Hybrid Intrusion Detection with Rule Generation

V.V. Korde, N.Z. Tarapore, S.R. Shinde, and M.L. Dhore

Department of Computer Engineering, Vishwakarma Institute of Technology, Pune
{korde.vaibhav,ntarapore}@yahoo.com

Abstract. This paper reports a new experimental hybrid intrusion detection system (HIDS). This hybrid system combines the advantages of Misuse-based intrusion detection system (IDS) having low false-positive rate and the ability of anomaly detection system (ADS) to detect novel unknown attacks. This is done by mining Internet connections records for anomalies. We have built ADS that can detect attacks not detected by Misuse-based systems like Snort or Bro systems. Rules are extracted from detected anomalies and then are added to Misuse-based system's rule database. Thus Misuse-based intrusion detection system can detect new attacks. The system is trained and tested using Massachusetts Institute of Technology/ Lincoln Laboratory (MIT/LL) DARPA 1999 dataset respectively. Our experimental results show a 69 percent detection rate of the HIDS, compared with 47 percent in using the Snort. This increase in detection rate is obtained with around 0.08 percent false alarms. This approach provides a better way to deal with novel attacks using ADS along with a trustworthy misuse-based Intrusion detection system.

1 Introduction

The widespread use of Internet and computer networks experienced in the past years has brought, with all its benefits, another kind of threat: those of people using illicit means to access, invade and attack computers. One can use a firewall as a preventive measure to maintain the security goals of computer networks. But, it simply restricts access to the designated points. A computer network intrusion is a sequence of related actions by a malicious adversary whose goal is to violate some policy regarding appropriate use of computer network. Since a preventive approach such as firewall is not enough to provide sufficient security for computer system, intrusion detection systems (IDSes) are introduced as a second line of defense.

IDSes can be distinguished by their differing approaches to event analysis. Misuse-detection is the most widely used approach in commercial IDS technology today. But the problem with these systems is that they cannot detect novel attacks. Another approach is called anomaly detection. It uses rules or predefined concepts about "normal" and "abnormal" system activity to distinguish anomalies from normal system behavior and to monitor, report, or block anomalies as they occur. Anomaly based Intrusion Detection System has the benefit of detecting novel attacks but has high false positive rate. On the other hand, misuse-based systems are rule-based having higher accuracy. Misuse-based Intrusion Detection System fails to detect novel attacks. To overcome these limitations, both Anomaly-based and Misuse-based Intrusion Detection Systems should be combined.

In this paper, a new hybrid intrusion detection system (HIDS) is presented. This system combines the positive features of both intrusion detection models to achieve higher detection accuracy, lower false alarms, and thus a raised level of cyber-trust. Our HIDS is network-based, which should not be confused with the host-based IDS with the same abbreviation by other.

The rest of the paper is organized as follows: Section 2 reviews related works and distinguishes the new approach from previous solutions. Section 3 gives system design internals. We present system implementation details in Section 4. Experimental performance results are reported in Section 5.

2 Related Works and Our Approach

A significant amount of research has been done and various approaches has been used for designing an Intrusion Detection System These approaches attempt to build some kind of a model over the normal data and then check to see how well new data fits into that model [3],[4],[5]. In the past, data mining techniques such as using association rules were suggested to build IDS [7].

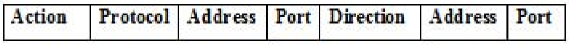
Daniel Barbara, et al., [3] described the design and experiences with the ADAM (Audit Data Analysis and Mining) system, which was used as a testbed to study how useful data mining techniques can be in intrusion detection. L. Ertoz, et al [4] introduced the Minnesota Intrusion Detection System (MINDS), which used a suite of data mining techniques to automatically detect attacks against computer networks and systems. H. Mannila and H. Toivonen [14] have compared different approaches to intrusion detection systems to supply a norm for the best-fit system. Snort [9] and Bro [10] are two widely used IDSes that are based on the misuse mode. The MIT/LL IDS (DARPA) evaluation data set and reported IDS performance results were analyzed in [12], [13], [11]. This data set is used to test the effectiveness of proposed HIDS. The concept of frequent episode rules (FERs) was first proposed by Mannila and Toivonen [14]. Tung-Ying Lee, et al [9] made first attempt to study a special episode rule, named serial episode rule with a time lag in an environment of multiple data streams.

In this paper, we propose the HIDS architecture. It is tested for its effectiveness through experiments. The HIDS integrates the flexibility of ADS with the accuracy of misuse-based IDS. ADS is designed by mining FERs [6], [14], over connections from Internet traffic. New rules are generated from anomalies detected by ADS. This new approach automatically enables HIDS to detect similar attacks in the future.

3 System Design Internals

In this section, we introduce the overall system design. Our aim is to combine a misuse based detection system with an Anomaly Detection System. The misuse based system used for this experimentation is Snort.

Snort is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. Snort has three primary modes: sniffer, packet logger, and network intrusion detection. One can write one's own rules with Snort to suit the particular needs of one's network. A Snort rule is composed of two major parts: rule headers and rule options. The rule header contains information about what action a rule takes. It also contains criteria for matching a rule against data packets The action part



Action	Protocol	Address	Port	Direction	Address	Port

Fig. 1. Snort rule format

of the rule determines the type of action taken when criteria are met and a rule is exactly matched against packet. Typical actions are generating an alert or log message or invoking another rule. The protocol part is used to apply the rule on packets for a particular protocol only. The address parts define source and destination addresses. Source and destination addresses are determined based on direction field. In case of TCP or UDP protocol, the port field determines the source and destination ports of a packet on which the rule is applied. In case of network layer protocols like IP and ICMP, port numbers have no significance.

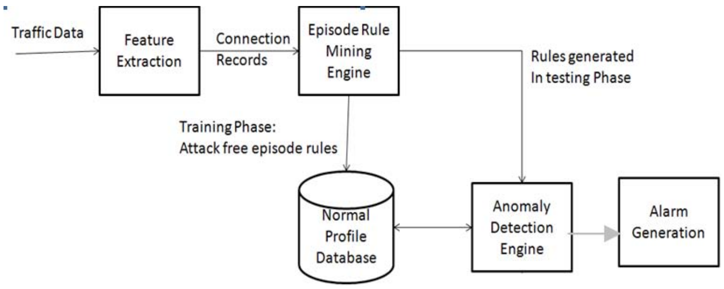


Fig. 2. Anomaly Detection System Architecture

3.1 Anomaly Detection System

Figure 3 explains the Data mining scheme for network anomaly detection. As shown in the figure, there are following major components.

Feature Extraction: Extracts required features from the audit data.

Episode Rule Mining Engine: This component works in two phases: training and testing.

Anomaly Detection Engine: In accordance with normal profile database the anomaly detection engine checks the newly received rules with that of those stored in the database. Depending on the results it triggers the Alarm generation component.

Alarm Generation: This component will generate alarm.

3.2 Episode Rules

An Internet episode is represented by a sequence of connection events. An episode can be generated by legitimate users or malicious attackers. Frequent episodes are mostly resulted from normal users. A rare episode is likely caused by intruders.

Let 'T' be a set of traffic connections and A be a set of attributes defined over 'T'. For example, 'A' consists of timestamp, duration, service, srchost, desthost for TCP connections. Let 'I' be a set of attribute-value pairs defined over 'A'. For example, I = timestamp = 15 sec, duration = 1 sec, service = http, srchost = 192.168.1.1, desthost = 192.168.1.10 for a typical http connection.

1. **Frequent Episode Rules**
   In general, an FER is expressed by the expression:

$$L_1, L_2, ...L_n \rightarrow R_1, R_2, ...R_m(c, s, window) \tag{1}$$

   where $L_i (1 \leq i \leq n)$ and $R_j (1 \leq j \leq m)$ are ordered itemsets in a traffic record set T. We call $L_1, L_2, ...L_n$ the LHS (left hand side) episode and $R_1, R_2, ...R_m$ the RHS (right hand side) of episode of the rule. Note that all itemsets are sequentially ordered, that is $L_1, L_2, ...L_n, R_1, R_2, ...R_m$ must occur in the ordering as listed. However, other itemsets could be embedded within our episode sequence. We define the support and confidence of rule by the following two expressions:

$$s = Support(L_1 \cup L_2 \cup .... \cup L_n) \tag{2}$$

$$c = \frac{Support(L_1 \cup L_2 \cup .... \cup L_n \cup R_1 \cup R_2 \cup .... \cup R_m)}{Support(L_1 \cup L_2 \cup .... \cup L_n)} \tag{3}$$

2. **Axis Attributes**
   Because the FER generation does not take any domain-specific knowledge into consideration, many ineffective FERs are generated. How to eliminate these ineffective rules is a major problem in traffic data mining for effective rule generation. For example, the association rule:

$$(srcbytes = 200) \rightarrow (destbytes = 300)$$

   is of little interest to the intrusion detection process, since the number of bytes sent by the source (src_bytes) and destination (dst_bytes) is irrelevant to the threat conditions. Lee et al [7] had introduced the concepts of axis attributes to constrain the generation of redundant rules. All Itemsets in an

FER must be built only with axis attributes. Axis attributes are independent of attacks being detected. The choice of axis attributes will reduce the number of FERs generated. According to Lee, axis attributes are selected from essential attributes which are enough to identify a connection. Different combinations of the essential attributes form the axis attributes. The axis attributes can easily be chosen with domain knowledge.

3. **Pruning of Ineffective Episode Rules**
   Keeping all rules generated will enlarge the search space and thus the overhead. The following FER transformation laws will reduce the rule search space significantly.
   - *Transposition of Episode Rules:* Suppose we have two FERs as and . Therefore, the second rule can be induced by the first rule. We only need to keep the first rule. The general rule of thumb is to make the LHS as short as possible. In general, rules with shorter LHSs are more effective than rules with longer LHSs.
   - *Elimination of Redundant Episode Rules:* Many FERs detected from the network traffic have some transitive patterns. Suppose we have two rules A $\rightarrow$ B and B $\rightarrow$ C in the rule set. Then, the longer rule A $\rightarrow$ B, C is implied. Since we reconstruct this rule from two shorter rules, the longer rule A $\rightarrow$ B, C becomes redundant.
4. **Rule Generation**
   Generating the rules is the most CPU-intensive task. This is because we need to generate all possible combinations of the sequences in a given window size. To generate these episode rules, we first need to find the sequence of the events occurring in the given window size. This is done by querying the database of extracted features to retrieve the event sequence in particular window size. Once the event sequence is retrieved it is fetched to the script for generation of Episode rules. The script generates all Episode rules and also finds the support and confidence for each episode rule.

3.3 MIT/LL DARPA Dataset

We have used MIT/LL's DARPA 1999 dataset for system training and testing purpose. DARPA 1999 dataset is a five week dataset. (Twenty-two hours a day and five days per week). The first and third weeks of the training data do not contain any attacks. The fourth and fifth weeks of data are the "Test Data". These two weeks of testing data consists of network based attacks along with the normal background data.

3.4 System Architecture

Initially the incoming traffic will be fed to Snort to filter out known attacks with existing rules of Snort. Then remaining traffic will be passed to Episode Mining Engine. Episode Mining Engine will generate Frequent Episode Rules (FER). Newly generated FER are compared with those stored in a normal profile database. The anomalous episodes are used to generate rules. Newly generated rules are inserted into Snort's attack rules database.

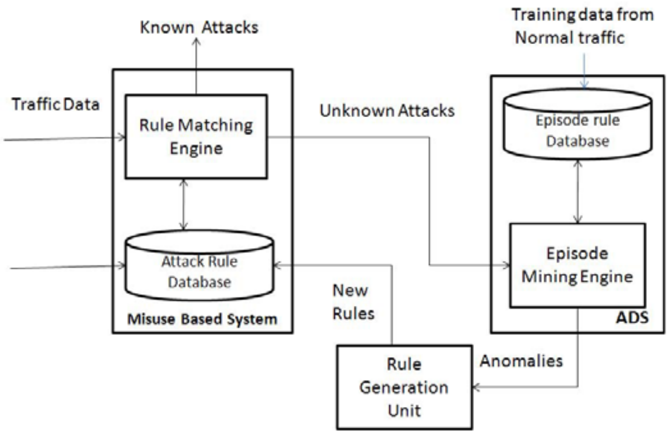


Fig. 3. Proposed System Architecture

4 System Implementation

In this section we first describe some tools used for implementation of the proposed system. Then we give details about stepwise implementation of the proposed system.

The system is implemented and tested on Fedora operating system. Snort 2.9.3 is used for the experimentations. The system is implemented using Java along with Shell script and some Linux based tools. Since the MIT/LL dataset is in the form of TCP dump files, we need to use the tools that can read them as fast as possible and extract the required detail. We have used tcptrace and Tcpdump utilities available on Linux.

4.1 Feature Extraction and Preprocessing

Since we aim at detecting anomalies of network traffic, the content features are not used in this work. The connection features and temporal statistics will be used in HIDS construction. Total features extracted from the connections are enlisted in table 1. The axis attributes described in section III are chosen from these features. Connection level features are used in both FER and rules generation. Temporal statistics are related to connections with the same reference features. They can be used to improve the accuracy of rules generated. Table 2 gives the list of temporal features calculated from connection features extracted.

Table 1. Connection Features Extracted

Feature Name	**Description**	**Feature Name**	**Description**
timestamp	the time when Connection Begins	service	Network Service on destination
duration	duration of the connection	icmp_type	ICMP message type
ip_proto	IP protocol type	src_bytes	Bytes sent by source
src_ip	Source IP address	dst_bytes	Bytes sent by destination
dst_ip	Destination IP address	flags	SYN/FIN Flags in the connection

Table 2. Temporal Features Extracted

Feature Name	**Description**
src_count	No of connections from same source
dst_count	number of connection to same destination
service_count	Number of connection for same service
avg_duration	Average duration of connection for same service
avg_src_bytes	Average bytes sent by source
avg_dst_bytes	Average bytes sent by Destination

After the features are extracted, we need to calculate relative timestamp, order them and assign a transaction ID for each one of them. The axis attributes chosen for ADS built here are service, IP protocol, and flags. These attributes are enough to classify the instance of a connection. Each such unique possible set of these three values is considered as an Event. These events with unique Event IDs are stored in the EventList database. In the pre-processing phase, every transaction ID is mapped with the Event ID and stored with the relative timestamp in EventOrder database.

4.2 Rule Generation

For generating rules we need to find all the connection records related to the anomalies detected. The table 3 gives mapping between Connection Attributes and Snort Rule Keywords. Let's say we have got an anomaly detected as A $\rightarrow$ B. Where A and B denote events with the following attributes.

$$A(Event\_ID = A, ip\_proto = icmp, icmp\_type = echo\_req, flags = 0)$$
$$B(Event\_ID = B, ip\_proto = icmp, icmp\_type = echo\_req, flags = 0)$$

Here flag=0 means no SYN/FIN flags are observed (obvious for an ICMP connection).Now after searching back in the EventOrder database if we find that these events have occurred many times in a given window size of say 10 sec, it will produce a higher support as well as confidence values for the same. Thus we can conclude to the following Snort rule
alert icmp EXTERNAL_NETany <>HOME_NET any (msg: "Anomalous Behaviour"; itype: 8; threshold : type both; count 10 seconds; sid:100001;rev:0)

Table 3. Mapping between connection attributes and Snort rule keywords

Attribute Name	**Snort Rule Keyword**	**Short Description**
protocol	protocol	IP protocol type
src_ip	source IP address	Source IP address
dst_ip	destination IP address	destination IP address
service	destination port number	Service type
icmp_type	itype	ICMP message type
src_bytes	Dsize	Packet Payload size
flags	Flags	TCP flags
land	Sameip	Same source and destination Address
src_count	threshold: track by_src, count $< n >$	No. of connection to the same destination
dst_count	threshold: track by_dst, count $< n >$	No. of connection from the same source

5 Experimentation and Results

This section gives brief details about the experiments that were performed on the system built to generate the performance results. The user needs to select a window size (in seconds). The window size will be used by the system as a window size in sliding window implementation of episode rule generation. User needs to specify values of support and confidence for selection of the episode rule into the database.

The system was tested with different values for window size, support and confidence values. Thus a normal profile is created by training the system with week 1 and week 3 dataset. The system was tested by using DARPA 1999's week 4 and week 5 testing dataset. The graph in figure 4.a shows the number

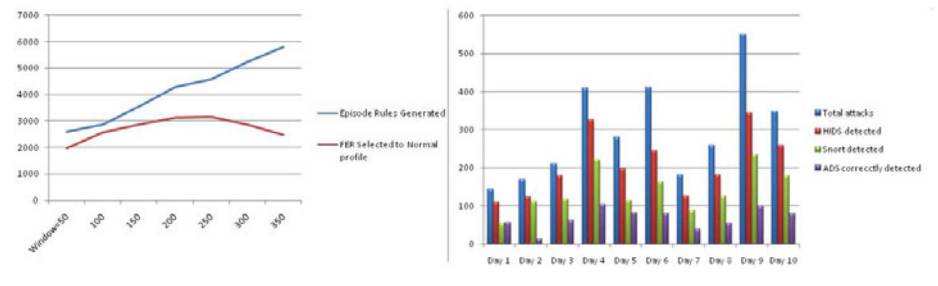


Fig. 4. a)Episode Rule Generation Vs. Window size b)Total Number of Intrusions Detected

of Episode Rules generated for different window sizes. Episode rules here are different from those discussed above in such a way that, rules stated here are unique episode rules among all generated above. Frequent Episode Rules are chosen by applying the criteria of support and confidence to them. From the graph as shown in figure 4.a it is clear that the number of FER that are selectable for normal profile dataset are steady between window sizes 200 and 250. After window size = 250, the total Episode rules becomes higher but comparatively very few FERs are produced. For system testing purpose, DARPA 1999's week 4 and week 5 was used. The detection list is provided by them. The attack results obtained from the detection by the HIDS were compared to that of the attacks in the attack list file. This is shown by the graph as shown in Fig 4.b. The misuse-based Intrusion Detection System: Snort is quite capable of detecting the attacks present in the test dataset. The anomaly detection system has also performed well. There were in total 2959 attacks present in the dataset. The intrusion detection rate (denoted by δ) is formally defined by

$$Detectionrate\delta = \frac{detectedattacks}{totalattacks} \tag{4}$$

Snort could detect 1407 attacks with the help of its own rules. Detection rate

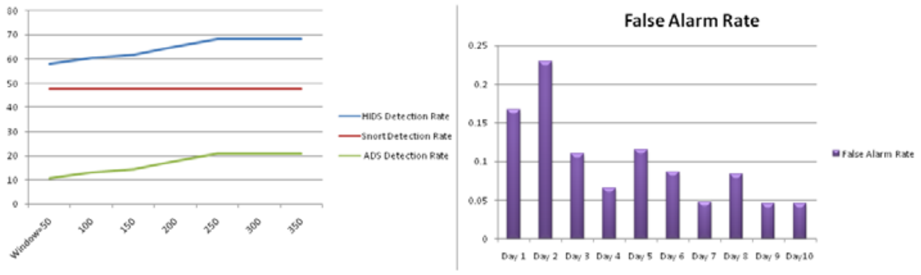


Fig. 5. a)Intrusion Detection Rate of HIDS b)False Alarm Rate per day of testing dataset

for Snort becomes 47.55%. Snort could detect 1667 attacks with only ADS Rule but out of it 1053 were false positives. Thus 614 were correctly detected. Thus detection Rate for Snort with only ADS rules =20.75%. It implies the Total Detection rate = 68.30%. Thus the new hybrid Intrusion Detection System has achieved a detection rate of 68.30% which is better than that of Snort which is 47.55%. The resultant system just had a false alarm of about 0.08%. Note that number of attacks in MIT/LL dataset are much lower than number of normal connections. This affects false alarm rate. The system had false positives of an average of 100 per day. From the figure 5.a, it is clear that the detection rate is almost steady after window =250. The false alarm rate (denoted by η) measures the percentage of false positives among all normal traffic events. A formal definition is given by

$$\eta = \frac{p}{k} * 100 \tag{5}$$

where p is the total number of false positive alarms and k accounts for the total number of connection events.

6 Conclusion

The hybrid intrusion detection system proposed here combines the advantages of low false-positive rate of misuse-based intrusion detection systems and the ability of an anomaly detection system to detect novel unknown attacks. By mining anomalous traffic episodes from Internet connections, an anomaly detection system is built that detects anomalies not detected by misuse-based Snort. Frequent episode rule concept can successfully be applied for detecting the anomalies in the computer network and hence for performing Intrusion detection in the network. The performance of detection rate was increased from 47.55% to 68.30% with the combined rules from Snort and ADS. This increase is just at a cost of average 0.08% false alarm rate.

The ADS is efficient for an offline evaluation since the computational resource requirement is higher. This kind of system can be deployed at network layer along with Snort to locally generate the rules specific to the traffic of the deployed network.

References

1. Qin, M., Hwang, K.: Anomaly Intrusion Detection by Internet Data mining of Traffic Episodes. ACM Transactions on Information and System Security (2004)
2. Yang, J., Chen, X., Xiang, X., Wan, J.: HIDS-DT: An Effective Hybrid Intrusion Detection System Based on Decision Tree. In: International Conference on Communications and Mobile Computing (2010)
3. Barbara, D., Couto, J., Jajodia, S., Popyack, L., Wu, N.: ADAM: Detecting Intrusions by Data Mining. Proceedings of the IEEE (2001)
4. Ertoz, L., et al.: The MINDS-Minnesota Intrusion Detection System. In: Next Generation Data Mining. MIT Press (2004)
5. Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A., Srivastava, J.: A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. In: Proc. Third SIAM Conference Data Mining (2003)
6. Lee, W., et al.: A Framework for Constructing Features and Models for Intrusion Detection Systems. ACM Transactions on Information and System Security (2000)
7. Lee, T.-Y., et al.: Mining Serial Episode Rules with Time Lags over Multiple Data Streams. Springer, Heidelberg (2008)
8. Snort 2.1 Intrusion Detection, 2nd edn. Syngress Publication
9. Roesch, M.: SNORT-Lightweight Intrusion Detection for Networks. In: Proc. USENIX 13th Systems Administration Conf., LISA 1999 (1999)
10. Paxson, V.: Bro: A System for Detecting Network Intrusions in Real Time. In: Proc. Seventh USENIX Security Symposium (January 1998)
11. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation. In: Debar, H., Mé, L., Wu, S.F. (eds.) RAID 2000. LNCS, vol. 1907, pp. 162–182. Springer, Heidelberg (2000)
12. Mahoney, M.V., Chan, P.K.: An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 220–237. Springer, Heidelberg (2003)
13. McHugh, J.: Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Off-line Intrusion Detection System Evaluation as Performed by Lincoln Laboratory. ACM Transactions on Information and System Security (November 2000)
14. Mannila, H., Toivonen, H.: Discovering Generalized Episodes Using Minimal Occurrences. In: Proc. Second International Conference on Knowledge Discovery and Data Mining (August 1996)