OPEN ACCESS

International
Journal
Of Modern Engineering Research (IJMER)

Review of Intrusion and Anomaly Detection Techniques

Qamar Rayees Khan[1], Muheet Ahmed Butt[2], Majid Zaman[3], Mohammad Asger[4]

[1, 4] *Baba Ghulam Shah Badshah University, Rajouri (J&K), India.*
[2, 3] *University of Kashmir, Srinagar, (J&K), India*

Abstract: *Intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a resource. With the tremendous growth of network-based services and sensitive information on networks, network security is getting more and more importance than ever. Intrusion poses a serious security threat in a huge network environment. The increasing use of internet has dramatically added to the growing number of threats that inhabit within it. Intrusion detection does not, in general, include prevention of intrusions. Now a days Network intrusion detection systems have become a standard component in the area of security infrastructure. This review paper tries to discusses various techniques which are already being used for intrusion detection.*

Keywords: *Anomaly, Data Confidentiality, False Positive, Intrusion, Misuse.*

I. Introduction

Intrusion detection starts with instrumentation of a computer network for data collection. Pattern-based software sensors monitor the network traffic and raise alarms when the traffic matches a saved pattern. Security analysts decide whether these alarms indicate an event serious to warrant a response. A response might be to shut down a part of a network, to phone the internet service provider associated with suspicious traffic, or to simple make note of usual traffic for future reference [19].

This paper presents overview in the area of intrusion detection. We have divided the work into three sections: the first section analyzes anomaly detection systems and their performance. It gives a short overview of statistical methods for anomaly detection, Predictive pattern generation for anomaly detection and program based anomaly detection. The second section presents an overview of misuse detection techniques: language base misuse detection, expert systems and high level state machines for misuse detection. The third section presents an overview of specification based approaches to intrusion detection: process, behavior monitoring and process state analysis. Each of the subsections gives a short overview of the techniques of interest and then presents the data set used.

Background

If the network is small and signatures are kept up to date, the human analyst solution to intrusion detection works well. But when organizations have a large, complex network the human analyst quickly become overwhelmed by the number of alarms they need to review [19]. The sensors on the large networks, will generate over millions alarm per day and that number is increasing. This situation arises from ever increasing attacks on the commercial tools typically do not provide an enterprise level view of alarms generated by multiple sensor vendors. Commercial intrusion detection software packages tend to be signature oriented with little or no state information maintained. These limitations led us to investigate the application of data mining to this problem.

A secure network must provide the following:

1. **Data Confidentiality:** Data that are being transferred through the network should be **accessible** only to those that have been properly authorized.
   Data integrity: Data should maintain their integrity from the moment they are transmitted to the moment they are actually received. No corruption or data loss is accepted either from random events or malicious activity.
2. **Data availability:** The network should be resilient to Denial of Service attacks.

The first threat for a computer network system was realized in 1988 when 23-year old Robert Morris launched the first worm, which override over 6000 PCs of the ARPANET network. On February 7th, 2000 the first DoS attacks of great volume where launched, targeting the computer systems of large companies like Yahoo!, eBay, Amazon, CNN, ZDnet and Dadet. More details on these attacks can be found at [20].

The ideal approach to securing the network is to remove all security flaws from individual hosts, but less expensive and more feasible approaches are used for securing the computer systems. Most computer systems have some kind of security flaw that may allow intruders or legitimate users to gain unauthorized access to sensitive information. Moreover, many computer systems have widely known security flaws that are not fixed due to cost and some other limitations. Even a supposedly secure system or network can still be vulnerable to insiders misusing their privileges or it can be compromised by improper operating practices. So the ideal approach to secure a system can be to identify the process (system) with abnormal behavior and try to fix it.

II. Techniques

The two basic approaches to intrusion detection are misuse intrusion detection and anomaly intrusion detection. In *misuse intrusion* detection, know patterns of intrusion (intrusion signatures) are used to try to indentify intrusions when they happen.
In *anomaly intrusion* detection, it is assumed that the nature of the intrusion is unknown, but that the intrusion will result in behavior different from that normally seen in the system. Many detection systems combine both approaches

1.1 Anomaly Detection

1.1.1 Statistical Methods for Anomaly Detection

In statistical Method for Anomaly detection, the NIDS observes the activity of systems under study and generates some valuable information that is stored in the form of profiles to represent the behavior of the system. The profile contains four different sets of measures i.e. activity intensity measure, audit record distribution measure, categorical measures and ordinal measures. At every point of time two profiles are maintained for each system a current profile and a stored profile. As the records are processed the NIDS updates the current profile and calculates the abnormal behavior if any by comparing the current profile with the stored profile using a function of abnormality for all the measures in the profile. The stored profile is updated from time to time by merging it with the current profile. Statistical methods suffers from a major drawback i.e. a skilled attacker can train the method so that it can accept an abnormal behavior as normal secondly not all the behaviors can be modeled as (Kumar 95).

a) Haystack

Haystack [1] is an example of statistical anomaly-based IDS. It uses both user and group-based anomaly detection. It models system parameters as independent, Gaussian random variables. Parameters are considered as abnormal when they fall out of 90% of the data range for the variable. It maintains a database of user groups and individual profiles. If a user has not previously been detected, a new user profile with minimal capabilities is created using restrictions based on the user's group membership. It is designed to detect six types of intrusions: attempted break-ins by unauthorized users, masquerade attacks, penetration of the security control system, leakage, DoS attacks and malicious use.

b) Bayesian classification

Bayesian classification automatically determines the most probable number of classes for the data, continuous and discrete measures can be freely mixed. On the other hand, this approach does not take into account the interdependence of the measures, it is unclear whether the algorithm can perform classification incrementally as needed for on-line processing, suffers from difficulty to determine thresholds, susceptibility to be trained by a skillful attacker etc.

1.1.2 Predictive Pattern Generation for Anomaly Detection

Predictive pattern generation for anomaly detection takes into account the relationship and ordering between events. It assumes that events are not random, but follow a distinguishable pattern. Such methods learn to predict future activities based on the most recent sequence of events and flag events that deviate significantly from the predicted.

a) Machine learning

T. Lane and C. Brodley applied the machine learning approach to anomaly detection in [2]. They built user profiles based on input sequences and compared current input sequences to stored profiles using a similarity measure. To form a user profile, the approach learns characteristic sequences of actions generated by users. For learning patterns of actions the system uses the sequence as the unit of comparison. A sequence is defined as an ordered, fixed-length set of temporally adjacent actions, where actions consist of UNIX shell commands and their arguments. To characterize user behavior they used only

positive examples. For the purpose of classification, sequences of new actions are classified as consistent or inconsistent in reference to sequence history using a similarity measure. To classify the attack, they classify the similarity of each input sequence, yielding a stream of similarity measures. If the mean value of the current window is greater than the threshold then it is classified as normal.

b). Time-based inductive generalization

Teng, Chen and Lu [3] developed a technique that applies a time-based inductive learning approach to security audit trail analysis. The system developed sequential rules. The technique uses time-based inductive engine to generate rule-based patterns that characterize the normal behavior of users. Each event is one single entry in the audit trail and is described in terms of a number of attributes (event type, image name, object name, object type, privileges used, status of execution, process ID, etc). The events in the audit trail are considered to be sequentially related. Rules identify common sequences of events and the probability of each sequence being followed by other events. Rules with a high level of accuracy and high confidence are kept, less useful rules are dropped. They define two types of anomaly detection: deviation detection and detection of unrecognized activities. Low probability events are also flagged, using a threshold. Anomaly is detected if either deviation or unrecognized activity is detected. This method can identify the relevant security items as suspicious rather than the entire login session and it can detect some anomalies that are difficult to detect with traditional approaches. It's highly adaptive to changes, can detect users who try to train the system, activities can be detected and reported within seconds. On the other hand, some unrecognized patterns might not be flagged since the initial, predicting portion of the rule may be matched.

2.1.3. Program-Based Anomaly Detection

a) Program Modeling

The program modeling approach (Garth Barbour Ph.D. thesis [4]) presents an algorithm that efficiently learns the program behavior. The author claims that the number of false positives never increases with additional training. If there is a false positive, the representation can learn to accept the run in the future without negatively impacting the performance of other runs. He also claims that his technique can detect novel attacks as they occur, without manual specification of program behavior or attack signatures. His algorithm learns an approximation of the program's behavior. The number of false positives decreases with training set size. He uses a non-statistical method since statistical methods would miss minor deviations.Barbour presents an algorithm that learns program behavior, without any input parameters presented in addition to his algorithm, which would add more reliability to his results. Input parameters may add additional reliability to detection because in programs with significant number of system calls, we may have different arguments associated with those system calls. If a program with limited capabilities is observed, it does not generate too many different system calls and, hence it is easy to detect any misuse of that program. On the other hand, if we are dealing with a powerful program, it generates a significant number of system calls and therefore, it is more difficult to detect misuse of that program.

b) Computational Immunology

Stephanie Forrest is investigating anomaly detection on system processes from the perspective of immunology [6], [7], [8]. In [6] the authors base their approach on the immunological mechanism of distinguishing self from non- self and [7] use look ahead pairs. They take short sequences of system calls, called n-grams that represent samples of normal runs of programs and compare them to sequences of system calls made by a test program. If any run has a number of mismatches that is higher than a certain number (percent), it is flagged as anomalous. They extended their work to variable length sequences, based on random generation of examples of invalid network traffic to detect anomalous behavior. Unlike Barbour [4], where there is no delay in detecting anomalies, the approach of Forrest has a certain delay due to the fact that the program computes the percent mismatch to be able to flag or not flag the run as anomalous. Hofmeyr [7] used the rule of *r* contiguous bits and showed that fixed length sequences give better discrimination than look-ahead pairs. Christina Warrender [8] modelled normal behavior of data sets described below using stide, t-stide, RIPPER and HMMs. She compared learning methods for application to n-grams: enumerating sequences using look-ahead pairs and contiguous sequences, methods based on relative frequency of different sequences, RIPPER (a rule-learning system) and HMMs. They used RIPPER to learn rules representing the set of n-grams. Training samples consisted of set of attributes describing the object to be classified and a target class to which the object belongs. RIPPER takes training samples and forms a list of rules to describe normal sequences. For each rule a violation score is established. Each set that violates a high confidence rule is a mismatch. They created HMMs so that there is one state for each n-gram.

The results showed that HMMs performed marginally better than other methods. She showed that HMMs performed only marginally better than other simple and not computationally expensive methods and concluded that the choice of data stream (short sequences of system calls) was more important than the particular method of analysis.

c). Variable-length audit trail patterns

Wespi [9] points out that usage of fixed-length patterns to represent the process model in [6] has no rule for selecting the optimal pattern length. On the other hand, Forrest *et.al. [7],* [8] point out that the pattern length has an influence on the detection capabilities of the IDS. Wespi *eta* Initially were not able to prove significant advantage of variable-length patterns in [10]. Later, in [9] they showed that the previously stated method outperforms the method of Forrest *et. Al [7], [8].* They use functionality verification test (FVT) suites provided by application developers to generate a training set based on the normal program's specified behaviors. Their system consists of an off-line and on-line part.

In the next step they apply the Teiresias algorithm [11] to generate the list of maximal variable-length patterns followed by some pruning during their preprocessing. During operation, a recursive pattern matching algorithm attempts to find the best matches for the test run. The measure of anomaly is the longest sequence of unmatched events (more than six consecutive unmatched events are treated as an intrusion). Initial work generated variable-length patterns using a suffix-tree constructor, a pattern pruner, and a pattern selector. ID based on this technique can detect the attacks but is prone to issue false alarms. Then they used a modified version of the pattern generator to create variable-length patterns that reduce the number of false alarms substantially without reducing the ability to detect intrusions. The authors claim that their algorithm performs better than the fixed-length sequences of Stephanie Forrest, but they compare their algorithm only for the ftpd process and then claim that their algorithm outperforms algorithms presented in [6],[7]. This is arguable because it might be the case that they presented one example where their algorithm outperformed the other algorithms and did not present the instances where their algorithm failed. Hence, we can conclude that on that specific process algorithm of Wespi *et. al.* [9] did outperform the algorithm of Forrest *et. al. [6],* [7] but, on the other hand, that may or may not be true for other processes.

d) Program Behavior Profiles

The work of Ghosh *et. al.* [5] is based on the work of the UNM group. The goal of their approach was to employ machine learning techniques that can generalize from past behavior so that they are able to detect new attacks and reduce the false positive rate. The first approach they used was equality matching. The database used was the normal behavior of programs. Instead of using strace, they use BSM data to collect the system call information and collect them into fixed size windows. They used the fact that the attacks cause anomalous behavior in clusters, like Kosoresow in [12]. Their decision choice looks for local clustering of the mismatches. The second model was the feed-forward topology with back propagation learning rules. Database was not on per-user basis like in [2], but they built profiles of software behavior and malicious software behavior. During training many networks were trained for each program and the network that performed the best was selected. The training process consisted of exposing networks to four weeks of labeled data and performing back propagation algorithm to adjust weights. They trained Artificial Neural Networks (ANN) to recognize whether small, fixed sequences of events are characteristic of the programs in which they occur. For each sequence, the ANN produces an output value that represents how anomalous the sequence is the leaky bucket algorithm was used to classify the program behavior and using leaky bucket they made use of the rule that two close anomalies have higher influence than when they are apart. The final approach was Elman network, developed by Jeffrey Elman, since they wanted to add information about the prior sequences (DFA approach lacked flexibility and ANNs have the ability to learn and generalize). In order to generalize they employ a recurrent ANN topology of an Elman network. They train the network to predict the next sequence that will occur at any point in time. The Elman network approach had the best results, with 77.3% detection and no false positives and 100% detection and fewer false positives than in two other cases. They also tried two approaches in which they used FSMs for their representation. Audit data was condensed into a stream of discrete events, where events are system calls recorded in data. Separate automata are constructed for different programs whose audit data are available for training. The training algorithm is presented with a series of n-grams taken from non-intrusive BSM data for a given program. The audit data is split into sub-sequences of size $n + I$ (*n* elements define a state and / elements are used to label a transition coming out of that state). The second FA-like approach, called a string transducer makes an attempt to detect subtler statistical deviations from normal behavior. It associates a sequence of input symbols with a series of output symbols. During training they estimate the probability distribution of the symbols at each state and during testing, deviations fr8m this probability distribution indicate anomalous behavior. They use FA whose states correspond to n-grams in the BSM data. For each state they also record information about successor /-grams that are observed in the training data. During training their goal is to gather statistics about successor /-grams. They estimate the probability of each /-gram by counting.

1.1.3 Anomaly detection using data mining

The ADAM (Audit Data Analysis and Mining) system [13] is an anomaly detection system. It uses a module that classifies the suspicious events into false alarms or real attacks. It uses data mining to build a customizable profile of rules of normal behavior and then classifies attacks (by name) or declares false alarms. ADAM is a real-time system. To discover attacks in TCPdump audit trail, ADAM uses a combination of association rules, mining and classification. The system builds a repository of normal frequent item sets that hold during attack-free periods. Then it runs a sliding window online algorithm that finds frequent item sets in the last D connections and compares them with those stored in the normal item set repository. With the rest, ADAM uses a classifier which has previously been trained to classify the suspicious connections as a known type of attack, unknown type or a false alarm.

Association rules are used to gather necessary knowledge about the nature of the audit data. They derive a set of rules in form *X* —> Y, where X and Y are sets of attributes. There are two parameters associated with a rule: *support s* and *confidence c*. The definitions of *s* and *c* are as follows. The rule *x* Y has support *s* in the transition set *T* if s% of transactions in T contains X or Y. The rule *x* 4 Y has confidence *c* if c% of transactions in T that contain X also contain Y. If the item set's support surpasses a threshold, that item set is reported as suspicious. The system annotates suspicious item sets with a vector of parameters. Since the system knows where the attacks are in the training set, the corresponding suspicious item set along with their feature vectors are used to train a classifier. The trained classifier will be able to, given a suspicious item set and a vector of features, classify it as a known attack (and label it with the name of attack), as an unknown attack or a false alarm.

1.2 Misuse Detection

1.2.1 Language-based Misuse Detection

Language-based Misuse Detection systems accept a description of the intrusions in a formal language and use this to monitor for the intrusions. Most languages for misuse systems, including the one used by NIDES, are low-level and have limited expressiveness.

a) ASAX

Habra *et. al.* define Advanced Security audit trail Analysis for universal audit trail analysis [14]. ASAX (Advanced Security audit trail Analysis on UNIX) uses RUSSEL, a rule-based language, specifically appropriate for audit trail analysis. ASAX sequentially analyzes records using a collection of rules that are applied to each audit record. A subset of rules is active at any time. They define a normalized audit data format (NADF) as a canonical format of the Operating System's audit trail.

1.2.2 Expert Systems

Expert systems use lists of conditions that, if satisfied, indicate that an intrusion is taking place. The conditions are rules that are evaluated based on system or network events. These rules are specified by experts familiar with the intrusions, generally working closely with the developers of the system.

1.2.3 High-level state machines for misuse detection

The State Transition Analysis **Tool (STAT)** [15] describes computer penetrations as attack scenarios. It represents attacks as a sequence of actions that cause transitions that lead from a safe state to a compromised state. A state represents snapshot of the system's security-relevant properties that are characterized by means of assertions, which are predicates on some aspects of the security state of the system. The initial state of a transition diagram does not have any assertions. Each state transition diagram starts with a signature action that triggers monitoring of the intrusion scenario. Transitions between states are labeled by the actions required to switch from one state to another. These actions do not necessarily correspond to audit records. The resulting state transition diagram forms the basis of a rule-based intrusion detection algorithm.

a). USTAT [16] is an implementation of the STAT tool developed for Solaris BSM. It reads specifications of the state transitions necessary to complete an intrusion and evaluates an audit trail. Two preconditions must be met to use

USTAT: the intrusion must have a visible effect on the system state and the visible effect must be recognizable without knowing the attacker's true identity.

b). STATL [17] is a language that allows description of computer penetrations as sequences of actions that an attacker performs to compromise a computer system. The attack is modeled as a sequence of steps that bring a system from an initial safe state to a final compromised state. An attack is represented as a set of states and transitions, where each transition has an associated action. The notion of *timers* is used to

express attacks in which some events must happen with an interval following some other event (set of events). Times are declared as variables using the built-in type timer. STAT can detect cooperative attacks and attacks that span multiple user sessions, can specify rules at higher level than audit records, is easier to create and maintain than other rule-based methods, can represent a partial ordering among actions, can represent longer scenarios than other rule-based systems. On the other hand, it has no general-purpose mechanism to prune partial matches of attacks other than assertion primitives built into the model.

1.2.4 Emerald

EMERALD (Event monitoring enabling responses to anomalous live disturbances) [18] employs both anomaly and misuse detection. It includes *service analysis* that covers the misuse of individual components and network services within a single domain, *domain-wide analysis* that covers misuse visible across multiple services and components and *enterprise-wide analysis* that covers coordinated misuse across multiple domains. They also introduce the notion of *service monitors* that provide localized analysis of infrastructure and services. EMERALD consists of three analysis units: profiler engines, signature engines and resolver. Profiler engine performs statistical profile-based anomaly detection given a generalized event stream of an analysis target. Signature engine requires minimal state management and employs a rule-coding scheme. The profiler and signature engines receive large volumes of event logs and produce a smaller volume of intrusion/suspicion reports and send that data to the resolver. The signature analysis subsystem allows administrators to instantiate a rule set customized to detect known problem activity occurring on the analysis target.

III. Conclusion

This paper elaborates the foundations of the main anomaly based network intrusion detection technologies. Considering the surveyed literature, it is clear that in order to be able to secure a network against the novel attacks, the anomaly based intrusion detection is the best way out. However, due to its immaturity there are still problems with respect to its reliability. These problems will lead to high false positives in any anomaly-based IDS. In order to solve this problem, usually different anomaly detection techniques can be used. Method like Haystack can be used to detect a different set of attacks like break-in's Dos etc. In the same way some authors have used Machine Learning, Program based, and time based approaches for network intrusion detection. All the approaches that have been surveyed in the work perform better from one another in some cases but above all the techniques still need some improvements in terms of false positive rates and true negative rate.

However, there are still many challenges to overcome in the field of intrusion detection, But the presented information constitutes an important point to start for addressing Research & Development in the field of IDS. We find that the majority of surveyed works can be used as starting point for continuing research work in the field of intrusion detection.

BIBLIOGRAPHY

[1] Stephen E. Smaha "Haystack: An Intrusion Detection System", Proc. of the IEEE 4th Aerospace Computer Security Applications Conference, Orlando, FL, Dec. 1988, 37 — 44 97

[2] T. Lane, C. Brodley, "An application of machine learning to anomaly detection", Proc. 20th NIST-NCSC National Information Systems Security Conference,1997.

[3] H. Teng, K. Chen, S. Lu, "Security Audit Trail Analysis Using Inductively Generated Predictive Rules", Proceedings of the sixth conference on Artificial intelligence applications January 1990.

[4] Garth Barbour "Program Modeling: A Machine Learning Approach To Intrusion Detection", PhD Thesis, University of Marryland at College Park, 2002.

[5] A. K. Ghosh, A. Schwartzbard, M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection", Proceedings of the Workshop on Intrusion Detection and Network monitoring, USENIX 1999.

[6] S. Forrest, S. A. Hofmeyr, A. Somayaji and T. A. Longstaff "A sense of self for Unix processes". In Proceedings of the 1996 IEEE Symposium on security and Privacy, pages 120-128, Los Alamitos, CA, 1996. IEEE Computer Society Press

[7] S. A. Hofmeyr, S. Forrest, A. Somayaji "Intrusion detection using sequences of system calls" Journal of Computer Security Vol. 6, pp. 151-180 (1998).

[8] C. Warrender, S. Forrest, B. Pearlmutter "Detecting Intrusions Using System Calls: Alternative Data Models", 1999 IEEE Symposium on Security and Privacy, May 9-12, 1999.

[9] A. Wespi, M. Dacier, H. Debar "Intrusion Detection Using Variable-Length Audit Trail Patterns", Recent Advances in Intrusion Detection — Lecture Notes in Computer Science ed. by H. Debar, L. M, S.F. Wu., Berlin, Springer-Verlag, vol.1907, p.110-29 in 2000

[10] A. Wespi, M. Dacier, H. Debar "An Intrusion-Detection System Based on the Teiresias Pattern Discovery Algorithm ", Technical Report RZ3103, Zurich Research Laboratory, IBM Research Division, 1999.

[11] I. Rigoutsos, A. Floratos "Combinatorial Pattern Discovery in Biological Sequences: the TEIRESIAS Algorithm",

Bioinformatics, Vol. 14, No. 1, 1998,pp. 55-67.
[12] A. P. Kosoresow, S. A. Hofmeyr, "Intrusion Detection via System Call Traces", IEEE Software, vol. 14, pp. 24-42, 1997.
[13] D. Barbara, *etaL,* "ADAM: Detecting Intrusions by Data Mining", proceedings of the 2001 IEEE Workshop on Information Assurance and Security, June 2001.
[14] J. Habra, B. Charlier, A. Mounji, I. Mathieu, "ASAX: Software architecture and rule-based language for universal audit trail analysis", Second European Symposium on Research in Computer Security (ESORICS 92).
[15] G. Vigna, S. T. Eckmann, R. A. Kemmerer, "The STAT Tool Suite", in Proceedings of DISCEX 2000, Hilton Head, South Carolina, January 2000,IEEE Press.
[16] K. Ilgun, "USTAT: A real-time intrusion detection system for UNIX", in Proceedings of the 1993 IEEE Symposium on Security and Privacy, Oakland, CA, May 1993.
[17] S. T. Eckmann, G. Vigna, R. A. Kemmerer, "STATL: An Attack Languagefor State-based Intrusion Detection", in Proceedings of the ACM WOokshopin Intrusion Detection, Athens, Greece, November 2000.
[18] A. Porras and P. G. Neumann, "EMERALD: Event monitoring EnablingResponses to Anomalous Live Disturbances", In Proceedings of the National Information Systems Security Conference, 1997, pp. 353- 365.
[19] Eric Bloedorn, Alan D. Christiansen, William Hill,Clement Skorupka, Lisa M.Talbot, Jonathan Tivel "Data Mining for Network Intrusion Detection: How to Get Started"
[20] http://www.securitvfocus.com/news/2445 .