SRI International

Software Users Manual • December 1, 1994

Next Generation Intrusion Detection Expert System (NIDES)

Software Users Manual

Beta-Update Release

Debra Anderson, Computer Science Laboratory
Thane Frivold, System Technology Division
Ann Tamaru, Computer Science Laboratory
Alfonso Valdes, Applied Electromagnetics and Optics Laboratory

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Next Generation Intrusion Detection Expert System (NIDES) is powerful state-of-the-art software that supports intrusion detection on single or multiple computers.

1.1 How to Use This Manual

This edition of the User's Manual covers features specific to the NIDES Version 1 Beta-Update Release and comprises the following:

- Chapter 2 — NIDES Architecture and Operation
- Chapter 3 — Tutorial
- Chapter 4 — Statistical Analysis Configuration
- Chapter 5 — Rulebased Analysis Configuration
- Chapter 6 — Reference Manual
- Chapter 7 — Utility Programs
- Chapter 8 — Audit Data Source Customization
- Chapter 9 — Installation Instructions
- Glossary
- Index
- References

If you are new to NIDES, review Chapter 2, which provides information about general NIDES concepts and the NIDES system architecture and modes of operation. Additional information on the NIDES software design can be found in [1].

The tutorial, Chapter 3, can help you get NIDES up and running in your computing environment. It describes in detail each step needed to interact with the NIDES user interface to

- Configure the NIDES analysis components
- Configure alert mechanisms
- Start NIDES analysis and arpool servers
- Select target hosts
- Terminate execution of NIDES
- Exit NIDES

With the tutorial, you can successfully configure and initiate NIDES audit data collection, analysis, and anomaly reporting. The reference manual, Chapter 6, will be useful for your day-to-day interaction with NIDES.

1.2 NIDES Support and Training

Technical support for the NIDES software is available through SRI at a reasonable cost. SRI also offers a NIDES training course. For information on technical support or the training course, contact Debra Anderson via e-mail at debra@csl.sri.com, or by phone at (415) 859-3177.

Chapter 2

NIDES Architecture and Operation

NIDES is a comprehensive intrusion-detection system that performs real-time monitoring of user activity on one or more target system computers. NIDES runs on its own workstation and analyzes audit data collected from the monitored systems to detect unusual and suspicious user behavior.

NIDES analysis runs on a host we refer to as the *NIDES host,* which is not monitored. The NIDES host monitors usage on a number of computers connected via an Ethernet[1] network. These monitored systems, called *target hosts,* communicate their audit data (in a system-independent form called NIDES Audit Records) to the NIDES host. All interactions with NIDES are performed on the NIDES host via the NIDES user interface. Only one user interface runs on the NIDES host at any time. Different hosts can have the role of the NIDES host at different times; they cannot effectively collect data from overlapping sets of target hosts.

2.1 NIDES Components

NIDES comprises several components that interact via remote procedure calls (RPCs) or library calls. The key NIDES components are

- Persistent Storage
- Agend
- Agen
- Arpool
- Statistical Analysis
- Rulebased Analysis

[1] All product names used in this manual are trademarks of their respective holders.

- Resolver
- Archiver
- Batch Analysis
- User Interface

2.1.1 Persistent Storage

The persistent storage component comprises a complete set of library-based functions that provide data management services to NIDES processes. The persistent storage data includes the audit record archive, the result archive, user statistical profiles, and analysis configuration information. A separate set of results data, user profiles, and analysis configuration information is stored for each NIDES test.

Instances

NIDES uses a storage concept called an *instance* to separate different versions of the same type of in format ion. For example, each user-executed test has a name representing an instance of a NIDES test. All results, user profiles, and configuration information for a particular test are identified by the name of the instance associated with the test. Each test must have an instance associated with it, and test instances may be reused (i.e., used for multiple tests). NIDES has a special instance called "real-time", which is reserved for the storage of NIDES real-time operation (results, profiles, and configuration) information.

2.1.2 Agend

The `agend` process is a daemon that should constantly run on all NIDES target host systems. The `agend` program is normally started at system boot-up by all hosts that are to be monitored by NIDES. `Agend` runs as an RPC server process and listens on a "well-known port" for requests to start and stop the native audit data conversion program, `agen`. The NIDES user interface is responsible for sending start and stop requests to the `agend` processes on the target host systems. For additional information on `agend` see Section 7.4.

2.1.3 Agen

The `agen` process runs on each target host system that is actively providing audit data to NIDES for analysis. The `agen` program reads native audit record data, converts it into NIDES audit records, and delivers these records to the `arpool` process. The NIDES Beta release comes with a UNIX version of `agen`, which supports three native audit data types: Sun OS BSM version 1, Sun OS C2, and standard UNIX accounting. When started, the default behavior for `agen` is to process all available types of audit data. Typically, target hosts have one or perhaps two (either BSM or C2, and UNIX accounting) audit data sources.

`Agen` is normally started by `agend` when the user initiates target host monitoring through the NIDES user interface. For additional information on `agen` see Section 7.3.

2.1.4 Arpool

The NIDES `arpool` process collects audit data from the various target hosts and provides the data to the analysis components for analysis and anomaly detection. For additional information on `arpool` see Section 7.7.

2.1.5 Statistical Analysis Component

The NIDES statistical analysis component maintains historical statistical profiles for each user and raises an alarm when observed activity departs from established patterns of use for an individual. The historical profiles are updated regularly, and older data "aged" out with each profile update, so that NIDES adaptively learns what to expect from each user. This type of analysis is intended to detect intruders masquerading as legitimate users. Statistical analysis may also detect intruders who exploit previously unknown vulnerabilities and who cannot be detected by any other means. Statistical anomaly detection can turn up interesting and unusual events that could lead to security-relevant discoveries upon investigation by a security officer. The statistical analysis is customizable: several parameters and thresholds can be changed from their default values, and specific intrusion-detection "measures" (the aspects of behavior for which statistics are kept) can be turned ON or OFF. See Chapter 4 for information on configuration of the statistical analysis component and a summary of the algorithms used. For more detailed information on the statistical algorithms see [2].

2.1.6 Rulebased Analysis Component

The NIDES rulebased analysis component uses rules that characterize known intrusion types to raise an alarm if observed activity matches any of its encoded rules. This type of analysis is intended to detect attempts to exploit known security vulnerabilities of the monitored systems and intruders who exhibit specific patterns of behavior that are known to be suspicious or in violation of site security policy. Observed activity that matches any of these predefined behaviors is flagged. The rulebase is customizable: new rules can be defined and compiled into the running system, and existing rules can be turned ON or OFF. Although NIDES comes with a basic rulebase designed for Sun UNIX operating systems, you will want to customize the rulebase for your particular environment and to keep it up to date with the changing vulnerabilities of new system releases and discovered vulnerabilities of current releases. See Chapter 5 for more information on the rulebased analysis component.

2.1.7 Resolver

The NIDES resolver component screens the alarms generated by the statistical and rulebased components before reporting them to the security officer. Because typically tens to hundreds of audit records can be generated by a single user action, an unusual action could result in tens to hundreds of alarms being reported by statistical analysis, in rapid sequence. To avoid flooding the security officer with redundant alarms, the resolver filters the alarms to remove such redundancies. Alerts can be reported to the NIDES console or to a list of e-mail recipients. Some user-configurable filters are also provided. For example, you can turn off alert reporting for specific users, if you know they will be doing something unusual and would otherwise generate a lot of false alarms. Although filtered alerts are not reported, they are still logged.

2.1.8 User Interface

A NIDES user accesses all NIDES capabilities through an interface, that is the primary focus of this manual. The NIDES user interface is written using the MOTIF toolkit to operate under the X-Window system. Access to the various NIDES functions is provided via pulldown menus, point-and-click selections, and occasional text entry. An extensive multitiered context-sensitive help system is included. The user interface includes a system monitoring facility that displays information on monitored systems, the status of the audit data archiver, an hourly summary of system throughput, and an hourly summary of alert generation.

2.1.9 Archiver

The NIDES `archiver` process stores audit records, analysis results, and alerts. Browsing of the archive is supported through the user interface.

2.1.10 Batch Analysis

The NIDES batch analysis facility allows a security officer to experiment with new statistical parameter settings or new rulebase configurations before committing them to the running NIDES. The NIDES user may construct test data sets from the audit record archive for a specific time window and set of user names. The candidate rulebase and statistical parameters can then be tested against these test data sets concurrent with the running NIDES. Test results are archived for comparison. See Section 6.7 for a description of the NIDES experimentation facility and Section 7.9 for a description of the `batch_analysis` program.

2.2 NIDES Operation

NIDES runs on most Sun SPARCstations. A target host computer can be any type of system that can communicate with the NIDES host and has audit data conversation and transmission software supporting its native audit record format. NIDES can operate either in real time, for continuous monitoring and analysis of user activity, or in batch mode for periodic analysis of audit data.

2.2.1 Real-time Operation

The NIDES primary mode of operation is to analyze data and report suspicious activity in real time. NIDES can monitor numerous, possibly heterogeneous, machines. Figure 2.1 shows how the various NIDES components interact under real-time operation. The flow of

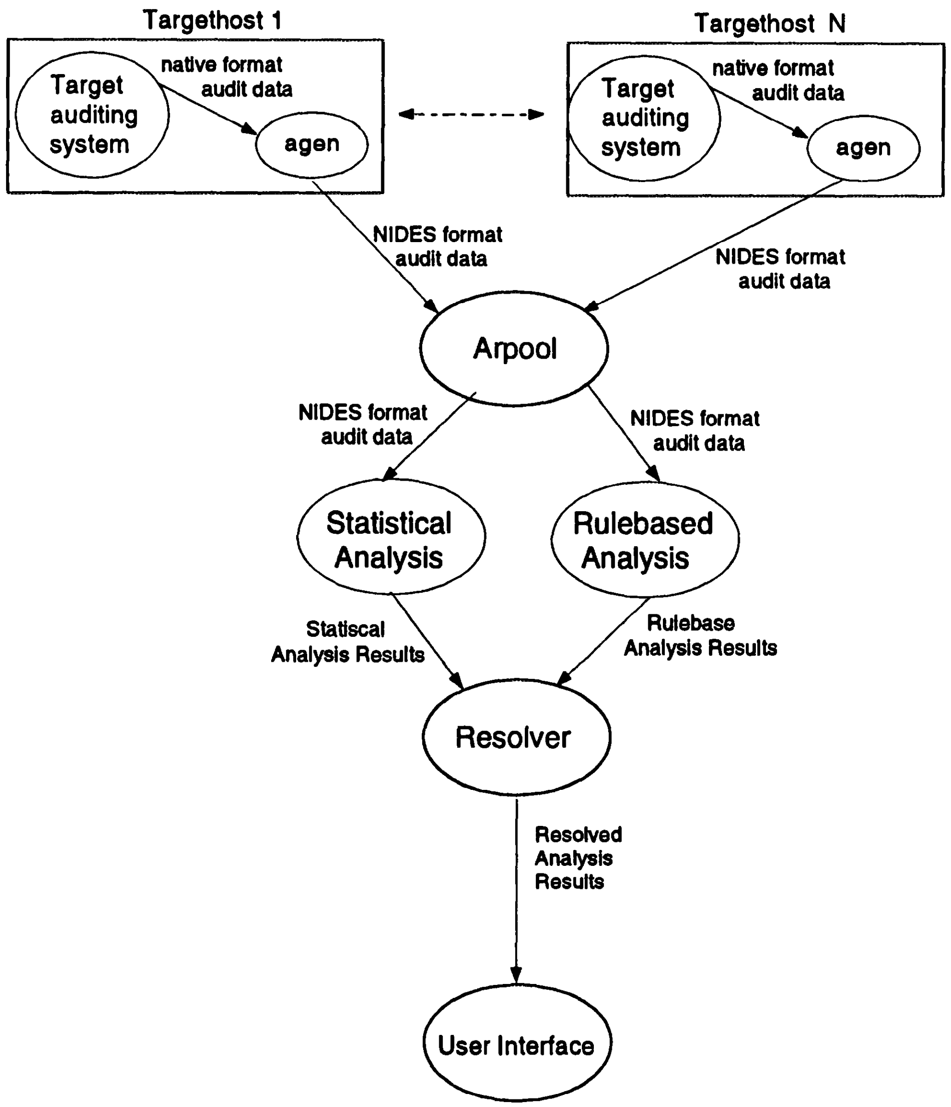


Figure 2.1: NIDES Process Graph (Real Time)

data starts with the target hosts (top of graph). The `agen` process converts data in the target host's native audit record format to a generic audit data format used by NIDES and transmits the NIDES-formatted audit data to the `arpool` process running on the NIDES host. The `arpool` process takes data received from multiple target hosts and coalesces it into a single audit record stream, assigning a unique sequence number to each record during this process. Because NIDES uses a generic audit record format, it is easily adapted to monitor new system types by writing a simple audit data mapping routine. The rulebased and statistical analysis components obtain audit data from the `arpool` process. `Arpool` provides an audit record to all its consumers (i.e., those processes that have made a connection to `arpool` for the purpose of obtaining audit data) and then discards it. After the analysis components have analyzed a single audit record, the results of the analysis are reported to the `resolver.` The `resolver` analyzes the rulebased and statistical results to determine if an alert should be reported. The `resolver` reports an alert to the user interface, provided the user has turned ON at least one of the alert reporting mechanisms (i.e., e-mail or popup window). Alerts generated by the resolver are also archived in the NIDES result archive.

The NIDES user interface initiates and terminates NIDES host processes and the agen processes (via the `agend` daemon).

2.2.2 Batch Operation

NIDES provides a batch mode of operation that allows the user to run NIDES tests on archived audit data, with user-specified NIDES configurations. One or more NIDES `batch_analysis` processes may run concurrently with the real-time mode of operation. A user can initiate a NIDES batch run from the NIDES user interface or from the UNIX command line. For more information on the `batch_analysis` program see Section 7.9. Figure 2.2 shows how a batch analysis process operates. Once this process has been started through the NIDES user interface or the UNIX command line, the user-specified configuration is read from the NIDES instance database. The audit data used for the batch run is specified when the batch process is invoked. The `batch_analysis` process reads audit data from the specified archive, analyzes the audit data using the NIDES statistical and rulebased analysis functions, and writes the result of the analysis to the result archive. If the *reporting* flag is set when the batch job is started, the `batch_analysis` process will periodically report its status to the user interface.

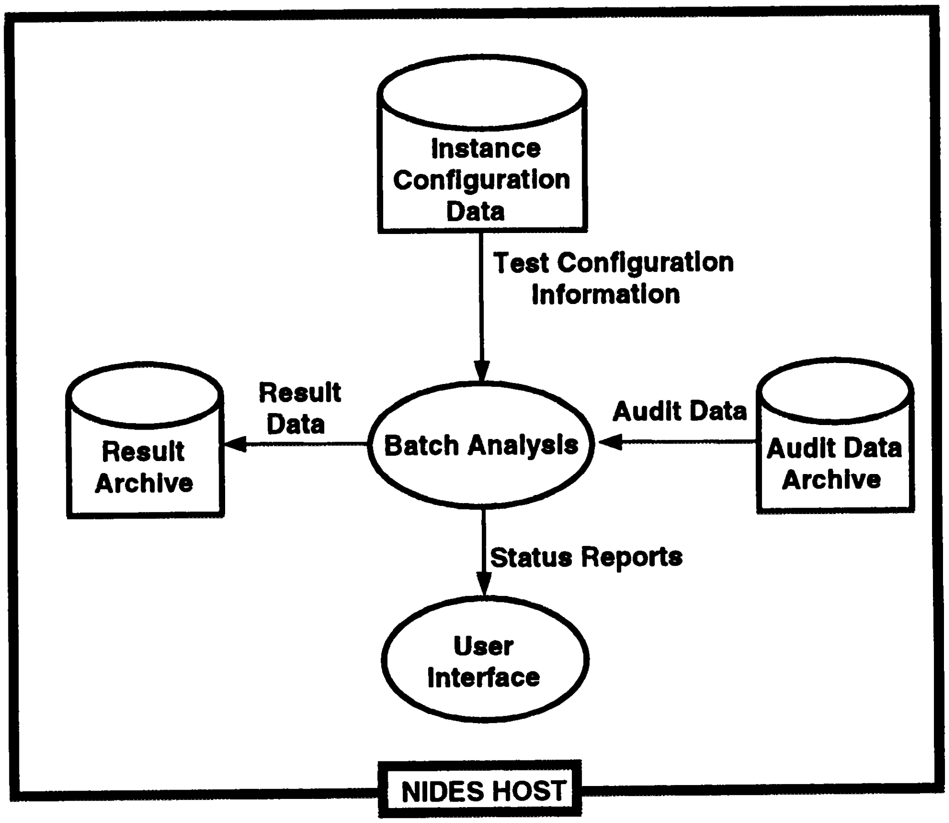


Figure 2.2: NIDES Process Graph (Batch Mode)

Chapter 3

Tutorial

Several configuration steps are required before you can use NIDES. You must begin by properly configuring your computer. Because the NIDES user interface runs under the X Window System, it may be helpful to use an introductory X windows manual with this tutorial. It is also useful to understand how to get help information.

3.1 System Configuration

Before starting up NIDES, make sure your NIDES host machine and all target host machines have been properly configured. Section 6.1 reviews NIDES host configuration and Chapter 9 contains detailed instructions for NIDES installation and configuration.

3.2 Getting On-line Help

The NIDES user interface includes a comprehensive Help system, and each currently active window has a *HELP* button. To get initial help on the NIDES user interface, choose the HELP Option on the Help Menu of the Main Window, Figure 3.1. Refer to Section 6.9 for a detailed description of the Help system.

3.3 Running NIDES

Once you have configured your system to run NIDES and you have successfully started up X windows, you are ready to bring up the NIDES user interface. To do this, bring up an xterm window and type `nides` at the UNIX prompt to display the Main Window (Figure 3.2), which will serve as the backdrop for all your interactions with the user interface.

The Main Window comprises two areas. At the top is a pulldown menu bar with seven options: Setup, Monitor, Browse, Customize, Experiment, Quit, and Help.

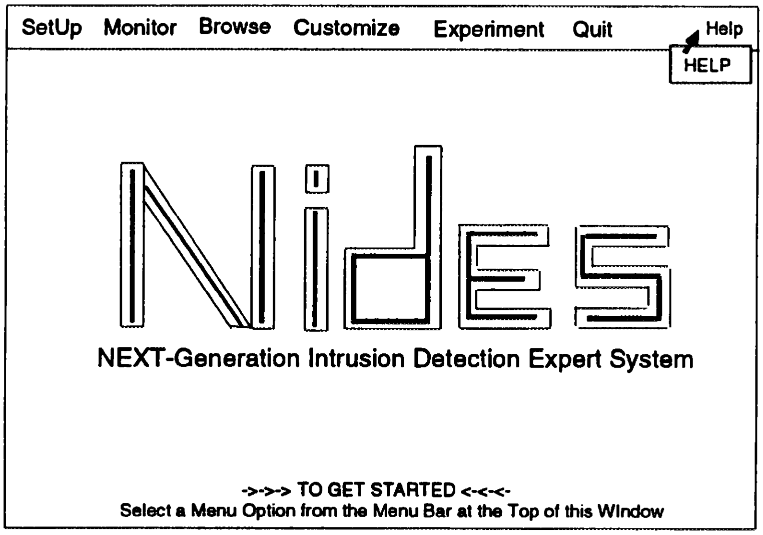

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
HELP
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.1: Help Menu

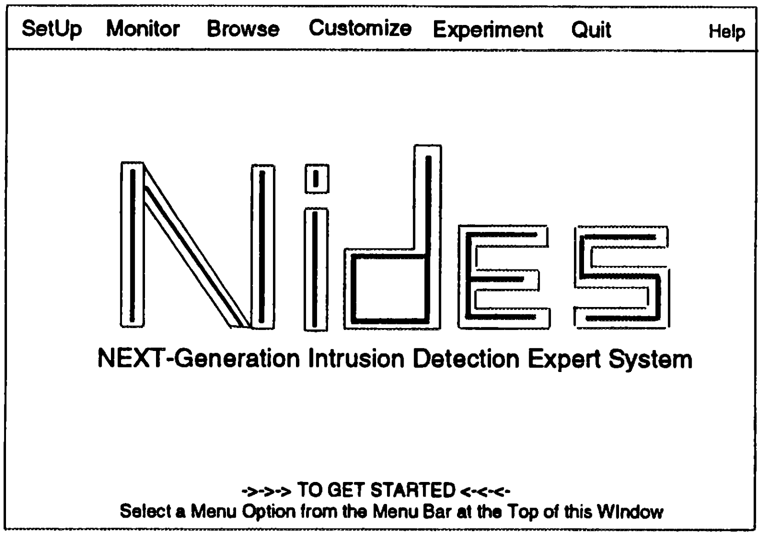

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.2: Main Window

The larger part of the window contains textual information and the NIDES logo. While you are executing NIDES commands from the menu bar, smaller windows containing messages or asking questions are displayed on top of the Main Window.

Once you have brought up the user interface, follow three steps to get NIDES analysis and anomaly detection running in your environment:

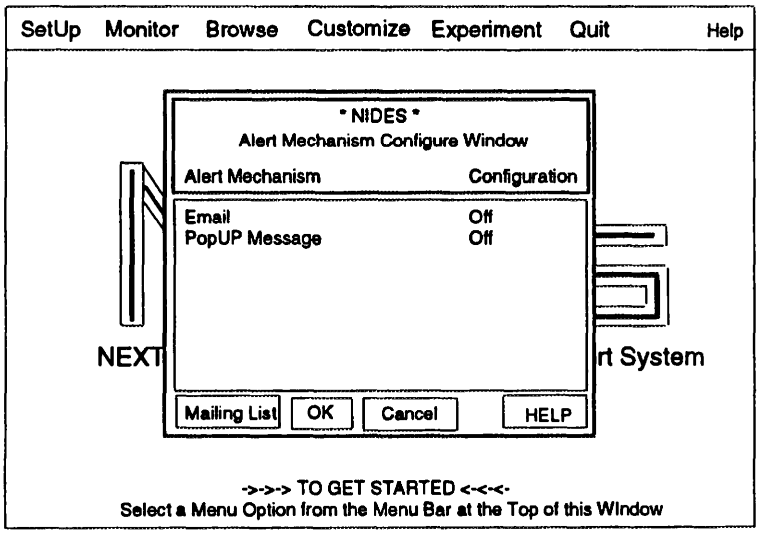
1. Configure the alert reporting mechanism.
2. Start the NIDES analysis and arpool processes.
3. Select target hosts that will provide audit data to NIDES.

3.3.1 Configuring Alert Mechanisms

After you have successfully started NIDES, you must select the alert mechanisms you want NIDES to use when notifying you about anomalous events. To configure these mechanisms, select the Alert Method option of the SetUp Menu of the Main Window. The Alert Mechanism Configure Window, as shown in Figure 3.3, is displayed and comprises two main areas:

1. A list of all the available alert mechanisms and their current configurations
2. A panel of buttons located below the display area

Click on either mechanism to toggle it ON or OFF.



* NIDES *
Alert Mechanism Configure Window

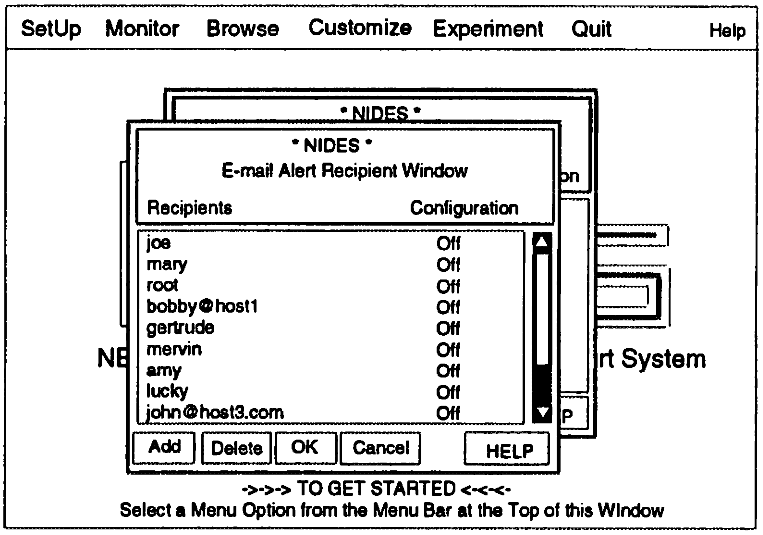
Alert Mechanism	Configuration
Email	Off
PopUP Message	Off

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.3: Alert Mechanism Configure Window

You must take two steps to activate your selected mechanisms. If you have turned ON the e-mail alert mechanism, then configure a mailing list for your e-mail alerts by selecting the

Mailing List option on the Alert Mechanism Configure Window. An E-mail Alert Recipient Window is displayed as shown in Figure 3.4, with its two main areas. One area lists potential e-mail recipients and their current configurations (ON or OFF). Below the recipient listing is a panel of buttons labeled Add, Delete, OK, Cancel, and HELP.

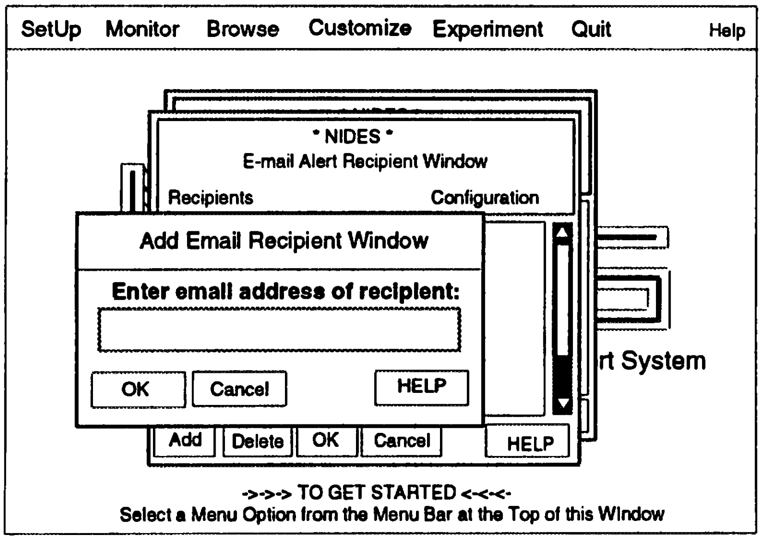


Recipients	Configuration
joe	Off
mary	Off
root	Off
bobby@host1	Off
gertrude	Off
mervin	Off
amy	Off
lucky	Off
john@host3.com	Off

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.4: E-mail Alert Recipient Configure Window

When you first start up NIDES your mailing list will be empty and you will need to add names. To do this, select the *Add* option on the E-mail Alert Recipient Window. A window is displayed as shown in Figure 3.5. To enter a recipient name, click in the entry box, type in the name, and select *OK*.



->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.5: E-mail Alert Recipient Add Window

To select the recipients of NIDES e-mail alerts, click ON all recipients you want on your mailing list. Once you have configured the mailing list, select *OK* to display a Confirmation Window that summarizes the changes. If you are satisfied with the changes displayed in the window, select *OK* to put your changes into effect and return to the Alert Mechanism Configure Window. If you are satisfied with the alert mechanisms you have chosen, select *OK* to record your alert choices and return to the Main Window.

3.3.2 Starting NIDES Servers

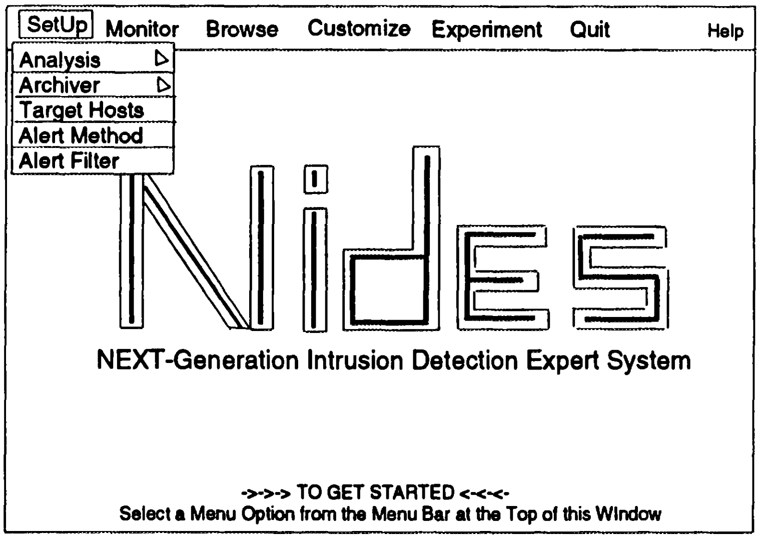

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
Analysis
Archiver
Target Hosts
Alert Method
Alert Filter
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.6: NIDES Main Window Setup Menu

The second step in getting NIDES operational in your computing environment is to start up the NIDES analysis and arpool servers. To begin, select the Analysis option located on the SetUp Menu, shown in Figure 3.6, of the Main Window. Note that while this menu contains five options, three of them are inactive or unavailable as indicated by the lighter text used for their labels.

To start NIDES analysis, bring up the SetUp Menu by selecting the SetUp option on the Main Menu bar. Then select the Analysis option on the menu; since this is a walking menu, drag your mouse to the right until the START option is highlighted, as shown in Figure 3.7, and release the mouse button.

A Confirmation Window as shown in Figure 3.8 is displayed. As with all NIDES Confirmation Windows, this window contains a message in the form of a question and three buttons, labeled OK, Cancel, and HELP. Select one of the three buttons:

- **OK**— Initiates the start-up of the analysis and arpool servers
- **Cancel** — Cancels the Start option and returns you to the Main Window

- **HELP —** Displays a Help Window with information about the currently activated option

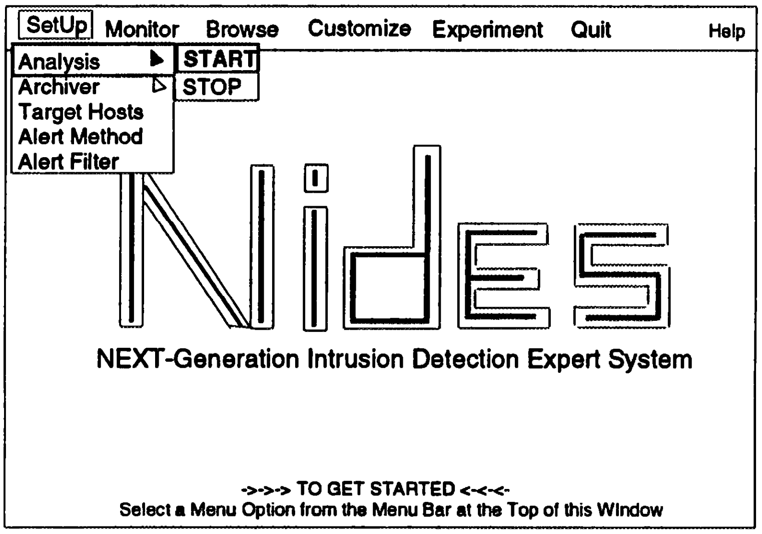

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
Analysis
Archiver
Target Hosts
Alert Method
Alert Filter
START
STOP
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.7: NIDES Analysis Option on SetUp Menu

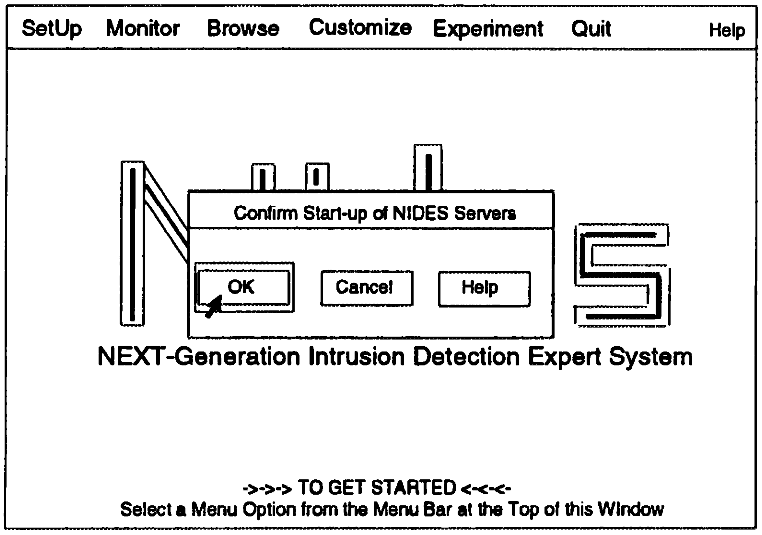
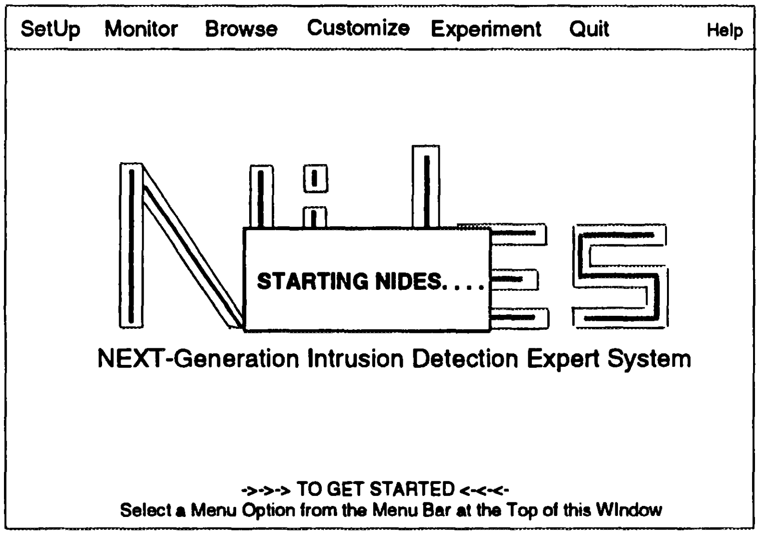

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
Confirm Start-up of NIDES Servers
OK
Cancel
Help
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.8: Start NIDES Confirmation Window

After you have confirmed the start-up of the servers, a message is displayed that tells you the system is initiating the start-up, as shown in Figure 3.9. If the system encounters an error while attempting to start up the servers, an error message is displayed as shown in Figure 3.10.

After the servers have been successfully started, the NIDES System Status Window under the Monitor Menu is updated to reflect the current status of the servers, as shown

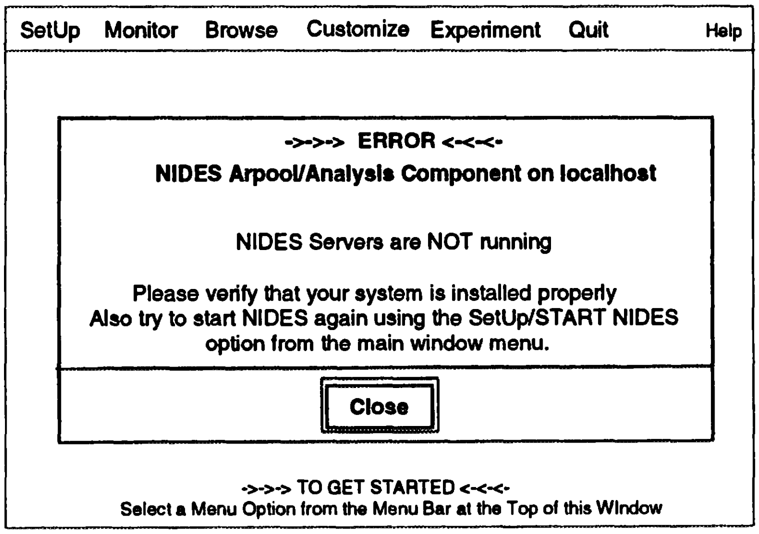


STARTING NIDES....

NEXT-Generation Intrusion Detection Expert System

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.9: Starting NIDES Message Window



->->-> ERROR <-<-<-
NIDES Arpool/Analysis Component on localhost

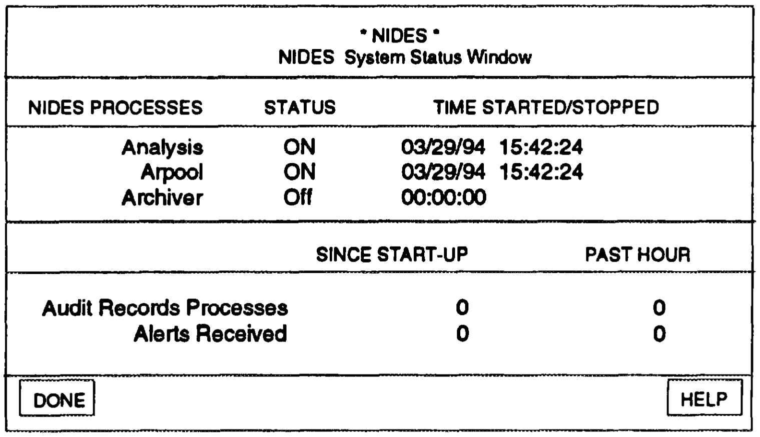
NIDES Servers are NOT running

Please verify that your system is installed properly
Also try to start NIDES again using the SetUp/START NIDES option from the main window menu.

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.10: Starting NIDES Error Message Window

in Figure 3.11. In addition, all the options on the SetUp Menu of the Main Window are now available. The START option of the Analysis menu option is now deactivated, and the STOP option is activated.



* NIDES *
NIDES System Status Window

NIDES PROCESSES	STATUS	TIME STARTED/STOPPED
Analysis	ON	03/29/94 15:42:24
Arpool	ON	03/29/94 15:42:24
Archiver	Off	00:00:00

	SINCE START-UP	PAST HOUR
Audit Records Processes	0	0
Alerts Received	0	0

Figure 3.11: NIDES System Status Window

3.3.3 Selecting Target Hosts

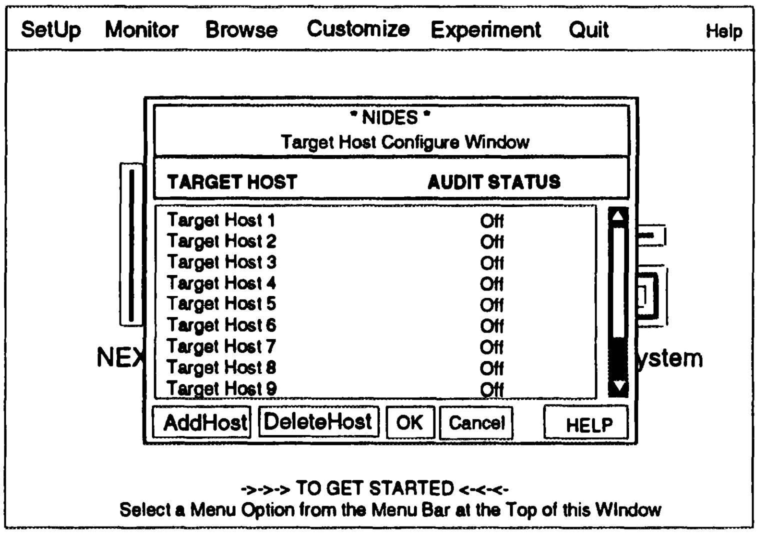
Once you have started the NIDES servers and have selected the initial mechanism(s) for reporting alerts, you can start audit data generation and transmission to NIDES for the target hosts you want to monitor. To configure your target hosts, select the Target Hosts option from the SetUp Menu in the Main Window.

The Target Host Configure Window, Figure 3.12, is displayed and is composed of three areas. At the top is a title area with headings for each column of data that is presented. Below the title area is the target host list, showing the current audit configuration flag (ON or OFF). If the displayed list is longer than the window, a scrollbar is available on the right side of the list window area. At the bottom of the window are five buttons labeled AddHost, DeleteHost, OK, Cancel, and HELP.

3.3.3.1 Adding New Target Hosts

If this is the first time you have run NIDES, your target hosts list will be empty. Before you can initiate NIDES monitoring of a target host's audit trail, you must include the host in your list.

Select *AddHost* to display an entry window as shown in Figure 3.13. Click in the entry window box, enter the hostname, and select *OK* to add the host to your list. A new host's audit status is initially set to OFF. If you decide not to add the host, select *Cancel* to return to the Target Host Configure Window.


SetUp Monitor Browse Customize Experiment Quit Help

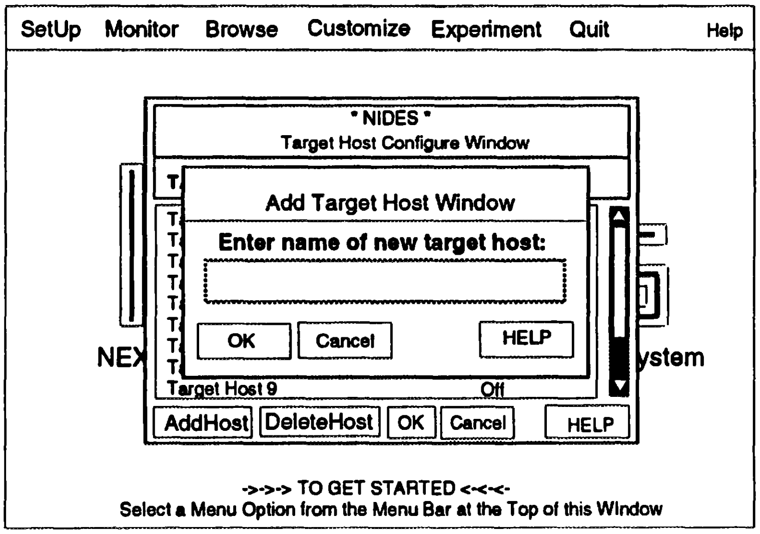
* NIDES *
Target Host Configure Window

TARGET HOST	AUDIT STATUS
Target Host 1	Off
Target Host 2	Off
Target Host 3	Off
Target Host 4	Off
Target Host 5	Off
Target Host 6	Off
Target Host 7	Off
Target Host 8	Off
Target Host 9	Off

AddHost DeleteHost OK Cancel HELP

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.12: Target Host Configure Window


SetUp Monitor Browse Customize Experiment Quit Help

* NIDES *
Target Host Configure Window

Add Target Host Window

Enter name of new target host:

OK Cancel HELP

Target Host 9 Off

AddHost DeleteHost OK Cancel HELP

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 3.13: Add Target Host Window

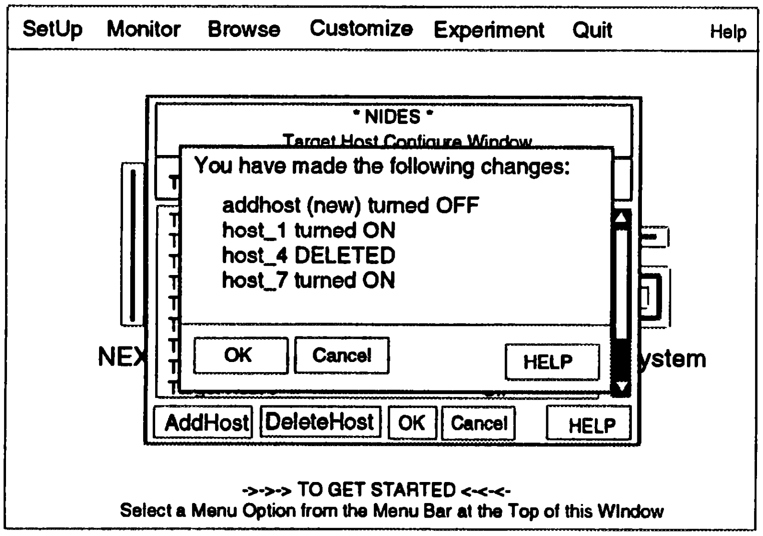
Target Host Name Verification When a new target host name is entered, it is verified in two ways. Only alphanumeric characters and a few special characters (_ . -) are allowed. If the name entered passes this first test, NIDES then checks your system's host tables to see if the host is known to your network. If it is known, NIDES edits your entered name to match the host table primary entry. You may enter a host's alias, if it is listed in your host tables; NIDES will convert the name to the primary entry. If the host cannot be found in any of your system's host tables, an error is reported and your entry is not added.

3.3.3.2 Starting Target Host Audit Data Transmission to NIDES

To start audit data generation and transmission on a particular target host, set the audit configuration flag to ON by clicking on the target host name.

Selecting *OK* initiates configuration of the target hosts. If you decide not to change the target host configuration, select *Cancel* to return to the Main Window. Select *HELP* for information on configuring your target hosts.

After you select *OK*, a Confirmation Window is displayed summarizing the changes. Figure 3.14 shows a Target Host Configure Confirmation Window with a list of the changes and a set of buttons labeled OK, Cancel, and Help.


SetUp Monitor Browse Customize Experiment Quit Help
* NIDES *
Target Host Configure Window
NE
ystem
AddHost DeleteHost OK Cancel HELP
OK Cancel HELP
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

You have made the following changes:

addhost (new) turned OFF
host_1 turned ON
host_4 DELETED
host_7 turned ON

Figure 3.14: Target Host Confirmation Window

If you are satisfied with the configuration specified, select *OK*. A Message Window is displayed to inform you that your request is being processed.

Messages will be sent to those target hosts that were switched ON to begin sending audit data to NIDES, and messages will be sent to hosts that were switched OFF to stop sending audit data to NIDES. If an error occurs during an attempted target host start-up, an error message is displayed. If this occurs, the target host is switched OFF. You can attempt to start the target host again by turning ON that host's configuration.

3.3.4 Stopping Servers

When you no longer want the NIDES servers running on your system, select the Analysis option of the Main Window SetUp Menu to display two options — START and STOP. START is not selectable. Select STOP to halt NIDES analysis.

A Confirmation Window is displayed. If you are sure you want to stop the NIDES servers, select *OK*. When the servers are stopped, all target hosts are turned OFF automatically. When you have confirmed the STOP option, a message window is displayed while the servers are being stopped and the NIDES server Status Window is updated.

To run NIDES again, simply select the START option and then select the target hosts. When NIDES servers are stopped, the alert mechanism setting is not changed and you do not need to set it again — unless you want to change it.

3.3.5 Exiting the User Interface

When you have finished running NIDES and are ready to exit the NIDES user interface, select the Quit option from the Quit Menu in the Main Window. A Confirmation Window is displayed to confirm your exit. If the NIDES arpool and analysis processes are running when the Quit option is confirmed, they are turned OFF. All target hosts that were sending data to NIDES are notified to stop transmitting. Once you have confirmed the exit, the user interface is exited and you are returned to the UNIX shell prompt.

Chapter 4

Statistical Analysis Configuration

The NIDES statistical analysis component analyzes audit data received by NIDES and reports suspicious user behavior through the resolver component. The ability to configure the statistics component is a new feature just introduced in the NIDES beta release. Users with privileged access can configure any items described here. Users with limited NIDES access can configure most statistics parameters, except measure parameters (Qmax, Scalar, Short-term half-life, and Minimum Effective-N) and subject profiles. See Table 6.1 for a complete listing of privileged and non-privileged NIDES functions. If you plan to configure any aspects of the NIDES statistical analysis component, read through this chapter prior to performing any statistical reconfigurations. For more information about NIDES statistical analysis, see [2, 3].

4.1 Algorithms

The statistical approach used in NIDES is to compare a subject's short-term behavior with the subject's historical or long-term behavior. A subject is a user of a computer system. In comparing short-term behavior with long-term behavior, the statistical component is concerned both with long-term behaviors that do not appear in short-term behavior, and with short-term behaviors that are not typical of long-term behavior. Whenever short-term behavior is sufficiently unlike long-term behavior, a warning flag is raised. In general, short-term behavior is somewhat different from long-term behavior, because short-term behavior is more concentrated on specific activities and long-term behavior is distributed across many activities. To accommodate this expected deviation between short-term and long-term behavior, the NIDES statistical component keeps track of the amount of deviation that it has seen in the past between a subject's short-term behaviors and long-term behaviors. The NIDES statistical component issues a warning only if the current short-term behavior is very unlike long-term behavior relative to the amount of deviation between these types of behaviors that it has seen in the past.

The NIDES statistical approach requires no *a priori* knowledge about what type of behavior would result in compromised security. It simply compares short-term and long-term behaviors to determine whether they are statistically similar. A masquerading user may not compromise security in any preconceived way. Nevertheless, if the masquerading user has different characteristics from the user whose identity is being expropriated (and the short-term behavior for the user whose identity is being expropriated is not too erratic), the NIDES statistical component is able to recognize these differences and identify the masquerade.

4.1.1 Measures

Aspects of subject behavior are represented as *measures* (e.g., file access, CPU usage, hour of use). Tables 4.1 and 4.2 (Section 4.6) list and describe the NIDES statistical component measures. For each measure, we construct a probability distribution of short-term and long-term behaviors. For example, for the file access measure, the long-term probability distribution would consist of the historical probabilities with which different files have been accessed, and the short-term probability distribution would consist of the recent probabilities with which different files have been accessed. In this case, the categories to which probabilities are attached are the file names, which are "learned" by the system as they are encountered. In the case of continuous measures, such as CPU time, the categories to which probabilities are attached are ranges of values, which we sometimes refer to as *bins* (these ranges of values are mutually exclusive and span all values from 0 to infinity). We refer to the collection of measures and their long-term probability distributions as the subject's profile.

We have classified the NIDES measures into five groups: activity intensity, audit record distribution, categorical, continuous, and binary. These different types of measures serve slightly different purposes. The activity intensity measures determine whether the volume of activity generated is normal. The audit record distribution measure determines whether, for recently observed activity (say, the last few hundred audit records generated), the types of actions being generated are normal. The categorical and continuous measures determine whether, within a type of activity (say, accessing a file), the types of actions being generated are normal. The binary measures determine if a particular activity was observed in the current audit record. The binary measures do not contribute to the T2 score regardless of their ON/OFF status. Rather, they are used internally for the proper calculation of the audit record distribution measure. User reconfiguration of the binary measures has no effect on NIDES function.

4.1.2 Half-life

The number of audit records or days of audit record activity that constitute short-term and long-term behavior (i.e., that show up in the short-term and long-term probability distributions for a measure) can be set through the specification of a half-life. For the long-term probability distributions, we typically set the half-life at 20 profile updates, which are typically performed once daily. With this setting, audit records that were gathered 20 updates in the past contribute half as much weight toward the probability distribution as do the most recent records, audit records that were gathered 40 updates in the past contribute one-quarter as much weight, and so forth. Thus, the most recent days of activity contribute more than the more distant days of activity, and eventually the long-term profile "forgets" about very distant behavior. For the long-term profile, the long-term aging factor is applied to the historical data at each update, and then the new information is folded in. For the short-term profile, the short-term aging factor is applied to the profile with each audit record, and the current audit record is folded in.

The concepts of aging rate, half-life, and effective length of a profile are related as follows. The aging rate is a multiplicative factor less than or equal to one, by which the existing information in a profile is aged. The smaller the rate, the more rapidly the information in the profile is "forgotten." For the short-term profile, the half-life is the number of audit records that must transpire for the subject before the contribution of a given data item is decayed (downweighted) by one half. For example, if the aging rate is 0.8, the third most recent audit record has a weight of 0.8*0.8*0.8 or 0.512, so we would say that an aging rate of 0.8 corresponds to a short-term profile half-life of approximately three audit records. The effective length of a profile is given by the series sum of all powers of the aging factor, which will converge if this factor is strictly less than 1. For example, an aging rate of 0.8 corresponds to an effective profile length of 5 audit records. As a rule of thumb, the effective length of the profile is approximately 1.5 times the half-life. We refer to the effective length of a profile as "Effective-N."

4.1.3 Differences between Long- and Short-term Profiles – The *Q* Statistic

The degree of difference between a long-term profile for a measure and the short-term profile for a measure is quantified using a chi-square-like statistic, comparing observation (the short-term profile) to expectation (the long-term profile). We call the resultant numerical value Q. Each measure has a different Q value. Large values of Q mean that the long-term and short-term profiles for the measure are very different from one another and therefore warrant some suspicion; a Q value of zero means that they agree completely.

In calculating the Q statistic, we need to refer to the categories that have been defined for each measure. For numerical reasons, it is desirable that categories that are rarely or never observed in the long-term distribution be collapsed together. In NIDES we have defined a parameter, called "maxsumrareprob", that controls the cumulative probability of categories that are collapsed together.

4.1.4 Scoring Anomalous Behavior – The S and T2 Statistics

As with all other long-term distributions in NIDES, the long-term distribution of Q is categorized into ranges of values or bins (see Section 4.2). For each audit record, the NIDES statistical component generates a vector of Q values, with large values indicating suspicious activity. Unfortunately, it is not possible to refer Q directly to a chi-square table of cutoff values to determine if the difference between the two probability distributions is statistically significant. The traditional chi-square method cannot be applied in environments where the audit records that contribute to the short-term profiles are not independent, nor in systems where the short-term profiles are based upon very few audit records. Since the distribution of Q is not chi-squared, we need to track its values to determine what its distribution looks like. Every time another audit record arrives, a new value of Q is generated. We observe these values over a substantial portion of the profile-building period. We begin to observe Q values during the second stage of the profile-building period, which commences as soon as we have constructed reasonably stable historical profiles for the observed categories. We use these values for Q to build a long-term probability distribution for Q. (Each measure has its own Q and a long-term distribution for that Q statistic.) The Q distributions look somewhat like long-tailed and stretched-out chi-square distributions.

From this distribution, we can obtain the tail probability of obtaining a value for Q at least as great as the observed value. We transform this tail probability and denote the transformed variable as S, and define the transformation so that S has a half-normal distribution. (A half-normal distribution looks like the right-hand side of a normal distribution, except that the height of the probability distribution is doubled so that there is still unit area under the curve. This is also the distribution of the absolute value of a normally distributed variable.) The mapping from tail probabilities of the Q distribution to half-normal values requires only a table of normal probabilities. For example, if the current Q statistic is at its 80th percentile, then the corresponding value for S is 1.28 (i.e., the same as the 90th percentile of a normal distribution). If the current Q statistic is at its 90th percentile, the corresponding value for S is 1.65 (i.e., the same as the 95th percentile of a normal distribution).

As each audit record is received, it is possible to generate the corresponding vector of S values. High S values correspond to measures that are unusual relative to the usual amount of discrepancy that occurs between long-term and short-term profiles. Small S values correspond to measures that are not unusual relative to the amount of discrepancy that typically occurs between long-term and short-term profiles. We combine the S scores into an overall statistic that we call T2. The T2 statistic is a summary judgment of the abnormality of many measures, and is in fact the sum of the squares of the S statistics (i.e., $S_1^2 + S_2^2 + \ldots + S_m^2$, where there are m measures).

Because each S statistic follows the same half-normal distribution (when the audit records being observed come from the subject who owns the account), the T2 statistic is approximately distributed like a chi-squared variable with *m* degrees of freedom. However, the S; are not necessarily independent, so rather than rely on the chi-square distribution to provide threshold values for T2 we build a long-term distribution for T2. We then observe the upper threshold values for T2. (The long-term distribution for T2 is built during the last stage of the profile building period, after reasonably stable long-term distributions for the Q statistics have been constructed.) We declare recent audit records to be anomalous at the yellow level whenever the T2 value is over the 1% value of the long-term profile for T2 and at the red level whenever the T2 is over the 0.1% value of the long-term profile for T2.

4.2 Statistical Component Operation

The NIDES Statistical Analysis component performs a complete analysis of each audit record received. Every day (or according to a user-defined update schedule), it also updates all long-term profiles for subjects who have had activity in that day. Initially the component is in *training* because long-term profiles are being created. A new profile (long-term and short-term) is created whenever a new subject is encountered– that is, when the first audit record for a subject is received by NIDES. The Statistical Analysis component will continue to *train* by recording and updating a subject's behavior in the subject's long-term profile. A subject's long-term profile is considered trained when at least one measure has gone through the C, Q, and T2 training phases and has at least the minimum number of observations as defined by the measure's Minimum Effective-N configuration– at this point anomalies may be reported.

4.2.1 Audit Record Processing

For every audit record processed by the Statistical Analysis Component, these steps are followed:

- **Activity Vector Construction** — The first step in processing an audit record is the construction of an activity vector. For every measure represented in the audit record, the associated audit data is converted to a continuous or categorical value, depending on the type of measure, and placed in the activity vector entry for the measure. CPU time associated with an audit record is an example of a continuous measure, while file name is an example of a categorical measure. Continuous measures are recoded to 32 categorical bins (ranges for each bin are calculated based on the Scalar value for the measure) so that most measures are treated as if they were categorical. Special measures capture intensity of activity in the last minute, ten minutes, and hour. The audit record distribution measure captures the frequency with which the measures were touched by recent audit records (an audit record might touch multiple measures). For this measure, the categories are the measure names themselves.

- **Adjust Short-term Category Counts** — For each measure affected by the current audit record, the measure value recorded in the activity vector is used to adjust the category counts that are maintained in the short-term profile.

- **Q Calculation** — After the short-term profile counts are adjusted, they are compared to the expected counts based on the long-term profile, and for each measure a statistic called Q is computed. This statistic is similar in some respects to a chi-square difference on a per-measure basis between the long- and short-term profiles.

- **S Calculation** — After the Q values have been calculated, for each measure, the tail probabilities are calculated from previously trained empirical distributions. This distribution of tail probabilities is inverted using a half-normal distribution function to represent the score for all measures in similar terms, which we term the S statistic.

- **T2 Calculation** — The S values are squared and summed across measures to give the overall score value or T2 statistic.
- **Anomaly Determination** — The T2 score is compared against the subject's red and yellow thresholds. If the score exceeds the red threshold, a critical level result is reported. If it is below the red but exceeds the yellow threshold, a warning level result is reported. If it is below both thresholds, a safe level result is reported. The resolver component of NIDES decides if any critical level results generated should be converted to alert status.

Normalization by the number of categories at the Q calculation stage and by the number of measures at the T2 calculation stage gives the system stability as the number of categories or active measures changes.

4.2.2 Long-Term Profile Updating

The statistical analysis component performs a complete update of the long-term profiles every 24 hours. Profiles for subjects with activity in the past 24 hours are updated. Prior to updating, category counts for each measure (including any new categories not in the long-term profile) are maintained. When the update occurs, historical category probabilities are combined with the new counts using a weighting proportional to historical and new counts and exponentially aging the long-term data to permit adaptation to the new behavior (the aging constant is defined by the long-term profile half-life). Categories for which the probability is below some minimum are dropped from the long-term profile, and some other categories might be combined into the RARE category (although for these the actual counts are maintained so that a category can cease to be RARE if its usage increases). The update process yields an updated category probability distribution for the observed values for each measure. The distributions for Q and T2 are similarly updated except that there is no provision for "drop" and "rare" with respect to Q and T2. Score thresholds are computed from the T2 distribution based on the configured yellow and red threshold percentages (which are in fact nominal false-positive rates) and the empirical T2 distribution. Updates usually occur at a time of low system usage, as they are compute-intensive, although the user can cause an update to be done at any time. Section 4.8 discusses profile update configuration options.

4.2.3 Profile Training Phases

The statistical component training period is the interval of time (measured in the number of profile updates) that is required before the scoring mechanism comes into play. The default training period is 20 updates. To get to a trained state, the NIDES statistical component must pass through three training phases, respectively, to learn raw categories (the C phase), the Q empirical distribution (the Q phase), and the T2 empirical distribution (the T phase). The number of updates required to complete each training phase is the training period (by default 20 updates) divided by the number of phases, 3 and rounded up to the nearest whole number. By default each training phase, C, Q, and T requires 7 updates to complete. The training status of each measure can be ascertained by viewing the subject's profile. Section 4.7.1 discusses profile viewing functions. All measures are untrained at system startup; various configuration procedures can cause previously trained measures to revert to an "untrained" status with one or more of the phases (C, Q, T2) required for the measure to contribute to the T2 score again.

4.3 Configuration Application

The user can configure many statistical parameters to tune the NIDES statistical analysis component for a given environment. The configurable parameters are listed in Tables 6.17, 6.18, and 6.19 on pages 170, 171, and 172. Although all items shown can be reconfigured while the system is running, some take effect immediately, while others require at least one profile update to take effect. In addition, changes to some measure configurations require partial or full retraining for the affected measure. The statistical analysis component automatically handles all instantaneous and deferred reconfigurations as well as associated retraining, but the user should be aware of consequences before performing any reconfiguration.

The discussions that follow describe the statistical parameters that can be reconfigured, indicating cases when reconfiguration is warranted. As a general rule, the statistical analysis component is sufficiently robust that a wide range of values for parameters such as profile half-life or measure scaling parameters give good detection sensitivity, so that changes should be made only if there is evidence that the parameter setting in question is significantly far from optimal.

4.4 Classes

NIDES has the capability of grouping several kinds of activity into class lists. As an example, compilers are a class of activity, and the categories for the class are the compilers available on a system. Other activity classes include editors, mailers, shell environments, window commands, network commands, and local hosts, all of which are potentially indicative of a user's style. The temporary file class list functions differently in that items in this list represent uninteresting files for which profile categories are not kept. The user may add or delete class members as part of initial configuration or reconfiguration. Table 6.10 on page 156 provides a summarized description of each statistics class, and Table 6.14 on page 167 lists the default configurations of the statistics class lists. If possible, items should be added before they are observable (that is, if the system administrator anticipates installing a new compiler, this compiler's name should be added to the compiler class list beforehand).

Although items can be added or deleted from class lists at any time (with the change taking place at the next profile update), it is worth understanding the functionality of these lists so that changes are made in a way that maintains throughput and sensitivity of the measures. When NIDES encounters an observation for a category in a class list measure, special class list profiling is invoked. Items that should be in the class list but are omitted (for example, an editor not known to the editor list) will affect the profile via their contribution to other measures such as command usage and files. The only information not exploited is the fact that the command represented an editor. For proper scoring, therefore, it is preferable that these lists be configured to recognize a new item (such as a new editor) before data arrives for the item. Removing an item from a class list causes future observations for the item (if any) to be treated as a non-class-list event, and the item contributes to the other NIDES measures as appropriate. However, the category for the deleted class item is not removed from the long-term profile, but is allowed to "age out" through the profile aging mechanism.

4.4.1 Compilers

The compilers used are very characteristic of the individual. The compiler class list should include all the compilers that are available on the monitored system(s). Members of the compiler class should be added or deleted as compilers are added to or removed from the system (additions to this list should take place before the installation of the compiler). The interpretation of the category probabilities for this measure represents the distribution of compiler usage across different compilers. It is desirable to categorize compiler usage with its own measure because compiler usage is of sufficient interest to observe more directly. Otherwise, compiler usage would form only part of the command measure usage categories for a user and would be diluted.

4.4.2 Editors

The editors used are also very characteristic of the individual. The editor class list should include all the editors that are available on the monitored system(s). The editor class maintains the distribution of editors used, so that a user who prefers `emacs` would appear to be different from a user who primarily uses vi, for example, even if the broader measures (i.e., CPU, IO, MEM, and FILE) for the editing session are comparable. Additions to the editor class list should be made prior to installation of the editor.

4.4.3 Mailers

The mailer class characterizes a subject's use of mail-oriented programs. As with editors, most users have a favorite mail program that captures most of their e-mail activity. The mail class list should include all mail programs available on the monitored systems(s). If new mail programs are installed, the programs should be added to the mail class list prior to installation.

4.4.4 Shell Environments

This class captures the shell environments used. A user tends to choose a familiar shell environment, so that for this class the probability is usually concentrated in the category corresponding to the preferred environment. The shell class list should include the names of all shell environments available on the monitored system(s). Additions and deletions to this lists should take place as environments are added or removed.

4.4.5 Window Commands

The window class characterizes a subject's use of window-oriented programs — for example, `xinit` or `suntools`. The window class list should include all the window commands available on the monitored system(s). If new window programs are installed, the programs should be added to the window class list prior to installation.

4.4.6 Network Commands

The network command class characterizes a subject's usage of a system with respect to network activity such as file transfer and remote login. The network class list should include all the network services that are available on the monitored system(s).

4.4.7 Local Hosts

The local hosts class maintains a distribution of a subject's activity within the local host network. The local host class list should include the names of all hosts considered to be part of the local environment. Users have a set of frequently used local hosts. NIDES also uses the local host class list to determine if an activity is executed from a remote host — if a host used is not in the local host class list, it is assumed to be a remote host.

4.4.8 Temporary Files

While the general activity classes characterize a type of activity into constituent classes, the temporary file class is used to screen out uninteresting files that can flood a subject's profile and thus reduce sensitivity. At system start-up, the user should place in the configuration file all directory names that include these type of files (i.e., tmp file directories, news files, and possibly spool areas). Accesses to files in directories identified here do not generate categories in the profiles; rather, the temporary file filter screens them out. Deleting an item from this list results in future observations of that item entering profiles as a normal category. Items added to the temporary file list that have already been observed are not immediately removed from profiles, but are instead aged out by the long-term profile aging mechanism. Thus, it is preferable to add items to this list before they are observed.

The temporary file filter enables the user to improve the sensitivity of file-related measures. Without this filter, temporary files created by any number of programs would flood the category list with entries for which there are only a handful of observations and which are typically not encountered again. This can lead to an explosion in the category list, which can severely slow down NIDES in addition to diluting the detection statistic. Usage of files in screened directories contributes to the profile and scoring mechanism via other measures — for example, resource use measures such as CPU and IO.

4.5 Parameters

NIDES allows the user to configure the following parameters to tune the statistical analysis component to suit the environment:

- Long-term half-life
- Training period
- Thresholds
- Maximum sum for RARE category probability
- Profile cache size

4.5.1 Long-term Half-life

The long-term profile half-life is that time period (measured in the number of profile updates) by which the contribution of a given day's data is downweighted by one-half. The larger this value, the more representative the long-term profile will be with respect to the full spectrum of activity for a subject. Smaller values permit more rapid adaptation to recent behavior. The default value of 20 represents a month of working days and is appropriate for most purposes. NIDES continues its scoring if this value is changed, but the user must keep in mind that profiles based on very short half-life values are less stable and thus more likely to give false alarms. A value that is too long, on the other hand, may not enable the profile to adapt to gradual changes in a subject's activity patterns, and the profile's sensitivity may be reduced.

4.5.2 Training Period

The NIDES training period is the interval of time (measured in the number of profile updates) from system startup or for a new subject that is required before the full scoring mechanism comes into play. The default value is 20 updates, divided equally among C, Q, and T training (see Section 4.2.3). Changing the training period does not affect measures already trained (even if they would not be considered trained under a new, larger value). For measures in training, the current training phase completes under the previously configured value, then subsequent phase(s) are completed in a number of updates equal to one-third of the new value. It is our experience that false alarm rates can be high even with a 20-update training period, so we do not generally recommend reducing this value. The length of the training period does not affect the steady-state behavior of NIDES once profiles have been trained.

4.5.3 Critical (Red) and Warning (Yellow) Thresholds

NIDES by default declares activity as anomalous if it exceeds the 99th percentile (for Warning status) or the 99.9th percentile (for Critical status). These correspond to nominal yellow and red thresholds of, respectively, 1% and 0.1%. These are nominal false positive rates; the actual observed rates may be higher, particularly in the first few days after scoring is activated. If these values are reset, the new values take effect at the next update of the long-term profiles.

The user may change these values, with the only requirements that the "yellow" percent must be greater than the "red" percent and that both values must be positive. Increasing these values flags a greater proportion of normal data as anomalous but may allow NIDES to be more sensitive to borderline intrusions. Decreasing them lowers the false alarm rate but may also reduce detection sensitivity.

4.5.4 Maximum Sum RARE Category Probability

This is a feature included to prevent very small expected counts from rendering Q calculation unstable. Categories whose summed probability is lower than this value are scored as a group called RARE. A robust calculation in this version of NIDES allowed us to reduce this value considerably and more faithfully represent observed categories. This parameter should probably not be changed by the user without recommendation from the NIDES development team.

4.5.5 Profile Cache

NIDES maintains an internal cache of the most recently needed subject profiles. When a subject's profile that is not in the cache is needed, the least recently needed profile in the cache is written out to disk and the internal cache entry replaced with the needed subject's profile. Modifications to the cache size do not affect the detection capability of the statistical algorithms, but can affect system performance significantly. If users in your environment generate large profiles (i.e., the profiles include a large number of categories), keeping the cache size smaller will help keep the NIDES process sizes smaller. On the other hand, if you have a large number of users, you may want to increase the size of the cache to reduce the number of times profiles are swapped in and out of the cache.

4.6 Measures

Intrusion detection measures are aspects of user behavior for which NIDES keeps statistical information in the user profiles. Measures can be continuous (such as CPU usage), categorical (such as file usage), intensity (the rate of arrival of audit records), binary, and the special audit record distribution measure (U_ARECDIST). The statistical component converts values for continuous measures into categories representing value ranges, so that all measures are categorical from a computational standpoint. NIDES binary measures represent a "yes" or "no" value indicating whether a particular type of activity was observed in the audit record analyzed. The measures employed by the NIDES statistical component are shown in Tables 4.1 and 4.2.

4.6.1 Measure Activation

The user activates measures from the *Measures* option of the Instance Management Window available from the Customize Menu. This window shows available measures, their type ("CONT" for continuous, "CAT" for categorical and "BINARY" for binary), and status (active/inactive or ON/OFF). The active/inactive status and training status of a measure are independent, so that inactive measures are training continuously and when activated (turned ON) can immediately contribute to the score, if they are trained. Measures that are active but not fully trained do not contribute to the T2 score. The statistical component begins T2 scoring for a subject when at least one active measure for the subject reaches "trained" status. When the active/inactive status for a measure is changed, the change takes effect immediately, and no profile update is required. NIDES readjusts its normalization parameters for T2 accordingly. Nonetheless, a radical change in the active measure mix may result in unreliable system performance (as measured by false alarms) until the system restabilizes. Users may activate any arbitrary subset of these measures. We recommend that the intensity and audit record distribution measures always be configured as active. Measures that are most likely to aid in differentiating users should be activated. Those that are more likely to be similar across different users may be deactivated. If you are seeing many false alarms in which the top measures that contributed to the alarm are the same, you may consider deactivating one or more of the top contributing measures.

4.6.2 Scalar

For each continuous measure, NIDES categorizes the range of values based on the Scalar for that measure. This value should be larger than the largest value ever likely to be seen for the measure across all subjects. Since this is not known beforehand, the default configuration file sets these values on the high side. Since NIDES scales the value range logarithmically (see Section 4.2.1), it is not too critical if this value is considerably higher than the true maximum; the robustness of the scaling mechanism permits good performance even if the value is high by a factor of 10 or more.

The symptom indicating that the Scalar is set too high is that the category probabilities are concentrated in the low numbered bins (or in the zero bin in the extreme case). Figure 4.1 shows an example of the category view screen with values indicating the Scalar is set too

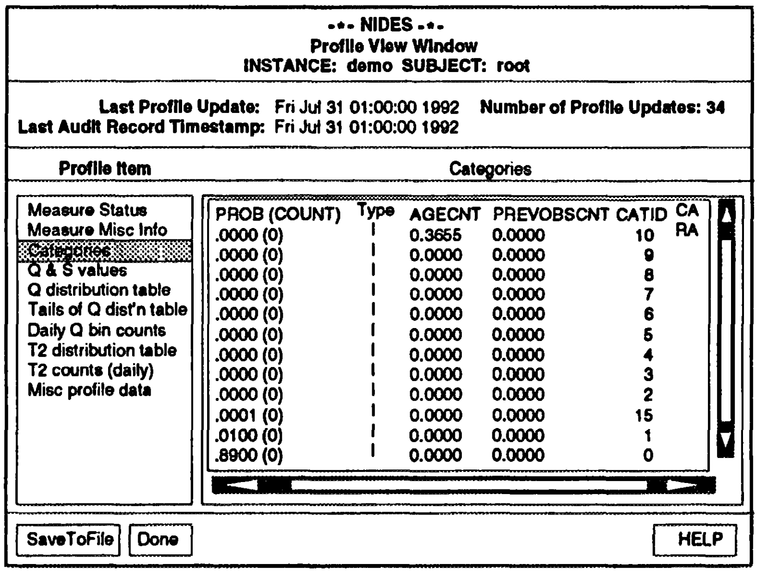
Measure Name	Code	Type	Description
U_CPU	CPU	Continuous	CPU time used
U_IO	IO	Continuous	I/O activity
U_MEM	MEM	Continuous	Memory usage
U_LOC	LOC	Categorical	TTYs used by subject
U_MAIL	MAIL	Categorical	Mail programs/commands used, categories for this measure are the MAIL class list members
U_EDIT	EDIT	Categorical	Editor programs/commands used, categories for this measure are the EDITOR class list members
U_COMPILER	COMPILE	Categorical	Compiler programs/commands used, categories fo this measure are the COMPILER class list membe
U_SHELL	SHELL	Categorical	Shells used (f.e., csh,sh,tsh), categories for this measure are the SHELL class list members
U_WINDOW	WINDOW	Categorical	Window programs/commands used, categories for this measure are the WINDOW class list member
U_COMMD	CMD	Categorical	Commands invoked
U_COMMDB	CMDB	Binary	Notes if a command was invoked
U_COMMDC	CMDC	Binary	Notes if a new command was invoked
U_SYSCALL	SYSCL	Categorical	UNIX system calls invoked
U_DIR	DIR	Categorical	Directories accessed
U_DIRB	DIRB	Binary	Notes if a directory was accessed
U_DIRNEW	DNEW	Binary	Notes if a directory was created
U_DIRDEL	DDEL	Binary	Notes if a directory was deleted
U_DIRMOD	DMOD	Binary	Notes if a directory was modified
U_DIRREAD	DREAD	Binary	Notes if a directory was read
U_FILENEW	FNEW	Binary	Notes if a file was created
U_FILEREAD	FREAD	Binary	Notes if a file was read
U_FILEMOD	FMOD	Binary	Notes if a file was modified
U_FILEDEL	FDEL	Binary	Notes if a file was deleted
U_FILETMP	FTMP	Binary	Notes if a temporary file was accessed
U_FILE	FILE	Categorical	Files accessed
U_FILEB	FILEB	Binary	Notes if any file was accessed
U_UID	UID	Categorical	User IDs used
U_UIDB	UIDB	Binary	Notes is a new user ID was used

Table 4.1: Statistical Component Measure Descriptions (part 1)

Measure Name	Code	Type	Description
U_SYSERR	SYSRR	Binary	Notes if a system error occurred
U_SYSERRTYP	SYSERRTYP	Categorical	System error codes encountered
U_AUDREC	AUDREC	Categorical	Not used in this NIDES release
U_HOUR	HOUR	Categorical	Time of activities
U_HOURB	HOURB	Binary	Notes change in hour of activities
U_DAILY	DAILY	Categorical	Usage by day of week
U_DAILYB	DAILYB	Binary	Notes change in day of usage
U_RNET	RNET	Binary	Notes if activity is remote
U_RNETTYP	RNETTYP	Categorical	Remove network activity
U_RNETHOST	RNETHOST	Categorical	Remote hosts used
U_LNET	LNET	Binary	Notes if activity is local
U_LNETTYP	LNETTYP	Categorical	Local network activity
U_LNETHOST	LNETHOST	Categorical	Local hosts used
U_INTARR	INTARR	Continuous	Audit record interarrival times
U_FCLASS	FCLASS	Categorical	Classes of files accessed
U_FCLSRD	FCLSRD	Binary	Notes a file class read violation
U_FCLSWR	FCLSWR	Binary	Notes a file class write violation
U_ARECDIST	ARECDIST	Categorical	Keeps track of which measures have been observed with each audit record. The categories for this measure are the 49 measures.
U_INT60	INT60	Continuous	1-minute audit record intensity(volume)
U_INT600	INT600	Continuous	10-minute audit record intensity(volume)
U_INT3600	INT3600	Continuous	1-hour audit record intensity(volume)

Table 4.2: Statistical Component Measure Descriptions (part 2)

high. Figure 4.2 represents an example of a histogram plotted from the category probabilities as obtained from the categories viewed via the profile viewing option.



```
                    -*- NIDES -*-
                 Profile View Window
            INSTANCE: demo  SUBJECT: root

       Last Profile Update:  Fri Jul 31 01:00:00 1992   Number of Profile Updates: 34
Last Audit Record Timestamp: Fri Jul 31 01:00:00 1992

    Profile Item                         Categories

Measure Status          PROB (COUNT)  Type  AGECNT  PREVOBSCNT  CATID  CA
Measure Misc Info       .0000 (0)      I    0.3655  0.0000        10   RA
Categories              .0000 (0)      I    0.0000  0.0000         9
Q & S values            .0000 (0)      I    0.0000  0.0000         8
Q distribution table    .0000 (0)      I    0.0000  0.0000         7
Tails of Q dist'n table .0000 (0)      I    0.0000  0.0000         6
Daily Q bin counts      .0000 (0)      I    0.0000  0.0000         5
T2 distribution table   .0000 (0)      I    0.0000  0.0000         4
T2 counts (daily)       .0000 (0)      I    0.0000  0.0000         3
Misc profile data       .0000 (0)      I    0.0000  0.0000         2
                        .0001 (0)      I    0.0000  0.0000        15
                        .0100 (0)      I    0.0000  0.0000         1
                        .8900 (0)      I    0.0000  0.0000         0

[SaveToFile] [Done]                                            [HELP]
```

Figure 4.1: Category View Screen when Value of Scalar Too High

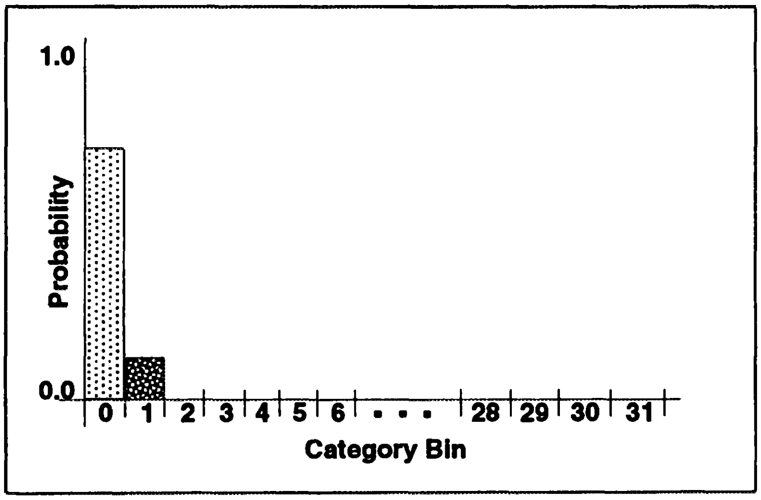
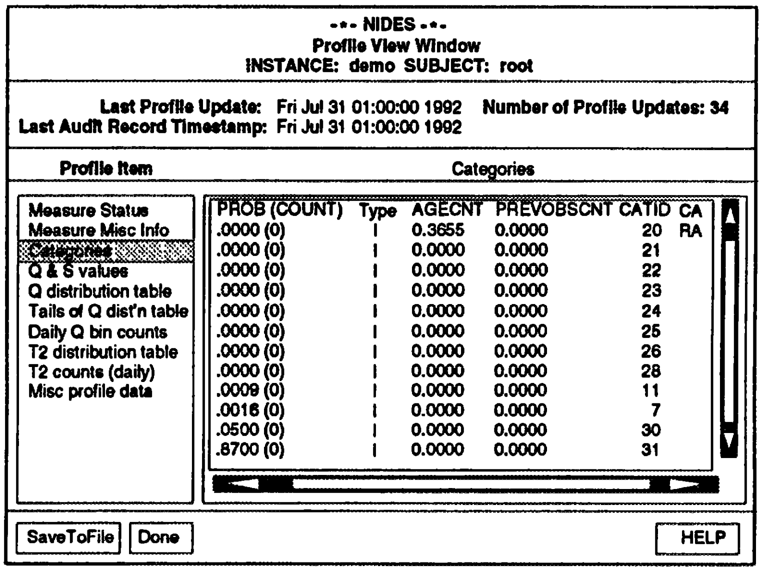


Figure 4.2: Category Histogram with Value of Scalar Too High

Conversely, if the Scalar is too low, the category probabilities will crowd the upper end of the distribution, as shown in Figure 4.3 and plotted in Figure 4.4. These examinations should be made on the profiles no sooner than the end of the " C" training phase.

It is not necessary that all categories be populated for good detection performance, merely that neither extreme has too much probability. Therefore, we recommend you do not change the Scalar unless either of the histogram types, shown in Figures 4.2 or 4.4, is evident after

C training completes for a number of subjects. Changing the Scalar value requires full retraining for the measure modified.



```
-*- NIDES -*-
Profile View Window
INSTANCE: demo SUBJECT: root

        Last Profile Update:  Fri Jul 31 01:00:00 1992   Number of Profile Updates: 34
Last Audit Record Timestamp:  Fri Jul 31 01:00:00 1992

Profile Item                              Categories

Measure Status           PROB (COUNT)  Type  AGECNT  PREVOBSCNT CATID  CA
Measure Misc Info        .0000 (0)      I    0.3655  0.0000       20   RA
Categories               .0000 (0)      I    0.0000  0.0000       21
Q & S values             .0000 (0)      I    0.0000  0.0000       22
Q distribution table     .0000 (0)      I    0.0000  0.0000       23
Tails of Q dist'n table  .0000 (0)      I    0.0000  0.0000       24
Daily Q bin counts       .0000 (0)      I    0.0000  0.0000       25
T2 distribution table    .0000 (0)      I    0.0000  0.0000       26
T2 counts (daily)        .0000 (0)      I    0.0000  0.0000       28
Misc profile data        .0009 (0)      I    0.0000  0.0000       11
                         .0016 (0)      I    0.0000  0.0000        7
                         .0500 (0)      I    0.0000  0.0000       30
                         .8700 (0)      I    0.0000  0.0000       31
```

Figure 4.3: Category View Screen when Value of Scalar Too Low

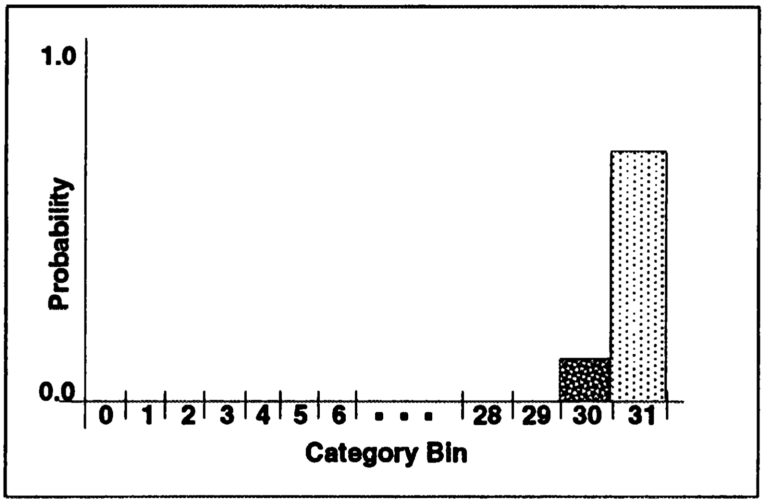


Figure 4.4: Category Histogram with Value of Scalar Too Low

4.6.3 Qmax

The parameter Qmax plays the role for the Q distribution that Scalar does for the category distribution, except that it is defined for all measures, not just continuous ones. Typical Q values are less than 10 for most measures, but they can be considerably higher for measures for which the short-term profile consists of one category exclusively. For example, the "hour of use" (U_HOUR) measure has 24 categories, and typically most users have a near-uniform distribution in the 10 or so hours spanning the normal work day and a scattering at other hours. At any given time, however, the short-term profile might contain only one category — the current hour. For these measures, Q can be as high as 100. Again, based on the robustness of the binning mechanism, we set Qmax values high and recommend a Qmax of 100 for most measures and 200 for measures such as U_HOUR. These are reflected in the default configuration Qmax values for the measures.

The symptoms indicating that Qmax is set too high or too low are that the Q Probability histogram has the appearance of the category histogram (see Figures 4.2 and 4.4). As with the measure Scalar value, Qmax should be changed only if the observed histogram is extremely off for a large number of subjects. Changing Qmax requires Q retraining for the measure.

4.6.4 Minimum Effective-N

The user may prevent a measure from contributing to the score even for trained profiles through the use of the Minimum Effective-N feature. The Minimum Effective-N for a measure represents the minimum number of observations, modified by aging factors, that must be observed for the measure before the measure can contribute to scoring regardless of the measure's training status. This prevents the first observation of a new category for a rarely seen measure from skewing the overall T2 score (for example, a user might go months before using the system from a remote host). A value of, say, 100 means that 150 to 200 observations have to be made (due to the aging mechanism and depending on how these observations are scattered throughout the training period) before a measure contributes to the overall score, irrespective of its training status. This value may be reduced somewhat for rarely observed measures, although too low a setting may lead to a scoring for the measure that is based on very few observations.

4.6.5 Short-term Half-life

The short-term half-life is a measure-specific parameter that controls the time window reflected in the short-term profile. The larger the value, the longer the time period represented in the short-term profile and the more stable the category counts. A smaller short-term profile half-life value reflects a shorter time interval and potentially provides more timely detection, but the category counts reflected in the short-term profile might be less statistically stable. The short-term half-life for a measure should be set to something like 5% of a typical user's daily audit record activity for the measure in question. For example, if 2000 of a typical user's audit records are relevant (i.e., contribute) to the CPU measure on a typical day the short-term half-life for the CPU measure should be about 100. The time window for the short-term profile for this measure is thus on the order of a half an hour (assuming most activity is concentrated in an 8- to 10-hour period). Unusual activity can nudge the detection score up above the threshold in a fraction of this time window (a very small fraction if the activity is sufficiently unusual). Over a broad range of values, the effective profile length (in number of audit records) is approximately one and one half times the half-life value. Setting the half-life too high might dilute intrusions that generate a very small number of audit records, while setting it too low might result in a false alarm problem. The exact value is not critical, and users may simply use the same value for most or even all measures. This means that the short-term profile may reflect significantly longer time scales for rarely seen measures. Changing this value requires retraining of the Q and T2 stages.

4.7 Profiles

NIDES maintains a compact statistical representation of the observed behavior for each subject in two profiles — a long-term (historical) and a short-term (current) profile. The long-term profile is an adaptive structure that is "trained" to a subject's long-term behavior patterns and is modified to follow shifts in user behavior through the profile update mechanism. The short-term profile tracks a subject's activity over shorter time windows (minutes to hours). A description of the components of the profiles and how and when a user would manipulate them is given below.

4.7.1 Viewing

NIDES enables the user to view profiles during and after training. See Section 6.5.3.2.4 on page 139 for a complete description of the profile viewing function.

Profile viewing provides a useful diagnostic to the experienced NIDES user. The user can ascertain training status at a glance, and by examining category distributions can determine if any rescaling is needed. (Section 4.6.2 discusses adjustment of the scaling parameter.) The View option presents the user with a summary showing the time of the last long-term profile update, the timestamp of the last audit record processed through the profile, and the number of profile updates to date; it also allows the user to examine various components of the profile, which are discussed next.

The user is advised against changing any rescaling parameters (measure Scalar and Qmax) until the underlying training phase (C for the measure Scalar value, Q for the measure Qmax) is complete.

4.7.1.1 Measure Status

The *Measure Status* option in the Profile View window brings up a display of the measures in the profile, their status, the number of updates to go in the current training phase, the remaining training phases, and the effective number of observations for the measure. Reviewing measure status can tell you which measures have become active (i.e., are contributing to score calculation and anomaly reporting), and the activity for each measure. Measures that are trained are indicated by a READY status.

4.7.1.2 Measure Misc Info

The *Measure Misc Info* option shows the aged number of categories, the sum of category probabilities for measures in the RARE group, the highest probability in the RARE group, and the next available category (always 34 for continuous measures). These entries are not reliable until the C training phase is complete. The sum of category probabilities in the RARE group should be less than or equal to the *Max Sum of Rare Cat Probs* parameter. Increasing or decreasing this configurable parameter (and we recommend only slight changes) can increase or decrease the number of categories that are considered RARE.

4.7.1.3 Categories

The *Categories* option shows categories for each measure. Continuous measures have a fixed number of category bins — 32. For continuous measures, the bulk of the probability should be concentrated in the middle bins (roughly bins 10 to 20). Probability concentrated in the lower bins indicates that the Scalar value for the measure is set too high, while concentration at the high end indicates that the Scalar is set too low. A Scalar value set somewhat high is preferable, since the same value applies to all subjects. For the file measure, uninteresting categories with high probabilities are candidates for addition to the temporary file class.

For continuous measures, the category name and id match the category index; for categorical measures, the actual name (e.g., a file name) is given. For categorical measures, the special categories OTHERCAT (used to score a new category not in the long-term profile) and DROPCAT (an aged cumulative total of category probability for dropped categories) are defined to enable the system to properly track new activity and obsolete activity that should be dropped from the long-term profile.

4.7.1.4 Q and S Values

The *Q and* S *Values* view option shows the chi-square-like difference between the recent and historical past (Q, discussed in Sections 4.1.3 and 4.2.1) and the half-normal transformation of Q based on the historical Q distribution (S, discussed in Sections 4.1.4 and 4.2.1). Consistently high or low Q values may indicate a need to change Qmax (and may also indicate unusual behavior for the measures involved). The user is advised against changing any measure scaling parameters (i.e., Scalar and Qmax) until the underlying training phase (C for the measure Scalar value, Q for the Qmax) is complete. The overall score, before normalization, is the sum of the squares of the S values. The system limits S values to 4.0; values for S approaching this should be considered suspicious.

4.7.1.5 Q Distribution Table

The *Q Distribution Table* option shows the empirical distribution of Q as binned into ranges based on the Qmax for each measure. Information in the table should not be considered reliable until Q training is complete. As with the Scalar for continuous measures, there should not be a concentration of probability in either the high or the low numbered bins. An extreme concentration at the low end indicates that Qmax can be lowered, while concentration at the high end indicates it should be raised. Since the same value applies to all subjects, the NIDES user should be sure that the value is significantly misscaled for the entire group of subjects before changing it. NIDES functions satisfactorily with a value that might be somewhat high for most subjects but successfully contains the data for extreme cases.

4.7.1.6 Tails of Q Distribution Table

The *Tails of the Q Distribution Table* option shows the same bins as the Q distribution table, but now each bin entry contains a tail probability rather than a bin probability. That is, the value for each bin is the sum of bin probabilities for that bin and all bins to the right. It is used by NIDES in the half-normal transformation of the Q value. The ideally scaled Q distribution has half its probability concentrated in bins 16 and above. This corresponds to a tail probability of 0.5 for bin 16 in the tail distribution. If the median bin (the first bin for which the tail probability is 0.5 or lower) is somewhere between the tenth and twentieth bins for most subjects, no change to Qmax is indicated.

4.7.1.7 Daily Q Bin Counts

The *Daily Q Bin Counts* option shows the counts in each Q bin from which the Q probability distribution is updated (the counts for each day are folded in with the historical counts after the latter are aged, and the total is converted to a probability distribution). This table usually contains all zeros, which is normal right after a profile update, except in the case where a snapshot of the profile is taken between update periods — that is, usually when a profile is swapped out of the profile cache. In this case, high counts for bins in which the tail probability is low lead to high S values and eventually high scores.

4.7.1.8 T2 Distribution Table

The *T2 Distribution Table* option shows the empirical distribution of the score values. Until training is completed, T2 is a distribution with all its probability in the first bin; afterwards, it reflects the actual probability. It is binned from 0 to 20 in increments of 0.1, and from 20 to 200 by increments of 1.0. As an example, bin 151 corresponds to T2 values between 15.0 and 15.1. The subject-specific thresholds are calculated from this distribution and the red and yellow threshold percentages; the distribution is examined from the high end to find the bins bracketing the desired percentage. The score value is then interpolated, and observations for which the T2 score exceeds this value represent a critical (in the case of the red threshold) or warning (in the case of yellow) status for the audit record.

4.7.1.9 Daily T2 Counts

The *T2 counts (daily)* option shows the counts in each T2 bin from which the T2 probability distribution is updated (the counts for each day are folded in with the historical counts after the latter are aged, and the total is converted to a probability distribution). As with the daily Q counts, this table will contain all zeros, except when the profile snapshot was taken between update periods — that is, usually when a profile is swapped out of the profile cache.

4.7.1.10 Miscellaneous Profile Data

Until training is complete, the yellow and red thresholds are set to the arbitrary values 2.0 and 3.0, respectively. These values are not actually used for scoring, since scoring does not occur until training is completed. After training is completed, the yellow and red thresholds are set to those values from the empirical T2 distribution that are exceeded by the percentage of scores equal to the user-configured yellow and red detection percentages; some interpolation is done for values between the T2 distribution bins.

4.7.2 Copying

The copy option enables the user to copy an existing subject's profile to a new subject. This is useful if you want to establish a trained profile for a new subject quickly. However, this can result in a very high initial false alarm rate until the profile adapts to the new subject's usage pattern.

4.7.3 Replacement

The profile replacement feature is useful to set up cross-profiling experiments where the data for one subject is processed through the profile of another — see Section 4.9.2 and [3]. Such experiments provide data on NIDES's ability to detect "true positives" (i.e., data labeled as belonging to the wrong subject). If desired, the target profile may be copied beforehand to a new (nonexistent) subject profile, and replaced after the experiment. Profile replacement can also be accomplished by deleting the target profile and then copying the source profile to the target as if it were a new profile, but the replacement operation saves a step.

4.7.4 Deletion

The NIDES user may want to delete a profile for a subject who is no longer an authorized user of the monitored system(s). When running experiments, profile deletion can also be used to control the data used to train some or all profiles. Additional data for subjects whose profiles have been deleted appears to NIDES as data from a new subject (with all the attendant training implications). This feature should be employed cautiously.

4.8 Long-term Profile Updating

NIDES allows you to configure long-term profile updating in several ways. For real-time analysis you can have updates performed on a user-defined schedule based on either the audit record timestamps or the system clock. Long-term profile updating can be turned ON or OFF. You may also initiate an instantaneous long-term profile update for selected subjects under real-time analysis. When a long-term profile update occurs, the new long-term profiles are saved to disk and the short-term profiles are also checkpointed (i.e., written out to the disk).

4.8.1 Real-time Profile Updating

For the real-time instance, updates of the long-term profile may be triggered according to the NIDES host system clock or the timestamp on the arriving audit records. Updating based on the NIDES host system clock ensures that long-term profiles are updated once daily. As updating is somewhat compute-intensive, it is recommended that updates be scheduled for times of low activity on the NIDES host (such as during the night or early morning).

The user may turn off long-term profile updating for one or any subset of the subject list. Updating of the long-term profiles can be suspended in anticipation of a major discontinuity in system activity. For example, if a user will undertake legitimate activity that would normally trigger alerts (such as installing software in certain system directories), the security officer might turn off both alert reporting and updating of the long-term profile.

4.82 Profile Updating during Experiments

For NIDES test instances, long-term profile updates take place according to audit record timestamps. Updates always happen at the first occurrence of an audit record past midnight for any subject. At this time, all profiles are updated (even though it is likely that the received audit records for most subjects have not yet crossed the date boundary). Long-term profile updating can be turned OFF for all subjects. Under the test facility, this means that the long-term profiles with which you started your test will remain unchanged for the duration of the test. Users running experiments assessing detection performance only (for example, in cross-profiling experiments — see Section 4.9.2) should disable long-term profile updating. This will ensure that your profiles are not modified during the experiment. If you are using the test facility to train a set of long-term profiles, make sure profile updating is switched ON.

4.8.3 Manual Updates of Long-term Profiles

The user can force an immediate update of long-term profiles at any time in the real-time instance. This can be used to force pending configuration items to take effect. An update may, however, take a considerable amount of time (particularly on systems with many profiles).

Remember that NIDES bases its training period on the number of long-term profile updates, but invoking a large number of immediate updates to accelerate training is not recommended as these profiles will generally be less stable than those representing an appropriate time interval.

The updater updates valid categories in the long-term profile and drops categories whose probability is below some threshold. Sometimes a subject generates a very large number of categories for a measure (typically a file-related measure), each of which has only one or a small number of counts. These can be cleared out of the short-term profile by invoking a manual profile update for the subject.

4.9 Statistical Component Experiments

The NIDES test facility provides the user with a powerful tool to assess NIDES detection performance and to guide NIDES configuration in a specific environment. This facility enables the user to train profiles quickly from archived data using different parameter settings, coupled with the ability to replace or remove profiles. Most experiments attempt to provide an estimate of the false positive detection rate, the true positive detection rate, and system throughput. For a description of some statistical experiments performed at SRI, see [3].

4.9.1 False-positive Detection Rate Experiments

The false-positive detection rate is that percent of audit records flagged at either the yellow or red threshold for a subject when run against that subject's profile. With a stable, trained system, the declarations above the yellow or red threshold should be approximately equal to the respective user-configured threshold percentages. The user might run several experiments using different values for the parameters to control the profile training period and short-term half-life by splitting the available data into a training set and test set (i.e., audit data not seen by the system during the training phase). The training set should be long enough to include a number of updates at least equal to the user-configured number of training days. Profiles are trained using this set, and then records from the second set are run through these trained profiles. The test will report both the number of alerts as declared by the resolver as well as the number of records above each of the respective thresholds. It is not unusual to observe a higher than nominal false positive rate for some users.

4.9.2 True-positive Detection Rate Experiments (Cross-profiling)

The true-positive detection rate is that percent of audit records flagged at either the red or yellow threshold for a subject when run against the profile of another subject. Ideally, the new data should represent an attempted intrusion, but typically such tests are carried out by running archived data for one authorized user against the profile of another. This is known as a cross-profiling experiment. To run such an experiment, the NIDES user would use the profile management facility and replace the profile for a subject with that of a second subject. Data for the first subject would then be scored against a profile reflecting activity patterns for the second subject. Ideally, the detection rate in such an experiment should be much higher than the false-positive detection rate. It is often observed that true-positive rates are often very asymmetric; for example, user A's data through user B's profile might generate 15% detections, while user B's data through user A's profile might generate 80%.

The NIDES test facility not only provides the capability to replace profiles, but also allows the NIDES user to copy the replaced profile to that of a new subject, from which it can be restored after the experiment. For cross-profiling experiments, it is recommended that profile updating be turned off.

4.9.3 Evaluating System Performance

The test facility also provides a display of the count of audit records processed as the test is running. This enables the NIDES user to estimate throughput in terms of number of audit records per second. Running the same data through with a different filter for the file class measure or different cache size can affect this throughput, at times dramatically. Experimentation manipulating these parameters can guide the NIDES user to optimal settings in a given environment.

The test facility can be used to explore these and other issues affecting NIDES, such as optimal settings of configuration parameters other than those cited above. It can also be used to replay data for which NIDES performance was other than expected, perhaps after changing some configuration parameters to better tune the system. The utility of the test facility is limited only by the availability of archive data and the imagination of the user.

Chapter 5

Rulebased Analysis Configuration

The rulebased component of NIDES includes 39 rules that generate alerts and many additional rules that support case building and fact maintenance. In this version of NIDES, the rulebased component can have new rules added dynamically at run time. Writing and installing new rules in NIDES is similar to programming. The process requires thoughtful preparation and care to ensure that rules added to the NIDES rulebase provide increased detection functionality without a degradation in rulebase performance or in the accuracy of other rules in the rulebase.

Before you add new rules to the NIDES rulebase, read this chapter on rulebase development. When you have determined that the existing NIDES rulebase does not address a vulnerability in your environment, follow these steps to introduce a new rule or rules into the NIDES environment:

1. Review the `rb_config` file documentation in Section 5.4 to determine if a change to the `rb_config` file can address your needs. A change to the `rb_config` file is much easier to perform and is less prone to introduce errors than the adding of new rules.

2. Determine the scenario you want your new rule or rules to address. Also determine how the audit trail will provide the data needed to recognize the scenario. In some cases your audit trail may be deficient and cannot provide the data needed for recognition of a particular scenario. If this is the case, a new rule cannot address your scenario.

3. Once you have determined the new rule's scenario and where in the audit trail the relevant data will be found, write up a prototype rule (or, if needed, rules) to address the scenario. Compile the rules using the NIDES `makerule` script.

4. Collect some audit data that includes the scenario you are attempting to capture. If you can provide more than one variation of this audit data, that is even better.

5. Run some experiments using your new rule(s) and your sample audit data.

6. If the experiment results are satisfactory (i.e., the planted "bad" behavior is correctly detected but false alarms aren't generated), introduce the new rule(s) into your real

time NIDES operation. If your results are not acceptable, review the steps taken and see how your rules can be refined/modified. Also review your audit data to be sure the scenario is accurately represented.

Read Section 5.5 on the default rulebase to familiarize yourself with the capabilities NIDES provides initially. If you decide to write new rules, review Sections 5.1 and 5.2 before you write and introduce them. Be sure to review how the `rb_config` file works (Section 5.4). Much customization can be achieved by "fleshing out" the `rb_config` file to match your environment. Section 5.3 is a tutorial introduction to rule writing; it guides you through the writing of a simple rule, compiling it, and installing it so NIDES can use it. If you do write new rules, be sure to test them by using the test environment NIDES provides before introducing them into your real-time analysis.

This release of NIDES does not include any automatic mechanisms for protection of your rulebase from unauthorized tampering or reverse engineering. Guidelines for protecting the rulebase are presented in Section 9.4.2.3. You should review and follow these guidelines when working with the NIDES rulebase.

5.1 Writing Rules

Writing rules for the NIDES rulebase is like programming — you'll need to know how to control the flow of execution, how data is managed during execution, and the basic syntax of the rulebase language.

5.1.1 Rulebase Concepts and Terms

If you have never worked with a rulebased system, some of the terms used here may be unfamiliar:

Antecedent (Rule Antecedent) The first of the two parts that comprise the body of a NIDES rule. The antecedent contains the tests that are performed on the rulebase's factbase to determine if a particular condition is met.

Consequent (Rule Consequent) and Rule Firing The second of the two parts that comprise the body of a NIDES rule. The consequent contains the actions that are performed if the tests performed in the rule's antecedent are satisfied. If the consequent actions are executed, the rule is said to have "fired". Actions that may be performed in the consequent of a rule include additions or deletions to the rulebase's factbase and generation of an alert report.

Facts, Ptypes and the Factbase The NIDES rulebased component stores transitory information needed for its analysis in *facts*. Facts are stored in a database we call the *factbase*. The structures for facts are defined by *ptype* declarations. A ptype declaration is similar to a C language structure declaration. Multiple facts of the same type can be contained in the factbase. When a rule searches the factbase for a fact type that contains multiple entries, the most recently asserted fact that matches the rule search specification will be returned to the rule.

Assertion and Deletion of Facts When a rule wants to add a new fact to the factbase, it does *so* by *asserting* the fact. If a fact needs to be removed from the factbase, a rule does so by *deleting* the fact.

A NIDES rulebase rule contains two basic parts:

- **Rule Heading** — declares the rule name, priority and any special operating modes
- **Rule Body** — defines what the rule does. The body of a NIDES rule has two parts:
  - **Antecedent** — defines the tests that the rule performs
  - **Consequent** — defines the actions the rule will perform if the antecedent tests are satisfied

5.1.2 Rulebase Execution Flow

The NIDES rulebase is structured to analyze audit records one at a time. The overall execution flow is

1. Assert audit record (i.e., event fact).
2. Process audit record.
3. Remove audit record from factbase.
4. Go to 1.

5.1.2.1 Audit Record Assertion

For each audit record sent to NIDES, the rulebase analyzes the audit record to determine if an anomaly should be reported. This is done by asserting a fact of type *event* into the factbase (Table 5.2 shows the structure of event facts). Audit records asserted into the factbase are received from the arpool process.

5.1.2.2 Audit Record Processing

Once an event fact has been asserted into the factbase, each currently active rule examines the event fact to determine if it can use the fact to satisfy its test conditions. Rules are ordered by priority; those with a higher priority see the event fact before rules of lower priorities.

5.1.2.3 Removal of Audit Records

The rulebase allows only one event fact in the factbase at a time. Additional audit records are not accessed until all rules have completed examination of the current audit record's event fact. When all rules have examined the event fact, a special rule removes it. At this point the rulebase reads another audit record, if one is available, and asserts it into the factbase as an event fact.

5.1.3 Facts and Ptypes

Since audit records are not stored permanently in the factbase, any inference requiring information from multiple audit records requires that the information be saved into facts that are preserved after the audit record has been deleted. The templates that describe the structure of facts are called ptypes. Note that all ptype definitions are built into the NIDES rulebase libraries. You may not declare any ptypes in your rule files. However, you may use several predefined ptypes, which are described in Section 5.1.4.

The following is an example of a ptype declaration:

```
ptype [count value:int]
```

The purpose of this declaration is to establish a pattern or template for facts. Each fact that exists in the expert system's factbase is an instance of some ptype. Facts of ptype *count* have one field, an integer called *value.* This declaration allows rules to refer to count facts, and to inspect and modify the value field of these facts. There are usually many facts of a given ptype.

The C programmer may find it helpful to think of ptype declarations as analogous to structure declarations. In the same way that there can be many instances of a single structure in a C program, there can be many facts of the same ptype in a NIDES factbase.

Like a C structure, a ptype may have more than one field:

```
ptype[session  userid:string,
               terminal:string,
               timeoutflag:int]
```

This ptype declaration establishes a pattern for facts with three fields: *userid* and *terminal,* both strings, and *timeoutflag,* an integer.

5.1.3.1 Factbase Testing

Rules can refer to facts that have fields matching particular things. For example, a rule could check for a session fact whose timeoutflag field is 1 by including the following clause in its antecedent:

```
        [+session|timeoutflag == 1]
```

The + sign after the opening bracket is used as an "existential quantifier". That is, it allows a rule to determine if any fact having certain characteristics exists. This clause, then, will match any session fact having a timeoutflag field with the value 1. Think of it as meaning "For all sessions whose timeout flag equals 1..."

A rule may also determine if there is no fact of a given type using the - sign. This type of test can be qualified by including tests on the ptype fields. The following example determines if there is no session fact with the userid field equal to "THISUSER":

```
[-session|userid == "THISUSER"]
```

A rule can also identify, or "alias", facts as follows:

```
[+se:session|timeoutf lag == l]
```

This clause tries to match a session fact whose timeoutflag field is 1. Once it matches such a fact, the fact that matched will have the name 'se' for the duration of the rule (if there are many matching facts, the most recently added or modified one is used). The rule can then refer to the fields of that particular fact later on:

```
[?|se.userid == "THISUSER"]
```

This clause determines whether the fact named 'se' has a userid field equal to "THISUSER". The above two clauses in conjunction make the rule try to find a single session fact whose timeoutflag field is 1 and whose userid field is "THISUSER". It is not necessary to use two separate clauses; instead you can do the following, which would check both fields:

```
[+se:session|timeoutflag == 1, userid == "THISUSER"]
```

5.1.3.2 Asserting and Deleting Facts

Both the + and - tests have corresponding assert and delete actions that appear in consequents. If a rule wants to assert a new fact of a given ptype, it uses the + action. For example, a rule asserting a new *count* fact whose *value* field is 0 would include the following clause in its consequent:

```
[+count|value = 0]
```

Notice the C-like use of the equal sign. In antecedents, tests for equality use the == (double-equal) sign, while in consequents, assignments use the = (single-equal) sign.

The assert action must specify an initial value for every field in the ptype for that fact.

If a rule wants to delete a fact from the factbase, it must first alias the fact in its antecedent, and then use the negation action on the alias in its consequent. The following is an example:

```
rule[SimuLogon(#l;*):
          .
          .
        [+tr:trans]         ' find and alias the fact we want to negate
          .
          .
==>
        [-tr]               ' delete matched fact
        etc.
```

A rule can also modify one or more fields in a fact. Again the rule must find and give an alias to the fact in its antecedent, and then operate on the fact in its consequent using the /operator. Here is an example:

```
rule [r1:
        [+c:count|value<100]   ' Find a count fact with value field < 100
         .
         .
==>
        [/c|value+=1]          'Increment count fact's value field
        etc.
```

Modifications are implemented as if the original fact had been deleted and a new fact with the modified fields had been asserted. This allows the system to give priority in binding to modified facts over facts that were created more recently but are "inactive".

5.1.3.3 Factbase Maintenance

Facts can be added to the factbase by any rule that needs to save information across multiple audit records. However, this process cannot continue indefinitely; these intermediate facts must be removed from time to time.

There are three reasons to remove facts. First, doing so prevents a rule from firing over and over with the same facts satisfying it. Second, removing facts from the factbase when they are no longer needed improves system performance, because, as the number of facts in the factbase grows, the overall throughput decreases. This is due to two factors. First, facts are arranged in lists, and as these lists grow, the time to traverse them becomes more significant. Second, when a rule does inter-fact testing, it can end up comparing all facts in each of its fact lists with all facts in all the other lists (i.e., making $m \times n \times o \times \ldots$ comparisons, where m is the number of facts in the first list, n is the number in the second, o the number in the third, and so on). This effect is most significant when the rule fails to fire; however, note that most rules fail to fire the vast majority of the time.

To minimize these effects, several approaches are taken. First, only one audit record is maintained in the factbase at a time, minimizing the length of the list of the type of fact that is accessed most often, the event fact. Second, most rules start out by checking for an event fact, as opposed to checking for other facts first; this minimizes list traversal. Finally, facts are removed from the factbase whenever possible. Every type of fact that would ordinarily be added during execution should have a rule that removes it when it is no longer needed.

A third reason to remove facts once they are not needed is to conserve memory. Obviously, if the system is to run for long periods of time, things must be removed from the factbase at the same rate, overall, as they are added to it, or the factbase will ultimately overflow the memory of the system. For example, when a login event comes in, the rulebase creates a session fact to store information about the user who has logged in. If the user is inactive for a long period of time, the session fact is removed.

Note that if a fact can match an antecedent clause in more than one rule, no rule should remove it unless all the rules that may need it have used it.

5.1.4 NIDES Default Fact/Ptype Descriptions

The NIDES rulebase has three ptypes that define facts you can use in rules that you write:

- **event** – This ptype structure defines the fact template for audit records. Every time the rulebase processes a NIDES audit record, this processing begins with the assertion of an event fact. It is likely that most if not all the rules you write will access this fact, probably as the first test in the antecedent clause. Event facts should not be asserted or deleted by any rules you write. You may, however, apply marks to the event fact, which is usually required to prevent rules that have already seen the event fact from firing repeatedly (Section 5.1.6 discusses marks). The NIDES rulebase has default rules that manage the event facts, so there is only one event fact in the factbase at one time. The `remove_event` rule, which has a priority of -97, the lowest priority of all rules, removes the event fact after all rules have had a chance to see it. The event ptype structure was modified for the Beta-update release. If you have rules written under earlier NIDES releases you will need to update any references to event facts to match the new event ptype template. Table 5.1 shows the earlier structure for the event ptype and the current structure for the event ptype is shown in Table 5.2.

- **generic** – This is the only fact type that you can assert, modify or delete. None of the NIDES default rules utilize this fact template. The structure for the generic ptype is shown in Table 5.3.

- **generic_config** – This is the only fact type that you can configure via the `rb_config` file. You should not assert, modify (including marks), or delete facts of this type in any of your rules. This fact type is initialized via entries in the `rb_config` file. The structure for the generic ptype is shown in Table 5.4. See Section 5.4 for more information on this fact type.

PTYPE event

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'    This ptype is the fact asserted when the rulebase reads '
'    a NIDES audit record.    '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
ptype[event
         targid:string,           ' local (target) host name
         real_userid:string,      ' audit user name (never changes)
         current_userid:string,   ' effective user id for this record.
         otheruser:string,        ' other user (for su actions)
         file:string,             ' File name (if present)
         action:ia,               ' NIDES audit data action code.[a]
         response:int,            ' any processes response
         rhost:string,            ' remote host name
         term:string,             ' tty
         process_id:int,          ' Process id (unix).
         cmd:string,              ' Nides command field.
         cputime:float,           ' process cpu time.
         audit_src:src,           ' Source of the data.
         hi_sequence:int,         ' The sequence number is a 64-bit
         lo_sequence:int,         ' quantity, or 2 32-bit fields.
         timerec:ptime,           ' time audit record received by IDES
         timegen:ptime            ' time record was generated by target
]
```

[a]See Table 5.5 for a list of NIDES action codes and Table 6.6 for a description of the action codes.

Table 5.1: Ptype event For Older Releases (Pre Beta-update)

PTYPE event

```
‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
‘    This ptype is the fact asserted when the rulebase reads ‘
‘    a NIDES audit record.    ‘
‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘‘
ptype[event
rseq_hi:int,            ‘ The arpool sequence number is a 64-bit (was hi_sequence
)
rseq_lo:int,            ‘ quantity, or 2 32-bit fields (was lo_sequence)
recvtime:ptime,         ‘ Time audit record received by IDES (was timerec)
tseq_hi:int,            ‘ The target sequence number is a 64-bit (new field)
tseq_lo:int,            ‘ quantity, or 2 32-bit fields (new field)
atime:ptime,            ‘ time record was generated by target (was timegen)
hostname:string,        ‘ local (target) host name (was targid)
audit_src:src,          ‘ Source of data (was audit_src)
action:ia,
auname:string,          ‘ audit user name (never changes) (was real_userid)
uname:string,           ‘ was (current_userid)
pid:int,                ‘ Process id (UNIX) (was process_id)
ttyname:string,         ‘ tty (was term)
cmd:string,
argcount:int,           ‘ New field
arglist:string,         ‘ New field
syscall:int,            ‘ New field
errno:int,              ‘ Was field response
rval:int,               ‘ New field
res_utime:float,        ‘ Was field cputime
res_stime:float,        ‘ New field system time
res_rtime:float,        ‘ New field
res_mem:float,          ‘ Memory (New field)
res_io:float,           ‘ I/O (new field)
res_rw:float,           ‘ Read/Writes (new field)
ouname:string,          ‘ other user (for su actions) (was otheruser)
remoteuname:string,     ‘ remote user name (new field)
remotehost:string,      ‘ remote host name (was rhost)
path0:string,           ‘ First File name (if present) (was file)
pathl:string            ‘ Second File name (if present) (New field)
]
```

Table 5.2: Ptype event (Beta-update Release)

PTYPE generic

```
ptype[generic
         id:string,
         s1:string,
         s2:string,
         s3:string,
         s4:string,
         il:int,
         i2:int,
         i3:int,
         i4:int
]
```

Table 5.3: Ptype generic

PTYPE generic_config

```
ptype[generic_config
         id:string,
         sval:string,
         ival:int
```

Table 5.4: Ptype generic_config

Rulebase Sets ia, m, and src

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' NIDES actions as used in audit records.[a]
'
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
set[ia: IA_VOID,      IA_DISCON,     IA_ACCESS,     IA_OPEN,
IA_WRITE,     IA_READ,         IA_DELETE,     IA_CREATE,     IA_RMDIR,    IA_CHMOD,
IA_EXEC,      IA_CHOWN,        IA_LINK,       IA_CHDIR,      IA_RENAME,   IA_MKDIR,
IA_MOUNT,     IA_UNMOUNT,      IA_LOGIN,      IA_BAD_LOGIN,  IA_SU,       IA_BAD_SU,
IA_EXIT,      IA_LOGOUT,       IA_UNCAT,      IA_RSH,        IA_BAD_RSH,  IA_PASSWD,
IA_RMOUNT,    IA_BAD_RMOUNT,   IA_PASSWD_AUTH,               IA_BAD_PASSWD_AUTH,
IA_CONNECT,   IA_ACCEPT,       IA_BIND,       IA_SOCKET_OPTION,           IA_KILL,
IA_CORE,      IA_PTRACE,       IA_TRUNCATE,   IA_UTIMES,     IA_FORK,     IA_CHROOT,
IA_MKNOD,     IA_HALT,         IA_REBOOT,     IA_SHUTDOWN,   IA_BOOT,     IA_SET_TIME,
IA_SET_HOSTNAME,               IA_SET_DOMAIN, IA_SET_UID,    IA_SET_GID,
IA_AUDIT_CONFIG,               IA_IS_PROMISCUOUS,
IA_ACTION_RESERVED00,          IA_ACTION_RESERVED01, ... , IA_ACTION_RESERVEDl44,
IA_ACTION_USER00,              IA_ACTION_USER01, ... ,     IA_ACTION_USER49,
]

''''''''''''''''''''''''''''''
' NIDES rulebase result codes '
'                             '
''''''''''''''''''''''''''''''
set[m: SAFE, WARNING, CRITICAL]

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' NIDES audit data sources as defined in NIDES audit records '
'                                                            '
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
set[src: IA_SRC_VOID, IA_SRC_CZ, IA_SRC_PACCT, IA_SRC_ADABAS,
         IA_SRC_LINK, IA_SRC_BSMVl, IA_SRC_BSMV2,
         IA_SRC_SYSLOG,  IA_SRC_AGEN_HOST,  IA_SRC_AGEN_NETWORK
         IA_SRC_RESERVED00, ... , IA_SRC_RESERVED89
         IA_SRC_USER00, ... , IA_SRC_USER49]
```

[a] See Table 6.6 for a description of NIDES action codes.

Table 5.5: Rulebase Sets

5.1.5 Sets

Sets are lists that associate a name with an integer value; they are comparable to enumerated types in C. The NIDES rulebase contains three sets (`ia`, `m`, and `src`), which are useful when writing new rules. For the Beta-update release of NIDES the `src` and `ia` sets were expanded to support user audit data customization. Table 5.5 lists the members of these three sets. A set declaration looks like this (note — NIDES users cannot define new sets for the NIDES rulebase):

```
set[letters: a, b, c, d]
```

This declaration creates a set *letters.*

The elements of the set can be referenced in a test in the antecedent of a rule as follows:

```
    [+ev:event|action == ia#DELETE,
              audit_src == src#IA_SRC_BSMV1]
```

This test looks for an event fact with its action field equal to the ia set member DELETE and the `audit_src` field equal to the `src` set member IA_SRC_BSMV1. If you review the definition for the event fact template (Table 5.2) you'll notice that the act ion field is declared as a type `ia` and the `audit_src` field is declared as a type `src`.

Set elements can also be used in the consequent clause. The following clause shows how the rulebase `inform` function uses the `m` set:

```
       [!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
       ev.hi_sequence, ev.lo_sequence, 'prstr,
       "Sample")]
```

This has the effect of passing the value for the CRITICAL member of the `m` set to the inform function.

The main purpose of using set declarations is to associate a name with an integer value. When you refer to a set, you must reference only valid members of the set.

5.1.6 Marks

Another important feature of the rulebase is the ability to mark and unmark facts, and to test for these marks. Since marks can have names, rules can mark a fact with different marks, and check for these marks by name. One use of this feature is to make sure that all the rules that can possibly use a fact have had the chance to do so. To do this, it helps to organize rules in mutually exclusive groups. Each rule in the group marks the fact with the same mark if it fires. Rules should be ordered so that those with hard-to-satisfy conditions have higher priority than those with easy-to-satisfy ones. A rule group might consist of, in effect, a set of filters with successively smaller holes. Here is an example, of how this would work. (Note: this example is intended to illustrate syntax and concepts only. It is not a real-life example of how intrusion detection would be done.) This rule assumes a built-in ptype called blucount, with the following structure. If you need to assert and delete intermediate facts, as shown in this example, you must use the generic ptype structure described in Section 5.1.4. Before writing any rules that apply marks to facts review Section 5.5.2, which discusses marks used by the default NIDES rulebase.

```
ptype[blucount
        user_id:integer,
        day:integer,
        second:integer,
        maximum:integer,
        current:integer]
```

These rules check for unsuccessful logins for a single user.

```
' Rule for when we have seen the maximum number of bad logins for this
' user (now we can give a security alert). It is the hardest to
' satisfy so it has the highest priority.
'

rule[UNSLOG(#50;*):
        [+ev:event^UNSLOG|action    == ia#BAD_LOGIN]
        [+bluc:blucount|user_id     == ev.real_userid]
        [?|bluc.current             == bluc.maximum]
==>
        [$|ev:UNSLOG]
        [!|sprintf('prstr,
                   "User %s executes %d Bad logins!!\n",
                    ev.real_userid,bluc.current)]
        [!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
        ev.hi_sequence, ev.lo_sequence, 'prstr,
        "UNSLOG")]
]


' Rule for counting unsuccessful logins after the first,
' up to bluc.maximum.  This rule has more conditions than UNSLOG2, so
' its priority is higher than UNSLOG2.
'

rule [UNSLOG1(#40;*):
        [+ev:event^UNSLOG|action    == ia#BAD_LOGIN]
        [+bluc:blucount|user_id     == ev.real_userid]
        [?|bluc.current             <= bluc.maximum - 1]
==>
        [$|ev:UNSLOG]
        [/bluc|current              += 1]
]

‘
‘ Rule for first unsuccessful login for this user. It has a lower
‘ priority than UNSLOG and UNSLOGl because it is the easiest
‘ to satisfy (it will catch all unmarked
‘ bad login event facts).  If one of the other rules
‘ fires, that rule will mark the fact, preventing this rule from
‘ firing.  That's the way we want it to work since this rule should
‘ only fire on the first bad login for a particular user.  The effect
‘ is that if we’ve already seen a bad login for that user, this
‘ rule won't fire.

rule[UNSLOG2(#30;*):
        [+ev:event^UNSLOG|action == ia#BAD_LOGIN]
==>
        [+blucount|user_id        = ev.user_id,
                   day            = ev.days,
                   second         = ev.seconds,
                   maximum        = 4,
                   current        = 1]
        [$|ev:UNSLOG]
]


‘
‘ Remove bad login fact.
‘ This rule should fire only after all the other rules have seen the
‘ fact. For this reason it gets assigned the lowest priority of the
‘ rules in the group.


rule[UNSLGGC(#-1;*):
        [+ev:event|action == ia#BAD_LOGIN]
==>
        [!|printf("Removing bad login fact with timestamp %s %s\n",
                 ev.days, ev.seconds)]
        [-|ev]
```

```
]
```

This example shows a group of rules named UNSLOG, UNSLOG1 and UNSLOG2. Each rule looks for event facts with a BAD_LOGIN action and mark facts with the UNSLOG mark and also determine if each fact it is checking has not already been marked. The example includes a rule that removes facts that have been "seen" by the rules, UNSLGGC. *Note that under normal NIDES operation the event fact should not be removed by any rules you write.*

The syntax for all this is as follows. If a rule wants to mark a fact, it includes a clause like

```
[$|ev:UNSLOG]
```

in its consequent. The $| followed by the fact alias name and the mark name indicates the marking of the fact. If a rule wants to check for a fact marked with UNSLOG, it includes the clause

```
[+ev:event$UNSLOG]
```

in its antecedent.

If a rule wants to check for a fact that has not been marked with a certain mark, it should include a clause of the form

```
[+ev:eventˆUNSLOG]
```

in its antecedent. Similarly, if it wants to remove the mark from a fact, it should include a clause of the form

```
[^|ev:UNSLOG]
```

in its consequent.

The UNSLGCC rule gives an example of the technique for removing facts that multiple rules may want to use. This rule has a lower priority than rules that may fire as a result of the facts that this rule removes. The conflict resolution mechanism takes rule priorities into account when deciding which rule will be allowed to fire when the conditions of multiple rules are satisfied by the same fact. Since the default rule priority is zero (0), a priority of negative one (-1) could be assigned to the UNSLGCC rule. The conflict resolution mechanism would cause all the rules with a higher priority to fire before the UNSLGCC rule fires.

5.1.7 Rule Priorities

Many of the default rules that are grouped to perform a single type of inference are ordered so that tests are performed in a well-defined sequence. The unsuccessful login rule group described in Section 5.1.6 is an example of rule ordering. If you create a group of rules to address a particular scenario, to ensure that the rules are tested in the proper order you should assign them priorities. Any rules you write should have a priority that is -96 or higher. Rules defined without a priority are given a default priority of zero. Priorities are ordered from high to low; rules with higher priorities are tested before rules with lower priorities. Most of the rules included in the NIDES release have a priority of 0. You assign a priority to a rule in the rule declaration as follows:

```
rule [Rulename(#71;*):
```

The `#71` assigns the rule a priority of 71. Assigning a priority to a rule is optional.

Below is an example of a group of pseudo-rules that are tested in a specific order (TryFirst, TrySecond, and TryThird). The TryFirst rule has the highest priority (specified by the `#22`), and the most specific antecedent conditions; it is tested first. If it does not fire and therefore applies no mark to the event fact, TrySecond is tested next; its conditions are easier to satisfy than those of TryFirst. If TrySecond does not fire, TryThird, which has the lowest priority of the three rules, fires by marking the event fact with the `TRY` mark. This mark allows only one of the three rules to fire on the current event fact, and prevents that rule from firing more than once.

```
rule[TryFirst(#22;*):
        [+ev:event^TRY|action == ia#LOGIN,targid==fileserver]
        [+vacation_schedule|userid=ev.real_userid]
        [+no_fileserver|userid=ev.real_userid]

==>
        [$|ev:TRY]
        [!|sprintf('prstr,
                 "User %s is on fileserver and is on vacation!!\n",
                  ev.real_userid)]
        [!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
        ev.hi_sequence, ev.lo_sequence, 'prstr,
        "TryFirst")]
]

rule[TrySecond(#21;*):
        [+ev:event^TRY|action == ia#LOGIN,targid==fileserver]
        [+no_fileserver|userid=ev.real_userid]

==>
        [$|ev:TRY]
        [!|sprintf('prstr,
                 "Non-approved User %s is on fileserver !!\n",
                  ev.real_userid)]
        [!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
        ev.hi_sequence, ev.lo_sequence, 'prstr,
        "TrySecond")]
```

```
rule[TryThird(#20;*):
        [+ev:event^TRY]
==>
        [$|ev:TRY]
]
```

5.1.8 Inference Groups

Rules that process audit records are arranged in groups of one or more rules that implement a particular piece of inference. Rules in the group are tested in a fixed order, based on their priorities. Ordinarily, these rules are mutually exclusive; that is, at most one from the group will fire on any particular audit record. To enforce this, each member of a group of rules marks the audit record with a mark unique to that set of rules, and each rule checks the event fact for that mark. Thus, if one rule in the group fires, the rest are prevented from firing, and the rule that has fired is prevented from firing repeatedly.

An example from the NIDES rulebase of the use of rule ordering and marks for execution flow control is the set of rules that infer the login type for a given login (Section 5.5.4.5.1 describes the login rules). Each of these rules checks for the `LOG` mark on the event fact. If the rule fires, it marks the event fact with the `LOG` mark. In particular, the RemoteLogin rule is prevented from firing only because of the mark, assuming the event fact contains a `LOGIN` action.

The login rules are organized conceptually as a set of sieves, with each sieve having smaller holes than the one before it. That is, each rule has a set of conditions that are easier to satisfy than the ones before. The final RemoteLogin rule fires on any unmarked event fact that has a login or rsh action.

In summary, inference can be organized by writing a set of rules that all use one particular mark. In general, every rule should check the event fact for some mark, and, if it fires, mark the event fact with the same mark it checks for. This guarantees a uniform method of execution flow and minimizes undesired interactions between rules. The rules should be arranged so that the ones with more-difficult-to-satisfy conditions are tested before those that are easier to satisfy. Possibly some rule that will always fire should be tested last, as a default. The effect is to minimize the tests required to achieve a particular inference.

5.1.9 Generating Alerts

The NIDES rulebase contains a built-in C function, `inform`, which you call when you want to generate a NIDES alert. The function is activated from the consequent clause of a rule. An inform statement looks like this:

```
        [!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
                ev.hi_sequence, ev.lo_sequence, 'prstr,
                    "RULENAME")]
```

The `inform` function has seven arguments, as follows:

- **Inform level —** Any item from the `m` set (see Table 5.5 for a listing of the members of the `m` set); a value of CRITICAL indicates an alert
- **User Name —** Name of the subject responsible for the alert, usually the `ev.real_userid` field of the event fact
- **Time —** Usually the `ev.timegen` field of the event fact
- **Sequence Number (High-order bits) —** The `ev.hi_sequence` field of the event fact; you should not use any other value in this field
- **Sequence Number (Low-order bits) —** The `ev.lo_sequence` field of the event fact; you should not use any other value in this field
- **Message —** A free form text message, usually created via the sprintf function
- **Rule Name —** The name of the rule that generated the alert

To create the string used for the alert message, use the built-in string variable `prstr` and the sprintf function:

```
        [!|sprintf('prstr,
                  "User %s executes %s command %d times!!\n",
                   ev.real_userid,ev.cmd,count)]
```

You can put any information in the string you want. The string is useful for providing additional information about the alert. See Section 5.3 for an example.

5.1.10 Sample Rule Declaration

Here is an example of a complete rule declaration:

```
rule[SimuLogon(#l;*):
        [+ev:event^SIMU|action == ia#LOGIN]
        [+se:session|userid == ev.userid]
        [?|se.terminal!= ev.terminal]
==>
        [$|ev:SIMU]
        [!|sprintf('prstr,"SimuLogon: user %s at terminals %s, %s\n",
                ev.userid, ev.terminal, se.terminal)]
        [!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
                ev.hi_sequence, ev.lo_sequence, 'prstr,
                "SimuLogon")]
        [-|se]
]
```

This rule detects a user logging in on a terminal while already logged in somewhere else. It works by checking for an *event* fact that is reporting a *LOGIN* action, and checking to see if there is any *session* fact with the same *userid* field. If such a session fact exists, it compares the *terminal* fields of the event and session facts to see if they are different. If they are different, the rule fires. This rule assumes that some other rule will look for login actions and create session facts for each login. The rule syntax is completely determined by the delimiters (brackets and ==>.) This rule is formatted in a readable fashion, but such formatting is optional.

A rule declaration begins with the keyword rule. It then has a name section followed by a colon (:). The name section of the above rule consists of the rule name *(SimuLogon)* followed by a set of options in parentheses. The `#1` option gives the rule a priority of 1. This means that if several rules can fire, this rule will fire before rules with a lower priority. Thus if this rule and a rule with a priority of -1 can both fire, this one will be selected to fire first. The asterisk (*) option means that the rule can fire repeatedly without some other rule firing in the meantime. By default, rules are not repeatable, since once a rule's antecedents are satisfied, they will continue to be satisfied forever, and if the rule were repeatable (indicated by the *) it would fire again and again. As we will see below, this rule deletes or marks the facts that satisfy it and so prevents such a loop.

The body of a rule consists of a sequence of antecedent clauses, also known as tests; the delimiter ==>; and a sequence of consequent clauses, also known as actions. Each antecedent clause consists of some test. The first antecedent clause in the above rule

```
[+ev:event^SIMU|action == ia#LOGIN]
```

checks for an event fact that is not marked with the SIMU mark and whose action field contains the LOGIN element from the set named ia (sets are described in Section 5.1.5). If it finds such a fact, it gives it the alias of *ev*. The clause

```
[+se:session|userid == ev.userid]
```

checks for a session fact with the same value in the userid field as the ev fact already found. If it finds such a fact, it gives it the alias of se. The third clause

```
[?|se.terminal != ev.terminal]
```

tests whether the terminal field in the session fact is different from the one in the event fact.

If all the antecedent clauses of a rule are satisfied, the rule "fires"; that is, its consequent clauses (actions) get executed. This rule has four actions. The first clause

```
[$|ev:SIMU]
```

marks the event fact with the SIMU mark; this prevents the rule from firing again on the same event fact.

The clause

```
[!|sprintf(‘prstr,"SimuLogon: user %s at terminals %s, %s\n”,
        ev.userid, ev.terminal, se.terminal)]
```

is a call to a C function. The !| syntax indicates such a call. The NIDES rulebase recognizes several built-in C functions/library calls; these functions are listed in Table 5.10. Such calls can reference the fields in the facts as if they were C structure elements (which they in fact are). This is done using the fact's alias name followed by a dot (.) and the field name.

The clause

```
[!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
        ev.hi_sequence, ev.lo_sequence, ’prstr,
        "SimuLogon")]
```

is a call to the special rulebase C function inform. This is the function used to generate an alert. The clause

```
[-|se]
```

removes the se fact from the factbase. This is indicated by the -| syntax followed by the fact's alias name.

5.2 Installing Rules

Once you have designed and successfully written a new rule, you need to install the rule into the NIDES environment. The NIDES rulebase uses a special rule translator, pbcc. The rule translator accepts a rule definition, written in the NIDES rule specification language, and produces a C language representation of the rule, which is then compiled into an object file that can be used by the NIDES rulebased analysis component.

This release of NIDES does not include any automatic mechanisms for protection of your rulebase from unauthorized tampering or reverse engineering. Guidelines for protecting the rulebase are presented in Section 9.4.2.3. You should review and follow these guidelines when working with the NIDES rulebase.

5.2.1 Development Environment

The NIDES rulebase development environment contains five directories located under $IDES_ROOT/exsys:

- **bin** — executable programs for compiling and installing new rules; makerule command script, gcc C compiler, and pbcc rule compiler.
- **include** — header files that define structures and constants needed by rules compiled for NIDES usage.

- **single-rules** — directory where the `makerule` script will place compiled rule object files; copies of the rule object files are also placed in the $IDES_ROOT/etc/rulebase directory.

- **src** — directory where source code files for new rules should be created; the `makerule` script will look for rule source code in this directory. When the rule source code is not needed or has already been compiled, we recommend removing it from this area, or encrypting it to prevent unauthorized access to the rule.

5.2.2 Compiling and Installing

To compile and install a new rule that can be used by NIDES, create a file containing the rule definition; place this file in the $IDES_ROOT/ exsys/src directory. The name of the file created should be *rulename.* pb, where *rulename* is the name of the rule contained in the file. You must put only one rule in each .pb file. You cannot put any ptype or set declarations in your rule file.

After creating the rule file, you can compile, link, and install the rule using the NIDES `makerule` script by typing

```
%$IDES_ROOT/exsys/bin/makerule  rulename
```

at the shell prompt — make sure your $IDES_ROOT environment variable is set prior to running `makerule.` The script will execute the commands necessary to produce an executable version of the rule. If an error occurs during the compilation, an appropriate error message is displayed and the rule is not installed. The `makerule` script processes your rule file using the `pbcc` translator.

The `makerule` script installs your new rule in the $IDES_ROOT/etc/rulebase directory. After installation is completed, the rule is available for use in NIDES.

5.3 Rulebase Tutorial

This tutorial takes you through the process of writing, compiling, installing, and using a new rule in the NIDES rulebase. Below is an example of a single rule that addresses a pseudo-scenario. We discuss its format and semantics, go through its compilation and installation, and show how the rule fires.

First create a file called `Identitycrisis.pb` in your IDES_ROOT/exsys/src directory. Enter the following text into the file:

```
‘
‘  Everything on the line after a backquote is a comment. You
‘  need not type comments in for this exercise if you don’t
‘  want to!
```

```
‘
‘  This rule fires if a user executes the “whoami” command.
‘  While this rule is not addressing a real world intrusion
‘  scenario, it demonstrates a simple rule that looks up
‘  data in the audit record “event” fact, tests the fact for
‘  a condition, and if the condition is met generates an alert.


rule[IdentityCrisis(*):
        [+ev:event^IDC|cmd == "whoami",
                      real_userid != "root"]
==>
        [$|ev:IDC]
        [!|sprintf(’prstr,
                  “User %s on %s can’t remember who they are!!!\n
                   Perhaps an identity crisis is in the cards!!!!",
                   ev.real_userid,ev.targid)]
        [!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
                 ev.hi_sequence, ev.lo_sequence, ’prstr,
                 “Identitycrisis”)]
]
```

This rule does the following:

- Checks in the factbase for an event whose cmd field equals “whoami”. Any event fact that contains the mark “IDC” is disregarded, because this mark indicates that this event has already been seen by this rule. If the userid in the event fact is root, this rule will not fire.
- If all the conditions in your antecedent (those lines before the ==> characters) are satisfied, then the actions in the consequent (those statements after the ==> characters) are executed. In this case the event fact is marked IDC, an alert message is created using the sprintf system call, and an alert is generated using the NIDES rulebase *inform* function.

When you have finished typing in this sample rule, save it into the file called `IdentityCrisis.pb`. You can have only one rule per file, and the name of the file must be *rulename.*pb.

Now you can translate, compile, and link the file to produce an executable version of the rule using the NIDES `makerule` script. Type

```
%$IDES_ROOT/exsys/bin/makerule  IdentityCrisis
```

at the shell prompt. The script executes the commands necessary to produce an executable version of the rule. This rule is automatically installed in your IDES_ROOT/etc/rulebase directory. After installation is completed, the rule is available for use in NIDES.

After your new rule is successfully compiled, start NIDES analysis and turn ON your new rule. Make sure you have some alert reporting mechanism turned ON so you can see the alert when it is generated. Turn ON any target host that you have set up to provide audit data to NIDES, and log onto that host and execute the `whoami` command — be sure you are not logged in as `root,` or your rule will not generate an alert. Wait to see your new rule fire and generate an alert.

5.4 Rulebase Configuration File (rb_config)

The rulebase configuration file, `rb_config,` resides in the $IDES_ROOT/etc directory. The `rb_config` file is a simple text file that you modify using a text editor. This file consists of sections describing each rulebase item that is configurable. A default `rb_config` file is included with the NIDES software. The default file includes Sun OS and UNIX specific configurations. You should review the defaults and modify them to suit your computing environment prior to running NIDES. Section 5.4.2 lists the default values for the `rb_config` file.

If you decide to modify the configuration file, we recommend making a backup copy of the file before making any changes. Remember that the NIDES rulebased component will get configuration information from this file. This file is read each time the NIDES real time analysis components are started and when a test is started. If you want to change the `rb_config` file, you must stop NIDES analysis and start it again after modifying the `rb_config` file.

5.4.1 Configuration File Syntax

The `rb_config` file is organized as a series of sections. Every blank line or line that begins with a whitespace character is ignored. The following syntax rules must be followed in the rulebase configuration file:

- Lines beginning with the '#' character are ignored (treated as comments).
- Each section begins with the name of the section, and ends with the keyword 'NO_MORE'. For example, the following is a section called PROGLOCATION:

```
PROGLOCATION
/bin/ 1
/usr/bin/ 1
/usr/ucb/ 1
/etc/ 0
/usr/etc/ 1
NO_MORE
```

- Each section requires a fixed number of arguments per line (separated by white space). The format of the arguments is also fixed. Refer to Section 5.4.2 to determine the appropriate format for each section.

5.4.2 Configuration File Sections

The `rb-config` file has 25 sections. Table 5.6 lists the sections and the rules that utilize them. The following paragraphs describe each section in the `rb_config` file. For each section, the default configuration is listed to serve as a format example. Sections without default settings are noted and sample data is used to illustrate the item's format. You should review the default values carefully and make changes, as needed, to adapt the file to your computing environment.

DOMAIN Internet domain names. The system considers logins from these domains to be 'local', and logins from any other domains to be 'remote'. This section's default configuration contains no domain names. It is important that you enter in all local domains here prior to running NIDES, since this is how the NIDES rulebased component distinguishes between local and remote network activity (and local and remote users). The format for this configuration item is a simple string representing a domain. This example is sample data only and should be replaced by actual data.

Example only

```
DOMAIN
mydeptl.mycompany.com
mydept2.mycompany.com
NO_MORE
```

GENERIC_CONFIG User-defined configurations. This section can be used by any user-developed rules. The format for this configuration item, shown in the example, is two strings followed by an integer. This example is sample data only and should be replaced by actual data.

Example only

```
GENERIC_CONFIG
my_config1 stringvaluel 1
my_configl stringvalue2 1
my_configl stringvalue3 4
my_config2 stringvaluel 0
my_config2 stringvalue2 0
my_config2 stringvalue3 0
NO_MORE
```

Section	**Rules That Utilize Section**
DOMAIN	MultiLogin1,LocalLogin, RemoteRootBadLogin RemoteRootBadPassword
GENERIC_CONFIG	None – To be used by user-generated rules
HOME_DIR	ChmodOtherUser,AccessPrivateFile1
KNOWN_LOGIN	KnownLogin1
LOG_DIR	TruncateLog
LOGIN_CONFIG	ChangeLoginFile
NOEXEC	BadUserExec
PARANOID_PROG	ParanoidUser1,ParanoidUser2,ParanoidUser3 ParanoidUser4
PRIVATE_DEVICE	AccessPrivateDevice
PRIVATE_FILE	AccessPrivateFile1
PROGLOCATION	TrojanHorse,ModSystemExec,LinkSystemExec, ReadSystemExec,ChmodSystemFile
PROGRAM	TrojanHorse
RAREEXEC	RunsRareExec,SuspiciousUser
REMOTE_FILE_NO_ACCESS	RemoteFile2
REMOTE_FILE_NO_MODIFY	RemoteFile1
REMOTE_NO_EXEC	RemoteExec
REMOTE_NOT_OK	NoRemote
ROOT_OK	BadRoot
SPECIAL_FILE	AccessSpecialFile
SPECIAL_PROGRAM	SpecUserProgram
SPECIAL_USER	SpecUserExec
SYSTEM_SCRIPTS	ReadSystemExec
TMP_DIRNAME	DotFile
TMP_FILE	DotFile
USER_TYPE	AccessSpecialFile

Table 5.6: Rulebase Configuration File Sections

HOME_DIR Users and their home directories. Requires user name and full pathname (slash-terminated) of home directory. This information is used by several rules that report access to sensitive or private files in a user's home directory. This section does not contain any default data. The list should be initialized with your users and their home directories. This example is sample data only and should be replaced by actual data.

Example only

```
HOME-DIR
avon          /homes/b/avon/
holly         /homes/a/holly/
Sarah         /homes/b/sarah/
NO_MORE
```

KNOWN_LOGIN Names of accounts that are known to be commonly left unprotected. Default values are shown below; you may add or delete accounts from the list.

Default Configuration

```
KNOWN_LOGIN
guest
anonymous
bin
sync
NO_MORE
```

LOG_DIR Names of directories where log files are kept. Make sure the LOG_DIR entries match the locations where your audit data is stored.

Default Configuration

```
LOG_DIR
/var/adm
/usr/audit/audit_trails
NO_MORE
```

LOGIN_CONFIG Scripts automatically executed at login or shell execution time. NIDES uses this information in rules that detect changes or attempts to change these files by non-owning users. You should list here names of initialization files that should not be modified by other (i.e., non-owning) users.

Default Configuration

```
LOGIN_CONFIG
 .login
 .cshrc
 .profile
 .xinitrc
NO_MORE
```

NOEXEC Programs normal users shouldn't run; only root should run them. NIDES uses this information in rules that detect execution or attempted execution of these programs.

Default Configuration

```
NOEXEC
ypset
NO_MORE
```

PARANOID_PROG Programs paranoid users may execute frequently to see if anyone may be observing their activities.

Default Configuration

```
PARANOID_PROG
finger
ps
w
who
NO_MORE
```

PRIVATE_DEVICE Private devices that abusers can use to eavesdrop on or spoof another user. NIDES uses this information in rules that detect when these private devices are accessed by a user not logged in at the console. The default list contains Sun-specific items.

Default Configuration

```
PRIVATE_DEVICE
# These are sun-specific devices.
/dev/audio
/dev/fb
/dev/kbd
NO_MORE
```

PRIVATE_FILE Files in a user's home directory that should be accessed only by that user. NIDES uses this information in rules that detect access or attempted access to these files by other users.

Default Configuration

```
PRIVATE_FILE
.rhosts
.netrc
RMAIL
NO_MORE
```

PROGLOCATION Names of directories (slash terminated) where system files reside. If the name of the directory is followed by a '1'(one), it indicates that the directory holds executable files. A '0'(zero) indicates that the directory is not normally used to hold executable files. NIDES uses this information in rules that detect execution or attempted execution of programs from directories where programs are not expected to be stored.

Default Configuration

```
PROGLOCATION
/bin/          1
/usr/bin/      1
/usr/ucb/      1
/etc/          0
/usr/etc/      1
NO_MORE
```

PROGRAM System programs that should be executed only from system directories. If a program with one of these names is executed from a directory not found in the PROGLOCATION section, the rule-based component reports it as a Trojan horse. The default list contains 35 items. If some of the programs listed here are not normally contained in a directory listed in the PROGLOCATION configuration and labeled with a '1', you should remove them from this list or add the directory where they are located to the PROGLOCATION configuration.

Default Configuration

```
PROGRAM
auk
cat
chgrp
chmod
```

```
cmp
cp
cpio
csh
date
diff
dirname
du
echo
ed
eject
env
ex
fgrep
file
find
grep
hostname
ln
login
ls
mkdir
mv
pwd
rm
sh
sleep
sort
sync
tar
tcsh
touch
who
NO_MORE
```

RAREEXEC Programs users don't ordinarily run. A user who runs one of them is considered suspicious by NIDES.

Default Configuration

```
RAREEXEC
adb
NO_MORE
```

REMOTE_FILE_NO_ACCESS Names of files that remote users should not access, no matter where they occur in the file system. NIDES uses this information to detect access or attempted access to these files by remote users.

Default Configuration

```
REMOTE_FILE_NO_ACCESS
.rhosts
hosts.equiv
NO_MORE
```

REMOTE_FILE_NO_MODIFY Full pathnames of files that remote users should not modify. NIDES uses this information to detect modification or attempted modification to these files by remote users.

Default Configuration

```
REMOTE_FILE_NO_MODIFY
/etc/passwd
/etc/aliases
/etc/aliases.pag
/etc/aliases.dir
NO_MORE
```

REMOTE_NO_EXEC Programs that remote users should not execute. NIDES uses this information to detect execution or attempted execution of these programs by remote users.

Default Configuration

```
REMOTE_NO_EXEC
a.out
adb
cc
chfn
gcc
kermit
ld
lp
lpd
lpr
mount
rz
sz
```

```
umount
xmodem
ymodem
zmodem
NO_MORE
```

REMOTE_NOT_OK Users who are not authorized to log in remotely. NIDES reports an anomaly if any user from this list logs in from a remote site. This example is sample data only and should be replaced by actual data.

Example only

```
REMOTE_NOT_OK
godzilla
sleer
NO_MORE
```

ROOT_OK Users authorized to become root. If any other user becomes root, an anomaly is reported. This example is sample data only and should be replaced by actual data.

Example only

```
ROOT_OK
batman
superman
wonderwoman
NO_MORE
```

SPECIAL_FILE This `rb_config` file section, along with the USER_TYPE `rb_config` file section, describes an access control mechanism. Files (full pathnames) are correlated with a type id number in the SPECIAL-FILE section. Multiple files can be correlated with the same type id number. The type id numbers correlate with type id's listed in the USER-TYPE `rb_config` file section. For example, in the default configuration, shown here, all files are given a type id number of 1. In the USER-TYPE configuration section, only one user is listed, root with an id of 1. This indicates that user root is the only user that should access files listed here with a type id of 1. If user "admin" were added to the USER-TYPE section with an id of 1, then user "admin" would also be allowed to access the files with type id 1.

Default Configuration

```
SPECIAL_FILE
/etc/exports            1
/etc/netgroup           1
```

```
/etc/inetd.conf        1
/etc/syslog.conf       1
/etc/ntp.conf          1
/etc/syslog.conf       1
/usr/lib/uucp/l.sys    1
NO_MORE
```

SPECIAL_PROGRAM Special programs that shouldn't be executed by anyone but a specified set of users. Both program name and user name must be entered. This list is used to specify programs that are required for tasks delegated to a particular user other than root (programs that only root uses should be placed in the NOEXEC section). NIDES uses this list to detect execution or attempted execution of programs in the list by users other than those users listed.

Default Configuration

```
SPECIAL_PROGRAM
audit          audit
in/ftpd        ftp
NO_MORE
```

SPECIAL_USER Special (pseudo) users that shouldn't execute anything but a specified set of programs. You must list both the user name and the name of the program that the user is allowed to run (or NULL if the user is not allowed to run any programs). A user can be limited to a small set of programs by making multiple entries for that user. NIDES will detect any execution or attempted execution by a user in this list of a program that is not listed here for that special user.

Default Configuration

```
SPECIAL_USER
bin NULL
sync sync
AUpwdauthd NULL
NO_MORE
```

SYSTEM_SCRIPTS Shell scripts that reside in system directories. Ordinarily, users should not read system executables (files residing in one of the directories listed in the PROGLOCATION section), but to execute shell scripts the user must read these script files. NIDES uses this information to detect acceptable accesses to files located in the PROGLOCATION directories (i.e., files listed here).

Default Configuration

```
SYSTEM_SCRIPTS
/bin/arch
/bin/mach
/bin/sparc
/bin/sun4
/bin/sun4c
/bin/false
/bin/true
/usr/bin/arch
/usr/ucb/which
/usr/etc/yp
NO_MORE
```

TMP_DIRNAME Temporary directories for the system. Directories listed must be terminated with a "/" character. An intruder sometimes writes invisible files into temporary directories because they are cleared at boot time, thus destroying evidence of the intrusion. Writing invisible files (files beginning with a ".") into these directories is considered a suspicious activity and will be flagged by NIDES.

Default Configuration

```
TMP_DIRNAME
/tmp/
/var/tmp/
/usr/tmp/
NO_MORE
```

TMP_FILE Dot files that are OK to write into the directories specified by the rb_config file section TMP_DIRNAME. These files are written by system utilities or window systems. Writing files not listed here into any directories listed in TMP_DIRNAME is considered suspicious and will be reported by NIDES. The default configuration contains no entries for this item; therefore, any files starting with a "." character written to any TMP_DIRNAME directory will be flagged.

Example Only

```
TMP_FILE
.spsinfo
NO_MORE
```

USER_TYPE This `rb_config` file section, along with the SPECIAL_FILE `rb_config` file section, describes an access control mechanism. Users are listed with a type id number. Files (full pathnames) are correlated with the same type id numbers in the SPECIAL_FILE section. Multiple users can be correlated with the same type id number, allowing them to access any of the files within that type. For example, in the default configuration, only one user is listed, root with an id of 1. This indicates that user root is the only user that should access files listed in the SPECIAL_FILE list with a type id of 1. If user "admin" were added with an id of 1, then user "admin" would also be allowed to access the files with type id 1.

Default Configuration

```
USER_TYPE
root 1
NO_MORE
```

5.4.3 User-Defined Configurable Rules

The NIDES rulebase contains a number of rules that can be configured at start up. The configuration is done by means of the `rb_config` file. User-defined rules can also be configured via the `rb_config` file. The `rb_config` file section GENERIC_CONFIG can be used to configure user-generated rules. The generic_config ptype definition looks like this:

```
ptype[generic_config    id:string,
                        sval:string,
                        ival:int]
```

New facts using the generic_config ptype cannot be asserted by any rules; facts are initialized when the rulebase starts; the facts are derived from the `rb_config` file GENERIC_CONFIG section only.

As an example of how to use the `rb_config` file to configure a new rule, suppose you want to store a list of users whose access is limited to a certain list of hosts (that is, you want to report an alert if a restricted user uses an unauthorized host). You could specify two lists in the `rb_config` file — one for the user list, and one for the list of hosts — by entering the following data in the `rb_config` file under the GENERIC_CONFIG section:

```
GENERIC_CONFIG
#begin limited access user list
limited_host_user    joe     0
limited_host_user    mary    0
limited_host_user    zeus    0
limited_host_user    isis    0
#begin limited host list
limited_host         bluejay 0
limited_host         dolphin 0
limited_host         spruce  0
limited_host         daisy   0
NO_MORE
```

Because the generic_config ptype has three fields (id, sval and ival — shown in the generic_config ptype declaration), the `rb_config` file entries must match the number and types of the fields specified for the ptype. The 0's are included in our example as filler for the `ival` field.

After you have included the configuration information in the `rb_config` file, you could write a rule that would check to see if a user was a limited_host_user and if so could report an anomaly if the user was not using an acceptable host — that is, one listed as a limited_host in the GENERIC_CONFIG section — for example:

```
rule[LimitedHost(*):
          [+ev:event^LMTDHOST]
          [+generic_config|id == "limited_host_user”,sval == ev.real_userid]
          [-generic_config|id == "limited_host",sval== ev.targid]
==>
          [$|ev:LMTDHOST]
          [!|sprintf(’prstr,
          "User %s should not be on host %s.",
           ev.real_userid,ev.targid)]
          [!|inform(m#CRITICAL, ev.real_userid, ev.timegen,
          ev.hi_sequence, ev.lo_sequence, ’prstr,
          "LimitedHost")]
]
```

5.5 Default NIDES Rulebase

The default rulebase delivered with the beta release of NIDES includes 69 rules, 39 of which generate alerts when their conditions are met. Some of the rules included with the release function as a group; rules that are not part of a group may depend on rule groups to collect information they need to satisfy their test (antecedent) conditions.

5.5.1 Rule Groups and Dependencies

The NIDES rulebase has several groups of rules that function together to perform some inference. These groups of rules should be turned ON or OFF together. If a subset of group members is activated or deactivated, NIDES rulebase performance will be unpredictable. Rules that should generate alerts may not be able to, if related rules in the group that generate needed facts are turned OFF. In addition, the factbase may become overly large if rules that support factbase maintenance (by deleting unneeded facts) are deactivated, while other rules that create those facts remain active.

Table 5.7 lists the four NIDES rule groups (Password/Login, Session, Paranoid User, and TFTP) included with the NIDES Beta release. Rules that can generate an alert are listed in boldface. Rules that are part of the same group should always be activated/deactivated together.

Group Dependencies

None of the rules in the Session Rule Group generate an alert. This group is responsible for maintaining LOGIN session information used by many other rules.

Table 5.8 lists all rules that depend on the Session Rule Group for their correct functioning. We recommend that you do not deactivate any rules that are part of the Session Rule Group.

5.5.2 Rulebase Marks

As described in Section 5.1.6, most NIDES rules use a marking mechanism to tag, in the factbase, facts that have been seen by a rule or group of rules. When you are writing new rules, you will probably use marks in most of your rules. When you write new rules, it is important that you do not use a mark that is used by one of the NIDES default rules.

Table 5.9 lists all marks used by the default rules. When writing new rules, be sure to check this table to ensure that any marks you apply to facts are unique and are not listed in the table.

5.5.3 Rulebase C Functions

The NIDES rulebase recognizes several built-in C functions/library calls, as listed in Table 5.10. You can use these functions in rules you write. UNIX manual pages can provide details on the parameters for these functions. Such calls can reference fields in facts as if they were C structure elements. This is done using the fact's alias name followed by a dot (.) and the field name. For example:

```
    [+ev:event|action == ia#LOGIN]
==>
    [!|sprintf('prstr,"User %s executes commands on terminal %s\n",
   ev.userid, ev.terminal)]
```

Here we have aliased an event fact from the factbase whose action field equals LOGIN. The alias ev is referenced in the system call `sprintf`, which constructs a string using two fields in the event fact — `userid` and `terminal`.

5.5.4 Default Rule Descriptions

Table 5.11 lists the default rules contained in the NIDES rulebase. Rules whose names are in boldface in the table generate alerts. If you want to see the source code for the default rules, look at the sample file `rulebase.src`. This file was included with the NIDES release; check with your system administrator to obtain the location of the file. All rulebase source

Rule Group	**Description**
Password/Login BadPassword1 BadPassword2 **BadPasswordAnomaly** **GoodPassword1** GoodPassword2 BadLogin1 BadLogin2 **BadLoginAnomaly** BadLoginBadPassword **GoodLogin1** GoodLogin2 **GoodSU1** GoodSU2	Maintains password and login information. This group counts bad password/login entries for a user, and reports an alert if a threshold is exceeded. Some of these rules update or remove bad password/login counts.
Session MultLogin1 MultLogin2 FlagRSH ConsoleLogin DialInLogin LocalLogin RemoteLogin Logout1 Logout2 Su1 Exec ClearSession TouchSession	Maintains information about a user's current session. Includes session type, counts of various activities, and removal of session facts when the session is terminated or remains inactive for a period of time.While none of the Session rule group rules generate an alert, many other NIDES rules rely on the Session rule group information to function. Table 5.8 lists the rules that rely on the Session group. *We recommend leaving all the Session group rules ON.*
Paranoid User ParanoidUser1 ParanoidUser2 ParanoidUser3 ParanoidUser4 **ParanoidUserAnom** ClearParanoidUser	Maintains information about paranoid user activity.
TFTP TFTPUse **TFTPAnomaly**	Records tftp usage.

Table 5.7: NIDES Default Rulebase - Rule Groups

Rules Dependent on Session Rule Group	
AccessPrivateDevice	ParanoidUserAnom
BrokeRoot	RemoteExec
ClearParanoidUser	RemoteFile1
InvisibleDirectory	RemoteFile2
Leapfrog1	RemoteFile3
NoRemote	RemoteMount1
ParanoidUser1	RemoteMount2
ParanoidUser2	RunsRareExec
ParanoidUser3	SuspiciousUser
ParanoidUser4	

Table 5.8: Rules Dependent on Session Rule Group

Default Rulebase Marks			
APD	CSF	NR	RRBP
APF	DF	PFA	RRE
ASF	EX	PU	RSE
BAR	FA	PUA	RSH
BE	ID	RE	SSU
BLOG	KL	RF1	SU
BP	LF	RF2	SUE
BR	LO	RF3	TH
BT	LOG	RM	TL
CLF	LSE	RRBL	TU
COU	MSE		

Table 5.9: Marks Used by Default Rulebase

Rulebase C Functions		
atof	getint	seek
atoi	getline	sprintf
close	getword	sscanf
creat	itoa	strcat
fclose	lower	strcmp
fgets	lseek	strlen
fopen	printf	strstr
fprintf	putc	tolower
fputs	putchar	toupper
fscanf	read	unlink
getc	readlines	write
getch	scanf	writelines
getchar		

Table 5.10: C Functions Available in NIDES Rulebase

files should be protected from unauthorized access; refer to Section 9.4.2.3 in the system installation instructions for guidelines on safeguarding the rule files.

5.5.4.1 Housekeeping Rules

The NIDES rulebase has four "housekeeping" rules:

remove-event Removes an event fact (i.e., an audit record) from the factbase after all the other rules have had an opportunity to examine it. This rule also produces a default result of *SAFE*. Its priority is set to -97. Any new rules should have a higher priority, that is -96 or more.

set_time Sets the rulebased component's internal time based on the audit record timestamp. This time is used by some rules for expiring facts.

ClearSession Removes a session fact if no audit records have been received for the user for about a week. The next time the user executes a command, a new session fact is created.

TouchSession Updates the timestamp in the session fact for that user.

5.5.4.2 Bad Password Rules

Five rules are used to detect bad passwords. A bad password event occurs if a known user name was given and a bad password was given, or if a password authentication failed from within a program. The GoodPassword rules remove bad-password facts when a good login

Default NIDES Rules	
AccessPrivateDevice	Leapfrog1
AccessPrivateFile1	LinkSystemExec
AccessPrivateFile2	LocalLogin
AccessSpecialFile	Logout1
BackwardsTime	Logout2
BadLoginl	ModSystemExec
BadLogin2	MultLogin1
BadLoginAnomaly	MultLogin2
BadLoginBadPassword	NoRemote
BadPassword1	ParanoidUser1
BadPassword2	ParanoidUser3
BadPasswordAnomaly	ParanoidUserAnom
BadRoot	PasswordFileAccess
BadUserExec	ReadSystemExec
BrokeRoot	RemoteExec
ChangeLoginFile	RemoteFile1
ChmodOtherUser	RemoteFile2
ChmodSystemFile	RemoteFile3
ClearParanoidUser	RemoteLogin
ClearSession	RemoteMount1
configured	RemoteMount2
ConsoleLogin	RemoteRootBadLogin
DialInLogin	RemoteRootBadPassword
DotFile	remove_event
Exec	RunsRareExec
FTPAnomaly	set_time
FlagRSH	SpecUserExec
GoodLogin1	Su1
GoodLogin2	SuspiciousUser
GoodPassword	TFTPAnomaly
GoodPassword	TFTPUse
GoodSU1	TouchSession
GoodSU2	TrojanHorse
InvisibleDirectory	TruncateLog
KnownLogin1	

Table 5.11: NIDES Default Rules

comes in for a given user. The first GoodPassword rule generates an alert because it checks for users who have had several bad password authentications followed by a good login.

BadPassword1 This rule creates a 'bad_password ' fact in the factbase. This bad-password fact stores information about the bad password attempt: the host reporting it, the terminal, the remote host from which the bad password came, and the sequence number of the audit record.

BadPassword2 If a bad_password fact exists for this user and the user generates another bad password, this rule increments the count of bad passwords contained in the bad_password fact. It counts bad passwords only until the threshold for reporting a login attack is reached, and then the BadPasswordAnomaly fires instead.

BadPasswordAnomaly If more than the threshold number of bad passwords have been received for a given user without a successful one, this rule fires, reporting the threshold number of bad password entries.

GoodPassword1 When a good login comes in after a user has had several bad password authentications, this rule reports an alert. The rule is reporting the possibility that a login attack has succeeded.

GoodPassword2 This rule fires if a login has been successful after one or more (but less than the threshold number of) bad passwords have been received. It removes the bad_password fact, effectively causing NIDES to forget about previous bad passwords for that user.

5.5.4.3 Bad Login Rules

Five rules are used to detect bad logins. A bad login event occurs if a bad user name was given during a login attempt. The GoodLogin rules remove bad_login facts when a good login comes in for a given user. The first GoodLogin rule generates an alert because it checks for users who have had several bad logins followed by a good login.

BadLogin1 This rule creates a 'bad_login 'fact in the factbase. This bad_login fact stores information about the bad login: the host reporting it, the terminal, the remote host from which the bad login came, and the sequence number of the audit record.

BadLogin2 If a bad_login fact exists for this user and the user generates another bad login, this rule increments the count of bad logins contained in the bad_login fact. It counts bad logins only until the threshold for reporting a login attack is reached, and then the BadLoginAnomaly fires instead.

BadLoginAnomaly If more than the threshold number of bad logins have been received for a given user without a successful one, this rule fires, reporting the threshold number of bad logins.

GoodLogin1 When a good login comes in after a user has had several bad logins, this rule reports an alert. The rule is reporting the possibility that a login attack has succeeded.

GoodLogin2 This rule fires if a login has been successful after one or more (but less than the threshold number of) bad logins have been received. It removes the bad_login fact, effectively causing NIDES to forget about previous bad logins for that user.

5.5.4.4 Bad Login/Password Combination Rules

One rule is used to manage duplicate bad password events — BadLoginBadPassword.

BadLoginBadPassword The Sun C2 audit system and the UNIX accounting system both provide audit records to NIDES. These two systems report the same bad password event in two different forms. This rule attempts to merge bad password events reported by Sun C2 auditing and bad login events generated by the UNIX accounting system when it sees a bad password.

5.5.4.5 Login Rules

Twelve rules address login scenarios. The login rules notice logins and attempt to classify them. The twelve rules are categorized into four subsets: general logins, rsh, logouts, and special logins.

5.5.4.5.1 General Logins Six rules are considered general login rules. The general login types are *console, local, dialin, local_network, remote_network,* and *multiple.* All these rules look for the same event, a login. The general login rules are mutually exclusive; they use and check for the LOG mark on a fact to prevent the other general login rules from firing on the same fact. Using rule priorities and marks allows some economy in the test conditions. Once a login is found, the appropriate rule creates a session fact, and annotates it with the user id, the name of the host to which the login was directed, the host from which it came, the terminal id, and the login type. If the login type is remote (i.e., from a remote network), then the session is marked as suspicious.

ConsoleLogin This rule finds logins from the console.

DialInLogin This rule finds dialin (i.e., serial-port) logins. These are deduced from the fact that the remote host is the same as the host to which the login is going, but the terminal is not the console.

LocalLogin Finds logins from the local network by examining the rhost field of the event fact to determine if the domain name in that field matches a configured local domain name. Local domain names are configured using the `rb-config` file section DOMAIN (see page 70).

RemoteLogin The RemoteLogin rule sees logins from non-local machines. If none of the other login rules fire, this rule fires, treating the login as remote. The session that this rule creates is marked as suspicious. This suspicious marking is used by the aggregate suspicious behavior rules, which report alerts if a user exhibits multiple instances of suspicious behavior (see Section 5.5.4.11.3 on page 94).

MultLogin1 This rule fires if the user makes another login, this one coming from a remote location (regardless of where the original login came from). It increments the count of logins maintained by the session fact, marks the session as a remote login, and marks it as suspicious (all remote sessions are suspicious).

MultLogin2 This rule notices multiple logins if MultLogin1 did not fire. It increments the count of logins that the session fact maintains.

5.5.4.5.2 Rsh One rule notices users obtaining access to a host via an rsh mechanism.

FlagRSH This rule detects a user coming in through rsh rather than through a login process and marks the session as suspicious. See Section 5.5.4.11.3 (page 94) for a discussion of suspicious behavior rules.

5.5.4.5.3 Logouts Two rules track a user's logouts, maintaining a count of the number of login events for a session.

Logout1 If the user logs out, this rule decrements the login count in the corresponding session fact.

Logout2 This rule deletes session facts when the login count goes to zero.

5.5.4.5.4 Other Login Rules Three rules track special kinds of logins.

KnownLogin1 This rule checks for logins using known security holes (e.g., default user accounts with well-known default passwords). The login names checked are configured using the `rb_config` file section KNOWN_LOGIN (see page 72).

Leapfrog1 This rule checks for a user logged in remotely and executing one of the commands (telnet, rlogin or rsh) to login to another remote machine.

Exec If the user executes a program, but no session fact exists for this user, the login has been missed. This rule creates a session fact for the user as if the user had logged in.

5.5.4.6 Trojan Horse Rules

Two Trojan horse rules check for someone creating files or programs that will be executed involuntarily by another user.

TrojanHorse This rule checks for the user executing system programs from other than the list of directories specified by the `rb_config` file section PROGLOCATION (see page 74). For example, if the user executes a program named 'ls' that resides in the directory /home/users/user1 (and this directory is not listed in the PROGLOCATION list), this rule produces a "possible Trojan horse" alert.

ChangeLoginFile This rule checks for a user modifying another user's login configuration files. The list of those files is configured in the `rb_config` file section LOGIN_CONFIG (see page 72).

5.5.4.7 File and Device Access Rules

Eleven rules check for access to various files or devices that are either private or sensitive with regard to system operation.

AccessPrivateFilel This rule checks for one user accessing a private file in another user's home directory. The list of private files is configured via the PRIVATE_FILE section in the `rb_config` file (see page 74).

AccessPrivateFile2 This rule checks for a user accessing someone else's mail spool file. These files are found in /var/spool/mail.

AccessSpecialFile This rule checks for unauthorized access to a configurable set of files defined in the `rb_config` file section SPECIAL-FILE (see page 77). A file to be protected is given a type id number; a user allowed to access that file is given the same type id number in the USER-TYPE `rb_config` file section (see page 79). Multiple files and multiple users can have the same type id number. This allows NIDES to report access violations on a set of files by users not in a specified group. In addition, a single user can have multiple type id entries. This allows NIDES to recognize that the user has access to multiple types of files.

PasswordFileAccess This rule checks for access to the shadow password file. Only the password authorization daemon should access this file. This rule generates an alert if any other user touches that file.

ModSystemExec This rule reports modification of system executables located in system directories listed in the `rb_config` section PROGLOCATION (see page 74).

ReadSystemExec Users ordinarily should not read system executables. This rule reports users doing so. Since scripts must be read in order to be executed, the rule ignores reads of any files listed in the `rb_config` file section SYSTEM-SCRIPTS (see page 78).

BadUserExec This rule fires if any user other than root executes any of the programs listed in the `rb_config` file section NOEXEC (see page 73).

RunsRareExec This rule reports users running any programs listed in the `rb_config` file section RAREEXEC (see page 75). Certain rarely run programs can raise suspicions if used.

ChmodOtherUser This rule reports a user changing the permissions on a file in another user's home directory. Users and home directories are listed in the `rb_config` file section HOME_DIR (see page 72).

ChmodSystemFile This rule reports a user changing the permissions on a file in a system directory listed in the `rb_config` file section PROGLOCATION (see page 74).

AccessPrivateDevice This rule checks for access to a private device except by a user logged into the console. The private devices checked are configured in the `rb_config` file section PRIVATE_DEVICE (see page 73) — for example, /dev/audio (microphone) or /dev/fb (the screen's frame buffer). Accessing these devices could allow eavesdropping.

5.5.4.8 Remote User Rules

Nine rules monitor the actions of remote users who are more restricted than local users. The remote user rules are grouped into six categories: Remote File Access, Remote Execution, Remote Authorization, Remote Root, and Remote Mount.

5.5.4.8.1 Remote File Access

RemoteFile1 This rule reports a remote user modifying sensitive files. The sensitive files checked are those configured in the `rb_config` file REMOTE_FILE_NO_MODIFY section (see page 76).

RemoteFile2 This rule reports a remote user accessing any sensitive files. The sensitive files checked are those configured in the `rb_config` file REMOTE_FILE_NO_ACCESS section (see page 76). For example, this rule finds a remote user accessing a `.rhosts` file (if it is included in the `rb_config` file) no matter what directory it resides in.

RemoteFile3 This rule reports a remote user modifying a file in /etc or /usr/etc. These directories contain many sensitive configuration files that a remote user should not modify.

5.5.4.8.2 Remote Execution

RemoteExec This rule reports a remote user executing a sensitive program. The sensitive programs checked are those configured in the `rb_config` file REMOTE_NO_EXEC section (see page 76).

5.5.4.8.3 Remote Authorization

NoRemote This rule reports a remote login from a user who is not allowed to log in remotely. The list of prohibited users is configured in the **`rb_config`** file REMOTE-NOT-OK section (see page 77).

5.5.4.8.4 Remote Root In general, a user trying to obtain root access from a remote system is suspicious. Two rules monitor attempts to gain root access from a remote site.

RemoteRootBadLogin This rule reports users trying to log in as root from remote sites, and failing. This rule responds to events generated from accounting data. The rule determines if the login is remote using the **`rb_config`** file DOMAIN section (see page 70). All items in the DOMAIN list are local; anything not in the list is remote.

RemoteRootBadPassword This rule reports users trying to log in as root from remote sites and failing. This rule responds to events generated from the Sun UNIX BSM/C2 data. The rule determines if the login is remote using the **`rb_config`** file DOMAIN section (see page 70). All items in the DOMAIN list are local; anything not in the list is remote.

5.5.4.8.5 Remote Mount These rules check for a machine from a remote site attempting to mount the filesystem of a monitored machine. Note that these rules depend on the mount program being modified to audit remote mounts.

RemoteMount1 This rule reports successful remote mount attempts.

RemoteMount2 This rule reports unsuccessful remote mount attempts.

5.5.4.9 User ID Rules

Six rules monitor and observe changes in user identity. Most involve the root account.

Su1 This rule notes a user executing an **`su`** action. The user's session is marked as being suspicious (see the aggregate suspicious behavior rules on page 94).

GoodSU1 This rule reports a user successfully executing an **`su`** action when more than two bad root password attempts for this host have been seen without a successful password authorization.

GoodSU2 This rule removes any bad_password facts for root after a successful password authorization.

BrokeRoot If the user is able to get root privilege without executing an **`su`** action, this rule notes it and reports it.

BadRoot This rule reports unauthorized use of root privilege. This rule fires if any user not listed in the **`rb_config`** file section ROOT-OK (see page 77) executes a program when the current user id is root.

SpecUserExec This rule reports when a special user executes a program that they are not allowed to execute. The list of users and allowable programs, if any, are in the **`rb_config`** file SPECIAL-USER section (see page 78).

5.5.4.10 FTP

Three rules observe FTP activity.

FTPAnomaly This rule reports an ftp user accessing any file outside the **`/ftp`** directory.

TFTPUse This rule notes a user executing tftp.

TFTPAnomaly This rule reports any tftp access to a program outside the **`/tftpboot`** directory.

5.5.4.11 Suspicious Behavior

Twelve rules address suspicious user behavior. They are grouped into three categories: Hiding Tracks, Paranoia, and Aggregate Suspicious Behavior.

5.5.4.11.1 Hiding Tracks Many intruders attempt to hide their tracks during a break-in. The Hiding Tracks rules look for behavior that would indicate a user attempting to cover up activities.

LinkSystemExec This rule reports creation of links to system executables. Intruders make these links to hide what they are doing.

InvisibleDirectory This rule notes, and marks as suspicious, the creation of an invisible directory in which an intruder could hides files. See page 94 for a description of the aggregate suspicious behavior rules that utilize the suspicious mark.

DotFile This rule reports users writing invisible files (files whose names begin with a '.') into temporary directories. The set of temporary directories is configured via the **`rb_config`** file sections TMP_DIRNAME and TMP_FILE (see pages 79 and 79). Some programs, such as window managers, normally write invisible files into temporary directories. Using the **`rb_config`** file to list only those directories that are suspicious prevents NIDES from generating an alert in all other cases.

TruncateLog This rule reports users covering tracks by truncating log files. If a user creates or deletes any files in the directories listed in the **`rb_config`** file section LOG_DIR (see page 72), the rule generates an alert.

BackwardsTime This rule looks for an audit record whose timestamp is significantly earlier than the previous audit record's timestamp. An intruder sometimes does this to hide file modifications, making it look like the modified file was written at the same time as the original file.

5.5.4.11.2 Paranoia Users who execute programs like 'who', 'ps' or 'finger' may worry that someone will log on and notice them. This may indicate that the user is an intruder or at least is up to no good. This set of rules notes the user executing any of the programs configured in the `rb_config` file PARANOID_PROG section (see page 73). The rules count instances of this behavior and report an alert if the behavior is too frequent.

ParanoidUserl, ParanoidUser2 This rule notes the first instance of 'paranoid' behavior, creating a fact to keep track of this behavior for the user.

ParanoidUser3, ParanoidUser4 This rule notes second and further instances of paranoid behavior.

ParanoidUserAnom This rule reports an alert if the user executes paranoid programs at a rate greater than five per minute.

ClearParanoidUser This rule cleans the slate (that is, gets rid of the paranoid fact) if the user hasn't executed a paranoid command in more than 1 hour.

5.5.4.11.3 Aggregate Suspicious Behavior This rule tries to decide whether a user is acting suspiciously. We have a set of criteria for suspicious behavior: remote logins, multiple remote logins, obtaining root privilege, using rsh, using any of the 'rarely executed' programs, being paranoid, or creating an invisible directory.

SuspiciousUser This rule reports a user having more than ten suspicious events, divided between at least two of the suspicious categories.

5.6 Rulebase Syntax Diagrams

The syntax diagrams for the NIDES rulebase rule specification language may be useful for writing new rules for NIDES. The diagrams use fairly standard conventions. For example,

<object a> ::= <object b>

means that the object on the left has the syntax shown on the right. Literals (or keywords) are in bold type. These include brackets, bars, and plus signs. The same symbols in non-bold type have different meanings, as follows:

- | (non-bold vertical bar) indicates alternatives; that is, we would say

```
<statement>     ::=  <ptypedef>
                  |   <ruledef>
                  |   <comment>
```

  to indicate that a statement can be a ptypedef, a ruledef, or a comment.

- * (non-bold superscript asterisk) indicates that an item can occur as many times as desired, but need not appear. For example,

```
<ante>          ::=  [<clause>]*
```

  says that an antecedent consists of zero or more clauses.

- + (non-bold superscript plus) indicates that an item can occur as many times as desired, but must appear at least once. For example,

```
<cons>          ::=  [<action>]+
```

  says that a consequent must consist of at least one action clause, though it may have as many more as desired.

- [ ] (non-bold brackets) surrounding an item are used to indicate grouping. In the examples above, the brackets indicate that the asterisk and plus apply to whatever is inside the brackets. If the brackets surround an item, but are not followed by an asterisk or plus, they indicate that the item may occur zero or one times. For example,

```
<marktest>      ::=  $[<name>]
                  |   ^[<name>]
```

  indicates that a mark test has an optional name after the dollar sign or up caret.

Several abbreviations are used, including

- relop — relational (comparison) operator
- assop — assignment operator
- funcall — C language function call

The syntax diagrams are as follows.

```
<ptypedef>      ::= ptype [ <name> <fields> ]

<fields>        ::=  <field>[,<field>]*

<field>         ::=  <name>
                 |   <name>:<typename>
```

```
<typename>      ::=  integer
                |    float
                |    list
                |    string
                |    symbol
                |    setname

<ruledef>       ::=  rule [ <name> [(<opts> )] : <ante> ==> <cons> ]

<opts>          ::= <opt>[;<opt>]*

<opt>           | <rank>
                |   <repeat>
<rank>          ::= # <integer>

<repeat>        ::=  *

<ante>          ::=  [<clause>]*

<clause>        ::=  [+<pname>[<marktest>][<restrictions>]]
                | [-<name>[<marktest>][<restrictions>]]
                | [?<restrictions>]

<pname>         ::=  <name> : <name>
                |    <name>

<marktest>      ::= $[<name>]
                |    *[<name>]

<restrictions>  ::=| <restricts>

<restricts>     ::= <restrict>[,<restrict>]*

<restrict>      ::= <expr> <relop> <expr>
                |    <expr>
```

```
<expr>          ::=  ( <expr> )
                 |   <expr> + <expr>
                 |   <expr> - <expr>
                 |   <expr> * <expr>
                 |   <expr> / <expr>
                 |   <expr> % <expr>
                 |   <expr> >> <expr>
                 |   <expr> << <expr>
                 |   <expr> & <expr>
                 |   <expr> ^ <expr>
                 |   <expr> | <expr>
                 |   <expr> && <expr>
                 |   <expr> || <expr>
                 |   <pvalue>

<relop>         ::=  ==
                 |   !=
                 |   >
                 |   <
                 |   >=
                 |   <=

<pvalue>        ::=  <name>
                 |   '<name>
                 |   <integer>
                 |   <real>
                 |   <string>
                 |   <pfield>
                 |   <list>
                 |   <setref>
                 |   <funcall>
```

```
<assop>          ::=  =
                  |    +=
                  |    -=
                  |    *=
                  |    /=
                  |    %=
                  |    >>=
                  |    <<=
                  |    &=
                  |    ^=
                  |    |=
                  |    %=

<pfield>         ::=  <name> . <name>

<list>           ::= [ [<pvalue>]* ]

<setref>         ::=  <name> # <name>

<funcall>        ::= <name> ( [<arglist>] )

<arglist>        ::= [<pvalue>],+

<cons>           ::=  [<action>]+

<action>         ::= [ <negate> ]
                  |    [ <mark> ]
                  |    [ <unmark> ]
                  |    [ <assert> ]
                  |    [ <modify> ]
                  |    [ <execute> ]

<negate>         ::=  -| [<name>]+,

<mark>           ::=  $| <name> [: <name> ]

<unmark>         ::=  ^| <name> [: <name> ]

<assert>         ::= + <name> [<arestricts>][<rcf>]
```

```
<modify>        ::=  /  <name>  [<arestricts>][<rcf>]

<arestricts>    ::=  | <arestrict>[,<arestrict>]*

<arestrict>     ::=  <expr> <assop> <expr>
                 |   <expr>

<execute>       ::=  ! | [<exstat>]+,

<exstat>        ::=  <xassign>
                 |   <funcall>

<xassign>       ::=  [']<name> <assop> <expr>

<rcf>           ::=  ~ <real>

<name>          ::=  [A-Za-z]+[A-Za-z0-9_]*

<string>        ::=  " [<any character>]* "

<integer>       ::=  [0-9]+

<real>          ::=  [0-9]*.[0-9]*

<comment>       ::=  ` [<any character>]* <newline>
```

Chapter 6

Reference Manual

6.1 NIDES Host Configuration

Before you can run NIDES on your system, several configuration procedures must be completed. Most procedures should be done when your NIDES software is installed. If you encounter problems while running NIDES, make certain that all items are installed properly. If you are concerned about your NIDES installation or you have any software problems, consult your system administrator. For additional information on configuration of the NIDES host and target hosts, refer to Chapter 9.

6.1.1 X Windows

We recommend using the X11R5 version of X windows and the *twm* window manager, which is included with the standard release of X.

6.1.2 Host Software

NIDES should be installed on a system that will not be a target host (audit data provider) to NIDES. When your NIDES system was installed, all required software should have been loaded on the NIDES host. Instructions for installing NIDES software are in Chapter 9.

6.1.3 IDES-ROOT

Prior to starting any processes on your NIDES host you must set your IDES-ROOT environment variable to the NIDES top-level root directory. This information is used by the various NIDES processes in finding files, executables and configuration information. To set the IDES_ROOT environment variable, type the following command on the NIDES host computer:

```
unix_prompt% setenv IDES_ROOT /ides_top_level_directory
```

If you run the `nides_init` script in your *.cshrc* file, this variable is set automatically; this procedure is explained in Chapter 9. The `ides_top_level_diretory` should be the directory where NIDES was installed on your system.

6.1.4 IPC Nameserver

Because NIDES is a distributed system, all NIDES programs running on the NIDES host need to know the IPC nameserver's location. To tell the NIDES processes this name, you should have an environment variable set to identify the nameserver. To set the nameserver, type the following command on the NIDES host computer:

```
unix_prompt% setenv IPC_NAMESERVER nideshost:7001
```

If you run the `nides_init` script in your *.cshrc file,* this variable is set automatically, this procedure is explained in Chapter 9. The `nideshost` should be the hostname where you will be running the NIDES processes. The number following `nideshost:` should be the number of a free TCP port. We recommend using 7001 if it is available. If the port number you select is not free, you will get an error when you run the *ipc_nameserver* program. You can set the port number to any number under 65535. Keeping the number in the 7000s is recommended.

In addition to setting your IDES-ROOT and IPC_NAMESERVER environment variables, make sure a process called *ipc_nameserver is* running on the NIDES host. If this process is not running, you can start it with the following command; be sure you set the IDES-ROOT and IPC_NAMESERVER variables before starting the nameserver:

```
unix_prompt% $IDES_ROOT/bin/bin.sun4/ipc_nameserver &
```

6.1.5 Target Host agend Processes

Every potential target host system (that is, a system that may provide audit data to NIDES) must have an *agend* daemon process running on it prior to execution of NIDES and an attempt to receive audit data from the target host. Your system administrator should modify your system's *rc.local* file to run *rc.nides_target*, which will automatically start up the daemon at boot time. If the *agend* daemon is not running on a machine that you want to use as a target, you can start it manually on the target host by logging in as user audit on the target host and issuing the following command:

```
unix_prompt% agend
```

The *agend* daemon does not need to run on the NIDES host, only on target hosts that provide audit data to the NIDES host.

6.1.6 Analysis Component Default Configurations

During NIDES initial installation, default configurations for the statistical and rulebased analysis components are installed. It is highly recommended that you review these default configurations and modify them to suit your environment prior to running NIDES. A discussion of the default configuration begins on page 164.

6.2 Main Window and Privileged Commands

Figure 6.1 shows the window that is displayed when NIDES is first executed. All NIDES functions are invoked from the Main Menu. At the top of the window are seven menu items with various functions:

- **SetUp** – Activates/deactivates the NIDES analysis and audit record collection processes, selects target hosts, and configures alert methods and filters
- **Monitor** – Displays the status of the real-time NIDES audit data processing and intrusion detection functions
- Browse – Displays audit data, real-time and test result data, and instance configurations
- **Customize** – Configures the real-time NIDES analysis components and creates and configures instances and audit data sets that can be used in running NIDES experiments
- **Experiment** – Sets up, executes, and displays the status and results of NIDES experiments
- **Quit** – Exits the NIDES program
- **Help** – Accesses help about the NIDES Main Menu window functions

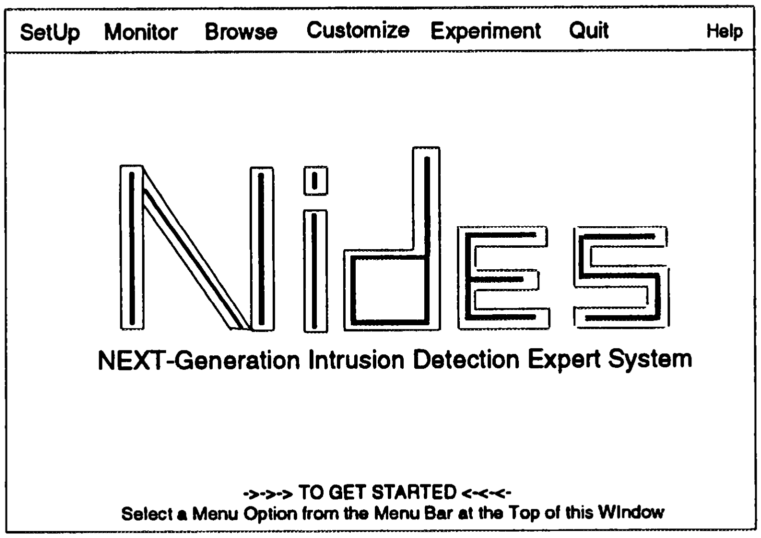

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.1: NIDES Main Window

Privileged Commands Some NIDES options can be invoked only by users configured as privileged by the NIDES system administrator. Table 6.1 lists all the NIDES menu functions and shows which functions are privileged. When you start up NIDES, if you are not a privileged user, the following message is displayed in the xterm window where NIDES was invoked:

```
Non-privileged user, limited capability
```

NIDES continues to run and you will be able to execute all the commands that are non-privileged.

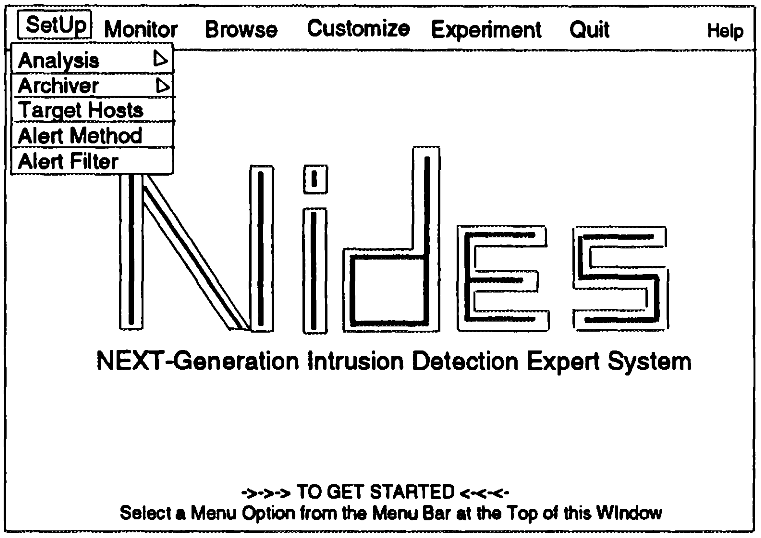
If you need to access a privileged function, have your system administrator add you to the NIDES privileged-user list.

NIDES Menu Item	Privileged	Non-privileged
SetUp Menu		
Analysis-START		X
Analysis-STOP		X
Archiver-START		X
Archiver-STOP		X
Target Hosts		X
Alert Method		X
Alert Filter		X
Monitor Menu		
System		X
Targets		X
Browse Menu		
Audit Data		X
Live Results		X
Test Results		X
Pending Reconfig		X
Customize Menu		
Live Instance		
Profile Modification	X	
Measure Parameters	X	
All Other Items		X
Test Instances	X	
Audit Data Sets	X	
Experiment Menu		
Setup and Exec	X	
Status and Results	X	
Quit Menu		
QUIT		X
Help Menu		
HELP		X

Table 6.1: NIDES Privileged and Non-privileged Functions

6.3 SetUp Menu

The SetUp Menu of the Main Window contains commands to start/stop NIDES, start/stop audit data archival, select target hosts, select the alert reporting method, and configure alert filtering. Figure 6.2 shows the SetUp Menu. Some of these options are activated or deactivated, depending upon the current status of the NIDES analysis and arpool servers.



NEXT-Generation Intrusion Detection Expert System

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.2: NIDES Main Window SetUp Menu

When NIDES first starts up, the Analysis and Alert Method options are the only active options on the SetUp Menu. Once the servers are running, the other options are activated. If the text is faint, the option is not currently active.

6.3.1 Analysis Option

The Analysis option of the SetUp Menu is a walking menu, shown in Figure 6.3, with two mutually exclusive options — START and STOP. When NIDES first starts, the analysis components are not running, START is selectable and STOP is deactivated. The START option initiates NIDES real-time analysis, invoking the analysis components (statistics, rulebased, and resolver) and arpool component. Once NIDES analysis has been started, you can reconfigure the real-time instance with any initial configuration changes. Then target hosts should be activated to provide data to NIDES. When NIDES analysis is running, the STOP option is selectable and the START option is deactivated.

6.3.2 Archiver Option

NIDES includes an archival function that stores audit data received by NIDES into an archive. Data from the archive can be retrieved by using the Audit Data option of the Browse Menu. Because the NIDES archive can grow quite large, the NIDES system administrator must implement regular back-up and purge procedures to ensure that the NIDES disk does not get full. If data archival is

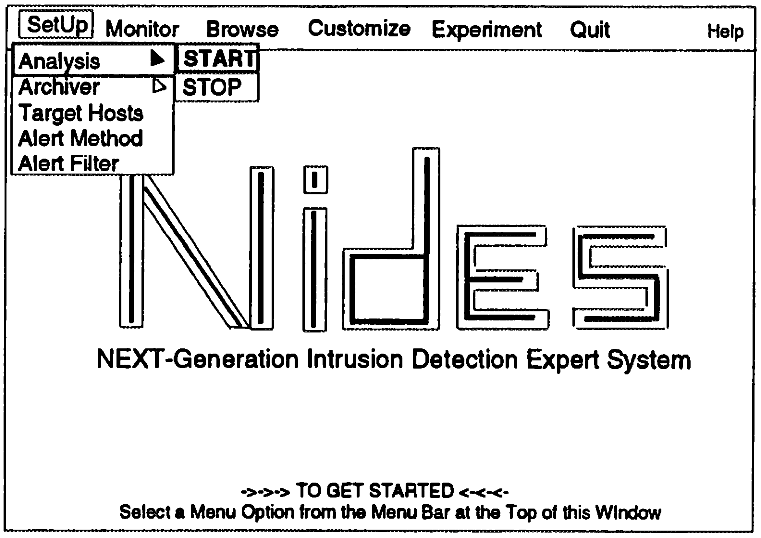

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
Analysis
Archiver
Target Hosts
Alert Method
Alert Filter
START
STOP
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.3: SetUp Menu Analysis Option

not required, the archiver should be switched OFF. If you choose to use the Archiver option, you should not also store audit data on the target systems.

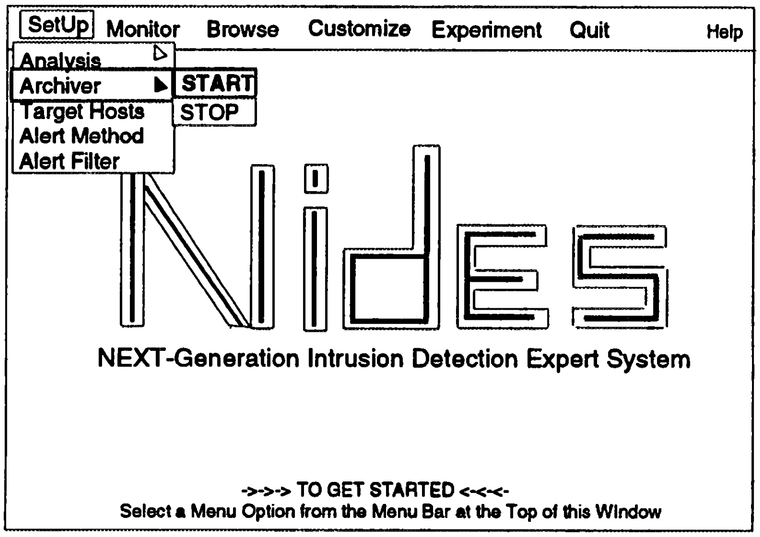

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
Analysis
Archiver
Target Hosts
Alert Method
Alert Filter
START
STOP
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.4: SetUp Menu Archiver Option

The Archiver option of the SetUp Menu is a walking menu, shown in Figure 6.4, that contains two mutually exclusive options — START and STOP. When NIDES first starts, the archiver component is not running, and both START and STOP options are deactivated.

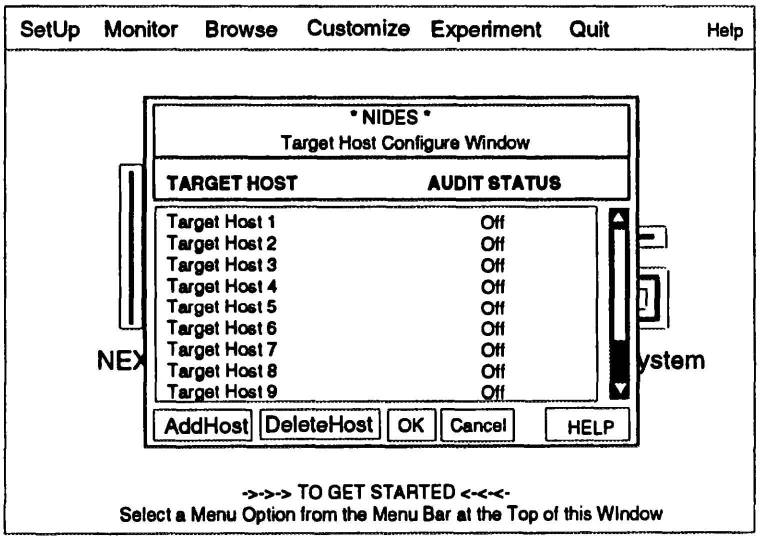
Once NIDES analysis has been started, selecting START initiates archival of all audit data received by the arpool component. When the archiver is running, the STOP option is selectable and the START option is deactivated.

6.3.3 Target Hosts Option

The Target Hosts option of the SetUp Menu allows you to add target hosts to the list of available target hosts, delete hosts from that list, and activate or deactivate audit data collection and transmission to NIDES for any target hosts in your list.

After NIDES analysis has been activated, the Target Hosts option is selectable. When the option is selected, the Target Host Configure Window as shown in Figure 6.5 is displayed. The window contains the list of currently available hosts and their audit status and a row of command buttons:

- **AddHost** — Allows you to add new target hosts to your list of available hosts. When NIDES is invoked for the first time, your target host list is empty so you will need to use this feature to enter the hosts you will be monitoring with NIDES
- **DeleteHost** — Allows you to delete target hosts from your list of available hosts
- **OK** — Executes the requested changes, including addition/deletion of hosts from your list and activation/deactivation of audit data collection on your target hosts
- **Cancel** — Cancels the target hosts configuration option without making any of the changes entered; you are returned to the Main Window
- **HELP** — Provides help information on the target hosts configuration features


SetUp Monitor Browse Customize Experiment Quit Help
* NIDES *
Target Host Configure Window
AddHost DeleteHost OK Cancel HELP
NEX
ystem

TARGET HOST	AUDIT STATUS
Target Host 1	Off
Target Host 2	Off
Target Host 3	Off
Target Host 4	Off
Target Host 5	Off
Target Host 6	Off
Target Host 7	Off
Target Host 8	Off
Target Host 9	Off

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.5: Target Host Configure Window

6.3.3.1 AddHost Option

When you select the *AddHost* option, an entry window as shown in Figure 6.6 is displayed. Click on the entry box, enter the host name, and select *OK* to add the host to your list. When a new host is added, its audit status is initially set to OFF. If you decide not to add the host, select *Cancel* to return to the Target Host Configure Window.

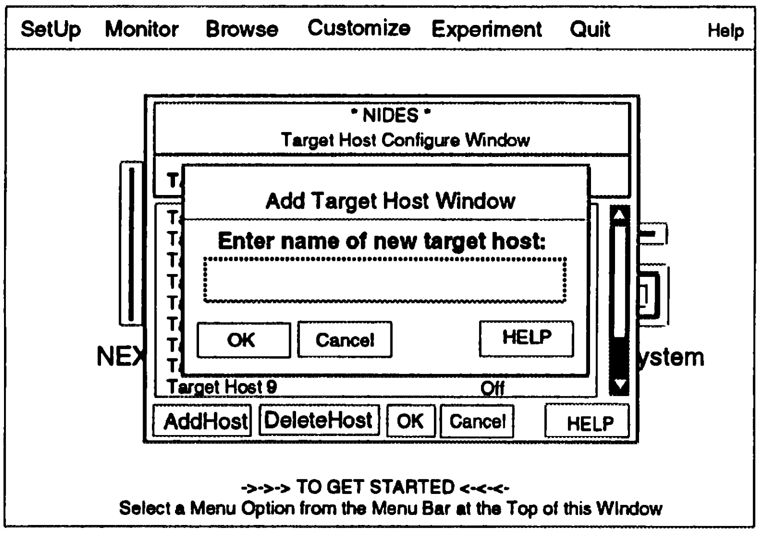

SetUp Monitor Browse Customize Experiment Quit Help
* NIDES *
Target Host Configure Window
Add Target Host Window
Enter name of new target host:
OK
Cancel
HELP
Target Host 9
Off
AddHost
DeleteHost
OK
Cancel
HELP
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.6: Add Target Host Window

6.3.3.2 Target Host Name Verification

When a new target host name is entered, it is verified in two ways. Only alphanumeric characters and a few special characters (_.-) are allowed. If the name entered passes this first test, NIDES then checks your system's host tables to see if the host is known to your network. If it is known, NIDES edits your entered name to match the host table primary entry. You may enter a host's alias, if it is listed in your host tables; NIDES will convert the name to the primary entry. If the host cannot be found in any of your system's host tables, an error is reported and your entry is not added.

6.3.3.3 DeleteHost Option

The *DeleteHost* option deletes hosts in your target host list. To delete a host,

1. Click on the host name. The host name is highlighted and the *DeleteHost* option becomes active.
2. Select *DeleteHost* to delete the host from your list. The host deletion does not become permanent until you select *OK*.

If you decide not to delete any of the hosts that you have removed from the list, select Cancel to return to the Main Window.

When you delete a target host from your target host list, it is possible that the target is switched ON and is currently providing data to NIDES. If this is the case, when the host deletion is made, the audit data collection on the host will be switched OFF.

6.3.3.4 Target Host Activation

Once you have started NIDES analysis and have added some target hosts to your target host list, you can activate audit data collection and transmission to NIDES on any of these target hosts. All hosts are added with their audit flags switched OFF. To turn a host's audit state to ON, click on the host's entry in the Target Host Configure Window to toggle the host's configuration. Select *OK* to initiate your changes. A confirmation window lists all the changes that will take place once you have confirmed them.

6.3.3.5 Confirming Changes

If you are satisfied with the configurations specified, select *OK* on the confirmation window. A message window informs you that your changes are being made. Messages will be sent to those targets that were switched ON to begin sending audit data to NIDES, and messages will be sent to those targets switched OFF to stop sending audit data to NIDES. Each target host that was previously sending audit data to NIDES and was deleted from the target host list will also be sent a message to stop sending data to NIDES. Once you have started audit data transmission on some target hosts, you may review the status of those hosts via the Browse Menu, Target Hosts option, which is described in Section 6.4.2.

If you decide not to keep all the changes made, select *Cancel* to return to the Target Host Configure Window where you may make additional changes. If you decide not to make any changes, select *Cancel* to return to the Main Window.

6.3.4 Alert Method Option

The SetUp Menu Alert Method option allows you to select and configure the real-time alert reporting mechanisms used by NIDES to notify you about anomalous events. If you do not need real-time alert reporting, do not select any alert mechanism; alert results generated by NIDES are archived and can be reviewed at any time. We recommend that you select at least one alert mechanism.

The Alert Mechanism Configure Window as shown in Figure 6.7 comprises two main areas:

1. A list of all the available alert mechanisms and their current configurations

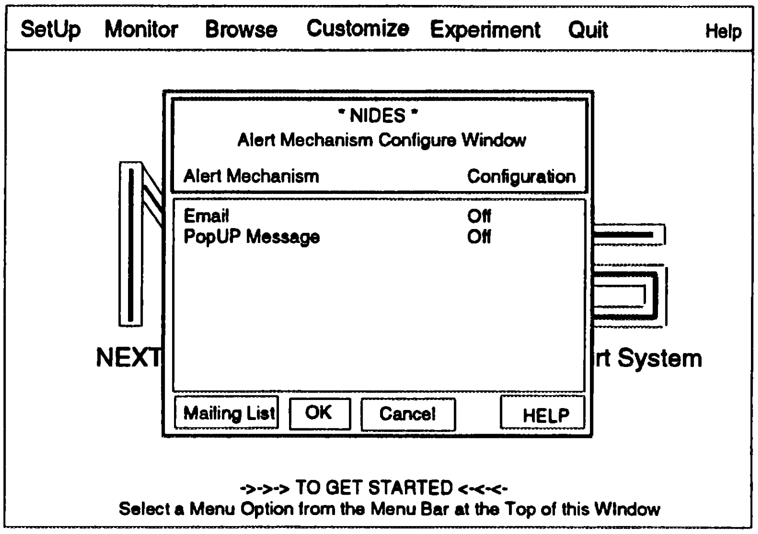
2. A panel of buttons, located below the display area

You may select one or more alert mechanisms listed in the window. Click on the mechanism to turn it ON or OFF. Select *OK* to record your alert choices; once confirmed, you are returned to the Main Window.

NIDES currently supports two methods of real-time alert notification: PopUp message reporting and e-mail reporting.

6.3.4.1 PopUp Window Alert Reporting

If you select the PopUp Message method, a PopUp Window is immediately displayed and a beep sounds each time NIDES generates an alert. The PopUp Window contains summarized information about the alert. If multiple alerts are generated, PopUp Windows are displayed in succession until all alerts have been reported. If you are often absent from your screen for more than an hour or so, we do not recommend use of the PopUp Message method, as you may have a large queue of Alert PopUp Windows when you return to the Main Window. You will have to acknowledge each PopUp Window before you continue any NIDES activities. However, if you will be sitting at or



* NIDES *
Alert Mechanism Configure Window

Alert Mechanism	Configuration
Email	Off
PopUP Message	Off

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.7: Alert Mechanism Configure Window

near the NIDES console most of the time, the PopUp Message is the most timely method for alert reporting. One possible way to use the PopUp Message method is to leave it ON while you are sitting at the console, and switch it OFF when the system is not attended.

6.3.4.2 E-mail Alert Reporting

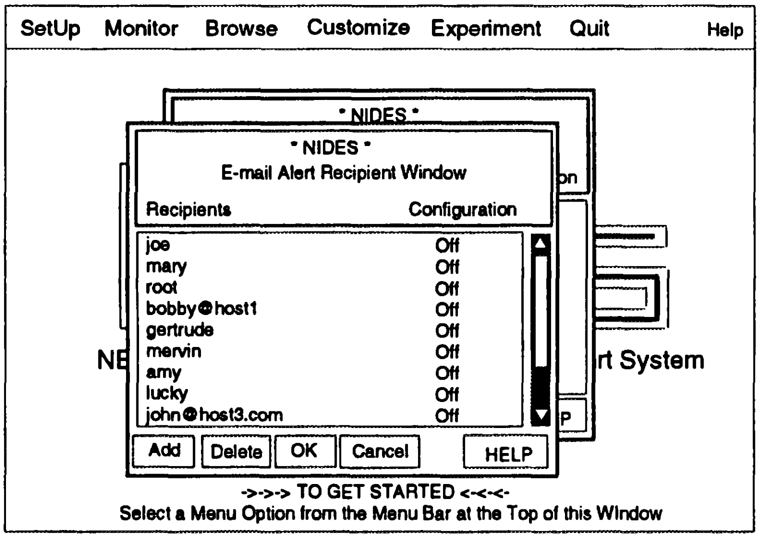
If you select the e-mail method of alert reporting, each time NIDES generates an alert, an e-mail message is sent to your selected recipients immediately following an alert. The e-mail message contains summarized information about the alert. If multiple alerts are generated, multiple e-mail messages are sent to your list of recipients. We recommend use of e-mail alert reporting if you will not be in front of your NIDES window consistently.

6.3.4.3 E-mail Recipients List

If you have turned ON the e-mail alert mechanism, you must configure a mailing list by selecting the *Mailing List* option on the Alert Mechanism Configure Window. An E-mail Alert Recipient Window is displayed as shown in Figure 6.8, with its two main areas. One area contains a list of potential e-mail recipients and their current configurations (ON or OFF). Below the recipient listing is a panel of buttons labeled Add, Delete, OK, Cancel, and HELP.

When you first start up NIDES, your mailing list will be empty and you will need to add names to the list. To do this, select Add. A window will be displayed as shown in Figure 6.9. To enter a recipient, click on the entry box and type in the name. Make sure the cursor is within the name entry window. Select *OK* to add the name to your list. Be sure to enter each recipient's full e-mail address.

To select the recipients of e-mail alerts, turn ON all recipients you want on your mailing list. To delete a recipient from your list, select the recipient's name, and then select *Delete.*


SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
* NIDES *
E-mail Alert Recipient Window
Recipients
Configuration

Recipients	Configuration
joe	Off
mary	Off
root	Off
bobby@host1	Off
gertrude	Off
mervin	Off
amy	Off
lucky	Off
john@host3.com	Off

Add
Delete
OK
Cancel
HELP
rt System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.8: E-mail Alert Recipient Window

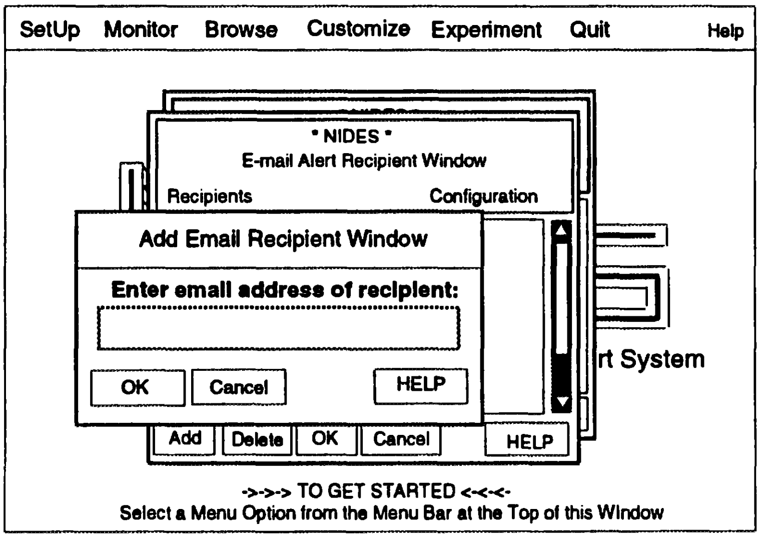

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
* NIDES *
E-mail Alert Recipient Window
Recipients
Configuration
Add Email Recipient Window
Enter email address of recipient:
OK
Cancel
HELP
Add
Delete
OK
Cancel
HELP
rt System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

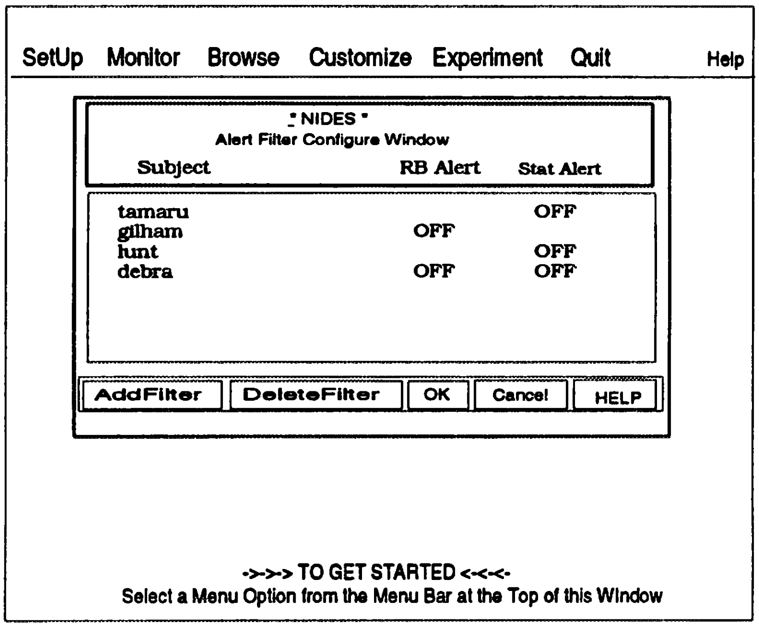
Figure 6.9: E-mail Alert Recipient Add Window

Once you have configured the mailing list, select *OK* to display a Confirmation Window summarizing the changes. Select *OK* to put your changes into effect and return to the Alert Mechanism Configure Window.

6.3.5 Alert Filter Option

The Alert Filter option of the SetUp Menu allows you to configure an alert filter that will make NIDES suppress real-time alert reporting on specific users by type of alert. Any alerts that are not reported via the chosen alert mechanism because of alert filtering are still included in the result archive, so no alerts are lost; they are simply not reported via a PopUp window or e-mail. You may add a filter for any user, and the filter can have one of three values:

- **Filter Statistics Alerts** — Suppresses reporting of statistical alerts; rulebased alerts are still reported
- **Filter Rulebased Alerts** — Suppresses reporting of rulebased alerts; statistical alerts are still reported
- **Filter Rulebased and Statistics Alerts** — Suppresses real-time reporting of all alerts, both statistical and rulebased


SetUp Monitor Browse Customize Experiment Quit Help
* NIDES *
Alert Filter Configure Window
AddFilter
DeleteFilter
OK
Cancel
HELP
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Subject	RB Alert	Stat Alert
tamaru		OFF
gilham	OFF	
hunt		OFF
debra	OFF	OFF

Figure 6.10: Alert Filter Configure Window

Alert filters can be configured only when the real-time analysis is running. Once you have started NIDES analysis, you may configure alert filtering by selecting the Alert Filter option of the SetUp Menu to display the Alert Filter Configure Window, as shown in Figure 6.10. When you first start up NIDES this window will be empty because no filters have been entered. You can enter in any filters you want to use.

The sample window shown in Figure 6.10 has three columns. The first column shows the subject whose alerts will be filtered. The next two columns show which types of alerts will be filtered, Rulebased and/or Statistics. If the alert will be filtered, the word OFF appears under the type of alert that is switched OFF. For example, for user tamaru, rulebased alerts will be reported but statistical alerts are switched OFF and hence will not be reported. For user debra, neither rulebased nor statistical alerts will be reported because both columns show alert reporting switched OFF.

6.3.5.1 Adding a New Alert Filter

To filter alerts, you must add users to the alert filter list. After you have brought up the Alert Filter Configure Window, via the Alert Filter option on the SetUp Menu, select *AddFilter.* A filter entry window will be displayed as shown in Figure 6.11.

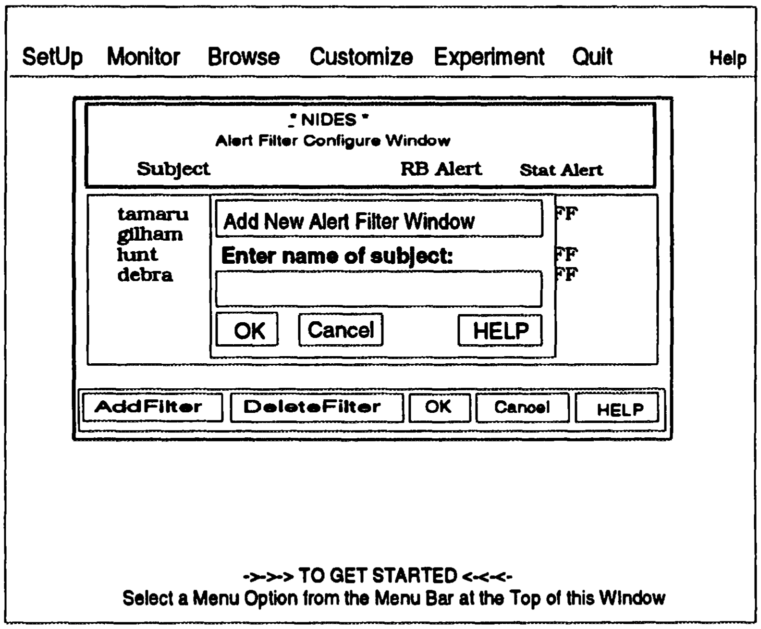


Figure 6.11: Add New Alert Filter Window

To add a filter, enter the name of the subject whose alerts you want to filter. The name entered should be the subject's computer account name as it appears in the audit trail. NIDES uses this name to match the subject name reflected in audit data it analyzes. If the name does not match, the alerts will not be filtered. When you enter a new subject into the alert filter list, the default configuration will have both statistical and rulebased alerts switched OFF; if you want to change the configuration so only statistics or only rulebased alerts are suppressed, click on the subject you want to modify. The clicking action serves as a three-way toggle, between the three possible filter states (rulebased filter, statistical filter, or both filtered).

6.3.5.2 Deleting an Alert Filter

If you have a subject in your Alert filter list whose alerts you no longer want to filter, select the subject and then select *DeleteFilter.*

6.3.5.3 Alert Filter Activation

Once you have made all your Alert Filter changes, select *OK* on the Alert Filter Configure Window to begin activation of the new alert filters. A confirmation window is displayed listing changes made. Once confirmation is made, all alerts produced by NIDES are filtered using the current filter configuration. If you decide you do not want to confirm the changes, select *Cancel* to return to the Alert Filter Configure Window where you can make additional changes to the Alert Filters. If you again select *Cancel* you will be returned to the Main Window and your alert filters will be unchanged.

If the NIDES user interface is exited then restarted any previous alert filter configuration is lost and the alert filter list is empty.

6.4 Monitor Menu

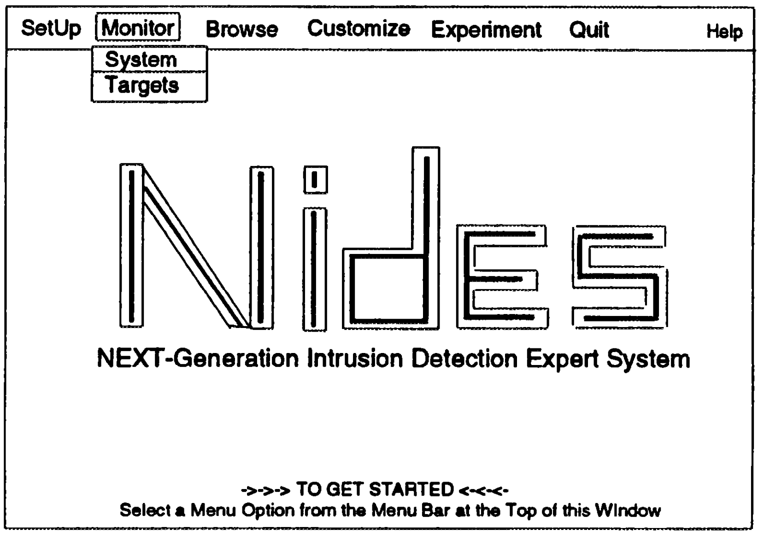

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
System
Targets
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.12: NIDES Main Window Monitor Menu

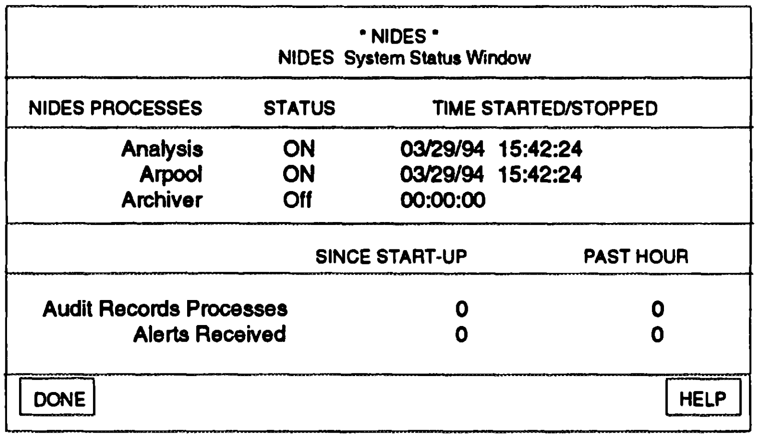
The Monitor Menu contains options that display system and target host status information. Unlike most of the windows that are part of the NIDES interface, these windows can be displayed at any time, and while they are displayed other windows may also be accessed.

Figure 6.12 shows the Monitor Menu, which has two options:

- **System** — Displays status of the NIDES real-time audit data analysis and intrusion detection functions

- **Targets** — Displays status of all the target hosts known to NIDES, regardless of whether they are currently providing audit data to NIDES

6.4.1 System Option



```
                         * NIDES *
               NIDES System Status Window

NIDES PROCESSES     STATUS      TIME STARTED/STOPPED

       Analysis      ON         03/29/94  15:42:24
         Arpool      ON         03/29/94  15:42:24
       Archiver      Off        00:00:00

                         SINCE START-UP        PAST HOUR

Audit Records Processes         0                  0
         Alerts Received        0                  0

DONE                                                HELP
```

Figure 6.13: NIDES System Status Window

Figure 6.13 shows the window displayed when the System option is selected from the Monitor Menu. This window contains information about the real-time audit data processing and intrusion detection components of NIDES.

6.4.1.1 System ON/Off Status

The top part of the window lists the three primary NIDES processes that are run during real-time analysis:

- **Analysis** — The statistical analysis, rulebased analysis, and resolver components. When real-time analysis is initiated, these three processes are started.
- **Arpool** — Interacts with the various target host processes; arpool (audit record pool) collects the audit data from all target hosts and coalesces it into a single audit record stream. The analysis processes obtain their audit data from the arpool process. When NIDES real-time analysis is initiated, the arpool process is started.
- **Archiver** – An optional process that archives the audit data collected by arpool. The archiver is started and stopped separately from the analysis and arpool processes. The archiver must have analysis running in order to function.

The top part of the window displays the status of each process (ON or OFF) and the time the process status changed (i.e., the time the process was last started or stopped).

The status and times for each process listed will change under two conditions. If the user activates or deactivates the process, the status and time are updated. If the process encounters an error and is shut down, the status automatically switches to OFF and the time is set to the time the process is terminated. If this occurs, the user can usually restart the processes via the SetUp Menu.

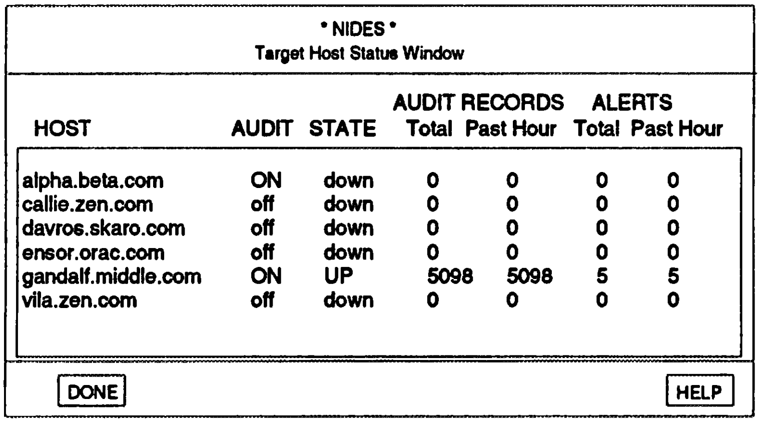
6.4.1.2 System Audit Data and Alert Counts

The lower part of the System Status Window shows counts of audit record and alert activity for the real-time NIDES analysis. The audit record counts are provided by arpool and are tabulated as the summation of all the audit data provided by all target hosts since the analysis components were started. If the analysis components are turned OFF or go DOWN, the count for audit records starts again at zero when the NIDES analysis and arpool processes are restarted. The past hour count represents the number of audit records received by arpool in the most recent hour.

The alert information shown in this window is provided by the NIDES resolver. The since start-up alert count represents the summation of all alerts generated by all target hosts since the NIDES analysis was initiated. The past hour count represents alert activity during the most recent hour.

The NIDES System Status Window is read-only, and this window can be displayed while you are using other NIDES windows. The Done option at the bottom of the window removes the window. *HELP* provides information on the window's contents.

6.4.2 Targets Option



HOST	AUDIT	STATE	AUDIT RECORDS Total	AUDIT RECORDS Past Hour	ALERTS Total	ALERTS Past Hour
alpha.beta.com	ON	down	0	0	0	0
callie.zen.com	off	down	0	0	0	0
davros.skaro.com	off	down	0	0	0	0
ensor.orac.com	off	down	0	0	0	0
gandalf.middle.com	ON	UP	5098	5098	5	5
vila.zen.com	off	down	0	0	0	0

Figure 6.14: Target Host Status Window

Figure 6.14 shows the window that is displayed when the Targets option is selected from the Monitor Menu. This window contains information about the target hosts that provide real-time audit data to the NIDES analysis components.

All target hosts that are currently in the NIDES target host list are shown on the screen. Each target has six columns of information:

- **AUDIT** — Audit configuration setting for the target host. If the audit configuration is OFF the target host is not currently configured to send audit data to NIDES. If the audit configuration is ON, the target host is running the NIDES process agen, which reads the native format audit data on the target hosts, converts it to NIDES format, and sends it to arpool.

- **STATE** — Status of audit data received from the target host. If the value for the STATE column is DOWN, arpool has not yet received any audit data from the target host. If the value is UP, arpool has received audit data from the target host. In some cases you may have a target that you have switched ON, but which is listed as DOWN. This can be acceptable, especially when you first turn the target host ON, if the target has not generated any audit data since its agen process was initiated by NIDES.

Once a target host's state switches to UP, you should begin to see the numbers change in the counting columns.

- **AUDIT RECORDS Total** — Total number of audit records received by arpool from the target host since the target host was last turned ON. If the target host is switched OFF or goes DOWN, this number is reset to zero.
- **AUDIT RECORDS Past Hour** — Number of audit records received by arpool from the target host during the most recent hour. If the target host is switched OFF or goes DOWN, this number continues to be updated, and if the target remains OFF, eventually goes to zero.
- **ALERTS Total** — Total number of alerts generated by a target host since the target host was last turned ON. If the target host is switched OFF or goes DOWN, this number is reset to zero.
- **ALERTS Past Hour** — Number of alerts generated by the target host for the past hour of elapsed time. If the target host is switched OFF or goes DOWN, this number continues to be updated, and if the target remains OFF, eventually goes to zero.

The Target Host Status Window is read-only, and this window can be displayed while you are using other NIDES windows. The *Done* option at the bottom of the window removes the window. *HELP* provides information on the window's contents.

6.5 Browse Menu

The Browse Menu contains options that support the retrieval of audit and result data and the review of instance configurations, including any pending reconfigurations. Figure 6.15 shows the NIDES Browse Menu, which has four options:

- **Audit Data** — Retrieves, displays, and saves audit data from a NIDES audit data archive
- **Live Results** — Retrieves, displays, and saves result data generated by the real-time NIDES analysis components
- **Test Results** — Retrieves, displays, and saves result data generated by audit data analysis experiments run by the NIDES batch analysis component
- **Instances** — Displays and saves instance configuration and pending analysis reconfiguration data for both real-time and experimental analysis

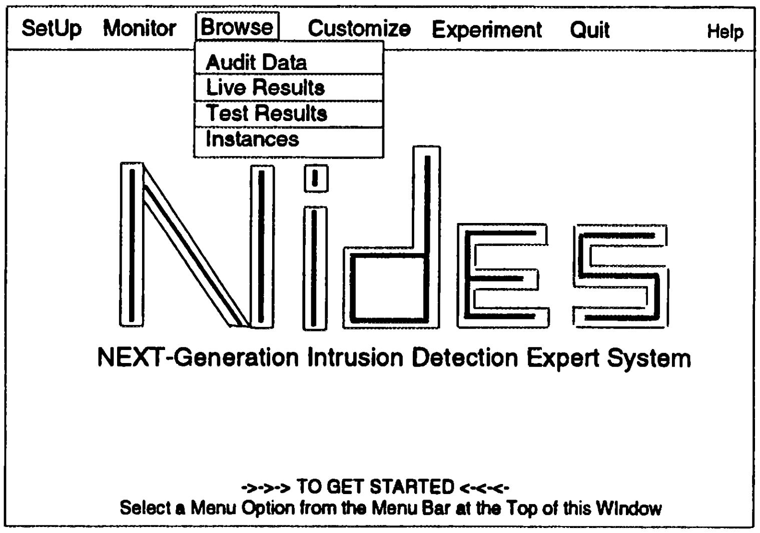

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
Audit Data
Live Results
Test Results
Instances
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.15: NIDES Main Window Browse Menu

6.5.1 Audit Data Option

When you select the Browse Menu Audit Data option, the Audit Data Browse Window as shown in Figure 6.16 is displayed. The window comprises three primary areas, from top to bottom:

- **View Criteria Selection Area** — Selects the audit data archive to retrieve data from, and the search parameters to use when retrieving records from the archive's files.
- **Data View Area** — Displays retrieved data. This area normally includes vertical and horizontal scroll bars.
- **Button Panels** — The buttons in the upper row correspond to a set of *view options* that let you choose which fields in the audit data records are retrieved and displayed. Selecting any of these buttons initiates the data retrieval process. The buttons in the lower row control window actions and the SaveToFile feature.

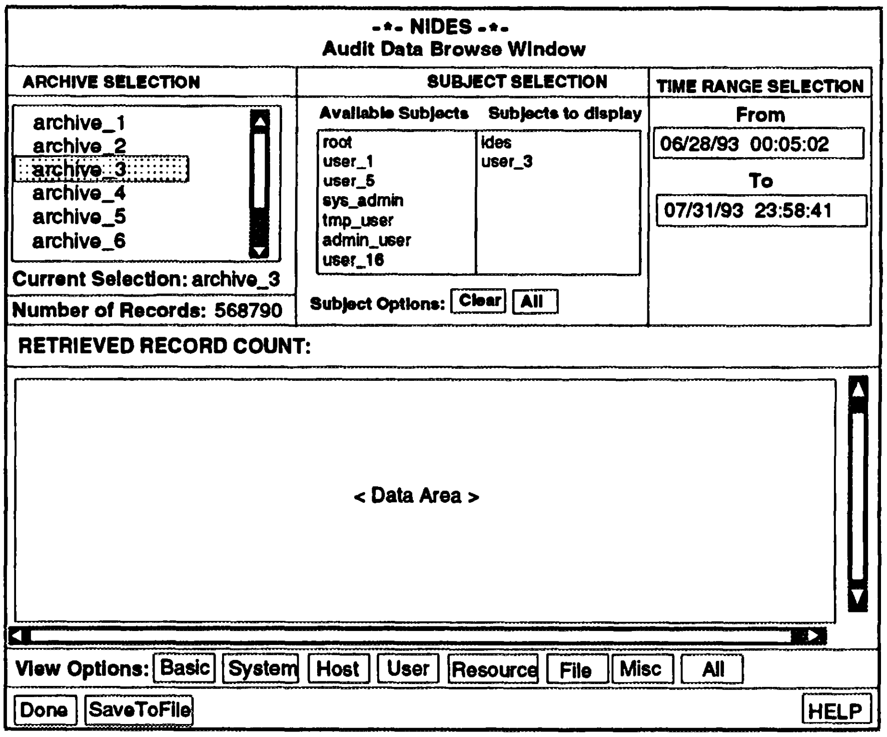
6.5.1.1 View Criteria Selection

At the top of the Audit Data Browse window is the view criteria selection area. Before any audit data can be retrieved, you must specify the archive and search parameters for the retrieval by configuring three items:

é Archive Selection

é Subject Selection

é Time Range Selection



archive_1
archive_2
archive_3
archive_4
archive_5
archive_6

Current Selection: archive_3
Number of Records: 568790

Available Subjects: root, user_1, user_5, sys_admin, tmp_user, admin_user, user_16
Subjects to display: ides, user_3

From: 06/28/93 00:05:02
To: 07/31/93 23:58:41

RETRIEVED RECORD COUNT:

Figure 6.16: Audit Data Browse Window

6.5.1.1.1 Selecting an Archive When the Audit Data Browse Window is first displayed, the Archive Selection area lists all available archives. The Subject and Time Range Selection areas will be empty until an archive has been selected. To retrieve audit data, select the archive from which you want to retrieve data. The selected archive is highlighted and the remainder of the criteria selection area is filled in with data about the currently selected archive. Below the archive list, the name of the currently selected archive and the total number of audit data records in the archive are displayed. The Subject Selection area is filled in with the list of all subjects represented in the audit data archive chosen. The Time Range Selection area is filled in with the earliest and latest times represented in the archive.

6.5.1.1.2 Selecting Subjects Once you have selected an audit data archive, you will need to select the subject or subjects for which you want to retrieve data. When an archive is selected, the Subject Selection area is updated to list all subjects who are represented in the archive. Initially all the subjects are listed in the Available Subjects column of the Subject Selection area. Next to the Available Subjects list is the Subjects to Display list — the subjects whose data will be retrieved. Below the two lists are two convenience buttons labeled Clear and All. Select *All* to move all subjects in the Available Subjects list to the Subjects to Display list. Select Clear to reverse the procedure. Click on any single subject to move it from one list to the other.

6.5.1.1.3 Selecting a Time Range Once you have selected an audit data archive, the Time Range Selection *From* and *To* times are filled in with the archive's earliest and latest timestamps.

To retrieve audit records for the entire time range represented in your chosen archive, you do not need to do anything. However, if the archive contains a large number of records you should reduce the time range.

You may change either the *From* or *To* time value or both. The only constraints on the data you enter are

1. The *From* time cannot be before the earliest time represented in the archive. The earliest time is shown in the *From* field when the archive is initially selected.
2. The *To* time cannot be after the latest time represented in the archive. The latest time is shown in the *To* field when the archive is initially selected.
3. The *From* time must be earlier than the *To* time entered.

To change time values, click on the time. The box surrounding the time entry is highlighted and you can edit its text. Be sure to follow the proper format for your time-range entry **`(MO/DAY/YR HH:MM:SS)`**, for example **`(12/16/90 15:13:45)`**. If you enter an invalid time-range value, or do not follow the proper format, a message will point out your error.

6.5.1.2 View Options

Once you have entered your selection criteria, you are ready to initiate the retrieval process by selecting a view option. Below the audit data viewing area is a row of eight buttons that determine which fields in the audit data records are presented:

- **Basic** — Subject, timestamp, record sequence number, and action
- **System** — System information, such as process ID, system calls, and command names
- **Host** — Target host and remote host information
- **User** — User ID information; if the user ID changes, known user names are present in these fields
- **Resource** — System resource information, such as system time, I/O and reads and writes performed
- **File** — Information on files accessed
- **Misc** — Source of the audit data (i.e., UNIX accounting files, Sun C2, Sun BSM, or perhaps an application)
- **All** — All data from the other seven categories

Table 6.2 shows which fields in the audit record are presented with each option. Fields in the *Basic* option are included in all other options, and the *All* option presents all data.

Tables 6.3, 6.4, and 6.5 describe the audit data fields listed in Table 6.2. These descriptions are valid for UNIX accounting, Sun C2 and Sun BSM audit data that is mapped into the NIDES audit record. Other mappings, such as specialized applications, may apply different meanings to some of the data fields.

Tables 6.6 and 6.7 provide descriptions of the audit data actions listed in Table 6.3.

	Option							
Fields Displayed	**Basic**	**System**	**Host**	**User**	**Resource**	**File**	**Misc**	**All**
Subject	X	X	X	X	X	X	X	X
Timestamp	X	X	X	X	X	X	X	X
Sequence number	X	X	X	X	X	X	X	X
Action	X	X	X	X	X	X	X	X
Command		X						X
System call		X						X
Error number		X						X
Process ID		X						X
Return value		X						X
Target host			X					X
Target sequence number			X					X
Tty			X					X
Arpool timestamp			X					X
Remote host			X					X
Audit user name				X				X
Audit user label				X				X
User name				X				X
User label				X				X
Other user name				X				X
Other user label				X				X
User time					X			X
System time					X			X
Real time					X			X
Memory used					X			X
I/O					X			X
Read/write					X			X
File0						X		X
File0 type						X		X
File0 label						X		X
File1						X		X
File1 type						X		X
File1 label						X		X
Audit data source							X	X
Argument list							X	X

Table 6.2: Audit Data Browse Options

Audit Data Field	**Description** *Valid for UNIX accounting, C2, and BSM data mappings; some application mappings may apply different meanings to some of the audit data fields.*
Subject	The real identifier of the user who generated the audit record. This value should never change, even if users change their effective user identifiers (e.g., with su). This data comes from the "Audit user name" field listed below.
Timestamp	The time at which the audit record was generated on the target host. This value is normally provided by the native auditing system (e.g., C2 or BSM).
Sequence number	The sequence number assigned by arpool. Arpool assigns unique sequence numbers to all incoming audit records. These numbers are unique only within a given run of arpool.
Action	The canonical action type associated with the audit record. The actual values encountered depend upon the types and configurations of the native auditing systems active on any given target host. Descriptions of the action types are listed in Table 6.6. The possible action types are VOID, DISCON, ACCESS, OPEN, WRITE, READ, DELETE, CREATE, RMDIR, CHMOD, EXEC, CHOWN, LINK, CHDIR, RENAME, MKDIR, MOUNT, UNMOUNT, LOGIN, BAD_LOGIN, SU, BAD_SU, EXIT, LOGOUT, UNCAT, RSH, BAD_RSH, PASSWD, RMOUNT, BAD_RMOUNT, PASSWD_AUTH, BAD_PASSWD_AUTH KILL, CORE, PTRACE, TRUNCATE, UTIMES, FORK, CHROOT, MKNOD HALT, REBOOT, SHUTDOWN, BOOT, SET_TIME, SET_UID, SET_GID AUDIT_CONFIG, IS_PROMISCUOUS, CONNECT, ACCEPT, BIND, SOCKET_OPTION
Command	The name (not path) of the command executed.
System call	The name of the UNIX system call that generated the audit record.
Error number	The error value generated by the UNIX system call that generated the audit record. An error value of zero indicates the absence of an error.
Process ID	The UNIX process identifier of the process that generated the audit record.
Return value	The return value generated by the UNIX system call that generated the audit record. For most UNIX system calls, a zero or positive integer value indicates the absence of an error.

Table 6.3: Audit Data Field Descriptions (part 1)

Audit Data Field	**Description**
Target host	The name of the target host on which the audit record was generated. This value is extracted from the audit records of the native auditing system (e.g., C2 or BSM).
Target Sequence number	The sequence number assigned by agen. Agen assigns unique sequence numbers to all generated audit records. These numbers are unique only within a given run of agen.
	Valid for UNIX accounting, C2, and BSM data mappings; some application mappings may apply different meanings to some of the audit data fields.
Tty	The name of the controlling tty of the UNIX process that generated the audit record. This value is not always known.
Arpool timestamp	The time at which the audit record was received by arpool.
Remote host	The name of the remote host involved with a distributed or network related operation (e.g., initiating host for remote login session).
Audit user name	The real identifier of the user who generated the audit record. Even if the effective user identifiers change (e.g., via su) this value should never change.
Audit user label	The security label of the real user identifier. This value will be assigned only by native audit systems that support object labeling.
User name	The effective identifier of the user who generated the audit record. This value will change when the user changes their effective user identifier (e.g., with su).
User label	The security label of the effective user identifier. This value will be assigned only by native audit systems that support object labeling.
Other user name	An additional, action-specific user identifier (e.g., user name of a failed remote login).
Other user label	The security label of the action-specific user identifier. This value will be assigned only by native audit systems that support object labeling.
User time	The total CPU time spent by a UNIX process executing non-kernel program code (e.g., everything but system calls). Reported only upon the termination of a UNIX process under UNIX accounting.
System time	The total CPU time spent by a UNIX process executing kernel program code (e.g., system calls). Reported only upon the termination of a UNIX process under UNIX accounting.
Real time	The total elapsed lifetime of a UNIX process. Reported only upon the termination of a UNIX process under UNIX accounting.

Table 6.4: Audit Data Field Descriptions (part 2)

Audit Data Field	**Description** *Valid for UNIX accounting, C2, and BSM data mappings; some application mappings may apply different meanings to some of the audit data fields.*
Memory used	The average memory usage of a UNIX process. Reported only upon the termination of a UNIX process under UNIX accounting.
I/O	The number of characters transferred (both input and output) during the lifetime of a UNIX process. Reported only upon the termination of a UNIX process under UNIX accounting.
Read/write	The number of blocks transferred (both input and output) during the lifetime of a UNIX process. Reported only upon the termination of a UNIX process under UNIX accounting.
File0	The absolute pathname of the first file argument on the target system relevant to the audit record action. This field is filled in for all single argument file operations (e.g., read, write, create, delete) and all double argument file operations (e.g., link, rename).
File0 type	The type of the first file argument on the target system. This field will be either "regular" or "temporary."
File0 label	The security label of first file argument on the target system. This value will be assigned only by native audit systems that support object labeling.
File1	The absolute pathname of the second file argument on the target system relevant to the audit record action. This field is blank for all single argument file operations (e.g., read, write, create, delete) and filled in only for double argument file operations (e.g., link, rename).
File1 type	The type of the second file argument on the target system. This field will be either "regular" or "temporary."
File1 label	The security label of the second file argument on the target system. This value will be assigned only by native audit systems that support object labeling.
Audit data source	The type of the native audit system that generated the raw audit data that was translated into a NIDES audit record. This field may be C2, BSM1, BSM2 (all SunOS native auditing systems), PACCT (standard UNIX accounting), ADABAS, or LINK (arpool-generated disconnect record). New sources may be added to NIDES at a later date.
Argument list	The command line arguments to a command. This field is rarely filled in.

Table 6.5: Audit Data Field Descriptions (part 3)

Action	Description *Valid for UNIX accounting, C2, and BSM audit data sources; NIDES may apply different meanings to other data sources.*
VOID	When no other recognized action occurs, the audit record action field is set to VOID.
DISCON	A NIDES DISCON record is generated when a target host disconnects from the NIDES arpool process. This action is generated whether the disconnection is intentional (i.e., turning OFF a target host) or not.
ACCESS	A file or directory's status information has been accessed. For example when the `ls -l` command is executed.
OPEN	A file/directory was opened. For example when a `cat` or `more` command is executed.
WRITE	A file/directory was written/modified. For example when a editor is used to modify a file.
READ	A file/directory was read. For example when a `cat` or `more` command is executed.
DELETE	A file is deleted. For example using `rm`.
CREATE	A file is created. For example using an editor.
RMDIR	A directory is removed using `rmdir`.
CHMOD	A file/directory mode is changed using `chmod`.
EXEC	A command is executed.
CHOWN	A file/directory owner is changed using `chown`.
LINK	Indicates the creation of either a symbolic link (i.e., `ln -s`) or a hard link (i.e., `ln`).
CHDIR	The `chdir` command is executed to change directories.
RENAME	A file/directory is renamed. For example, using `mv`
MKDIR	A new directory is created using `mkdir`.
MOUNT	A `mount` command is executed.
UNMOUNT	A `umount` command is executed.
LOGIN	A login occurred. For example, using `login` or `rlogin`
BAD_LOGIN	An unsuccessful login attempt occurred.
SU	A su occurred. For example, using `su` or `login`.
BAD_SU	An unsuccessful `su` attempt occurred.
EXIT	A command exited.
LOGOUT	A logout occurred.
UNCAT	Not used in the current NIDES release.
RSH	An `rsh` command was executed.
BAD_RSH	An unsuccessful `rsh` attempt occurred.
PASSWD	A password authentication occurred.
RMOUNT	A remote mount using `mount` occurred.
BAD_RMOUNT	An unsuccessful remote mount attempt was made.
PASSWD_AUTH	A password authentication occurred.
BAD_PASSWD_AUTH	An unsuccessful password authentication occurred.

Table 6.6: Audit Record Action Types (part 1)

Action	Description *Valid for UNIX accounting, C2, and BSM audit data sources; NIDES may apply different meanings to other data sources.*
KILL	A process is killed via the "kill" command.
CORE hline PTRACE	When a process dumps core a record is generated of this type.
TRUNCATE	
UTIMES	
FORK	When a processes is forked/ or forks other processes this type of record is generated
CHROOT	
MKNOD	
HALT	When the system is halted using the "halt" command this action type is assigned to the audit record that indicates execution of the "halt" command.
REBOOT	When a system is rebooted using the "reboot" command this action type is assigned to the audit record that indicates execution of the "reboot" command.
SHUTDOWN	When a system is halted using the "shutdown" command this action type is assigned to the audit record that indicates execution of the "shutdown" command.
BOOT	When a system is booted using the "boot" command this action type is assigned to the audit record that indicates execution of the "boot" command.
SET_TIME	An audit record indicating a change in the system's clock using the "date" (is that right?) command.
SET_UID	An audit record indicating that the user id was changed to something other than root via the su command
SET_GID	An audit record indicating that the group id was changed via the ??? command
AUDIT_CONFIG	When the configuration of the C2/BSM audit_config is changed via the audit -s command this action type is assigned to the audit record indicating a change in the audit configuration.
IS_PROMISCUOUS	When agen detects that the ethernet controller of a particular host is in promisucous mode an audit record of this action type is generated by agen.
CONNECT	When another host connects to the target host using the ??? command this action type is assigned to the audit record for the command ??.
ACCEPT	When a target host accepts a connection from another host using the ??? command this action type is assigned to the audit record for the command ??.
BIND	
SOCKET_OPTION	

Table 6.7: Audit Record Action Types part 2)

6.5.1.3 Viewing Audit Data

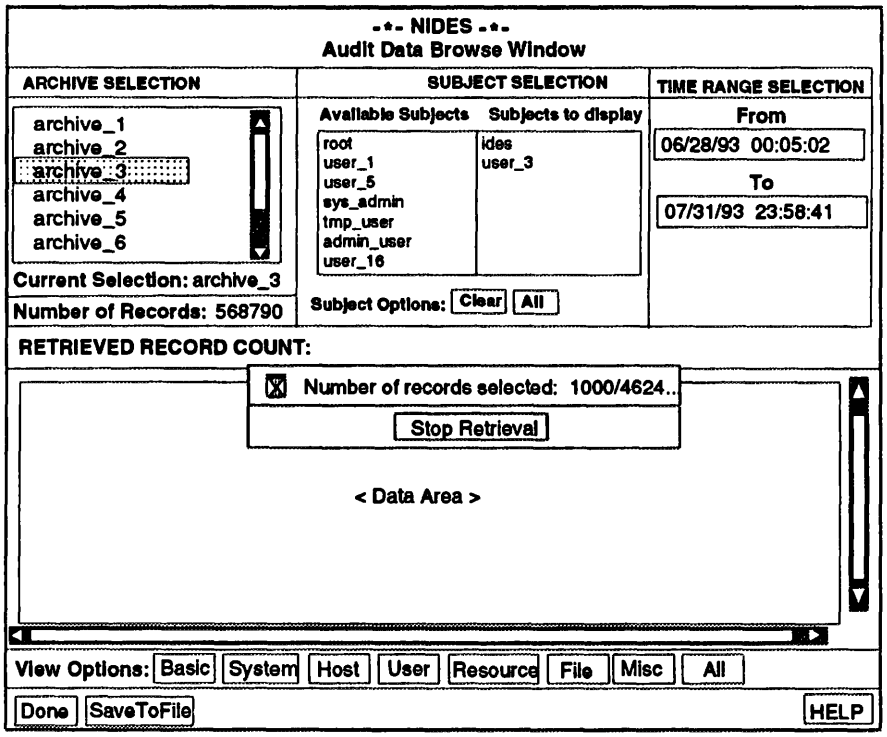


Figure 6.17: Audit Data Browse Counting Window

To view archived audit data, enter your search criteria (subject and time range) and select one of the eight view options to display a confirmation window. After you have confirmed the retrieval, a "selecting records" message is displayed. Once the record selection is completed, a counter window is presented, like the one shown in Figure 6.17. The window shows two numbers; the first, which will be updated, is the number of records that have been retrieved from the archive so far. The second is the total number of records that matched the search criteria and therefore will be retrieved. Below the numbers is a button labeled Stop Retrieval which terminates the process and displays those records retrieved so far.

Note that this release of NIDES limits each retrieval to 5,000 records. When the retrieval is initiated, an estimate of the number of records likely to match the search criteria is made. If this estimate exceeds the 5,000-record limit, an error message is displayed and the retrieval is canceled. If you receive this warning narrow your search criteria by reducing the number of subjects, shortening the time range, or both.

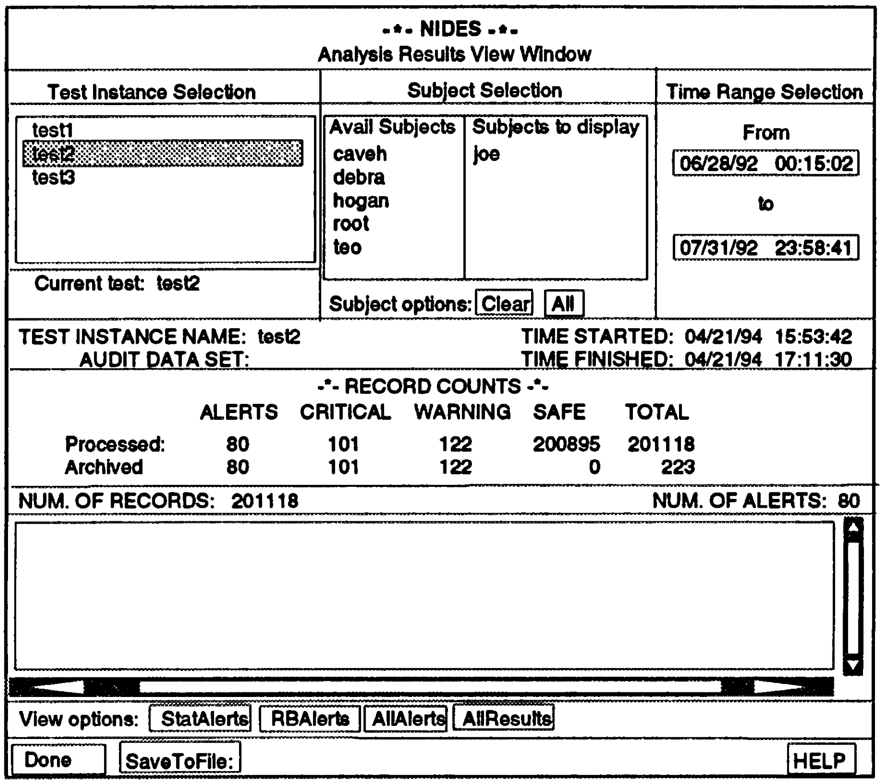
6.5.1.4 Window/File Options

Below the row of View Option buttons are three buttons:

- **Done** — Exits the Audit Data Browse Window and returns you to the Main Window

- **SaveToFile** — Saves the currently displayed audit data to an ASCII text file; when you select this button, you are prompted for the name of the file
- **HELP** — Presents information on the Audit Data Browse Window

6.5.2 Live Results and Test Results Options


-*- NIDES -*-
Analysis Results View Window

Test Instance Selection: test1, test2, test3
Current test: test2

Subject Selection
Avail Subjects: caveh, debra, hogan, root, teo
Subjects to display: joe
Subject options: Clear All

Time Range Selection
From 06/28/92 00:15:02 to 07/31/92 23:58:41

TEST INSTANCE NAME: test2 TIME STARTED: 04/21/94 15:53:42
AUDIT DATA SET: TIME FINISHED: 04/21/94 17:11:30

-*- RECORD COUNTS -*-

	ALERTS	CRITICAL	WARNING	SAFE	TOTAL
Processed:	80	101	122	200895	201118
Archived	80	101	122	0	223

NUM. OF RECORDS: 201118 NUM. OF ALERTS: 80

Figure 6.18: NIDES Analysis Results View Window

The Live Results and Test Results options of the Main Window Browse Menu allow you to view the results of NIDES real time and experimental analysis. Because these two functions are nearly identical, the viewing functions for both live and test results are described here, with differences between real time and tests noted. Figure 6.18 shows the Analysis Results View Window for tests. The real-time window is identical except that under the Test Instance Selection area *real-time* is the only option listed and it is preselected when the window is displayed. The Analysis Results View Window comprises five areas. The top section contains several boxes that allow you to select the search parameters for result data retrieval. There are three criteria — test selection (except for the real-time option, where no selection is needed), subject selection, and time range selection.

Below the search criteria selection area is a heading showing information about the test selected. This information includes the test/instance name, the date the test was run, and the audit data used.

Below this general information are several counts for the archive. The first row lists the record counts for all data processed, and below this row is a listing of records that are included in the archive. The archive counts will be less than or equal to the processed numbers. For each row, the number of alerts, and critical-, warning- and safe-level results are shown, followed by totals for processed records and archived records.

Below the test information area is a window where the selected data will be displayed. The box includes both vertical (on the right) and horizontal (on the bottom) scroll bars. At the top of this area are two numbers — the records retrieved and alerts retrieved.

Under the data view area is a panel of view-option buttons labeled StatAlerts, RBAlerts, AllAlerts, and AllResults.

The bottom section of the window contains three buttons labeled Done, SaveToFile, and HELP.

6.5.2.1 Search Criteria Selection

When the window is initially displayed, the test selection list is filled in with available test names. Once a test is selected, the Subject Selection and Time Range Selection areas are initialized with appropriate selections. The Available Subjects box lists all subjects that were part of the test chosen. The Time Range Selection area shows the earliest and latest times represented in the audit data used for the test run.

To view the results of a test, follow these steps:

1. **Select a test.** If you are browsing test results, you will need to select a test from the Test Instance Selection area. Under real-time results browsing, the real-time option is preselected when the window is presented and this step is not needed.

2. **Select subjects.** Select one or more subjects from the Subject Selection area.

3. **Modify the time range.** You do not need to modify the time range if you want to see results for the entire duration of results data.

4. **Select a view option.** Select one of the four view options from the View Options button panel.

6.5.2.1.1 Selecting a Test This step is not needed if you have selected the Live Results option of the Browse Menu (i.e., you are viewing real-time results). When the Analysis Results View Window is first displayed, the Test Instance Selection area lists all available tests. The subject and time-range selection areas will be empty until a test has been selected. To retrieve results data, select the test from which you want to retrieve data. The test selected is highlighted and the remainder of the criteria selection area is filled in with data. Below the test list, the name of the currently selected test is shown. The Subject Selection area is filled in with the list of all subjects represented in the test chosen or, for real-time, any subjects who have results data for the real-time analysis. The Time Range Selection area is filled in with the earliest and latest times represented in the results archive.

Below the search criteria selection area the following information is shown once a test is selected:

- **Test Name.**

- **Time Started and Time Finished** — The dates when the test was run. For real time, these are the earliest and latest times for the real-time analysis.
- **Audit Data Set** — The name of the audit data set used for the test run. The dates listed in the criteria selection area coincide with the dates of the audit data in the audit data set used for the experiment. This field is not relevant for real-time results.
- **Record Counts** — Below the general test information is another boxed area containing two sets of five numbers. One set represents record counts for all data processed. The second represents record counts for data that was archived. The five numbers reported are
  - **Alerts** — Number of records processed that generated an alert.
  - **Critical** — Number of records processed that generated a critical-level result from the statistical analysis component or the rulebased analysis component. A record is considered critical by the statistics component if the score calculated for the record exceeds the red/critical score threshold.
  - **Warning** — Number of records processed that generated a warning-level result from the statistical analysis component or the rulebased analysis component. A record is considered a warning by the statistics component if the score calculated for the record exceeds the yellow/warning score threshold.
  - **Safe** — Number of records processed that generated a safe-level result. In this case the statistical score calculated was below all thresholds and the rulebase did not report any problems with the record processed.
  - **Total** — Totals for both processed and archived records.

6.5.2.1.2 Selecting Subjects After you have selected a test, select the subject or subjects for which you want to retrieve result data. When a test is selected, the Subject Selection area is updated to list all subjects who are represented in the test. Initially, all the subjects are listed in the Available Subjects column of the Subject Selection area. Next to the Available Subjects list is the Subjects to Display list — the subjects whose data will be retrieved. Below the two lists are two convenience buttons labeled Clear and All. Select *All* to move all subjects in the Available Subjects list to the Subjects to Display list. Select Clear to reverse the procedure. Select any single subject to move it from one list to the other.

6.5.2.1.3 Selecting a Time Range Once you have selected a test, the Time Range Selection *From* and *To* times are filled in with the result data's earliest and latest timestamps. To retrieve audit records for the entire time range represented in your chosen test, you do not need to do anything. However, if your test contains a large number of records, you should reduce the time range used for the data selection.

You may change either the *From* or *To* time value or both. The only constraints on the data you enter are

1. The *From* time entered cannot be before the earliest time represented in the test results. The earliest time is shown in the *From* field when the test is initially selected.

2. The *To* time entered cannot be after the latest time represented in the test results. The latest time is shown in the *To* field when the test is initially selected.

3. The *From* time must be earlier than the *To* time entered.

To change time values, select the time. The box surrounding the time entry is highlighted, and you can edit its text. Be sure to follow the proper format for your time-range entry **(MO/DAY/YR HH:MM:SS),** for example **(12/16/90 15:13:45).** If you enter an invalid time-range value, or do not follow the proper format, a message points out your error.

6.5.2.2 View Criteria Selection

Below the data viewing area is a row of four View Option buttons that allow you to further specify which records in the results archive are retrieved. The four options for viewing results data are

1. StatAlerts — Retrieve only results that were statistical alerts

2. RBAlerts — Retrieve only results that were rulebased alerts

3. AllAlerts — Retrieve all results that contained any alert, either rulebased, statistics, or both

4. AllResults — Retrieve all results generated, including both alert and nonalert results

When you have entered your search criteria and selected one of the four view options, a confirmation window is displayed. After you have confirmed the retrieval, a "selecting records" message is displayed, as shown in Figure 6.19. After the record selection has been completed, a counter window is presented, like the one shown in Figure 6.20. The window shows two numbers; the first, which will be updated, is the number of records that have been retrieved from the archive so far. The second is the total number of records that matched the search criteria and therefore will be retrieved. Below the numbers is a button labeled Stop Retrieval, that you can select to terminate the retrieval; those records that were retrieved so far are displayed.

Note that this release of NIDES limits each retrieval to 5,000 records. When the retrieval is initiated, an estimate of the number of records likely to match the search criteria is made. If this estimate exceeds the 5,000-record limit, an error message is displayed and the retrieval is canceled. If you receive this warning, narrow your search criteria by reducing the number of subjects, shortening the time range, or both.

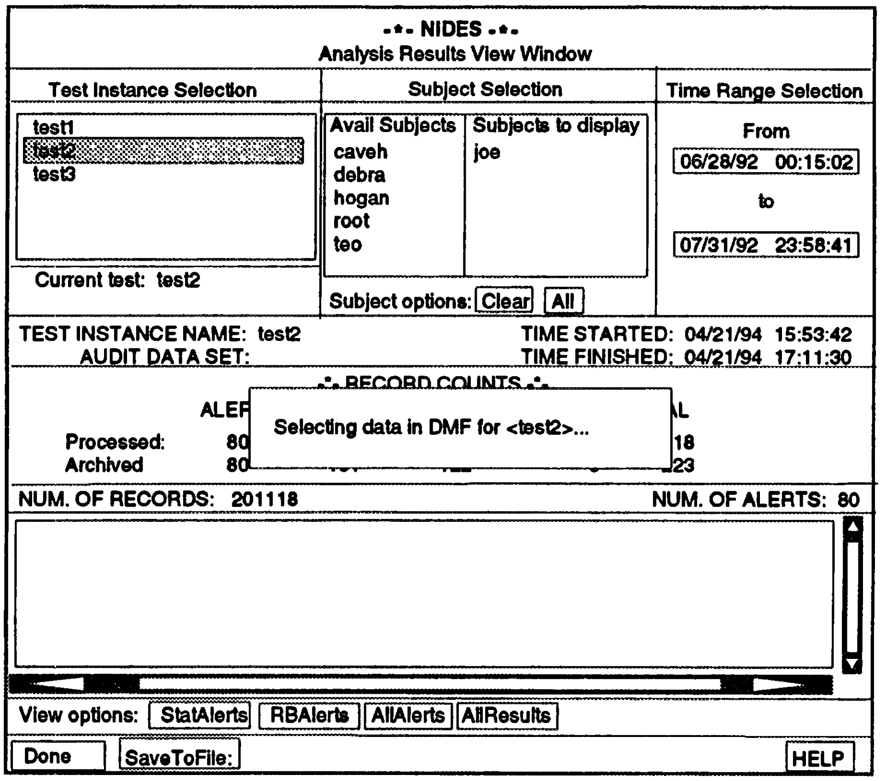
6.5.2.3 Description of Data Displayed

Figure 6.21 shows samples of result records (both rulebased and statistical) with and without alerts. Each record of result data is displayed in one row of eight columns, from left to right:

- **User@host** — User name and target host name.

- **Date** — Arpool timestamp, i.e., date/time audit record was received by arpool.

- **Sequence Number** — Unique arpool sequence number. Sequence numbers are unique only under the same invocation of arpool. If arpool is stopped and then restarted, sequence numbers will begin again at 0.


-•- NIDES -•-
Analysis Results View Window

Test Instance Selection: test1, test2, test3
Current test: test2

Subject Selection
Avail Subjects: caveh, debra, hogan, root, teo
Subjects to display: joe
Subject options: Clear All

Time Range Selection
From 06/28/92 00:15:02 to 07/31/92 23:58:41

TEST INSTANCE NAME: test2 TIME STARTED: 04/21/94 15:53:42
AUDIT DATA SET: TIME FINISHED: 04/21/94 17:11:30

RECORD COUNTS
Processed: 80 ... 18
Archived 80 ... 23

Selecting data in DMF for <test2>...

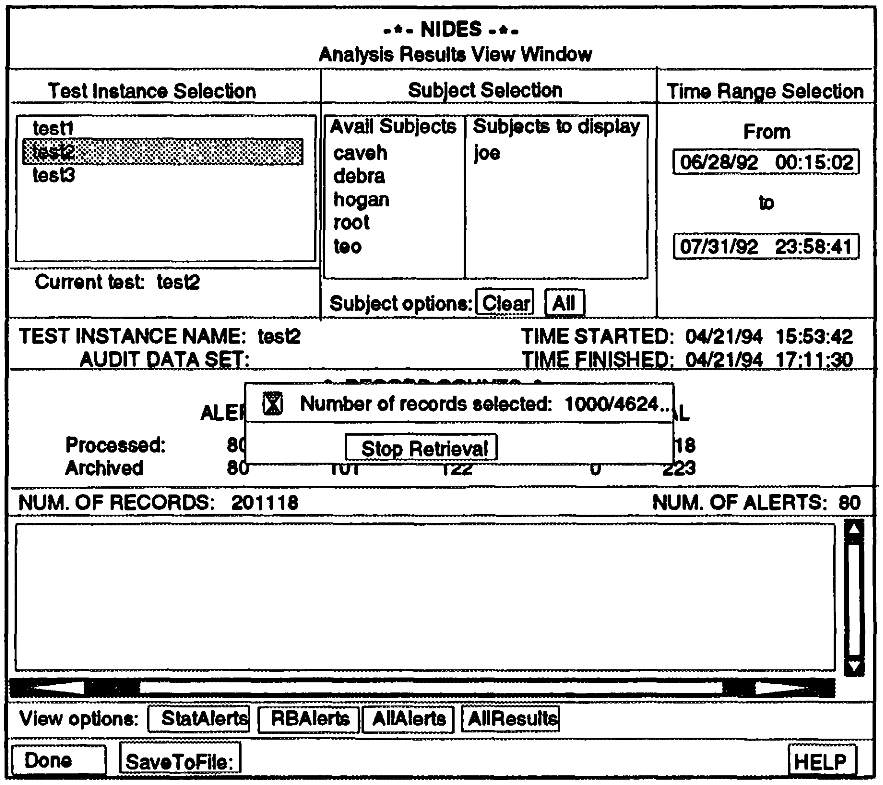
NUM. OF RECORDS: 201118 NUM. OF ALERTS: 80

View options: StatAlerts RBAlerts AllAlerts AllResults
Done SaveToFile: HELP

Figure 6.19: NIDES Analysis Results View Window Data Selection

- **Level** — Safe, warning or critical result; displayed as (S), (W), or (C).
- **Score** — Statistical analysis component score calculated for the audit record.
- **Red Threshold** — Value of the critical/red threshold for statistics when audit record was analyzed.
- **Top 5 Measures** — Measures having the greatest contribution to the statistical score (i.e., those measures considered the most abnormal by NIDES).
- **Top 5 S-values** — Value of the S statistic for the five measures considered most abnormal by NIDES. Section 4.1.4 describes how the S statistic is calculated. The maximum value for S for any measure is 4.0. For each measure observed in an audit record an S value is calculated. These five values allow the security officer to determine if an anomaly was triggered largely by abnormal activity with respect to a few measures (as evidenced by the top one or two scores being near four and the remaining three or four significantly lower) or due to abnormal behavior across several measures (as evidenced by several of the top five scores close to 4, say 3.5 or above).

Below the result data, corresponding alert data is displayed if an alert was generated.



test1
test2
test3

caveh
debra
hogan
root
teo

joe

06/28/92 00:15:02
07/31/92 23:58:41

Current test: test2

TEST INSTANCE NAME: test2
AUDIT DATA SET:
TIME STARTED: 04/21/94 15:53:42
TIME FINISHED: 04/21/94 17:11:30

Number of records selected: 1000/4624...

```
           ALE                                    L
Processed:   8                                    18
Archived     80      101      122        0       223
```

NUM. OF RECORDS: 201118
NUM. OF ALERTS: 80

Figure 6.20: NIDES Analysis Results View Window Progress Reporting

6.5.2.4 Window/File Options

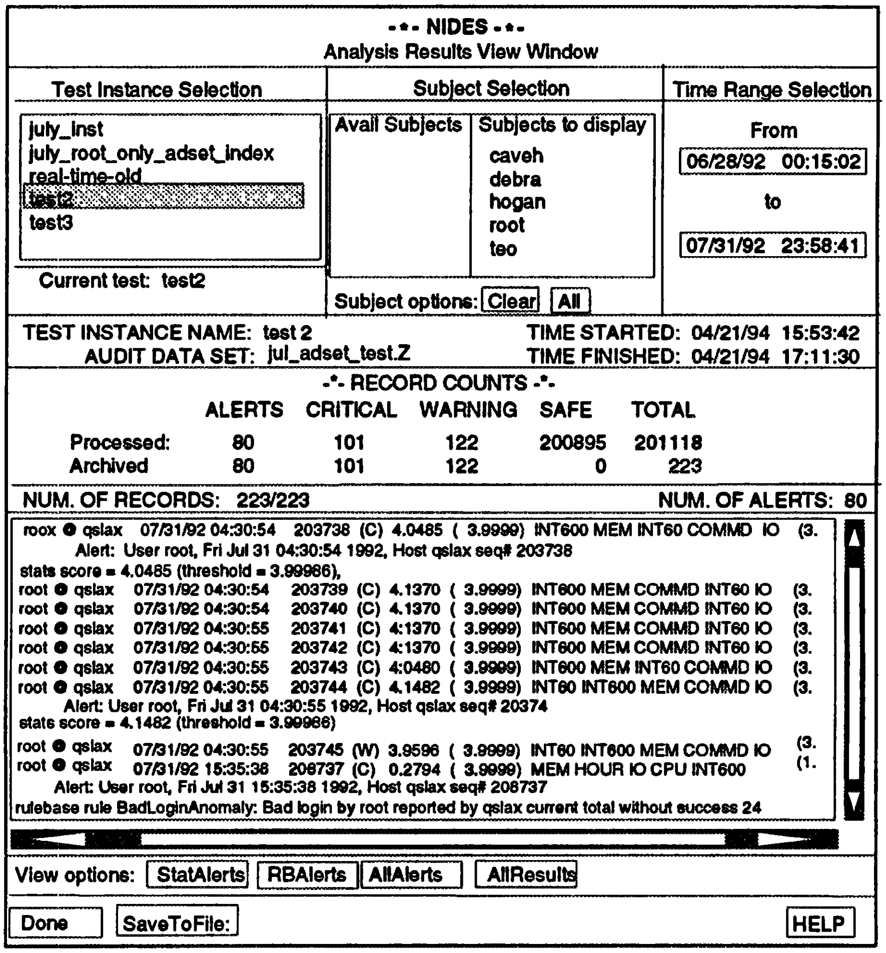
Below the row of View Option buttons are three buttons:

- **Done** — Exits the Analysis Results View Window and returns you to the Main Window
- **SaveToFile** — Saves the currently displayed result data to an ASCII text file; when you select this button, you are prompted for the name of the file
- **HELP** — Presents information about using the Analysis Results View Window

6.5.3 Instance Option

The Instance option of the Main Window Browse Menu allows you to view configuration information for the real-time instance and for test instances. When the Instance option is selected, the Instance View Window as shown in Figure 6.22 is displayed. The window comprises three main areas, from top to bottom:

- **Instance List** — List of available instances; the real-time instance is always at the top
- **View Options** — Box containing nine buttons, which are the instance view options
- **Window/File Options** — *SaveToFile, Done,* and *HELP* option buttons



-•- NIDES -•-
Analysis Results View Window

Test Instance Selection | Subject Selection | Time Range Selection

july_inst
july_root_only_adset_index
real-time-old
test2
test3

Current test: test2

Avail Subjects | Subjects to display
caveh
debra
hogan
root
teo

Subject options: Clear All

From
06/28/92 00:15:02
to
07/31/92 23:58:41

TEST INSTANCE NAME: test 2 TIME STARTED: 04/21/94 15:53:42
AUDIT DATA SET: jul_adset_test.Z TIME FINISHED: 04/21/94 17:11:30

-•- RECORD COUNTS -•-

	ALERTS	CRITICAL	WARNING	SAFE	TOTAL
Processed:	80	101	122	200895	201118
Archived	80	101	122	0	223

NUM. OF RECORDS: 223/223 NUM. OF ALERTS: 80

```
 roox ● qslax  07/31/92 04:30:54  203738 (C) 4.0485 ( 3.9999) INT600 MEM INT60 COMMD IO  (3.
        Alert: User root, Fri Jul 31 04:30:54 1992, Host qslax seq# 203738
stats score = 4.0485 (threshold = 3.99986),
root ● qslax  07/31/92 04:30:54  203739 (C) 4.1370 ( 3.9999) INT600 MEM COMMD INT60 IO  (3.
root ● qslax  07/31/92 04:30:54  203740 (C) 4.1370 ( 3.9999) INT600 MEM COMMD INT60 IO  (3.
root ● qslax  07/31/92 04:30:55  203741 (C) 4:1370 ( 3.9999) INT600 MEM COMMD INT60 IO  (3.
root ● qslax  07/31/92 04:30:55  203742 (C) 4:1370 ( 3.9999) INT600 MEM COMMD INT60 IO  (3.
root ● qslax  07/31/92 04:30:55  203743 (C) 4:0480 ( 3.9999) INT600 MEM INT60 COMMD IO  (3.
root ● qslax  07/31/92 04:30:55  203744 (C) 4.1482 ( 3.9999) INT60 INT600 MEM COMMD IO  (3.
      Alert: User root, Fri Jul 31 04:30:55 1992, Host qslax seq# 20374
stats score = 4.1482 (threshold = 3.99986)
root ● qslax  07/31/92 04:30:55  203745 (W) 3.9596 ( 3.9999) INT60 INT600 MEM COMMD IO  (3.
root ● qslax  07/31/92 15:35:38  208737 (C) 0.2794 ( 3.9999) MEM HOUR IO CPU INT600     (1.
     Alert: User root, Fri Jul 31 15:35:38 1992, Host qslax seq# 208737
rulebase rule BadLoginAnomaly: Bad login by root reported by qslax current total without success 24
```

Figure 6.21: NIDES Test Results View Window with Data

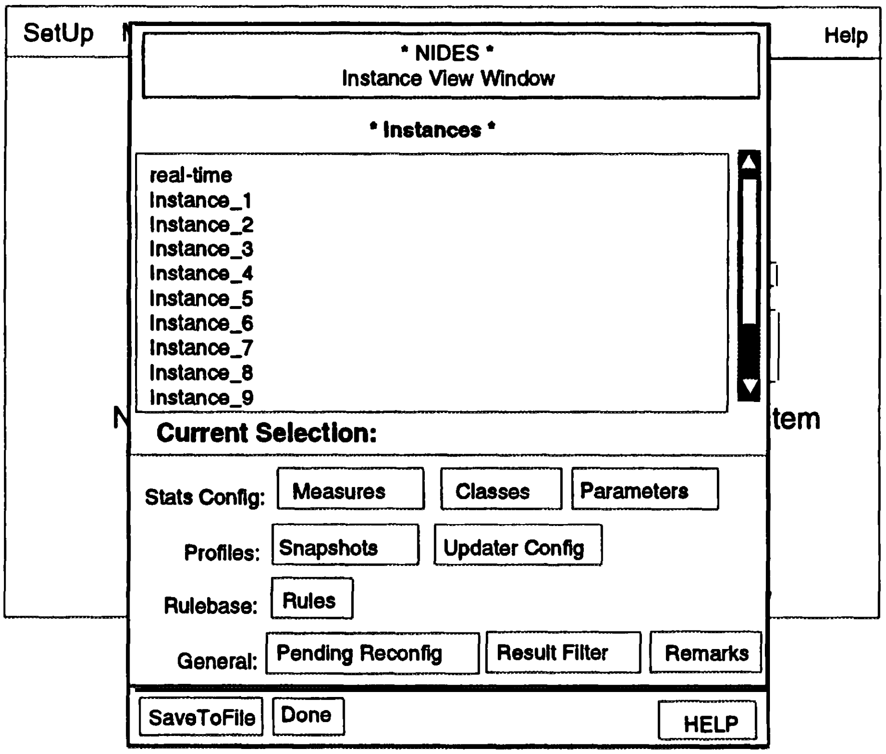


Figure 6.22: NIDES Instance View Window

6.5.3.1 Instance Selection

Prior to viewing any instance configuration data, you must select the instance from the list. The selected instance is highlighted, and the Current Selection area is filled in with the instance name.

6.5.3.2 Instance View Options

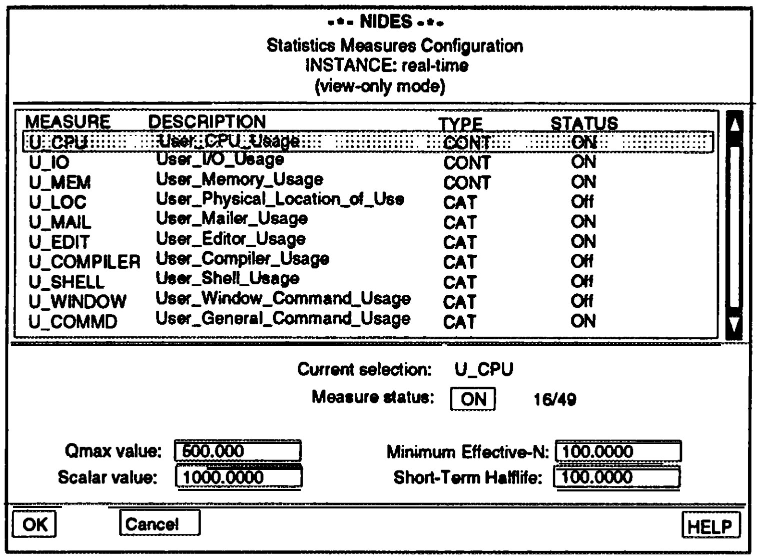
After selecting an instance, you may select one of the nine view options:

- **Measures** — Displays measure states (ON or OFF) and the configurations of each measure
- **Classes** — Displays the available statistics classes and the list of members for each class
- **Parameters** — Displays general statistical analysis component parameters, including long-term half-life, training period, threshold settings and profile cache size
- **Snapshots** — Displays the profiles of subjects, including training status, number of updates, category lists, and other profile data
- **Updater Config** — Displays the configuration of the profile update mode, schedule and status
- **Rules** — Displays the names of all rules available in your current rulebase, and the state of each rule, ON or OFF

- **Pending Reconfig** — Selectable only if there is a pending reconfiguration for the instance selected; displays configuration changes that are pending until the next profile update
- **Result Filter** — Displays the result filter configuration for the selected instance
- **Remarks** — Displays instance remarks, the date the instance was created, and the date of the last audit record processed through the selected instance

All the view options have comparable windows under the Customize Menu options for real-time and test instance configuration. The following paragraphs highlight the view-only versions of the windows. For more information, refer to Section 6.6, which describes the configuration process.

6.5.3.2.1 Instance View — Measure Option When you select the *Measures* option under the Instance View Window, a window as shown in Figure 6.23 is displayed. In the main



-•- NIDES -•-
Statistics Measures Configuration
INSTANCE: real-time
(view-only mode)

MEASURE	DESCRIPTION	TYPE	STATUS
U_CPU	User_CPU_Usage	CONT	ON
U_IO	User_I/O_Usage	CONT	ON
U_MEM	User_Memory_Usage	CONT	ON
U_LOC	User_Physical_Location_of_Use	CAT	Off
U_MAIL	User_Mailer_Usage	CAT	ON
U_EDIT	User_Editor_Usage	CAT	ON
U_COMPILER	User_Compiler_Usage	CAT	Off
U_SHELL	User_Shell_Usage	CAT	Off
U_WINDOW	User_Window_Command_Usage	CAT	Off
U_COMMD	User_General_Command_Usage	CAT	ON

Current selection: U_CPU
Measure status: ON 16/49

Qmax value: 500.000
Scalar value: 1000.0000
Minimum Effective-N: 100.0000
Short-Term Halflife: 100.0000

Figure 6.23: Statistics Measures Configuration Window

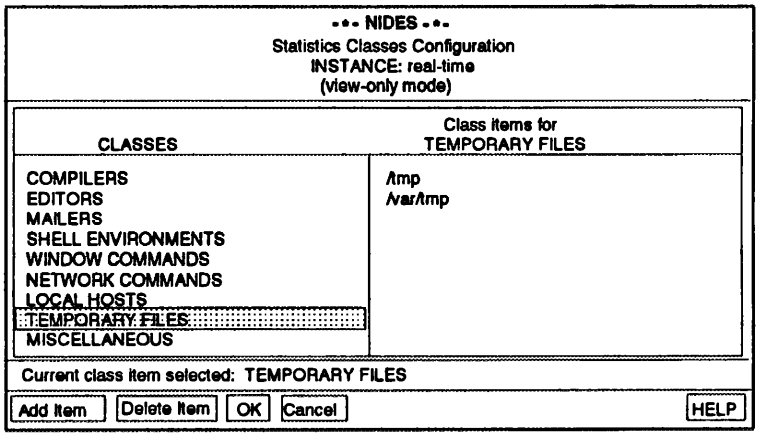
view area of the window is a scrollable list of all statistical measures. The type and status of each measure are listed after the measure ID and description. If the measure is ON and has been trained, it will contribute to statistical anomaly detection; if a measure is OFF it will not contribute, but will be trained so that if it is switched ON it will be able to contribute as soon as possible. Below the list of measures is additional information about the currently selected measure. When you select a measure, it is highlighted and the Current Selection field is updated with the selected measure's ID. The fields below the measure list are filled in with the values for the selected measure:

- **Measure status/count** — A count showing how many measures are activated (turned ON) and the total number of measures available; for example 16/49 means that out of a total of 49 available measures 16 are currently turned ON; see Section 4.6.1 for information on measure activation
- **Qmax value** — Determines binning ranges for the Q distribution; see Section 4.6.3

- **Scalar Value** — Relevant only for continuous measures; used to determine bin ranges, see Section 4.6.2
- **Minimum Effective-N** — Represents the minimum number of observations modified by aging factors that must be observed before the measure will contribute, regardless of the measure's training status, see Section 4.6.4
- **Short-term Half-life** — Represents the number of observations that are made before the contribution of a given audit record is downweighted by one half in the short-term profile, see Section 4.6.5

At the bottom of the measure configuration window are buttons labeled OK, Cancel, and HELP. The *OK* option is disabled in the view-only mode. *Cancel* returns you to the Instance View Window, and the *HELP* option gives information on the Statistics Measures Configuration Window.

6.5.3.2.2 Instance View — Classes Option When you select the *Classes* option under the Instance View Window, a window as shown in Figure 6.24 is displayed.


-•- NIDES -•-
Statistics Classes Configuration
INSTANCE: real-time
(view-only mode)

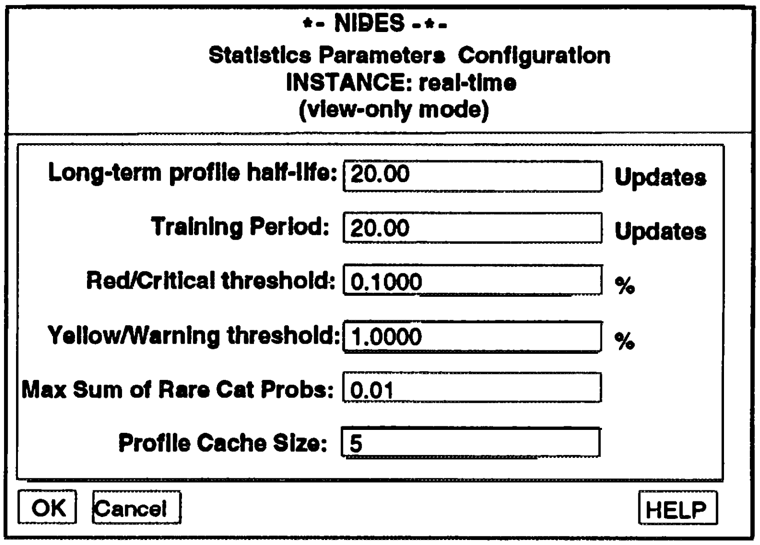
CLASSES	Class items for TEMPORARY FILES
COMPILERS	/tmp
EDITORS	/var/tmp
MAILERS	
SHELL ENVIRONMENTS	
WINDOW COMMANDS	
NETWORK COMMANDS	
LOCAL HOSTS	
TEMPORARY FILES	
MISCELLANEOUS	

Current class item selected: TEMPORARY FILES

Figure 6.24: Statistics Classes Configuration Window

The main view area of the window is divided into two halves. On the left side is a list of available classes. On the right is the area where members of the selected class are displayed. When you select a class, the selected class is highlighted and the right side of the window is populated with the list of class members. A class lists contains the categories grouped under the associated measure that model activity of the type associated with the measure. For example, the compiler class list is used to track compiler usage; its associated category list contains the compilers used on the system. One exception exists – the temporary files class is not tied to a specific measure; it represents the list of files and directories that do not generate categories under the statistics file measures, so in a sense it is a "negative" class. See Section 4.4 for more information on class lists. At the bottom of the window are buttons labeled Add Item, Delete Item, OK, Cancel, and HELP. The *Add Item, Delete Item,* and *OK* options are disabled in the view-only mode. The *Cancel* option returns you to the Instance View Window, and *HELP* gives information on the Statistics Classes Configuration Window.

6.5.3.2.3 Instance View — Parameters Option When you select the Parameters option under the Instance View Window, a window as shown in Figure 6.25 is displayed.


- NIDES --
Statistics Parameters Configuration
INSTANCE: real-time
(view-only mode)

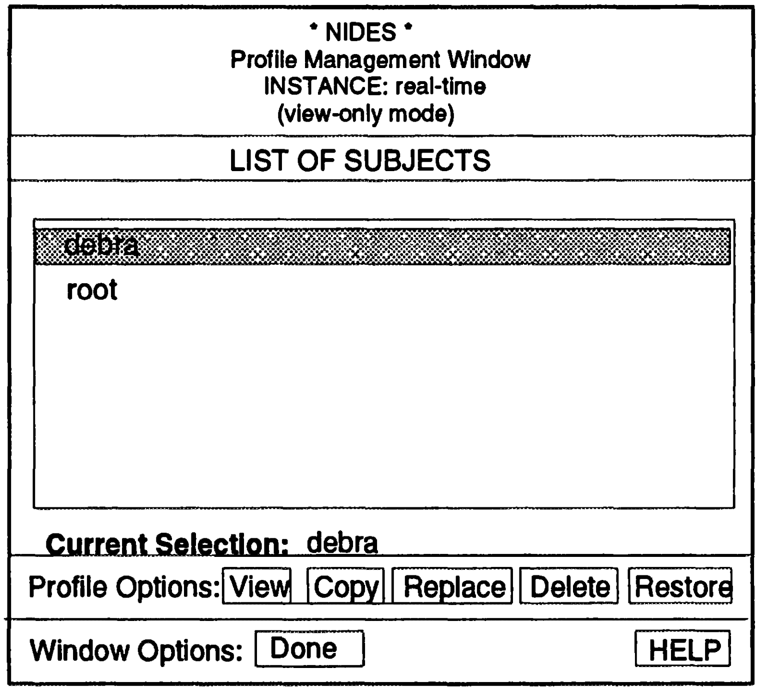
Long-term profile half-life: 20.00 Updates
Training Period: 20.00 Updates
Red/Critical threshold: 0.1000 %
Yellow/Warning threshold: 1.0000 %
Max Sum of Rare Cat Probs: 0.01
Profile Cache Size: 5

Figure 6.25: Statistics Parameters Configuration Window

The main view area of the window lists the six statistics parameters that are configurable across all measures (Table 6.15 lists the default values for these parameters):

- **Long-term Profile Half-life** — Time period (measured in number of profile updates) after which the contribution of a given day's data is downweighted by one half, see Section 4.5.1 for more information on this parameter.
- **Training Period** — Interval of time (measured in number of profile updates) required before the statistical analysis scoring mechanism generates alerts, see Section 4.5.2 for more information on this parameter.
- **Red/Critical Threshold** — Percentage indicating the percentile of activity considered critical; this percentage is used in the calculation of the red threshold score value, see Section 4.5.3 for more information on this parameter.
- **Yellow/Warning Threshold** — Percentage indicating the percentile of activity considered a warning; this percentage is used in the calculation of the yellow threshold score value, see Section 4.5.3 for more information on this parameter.
- **Max Sum of Rare Cat Probs** — Maximum sum of probability that is totaled for any measure's RARE category (i.e., those categories that are seen infrequently); see Section 4.5.4 for more information on this parameter.
- **Profile Cache Size** — Most recently used profiles during NIDES analysis are kept in a cache; the number of profiles maintained in the cache is determined by the profile cache size; see Section 4.5.5 for more information on this parameter.

6.5.3.2.4 Instance View — Snapshots Option When you select the *Snapshots* option under the Instance View Window, a window as shown in Figure 6.26 is displayed.


* NIDES *
Profile Management Window
INSTANCE: real-time
(view-only mode)

LIST OF SUBJECTS

debra
root

Current Selection: debra

Profile Options: View Copy Replace Delete Restore
Window Options: Done HELP

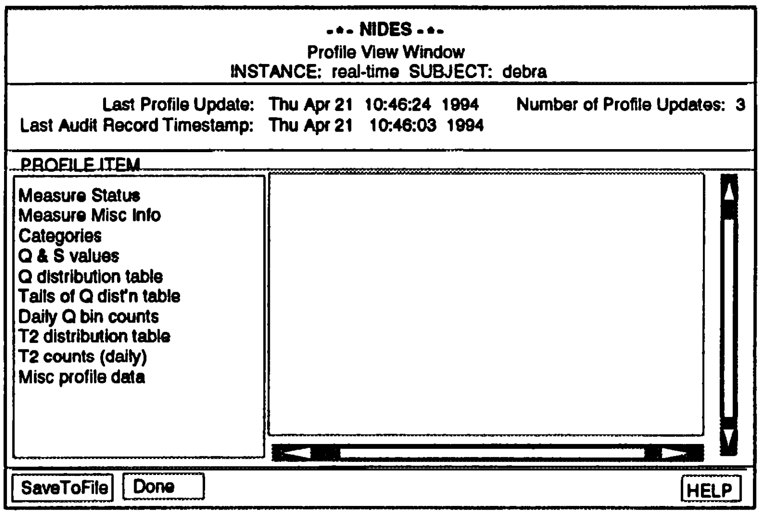
Figure 6.26: Profile Management Window

The main view area of the window lists all subjects that have profiles (both short-term and long-term) for the instance selected. Below the list of subjects are buttons labeled View, Copy, Replace, Delete and Restore. Only the *View* option is selectable in the view-only mode. Profiles that are displayed under this option are current as of the last profile update or are checkpointed versions of the profile if it has been swapped out of the internal profile cache. The profile information displayed comes from both the short-term and long-term profile for the selected subject. At the bottom of the window are buttons labeled Done and HELP. *Done* returns you to the Instance View Window, and *HELP* gives information on the Profile Management Window.

To view the contents of a subject's profile, select the subject, which is then highlighted. The Current Selection area is updated with the selected subject. Select *View* to display a Profile View Window as shown in Figure 6.27.

The top portion of the window shows the subject's name and the following information:

- **Last Profile Update** — Date and time of the last update of the historical (long-term) profile. The profile you are viewing contains information as of this date.
- **Last Audit Record Timestamp** — Date of the last audit record to have an effect on the profile you are viewing as of the last profile update. If additional audit data has been processed by NIDES, but an update has not occurred, the last audit record timestamp will not coincide with the last audit record seen by NIDES for the chosen subject. A special case occurs if the profile has been swapped out of the internal profile cache and therefore


-•- NIDES -•-
Profile View Window
INSTANCE: real-time SUBJECT: debra

Last Profile Update: Thu Apr 21 10:46:24 1994 Number of Profile Updates: 3
Last Audit Record Timestamp: Thu Apr 21 10:46:03 1994

PROFILE ITEM

Measure Status
Measure Misc Info
Categories
Q & S values
Q distribution table
Tails of Q dist'n table
Daily Q bin counts
T2 distribution table
T2 counts (daily)
Misc profile data

Figure 6.27: Profile View Window

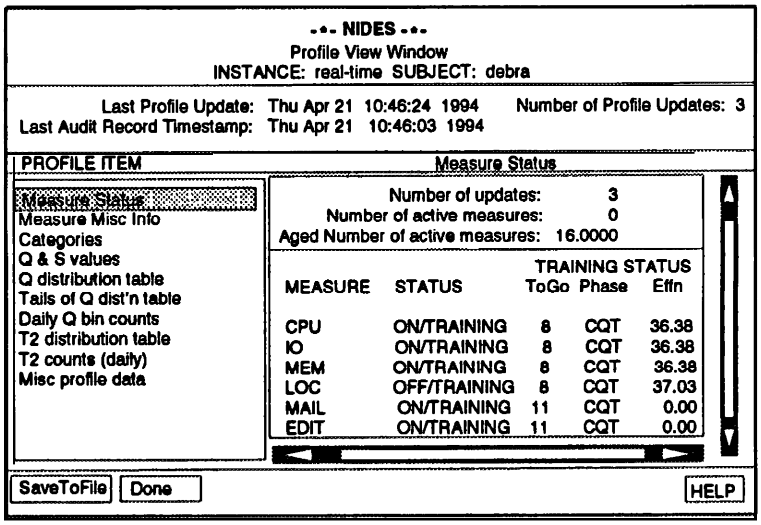
checkpointed. Under this condition the timestamp will match the timestamp of the last audit record processed through the profile even though a profile update has not occurred.

- **Number of Profile Updates** — Number of times the profile has been updated. This number has a direct bearing on the training status of the profile.

Below the top portion of the Profile View Window is an area split into two halves. On the left is a list of profile items that can be viewed. On the right is the area where information on the selected option is presented:

- **Measure Status** — Training status of all measures (see Sections 4.2.3 and 4.7.1.1 and Table 6.8); Figure 6.28 is an example of this data
- **Measure Misc Info** — Summary information about measure categories (see Section 4.7.1.2 and Table 6.8)
- **Categories** — Detailed lists of categories for each measure (see Section 4.7.1.3 and Table 6.9)
- **Q & S values** — Current Q and S values for each measure (see Section 4.7.1.4 and Table 6.9)
- **Q distribution table** - An empirical distribution of the Q values for each Q bin, for each measure (see Section 4.7.1.5 and Table 6.9)
- **Tails of Q dist'n table** — Sum of probability for all bins to the right of the Q distribution (see Section 4.7.1.6 and Table 6.9)
- **Daily Q bin counts** — Daily count for each Q bin for (see Section 4.7.1.7 and Table 6.9) every measure.
- **T2 distribution table** — Empirical distribution of the T2 values (see Section 4.7.1.8 and Table 6.9)

- **T2 counts (daily)** — Daily count for T2 (see Section 4.7.1.9 and Table 6.9)
- **Misc profile data** — current red and yellow score threshold values, and the aged number of total records for which scoring has occurred (see Section 4.7.1.10 and Table 6.9)



```
-*- NIDES -*-
Profile View Window
INSTANCE: real-time SUBJECT: debra

      Last Profile Update: Thu Apr 21 10:46:24 1994    Number of Profile Updates: 3
Last Audit Record Timestamp: Thu Apr 21 10:46:03 1994

PROFILE ITEM                          Measure Status

Measure Status                         Number of updates:       3
Measure Misc Info                Number of active measures:     0
Categories                 Aged Number of active measures: 16.0000
Q & S values
Q distribution table                                  TRAINING STATUS
Tails of Q dist'n table    MEASURE   STATUS          ToGo  Phase   Effn
Daily Q bin counts
T2 distribution table      CPU       ON/TRAINING      8    CQT    36.38
T2 counts (daily)          IO        ON/TRAINING      8    CQT    36.38
Misc profile data          MEM       ON/TRAINING      8    CQT    36.38
                           LOC       OFF/TRAINING     8    CQT    37.03
                           MAIL      ON/TRAINING     11    CQT     0.00
                           EDIT      ON/TRAINING     11    CQT     0.00

SaveToFile  Done                                                  HELP
```

Figure 6.28: Profile Measure Status Data

The component data items listed for each of the ten items are shown in Tables 6.8 and 6.9.

To display a particular option in the view area, select the desired option and use the vertical and horizontal scrollbars, if necessary. Repeat the process to select another item and replace the contents of the view window.

At the bottom of the Profile View Window are buttons labeled SaveTofile, Done, and HELP. The *SaveToFile* option prompts you for a filename in which to write an ASCII version of the currently selected profile view option. The *Done* option returns you to the Profile Management Window, and the *HELP* option gives information on the Profile View Window.

6.5.3.2.5 Instance View - Updater Config Option When you select the *Updater Config* option under the Instance View Window, a window as shown in Figure 6.29 is displayed if you are viewing the real-time instance, and a window as shown in Figure 6.30 is displayed if you are viewing any other instance.

The real-time updater configuration window has three information areas:

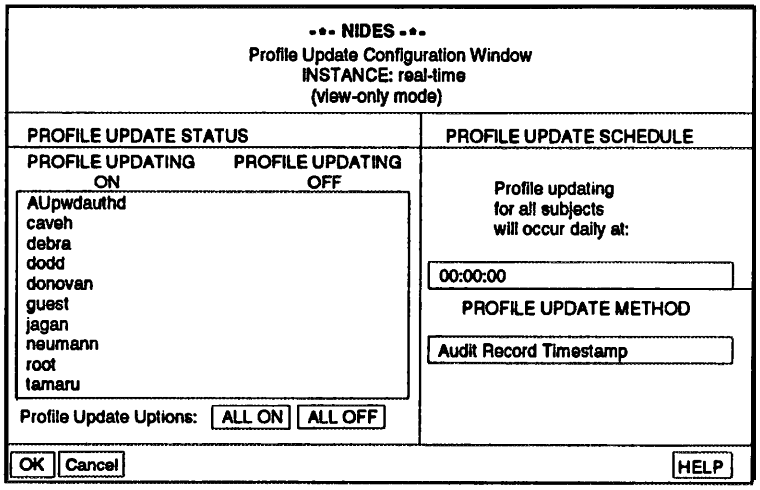
- **Profile Update Status** — The left side of two sections lists subjects whose profiles will be updated when profile updating is scheduled to occur. On the right side is the list of subjects whose profiles will not be updated.
- **Update Schedule** — The time when updates are scheduled to occur, based on a 24-hour clock.
- **Update Method** — The current update method in use. There are two possible methods. The audit record timestamp method updates profiles based on the timestamps in the audit

Profile Option	**Description**
Measure Status	
Number of Updates	Number of times the long-term profile was updated. i.e., the number of update intervals in which there was at least one observation for the measure.
Num. active measures	Number of trained measures turned on by user.
Aged Num. active measures	Aged number of measures used to average score distribution
Training Status	The measure's configuration (ON or OFF) and the training status (TRAINING or READY).
Training Status Fields	
ToGo	The number of updates remaining for the current phase of training.
Phase	The training phases that remain (C = category training, Q = Q training and T = T2 training). For example, if the ToGo number is 7, and the Phase listed is QT, C training has completed, Q training has 7 updates to go, and then T training will begin.
Effn	The historical observed count for the measure in the current training phase.
Short Effn	The effective-n for the short-term profile.
Hist Effn	The historical observed count for the measure aged by the long-term half-life.
Todays Count	Number of times the measure was observed since the last profile update.
Misc. Measure Info	
Aged Number of Cats	Aged number of categories observed for the measure. Aging smooths dropped and new categories.
Sum Rare Catprobs	Actual sum of probabilities of categories in the rare class. Will be less than or equal to max sum rare prob configuration value.
Max prob rare cats	Highest probability of a rare category.
Next Avail cat ID	Used for category hash tables.

Table 6.8: NIDES Profile View Options (part 1)

Profile Option	**Description**
Categories	
CATPROB (Count)	Number of times category seen since last historical profile update.
Type	Measure type (categorical, continuous, binary).
Agecnt	Aged count of times category was observed.
Prevobscnt	Used internally for Q statistic calculation.
Catid	Unique category ID. (unique across all measures)
Catname	Name of category.
Q and S values	
Q value	The normalized chi-square-like difference statistic to measure the long-term/short-term differences for the measure. Value shown is as of most recent update.
S value	The half-normal transformation of Q.
Q Distribution Table	
Bin Numbers	The historical distribution of the Q values used for half-normal inversion, aged by the long-term half-life. Presented in 32 bins. One column for each of the thirty-two Q bins numbered 0 to 31.
Tails of Q Dist'n Table	
Bin Numbers	Tails for the Q distribution table.
Daily Q bin counts	
Bin Numbers	One column for each of the thirty-two daily Q bin counts, bins numbered 0 to 31.
T2 Dist. table	The historical distribution of T2 values, from which score thresholds are estimated. Presented in bins of increments in tenths for T2 values less than 20, then by whole numbers to maximum value.
T2 counts (daily)	Daily counts from which T2 distribution is computed.
Misc. profile Data	
Yellow Threshold Score	That T2 score value which is exceeded at a rate approximately equal to the yellow threshold percentage.
Red Threshold Score	That T2 score value which is exceeded at a rate approximately equal to the red threshold percentage.
Aged num of total records	Aged number of records contributing to the T2 profile (after training).

Table 6.9: NIDES Profile View Options (part 2)


•*• NIDES •*•
Profile Update Configuration Window
INSTANCE: real-time
(view-only mode)

PROFILE UPDATE STATUS

PROFILE UPDATING ON | PROFILE UPDATING OFF

AUpwdauthd
caveh
debra
dodd
donovan
guest
jagan
neumann
root
tamaru

Profile Update Uptions: ALL ON ALL OFF

PROFILE UPDATE SCHEDULE

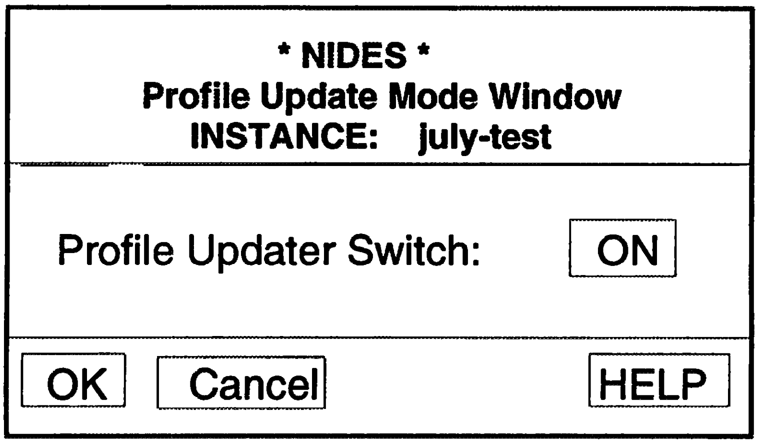
Profile updating
for all subjects
will occur daily at:

00:00:00

PROFILE UPDATE METHOD

Audit Record Timestamp

Figure 6.29: Profile Update Configuration Window (Real-time Instance)


* NIDES *
Profile Update Mode Window
INSTANCE: july-test

Profile Updater Switch: ON

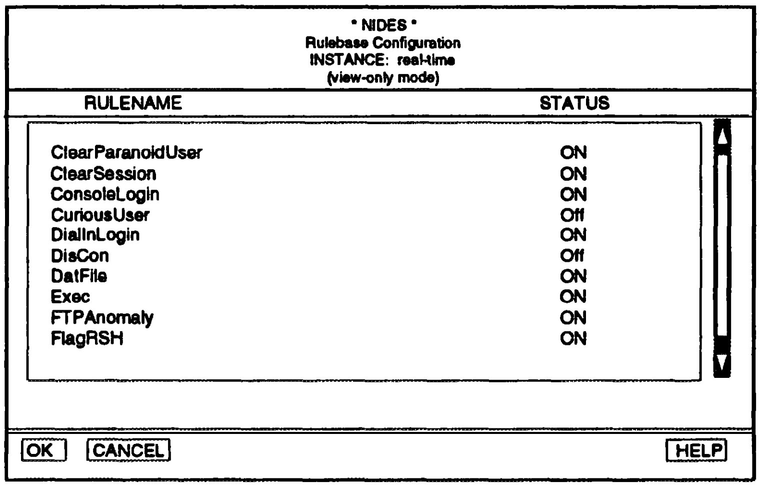
Figure 6.30: Profile Update Mode Window (Test Instances)

records processed. The system clock method updates profiles based on the NIDES computer's system clock regardless of the timestamps of the audit records.

At the bottom of the Update Configuration Window for the real-time instance are buttons labeled OK, Cancel, and HELP. The *OK* option is deactivated in the view-only mode of this window. The *Cancel* option returns you to the Instance View Window, and the *HELP* option gives information on the Update Configuration Window.

The test instance Profile Update Mode Window shows the update mode for the selected instance, either ON or OFF. If the updater switch is set to ON, any experiment run using this instance updates profiles based on the audit record timestamps, with a update schedule set to 00:00:00. If the update switch is set to OFF, no updates will occur during a test run using the instance. At the bottom of this window are buttons labeled OK, Cancel, and HELP. The *OK* option is deactivated in the view-only mode of this window. The *Cancel* option returns you to the Instance View Window, and the *HELP* option gives information on the Profile Update Mode Window.

6.5.3.2.6 Instance View — Rules Option When you select the Rules option under the Instance View Window, a window as shown in Figure 6.31 is displayed.



* NIDES *
Rulebase Configuration
INSTANCE: real-time
(view-only mode)

RULENAME	STATUS
ClearParanoidUser	ON
ClearSession	ON
ConsoleLogin	ON
CuriousUser	Off
DialInLogin	ON
DisCon	Off
DatFile	ON
Exec	ON
FTPAnomaly	ON
FlagRSH	ON

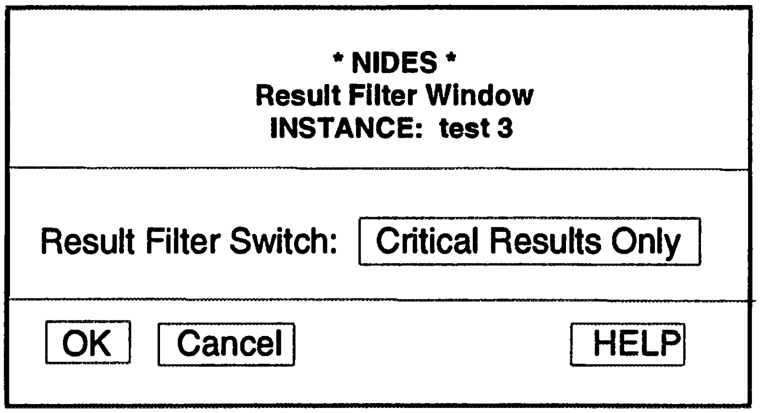
Figure 6.31: Rulebase Configuration Window

This window contains a list of all available rules and their configurations (ON or OFF). If the rule is switched ON it will be used to analyze audit data received by NIDES, if it is switched OFF it will not be used.

Below the list of rules are buttons labeled OK, Cancel, and HELP. The *OK* option is deactivated in the view-only mode of this window. The *Cancel* option returns you to the Instance View Window, and the *HELP* option gives information on the Rulebase Configuration Window.

6.5.3.2.7 Instance View — Result Filter When you select *Result Filter* under the Instance View Window, a window as shown in Figure 6.32 is displayed. This window shows the current configuration for the result data archive filter. There are three possible values for the filter:

- **Critical Results Only** — Archives results at the critical level only


* NIDES *
Result Filter Window
INSTANCE: test 3

Result Filter Switch: Critical Results Only

Figure 6.32: Result Filter Configuration Window

- **Warning Level and Above** — Archives results at the critical or warning level
- **All Results** — Archives all results generated; since for each audit record processed a result record is generated, under this filter setting each audit record seen will generate a result record in the archive

Below the window showing the result filter setting are buttons labeled OK, Cancel, and HELP. The *OK* option is deactivated in the view-only mode of this window. The *Cancel* option returns you to the Instance View Window, and the *HELP* option gives information on the Result Filter Window.

6.5.3.2.8 Instance View — Pending Reconfig Option When you select the *Pending Reconfig* option under the Instance View Window, a window like the one shown in Figure 6.33 is displayed. This window lists all pending reconfigurations for the selected instance. These reconfigurations will be applied at the next profile update.

At the bottom of the window are buttons labeled Done and HELP. *Done* returns you to the Instance View Window, and *HELP* gives information on the Pending Reconfig Window.

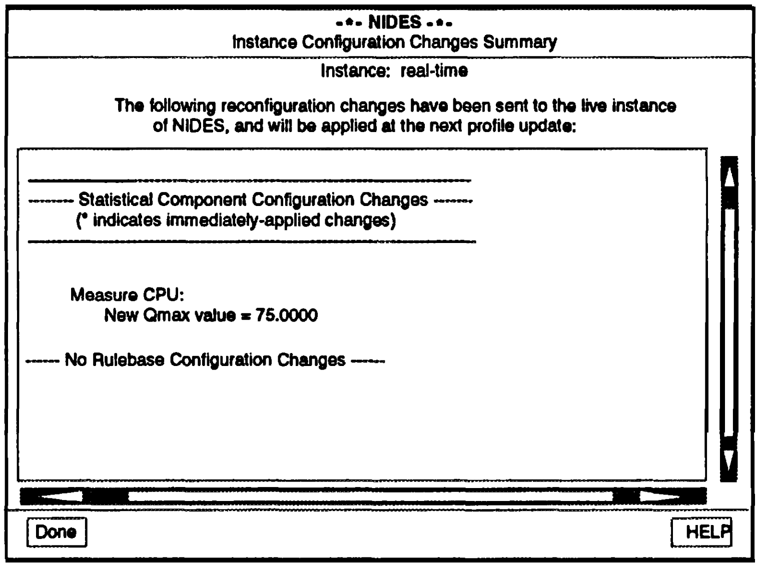
6.5.3.2.9 Instance View — Remarks Option When you select the *Remarks* option under the Instance View Window, *a* window as shown in Figure 6.34 is displayed. At the top of the window are two fields. The first lists the time the instance was initially created, and the second shows the time of the last audit record processed by the instance. Below these two items is a window containing any comments entered for the selected instance.

At the bottom of the window are buttons labeled OK, Cancel, and HELP. The *OK* option is deactivated in the view-only mode of this window. The *Cancel* button returns you to the Instance View Window, and the *HELP* option gives information on the Remarks Window.

6.5.3.3 Window/File Options

Below the row of View Option buttons are three buttons:

- **Done** — Exits the Instance View Window and returns you to the Main Window.



-•- NIDES -•-
Instance Configuration Changes Summary
Instance: real-time

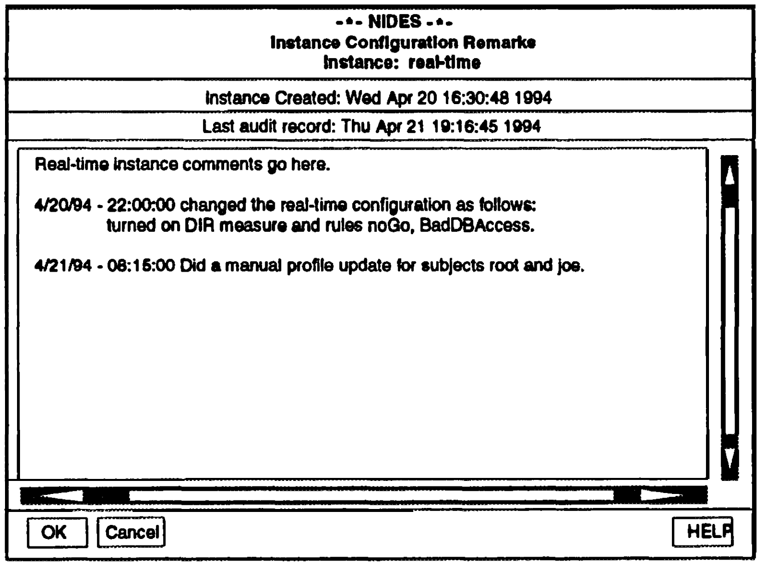
The following reconfiguration changes have been sent to the live instance of NIDES, and will be applied at the next profile update:

```
------ Statistical Component Configuration Changes ------
       (* indicates immediately-applied changes)

    Measure CPU:
        New Qmax value = 75.0000

----- No Rulebase Configuration Changes -----
```

Figure 6.33: Pending Reconfig Window



-•- NIDES -•-
Instance Configuration Remarks
Instance: real-time

Instance Created: Wed Apr 20 16:30:48 1994

Last audit record: Thu Apr 21 19:16:45 1994

```
Real-time instance comments go here.

4/20/94 - 22:00:00 changed the real-time configuration as follows:
          turned on DIR measure and rules noGo, BadDBAccess.

4/21/94 - 08:15:00 Did a manual profile update for subjects root and joe.
```

Figure 6.34: Instance Remarks Window

- **SaveToFile** — Saves the currently displayed instance's configuration data to an ASCII text file; when you select this option, you are prompted for the name of the file.
- **HELP** — Presents information about using the Instance View Window.

6.6 Customize Menu

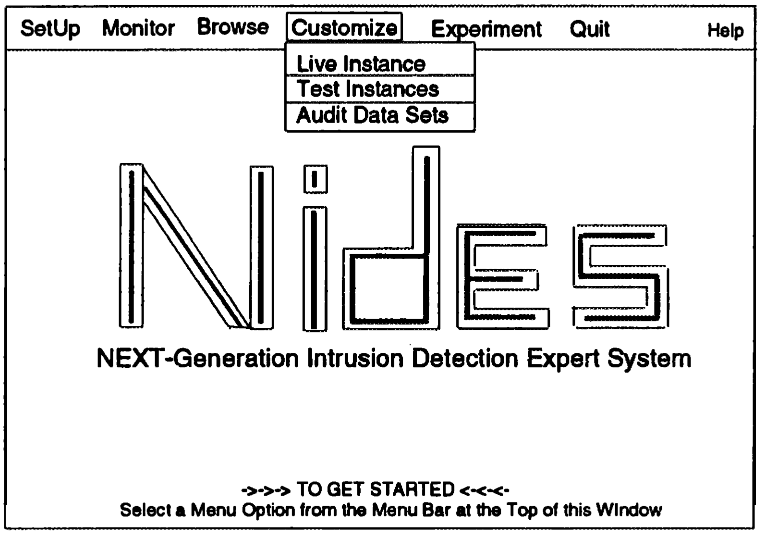

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
Live Instance
Test Instances
Audit Data Sets
NIdES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.35: NIDES Main Window Customize Menu

The Customize Menu contains options that support the customization of the NIDES analysis components for both real-time and experimental audit data analysis. The Customize Menu, shown in Figure 6.35, contains three options:

- **Live Instance** — Supports configuration of the NIDES real-time intrusion detection analysis; this option is available only if the real-time analysis is currently running
- **Test Instance** — Supports configuration of NIDES test instances that are used to run intrusion detection experiments; this is a privileged user option
- **Audit Data Sets** — Supports the creation of audit data sets that are used as input for NIDES experiments; this is a privileged user option

6.6.1 Live Instance and Test Instances Options

The Live Instance and Test Instances options of the Main Window Customize Menu allow you to customize the analysis components for real-time analysis and experimental analysis, respectively. Because these two functions are nearly identical, the Live Instance and Test Instances customization functions are described together; their differences are noted when appropriate. Many of the configuration options have associated profile re-training costs. See Section 6.6.1.4 and Tables 6.17, 6.18, and 6.19, which describe configuration retraining costs and application methods.

6.6.1.1 Managing Test Instances

One of the key differences between the real-time instance and test instances is that test instances are created and managed by the user. When the Test Instances option is selected from the Customize Menu, the Instance Management Window like the one shown in Figure 6.36 is displayed.

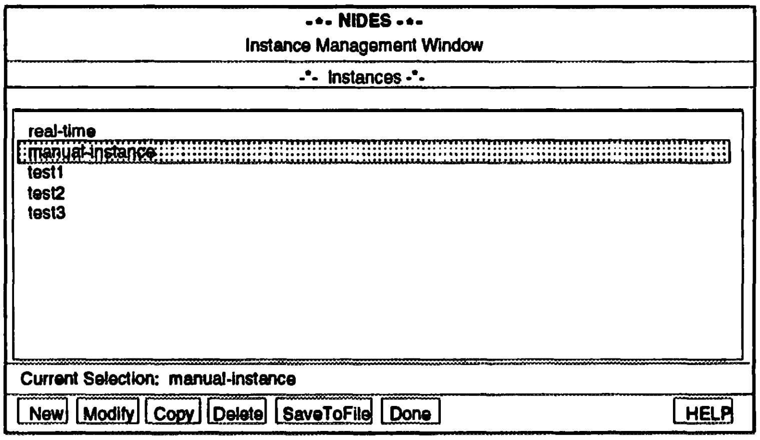


Figure 6.36: Instance Management Window

The top part of the window lists current instances and highlights the currently selected instance. Below the instance list is a row of seven option buttons:

- **New** — Creates a new instance. A name entry window is presented when this option is selected.
- **Modify** — Modifies the configuration of the currently selected instance. If no instance has been selected this button is deactivated. To modify the real-time instance, use the Live Instance option of the Customize Menu. The modify option under the Instance Management Window is deactivated when the real-time instance is selected.
- **Copy** — Makes a copy of an existing instance into a new instance. A name entry window is presented when this option is selected. If no instance has been selected as the source of the copy, this button is deactivated.
- **Delete** — Deletes the currently selected instance, including all experiment results associated with the instance. When this option is chosen, a confirmation window is displayed. If no instance has been selected this button is deactivated.
- **SaveToFile** — Saves information about the selected instance to an ASCII text file. If no instance has been selected this button is deactivated.
- **Done** — Exits the Instance Management Window and returns you to the Main Window.
- **HELP** — Presents help on the instance management functions.

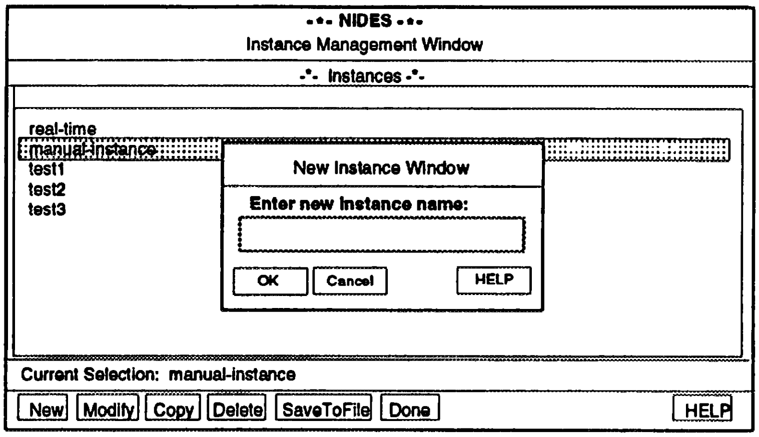


Figure 6.37: New Instance Window

6.6.1.1.1 New Option When you select the *New* option on the Instance Management Window, a name entry window, as shown in Figure 6.37, is displayed. To enter in the new instance's name, click in the name entry box and the box is highlighted. If you type in characters that are not acceptable for an instance name, the system beeps, and your characters are not placed in the name entry box. Select *OK* to confirm the entry, and a new instance is created using the name entered. The *Cancel* option cancels the new instance function and returns you to the Instance Management Window. The *HELP* option gives you guidance on how to enter the name, and format guidelines for instance names.

6.6.1.1.2 Modify Option When you select the Modify option, a window showing all modify options available for test instances is displayed. Refer to Section 6.6.1.2 on instance configuration for a description of options.

6.6.1.1.3 Copy Option The *Copy* option is very similar to the New option, except that the new instance is a copy of an existing instance. The new instance's configuration and profiles are copied from the selected instance. When you select the Copy option on the Instance Management Window, a name entry window, similar to the one shown in Figure 6.37, is displayed. Entry of the new instance name is the same as in the New instance option. Select *OK* to confirm the entry, and a new instance is created that is a copy of the configuration and profiles from the selected source instance. The *Cancel* option cancels the copy instance function and returns you to the Instance Management Window. The *HELP* option gives you guidance on name entry and name format guidelines.

6.6.1.1.4 Delete Option The Delete option deletes the selected instance, including all profiles created for the instance, and all test result data created during experiments using the instance. When the Delete option is selected, a confirmation window is displayed. The *OK* option in the confirmation window initiates the deletion process. If the instance contains a large number of profiles and/or test results data, it may take a minute or two to complete the deletion. The *Cancel* option in the confirmation window cancels the deletion (nothing is deleted) and returns you to the

Instance Management Window. The *HELP* option gives you information on confirmation window usage.

6.6.1.1.5 SaveToFile Option When you select the *SaveToFile* option on the Instance Management Window, a filename entry window is displayed. To enter the filename, click in the entry box and the box is highlighted. If you type in characters that are not acceptable for a filename, the system beeps, and your characters are not placed in the entry box. When you have entered the filename, select *OK* to confirm it, and the selected instance's data is written to the file. The *Cancel* option cancels the SaveToFile function and returns you to the Instance Management Window. The *HELP* option gives you guidance on how to enter the filename, and format guidelines for names.

6.6.1.2 Instance Configuration Options

The instance configuration options for the Live Instance and Test Instances are slightly different. Figure 6.38 shows the window that is displayed when the Live Instance option is selected from the Customize Menu. Figure 6.39 shows the window displayed when the *Modify* option is selected from the Instance Management Window.

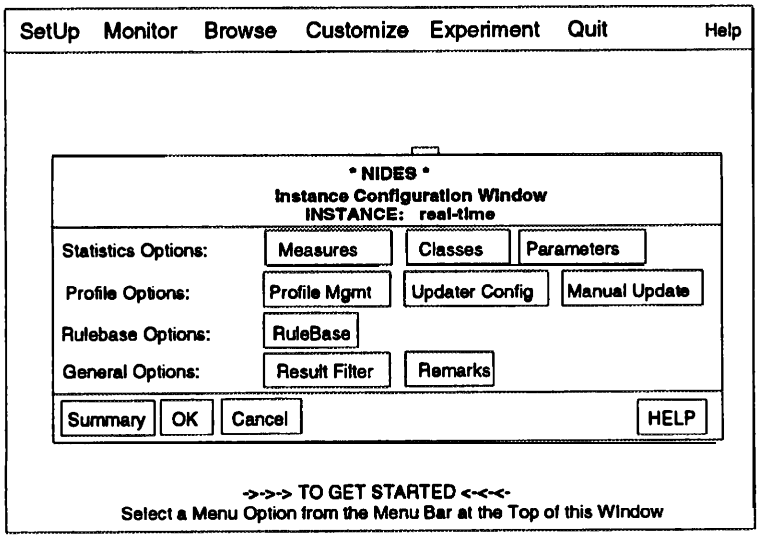


Figure 6.38: NIDES Real-time Instance Configuration Window

There are a total of ten options for instance configuration. Seven of the options are common to both Live and Test Instances. Two options (Updater Config and Manual Update) are valid for the Live Instance only. One option (Updater Mode) is valid for Test Instances only. The ten options are

- **Measures** — Configures measures ON or OFF and the parameters of each measure (Qmax, Scalar, Short-term half-life and Minimum Effective-N)
- **Classes** — Adds or deletes members of the eight statistics classes
- **Parameters** — Configures general statistical analysis component parameters, including long-term half-life, training period, threshold settings, and profile cache size

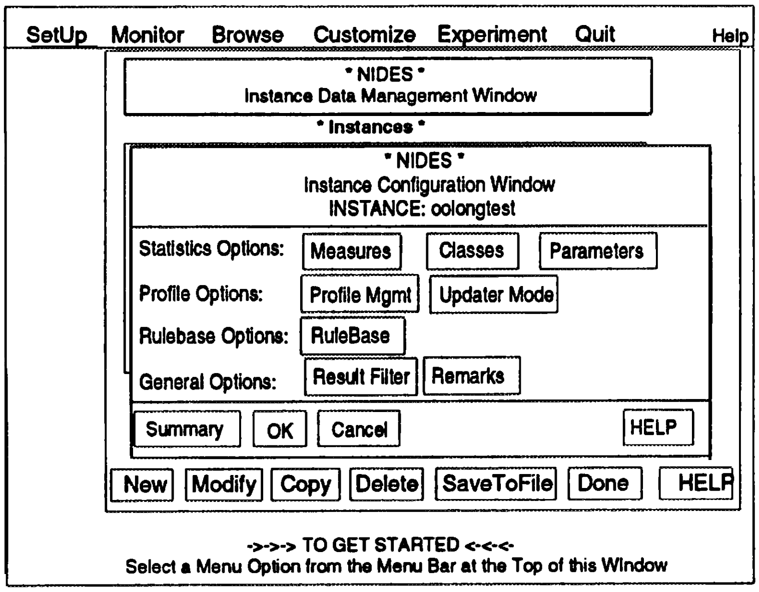
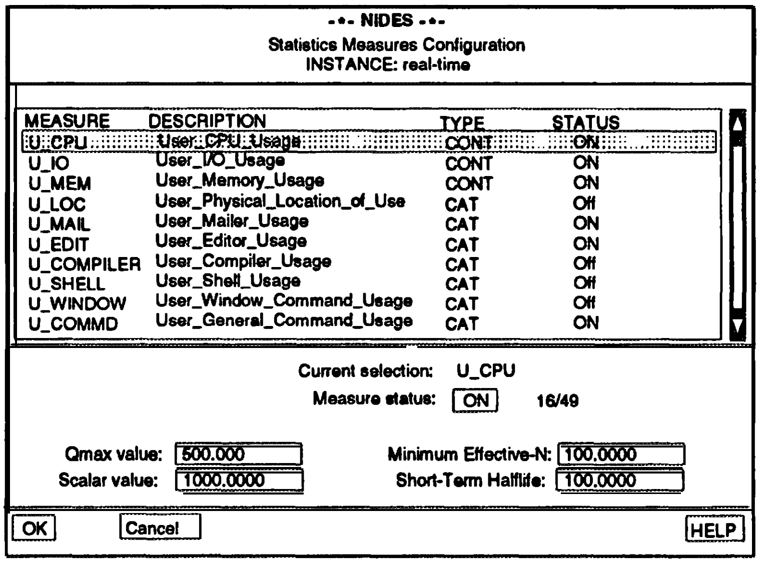

SetUp Monitor Browse Customize Experiment Quit Help
* NIDES *
Instance Data Management Window
* Instances *
* NIDES *
Instance Configuration Window
INSTANCE: oolongtest
Statistics Options: Measures Classes Parameters
Profile Options: Profile Mgmt Updater Mode
Rulebase Options: RuleBase
General Options: Result Filter Remarks
Summary OK Cancel HELP
New Modify Copy Delete SaveToFile Done HELP
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.39: NIDES Test Instance Configuration Window

- **Profile Mgmt** — Displays, copies, replaces, and deletes profiles of subjects included in the selected instance. Profile viewing allows you to review the profile's training status, number of updates, category lists, and other aspects of the profile data
- **Updater Config** — Configures the real-time profile update method and schedule, and selects which subjects'profiles will be updated; available for the real-time instance only
- **Manual Update** — Performs an instantaneous profile update on selected subjects' profiles; available for the real-time instance only
- **Updater Mode** — Turns profile updating ON or OFF; available for test instances only
- **Rulebase** — Configures the rulebase by turning available rules ON or OFF
- **Result Filter** — Determines the types of NIDES analysis results that are written into the results archive
- **Remarks** — Enters general remarks about the selected instance

The following paragraphs describe the ten configuration options in greater detail, discuss the default values for the various configuration items, and explain how the various configuration changes are applied to the analysis components. We recommend reviewing all these sections prior to making any configuration changes. Also see Chapter 4, which discusses statistical analysis configuration and Chapter 5, which discusses rulebased analysis configuration.

6.6.1.2.1 Measure Configuration Option When you select the *Measures* option under the Instance Configuration Window, a window as shown in Figure 6.40 is displayed. If you configure any of the measure parameters including the ON/OFF status, review Section 4.6 in this manual, which discusses measure configuration. Normally, you may decide to turn a measure ON or OFF based on examination of profile training status or on knowledge of the audit data your environment will provide.



-•- NIDES -•-

Statistics Measures Configuration

INSTANCE: real-time

MEASURE	DESCRIPTION	TYPE	STATUS
U_CPU	User_CPU_Usage	CONT	ON
U_IO	User_I/O_Usage	CONT	ON
U_MEM	User_Memory_Usage	CONT	ON
U_LOC	User_Physical_Location_of_Use	CAT	Off
U_MAIL	User_Mailer_Usage	CAT	ON
U_EDIT	User_Editor_Usage	CAT	ON
U_COMPILER	User_Compiler_Usage	CAT	Off
U_SHELL	User_Shell_Usage	CAT	Off
U_WINDOW	User_Window_Command_Usage	CAT	Off
U_COMMD	User_General_Command_Usage	CAT	ON

Current selection: U_CPU

Measure status: ON 16/49

Qmax value: 500.000

Scalar value: 1000.0000

Minimum Effective-N: 100.0000

Short-Term Halflife: 100.0000

OK Cancel HELP

Figure 6.40: Measure Configuration Window

In the main view area of the window is a scrollable list of all statistical measures. The type and configuration of each measure are listed after the measure ID and description. If a measure is ON and has been trained, it will contribute to statistical anomaly detection; if a measure is OFF it will not contribute, but will be trained so that if it is switched ON it will be able to contribute as soon as possible. Below the list of measures are additional configuration items for the currently selected measure. Select the measure you want; it is highlighted and the Current Selection field is updated with the selected measure's ID. The fields below the measure list are filled in with the values for the selected measure. The items shown below the listing of measures are

- **Measure status/count** — The selected measure's status is listed in a box below the selected measure's ID. Displayed to the right is the number of measures activated (turned ON) and the total number of measures available. For example, 16/49 indicates that out of a total of 49 available measures, 16 are currently turned ON. If you want to change the state of the selected measure, click in the Measure status (ON/OFF) box located below the Current Selection area. The state changes with each mouse click. For information on measure activation see Section 4.6.1.

- **Qmax value** — Determines binning ranges for the Q distribution. If you want to change the Qmax value, review Sections 4.7.1.4, 4.7.1.5 and 4.6.3 prior to making any change. Since the same Qmax value applies to all subjects, the NIDES user should be sure that the value is significantly misscaled for the entire group of subjects before changing it. NIDES functions satisfactorily with a value that might be somewhat high for most subjects but successfully contains the data for extreme cases.

To change the value, click in the box containing the current Qmax value. The box is highlighted and you may edit the value. If you enter invalid characters (non-numeric), the system beeps and your characters are not entered. Changing the Qmax value is a privileged user function.

- **Scalar value** — Used to determines category bin ranges for continuous measures only. Review Sections 4.7.1.3 and 4.6.2 prior to making any change in the Scalar value. A Scalar value set somewhat high is preferable, since the same value applies to all subjects.

  To change the value, click in the box containing the current Scalar value; if the box does not contain a number, then the current measure selected does not require a scalar configuration (i.e., it is not a continuous measure). The box is highlighted and you may edit the value. If you enter invalid characters (non-numeric), the system beeps and your characters are not entered. Changing the Scalar value is a privileged user function.

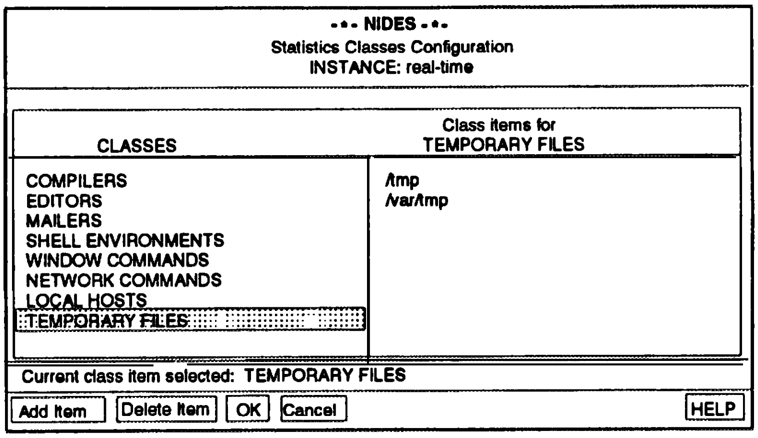
- **Minimum Effective-N** — Represents the minimum number of observations modified by aging factors that must be observed before the measure will contribute to the score, regardless of the measure's training status. Review Section 4.6.4 prior to making any change in the Minimum Effective-N value. To change the value, click in the box containing the current Minimum Effective-N value. The box is highlighted and you may edit the value. If you enter invalid characters (non-numeric), the system beeps and your characters are not entered.

- **Short-term Half-life** — Represents the number of observations that are made before the short-term profile entry for the measure ages the observations by one half. Review Sections 4.1.2 and 4.6.5 prior to making any change in Short-term half-life value. To change the value, click in the box containing the current Short-term half-life value. The box is highlighted and you may edit the value. If you enter invalid characters (non-numeric), the system beeps and your characters are not entered.

At the bottom of the measure configuration window are buttons labeled OK, Cancel, and HELP. When the *OK* option is selected, all errors in the entered data are reported (Table 6.11 lists valid values for each measure parameter). If there are no errors in the entered data, a window summarizing the configuration for all active measures is displayed. *Deactivated measures are not displayed in the summary window, even if their parameter values were modified.* The *OK* option confirms your changes and stores them. The changes are applied when you confirm all configuration changes in the Instance Configuration Window. After *OK* is selected in the confirmation window, you are returned to the Instance Configuration Window. The *Cancel* option returns you to the Statistics Measures Configuration Window, and the *HELP* option gives you information on the confirmation window.

The *Cancel* option on the Statistics Measures Configuration Window returns you to the Instance Configuration Window and no changes are made. The *HELP* option gives information on the Statistics Measure Configuration Window.

6.6.1.2.2 Class Configuration When you select the *Classes* option under the Instance Configuration Window, a window as shown in Figure 6.41 is displayed. The left half of the main area lists available classes. The right half displays members of the selected class. When you select a class, it is highlighted, and the right side of the window is populated with the list of class members.


-•- NIDES -•-
Statistics Classes Configuration
INSTANCE: real-time

CLASSES

COMPILERS
EDITORS
MAILERS
SHELL ENVIRONMENTS
WINDOW COMMANDS
NETWORK COMMANDS
LOCAL HOSTS
TEMPORARY FILES

Class items for
TEMPORARY FILES

/tmp
/var/tmp

Current class item selected: TEMPORARY FILES

Figure 6.41: Class Configuration Window

Class lists represent the categories that are generated by associated statistical measures. One exception, the temporary files class, is not tied to a particular measure; it represents the list of files and directories that will not generate categories under the statistics file measures, so in a sense it is a "negative" class. Table 6.10 describes the classes available. Review Section 4.4 prior to making any changes to the class lists.

At the bottom of the class configuration window are five buttons labeled Add Item, Delete Item, OK, Cancel, and HELP:

- **Add Item** — Adds a new member to the currently selected class. When the *Add Item* option is selected, a text entry window is displayed where you may enter the new items name. Selecting the *OK* option on the name entry window adds the item to your class list, the *Cancel* option cancels the entry and returns you to the Statistics Classes Configuration Window, and the *HELP* option gives guidance on the text entry window.

- **Delete Item** — Activates when you have selected a class item from the currently displayed class list. The current selection is listed in the Current class item selection area, just above the row of option buttons. The *Delete Item* option removes the item from the list. The item is not permanently removed until all instance configurations are confirmed at the Instance Configuration Window.

- **OK** — Displays a confirmation window listing all the class changes made. The *OK* option saves your changes, but they will not be permanent until confirmation is made in the Instance Configuration Window. The *Cancel* option in the confirmation window returns you to the Statistics Classes Configuration Window.

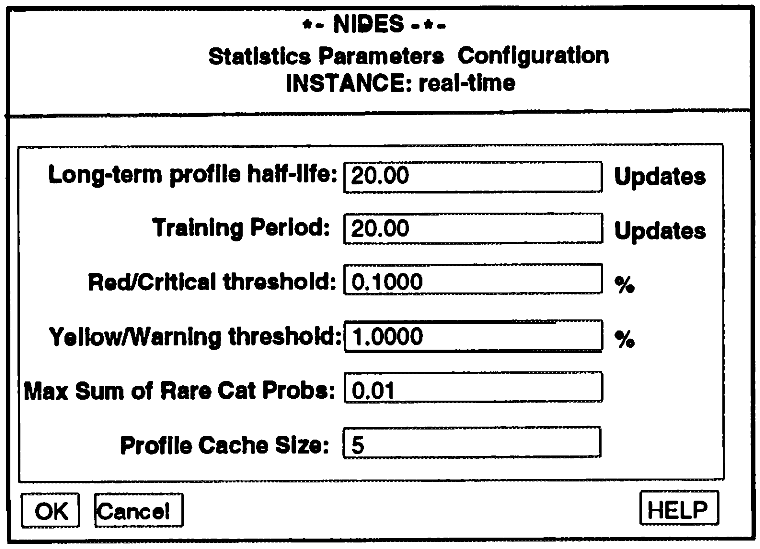
- **Cancel** — Cancels the changes made and returns you to the Instance Configuration Window.

- **HELP** — Gives you information about the class configuration window.

6.6.1.2.3 Parameter Configuration When you select the *Parameters* option under the Instance Configuration Window, a window as shown in Figure 6.42 is displayed. The main area of the window lists the six statistics parameters that are configurable across all measures:

Class/Description	Measures Using Class	Example Members
COMPILERS Commands/programs invoked that are compilers.	U_COMPILER	bison,cc,f77,g++ gcc,m4,yacc.
EDITORS Commands/programs invoked that are editors.	U_EDIT	awk,e,ed,edit,emacs,ex, lemacs,less,more,perl,sed,
MAILERS Commands/programs invoked that are associated with e-mail.	U_MAIL	Mail,comp,dist,folders, forw,inc,mail,mailtool, mh-eMail,mhmail,mhpath,
SHELL ENVIRONMENTS Commands/programs that are shells.	U_SHELL	/bin/bash,/bin/csh, sh,csh,bash,
WINDOW COMMANDS Commands/programs associated with windows.	U_WINDOW	X,xcalc,sunview,mailtool, suntools,xinit,shelltool
NETWORK COMMANDS Command/programs that are network based or remote.	U_RNETTYP	cu,fingerd,ftp,in.fingerd, kermit,mount,telnet,on, rcp,rdate,rdist,rsh,tip,
LOCAL HOSTS Hosts that are local to your network.	U_RNET,U_LNET	List must be initialized with user's local hosts.
TEMPORARY FILES Files and directories that contain temporary files. These files are not relevant for statistical analysis because users will not establish patterns of usage regarding these filenames	U_FILE,U_DIR U_DIRNEW,U_FILENEW U_DIRDEL,U_FILEDEL U_DIRMOD,U_FILEMOD U_DIRREAD,U_FILEREAD	/tmp, /var/tmp

Table 6.10: Statistics Classes Descriptions


- NIDES --
Statistics Parameters Configuration
INSTANCE: real-time

Long-term profile half-life:	20.00	Updates
Training Period:	20.00	Updates
Red/Critical threshold:	0.1000	%
Yellow/Warning threshold:	1.0000	%
Max Sum of Rare Cat Probs:	0.01	
Profile Cache Size:	5	

Figure 6.42: Statistics Parameters Configuration Window

- **Long-term Profile Half-life** — Time period (measured in number of profile updates) after which the contribution of a given day's data is downweighted by one half. Review Section 4.5.1 before changing this value.
- **Training Period** — Interval of time (measured in number of profile updates) required before the statistical analysis scoring mechanism generates alerts. Review Section 4.5.2 before changing this value.
- **Red/Critical Threshold** — Percentage indicating percentile of activity that will be considered critical; this percentage is used in the calculation of the red threshold score value. Review Section 4.5.3 before changing this value.
- **Yellow/Warning Threshold** — Percentage indicating percentile of activity that will be considered a warning; this percentage is used in the calculation of the yellow threshold score value. Review Section 4.5.3 before changing this value.
- **Max Sum of Rare Cat Probs** — Maximum sum that is totaled for any measure's RARE categories (i.e., those categories that count infrequent/rare observations). Increasing or decreasing this parameter (and we recommend only slight changes) can increase or decrease the number of categories that are considered RARE. Review Section 4.5.4 before changing this value.
- **Profile Cache Size** — Most recently used profiles during NIDES analysis are kept in a cache; the number of profiles maintained in the cache is determined by the profile cache size. NIDES performance can be tuned by changing the cache size. Review Section 4.5.5 before changing this value.

If you click on the item you want to modify, its box is highlighted and you may edit the contents. Values entered are verified when you select *OK*. Table 6.11 lists valid data values for each statistics parameter.

Item	Valid Values
Measure Parameters	
Qmax	10 to 1000.
Scalar	0 to 100,000,000.
Minimum Effective-N	0 to 100,000.
Short-term Half-life	0 to 100,000.
Classes	
Class Member	alpha-numeric and /
Statistics Parameters	
Long-term Half-life	1-365 days.
Training Period	1-365 days.
Red/Critical Threshold	0.001% to 100.0%.
Yellow/Warning Threshold	0.001% to 100.0%.
Max Sum Rare Prob.	.0001 to .25
Cache	1 to 100.

Table 6.11: Statistics Configuration Items - Valid Values

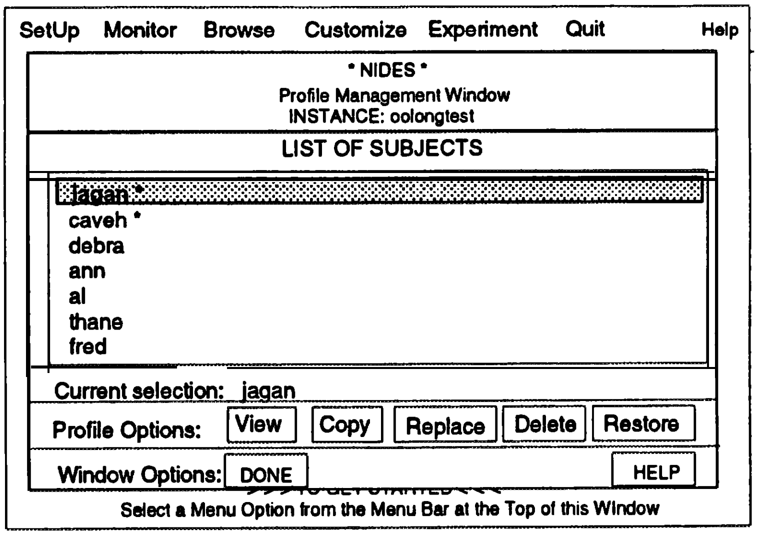
When you have completed your modifications, select OK to display a confirmation window listing your changes. The *OK* option saves your changes, but they will not be permanent until confirmation is made in the Instance Configuration Window. The *Cancel* option in the confirmation window returns you to the Statistics Parameters Configuration Window.

The *Cancel* option in the Statistics Parameters Configuration Window cancels your changes to the parameters and return you to the Instance Modify Window. The *HELP* option gives information about the Statistics Parameters Configuration Window.

6.6.1.2.4 Profile Management When you select the *Profile Management* option under the Instance Configuration Window, a window as shown in Figure 6.43 is displayed. The main area of the window lists all subjects that have profiles for the selected instance. The * annotation for a subject names indicates that at least one measure for the subject is trained; therefore statistical alert reporting is active for that subject. Below the list of profiles are two sets of buttons — five option buttons, and two window buttons:

- **View** — Displays the contents of a subject's profile. Refer to Section 6.5.3.2.4, starting on page 139, for a discussion on viewing subject profiles.

- **Copy** — Copies an existing profile into a new profile. This is useful when you want to initialize a new subject's profile with an existing subject's trained profile. The Copy option is activated only when a profile has been selected. To copy an existing subject's profile, select the subject. The selected subject is highlighted and the Current selection area is updated with the selected subject. When you select *Copy,* a window is presented for entry of the new profile name. To confirm the copy, select *OK* in the entry window; to cancel, select *Cancel.* Actual copies are not made until final confirmation is made in the Instance Management Window.



* NIDES *
Profile Management Window
INSTANCE: oolongtest

LIST OF SUBJECTS

jagan *
caveh *
debra
ann
al
thane
fred

Current selection: jagan

Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.43: Profile Management Window

- **Replace** — The *Replace* option is similar to the *Copy* option except that an existing profile is copied into another existing profile. The *Replace* option is activated only when a profile has been selected. To replace an existing subject's profile, select the subject. When you select the *Replace* option, a window is presented for entry of the profile that should replace the selected profile. The replacement is recorded when you select *OK* in the entry window, and the profile listing is annotated to reflect that the profile has been changed. To cancel the replacement, select *Cancel* in the name entry window. If, after confirming the replacement you decide not to perform the replacement, the *Restore* option restores the replaced profile to its original state, so long as the profile changes have not been activated by final confirmation in the Instance Management Window.

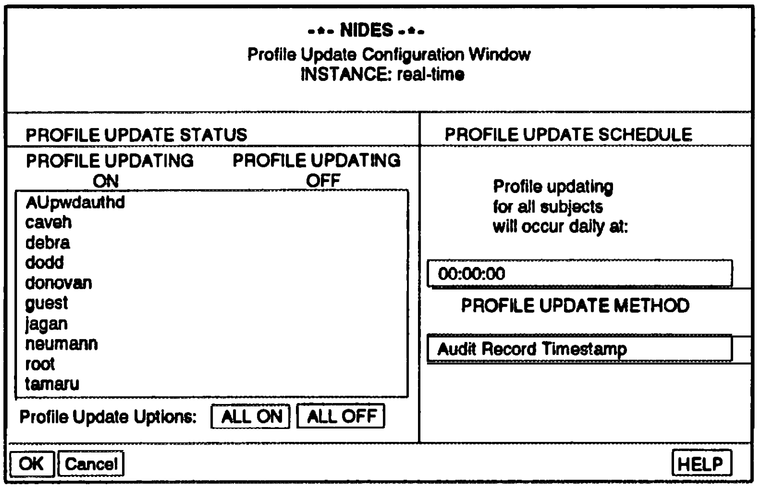
- **Delete** — Deletes a profile from the selected instance. Note that this is the *only* option under this window that is activated immediately upon confirmation. The profile is deleted when confirmed and cannot be restored. The *Delete* option is activated only when a profile has been selected. To delete a subject's profile, select the subject. When you select *Delete,* a confirmation window is displayed. To delete the profile, select *OK* in the confirmation window. To cancel, select *Cancel.* Remember that once you have confirmed the deletion the profile is gone for good!

- **Restore** — Restores the selected profile to its original state, if it has been modified (replaced) by another profile. The *Restore* option is activated only when a modified profile has been selected. To restore a modified profile, select the profile. When you select *Restore* a confirmation window is displayed. To restore the profile, select *OK* in the confirmation window. To cancel, select *Cancel.*

- **Done** — Returns you to the Instance Management Window.

- **HELP** — Gives information on the Profile Management Window.

6.6.1.2.5 Updater Config The *Updater Config* option is available only for the real-time instance. When you select the *Updater Config* option under the Instance Configuration Window, a window as shown in Figure 6.44 is displayed. The Profile Update Configuration Window has three configuration areas:



-•- NIDES -•-
Profile Update Configuration Window
INSTANCE: real-time

PROFILE UPDATE STATUS

PROFILE UPDATING ON	PROFILE UPDATING OFF
AUpwdauthd	
caveh	
debra	
dodd	
donovan	
guest	
jagan	
neumann	
root	
tamaru	

Profile Update Uptions:

PROFILE UPDATE SCHEDULE

Profile updating for all subjects will occur daily at:

00:00:00

PROFILE UPDATE METHOD

Audit Record Timestamp

Figure 6.44: Real-Time Updater Configuration Window

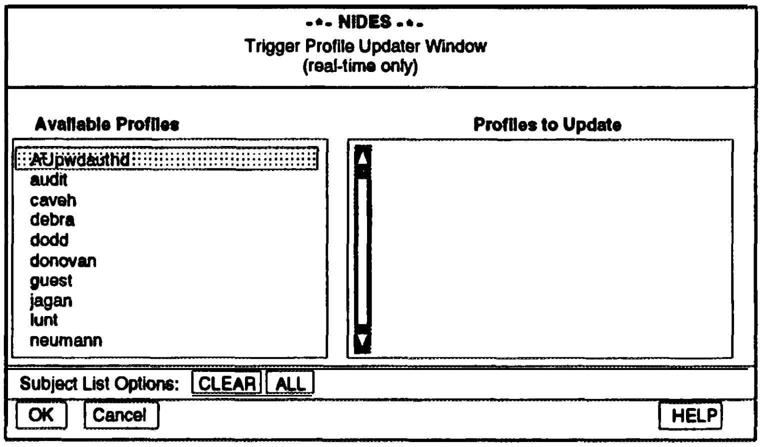
- **Profile Update Status** — Allows you to select which subjects'profiles will be updated when a scheduled profile update occurs. The left side lists subjects whose profiles will be updated when profile updating is scheduled to occur. The right side lists subjects whose profiles will not be updated. Selecting any subject moves it from one list to the other. Changes to the lists are confirmed when you select *OK.*

- **Update Schedule** — Allows you to select the time when scheduled daily profile updates will occur. Profile updating is a compute-intensive process, so it is a good idea to schedule the updates at a time when the system is not busy. The update schedule is based on a 24-hour clock. To change the value, click in the box containing the current time. The box will be highlighted, and you may edit the time in the box. Be sure to enter a valid time from 00:00:00 to 23:59:59. Changes to this value are confirmed when you select *OK.*

- **Update Method** — Allows you to choose how NIDES determines when an update should occur. There are two possible methods. The audit record timestamp updates profiles based on the timestamps in the audit records being processed. The system clock method updates profiles based on the NIDES computer's system clock regardless of the timestamps of the audit records. To change the update method, click on the method listed. The method will be toggled.

At the bottom of the Profile Update Configuration Window are buttons labeled OK, Cancel, and HELP. When the *OK* option is selected, a confirmation window is displayed listing the changes made. If the *OK* option in the confirmation window is selected, the changes are recorded but are not permanent until the *OK* option is selected in the Instance Configuration Window. The *Cancel* option returns you to the Instance Configuration Window, and the *HELP* option gives information on the Profile Update Configuration Window.

6.6.1.2.6 Manual Update The *Manual Update* option is available only for the real-time instance. This option allows you to update profiles immediately regardless of the profile updater configuration. When *you* select the *Manual Update* option under the Instance Configuration Window, a window as shown in Figure 6.45 is displayed.



AUpwdauthd
audit
caveh
debra
dodd
donovan
guest
jagan
lunt
neumann

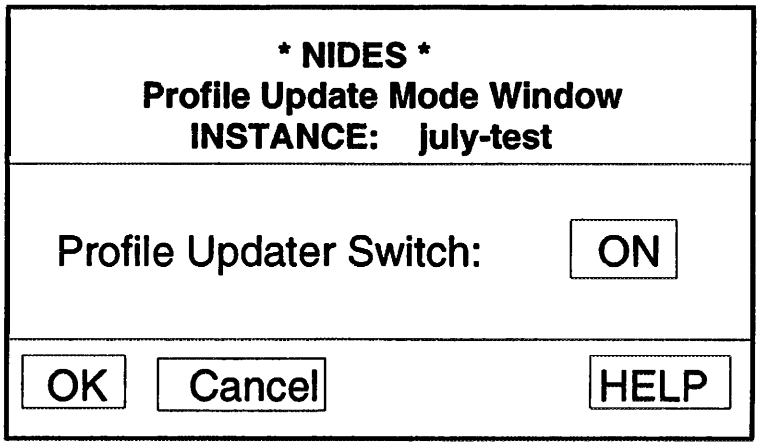
Figure 6.45: Trigger Profile Updater Window

The Trigger Profile Updater Window has one area, allowing you to select which subjects' profiles will be updated, and is divided into two sections. On the left side is the list of subjects whose profiles are available for update (this list includes profiles whose update configuration is switched OFF). On the right side is the list of subjects whose profiles will be updated when the update is confirmed. You can move a subject from one list to the other by selecting it. Below the list of available profiles are two convenience buttons, Clear and All. The *Clear* option places all profiles in the Available Profiles list, and the *All* option places all profiles in the Profiles to Update list.

The *OK* option presents a confirmation window listing profiles that will be updated. Once confirmed, the profiles are updated *immediately.* Depending on the number of profiles selected for update, the updates could take several minutes to complete. To cancel the update, select *Cancel* in the confirmation window. To exit the Trigger Profile Updater Window, select *Cancel.* The *HELP* option provides information on the manual update function.

6.6.1.2.7 Updater Mode The *Updater Mode* option is available only for test instances. When you select the *Updater Mode* option under the Instance Configuration Window, a window as shown in Figure 6.46 is displayed.

The Profile Update Mode Window shows the update mode for the selected instance, either ON or OFF. If the updater switch is set to ON, any experiment run using this instance will update profiles based on the audit record timestamps, with an update schedule set to 00:00:00. If the updater switch is set to OFF, no updates will occur during an experiment using the instance. To change the updater mode, click on the Profile Updater Switch. The value of the updater switch will be toggled. At the bottom of the window are buttons labeled OK, Cancel, and HELP. *OK*



* NIDES *
Profile Update Mode Window
INSTANCE: july-test

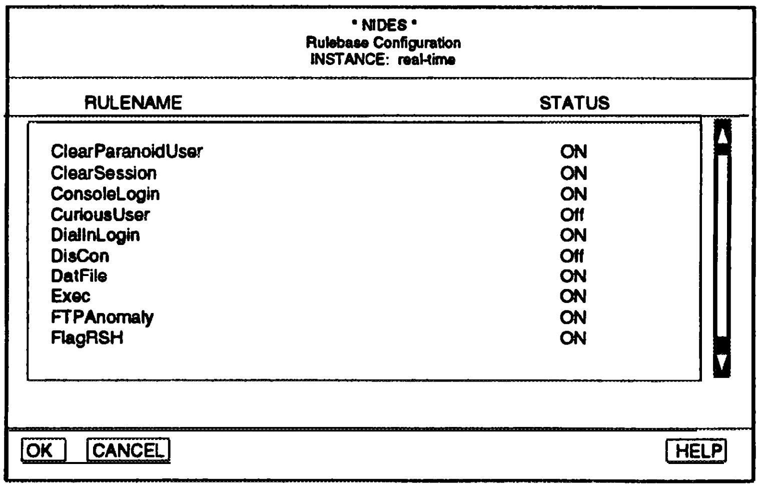
Profile Updater Switch: ON

Figure 6.46: Profile Update Mode Configuration Window

confirms your setting. Cancel returns you to the Instance Configuration Window, and *HELP* gives information on the Profile Update Mode Window.

6.6.1.2.8 Rulebase Configuration When you select the *Rules* option under the Instance Configuration Window, a window as shown in Figure 6.47 is displayed. This window contains a list of the names of all available rules and their configurations (ON or OFF). If the rule is switched ON, it is used to analyze audit data received by NIDES; if it is switched OFF it is not used. Click on a rule to change its setting. The rule setting is toggled.

Before switching rules ON and OFF, review Chapter 5 on rulebase configuration. In particular, Section 5.5 describes the default rulebase configuration and rules. Some rules function as a group and must be turned ON or OFF as a group. Rule groups are listed in Table 5.7 and discussed in Section 5.5.1. Also, if you write a new set of rules that function as a group, you will need to turn them ON or OFF as a group.

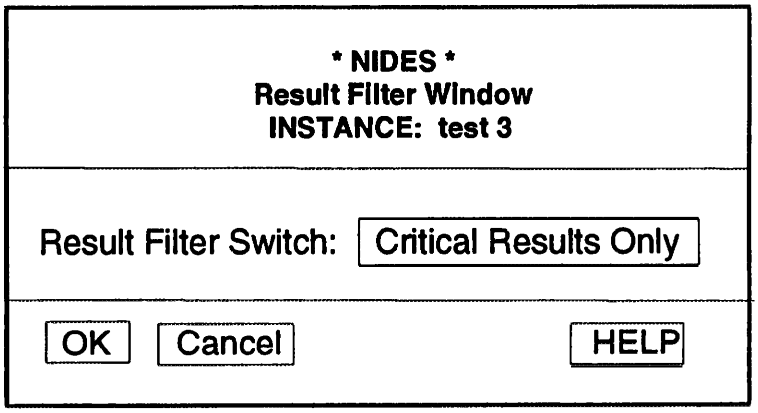


* NIDES *
Rulebase Configuration
INSTANCE: real-time

RULENAME	STATUS
ClearParanoidUser	ON
ClearSession	ON
ConsoleLogin	ON
CuriousUser	Off
DialInLogin	ON
DisCon	Off
DatFile	ON
Exec	ON
FTPAnomaly	ON
FlagRSH	ON

Figure 6.47: Rulebase Configuration Window

Below the window containing the list of rules are buttons labeled OK, Cancel, and HELP. *OK* confirms changes made to the rulebase configuration. *Cancel* returns you to the Instance Configuration Window, and *HELP* gives information on the Rulebase Configuration Window.


* NIDES *
Result Filter Window
INSTANCE: test 3

Result Filter Switch: Critical Results Only

Figure 6.48: Result Filter Configuration Window

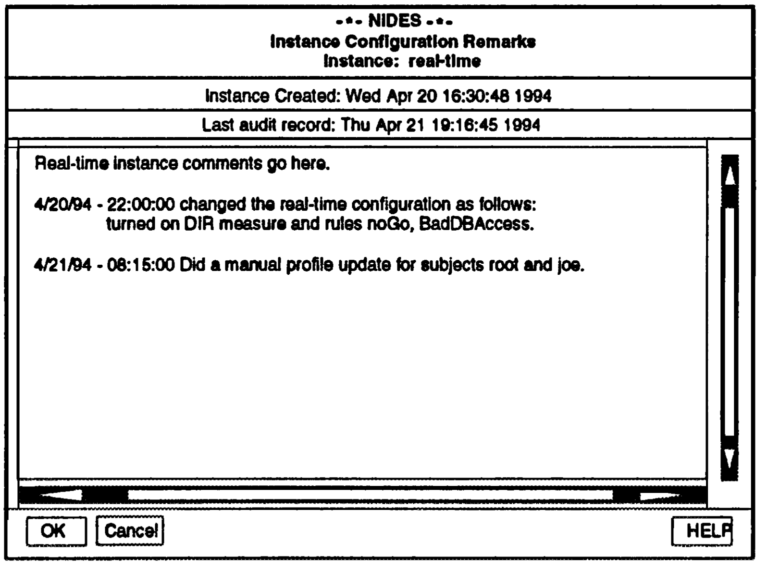
6.6.1.2.9 Result Filter Configuration When you select the *Result Filter* option under the Instance Configuration Window, a window as shown in Figure 6.48 is displayed. This window shows the current result filter setting in a box whose value can be toggled between the three possible result filter settings by clicking in the box. The three possible result filter configurations are

- **Critical Results Only** — Indicates that results at the critical level will be archived.
- **Warning Level and Above** — Indicates that results at the critical or warning level will be archived.
- **All Results** — Indicates that all results generated will be archived. Since for each audit record processed a result record is generated, each audit record seen will generate a result record in the archive.

Below the result filter switch setting are buttons labeled OK, Cancel, and HELP. *OK* confirms the changes made to the result filter configuration. *Cancel* returns you to the Instance Configuration Window, and *HELP* gives information on the result filter configuration window.

IMPORTANT: Configuration of the result filter to the higher levels (i.e., critical or warning and above) can speed up NIDES processing and also save space on your disk. Set the result filter to the highest level that will suit your needs.

6.6.1.2.10 Remarks Configuration When you select the *Remarks* option under the Instance Configuration Window, a window as shown in Figure 6.49 is displayed. At the top of the window are two fields. The first lists the time the instance was initially created, and the second shows the timestamp of the last audit record processed by the instance. Below these two fields is a window where comments about the instance may be entered by clicking in the comment text area. The comment area is highlighted and you may edit/enter any free text.



Instance Created: Wed Apr 20 16:30:48 1994

Last audit record: Thu Apr 21 19:16:45 1994

Real-time instance comments go here.

4/20/94 - 22:00:00 changed the real-time configuration as follows:
turned on DIR measure and rules noGo, BadDBAccess.

4/21/94 - 08:15:00 Did a manual profile update for subjects root and joe.

Figure 6.49: Instance Remarks Window

At the bottom of the window are buttons labeled OK, Cancel, and HELP. *OK* confirms changes to the comments. Changes to the remarks are recorded when confirmed, not later when the Instance Configuration Window is OK'd. *Cancel* returns you to the Instance Configuration Window, and *HELP* gives information on the Instance Configuration Remarks Window.

6.6.1.3 Configuration Default Values

Tables 6.12 and 6.13 show the default configurations for statistical measures: ON or OFF status, Qmax, Scalar, Short-term half-life, and Minimum Effective-N. For a description of the measures, refer to Tables 4.1 and 4.2.

Table 6.14 lists the default class members for the statistics classes.

Table 6.15 lists the default parameter values for the long-term profile half-life, profile training period, red/critical threshold, yellow/warning threshold, maximum sum for rare probabilities, profile cache size, profile update mode and configuration, and result archive filter.

Table 6.16 lists the default rules (all are switched ON) and whether or not they generate an alert.

6.6.1.4 Configuration Activation and Ramifications

Table 6.17 lists all the analysis configuration options and shows when the configurations will be applied. Configuration changes can be applied either immediately or when the next profile update occurs.

6.6.1.4.1 Immediate Reconfiguration Application For the real-time analysis operation, immediate configurations are applied as soon as the reconfiguration message is received by the analysis processes. After the changes have been applied, the real-time instance configuration file is updated to reflect the new configuration in use.

Measure Default Configuration Values					
Measure	Status	QMAX	Scalar	Min Eff-N	Half-life
U_CPU	**ON**	100	1,000	100	100
U_IO	**ON**	100	10,000,000	100	100
U_MEM	**ON**	100	10,000,000	100	100
U_LOC	OFF	100	n/a	100	100
U_MAIL	**ON**	100	n/a	100	100
U_EDIT	**ON**	100	n/a	100	100
U_COMPILER	OFF	100	n/a	100	100
U_SHELL	OFF	100	n/a	100	100
U_WINDOW	OFF	100	n/a	100	100
U_COMMD	**ON**	100	n/a	100	100
U_COMMDB	OFF	100	n/a	100	100
U_COMMDC	**ON**	100	n/a	100	100
U_SYSCALL	OFF	100	n/a	100	100
U_DIR	OFF	100	n/a	100	100
U_DIRB	OFF	100	n/a	100	100
U_DIRNEW	OFF	100	n/a	100	100
U_DIRDEL	OFF	100	n/a	100	100
U_DIRMOD	OFF	100	n/a	100	100
U_DIRREAD	OFF	100	n/a	100	100
U_FILENEW	OFF	100	n/a	100	100
U_FILEREAD	OFF	100	n/a	100	100
U_FILEMOD	OFF	100	n/a	100	100
U_FILEDEL	OFF	100	n/a	100	100
U_FILETMP	OFF	100	n/a	100	100
U_FILE	OFF	100	n/a	100	100
U_FILEB	OFF	100	n/a	100	100
U_UID	OFF	100	n/a	100	100
U_UIDB	**ON**	100	n/a	100	100

Table 6.12: Statistical Analysis Component - Default Measure Configuration (part 1)

Measure Default Configuration Values					
Measure	Status	QMAX	Scalar	Min Eff-N	Half-life
U_SYSERR	ON	100	1,000	100	100
U_SYSERRTYP	OFF	100	n/a	100	100
U_AUDREC	OFF	100	n/a	100	100
U_HOUR	ON	200	n/a	100	100
U_HOURB	OFF	100	n/a	100	100
U_DAILY	OFF	100	n/a	100	100
U_DAILYB	OFF	100	n/a	100	100
U_RNET	ON	100	n/a	100	100
U_RNETTYP	OFF	100	n/a	100	100
U_RNETHOST	OFF	100	n/a	100	100
U_LNET	OFF	100	n/a	100	100
U_LNETTYP	OFF	100	n/a	100	100
U_LNETHOST	OFF	100	n/a	100	100
U_INTARR	ON	100	172,800	100	100
U_FCLASS	OFF	100	n/a	100	100
U_FCLSRD	OFF	100	n/a	100	100
U_FCLSWR	OFF	100	n/a	100	100
U_ARECDIST	ON	100	n/a	100	100
U_INT60	ON	100	1	100	100
U_INT600	ON	100	1	100	100
U_INT3600	ON	200	1	100	100

Table 6.13: Statistical Analysis Component - Default Measure Configuration (part 2)

Class	Default Members
COMPILERS	bison,cc,f77,g++ gcc,m4,yacc.
EDITORS	awk,e,ed,edit,emacs,ex, lemacs,less,more,perl, sed,textedit,vi
MAILERS	Mail,comp,dist,folders,forw, inc,mail,mailtool,mh-eMail, mhmail,mhpath,mm,msgchk,msh, mush,next,pick,prev,refile, repl,rmail,rmf,rmm,scan, sendmail,show,vmh.
SHELL ENVIRONMENTS	/bin/bash,/bin/csh,/bin/sh, /bin/tcsh,bash,cmdtool,csh, sh,tcsh.
WINDOW COMMANDS	X,Xsun,XsunMONO,bdftopcf,bdftosnf,imake, kbd_mode,mftobdf,mwm,tvtwm,twm,uwm,xauth, xbiff,xcalc,xclock,xconsole,xdbx,xdpyinfo, xdvi,xedit,xhost,xinfo,xinit,xload,xloadimage, xlock,xlogo,xlsatoms,xlsclients,xlsfonts, xlswins,xmessage,xmh,xmodmap,xplaces,xprop, xrdb,xrefresh,xselection,xset,xsetbg,xsetroot, xshowcmap,xstdcmap,xterm,xv,xwd,xwininfo,xwud, adjacentscreens,clock,cmdtool,defaultsedit, fileview,fontedit,get_selection,gfxtool, iconedit,lockscreen,mailtool,overview, perfmeter,selection_svc,shelltool,suntools, sunview,sv_acquire,sv_release,swin,switcher, tektool,textedit,toolplaces
NETWORK COMMANDS	cu,fingerd,ftp,in.fingerd, kermit,mount,nntpxmit,on,rcp,rdate, rdist,rex,rexec,rlogin,rmt, rnusers,rquota,rsh,rstat,rtime, rusers,rwall,showmount,telnet, tip,uux.
LOCAL HOSTS	<empty list> – should be initialized.
TEMPORARY FILES	/tmp, /var/tmp

Table 6.14: Statistics Classes Default Members

Statistics Parameters	Default Value
Long-term Half-life	20
Training Period	20
Red/Critical Threshold	0.1%
Yellow/Warning Threshold	1.0%
Max Sum Rare Prob.	0.01
Cache	5
Miscellaneous Parameters	**Default Value**
Update Mode (Tests Only)	ON
Updater Configuration (Real-time Only)	Audit Record Timestamps
Update Schedule (Real-time Only)	00:00:00
Result Archive Filter	Warning level and above.

Table 6.15: NIDES Statistics and Miscellaneous Parameters - Default Values

For test instances, the immediate reconfigurations are applied as soon as a test is initiated. Changing the configuration of a test instance used for an executing test is not allowed. However, the test instance configuration can be changed after the test is completed.

6.6.1.4.2 Deferred Reconfiguration Application Some configuration actions are performed at profile update time. For real-time operation, the profile update that triggers the application of any pending reconfiguration can be a regularly scheduled profile update or can be user-initiated via the manual profile update option. After the reconfigurations have been applied, the real-time instance configuration file is updated to reflect the current configuration. The pending reconfiguration information, that was available via the Browse Menu, is cleared and the Pending Reconfiguration option will not be available until another reconfiguration is made.

For test instances, deferred reconfiguration options are applied in one of two ways. If the test instance that has been reconfigured is a new instance and therefore contains no profiles, the deferred reconfiguration options are applied when the test is initiated. However, if the modified test instance has been used for previous experiments, and therefore contains some profiles, the pending reconfiguration items are applied at the next profile update. After the deferred reconfiguration has been applied during the experiment, the pending reconfiguration information, that was available via the Browse Menu, is cleared and the Pending Reconfiguration option for the test instance will not be available until another reconfiguration is made.

6.6.1.4.3 Configuration Ramifications Many of the items that can be reconfigured have a profile retraining cost associated with them. When you modify these items, statistical profiles may need retraining, and therefore statistical anomaly reporting will be switched off until the retraining process is completed. Tables 6.18 and 6.19 show profile retraining requirements for all of the reconfiguration options. To get an idea of how long retraining will take, review the training period configuration (this is measured in number of updates, which equates to number of days, unless manual profile updates are made). There are three training phases, C, Q, and T2. Each

Rulebase Default Rules *Rules listed in* **boldface** *generate alerts*	
AccessPrivateDevice	**Leapfrog1**
AccessPrivateFile1	**LinkSystemExec**
AccessPrivateFile2	LocalLogin
AccessSpecialFile	Logout1
BackwardsTime	Logout2
BadLogin1	**ModSystemExec**
BadLogin2	MultLogin1
BadLoginAnomaly	MultLogin2
BadLoginBadPassword	**NoRemote**
BadPassword1	ParanoidUser1
BadPassword2	ParanoidUser3
BadPasswordAnomaly	**ParanoidUserAnom**
BadRoot	**PasswordFileAccess**
BadUserExec	**ReadSystemExec**
BrokeRoot	**RemoteExec**
ChangeLoginFile	**RemoteFile1**
ChmodOtherUser	**RemoteFile2**
ChmodSystemFile	**RemoteFile3**
ClearParanoidUser	RemoteLogin
ClearSession	**RemoteMount1**
configured	**RemoteMount2**
ConsoleLogin	**RemoteRootBadLogin**
DialInLogin	**RemoteRootBadPassword**
Dot File	remove_event
Exec	RunsRareExec
FTPAnomaly	set_time
FlagRSH	**SpecUserExec**
GoodLogin1	**Su1**
GoodLogin2	**SuspiciousUser**
GoodPassword1	**TFTPAnomaly**
GoodPassword2	TFTPUse
GoodSU1	TouchSession
GoodSU2	**TrojanHorse**
InvisibleDirectory	**TruncateLog**
KnownLogin1	

Table 6.16: NIDES Default Rules

Analysis Configuration Activation	
Configuration Item	**Activation**
Measure Configuration	
Activation(ON)/Deactivation(Off)	Immediate
Qmax	Next Profile Update
Scalar	Next Profile Update
Minimum Effective-N	Next Profile Update
Short-term Half-life	Next Profile Update
Class Configuration	
Add Members	Next Profile Update
Delete Members	Next Profile Update
Statistics Parameter Configuration	
Long-term Profile Half-life	Next Profile Update
Training Period	Next Profile Update
Red/Critical Threshold	Next Profile Update
Yellow/Warning Threshold	Next Profile Update
Sum Rare Prob.	Next Profile Update
Profile Cache Size	Immediate
Profile Management	
Copy	Immediate
Replace	Immediate
Delete	Immediate
Profile Updater Configuration	
Profile Update Time	Immediate
Profile Update Method	Immediate
Subject Profile Update (Update Flag ON/Off)	Next Profile Update
Manual Profile Update Option	
Subject Profile Update	Immediate
Updater Mode Option	
Update Profiles (ON/OFF)	Immediate
Rulebase Configuration	
Rules Turned ON	Immediate
Rules Turned Off	Immediate
Result Filter Configuration	
Result Filter Modifications	Immediate

Table 6.17: Analysis Configuration Activation Modes

	Phases Retrained				
Configuration Item	C	Q	T2	None	Ramifications
Measure Items					
Activation(ON) and Deactivation(Off)				X	Measures turned **ON** and already trained will contribute to statistical scoring on the next audit record. Measures turned **ON** and not yet trained will continue to train and will contribute only when training is completed. Measures turned **OFF** will continue to train but will not contribute to scoring.
Qmax		X	X		***(2/3 of training period measure is off-line)*** During retraining the measure will *NOT* contribute to scoring
Scalar	X	X	X		During retraining the measure will *NOT* contribute to scoring
Minimum Effective-N				X	Affects measure in training ONLY. Measures already trained are not affected.
Short-term Half-life	X	X	X		During retraining the measure will *NOT* contribute to scoring
Class Items					
Add Members				X	If the member exists in some profiles as a normal (not class-list) category, it will take time to age out
Delete Members				X	Future observations for deleted class members will enter the profiles as normal (non-class item) categories.
Statistics Parameters					
Long-term Profile Half-life				X	Possible increase in false alarm rate if set too low; slow adaptation to new behavior if too high.
Training Period				X	Only affects future training phases of untrained measures.
Red/Critical Threshold				X	Allows user control of false alarm rate.
Yellow/Warning Threshold				X	Allows user control of false alarm rate.
Sum Rare Prob.				X	If set too high, distorts the notion of rare occurrences. Default value should be appropriate in virtually all situations.
Profile Cache Size				X	Affects system throughput. May be configured larger than default on machines with large swap space/memory.

Table 6.18: Analysis Re-configuration Ramifications (part 1)

Configuration Item	Phases Retrained C	Q	T2	None	Ramifications
Profile Management Copy				X	Useful to supply a new subject with a default profile quickly, but may have a high false alarm rate.
Replace				X	Useful for cross-profiling studies.
Delete				X	New records for the subject will appear as a new subject. A new profile will be built for the subject
Profile Updater *(real-time only)* Profile Update Time				X	Should be set to a time of low system activity as updates are compute-intensive.
Profile Update Method				X	Update by system clock ensures that profiles will be updated the same time each day. Updates based on timestamps occur only when activity past the update time boundary is observed. So, exact time of update is not guaranteed.
Subject Profile Update (Update Flag ON/Off)				X	Setting updating OFF can be used to prevent updates for a subject during periods of unusual but legitimate activity.
Manual Update *(real-time only)* Subject Profile Update				X	Too many manual updates will cause the profile to be considered trained when it is not stable. A single manual update will force pending reconfigurations to be applied. Can help profile adapt more rapidly to new user behavior as long as the user already has a trained profile.
Update Mode *(test instances only)* Updating ON/Off				X	Useful for cross-profiling experiments (e.g., to run several data streams through the same stable profile).
Rulebase Config Rules Turned ON				X	Rules that are part of a group should be turned ON at the same time.
Rules Turned Off				X	Rules that are part of a group should be turned Off at the same time.
Result Filter Result Filter Changes				X	Volume of result data will increase if data levels are added and will decrease if levels are deleted. Increased data volume affects system performance.

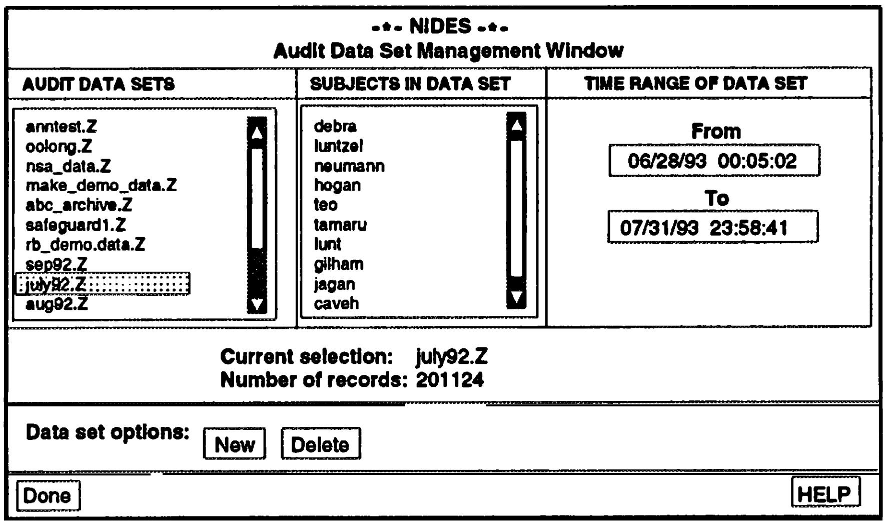
Table 6.19: Analysis Re-configuration Ramifications (part 2)

phase will take one third of the training period to retrain; if the training period does not divide evenly by 3, round up to the next whole number. As you will see in the tables, many of the items do not, require full (C, Q and T2) retraining, but rather some subset of the three phases.

6.6.2 Audit Data Sets Option

The Audit Data Sets option of the Main Window Customize Menu lets you manage and create NIDES audit data sets. Audit data sets are subsets of archived data that can be used as input for NIDES test experiments. They can be represented in two ways:

- **Adset File** — The audit data file is an actual file that contains NIDES audit records. When a test is executed, this file is read directly. This file is normally compressed with the standard UNIX compression utility (identified by a .Z tag at the end of the file name).
- **DMF Index** — The index file contains key information (such as subject names and timestamp ranges) that will enable the batch analysis process to extract the relevant audit data from the selected DMF archive. This type of audit data set is sometimes called a "virtual" data set. Using the DMF Index mode can save considerable disk space.



-*- NIDES -*-
Audit Data Set Management Window

AUDIT DATA SETS	SUBJECTS IN DATA SET	TIME RANGE OF DATA SET
anntest.Z	debra	From
oolong.Z	luntzel	06/28/93 00:05:02
nsa_data.Z	neumann	To
make_demo_data.Z	hogan	07/31/93 23:58:41
abc_archive.Z	teo	
safeguard1.Z	tamaru	
rb_demo.data.Z	lunt	
sep92.Z	gilham	
july92.Z	jagan	
aug92.Z	caveh	

Current selection: july92.Z
Number of records: 201124

Data set options: New Delete

Done HELP

Figure 6.50: Audit Data Set Management Window

When you select the Audit Data Sets option from the Customize Menu, the Audit Data Set Management Window as shown in Figure 6.50 is displayed. On the left side of this window is a list of audit data sets that are available for test runs. You may select any of these audit data sets by clicking on the desired name. If the list of data sets is longer than the size of the window, you can use the vertical scrollbar on the right side of the list. Your selection is highlighted, and other parts of the window are populated.

The middle section lists subjects whose audit records are contained in the audit data set. This list is populated when you make a selection from the audit data set list. A vertical scrollbar is displayed if the list of subjects is longer than the size of the list window.

The right side shows the timestamps of the first and last audit records contained in the audit data set you have selected, giving you the full time span covered by the audit data set.

Below the lists and time range windows, there is a panel of information indicating the current selection you have made, the total number of records contained in the audit data set, and what kind of audit data set it is (file or index only). If the data set is an index only set, then the name of the DMF archive from which the audit data will be extracted is displayed. Otherwise, the string "n/a" is shown, indicating that the selected data set is a file containing actual audit data, not simply an index.

Below the information panel are two data set management options:

- **New** – Allows you to create a brand new audit data set. A name entry window is displayed when you select this option.
- **Delete** — Deletes the audit data set you have selected.

At the bottom of the window are two options, *Done* and *HELP*. The *Done* option exits the Audit Data Set Management Window and returns you to the Main Window. *HELP* gives overall guidance for the window.

6.6.2.1 New Option

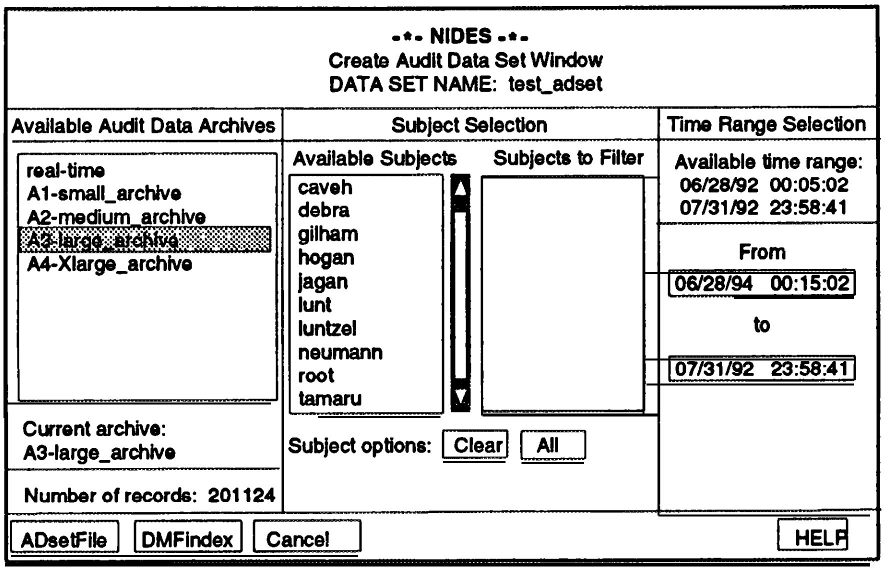
When you select New on the Audit Data Set Management Window, a name entry window appears. To enter the name of the audit data set you want to create, click in the name entry box. The box is highlighted, and you can type in the desired name. If you type in characters that are not allowable for audit data set names, the system beeps, and those characters are not entered. The *OK* option confirms your entry, and *Cancel* cancels the new audit data set function and returns you to the Audit Data Set Management Window. *HELP* gives guidance on how to enter the data set name, and which characters are allowable.

When you have confirmed your data set entry, the Create Audit Data Set Window is displayed, as shown in Figure 6.51.

The left side of the Create Audit Data Set window lists DMF archives that are available as audit data sources. Click on an archive to select it. Your selection is highlighted, and other parts of the window are populated, including the total number of audit records contained in the archive.

The middle section lists subjects whose audit records are in the selected archive. You *must* select at least one subject for the audit data set you are creating. The list is divided into two sections. The left side is labeled Available Subjects, and is initially populated with the entire list of subjects contained in the selected archive. The right side is labeled Subjects To Filter and lists all subjects whose data will be included in the audit data set. To select the subject(s) with which you want to create your audit data set, click on the desired names (one at a time). Each name automatically appears on the right side in the Subjects to Filter list. To remove a name from the Subjects to Filter list, click on the desired name. The *Clear* and *All* buttons are available for your convenience. The *Clear* option moves all subjects to the Available Subjects list, and the *All* option moves all subjects to the Subjects to Filter list.

The right side of the window shows the timestamps of the first and last audit record contained in the audit data archive you have selected, and below these two timestamps is an area where you can modify the time ranges for the audit data set you want to create (note that the default setting


-*- NIDES -*-
Create Audit Data Set Window
DATA SET NAME: test_adset

Available Audit Data Archives: real-time, A1-small_archive, A2-medium_archive, A3-large_archive, A4-Xlarge_archive

Current archive: A3-large_archive

Number of records: 201124

Subject Selection

Available Subjects: caveh, debra, gilham, hogan, jagan, lunt, luntzel, neumann, root, tamaru

Subjects to Filter

Time Range Selection

Available time range:
06/28/92 00:05:02
07/31/92 23:58:41

From 06/28/94 00:15:02

to 07/31/92 23:58:41

Figure 6.51: Create Audit Data Set Window

is the full time range of the archive). You may change either or both of the *From* and *To* fields, but must adhere to the following constraints:

1. The *From* time cannot be earlier than the timestamp of the first audit record represented in the archive (indicated in the upper timestamp set).
2. The *To* time cannot be later than the timestamp of the last audit record represented in the archive (indicated in the upper timestamp set).
3. The *From* time must be earlier than the *To* time entered.

To change the time values, click in the time field you want to change. The box surrounding the time entry is highlighted, and you can enter your new time. Be sure to follow the proper format: `(MO/DAY/YR HH:MM:SS)` (e.g., `(12/16/93 15:13:45)`). If you enter an illegal time stamp value or do not use the proper format, an error message is displayed.

At the bottom of this window are four options. When you have finalized your selection criteria, select *ADsetFile* to create an actual audit data set containing the audit data records you have selected from the archive, or select *DMFindex* to create the virtual version of the audit data set. To cancel your request entirely, select *Cancel* to return to the Audit Data Set Management Window. The *HELP* option provides information on the Create Audit Data Set window.

Note that audit data archives tend to be very large, and depending on the amount of audit data you want to select from an archive, creating actual audit data sets can be time-consuming. Hence, the actual process of creating these audit data sets is independent from the NIDES user interface (in UNIX terminology, the process is "forked" off), so that the user can continue to do other NIDES functions without having to wait for the audit data set to be completed. To find out if an audit data set is available, bring up either the Audit Data Set Management Window or the Test Facility Configuration Window (see Figure 6.53), and if the audit data set you have created is listed, then it is ready for use. Using the *DMFindex* option makes your audit data set available immediately and saves disk space. Running a test using a "virtual" audit data set may take slightly longer, as the test facility will have to retrieve the audit data from the archive used to create the audit data set.

6.6.2.2 Delete Option

The *Delete* option deletes the selected audit data set. A confirmation window is displayed. Select OK in the confirmation window to delete the audit data set. If you change your mind and decide to keep the audit data set, select *Cancel* in the confirmation window. The *HELP* option gives you information on how to use the confirmation window.

6.7 Experiment Menu

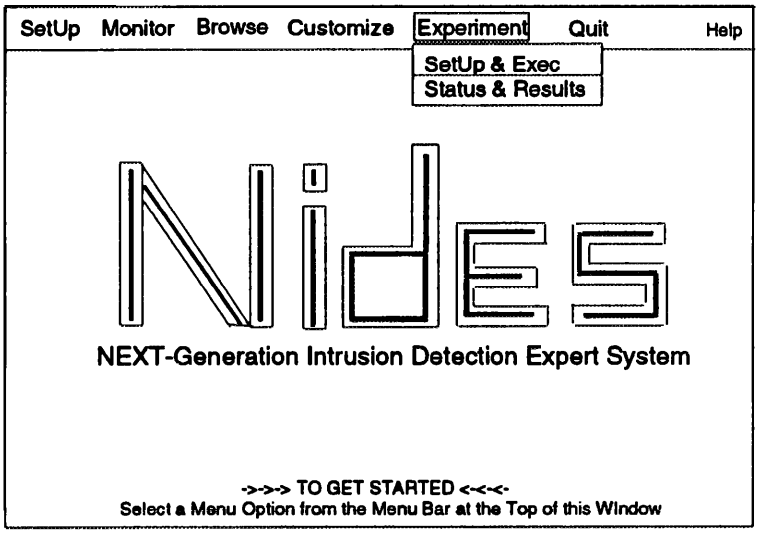
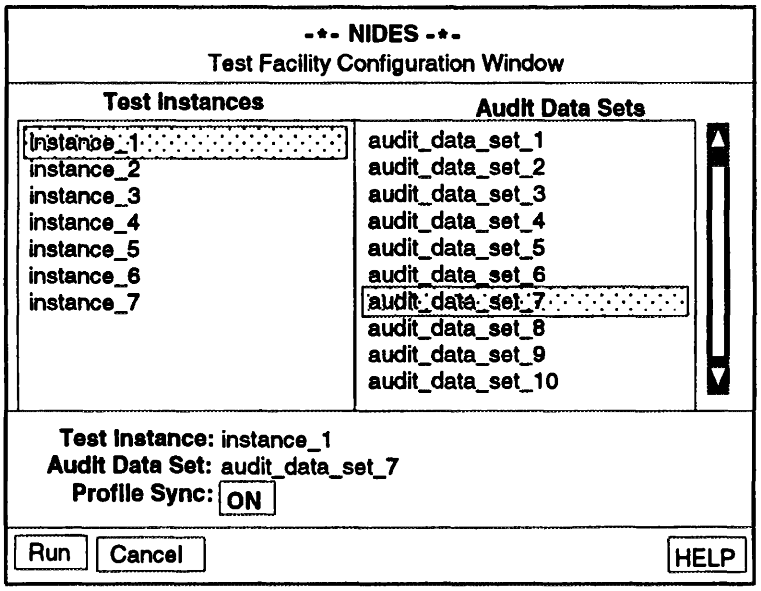

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
SetUp & Exec
Status & Results
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.52: NIDES Main Window Experiment Menu

The Experiment Menu, Figure 6.52, contains options that allow you to run NIDES analysis experiments using test instances and audit data sets. The menu contains two options:

- **SetUp & Exec** — Initiates NIDES test runs after you specify the test instance and audit data set.
- **Status & Results** — Displays the status of all NIDES test runs, both those currently running and those completed. You may also access the Analysis Results View Window from the Test Status/Results Window.

6.7.1 SetUp & Exec Option


-*- NIDES -*-
Test Facility Configuration Window

Test Instances
instance_1
instance_2
instance_3
instance_4
instance_5
instance_6
instance_7

Audit Data Sets
audit_data_set_1
audit_data_set_2
audit_data_set_3
audit_data_set_4
audit_data_set_5
audit_data_set_6
audit_data_set_7
audit_data_set_8
audit_data_set_9
audit_data_set_10

Test Instance: instance_1
Audit Data Set: audit_data_set_7
Profile Sync: ON

Figure 6.53: NIDES Test Facility Configuration Window

To initiate a NIDES experiment, select the SetUp & Exec option on the Experiment Menu. Prior to this step, you should have created a test instance and configured it to meet your test requirements via the Customize Menu Test Instances option. When you select the SetUp & Exec option, a window like the one shown in Figure 6.53 is displayed. The window comprises three areas (from top to bottom):

- **Test Instance and Audit Data Set Selection** — The list on the left side of the area contains test instances currently defined. The list on the right shows available audit data sets. To perform a test, select a test instance and audit data set. Your selections are highlighted and the configuration display area is updated with your selection.
- **Configuration Display** — This area, located beneath the two lists, shows the current selection, and contains one configuration button, for profile synchronization. The profile synchronization flag can be set to ON or OFF. Click on the button to change the configuration. Section **6.7.1.1** discusses profile synchronization.
- **Buttons** — At the bottom of the window are three buttons:
  - **Run** — Initiates your test run; a confirmation window will be displayed.
  - **Cancel** — Exits the Test Facility Configuration Window and returns you to the Main Window.
  - **HELP** — Gives information about the Test Facility Configuration Window.

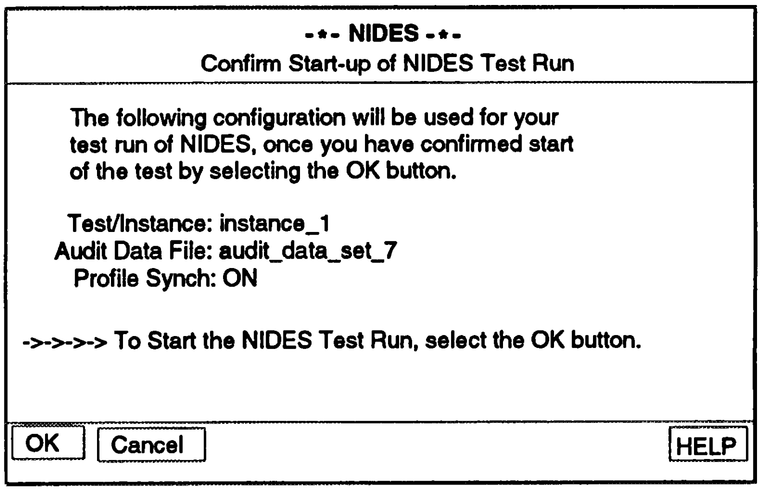
6.7.1.1 Test Profile Synchronization

The Test Facility Configuration Window includes one test configuration item, a profile synchronization flag. If you are running an experiment with an instance that has been used before and therefore contains profiles, and the timestamps in the audit data set you plan to use for your experiment are earlier than previous audit data set timestamps used with the instance, the profile synchronization option ensures that the existing profiles in your instance are updated properly.

The flag can be set to ON or OFF. If the flag is switched ON any profiles that are already part of the test instance will have the last audit record timestamp data synchronized with the audit data set used for the test. This means that the profile date will equal the earliest audit record timestamp in the audit data set used for the test. This can be useful if you are running a test where the dates in the audit data set are earlier than audit data set dates used in a previous test run. If you do not synchronize the profiles, profile updates will not occur until the audit data set timestamps are later than the last update timestamps reflected in the existing profiles. If your current data set timestamps are all earlier than previously used audit data set timestamps, no profile updates will occur during the test run.

If you are performing multiple experiments using the same instance, which is quite reasonable if you are attempting to train a set of profiles, and the audit data sets you are using run in time sequence chronologically, you should not synchronize the profiles in the instance.

6.7.1.2 Test Initiation


-*- NIDES -*-
Confirm Start-up of NIDES Test Run

The following configuration will be used for your test run of NIDES, once you have confirmed start of the test by selecting the OK button.

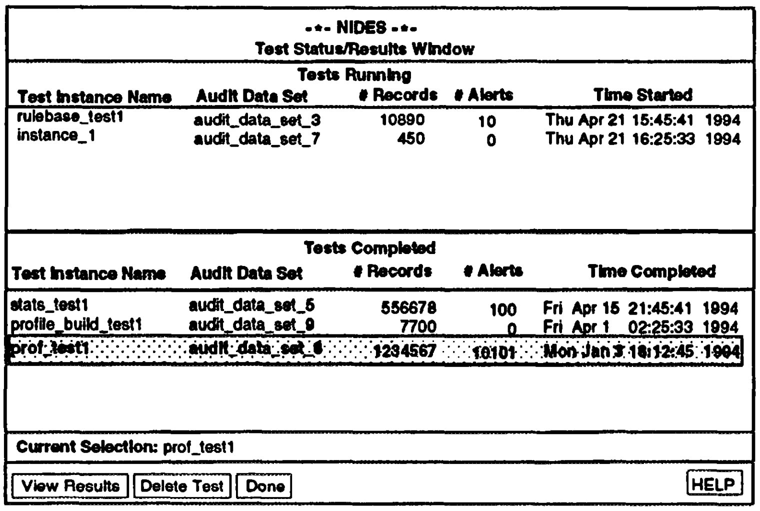
Test/Instance: instance_1
Audit Data File: audit_data_set_7
Profile Synch: ON

->->->-> To Start the NIDES Test Run, select the OK button.

Figure 6.54: NIDES Test Start Confirm Window

Once you have entered in your test configuration, by specifying a test instance and audit data set and setting the profile synchronization flag to the desired value, selecting *Run* initiates your test. A confirmation window as shown in Figure 6.54 is displayed. The confirmation window shows the configuration that will be used for the test. To start your test run, select *OK.* To cancel select *Cancel.* The *HELP* option provides information on the confirmation window. After you have confirmed, a message is displayed indicating that your test is being started.

6.7.2 Status & Results Option


-*- NIDES -*-
Test Status/Results Window

Tests Running

Test Instance Name	Audit Data Set	# Records	# Alerts	Time Started
rulebase_test1	audit_data_set_3	10890	10	Thu Apr 21 15:45:41 1994
instance_1	audit_data_set_7	450	0	Thu Apr 21 16:25:33 1994

Tests Completed

Test Instance Name	Audit Data Set	# Records	# Alerts	Time Completed
stats_test1	audit_data_set_5	556678	100	Fri Apr 15 21:45:41 1994
profile_build_test1	audit_data_set_9	7700	0	Fri Apr 1 02:25:33 1994
prof_test1	audit_data_set_6	1234567	10101	Mon Jan 3 18:12:45 1994

Current Selection: prof_test1

Figure 6.55: NIDES Test Status Window

To review the status of tests, you can select the Status & Results option on the Experiment Menu. When you select the Status & Results option a window as shown in Figure 6.55 is displayed. The window comprises three areas:

- **Test Running Status** — At the top portion of the window is the Tests Running status area that shows the status of currently active test runs.
- **Tests Completed List** — Below the Tests Running area is a list of all completed test runs. Items in this list may be selected for viewing or deletion. Below this list is an area showing the current selection from the completed tests list.
- **Option Buttons** — At the bottom of the window are four buttons:
  - **View Results** — Displays the Analysis Results View Window, with the chosen test preselected in the window. For a complete description on the usage of the Analysis Results View Window, see Section 6.5.2.
  - **Delete Test** — Deletes the test results, once the deletion is confirmed.
  - **Done** — Exits the Test Status/Results Window and returns you to the Main Window.
  - **HELP** — Provides help on usage of the Test Status/Results Window.

6.7.2.1 Tests Running Status

The top portion of the Test Status/Results Window shows currently active tests. Information on the test status is updated approximately every 10 seconds (only the record and alerts counts are updated). Five columns of data are displayed:

- **Test/Instance Name** — Name of the instance used for the test; in NIDES the instance and test name are synonymous.
- **Audit Data Set** — Name of the audit data set used for the test.

- **Number of Records** — Number of audit records processed by the test so far; updated approximately every 10 seconds.
- **Number of Alerts** — Number of alerts generated by the test so far; updated approximately every 10 seconds.
- **Time Started** — Time the test started.

Note that if you exit the NIDES user interface, any active tests will continue to run and if the NIDES user interface is subsequently invoked, active tests will resume status reporting to the new invocation of the NIDES user interface provided NIDES is run under the same environment that was in effect when the test was started.

6.7.2.2 Tests Deletion and Result Viewing

Tests that have completed are listed below the running tests. Once a test has finished, you may view the results of the test and/or delete the test results from the result archive.

6.7.2.2.1 Test Result Viewing To view the results of a test from the Test Status/Results Window, select the test you want to view and select *View Results.* The Analysis Results View Window described in Section 6.5.2 is displayed. When you exit the Analysis Results View Window, you will be returned to the Test Status/Results Window.

6.7.2.2.2 Test Deletion If you have finished using the results of a test, it is a good idea to delete the test results to save disk space. When a test is deleted, only the results are removed; the instance and profiles are maintained. This is useful if you are running tests to build up profiles and are not really interested in the results generated during the profile-building phase. If you want to delete the instance and profiles as well as the test results, use the Delete option on the Instance Management Window. See Section 6.6.1.1.4 on page 150 for a description of this option.

To delete a test from the Test Status/Results window, select the test you want to delete and select *Delete Test.* A confirmation window is displayed. Once the deletion is confirmed, all test results for the selected test are deleted. If your test contained a lot of data, it may take a few moments for the test to be deleted. While the test result data is being removed, a message lets you know that the system is working on the deletion.

6.8 Quit Menu

The Quit Menu of the Main Window, Figure 6.56, contains one option — Quit SOUI, which causes you to exit the NIDES user interface. Once this option is confirmed, the NIDES analysis and arpool servers are stopped and all target hosts are turned OFF. Any experiments started continue to run even after the NIDES user interface exits. When all NIDES real-time analysis processing has been stopped, the user interface exits and returns you to the UNIX shell window.

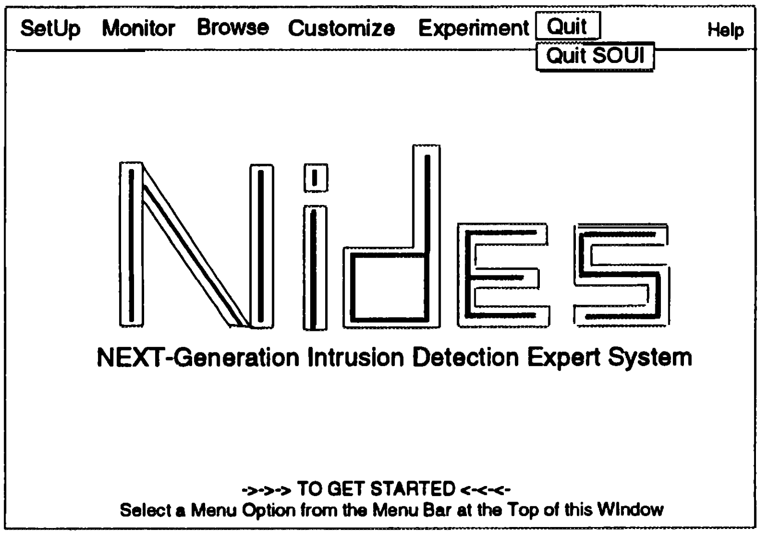

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
Quit SOUI
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.56: NIDES Main Window Quit Menu

6.9 Help System

The NIDES User Interface includes a comprehensive help system. All screens contain a *HELP* option that provides relevant information for the currently active window. To get initial help, select the *Help* option on the Main Window as shown in Figure 6.57, and one of three help window types is displayed. The type of window depends upon where in NIDES the *HELP* option was selected. The window types are

- Top-level Help Information
- Help Topic Menu
- Simple Help Information

6.9.1 Top-Level Help Information Windows

When a *HELP* option is selected, a Top-Level Help Information Window as shown in Figure 6.58 is displayed, if appropriate. As with all Help Windows, this window has two main areas. The top part is an information area, which in this case contains general information relevant to the activities under way. Below the information area is a panel with one or more buttons. In this case, there are two buttons. The Close button closes the window and returns you to where you were when you selected the *HELP* option. The *More HELP* button brings up a Help Topic Menu Window. The *More HELP* button is available only in some windows.

6.9.2 Help Topic Menu Windows

When a *More HELP* option is selected from a Top-Level Help Window, a Help Topic Menu Window is displayed as shown in Figure 6.59. At the bottom of the window is a panel with buttons labeled

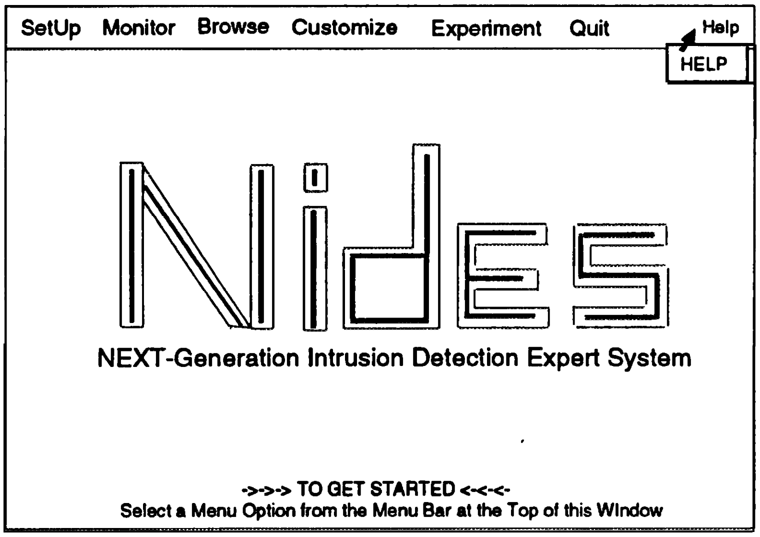

SetUp
Monitor
Browse
Customize
Experiment
Quit
Help
HELP
NIDES
NEXT-Generation Intrusion Detection Expert System
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.57: Help Menu

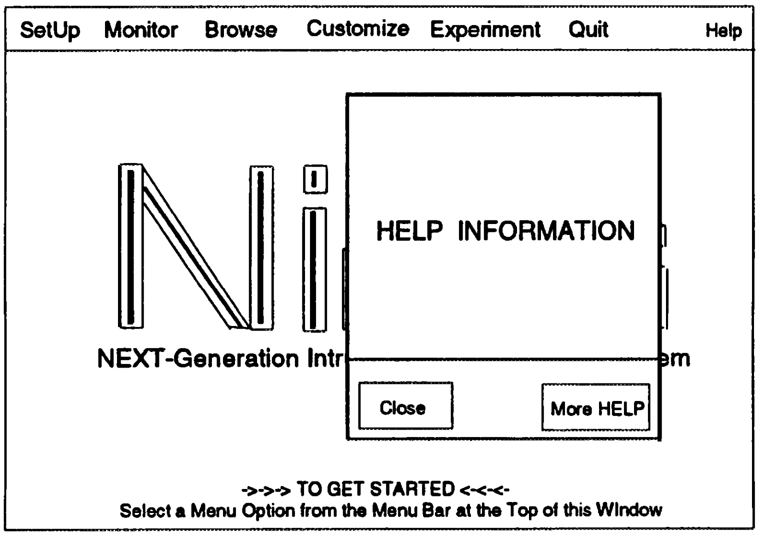


Figure 6.58: NIDES Top-level Help Window

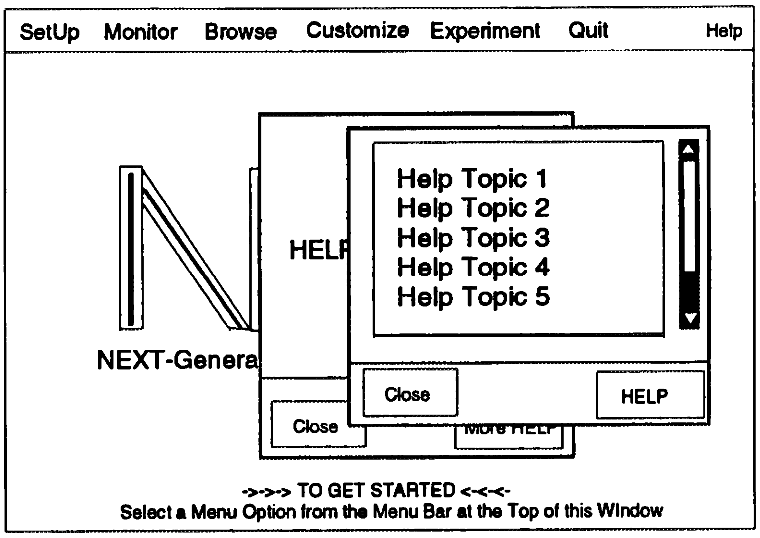


Figure 6.59: Help Topic Menu Window

Close and Help. Above the button panel is a list of available help topics. If the number of topics available cannot be displayed in the window, a scrollbar is located on the right side of the window.

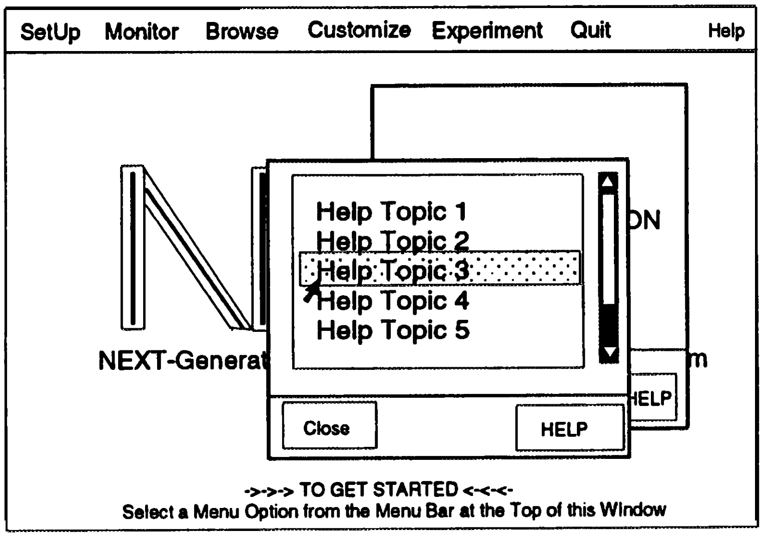


Figure 6.60: Selection of a Help Topic

To select a topic, point to the topic of interest and click the left mouse button, as shown in Figure 6.60. A Simple Help Information Window with information on the selected topic is then displayed as shown in Figure 6.61.

When you have finished selecting topics from the Help Menu Window, select Close to return to the top-level Help system.

6.9.3 Simple Help Information Windows

When a *HELP* option is selected, a Simple Help Information Window as shown in Figure 6.61 is displayed, if appropriate. This window has two main areas. The top part contains general information relevant to the activities under way. Below the information area is a panel with a single button labeled Close. When you have finished reading the help information, select Close to return to where you were when you selected the *HELP* option.

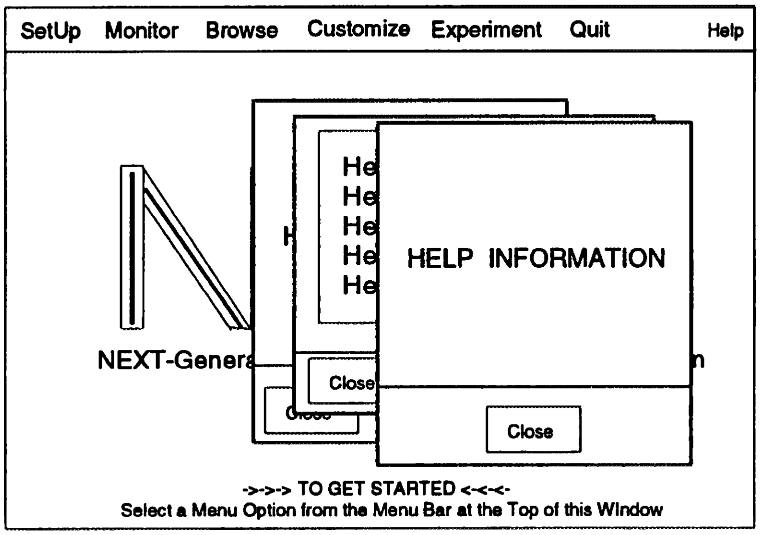

SetUp Monitor Browse Customize Experiment Quit Help
He
He
He
He
He
HELP INFORMATION
NEXT-Gener
n
Close
Close
Close
->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.61: Simple Help Information Window

Help windows are not modal — that is, you can access the window from which the Help window was invoked as you are reading the help information.

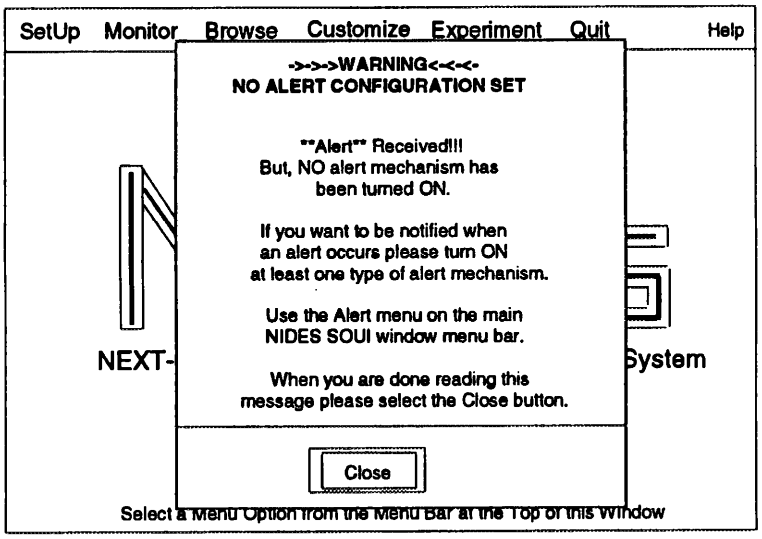
6.10 Receiving Alerts

When an anomalous event is detected by NIDES, you are notified by the selected alert mechanism(s). If an alert is received but all of the alert mechanisms are switched OFF, a warning message as shown in Figure 6.62 is displayed. In all other cases, the alert information is presented in a PopUp message, e-mailed to a list of users, or provided using both methods of reporting.

6.10.1 PopUp Alert Message

When the PopUp Window alert mechanism is switched ON, an Alert Window is displayed when an anomalous event occurs. Figures 6.63 and 6.64 show the types of Alert Windows displayed when an alert is reported. The Statistics Alert Window includes a list of the top five measures that contributed the most to the result. Tables 4.1 and 4.2 list codes that may be presented when an alert is reported by the NIDES statistical analysis component.

After you read the alert information, you must select *Acknowledge* prior to interacting in any way with the NIDES user interface. Until *Acknowledge* is selected, all other menu options are deactivated.



->->->WARNING<-<-<-
NO ALERT CONFIGURATION SET

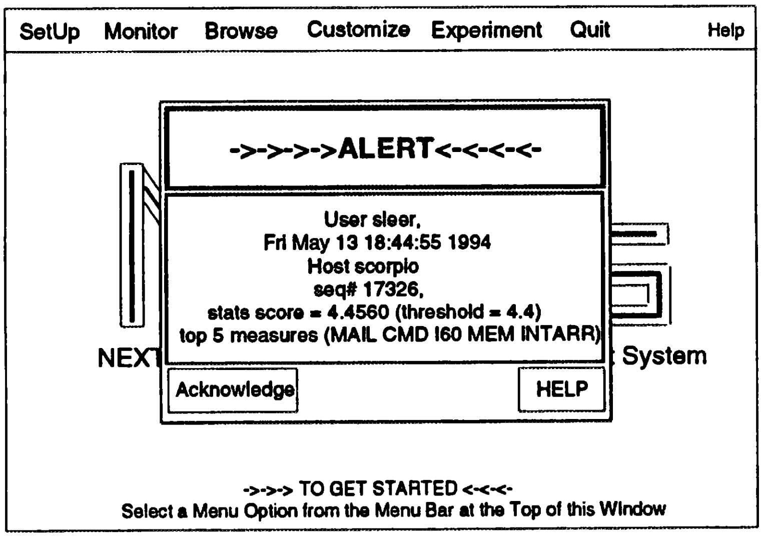
Alert Received!!!
But, NO alert mechanism has
been turned ON.

If you want to be notified when
an alert occurs please turn ON
at least one type of alert mechanism.

Use the Alert menu on the main
NIDES SOUI window menu bar.

When you are done reading this
message please select the Close button.

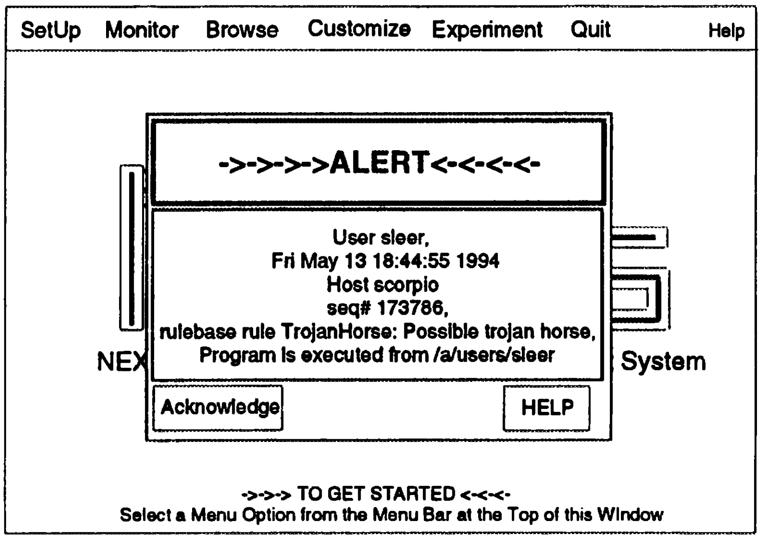
Figure 6.62: Alert Received Warning Window



User sleer,
Fri May 13 18:44:55 1994
Host scorpio
seq# 17326,
stats score = 4.4560 (threshold = 4.4)
top 5 measures (MAIL CMD I60 MEM INTARR)

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.63: NIDES Alert Window — Statistics



->->->->ALERT<-<-<-<-

User sleer,
Fri May 13 18:44:55 1994
Host scorpio
seq# 173786,
rulebase rule TrojanHorse: Possible trojan horse,
Program is executed from /a/users/sleer

->->-> TO GET STARTED <-<-<-
Select a Menu Option from the Menu Bar at the Top of this Window

Figure 6.64: NIDES Alert Window — Rulebase

6.10.2 E-mail Alert Message

When the E-mail alert mechanism is switched ON, an e-mail message containing the alert information is sent to the mailing list of selected users. The e-mail message contains the same information as the alert windows shown in Figures 6.64 and 6.63.

6.11 Error Messages

The NIDES user interface reports errors that occur during NIDES execution through popup error windows. There is a one-sentence error message at the top of the window, followed by a brief error description. When you have finished reading the error message, select *Close* to return operation to the NIDES user interface.

Below is an exhaustive list of all error conditions reported by NIDES organized by NIDES functionality, and alphabetized by their one-sentence messages.

Most of the error messages displayed on the screen have enough detail to describe the conditions that led to the error, and offer possible remedies. If you cannot solve the problem described by the message, consult your local system administrator.

6.11.1 NIDES Startup Errors

NIDES startup errors are invoked when there is a problem with the installation of NIDES, improperly configured NIDES environment variables, or privileged usage problems. These error messages will appear in the shell window from which you have invoked the NIDES user interface (i.e., an error message window will not pop up).

NIDES Startup ERROR

The NIDES `ipc_nameserver` process must be installed and running on your machine before you can start any NIDES processes. Check with your NIDES system administrator to make sure that the `ipc_nameserver` is running.

Non-privileged user, limited capability
Your account has not been included in the privileged users file, and hence you will have limited NIDES capability. See Section 6.2 on page 103 for details on what functionalities are available for non-privileged users.

Problems reading default target host file
This is a warning message only. Apparently there were problems reading the target host file that was created from a previous NIDES session, possibly due to permission problems. The default target host list will be empty, and you must enter in all/any configurable target hosts via the SetUp Menu Target Hosts option in the Main Menu.

Problems reading privileged user list
There were problems with the privileged user list that was installed by your NIDES system administrator, possibly due to file or directory permissions. Hence, you will be allowed only limited use of the NIDES system. See Section 6.2 on page 103 for details on what functionalities are available for non-privileged users.

This user is unknown! NIDES aborting...
The account you are using is not a valid user account for the host on which you are running NIDES. You should report this problem to your local system administrator.

Your IDES_ROOT environment variable is not set.
The IDES_ROOT environment variable must be set in order to start NIDES processes. To set the environment variable, type "`setenv IDES_ROOT` *pathname*" at the UNIX prompt in the window from which you invoke NIDES. *Pathname* is the full path name of the directory where NIDES is installed.

6.11.2 NIDES Server Errors

These errors pertain to the NIDES arpool and analysis components.

NIDES analysis has already been started
You have attempted to start the NIDES Analysis when it is already running. Generally, you should not see this error message, since the Analysis Start/Stop options are automatically set according to the Analysis process state. If you do see this message, there may have been a synchronization problem between the NIDES agent processes and the user interface. This should be reported to your local NIDES system administrator.

NIDES servers are NOT running
There was either a problem starting up the NIDES Arpool/Analysis components, or one of these processes just went down unexpectedly. If there was a problem starting them up, check with your NIDES system administrator to make sure NIDES has been installed properly. If the processes went down for some reason, wait a couple of minutes and try to restart NIDES again by using the SetUp Menu Analysis Option in the Main Menu. If this does not work after a few attempts, consult your NIDES system administrator.

NIDES servers could NOT be stopped

There was a problem stopping the NIDES Arpool/Analysis components. If you want to terminate your NIDES session, select the Quit option from the main menu, and then check the system status to make sure that there are not any NIDES processes still active; if they exist, you should manually kill these with the UNIX kill command before you start another NIDES session.

No NIDES to STOP

You have attempted to stop the NIDES Analysis when there is no analysis running. Generally, you should not see this error message, since the Analysis Start/Stop options are automatically set according to the Analysis process state. If you do see this message, there may have been a synchronization problem between the NIDES agent processes and the user interface.

6.11.3 NIDES Archiver Errors

These errors pertain to the NIDES archiver component.

Archiver is already ON/OFF

You have attempted to either start the archiver process when one is already running, or to turn it off when there is no NIDES archiver process running. Generally, you should not see either version of this message, since the Archiver Start/Stop options are automatically set according to the archiver process state. If you do see this message, there may have been a synchronization problem between the NIDES agent processes and the user interface.

Can't start archiver (no arpool)

The archiver cannot run without the Arpool process. Generally, you should not see this error message, since the Archiver options are automatically turned OFF if the Arpool has not been started. If you do see this message, there may have been a synchronization problem between the NIDES agent processes and the user interface.

NIDES archiver went down

An error occurred while the NIDES archiver was saving audit data to archive storage area, and hence the archiver was automatically turned off. Archiver problems may result from permission problems in the archive area, or from the lack of disk space for storing audit records.

6.11.4 Target Host Errors

These error conditions may result from improperly configuring target hosts.

Can't display target hosts

This is a NIDES internal error (see Section 6.11.12 on page 207).

Duplicate target name entered

You have attempted to add a target host that is already in the list of NIDES configurable hosts. Make sure you have typed in the desired name correctly. It is also possible that the host you have entered is actually an alias for one of the hosts already in the list.

Entered host name not found: hostname
The target host name you have entered is not a valid host name according to your machine's official host list. Make sure you have typed in the correct name, and that such a host really exists on your network.

No host name specified
Apparently you did not enter any name in the name entry window provided, or the name entered has some invalid characters. Retype the host name you want to add.

Problems saving target host list to persistent storage
There was a problem writing out the target host list to persistent storage. It is possible that the permissions of the directory (`$IDES_ROOT/etc`) are not set properly, or that the existing target host file is not overwritable.

Problems with internal target list
This is a NIDES internal error (see Section 6.11.12 on page 207).

Target Host not started: hostname
The auditing on the specified host name could not be started for some reason, and as a result the configuration of this target host has been automatically turned to OFF. Verify that your system is installed properly, and then try to restart auditing on this host via the SetUp Menu Target Host option in the Main Menu.

Target Host not stopped: hostname
Auditing on the specified target host could not be stopped. It is possible that the `agen` process on this host went down without reporting its status to the user interface. If this happens consistently with this particular host, make sure that NIDES was installed properly. If auditing was never turned on for this host, it is possible that audit data is/was being sent to NIDES by some other means (i.e., not `agen`). This *may* be considered suspicious activity. Report this to your local NIDES system administrator.

The following target host(s) has (have) gone down
This error message will list one or more target hosts that have gone down unexpectedly (e.g., system crash, disconnected from the network). The configuration for these hosts will automatically be turned OFF, and you should see the target status as DOWN in the Target Status window. Wait a few minutes before turning auditing back ON for the host. You can restart auditing on this host via the SetUp Menu Target Hosts option in the Main Menu.

The following target host(s) was (were) not found
If you see this error message, the user interface has received status of one or more target hosts that were not originally configured by NIDES, nor are they recognizable by your machine's official host list. This *may* be considered suspicious activity. Report this to your NIDES system administrator.

6.11.5 Alert Configuration Errors

Duplicate recipient added
You have attempted to add a recipient who is already in the list of NIDES configurable

recipients. Make sure you have typed in the desired name correctly, and that you have entered the recipient's full net address (e.g., root@myhost.mydept.com).

No alert mechanism turned on
When NIDES initially starts up, all alert mechanisms are switched to OFF. If the user has not selected an alert mechanism when the first alert is received, a warning message is displayed as shown in Figure 6.62. This message is displayed only after the first alert is received. If no alert mechanisms are turned ON and another alert is received, the warning message will *not* be displayed again (unless the alert mechanism had been turned ON and back OFF again).

No recipient name specified
Apparently you did not enter any name in the name entry window provided, or the name you entered had some invalid characters. Retype the recipient's name.

Problems getting e-mail list
There were problems retrieving the list of e-mail alert recipients from persistent storage, possibly related to permissions for an e-mail recipient file that was created from a previous NIDES session. As a result, the default e-mail list will be empty, and you must enter in all/any recipients via the SetUp Menu Alert Config option in the Main Menu.

Problems saving recipient list to persistent storage
There was a problem writing out the e-mail recipient list to persistent storage. It is possible that the permissions of the directory (`$IDES_ROOT/etc`) are not set properly, or that the existing recipient file cannot be overwritten. Check the permissions of both the directory and file itself.

Problems with internal e-mail list
This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.6 Alert Filter Errors

Problems creating alert filter list.
There were problems creating the internal list of alert filters, possibly due to memory allocation.

Problems saving alert filter list to persistent storage.
There is a problem writing out the alert filter list to persistent storage. It is possible that the permissions of the directory (`$IDES_ROOT/etc`) are not set properly, or that the existing alert filter file cannot be overwritten. Check the permissions of both the directory and file itself.

No subject name specified.
Apparently you did not enter any name in the name entry window provided, or the name you entered had some invalid characters. Retype the subject's name.

Duplicate name added.
You have attempted to add a subject who is already in the alert filter list. Make sure you have typed in the desired name correctly.

6.11.7 Browse Audit Data Errors

Can't get list of archives
There is a problem retrieving the list of NIDES audit data archives. It is possible that you do not have read permission on the archive directory (`$IDES_ROOT/storage/dmf`), or that there are no archives in the archive directory.

Illegal time stamp format
The timestamp(s) you have entered is (are) improperly formatted. Timestamp entries must be in the format MM/DD/YY hh:mm:ss. MM=month(1-12), DD=day(1-31), YY=year(00-99), hh=hour(0-23), mm=minutes(0-59), and ss=seconds(0-59).

Illegal time stamp value
You have entered in an illegal timestamp value. The starting and ending timestamps must be within the default time range of the selected archive. In addition, the ending timestamp must be later than the starting timestamp. Make sure that your timestamp entries fall within these two restrictions. If you do not remember the default time range, re-select the archive name and the default time ranges will reappear in the timestamp boxes.

No file name specified
Apparently you did not enter any file name in the entry window provided, or the file name you entered had some invalid characters. Retype the file name.

No subjects selected
There must be at least one subject selected from the Available Subjects list in order to retrieve any records from the archive. Select relevant subjects from the Available Subjects list.

Problems accessing audit data archive
There was a problem obtaining a handle for the selected archive. This is an internal error, which should be reported to your system administrator.

Problems accessing temporary file
There was a problem reading the temporary file that was created as an intermediate step to writing out the displayed records to a UNIX file. It is possible that this temporary file has been corrupted. Try specifying a different file name, and if the problem persists, notify your NIDES system administrator.

Problems creating output file
There was a problem creating the output file specified by you. It is possible that you do not have write permission in the current directory from which you have invoked NIDES, or that a file already exists by the same name and cannot be overwritten by you.

Problems displaying list of subjects
This is a NIDES internal error (see Section 6.11.12 on page 207)

Problems getting archive name from list
This is a NIDES internal error (see Section 6.11.12 on page 207)

Problems getting audit data from archive
There was a problem selecting the data from the selected archive. This is an internal error. It is possible that the archived data is corrupted or improperly formatted.

Problems opening temporary file
There was a problem creating the temporary file that was to exist as an intermediate step to writing out records to a UNIX file. It is possible that you do not have write permission in the current directory from which NIDES is invoked, or that there already exists a temporary file which cannot be overwritten. Check the permissions on the file and directory.

Problems reading records from temp file to display
This is a NIDES internal error (see Section 6.11.12 on page 207)

Problems with internal subject list
This is a NIDES internal error (see Section 6.11.12 on page 207)

Problems with the audit data index file
There were problems with the index file associated with the selected archive. An archive cannot be examined without a valid index file. It is possible that the index file does not exist, is corrupted, or is unreadable by you. Check the file and directory permissions. If you have selected the real-time archive, it is possible that the archiver has not yet been activated, and hence there is no audit data index file at this time (and hence no audit data to browse).

Too many records to select (MAX=value)
The NIDES user interface imposes a limit on the number of records that can be retrieved from an archive to prevent the process size from getting too large. The selection criteria you have specified may cause this maximum value to be exceeded. You can decrease this potential value by reducing the number of subjects to be selected, and/or decreasing the selected time range to a narrower time window.

6.11.8 Browse Results Errors

The following messages are displayed while you use either the Browse Live Results or Browse Test Results option.

Can t display list of subjects
This is a NIDES internal error (see Section 6.11.12 on page 207)

Can t get list of subjects
There is a problem getting the list of subjects from the archive. It is possible that you do not have permission to read the results directory for the selected test. It is also possible that there are no results archived for this test; if you have requested to archive only warning and/or critical level results, there may not be any results at the requested level.

Can t get list of test names
There is a problem retrieving the list of NIDES result archives. It is possible that you do not have read permission on the archive directory `(IDES_ROOT/storage/dmf),` or that there are no result archives in the archive directory. Check the archive directory contents and permissions.

Can't write out results

There was a problem reading the temporary file that was created as an intermediate step to writing out the displayed records to a UNIX file. It is possible that this temporary file has been corrupted. Try specifying a different file name, and if the problem persists, notify your NIDES system administrator.

No subjects selected

There must be at least one subject selected from the Available Subjects list in order to retrieve any records from the test archive. Select relevant subjects from the Available Subjects list.

Illegal time stamp format

The timestamp(s) you have entered is (are) improperly formatted. Timestamp entries must be in the format MM/DD/YY hh:mm:ss.

Illegal time stamp value

The ending timestamp must be later than the starting timestamp. Make sure that your timestamp entries fall within these two restrictions. If you do not remember the default time range, reselect the test name and the default time ranges will reappear in the timestamp boxes.

No file name specified

Apparently you did not enter any file name in the entry window provided, or the file name you entered had some invalid characters. Retype the file name.

Problems accessing results database

There was a problem obtaining a handle for the selected test archive. This is an internal error (see Section 6.11.12 on page 207)

Problems creating result output file

There was a problem creating the output file specified by you. It is possible that you do not have write permission in the current directory from which you have invoked NIDES, or that a file already exists by the same name and cannot be overwritten by you. Check the directory permissions and files.

Problems getting results from database

There was a problem selecting the data from the selected test archive. This is an internal error. It is possible that the archived data is corrupted or improperly formatted. You can try using another archive or check the archive directory and files.

Problems opening temporary file

There was a problem creating the temporary file that needs to exist as an intermediate step to writing out records to a UNIX file. It is possible that you do not have write permission in the current directory from which NIDES is invoked, or that there already exists a temporary file which cannot be overwritten. Check the directory permissions and files.

Problems reading results from temp file to display

This is a NIDES internal error (see Section 6.11.12 on page 207)

Problems with getting test name from list
: This is a NIDES internal error (see Section 6.11.12 on page 207)

Problems with internal subject list
: This is a NIDES internal error (see Section 6.11.12 on page 207)

Problems with the result index file
: There were problems obtaining the result index file for the selected test. Check the permissions on the test results directory. If you are trying to view real-time results, it is possible that no audit data has been processed yet (and hence no results to view). If you are trying to view test results, it is possible that the test has not yet completed (the index file is not created until the test has finished).

Time stamp value out of range
: You have entered in an illegal timestamp value. The starting and ending timestamps must be within the default time range of the selected test. If you do not remember the default time range, reselect the test name and the default time ranges will reappear in the timestamp boxes.

Too many records to select (MAX=value)
: The NIDES user interface imposes a limit on the number of records that can be retrieved to avoid the process size from getting too large. The selection criteria you have specified may cause this maximum value to be exceeded. You can decrease this potential value by reducing the number of subjects to be selected, and/or decreasing the selected time range to a narrower window.

6.11.9 Instance Configuration/Browse Errors

The following error conditions might occur while you are configuring or browsing NIDES instances. Error conditions for a particular instance configuration or browse window are listed separately. Differences between real-time and test instances are noted.

Can t get general instance configuration
: There were problems retrieving general configuration information from persistent storage for the selected instance. It is possible that you do not have read permission for the *config* file, or that this file does not exist or is corrupted. Make sure that the instance directory itself is readable by you.

Can t get rule base configuration
: There is a problem reading the rulebase configuration data from persistent storage for the selected instance. It is possible that you do not have read permission for the *kb* or *rb_config* file, or that these files are out of date or corrupted. Make sure that the instance directory itself is readable by you.

Can t get stat configuration
: There is a problem reading the stats configuration data from persistent storage for the selected instance. It is possible that you do not have read permission for the *stats_config* file, or that

this file does not exist or is corrupted. Make sure that the instance directory itself is readable by you.

Can't write out instance reconfig
There were problems writing out the general configuration information to persistent storage for the selected instance. It is possible that you do not have write permission for the *config* file. Make sure that the instance directory itself is writable by you.

Can't write out rulebase configuration
There is a problem writing the list of rules to persistent storage for the selected instance. It is possible that you do not have write permission for the *kb* or *rb_config* files. Make sure that the instance directory itself is writable by you.

Can't write out stat configuration
There is a problem writing the stats configuration data to persistent storage for the selected instance. It is possible that you do not have write permission for the *stats_config* file. Make sure that the instance directory itself is readable by you.

Illegal rulebase action code for (rulename)
This is a NIDES internal error (see Section 6.11.12 on page 207).

Invalid instance name
This is a NIDES internal error (see Section 6.11.12 on page 207).

No new reconfiguration
You have requested display of summary reconfiguration data for the selected instance, but there isn't any. All pending reconfiguration data is applied to the actual instance configuration at profile update time, so it is possible that any recent reconfiguration you have made has already been applied.

Problems activating rule `rulename`
There was a problem configuring this rule to be ON. This is a rulebase internal error (see Section 6.11.12 on page 207).

Problems deactivating rule `rulename`
There was a problem configuring this rule to be OFF. This is a rulebase internal error (see Section 6.11.12 on page 207).

6.11.9.1 Instance View Errors

Can't get general instance configuration
There were problems retrieving general configuration information from persistent storage for the selected instance. It is possible that you do not have read permission for the *config* file, or that this file does not exist or is corrupted. Make sure that the instance directory itself is readable by you.

Can't get instance names
There were problems getting the list of available instances from persistent storage. It is

possible that there are no instances to view (including the real-time instance), or that the instances are unreadable by you. Check the permissions on the persistent storage directory.

Can't get instance selection
This is a NIDES internal error (see Section 6.11.12 on page 207).

Can't get reconfig data
There were problems reading in the reconfig structure for this instance. It is possible that the *re-config* file is unreadable by you, or that it is corrupted. Make sure that the instance directory itself is readable by you.

Can't get stat configuration
There were problems reading in the stats configuration file. It is possible that there is no *stats_config* file for this instance, or that it is corrupted or unreadable by you. Make sure that the instance directory itself is readable by you.

Can't open file
There were problems opening or writing to the specified file. Make sure that this file name is available and writable by you, and that you also have write permission in the current directory from which you have invoked NIDES.

Can't read in rules
There were problems reading in the rulebase configuration for this instance. It is possible that there is no rulebase config file for this instance, or that it is corrupted or unreadable by you. Make sure that the instance directory itself is readable by you.

No file specified
Apparently you did not enter any file name in the entry window provided, or that the file name you entered had some invalid characters. Retype the file name.

6.11.9.2 Instance Management Errors

Note: The Instance Management window applies only to test instances.

No instance name specified
Apparently you did not enter an instance in the name entry window provided, or the instance name you entered had some invalid characters. Retype the instance name.

No instance name selected
You need to specify which instance you want to copy. Select an instance from the Instance list, and then enter a new (different) name for the instance copy.

Duplicate instance entry
The instance name you have specified already exists. Enter a different name for this new instance. If you insist on using this name for the new instance, you must first delete the existing instance (associated results are automatically deleted).

Instance in use

The instance name you have specified is currently being used in a NIDES test experiment, and should not be tampered with until the test is finished. Wait until the active test has completed. You can view actively running tests via the Experiment Menu Status and Results option in the Main Menu.

Problems deleting test results

There were problems deleting the test results associated with the selected instance. Whenever you delete an instance, any test results processed using this instance will also be removed. It is possible that the test result directory is not writable by you, or there were some result records that were not removed for some reason. As a result, the test instance will not be deleted. Check the permissions of the test results directory and the instance directory.

Problems deleting instance

There were problems deleting the instance you have selected. It is possible that you do not have permission to remove this instance or any of its profiles, or that this instance does not actually exist. *Note: if you see this message, then you should assume that test results associated with this instance (if they exist) have been successfully deleted.*

Test instance not created

There were problems writing out the instance to persistent storage. It is possible that you do not have write permission to the instance directory. Check the permissions on the storage/instance directory under IDES_ROOT.

Could not create live instance

If there was no real-time instance, then NIDES automatically tries to create one. If you see this error message, then there were problems creating the real-time instance (probably due to permission or space problems), and you should report this immediately to your local NIDES system administrator.

No file specified

Apparently you did not enter a file name in the entry window provided, or the name you entered had some invalid characters. Retype the file name.

Can't open file

There were problems opening or writing to the specified file. Make sure that this file name is available and writ able by you, and that you also have write permission in the current directory from which you have invoked NIDES.

Can't get general instance configuration

There were problems retrieving general configuration information from persistent storage for the selected instance. It is possible that you do not have read permission for the *config* file, or that this file does not exist or is corrupted. Make sure that the instance directory itself is readable by you.

Can't get stat configuration

There is a problem reading the stats configuration data from persistent storage for the selected instance. It is possible that you do not have read permission for the *stats_config file,* or that this file does not exist or is corrupted. Make sure that the instance directory itself is readable by you.

Can't get rulebase configuration

There is a problem reading the rulebase configuration data from persistent storage for the selected instance. It is possible that you do not have read permission for the *kb* or *rb_config* file, or that these files are out-of-date or corrupted. Make sure that the instance directory itself is readable by you.

6.11.9.3 Measure Configuration Errors

The following error conditions may occur while you are working with the Statistical Measures Configuration window.

Illegal value entered

This error message is displayed if an invalid value was entered for any of the measure-specific parameters. The following constraints are imposed:

QMAX values.

Qmax must be a positive value between the indicated ranges.

Scalar values.

A Scalar value must be a positive value between the indicated ranges. Due to precision factors, if the Scalar value is high (say above l,000,000), NIDES does some rounding-off. For example, if you enter values between l,000,000 and 1,000,004, the computer will round off this value to 1,000,000 (and thus will not trigger the error message), and if you enter values between 1,000,005 and 1,000,009, the value will be rounded off to 1,000,001, which would trigger the error message.

Minimum effective-N.

Minimum Effective-N must be a positive value no larger than the indicated limit.

Short-term Half-life.

Short-term half-life must be a positive value no larger than the indicated limit.

6.11.9.4 Classes Configuration Errors

The following error conditions may occur while you are working with the Statistical Classes Configuration window.

Can't drop class item

There were problems trying to remove an item from the list of command classes. This is an internal NIDES error (see Section 6.11.12 on page 207).

Entered host name not found: hostname

The target host name you have entered is not a valid host name according to your machine's official host list. Make sure that you have typed in the correct name, and that such a host really exists on your network.

Invalid instance name
: This is a NIDES internal error (see Section 6.11.12 on page 207).

No item name specified
: Apparently you did not enter any item name in the entry window provided, or the name you entered had some invalid characters. Retype the item name.

Problems getting stat classes
: There was a problem obtaining the list of statistical classes from the statistical configuration structure. It is possible that this structure has some invalid data.

Problems with internal item list
: This is a NIDES internal error (see Section 6.11.12 on page 207).

Too many items to manage (MAX=value)
: This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.9.5 Parameters Configuration Errors

The following error conditions may occur while you are working with the Statistical Parameters Configuration window.

Cache size is smaller than original
: This message is a warning only. Contrary to intuition, specifying a smaller cache size than the previously configured size does not decrease the NIDES overall process size. It may, however, prevent the process size from increasing beyond its current size.

Illegal cache value
: The profile cache size must be at least 1 and no greater than the specified maximum.

Illegal long-term half-life value
: The long-term profile half-life value must be between 1 and 365 days (one year).

Illegal max sum rare prob value
: This number should be a valid decimal number within the specified range.

Illegal red threshold value
: The red (critical) threshold percentage value must be less than the yellow (warning) threshold percentage.

Illegal threshold value
: The threshold value is a percentage value, and must be between 0.001% and 100%. Note that you can specify fractional percentages, but it cannot be zero.

Illegal training days value
: You have specified an invalid value for the number of profile training days. You must specify a minimum of 3 days and a maximum of 365 days.

Illegal yellow threshold value
The yellow (warning) threshold percentage value must be greater than the red (critical) threshold percentage.

Invalid instance name
This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.9.6 Profile Management Errors

The following error conditions may occur while you are working with the Profile Management window. Note that there are two profiles per subject: a current (short-term) profile and a historical (long-term)profile; the error messages listed below are displayed if there are problems with either one.

Duplicate profile
You have attempted to make a copy of a profile, but there is already a profile for the specified subject. Use the Replace option to rename a profile, or specify a new subject name for the profile copy.

Illegal character in entry
You have entered an illegal character in the name entry box. Subject names can contain only alphanumerics, the hyphen (-) and the underscore (_) characters. Make sure you enter the subject name properly.

Illegal restore request
You have attempted to restore either a newly created profile or a profile that is already in its original state. The Restore option is to be used only to 'undo' a previous Replace operation. Use the Delete option to 'undo' a copy or to remove a profile.

No profile name entered
Apparently you did not enter any subject name in the entry window provided, or the name you entered had some invalid characters. Retype the subject name.

No profiles available in *this instance*
There were no profiles found for this instance, and hence there is nothing to manage in this window. Make sure that audit records have been processed for this instance.

No such subject found
You have attempted to replace this profile with a non-existent profile. Make sure that you enter a subject that is listed in the display.

Problems copying profile
There was a problem copying the selected profile for the subject you have specified. This may have something to do with the permissions of the profiles or the profile directory for this instance (must be writable by you). Check the permissions on the profile directory for your selected instance.

Problems deleting profile
: There was a problem deleting this profile from persistent storage. It is possible that you do not have permission to delete the profile, or that the instance directory is not modifiable by you. Check the permissions on the instance and the profile directories.

Problems getting list of profiles
: There was a problem obtaining the list of subject profiles for this instance. It is possible that you do not have permission to read the profile directory for this instance. It is also possible that there are too many profiles for the user interface to handle.

Problems reading profile from persistent storage
: There was a problem reading this profile from persistent storage. It is possible that you do not have permission to read the profile, or that the instance directory is not readable by you. It is also possible that this profile does not exist or is corrupted.

Problems replacing profile
: There was a problem replacing the selected profile with the subject you have specified. This may have something to do with the permissions of the profiles or the profile directory for this instance (must be writable by you). Check the permissions on the instance and profile directories and on the profile files.

Problems restoring profile
: There was a problem restoring the selected profile to its original state. This may have something to do with the permissions of the profiles or the profile directory for this instance (must be writable by you). Check the permissions on the instance and profile directories and on the profile files.

Problems with internal subject list
: This is a NIDES internal error (see Section 6.11.12 on page 207).

Replacing identical profiles.
: You have attempted to replace a profile with the same identical profile. Select another (different) profile.

Too many profiles to handle (MAX=value)
: This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.9.7 Updater Configuration Errors

The following error conditions may occur while you are working with the Profile Updater Configuration window. Note: This window applies to the real-time instance only.

Illegal timestamp format
: The timestamp you have entered is improperly formatted. The NIDES profile updater is on a 24-hour clock cycle, and hence must be in the format hh:mm:ss.

Invalid instance name
: This is a NIDES internal error (see Section 6.11.12 on page 207).

No profiles in this instance
: There are no profiles defined for this instance, and hence there is nothing to manage. You must select an instance with at least one profile for this option to be available. It is possible that no audit records have been processed yet for this instance.

Problems getting list of subjects for instance
: There was a problem obtaining the list of subjects for the selected instance. It is possible that you do not have permission to read this instance or any of the profiles. Check the permissions on the instance and profile directories.

Problems with internal subject list.
: This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.9.8 Manual Update Errors

The following error conditions may occur while you are working with the Trigger Profile Updater window. *This window applies to the real-time instance only.*

Can t get list of subjects
: There were problems retrieving the list of subject profiles for this instance. It is possible that there are no profiles for this instance (you need at least one profile to work with in this window), or that the profile directory or the profiles themselves are not readable by you. Check the permissions on the profile and instance directories.

Illegal instance specified
: This is a NIDES internal error (see Section 6.11.12 on page 207).

Problems with internal subject list
: This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.9.9 Profile Update Switch

The following error conditions may occur while you are working with the Profile Update Switch window. Note: This window applies only to test instances.

Invalid instance name
: This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.9.10 Instance Remarks Configuration Errors

Can t get general instance configuration
: There were problems retrieving general configuration information for the selected instance. It is possible that you do not have read permission for the *config* file, or that this file does not exist or is corrupted. Make sure that the instance directory itself is readable by you.

Problems writing instance config
: There were problems writing out the general configuration information to persistent storage for the selected instance. It is possible that you do not have write permission for the *config* file. Make sure that the instance directory itself is writable by you.

6.11.9.11 Instance Reconfiguration Errors

The following error conditions may occur while you are working with the Instance Reconfiguration window.

No pending reconfiguration
: If you see this error message, it means that there are no pending reconfigurations for the selected instance. If you are examining the real-time instance, all (if any) pending reconfigurations (including the deferred ones) have already been applied to the analysis components during the most recent profile update. If you are examining a test instance, all (if any) pending reconfigurations have already been applied when this instance was last used for a test run.

Can't create temp file
: There was a problem creating the temporary file that exists as an intermediate step to displaying information in the window. It is possible that you do not have write permission in the current directory from which NIDES is invoked, or that there already exists a temporary file that cannot be overwritten. Check the permissions in the directory where NIDES was invoked.

Problems reading records from temp file to display
: This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.9.12 Result Filter Errors

The following error conditions may occur while you are working with the Result Filter window.

Invalid instance name
: This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.9.13 Rulebase Configuration Errors

The following error conditions may occur while you are working with the Rulebase Configuration window.

Can't find mandatory rules
: There was a problem reading in the list of mandatory rules. It is possible that this list does not exist, or that you do not have permission to read the file containing this list. Check the permissions on the IDES_ROOT/etc directory and the mandatory_rules file.

Can't find rules
: There is a problem getting the complete list of available rules. It is possible that file containing this list is unreadable by you, or that the file has been corrupted in some way. Check the permissions on the rulebase directory.

Can't obtain rule configuration
: There were problems retrieving the list of activated rules for this instance. This is an internal rulebase error (see Section 6.11.12 on page 207).

Can t read in rules
There were problems reading in the knowledge base for this instance. This is an internal rulebase error (see Section 6.11.12 on page 207).

Problems with internal rule list
This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.10 Audit Data Set Errors

The following error conditions may occur while you are working with the Audit Data Set Management window or the Create Audit Data window.

Audit data set name is *NULL*
This is a NIDES internal error (see Section 6.11.12 on page 207).

Can t get index info for (database)
There were problems with the index file for the audit data archive you have selected. It is possible that this index file does not exist, is unreadable by you, or is corrupted. If you have selected the real-time archive, it is possible that there is no audit data available yet (i.e., no records have been processed) by the analysis components. If this is the case, you will need to wait until NIDES processes some audit data.

Can t get list of users for live audit data
This is a NIDES internal error (see Section 6.11.12 on page 207).

Can t load in database names
There was a problem retrieving the list of audit data archives. Make sure that you have read permission for the archive directory.

Could not create audit data set
There was a problem with the process that is invoked to create actual audit data sets. It is possible that the audit data set directory does not have the correct permissions set, or that the audit data archive is unreadable by you. Check the permissions on the audit data archive directories.

Duplicate name entered
You have specified a data set name that already exists. Either enter a different name, or delete the existing data set first and then reuse this name for the new data set.

Exceeded maximum list capacity (MAX=value)
This is a NIDES internal error (see Section 6.11.12 on page 207).

Illegal time stamp format
The timestamp(s) you have entered is (are) improperly formatted. Timestamp entries must be in the format MM/DD/YY hh:mm:ss.

Illegal time stamp value
You have entered an illegal timestamp value. The starting and ending timestamps must be

within the default time range of the selected archive. In addition, the ending timestamp must be later than the starting timestamp. Make sure that your timestamp entries fall within these two restrictions. If you do not remember the default time range, reselect the archive name and the default time ranges will reappear in the timestamp boxes.

Invalid list specified
This is a NIDES internal error (see Section 6.11.12 on page 207).

No audit data set name specified
Apparently you did not enter a name in the entry window provided, or the name you entered had some invalid characters. Retype the name of the data set.

No records selected
There were no audit records found with the selection criteria you have specified, and hence the audit data set could not be created. You may want to reexamine the contents of the archive via the Browse Audit Data option to redefine the selection criteria.

No subjects found for this audit data set
There were no subjects found in the audit data set you have selected. The corresponding index file may be invalid.

No subjects to filter
You must specify at least one subject in the filter list in order to create a new audit data set. Choose one or more subjects from the Available List.

Problems accessing archive *database*
There were problems obtaining a proper list handle for the archive specified. It is possible that the archive may have not been found or that the directory or archive itself does not have the proper permissions set. Check the permission on the archive directory.

Problems creating index file for dataset
There were problems creating the index file for the audit data set you have specified. It is possible that you do not have permission to write into this file or to the index directory. Check the permissions on the index directory.

Problems deleting data set
There were problems deleting the audit data set. It is possible that you do not have write permission for this data set or perhaps for the data set directory itself. Check the permissions on the audit data set directory and make sure that the data set has not already been removed.

Problems displaying audit data sets list
There were problems retrieving the list of audit data sets from the data archive. It is possible that you do not have read permission for the directory containing the index files for the audit data sets. Check the permissions on the index directory for audit data sets.

Problems getting selected records
There were problems selecting the records from the archive. It is possible that some or all of the audit records are unreadable by you or may even be invalid. Check the permissions on the audit data archive directory.

Problems retrieving records into file

There was a problem writing the audit data out to a file. Make sure that you have the proper permission to write into the IDES_ROOT/storage/adsets directory. Otherwise, this error may be due to an internal NIDES problem (see Section 6.11.12 on page 207).

Problems with audit index file

There were problems with the index file for the audit data set you have selected. If this data was created via the NIDES user interface, then it is possible that this index file does not exist or is corrupted. Make sure that it is readable by you. If this data set was created external to the NIDES user interface, then you must make sure that an appropriate index file has been created for this audit data set (use the *adset_index* utility).

Problems with internal subject list

This is a NIDES internal error (see Section 6.11.12 on page 207).

6.11.11 Test Facility Errors

No instances found

There are no NIDES test instances available to configure a test run with. A test run requires an instance and an audit data set. Instances may be created via the Customize Menu Test Instances option in the Main Menu. Make sure that the instance directory is readable by you.

No audit data sets found

There are no NIDES audit data sets available to configure a test run with. A test run requires an instance and an audit data set. Audit data sets may be created via the Customize Menu Audit Data Set option in the Main Menu. Make sure that the `adset` directory under `IDES_ROOT/storage/adsets` is readable by you.

No instance selected

Apparently you did not select an instance from the Instance List. Select an instance from this list.

No audit data set selected

Apparently you did not select an audit data set from the list. Select a data set from the Audit Data Set list.

Test name already in use

You have selected a test name/instance currently being used in a test run. You must wait until the test run has completed with this instance. Otherwise, you must select a different instance for the test.

No test results

You may only view results of tests that have been completed by the Test Facility. If you see this error message, this means that there are no completed tests to view. You will need to run at least one test before you can use this option.

Problems getting test results
There were problems accessing one or more of the test results. It is possible that you do not have read permission to the test results or the test archive itself. Check the permissions on the test results directory and make sure that the test archive has the proper archive suffix (.res).

Problems getting test indexes
There were problems accessing the index file for at least one of the tests in the test archive, and hence the list of test names could not retrieved. It is possible that the index file is not available, has the wrong permissions set, or is corrupted. Check the permissions on the test index directory.

No test results selected
You must select a test name from the list of completed tests before viewing the test results. Select a test name from the list.

Can't get list of subjects
There were problems getting the list of subjects for the selected test. Make sure that this test does have results available, and that the test result directory and its contents are readable by you.

Problems with the test index file
There were problems accessing the index file for the test you have selected. It is possible that the index file is not available, has the wrong permissions set, or is corrupted. Check the permissions on the test index directory and the files for the selected test.

Problems deleting test results
There were problems deleting the selected test results. It is possible that you do not have write permission in the test result directory, or the test archive itself. Check the permissions on the test result directory and files.

6.11.12 General Error Messages

While best efforts were made to make the NIDES system as robust as possible, it is inevitable that there still exist some software bugs in the system. These are called internal errors. If you see this type of error message displayed on your screen, notify your NIDES system administrator immediately. If this problem is repeatable, provide a detailed report on the steps taken that cause this problem, so that it may be addressed and fixed by the NIDES software developers.

Chapter 7

Utility Programs

This release of NIDES includes 15 programs, described here in *manual page* style, that are useful for everyday NIDES operation:

- **acc2ia** — reads in standard UNIX accounting audit data and writes out NIDES format audit records
- **adset_index**— creates an index file for a NIDES audit data set
- **agen** — reads native audit record data, converts it into NIDES audit records, and delivers these records to the `arpool` program
- **agend** — is normally started at system boot-up by all hosts that are to be monitored by the NIDES system; the `agend` process runs as an RPC server process and listens on a "well-known port" for requests to start and stop the native audit data conversion program, agen
- **apstat** — provides status information on an `arpool` process
- **archiver** — takes NIDES format audit record data and creates an archive that can be reviewed using the NIDES user interface
- **arpool** — runs as an RPC server process that collects NIDES audit records from multiple target host `agen` processes and delivers the collected records to all active RPC client processes
- **audit2ia** — reads in native audit record data and writes out NIDES audit records; the UNIX version of the `audit2ia` process currently supports two native audit data types: Sun OS BSM version 1 and Sun OS C2
- **batch_analysis** — incorporates the real-time intrusion detection modules (statistical and rulebased) to allow NIDES users to run NIDES in "batch" mode to analyze audit data in the same way live data is analyzed
- **iamerge** — reads in NIDES audit records from two sources and merges the records in time-sorted order

- **iapr** — reads NIDES audit records, either from a file or directly from the `arpool` process, and prints out, in ASCII format, specified fields from each record
- **init_priv_user_list** — initializes the list of NIDES privileged users
- **init_stat_config** – reads in an ASCII (human-readable) version of the NIDES statistical component configuration file and converts it into a binary format that is used by the various NIDES components
- **ipc_nameserver** — runs in the background, providing RPC lookup service to all NIDES processes
- **nides** — monitors usage on computers connected via an Ethernet network.

Many of the NIDES programs use the IDES_ROOT environment variable to locate NIDES files and programs and the IPC_NAMESERVER environment variable to communicate with other NIDES processes via RPCs. Setting these two environment variables prior to executing any NIDES programs is recommended. See Section 9.4 for a discussion on setting these values automatically using your `.cshrc` file. You can set them manually by executing the following commands:

```
unix-prompt%setenv  IDES_ROOT  <ides-root-directory>
unix-prompt%setenv  IPC_NAMESERVER  nides-host:port-number
```

The `ides-root-directory` should be replaced with the full path name of your NIDES installation directory. The `nides-host` should be replaced with the name of the host computer that will run NIDES. The `port-number` should be replaced with a free port (we recommend something in the 7000's, 7001 for example). If you need more information on setting these environment variables refer to Section 9.4.

7.1 acc2ia – Converting Accounting Data to NIDES Audit Records

The `acc2ia` program reads in standard UNIX accounting audit data and writes out NIDES audit records. This functionality is useful for generating files of NIDES audit records from accounting audit data collected on remote hosts or archived. Data files created by `acc2ia` can be used for NIDES experiments once you have created an index file for them using the `adset_index` utility (see Section 7.2).

Usage

acc2ia [-i <*input file*>] [-o <*output file*>] [-host <*hostname*>] [-head <*number*>] [-tail <*number*>] [-d]

Flags

--	Causes a brief description of the usage to be printed. No processing occurs.
-i <*input file*>	Indicates the file from which standard UNIX accounting audit data should be read. This data is often stored in the file /usr/adm/pacct. The input file may be any uncompressed file, any compressed file (with a “.Z” suffix), or standard input (indicated by a single dash “-”). This flag is mandatory.
-o <*output file*>	Indicates the file to which NIDES audit records should be written. The output file may be any uncompressed file, any compressed file (with a “.Z” suffix), or standard output (indicated by a single dash “-“). This flag is mandatory.
-host <*hostname*>	Indicates what hostname should be used as the reporting audit host in the NIDES audit records generated. The standard UNIX accounting audit data does not provide this information, so this value must be supplied by the NIDES user. This flag is mandatory.
-head <*number*>	Indicates how many records should be processed starting from the beginning of the input file. Without this flag all records in the file are processed.
-tail <*number*>	Indicates how many records should be processed starting from the end of the input file. Without this flag all records in the file are processed.
-d	Indicates that debugging information should be printed out during normal execution.

Configuration

The `acc2ia` program does not rely on any environment variables or configuration files.

Examples

The following examples are sample usages for the acc2ia utility:

- `acc2ia -i /usr/adm/pacct -o /archive/godzilla.ia.Z -host godzilla`

  Reads from the file `/usr/adm/pacct` and writes the output, in compressed format, to the file `/archive/godzilla.ia.Z;` all NIDES audit records will show the host `godzilla` as the audit host generating the records.

- `acc2ia -i /archive/mothra.pacct.Z -o - -host mothra | iapr -i -`

  Reads from the compressed file `/archive/mothra.pacct.Z` and writes the output to standard output; all NIDES audit records will show the host `mothra` as the audit host generating the records. The output is then read by the `iapr` utility program (see Section 7.11) that prints out an ASCII description of each NIDES audit record.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages are displayed in each case.

- The input file does not exist or cannot be accessed.
- The output file cannot be created.
- The audit host is not specified.

7.2 adset_index– Creating an Index File For Audit Data

The `adset_index` program creates an index file for a NIDES audit data set. This utility is useful if you have an audit data file that you want to use for NIDES experiments — for example, a file generated by `acc2ia` or `audit2ia`.

Audit data sets are UNIX files that contain NIDES-formatted audit records, and are generally compressed (by the UNIX `compress` utility). To be used for NIDES experiments, these files must have an index file.

The *adset_index* utility creates an index for any given audit data set and places it in the appropriate location for NIDES.

Usage

adset_index [-i <*adset-file-name*>] [-p] [-v]

Flags

-- Causes a brief description of the usage to be printed. No audit data set processing will occur.

-i <*adset-file-name*> Provides the name of the audit data set for which the index is to be created. This flag is mandatory. Any existing index file will be overwritten with the new one. If the input file has been compressed with the standard UNIX compression utility, be sure to include the `.Z` suffix. The audit data file must be located in the `IDES_ROOT/storage/adset` directory, otherwise the file will not be processed.

-p <*adset-file-name*> Prints out index information on the specified audit data set. No index file is created.

-v Prints out the status of the adset index creation process. If the input file is large, it may take several minutes to scan the entire audit data file, and hence this flag will allow you to monitor its progress. The display updates every 1000 records.

Configuration

The adset_index program uses the following configuration item when executing.

- IDES_ROOT is used to determine where the audit data file and index should be placed. Specifically, index files for audit data sets will be stored in the directory `IDES_ROOT/storage/adsets/INDEX`.

Examples

The following examples are sample usages for the *adset_index* utility:

- `adset_index -i foo.Z`
  An index file will be created in `IDES_ROOT/storage/adsets/INDEX/foo.Z.audinfo`.

- `adset_index -i godzilla_pacct.Z -v`
  Index file will be created in
  `IDES_ROOT/storage/adsets/INDEX/godzilla_pacct.Z.audinfo`.
  In this example, `adset_index` reports counts of audit records processed while it is generating the index file.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages are displayed in each case.

- The input file does not exist or cannot be accessed.
- The index file cannot be created (perhaps due to problems with write access or the INDEX file directory).
- There were problems with the audit records in the input file.
- The IDES_ROOT environment variable is not set. The program will exit without creating any index files, as it does not know where the audit data file exists.

7.3 agen — Audit Data Conversion and Transmission

The `agen` program reads native audit record data, converts it into NIDES audit records, and delivers these records to the `arpool` program.

The UNIX version of `agen` currently supports three native audit data types: SunOS BSM version 1, SunOS C2, and standard UNIX accounting. When started, the default behavior for `agen` is to process all available types of audit data. Typically, target hosts will have a single or perhaps two (either BSM or C2, and UNIX accounting) audit data sources.

Normally, `agen` is started by `agend` while the NIDES system is monitoring target hosts. Because `agen` examines standard configuration files for each audit system, few flags are required if a NIDES user needs to invoke `agen` directly.

Usage

agen [-c2-only] [-bsm-only] [-acc-only] [-rewind] [-remove] [-replay] [-i <*audit file*>] [-index <*index file*>] [-conf <*audit config file*>] [-nostats] [-mounts] [-nofork] [-nosuid] [-noretry] [-sun3] [-sun4] [-d]

Flags

- -	Causes a brief description of the usage to be printed. No processing will occur.
-c2-only	Indicates that only SunOS C2 audit data should be examined.
-bsm-only	Indicates that only SunOS BSM audit data should be examined.
-acc-only	Indicates that only standard UNIX accounting data should be examined.
-rewind	Indicates that any audit data files encountered should be read from the beginning of the file. This overrides the default behavior of reading only new audit data appended to the audit data files.
-remove	Indicates that any audit data files encountered should be deleted once the file has been entirely processed.
-replay	Indicates that any audit data files encountered should be read from the beginning of the file and that processing should terminate as soon as the file has been entirely processed.
-i <*audit file*>	Indicates the file from which native audit records should be read. Use of this flag implicitly has the effect of deactivating the standard UNIX audit data processing.
-index <*index file*>	Indicates the file from which a list of absolute path names for native audit records should be read. Use of this flag implicitly has the effect of deactivating the standard UNIX audit data processing.

-conf <*audit config file*>	Indicates the file from which audit configuration information should be read. For both BSM and C2, the audit system writes a file that contains the absolute path of the file to which audit data is currently being appended. Use of this flag implicitly has the effect of deactivating the standard UNIX audit data processing.
-nostats	Indicates that audit data generated by the UNIX stat(2) system call should not be reported. This reduces the number of NIDES audit records generated if the native audit system is reporting "read" events (e.g. `ls -l` will generate a stat(2) record for each file found).
-mounts	Indicates that audit data generated by the UNIX mount(2) and umount(2) system call should be reported. This can increase the number of NIDES audit records generated if the native audit system is using the automount facility; file systems are automatically mounted as needed and idle file systems are automatically unmounted.
-nofork	Indicates that `agen` should not go into daemon mode by detaching itself from the controlling TTY by forking. This option is useful only if a user wants to control the program from an interactive shell (e.g., using "Control-C" or "Control-Z") or to run `agen` under the control of a debugger.
-nosuid	Indicates that `agen` should not attempt to change its effective user ID to the `audit` user. The default behavior is to run as the `audit` user.
-noretry	Indicates that `agen` should not try to contact the `arpool` program after any failure.
-sun3	The binary formats of C2 audit data generated on Sun 3s and Sun 4s differ and are not compatible. This flag indicates that the C2 audit data was generated on a Sun 3.
-sun4	The binary formats of C2 audit data generated on Sun 3s and Sun 4s differ and are not compatible. This flag indicates that the C2 audit data was generated on a Sun 4.
-d	Indicates that debugging information should be printed out during normal execution.

Configuration

The `agen` program does not rely on any environment variables or configuration files.

Examples

The following is an example for the `agen` utility:

- `agen -i /usr/zen/my_pacct_data -bsm-only`

  Reads from the bsm format audit data file `/usr/zen/my_pacct_data` and sends it to arpool.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages are displayed in each case.

- The input file does not exist or cannot be accessed.
- The `arpool` process cannot be contacted.

7.4 agend — Audit Data Transmission Daemon

The agend program is normally started at system boot-up by all hosts that are to be monitored by the NIDES system. The agend process runs as an RPC server process and listens on a "well-known port" for requests to start and stop the native audit data conversion program, agen.

The agend process and invocation script are installed as part of the standard NIDES installation procedure, so the NIDES user should not have to deal with agend directly.

NIDES-related changes to the configuration of a target host should not be made by a NIDES user directly. Rather, any configuration changes should be made by the NIDES system administrator.

Usage

agend [nofork] [-d]

Flags

- - Causes a brief description of the usage to be printed. No execution will occur.

-nofork Indicates that the agend process should not go into daemon mode by detaching itself from the controlling TTY by forking. This option is useful only if a user wants to control the process from an interactive shell (e.g. using "Control-C" or "Control-Z") or to run the agend process under the control of a debugger.

-d Indicates that debugging information should be printed out during normal execution.

Configuration

The agend program uses the following configuration item when executing.

- IDES_ROOT is used in determining the location of the native audit data conversion program, agen, which is located in the file $IDES_ROOT/bin/bin.sun4/agen.

Examples

The following examples are sample usages for the agend utility:

- agend

  Starts the agend process in daemon mode. The process will detach from the controlling TTY and will not be accessible for interactive job control.

- agend -nofork

  Starts the agend process without going into daemon mode. The process will not detach from the controlling TTY and will be accessible for interactive job control.

Error Conditions/Return Values

Runtime errors in starting and stopping the agen process are reported directly back to the RPC client that made the request. On startup, errors will occur for the following conditions. Appropriate messages will displayed in each case.

- The IDES_ROOT environment variable is not set.
- The process cannot detach from the controlling TTY.

7.5 apstat — Audit Record Pool (arpool) Status

The `apstat` program makes a single RPC request from a running `arpool` process, and prints out `arpool` status information. This functionality can be useful in examining real-time NIDES audit record flow through the `arpool` process.

The status reported includes the total number of records processed as well as total and hourly counts for producers (i.e., `agen` processes) and status on the consumers (i.e., `exsys_client` and `stats_client` processes).

Usage

apstat [-d]

Flags

- - Causes a brief description of the usage to be printed. No processing will occur.

-d Indicates that debugging information should be printed out during normal execution.

Configuration

The `apstat` program uses the following configuration item when executing.

- IPC_NAMESERVER is used by the `apstat` process to locate the `arpool` process. If the nameserver is not running or the `arpool` process is not registered, the `apstat` process will terminate.

Examples

The following example is a sample usage for the `apstat` utility:

- `apstat`

  Reads status from the `arpool` process (if registered with the nameserver indicated by $IPC_NAMESERVER) and prints the status.

An example of the output from the `apstat` process is

```
audit record status:
    lowater mark 512
    hiwater mark 1024
    stored records 736
    max rseq 0:10512
client status:
    #of agens 3
        on <godzilla.tokyo.com> rec_count 5096, hour_count 5096
        on <zen.liberator.com> rec_count 1894, hour_count 1894
        on <orac.liberator.com> rec_count 3522, hour_count 3522
    #of consumers 2
        on <(null)> rseq 9808
        on <(null)> rseq 9776
```

Error Conditions/Return Values

Errors will occur for the following condition. Appropriate messages are displayed.

- The `arpool` process cannot be contacted.

7.6 archiver — Audit Data Archiving

The `archiver` program takes NIDES format audit record data and creates an archive that can be reviewed using the NIDES user interface. The `archiver` can obtain audit data from two sources: `arpool`, in the case of real-time archival, or an input audit data file. Archival of audit data obtained from `arpool` is normally initiated and terminated through the NIDES user interface.

Usage

archiver [-i <*input file*>] [-a <*archive*>] [-d]

Flags

- -	Causes a brief description of the usage to be printed. No processing will occur.
-i <*input file*>	Indicates the file from which NIDES audit records should be read. The input file may be any uncompressed file, any compressed file (with a ".Z" suffix), or standard input (indicated by a single dash "-"). If this flag is not specified, the `archiver` process will attempt to contact the `arpool` process and retrieve NIDES audit records though RPC calls.
-a <*archive*>	Indicates which archive should be used for storing the NIDES audit records. If no archive is specified, the "real-time" archive is assumed. The "real-time" archive is where audit records processed by NIDES in real-time are normally stored. The `archiver` program would normally obtain the "real-time" archive records from the `arpool` process via RPC calls.
-d	Indicates that debugging information should be printed out during normal execution.

Configuration

The `archiver` program uses the following configuration items when executing.

- IDES_ROOT is used in determining the locations of many files needed for maintaining an archive:
  - $IDES_ROOT/storage/dmf/<*archive*>.aud (NIDES audit records are located in the subdirectories contained in this directory).
  - $IDES_ROOT/storage/dmf/INDEX/<*archive*>.audinfo (Associated archive description files are located here).
- IPC_NAMESERVER is used by the `archiver` process to locate the `arpool` process. If the nameserver is not running or the `arpool` process is not registered, the `archiver` process will terminate unless the `-i` option is used.

Examples

The following examples are sample usages for the `archiver` utility:

- `archiver`

  Reads data from the `arpool` process (if registered with the nameserver indicated by $IPC_NAMESERVER) and writes the NIDES audit records to the "real-time" archive.

- `archiver -i /archive/zen.ia.Z -a zen-test`

  Reads from the compressed file `/archive/Zen.ia.Z` and writes the output to the archive `zen-test`.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages are displayed in each case.

- The input file does not exist or cannot be accessed.
- The archive cannot be created or accessed.
- The `arpool` process cannot be contacted.

7.7 arpool — Audit Record Collection

The `arpool` program runs as an RPC server process and collects NIDES audit records from multiple target host `agen` processes and delivers the collected records to all active RPC client processes.

The `arpool` process is started automatically by the NIDES user interface process when real-time analysis is activated, so the NIDES user should not have to deal with `arpool` directly.

Usage

arpool [-nofork] [-noretry] [-pool <*number*>] [-d]

Flags

--	Causes a brief description of the usage to be printed. No processing will occur.
-nofork	Indicates that `arpool` should not go into daemon mode by detaching itself from the controlling TTY by forking. This option is useful only if a user wants to control the process from an interactive shell (e.g., using "Control-C" or "Control-Z") or to run `arpool` under the control of a debugger.
-noretry	Indicates that `arpool` should not try to contact the nameserver after any failure.
-pool <*number*>	Indicates the maximum number of audit records `arpool` should keep in its pool of active NIDES audit records. The default size of the audit record pool is 1024.
-d	Indicates that debugging information should be printed out during normal execution.

Configuration

The `arpool` program uses the following configuration item when executing.

- IPC_NAMESERVER is used by `arpool` to register its current RPC information. All processes intending to make RPC requests of `arpool` must be able to contact the nameserver.

Examples

The following examples are sample usages for the `arpool` utility:

- `arpool -noretry`

  Starts the `arpool` process in daemon mode. The process will detach from the controlling TTY and will not be accessible for interactive job control. If the `arpool` process cannot contact the nameserver, the process will terminate immediately.

- `arpool -nofork -pool 2048`

  Starts the `arpool` process without going into daemon mode. The process will not detach from the controlling TTY and will be accessible for interactive job control. The maximum audit record pool size will be 2048.

Error Conditions/Return Values

On startup, errors will occur under the following conditions.

- The IPC_NAMESERVER environment variable is not set.
- The ipc_nameserver process is not running. `Arpool` will continue trying to connect to the ipc_nameserver process if it does not respond. This behavior can be disabled with the `-noretry` command line option.

7.8 audit2ia – Converting Audit Data to NIDES Format

The `audit2ia` program reads in native audit record data and writes out NIDES audit records.

The UNIX version of the `audit2ia` process currently supports two native audit data types: SunOS BSM version 1 and SunOS C2; `audit2ia` can process only one type of audit data at a time and the type must be specified.

Usage

audit2ia [-c2] [-bsm] [-host <*hostname*>] [-i <*audit file*>] [-o <*output file*] [-nostats] [-mounts] [-sun3] [-sun4] [-d]

Flags

--	Causes a brief description of the usage to be printed. No processing will occur.
-c2	Indicates that only SunOS C2 audit data should be examined. Either the **-c2** or the **-bsm** flag must be used.
-bsm	Indicates that only SunOS BSM audit data should be examined. Either the **-c2** or the **-bsm** flag must be used.
-i <*audit file*>	Indicates the file from which native audit records should be read. This flag is mandatory.
-o <*output file*>	Indicates the file to which NIDES audit records should be written. The output file may be any uncompressed file, any compressed file (with a ".Z" suffix), or standard output (indicated by a single dash "-"). This flag is mandatory.
-host <*hostname*>	Indicates what hostname should be used as the reporting audit host in the NIDES audit records generated. This flag is mandatory.
-nostat	Indicates that audit data generated by the UNIX stat(2) system call should not be reported. This will reduce the number of NIDES audit records generated if the native audit system is reporting "read" events (e.g., `ls -l` will generate a stat (2) record for each file found).
-mounts	Indicates that audit data generated by the UNIX mount(2) and umount(2) system call should be reported. This can increase the number of NIDES audit records generated if the native audit system is using the automount(8) facility; file systems are automatically mounted as needed and idle file systems are automatically unmounted.

-sun3	The binary formats of C2 data generated on Sun 3s and Sun 4s differ and are not compatible. This flag indicates that the C2 audit data was generated on a Sun 3.
-sun4	The binary formats of C2 data generated on Sun 3s and Sun 4s differ and are not compatible. This flag indicates that the C2 audit data was generated on a Sun 4.
-d	Indicates that debugging information should be printed out during normal execution.

Configuration

The `audit2ia` program does not rely on any environment variables or configuration files.

Examples

The following examples are sample usages for the `audit2ia` utility:

- `audit2ia -bsm -i /usr/audit/file -o /archive/godzilla.ia.Z`

  Reads BSM audit records from the file `/usr/audit/file` and writes the output, in compressed format, to the file `/archive/godzilla.ia.Z`.

- `audit2ia -c2 -i /archive/mothra.c2.Z -o - | iapr -i -`

  Reads from the compressed file `/archive/mothra.c2.Z` and writes to standard output. The output is then read by the `iapr` utility program, which prints out an ASCII description of each NIDES audit record.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages will displayed in each case.

- The input file does not exist or cannot be accessed.
- The output file could not be written.
- The input file is not in the proper format.

7.9 batch_analysis — Running NIDES in Batch Mode

The `batch_analysis` process incorporates the real-time intrusion-detection modules (statistical and rulebased) to allow NIDES users to run NIDES in "batch" mode to analyze audit data in the same way live data is analyzed.

This functionality can be useful in determining optimal statistical and/or rulebase configurations by experimenting with known data sets. Once tuned, the resultant configurations can be applied to the real-time analysis.

Furthermore, this functionality can be useful in processing data that has been collected remotely and only reported periodically (e.g., data from a gateway machine that is reported to the NIDES operator every other day).

While the `batch_analysis` process is typically invoked only by the test facility in the NIDES user interface, it can easily be invoked by a NIDES user from a command line.

Usage

batch_analysis [-a <*adset*>] [-i <*input file*>] [-I <*instance*>] [-report] [-d]

Flags

- -	Causes a brief description of the usage to be printed. No analysis will occur.
-a <*adset*>	Indicates which adset (audit data set) should be used as a source of NIDES audit records. The adset description will be read from the directory $IDES_ROOT/storage/adsets/INDEX. Once all the records in the specified adset have been processed, the `batch_analysis` process will terminate normally. An input source is mandatory. Either the **-a** <*adset*> or the **-i** <*input file*> flag must be specified.
-i <*input file*>	Indicates which input file should be used as a source of NIDES audit records. The input file may be any uncompressed file, any compressed file (with a ".Z" suffix), or standard input (indicated by a single dash "-"). The file must be a NIDES format file, possibly generated by `acc2ia` or `audit2ia`. Once all the records in the specified input file have been processed, the `batch_analysis` process will terminate normally. An input source is mandatory. Either the **-a** <*adset*> or the **-i** <*input file*> flag must be specified.
-I <*instance*>	Indicates which instance should be used for analysis. An instance dictates the configuration of both the statistical and rulebase modules and provides the set of subject profiles. The provided instance must already exist and reside in the directory $IDES_ROOT/storage/instances. Instances can be created, modified, and deleted using the NIDES user interface. This flag is mandatory.
-report	Indicates that the `batch_analysis` process should periodically report progress. This flag is always used when a test is started from the NIDES user interface. This flag, while not detrimental to normal analysis, should be used only when a user plans to track the progress of a test with the NIDES user interface.

In order for `batch_analysis` to locate the NIDES user interface to report progress, the IPC_NAMESERVER environment variable must be set, and must have the same value as when the NIDES user interface is/was invoked.

-d Indicates that debugging information should be printed out during normal execution.

Configuration

The `batch_analysis` program uses the following configuration items when executing.

- IDES_ROOT is used in determining the locations of many files needed for analysis. Instance configuration information is located in two directories:
  - $IDES_ROOT/storage/instances/<*instance*> (location of analysis configuration files).
  - $IDES_ROOT/storage/instances/ <*instance*>/profiles (Location of instance profiles.)
  - $IDES_ROOT/storage/dmf/ <*instance*>.res. (Test results are placed in the subdirectories contained in this directory.)
  - $IDES_ROOT/storage/adsets/INDEX/<*datafile*>.audinfo. (If an adset is used, the adset description is located in this file.)
- IPC_NAMESERVER is used by the `batch_analysis` process to locate the NIDES user interface to report progress when the **-report** option is specified. For the `batch_analysis` process to successfully locate the NIDES user interface process, both processes must have been started with IPC_NAMESERVER set to the same value.

Examples

The following examples are sample usages for the batch-analysis utility:

- `batch_analysis -a adset_july -I july_experiment -report`

  Starts a `batch_analysis` process that will use the adset called adset_july and the instance called july_experiment, and will report status to the NIDES user interface, if it is running.

- `batch_analysis -i /a/b/mydata/sept.dat -I sept_experiment -report`

  Starts a `batch_analysis` process that will use the NIDES audit data file located in /a/b/mydata/sept.dat and the instance called sept_experiment, and will report status to the NIDES user interface, if it is running.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages will displayed in each case.

- The input file does not exist or cannot be accessed.
- The instance configuration does not exist or is incomplete.
- The test results directory cannot be created or accessed.

7.10 iamerge — Merging Audit Data Files

The `iamerge` program reads in NIDES audit records from two sources and merges the records in time-sorted order. Both sets of NIDES audit records inputs must already be in time-sorted order for the `iamerge` process to work correctly.

This functionality can be useful in combining smaller NIDES audit record files gathered from different sites into a single file.

Usage

iamerge [-il <*input file #1*>] [-i2 <*input file #2*>] [-o <*output file*>] [-d]

Flags

--	Causes a brief description of the usage to be printed. No processing will occur.
-i1 <*input file1*>	Indicates the first file from which NIDES audit records should be read. The input file may be any uncompressed file, any compressed file (with a ".Z" suffix), or standard input (indicated by a single dash "-"). This flag is mandatory.
-i2 <*input file2*>	Indicates the second file from which NIDES audit records should be read. The input file may be any uncompressed file, any compressed file (with a ".Z" suffix), or standard input (indicated by a single dash "-"). This flag is mandatory.
-o <*output file*>	Indicates the file to which NIDES audit records should be written. The output file may be any uncompressed file, any compressed file (with a ".Z" suffix), or standard output (indicated by a single dash "-"). This flag is mandatory.
-d	Indicates that debugging information should be printed out during normal execution.

Configuration

The `iamerge` program does not rely on any environment variables or configuration files.

Examples

The following examples are sample usages for the `iamerge` utility:

- `iamerge -i1 /archive/file1 -i2 /archive/file2 -o /archive/merge.Z`

  Reads from the files `/archive/file1` and `/archive/file2` and writes the merged output, in compressed format, to the file `/archive/merge.Z`.

- `iamerge -i1 A.Z -i2 B.Z -o - | iamerge -i1 - -i2 C.Z -o merge.Z`

  The first `iamerge` process reads from the compressed files `A.Z` and `B.Z` and writes the output to standard output; the second `iamerge` process reads from standard input and the compressed file `C.Z` and writes the output, in compressed format, to the file `merge.Z`.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages are displayed in each case.

- Either of the input files does not exist or cannot be accessed.
- Both input files are specified to be standard input.
- The output file cannot be created.

7.11 iapr — Printing Audit Record Data

The `iapr` process reads NIDES audit records, either from a file or directly from the `arpool` process, and prints out, in ASCII format, specified fields from each record.

This functionality can be useful in examining NIDES audit records archived in data files or in viewing, in real time, the flow of audit records passing through the `arpool` process.

The ASCII output of `iapr` is designed to allow NIDES users an easy way to view NIDES audit records. The ASCII output is not interpreted by any other NIDES process, although the output can be used as input to standard UNIX utilities such as `awk` or `perl`.

Usage

iapr [-i <*input file*>] [-format <*format*>] [-lbuffer] [-d]

Flags

- - Causes a brief description of the usage to be printed. No processing will occur.

-i <*input file*> Indicates the file from which NIDES audit records should be read. The input file may be any uncompressed file, any compressed file (with a ".Z" suffix), or standard input (indicated by a single dash "-"). If this flag is not specified, the `iapr` process will attempt to contact the `arpool` process and retrieve NIDES audit records through RPC calls.

-format <*format*> Allows the user to specify the exact set of fields and format of the audit record data output. The format specification is treated as a text string where the specific NIDES audit record fields are represented as { <*field*> }. For example, the format used to print the action type and the effective user of a record, separated by a colon, would be `{action} : {uname}`. Because white space and certain special characters are interpreted by most UNIX shells, NIDES users should be careful to properly use quoting characters when this flag is used. If this flag is not specified, the default format will be used.

The available NIDES audit record fields are

- version

  The version of the NIDES audit record. The current version is 4.

- rseq

  The sequence number assigned by the `arpool` process. The `arpool` process assigns unique sequence numbers to all incoming audit records. These numbers are only unique within a given run of the `arpool` process.

- recvtime

  The time at which the audit record was received by the `arpool` process, printed in the format:

  DayOfWeek Month DayOfMonth TimeOfDay Year

  as in:

  `Sun Apr 10 12:45:27 1994`

- raw-recvtime

  The time at which the audit record was received by the `arpool` process, printed in the format:

  Year Month DayOfMonth Hours Minutes Seconds

  as in:

  `19940410124527`

- tseq

  The sequence number assigned by the `agen` process. The `agen` process assigns unique sequence numbers to all generated audit records. These numbers are unique only within a given run of the `agen` process.

- atime

  The time at which the audit record was generated on the target host, printed in the format:

  DayOfWeek Month DayOfMonth TimeOfDay Year

  as in:

  `Sun Apr 10 12:45:27 1994`

- raw-atime

  The time at which the audit record was generated on the target host, printed in the format:

  Year Month DayOfMonth Hours Minutes Seconds

  as in:

  `19940410124527`

- hostname

  The name of the target host on which the audit record was generated.

- audit_src

  The type of the native audit system that generated the raw audit data. This field is numeric and the possible values are

  ```
  1 (C2),
  2 (PACCT -- standard UNIX accounting),
  3 (Obsolete -- no longer used),
  4 (LINK -- arpool generated disconnect),
  5 (BSM version 1 -- SunOS 4.1.X), or
  6 (BSM version 2 -- SunOS 5.X/Solaris2.X).
  ```

- action

  The canonical action type associated with the audit record. See Table 6.6 for a description of the NIDES audit record actions. The actual

values encountered depend upon the types and configurations of the native auditing systems active on any given target host. The possible values are
`VOID, DISCON, ACCESS, OPEN, WRITE, READ, DELETE, CREATE, RMDIR, CHMOD, EXEC, CHOWN, LINK, CHDIR, RENAME, MKDIR, MOUNT, UNMOUNT, LOGIN, BAD_LOGIN, SU, BAD, EXIT, LOGOUT, RSH, BAD_RSH, PASSWD, RMOUNT,BAD_RMOUNT,PASSWD_AUTH,` and `BAD_PASSWD_AUTH`.

- auname

  The real user identifier of the user who generated the audit record. This value should never change, even if the user changes the effective user identifier (e.g., with `su`).

- auname-label

  The security tag of the real user identifier. This value will be assigned only by native audit systems that support object labeling.

- uname

  The effective user identifier of the user who generated the audit record. This value will change when the user changes the effective user identifier (e.g., with `su`).

- uname-label

  The security tag of the effective user identifier. This value will be assigned only by native audit systems that support object labeling.

- pid

  The UNIX process identifier of the process that generated the audit record.

- ttyname

  The name of the controlling TTY of the UNIX process that generated the audit record. This value is not always known.

- cmd

  The name (not path) of the command executed.

- arglist

  The command line arguments to a command. This field is rarely filled in.

- syscall

  The name of the UNIX system call that generated the audit record.

- errno

  The error value generated by the UNIX system call that generated the audit record. An error value of zero actually indicates the absence of an error.

- rval

The return value generated by the UNIX system call that generated the audit record. For most UNIX system calls, a zero or positive integer value indicates the absence of an error.

- utime

  The total CPU time spent by a UNIX process executing non-kernel program code (e.g., everything but system calls). Reported only upon the termination of a UNIX process under UNIX accounting.

- stime

  The total CPU time spent by a UNIX process executing kernel program code (e.g., system calls). Reported only upon the termination of a UNIX process under UNIX accounting.

- rtime

  The total elapsed lifetime of a UNIX process. Reported only upon the termination of a UNIX process under UNIX accounting.

- mem

  The average memory usage of a UNIX process. Reported only upon the termination of a UNIX process under UNIX accounting.

- io

  The number of characters transferred (both input and output) during the lifetime of a UNIX process. Reported only upon the termination of a UNIX process under UNIX accounting.

- rw

  The number of blocks transferred (both input and output) during the lifetime of a UNIX process. Reported only upon the termination of a UNIX process under UNIX accounting.

- ouname

  An additional, action-specific user identifier (e.g., user name of a failed remote login).

- remoteuname

  An additional, action-specific user identifier (e.g., user name of a failed remote login).

- remotehost

  The name of the remote host involved with a distributed or network related operation (e.g., initiating host for remote login session).

- path0

  The absolute pathname of file argument #1 on the target system relevant to the audit record action. This field is filled in for all single argument file operations (e.g., read, write, create, delete) and all double argument file operations (e.g., link, rename).

- path1

  The absolute pathname of file argument #2 on the target system relevant to the audit record action. This field is blank for all single argument file operations (e.g., read, write, create, delete) and filled in for all double-argument file operations (e.g., link, rename).

-lbuffer Indicates that line buffering should be used when writing ASCII data to standard output. This flag causes data to be output only when a complete line is available. This can be desirable if the output of `iapr` is piped to another process.

-d Indicates that debugging information should be printed out during normal execution.

Configuration

The `iapr` program uses the following configuration item when executing.

- IPC_NAMESERVER is used by the `iapr` process to locate the `arpool` process. If the nameserver is not running or the `arpool` process is not registered, the `iapr` process will terminate, unless a file is specified using the `-i` option.

Examples

The following examples are sample usages for the `iapr` utility:

- `iapr`

  Reads data from the `arpool` process (if registered with the nameserver indicated by $IPC_NAMESERVER) and prints ASCII output using the default format.

- `iapr -i /archive/orac.ia.Z -lbuffer | more`

  Reads from the compressed file `/archive/orac.ia.Z` and prints ASCII output using the default format to a pipe, with line buffered output, to the viewing program `more`.

- `iapr -format '{action}:{auname}:{uname}:{path0}'`

  Reads data from the `arpool` process (if registered with the nameserver indicated by $IPC_NAMESERVER) and prints ASCII output showing four fields, separated by colons. The fields are the action type, the real user ID, the effective user ID, and the first file argument (if any) for the action.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages are displayed in each case.

- The input file does not exist or cannot be accessed.
- The format specification is invalid.
- The `arpool` process cannot be contacted.

7.12 init_priv_user_list — Initializing the List of Privileged Users

The `init_priv_user_list` program initializes the list of NIDES privileged users. It uses as input a file containing the names of users who are allowed to execute NIDES privileged commands. See Table 6.1, on page 104, for a listing of privileged and nonprivileged functions. The input file is simply an ASCII text file, with one user name per line. The file is converted to an internal NIDES format by the init_priv_user_list program for use by NIDES.

Usage

init_priv_user_list [-init <*file*>] [-v] [-p]

Flags

-	Causes a brief description of the usage to be printed. No processing will occur.
-init <*file*>	Allows you to specify an optional file name to use for initialization of the privileged user list. If this argument is not specified the default file is used. The default file is IDES_ROOT/etc/pusers. The `pusers` file should contain all the user accounts for privileged users of NIDES. The format of the `pusers` file is simple ASCII text, one user name per line in the file.
-v	Puts the program into verbose mode. When this flag is used, messages will be displayed showing progress of the program as execution proceeds. The default setting of the program is non-verbose and no messages are printed during execution.
-p	Causes the program to print the contents of the current privileged user list without making any changes to the list.

Configuration

The init_priv_user_list program uses the following configuration item when executing.

- IDES_ROOT is used to determine where to write the new privileged user file. The file is written to IDES_ROOT/etc.

Examples

- Privileged user file initial configuration. When you first install NIDES you will need to initialize the privileged user list. Input the names of all NIDES privileged users in the IDES_ROOT/etc/pusers file. Then execute the command as follows:

  ```
  %init_priv_user_list  -v
  ```

  The -v flag is optional and is useful as it acknowledges the contents of your `pusers` file, by listing the names of all the users added to the internal NIDES privileged user list.

- Reviewing the privileged user list. Periodically you may want to review who is in the list of privileged users, to help determine if changes are needed. To do this, execute the command as follows:

  ```
  %init_priv_user_list  -p
  ```

- Using a different `pusers` file. If you have entered your list of users in a different file, perhaps even located in a different directory, you can use this file to update/initialize your privileged user list by executing the command as follows (say your file is located in /root/config/files/priv_users):

  ```
  %init_priv_user_list  -init  /root/config/files/priv_users  -v
  ```

  Once again the -v flag is optional.

- Getting the list of program options. To get the list of possible program flags/options, execute the command as follows:

  ```
  %init_priv_user_list  --
  ```

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages are displayed in each case.

- The input file specified does not exist or cannot be accessed.
- The privileged user file cannot be updated, perhaps because the user executing the program cannot write to the IDES_ROOT/etc directory.
- The IDES_ROOT environment variable is not set. This will cause the program to exit without making any changes to the privileged user list, because it does not know where to write the internal privileged user list.

7.13 init_stat_config — Converting the Statistical Configuration File

The `init_stat_config` program is a utility that will read in an ASCII (human-readable) version of the NIDES statistical component configuration file and convert it into a binary format that is used by the various NIDES components. It is an alternate method for reconfiguring the stats configuration and should be used only at installation time (see Section 9.4.2.4.1).

The binary version of the stats configuration file is called `stat_config,` and there should always be one for each instance, as well as one in the `IDES_ROOT/etc` directory (this file is used as the default stat configuration for newly created instances).

Usage

init_stat_config [-I <*instance*> | -etc] [-f <*filename*>] [-v]

Flags

- - Causes a brief description of the usage to be printed. No processing will occur.

-f <*filename*> Indicates the name of the ASCII file that contains the stat configuration data. This option is mandatory — that is, an ASCII file must be specified for this utility to work.

-I <*instance*> Provides the name of the instance for which the stat configuration file will be created. Make sure that the instance is created beforehand.

-etc An alternative to the **-I** option. Instead of writing the binary config file to an instance, the config file will be written to the `IDES_ROOT/etc` directory. This version of the config file will subsequently be used for creating default (new) instances, and will overwrite the existing default file.

-v Prints out diagnostics for the *init_stat_config* program. For example, it will print out the entire contents of the config file after it has been converted to the binary format for verification purposes.

Configuration

The `init_stat_config` utility uses the following configuration item when executing.

- IDES_ROOT is used to determine where the binary config file should be placed. Specifically, the config file will be stored in the directory `IDES_ROOT/storage/instances/` *`<instance>`* in `IDES_ROOT/etc` if the **-etc** option is specified.

Examples

The following examples are sample usages for the `init_stat_config` utility:

- `init_stat_config -I foo -f myconfig`
  Reads from the ASCII file `myconfig` and places the binary config file in `IDES_ROOT/storage/instances/foo/stat_config`.
- `init_stat_config -etc -f yourconfig`
  Reads from the ASCII file `yourconfig` and places the binary config file in `IDES_ROOT/etc`.

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages are displayed in each case.

- The ASCII input file does not exist or cannot be accessed.
- The ASCII input file is improperly formatted.
- The instance cannot be accessed (doesn't exist or no read/write access).
- The `etc` directory had problems (current `stats_config` file not overwriteable).
- The IDES_ROOT environment variable is not set. This will cause the program to exit without making any changes to the stat config file, since it does not know where to locate the file.

7.14 ipc_nameserver — Providing IPC Name Service

The `ipc_nameserver` runs in the background, providing RPC lookup service to all NIDES processes. The `ipc_nameserver` program must be running before any NIDES programs are executed. You can keep the `ipc_nameserver` running indefinitely; it will provide service across invocations of other NIDES programs. Only one `ipc_nameserver` process should be running on your NIDES host at one time.

Usage

ipc_nameserver [-debug]

Flags

-debug Indicates that debugging information should be printed out during normal execution.

Configuration

The `ipc_nameserver` program uses the following configuration item when executing.

- IPC_NAMESERVER specifies the port used for any interprocess communications between the `ipc_nameserver` and other NIDES processes.

Examples

The following example shows usages for the `ipc_nameserver` utility:

- `ipc_nameserver &`

  Starts the `ipc_nameserver` process in the background; this is the preferred method for the ipc_nameserver.

Error Conditions/Return Values

On startup, errors will occur under the following conditions.

- The IPC_NAMESERVER environment variable is not set.
- The IPC_NAMESERVER address is already in use. This indicates that you already have an `ipc_nameserver` process running on your system.

7.15 nides — Invoking NIDES From the Command Line

The `nides` program is an intrusion-detection expert system that monitors usage on a number of computers connected via an Ethernet network. It is an X/MOTIF-based user interface program that will allow you to set up target hosts and real-time intrusion-detection analysis, as well as to run NIDES experiments.

Usage

nides [-d]

Flags

-d Produces a variety of debugging information while `nides` is running.

Configuration

The `nides` program uses the following configuration items when executing.

- IDES_ROOT is used in determining the locations of many files needed for analysis.
- IPC_NAMESERVER is used by `nides` to locate and communicate with other NIDES processes.

Examples

To invoke `nides` from the UNIX command line enter:

- unix-prompt%`nides`

Error Conditions/Return Values

Errors will occur for the following conditions. Appropriate messages will displayed in each case.

- The `IDES_ROOT` environment variable is not set. You need to specify `IDES_ROOT` in your environment, or `nides` will not be able to access any start-up or configuration information.
- NIDES start-up error. Check your `IPC_NAMESERVER` environment variable and verify that the `ipc_nameserver` is running on your NIDES host.
- Unknown user. Your login account is not known to the host on which you have invoked `nides.` Notify your system administrator, because something may be wrong with your account.
- Problems reading the privileged user list. There is a problem reading the NIDES privileged user list so you will have limited capability with the `nides` program. Section 6.2 describes privileged and non-privileged capabilities.

- Problems reading the default target host file. This is not an error (i.e., `nides` will not abort), but your target host list will be empty when you begin configuring your target hosts for NIDES monitoring.
- Cannot read list of e-mail recipients. This is not an error (i.e., `nides` will not abort), but the e-mail list will be empty when you begin configuring your alert recipient list.
- Problems deleting old alert_filter list. A leftover `alert_filter` list in the `IDES_ROOT/etc` directory cannot be removed (NIDES always starts off with no filters for alerts – that is, all alerts are reported initially). Make sure you have write permission to this directory, or have your system administrator remove this file for you.
- No real-time instance available. There is a problem creating a default real-time instance, so `nides` cannot continue. Make sure that you have write permission for the `IDES_ROOT/storage/instances` directory, or have your system administrator create the default instance for you.

Chapter 8

Audit Data Source Customization

Previous versions of NIDES relied upon agen processes written in C and compiled for the target architecture (Sun SPARC). Thus, the number of different platforms that NIDES could monitor was severely limited. In an effort to permit monitoring of multiple platforms, combined with the desire to permit substantial "site customization," we have developed a working agen process written in Perl.

Perl is a powerful scripting language that provides access to many routines normally accessible only through compiled system libraries. Furthermore, Perl is available at no cost, easy to install, and operational on a large number of operating systems and hardware platforms. For more information on Perl, see [4, 5].

8.1 Caveats

It should be noted that the primary intent of the Perl agen was to permit monitoring of a variety of systems as well as allowing individual sites to customize the monitoring to fit their native audit systems and needs. The primary developer is not a Perl expert, so the Perl code looks rather like C code, is not inordinately fast, and probably could be well served by a revision from a Perl expert. Thus, anyone who has suggestions, modifications, enhancements and/or code to process new audit sources is greatly encouraged to share them with the NIDES team so that they can be integrated into the official version.

Furthermore, because the NIDES team does not have access to systems other than Sun SPARC, the Perl code has been tested only on Sun SPARC systems. We hope that the code will work on other systems without modification.

If you find that the Perl code does not work, consider issues such as "big endian" versus "little endian" differences (Sun is "big endian") in the XDR and IPC code, sysread/syswrite differences, or other problems of incompatibility.

8.2 Perl Files

The Perl agen release consists of three categories of files — main programs, NIDES library files, and audit customization files:

1. Main programs
   - agen.pl — the "main" code for the Perl agen process
   - agend.pl — the "main" code for the Perl agend process
2. NIDES library files
   - audit_rec.ph — the definition of important constants
   - audit.pl — the code to process NIDES formatted audit records
   - xdr.pl — the code to handle reading and writing data structures
   - ipc.pl — the code to handle inter-process communication
3. Audit customization files
   - custom.pl — the code that "defines" the audit sources to monitor
   - syslog_to_nides.pl — sample code for handling syslog audit data
   - tcpd_to_nides.pl — sample code for handling TCP wrapper data
   - acct_to_nides.pl — sample code for handling UNIX accounting data
   - nides_to_nides.pl — sample code for handling SunOS BSM data using the compiled version of agen

All of these files, once installed, can be found in the directory $IDES_ROOT/bin/bin.perl.

8.3 Customization Decisions

Now that Perl versions of agen and agend are available, the NIDES administrator must make some decisions regarding how each host should be configured. Specifically, the administrator should consider the following questions:

1. Can your target host run the standard (compiled) agen/agend?
   (a) If so, are UNIX accounting and/or one of C2 or BSM sufficient?
   (b) If not, what other audit sources do you want to incorporate?
2. If your target hosts cannot run the standard (compiled) agen/agend, do you have (or can you acquire) Perl?
3. Do the audit sources you want to incorporate already have Perl agen code to process the data.
   (a) If not, are you capable of writing, and willing to write Perl code for that source?
4. Do all of your target hosts have TCP/IP capabilities?
   (a) If so, do you want to run the Perl agen in real-time mode?
   (b) If not, do you want to use the Perl agen in batch mode to process collected audit data after the fact?

8.4 Customizing the Perl agen

In an effort to make customization as simple and flexible as possible, the Perl `agen` has table-driven audit source customization. Each entry in the table must have six parameters properly defined. Upon invocation, the Perl `agen` will attempt to "activate" all of the defined sources. One record is obtained from each active source and the earliest record is processed (and the next record from that source acquired), thus providing global time sorting if each source has correct time sorting.

Six parameters, each of which is a string in Perl, must be supplied for each audit source:

1. A source name (used internally only), for example: `TCP-WRAPPER`. This value is used to distinguish sources within the Perl `agen` code, and *must* be unique across all other configured audit sources.

2. The type of data to read ('nides', 'line', or 'other'), for example: 'nides'. This value indicates the kind of data to expect from the source:

   - `'nides'` — NIDES records (e.g., from audit2ia or acc2ia)
   - `'line'` — single line, ASCII records (e.g., syslog or tcpd)
   - `'other'` — other, either ASCII or binary record format

3. The type of the audit source stream ('once' or 'continuous'), for example: 'continuous'. This value indicates how to expect data to be generated by the source:

   - `'once'` — processing should stop at the end of file (EOF)
   - `'continuous'` — processing should continue to expect more data even after the end of file has been reached

4. The position at which to start reading data (`'head'` or `'tail'`), for example: `'head'`. This value indicates where to start processing within the audit stream:

   - `'head'` — start reading at the beginning of the stream
   - `'tail'` — start reading at the end of the stream

   Note, the 'tail' option is valid only when the audit source is really a normal file (as opposed to a process).

5. The source of the data, for example:

   ```
   '/var/adm/pacct'
   '/usr/nides/bin/bin.sun4/agen -bsm -o - |'
   'zcat /usr/nides/data/jul94.Z |'
   'grep -vi local.com /usr/data/tcpd/archive | grep refused |'
   ```

   This value indicates what command to execute in order to initiate the audit data stream. The open command in Perl has a nice feature allowing you to simply open a file or to invoke a series of commands with the same function. Thus, the real requirement is that whatever is provided for this field, when passed to the Perl `open` command, provides the intended data.

The first example above causes data to be read from the file `/var/adm/pacct;` the second example invokes a compiled version of `agen` that will read the standard BSM audit trail and output the data; the next example invokes a process (zcat) that causes the file `/usr/nides/data/jul94.Z` to be decompressed; the last example selects lines that do not contain the string "local.com" and do contain the string "refused" from the file `/usr/data/tcpd/archive.`

6. The name of the conversion function, if any, for example:

```
convert_tcpd
convert_acct
```

This value indicates which conversion function, if any, should be called to convert the input data into a NIDES record. Any source that is of type `line` or `other` must provide a conversion function; a conversion function is optional for a `nides` source.

The Perl `agen` expects all customization to occur in the file `custom.pl.` The six parameters used to describe a specific audit source must be stored together in a single array, a reference to which is added to the variable `source_data` that contains a list of potential sources. An example `custom.pl` file is shown in Table 8.1.

While the goal of the table-driven customization was to offer both consistency and ease of use, there is one cross-field dependency that a NIDES administrator using the Perl `agen` must keep in mind. The conversion function will be invoked with different arguments, depending upon whether the source has been identified as `nides`, `line`, or `other`. In the case of the `line` and `other` sources there are two parameters, and the second parameter is always the audit record to fill in. The `nides` source contains only one parameter — the audit record.

- If the source is identified as `nides,` the parameter will be a NIDES audit record read in from the source; it is the responsibility of the conversion function to copy or modify all fields in the provided audit record. The sample Perl file `nides_to_nides.pl` contains example functions that work with a `nides` source type.
- If the source is identified as `line,` the first parameter will be the single line of ASCII text read from the source; it is the responsibility of the conversion function to fill in all the required fields in the outgoing record. The sample Perl file `tcpd_to_nides.pl` contains example functions that work with a line source type.
- If the source is identified as `other,` the first parameter will be the file handle of the source that has data to be processed; it is the responsibility of the conversion function to both read the raw data from the file handle and fill in all the required fields in the outgoing record. The sample Perl file `acct_to_nides.pl` contains example functions that work with an `other` source type.

8.5 Audit Source Customization

All data processed by NIDES must be in the form of NIDES audit records. The Perl `agen` permits a NIDES administrator to fill in the fields of a NIDES audit record with whatever data is available

```
#
# Include modules required for converting other source data to NIDES format
#
require  "nides_to_nides.pl";
require  "syslog_to_nides.pl";
require  "tcpd_to_nides.pl";
require  "acct_to_nides.pl";

#
@BSM_params     = ('BSM',
                    'nides',
                    'continuous',
                    'head',
                    '$IDES_ROOT/bin/bin.sun4/agen -bsm -o - |',
                    'convert_bsm_nides');


# Read the UNIX accounting log file from the current EOF
#
@ACCT_params = ('ACCT',
                    'other',
                    'continuous',
                    'tail',
                    '/var/adm/pacct',
                    'convert_acct');

#
# Process all "TCP wrapper refused" lines in file "/usr/data/tcpd/archive"
#
@TCPD_params = ('TCPD',
                    'line',
                    'once',
                    'head',
                    'grep refused /usr/data/tcpd/archive |',
                    'convert_tcpd');


@source_data = (*BSM_params, *ACCT_params, *TCPD_params);
```

Table 8.1: Sample custom.pl Perl agen customization file

and relevant from the native auditing system. Appropriate access functions have been supplied with the Perl library files to simplify the manipulation of NIDES audit records.

8.5.1 Audit Data Action and Source Codes

To allow introduction of new sources and distinction between sources at the analysis phase, a NIDES administrator can create new source and action codes to extend the audit data as necessary. The Beta-update version of NIDES includes fifty user-definable audit action codes and fifty user-definable audit source codes. Tables 8.2 and 8.3 list the available codes. These codes are defined as C enumerated types and are recognized by all NIDES processes. The file `IDES_ROOT/bin/bin.perl/audit_rec.ph` defines these mappings for Perl.

In addition, two configuration files, `audit_actions` and `audit_sources`, are included in the IDES_ROOT/etc directory to support specification of the text strings used by the NIDES user interface when presenting audit data containing user-defined audit action and source codes. Sample versions of these files are included with the NIDES release and are also shown in Tables 8.4 and 8.5. If names for user-defined action and source codes are not specified default values are presented.

8.5.2 Example Customization File

Example code shown in Table 8.6 processes single-line reports from Wietse Venema's TCP wrapper package. The example Perl code uses a numeric code reserved for site-specific source customization (IA_SRC_USER00 = 100) as the distinct code to identify TCP wrapper audit data (IA_SRC_TCPD). Likewise, because there is no appropriate existing action code, the Perl code also assigns a site-specific action code (IA_ACTION_USER00 = 200) for reporting hostname-to-address mismatches (IA_HOST_ADDR_MISMATCH).

The Perl code goes on to define a conversion function for transforming TCP wrapper reports to NIDES audit records. The function `convert_tcpd` was "registered" in the `custom.pl` example shown in the previous section; thus, each line read from the TCP wrapper log file will be passed to the function `convert_tcpd` defined below.

The first parameter, `$_[0]`, is the line of text read from the log file, and the second parameter, `$_[1]`, is the NIDES audit record to fill. The return value from the function should be -1 (to indicate the source is at the end of file), 0 (to indicate that no record was generated, or 1 (to indicate that a record was generated). Currently, only one record can be generated per call to a conversion function.

The conversion function `convert_tcpd` goes through the following steps to process a single line report:

1. Converts the timestamp to a UNIX time (call to `syslog_time`)

2. Breaks the line into separate tokens (call to `split`)

3. Clears the provided audit record (call to `clear_audit_record`)

4. Determines if line contains "address mismatch" (use of `=~` operator)

5. Fills in the NIDES audit record fields according to the detected event

6. Returns the number of records generated (either 0 or 1)

Action Code	Value	Action Code	Value
IA_ACTION_USER00	200	IA_ACTION_USER25	225
IA_ACTION_USER01	201	IA_ACTION_USER26	226
IA_ACTION_USER02	202	IA_ACTION_USER27	227
IA_ACTION_USER03	203	IA_ACTION_USER28	228
IA_ACTION_USER04	204	IA_ACTION_USER29	229
IA_ACTION_USER05	205	IA_ACTION_USER30	230
IA_ACTION_USER06	206	IA_ACTION_USER31	231
IA_ACTION_USER07	207	IA_ACTION_USER32	232
IA_ACTION_USER08	208	IA_ACTION_USER33	233
IA_ACTION_USER09	209	IA_ACTION_USER34	234
IA_ACTION_USER10	210	IA_ACTION_USER35	235
IA_ACTION_USER11	211	IA_ACTION_USER36	236
IA_ACTION_USER12	212	IA_ACTION_USER37	237
IA_ACTION_USER13	213	IA_ACTION_USER38	238
IA_ACTION_USER14	214	IA_ACTION_USER39	239
IA_ACTION_USER15	215	IA_ACTION_USER40	240
IA_ACTION_USER16	216	IA_ACTION_USER41	241
IA_ACTION_USER17	217	IA_ACTION_USER42	242
IA_ACTION_USER18	218	IA_ACTION_USER43	243
IA_ACTION_USER19	219	IA_ACTION_USER44	244
IA_ACTION_USER20	220	IA_ACTION_USER45	245
IA_ACTION_USER21	221	IA_ACTION_USER46	246
IA_ACTION_USER22	222	IA_ACTION_USER47	247
IA_ACTION_USER23	223	IA_ACTION_USER48	248
IA_ACTION_USER24	224	IA_ACTION_USER49	249

Table 8.2: NIDES User-defined audit record action codes

Source Code	Value	Source Code	Value
IA_SRC_USER00	100	IA_SRC_USER25	125
IA_SRC_USER01	101	IA_SRC_USER26	126
IA_SRC_USER02	102	IA_SRC_USER27	127
IA_SRC_USER03	103	IA_SRC_USER28	128
IA_SRC_USER04	104	IA_SRC_USER29	129
IA_SRC_USER05	105	IA_SRC_USER30	130
IA_SRC_USER06	106	IA_SRC_USER31	131
IA_SRC_USER07	107	IA_SRC_USER32	132
IA_SRC_USER08	108	IA_SRC_USER33	133
IA_SRC_USER09	109	IA_SRC_USER34	134
IA_SRC_USER10	110	IA_SRC_USER35	135
IA_SRC_USER11	111	IA_SRC_USER36	136
IA_SRC_USER12	112	IA_SRC_USER37	137
IA_SRC_USER13	113	IA_SRC_USER38	138
IA_SRC_USER14	114	IA_SRC_USER39	139
IA_SRC_USER15	115	IA_SRC_USER40	140
IA_SRC_USER16	116	IA_SRC_USER41	141
IA_SRC_USER17	117	IA_SRC_USER42	142
IA_SRC_USER18	118	IA_SRC_USER43	143
IA_SRC_USER19	119	IA_SRC_USER44	144
IA_SRC_USER20	120	IA_SRC_USER45	145
IA_SRC_USER21	121	IA_SRC_USER46	146
IA_SRC_USER22	122	IA_SRC_USER47	147
IA_SRC_USER23	123	IA_SRC_USER48	148
IA_SRC_USER24	124	IA_SRC_USER49	149

Table 8.3: NIDES User-defined audit record source codes

```
#
# Host/address name mismatch reported by TCP wrapper
#
200 NS_MISMATCH
```

Table 8.4: Sample audit-actions file

```
#
# Reports from the TCP wrapper
#
100 TCPD
```

Table 8.5: Sample audit_sources file

8.6 Audit Record Support Functions

To facilitate the manipulation of NIDES audit records, a library of Perl functions has been supplied to allow easy assignment of audit record fields. All the functions are of the form `set_audit_`*`field`* where *field* is a field in the audit record. The audit record parameter should always be passed by "reference" using the * designator, for example:

```
&set_audit_source(*record,  $IA_SRC_TCPD);
&set_audit_real_user(*record,  "root");
```

Tables 8.7 and 8.8 list all of the audit record assignment functions, including examples of how each function would be called in actual Perl code. These functions are defined in the Perl file `audit.pl`.

Tables 8.9 and 8.10 list additional Perl functions that may be useful. These functions are also defined in the Perl file `audit.pl`.

8.7 Analysis Customization

The addition of new audit sources does not inherently extend the statistical analysis if the event codes generated do not map into the original Beta release event codes. However, as long as Beta event codes are used, the Perl `agen` can be used to allow statistical analysis of target hosts that could not previously be monitored because of an unsupported audit system or an unsupported hardware platform.

The utility of generating new types of data using Perl scripts is thus best realized when used in conjunction with the NIDES rulebase. In this way, event- or scenario-specific monitoring can occur. An example rule, shown in Table 8.11, works in conjunction with the TCP wrapper conversion script described earlier and shown in Table 8.6.

This rule is very rudimentary and is not intended to be used without further configuration. A more complex rule might generate an alert after a certain number of hostname-to-address mismatch records are observed because that condition might reflect a network-based attack or simply a DNS inconsistency or error.

```
eval 'sub IA_SRC_TCPD           { &IA_SRC_USEROO }';
eval 'sub IA_HOST_ADDR_MISMATCH { &IA_ACTION_USEROO  }';

sub convert_tcpd
{
    local($line)   = $\_[0];
    local(*record) = $\_[1];
    local($time)    =  &syslog_time($line);
    local(@tokens) = split(/[:\s\[\]]+/,  $line);
    local($count) = 0;

    # Clear out the audit record
    &clear_audit_record(*record);

    # Look for host/address mismatch report
    if( $line =~ /address mismatch/ )
    {
        # MMM DD HH:MM:SS HOST PROC[PID]: host name/address mismatch IP != HOST
        &set_audit_source(*record,  &IA_SRC_TCPD);
        &set_audit_action(*record,   &IA_HOST_ADDR_MISMATCH);
        &set_audit_ttime(*record, $time,  0);
        &set_audit_real_user(*record,  "tcpd");
        &set_audit_effective_user(*record,  "tcpd");
        &set_audit_host(*record,  $tokens[5]);
        &set_audit_command(*record,  $tokens[6]);
        &set_audit_pid(*record,  $tokens[7]);
        &set_audit_remote_user(*record,  $tokens[11]);
        &set_audit_remote_host(*record,  $tokens[13]);

        # Success => 0
        &set_audit_errno(*record,  0);
        &set_audit_retval(*record,  0);

        $count += 1;
    }

    return $count;
}
```

Table 8.6: Sample conversion function from `tcpd_to_nides.pl`

Function Name	Arguments Name	Arguments Type	Example Call
set_audit_mark	record mark	audit record int	set_audit_mark(*record, $mark)
set_audit_rseqno	record seqno_hi seqno_lo	audit record int int	set_audit_rseqno(*record, $seqno_hi, $seqno_lo)
set_audit_rtime	record seconds microseconds	audit record int int	set_audit_rtime(*record, $seconds, $microseconds)
set_audit_tseqno	record seqno_hi seqno_lo	audit record int int	set_audit_tseqno(*record, $seqno_hi, $seqno_lo)
set_audit_ttime	record seconds microseconds	audit record int int	set_audit_ttime(*record, $seconds, $microseconds)
set_audit_host	record host	audit record string	set_audit_host(*record, $host)
set_audit_source	record source	audit record int	set_audit_source(*record, $source)
set_audit_action	record action	audit record int	set_audit_action(*record, $action)
set_audit_real_user	record user	audit record string	set_audit_real_user(*record, $user)
set_audit_effective_user	record user	audit record string	set_audit_effective_user(*record, $user)
set_audit_pid	record pid	audit record int	set_audit_pid(*record, $pid)
set_audit_tty	record tty	audit record string	set_audit_tty(*record, $tty)
set_audit_command	record command	audit record string	set_audit_command(*record, $command)
set_audit_args	record count array	audit record int array of strings	set_audit_args(*record, $count, $array)
set_audit_syscall	record syscall	audit record int	set_audit_syscall(*record, $syscall)
set_audit_errno	record errno	audit record int	set_audit_errno(*record, $errno)
set_audit_retval	record retval	audit record int	set_audit_retval(*record, $retval)

Table 8.7: Perl script audit record manipulation functions (part 1)

	Arguments		
Function Name	Name	Type	Example Call
set_audit_res_utime	record time	audit record double	set_audit_res_utime(*record, $time)
set_audit_res_stime	record time	audit record double	set_audit_res_stime(*record, $time)
set_audit_res_rtime	record time	audit record double	set_audit_res_rtime(*record, $time)
set_audit_res_mem	record mem	audit record double	set_audit_res_mem(*record, $mem)
set_audit_res_io	record io	audit record double	set_audit_res_io(*record, $io)
set_audit_res_rw	record rw	audit record double	set_audit_res_rw(*record, $rw)
set_audit_other_user	record user	audit record string	set_audit_other_user(*record, $user)
set_audit_remote_user	record user	audit record string	set_audit_remote_user(*record, $user)
set_audit_path0	record path type	audit record string int	set_audit_path0(*record, $path, $type)
set_audit_path1	record path type	audit record string int	set_audit_path1(*record, $path, $type)

Table 8.8: Perl script audit record manipulation functions (part 2)

Function Name	Arguments		Example Call and Description
	Name	Type	
read_audit_record	result stream record	int file stream audit record	$result = &read_audit_record($stream, *record) Reads (in XDR format) the record from the file stream
write_audit_record	result stream record	int file stream audit record	$result = &write_audit_record($stream, *record) Writes (in XDR format) the record to the file stream
send_arpool_record	result stream record	int file stream audit record	$result = send_arpool_record($stream, *record) Writes (as an arpool RPC request) the record to the file stream
print_entire_audit_record	result	int	$result = &print_entire_audit_record($stream, *record Prints all fields in the record to the file stream
print_audit_record	result stream record	int file stream audit record	$result = &print_audit_record($stream, *record) Prints all defined fields in the record to the file stream
clear_audit_record	result record	int audit record	$result = &clear_audit_record(*record) Sets all fields in the record to default values
free_audit_record	result record	int audit record	$result = &free_audit_record(*record) Frees (i.e., makes undefined) all fields in the record

Table 8.9: Perl utility functions (part 1)

These functions extract specific fields from the audit record
All these functions take a single argument which is a NIDES format audit record

Function Name	Arguments		Example Call
	Name	Type	
get_audit_rseqno	seqno_hi seqno_lo	int int	(seqno_hi, seqno_lo) = &get_audit_rseqno(*record)
get_audit_rtime	sec usec		($sec, $usec) = &get_audit_rtime(*record)
get_audit_tseqno	seqno_hi seqno_lo	int int	(seqno_hi, seqno_lo) = &get_audit_tseqno(*record)
get_audit_ttime	sec usec		($sec, $usec) = &get_audit_ttime(*record)
get_audit_host	host	string	$host = &get_audit_host(*record)
get_audit_source	source	int	$source = &get_audit_source(*record)
get_audit_action	action	int	$action = &get_audit_action(*record)
get_audit_real_user	user	string	$user = &get_audit_real_user(*record)
get_audit_effective_user	user	string	$user = &get_audit_effective_user(*record)
get_audit_pid	pid	int	$pid = &get_audit_pid(*record)
get_audit_tty	tty	int	$tty = &get_audit_tty(*record)
get_audit_command	command	string	$command = &get_audit_command(*record)
get_audit_syscall	syscall	int	$syscall = &get_audit_syscall(*record)
get_audit_errno	errno	int	$errno = &get_audit_errno(*record)
get_audit_retval	retval	int	$retval = &get_audit_retval(*record)
get_audit_res_utime	time	int	$time = &get_audit_res_utime(*record)
get_audit_res_stime	time	int	$time = &get_audit_res_stime(*record)
get_audit_res_rtime	time	int	$time = &get_audit_res_rtime(*record)
get_audit_res_mem	mem	int	$mem = &get_audit_res_mem(*record)
get_audit_res_io	io	int	$io = &get_audit_res_io(*record)
get_audit_res_rw	rw	int	$rw = &get_audit_res_rw(*record)
get_audit_other_user	user	string	$user = &get_audit_other_user(*record)
get_audit_remote_user	user	string	$user = &get_audit_remote_user(*record)
get_audit_remote_host	host	string	$host = &get_audit_remote_host(*record)
get_audit_path0	path type	string int	($path, $type) = &get_audit_path0(*record)
get_audit_path1	path type	string int	($path, $type) = &get_audit_path1(*record)

Table 8.10: Perl utility functions (part 2)

```
' NOTE: Audit source IA_SRC_USEROO (100)    => TCP wrapper
' NOTE: Audit action IA_ACTION_USEROO (200) => host/address mismatch
'
rule[HostAddressMismatch(*):
        [+ev:eventˆHAM|audit_src == src#IA_SRC_USEROO,
                      action == ia#IA_ACTION_USEROO]
==>
        [$|ev:HAM]
        [!|sprintf('prstr,
                  "Host %s does not match reported address %s trying to access %s on %s.",
                  ev.remotehost, ev.remoteuname, ev.cmd, ev.hostname)]
        [!|inform(m#CRITICAL, ev.auname, ev.atime,
                 ev.rseq_hi, ev.rseq_lo, 'prstr,
                 "HostAddressMismatch")]
]
```

Table 8.11: Sample customized audit source rule HostAddressMismatch

8.8 Installation

Chapter 9 contains complete installation instructions for all NIDES software. Section 9.5.3.2 discusses configuration and use of Perl agens.

Chapter 9

Installation Instructions

Before installing NIDES, read this entire chapter, which details the installation and configuration process for the Beta-update version of the NIDES software. Table 9.1 summarizes the requirements for the Sun system that will serve as the NIDES host and the Sun systems that will serve as NIDES target hosts (audit data providers). With this release of NIDES you may also use non-Sun hosts as targets by using the audit data customization facility described in Chapter 8. The configuration of

NIDES Host Configuration	Recommended Configuration	Minimal Configuration
Sun OS:	4.1.3 or Solaris 1.1	
RAM	64 MB	32 MB
Swap Space	150 MB	75 MB
NIDES Directory Space	500 MB	100 MB
X windows	X11R5	X11R4
NIDES Target Host Configuration		
Sun OS:	4.1.3 or Solaris 1.1	
RAM	32 MB	16 MB
Swap Space	100 MB	50 MB
Sun BSM or C2 Auditing Installed	Yes	Optional (highly recommended)
UNIX Accounting	Yes	Yes

Table 9.1: Host Configuration Guidelines

non-Sun targets will depend upon the amount of audit data produced and the processing overhead associated with the audit data collection mechanisms. To monitor non-Sun targets in real time, the host must support TCP/IP and have a connection to the NIDES host to support data transfer.

9.1 Pre-installation Steps for NIDES Alpha and Beta Release Users

If you have been running the NIDES Alpha, Alpha-patch or Beta release, you must install the Beta-update release so that any previously installed release does not interfere with it. If you have never run NIDES, start with Section 9.3.

Install the Beta-update release in an area separate from earlier NIDES release software. If you do not want to keep the earlier release software on your computer, you can remove it completely, and install the Beta-update release in the same area. Make sure that if you do keep both releases on your system, they are carefully separated. All binary files and some data files are incompatible between releases.

NIDES audit data files are the only Alpha files compatible with any Beta release. Under any Beta release, audit data files require an index file. The utility `adset_index` can be used to create index files for existing NIDES audit data files; Section 7.2 describes this utility. No other NIDES configuration and profile files are compatible between any NIDES Alpha release and any NIDES Beta release. Do not attempt to use any Alpha release configuration or profile files with any Beta release.

9.2 Post-installation Steps for NIDES Beta Release Users

If you have been using the original Beta release of NIDES you may have created and used instances, audit data sets, and rulebase files that you want to use under the Beta-update release. Most of the data generated under the Beta release of NIDES is compatible with the Beta-update release. However, no automated process has been included with the installation scripts to transfer NIDES Beta release data into the Beta-update release IDES_ROOT directory area. Recommended procedures for transfer of NIDES Beta release data into the Beta-update release area are described below.

9.2.1 Audit Data

Two directories under the IDES_ROOT directory are used to store NIDES audit data sets and audit data archives — `IDES_ROOT/storage/adsets` and `IDES_ROOT/storage/dmf`. The `adsets` directory holds individual audit data set files and their index files. The `dmf` directory stores audit data archives in subdirectories containing a `.aud` suffix. Indices for the archives are stored in the INDEX directory located under the `dmf` area (review figure 9.1 on 268 for a complete description of the NIDES directory structure). The `dmf` directory also stores result data archives, which are directories with a `.res` suffix. When NIDES is initially installed, all these directories are empty. If you want to utilize all audit data sets and audit data archives from your NIDES Beta release, simply do a recursive copy from the adsets and dmf directories under your Beta release storage directory to the Beta-update release storage directory as follows:

```
%cp  -r  <Beta_IDES_ROOT>/storage/adsets/*  <Beta-update_IDES_ROOT>/storage/adsets
```

```
%cp -r <Beta_IDES_ROOT>/storage/dmf/*.aud <Beta-update_IDES_ROOT>/storage/dmf
%cp -r <Beta_IDES_ROOT>/storage/dmf/INDEX/*.audinfo <Beta-update_IDES_ROOT>/storage/dmf/INDEX
```

If you want to use only selected audit data sets and archives, you can copy each one individually. Each audit data set is composed of two files, a data file and an index file that is stored in the `IDES_ROOT/storage/adsets/INDEX` directory. Audit data archives are stored in directories named for the archive with a `.aud` suffix appended. The corresponding index file is located in the `IDES_ROOT/storage/dmf/INDEX` directory; the file will have the same name as the archive directory except with a `.audinfo` suffix. If you copy selected audit data sets or archives you must be sure to copy the corresponding index file as well. If some of the audit data sets in your adsets area were created as "virtual" audit data sets, their corresponding audit record archives must also be copied.

9.2.2 Test Result Data

Test result data generated under the NIDES Beta release can be copied into the Beta-update release area if test results from earlier experiments need to be reviewed using the NIDES User Interface browse functions. Test results are stored under the `IDES_ROOT/storage/dmf` directory. Each test instance will have a directory in the `dmf` area called <Instance-name>.res and a corresponding index file under the `dmf/INDEX` area called <Instance-name>.resinfo. Copy each directory and corresponding index file for every set of test results you want to use with the Beta-update release. To copy all test results to the Beta-update area, do the following:

```
%cp -r: <Beta_IDES_ROOT>/storage/dmf/*.res <Beta-update_IDES_ROOT>/storage/dmf
%cp -r <Beta_IDES_ROOT>/storage/dmf/INDEX/*.resinfo <Beta-update_IDES_ROOT>/storage/dmf/INDEX
```

However, we recommend that you copy only desired result directories to save disk space.

9.2.3 Instance Data

Under the `IDES_ROOT/storage` directory area is a directory called `instances` containing one directory for each NIDES instance. Each instance directory contains the following five entries:

1. `config` — the instance configuration file
2. `kb` – the rulebase file
3. `rb_config` — the rulebase configuration file
4. `stat_config` — the statistical configuration file
5. `profiles` – a directory containing all user statistical profiles

Any `kb` rulebase files created under the original Beta release are not compatible with the Beta-update release. We also recommend that any `config`, `rb_config`, and `stat_config` files created under the original Beta release not be copied into Beta-update release instances. However, it is quite likely that statistical profiles generated under the Beta release will be of use in the Beta-update release, especially if you have a set of trained or nearly trained profiles that you want to continue to use under the Beta-update software. The following procedure is recommended for recreating Beta release instances for use in the Beta-update release:

1. If you have created customized default `rb_config` and/or `stat_config` files, which are stored in your Beta Release `IDES_ROOT/etc` area, you can reuse these configuration files in your Beta-update release. First, back up the default configuration files, located in your Beta-update release `IDES_ROOT/etc` area, that you will be replacing. Then copy the configuration files from the Beta release area into your new Beta-update area.

2. For every instance you want to duplicate from the Beta release, create a new instance in the Beta-update area using the NIDES User Interface Customize Menu Test Instances Option. For each instance created, default `kb`, `config`, `rb_config`, and `stat_config` files are created (if you updated the default configuration files in the `IDES_ROOT/etc` as described in Step 1, your new instance will use these files). In addition, when the instance is created, an empty `profiles` directory is created.

3. After the new instance has been created you may copy user profiles from a Beta release instance into the `profiles` directory for the newly created instance. For each subject you must be sure to copy the current profile file (stored in a file with the same name as the user), the historical profile file (stored in a file with the same name as the user with a `.hist` suffix), and if it exists, the optional temporary category file (stored in a file with the same name as the user with a `.lemm` suffix).

4. The first time you use one of your copied instances for a NIDES batch run, check to see if the profile last update timestamps are later than the audit data you will be using. If they are, synchronize the profiles before running any experiments.

9.2.4 Rulebase Files

The format for the rulebase `event` fact was changed under the Beta-update release. The `event` fact is used to store the NIDES audit records analyzed by the rulebased analysis component. Because of this format change, rulebase files created with the earlier Beta release are not compatible with the Beta-update release software. Any rule source files that were created and compiled under the Beta release will need to be modified if they contain references to `event` facts (Table 5.2 shows the new format for the NIDES `event` fact). Once you've modified your rule source files you can compile and install them in your Beta-update release area. See Section 5.2 if you need additional information on rule compilation and installation.

9.3 Installing NIDES Software from Tape

Installing NIDES requires root access, so it should be performed by your system administrator.

9.3.1 NIDES Software Installation

Follow these four steps to install the NIDES software:

Step 1 — Create an ides account and group

Create an "ides" account and group, both called ides. The ides group must include the ides user and any additional users who will run NIDES.

Step 2 — Determine the directory where NIDES will reside

Decide where you want to load NIDES. This directory will be referred to as IDES_ROOT throughout these instructions. This directory should *not* be the home directory of the ides user. You will be loading about 40 MB of software, and you should reserve at least 100 MB of additional disk space where NIDES can write information. The space needed depends on the volume of audit and result data generated.

Step 3 — Load the contents of the tape onto your NIDES machine

Making sure you are superuser, place the tape in an appropriate tape drive, and type the following command:

```
# /usr/etc/extract_unbundled
```

Typing this command executes an install script that automatically reads the contents of the tape onto your system. During the install process, you will be asked for the name of the directory where NIDES will reside on your system.

If the tape drive is a remote device, root on the local machine must have rsh access as root to the remote machine.

Step 4 — NIDES software has been loaded onto your system

When software loading is completed, the NIDES distribution is contained in the directory specified. From now on we will call this directory IDES_ROOT. Figure 9.1 shows the directory structure and files included on the NIDES release tape. You can now configure NIDES to run in your environment.

9.3.2 GCC Compiler Installation

The Beta-update version of NIDES needs a C compiler to compile rules for use in the NIDES rulebase; we recommend using `gcc` version 2.5.6 or later. The standard C compiler included with the SunOS release is not compatible with the NIDES rulebase. Once you have determined which C compiler will be used, make sure it is accessible to any NIDES users who may compile rules. Modify the `makerule` script located in $IDES_ROOT/exsys/bin to use the chosen C compiler.

9.4 Configuring NIDES

After you have installed NIDES from tape, you must configure the host system before running NIDES. Perform the following steps when you first install NIDES on your computer. To configure the NIDES host computer, you should be logged into the NIDES host system as user "ides".

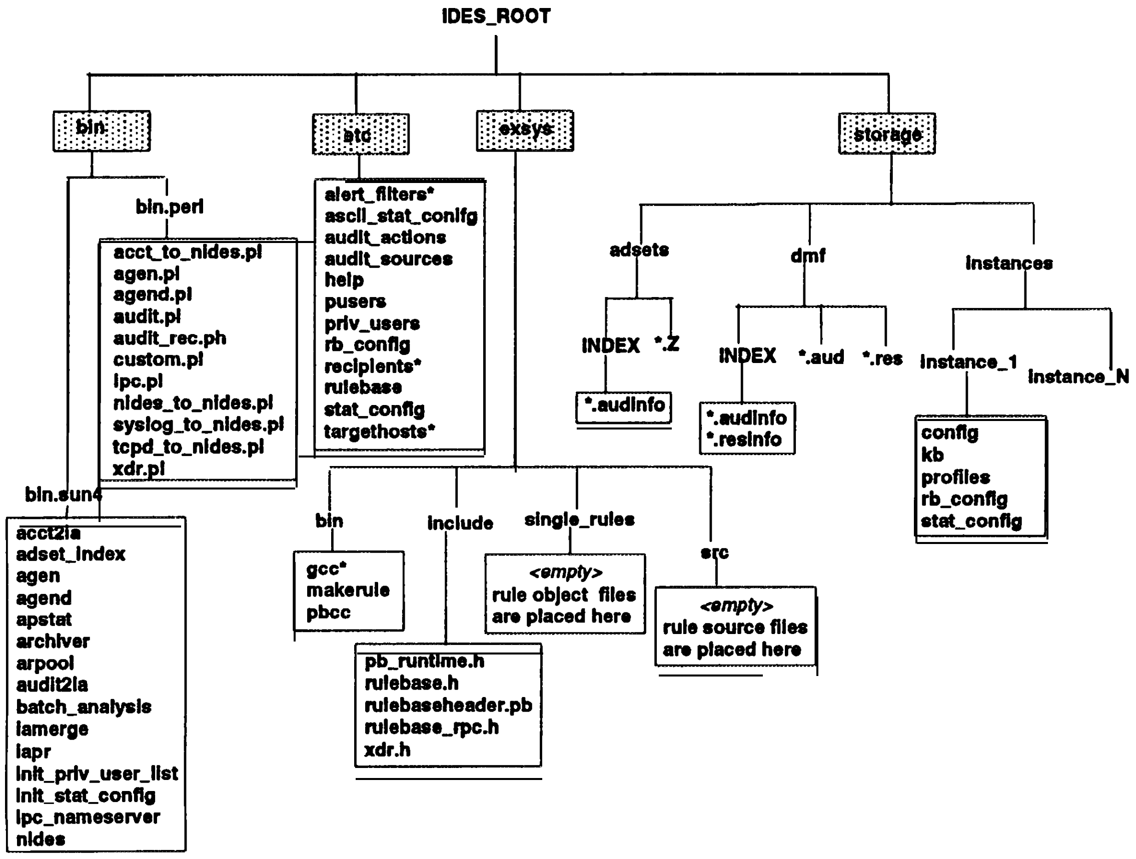


Figure 9.1: NIDES Beta-update Release Filesystem

9.4.1 Environment Configuration

NIDES depends on two environment variables and an IPC server process to execute. The following procedures should be done to ensure that these items are configured correctly.

9.4.1.1 NIDES C-shell File (`nides_init`)

Your NIDES installation includes a script file called `nides_init`. This file is located in the IDES_ROOT/etc directory. Remember that IDES_ROOT is the directory where your NIDES release was installed. Modify this file as follows:

1. Change _SOMEHOST_ to the NIDES hostname. You will see a line in the file that looks like this:

   ```
   setenv IPC_NAMESERVER _SOMEHOST_:7001
   ```

   Change the _SOMEHOST_ string to the name of the host on which you will run the NIDES `ipc_nameserver` process. The number following ':' is a port number; if necessary, it can be changed to any other non-privileged port. If you need more information, see Sections 6.1.4 and 7.14.

2. Set IDES_ROOT to the proper directory. Change the "setenv IDES_ROOT /ides" line by changing the "/ides" string to the location where NIDES is actually installed at your site.

3. Modify your .cshrc file to read the NIDES shell file nides_init by adding the following line to your .cshrc. Note that the `<IDES_ROOT>` path should be the same as the new path entered in step 2:

   ```
   source  <IDES_ROOT>/etc/nides_init
   ```

   This should be done for each user who will run NIDES.

4. Run `nides_init` script. Run the script so the environment assignments take effect immediately:

   ```
   %  source  <IDES_ROOT>/etc/nides_init
   ```

9.4.1.2 IPC Nameservices

The NIDES `ipc_nameserver` must be invoked for NIDES processes to communicate with each other. The program `ipc_nameserver` should be run on the NIDES host system. NIDES audit data providers (target hosts) do not need to have this process running. To invoke the proper IPC name services, type the following commands (note that you do not have to use the setenv command if you use the `nides_init` script):

```
% setenv IPC_NAMESERVER 'hostname':7001
% ipc_nameserver &
```

This has to be done only when no `ipc_nameserver` instance is running; even if NIDES programs are started and stopped many times, the same invocation of the `ipc_nameserver` can be used. The invocation and running of the `ipc_nameserver` is completely independent of NIDES programs; however, the `ipc_nameserver` must be running before any NIDES programs can run. If the `ipc_nameserver` is not running, NIDES programs will ask you to start it. Do not kill the `ipc_nameserver` while any NIDES programs are running.

9.4.2 NIDES Analysis Configuration

Your NIDES software comes with default configurations for the privileged user list and the rulebased and statistical analysis components. You should review these configurations and make modifications to suit your environment before running NIDES under real-time operation.

9.4.2.1 Privileged User List Configuration

The NIDES Beta-update release supports the concept of privileged and non-privileged users. You can configure the list of privileged users. A discussion of privileged user commands begins on page 103, and Table 6.1 shows which functions are privileged in NIDES.

The default privileged user list contains one user, ides. Update the list to include users who need access to privileged NIDES functions by following these steps:

1. Edit the `pusers` file, which is an ASCII file containing the list of privileged users, located in $IDES_ROOT/etc/pusers. Modify the file to contain all users who are privileged. You can put your list in a different file if you want to.

2. Process the `pusers` file using the `init_priv_user_list` utility, which is described in Section 7.12.

3. Update the `pusers` list as needed. You may add or delete users from the list at any time; edit the ASCII file, and then process it using `init_priv_user_list`. The privileged user list is read each time the `nides` program is started.

9.4.2.2 Rulebase Configuration

The NIDES rulebased analysis component includes a configuration file, `rb_config`, located in $IDES_ROOT/etc. Many rules in the default rulebase rely on the `rb_config` file to function properly. Section 5.4 discusses configuration of the `rb_config` file and describes default values. Of particular importance are the DOMAIN, HOME_DIR, LOG_DIR, PROGLOCATION, PROGRAM, and ROOT_OK options. Make a backup copy of the default `rb_config` file, and then modify it to match your environment.

The NIDES Beta-update release supports user development of expert system rules and activation/deactivation of rules during run time. You can write and install new rules at any time. However, if you are aware of some urgent vulnerabilities in your environment, review Table 6.16 and Section 5.5, which discuss the default rulebase. After reviewing the default configuration you can write additional rules, if needed, to address any known vulnerabilities in your system.

9.4.2.3 Protection of the NIDES Rulebase

The NIDES Beta release includes a default rulebase and `rb_config` file specific to the Sun UNIX environment. A copy of the source code for the NIDES default rules is also included with the release. The rule source files and `rb_config` file should be considered sensitive information. Protecting the rulebase files from unauthorized access ensures that potential intruders cannot reverse engineer the NIDES rulebase or tamper with the rulebase files.

This release of NIDES does not contain any built-in mechanisms for safeguarding the rulebase files, but the following procedures can help protect these files from unauthorized access and review:

- **Encryption of Rulebase Files Located in the NIDES Release Area** — The directory $IDES_ROOT/exsys contains the rulebase source, programs and scripts that are used to tailor the NIDES rulebase. We recommend that all files in this directory be encrypted when they are not needed. We also recommend that if you will be making additions or changes to the rulebase infrequently, you remove these files completely from the system and reload them when you need to make any changes.

- **Removal of Default Rulebase Source Code** — The file `rulebase.src` located in the $IDES_ROOT/exsys directory was provided with the NIDES release as an example of NIDES rulebase source code. The rules in the file have already been compiled for NIDES use, so the file is not needed for NIDES operation. We strongly suggest that you remove this file from the system immediately following installation of NIDES software. You may want to print copies of the file or put it on a floppy disk that can easily be loaded on a system for short-term review.

- **Encryption of Rule Object Files and `rb_config` File** — All the rule object files used by NIDES are stored under the $IDES_ROOT/etc/rulebase directory. We recommend that you encrypt all the files in this directory when you will not be running NIDES for an extended period. The rulebase configuration file, `rb_config`, is in the $IDES_ROOT/etc directory. We recommend that you encrypt this file when NIDES is not in use for an extended period. Be sure to decrypt these files before running NIDES.

9.4.2.4 Statistical Configuration

Under the Beta-update release of NIDES, most parameters pertaining to the statistical analysis component can be configured. Here we mention only the parameters that should be reviewed prior to real-time NIDES operation. Section 6.6.1.2 discusses the mechanics of analysis configuration using the NIDES user interface, Chapter 4 discusses statistical analysis configuration, and a discussion of the default statistical component configuration begins on page 164.

9.4.2.4.1 Installation of Default Statistics Configuration Configuration of the statistical component is usually done through the NIDES user interface. However, for initial configuration you may create an ASCII statistics configuration file (the default configuration is located in the file `ascii_stat_config` located in $IDES_ROOT/etc) and process it using the `init_stat_config` utility. To create and install a default statistics configuration for your environment, do the following:

1. Review the default statistics configuration and determine changes you need to make. The file `ascii_stat_config` contains the default configuration. Review Chapter 4, which discusses statistical analysis configuration, and the default statistical component configuration discussed on page 164.

2. Make a backup copy of the `ascii_stat_config` file and edit the file to suit your needs. Note that your default configuration file can have any name; you do not need to use the `ascii_stat_config` file.

3. Make a backup copy of the binary statistics configuration file, `stat_config`, which is located in $IDES_ROOT/etc.

4. Process the ASCII configuration file using the `init_stat_config` utility. Be sure to use the *-etc* flag to ensure that your new `stat_config` file is created in the $IDES_ROOT/etc directory.

9.4.2.4.2 Statistics Configuration File (ascii_stat_config) The ASCII statistical configuration file, `ascii_stat_config` on the NIDES release tape, should be modified as part of the installation procedure. It contains three sections — for parameters, for command classes, and for measures. The syntax of the file is as follows: comment lines are any lines that begin with a "#" character; each section starts with a `BEGINxxx` line and ends with an `ENDxxx` line. For example, the parameters section begins with a line `BEGINPARAMS` and ends with `ENDPARAMS`, the command classes section begins with `BEGINCOMMANDCLASSES` and ends with `ENDCOMMANDCLASSES`, and the measures section begins with `BEGINMEASURES` and ends with `ENDMEASURES`. In between the `BEGINxxx` and `ENDxxx` declarations are lines containing configuration definitions.

Parameters The parameters section controls such items as a half-life, score thresholds, and training periods. For a discussion of the meaning of these fields, refer to Section 4.5. The format for the parameters section is

```
BEGINPARAMS
AR_HALFLIFE=100.0
PROF_HALFLIFE=20.0
CORR_CUTOFF=99.0
MIN_EFFN=100.0
YELLOW_PERC=0.01
RED_PERC=0.001
TRAINING_DAYS=20
MAXSUMRARE=0.01
NO_UPDATE_MODE=
ENDPARAMS
```

The default values listed in the parameters section should be suitable for most installations.

Command Classes The command classes section contains class member lists (see Section 4.4). Seven statistical classes are used to determine measure categories, and the TMPDIRS class list is used to filter out uninteresting files. Seven of the classes have default membership lists (COMPILER, EDITOR, MAILER, SHELL, WINDOW, NETWORK, and TMPDIRS). The MISC class list is not used in this release of NIDES. The LOCALHOSTS class has no members under the default configuration. The LOCALHOSTS class should list all hosts you consider local, and any host not listed will be categorized as remote by the statistics. All these classes are likely to be installation-dependent to some degree; you should review the membership lists of the statistical classes and modify them to describe your environment, as appropriate, during NIDES installation. The format for the class section is

```
BEGINCOMMANDCLASSES
COMPILER=gcc,cc,g++,f77,yacc,bison,m4
EDITOR=emacs,vi,ed,edit
MAILER=mm,mail,mh
SHELL=sh,csh,tcsh
WINDOW=X,Xsun,suntools,xcalc,cmdtool
NETWORK=rsh,ftp,kermit,rcp,rdist
MISC=
LOCALHOSTS=
TMPDIRS=/tmp,  /var/tmp
ENDCOMMANDCLASSES
```

For each class list, the file contains a line with the list mnemonic name (one of `COMPILER`, `EDITOR`, `MAILER`, `SHELL`, `WINDOW`, `NETWORK`, `MISC`, `LOCALHOSTS`, or `TMPDIRS`), an `=` sign, and the class members separated by commas. All members of a class must be listed on one line, followed by a carriage return. A carriage return in the middle of a list is not allowed. The maximum line length is 1000 characters.

Measures The final section in the ASCII statistics configuration file describes measure configurations (see Section 4.6). The format of this section is

```
BEGINMEASURES
U_CPU         ON   CONT    100.0  0.0   1000.0      User_CPU_Usage
U_IO          ON   CONT    100.0  0.0   10000000.0  User_I/O_Usage
U_MEM         ON   CONT    100.0  0.0   10000000.0  User_Memory_Usage
U_LOC         OFF  CAT     100.0  0.0   0.0         User_Physical_Location_of_Use
...
...
U_ARECDIST    ON  CAT     100.0   0.0   0.0         User_Audit_Rec_Distribution
U_INT60       ON  CONT    100.0   0.0   0.0         User_AudRec_Intensity_60
U_INT600      ON  CONT    100.0   0.0   0.0         User_AudRec_Intensity_600
U_INT3600     ON  CONT    200.0   0.0   0.0         User_AudRec_Int
ENDMEASURES
```

All measures must be listed in the file. There must be one line for each measure, which contains seven entries in the following order:

1. Measure ID. You should not change this entry.

2. Status. There are two valid values — `ON` and `OFF`. See Section 4.6.1.
3. Measure Type. You should not change this entry. There are three possible values (see Section 4.1.1 for a description of measure types):
   - CAT — categorical measure
   - CONT — continuous measure
   - BINCONT — binary measure
4. QMAX. This is a floating point number. See Section 4.6.3.
5. Weight. This field is not used under this NIDES release.
6. Scalar. This is a floating point number. See Section 4.6.2.
7. Measure Description. You should not change this entry.

Measure Scalar is applicable only to continuous measures (it is set to 0.0 for the categorical measures (e.g., `U_IO` above). Generally, it should not be necessary to change any of the measure parameters for your initial installation except possibly the measure status (`ON/OFF`).

9.4.2.4.3 Statistical Component Performance Parameters Two configuration items affect the performance of the statistical analysis component: profile cache and temporary file class.

- **Profile Cache** — The size of the profile cache determines how many profiles the statistical analysis component will store internally while processing audit data. The profile cache cannot be configured in the `ascii_stat_config` file; it must be configured via the NIDES user interface (see Sections 4.5.5 and 6.6.1.2.3).
- **Temporary File Class** — This class list determines files the statistical analysis considers uninteresting, and for which, therefore, it will not create categories. This is an important configuration item. It not only affects the speed and size of the statistical process, but also improves detection ability by preventing subject profiles from being diluted with uninteresting information.

9.5 Target Host Installation and Configuration

To run NIDES on "live" audit data, you must configure additional computers (target hosts) to serve as audit data providers to NIDES. Configuring NIDES target hosts requires root access, so this should be performed by your system administrator.

9.5.1 Audit Data Sources

NIDES relies upon existing, platform-specific audit systems for the generation of native audit data as well as user-defined data sources. This audit data is converted into canonical NIDES audit records for analysis.

The default NIDES configuration supports three different native audit systems — SunOS BSM, SunOS C2, and standard UNIX accounting. A feature, new to this release of NIDES, provides a facility for tailoring NIDES to use additional sources of audit data. See Chapter 8 for a complete description of this facility.

Two conflicting principles govern the monitoring requirements of the target systems that are to be analyzed by NIDES:

- The finer the resolution of the generated audit data, the greater the ability to detect unauthorized behavior.

- The greater the volume of the generated audit data, the greater the likelihood that computational and/or storage resources will be adversely impacted.

In particular, both the SunOS BSM and SunOS C2 audit systems can be configured to report *all* file accesses, creations, deletions, and modifications, thus providing a very fine-grained view of user activity. At the same time, the volume of audit data generated with such a configuration can adversely affect the responsiveness of the target system itself (independent of NIDES) as well as the responsiveness of NIDES.

On the other hand, the standard UNIX accounting data, which was designed primarily for charging users based upon CPU, I/O, and memory utilization, provides only minimal information about the programs invoked by individual users, and thus provides only a coarse-grained view of user activity.

An assessment must be made to determine the types of unauthorized behavior that NIDES is to detect. For example, the requirement that all attempts to browse unauthorized files (even if the file permissions permit such access) be detected will require a different configuration (and generate *many more* records) than the requirement to detect all attempts to write the same unauthorized files. With the introduction of the audit data customization facility, users may be able to reduce the need for SunOS C2 or BSM data by supplementing or replacing the data with other sources.

9.5.1.1 Practical Experience

An eight-day test, with 10 workstations running C2 auditing with all audit flags on, was conducted at SRI International in April 1993. Out of a total of 10,296,854 records generated by the SunOS C2 audit system reporting all event types, 92% reported read access to files. Moreover, more than 99% of these read accesses were "legitimate", which suggests that, depending upon the monitoring requirements, the utility of collecting "read" access to files is of marginal utility. Furthermore, the inherent overhead of the SunOS C2 audit system reduced the responsiveness of some diskless workstations by more than a factor of 2 — primarily because the diskless systems write their audit trails across the network file system (NFS).

No similar experiment has been conducted with SunOS BSM, but a similar result is expected based upon the underlying similarity of the SunOS C2 and SunOS BSM auditing systems.

9.5.1.2 Recommended Configuration for SunOS BSM Auditing

The SunOS BSM audit system classifies users' actions into 12 basic categories. The resolution of auditing can be controlled by setting default system flags as well as individual user flags indicating the categories of events to report. Care should be taken in determining the flags that are required in order to provide the desired level of target system monitoring. Consult the SunOS BSM installation and configuration procedures for more detail.

A recommended starting configuration is to indicate all flags *except* read access to files given by

```
dw,dc,da,lo,ad,p0,p1,ex,nt,io,other
```

If it is necessary to collect read audit records, limiting the reporting to failed reads should be considered; failure is typically linked to permission denial or, more often, an attempt to read a non-existent file. The flags to report all events *except* successful read access to files is given by

```
-dr,dw,dc,da,lo,ad,p0,p1,ex,nt,io,other
```

9.5.1.3 Recommended Configuration for SunOS C2 Auditing

The SunOS C2 audit system classifies users'actions into eight basic categories. The resolution of auditing can be controlled by setting default system flags as well as individual user flags indicating the categories of events to report. Care should be taken in determining the flags that are required in order to provide the desired level of target system monitoring. The SunOS C2 installation and configuration procedures should be consulted for more detail.

A recommended starting configuration is to indicate all flags *except* read access to files given by

```
dw,dc,da,lo,ad,p0,p1
```

If it is necessary to collect read audit records, limiting the reporting to failed reads should be considered; failure is typically linked to permission denial or, more often, an attempt to read a non-existent file. The flags to report all events *except* successful read access to files is given by

```
-dr,dw,dc,da,lo,ad,p0,p1
```

A major drawback of the event classification system in SunOS C2 is that the flag for read access to files is required to provide program execution events. This deficiency can be overcome, to a certain extent, by using standard UNIX accounting in conjunction with SunOS C2 if read access to files is deemed to be unwarranted.

9.5.1.4 Standard UNIX Accounting Data

The standard UNIX accounting system generates records only when program execution has terminated. Unlike both SunOS BSM and SunOS C2, the UNIX accounting audit records give information about program resource utilization that is of particular use in the statistical component.

In general, all target systems running UNIX should generate standard UNIX accounting data. If neither SunOS BSM nor SunOS C2 is active, UNIX accounting is a bare minimum. Even if either SunOS BSM or SunOS C2 is active, UNIX accounting provides useful information that the other systems do not.

9.5.2 Selecting a NIDES Software Directory on Each Target

The NIDES target host software must be installed on a file system that will always be available for the daemons that run on the target hosts. You will probably want to install the software in the "/usr" hierarchy for each host. If different hosts use different "/usr" hierarchies, you will need to install the software on each hierarchy. If the target hosts all use the same hierarchy, you will need to install the target host software only once for each audit data configuration.

Select a directory where the NIDES target code is to be installed. Here, we will assume "/usr/ides" is selected. Note that this must be different from the location of the NIDES distribution hierarchy. Make sure this directory does *not* already exist.

9.5.3 Configuring Target Hosts

The NIDES Beta-update release includes a new facility that allows users to customize their audit data sources. With this new facility NIDES can monitor a mixture of SunOS hosts running the standard SunOS C2 or BSM auditing and UNIX accounting, and hosts which collect other types of audit data. Users may even collect additional types of data on SunOS hosts running C2/BSM and include that data in the NIDES analysis. To configure a Sun Target Host that will provide only C2/BSM and/or UNIX accounting data, follow the procedures for target host installation described in Section 9.5.3.1. To configure target hosts that will utilize the new audit data customization facility, follow the procedures described in Section 9.5.3.2.

9.5.3.1 Configuration of Sun C2/BSM and UNIX Accounting Target Hosts

Perform the following steps to configure each target host that will utilize only SunOS C2/BSM and UNIX accounting data:

1. Log on to the target host as root.

2. Set your IDES_ROOT environment variable to the NIDES distribution directory:

   ```
   % setenv IDES_ROOT __SOMEWHERE__
   ```

   If your machines do not share a common file system, the file system containing the NIDES distribution should be NFS mounted. Root on the target host system should have root access to that file system. If you cannot NFS mount the NIDES software hierarchy, use ftp (in binary mode) to transfer the followings files to your target host's NIDES directory:

   - $IDES_ROOT/bin/bin.sun4/agend
   - $IDES_ROOT/bin/bin.sun4/agen
   - $IDES_ROOT/bin/bin.sun4/rc.ides-target

3. Install the NIDES target host software on the target computer by issuing the following command:

   ```
   % $IDES_ROOT/bin/bin.sun4/install_ides_target /usr/ides
   ```

   Here "/usr/ides" is the directory where the target host software will be located.

4. Update the target host's *rc.local* file by adding the following line to the end of /etc/rc.local. This will automatically start the NIDES target host daemon the next time the target host is rebooted:

   ```
   if [ -f /usr/ides/bin/rc.ides-target ]; then
           /usr/ides/bin/rc.ides-target
   fi
   ```

5. Edit /usr/ides/bin/rc.ides-target, changing

   ```
   ides_root="/ides"
   ```

   to

   ```
   ides_root="/usr/ides"
   ```

9.5.3.2 Configuration of Non-Sun C2/BSM Target Hosts

Before configuring any target hosts that will use the new audit data customization facility, review Chapter 8. Perform the following steps to configure target hosts that will use the new audit data customization facility:

1. Log on to the target host as root.
2. Set your IDES_ROOT environment variable to the NIDES distribution directory:

   ```
   % setenv IDES_ROOT __SOMEWHERE__
   ```

   If your machines do not share a common file system, the file system containing the NIDES distribution should be NFS mounted. Root on the target host system should have root access to that file system. If you cannot NFS mount the NIDES software hierarchy, use ftp (in binary mode) to transfer the followings files to your target host's NIDES directory:

   - $IDES_ROOT/bin/bin.sun4/agend
   - $IDES_ROOT/bin/bin.sun4/agen
   - $IDES_ROOT/bin/bin.sun4/rc.ides-target

3. Install the NIDES target host software on the target computer by issuing the following command:

   ```
   % $IDES_ROOT/bin/bin.sun4/install_ides_target /usr/ides
   ```

   Here "/usr/ides" is the directory where the target host software will be located.

   Now the compiled default versions of `agen` and `agend` have been installed in your target host's NIDES area. A shell script `rc.ides-target` is now installed as well.

4. Determine which data sources you will use with this target host and write Perl script translation routines for each source. Be sure to put these scripts in your $IDES_ROOT/bin/bin.perl directory. Chapter 8 describes this process in detail.

5. Install all Perl customization files on your target host by copying all Perl script files in your $IDES_ROOT/bin/bin.perl directory into your target host's NIDES directory:

    ```
    %cp -p $IDES_ROOT/bin/bin.perl/* /usr/ides
    ```

6. Move the default NIDES `agen` executable to back up a file and replace it with the Perl version.

    ```
    %mv /usr/ides/agen /usr/ides/agen-compiled
    %ln -s /usr/ides/agen.pl /usr/ides/agen
    ```

    If your target host is not running SunOS 4.1.x you will also need to back up the default NIDES `agend` executable because you must use the Perl version of `agend` with any non-SunOS 4.1.x target hosts.

    ```
    %mv /usr/ides/agend /usr/ides/agend-compiled
    %ln -s /usr/ides/agend.pl /usr/ides/agend
    ```

7. Verify that your Perl `custom.pl` file lists all the audit data sources you intend to use on your target host and that all referenced Perl files are contained in your target host's NIDES directory.

8. Update the target host's *rc.local* file by adding the following line to the end of /etc/rc.local. This will automatically start the NIDES target host daemon the next time the target host is rebooted:

    ```
    if [ -f /usr/ides/bin/rc.ides-target ]; then
          /usr/ides/bin/rc.ides-target
    fi
    ```

9. Edit /usr/ides/bin/rc.ides-target, changing

    ```
    ides_root="/ides"
    ```

    to

    ```
    ides_root="/usr/ides"
    ```

9.5.4 Starting the NIDES Target Daemon

After you have configured your target host, start the NIDES target daemon by issuing the following command:

```
% /usr/ides/bin/rc.ides-target
```

9.5.5 Additional Target Hosts

Repeat the target host installation process for every host computer that will provide audit data to NIDES. With the new audit data customization facility, different target host platforms may concurrently provide data of different types to NIDES.

9.6 Starting and Running NIDES

After you have configured your NIDES host system and target host (audit data provider) systems, you can run NIDES.

9.6.1 X windows

You will need to run NIDES under X windows version X11R4 or X11R5 (we recommend X11R5). If you are not set up to run X windows, have your system administrator help you. We recommend using the *twm* window manager, which is supplied with X windows.

Your system administrator must make available to NIDES the X11 files used by the NIDES user interface. NIDES will look for the standard X directories under /usr/X11R5. If your X system is installed under another directory hierarchy, you must create a link to that directory from /usr/X11R5. You will need the following three directories under your X directory: bin, include, and lib. If these directories for the standard X distribution are not located under the same directory, you must to create three links instead of one — /usr/X11R5/bin, /usr/X11R5/include, and /usr/X11R5/lib.

9.6.2 Final Configuration Checks

Make sure that your shell environment has been correctly set up, using the `nides_init` script. This is done automatically if you include execution of this file in your .cshrc file and you have created a new shell window after modification of your .cshrc file. If in doubt, execute the `nides_init` script.

The `ipc_nameserver` process must be running on your NIDES host computer. If it has not already been started, start it by issuing the following command on the NIDES host:

```
% $IDES_ROOT/bin/bin.sun4/ipc_nameserver &
```

You should have only one `ipc_nameserver` process running on your NIDES host computer. NIDES target host computers do *not* need to have this process running on them.

9.6.3 Starting Up NIDES

Log in to the NIDES host as a user who is a member of the ides group. NIDES users should be in the ides group in order to run NIDES.

Since the NIDES user interface is X-windows based, you need to have X windows running and a window manager invoked.

Once you have completed the configuration steps outlined above and have configured all NIDES target hosts, type `nides` at the system prompt, as shown below, and the NIDES user interface will start. You must do this from an xterm window that is connected to the NIDES host computer:

```
% nides
```

After `nides` has successfully started, consult Chapters 3 and 6 for information on the NIDES user interface.

Glossary

Accounting Audit Data The standard UNIX accounting system. Designed primarily for keeping track of resource utilization (e.g., connection time, CPU usage) for billing purposes. The accounting records generated are of minimal utility when other forms of audit data are available (e.g., C2 or BSM).

Activity Intensity Measure A group of measures that capture intensity of activity measured in rate of arrival of audit records. Three measures track intensity over the last minute, ten minutes, and hour, comparing the rates observed in real time to the rates as learned in the profile. These are intended to detect intrusions that flood the system with audit records.

Activity Vector Each time the NIDES Statistical Analysis component analyzes an audit record, the first processing step is the construction of an activity vector. This vector of observed measure values (at most one per NIDES measure) is obtained by processing the data contained in the NIDES audit record. For every measure represented in the audit record, the associated audit data is converted to a continuous or categorical value, depending on the type of measure, and placed in the activity vector entry for the measure.

Adset A mnemonic term for Audit Data Set. See Audit Data Set.

Aging Factor The factor by which past data is multiplied so as to fade its value at a desired rate. For a half-life of k audit records, for example, the factor is set at the kth root of 1/2, so that after k steps the data are faded to one-half of their original contribution. Storing profiles as aged cumulative totals permits relatively compact profile structures and allows the system to adapt to changes in subject behavior. NIDES has a short-term aging factor applied to each audit record and a long-term aging factor applied to daily totals at update time.

Agen One of the core NIDES processes. A single `agen` process runs on each of the actively monitored target hosts, translating all the supported, native audit data into canonical NIDES audit records, and providing then to the `arpool` process. The UNIX version of the agen process currently supports three native audit record formats: SunOS BSM version 1, SunOS C2, and standard UNIX accounting.

Alert NIDES has two analysis components that process 'audit data and determine if a suspicious event has occurred - rulebased and statistics. A resolver component takes the results of the rulebased and statistical analysis and determines if an alert should be reported. Currently, the resolver reports all rulebased results that are critical as alerts.

For the statistical analysis, when the T2 score as of the current audit record exceeds a declaration (red or critical) threshold and the previous audit record did not exceed the threshold, an alert is reported. The threshold is set to achieve a nominal false positive rate (user configurable, 0.1% by default). As the statistical analysis employs a short-term memory of recent activity, an alert occurs on the record that nudges the score above the threshold, but the alert should be considered as reflecting a sequence of unusual activity in the recent past. If subsequent audit records keep the statistical score above the threshold, additional alerts are not reported unless the top (most significant) measure that contributed to the score changes.

Antecedent See Rule Antecedent.

Arpool One of the core NIDES processes. The `arpool` process accepts canonical NIDES audit records from the `agen` process on all the actively monitored target hosts and presents the audit records as a single data stream to the analysis components of NIDES.

Archiver One of the core NIDES processes. The `archiver` process accepts canonical NIDES audit records from the `arpool` process and stores them on disk, in a compressed format, to facilitate future reference when investigating activity that generated alerts.

Audit Data Set A source of NIDES audit records, generally used as input to run NIDES experiments using the test facility. An audit data set can be either *real* or *virtual*. A real audit data set consists of a single UNIX file (usually compressed) containing NIDES audit records. A virtual audit data set consists of parameters used to select audit data from an audit data archive; the audit data is retrieved from the specified audit data archive at the time a test is run.

Audit Record Distribution Measure A special measure whose categories are the names of all other measures and which tracks the number of times the respective measures are touched in the short-term profile. Its purpose is to assess the normalcy of the distribution of the user's recent activity across the measures.

Audit Record Half-life See short-term half-life.

Bin Table entry to which an observed value is assigned. For categorical measures, such as ERRTYP, there is a one-to-one correspondence between bins and observed category values. For continuous measures there are 32 bins which correspond to value ranges.

Binary Measure A group of measures that track whether or not a given type of activity is observed in the current audit record. Binary measures are used as a mechanism to maintain counts in the audit record distribution measure and do not directly affect the score.

BSM The most recent auditing system developed for SunOS. The BSM (Basic Security Module) generates audit records derived from low-level UNIX activity (e.g., reading, writing, assessing, or deleting a file, changing directory, running a program).

Categorical Measure A measure that assumes values in discrete categories. For some such measures, such as HOUR, the values are known beforehand (the hours 0, 1, 2, ..., 23). For others, new categories are allocated by NIDES as they are encountered.

Category An observed value (such as error type or hour of use on a 24-hour clock) for categorical measures, or a value range for a continuous measure such as CPU. By logarithmically recoding the ranges of continuous measures, NIDES in fact treats all measures as "categorical".

Class A list of commands or objects belonging to the same "class" of activity (e.g., compilers, editors, or mail commands). Classes are used by the statistical analysis component to determine categories for class measures. The classes used in NIDES are: compilers, editors, mail programs, shell environments, window commands, network commands, local hosts, and temporary file directories.

Class Measure A measure with a predefined set of categories that captures a given class of computer activity. For example, the compiler measure has as its predefined categories the various compilers available on the system. The profile for this measure tracks the percent of compiler usage attributable to each compiler. This is useful because, for example, compiler usage may comprise a relatively small percentage of total command usage (and hence be somewhat diluted in the command usage measure) but may be especially interesting with respect to intrusion detection.

Consequent See Rule Consequent.

Continuous Measure A measure that takes continuous values, such as CPU in time units.

Cross-profiling An experiment in which data for each subject is tested against the trained profile for each other subject. Long-term profile update is disabled for such experiments.

C2 An older, now obsolete, auditing system developed for SunOS. C2 generates audit records derived from low-level UNIX activity (e.g., reading, writing, assessing, or deleting a file, changing directory, running a program). Its name is derived from a specific security rating described in the "Orange Book".It should not be confused with the generic computer security rating of C2.

Detection/Detection Rate A declaration by NIDES that a stream of audit data contains anomalous activity, which can be at a yellow (caution) or red (critical) threshold. Detection rate is the percent of audit records in a given audit data stream that trigger detections.

Effective n The effective length of the short-term profile, which equals the series sum of all powers of the aging factor (or approximately 1.5 times the short-term half-life). This can be thought of as the number of audit records that, after aging, still make a contribution to the short-term profile.

Experiment See Test.

Fact The NIDES rulebased component stores transitory information needed for its analysis in facts. Facts are stored in a database (see Factbase) internal to the rulebased component. The rulebase can define many different kinds of facts. The structures for facts are defined by ptype declarations. Facts are asserted (added) and removed from the internal database by rules during runtime.

Factbase A database of transitory information (See Fact) created, used, and maintained by the NIDES rulebased analysis component. Multiple facts of the same type can be contained in the factbase. If a rule searches the factbase for a fact type that contains multiple entries, the most recently asserted fact matching the rule search specification will be returned to the rule.

False-positive A detection, by the statistical analysis component, for a subject against its own profile.

Half-life The number of audit records (in the case of the short-term profile) or the number of profile updates (in the case of the historical profile) by which time the contribution of a data item to the present cumulative totals is reduced by one half.

Historical effective n The effective count of audit records contributing to the long-term profile. It consists of the sum of all daily totals each weighted by the appropriate power of the long-term aging factor. This value can be thought of as the number of audit records that, after aging, still contribute to the long-term profile.

Historical Profile See Long-term Profile.

IDES_ROOT The NIDES environment variable that determines the directory where the NIDES software resides. This variable must be set prior to running any NIDES software.

Instance An analysis configuration, and the set of profiles associated with that configuration.

Intensity Measure See Activity Intensity Measure.

Inter-arrival Time The difference in timestamps between successive audit records for the same subject. Used by the statistical analysis to monitor intensity (rate of activity in l-minute, lo-minute, and 60-minute windows) and thereby potentially detect an intrusion that floods the system with audit records.

Long-term Half-life That time interval (measured in profile updates) by which time the contribution of a given data item in the long-term profile is "aged out" by a factor of one-half. The system default is 20 updates (one month of nonweekend days), configurable by the user.

Long-term Profile For each subject and measure, the observed categories and the observed long-term probabilities for each category, the historical effective n, and the empirical Q distributions. For the subject there is also an empirical score (T2) distribution, which is aggregated across all measures. At the end of each day, this profile is aged by the long-term aging factor and combined with the new daily totals.

Max **Sum of Rare Category Probabilities (Max Sum Rare Prob)** A configurable constant that represents the maximum sum of probabilities of categories classified as rare. Categories are sorted in ascending order of probability and then summed to the largest index for which the sum is less than or equal to this constant. All categories up to and including this index are classified as rare until the next update interval. For numerical stability, this value should be between 0.01 and 0.05 .

Measure A measure is an aspect of subject behavior. This is the unit used by the statistical analysis component of NIDES. The measure is used to monitor activity on a particular dimension of subject behavior. Measure types are continuous (such as CPU in seconds on the present audit record), categorical (such as file name), intensity (rate of arrival of audit records in various time windows), and a special audit record distribution measure to monitor recent types of activity. A single audit record can generate observed values for more than one measure.

Minimum effective n The minimum count of records in the long-term profile that must be accumulated before the scoring mechanism is considered reliable. It is measure-specific.

Native Audit Record An audit record specific to a given auditing system. Native audit records are converted by the agen process into a canonical NIDES audit record format for analysis and storage. Once the audit data are converted, NIDES no longer makes use of a native audit record. The UNIX version of the agen process currently supports three native audit record formats: Sun OS BSM version 1, Sun OS C2, and standard UNIX accounting.

NIDES Audit Record A canonical audit record format capable of representing all supported native audit record information. NIDES audit records are used for analysis and storage. Once the audit data are converted, NIDES no longer makes use of a native audit record.

Orange Book The common name of a document describing different levels of computer security ratings and the associated requirements.

Perl A UNIX shell script language. Perl stands for *Practical Extraction and Report Language.* For a complete description of Perl see [4, 5].

Persistent Storage NIDES maintains databases of many types under its normal operation. These databases include an audit record archive, analysis result archive, instances (user profiles and analysis configuration data) and miscellaneous configuration files (e.g., privileged user lists). All of these databases and files are part of the NIDES persistent storage facility. The persistent storage facility provides a set of library functions to all NIDES components, allowing them to read and write data to the various databases and configuration files.

Profile The statistical analysis component of NIDES generates a profile of behavior for each subject it sees in the audit data stream. The profile is comprised of two parts, a long-term profile and a short-term profile. The long-term profile contains the category probabilities, aged counts, system thresholds, and so forth for each subject, aged with a long-term half-life on the order of several weeks (set to achieve a trade-off between stability and adaptability to new behavior). The short-term profile contains the observed categories and aged counts in the recent past, aged with a short-term half-life of tens to hundreds of audit records (representing minutes to tens of minutes of activity). For computational efficiency, the short-term profile maintains aged counts, while the long-term profile maintains probabilities that do not change between updates.

Profile Snapshot An instantaneous view of the profile available immediately after an update or when a profile is swapped out of the profile cache and into persistent storage. The NIDES profile viewing utilities show the most recent snapshot.

Profile Synchronization A means of adjusting time stamps in experimental data sets that enables updating to take place in the test facility even when the time stamps in the audit data set are earlier than the last update time stamp in the profile.

Profile Training The general procedure of updating profiles, adding and dropping categories, and adjusting the empirical distributions for Q and T2. It proceeds in three stages. In the first, category probabilities are obtained from a number of days of raw data. In the second, the Q distribution is estimated over an additional number of days. Finally, the T2 distribution is estimated, after which time NIDES is ready to score audit records. In a production environment, profile training continues indefinitely. For experimentation with known masquerader data, profile updating and training are disabled.

Profile update The merging of the historical profile with new information at the end of each day. Long-term probabilities are converted to effective counts (by multiplying by the historical effective n). The new daily counts are summed in, and the results converted back to probabilities. Categories that have too low a probability are folded into a RARE category, which can change daily.

ptype A declaration that defines the structure of facts that are created and stored in the NIDES rulebased component's fact base. A ptype declaration is similar in concept to a structure declaration in C. An example of a ptype declaration is

```
ptype[event  subject:string,
    action:string,
    object:string,
    time:int]
```

Here the structure for the *event* ptype is defined to contain four fields: subject, action and object are strings, and time is an integer. Using this ptype, facts of type *event* can be added to or removed from the NIDES rulebased component's factbase.

Q-score A chi-square-like square difference statistic based on the difference between the short- and long-term profiles for each measure.

QMax A scale value used to assign the Q-score into bins to obtain its empirical distribution.

Rare Probability A configurable system constant (default 0.01 or 1%) used for collapsing categories into a RARE class (which are scored by NIDES as a group rather than as individual categories). Categories whose cumulative sum is less than this constant are tagged as RARE in a given update.

Red/Critical threshold That value which, when exceeded by the T2 score, causes NIDES to issue a red or critical result from the statistical analysis. It is configurable (default of 0.1% seeks to achieve a false positive rate of 0.1% on normal data).

Remote Procedure Call (RPC) An action in which a process calls a procedure that is executed by another process. The NIDES architecture is composed of many processes that communicate via RPCs. For example, when the NIDES analysis components (statistical and rulebased) need an audit record to analyze, both components make an RPC to the arpool process to ask for the next audit record; the arpool process makes an RPC in the form of a response providing an audit record to the analysis processes.

Resolver The NIDES analysis process that receives results from the statistical and rulebased analysis components and determines if an alarm should be reported.

Result A result is generated for every audit record processed by the NIDES analysis components. Results are categorized into three levels: safe, warning, and critical. The level of a result is assigned by the resolver component based on the levels assigned by the statistical and rulebased analysis components. An NIDES alert is reported when the resolver determines that a critical-level result should be assigned alert status.

Rule Antecedent The first part of the two parts that comprise the body of a NIDES rule. The antecedent contains the tests that are performed on the rulebase's factbase to determine if a particular condition is met. If the condition is met, the second part of the rule, the consequent, is executed.

Rule Consequent The second part of the two parts that comprise the body of a NIDES rule. The consequent contains a set of actions that are performed if the tests performed in the rule's antecedent are satisfied. If the consequent actions are executed, the rule is said to have "fired". Actions that may be performed in the consequent of a rule include additions or deletions to the rulebase's factbase and generation of an alert report.

Rule Priority A priority assigned to the NIDES rulebased component rules when they are written. The priority determines the order in which rules are tested. Rules with higher priorities are tested first. Higher numbers equate to a higher priority (e.g., a priority of 5 is higher than a priority of 1).

S-value A unitless quantity obtained by inverting the observed Q-score using the Q empirical distribution and a half-normal transform. This results in all measure scores being comparably distributed.

Scalar A value used to scale observed (raw) values to assign them to category (range) bins.

Score The multivariate aggregate statistic on which the statistical analysis bases anomaly detection. Up to various normalizations, it is proportional to the sum of squares of the S values. Also called the T2 score.

Sequence Number Numbers assigned by the NIDES `agen` and `arpool` processes to the audit records processed by NIDES. Two sequence numbers are assigned to each audit record. The `agen` process assigns a target host sequence number that is unique for the duration of the current `agen` process execution on the target host. This number is referred to as the *target sequence number.* The `arpool` process assigns a sequence number to all audit records it receives; this number is unique across all NIDES target hosts and monotonically increases for the duration of the current `arpool` process. This number, referred to as the *audit record sequence number,* is used to identify the audit record when alerts are reported by NIDES. When `arpool` is first started it begins with a sequence number of `0`.

Short-term Half-life See Half-life.

Short-term profile For each subject and measure, the number of counts recently observed for each category in the long-term profile with special handling for new categories. Due to the aging procedure, these counts are generally fractional.

Short-term Profile Length The effective number of audit records in the short-term profile. It is approximately 1.4 times the short-term half-life.

Subject The entity for which NIDES maintains profiles and performs anomaly detection. In the NIDES paradigm, the subject (e.g., a user of the system) initiates actions (e.g., file copy) that act on objects (e.g., files).

Subject Profile See Profile.

Target Host A host computer that is monitored (or can be monitored) by NIDES.

Test A batch run of NIDES with archived data, typically done to examine the impact of parameter changes or establish detection rates

Threshold The NIDES-estimated value for T2 at which a detection is declared. It is set to achieve no greater than some user-specified percent (usually 1% for yellow, 0.1% for red) of false positives.

Training The process by which the NIDES statistical component learns normal activity for a subject. It consists of category training (wherein the system learns the observed categories for each measure), Q training (wherein the system builds an empirical distribution for the Q statistic, which measures the measure-by-measure difference between the long- and short-term profiles), and T2 training (wherein the system establishes the threshold for the measure statistic, which is collected across all active measures). All three phases have a minimum training period before anomaly scoring begins. Training continues in the steady state, permitting a degree of adaptation to new subject behavior.

Training Status The status of a measure with respect to the three training phases (see Training). A measure can be trained (ready to contribute to scoring) or under one of the three phases.

Training Period The length of time (measured in number of profile updates) before measures may contribute to anomaly scoring. It is user configurable. A number of updates equal to one third this quantity (rounding any fraction upward to the next integer) is required before a measure exits each of the three training phases (see Training).

True-positive A detection for a subject (possibly a masquerader) against another subject's profile.

T2 The overall NIDES statistical analysis score on which anomalies are declared, aggregated across all measures. (See Score)

Yellow/Warning threshold That value which, when exceeded by the T2 score, causes NIDES to issue a yellow or warning alert from the statistical analysis. It is configurable (default of 1.0% seeks to achieve a false positive rate of 1.0% on normal data).

Index

Bibliography

[1] Debra Anderson, Thane Frivold, Ann Tamaru, and Alfonso Valdes. Next generation intrusion detection expert system (NIDES): Software design document and version description document. Document A002 and A005, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, July 1994.

[2] Harold S. Javitz and Alfonso Valdes. The NIDES statistical component description and justification. Annual report, SRI International, Menlo Park, CA, March 1994.

[3] Debra Anderson, Teresa Lunt, Harold Javitz, Ann Tamaru, and Alfonso Valdes. Safeguard final report: Detecting unusual program behavior using the NIDES statistical component. Final report, SRI International, Menlo Park, CA, December 1993.

[4] Randal L. Schwartz. *UNIX Programming: Learning Perl.* O'Reilly & Associates, Inc., 632 Petaluma Avenue, Sebastopol, CA 95472, 1993.

[5] Larry Wall and Randal L. Schwartz. *UNIX Programming: Programming perl.* O'Reilly & Associates, Inc., 632 Petaluma Avenue, Sebastopol, CA 95472, 1991.