SRI International

Next-generation Intrusion Detection Expert System (NIDES) A Summary[1]

Debra Anderson
Thane Frivold
Alfonso Valdes

Computer Science Laboratory

SRI-CSL-95-07, May 1995

333 Ravenswood Avenue • Menlo Park, CA 94025-3493 • (415) 326-6200 • FAX: (415) 326-5512 • Telex: 334486

[1]This report was prepared for the Department of the Navy, Space and Naval Warfare Systems Command, under Contract N00039-92-C-0015

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Existing security mechanisms protect computers and networks from unauthorized use through access controls, such as passwords. However, if these access controls are compromised or can be bypassed, an abuser may gain unauthorized access and thus can cause great damage and disruption to system operation.

Although a computer system's primary defense is its access controls, it is clear from numerous newspaper accounts of break-ins, viruses, and computerized thefts that we cannot rely on access control mechanisms in every case to safeguard against a penetration or insider attack. Even the most secure systems are vulnerable to abuse by insiders who misuse their privileges, and audit trails may be the only means of detecting authorized but abusive user activity.

Other modes of protection can be devised, however. An intruder is likely to exhibit a behavior pattern that differs markedly from that of a legitimate user. An intruder masquerading as a legitimate user can be detected through observation of this statistically unusual behavior. This idea is the basis for enhancing system security by monitoring system activity and detecting atypical behavior. Such a monitoring system will be capable of detecting intrusions that could not be detected by any other means, for example, intrusions that exploit *unknown* vulnerabilities. In addition, any computer system or network has known vulnerabilities that an intruder can exploit. However, it is more efficient to detect intrusions that exploit these known vulnerabilities through the use of explicit expert system rules than through statistical anomaly detection.

While many computer systems collect audit data, most do not have any capability for automated analysis of that data. Moreover, those systems that *do* collect audit data generally collect large volumes of data that are not necessarily security relevant. Thus, for security analysis, a security officer (SO) must wade through stacks of printed output of audit data. Besides the pure tedium of this task, the sheer volume of the data makes it impossible for the security officer to detect suspicious activity that does not conform to a handful of obvious intrusion scenarios. Thus, the capability for automated security analysis of audit trails is needed.

The Next-generation Intrusion-Detection Expert System (NIDES) is the result of research that started in the Computer Science Laboratory at SRI International in the early 1980s and led to a series of increasingly sophisticated prototypes that resulted in the current NIDES Beta release. The current version, described in this final report and in greater detail in [1, 2, 3], is designed to operate in real time to detect intrusions as they occur. NIDES is a comprehensive system that uses innovative statistical algorithms for anomaly detection, as well as an expert system that encodes known intrusion scenarios.

1.1 Previous Work

One of the earliest works on intrusion detection was a study by Jim Anderson [4], who suggested ways of analyzing computer system audit data. His methods use data that are collected for other reasons (e.g., performance analysis) and were designed for "batch" mode processing; that is, a day's worth of audit data would be analyzed at the end of the day.

Subsequent to Anderson's study, early work focused on developing procedures and algorithms for automating the offline security analysis of audit trails. The aim of such algorithms and procedures was to provide automated tools to help the security officer in a daily assessment of the previous day's computer system activity. One of these projects, conducted at SRI, used existing audit trails and studied possible approaches for building automated tools for audit trail security analysis [5]. This work involved performing an extensive statistical analysis on audit data from IBM systems running MVS[1] and VM. The intent of the study was to develop analytical statistical techniques for screening computer system accounting data to detect user behavior indicative of intrusions. One result of this work was the development of a high-speed algorithm that could accurately discriminate among users, based on their behavior profiles.

Another such project, led by Teresa Lunt at Sytek [6], considered building special security audit trails and studied possible approaches for their automated analysis. These projects provided the first experimental evidence that users could be distinguished from one another through their patterns of computer system use [5], and that user behavior characteristics could be found that could be used to discriminate between normal user behavior and a variety of simulated intrusions [7, 8, 9, 10, 11, 12].

The evidence of the early Sytek and SRI studies was the basis for SRI's real-time intrusion-detection system, that is, a system that can continuously monitor user behavior and detect suspicious behavior as it occurs. This system, NIDES, is based on two approaches: (1) intrusions, whether successful or attempted, can be detected by flagging departures from historically established norms of behavior for individual users, and (2) known intrusion scenarios, known system vulnerabilities, and other violations of a system's intended security policy (i.e., *a priori* definition of what is to be considered suspicious) are best detected through use of an expertsystem rulebase.

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

Largely as a result of the pioneering work at SRI, several other intrusion-detection projects are under way at other institutions. For a survey of these, see [13].

1.2 Related Work

During the period of the NIDES work funded under this contract, four related efforts were undertaken that utilized the NIDES software prototype. These projects benefited from the work performed under this contract and also contributed to the improvements made to the NIDES software prototype:

- **Advisor Project** — Under contract to Trusted Information Systems (TIS), SRI helped to develop a modified version of NIDES to support the monitoring of Trusted Xenix systems.
- **FBI FOIMS-IDES Project** — SRI provided customized versions of the NIDES prototype to the FBI to support analysis of database transaction logs from the FBI-FOIMS database application.
- **Safeguard Project** — Under contract to TIS, SRI performed a study to analyze the effectiveness of the NIDES statistical analysis component in recognizing masquerading applications.
- **NIDES Training Course** — SRI developed and hosted a four-day, intensive, hands-on training course offered free of charge to NIDES users.

1.2.1 Advisor Project

This project was sponsored by Rome Laboratory, and SRI was a subcontractor to Trusted Information Systems. SRI added specific new functionality to NIDES to support the customization and use of NIDES with Trusted Xenix hosts. Changes made to NIDES included an analysis customization facility whereby the statistical component could be tuned and specific measures enabled and disabled, and the rulebase could be customized by enabling/disabling rules and introducing new rules in the running system. A test facility was added whereby a candidate NIDES configuration could run alongside the real-time NIDES software. An archival facility was developed that supports archival and retrieval of audit records and result data. The NIDES audit record structure was expanded to include fields that store security label and integrity information for user ids and file objects. New measures were added to the NIDES statistical analysis component to support the new security label information. SRI performed an additional task to study the feasibility of introducing a neural network component into the NIDES analysis. See [14] for a description of this neural network study.

1.2.2 FBI FOIMS-IDES Project

Under this FBI-sponsored project, an application of NIDES was developed to support evaluation of database transaction logs generated by the FBI's FOIMS application. Customizations made by SRI included development of specific expert system rules, statistical parameters, and audit data conversion routines to map the FOIMS audit trail data to the NIDES canonical audit record format. Several versions of the FOIMS-NIDES system were installed at FBI headquarters for evaluation. In addition, a study was performed that considered the security implications anticipated in the use of NCIC-2000. The results of this study are discussed in [15].

1.2.3 Safeguard Project

Under contract to Trusted Information Systems (TIS), SRI evaluated the ability of NIDES to detect the use of computer systems for other than authorized applications.

The principal goal of the Safeguard Audit Data Analysis task was to explore methods of detecting and reporting attempts, by those whose use of exported computer systems is restricted to approved applications or classes of applications, to bypass or defeat software execution restrictions. As part of this task, audit trail analysis techniques were considered as potentially useful in detecting misuse of restricted technology.

In the paradigm explored by SRI [16, 17, 18, 19, 20], the subjects were considered to have been computer users. For the Safeguard study, the NIDES methodology was adapted to consider applications as subjects. The methodology as adapted for Safeguard would be useful in the detection of trojan horses and other masquerading applications. The analysis software developed for this project was a straightforward adaptation of the current NIDES statistical analysis component.

Overall, the NIDES statistical component was successful in its efforts to detect masquerading applications, whether these were special resource-intensive applications or standard applications run through historical profiles other than their own. Simulated masquerading applications were detected 101 times in 130 opportunities, with a false positive detection rate of 1.3%.

For additional information on the Safeguard project, see [21, 22].

1.2.4 NIDES Training Course

In August of 1994 SRI hosted a training course attended by ten students. The course taught by Debra Anderson, Thane Frivold, and Al Valdes was designed to give students a hands-on introduction to the NIDES Beta release. The following topics were covered:

- System installation and configuration
- Real-time operation
- Rulebased analysis configuration

- Statistical analysis configuration
- Target host configuration
- Audit data concepts and sources
- Performance tuning
- Experimentation

Students completed course evaluations indicating that they were pleased with the course content and pacing. The rulebase sessions were the most popular. Several students successfully wrote, installed and used realistic rules during the rulebased configuration exercises. By learning about all the key features of NIDES the students were able to gain a comprehensive understanding and appreciation for the system's utility and flexibility.

1.3 Project Overview

Progress on this project concentrated on the rearchitecting of the IDES software prototype into the NIDES prototype and the subsequent refinement and enhancement of the NIDES software. Key achievements were

- IDES prototype rearchitected to become NIDES
- First NIDES release delivered (NIDES alpha)
- Alpha-patch release delivered with performance improvements
- Beta release delivered with greatly expanded configuration features and system monitoring functions
- Beta update release delivered with additional performance improvements and user customization of audit data sources for NIDES analysis
- Reports completed
  - Software test report [23]
  - Users' manuals [2, 24, 25]
  - Design documents [3, 18]
  - Requirements specification [1]
  - Technology transfer plan [26]

Chapter 2

Software Prototypes

SRI developed four major versions of the NIDES software prototype, each with increased functionality and performance:

- **February 1993 — NIDES Alpha Release:** A rearchitected prototype was completed and christened NIDES alpha. This prototype incorporated a client-server architecture and modular design into the functionality of the IDES prototype [16]. An X-windows user interface using the MOTIF toolkit was also incorporated.

- **October 1993 — NIDES Alpha-patch Release:** An updated version of the NIDES Alpha release was delivered with performance improvements, primarily involving the statistical analysis component.

- **May 1994 — NIDES Beta Release:** A greatly enhanced version of NIDES was released with numerous new features, including real-time reconfiguration of NIDES analysis, system status reporting functions, and a greatly expanded users' manual. An expanded rulebase was also developed. Thirty copies of the Beta release were distributed.

- **November 1994 — NIDES Beta-update Release:** An updated version of the NIDES Beta Release was developed to address reported bugs and to incorporate improved statistical algorithms, enhanced performance and a Perl scripted `agen`.

2.1 Alpha Release

The first NIDES release was a rearchitected version of the previous IDES prototypes [16]. The architecture was based on core components and infrastructural facilities [18]. The NIDES core components included the user interface, the resolver, the statistical, the rulebased, the audit data collection, and the audit data generation components. The NIDES infrastructural facilities included process management, interprocess communication, and graphical user interface facilities. The rearchitecting process imposed a modular design on the NIDES components, and functional interfaces for each component were specified [18] and developed. The algorithms for the rulebased analysis component were unchanged, and the algorithms for the statistical analysis component were enhanced. The rulebased and statistical analysis component interfaces to all other NIDES functions were modified to support a strict client-server model and a modular design. The alpha release also included a comprehensive user interface that supported access to all NIDES capabilities and contained a context-sensitive on-line help system for all NIDES functions. A complete NIDES user manual [24] was included with the alpha release as well. Automated installation procedures were added to the first release to facilitate ease and success of system installation. All subsequent releases (alpha-patch, beta, and beta-update) were based on the architecture developed for the initial NIDES offering. See Section 2.5 and [3] for more information on the NIDES architecture.

Statistical Analysis Algorithm Enhancements

Under the NIDES Alpha release the statistical algorithms were modified so that they could better model not just historical modes of behavior but also individual deviations about these historical modes. With the Alpha release and all subsequent releases the algorithms were no longer dependent on parametric distributions and a better representation for multimodal behavior was developed that could comprehend the behavior of subjects who, for example, might spend part of their computer activity in document preparation with intermittent periods of something significantly different such as financial analysis.

Some of the key changes made were:

- Enhancement of the definition of the transform from Q to S, permitting scoring of user activity by how unusual it is rather than by where it is on the range of possible activity
- Introduction of intensity and audit record distribution measures
- Treatment of formerly continuous measures as categorical from a computational standpoint

2.2 Alpha-patch Release

The NIDES Alpha-patch release was developed in response to user feedback regarding NIDES performance. Three key changes were made to improve system performance and increase configuration capabilities. To expedite the completion of the Alpha-patch release, no additional changes were made to the NIDES user interface. Because the user interface was unchanged, the three configuration features added to NIDES for the Alpha-patch version were accessed either via an external configuration file that a user could edit to make configuration changes or via UNIX environment variables that would be configured prior to running NIDES. Since the NIDES user interface was scheduled to be enhanced with the next (Beta) release we incorporated user interface support for the new configuration items at that time. Only two NIDES components were modified for the Alpha-patch release — the statistical analysis component `(stats_client)` and the batch analysis component `(batch_analysis)`. The three configuration features added to the Alpha-patch release were

1. **File/Directory Categorical Measures**

   The statistical analysis component includes two categorical measures that store file and directory-related activity as lists. It was discovered that these lists could become very large, possibly on the order of several thousand items, and computations performed on these lists would become more time-consuming as the list grew. We also observed that many of the files and directories in these lists were of a transitory nature and therefore of little interest to the statistical analysis. Hence, the following changes were made to combat these problems:

   - **Category List Processing Algorithms**
     The processing algorithms for category lists were improved so that the entire list need not be traversed each time the corresponding measure was observed, thus speeding up the per-audit-record processing in the statistical component. As a result, some computation had to be "pushed back" to the profile updating process, and hence the profile updating process takes a little longer. Because this process occurs only once a day, the performance difference at update time is negligible.
   - **Temporary File Configuration**
     A new feature was added to allow a NIDES system administrator to configure a list of transitory files and directories. These files are ignored by the statistical component and hence reduce the number of files and directories in the category lists for the relevant measures. For the Alpha-patch release this list of temporary files was stored in **a** file called `tmp_config` located under the `$IDES_ROOT/etc` directory.

     With the Alpha-patch release the default `tmp_config` file contained two default entries `/tmp/` and `/var/tmp/`. Additional entries defined in the `tmp_config` file are automatically added to the list. Duplicate values are eliminated, and the final list is sorted so that pattern searching can be done efficiently. The `tmp_config` file was replaced by user interface support in the NIDES Beta release.

2. **Timestamp Profile Update Mode**

   A second feature included in the Alpha-patch release was the ability to have the real-time NIDES optionally update profiles based upon the audit record timestamps instead of the system clock (the original Alpha release updated profiles using the system clock only). The user can configure the updating mode via the environment variable IDES_USE_TIMESTAMPS. If this variable is not set, the system clock value is used to determine when profile updating should occur. The environment variable configuration of this value was replaced by user interface support in the Beta release.

3. **Profile Cache Size for Statistics Client**

One of the primary areas of overhead processing in NIDES is the management of the statistical profiles. A caching mechanism was added to the Alpha-patch release to enhance profile management, and allow a NIDES system administrator to set the size of this cache as appropriate. If an organization has a large number of users or if the target systems generate a large number of file- or directory-related audit data activity, setting the size of this profile cache to a number smaller than the total number of users in the monitored environment reduces the size of the statistics process. This may improve performance because NIDES's processes should not swap as frequently. For the Alpha-patch release the size of the profile cache is set via an environment variable NIDES_STAT_CACHE_SIZE. The default cache size for the Alpha-patch release is 20, which is used if the environment variable is not set. The environment variable configuration of the profile cache was replaced by user interface support in the Beta release.

2.3 Beta Release

The first version of the Beta release represented a significant increase in the functionality provided by NIDES.

2.3.1 Documentation

Along with the updated software, an updated and greatly expanded user manual [25] was completed. Subsequent to the delivery of the release, an updated software design document was also completed [3].

2.3.2 Features

These key features were added to the NIDES prototype in the Beta release:

- Optimized profile storage structure
- Real-time configuration of NIDES analysis
- Configuration of NIDES batch analysis
- Expanded status reporting
- Data management facility
- Expanded rulebase

2.3.2.1 Optimization of Profile Structure

Some the NIDES user community expressed concern about the storage needed for subject profiles used by the statistical analysis. Most of these users had large numbers of subjects, and NIDES generates two files for each subject in which statistical information (the current or short-term profile and the historical or long-term profile) is stored.

An analysis of the data structures used to store the statistical profiles was made to determine if fields could be removed or reduced in size. The NIDES Beta release achieved a more compact profile representation than the Alpha releases by eliminating excessive use of the type `double` as well as by removing some large matrices that were no longer used. By reducing the number of fields in the structures and reducing the size of several fields we were able to reduce the baseline size of the two files by 62%. Table 2.1 shows the sizes for the Alpha version versus the sizes for the Beta version. The benefits realized by this reduction were that the amount of disk space needed to store subject profiles and the process size of the statistical analysis were reduced, thus increasing performance by decreasing the amount of swapping. Because the profile structures were modified, profiles generated under the NIDES Alpha releases cannot be used in the Beta releases.

NIDES Baseline Profile Structure Sizes

	Size Of Structure (Bytes)	
Structure	Alpha Releases	Beta Releases
Current/Short-term Profile	19112	13140
Historical/Long-term Profile	57424	15308

Table 2.1: NIDES Baseline Profile Structure Size Comparison

2.3.2.2 Analysis Configuration (Real-time and Batch)

The NIDES Beta release represented the first NIDES version that supported user configuration of most aspects of the NIDES analysis process. The following items became configurable under the NIDES Beta release. For a more complete discussion of the release's configuration options refer to [25].

Target Host Target hosts can be added or deleted from the list of available target hosts at any time during NIDES execution. The Beta release user interface contains a complete facility for editing the target host list and configuring each target host.

Alert Reporting Configuration As with the Alpha release, the user can configure the method used to report anomalies. Current options are e-mail reporting and popup window reporting. With the Beta release the user can add or delete e-mail recipients at any time during NIDES operation via the user interface.

Alert Filter The configuration of alert filters is a new concept introduced with the Beta release. Users may suppress real-time alert reporting for selected subjects by configuration of alert filters. Rulebased and/or statistical alerts can be filtered for any subject known to NIDES.

Statistical Analysis Most parameters of the NIDES statistical analysis component are configurable under the Beta release. All the statistical configuration items are accessed through the NIDES user interface. The following statistical items can be configured:

- Measure parameters — Scalar, half-life, QMAX, minimum effective-N, and measure state (On or Off)
- Global parameters — training period, profile half-life, cache size, red and yellow thresholds, and rare category probability sum
- Category class lists
- Profile management functions — copying, replacement, and deletion of subject profiles
- Profile updating — update time determination (audit record timestamps or system clock), updating state (ON or OFF) globally or on a subject-by-subject basis (real-time analysis only)
- Profile synchronization — under the batch mode of operation subject profile update times can be synchronized with the audit data used for the batch run

Rulebased Analysis Two new features were added to the Beta release that allow the user to modify the rulebased analysis component during runtime. New rules may be compiled, installed, and then added to the real-time or batch analysis processing. Rules already available to NIDES may be turned ON or OFF dynamically during runtime (real-time analysis) or during the test configuration process (batch analysis mode).

Result Filter Each NIDES analysis result is assigned one of three levels, SAFE, WARNING, or CRITICAL. With the result filter configuration option a user can determine which level of results is written to the NIDES result database. By restricting the results that are archived, the user can reduce the amount of disk space required to store the analysis results and improve NIDES performance.

Remarks For each instance created, the user can enter or update information in a log. The recorded information could include experiment notes and configuration changes.

2.3.2.3 Status Reporting

With the NIDES Beta release the status reporting functions provided under the Alpha release were greatly enhanced in response to user feedback. The Beta release provides status reports on three areas:

- **System Status**
  NIDES reports the state (ON/OFF) and throughput numbers for the NIDES analysis and archival processes. Numbers reported include audit record and alert counts since a process state change and during the most recent hour.

- **Target Most Status**
  NIDES reports the configuration (ON/OFF) and the status (UP/DOWN) of each known target host. Audit record and alert counts for each target host since its activation and during the most recent hour are also reported.

- **Experiment Status**
  For each active batch analysis process the number of audit records processed and alerts generated are reported periodically (approximately every 10 seconds).

2.3.2.4 Data Management Facility

A comprehensive data management facility was incorporated into the Beta release to support archival and retrieval of audit record and result record data. The Beta release user interface controls data archival and review of the contents of the various NIDES databases. The batch analysis facility can use data stored in the audit record database.

2.3.2.5 Expanded Rulebase

The number of alert-generating rules was increased from 21 for the Alpha release to 39 for the Beta release. Rules added to the Beta release detect suspicious FTP/TFTP use, paranoid user behavior, suspicious remote execution, and truncation of log files. Table 2.2 shows the rules included with each NIDES release.

2.4 Beta-update Release

The final release, the NIDES Beta-update, includes bug fixes, performance improvements, and expanded features.

2.4.1 Bug Fixes

Many users of the initial Beta release provided SRI with valuable comments on the software. Some of the feedback described bugs which we attempted to correct in the Beta-update version. These bug fixes occurred in the following areas:

Rulebase Default Rules		
	NIDES Release	
Rule Name	Alpha and Alpha-patch	Beta Releases
AccessPrivateDevice		X
AccessPrivateFile1	X	X
AccessPrivateFile2	X	X
AccessSpecialFile	X	X
BackwardsTime		X
BadLoginAnomaly	X	X
BadPasswordAnomaly	X	X
ParanoidUserAnom		X
BadRoot	X	X
BadUserExec	X	X
BrokeRoot	X	X
ChangeLoginFile	X	X
ChmodOtherUser		X
ChmodSystemFile		X
DotFile		X
FTPAnomaly		X
GoodLogin1		X
GoodPassword1		X
KnownLogin1	X	X
Leapfrog1	X	X
LinkSystemExec		X
ModSystemExec	X	X
NoRemote	X	X
PasswordFileAccess	X	X
ReadSystemExec		X
RemoteExec		X
RemoteFile1	X	X
RemoteFile2	X	X
RemoteFile3	X	X
RemoteMount1	X	X
RemoteMount2	X	X
RemoteRootBadLogin		X
RemoteRootBadPassword		X
SpecUserExec	X	X
Su1		X
SuspiciousUser		X
TFTPAnomaly		X
TrojanHorse	X	X
TruncateLog		X
Total Rules	21	39

Table 2.2: NIDES Default Rules

- Audit data set functions
- Test SetUp & Exec Menu option
- Profile update processing
- Confirmation windows
- `audit2ia` utility program

2.4.2 Performance Improvements

NIDES users reported performance degradation on certain types of data files. These users supplied samples of audit trails that appeared to exacerbate the performance degradation, and SRI performed an analysis of those audit trails. It was discovered that the audit trails contained examples of single users who had accessed on the order of 100,000 to 300,000 unique files over a four day period. The slowdown was attributed to the statistical analysis component. For each file a user accesses, a file category is created. While the daily profile updating removes most rarely seen file categories from a user's profile, the daily processing of each file category was too much for NIDES. Over 90% of the files seen would normally not remain in a user's profile after the daily updating was completed. SRI devised a method for preventing these files from becoming categories if it was likely that they would be dropped from the profile the same day they were added.

We implemented a temporary list of newly observed categories; the members are promoted to become actual categories in a user's profile only after they are observed enough times to indicate that they are likely to enter the permanent category list at the next update. This list is bounded in size and drops categories for which the observations are too few and/or too distant in time, so that the category list will not grow without bound. NIDES drops categories for which the historical probability falls below the system constant MINPROB. In situations where explosive growth in new categories is observed, most new categories do not achieve high enough counts to make their probability exceed MINPROB, and therefore the categories are dropped at the first update after their initial observation. However, these categories severely cluttered the short-term profile until the next profile update. To solve this problem, a temporary category is created that is not promoted (i.e., added to the actual short-term profile) until its observed count exceeds the product of MINPROB and the historical effective n. This causes the promotion of just those categories that are likely to become part of the historical profile at the next update. Other newly observed categories are kept in the temporary list, with a provision for being removed from the list once it is full. Removal is based on category rank, which is based on observation count and time of last observation, with higher ranks corresponding to categories that are seen more frequently and recently. When a profile update occurs, the temporary category list is cleared and a new list is started.

SRI performed some initial experiments to evaluate the performance gain realized by the improvements made to category management functions. In cases where the audit trail did not contain a large number of unique files, performance improvements were slight. However, during running of the audit trails that had initially brought the problem to light, performance improvements were dramatic, and in some cases experiments that would not run to completion on a minimally configured Sun workstation because of memory limitations would easily complete in a matter of hours.

Table 2.3 shows some of the statistics collected during our performance tests.

		Beta Release		Beta-Update Release	
Data Description	# of Records	Time	Rate	Time	Rate
60 Days Database Logs	8,650,824	20:40:00	116/s	23:26:00	10
31 Days Sun C2 Data (4 Flags/14 Hosts)	201,124	40:00	84/s	34:00	9
5 Days Sun C2 Data (8 Flags/1 Host)	1,603,525	26:33:00	17/s	8:02:00	5
4 Days Sun BSM Data (11 Flags/1 Host)	1,111,325	Over 120 hours	< 1/s	5:33:00	5

Table 2.3: Beta Release Performance

An additional benefit to this enhancement was that the sizes of user profiles were reduced, in some cases by a factor of 100, which improved file I/O performance and hence NIDES performance, and the size and growth of the NIDES statistical analysis process was reduced, in some cases by a factor of 5.

2.4.3 New Features

Several new features added to the Beta-update release of NIDES primarily focused on providing a mechanism by which users could customize NIDES to use any source of audit data:

- Perl script `agen` utility
- UNIX `agen` monitor for Ethernets in promiscuous mode
- Expanded rulebase `event` fact template
- Expanded audit record source codes
- Expanded audit record action codes
- Updated installation procedures and user documentation

2.4.3.1 Perl Script agen Utility

Previous versions of NIDES relied upon `agen` processes written in C and compiled for the target architecture (namely Sun SPARC). Thus, the number of different platforms that NIDES could monitor was severely limited. In an effort to permit both the monitoring of multiple platforms and 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 for free, easy to install, and operational on a large number of different operating systems and hardware platforms.

The Perl `agen` has table driven audit source customization that is simple and flexible. Users may enter multiple audit data sources in the Perl `agen` configuration file and then simply write a script to convert each data source into NIDES audit records. Library functions, which generate NIDES audit records, are included with the software to facilitate user development of data conversion scripts.

Versions Developed for Beta-update Release To aid users in development of their own `agen` Perl scripts, SRI has provided two realistic sample scripts for TCP-wrapper data and UNIX accounting data file formats. The TCP-wrapper writes log entries using the standard UNIX syslog facility. Three types of records are generated by the TCP-wrapper — connections, refused connections, and mismatched host name lookups. The UNIX accounting data file is fairly standard for most UNIX systems; therefore, the Perl `agen` for UNIX accounting files is comparable to the regular compiled `agen` program that can process UNIX accounting logs on a Sun workstation. The Perl script version allows users to analyze comparable data from target hosts that are not Suns. For the TCP-wrapper data, two NIDES rules were written to serve as examples of how the new data might be used. One rule looks for TCP wrapper log records that report refused connections; the other rule looks for mismatched host names and addresses.

2.4.3.2 UNIX `agen` Ethernet Enhancement

The Sun version of `agen` was modified to determine whether the host's Ethernet is in *promiscuous* mode, which could indicate the presence of a network "sniffer" that could capture users' passwords or other sensitive data. If `agen` discovers that the Ethernet controller is in promiscuous mode, a special audit record is generated and passed into `arpool` and to be checked for by the rulebased analysis component so that an alert can be generated if appropriate. The `agen` process checks the Ethernet controller every 10 seconds and reports an event to `arpool` each time the controller is in promiscuous mode.

2.4.3.3 Updated Rulebase and `event` Fact Template

With the addition of an audit data customization feature using Perl scripts, changes were made to the rulebased analysis component to support user development of Perl scripted `agens.`

The `event` fact template used to store NIDES audit records was updated and expanded to include all data fields available in an audit record. Prior versions of the NIDES rulebase software provided only a subset of the audit record fields available in NIDES, namely, the fields most commonly used under SunOS auditing systems. With the addition of user audit data customization facilities in the Beta-update release it was deemed necessary that all fields in the audit record be made available to the rulebased component so that users would have complete flexibility in their use of that component and the audit record format. Because the structure of the `event` fact was modified, existing rulebases (pre Beta-update release) are not compatible with the Beta-update release of NIDES. Instances created and modified under the earlier Beta release cannot be used with the new software. In addition, any user-developed rules that access fields in `event` facts must be modified to refer to the fields by their new names. The Beta-update user manual describes in detail the differences between the old `event` fact template and the updated `event` fact template.

2.4.3.4 Expanded Audit Record Codes

To accommodate users customization of the NIDES audit data sources using the new PERL scripted `agen,` the audit record action codes and source codes available to NIDES were expanded. The audit record action codes indicate the type of action represented in the audit record. Table 2.4 shows the action codes available under the NIDES Beta and Beta-update releases. These codes are defined as an enumerated C'type.

The audit data source codes indicate the source of the audit data. Table 2.5 shows the old and new source codes available under NIDES. These codes are also defined as an enumerated C'type.

2.4.3.5 Updated Installation Procedures and Documentation

To accommodate both existing and new NIDES users, the installation procedures and user documentation were updated for the Beta-update release.

Installation scripts and procedures were modified to accommodate changes made to the NIDES directory tree and configuration changes to NIDES rulebase files. Procedures were outlined for Beta users on how to reuse data with the Beta-update version. Installation of Beta-update software is similar to that of previous NIDES install scripts. The user specifies the IDES_ROOT directory, and the NIDES release is installed in that directory. The IDES_ROOT directory specified should not already exist, and is created during the installation process. Some post-installation steps are required for Beta users who want to reuse NIDES data or instances created under the NIDES Beta release. All NIDES audit data sets and archives are compatible between the two Beta versions. Users can transfer any audit data sets or archives to their new IDES_ROOT area. For existing instances, profiles are compatible and can be copied to Beta-update instances. Because of rulebase fact format changes, existing rulebases are not compatible with the Beta-update release. Any user-developed rules must be modified to take into account the modified `event` fact format.

Minor corrections were made and new material was added to the Beta release user's manual [25], and a new version of the manual was produced [2]. A new chapter was added discusses the new Perl scripted `agen` facility. The installation section was also updated.

Action Code	Beta	Beta-update	Action Code	Beta	Beta-update
IA_VOID	**X**	**X**	IA_RMOUNT	**X**	**X**
IA_DISCON	**X**	**X**	IA_BAD_RMOUNT	**X**	**X**
IA_ACCESS	**X**	**X**	IA_PASSWD_AUTH	**X**	**X**
IA_OPEN	**X**	**X**	IA_BAD_PASSWD_AUTH	**X**	**X**
IA_WRITE	**X**	**X**	IA_KILL		**X**
IA_READ	**X**	**X**	IA_CORE		**X**
IA_DELETE	**X**	**X**	IA_PTRACE		**X**
IA_CREATE	**X**	**X**	IA_TRUNCATE		**X**
IA_RMDIR	**X**	**X**	IA_UTIMES		**X**
IA_CHMOD	**X**	**X**	IA_FORK		**X**
IA_EXEC	**X**	**X**	IA_CHROOT		**X**
IA_CHOWN	**X**	**X**	IA_MKNOD		**X**
IA_LINK	**X**	**X**	IA_HALT		**X**
IA_CHDIR	**X**	**X**	IA_REBOOT		**X**
IA_RENAME	**X**	**X**	IA_SHUTDOWN		**X**
IA_MKDIR	**X**	**X**	IA_BOOT		**X**
IA_MOUNT	**X**	**X**	IA_SET_TIME		**X**
IA_UNMOUNT	**X**	**X**	IA_SET_UID		**X**
IA_LOGIN	**X**	**X**	IA_SET_GID		**X**
IA_BAD_LOGIN	**X**	**X**	IA_AUDIT_CONFIG		**X**
IA_SU	**X**	**X**	IA_IS_PROMISCUOUS		**X**
IA_BAD_SU	**X**	**X**	IA_CONNECT		**X**
IA_EXIT	**X**	**X**	IA_ACCEPT		**X**
IA_LOGOUT	**X**	**X**	IA_BIND		**X**
IA_UNCAT	**X**	**X**	IA_SOCKET_OPTION		**X**
IA_RSH	**X**	**X**	IA_ACTION_RESERVED00— 144		**X**
IA_BAD_RSH	**X**	**X**	IA_ACTION_USER00— 49		**X**
IA_PASSWD	**X**	**X**			

Table 2.4: NIDES Audit Record Action Codes Comparison Beta and Beta-update Releases

Audit Source Code	Beta Release	Beta-Update Release
IA_SRC_VOID	X	X
IA_SRC_C2	X	X
IA_SRC_PACCT	X	X
IA_SRC_ADABAS	X	X
IA_SRC_LINK	X	X
IA_SRC_BSMV1	X	X
IA_SRC_BSMV2		X
IA_SRC_SYSLOG		X
IA_SRC_AGEN_HOST		X
IA_SRC_AGEN_NETWORK		X
IA_SRC_RESERVED00 thru 89		X
IA_SRC_USER00 thru 49		X

Table 2.5: NIDES Audit Record Source Codes

2.5 Architecture

In the NIDES Alpha release and all subsequent releases, the architecture follows a client-server model. Each NIDES component now has a modular design that provides a well-defined interface for other NIDES components while keeping algorithms and specific processing requirements hidden.

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 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.

2.5.1 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
- Resolver
- Archiver
- Batch analysis
- User interface

2.5.1.1 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 include 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 instance.

Instances NIDES uses a storage concept called an *instance* to separate different versions of the same type of information. 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.

2.5.1.2 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.

2.5.1.3 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-update 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 — and a Perl script functionality that allows users to develop their own `agen` scripts for other data sources. When started, the default behavior for `agen` is to process all available types of audit data. Typically, Sun OS 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.

2.5.1.4 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.

2.5.1.5 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 each user's behavior. This component 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. For more detailed information on the statistical algorithms see [20, 21].

2.5.1.6 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. Starting with the NIDES Alpha release the rulebase could be configured via a file `rb_config,` which supports users' configuration of existing NIDES rules. Rulebase customization capabilities were increased with the NIDES Beta release; 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, users should customize the rulebase for their own environments and continually update the rules accomodate changing vulnerabilities of new system releases and newly discovered vulnerabilities of current releases.

2.5.1.7 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, the security officer can turn off alert reporting for specific users, if it is known that 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.

2.5.1.8 Archiver

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

2.5.1.9 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 real-time NIDES operation. 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 real-time NIDES operation. Test results are archived for comparison.

2.5.1.10 User Interface

A NIDES user accesses all NIDES capabilities through the NIDES user interface. 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.

2.5.2 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.

2.5.2.1 Real-time Operation

The primary mode of NIDES 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.

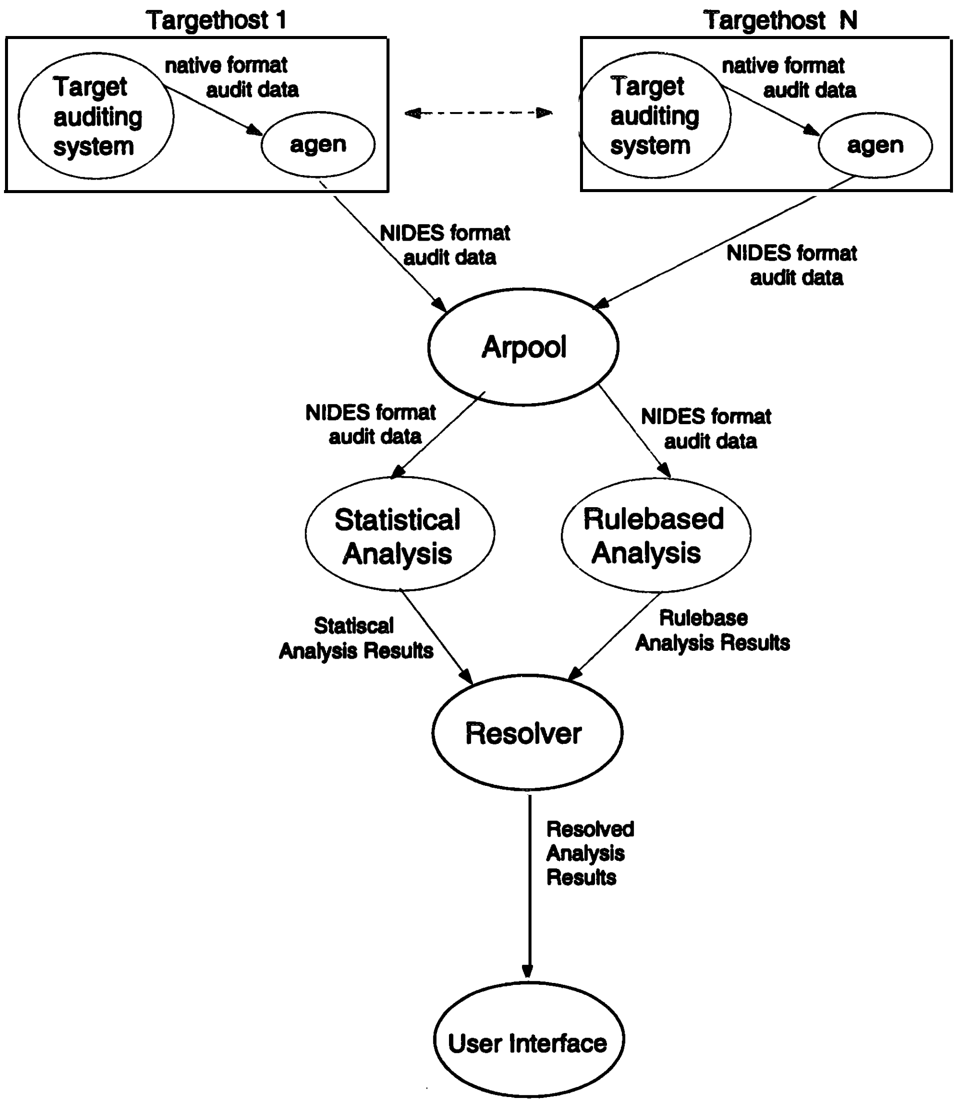

Figure 2.1: NIDES Process Graph (Real Time)

The flow of data starts with the target hosts (top of graph). The `agen` process converts data in the target hosts 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 them 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 or using Perl scripts. 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).

2.5.2.2 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. 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.

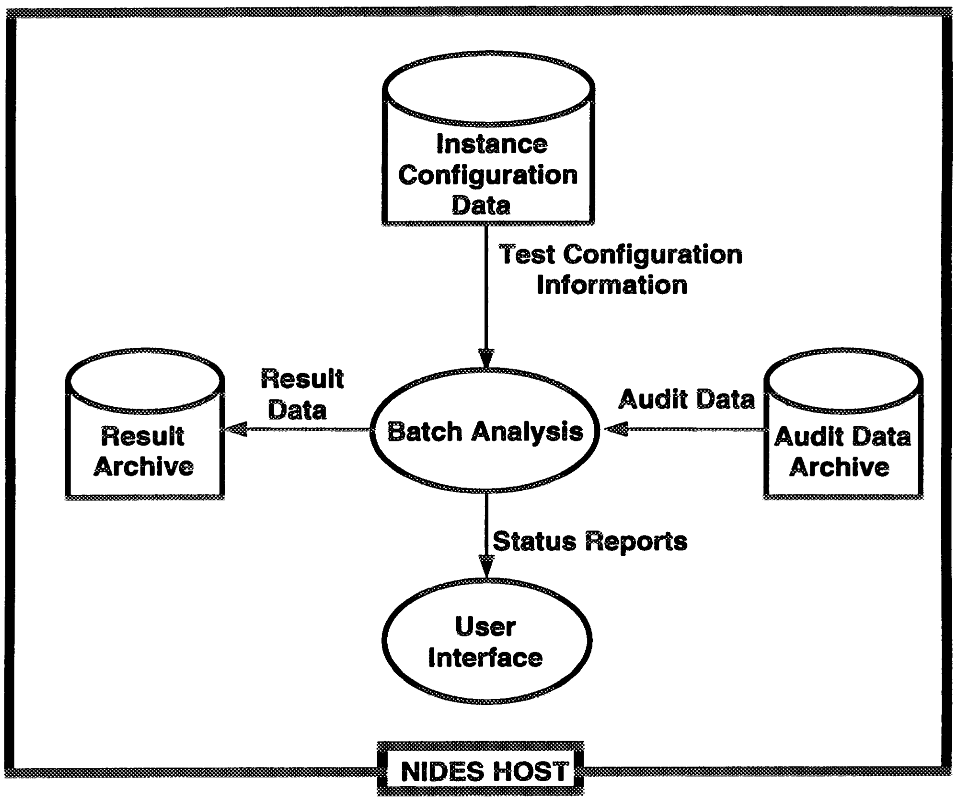

Figure 2.2: NIDES Process Graph (Batch Mode)

Chapter 3

Future Directions

The current NIDES prototype is a well-architected system that contains a large number of features facilitating component reuse and configuration. The NIDES technology could continue to grow and contribute to the improvement of system security in many areas. The following tasks could improve upon the current prototype and make NIDES a more useful tool to system security personnel:

- Technology transfer and operational evaluations
- User training and support
- Enhance NIDES security
- Network intrusion detection
- Intrusion detection testbed
- Comprehensive rulebase
- Profile other entities
- Enhance component independence

3.1 Technology Transfer and Operational valuation

The current NIDES prototype software has been delivered to more than thirty sites, many of which are U.S. government agencies. The final prototype version of NIDES, described in this report, is the most usable and mature NIDES to date.

The NIDES technology represents the culmination of many years of research and software prototyping in the computer intrusion-detection field. Now that the NIDES technology has matured, the next step is to begin transfer of the technology into user environments. The impetus behind this technology transfer is to discover how the advanced technology developed in the laboratory applies to an end-user environment. By introducing advanced prototype versions of NIDES into end-user communities, we can discover how the basic and applied research that has led to the prototype has succeeded and where its methods can be improved. Technology transfer issues related to NIDES are discussed in detail in [26].

3.2 User Support and Training

SRI has received valuable feedback from users of both the Alpha and Beta releases. As a result, many changes were made to the software in the current Beta-update version. As users continue their experimentation with the Beta-update release, the need for further changes will arise. For these sites to successfully use NIDES, our continued availability for maintenance and support is needed. As our experience with NIDES continues, additional bug fixes and feature improvements will result in a more usable and accepted system. The efforts to support users can be divided into the four areas described below.

3.2.1 NIDES Maintenance

We envision the need to provide bug fixes and performance improvements to NIDES, based on user feedback. When appropriate, SRI can obtain samples of the user data in sufficient quantity to duplicate reported problems in our own laboratory. For problems related to the false-alarm rate, we can make corrections to the statistics, the rulebase, and/or the resolver filtering and decision procedures so as to produce an acceptable false-alarm rate, while retaining the ability to detect the events of concern. For performance-related problems, we can conduct experiments with user data and our own data to discover performance bottlenecks. The fixes that are made may be specific to the reporting user (i.e., tuning the NIDES components for best results in a particular monitored community) or may apply to all NIDES users (i.e., a specific enhancement or bug fix). In the former case, the fix/enhancement can be provided to the reporting user. In the latter case, the fix/enhancement can form part of a new NIDES release.

3.2.2 Training

Based on the success of the our initial training course it is clear that NIDES users can benefit greatly from a comprehensive course. By exposure to all of NIDES in a classroom setting, users can have a high degree of success and be made aware of all the capabilities NIDES offers.

3.2.3 Telephone and On-site Support

SRI can provide installation support and on-site hands-on assistance as requested. Telephone and email support to answer user questions and address problems is also essential. In addition, SRI can assist NIDES users in tailoring the NIDES statistical, rulebased, and `agen` components to best suit their environments.

For more complex problems, where lengthy interaction is required, SRI can provide on-site support by sending NIDES-trained personnel to assist a NIDES user in solving a specific NIDES-related problem.

3.2.4 Configuration Management

The NIDES software resides on a secure host in a locked room at SRI that is not connected to any network or any other machine. This host was prepared by completely erasing its disks and installing a pristine copy of the Sun operating system. Following this, we installed NIDES source code that had been inspected and checkpointed. All NIDES binaries are built on the secure machine. Updates to the NIDES source code are verified and checkpointed before installation.

3.3 Security Goals

NIDES is itself a sensitive application and has security requirements in addition to those of the systems whose use is being monitored [27].

Malicious tampering with the audit data or with the analysis system itself could lead to intrusions going undetected. If an intruder can read the NIDES rulebase, then the penetrated site and other sites using a substantially similar rulebase could be jeopardized, especially if such knowledge is shared among the intruder community.

As is any computer system, NIDES is potentially subject to attack. For example, an attacker may try to break the security officer's account, try to cause a core dump, or try to exploit potential weaknesses in the design and implementation. Once NIDES is compromised, the attacker may disable the system.

The NIDES rulebase represents explicit scenarios that are indicative of potential system misuse. An attacker who obtains the rulebase may be able to devise attack strategies to defeat the rules, and thus evade detection. Therefore, it is vital to protect the rulebase from disclosure to intruders and from unauthorized modification.

We consider two types of threats — tampering and reverse engineering. Either type may be critical in a given operating environment. These threats may arise from authorized users or unauthorized outsiders. The following are NIDES security goals:

- **Goal 1: Target-system integrity.** NIDES must not have adverse effects on the integrity of the target systems. NIDES should authenticate interactions with target systems. NIDES should continually validate the status of all monitored systems to detect involuntary shutdown.

- **Goal 2: Audit data security.** NIDES must protect the audit data transmitted from the target systems to NIDES — from browsing, alteration, deletion, or spoofing.

- **Goal 3: System integrity.** NIDES must be protected from modification of its procedures and system data.

- **Goal 4: Availability.** NIDES must be protected from malicious denials of service.
- **Goal 5: Rulebase protection.** The NIDES rulebase must be protected from undesired reading and reverse engineering and from unauthorized modification.
- **Goal 6: User access.** Access to NIDES and its data must be restricted through user authentication within a tightly controlled set of authorized personnel.
- **Goal 7: User accountability.** User activities must be logged and analyzed in sufficient detail for the detection, by security officers and administrators, of malicious misuses of NIDES and its data, and of intrusions by unauthorized personnel.

With the current modular architecture of NIDES, enhancements could be easily made make NIDES more tamper-proof and resistant to reverse engineering. These enhancements could include

- Separation of roles
- Interprocess and server authentication
- Detection of and protection against denials of service
- Improved target system integrity/availability
- Arpool target host authentication
- Smokescreen detection
- Logging of NIDES user actions
- Improvement of NIDES processes integrity/security
- Improve alert reporting integrity
- Authentication of NIDES users
- Restriction of TCP/IP services
- Protection of the rulebase and software from reverse engineering

3.4 Network NIDES

Current computing environments are, more and more, massively networked sets of workstations, servers, mainframes, supercomputers, and special-purpose machines and devices. Focusing on the vulnerabilities of any single host may prove inadequate in such a distributed environment. Detection of intruders will require a comprehensive view of a network, possibly extending to other networks gated to the local network, to detect *itinerant intruders* — those who move through many machines in many autonomous administrative domains. Applying intrusion-detection technology to a networked computing environment will require expanding its scope to include the ability to detect network intrusions as well as intrusions into the individual host machines, and to recognize patterns of activity that are spread among many hosts. The current NIDES technology could be easily extended to perform network monitoring and intrusion detection in three key areas — network data collection, rulebased analysis, and statistical analysis. In addition, the NIDES architecture could be extended to support multiple cooperative NIDES processes that would each be responsible for a local domain, with a higher-level NIDES process responsible for the network that supports all the local domains.

3.4.1 Data Collection

NIDES currently uses the audit data available to track both successful and failed attempts to use network services. For example, under Sun's BSM module, certain telnet, rlogin, and ftp events are directly introduced into the system audit stream via system calls made available to application programs. Most other services (and their related server processes) provide little or no application-level auditing (e.g., YP/NIS, NFS, SMTP). The only access to any network service information, short of rewriting the concerned system processes, is to directly monitor network packets and to synthesize network-level audit records for the existing NIDES analysis modules. A network monitoring process could be incorporated into NIDES to read network packets and produce canonical NIDES audit records for analysis. Part of this process would entail substantial filtering of the network data prior to conversion to NIDES audit records.

3.4.2 Rulebase

The current NIDES rulebase can be extended to include rules specific to the domain of detecting suspicious network activity. Several types of scenarios can be addressed, for example:

- **Host Name Spoofing.** An intruder can attempt to have his host masquerade as a host that is trusted (e.g., for NFS (Network File System) or remote login service) by a target server by having a domain name server show the intruding host's address as that of the trusted host. An intrusion detection system could check for domain name server responses that claim to associate the name of a trusted host with some other host's IP address.

• **IP Address Spoofing.** An intruder can have his host masquerade as a host that is trusted by a target server by having its IP address pretend to be that of the trusted host. An intrusion-detection system, if it can tell which physical network a packet comes from, could look for packets with IP addresses that should not have come from that network. Within a local network, the network intrusion-detection system could verify that IP addresses are in fact assigned to the host with the physical (e.g., Ethernet) addresses from which they originate.

• **Doorknob Twisting.** "Doorknob twisting," refers to service access attempts that may be harmless individually, but taken together probably represent a break-in attempt. For example, on a network that allows remote connections (or connections to certain services such as login) from only a limited set of hosts, a series of connection attempts within a short time from a variety of unauthorized hosts is likely to represent a serious break-in attempt rather than simply someone poking around. A network intrusion-detection system could watch for such patterns.

3.4.3 Statistical Measures

The NIDES statistical component is easily adaptable to network intrusion detection with a reinterpretation of subject and action but otherwise no algorithmic changes. For network auditing, profiled entities can be network hosts. For the NIDES statistical component, a set of statistical intrusion-detection measures that are specific to network-level intrusions are easily developed. For example, such measures might profile the amount of traffic originating or being received by local and remote hosts for given times of the day and days of the week. This information could be correlated with information received from audit trails, such as port number and network activity type, to build additional measures of interest.

3.5 Intrusion-Detection Testbed

Intrusion-detection technology research has been ongoing for many years, and numerous concepts and prototypes have been developed [13]. Many of these prototypes address specific subsets of the security and privacy concerns of end users [28]. In addition, many security tools have been developed both as freeware and as commercial products.

It is likely that a combination of the currently available tools can increase the security of today's computer systems. The development of an intrusion-detection/security testbed would be very beneficial toward the development of an intrusion-detection/security architecture that could provide greater protection from both external abuse and internal misuse, while still permitting the required amount of local and remote access. The goals of an intrusion-detection testbed would be to

1. Develop filters for making data from different tools interoperate

2. Determine which existing technologies are most promising

3. Determine appropriate tool sets for specific monitoring and performance needs
4. Compare performance and detection capabilities of different tool sets
5. Determine goals for future systems based on observed deficiencies

The types of tools that could be evaluated and incorporated include real-time monitoring tools (e.g., TCP wrapper, NID, Stalker), state evaluation tools (e.g., COPS, Tripwire, TAMU), firewalls, application gateways, and other well-known network management protocols (e.g., DNS, RIP, SNMP).

The NIDES Beta-update release represents one of the most powerful intrusion detection prototypes developed to date. With the addition of a user-configurable audit data translation utility, NIDES is now able to import data from other audit data sources for its analysis. Within the architecture of an intrusion detection testbed, the NIDES analysis components could analyze data generated by all or most of the testbed systems, perform an integrated analysis of all data collected, and determine which data are of most value in the detection of anomalies.

3.6 Rulebase Expansion

The existing NIDES default rulebase includes 39 rules that detect anomalies and generate alert reports. The rulebase has evolved and been expanded to address a variety of known UNIX system vulnerabilities. The rulebase development has been undertaken with the intent to provide a good baseline set of rules serving as an example of what the NIDES rulebased analysis can detect. The next progression for the NIDES UNIX-oriented rulebase would be to perform a comprehensive survey and analysis of known UNIX vulnerabilities and develop rules to address as many of these weakness as existing or potential audit trails support. Many computer communities could contribute to the development of this list of weaknesses. Current NIDES users who have experience and responsibilities for system security would be valuable sources of intrusion scenarios; existing response teams — CERT, for example — could provide knowledgeable insights into current vulnerabilities. System security managers and administrators should be surveyed for their inputs and views on potential weaknesses on UNIX platforms. Operating system vendors may also provide valuable inputs.

3.7 Profiling Other Entities

The Safeguard study [21, 22] has demonstrated the adaptability of NIDES to profile entities other than users. To expand upon this, NIDES could monitor different entities at various levels. For example, in addition to user monitoring, NIDES could also monitor hosts, trusted programs, user groups, and networks. Information from a single audit record could contribute to the profiles of different kinds of subject entities. To facilitate this profiling, a more flexible mechanism for mapping between audit data and statistical measures is needed. NIDES could be adapted to allow a configurable mapping between audit data and statistical measures, thus allowing users to configure NIDES to profile many different types of entities simultaneously.

3.8 Enhanced Component Independence

Under the current architecture of NIDES [3] the real-time analysis components must be manipulated via the user interface component. The user interface provides an easy mechanism to manipulate NIDES analysis and review system data. However, it would be beneficial to have command-line or compiled library access to the separate analysis components (statistical and rulebased) without use of the user interface. Separate access would provide the following benefits:

1. Reduced system load achieved when only desired analysis components are executed
2. Improved recovery procedures in the event of a system error, if only the component that caused the error needs to be restarted, all other active NIDES processes could continue to operate
3. Easy incorporation of NIDES analysis into other security systems

To achieve separation of the NIDES analysis components, the following modifications could be made to the existing NIDES software:

- Allow the NIDES user interface to be started or stopped without interruption of ongoing real-time analysis
- Allow users to select which type of analysis is used for both real-time and batch processing
- Enhance the software modules for the statistical and rulebased components to allow for command line invocation without use of the NIDES user interface
- Provide 'C" libraries that permit statistical and/or rulebased analysis on individual records

Bibliography

[1] Teresa F. Lunt and Debra Anderson. Software requirements specification: Next-generation intrusion detection expert system. Final Report A00l, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, April 1994.

[2] Debra Anderson and Thane Frivold Alfonso Valdes. Next generation intrusion detection expert system (NIDES): Software users manual; computer system operators manual (beta-update release). Manual, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, December 1994.

[3] 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.

[4] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.

[5] H. S. Javitz, A. Valdes, D. E. Denning, and P. G. Neumann. Analytical techniques development for a statistical intrusion detection system (SIDS) based on accounting records. Technical report, SRI International, Menlo Park, California, July 1986. Not available for distribution.

[6] T. F. Lunt, J. van Horne, and L. Halme. Automated analysis of computer system audit trails. In *Proceedings of the Ninth DOE Computer Security Group Conference,* May 1986.

[7] T. F. Lunt, J. van Home, and L. Halme. Analysis of computer system audit trails — initial data analysis. Technical Report TR-85009, Sytek, Mountain View, California, September 1985.

[8] T. F. Lunt, J. van Horne, and L. Halme. Analysis of computer system audit trails — intrusion characterization. Technical Report TR-85012, Sytek, Mountain View, California, October 1985.

[9] T. F. Lunt, J. van Horne, and L. Halme. Analysis of computer system audit trails — feature identification and selection. Technical Report TR-85018, Sytek, Mountain View, California, December 1985.

[10] T. F. Lunt, J. van Horne, and L. Halme. Analysis of computer system audit trails — design and program classifier. Technical Report TR-86005, Sytek, Mountain View, California, March 1986.

[11] J. van Horne and L. Halme. Analysis of computer system audit trails — training and experimentation with classifier. Technical Report TR-85006, Sytek, Mountain View, California, March 1986.

[12] J. van Horne and L. Halme. Analysis of computer system audit trails — final report. Technical Report TR-85007, Sytek, Mountain View, California, May 1986.

[13] T. F. Lunt. Automated audit trail analysis and intrusion detection: A survey. In *Proceedings of the 11th National Computer Security Conference,* October 1988.

[14] Aviv Bergman. Intrusion detection with neural networks. Technical Report A0I2, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, February 1993.

[15] Peter G. Neumann. Security and integrity controls for federal, state, and local computers accessing NCIC. Technical report, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, June 1990.

[16] Teresa Lunt, Ann Tamaru, Fred Gilham, R. Jagannathan, Caveh Jalali, Harold Javitz, Alfonso Valdes, Peter Neumann, and Thomas Garvey. A real-time intrusion-detection expert system (IDES). Final technical report, Computer Science Laboratory, SRI International, Menlo Park, CA 94025, February 1992.

[17] H.S. Javitz and A. Valdes. The SRI statistical anomaly detector. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy,* May 1991.

[18] R. Jagannathan, Teresa Lunt, Debra Anderson, Chris Dodd, Fred Gilham, Caveh Jalali, Hal Javitz, Peter Neumann, Ann Tamaru, and Alfonso Valdes. System design document: Next-generation intrusion detection expert system (NIDES). Technical Report A007, A008, A009, A010, A012, A014, SRI International, Menlo Park, CA, March 1993.

[19] Harold S. Javitz, Alfonso Valdes, Teresa F. Lunt, Ann Tamaru, Mabry Tyson, and John Lowrance. Next generation intrusion detection expert system (NIDES); 1: Statistical algorithms rationale; 2: Rationale for proposed resolver. Technical Report A016, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, March 1993.

[20] Harold S. Javitz and Alfonso Valdes. The NIDES statistical component description and justification. Annual report, SRI International, Menlo Park, CA, March 1994.

[21] Alfonso Valdes and Debra Anderson. Statistical methods for computer usage anomaly detection using NIDES. In *Conference on Rough Sets and Soft Computing,* November 1994.

[22] 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.

[23] Debra Anderson. Next generation intrusion detection expert system (nides): Software test report. Technical Report A004, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, October 1994.

[24] Debra Anderson, Christopher Dodd, Fred Gilham, Caveh Jalali, Ann Tamaru, and Mabry Tyson. Next generation intrusion detection expert system (NIDES): User manual for security officer user interface (SOUI). User manual, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, March 1993.

[25] Debra Anderson, Thane Frivold, Ann Tamaru, and Alfonso Valdes. Next generation intrusion detection expert system (NIDES): Software users manual; computer system operators manual. Manual A006 and A007, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, June 1994.

[26] Debra Anderson, Teresa Lunt, and Peter Neumann. Next generation intrusion detection expert system (nides): Large network architecture. Technical report, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, November 1992.

[27] D.E. Denning and P.G. Neumann. Requirements and model for IDES – a real-time intrusion-detection expert system. Document A005, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, August 1985.

[28] Teresa Lunt. Detecting intruders in computer systems. In *1993 Conference on Auditing and Computer Technology,* 1993.

Computer Science Laboratory
SRI INTERNATIONAL

Technical Reports - 1995

SRI-CSL-95-01, March 1995
Formal Methods and their Role in the Certification of Critical Systems, John Rushby

SRI-CSL-95-02, January 1995
Adaptive Fault-Resistant Systems, Jack Goldberg, Li Gong, Ira Greenberg, Raymond Clark, E. Douglas Jensen, Kane Kim, and Douglas M. Wells

SRI-CSL-95-03, January 1995
Mechanized Formal Verification – Seven Papers, David Cyrluk, Patrick Lincoln, Steven P. Miller, Paliath Narendran, Sam Owre, Sreeranga Rajan, John Rushby, Natarajan Shankar, Jens Ulrik Skakkebæk, Mandayam Srivas, and Friedrich von Henke

SRI-CSL-95-04, July 1995
Formal Verification of a Commercial Microprocessor, Steven P. Miller and Mandayam Srivas.

SRI-CSL-95-05, July 1995
Execution Driven Distributed Simulation of Parallel Architectures, Livio Ricciulli, Patrick Lincoln, and José Meseguer.

SRI-CSL-95-06, May 1995
Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES), Debra Anderson, Teresa F. Lunt, Harold Javitz, Ann Tamaru, and Alfonso Valdes.

SRI-CSL-95-07, May 1995
Next Generation Intrusion Detection Expert System (NIDES): A Summary. Debra Anderson, Thane Frivold, and Alfonso Valdes.

SRI-CSL-95-08, June 1995
Correct Schema Transformations, Xiaolei Qian.

SRI-CSL-95-09, June 1995
Query Folding, Xiaolei Qian.

SRI-CSL-95-11, June 1995
Architecture and Techniques for Fast Computer Tomography, Iskender Agi.

Technical Reports - 1987 – 1994

SRI-CSL-94-01, January 1994
General Logics and Logical Frameworks, Narciso Martí-Oliet and José Meseguer.

SRI-CSL-94-02, April 1994
Semantic Interoperation: A Query Mediation Approach, Xiaolei Qian and Teresa Lunt.

SRI-CSL-94-03, March 1994
Elements of Trusted Multicasting, Li Gong and Nachum Shacham.

SRI-CSL-94-04R, March 1994, Revised October 1994
Adaptive Fault Tolerance: Two Papers Jack Goldberg, Li Gong, Ira Greenberg, and Thomas Lawrence.

SRI-CSL-94-05, April 1994
A Formal Approach to Correct Refinement of Software Architectures, Mark Moriconi, Xiaolei Qian, and R.A. Riemenschneider.

SRI-CSL-94-06, July 1994
GLU Programmer's Guide Version 0.9, R. Jagannathan and Chris Dodd

SRI-CSL-94-07, April 1994
Action and Change in Rewriting Logic, Narciso Martí-Oliet and José Meseguer.

SRI-CSL-94-08, May 1994
Authentication, Key Distribution, and Secure Broadcast in Computer Networks Using No Encryption or Decryption, Li Gong
Also included: Using One-Way Functions for Authentication, Li Gong
Secure, Keyed, and Collisionful Hash Functions, Thomas A. Berson, Li Gong, and T. Mark A. Lomas.

SRI-CSL-94-10, October 1994
Transformations in High Level Synthesis: Formal Specification and Efficient Mechanical Verification, P. Sreeranga Rajan.

SRI-CSL-94-11, May 1994
Specification, Transformation, and Programming of Concurrent Systems in Rewriting Logic, Patrick Lincoln, Narciso Martí-Oliet and José Meseguer.

SRI-CSL-94-12, Aug. 1994
A Policy Framework for Multilevel Relational Databases, Xiaolei Qian and Teresa F. Lunt.

SRI-CSL-94-14, Oct. 1994
Fail-Stop Protocols: An Approach to Designing Secure Protocols, Li Gong.

SRI-CSL-94-15, Oct. 1994
Efficient Network Authentication Protocols: Lower Bounds and Optimal Implementations, Li Gong.

SRI-CSL-93-01, March 1993
Critical System Properties: Survey and Taxonomy, John Rushby.

SRI-CSL-93-02, March 1993
A Formally Verified Algorithm for Interactive Consistency under a Hybrid Fault Model, Patrick Lincoln and John Rushby.

SRI-CSL-93-03, December 1993
Multidimensional Problem Solving in Lucid, A.A. Faustini and R. Jagannathan

SRI-CSL-93-04, May 1993
Eight Papers on Formal Verification, Patrick Lincoln, Sam Owre, Natarajan Shankar, John Rushby, and Friedrich von Henke.

SRI-CSL-93-05, August1993
Rewriting Logic as a Logical and Semantic Framework, Narciso Martí-Oliet and José Meseguer.

SRI-CSL-93-06, November 1993
A Model-Theoretic Semantics of the Multilevel Secure Relational Model, Xiaolei Qian.

SRI-CSL-93-07, November 1993
Formal Methods and the Certification of Critical Systems, John Rushby.

SRI-CSL-93-08, December 1993
A Lazy Approach to Compositional Verification, by N. Shankar

SRI-CSL-93-09, December 1993
Abstract Datatypes in PVS, by N. Shankar

SRI-CSL-93-10, December 1993
A Duration Calculus Proof Checker: Using PVS as a Semantic Framework, by J.U. Skakkebæk and N. Shankar

SRI-CSL-93-11, December 1993
Linear Logic and Proof Theory: Three Papers, by Patrick Lincoln, Andre Scedrov, Natarajan Shankar, and Timothy Winkler

SRI-CSL-93-12, December 1993
Microprocessor Verification in PVS: A Methodology and Simple Example, by David Cyrluk

SRI-CSL-93-13, December 1993
The Complexity and Composability of Secure Interoperation, by Li Gong and Xiaolei Qian.

SRI-CSL-92-01, July 1992
Formal Verification of an Oral Messages Algorithm for Interactive Consistency, by John Rushby.

SRI-CSL-92-02, July 1992
Noninterference, Transitivity, and Channel-Control Security Policies, by John Rushby.

SRI-CSL-92-03, March 1992
Introducing OBJ, by Joseph A. Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud.

SRI-CSL-92-04, March 1992
Survey of Object-Oriented Analysis/Design Methodologies and Future CASE Frameworks, by Donovan Hsieh.

SRI-CSL-92-05, April 1992
A Real-Time Intrusion-Detection Expert System (IDES), by Teresa F. Lunt, Ann Tamaru, Fred Gilham, R. Jagannathan, Caveh Jalali, and Peter G. Neumann.

SRI-CSL-92-06, May 1992
Multiparadigm Logic Programming, by José Meseguer.

SRI-CSL-92-07, July 1992
Simulation and Performance Estimation for the Rewrite Rule Machine, by Hitoshi Aida, Joseph A. Goguen, Sany Leinwand, Patrick Lincoln, José Meseguer, Babak Taheri, and Timothy Winkler.

SRI-CSL-92-08, July 1992
A Logical Theory of Concurrent Objects and its Realization in the Maude Language, by José Meseguer.

SRI-CSL-92-09, Sept. 1992
On the Semantics of Place/Transition Petri Nets, by José Meseguer, Ugo Montanari, and Vladimiro Sassone.

SRI-CSL-92-11, Sept. 1992
Using Rewriting Logic to Specify, Program, Integrate, and Reuse Open Concurrent Systems of Cooperating Agents, by José Meseguer, Kokichi Futatsugi, and Timothy Winkler.

SRI-CSL-92-12, November, 1992
Mechanized Verification of Real-Time Systems Using PVS, by N. Shankar.

SRI-CSL-92-13, December, 1992
GLu: A Hybrid Language for Parallel Applications Programming, by R. Jagannathan and A.A. Faustini.

SRI-CSL-92-14, December, 1992
Solving the Inheritance Anomaly in Concurrent Object-Oriented Programming, by José Meseguer.

SRI-CSL-92-15, December, 1992
A Logical Semantics for Object-Oriented Databases, by José Meseguer and Xiaolei Qian.

SRI-CSL-91-01, February 1991
Multilevel Security for Knowledge Based Systems, by Thomas D. Garvey and Teresa F. Lunt.

SRI-CSL-91-02, February 1991
An Introduction to Formal Specification and Verification Using EHDM, by John Rushby, Friedrich von Henke, and Sam Owre.

SRI-CSL-91-03, June 1991
Formal Specification and Verification of a Fault-Masking and Transient-Recovery Model for Digital Flight-Control Systems, by John Rushby.

SRI-CSL-91-04, June 1991
Mechanical Verification of a Schematic Protocol for Byzantine Fault-Tolerant Clock Synchronization, by N. Shankar.

SRI-CSL-91-05, February 1991
Conditional Rewriting Logic as a Unified Model of Concurrency, by José Meseguer.

SRI-CSL-91-06, March 1991
Final Algebras, Cosemicomputable Algebras, and Degrees of Unsolvability, by Lawrence S. Moss, José Meseguer, and Joseph A. Goguen.

SRI-CSL-91-07, April 1991
From Petri Nets to Linear Logic Through Categories: A Survey, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-91-08, November 1991
Parallel Programming in Maude, by José Meseguer and Timothy Winkler.

SRI-CSL-90-01, February 1990
Duality in Closed and Linear Categories, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-90-02R, February 1990, Revised June 1990
Rewriting as a Unified Model of Concurrency, by José Meseguer.

SRI-CSL-90-03R, February 1990, Rev. Dec. 1990
Compiling Concurrent Rewriting onto the Rewrite Rule Machine, by Hitoshi Aida, Joseph Goguen, and José Meseguer.

SRI-CSL-90-04, August, 1989
Secure Knowledge-Based Systems, by Teresa F. Lunt and Jonathan K. Millen

SRI-CSL-90-06, June, 1990
Order-Sorted Algebra Solves the Constructor-Selector, Multiple Representation and Coercion Problems, by José Meseguer and Joseph Goguen.

SRI-CSL-90-07, July, 1990
A Logical Theory of Concurrent Objects, by José Meseguer.

SRI-CSL-90-08, August, 1990
Decision Problems for Propositional Linear Logic, by Patrick Lincoln, John Mitchell, Andre Scedrov, and Natarajan Shankar.

SRI-CSL-90-09, October, 1990
On Location: Points about Regions, by Judith S. Crow and Peter B. Ladkin.

SRI-CSL-90-10, October 1990
On the Design of Dependable Computer Systems for Critical Applications, by Peter G. Neumann.

SRI-CSL-90-11, November 1990
The GLU Programming Language, by R. Jagannathan and A.A. Faustini

SRI-CSL-90-12, November 1990
Axiomatizing the Algebra of Net Computations and Processes, by Pierpaolo Degano, José Meseguer, and Ugo Montanari.

SRI-CSL-90-13, December 1990
Semantic Specifications for the Rewrite Rule Machine, by Joseph A. Goguen.

SRI-CSL-90-15, December 1990
A Logic to Unify Semantic Network Knowledge Systems with Object-Oriented Database Models, by Donovan Hsieh.

SRI-CSL-90-16, December 1990
Inclusions and Subtypes, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-90-17, December 1990
Architectural Design of the Rewrite Rule Machine Ensemble, by Hitoshi Aida, Sany Leinwand, and José Meseguer.

SRI-CSL-89-1, January 1989
An Intensional Parallel Processing Language for Applications Programming, by E.A. Ashcroft, A.A. Faustini, and R. Jagannathan.

SRI-CSL-89-2, January 1989
Dataflow-based Methodology for Coarse-Grain Multiprocessing on a Network of Workstations, by R. Jagannathan, Alan Downing, William Zaumen, and Rosanna Lee.

SRI-CSL-89-3R, February 1989, Revised August 1991
Formal Verification of the Interactive Convergence Clock Synchronization Algorithm using EHDM, by John Rushby and Friedrich von Henke.

SRI-CSL-89-4R, March 1989, Rev. June 1989
From Petri Nets to Linear Logic, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-89-5, March 1989
General Logics, by José Meseguer.

SRI-CSL-89-6, March 1989
The Rewrite Rule Machine Project, by Joseph Goguen, José Meseguer, Sany Leinwand, Timothy Winkler, and Hitoshi Aida.

SRI-CSL-89-8, July 1989
A Categorical Manifesto, by Joseph A. Goguen.

SRI-CSL-89-9, Sept. 1989
A Practical Approach to Semantic Configuration Management, by Mark Moriconi.

SRI-CSL-89-10, July 1989
Order-sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations, by Joseph A. Goguen and José Meseguer.

SRI-CSL-89-11, December 1989
An Algebraic Axiomatization of Linear Logic Models, by Narciso Martí-Oliet and José Meseguer.

SRI-CSL-88-1, January 1988
Higher Order Functions Considered Unnecessary for Higher Order Programming, by Joseph Goguen.

SRI-CSL-88-2R2, January 1988, Rev. 2, August 1988
What is Unification? A Categorical View of Substitution, Equation and Solution, by Joseph Goguen.

SRI-CSL-88-3R, January 1988, Rev. June 1989
Petri Nets Are Monoids, by José Meseguer and Ugo Montanari.

SRI-CSL-88-4R2, August 1988
OBJ as a Theorem Prover with Applications to Hardware Verification, by Joseph Goguen.

SRI-CSL-88-5, May 1988
A Descriptive and Prescriptive Model for Dataflow Semantics, by R. Jagannathan.

SRI-CSL-88-7R, Sept. 1988
Quality Measures and Assurance for AI Software, by John Rushby.

SRI-CSL-88-10R, Sept. 1988, Rev. Jan. 1989
Cell, Tile, and Ensemble Architecture of the Rewrite Rule Machine, by Joseph A. Goguen, Sany Leinwand, and Timothy Winkler.