AudES - an Expert System for Security Auditing

Gene Tsudik Rita Summers
IBM Los Angeles Scientific Center

Abstract

Computer security auditing constitutes an important part of any organization's security procedures. Because of the many inadequacies of currently used manual methods, thorough and timely auditing is often difficult to attain. Recent literature suggests that expert systems techniques can offer significant benefits when applied to security procedures such as risk analysis, security auditing and intrusion detection. This paper presents an example of a novel expert systems application, an Expert System for Security Auditing (AudES). Issues in development and use of the expert system that are unique to the application domain are discussed.

1 Introduction

The importance of effective computer security measures has become increasingly evident with the advent of recently publicized intrusion attempts and virus attacks. Any organization implementing computer security policies is faced with a wide range of potential threats. While some types of threats can be effectively countered using real-time methods [4, 5, 6], detection of others remains too time- or resource-intensive to address in real-time. Post facto security auditing is frequently used to detect anomalous events that fall out of scope of real-time security measures. A landmark study by Anderson [4] suggests that external intrusion attempts can be detected by auditing login records, while some internal intruders can be detected by analyzing resource access attempts.

Some recent literature indicates that AI techniques (expert systems methods, in particular) may have much to offer to computer security practitioners [1, 7]. The AudES expert system project described in this paper is an experiment in investigating potential expert systems applications in the area of computer security auditing. It is designed to automate manual security auditing procedures and to alleviate the burden on human auditors. AudES is interposed between a human auditor and Resource Access Control Facility (RACF)[3], a popular security mechanism for IBM mainframe systems.

2 Motivation

Currently, most RACF security auditing procedures are performed manually by scrutinizing system logs and identifying potential security violations. This is hardly surprising, since security auditing has always been considered a sensitive task which inherently requires human expertise. Although manual auditing methods are sometimes adequate, when the volume of the audit data is high and the data itself is diverse in nature, consistency and timeliness are frequently sacrificed. Therefore, effective security auditing is often beyond the limits of the available auditor time. Moreover, it sometimes demands too much from human memory and problem-solving abilities.

RACF provides a report generation facility, Report Writer, which is used to obtain listings of users' activity records. Since RACF records each potentially suspicious event and performs no filtering

of violation patterns, huge volumes of audit data can be generated. An average mainframe system produces on the order of several thousand records every day. As was observed at one site with a dozen systems, the entire auditing process (generating reports, printing them, auditing and following-up) consumed around eight hours per day for a team of two auditors.

Another problem plaguing manual auditing is the amount of paper generated. In addition to violation forms which are filled out (by hand) for each suspected violation, RACF reports are kept in printed form for extended periods of time to maintain audit trails. Altogether, the amount of paper generated can be enormous.

There is also a more subtle incentive for automation which is rooted in the social aspects of security auditing, namely, its relatively low prestige. In addition to being repetitive and rather unexciting, security auditing is frequently viewed as a low-level job function. For this reason, despite widespread awareness of the need for routine security audits, the job is frequently assigned to employees on a part-time or temporary basis. This results in a contradictory situation where a very sensitive task is being performed by insufficiently trained or motivated personnel. One of the consequences is a high turnover rate among security auditors which, in turn, results in a lack of experts.[1]

Existing automated tools, built with conventional automation methods, handle only part of the task. Since conventional programming does not afford flexibility, these tools tend to be difficult to transfer among sites and difficult to modify. It is due to the inadequacies of the existing approaches (both manual and automated) that security auditing has become a prime candidate for an expert system application.

3 Design Issues

In assessing the scope of applicability of an expert system, it is helpful to examine the intended application domain and identify the activities that can be potentially incorporated into the system. Since the primary goal of a human auditor is to identify security violations, security auditing can be viewed as being centered around the sequence of events that represent the violation life-cycle:

1) Occurrence 2) Detection 3) Follow-up 4) Archiving

Items 2 and 3 are of particular interest and are discussed below.

3.1 Detection

Detection of security violations entails isolating sequences of events that match certain patterns and identifying appropriate follow-up procedures. When performed manually, it can be an extremely time-consuming and tedious task, since violations tend to comprise only a small fraction of the audit data that a human auditor has to examine.

The biggest challenge in automating detection is the proper and robust representation of the violation patterns, i.e., sequences of events that constitute violations. Violation patterns can be determined by a number of variables, e.g.,

- VIOLATION TYPE - thresholds and follow-up actions tend to differ for different types of

[1] Even though, most organizations have personnel knowledgeable in auditing and security issues, when it comes to fine details, experts are scarce.

violations.

- USER TYPE - different classes of users are treated differently, e.g., violation criteria for external users can be harsher than for internal ones.
- LOCATION - location of the user at the time of perpetration can also have bearing upon auditor's decisions.
- DATE/TIME - time of the day (e.g., business hours, after hours) and date (workday, weekend, holiday, etc.) can be important in pinpointing suspicious events.
- THRESHOLD - each distinct combination of the above typically determines a threshold (maximum number of attempts) exceeding which triggers a violation.

Another issue of concern is the large volume of data in audit logs. At first glance, this seems to have little bearing upon the system design, but in reality, the size of audit logs is one of the most important characteristics that distinguish security auditing as an application domain.

Most available expert system packages target applications that employ *traditional* expert system methodology, i.e., a single consultation at a time. An example of a traditional system is a medical diagnostic system or a wine selection advisor. One common factor in all such systems is the two-step consultation process which begins with the expert system asking the user for some initial consultation data and then making a diagnosis, or a recommendation.

In contrast, the violation detection process consists of a large number of iterations, i.e.:

1. *While there are more records*
   a. *Acquire all records for a single user*
   b. *Analyze records for all violations types*
2. *Go to step 1*
3. *Compute report statistics*

The iterative nature of the auditing process which, as mentioned above, plagues human auditors, does not lend itself to most expert system tools. The main reason is the amount of internal bookkeeping necessary to support multiple instances. In particular, maintenance of long lists and records, complex memory management (efficient memory management is not a strong point of most expert system tools) due to iteration, are just a few factors contributing to the performance degradation. In order to combat this problem, AudES was made as "lean" as possible by implementing routine (not rule-based) tasks outside the knowledge base.

3.2 Follow-up

Follow-up is the sequence of events immediately following detection. It also requires timeliness and consistency. However, unlike detection, follow-up requires a more *active* approach. In other words, an auditor may have to:

- Contact Suspected Violator
- Contact Violator's Manager
- Notify Resource Owner(s)

- Alert Security, etc.

In summary, it involves much communication and paperwork on the part of the auditor - time that can be saved if the process is automated.

Although most organizations have guidelines for security auditing, they are rarely all-encompassing and often leave some aspects of detection to the individual auditor's judgement. At times, even when a sequence of events meets a specific violation criterion, the auditor has to decide whether or not an investigation is warranted. For this reason, it may beneficial not to automate follow-up procedures completely; instead, an advisory approach can be taken, so that suspicious events meeting violation criteria are displayed and/or recorded, but the final decision and the appropriate actions are left up to the human auditor.

3.3 Archiving and Verification

Verification of security audit procedures are frequently conducted by the management. The main purpose is to verify that prescribed auditing procedures are being performed and corporate security guidelines are being followed. This used to be a lengthy, manual process that involved a lot of paper-shuffling. RACF reports were reviewed at random and selected violations were traced to determine if appropriate actions had been taken. For that purpose, old RACF reports, individual violation reports, follow-up traces and numerous other paperwork had to be maintained.

With the detection process automated by the expert system, verification can be easily accomplished by validating the rules procedures implemented by the system. Accountability can be maintained by generating consultation traces that reflect the actions on the part of both the auditor and the expert system. Also, the amount of paperwork can be greatly reduced if instead of entire RACF reports, only suspected violation records detected by the expert system are archived.

4 Development Environment

Choosing the most appropriate tool for development is an issue of great importance in building an expert system. Ease of use, user interface, editing and run-time facilities, and speed of execution are just a few of many attributes that must be considered in identifying the right tool. Not surprisingly, an optimal trade-off among all desired properties is extremely difficult to find. Many expert system tools that target programmers emphasize speed while compromising user interface and run-time facilities, and others, that target the end-user, provide rich interfaces and easy syntax while sacrificing speed. In case of AudES, the latter was seen as the preferred (though not the optimal) approach. This choice may seem contrary to the goal of speeding up the auditing process, but the compensating factors (ease of use and modification, flexibility and extensive user interface) outweighed the performance consideration, at least for the initial implementation.

Expert System Environment (ESE)[2] was the tool adopted for AudES development. It is a commercially available product that has been successfully applied in a number of tasks including prediction, diagnosis, planning and decision making, in a variety of application domains.

ESE rules and commands employ a simple English-like syntax easily understood by non-programmers[2].

[2]See below for examples

Automatic error-checking and compilation is provided at edit-time. Furthermore, ESE provides an extensive interface to external procedures. This is especially important for AudES where some of the more routine tasks are implemented outside the knowledge base. An important run-time feature is the explanation facility, which allows the user to ask questions about the system's actions. Consultation traces are stored automatically so pre-recorded consultations can be easily reviewed and/or re-run.

However, as frequently happens in environments that facilitate AI methods, the emphasis on generality and ease-of-use greatly affects the performance. Due to ESE's performance limitations and the typical volume of audit data, many workarounds were necessary in order to meet the performance criteria. Whenever possible and appropriate, routine tasks not requiring rule-based techniques were implemented in PASCAL subroutines.

5 Knowledge Acquisition

As an inherently sensitive task, security auditing is relatively well-specified, unlike most other application domains. In most organizations, security auditing procedures are documented scrupulously. Furthermore, every attempt is made to cover all possible scenarios, so as to leave as little as possible to auditor's judgement. All this should make the process of knowledge acquisition rather uncomplicated. Experience with AudES, however, has shown that it is not always so.

For the most part, the guidelines and other documentation used by security auditors are written by non-programmers. This is neither surprising nor unusual. People responsible for writing this documentation are, in general, skilled security professionals well-qualified for this kind of work. On the other hand, written documents are not as conducive to covering all possible aspects of the task as is programming (even in a high-level expert system shell). This leads to a curious phenomenon whereby violation conditions previously ignored or missed are recognized as such at the knowledge acquisition stage.

An example of this phenomenon is the so-called *partial* login violation. A *mainstream* login violation is defined as a sequence of M or more failed login attempts for the same userid (M is also known as the login violation threshold). A typical sequence of failed login attempts consists of:

1. Three attempts to login. This constitutes a grace period.
2. Fourth failed attempt causes the userid to be revoked.
3. Subsequent login attempts are recorded but ignored, i.e., the user is not logged in even if the correct password is entered.

Human auditors are instructed to spot sequences of M or greater login attempts. If an auditor sees, say, M-3 attempts he ignores them. However, if these M-3 attempts begin with a revoke record, then, at least three attempts (grace period) must have occurred previously. Thus, even though M-3 attempts are recorded, M must have taken place. This (and other similar) situations occur when a sequence of records is split between two RACF reports (for example, when attempts are made on two consecutive days). Of course, events of this kind are not frequent and, thus, are not generally considered by those who write the security auditing guidelines.

Another discovery made during knowledge acquisition stage is the *multi-system* violation. Many users at the experimental site have accounts on several systems under the site's administrative control.

As described in Section 2, RACF reports are produced and audited individually for each system. Because no correlation between systems' activity is performed, potential violators can slip through by distributing their attempts among several systems. While this type of violation is very difficult to identify with manual methods, an expert system makes the task feasible.

6 Current Version

In its present state, the AudES system incorporates all of the functions pertaining to violation detection. All RACF auditing rules and procedures have been fully represented in the knowledge base.

There are twenty five events monitored and recorded by RACF. They range from login records to modifications to RACF itself. In addition, each event type has several (three to eight) possible return codes or qualifiers. Each $[event, qualifier]$ combination forms as a distinct exception type. There are a total of eighty nine exception types which AudES groups into four categories:

- Login - failed login attempts, e.g., invalid password supplied.
- Resource - unauthorized attempts to manipulate resources (e.g., access, rename, delete).
- Command - attempts to execute restricted/privileged commands, e.g., various RACF commands.
- Mixed - occurs when none of the above violations are detected, but the combined probability of all violation types exceeds a specified threshold.

RACF divides users into several classes: unprivileged, special, operations and auditor. Unprivileged users constitute the majority. Special users are the ones with the highest security clearance, such as the Site Security Administrators and they are given a virtual *carte blanche*. Operations are typically the systems programmers who are given some limited authority to manipulate RACF. Finally, auditors are the users permitted to access activity reports and use the RACF Report Writer.

AudES supports all RACF-supported user classes. Furthermore, it subdivides unprivileged users into internal and external (or outside) users. The internal users are (in our case) IBM employees, whereas external users are customers, vendors, contractors and various other non-IBM system users.

AudES treats each combination of user class, exception type as a distinct violation type. Each violation type has a corresponding (configured) threshold, i.e., a minimum number of records that cause AudES to suspect a violation. In addition, each violation type is associates with a prescribed sequence of follow-up actions which are presented to the auditor upon violation discovery.

Figure 1 shows an example ESE rule that AudES uses to identify suspected violations. The particular example is concerned with identifying anomalous resource access attempts.

6.1 Consultation

A typical AudES consultation proceeds as follows. First, the auditor runs a filter program on all raw RACF reports to produce AudES-readable input records. Next, ESE is used to load AudES. The auditor can pre-select a number of reports to audit by entering their names in a profile, or select them interactively at run-time. The consultation can be conducted either in the interactive or the unattended mode.

IF *(resource-violation-counter > resource-threshold)*
AND *(user-type* **IS** *CUSTOMER)*
AND *(resource-type* **IS** *INTERNAL)* **THEN**
recommended-action = "Contact the resource owner and
report incident to Customer Assistance"

Figure 1: AudES Rule Example.

In the interactive mode, AudES displays suspected violations as they are discovered and waits for the auditor to pronounce judgement (i.e., initiate or ignore). All suspected violations are tagged as *processed* when in this mode and auditor's decisions are duly recorded. In unattended (or batch) mode the auditor runs a consultation in the background without requiring any input from the auditor. In this mode, suspected violations are recorded but tagged as *suspended* . This is particularly convenient when auditing must be done in odd hours of the day or available auditor time is limited. Regardless of the mode, all suspected violations are recorded to disk to serve as the proof of audit. In the interactive mode, auditor decisions are similarly recorded.

Suspected violations are recorded and displayed (interactive mode only) along with all the necessary user data as well as the information on commands, resources, affiliations and, most importantly, appropriate actions that an auditor must take according to the local security auditing guidelines. When the consultation is completed, the auditor can obtain a hard-copy of the recorded violations. In addition to consultation reports, the entire consultation (at the keystroke level) is recorded in a separate file which can be subsequently used to re-play the entire consultation anew. This feature can be used for verifying past audits.

6.2 Portability

AudES can be easily adjusted to individual site's requirements. The configuration is done by creating several consultation profiles for AudES execution. These profiles are created by the Site Security Administrator (usually, the user with RACF *special* attribute) with the help of a separate knowledge base, AudINIT. AudINIT takes the user step-by-step through the configuration process and checks the configuration parameters for possible anomalies and inconsistencies. The profiles and their contents are described below.

The *site profile* contains site-specific information such as:

- systems under the site's control
- users authorized to use AudES (auditors)
- privileged users (special and operations)
- frequency of RACF report generation

The *event selection* profile determines how AudES detects suspected violations. Each violation type can either turned on or off (monitored or ignored) and associated with a threshold. Follow-up actions

for each violation type are also entered in this profile. Both the *site* and *event* selection profiles are usually modified infrequently. Finally, the run-time profile sets various AudES execution parameters. It specifies (among other things) the execution mode (interactive or unattended), list of reports to audit, and whether or not to print the suspected violations. This profile can be modified as often as needed by the auditors to afford more flexibility.

6.3 Size

AudES is a relatively small system. Its two major components are the ESE knowledge base and the system interface. The rules and procedures are embodied within the ESE knowledge base. There are on the order of eighty rules and fifteen Focus Control Blocks (FCBs) which are used to group rules, parameters and external actions. The system interface consists of ten PASCAL subroutines used for external actions, e.g., fetching user records from RACF reports, identifying the user, composing violation records, etc. All of the interface services are directly accessible from the knowledge base. In addition, there are two stand-alone utility programs: AudSTRIP and AudINIT. The former is used to re-format (strip) the raw RACF reports before running AudES, and the latter is an ESE knowledge base which is used to customize AudES.

6.4 Deployment

Before replacing manual auditing procedures, AudES had to be thoroughly field-tested to assure its correctness and to validate its rules. For a period of eight months it was used in parallel with the manual auditing. This interim period proved to be extremely valuable due largely to the auditors' feedback regarding the user interface, performance aspects and violation reporting. The reactions to AudES on the part of the auditors were extremely favorable. Ease-of-use, savings in time and paperwork, and the ubiquity of unattended/interactive consultation modes were the features most appreciated by the users.

At the time of this writing, AudES is in full deployment at one experimental site, having completely replaced the previous auditing procedures. It is being used on a daily basis for analysis of RACF reports generated on some 20 to 25 mainframes. The initial readjustment period turned out to be very short (less than a week). The auditors find AudES very easy to use and customize. In summary, the task that used to consume an entire day for a team of two and, sometimes three, auditors, now requires about 1.5 hours per day for a single auditor. Of this time only one third is spent in preparing input for, and running, AudES. The balance is taken by printing and filing of AudES-generated violation reports.

At the same time AudES is being field-tested at two other sites with radically different user populations and business needs. Both sites are quite large (about 80 mainframe systems at each). This presents a number of new challenges to AudES of which performance consideration are the most immediate.

In conclusion, the benefits from the use of AudES as opposed to manual RACF security auditing procedures are as follows:

- Greatly reduced routine work for auditors
- Greater consistency of audit

- Timelier violation detection and follow-up
- Ability to conduct more frequent audits
- Cost savings from reduced paperwork
- Clear verification of auditing rules and procedures used
- Easy customization to meet individual site requirements
- Reduced training time for auditors and a tool for training

The last three items also represent the benefits of an expert system approach as opposed to more conventional non-AI techniques.

7 Summary

In summary, AudES demonstrates the feasibility and the benefits of applying expert systems methodology to computer security auditing. As the system evolves, it can acquire more knowledge and gradually take over some additional, more judgmental, tasks. The working system is a start towards a longer-range goal of expert systems for security audit and administration. It is but one component in provision for maximizing security, and, its integrity is dependent upon the soundness and completeness of the auditing procedures that it implements. At the same time, AudES serves as an example of application of expert system technology in a field hereto largely untouched. It demonstrates the feasibility and scope of potential automation of the security auditing procedures and exhibits underlying issues, limitations and concerns.

8 Acknowledgements

The authors would like to thank Stan Kurzban, Ray Martin and the anonymous reviewers for their insightful suggestions and comments on the draft of this paper.

References

[1] R. C. Summers and S. A. Kurzban, *Potential Application of Knowledge-based Methods to Computer Security*, *Computers & Security Journal, vol. 7*, August 1988.

[2] IBM Corporation, *Expert System Environment*, *General Information Manual, GH20-9597.*

[3] IBM Corporation, *Resource Access Control Facility*, *General Information Manual, GC28-0722-10.*

[4] J. P. Anderson, *Computer Security Threat Monitoring and Surveillance*, *James P. Anderson Co.*, Fort Washington, Pa., April 1980.

[5] D. E. Denning, *An Intrusion-Detection Model*, *IEEE Transactions on Software Engineering*, February 1987.

[6] T. F. Lunt and R. Jagannathan, *A Prototype Real-Time Intrusion Detection Expert System*, *Proceedings of 1988 Symposium on Security and Privacy*, April 1988.

[7] T. F. Lunt, *Automated Audit Trail Analysis and Intrusion Detection: a Survey*, *Proceedings of 11th National Computer Security Conference*, October 1988.