NERD
Network Event Recording Device: An Automated System for Network Anomaly Detection and Notification

David G. Simmons and Ronald Wilkins
Network Engineering
Los Alamos National Laboratory
Los Alamos, NM 87545

Abstract

The Network Event Recording Device is an automated, real-time system for monitoring and detecting network anomalies, as well as providing timely notifcation to network maangers of significant network events. The NERD system allows for continuous monitoring of system and security logs, easily configurable notification options and a central, secure data collection point for distributed system logs.

1 Overview

The goal of the Network Event Recording Device (NERD) is to provide a flexible, reliable, and autonomous system for network logging and notification when significant network anomalies occur. The NERD is also charged with increasing the efficiency and effectiveness of currently implemented network security procedures. While it has always been possible for network and security managers to review log files for evidence of network irregularities, the NERD provides real-time display of network activity, as well as constant monitoring and notification services for managers. Similarly, real-time display and notification of possible security breaches will provide improved effectiveness in combating resource infiltration from both inside and outside the immediate network environment.

1.1 Background

The Network Event Recording Device, or NERD, was originally begun by Sally Wilkins, Craig Idler, and Ben Crane in May, 1991. The originally defined goals of the NERD were to be "a service on a network for recording network events and affecting some appropriate notification that is determined by the criticality of the events. NERD may be a stand alone host or may be one of several network services on a host dedicated for such tasks" [11]. While the basic functional description and purpose of the NERD has not changed substantially since that time, the implementation and features of the NERD have undergone radical revisions in order to make it a flexible, production quality system addressing the ever-widening problems of network management and security. The rewrite of the NERD, including the development of several new and important aspects of the system, was completed at Los Alamos National Laboratory by David G. Simmons under the direction of Ron Wilkins and Dale Land in the Network Engineering Group (CIC-5). Ongoing work to continuously upgrade the capabilities and services provided by the NERD, as well as to address new and emerging network and security issues, is also being conducted.

1.2 Environment

The target environment for the NERD System is the Integrated Computing Network (ICN) at Los Alamos National Laboratory. The ICN is the central computing network, serving over 9000 users on a variety of supercomputers, mainframe, minicomputers and workstations, as well as file storage devices, communications interfaces, routers, bridges and terminals. The variety of operating systems present in the ICN was a central driving force in the final design of the NERD System.

In addition to hardware variations, the ICN is divided into several separate partitions based on four defined security levels. At this writing, the ICN is undergoing a major reorganization (the ICN2 Project) to further isolate these partitions and provide greater security against intrusion. In so doing, however, it has also complicated the task of centralized network monitoring and notification by requiring a complete physical separation between networks processing classified data, and those processing unclassified material, and thus a need for simultaneous services on the separate networks [2].

1.3 Functional Overview

The NERD is not necessarily, as the name might imply, a single entity, but is rather a suite of programs that, when run together, constitute a sophisticated Network Event Monitoring System, a Notification System for significant Network Events, and a self-monitoring and diagnostic system to ensure reliable operation. The system also provides an intuitive, interactive, window-based interface to the information maintained by the NERD.

In order to minimize system maintenance requirements and to allow a high degree of portability, the NERD is built on a standard Berkeley System Designs 4.3+ UNIX **syslogd** daemon process, making the NERD effectively blind to hardware and operating system differences across a heterogeneous network such as the ICN at Los Alamos. The modified process is run only on the NERD data server, allowing the NERD to provide the full range of its services to all hosts, or a specified subset of hosts, without remote software modifications.

The NERD provides host-authentication routines which allow a high degree of granularity in the control of access to the various logging capabilities allowed to remote hosts, notification abilities granted to remote hosts, and access to the NERD's log files. NERD uses the network log files generated on the various service nodes to monitor network activity. Since the NERD uses existing sources of data, we are able to incorporate new clients with little impact on the client systems.

The Authentication, Logging and Notification system, and the User Interface to those systems will be described here. The suite of support processes written to ensure reliability of the NERD will not be described.

1.4 Motivating Factors

The NERD project is seen as a significant opportunity to increase efficiency and drive down costs in the areas of network management as well as network resource allocation. However, a significant aspect of the NERD's capabilities lie in increasing the effectiveness of present and future network security procedures through providing a flexible notification interface.

2 Network Efficiency

The economic benefits of improved network performance and security are well recognized [5]. Improved efficiency in managing networks, and network resources, has the potential to decrease the financial impact of network maintenance and management on network providers and network managers.

The problem of effective network management applies equally to large-scale computing facilities and Local and Wide Area Networks. A simple example of network resource mismanagement is an infinite loop in email forwarding, creating a "cycled user." A user forwards his mail from machine X to machine Y. Forgetting he has done this, he later forwards his mail from machine Y to machine X. Though most mail protocols will age messages and discard them, the effect of large numbers of cycling email messages on a network can be substantial. A security profiler may also see such activity as a potential security violation, and take unwarranted action against a user or a series of hosts on the network to curtail the activity [6]. In this way, resource degradation on even small Local Area Networks of workstations can significantly reduce productivity and drive up the cost of doing business.

2.1 Security Enhancement

An important, and often overlooked, avenue to increased security lies in the effective management of networks and their associated resources. Even using the most advanced network security technology available will not guarantee adequate security unless effective network management procedures are developed and followed. As with resource management, security management has the potential to directly influence the cost of maintaining and providing network services. Furthermore, even small breaches of security can lead to large-scale financial losses to network providers [9,3].

Given the growing presence of open systems, a large majority of which are based on Unix, and the inherent security weaknesses in Unix, providing effective means of security management in distributed, open systems becomes an increasingly important commitment. It is therefore imperative to implement a network-wide system of auditing and control to ensure adequate security and management of users and processes on open systems [7,1]. While much of this functionality already exists, often the data generated and stored in log files is never adequately examined due to a lack of an effective interface to those files.

2.2 Analysis of and Integration with Existing Systems and Software

In designing a new network monitoring system, we felt it was imperative to design a system that could be rapidly integrated into existing network monitoring technology. Such a design approach would upon implementation, maximize the actual use of the system, and minimize the impact on users.

In order to make remote system modification unnecessary, the NERD is based on a modified Berkeley UNIX **syslogd** (system logging daemon) process. By using a standard, well-known, and widely used network logging system, the NERD is able to be integrated quickly into an existing network structure without large-scale modifications to that structure. By running the modified **syslogd** process only on one machine, no software modifications are required network-wide.

Additionally, by using a widely available and well-understood mechanism, we were able to provide managers with an easy interface to using and customizing their use of the NERD. The NERD was designed to continuously monitor and log all network events on any machine or set of machines without significant changes to those machines.

2.3 Cross-platform Adaptation

In order to deal effectively with the wide variety of hardware platforms and operating systems present in the ICN, it has at times been necessary to implement different software for handling data from the various service nodes on the network [1]. For the purposes of the NERD, this approach was seen as placing too high a burden on both the administrators of the various nodes, and those maintaining the NERD software in the future. For this reason an approach was pursued which would require no specialized software for differing hardware platforms and operating systems. Instead, we rely on the wide availability of UNIX and UNIX-like processes to handle the differences among hardware and operating system platforms. This system-independent approach was not a

part of the original design of the NERD, but rather was implemented as the design parameters changed [1].

2.4 System Autonomy

By designing the NERD's operation to be independent of the presence or absence of any specific service nodes within the ICN, the NERD is able to function regardless of the status of any segment of the network environment, with the exception of the network connection of the NERD itself to the ICN backbone. Should a service node, or set of nodes, either cease operation, or suddenly begin utilizing the NERD, no significant changes need be made to the NERD's configuration. As new systems are brought into the ICN, they can be integrated into NERD without interruptions in existing services to other nodes.

2.5 Data Handling and Authentication

Authenticating and handling the large volume of data generated by the networks currently utilizing the NERD has proven to be a larger issue than at first imagined. The volume of data which the NERD handles has increased steadily as new systems are brought into service, and we have been forced to adapt our data handling procedures to accommodate these changes.

All NERD data transmissions are handled through a standard set of well-known ports. From remote hosts, the NERD looks for messages on UDP port 514 [8]. Additionally, the NERD listens for locally generated messages on /dev/log, and for messages generated by the kernel itself on /dev/klog. The scheme used by NERD represents no change from the implementation provided by BSD 4.3+ UNIX **syslogd**, and is used for that reason. These input routes are all opened at start-up, and are continuously polled for incoming messages.

All information sent to the NERD is stored in log files maintained by the NERD. The NERD also stores all of its own events in these log files. The formats of these files will be discussed in Sections 4.3 and 5.3. All NERD log files are backed up both locally and to the Common File System, the Laboratory's main file storage system, to ensure an accurate archive of network data, and the integrity of current network data in the event of a system crash.

2.6 Host Authentication

Upon receipt of a message from one of the input sources, the sending host is extracted from the data packet header, and is verified against the NERD's list of allowed hosts. Only hosts listed in the authorization file (/etc/nerdhosts) are permitted to log events to the NERD. This is a deviation from both the standard implementation of **syslogd**, which allows any incoming message to be logged, and from the original design specifications of the NERD, which would have followed the **syslogd** implementation. The policy change was made in order to prevent unauthorized access to the NERD as well as to better control the amount of data logged by the NERD.

In the event of an attempted logging from an unauthorized host, the attempt, including the hostname making the attempt, is logged by the NERD, and the message is then discarded. All messages are time- and host-stamped for identification purposes.

The authentication routine implemented by the NERD currently relies on the integrity of the underlying transmission protocol (in this case, UDP) for trustworthy host identification. Authorized hosts are kept in memory, and all incoming messages are validated against this list. For performance reasons, an additional check for allowed Notification functions (Section 3) is also performed during this authentication routine.

2.7 Data Authentication

In order to provide the needed consistency in the data gathered by the NERD, checks are run on all incoming messages to ensure the validity of the message and its contents. In the case of messages that are found to be invalid the NERD attempts to reformat the message into a usable form. For example, messages containing control or other non-printable characters are converted. Control characters are converted to their ASCII equivalent (^D, for control-D) so that managers can be aware of attempts to circumvent security measures by sending control strings across the network. Corrupt or missing time-stamps are also repaired and hostnames are appended to all messages to make clear the source of all messages. When known, the process id generating the message is also added to the message in order to make network debugging and problem resolution easier.

3 Notification

The Network Event Recording Device is designed to notify Network Managers of significant network events via electronic mail, digital pager, public address announcements, and video displays. Once validated for particular services by the NERD, a system manager can customize his or her system to provide varying degrees of notification should significant events occur. Again, this customization can, in all but the most elaborate cases, be performed without specialized software. In the event that an important Network Event occurs, and a System Manager has not specified a notification procedure, the Network Operations Center (CIC-5) on-call operator will be notified of the event.

3.1 Notification Options

Currently the NERD provides four levels of notification to authorized hosts: Announcement through DECTalker Synthesized speech, paging through the Los Alamos CIC Alphanumeric Pager Interface (CAPI), electronic mail, and display through the NERD interface (Nerdint) Graphical User Interface tool. Notifications are generated based on the mapping shown in Table 1. The levels are Facility and Priority levels as defined in <sys/syslog.h>.

3.2 Host Authentication

As with host authentication for incoming messages, hosts are also authenticated for allowed Notification services. Hosts that attempt to access a Notification Service for which they are not authorized are denied that service, and the attempt is logged by the NERD. Further processing of that message is then terminated. As described earlier, the authorization level of the

Facility	Priority	Action
LOG_LOCAL7	LOG_EMERG	Public Address Announcement
LOG_LOCAL7	LOG_WARN	CAPI Page
LOG_LOCAL7	LOG_NOTICE	Electronic Mail
ANY	ANY	Video Display

Table 1: Notification mappings

sending host is determined during the host authentication procedure. This authorization level is then consulted at the time when Notification is attempted. Most Notification Services have a default action (or the option of implementing a default option) in order to avoid lost Notifications.

3.3 General Message Format

In order for a host to initiate Notification services, the incoming message must be formatted in such a way that the NERD can determine both the Notification Service requested, and the designated recipient of the Notification Service. Since Notification Services are mapped by syslog facility and priority levels, this determination is provided by standard **syslogd** functions. Designated recipients, if any, are determined through routines specifically designed to perform the Notification Service.

In order to determine the desired recipient of the Notification, most Notification requests need to include a recipient address (described in the relevant sections below) followed by the designated delimiter character set '<?>'.

3.4 Notification Through Public Address Announcement

The NERD is capable of supporting up to 4 DECTalkers, allowing for announcements in different 'zones' of a facility. Announcement zones are configurable through a talker configuration file, allowing for changes in zones based on identified usage patterns. Multiple zones can be used simultaneously by OR-ing zone identifiers together in the Notification Recipient field of the message. By default, if a message from an authorized host, of the corresponding Facility and Priority levels is received, without a specified zone number, the NERD will announce the message to a defined default-zone.

3.5 Notification Through CIC Alphanumeric Pager Interface (CAPI)

Another of the key requirements of the NERD is the ability to notify responsible network managers in a timely manner of significant network events that might require attention. The NERD's interface to the CAPI pager system provides an extremely rapid response time to an urgent network situation. In order to make the system as useful as possible, the NERD provides the full text of the error message to the manager specified in the Notification Recipient field of the message. The NERD does no verification of the validity of the Recipient field, instead relying on the CAPI database to provide that service. Invalid pager identifiers are simply dropped by the CAPI system. In the event that no recipient is specified, the NERD will attempt to send a page to a default address defined in a configuration file. By allowing the NERD to accept a default pager address from a file, the NERD is able to follow an on-call system where different managers are responsible at different times of the day or week.

3.6 Notification Through Electronic Mail

Electronic mail notification can notify a single individual, or a list of responsible parties, when lower level network events occur. Mail recipients are specified in the Notification Recipient field of the incoming message. Again, the Recipient field of the message is not validated, relying instead on sendmail to deal with invalid mail addresses. Through a configuration file, a default mail address can be specified, allowing all messages of Facility LOG_LOCAL7 and Priority LOG_NOTICE to generate an email notification.

3.7 Notification Through Video Display

Through the NERDINT (Network Event Recording Device INTerface) messages received by the NERD can be viewed in real-time by system managers. Any message of any Facility or Priority level from a host authorized for video display will be sent to the video support process and will subsequently be available for examination on a NERDINT. In addition, all Notification Services, as described above, are logged to the video support process. The video display and associated processes will be described more fully in Section 5.

4 Implementation

The Network Event Recording Device is based on a modified Berkeley Systems Design (BSD) **syslogd** process. Though significant changes were made in the **syslogd** code, the basic functionality of the original **syslogd** process was maintained in the interest of providing both consistent service and a consistent interface for users. As with the original process, all modifications and additions were done using the C programming language.

A potential flaw in the original code was also uncovered, and significant effort was directed towards correcting what appears to be a non-standard approach to handling interprocess and network communications by **syslogd**.

4.1 Alterations to Basic Functionality

All basic functions of the original **syslogd** process have been maintained, though some have been altered.

In order to help create a more detailed audit trail of incoming events, changes were made in the way in which **syslogd** records and formats messages, as well as how messages to be forwarded to other hosts are treated. Also implemented was a change to **syslogd**'s handling of repeated, duplicate messages, again to improve the ability of managers to effectively track potential network and security problems.

4.2 Message Format

The original **syslogd** process, as provided by BSD, simply follows a configuration file (commonly /etc/syslog.conf) as to the manner in which incoming messages are to be processed. This functionality was maintained in order to provide consistency of service and communication between differing versions of **syslogd**.

Our modifications consisted of ensuring that all messages are tagged with the sending hostname, and preserving the originating process identifier for all incoming and outgoing messages. Hostnames as extracted from the datagram, are appended to all messages in the form ' hostname ' to allow network managers to quickly locate messages from a given set of hosts. (See Figure 1.)

In addition to preserving hostnames, a change was made to the manner in which **syslogd** formats messages before forwarding them to other designated hosts. In the original implementation, **syslogd** processes forwarding a message sent by a peer **syslogd** process first stripped the process identifier from the head of the message before forwarding the message string. This situation resulted in a loss of information on messages forwarded through multiple hosts. By changing this behavior and leaving the originating process identifier intact, a more accurate audit trail of system messages has been preserved.

4.3 File Format

The NERD maintains a log file similar to that maintained by a standard **syslogd** process. The name and location of the log file are different for the NERD process, which requires that log information be kept in the file /var/adm/nerd_log. This file can reside anywhere, as long as appropriate links are established.

A typical message written to the NERD's log file is shown in Figure 1. The originating process (in this case adgate.lanl.gov's syslog process) is kept, as is the id of the machine which generated the error (in this case, colman.lanl.gov).

4.4 Message Processing

After authentication, messages are processed by the NERD based on facility and priority levels. Duplicate messages are suppressed. This suppression is similar to standard **syslogd** protocol. Unlike standard **syslogd**, where a message stating "Last message repeated x times" is generated, the NERD preserves the repeated message, logging the message "The message <message text> was repeated x times." This functional enhancement was added to the NERD in order to make senders of repeated messages easier to track, isolate and control.

4.5 Notification Procedures

After an authenticated incoming message has been logged to the appropriate log files, and any message forwarding has been completed (as defined in the configuration file) the NERD's specialized notification procedures are invoked. Allowed services for each host are stored in a 16 bit integer, and are extracted by ORing the requested service with the allowed services for the given host.

After this second round of authentication, an allowed request is executed by subroutines for each service. For notification services requiring a Notification Recipient Field (NRF) messages are parsed to extract the required address. In the event that no NRF is found, a default address is attempted. If no default address is defined or available, the message is then discarded and control is returned.

Access to the DECTalkers is controlled through direct access to serial ports on the NERD. DECTalkers connected to the serial ports translate the incoming message and broadcast over an attached Public Address system. The NERD supports up to 4 distinct zones, though the number of zones possible is limited only by the number of serial ports available on the host machine.

Email and CAPI access are done via direct access to the host sendmail process. In order to prevent the transmission of restricted or classified data through the NERD, a SECURE compile flag is provided. If set at compile time, only a predefined (using #define) mail message can be sent by the NERD. Through this mechanism, we eliminate the possibility of transmission of restricted or classified data outside the secure partition of the ICN by the NERD. We felt that hard-coding certain aspects of the interface to electronic mail would make infiltrating and compromising the NERD system more difficult for possible intruders. Finally, success or failure of the email (or CAPI) routine is logged to the NERD itself, showing the intended recipient, and the included message.

Following the completion of the above routines, control is returned to the main event handling loop for processing further messages.

5 User Interface

The Network Event Recording Device provides a Graphical User Interface for real-time monitoring of current network events as well as reviewing past network logs. The interface, called Nerdint, provides a live connection to the NERD giving the user an up-to-date view of network activity. This interface is customizable by each user, allowing managers to view almost any redefinable subset of network events as they occur. By running this interface, a Network Manager can immediately see color-coded priority messages relating to her system.

In addition to providing real-time monitoring of Network Events, the Nerdint allows Network Managers to notify others of significant network events directly from within the Nerdint. In order to make notification easier, the Nerdint provides an interface to the Los Alamos Trouble Ticket System, the LANL Digital Paging system (CAPI), and standard electronic

```
Aug 9 12:50:00
adgate.lanl.gov syslog : colman.lanl.gov : 109586 AUTHENTICATE successful. {adgate.lanl.gov}
```

Figure 1: nerd_log file format.

mail. Users can forward highlighted messages within the NERD to any of these systems without leaving the Nerdint windowing system.

6 Current Usage

At the present time, the NERD is being used to log a variety of systems throughout the CCF and to provide specific notification features to each of these systems.

Also within the CCF, the NERD provides audit-trail logging of connections into and out of the Terminal Internet Gateways (TIGs), logging all incoming and outgoing connections, validations, and failures. This logging provides a security audit trail of all CCF access from dial-ups and remote sites. All kerberos authentication requests, completions, and failures are also logged in order to provide a complete audit trail.

The Desktop Group, responsible for administering LAN services throughout the Laboratory, is presently using the NERD to continuously monitor the wide variety of Local Area Networks for which it provides network support. System Managers within the group are using the NERD software package to provide central logging and notification for a large and complex system of LANs throughout the Laboratory.

7 Conclusions

In the course of development and implementation of this complex software system, we encountered several obstacles, as well as some interesting possible defects in legacy code. The difficulties we encountered, our solutions and rational for those solutions will be discussed in this section, along with our findings of possible defects in legacy code.

7.1 Implementation Problems

One of the most difficult problems we encountered in the development of the NERD was the ability of the communications protocols to deal with large amounts of traffic, often consisting of large messages. The original design relied on connected TCP sockets for all interprocess communication. While this system worked well in the test situation, its shortcomings were seen immediately upon implementation in a busy production environment. As the number of systems depending on the NERD for event recording and notification increased, the rate of failure increased as well. Since failures resulted in loss of network data, this situation was unacceptable, and a new approach to interprocess communications was pursued.

With the implementation of the User Datagram Protocol for all communications between **syslogd** and the **driver** process the degradation in system reliability has disappeared. While an unburdened (logging only local events, or logging for a limited number of hosts) **syslogd** process tends to have a network input queue length of $\emptyset$ bytes, the NERD, when fully loaded under the TCP communication scheme, kept an average input queue length of greater than 2000 bytes. Since the implementation of the UDP Communication Protocol on the NERD, we have seen an average input queue length, when the system is fully loaded, of less than 100 bytes, indicating that the new communications approach has significantly reduced the risk of data loss

7.2 Code Inconsistencies

In Berkeley Systems Design **syslogd**, a potential bug was discovered in the way in which that process handles open socket descriptors for incoming messages. We have been able to find no rationale for the way in which BSD code handles this situation. The BSD code used integers, and the macro FDMASK to set file descriptors, and passes this value to the select() system call, blocking until one or more of the opened file descriptors is ready for I/O. While this approach appears to work in most cases, this scheme often results in a large number of open file descriptors, eventually leading to a failure of the process, W. Richard Stevens [10] presents a method of I/O multiplexing for reading from multiple socket inputs that is more robust, allowing for reading on multiple socket descriptors while preventing blocking, and avoiding the open file descriptor limit.

While the original code provided by BSD4.3 for multiplexing the I/O for **syslogd** appears to have no significant, obvious flaws, when the load on the NERD increased dramatically, we saw a rapid rise in the number of unexplained data alignment errors and segmentation violations, traced back to the I/O routines. We have yet to arrive at a definite reason for why the errors began to occur and increased in frequency as the load on the process increased.

However, since replacing the multiplexing I/O code with W. Richard Stevens' approach, we have seen no alignment or segmentation violations originating from the corresponding section of code [10]. Again, we have not reached a definitive conclusion as to why these errors occurred or why the change in multiplexing approaches appears to have solved the problem. A complete cessation of alignment and segmentation violations is, we believe, fairly compelling evidence to support our change.

According to the original author of **syslogd**, Mr. Eric Allman, the FD_SET() macro abstraction which we used to solve the problem was not available under BSD4.2. This left **syslogd** with a limit of 32 file descriptors. It was, apparently, this limit which caused the difficulties discussed above. We have communicated our solution to Mr. Allman, and he has indicated that it will be incorporated into future releases of **syslogd**. We would like to acknowledge Mr. All-

man's time and input in providing his thoughts and reasoning on this issue. For more information on the NERD, contact:

David G. Simmons or Ronald Wilkins
Network Engineering
Los Alamos National Laboratory, Mail Stop B255
Los Alamos, NM 87545
davids@lanl.gov, ronw@lanl.gov,
or nerd@nerd.lanl.gov.

References

[1] G. Black, **Update on Open Systems Security**, Computers & Security, Elsevier Science Publishers Ltd., Vol. 11, pp. 699-702.

[2] Final Report of the Design Review Phase of the Independent Verification and Validation of the LANL ICN (Draft), February 15, 1994.

[3] G. Hardy, **Reviewing Logs Using Knowledge Based Systems**, *Proceedings of The Ninth World Conference on Computer Security, Audit and Control*, Elsevier Science Publishers Ltd., November 1992.

[4] J. Hochberg et al. **NADIR: An automated System for Detecting Network Intrusion and Misuse**, *Computers & Security*, Elsevier Science Publishers Ltd., Vol. 12, No. 3, p. 238.

[5] D. Lindsay, *An Overview of Leading Security Issues, Information Security An Integrated Approach*, Information Security An Integrated Approach, J.E. Ettinger, Ed., Chapman & Hall, New York.

[6] U. Manber, **Chain Reactions in Networks**, *Computer*, IEEE Computer Society Publications, Vol. 23, No. 10.

[7] K. Morgan and T. Hamer, **Network Security: Aid to Industry Introduction to Communications Audit and Security**, *Proceedings of The Ninth World Conference on Computer Security, Audit and Control*, Elsevier Science Publishers Ltd., November 1992.

[8] J. Reynolds, J. Postel, **Assigned Numbers**, *Network Working Group Request for Comments: 1060*, March 1990.

[9] S. Rowley, *Management: The key to effective distributed systems security*, Information Security An Integrated Approach, J. E. Ettinger, Ed., Chapman & Hall, New York.

[10] W. Stevens, UNIX Network Programming, Prentice Hall, New York.

[11] S. Wilkins, Network Event Recording Device Functional Description (Draft), May 6, 1991.