Proceedings of the 2001 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 5–6 June 2001

ADAM: Detecting Intrusions by Data Mining

Daniel Barbará Julia Couto Sushil Jajodia Leonard Popyack Ningning Wu
George Mason University
Center for Secure Information Systems and ISE Department
Fairfax, VA 22030
dbarbara,jics,jajodia,nwu@gmu.edu
popyack@rl.af.mil

Abstract— **Intrusion detection systems have traditionally been based on the characterization of an attack and the tracking of the activity on the system to see if it matches that characterization. Recently, new intrusion detection systems based on data mining are making their appearance in the field. This paper describes the design and experiences with the ADAM (Audit Data Analysis and Mining) system, which we use as a testbed to study how useful data mining techniques can be in intrusion detection.**

Keywords— **Intrusion Detection, Data Mining, Association Rules, Classifiers.**

I. Introduction

The widespread use of Internet and computer networks experienced in the past years has brought, with all its benefits, another kind of threat: that of people using illicit means to access, invade and attack computers. To understand that the threat is real it is enough to look at the statistics. Ten major government agencies, accounting for 98 % of the Federal budget had been compromised in the past [9]. Recently, a massive, coordinated attack directed at the major e-commerce sites was staged [22]. What is worse, it is estimated than less than 4 % of these attacks will ever be detected or reported. The issue is so pressing that has prompted the administration to propose a new Federal Intrusion Detection Network and a plan to put resources into what is called Defensive Information Warfare.

Lately, systems have been trying to use data mining techniques to meet the intrusion detection challenge. Data mining can be defined as a set of tasks that enable users to look for patterns in the data (good introductions to the topic can be found [8], [10]).

Intrusion detection techniques can be classified into two broad categories: misuse detection and anomaly detection. Misuse detection aims to detect well-known attacks as well as slight variations of them, by characterizing the rules that govern these attacks. Due to its nature, misuse detection has low false alarms but it is unable to detect any attacks that lie beyond its knowledge. Anomaly detection is designed to capture any deviations from the established profiles of the system normal behavior. Anomaly detection has the potential to generate too many false alarms, requiring a lot of time and labor to separate true intrusions from false alarms. ADAM has been designed and implemented as an anomaly detection system, but using a module that classifies the suspicious events into false alarms or real attacks. ADAM is unique in two ways. First, ADAM uses data mining to build a customizable profile of rules of normal behavior, and a classifier that sifts the suspicious activities, classifying them into real attacks (by name) and false alarms. Secondly, ADAM is designed to be used on-line (in real time), a characteristic achieved by using incremental mining algorithms that use a sliding window of time to find suspicious events.

L. Popyack is with the Air Force Research Lab, 2e Air Force Research Lab/IFGB 525 Brooks Rd Rome, NY 13441-4514, popyack@rl.af.mil .

This research has been funded by the Air Force Research Laboratory, Rome, NY under the contract F30602-00-2-0512.

II. ADAM

ADAM is essentially a testbed for using data mining techniques to detect intrusions. ADAM [4] uses a combination of association rules mining and classification to discover attacks in a TCPdump audit trail. First, ADAM builds a repository of "normal" frequent itemsets that hold during attack-free periods. It does so by mining data that is known to be free of attacks. Secondly, ADAM runs a sliding-window, on-line algorithm that finds frequent itemsets in the last D connections and compares them with those stored in the normal itemset repository, discarding those that are deemed normal. With the rest, ADAM uses a classifier which has been previously trained to classify the suspicious connections as a know type of attack, an unknown type or a false alarm.

Association rules are used to gather necessary knowledge about the nature of the audit data, on the assumption that discovering patterns within individual records in a trace can improve the classification task. The task of mining association rules, first presented in [1] consists in

ISBN 0-7803-9814-9/$10.00

deriving a set of rules in the form of $X \longrightarrow Y$ where X and Y are sets of attribute-values, with $X \bigcap Y = \emptyset$ and $\|Y\| = 1$. The set X is called the antecedent of the rule while the item Y is called consequent. For example, in a market-basket data of supermarket transactions, one may find that customers who buy *milk* also buy *honey* in the same transaction, generating the rule $milk \longrightarrow honey$. There are two parameters associated with a rule: *support* and *confidence*. The rule $X \longrightarrow Y$ has *support* s in the transaction set T if $s\%$ of transactions in T contain $X \cup Y$. The rule $X \longrightarrow Y$ has *confidence* c if $c\%$ of transactions in T that contain X also contain Y. The most difficult and dominating part of an association rules discovery algorithm is to find the itemsets $X \bigcup Y$, that have strong support. (Once an itemset is deemed to have strong support, it is an easy task to decide which item in the itemset can be the consequent by using the confidence threshold.)

ADAM uses connections as the basic granule, obtaining the connections from the raw packet data of the audit trail. This preprocessing results in a table with the following schema:

$R(T_s, Src.IP, Src.Port, Dst.IP, Dst.Port, FLAG)$.

In this schema, T_s represents the beginning time of a connection, $Src.IP$ and $Src.Port$ refer to source IP and port number respectively, while $Dst.IP$ and $Dst.Port$, represent the destination IP and port number. The attribute $FLAG$ describes the status of a TCP connection. The relation R contains the dataset that is subject of the association mining. The number of potential itemsets is large: connections may come from a large base of source IP addresses and ports. We focus in itemsets that contain items that indicate the source of the connection (like source IP and port), and items that indicate its destination (like destination IP and port). We also consider itemsets that are "aggregations" of source IP or Port values, e.g., connections that come from a source domain and have the same destination IP. We call these itemsets *domain-level* itemsets. (For instance, we want to discover frequent connections from Source IP X to Destination IP Y, or from Source Domain W to Destination IP Y.)

First, ADAM is trained using a data set in which the attacks and the attack-free periods are correctly labeled. In a first step, a database of frequent itemsets (those that have support above a certain threshold) for the attack-free portions of the data set is created. This serves as a profile against which frequent itemsets found later will be compared. The profile database is populated with frequent itemsets whose format was shown before, as well as frequent domain-level itemsets for attack-free portions of the data. The itemsets in this profile database can be cataloged according to the time of the day and day of the week, to further refine the specificity of these rules to variations of workload during the different time periods. The mining algorithm used for this first step of the training phase is an off-line algorithm. Thus, a conventional association rule mining algorithm can be used to drive this phase. (Although we use an algorithm tailored specifically for the kinds of itemsets we aim to find, and which runs considerably faster than a general-purpose association rules algorithm.)

Next, to complete the training phase, we use an incremental, on-line algorithm to detect itemsets that receive strong support within a period of time. This algorithm is driven by a sliding window of tunable size δ. The algorithm outputs itemsets (of the same format of those present in the profile database) that have received strong support during this window. We compare any itemset that starts receiving support with itemsets in the profile database for an analogous time and day of the week. If the itemset is present in the profile database, we do not pay attention to it (i.e., we do not devote storage resources to keep track of its support). On the other hand, if the itemset is not in the database, we keep a counter that will track the support that the itemset receives. If the itemset's support surpasses a threshold, that itemset is reported as suspicious. For a set of suspicious itemset, we provide two services. First the ability to drill down and find the raw data in the audit trail that gives rise to these rules. Secondly, We annotate suspicious itemsets with a vector of parameters (based on the raw audit trail data that gave rise to the rules). Since we know where the attacks are in the training set, the corresponding suspicious itemsets along with their feature vectors are used to train a classifier. The trained classifier will be able to, given a suspicious itemset and a vector of features, classify it as a known attack (and label it with the name of the attack), as an unknown attack (whose name is not known), or as a false alarm. It is important to remark here that ADAM has the ability of classifying a suspicious event (itemset and features) as an unknown attack. Notice that no training set can possibly prepare a classifier for an unknown attack (since there can be no examples of such an event). In general, labeling events as unknown attacks (or anomalies) is a very difficult problem. We are able to include such a provision by using an artifact present in some classifiers: the inclusion of a "default" label by which the classifier expresses its inability to recognize the class of the event as one of the known classes. We take the approach that any event flagged by the association rules software that cannot be classified as a known attack or as a normal event (false alarm) by the classifier, ought to be considered, conservatively, as an unknown attack. Using this assumption, we change the label in the classifier from "default" to "unknown." Our experiments have shown that this is a very efficient way to detect attacks whose nature is not fully understood. From there, an analyst can perform a more rigorous analysis by hand.

ADAM is then ready to detect intrusions online. Again, the on-line association rules mining algorithm is used to

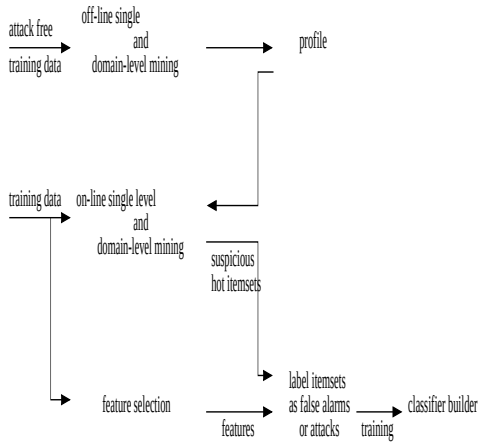

Figure 1: The training phase of ADAM.

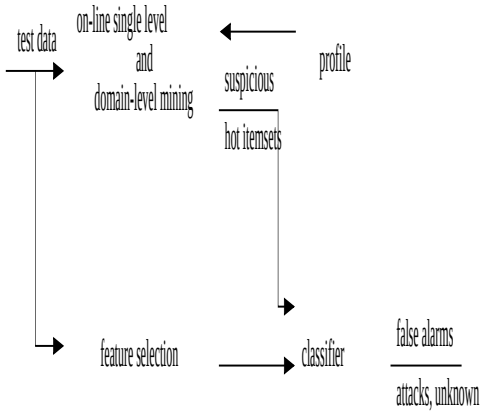

Figure 2: Discovering intrusions with ADAM.

process a window of the current connections. Suspicious connections are flagged and sent along with their feature vectors to the trained classifier, where they will be labeled accordingly.

Figures 1 and 2 show the basic architecture of our system. Our system performs its task in two phases. In the training mode , depicted in Figure 1, we use a data stream for which we know where the attacks (and their type) are located. The attack-free parts of the stream are fed into a module that performs off-line association rules discovery. The output of this module is a profile of rules that we call "normal," i.e., that depict the behavior during periods where there are no attacks. The profile along with the training data set is also fed into a module that uses a combination of a dynamic, on-line algorithm for association rules, whose output consists of frequent itemsets that characterize attacks to the system. These itemsets, along with a set of features extracted from the data stream by a features selection module are used as the training set for a classifier (decision tree). This whole phase takes place one time (off-line), before we use the system to detect intrusions.

The other phase, i.e., the actual detection of intrusions is implemented as depicted in 2. Here the dynamic algorithm is used to produce itemsets that are considered as suspicious and, along the features extracted by the features selection module are fed to the (already trained) classifier, which labels the events as attacks (including its presumed type), false alarms, or unknown. When the classifier labels connections as false alarms, it is filtering them out of the attacks set, avoiding passing these alarms to the security officer. The last class, i.e., unknown, is reserved for

max. number of false positives	1/day	10/day	100/day
total number of attacks detected	12	12	12
% of the total #	32.4%	32.4%	32.4%

Figure 3 PROBE attacks detected versus the maximum number of false positives

max. number of false positives	1/day	10/day	100/day
total number of DOS attacks detected	30	30	30
% of the total #	46.2%	46.2%	46.2%

Figure 4 DOS attacks detected versus the maximum number of false positives

events whose exact nature cannot be pinpointed by the classifier (they cannot be classified as known attacks). We consider those as attacks and include them in the set of alarms passed to the security officer.

Figures 3 and 4 show the results of DARPA 1999 test data. ADAM participated in DARPA 1999 intrusion detection evaluation. It focused on detecting DOS and PROBE attacks from tcpdump data and performed quite well.

Figures 5 and 6 show the results of the recent 1999 DARPA Intrusion Detection Evaluation, administered by MIT Lincoln Labs [19]. ADAM entered the competition and performed extremely well. We aimed to detect intrusions of the type Denial of Service (DOS) and Probe attacks (although ADAM can discover other types of intrusions as well). In those categories, as shown in Figure 5, ADAM ranked third, after EMERALD and the University of California Santa Barbara's STAT system. The attacks

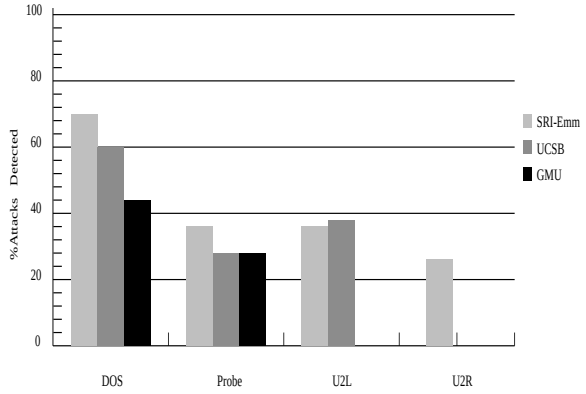

Figure 5: 1999 DARPA Intrusion Detection Evaluation Results. The graph shows the accuracy achieved by the best three ranking tools in the competition (EMERALD, UCSB STAT and ADAM –labeled as GMU–).

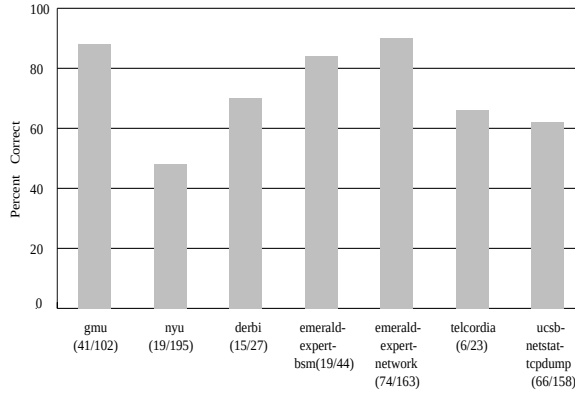

Figure 6: 1999 DARPA Intrusion Detection Evaluation Results. Overall ranking of the competing tools in terms of identifying attacks: EMERALD and ADAM –labeled as GMU– were the most accurate tools. (The numbers in parenthesis indicate the attacks the tools detected out of those they were supposed to detect.)

detected by those systems and missed by ADAM were those that fell below our thresholds, being attacks that involved usually only one connection and that can be best identified by signature-based systems. ADAM came very close to EMERALD in overall attack identification, as shown in Figure 6. Evaluation results can be found in [17].

III. Related Work

There are two kinds of intrusion detection systems: those which use "signatures" to detect attacks whose behavior is well understood and those which use some kind of statistical or data mining analysis to do the job. Of course, many tools have both kinds of engines present to maximize the likelihood of capturing the attacks.

A. Signature-based IDS

A.1 P-Best

P-Best [16] is a rule-based, forward-chaining expert system that has been applied to signature-based intrusion detection for many years. The main idea is to specify the characteristics of a malicious behavior and then monitor the stream of events generated by system activity, hoping to recognize one intrusion "signature". Using a traditional expert system is shown to yield good performance results in real-time detection. Also, it is easy to use and it integrates well into existing OS environments, thanks to its programmability at the C language level. In general, an expert system production rule consists of a predicate expression (rule antecedent) over a well-defined set of facts, and a consequent, which specifies which other facts are derived when the antecedent is true. When any facts are asserted that match the arguments of a rule antecedent, the predicate expression is evaluated. If it evaluates to true (the rule "fires"), then the consequent is executed, potentially resulting in other facts being asserted. This process may create a chain of rule firings that yield new deductions about the state of the system. In the context of intrusion detection, facts are generally system events, with a type such as "login attempt" and additional context attributes, e.g. "return-code" with value "bad-password". These attribute values can be used as arguments in the rules antecedents. P-BEST was developed at SRI International and it was first deployed in the MIDAS ID system at the National Computer Security Center. Later, P-BEST was chosen as the rule-based inference engine of NIDES, a successor to the IDES prototype [18]. The P-BEST expert system shell is also used in EMERALD's eXpert [21], a generic signature-analysis engine.

A.2 USTAT and NSTAT

USTAT [11], a real-time intrusion detection system for UNIX was developed in the Computer Science Department of the University of California, Santa Barbara, and the name stands for State Transition Analysis Tool for UNIX. The original design was first developed by P. A. Porras and presented in [20] as STAT, State Transition Analysis Tool. STAT employs rule-based analysis of the audit trails of multi-user computer systems. In STAT, an intrusion is identified as a sequence of state changes that lead the computer system from some initial state to a target compromised state. USTAT makes use of the audit trails that are collected by the C2 Basic Security Module of SunOS and it keeps track of only those critical actions that must occur for the successful completion of the penetration. This approach differs from other rule-based penetration identification tools that pattern match sequences of audit records.

NetSTAT [24] performs real-time network-based intrusion detection by extending the state transition analysis technique, first introduced in STAT [20], to the networked

environment. The system works on complex networks composed of several sub-networks. Using state transition diagrams to represent network attacks entails a number of advantages, including the ability to automatically determine the data to be collected to support intrusion analysis, resulting in a lightweight and scalable implementation of the network probes (speed advantages).

B. Statistic and Data Mining-Based IDS

B.1 IDES, NIDES and EMERALD

These three systems [18], [7], [2], [21] share a common background and are built with a traditional signature-based component that coexists with a statistical profiling unit. The statistical profiling unit has as its guiding principle finding behavior that looks anomalous with respect to a profile (in data mining this is known as finding "outliers").

The statistical unit of these IDS maintains a knowledge base of profiles, i.e., descriptions of normal behavior with respect to a set of selected measures. (Full details about the unit can be found in [12], [3].) The idea is to describe the audited activity as a vector of intrusion-detection variables and compare this vector to another one defined by the expected values stored in the profiles. If the audited activity vector proves to be sufficiently far from the expected behavior, an anomaly is flagged. This vector, or *summary test statistic* (in the terminology of IDES) is formed from many individual measures, such as CPU usage and file access. Each measure reflects the extent to which a particular type of behavior is similar to the historical profile built for it. The way that this is computed is by associating each measure to a corresponding random variable. The frequency distribution of is built (and updated) over time, as more audit records are analyzed. Examples of measures are the rate of audit record activity every 60 seconds and the user CPU time.

The frequency distribution is computed as an exponential weighted sum with a half-life of 30 days. The half-life value makes audit records that were gathered 30 days in the past to contribute with half as much weight as recent records; those gathered 60 days in the past contribute one-quarter as much weight, and so on. This is, in effect, a way to control the number of audit records that play a role in the distribution. The frequency distribution of Q_i can be computed in this manner for both continuous (numerical) and categorical measures. (For details see [12], [3].) The frequency distribution is kept in the form of a histogram with probabilities associated with each one of the possible ranges, or bins, that the measure can take. The cumulative frequency distribution is then built by using the ordered set of bin probabilities. Using this frequency distribution, and the value of the corresponding measure for the current audit record, it is possible to compute a value that reflects how far away from the "normal" value of the measure the current value is. The actual computation whose details can be found in [12], [3], renders a value that is correlated with how abnormal this measure is. Combining the values obtained for each measure, and taking into consideration the correlation between measures, the unit computes an index of how far the current audit record is from the normal state. Records beyond a threshold are flagged as possible intrusions.

B.2 JAM

The main idea in JAM [13], [14], [15] is to generate classifiers using a rule learning program on training data sets of system usage. The output from the classifier, a set of classification rules, is used to recognize anomalies and detect known intrusions. The main difference between JAM and ADAM is that JAM aims to be a missuse detection system by learning the characterization of the attacks (while ADAM uses an anomaly-based approach). Specifically, the approach of using classifiers is tested on two sets of data: one from attacks that use sendmail, the other from network attacks, using TCPdump. Sendmail data consists of two sets of traces, one with normal and one with abnormal data. The training data is fed to RIPPER [6], a rule-learning program. RIPPER rules classify the training data into the two classes "normal" and "abnormal". Each trace is then post-processed by comparing it with the RIPPER predictions, in order to filter out spurious prediction errors. The rationale for the post-processing scheme is that an actual intrusion is characterized by a majority of abnormal adjacent call sequences. A second experiment described in this work consists in computing classification rules using only the normal traces. In this case, in order to detect intrusions, the confidence information associated with the generated rules is used. Each trace is given a score according to whether a trace submitted to the classifier at runtime violates one of the generated rules. In this case, the trace score is incremented in proportion to the confidence of the violated rule. Given a long trace with many sequences, scores are assigned to each sequence, and the average score is used to decide whether the sequence represents an intrusion. The TCPdump experiment shows how classifiers can be induced from traffic data. Pre-processing is applied to the TCPdump raw data and then RIPPER is applied to the data. The paper reports on the results obtained, which are less encouraging than expected. Finally, the authors mention the use of a Meta-detection model that describes how multiple base classifiers can be combined in order to exploit combined evidence of multiple traffic patterns.

IV. Conclusions and Future Work

The behavior of ADAM in the DARPA 1999 competition gives us evidence about the usefulness of data mining techniques in intrusion detection. There are, however, a lot of issues to be solved:

- *Threshold tuning* It is important to obtain good thresholds for declaring a connection suspicious. We are currently performing statistical analysis of our results to obtain a better sense on how to adjust these thresholds.
- *Profiling* Profiles play a crucial role in ADAM, since they dictate what is considered "normal" behavior. We are currently experimenting with tailoring the profiles to different days of the week and times of the day, in an effort to get a good characterization of normality.
- Dependency on training data Unfortunately, for many installations, obtaining training data is not easy. It is difficult to distinguish (without a lot of manual labor) between normal connections and attacks in an audit trail to be capable of labelling the connections properly and use the data stream as training data. (So future audit trails can really be diagnosed automatically.) We are currently working in two directions to make this problem more manageable:

1. Decrease ADAM's dependency on attack training data: We want to be able to start running ADAM in an installation for which only "normal" behavior is known as training data. To do so, we are experimenting with a new technique, based in Pseudo-Bayes estimators [5], that will allow us to detect attacks even if no previous knowledge about the attacks exists. (Of course, we will not be able to label the attack name, but we will be able to attract the security's officer attention to it.) Preliminary results indicate that the technique is extremely succesful in identifying previously unknown attacks.

2. Automate the process of obtaining a profile of normal activity, starting from an audit trail for which the connections are not labelled. We are currently using statistical techniques to detect outliers in an unlabeled stream of connections. We are confident that by cleaning the data from outliers, the stream of connections that is left will give us a good profile of normal activity.

- *Detect other types of attacks.* So far, ADAM has concentrated in detecting PROBE and DOS attacks (and a handful of other attacks that make use of a relatively large number of connections). Our aim is to enhance the functionality of ADAM by using other data mining techniques for attacks that take place using a handful (or even one) of connections. An idea is to model normality of these attacks using time series and use a predictive tool that can distinguish when a series is abnormal, to flag it as an attack.

In summary, data mining is proving to play an important role in intrusion detection. We are very confident that ADAM will become, with time, a very strong tool to help security officers in their daily work against intruders.

References

[1] R. Agrawal, T. Imielinski, , and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, Washington D.C., May 1993.

[2] D. Anderson and T. Frivold and A. Valdes. NIDES: A Summary. In http://www.sdl.sri.com/nides/index5.html

[3] D. Anderson and T. Lunt and H. Javitz and A. Tamaru and A. Valdes. Detecting unusual program behavior using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES). Technical Report, SRI-CSL-95-06, Computer Science Laboratory, SRI International, May 1995.

[4] D. Barbará and S. Jajodia and N. Wu and B. Speegle. Mining Unexpected Rules in Network Audit Trails. Technical Report, George Mason University, ISE Dept. September 1999.

[5] Y.M.M. Bishop and S.E. Fienberg. Discrete Multivariate Analysis: Theory and Practice. The MIT Press, 1975.

[6] W.W. Cohen. Fast Effective Rule Induction. In *Proceedings of the 12th International Conference on Machine Learning,* Lake Taho, CA, 1995.

[7] D.E. Denning. An Intrusion Detection Model. In *IEEE Transactions on Software Engineering*, February 1997, pp. 222-228.

[8] U.M. Fayyad, G. Piatesky-Shapiro, P. Smyth, and R. Uthurusamy. Advances in Knowledge Discovery and Data Mining. AIII/MIT Press, 1996.

[9] General Accounting Office. Information Security: Computer Attacks at Department of Defense Pose Increasing Risks. GAO/AIMD-96-84, May, 1996.

[10] J. Han and M. Kamber Data Mining: Concepts and Techniques Morgan Kaufmann, 2000

[11] K. Ilgun. USTAT: A Real-Time Intrusion Detection System for UNIX. Master Thesis, University of California, Santa Barbara, November 1992.

[12] H.S. Javitz and A. Valdes, The SRI IDES Statistical Anomaly Detector. In http://www.sdl.sri.com/nides/index5.html

[13] W. Lee and S. Stolfo. Data Mining Approaches for Intrusion Detection. In *Proceedings of the 7th USENIX Security Symposium*, 1998.

[14] W. Lee and S.Stolfo and K. Mok. A Data Mining Framework for Building Intrusion Detection Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1999.

[15] W. Lee and S.J. Stolfo and K. Mok. Mining Audit Data to Build Intrusion Detection Models. In *Proceedings of the International Conference on Knowledge and Data Mining*, August 1998.

[16] U. Lindqvist, P.A. Porras. Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST). In *Proceedings of the 1999 IEEE Symposium on Security and Privacy.* pp. 146 -161.

[17] R. Lippmann, J.W. Haines, D.J. Fried, J. Korba and K. Das. The 1999 DARPA off-line intrusion detection evaluation. In *Computer Networks,* 34:579-595, 2000.

[18] T.F. Lunt and R Jagannathan. A Prototype Real-Time Intrusion-Detection Expert System. In *Proceedings of the IEEE Symposium on Security and Privacy,* 1988, pp. 18-21.

[19] MIT Lincoln Laboratories DARPA Intrusion Evaluation Detection. In http://www.ll.mit.edu/IST/ideval/

[20] P.A. Porras. STAT: A State Transition Analysis for Intrusion Detection. Master Thesis, Computer Science Department, University of California, Santa Barbara, 1992.

[21] P.A. Porras and P.G. Neumann EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proceedings of the National Information Systems Security Conference,* 1997, pp. 353-365.

[22] I. Sager et al. Cyber Crime. In *Business Week*, February 21, 2000.

[23] S. Smaha. Haystack audit trail analysis system. Status Report HS-STAT.TXT Haystack Laboratories, Colorado, Aug., 1990.

[24] G. Vigna and R. Kemmerer. NetStat: A Network-Based Intrusion Detection Approach. In *Proceedings of the 14th Annual Information Theory : 50 Years of Discovery Computer Security Application Conference*, Dec. 1998.

ISBN 0-7803-9814-9/$10.00