A FRAMEWORK FOR AN ADAPTIVE INTRUSION DETECTION SYSTEM WITH DATA MINING

Mahmood Hossain and Susan M. Bridges

Department of Computer Science
Mississippi State University, MS 39762, USA
E-mail: {mahmood, bridges}@cs.msstate.edu

ABSTRACT

The goal of a network-based *intrusion detection system* (IDS) is to identify patterns of known intrusions (misuse detection) or to differentiate anomalous network activity from normal network traffic (anomaly detection). Data mining methods have been used to build automatic intrusion detection systems based on anomaly detection. The goal is to characterize the normal system activities with a profile by applying mining algorithms to audit data so that abnormal intrusive activities can be detected by comparing the current activities with the profile. A major difficulty of any anomaly-based intrusion detection system is that patterns of normal behavior change over time and the system must be retrained. An IDS must be able to adapt to these changes, and be able to distinguish these changes in normal behavior from intrusive behavior. The goal of this paper is to provide a general framework for an adaptive anomaly detection module that utilizes fuzzy association rule mining.

1. INTRODUCTION

With the ever-increasing growth of computer networks and emergence of electronic commerce in recent years, computer security has become a priority. Since intrusions take advantage of vulnerabilities in computer systems or use socially engineered penetration techniques, intrusion detection is often used as another wall of protection in addtion to intrusion prevention techniques such as user authentication. Intrusion detection is not an easy task due to the vastness of the network activity data and the need to regularly update the IDS to cope with new, unknown attack methods or upgraded computing environments.

Mukherjee, Heberlein, and Levitt (1994) defined intrusion detection as identifying unauthorized use, misuse, and abuse of computer systems by both inside and outside intruders. There are many categories of network intrusions (Graham 1999). Examples include SMTP (SendMail) attacks, password guessing, IP spoofing, buffer overflow attacks, multiscan attacks, denial of service (DoS) such as ping-of-death, SYN flood, etc. Intrusion detection techniques can broadly be divided into two categories: *misuse detection* and *anomaly detection* (Sundaram 1996). Misuse detection is based on knowledge of system vulnerabilities and known attack patterns, while anomaly detection assumes that an intrusion will always reflect some deviation from normal patterns. Many AI techniques have been applied to both misuse detection and anomaly detection.

Pattern matching systems like rule-based expert systems, state transition analysis, and genetic algorithms are direct and efficient ways to implement misuse detection. On the other hand, inductive sequential patterns, artificial neural networks, statistical analysis and data mining methods have been used in anomaly detection.

The goal of mining association rules is to derive multi-feature (attribute) correlations from a database table (Agrawal, Imielinski, and Swami 1993). It has been observed that program executions and user activities exhibit frequent correlations among system features. Audit data can be formatted into a database table where each row is an audit record and each column is a field (system feature) of the audit records. Lee and Stolfo (1998) extended the basic association rule algorithms to capture consistent behavior in program execution and user activities The rules mined from audit data are merged and added into an aggregate rule set to form the user's normal profile. To analyze a user login session, frequent patterns are mined from the sequence of commands during the session and this new pattern set is compared with the established profile pattern set. Similarity functions are used to evaluate deviations to generate alarms in case of intrusive behaviors.

A major problem for such an IDS is that it can give false alarms in cases where there are modifications in the normal system behavior. The IDS must be capable of adapting to these changes and the user profile must be updated at regular intervals. One straightforward approach can be to generate a new user profile with each set of new audit data. This approach is not computationally feasible and can cause the system to incorporate patterns of intrusive behavior as normal. The paper discusses some of the issues encountered in developing an adaptive IDS using data mining techniques and outlines a general framework of an adaptive IDS.

The remainder of the paper is organized as follows. Section 2 provides a background of related work. Section 3 discusses some technical issues that need to be addressed to develop an adaptive intrusion detection system. Section 4 provides a rough archtitecture of an adaptive IDS. Finally, the paper ends with concluding remarks in Section 5.

2. BACKGROUND AND RELATED WORK

The goal of mining association rules is to derive correlations between the features of a database table. An association rule is an implication of the form $X \rightarrow Y_{[c,s]}$, where X and Y are disjoint itemsets, s is the support of $X \cup Y$ (indicating the percentage of total records that contain both X and Y), c is the confidence of the rule and is defined as $s_{X \cup Y}/s_X$ (Agrawal, Imielinski, and Swami 1993). It has been observed that program executions and user activities exhibit frequent correlations among system features. A typical example of an association rule obtained from audit data can be $ftp \rightarrow get_{[.4,.1]}$, which implies 40% of the time when the user uses the ftp command, get command is also invoked and doing so constitutes 10% of the commands issued by the user. Audit data can be formatted into a database table where each row is an audit record and each column is a field (system feature) of the audit records.

Lee and Stolfo (1998) utilized the basic association rules algorithms to mine rules from system audit data into an aggregate rule set to form the user's normal profile. Two rules are merged if their right and left hand sides are exactly the same or their RHSs can be combined and LHSs can also be combined, and the support and confidence values are close. For example, $service=http \rightarrow src\_bytes=20$ and $service=http \rightarrow src\_bytes=30$ can be combined into $service = http \rightarrow 20 \leq src\_bytes \leq 30$. Any subsequent system activities are analyzed to mine frequent patterns and the new pattern set is compared with the normal profile. Similarity functions are used to evaluate deviations involving missing or new rules, violation of the rules (same antecedent but different consequent), and significant changes in support of the rules. For example, if the new set has n patterns and m patterns among them can be merged with patterns in the profile set, then the similarity score can be m/n.

Audit data contains quantitative features. During mining, the quantitative data are partitioned into intervals. But a sharp boundary problem results from this partition that may create problems in intrusion detection. For example, let us assume $[a_1..a_p]$ and $[a_{p+1}..a_n]$ are two intervals for a quantitative attribute A, a_p has a support of 15%, a_{p+1} has a support of 5%, and the support threshold is 10%. Even if a_{p+1} lies near a high support value, it may not gain enough support. Now, if the interval $[a_1..a_p]$ is mined as normal pattern, the interval $[a_{p+1}..a_n]$ will be considered as abnormal. Similarly, an intrusive pattern with a small variance may fall inside $[a_{p+1}..a_n]$ and remain undetected. To overcome this boundary problem, Luo and Bridges (2000) developed an intrusion detection system that integrates fuzzy logic with data mining algorithms (association rules and frequent episodes). They categorize quantitative features into categories having fuzzy membership values. For example the feature *datasize* can be divided into three categories *low*, *med*, and *high*. If the feature is not fuzzy, then a particular value of *datasize* would fall into exactly one category and would have a membership of 1 for that category and 0 for all other categories. But if it is a fuzzy feature, then a particular value of *datasize* can fall into more than one category with some fuzzy membership values. The authors used a normalized measure to compute the fuzzy membership values. For example, a particular value of *datasize* can be "*low*" with 0.9 and "*med*" with 0.1.

Though the system works well generally, the selection of fuzzy membership function parameters is done by experience that may lead to some false alarms. Shi (2000) used genetic algorithms to automatically optimize the fuzzy-membership function parameters. In his approach, he defined a chromosome to consist of a sequence of the fuzzy function parameters. The process starts with a random initial population of chromosomes where each chromosome is a possible set of parameters. A fitness function is used that gives preferences to high similarity between rules mined from reference and non-intrusive data and to low similarity between rules mined from reference and intrusive data. The process evolves a population of chromosomes to come up with an optimized set of parameters. As a continuation of this work, Bridges and Vaughn (2000) proposed a prototype intelligent intrusion detection system (IIDS) utilizing fuzzy mining and genetic algorithms.

3. PROBLEM DESCRIPTION AND RELATED ISSUES

A major shortcoming of current IDSs that employ data mining methods is that they can give a series of false alarms in cases of a noticeable systems environment modification. There can be two types of false alarms in classifying system activities in case of any deviation from normal patterns: *false positives* and *false negatives*. False positive alarms are issued when normal behaviors are incorrectly identified as abnormal and false negative alarms are issued when abnormal behaviors are incorrectly identified as normal. Though it's important to keep both types of false alarm rates as low as possible, the false negative alarms should be the minimum to ensure the security of the system.

To overcome this limitation, an IDS must be capable of adapting to the changing conditions typical of an intrusiuon detection environment. For example, in an academic environment, the behavior patterns at the beginning of a semester may be different than the behavior patterns at the middle/end of the semester. If the system builds its profile based on the audit data gathered during the early days of the semester, then the system may give a series of false alarms at the later stages of the semester.

System security administrators can tune the IDS by adjusting the profile, but it may require frequent human intervention. Since normal system activities may change because of modifications to work practices, it is important that an IDS should have automatic adaptability to new conditions. Otherwise, an IDS may start to lose its edge. Such adaptability can be achieved by employing incremental mining techniques. Such an adaptive system should use real time data (log of audit records) to constantly update the profile.

One straightforward approach can be to regenerate the user profile with the new audit data. But this would not be a computationally feasible approach. When the current usage profile is compared with the initial profile, there can be different types of deviation as mentioned in section 2. Each of these deviations can represent an intrusion or a change in behavior. In case of a change in system behaviors, the base profile must be updated with the corresponding change so that it does not give any false positives alarms in future. This means that the system needs a mechanism for deciding whether to make a change or reject it. If the system tries to make a change to the base profile every time it sees a deviation, there is a potential danger of incorporating intrusive activities into the profile. The IDS must be able to adapt to these changes while still recognizing abnormal activities. If both intrusive behavior a a change in normal behavior occur during a particular time interval, the problem becomes more complicated. Again, determining which rules to add and which to remove is critical. There are also additional issues that need to be addressed in case of updating. The system should adapt to rapid changes as well as gradual changes in system behavior. Selecting the time interval at which the update should take place is also an important issue. If the interval is too long, the system may miss some rapid changes or short-term attacks. If the interval is too small, the system may miss some long-term changes.

So, we consider two problems as the major issues in developing an adaptive intrusion detection system. One is to select the time when the update should be made. The other is to select a mechanism to update the profile. To tackle the first issue, we can continuously measure the similarity between each day's activity and the profile and utilize this similarity trace. If the similarity stays above a threshold level, then the profile is taken to be a correct reflection of the current activities. If the similarity goes down below the threshold level, then there can be two possibilities: either the behavioral patterns are changing or the system is under attack. To take care of these two possibilities, we need to measure the rate of change in the similarity. If an abrupt change is encountered, it is interpreted as an intrusion, and that time window will not be used to update the profile. If a gradual negative change is encountered, then that time window will be used to update the profile. We can assume that behavioral change occurs gradually, not abruptly. This is illustrated in Figure 1. The activities before point A are considered to be normal and the profile does not need any update. Between points A and B, the patterns represent some behavioral change and the profile needs to be updated. Between points C and D, the patterns represent intrusive behavior and no update is made.

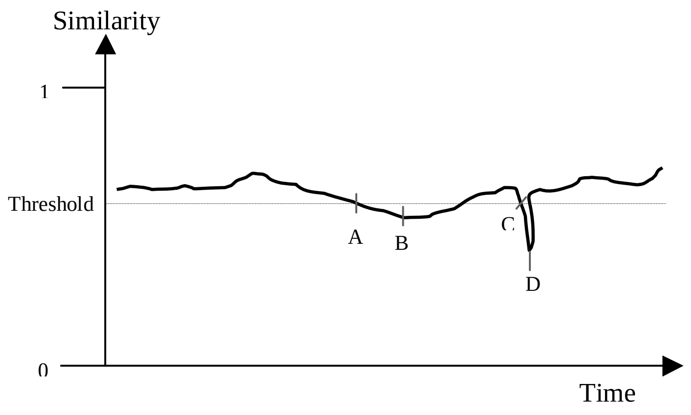

Figure 1: Change of similarity with time

It is not computationally feasible to archive audit data for a long time. Therefore, we will need to employ a sliding window technique to update the base profile. We can assume that system activities before a certain period of time are too old to characterize the current behavior, i.e., the audit records before that period are unlikely to contribute towards the rules that represent system activities. We can define an overlapping sliding window [t_1, t_2, ... , t_n] of n days. The update technique would reject transactions outside the sliding window as they are assumed to be old and outdated. As mentioned in Toivonen (1996), we would maintain both the strong rules (having high enough support and confidence) and the negative border (support and confidence are less than but close to the thresholds). As time goes on, a strong rule may start losing its support and confidence and rules in the negative border may start gaining support and confidence. We would discard some strong rules (loosing support and confidence in subsequent time windows) in the process and include some new rules. Though we will maintain both the strong rules and negative border of the strong rules, we will only use the strong rules to compute the similarity. This will facilitate the updating process in case of gradual behavior changes.

The last issue is which technique to apply to update the profile rule set that would minimize the amount of recomputation. The problem of maintaining discovered association rules was first studied in Cheung et al. (1996). They described the Fast Update Algorithm (FUP) for incrementally maintaining association rules from large databases. The incremental database is scanned for large itemsets of the original database to update their support counts in the modified database and only the itemsets passing the support threshold test with respect to the new modified database are retained. At the same time, all new large itemsets in the incremental database are created and the original database is scanned to retain ones passing the support threshold test with respect to the new modified database. The problem with FUP is that it can handle the maintenance problem only in case of insertions. Cheung, Lee, and Kao (1997) described a more general incremental updating technique FUP_2 for maintaining the association rules that can handler insertions, deletions, and modifications of transactions in the database. For insertions, FUP_2 is equivalent to FUP. For deletions, FUP_2 is a complementary algorithm of FUP. We will employ FUP_2 since in each transition from one time window to another, some audit records will be deleted (the least recent ones) and some audit records will be added (the most recent ones).

4. A FRAMEWORK FOR AN ADAPTIVE INTRUSION DETECTION SYSTEM

In this section we propose a rough framework for the adaptive maintenance of the profile rule set that can overcome the need for recomputation of the rules without sacrificing the detection capabilities. The profile rule ser can be updated by adding new rules, deleting old rules, or by modifying existing rules. Existing rules can be modified by changing their support and confidence. The flexible framework will exploit the rule generated during the earlier stages. We will store some rules that do not have sufficient support and confidence to be considered as strong at that time in addition to the strong rules. We will employ an overlapping sliding window approach that generates rules from recent data avoiding the use of old data. The profile will be maintained by periodic updates where strong rules and negative borders are added and other rules are discarded. The central idea behind the sliding window approach is the concept of a time window, an interval of time outside of which audit records are considered too old to reflect current system activities. The time window therefore acts to filter out outdated audit data and tries to build a profile based on only recent data that reflects the recent system activities.

Figure 2 presents an architecture for our framework. The process begins with an initial set of audit data. Genetic algorithms (Shi 2000) would be used to tune the fuzzy membership function parameters. Then fuzzy association rule mining will be applied to mine rules into a normal profile. During each time window, the audit data in the incremental part will be mined and compared with the profile rule set. There can be three possibilities. If the similarity stays above threshold, no update is needed and the system continues with the current profile. If similarity goes below threshold with a sharp negative change, intrusion will be signaled and the profile will not be updated. If similarity goes below threshold with gradual change, the profile will be updated with the audit data in the current time window.

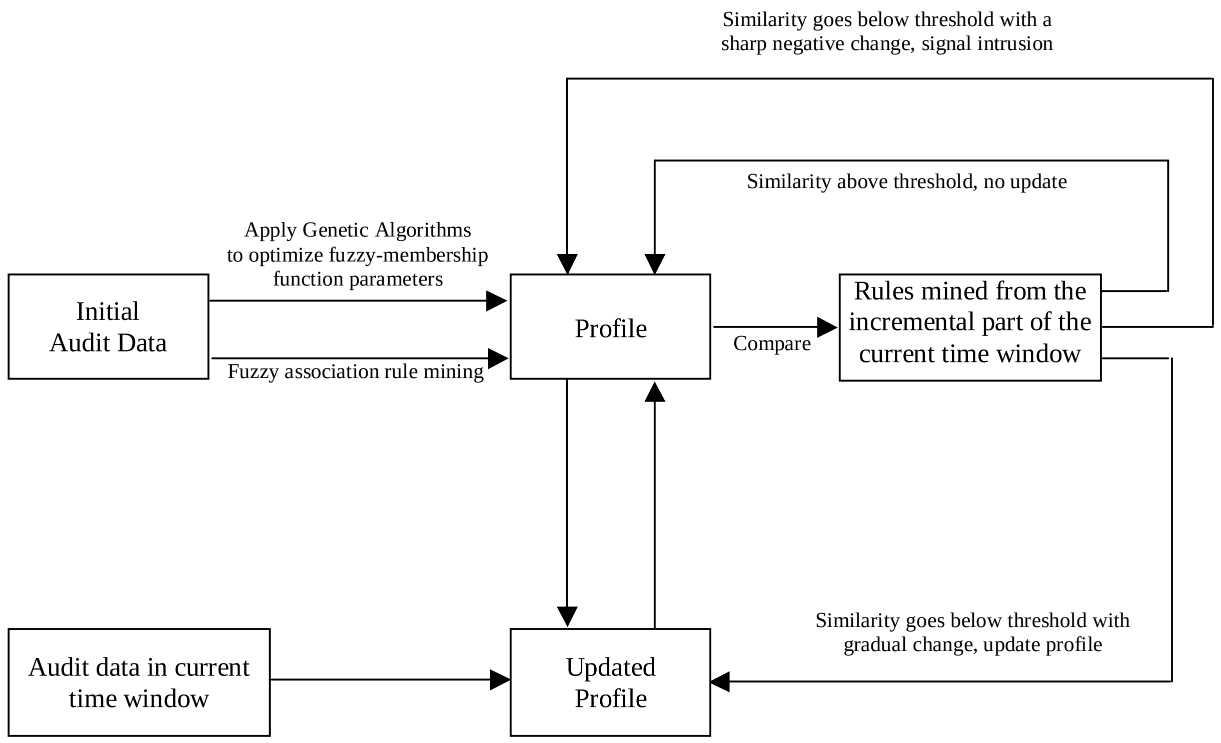
Similarity goes below threshold with a sharp negative change, signal intrusion
Similarity above threshold, no update
Apply Genetic Algorithms to optimize fuzzy-membership function parameters
Initial Audit Data
Fuzzy association rule mining
Profile
Compare
Rules mined from the incremental part of the current time window
Audit data in current time window
Updated Profile
Similarity goes below threshold with gradual change, update profile

Figure 2: The framework of the Adaptive Intrusion Detection System

5. CONCLUSION

Data mining methods provide automatic intrusion detection capabilities. They mine knowledge from audit data to characterize normal and abnormal user behavior. Some of the works done to-date were introduced in the paper. One of the major limitations of the systems is that they lack adaptability to changing behavior patterns. In this paper, we outlined a framework for an adaptive intrusion detections system using fuzzy data mining. There is one issue that we will address in future works. We haven't given any direction about how to tackle a time window that contains both intrusive data and non-intrusive data representing a behavioral change. We have to perform some drill-down operation to individual rules to distinguish between suspicious and non-suspicious rules as done in Barbara, Jajodia, and Wu (2000). There is one more area where we need to give some attention. The system in its current status, works with a static set of fuzzy membership function parameters determined at the beginning. But these values also need to tuned dynamically.

REFERENCES

Agrawal, R., T. Imielinski, and A. Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on management of data held in Washington, D.C., May 26-28, 1993*, 207-216.

Barbara, D., S. Jajodia, and N. Wu. 2000. Mining unexpected rules in network audit trails. Personal communications.

Bridges, S. and R. Vaughn. 2000. Fuzzy data mining and genetic algorithms applied to intrusion detection. In *Proceedings of the 23rd National Information Systems Security Conference held in Baltimore, MA, October 16-19, 2000*, 13-31.

Cheung, D., S. Lee, and B. Kao. 1997. A general incremental technique for updating discovered association rules. In *Proceedings of the 5th international conference on database systems for advanced applications (DASFAA'97) held in Melbourne, Australia, April 1-4, 1997*, 185-194.

Cheung, D., J. Han, V. Ng, and C. Wong. 1996. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of the 12th IEEE international conference on data engineering (ICDE'96) held in New Orleans, Louisiana, February 26 - March 1, 1996*, 106-114.

Graham, R. 1999. FAQ: Network intrusion detection systems. (Downloaded from http://www. infosyssec.com/infosyssec/netintrufaq.htm)

Lee, W. and S. Stolfo. 1998. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX security symposium* (*SECURITY '98*) *held in San Antonio, TX, January 26-29, 1998*, 79-94.

Luo, J. and S. Bridges. 2000. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. *International Journal of Intelligent Systems* 15(8): 687-703.

Mukherjee, B., L. Heberlein, and K. Levitt. 1994. Network intrusion detection. *IEEE Network* 8(3): 26-41.

Shi, Fajun. 2000. Genetic algorithms for feature selection in an intrusion detection application. Masters thesis report, Mississippi State University.

Sundaram, A. 1996. An introduction to intrusion detection. *Crossroads: The ACM Student Magazine* 2(4). (Downloaded from http://www.acm.org/crossroads/ xrds2-4/intrus.html)

Toivonen, H. 1996. Sampling large databases for association rules. In *Proceedings of 22nd international conference on very large data bases (VLDB'96) held in Mumbai, India, September 3-6, 1996*, 134-145.