Adaptive Real-time Anomaly Detection Using Inductively Generated Sequential Patterns

Henry S. Teng Kaihu Chen
290 Donald-Lynch Blvd.
Applied Intelligent Systems Group
Digital Equipment Corporation
Marlboro, MA 01752

Stephen C-Y Lu
Knowledge-Based Engineering Systems Research Laboratory
Department of Mechanical and Industrial Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801

ABSTRACT

This paper describes a time-based inductive learning approach to the problem of real-time anomaly detection. This approach uses *sequential* rules that characterize a user's behavior over time. A rulebase is used to store patterns of user activities, and anomalies are reported whenever a user's activity deviates significantly from those specified in the rules. The rules in the rulebase characterize either the sequential relationships between security audit records, or the temporal properties of the records. The rules are created in two ways: they are either dynamically generated and modified by a time-based inductive engine in order to adapt to changes in a user's behavior, or they are specified by the security management to implement a site security policy. This approach allows the correlation between adjacent security events to be exploited for the purpose of greater sensitivity in anomaly detection against seemingly intractable (or "erratic") activities using statistical approaches. Real-time detection of anomaly activities is possible.

1 Introduction

The business of many corporations is becoming increasingly dependent on secure data and secure information. There are multiple aspects in the proper protection of proprietary information stored on a computer system. An important one is to detect security vulnerabilities[1] on a system, another one is to detect security intrusions on a system. Both are time-critical. The sooner a security weakness is discovered, the less opportunity an intruder has to explore the vulnerability. The faster a system detects and responds to a hostile attack, the faster severe damage can be minimized.

Audit trail analysis refers to the analysis of the computer activity audit records. The capability of detecting and responding to hostile attacks via real-time audit trail analysis has become a critical business issue for security management and a challenging technical issue to the security community.

There are three security threats[2] that could be addressed by audit trail analysis:

- External penetrators (who are not authorized to use the computer).
- Internal penetrators (who are authorized to use the computer but not the data, program, or resource accessed, including:
  - Masqueraders (who operate under another user's ID and passwords)
  - Clandestine users (who evade auditing and access controls)
- Misfeasors (who are authorized to use the computer and resources accessed but misuse their privileges).

In the past few years, automated audit trail analysis techniques and intrusion-detection systems[3] have been developed to address these threats using various approaches. Systems such as IDES[4], MIDAS[5], and W&S[6], use either statistical techniques, or expert system techniques or both, to analyze security audit trails.

This paper describes a *time-based inductive learning* approach to adaptive real-time anomaly detection which is a critical step towards successful intrusion detection. This approach has the potential of detecting masqueraders or misfeasors based on

CH2884-5/90/0000/0278$01.00 © 1990 IEEE

deviations from the known sequential patterns of a user.

One of the major challenges to an audit trail analysis technique is its ability to deal with those "erratic" users with a wide range of activities. The time-based inductive learning approach shows promising results in handling those "erratic" users that have strong sequential patterns. Another strength of this approach is its ability to focus on a few security events that may be relevant to a hostile attack, instead of dealing with an entire user login session that has been labeled suspicious.

2 Background

Inductive learning is a branch in Artificial Intelligence where the goal is to endow a computer with the ability to learn from experience. The approach described in this paper uses a time-based inductive learning program, called TIM, that discovers temporal patterns in the form of sequential decision rules generalized from input data.

2.1 Introduction to Time-based Inductive Machine (TIM)

TIM (Time-based Inductive Machine) [7] was originally developed as a general-purpose tool with potential applications in many domains. TIM discovers temporal patterns from observations of a temporal process, where the patterns represent highly repetitive activities and can be used for prediction with high accuracy. The temporal patterns (represented in the form of rules) are generated and modified from the input data using a logical inference called *inductive generalization*. A set of *hypotheses* (i.e., temporal patterns represented in the form of *rules*) generalized from observed data is maintained and modified dynamically. Hypotheses are generated or modified so that eventually only the high quality ones are left in the rulebase. A *high quality* hypothesis has, among others, the following properties:

- High accuracy in prediction, i.e., when used for prediction the hypothesis is correct most of the time.
- High level of confidence, i.e. the hypothesis is confirmed by many previous observations.

The accuracy in prediction is expressed in the form of *entropy*[1], which measures the degree of randomness in its prediction when the rules are matched against the known data points. Although TIM always attempts to find better hypotheses using various entropy-based heuristics [7], the learning process is not hindered by the lack of perfect patterns in the data. The robustness of the system is enhanced by TIM's ability to produce temporal patterns that are only partially accurate when no better patterns can be generated.

Input data to TIM are called *episodes*. An episode consists of a sequence of *events*. Each *event* is a "snapshot" of a temporal process, which depicts the state of the process at a particular moment.

TIM attempts to generate rules that accurately predict the occurrence of a specific type (given by the user) of events. The function of TIM can be described as follows:

> Given a particular type of event E to be predicted, the goal is to discover from observations a set of temporally related conditions C that predicts within reasonable precision the time of occurrence of E relative to C.

During the learning process, the hypotheses generated by TIM are stored in the form of a *lattice*, where the generalization relationships between hypotheses are recorded. The lattice is used to guide the learning process in such a way so that high quality hypotheses can be systematically generated.

3 Applying TIM to Security Auditing

A session in a security audit trail may be viewed as an episode defined above, where:

[1] The entropy of a rule is defined as $\Sigma_i(-p_i \log (p_i))$, where p_i is the probability that event i will occur under the conditions as specified by the rule.

- Each event is one single entry in the audit trail, and is described in terms of a number of attributes.
- The events in the audit trail are considered to be sequentially related, i.e., expressions such as *next* event or *last* event are well-defined.

In security auditing, such an event could be a security record in the system audit trail or a command, such as a DCL command on a VAX/VMS[1] system, or a shell command on a UNIX[2] system, together with its corresponding command parameters and qualifiers/keys.

Assuming that sequential patterns do exist in the audit trail of a user, TIM can then be used to generate rules that predict the occurrences of certain specified events. Such rules are used to:

- Detect unusual activities that deviate from a user's established profile.
- Uncover activities that are unrecognized in the user's established profile.
- Offer a simplified view of the security audit trail where repetitive sequence of events are compressed into a few rules.

In addition, rules of the same form can be entered directly by security management or security intrusion experts to implement a site security policy in terms of proper or improper behavioral patterns. The behavioral patterns described in the site security policy can be specified in the form of either absolute time, the sequential relationships between specified events, or the combination of both.

TIM is implemented in Common Lisp, and is portable among different platforms that support the Common Lisp standard. For its application in the audit trail analysis, TIM is configured for continuous monitoring on an operating system. A profile database is incrementally modified to reflect the latest characteristic in a user's behavior. Auditing events are processed as soon as they are received. Anomaly activities can be detected and reported within seconds of receiving auditing events.

3.1 Representation of an Event

When applying the time-based inductive approach to security auditing, an event can be defined with a number of attributes. The following illustrates a few attributes that are typically used in an event description:

- TIME: the sequence number for a given event in a login session or the time stamp of the event.
- DESCRIPTION: a number of attributes associated with the event which could be as follows:
  - Event_type: type of event recorded such as file access or network breakin.
  - Image_name: the name of the executable image that has been executed.
  - Object_name: the object being accessed.
  - Object_type: the type of the object being accessed.
  - Privileges_used: the privileges used by the executable image.
  - Status: the status of the execution.
  - Process_ID: the process identification code.

  New attributes can be added without rebuilding the system.

A typical security event could look like this:

Event_type: Attempted file access
Event time: 6-JUL-1989 09:29:03.83
Process_ID: 00000055
User_name: BACKUP
Image_name: DISK:BACKUP.EXE
Object_name: DISK:AUTHORIZATION.DAT
Object_type: file
Access_requested: WRITE, CONTROL
Status: NORMAL, normal successful completion
Privileges_used: SYSTEM

Attribute hierarchies can also be specified to allow the system to infer information that is not explicitly given. The attribute hierarchy is used by TIM to generalize the security events into more abstract patterns that are not explicitly given in the audit

[1] VAX and VMS are trademarks of Digital Equipment Corporation.

[2] UNIX is a trademark of AT&T.

trail. The same mechanism is also used to allow the specification of a more abstract form of site security policy.

For example, given the following attribute hierarchy:

9 A.M. to 5 P.M. are business hours; and
EMACS is an editor; and
John is a member of the X project.

The event *"the command EMACS is invoked by John at 2 P.M."* can be generalized by the learning program to a more abstract description such as *"an editor is invoked during business hours by a member of the X project."* Such generalization is done by the learning program in order to create more predicative rules. On the other hand, a generalized form of a site security policy such as *"if an editor is invoked during business hours by a member of the X project, then invoke authentication procedure"* can also be specified.

3.2 User Profile Generation

Security event rules, which form the basis of a profile for each user or group of users, are generated via TIM. These rules describe the behavior patterns of either a user or a group of users based on past security audit history. Furthermore, each rule describes a *sequential pattern* that predicts the next possible events with satisfactory accuracy.

An example of a simplified rule produced in TIM is as follows:

E1 - E2 - E3 --> (E4 = 95%; E5 = 5%)

where E1, E2, E3, E4 and E5 are security events in the form described in previous section.

The rule indicates that if E1 is *followed by E2* (represented by the notation "-"), and E2 is followed by E3, then there is a 95% chance (based on the previous observations) that E4 will follow, and 5% chance that E5 will follow. Note that TIM can actually produce more generalized rules (in terms of temporal relationship between events, or the event descriptions) than the example given above. Temporal relationship such as

E1 - * --> (E2 = 100%)

where * matches any single event is supported. Any number of *'s in a rule is allowed. Temporal relationship between two events can also be specified as a range of values. For instance, a description such as "E1 and E2 are separated by *0, 1, or 2* events" is supported.

The following is a simplified example designed to demonstrate the process where rules are generated. Assuming that sequential rules are to be discovered from the following events where each letter represents an event:

A-B-C-S-T-S-T-A-B-C-A-B-C

The following rules may be generated:

R1: A - B --> (C, 100%)

R2: C --> (S, 50%; A 50%)

R3: S --> (T, 100%)

R4: T --> (A, 50%; S, 50%)

The rules can be viewed as alternative *explanations* that uncover what is happening in the process. R2 and R4 may be deleted from the rulebase eventually, because they may lead to wildly divergent predictions and are not "good" hypotheses. R1 and R3 may remain in the user profile, because they seemed to cover or explain more events more accurately, and could be used for deviation detection in the future.

3.3 Anomaly Detection

There are two ways by which an unusual activity can be detected. These are described in the following sections.

3.3.1 Deviation Detection

With the set of rules either produced by TIM, or entered by the security management, a deviation is detected if a sequence of events triggers the left-hand-side of a rule, R, while the subsequent events deviates significantly from the established long term pattern as predicted by R (i.e., the right-hand-side of R). For example, given the following rule, R,

A - B - -> (C = 100%)
(which indicates that if A is followed by B, then C is always expected to follow)

then the sequence of events

A - B - D

will be considered a violation of the pattern, because the first two events match the left-hand-side of R, while the third event, D, does not match the right-hand-side of the rule, C. Note that each event (given above simply as A, B, C and D) may be a complex description involving multiple attributes. For each rule, an *observed short term pattern* is matched against an *observed long term pattern* to decide whether a deviation has occurred.

3.3.2 Detection of Unrecognized Activities

On the other hand, the following sequence of events:

A - C - F

will not trigger the rule R above since the pattern at the left-hand-side, A followed by B, is not satisfied and will be considered as unrecognized activities. Such activities can be presented to security management for further examination. These activities could also be used to generate new rules, which could represent the inception of an evolving user profile.

3.4 Facilitating Security Management

When unusual sequence of events are detected, security management may take different courses of actions depending on the degree of deviation from a user's typical profile or the nature of the unrecognized activity. For instance, an attempt to modify the system authorization file may be considered as a serious offense.

Also the security event patterns generated are essentially a compression of all the security events that occurred in the past and could be used to facilitate security management . For example, a system manager may recognize a rule R1 as a *backup activity*, and another rule R2 as a *project development activity*. Thus instead of presenting the security manager with statistics that describe the activities of each individual user or group of users in terms of numbers, a more structured observation such as "there has been 50 backup activities and 37 project development activities" can be made. The consequence is that a security manager need only to deal with a small portion of the security audit information to understand what has happened to the system.

4 Preliminary Results

The approach described above has been implemented into a prototype system with four major modules: a data collection and conversion module, a user profile generation module based on TIM, an exception detection module, and a user interface module. The architecture of the system is shown in Figure 1.

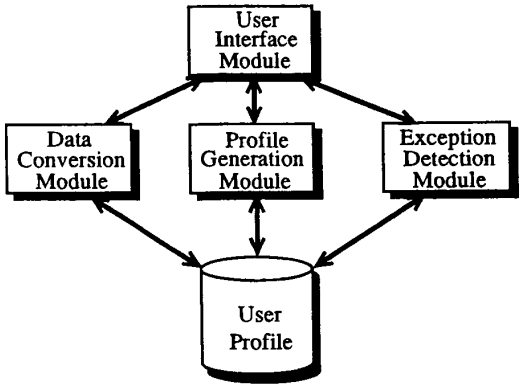

Figure 1. System architecture.

The program runs on a VAX 3500 computer with 32 megabytes of memory. Audit events were collected with users' consent. Proper warning messages were displayed when users logged onto the system. Anomaly activities can be detected within seconds of their occurrence. Since profile generation is very computation intensive, it is automatically delayed for later processing until sufficient computing resource is available.

The following is a report from the program where deviations from a user's profile were detected:

Account Name: FOO
Time of Analysis: 5-SEP-1989 13:31:00 - 14:31:00
Security Events Collected: 302
Rules in User Profile: 387
Rules Based on Site-specific Policy: 9
Unusual Sequences of Events: 6
Average Deviation for Unusual Sequences: 17%
Events Not Covered by Profile: 0 (0%)
Violations Based on Site-specific Policy: 0 (0%)

One of the rules in the user profile that had the deviation is as follows:

RULE: 155
The occurrence of the event:

```
        EVENT_TYPE =FILE- ACCESS
        USERNAME = FOO
        IMAGE_NAME = DISK:PREP.EXE
        OBJECT_NAME = DISK:PROFIL.LSP
        OBJECT_TYPE = FILE
        ACCESS_REQUESTED = READ
        STATUS = NORMAL
... implies that the event:
        EVENT_TYPE = FILE- ACCESS
        USERNAME = FOO
        IMAGE_NAME = DISK:MAIN.EXE
        OBJECT_NAME = DISK:UDP_IN.LSP
        OBJECT_TYPE = FILE
        ACCESS_REQUESTED = READ
        STATUS = NORMAL
... will occur 2 events later
... 100 percent of the time
... the total coverage is 15
... the entropy value is 0.0
```

In the rule above, the *total coverage* is the total number of times that the rule was triggered, and the *entropy value* indicates the consistency in the predictions made by the rule in the past.

The capability of focusing on a few unusual security events is important to a security officer. Security management may be able to better comprehend the relevance of these security events to a potential hostile attack, because the semantics and the ordering of the security events may provide stronger evidence.

An analysis of one experiment showed that one group of users activated an average of 46 different executable images, and accessed an average of 512 different files on a VAX/VMS system within a given period of time. Whereas an analysis of a second group of users showed that they activated an average of 16 different executable images, and accessed an average of 276 different files within a similar time period.

However, an analysis of the rules generated by TIM for the first group of users, who seemingly had a much wider range of activities than the second group of users, showed that the first group of users still had strong behavioral patterns in terms of sequence of activities. About 9.5% of the rules generated with an entropy value of less than 0.25 could explain or cover more than 63.5% of the security events over a given period of time. This indicated that the first group of users, although seemingly erratic, had a fairly consistent behavior describable in terms of sequential rules. Consequently the anomaly detection sensitivity of the system is increased.

5 Comparison with Statistical Approaches

The inductive approach described here is, to some extent, similar to statistical approaches used in systems such as IDES in that both can be used to offer a simplified view on a set of complex data. There are, however, some fundamental differences between the two approaches:

- The inductive approach conducts an heuristic search to find those hypotheses that satisfy certain given criteria, while statistical approaches are mainly used either for the evaluation of a given hypothesis, or for the systematic fitting of a given class of models.
- The inductive approach uses logical expressions as its representation, while statistical approaches use mainly analog models.

Depending on the level of audit trail such as auditing at the CPU and I/O level or auditing at the command level, one approach may have strength over the other. In particular, the inductive approach may generate more meaningful rules from auditing at the command level. The integration of the inductive and the statistical approaches has obvious advantages, and is a topic for future research.

6 Conclusions

The use of time-based inductive learning permits the sequential relationship embedded in a security audit trail to be uncovered. This in turn allows an audit trail to be viewed as chunks of temporally correlated events. This offers several important advantages:

- The use of rule-based sequential patterns offers a new perspective in the activities of a user. The detection of certain anomaly activities that may be difficult with other methods becomes possible.
- High quality sequential patterns are automatically generated using inductive generalization. Lower quality patterns are eventually eliminated from the system. With this feature, it is possible to build security auditing systems that are highly adaptive to changes in the problem domain. This

feature alleviates the problems of knowledge acquisition bottleneck and rulebase maintenance, which are typical issues encountered in the expert system approach.

- The overall sensitivity of the system is increased because the detections of violation are measured against a local context (the left-hand-side of a rule) provided by each *individual* rule. Even for rules that are rarely triggered, anomaly activities that deviate from such rules can be detected.
- In a limited sense, the meaning of each event (e.g., EDIT is a file-changing command) can be captured using an attribute hierarchy. The rules created in such a way are easy to understand, by the security management, therefore the chance of catching "cheaters" who intend to falsify their normal profiles during the learning period is improved.

Our preliminary experiments have shown that significant number of sequential patterns can be found in the activities of most users. Instantaneous response to deviations from established patterns is possible for these users.

Acknowledgments

We would like to thank Dr. Mitchell Tseng, Michael Gee, Tom Cerva, and Dr. Dorothy Denning at Digital Equipment Corporation for their continuing support and constructive comments.

References

[1] H. S. Teng, An Expert System Approach to Security Inspection of a VAX/VMS System in a Network Environment, *Proceedings of the 10th National Computer Security Conference, Baltimore, Sept. 1987.*

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

[3] T. F. Lunt, Automated Audit Trail Analysis and Intrusion Detection: A Survey, *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, October, 1988.

[4] D. E. Denning and P. G. Neumann, *Requirements and Model for IDES - a Real-Time Intrusion Detection System*, Computer Science Laboratory, SRI International, 1985.

[5] M. M. Sebring, Eric W. Shellhouse, R. Alan Whitehurst, Expert Systems in Intrusion Detection: A Case Study, *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, October 1988.

[6] H. S. Vaccaro and G. E. Liepins, Detection of Anomalous Computer Session Activity, *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, May 1989.

[7] K. Chen, *An Inductive Engine for the Acquisition of Temporal Knowledge*, PH.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1988.