Haystack: An Intrusion Detection System

Stephen E. Smaha

Tracor Applied Sciences, Inc.
6500 Tracor Lane
Austin, Texas 78725-2050

ABSTRACT

Haystack is a prototype system for the detection of intrusions in multi-user Air Force computer systems. Haystack reduces voluminous system audit trails to short summaries of user behaviors, anomalous events, and security incidents. This is designed to help the System Security Officer (SSO) detect and investigate intrusions, particularly by insiders (authorized users). Haystack's operation is based on behavioral constraints imposed by security policies and on models of typical behavior for user groups and individual users.

1.0 BACKGROUND

Three initiatives are changing the computer security environment:

- The development of explicit government standards and criteria (e.g. DOD 5200.28-STD [1]) for computer security;

- The development of multi-vendor operating system standards, like POSIX; and

- The insistence by the government that standards be used for computer procurements (e.g. DOD 5200.28-STD for security standards, GOSIP and forthcoming FIPS standards for POSIX).

1.1 National Computer Security Center (NCSC) Requirements

The NCSC Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, permit objective evaluation of the security of computer systems. For example, these Criteria require that the Trusted Computing Base (TCB) produce an audit trail including security-relevant information starting at the C2 level. Starting at the B3 level, the TCB must also notify the Security Administrator when audited events exceed predefined thresholds, and act to terminate the events if they continue to exceed thresholds.

1.2 POSIX

The IEEE P1003 POSIX Committee has defined a standard for an operating system interface. This interface standard has been proposed by the National Bureau of Standards to become a Federal standard for future computer acquisitions. The IEEE POSIX Security Working Group (P1003.6) is currently developing security extensions to the POSIX standards effort, including security auditing, discretionary and mandatory access controls, and access control lists. Their goal is to produce a system that is rated B1 by NCSC criteria. Since the government intends to make future systems procurements adhere to POSIX standards, this Working Group's efforts should be closely watched for impacts on emerging security architectures.

1.3 Insider Threat

Most computer security experts believe that the vast majority of computer crimes is committed by an organization's authorized computer users, its insiders. This is true both in commercial systems and in sensitive government computers. Within the Air Force, Paul Boedges, head of the Air Force Office of Special Investigations Computer Crime Division, reports [2] that 80 percent of known computer crimes were carried out by trusted employees, including government contractors. The current emphasis on access controls (like dial-back modems) and protecting against "hackers" does not deal with the high incidence of computer crimes committed by insiders. In any event, the distinction between insider and outsider becomes moot once an outsider has successfully logged onto or otherwise penetrated a computer.

1.4 Implications

We have noted several important initiatives imposing security constraints on the design and operation of computer systems. As computer system software and hardware improve in their ability to embody a security policy and to detect and respond to security policy violations, a greater proportion of the organization's security needs can be enforced by the system.

We can expect that future systems will better deal with threats to security.

However, these initiatives do not address the techniques needed by security administrators to deal with current systems. They also do not help administrators interpret the huge volume of audit trail information generated by audit mechanisms in current systems. This paper describes an approach to improving the security of current systems as well as those systems which will become operational in the future by analyzing system audit trails.

2.0 INTRODUCTION

Haystack is a system for the detection of intrusions in multi-user computer systems. It is currently under development as part of the Communications-Computer Security Program, sponsored by the Air Force Cryptologic Support Center, Kelly Air Force Base, San Antonio, Texas. The developer is Tracor Applied Sciences, Inc., Austin, Texas.

Haystack reports on anomalous events in each day's audit trail files, and analyzes user activity against predefined security constraints and models of typical user behavior.

We define an intrusion to be the inappropriate use of a computer: that is, use of the computer that violates the security and administrative policies established for that computer. Unlike many commercial computers, the operational military systems towards which Haystack is targeted have an explicit security policy which states how the computer may be used and likewise states prohibitions for its use.

2.1 The Target Computer System

The target computer system is a Unisys (Sperry) 1100/60 mainframe running the OS/1100 operating system. This system is used by all Air Force Bases as a standard computing platform. OS/1100 has complex security monitoring features available and many attributes of a Trusted Computing Base which could be rated at the C2 level according to NCSC criteria. Current security policies take advantage of some, but not all, of these security features.

There is little external networking in this environment excluding occasional dial-in use under dial-back control monitored by the system operator. The 1100/60 is mostly used for conventional data processing applications like those run on IBM mainframes in the private sector. The typical 1100/60 audit trail consists of about a million recorded events per week, and includes accounting, performance monitoring, crash recovery, and security related information. The 1100/60 normally handles unclassified but sensitive (Privacy Act) data. The primary threats to the data on this machine result from data aggregation (accumulating sufficient unclassified information to allow deriving conclusions equivalent to classified information), violation of privacy rights, alteration of key logistical data, and financial fraud.

2.2 The Organizational Context: The Big Picture

The target system is an operational military computing resource where security is a major consideration. Since Haystack is intended to augment, not replace, existing analysis capabilities and security personnel, its design must consider a "big picture" that includes the entire security and investigative process, as well as the information handling needs of the organization.

Figure 1 shows the use of audit trail information within the entire investigative process, starting from raw data (simple events occurring on the target system, like logins, file accesses, or changes in the security state of the system), and ending in a detected security violation that has been validated by standard investigative procedures.

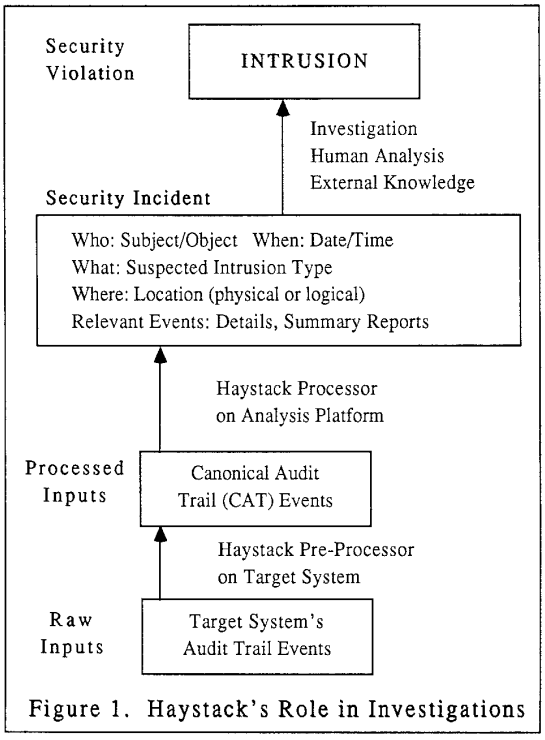

Figure 1. Haystack's Role in Investigations

3.0 IMPORTANT CONCEPTS

Because Haystack is a new system in a relatively new area of research and development, it reflects a particular understanding of its underlying problem domain of intrusion detection and its intended security-conscious military environment. There are several important concepts behind its development.

3.1 How To Detect Intrusions

The process of detecting intrusions on computer systems shares properties with other forms of detection and enforcement.

3.1.1 One method of intrusion detection is to look for behavior that is not good or typical, given some definition or measure of "good" or "typical". For example, if a user's print output suddenly increases by 2000% and its absolute value is also large, the Security Officer might become suspicious, and investigate the possibility of information leakage. Of course, the user might have valid reasons for such behavior, like a special year-end report. The SSO's investigative process is intended to look into such possibilities.

3.1.2 Another method of intrusion detection is to look for behavior that is "bad" by definition. For example, if a user tries to print the password file, the Security Officer should investigate this behavior, whether or not the user had authorization to read that file. Disadvantages of this approach are that it can only find pre-programmed "bad" behavior. "Creative malice" may go undetected.

Ideally, the computer security model, as adapted by the organization, and the operating system's security capabilities could together be used as a basis for logically deriving at least some of the definitions and rules that characterize "bad behavior" for an intrusion detection system. Unfortunately, there is no strong consensus on the formal characteristics of security policies in general.

3.2 User Models

To attempt to detect behavior that is not good or typical implies the existence of a user model that defines the norm for that user. There are two general approaches to user modeling, depending whether the user is treated as an individual or a member of a user class. It is likely that a realistic user model will require both approaches.

3.2.1 A user model may be based on information about the individual user's past behavior, giving a norm of "typical" behavior for that user.

A model based on knowledge of the user's past behavior is customized for the user, and reflects the customary behavior of that user. It takes into account the fact that different users have very individual use patterns on a system, depending on job assignment, level of experience, work style, and personality.

Disadvantages of this approach are that the model may be modified ("trained") by a cunning user over time. Thus it may have trouble detecting intrusions that do not involve abnormal resource or command utilization. It may also be hard to establish what default behavioral characteristics should be assigned to a new user. The new user may trigger too many suspicion flags, or too few, and the detection mechanism may be ineffective until it has acquired enough information about this new user to form a statistically valid basis. Also, the model may interpret innocent variations in user behavior, like those due to increased skill or changes in job assignments, as suspicious.

3.2.2 The user model may be based on generic notions of acceptable behavior for a group of users. The generic groups should distinguish among different groups of employees, with different expected and permitted behaviors for data entry clerks than for Security Officers or software developers.

Having predefined mutually exclusive categories of users is a refinement of the common understanding of multi-level security. Since the number of user categories will be much smaller than the number of users, it will be easier to maintain a set of user categories.

There are disadvantages to this approach, as well. One user may fill more than one role on the system. A user whose work characteristics are at the edge of applicability of a general group may trigger many false alarms. Also, it is hard to define a conceptual basis for adapting this kind of model to actual usage patterns over time.

3.2.3 Finally, a user model could combine these two approaches, considering both the individual's past behavior and membership in a user group. This is the approach used in Haystack. Such a combined model starts a new user with a "template" profile appropriate to the user's work category. This effectively supplies a hypothetical past for the subject plus the appropriate restrictions on the user due to group membership and work assignments.

3.3 Intrusion Types

Haystack is designed to detect six types of intrusions. Each may be detected in several ways, the most obvious of which are listed.

3.3.1 Attempted break-ins (by unauthorized users) are detected by monitoring login attempts. A successful break-in occurs when an outsider "convinces" the system that he/she is an authorized user by supplying a valid user identification and password. These are detected by atypical behavior profiles or violations of security constraints.

3.3.2 A masquerade attack occurs when the intruder attempts to "convince" the system that he/she is really a user other than the one the system expects him/her to be, presumably one with higher privilege. These are detected by atypical behavior profiles or violations of security constraints.

3.3.3 Penetration of the security control system (where a user attempts to modify the security characteristics of the system, like its passwords or authorizations) is detected by use of privileged logins or privileged system services.

3.3.4 Leakage (causing information to move out of the system, perhaps by printing a large number of files or displaying them on a terminal that can capture them) is detected by atypical usage of I/O resources.

3.3.5 Denial of service (making system resources unavailable to other users) is detected by atypical usage of system resources or use of special privileges to modify access rights to resources.

3.3.6 Malicious use (resource hogging, file deletion, and other miscellaneous attacks) may be detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

3.3.7 Although this is not a design goal of the current Haystack prototype, detecting some instances of creation and propagation of "virus" programs through audit trail analysis may be possible if certain conditions are met by the target operating system. Detecting modifications to executable files, a useful virus detection technique, could be done if all executable files are marked by attributes of their names, or if executability is a file access permission maintained by the target operating system, or if an exhaustive list of executable program files is available for comparison whenever files are modified. In the case of Haystack's target system, the Unisys 1100/60 operating under current versions of OS/1100, however, no evidence for either of the first two conditions is available in the audit trail. In addition, the number of files and file manipulations precludes use of the third technique in the current Haystack prototype. Obviously such techniques would not detect viruses targeted at interpretive programs, like command intepreters, that themselves process non-executable files.

3.4 Event Horizon

Analyzing the audit trail of a computer system could be likened to putting a day (or week or month) of the life of the target mainframe computer into the analysis platform - all at once! Since Haystack's analysis platform is a much smaller machine than the target mainframe, with significantly smaller memory and storage resources, we need to consider how much of the target's audit trail can be dealt with at one time.

We define the event horizon as the number of audited events the audit trail analysis system must "remember" in full detail at one time while processing a series of events recorded in the audit trail.

In the simplest case, where the event horizon is one, each individual event is analyzed without reference to specific details of preceding or succeeding events. All required information from an event is abstracted from the event data and stored as part of a data aggregate, like a statistical ensemble or the weights of the connections in a neural network. Analysis of the event is performed with respect to the recorded data aggregates assembled from the processing of previous events. After all events have been aggregated, the data aggregates themselves are analyzed. This aggregation process provides a significant reduction in the volume of data and improves throughput.

To identify certain kinds of behavior, the intrusion detection system would require an event horizon greater than one. For example, each event could be analyzed in light of all the events in the user's session, where a session is defined as the set of user activities bounded by logging in and logging out. Or each event could be analyzed with respect to all the events across multiple sessions belonging to that user. The latter case might help prevent cases where a malevolent user would attempt to "train" statistically-based analytical techniques to permit broader variations in behavior over time, until some particular intrusion becomes "acceptable" to the system.

Unfortunately, there is no reasonably small number N which could serve as an upper bound on the number of events which would allow us to perform meaningful pattern-matching, since plausible patterns may require an analysis of events covering at least an entire session, and one session can contain a very

large number of events. Also, it is hard to find actual audited intrusions to use as patterns to be matched. For these reasons, the prototype version of Haystack assumes an event horizon of one.

3.5 Learning (Self-modifying) Behavior

Because Haystack maintains a statistically-based model of a user, its individual user models evolve over time to adjust to changes in the users' work styles and task requirements. To prevent this characteristic from becoming a security liability, various techniques allow the SSO to observe and evaluate the effects of these changes over time, as described in section 4.2.2.3 below.

4.0 PROJECT SUMMARY

The basic approach to intrusion detection in Haystack is similar to Dorothy Denning's model presented in "An Intrusion-Detection Model" [3]. We have modified her model to suit the data available on our target machine and the security requirements of our users.

4.1 Design Goals

Several design goals have shaped our work with Haystack:

a Improve the System Security Officer's intrusion detection capabilities in a time-sharing data processing environment;

b Provide adequate throughput to keep up with a high-volume mainframe computer;

c. Monitor time-sharing and batch processing users, not users of specialized transaction processing applications;

> Users of specialized transaction processing applications on the target system are controlled by application-level security systems. The available audit information for these applications is not adequate for effective intrusion detection.

d. Maximize the portability of our design;

> Although the current target system is the Unisys (Sperry) 1100/60, we consider future U. S. Air Force POSIX-based computer system acquisitions as our secondary target. This dictates compatibility with the emerging POSIX standard, ANSI C, and standard SQL for database functions.
>
> By standardizing on a Canonical Audit Trail (CAT) file as a single representation for audit trail events, we maximize the portability of our audit trail analysis system to other computer systems. The CAT format was designed to be compliant with NCSC guidance in "A Guide to Understanding Audit in Trusted Systems" (NCSC-TG-001) for audit requirements at the B1 level [4], and also reflects our analysis of the formats of several computer vendors' audit trails.

e. Design a "friendly", usable system that will not be resisted by its intended users, the SSO's;

f. Gain experience in audit trail interpretation and develop a conceptual base for future work;

> This will advance a critical tool for deterring insider and other organized threats. For example, the prototype version of Haystack will operate in a batch processing mode. Field experience may show that a real-time monitoring mode is preferable and feasible.

g. Use a currently available Air Force standard computing platform for this system. This platform is the Zenith Z-248, an Intel 80286-based "clone" of the IBM PC-AT, running MS-DOS.

4.2 System Organization

The Haystack system consists of two program clusters, one executing on the Unisys (Sperry) 1100/60 mainframe and the other executing on the Z-248 PC. At the highest level, Haystack is a system that interacts with three external entities: the Unisys (Sperry) 1100 operating system, the System Security Officer, and the database management system on the analysis platform. The conceptual structure of the Haystack system is represented in Figure 2, and the operational structure of the Haystack system is shown in Figure 3.

4.2.1 Unisys (Sperry) Haystack Programs

The Unisys (Sperry) programs extract the required audit trail records from the operating system's audit trail logs, parse them with respect to the abstract elements that constitute a generalized audit trail event, transform them into the required Canonical Audit Trail (CAT) file format, and write them to a standard 9 track ANSI tape. The SSO may select particular users and time intervals for analysis, if desired, or may process the entire set of audit trail information.

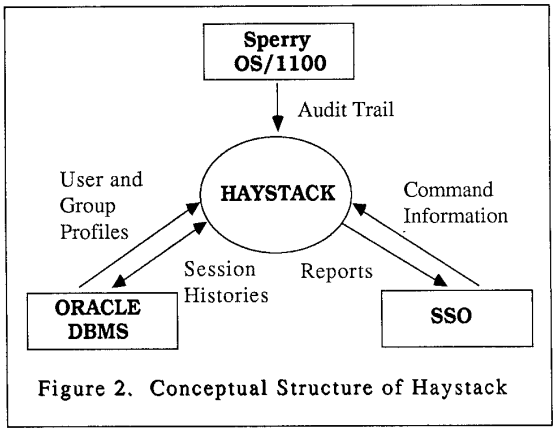

Figure 2. Conceptual Structure of Haystack

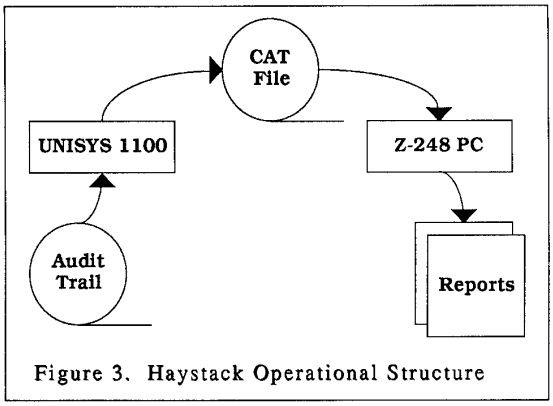

Figure 3. Haystack Operational Structure

4.2.2 PC Haystack Programs

The PC Haystack programs constitute most of the code of the system. There are four main functions implemented on the PC.

4.2.2.1 Processing the Canonical Audit Trail (CAT) File

The most I/O-intensive code on the PC is that which reads the CAT tapes from the mainframe, detects and logs any obvious anomalies, and creates new session records for the users whose activities were recorded on the tapes. The processing bandwidth of this stage of Haystack operation is inherently limited by the medium of transmission (1600 bits per inch on magnetic tape) and the hardware used to read the medium (less than 100 inches per second) to less than 160 kilobytes of data per second.

A session history record is created when a login event occurs for a user, based on the user's profile. That session record is cached for the duration of the processing of the tape to reduce ongoing access to the database on disk. The session history record is written to the database when a logout event for that user is detected.

Because the Air Force security policy for the 1100/60 defines what special privileges and capabilities may be assigned to specific classes of users, Haystack maintains a database of user groups as the basis for individual user profiles. If a user recorded in the CAT file has not previously been encountered by Haystack, a new user profile with minimal capabilities is created consistent with the principle of least privilege. This new user and the new user profile are flagged for review by the SSO in the summary report generated by this step of processing.

The user's session record is updated as audit events are recorded for each user. When security-relevant events are detected, their details are logged as "anomalous events" into a separate report for the SSO. Thus all security-relevant events are listed in detail for the SSO. Since they make up only a small percentage of the audit events, typically less than 0.5%, this does not burden the SSO.

Because of special characteristics of the 1100/60's audit trail mechanism, a user's session events (the login, logout, and intervening activities) are all guaranteed to be present in the same CAT file, unless the mainframe "crashes" catastrophically. If the PC detects an 1100/60 system "crash" event, all open user sessions are automatically closed.

When the tape has been processed, two reports are produced. The summary report has an overview of processing, a list of new users created automatically by Haystack, and a list of users whose session's "suspicion quotients" for intrusions exceeded a group- dependent threshold. The detailed report lists anomalous events recorded for each user. This provides a significant reduction of audit data compared to a "raw data" audit listing that could well be extremely voluminous.

4.2.2.2 Analysis of Current Sessions

After the CAT tape has been reduced to a new set of session history records in the database, Haystack analyzes the new sessions using statistical and pattern-based techniques, looking for evidence of predefined "bad" behavior and atypical or suspicious behavior. The pattern-based techniques assess multivariate characteristics of the sessions compared against expected characteristics of particular types of intrusions. Rule-based systems components could be added in future versions of Haystack to evaluate the overall impact of observations, including confidence levels.

4.2.2.3 Longitudinal Analysis of Sessions

Because Haystack retains past user sessions in its databases (deleting them under the SSO's control for archiving and maintenance purposes), it can help the SSO look for trends and tendencies in the user's behavior over time. If a malicious insider believed that a statistically-based monitoring technique was in place, he/she could try to "train" the system gradually over time to accept greater variations in behavior. For example, the insider might gradually increase the expected number of pages printed by the user until the range of "typical" print outputs for the user was large enough to allow him/her to dump an entire mainframe database. Some of this "training" can be detected as a statistical trend in user behavior over time.

The SSO may also apply a variety of "aging" techniques to the data to account for the effects of skill improvement in novice users, changes in job assignments, and frequency of use.

4.2.2.4 Utilities

Several SQL database applications help the SSO maintain the databases that control Haystack, and give the SSO the ability to tailor its function to his or her site. These include editors for user and group profiles, an editor for session histories, and an editor to manage the database containing the expected behaviors and relative weights for the session features analyzed for each group. Haystack also allows the SSO to make "ad hoc" queries against the databases. This will assist investigations by letting the SSO generate new reports and analyses.

4.3 Security in Haystack

The OS/1100 audit trail, the primary data source, is protected by strict system access controls while on the 1100/60. The 1100/60 preprocessing step and the resulting data files on tape are under the control of the SSO. The Haystack software running on the PC has password access control on the databases, and the PC itself is physically protected. Because the bulk of the processing and most of the data are on a physically separate computer, the degree of influence that an 1100/60 mainframe user can have on the intrusion detection process is restricted to the possible effects on "raw" input data. On the analysis platform, all SQL database transactions except the creation of new session history records are themselves audited to maintain a history of changes in Haystack and its databases.

4.4 Current Status

Development of Haystack was underway at the time this paper was written. The development team consisted of three software developers and one librarian/technical writer.

The analysis platform is an Intel 80286-based Zenith Z-248 with an ANSI standard 9 track tape drive, a large Winchester disk, floating point coprocessor hardware, and at least 4 megabytes of memory.

The software environment consists of the MS-DOS operating system, PC ORACLE, an SQL-based database management system, Microsoft C and Assembler, a commercially available windows and forms package, and CLIPS, an expert system shell available in portable C source code from NASA.

5.0 SUMMARY

Haystack is designed to be an operational utility for the System Security Officer to reduce enormous quantities of generally obscure audit trail data to short summaries of interpreted information for further investigation of potential computer intrusions. Because of the ambiguity of the data and the variety of possible interpretations for most user behaviors, the SSO remains the critical element in the investigative process. Haystack assists the SSO by providing "hunches," clues, and summaries of relevant data.

6.0 FUTURE RESEARCH

There are many areas for future research in the general area of computer intrusion detection systems.

- How do we test an intrusion detection system and measure its effectiveness? There are very few known recorded instances of system penetration; most scenarios used for testing have been generated by software developers based on their own understanding of system weaknesses. Testing and validation are required to "sell" the concept of intrusion detection to organizations, especially commercial ones.

- In what ways would alternate pattern-recognition techniques, such as rule-based (expert) systems, language-based approaches (like semantic analysis), and connectionist models (neural nets), improve our ability to detect event patterns?

- Would connectionist approaches to user modeling (e.g. neural nets) improve efficiency or effectiveness? If so, how would such systems be "trained" or programmed?

- What additional audit trail information, such as those expected from a B2-rated secure computing system, would facilitate the detection of virus attacks?

- Could we implement a real-time intrusion detection system with sufficient security and sufficient reliability to be entrusted with the ability to shut down an offending user or even the entire system?

- Would more complex statistical techniques for user modeling and data interpretation, including multidimensional and multivariate methods, improve the detection ability of the system? What effects would these techniques have on the rates of false positives and false negatives?

- What additional information could be extracted from the host system to improve our intrusion detection capability? Haystack's sole data source is the standard audit trail from the operating system.

- What visual metaphors are most effective for presenting computer security information to the SSO? Is there a security metaphor that is analogous to the spreadsheet for financial analysis?

- What are the relevant privacy and legal issues? It is clear that users must be notified of monitoring. But if monitoring is based solely on standard audit trail information from the operating system, is notification required? If notification is required, it must it be given at every login to the system (through an appropriately worded "banner" message), or is a single notice prior to first use sufficient? Also, if monitoring techniques with non-security applications (like keystroke capture and counting) are added to the audit trail analysis system, this may affect employee morale and reduce the perceived usefulness of the target computer system for exploratory or research work.

Acknowledgments:

I am grateful to Lieutenant Tim Grance, our Air Force technical contact, for his active support for our work, and Teresa Lunt of SRI International for helpful discussions. I especially thank Howard Herbert, Bill Steffan, Marshall Bruni, and the rest of the Haystack team at Tracor.

Bibliography

[1] National Computer Security Center, "DoD Trusted Computer System Evaluation Criteria," DoD 5200.28-STD, 1985.

[2] Paul Boedges, quoted in "Air Force Mounts Offensive Against Computer Crime," Government Computer News, July 8, 1988, p. 51.

[3] Dorothy E. Denning, "An Intrusion-Detection Model", Proceedings of the 1986 IEEE Symposium on Security and Privacy, April 1986, pp. 118-131.

[4] National Computer Security Center, "A Guide to Understanding Audit in Trusted Systems", 28 July 87, NCSC-TG-001.