AIIM

Association for Information and Image Management

1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

301/587-8202

MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.

1 of 1

UCRL-MA-116609

Network Intrusion Detector

NID User's Guide V 1.0

Robert Palasek, Editor

April 1994

Lawrence Livermore National Laboratory

NID has been developed jointly by Lawrence Livermore National Laboratory and the University of California, Davis

Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

Contents

Preface

Audience

The audience for this guide is a technical reader, for example a Unix system administrator, who is interested in learning how to use the *Network Intrusion Detector (NID).*

How This Guide Is Organized

This guide contains five chapters, an appendix, an index, and a comment form.

Chapter 1, Introduction: This chapter is an introduction to the NID application that briefly describes the various tools available to assist system managers and computer security officers in maintaining the security of their systems.

Chapter 2, Installation: This chapter contains instructions for installing the NID software onto your system.

Chapter 3, Usage: This chapter describes how NID is typically used and is aimed at the new NID user.

Chapter 4, Tools: This chapter contains a brief description of the NID tools.

Chapter 5 NID File Descriptions: This chapter describes the major NID files and directories.

Appendix A: Warning Notices.

Conventions Used in This Guide

The following conventions are used in this guide:

— **`bold terminal font`**

This font indicates text or code typed at the keyboard during an interactive session with the application.

— `terminal font`

This font indicates information displayed by the computer during interactive sessions. It is also used in code examples and text to indicate filenames, command names, and error names.

— *italic font*

This font is used in code examples and text to indicate user-specified parameters for insertion into programs or command lines. It is also used to introduce new terms or phrases the first time they are used in the text. Within narrative text, we also use italic font for the names of computer programs; this is similar to italicizing the titles of books.

— gray boxes

```
Examples and interactive dialogs with
NID are shown in gray  boxes.
```

— Return, Tab, Delete

Keyboard keys are given an initial capital letter. For example, "To run '`capture`', type **`capture`** and press the Return key."

The Version 1 NID codes were in large part developed at the University of California at Davis by Todd Heberlein and others, under a research and development contract from the Department of Energy through Lawrence Livermore National Laboratory's Computer Security Technology Center. The beta versions were known as NSM, the Network Security Monitor. This product is in no way related to the NSM computer security product recently available commercially from Network-1.

Todd Heberlein and Susan Izumi were contributors to earlier versions of this manual. I wish to thank L. Doug Brown for his help with parts of this version. Any errors that the reader might find are truly my responsibility.

Robert Palasek
April, 1994

Further Information

To obtain NID, or information about it, contact the NID Team Leader:

Robert Palasek, NID Project Team Leader
Computer Security Technology Center
Lawrence Livermore National Laboratory
Mail Stop L-303
P. O. Box 808
Livermore, CA 94551-9900

Telephone:	(510) 423-6224
e-mail:	*cstc@llnl.gov*

1

Introduction

The NID suite of software tools was developed to help detect and analyze intrusive behavior over networks.

NID combines and uses three techniques of intrusion detection: attack signature recognition, anomaly detection, and a vulnerability risk model. We have found from experience that the signature recognition component has been the most effective in detecting network based attacks.

The underlying assumption of NID is that there is a security domain that you are interested in protecting. NID monitors traffic that crosses the boundary of that domain, looking for signs of intrusion and abnormal activity.

The first component, *capture*, listens to the Ethernet broadcast network and captures IP traffic comprising the protocols specified by the operator. It saves this traffic in a temporary file space on disk for later, periodic analysis.

Typically once a day, the system manager or the network security officer runs the analysis program on the data that were captured over the previous monitoring epoch. The *analyze* routine organizes the data into sessions and then screens the sessions for signatures of intrusive activity, anomalous connections, and relative risk. The sessions are then scored according to these attributes; sessions having higher scores contain more indication of intrusive activity than do sessions with lower scores.

Once the *analyze* program has collated and scored the data, the security officer reviews its findings via a report generator named *report*. Typically, the report contains summaries of the many sessions of the past period's traffic. If the situation calls for it, the security officer can request a more detailed report that gives indications why a particular session has a high score.

Finally, if even further examination is warranted, the security officer can use the *transcript* utility to generate a detailed transcript of the session in question which he/she can then examine.

The chapter entitled "Tools", contains further description of these and other tools. Also, on-line synopses are given in the associated *manual (man)* pages in the `$NIDHOME/doc/man/man1` directory.

2

Installation

2.1. Environment

NID presently runs on a SPARC workstation under SunOS 4.1.2 or 4.1.3. (*This does not constitute an endorsement; see disclaimer.*) A typical instance has at least 0.25 gigabytes of free disk storage available for it, although it may be possible to get by with less.

Since NID is a security program and some of its data may be sensitive, we suggest that an inexpensive work station be dedicated to NID and no other applications run on it. Because we presently offer no means of transmitting this possibly sensitive data over the network, we recommend that the workstation be configured to run stand-alone, and that it be accessed only from the workstation's console.

Table 2.1 gives a rough estimate of acquisition cost for a typical stand-alone NID host.

ITEM	Approximate available cost, Spring '94
Inexpensive stand-alone work-station with 24 Mbyte memory, 15" monitor, 210 Mbyte disk.	$3400
1GB external disk for data.	$1200
Tape drive for backup.	$750

Table 2.1 Approximate range of hardware costs.

We also recommend that the security configuration of the node be tightened up; use of SPI, the Security Profile Inspector, is very helpful here.

Because the packet capture part of NID listens to the network in promiscuous mode, it must be run from a session of a user who is root. Running as root is not needed, however, by any other application in the NID toolbox. The system manager may find it advantageous to set up a group named `niduser` and to set permissions on all the NID files and directories such that the group `niduser` has access to them. In time constrained situations one may find it easier to run everything from the `root` user ID.

The configuration we use has most of its internet demons disabled; this means that the NID host cannot be accessed remotely, although it is possible to initiate Internet connections from the NID host to other nodes. In our stand-alone configuration running SunOS 4.1.2, with the Basic Security Module, BSM, and the Domestic Encryption Kit, we found that we had to allow the following demons to continue to run: `portmap` and `keyserv`. (In theory these should not be required for a stand-alone system; nonetheless, we could not get our system to run satisfactorily when we did disable them.)

We assume that Sun's "Domestic Encryption Kit" has been installed on the workstation. Alternately, DES decryption codes can be found on some FTP servers.

2.2. Installation

Set up a group **`niduser`**. Use whatever technique you usually use to set up groups on your system.

2.2.1 Become the root user and niduser

```
su root
newgrp niduser
```

2.2.2 Find a disk partition with as much space as possible, 100 Mbytes or more. It would also be good to choose the partition such that if it fills up, the system won't be impacted. Create a `nid` directory in that partition.

```
cd <mumble parent directory>
mkdir nid
chown root.niduser nid
chmod 770 nid

cd nid
ls -ldg .
drwxrwx---   2 root niduser 512 Mar 11
17:00   .
```

2.2.3 Copy the NID distribution file to the `nid` directory. This is done via an outbound `FTP` connection, or a `tar` from a tape drive, or a `bar` from a floppy disk drive, or however appropriate for the platform. The distribution file is named `NID_DIST.tar.des`.

2.2.4 Obtain the DES key from the NID support and distribution contact, and unpack the distribution set.

```
des -d NID_DIST.tar.des NID_DIST.tar

tar -xvf NID_DIST.tar
```

2.2.5 Set owner and group ids to those appropriate to the local system.

```
chown -R root.niduser NID_DIST
chmod -R 770 NID_DIST
```

2.2.6 For the remaining setup and use of NID, it will be convenient to have an environment variable `NIDHOME`. If you are using the C shell, it can be defined thusly:

```
cd <mumble parent directory>/nid/NID_DIST
setenv NIDHOME `pwd`
```

Also, it will be convenient to add a reference to your `MANPATH` for the NID on-line documentation:

```
setenv MANPATH $NIDHOME/doc:$MANPATH
```

At this time it is also appropriate to add similar statements to your `.login`, `.cshrc`, or your `.profile` file.

2.2.7 Edit the configuration files that came with the distribution to reflect the particular characteristics of your site. The following files need to be adapted to your site; further explanations are given below:

In the directory
`$NIDHOME/analysis/DB`

Configure the following files used by the *capture* program:

```
address_filter.file
service_filter.file
exceptions.file
```

Configure the following files used by the *analyze* program:

```
host.file
strings.file
```

The following files from the NID distribution can be left as they are.

```
tcp.file
udp.file
$NIDHOME/analysis/config.file
```

`address_filter.file` This file defines the security domain, a set of nodes and sub-nets such that traffic crossing the domain's boundary is captured.
Traffic between nodes within the security domain is not captured. Neither is traffic between any two nodes outside of the domain, even though this external traffic may traverse the monitored Ethernet.

You must edit this file to reflect the addresses of your site and also to remove the example lines. More details are given in the chapter entitled "NID File Descriptions" and also in the `man` pages.

`service_filter.file` This file defines the internet protocols that will be monitored and whose traffic is captured.

Edit this file to add or delete protocols which you are interested in monitoring. Initially, you may wish to leave it as it comes. See the chapter entitled "NID File Descriptions" and also the `man` pages.

`exceptions.file` This describes nodes outside of the security boundary whose traffic, such as print traffic and especially remote *backup traffic*, you especially do not wish to capture.

You may edit this file, at least to remove the example lines. See the chapter entitled "NID File Descriptions" and also the `man` pages.

`strings.file` This file describes strings associated with known attacks and whose presence suggests the possibility of an intrusion. Initially, you can leave the `strings.file` as it is initially configured in the distribution. Because some of the default strings are needed for the "expert system" component, we recommend not deleting any of the supplied strings, but rather to append to that set when needed.

`host.file` This file defines parameters used as part of the vulnerability risk model. Put the addresses of the hosts of your security domain and other hosts likely to have traffic with these in your `host.file`. Consider setting an initial value of 3 for the "security level" of the hosts in your domain. For those hosts outside your domain, consider either a 0 or a 3, depending on your level of trust of both the host and the path by which it communicates. Important or sensitive hosts may be given higher levels. See the chapter entitled "NID File Descriptions" and also the `man` pages.

`tcp.file` This file also defines parameters used as part of the vulnerability risk model. Leave the `tcp.file` as it is initially configured.

`udp.file` This file also defines parameters used as part of the vulnerability risk model. Leave the `udp.file` as it is initially configured.

config.file It is possible to have the raw data that the *capture* program captures written to a different directory or partition other than `$NIDHOME/tmp/`. Modify the third line of the `config.file` to do this. Most installations choose a "fat" partition for the NID installation and leave the `config.file` as it is initially configured. See the chapter entitled "NID File Descriptions" and also the `man` pages.

2.3. Before running NID for the first time

Before running any of the NID tools, you must go through an authentication procedure. Set your default directory to be the `analysis` directory, and execute the `run_install` program.

```
cd $NIDHOME/analysis

../bin/run_install
```

Follow the instructions presented by the `run_install` program.

Finally, start the `capture` program. Details are presented in the `man` pages and also in the chapter entitled "Basic Operations and Usage" in this manual.

3

Basic Operations and Usage

This chapter describes the simple operations needed to keep a moderate handle on network security at a particular site. Our assumption is that a single person (or perhaps a small group of people) will regularly examine the network activity for intrusive data.

Training time may require up to three hours per day for the first week. As the user gains experience, the daily load should decrease.

Note

Before starting, it is important to note that execution of all the NID tools should be from the `$NIDHOME/analysis` directory. These tools assume that this is your present working directory and they will try to access the necessary configuration files and write outputs based on this assumption.

3.1. *capture*

The first step is to start the collection of data. This is done via the *capture* program. *capture* depends on four different configuration files as is shown in Figure 3.1. Assuming these configuration files have been set up correctly and that the authentication step has been performed (see "Installation" description of Chapter 2), the collection of data is started simply by invoking the *capture* program (as root) from the `$NIDHOME/analysis` directory:

```
% cd $NIDHOME/analysis
% su root
Password:
# ../bin/capture &
# ^D
%
```

This command starts the data collection. It should run continuously, usually 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, we 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.

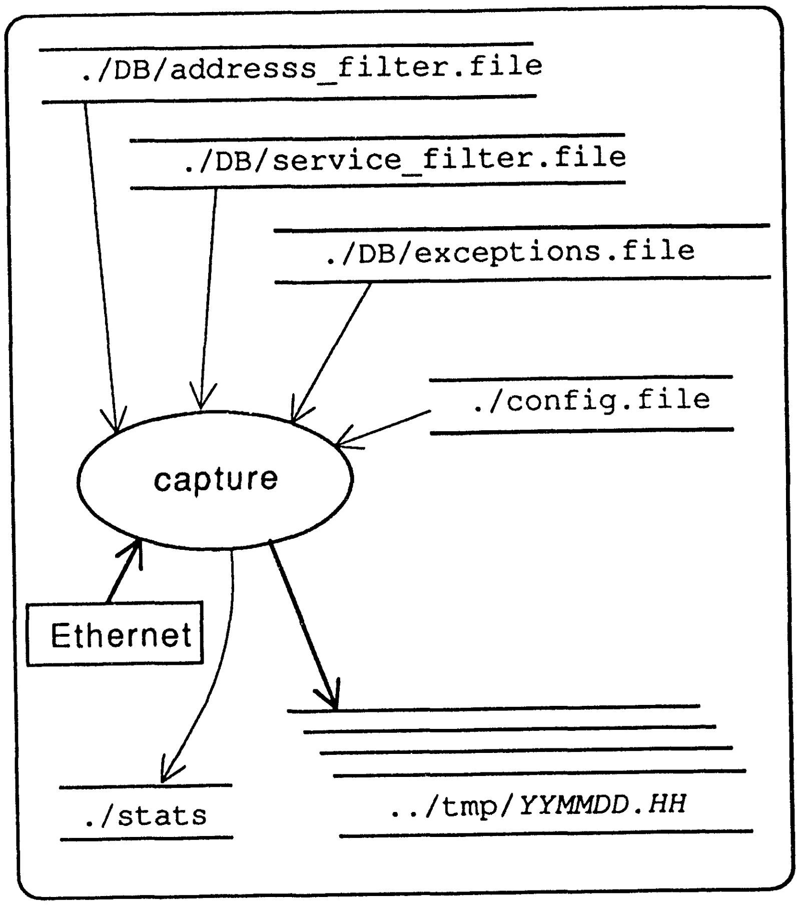


Figure 3.1. Data Flows for *capture* Program. Processes are indicated by circles or ovals. Data stores (files) are represented by names enclosed in parallel horizontal bars. *capture* reads four different configuration files at startup. The major data flow is from the Ethernet to the log files in the `../tmp` directory via the program *capture*.

Notes:
1. To make sure *capture* is not already running before trying to start *capture*, examine the output of the `ps` command.

```
ps -ax | grep capture
   727 co S     10:12 ../bin/capture
   831 co S      0:00 grep capture
```

The first line of output (the second line) in this example, above, indicates that the *capture* program is running. The next line of output indicates that the tail end of the user's command is executing.

2. If you wish to stop the *capture* program, find its process id and issue a simple `kill` command. The *capture* program will terminate gracefully.

```
% ps -ax | grep capture
  727 co S     10:12 ../bin/capture
  831 co S      0:00 grep capture

% su root
Password:
# kill 727
# ^D
%
```

In this example, examining the output of the `ps` command showed that *capture* had a process ID. of 727. Because *capture* had been started from a process having root user-ID, the user had to first `su` to root before issuing the `kill` command.

3.2. *analyze*

The *analyze* program will scan the captured data. The epoch or time period whose data is analyzed is specified as an argument to the program. *analyze* collates these data into distinct sessions and scores the sessions according to intrusion criteria.

Typically, *analyze* is run once each day on the previous day's captured traffic. In high volume situations it will be run more frequently. For incident handling situations, it may be run frequently on small time-segments of just-captured data, if required.

An example of a typical invocation of the *analyze* program is as follows. Assume the user, on March 19, 1994, wishes to process the previous day's data, from 7 a.m., March 18 through 7 a.m., March 19 (24 hours worth of data):

```
../bin/analyze -date 94 3 18 7 24
```

Figure 3.2 shows the data flows for the *analyze* program, illustrating the major inputs and outputs.

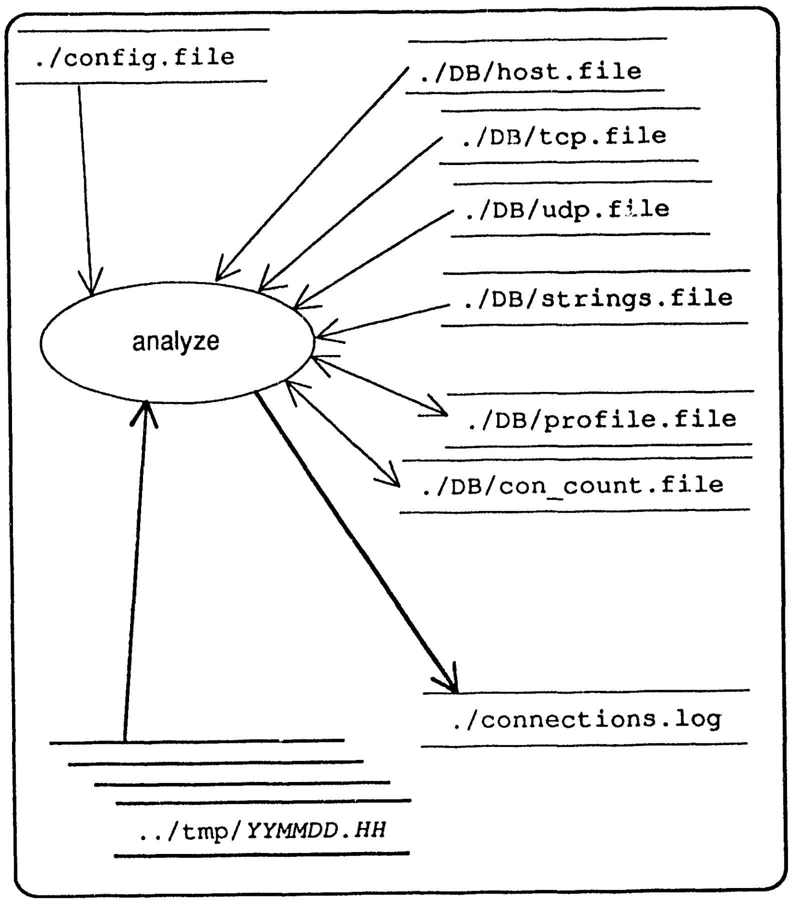


Figure 3.2 Data Flows for *analyze* Program. Processes are indicated by circles or ovals. Data stores (files) are represented by names enclosed in parallel horizontal bars. *analyze* reads five different configuration files as part of its initialization. The major data flow is from the log files in the `../tmp` directory to the `connections.log`. The `profile.file` is read as part of the initialization; it and the `con_count.file` are updated as a result of the processing done by *analyze*.

If the user wishes to keep several days' data on the system, he/she may wish to give each connections log a name that connotes when it was compiled. The `-o` option to the *analyze* program allows the user to override the default name of `connections.log` for *analyze*'s output.

For example on March 19, the user would enter the following command:

```
../bin/analyze -o CL3.19 -date 94
3 18 7 24
```

and the result would be that the connection data would be logged to the file `$NIDHOME/analysis/CL3.19`. Later on, in using other NID tools, the user would explicitly state that the connections log file to be used for processing is `CL3.19` .

For additional options on the *analyze* program, consult the *analyze* `man` pages.

3.3. *report*

Once *analyze* has scanned and processed the previous day's or epoch's data, the NID user will review its results via the *report* utility. A brief report consisting of one line per connection is the default.

The name of the connections log file is a mandatory argument. For example:

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

==================================================
ftp
    index  warning                source             destination
    -----  -------                ------             -----------
       13    2.997       science.college.EDU    aardvark.agency.gov
==================================================
login
    index  warning                source             destination
    -----  -------                ------             -----------
        3    3.410           128.259.99.46         128.259.99.48
        8    3.410           128.259.99.46         128.259.99.48
        9    3.410           128.259.99.46         128.259.99.48
       15    3.410           128.259.99.46         128.259.99.48
       19    3.410           128.259.99.46         128.259.99.48
       20    3.410           128.259.99.46         128.259.99.48
       23    3.410           128.259.99.46         128.259.99.48
==================================================
telnet
    index  warning                source             destination
    -----  -------                ------             -----------
        2    7.878       aardvark.agency.gov       128.259.99.48
        1    7.472          blue5.agency.gov   aardvark.agency.gov
       10    3.965        tooslim.agency.gov       128.259.99.50
```

To have *report* write its output to a file rather than to the terminal, redirect the standard output to your chosen file name, e.g.,

```
../bin/report CL3.19 > report.3.19
```

You may then browse through the report with your favorite read-only editor.

In the above report, the leftmost number in each line is a *connection index*. It uniquely identifies each session with respect to its `connections.log` file.

Detailed report

Note the relatively high warning value reported for the session with connection index 2 (the third last line of the above example). We can generate a detailed report and look for the detailed record for the session with connection index 2.

```
% ../bin/report -detailed \
      CL3.19 > report.3.19d
```

Now assume the user opens this newly generated `report.3.19d` with an editor and searches for a line whose first character is "2" (corresponding to connection index 2). The user will see the following lines on his screen from the detailed report:

```
. . .
==================================================
telnet
2       src:    128.257.99.32                       addvax.llnl.gov
        dst:    128.257.99.48                         128.257.99.48
        svc:  --  telnet --
         start: Sat-Jun-12-19:31:30-1993   stop: Sat-Jun-12-19:38:09-1993
        warning value: 7.878
        strings from source computer:
              <string-2>   2
              <string-3>   2
                  passwd   1
        strings from destination computer:
              Last login   1
              <string-2>   5
              <string-3>   6
             login: root   1
                  passwd   7
         Login incorrect   4
            login: guest   2
------------------------------------------------------
   .  .  .
```

There is enough information here that the reviewer may note that this session with the high warning value was concerned with doorknob rattling, authentication data, and other things.

Other report features

The *report* program has many other capabilities such as sorting by warning value, time, and other attributes. The reader is referred to the `man` pages.

3.4. *transcript*

If the previous filtering indicates a possibility of intrusion, the NID user can generate a detailed transcript of the session in question.

As an example:

```
../bin/transcript CL3.19 2
```

The first parameter is the connection log generated by the earlier run of *analyze*. The second parameter is the connection index for the session of that connection log.

transcript will produce two files, an "init" file and a "dest" file. The `init` file contains a transcript of the data from the initiating host, the host that opened the session. The `dest` file is a transcript of the data that the destination host has sent to the initiating host.

Most often the NID user will open the `dest` file with an editor and search for the occurrence of the matched strings, and then try to get an understanding of the context in which those strings occurred.

The names of the `init` and `dest` files follow the syntax

`<connections log filename>.<connection index>.init`
and
`<connections log filename>.<connection index>.dest`

In the above example, the generated transcript files would be named `CL3.19.2.init` and `CL3.19.2.dest` .

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 user. The second file is the data sent by the user to the target computer.

The transcript of the first file is shown below:

```
TRANSCRIPT

For connection file:  warn94-3-3
and connection index: 218

Initiating host:  128.120.259.251
Destination host: 128.120.257.60
Service:          telnet

Start time: Mon-Mar-03-18:12:03-1994
End time:   Mon-Mar-03-18:12:38-1994

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 (flash)



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

The transcript of the second file is shown below:

```
TRANSCRIPT

For connection file:  warn94-3-3
and connection index: 218

Initiating host:  128.120.259.251
Destination host: 128.120.257.60
Service:          telnet

Start time: Mon-Mar-03-18:12:03-1994
End time:   Mon-Mar-03-18:12:38-1994

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
```

3.5. Warning on the use of NID

The capturing and reviewing of user sessions is sometimes at the detail of "keystroke monitoring." In many circumstances, where there would normally be a reasonable expectation of privacy, such monitoring, beyond what is needed for network maintenance and support, is prohibited. The NID user must read and be familiar with the material presented in Appendix A of this document. It describes legal requirements placed on the use of NID.

4

Tools

This chapter describes the available NID tools. Additional information is found in the `man` pages. These tools will be found in the directory `$NIDHOME/bin/` .

4.1. `capture`

capture captures network traffic and saves it in temporary files for later processing. *capture* uses information from files in `$NIDHOME/analysis/DB` and from `configuration.file` to filter packets.

This program is designed to capture data flowing between a local site and the outside world. If your needs are more specific (e.g., capturing all TELNET and rlogin traffic in and out of a single machine no matter where the packet is from or to), we suggest you use `etherdump`.

4.2. `analyze`

analyze identifies individual network connections and assigns warning values to each connection. The program reads in data files captured from *capture*, and generates the output file of connection records called `connections.log`. You can sort the connections in `connections.log` by warning value with *warn_sort.*

4.3 `previewer`

previewer uses a log file of connections for input and generates a human readable printout of connection records in the log file. Its functions have been largely subsumed by *report.* It is simpler than *report* in that it does no internal sorting of records. This may be useful in dealing with very large connection logs.

4.4 `report`

report reports the information generated by *analyze* and generates a human readable printout of the connection records in the log file. It has several options for generating reports.

4.5. warn_sort

warn_sort uses a connection log file for input and generates a second connection log file containing the connection records sorted by warning value. The connections with the greatest values will be at the front of the new connection log file. This capability has been subsumed by *report*.

4.6. transcript

transcript is the program that generates script files (similar to those generated with the Unix script command) for a connection's input and output streams. The file names for the two transcript 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 log file passed to the program, and `<index>` is the connection number in the connection file for which the script files are generated.

4.7 top_con

top_con generates a shell script file to execute `transcript` for a number of connections. The input includes the connections log file (usually a sorted version of the `connections.log` file), the name of the shell script to be created, and the number of transcripts you want to generate. Executing the shell script then generates transcript files for all the specified connections.
This is used to quickly implement the policy: "Generate for review the transcripts of the sessions with the top n warning values."

4.8 etherdump

etherdump, a modification of the program *etherfind*, captures network traffic matching a filter entered at the command line and saves it in data files. Unlike *capture*, the filtering is targeted at a single host or a single protocol. The filter syntax is exactly the same as that of the Sun program *etherfind* — so for more information, see the man page for *etherfind*.

One bit of warning on the *etherfind* filter syntax: C shell apparently interprets the parenthesis passed to the program, so you need to precede each parenthesis with the back-slash, '\', character. For example, to capture all TELNET traffic (traffic to and from port 23) enter the following command:

```
etherdump -ip \( -srcport 23 -o -dstport
   23 \)
```

4.9. stream

stream generates two network packet data files for a specified connection: one representing the stream from the initiating host (source host), and one representing the stream from the destination host. You can then display the two files with the *packet_print* and/or *playback* programs.

4.10. playback

The *playback* program plays a recorded network connection directly back to the screen as if the events were "live." The screen or window in which *playback* runs will interpret all screen control commands. Therefore, you can view intruders that use visual editors (i.e., vi or emacs), the *talk* program, and other programs that use screen control.

4.11. packet_print

packet_print reads in a network packet data file and prints each file's packet information to the screen. Various options are available to control the amount of information displayed.

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

4.12. TCPdump_conv

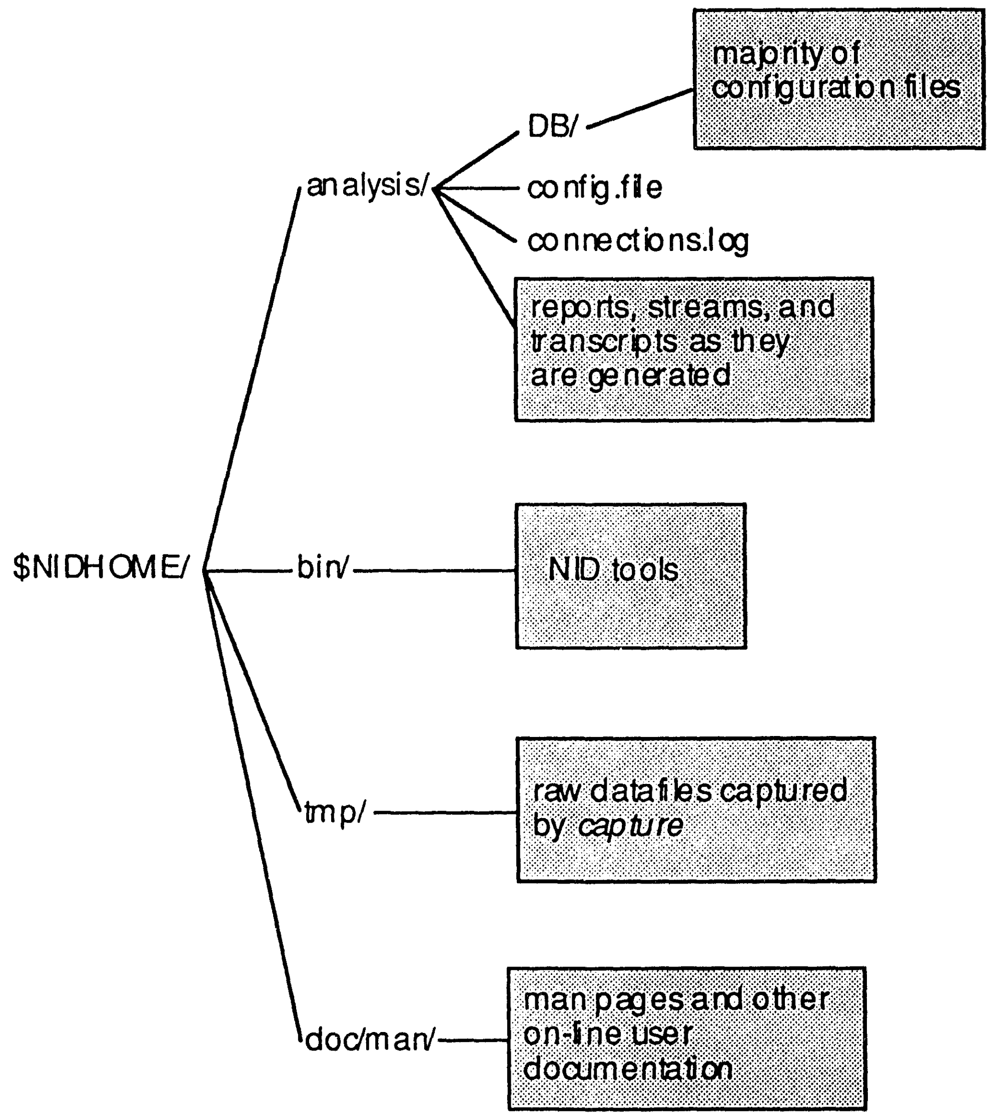
TCPdump_conv can be used to transform data packet files captured by the *TCPdump* program. *TCPdump* is available on several BSD Unix platforms. *TCPdump_conv* enables the NID user to apply the analysis and reporting tools to the data that was captured by *TCPdump* on those other platforms.

5

NID File Descriptions

Below is a description of the directories and files of the NID distribution. Not all files are shown — only major files and directories of interest.

Figure 5-1. Major NID Directories and Files.


$NIDHOME/
analysis/
DB/
majority of configuration files
config.file
connections.log
reports, streams, and transcripts as they are generated
bin/
NID tools
tmp/
raw datafiles captured by *capture*
doc/man/
man pages and other on-line user documentation

5.1. `$NIDHOME/`

`$NIDHOME` is the root directory for the NID product. Underneath `$NIDHOME` are the four directories: `analysis`, `bin`, `doc`, and `tmp`. These directories are described below.

5.1.1. `$NIDHOME/analysis/`

The `analysis` directory is the one from which you will do most of your work within NID. This directory contains the configuration file and the database directory, `DB`, which together direct the various programs of the NID to do their work. The resulting connection and transcript files from running *analyze* and other tools are stored in this directory as well.

5.1.2. `$NIDHOME/bin/`

The `bin` directory contains all the NID executable programs. This directory is static and should not change unless one of the programs is replaced. For a more detailed description, see the chapter entitled "Tools" and the associated `man` pages in `$NIDHOME/doc/man1` .

5.1.3. `$NIDHOME/doc/`

The `doc` directory contains all the NID documentation. The directory contains two sub-directories: `man1/` and `man5/`— the first containing the `man` pages for the NID tools, the latter containing the `man` pages describing the major NID files.

Additional user documentation in different formats will also be placed here depending on the distribution and the version of NID.

5.1.4. `$NIDHOME/tmp/`

The `tmp` directory contains the network traffic data files. The names of these files have the following format: *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. It is possible to use the `config.file` to change the path name and/or file name prefix of these files.

5.2. $NIDHOME/analysis/DB/

The DB directory contains the NID database files and the majority of the NID configuration files. NID reads from and writes to the database files. There is usually no need to access the database files directly once the product has been installed.

5.3. analysis/config.file

In one way or another, almost every NID tool relies on this file. config.file contains the information describing the location to store the data files (e.g., $NIDHOME/tmp/), and how often to switch data files (typically, every hour). The first two lines in the default config file are vestigial and are left in for backward compatibility with earlier, beta versions. The NID tools run fine with the default config.file. An example follows:

```
#num_of_local_class_B_nets 0
#num_of_local_gateways   0
#output_root_file_name   ../tmp/
#minutes_between_files   60
```

Each line contains two pieces of information — a comment and a value.

The third line
`#output_root_file_name  =  ../tmp/`

specifies the directory, ../tmp/, where the network traffic will be stored. It also specifies the prefix in the path name for the data files. In this example, the prefix is null.

The fourth line
`#minutes_between_files  =  60`

indicates that a new file to store the captured data will be created every hour (60 minutes). This mechanism, in general, keeps the raw data files from becoming too unwieldy.

5.4. analysis/connections.log

connections.log is a log file of all the connections observed by the last run of the program *analyze*. Although there is no file called "connections.log" distributed with the NID, the first run of the *analyze* program will create one. Because the *report* and *previewer* tools exist, this data file does not need to be examined directly. However, for completeness sake, a description of the file is included.

The connections.log file is also an important implicit input to many of the NID tools. While learning about those tools, the reader will notice that many of the processing and reporting tools use connections.log as the default log file.

First time readers should skip the rest of this section.

The first line of the connections.log file specifies the number of strings searched for by *analyze*. The following lines are the strings that *analyze* searched for. Following 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 lines on a normal 80 column display. An example follows:

```
7
login: guest
Login incorrect
daemon:
passwd:
login: root
Permission denied
CWD ~ROOT
  218 267389 8.944 5.778 10.000 10.000   128.120.259.251 128.257.99.60   6 25858
    23 telnet Mon-Mar-03-18:12:03-1994 Mon-Mar-03- 18:12:38-1994    35
    51  40  34  144 0-rec-1 1-rec-2
  199 267370 8.944 5.778 10.000 10.000   128.120.259.251 128.257.99.14   6 10498
    23      telnet   Mon-Mar-03-18:10:09-1994   Mon-Mar-03-18:10:36-1994    27
    126        93         71        278 0-rec-1 1-rec-5
  119 267290 8.944 5.778 10.000 10.000   128.120.25.259    128.257.99.67   6 11020
     23      telnet   Mon-Mar-03-17:59:48-1994   Mon-Mar-03-18:00:22-1994    34
    109        81        70        243 0-rec-3 1-rec-3
```

This file represents a connection file that has been sorted by warning value. A line-by-line description is given below.

The first line of the file, containing a number 7, indicates the number of strings searched for while processing the data files. 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 6, because (7 - 1) = 6). Therefore the following string :

```
login: guest
```

has the index number 0, and the string

```
CWD ~ROOT
```

has the index number 6.

Following the strings are the connection logs. The first connection log, although it is one line, is wrapped across three different text lines. The information in this first record represents the following:

`218`	connection index for this particular processing
`267389`	connection index for all time
`8.944`	composite warning value
`5.778`	vulnerability risk model warning value
`10.000`	anomaly detection warning value
`10.000`	attack signature warning value
`128.120.259.251`	address of the host initiating the connection
`128.257.99.60`	address of the host receiving the connection
`6`	protocol of service (6 - TCP, 17 - UDP)
`25858`	port used by initiating host
`23`	port used by receiving host
`telnet`	service name
`Mon-Mar-03-18:12:03-1994`	start time of the connection
`Mon-Mar-03-18:12:38-1994`	ending time of the connection
`35`	duration of the connection in seconds
`51`	number of packets initiator sent
`40`	number of bytes initiator sent
`34`	number of packets receiver sent
`144`	number of bytes receiver sent
`(0-rec-1)`	This is a string match. It 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.

`(1-rec-2)` This is another string match. It states that the string index `1`, "`Login incorrect`," was sent by the receiving host 2 times.

5.5. DB/con_count.file

The `con_count.file` contains a single number representing the total number of connections studied by the *analyze* program. This number is initially set to zero. For example, if you run *analyze* for three days during 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.

5.6. DB/host.file

`host.file` contains a list of Internet addresses and security level values. The data file is usually empty. However, it is possible to specify the security levels (between 0 and 10) of individual hosts. The vulnerability risk model considers a TELNET from a low-security machine to a high-security machine more suspicious than a TELNET from a high-security machine to a low-security machine. The *analyze* program uses the security level value in calculating the vulnerability risk portion of the warning value. Although this file can be empty, the file name must still exist. Any host not seen in this file is given the default security level of 0. An example follows:

Host Number	Host Name	Security Level
128.257.99.48	simbah	8
128.257.99.45	fatcat	8
128.257.99.49	slinky	5
128.257.99.61	catenate	4
128.257.99.42	ftpserver	1
128.257.99.46	bbserver	0

In the example shown above the host `128.257.99.49` is assigned a security level of `5`, and the host with the Internet address `128.257.99.45` is assigned a security level of `8`.

5.7. DB/profile.file

`profile.file` contains a record of past network activity observed by *analyze*. The *analyze* module uses it to calculate the anomalousness of an observed connection. `profile.file` is initially empty. An example follows:

```
555.257.1.1   128.257.99.1   6   25   0   32
555.86.71.1   128.257.99.1   6   79   0   1
555.86.71.2   128.257.99.20   6   513   1
   255
555.228.1.2   128.257.99.20   6   25   1   255
555.118.56.2   128.257.99.20   6   25   0
   111
555.39.1.2   128.257.99.20   6   25   0   48
555.86.71.2   128.257.99.20   6   514   1
   136
555.245.1.2   128.257.99.20   6   25   0   33
555.10.1.2   128.257.99.20   6   25   0   67
```

The first line of the profile represents a data path from the host `128.257.1.1` to `128.257.99.1` 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).

5.8. `DB/strings.file`

`strings.file` is perhaps the most useful file for detecting intrusions. `strings.file` contains a list of strings that the *analyze* program searches for in the network connections. For example, you can search for "`login: guest`" in the network connections. If it is matched, it usually implies that someone tried to log in as guest. A number of strings are automatically searched for (e.g., "`Login incorrect`"), with the results used by *analyze* to calculate the intrusion signature/expert system component of the warning value. However, the search for other strings is done by simply appending a new string at the end of the `strings.file`.
An example follows:

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

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

```
egrep "5-rec|5-init" connections.file
```

5.9. DB/tcp.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 requirements (each number has a value between 0 and 10). The higher the number, the greater the service's capabilities or authentication requirements. For example, TELNET has strong capabilities and strong authentication requirements (a password). *analyze* uses the capability and authentication requirement values in calculating the vulnerability risk/expert system component of the warning value. An example follows:

Port Number	Port Name	Capability	Authentication
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 1 (very low), and its authentication is also listed at a strength of 1 (again, very low).

5.10. `DB/udp.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). The higher the number, the greater the service's capabilities or authentication requirements. For example, TFTP has strong capabilities, but it has very weak authentication requirements (none). *analyze* uses the capability and authentication requirement values in calculating the vulnerability risk/expert system component of the warning value.
An example follows:

Port Number	Port Name	Capability	Authentication
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 the third row of the example above, the service "`tftp`" uses port number `69`. It has a capability strength of `8` (fairly high), but it has an authentication strength of `1` (very low).

5.11. `DB/address_filter.file`

`address_filter.file` defines the security domain of importance to NID. The security domain is a set of nodes and sub-nets. Only network traffic that crosses the security domain boundary is captured for analysis by NID. Traffic between two addresses within the security domain is ignored. Likewise, traffic between two addresses from "the outside" is ignored, even if it traverses the local network being monitored.
The `address_filter.file` has one entry per line. An entry designates either a node or a sub-network.
An example follows:

```
126.
128.259.
128.257.152.
128.257.99.61
bobcat
flatcat.agency.gov
```

The first three entries of the above example represent subnetworks. The trailing period is what distinguishes a subnetwork entry from an entry consisting of a node's address.

The fourth entry is a node, specified by its address.
The fifth and sixth entries are nodes specified by their node names. If the NID user specifies nodes by their names, as is done here, the host that runs NID must be able to resolve those names to IP addresses. This will happen either statically via entries in the `/etc/hosts` table, or dynamically by going out to the net and asking.

5.12. `DB/service_filter.file`

`service_filter.file` specifies the protocols that will be monitored by NID.

The `service_filter.file` has one entry per line; its format is similar to that of the file `/etc/services`.
The first field holds the protocol's name. The second field consists of a port number followed immediately by a slash, "/", followed immediately by either the string "tcp" or the string "udp". Anything beyond the second field is considered comments.
An example follows:

```
telnet   23/tcp
ftp      21/tcp
login    513/tcp
shell    514/tcp    cmd #no password used
name     42/udp     name server
tftp     69/udp
```

5.13. `analysis/stats`

`stats` is a file of traffic volume statistics written by the *capture* program.

Every hour when *capture* closes the current captured data file and opens a new one, it appends to the `stats` file the statistics that it has accumulated over that past hour.

5.14. `DB/exceptions.file`

[See the man pages for a description of the `exceptions.file` .]

Appendix A

A1. DOJ Letter

<letterhead> U.S. Department of Justice
Criminal Division
Office of the Assistant Attorney General
Washington, D.C. 20530

OCT 7 1992

Mr. James H. Burrows
Director, Computer Systems Laboratory
National Institute of Standards & Technology
U.S. Department of Commerce
B-154 Technology
Gaithersburg, Maryland 20899

Dear Mr. Burrows:

It has come to our attention that keystroke monitoring, a process whereby computer system administrators monitor both the keystrokes entered by a computer user and the computer's response, is being conducted by government agencies in an effort to protect their computer systems from intruders who access such systems without authority. We recognize that the unauthorized use of computers, particularly the insertion into a computer system of malicious code (e.g., viruses or worms) and backdoors (programming code that allows an intruder to reenter a system even if compromised passwords are changed), poses a serious threat to the integrity of that system and that keystroke monitoring is the most feasible means to assess and to repair the damage done by such activity. However, we have reviewed the legal propriety of such monitoring for the activities of intruders and, since you are responsible for providing computer security guidance to the federal government, I wish to share our legal conclusions with you. I would also appreciate it if you would, to the extent and in the manner you deem appropriate, circulate this letter to your colleagues in the federal government who are confronted with the keystroke monitoring issue.

The legality of such monitoring is governed by 18 U.S.C. § 2510 et seq. That statute was last amended in 1986, years before the words "virus" and "worm" became a part of our everyday vocabulary. Therefore, not surprisingly, the statute does not directly address the propriety of keystroke monitoring by system administrators.

Attorneys for the Department have engaged in a review of the statute and its legislative history. We believe that such keystroke monitoring of intruders may be defensible under the statute. However, the statute does not expressly authorize such monitoring. Moreover, no court has yet had an opportunity to rule on this issue. If the courts were to decide that such monitoring is improper, it would potentially give rise to both criminal and civil liability for system administrators. Therefore, absent clear guidance form the courts, we believe it is advisable for system administrators who will be engaged in such monitoring to give notice to those who would be subject to monitoring that, by using the system, they are expressly consenting to such monitoring. Since it is important that unauthorized intruders be given notice, some form of banner notice at the time of signing on to the system is required. Simply providing written notice in advance to only authorized users will not be sufficient to place outside hackers on notice.

An agency's banner should give clear and unequivocal notice to intruders that by signing onto the system they are expressly consenting to such monitoring. The banner should also indicate to authorized users that they may be monitored during the effort to monitor the intruder (e.g., if a hacker is downloading a user's file, keystroke monitoring will intercept both the hacker's download command and the authorized user's file). We also understand that system administrators may in some cases monitor authorized users in the course of routine system maintenance. If this is the case, the banner should indicate this fact. An example of an appropriate banner might be as follows:

> This system is for the use of authorized users only. Individuals using this computer system without authority, or in excess of their authority, are subject to having all of their activities on this system monitored and recorded by system personnel. In the course of monitoring individuals improperly using this system, or in the course of system maintenance, the activities of authorized users may also be monitored. Anyone using this system expressly consents to such monitoring and is advised that if such monitoring reveals possible evidence of criminal activity, system personnel may provide the evidence of such monitoring to law enforcement officials.

Obviously, each agency may want to tailor the banner to its precise needs. In addition to giving notice to users that keystroke monitoring may occur, the system administrator might decide that the banner should also contain a statement explaining the need for such monitoring (e.g., "To protect the system from unauthorized use and to insure that the system is functioning properly, system administrators monitor this system").

Lastly, we would note that the long-term-term monitoring of individuals using a system without authority, or in excess of their authority, should not be conducted routinely. The monitoring of such individuals should be limited to the extent reasonable and necessary to determine whether and how the system is being abused. Once that determination is made, the matter should be reported to law enforcement for consideration as to whether court orders authorizing continued monitoring should be obtained.

In sum, we believe that each banner should be crafted by the agency involved to fulfill its specific needs. <u>At a minimum</u>, however, those individuals who are using computers without or in excess of their authority, and those authorized users who are subject to monitoring, should be told expressly that by using the system they are consenting to such monitoring.

Your cooperation in this matter is appreciated.

Sincerely,

Robert S. Mueller, III
Assistant Attorney General
Criminal Division

By:
<signed> John C Keeney
John C. Keeney
Deputy Assistant Attorney General
Criminal Division

Index

User's Comment Form

NID Software Version 1.0 and *NID User's Guide*, UCRL-MA-116609
First Edition: April 1994

Your comments and suggestions concerning this software and publication are welcome. Please express your views regarding the usefulness and usability of the software or the organization, format, accuracy and usefulness of this manual in the space below:

Thank you for your comments.

User:

____
Name (optional)

____
Title

____
Organization or L-Code

Please complete and return to:
Robert Palasek c/o NID Support and Development
CSTC, Mail Stop L-303
LLNL
P.O.Box 808
Livermore, CA 94551-9900

END

DATE
FILMED
9/16/94