24

A METHODOLOGY TO DETECT TEMPORAL REGULARITIES IN USER BEHAVIOR FOR ANOMALY DETECTION

Alexandr Seleznyov
Computer Science and Information Systems Department
University of Jyväskylä
P.O. Box35, FIN-40351, Jyväskylä, Finland
alexandr@jytko.jyu.fi

Abstract

Network security, and intrusion detection in particular, represents an area of increased interest in security community over last several years. However, the majority of work in this area has been concentrated upon implementation of misuse detection systems for intrusion patterns monitoring among network traffic. In anomaly detection the classification was mainly based on statistical or sequential analysis of data often neglecting temporal events' information as well as existing relations between them. In this paper we consider an anomaly detection problem as one of classification of user behavior in terms of incoming multiple discrete sequences. We present an approach that allows creating and maintaining user behavior profiles relying not only on sequential information but taking into account temporal features, such as events' lengths and possible relations between them. We define a user profile as a number of predefined classes of actions with accumulated temporal statistics for every class, and matrix of possible relations between classes.

Keywords: Network Security, Intrusion Detection, Anomaly Detection, Online Learning, User Profiling, User Recognition

1. INTRODUCTION

Our society is becoming increasingly dependent on the rapid access and management of information. More information is being stored and processed on network-based computers. Increased connectivity not only provides access to larger and varied resources of data more quickly than ever before, it also provides an access path to the data from virtually anywhere on the network (Power, 1995). Thus, there is a need to have means to protect computer systems against abuse.

There are intrusion prevention and detection techniques used to protect computer systems. The intrusion prevention techniques such as authentication and authorization, safe programming, and information protection serve as a first line of defense in computer systems. However, recently the amount of successful intrusion incidents has grown quite high: even 99% of all major companies have reported at least one major intrusion incident (Sundaram, 1998). Computer systems tend to be more and more complicated introducing new weak points that allows to exploit them in order to penetrate systems' defenses. Hardware or software failures, incorrect system administration increase intrusion's chances to be successful. Software bugs also represent a great danger, since software designers are not learning from past mistakes, still reproducing "classical" programming mistakes (such as buffer overflow in *sendmail* (Sendmail, 2000)). In many cases, the security controls themselves introduce weaknesses. Thus, it shows that the usage of intrusion prevention alone is not sufficient to reliably defend computer system and there is a strong need to have another line of defense, such as intrusion detection.

Intrusion detection is a security technology that attempts to reveal and isolate intrusions against computer systems; therefore it is an important component of security system. Intrusion detection systems (IDSs) use a number of generic methods for monitoring of vulnerabilities' exploitation. They are useful not only in detecting successful breaches of security, but also in monitoring attempts to breach security, which provides important information for timely countermeasures. Thus, IDSs are useful even when a computer system has a high degree of confidence (Kumar, 1995). The intrusion detection approaches may be roughly divided into two main categories: misuse and anomaly detection systems (Smaha, 1993).

Misuse intrusion detection systems, for example (Kumar and Spafford, 1995) and STAT (Ilgun and Kemmerer, 1995), detect intrusions that follow well-known patterns of attack (or signatures) that exploit known software vulnerabilities. These misuse intrusion detection systems include encoded knowledge about poor or unacceptable behavior and directly search for it (Smaha, 1993).

The intrusion detection systems of the second category (for example IDES (Lunt et al., 1992)) are detecting abnormal behavior or use of computer resources. They classify usual or acceptable behavior and report other irregular behavior as potentially intrusive. Techniques used in anomaly detection are varied. Some of them rely mainly on statistical approaches and result in systems that have been used and tested extensively. Techniques based on prediction of future patterns of behavior utilizing already gathered patterns is an examples of approaches tried in intrusion detection (Lane and Brodley, 1998).

In this paper we formulate an anomaly detection problem as one of user behavior classification in terms of incoming multiple discrete sequences. Although, here we focus on user-oriented anomaly detection. Monitoring multiple streams of discrete events, such as GUI events, system call traces, keystrokes, a system learns in order to classify (or recognize) user according to his behavior. By developing our approach we aim to eliminate, as much as possible, manual and ad hoc elements from the creation and manipulation of the user profiles by introducing online learning. We develop an approach that allows creating and maintaining users' behavior profiles relying not only on sequential event information but taking into account events' lengths and possible relations between them. Information about user "normal" behavior is accumulated in user profile. We define it as a number of predefined classes of actions with accumulated temporal statistics for every class, and matrix of possible relations between classes. Every class contains a number of instances, i.e. a number of patterns that are allowed for this class. In other words, an instance of a certain class contains temporal information that is peculiar for a certain pattern. A relation matrix describing possible relations between classes gives us possibility not only to check the "normality" of each action in incoming sequence of events, but also to check whether current relations between actions are "normal" for a certain user.

In this paper we develop an approach to the problem of anomaly detection. In particular, automatically matching encoded patterns against current event streams in order to find deviations from normal behavior; and than decide whether it intrusion or normal user behavior changes. Our approach is based on the assumption that the user's behavior includes regularities that can be detected and coded as a number of patterns. The information derived from these patterns could be used to detect the abnormal behavior and to learn the intrusion detection system. It is a hitherto untried approach in the field of anomaly detection.

A definition and description of temporal-probabilistic trees are given in Section 2. In Section 3 we present an approach aimed to differentiate normal behavior from abnormal by monitoring deviations between incoming events and stored in profile. Finally, in Section 4 conclusions to this work are given.

2. CLASS APPROACH TO USER PROFILE BUILDING

Traditionally, in anomaly detection, user profiles have been built by calculating statistics for different characteristics, such as consumed resources, command count, typing rate, command sequences, etc. (Lane and Brodley, 1998). In our approach we construct a user profile by defining *classes* or *cases,* which are used as a bricks in constructing a model of user behavior, and analyze information inside classes basing on its context. We present an incoming information as a set of temporal intervals that gives us possibility to apply Allen's algebra (Allen, 1983) to discover relations between temporal intervals and to store them for further classification. In the following subsections we describe definitions and basic concepts of our approach.

2.1. DEFINITIONS AND BASIC CONCEPTS

In this section we provide necessary definitions and describe basic concepts of our approach. Here we consider a problem of user profile building as a bringing to conformity events, provided by operating system log facilities, with notions used to build a user behavioral model. Thus, the system may possibly have several streams of discrete events such as GUI events, system call traces, network packets, and keystrokes. It needs to automatically build a profile for every user in order to recognize him in a future fitting current behavior into behavioral model described in his profile. During this process, the system should use as less as possible manual and ad hoc elements. In other words our aim is not only to develop an approach for behavioral model description, but also to automate all processes used for creation and manipulation of the user profiles as much as possible.

At the beginning we introduce *a layer structure* of events (Seleznyov and Puuronen, 1999a), which is a three levels used for describing incoming information on different abstraction levels.

The term *event* [1] has been widely used within the temporal database area giving it different meanings. We define the event *as a single indivisible occurrence on the time axis.* As can be seen from this definition that the type of a possible occurrence is not defined. Therefore, we may conclude that the event meaning may depend on source of the incoming information. In other words, applying the concept of event on different types of source information we are going to have different results. For example, for a GUI log an event may be represented by a GUI message, for network packets it may be a header of a single packet, etc. Applying our definition to the examples we can conclude that in these cases a single record in a log file represents an event.

In order to describe an information abstraction way we define a notion of layer that reflects different levels of information generalization. The higher layer the more general and more descriptive the notions describing the user behavior are. Thus, on the highest layer we describe the user behavior using most general way not depending on a source where information is coming from.

In this work we are using an underlying assumption that a person's interaction with a computer consists of different activities that he performs in order to achieve his goals. These activities consist of actions. Each action causes series of events in the operation system. Each user performs similar activities which are expressed by repeated sets of actions and which differ on a per-user basis. This gives the possibility to differentiate an intruder from a valid user (Seleznyov and Puuronen, 1999a).

Layer is a concept generalization level of relations between occurrences, each of which represents a single event on a different abstraction level (Seleznyov and Puuronen, 1999b). At the lowest instant layer all occurrences are represented as a time points (instants) on an underlying time axis. A single occurrence on this layer is called an event. It is equivalent to a single line in the audit trail. An event is described by a single instant relative to a particular user. Information on this layer is source-dependent (for example, it depends on operating system or logging facility used to collect it). Thus, same occurrence may be defined differently on this layer.

On the *interval* or *action* layer events with their simple relations are described and they form actions. The action is considered as a temporal interval, as for example: LOGIN-LOGOUT. A *relation* defines a temporal relation between two events as one of Allen's point temporal relations (Allen, 1983) and has a *Name*. The difference of any two-time points is likewise a rational number (Kautz and Ladkin, 1991). There are three basic relations that are used to represent relations of events: < - "less" relation, = - "equal" relation, and > - "greater" relation.

The most complicated level is the *activity layer,* which is represented by actions and relations between them. It describes them in a general source-independent way. Because the actions are extended in time, different actions may overlap in time and interact. One LOGIN-LOGOUT temporal interval, for instance, includes dozens of mail check intervals. A single occurrence on this level we call an *activity.* A *Relation* between two actions (temporal intervals) is defined as one of Allen's interval temporal relations (Allen, 1983).

These three layers are the way by which systems classify certain patterns of change. No one is more correct than other, although some may be more informative for certain circumstances. They are aimed to manage incoming information from multiple sources. For example, if system detects that a WWW browser is active and it exchanges information using HTTP protocol then it may conclude that user is browsing WWW pages in Internet. If the system observes network packets' headers it may come to the same conclusion when it detects connection establishment between user workstation and some server on port 80 followed by an information exchange. It may come to a same conclusion when observing some sequence of system messages between different modules of an operating system. Therefore, our point is - by monitoring different sources (sequences of events) it is possible to come to the same conclusions or, in other words, build a string of user activities, which are source and platform independent. And layer structure is aimed to make this transformation from event to activity layer, making possible to combine multiple strings from different sources. Moreover, wide usage of I/O cashing introduced some uncertainty in determining exact time points for a certain events. System *read* and *write* operations may be delayed and appear later in system logs than the user actually performed them. Getting confirmations from different sources the system is able to reason about actual time the user has requested a certain operation[2].

2.2. USER PROFILE

Above we have described a possible structure of information that may be gathered from operation system. This structure is used for abstracting information obtained from audit log files to more general - source or platform independent, giving it more meaning in context of person-computer interaction. How to store and process the obtained information? Below we present a way to construct user profiles using information obtained from different sources.

In contrast to definitions related to the structure of information layers, where all notions were defined in a way that more general were given in terms of more specific. For a user profile description we are going to move in a reverse direction. Hence, the structure of user profile definitions resembles an inheritance mechanism in object-oriented programming languages. Thus, we go from general to more specific, where more specific definition is formed by inheriting all features of "parent" one and adding some additional value to it.

As a general concept we define *action class* which describes one of the possible kind of action. It provides a formal description of an action without providing any specific details. Action class contains descriptions of events that start and end that action and possible events between them. Continuing our previous example consider a hypothetic action class “Web browsing”. It may be defined by Web browser activity interval. Also if we monitor network packets we may define same action class by a long sequence of events. For example, a request for connection establishment between some server and a user workstation port 80 (handshake protocol) followed by some information exchange and closing connection. As a matter of fact a number of all possible actions is limited by operation system tools and additionally installed programs, therefore, a number of action classes is finite and known beforehand.

An *action class instance* is an instance that describes a certain group of actions that belong to the same action class and have similar temporal characteristics. By similar temporal characteristics we imply temporal distances (time lengths) that characterize actions. These distances must be distributed normally in order to be grouped into a same instance. In figure 3.a it is possible to see an example of an action class that contains three instances. These instances described by three normal distributions each of which has own parameters - μ and σ. Finally, every instance contains n - number of actions grouped in it.

It is used for calculation of probability distribution of instances inside a single action class.

As was mention before an action class instance is formed by a number of *actions* with similar temporal parameters. Every action has information to which action class it belongs and it represents a concrete happening. Since an action has a beginning and an ending it is described by a temporal interval that has length (action's length).

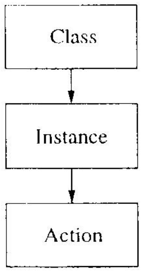

Figure 1 Structure of information stored in user profiles.

In this section we presented a way to represent information about user actions taking into account their temporal parameters. It describes user events by forming actions, classifying them, and splitting them into instances of the same action class. In figure 1 it is possible to see a structure that describes a part of a user profile presented in this section.

2.3. RELATIONS BETWEEN ACTION CLASSES

Looking at previous work we may summarize that traditionally, in anomaly detection, user profiles have been built basing on different characteristics, such as consumed resources, command count, typing rate, command sequences, etc. In these cases information analysis has been made using system log files, command traces, and audit trails. In most cases the classification was based on the sequence of events in time. However, the sequential data is not the only information that is possible to get from a stream of discrete happenings. It also contains some hidden information that is usually neglected: time relations between events are not taken into account at all or only very little attention is given to it. Sometimes time relations play a crucial role in attempting to classify events, i.e. determining whether an event is a part of anomalous or normal behavior. For instance, if an account has been under an IP spoof attack (Phrack, 1996), it is easier to recognize it relying on time relations between events, since the misuse activity appears as a continuation of the normal behavior within a single session. To be able to expose and later use of relations between actions it is necessary to introduce additional concepts.

Relation class is a notion that describes one of all possible relations between any two actions: $Action_i$ and $Action_j$. It is defined as one of Allen's interval temporal relations (Allen, 1983) and it has a *Name.*

$$Name \in \{\textit{before, after, meets, met - by, during, includes, overlaps, overlapped - by, starts, started - by, finishes, finished - by, equals}\} \quad (1)$$

According to (Allen, 1983):

- given any interval, there exists another interval related to it by each of the thirteen relationships;
- the relationships are mutually exclusive;
- the relations have a transitive behavior, e.g. if A is "before" B, and B "meets" C then A is "before" C.

Relation class instance describes some set of relations that have similar temporal characteristics and each of them belongs to the same relation class. In other words, a relation instance has a *Name* and describes some distribution of temporal parameters of relations grouped by this instance.

Relation is a relation (one of thirteen presented above) between two any actions $Action_i$ and $Action_j$. It is characterized by a *name* and a temporal distance between these actions. Thus, name is a qualitative parameter that describes what kind of relation it is, and temporal distance is a quantitative parameter that shows how strong the relation is or how much of it is possible to find between two current actions.

What is a temporal distance in context of user behavior expressed by a sequence of actions? Below we consider all possible relations between actions and define a notion of temporal distance for them. There are thirteen possible relationships between two actions (Allen, 1983). In our approach we are not using all of them. Since we have qualitative temporal characteristics we may define several *basic relations,* which with different temporal parameters produce all possible relations. For example, if $A1 : \textit{before}\ A2$ and temporal distance (distance between end of $A1$ and beginning of $A2$) is zero then $A1 : \textit{meets}\ A2$.

We define two relations as *basic:* "before", "during". Figure 2 shows them.

Below we define temporal distances t for basic relations.

$$\forall A1 : \textit{before}\ A2,\ t(A1 : \textit{before}\ A2) = A2_{\_begin} - A1_{\_end} \quad (2)$$

If $t(A1 : \textit{before}\ A2) = 0$ then we have case when $A1 : \textit{meets}\ A2$. If $t(A1 : \textit{before}\ A2) < 0$ in terms of Allen's temporal relations we may say that $A1 : \textit{overlaps}\ A2$.

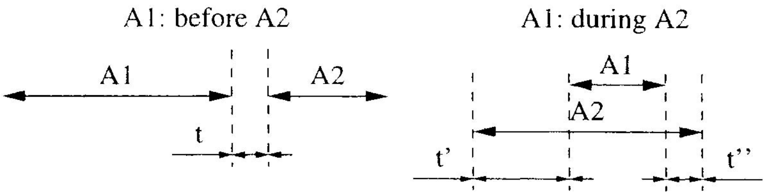

Figure 2 Basic relations.

$$\forall A1 : \textit{during} \;\; A2,\; t\,(A1 : \textit{during} \;\; A2) = A2_{\_begin} - A1_{\_begin} \qquad (3)$$

As it is possible to see from figure 2 a position of the interval $A1$ relatively to $A2$ is defined by t' and t''. If $t' = 0$ then this basic relation forms $A1$: *starts* $A2$ relation. In case when $t'' = 0$ we have $A1$: *finishes* $A2$. When both $t' = 0$ and $t'' = 0$ then $A1$: *equals* $A2$. For our purposes we do not need to calculate and store t''. Since we have a relation Name, t' and $A1$ length we are always able to reconstruct t'' when we need it. That is why we defined the temporal distance for a "during" relation as t'.

As we can see our two basic relations include all seven Allen's relations[3]. Therefore, in order to create a user profile we transform incoming information into vector of N classes with instances inside them. To take into account important relations between actions we need to discover them and store their temporal characteristics.

To represent relations between actions we use a square matrix $N \times N$ - *relational matrix.* In every cell $\{i, j\}$, matrix holds relational classes that are allowed between two classes i and j. Thus, cells i, j *when* $j > i$ contain direct relations for Ai and Aj, and cells i, j *when* $j < i$ contain reverse ones.

Using the relational matrix we may check whether there is a certain relation between any of two classes and if it fits into a certain relation instance.

3. DETECTING ABNORMAL BEHAVIOR

In this section we are going to discuss how to use described above action and relation classes to discover abnormal behavior by monitoring deviations between current user behavior and a model stored in profile. N action classes and a relational matrix are considered as tools that describes the model of user behavior. Every action class has one or more instances that represent some user action characterized by statistic parameters of time distribution - mean and standard deviation. Role of transactions' description between actions fills the relational matrix, every relation instance in which is characterized by same kind of temporal parameters. These tools are used to classify user behavior. Therefore, deviations from current values of its sequential and temporal parameters are considered as a consequence of abnormal behavior. To estimate the value of deviation we introduce *coefficient of reliability* - $r = \overline{[0, 1]}$. It is assigned to every active user and shows probability that the user is someone who he claims to be. During classification the system monitors deviations between expected or predicted user behavior and current one. According to this deviations it calculates some penalty (negative) or encouragement (positive) value, which reduces or increases the coefficient by Δ_r. At a root node the coefficient is assigned to 0.5 since there is no yet any evidence neither of distrust nor of trust. Therefore, the coefficient of reliability has an ability to grow as well as diminish. If it is crosses a certain threshold it means that there is a sequence of actions where parameters of each action are not in admissible intervals. It is considered as a case of abnormal behavior and thus, an alarm should be fired.

Coefficient Δ_r is calculated at the each step of classification. Each time the system gets a new case for classification it needs to determine correct action class and instance. Finding a correct class is relatively easy. Knowledge an action's name points to a certain class. After this it is necessary to find a correct instance inside the class. Below we present our way to automatically determine the instance, where the current action belongs, among several inside one class:

1 first it is necessary to determine a distribution where a new action belongs - $t \in [\mu - 2\sigma; \mu + 2\sigma]$;

2 if there are more then one interval found on step one, then we find *min* $|\mu - t|$ among them;

3 if there are more then one instance with same temporal distances between them and a current action, we chose the one with smallest standard deviation: *min* $[\sigma]$.

In normal distribution 95% of all cases are lying in interval $[\mu - 2\sigma; \mu + 2\sigma]$. During first step we use this to determine where a new case belongs. If there are some action instances that are overlapping each other and the new case is in overlapping area we use step two and three to introduce additional restrictions to find a right instance for the new case.

Below we consider calculations of the temporal-probabilistic characteristics for an action instance. In order to save computer resources a system does not need to keep a history of events, it needs only to have two parameters that describe a distribution inside this instance. Therefore, if a new case comes that supports a current instance we update its temporal parameters. For every *i* node its mean and standard deviation calculated each time it being used for classification:

$$\mu = \frac{\mu \times (n - 1) + t}{n} \tag{4}$$

where n - number of cases in this instance;
t - temporal length of being classified action.

$$\sigma = \sqrt{\frac{\sigma^2 \times (n-1) + (\mu - t)^2}{n^2}} \tag{5}$$

The same formulas may be applied for calculations of temporal parameters of relation instances. In this case t is a temporal length of being classified transition.

How we may use these temporal characteristics of action and relation instances to detect abnormal behavior? Well, if we take one instance it contains some distribution described by μ and σ. We separate all area outlined by this curve into three areas:

- Area limited by one σ - 68% of all area. In case when a new case is in this area we consider this case as a strong match that supports this instance. Therefore, the coefficient of reliability should be "encouraged".
- Area between one and two σ - 27% of all area. When a new case gets into this area it is a weak match. In other words, we have proofs that the new case belongs to this instance, but it does not support it as much as in previous case. Thus, the coefficient of reliability should not be changed.
- Area beyond two σ. In this situation the value of penalty for the coefficient of reliability should be calculated.

Below we present a formula for calculations of the coefficient of reliability changes:

$$\Delta_r = \begin{cases} 0 < t \leq \mu - 2\sigma, \nu_i \times exp(\frac{-(x-\mu)^2}{2\sigma^2}) - exp(-2) \\ \mu - 2\sigma < t \leq \mu - \sigma, 0 \\ \mu - \sigma < t < \mu + \sigma, \frac{n}{c} \times (exp(\frac{-(x-\mu)^2}{2\sigma^2}) - exp(-0.5)) \\ \mu + \sigma \leq t < \mu + 2\sigma, 0 \\ \mu + 2\sigma \leq t < \infty, \nu_i \times exp(\frac{-(x-\mu)^2}{2\sigma^2}) - exp(-2) \end{cases} \tag{6}$$

where c - coefficient that limits n and determines system's sensitivity to deviations;
v_i - coefficient of security significance of the action (in other words, how much danger improper usage of this action may cause); it is defined by system administrator for a certain kind of action.

Coefficient of reliability change calculations are based on following assumptions:

- how big difference is between predicted and current action lengths;

- how big difference is between predicted and current transition lengths;
- security significance of the current action;
- how big standard deviation a current action instance has;
- how big standard deviation a current relation transition has.

At the end of classification process if a coefficient of reliability r is lower than a certain threshold an alarm is fired. The alarm may also be issued during classification process when the coefficient of reliability diminishes very fast below the threshold. In the figure 3 we may see an example of some class "X" and a graphical representation of distribution of coefficient of reliability changes inside this class.

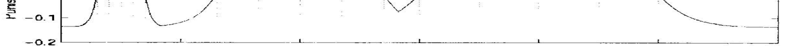

Figure 3 Class "X": a) three instances of the action class; b) dynamic Δ_r changes inside this action class.

In this section we presented an approach to expose regularities in user behavior. Using these regularities a user profile is built to monitor user's current behavior and detect anomalies in it. Basing on the amount of these anomalies it is possible to decide whether the current user is the same user he claims to be.

4. CONCLUSIONS

In this paper we proposed theoretical background for a new approach for anomaly detection. This approach allows to create a profile for every user by automatically finding regularities in his behavior and then constantly update these profiles. The main assumption behind this approach is that the behavior of each user follows regularities that may be discovered and presented using a limited number of action and relation classes.

The profiles are created by forming a vector of *N* action classes each of which contains several instances. In other words, it contains several temporal patterns of some action. Every profile also contains a matrix of relation classes (they are similar to action classes but describe relations). This matrix allows us to check whether a relation is valid between every two action classes.

Using described profiles a monitoring system evaluates every user action according to its length and relations with previous and next actions. During classifications a coefficient of reliability is changed. Basing on it a decision is made whether the current behavior is normal or anomalous.

The presented in this paper approach has some advantages. It is relatively simple and easy to visualize. It is quite fast since it does not require many calculations. The user profile is represented by action and relation classes, which are described by same temporal parameters, and thus, it is possible to use same formulas to calculate and update actions' temporal parameters as well as relations'.

At every time point every class contains several instances and therefore, may be described by some curve (as in figure 3a). To eliminate some calculations it is possible not to describe every instance as a distribution of temporal parameters, but a distribution of a coefficient of reliability change value (as in figure 3b). It would not require calculations of this value at every step and thus, increases classification speed.

The approach presented in this paper is interesting from theoretical point of view. However, we are still in the initial stages of our research and further development, numerous tests and evaluations with real implementation are needed to verify the practical aspects of this approach.

Notes

1. In this paper we attempt to use all definitions and terms in accordance with (?) and (?).
2. Since we are using a user-oriented approach we are interesting in time when a user has requested a certain action not when it actually has been performed by operation system.
3. Six reverse relations we do not take into account. Later we explain reasons for this.

References

Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM,* 26:832–843.

Ilgun, K. and Kemmerer, R. (1995). State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engeneering,* 21(3):181–199.

Kautz, H. and Ladkin, P. (1991). Integrating metric and qualitative temporal reasoning. In *Nine National Conference of Artificial Intelligence,* CA, USA.

Kumar, S. (1995). *Classification and Detection of Computer Intrusions.* Phd, Purdue University.

Kumar, S. and Spafford, E. (1995). A software architecture to support misuse intrusion detection. In *The 18th National information Security Conference,* pages 194–204.

Lane, T. and Brodley, C. (1998). Sequence matching and learning in anomaly detection for computer security.

Lunt, T., Tamaru, A., Gilham, F., Jagannathan, R., Neumann, P., Javitz, H., Valdes, A., and Garvey, T. (1992). A real-time intrusion detection expert system (ides) - final technical report. Technical, Computer Science Laboratory, SRI International.

Phrack (1996). Ip-spoofing demystified: Trust-relationship exploitation. *Phrack Magazine, available from http://www.fc.net/phrack/files/p48/,* 7(48).

Power, R. (1995). Current and future danger. Computer Security Institute, San Francisco, California.

Seleznyov, A. and Puuronen, S. (1999a). Anomaly intrusion detection systems: Handling temporal relations between events. In *2nd International Workshop on Recent Advances in Intrusion Detection,* Lafayette, Indiana, USA.

Seleznyov, A. and Puuronen, S. (1999b). Temporal aspects of user profiling in anomaly detection. In *Fourteen International Symposium on Computer and Information Sciences,* Izmir, Turkey.

Sendmail (2000). *Sendmail Mail Program.* Description and new version available from http:
www.sendmail.org.

Smaha, S. (1993). Tools for misuse detection. In *ISSA '93,* Crystal City, VA.

Sundaram, A. (1998). An introduction to intrusion detection. *ACM Crossroads.*