use the ../bin/transcript command as in the above example. This is a useful exercise to determine what sensitive data is being shipped off site.

NSM Layout


NSM
analysis
bin
src
tmp
con_count.file
config.file
connections.file
host.file
profile.file
strings.file
tcp.file
udp.file
analyze
network_capture
top_con
transcript
warn_sort
Analyze
Network_capture
Top_con
Transcript
Warn_sort
log910530.0
log910530.1
log910530.2
log910530.3