IDS False Alarm Reduction Using Continuous and Discontinuous Patterns

Abdulrahman Alharby and Hideki Imai

Institute of industrial Science, The university of Tokyo
4-6-1, Komaba, Meguro-ku, Tokyo, 153-8505 Japan
alharby@imailab.iis.u-tokyo.ac.jp, imai@iis.u-tokyo.ac.jp

Abstract. Intrusion Detection Systems *(IDSs)* are widely deployed in computer networks to stand against a wide variety of attacks. IDSs deployment raises a serious problem, namely managing of a large number of triggered alerts. This problem becomes worse by the fact that some commercial IDSs may generate thousands of alerts per day. Identifying the real alarms from the huge volume of alarms is a frustrating task for security officers. Thus, reducing false alarms is a critical issue in IDSs efficiency and usability. In this paper, we mine historical alarms to learn how future alarms can be handled more efficiently. First, an approach is proposed for characterizing the "normal" stream of alarms. In addition, an algorithm for detecting anomalies by using continuous and discontinuous sequential patterns is developed, and used in preliminary experiments with real-world data to show that the presented model can handle IDSs alarms efficiently.

Keywords: Intrusion detection, alarm reduction, sequential patterns.

1 Introduction

Over the past decade, the number as well as the severity of computer attacks has significantly increased. CSO magazine conducted a survey on the 2004 cyber crimes, the survey shows a significant increase in reported electronic crimes. Compared to the previous year, more than 40% of intrusions and electronic crimes are reported. Also, 70% of the respondents reported at least one electronic crime or intrusion was committed against their organization [1]. According to collected statistics, electronic crimes have an incredible impact on economy. Reports say that electronic crimes have cost more that $600 million in 2003.

In response, security services strongly recommend to deploy and implement suitable protection technology. Besides the first defence protections (e.g. firewalls, authentication, and cryptography), IDSs are recommended for attacks detection and to alert security officers for further actions. IDS has become one of the corner stones in computer security because of its triggered alarms to intrusive activities can greatly reduce the possible harm and data leakage due to attacks.

Although IDSs have been deployed widely across data networks during the last decade, and their value as security components have been demonstrated.

J. Ioannidis, A. Keromytis, and M.Yung (Eds.): ACNS 2005, LNCS 3531, pp. 192–205, 2005.

Most of them suffer from high false alarms rate. In fact, during their normal operation they generate thousands of fake alarms per day.

In this paper, we report our filtration technique to reduce fake alarms rate. In the remaining of this introduction, we describe intrusion detection methods, fake alarms problem, and the basic requirements of filtration algorithm. Section 2 shows our proposed model. Section 3 shows the experiments. In section 4 we discuss the experiments results. Our method is contrasted with existing methods in section 5. Finally, in section 6 we conclude the paper.

1.1 Detection Methodologies

IDSs are considered as powerful security tools in computer systems environments. These systems collect activities within the protected network and analyze them in order to detect intrusions. System activities are usually collected from two main sources, network packet streams and host log files. Once the information is collected, the detection algorithm starts looking for any evidence for intrusions existence.

There are two general methodologies of detection: *misuse* and *anomaly* detection [2, 3]. *Misuse* detection systems such as STAT [4] look for a known malicious behavior or signature, once it is detected an alarm is raised for further actions. While this type is useful for detecting known attacks, it can't detect novel attacks, and its signatures database needs to be upgraded frequently. The main feature of this model is its low false alarm rate. *Anomaly* detection models (e.g. IDES [5]) compare reference model of normal behavior with the suspicious activities and flag deviations as anomalous and potentially intrusive. Unlike *misuse* detection, *anomaly* detection systems identify unknown intrusions. The two detection approaches can be combined to detect attacks more efficiently. The most apparent drawback of these systems is the high rate of false alarms.

1.2 IDS False Positive Alarms Problem

A false positive alarm in IDSs is an attack alarm that is raised incorrectly. Suppose an IDS detects someone attempting to run *Nmap* and it turns out that the traffic pattern that caused the alarm reveals someone was running a peer-to-peer application: that is a false positive [8]. But if that scan is against a nonexistent machine, this alarm is not a false positive, someone actually was attempting malicious activities.

An ideal detection system is the one that has 0% of miss-classified normal behavior with 100% attack detection. However, current detection systems suffer from high false alarms and low attack detection rate [6]. IDSs can be fine-tuned to reduce false alarm generation, but this may degrade the security level. Usually, with *anomaly-based IDSs* the abnormality is determined by measuring the distance between the suspicious activities and the norm, and based on a chosen threshold the observed behaviour is classified. Increasing this threshold leads directly to induce more false alarms, while many of them are actually not true. In case of *misuse-based IDSs* the detection depends on a set of rules or signatures

to detect intrusive behaviour, the tighter the rule set, the higher the false alarms rate. On the other hand, the higher threshold in the case of anomaly detection and the tighter the rules set in the case of misuse detection, the stronger the security can be achieved. Reducing anomaly detection threshold and relaxing some misuse detection rules can reduce the number of generated false alarms, but such action is most risky, causing IDS to be unable to detect major attacks. Therefore, the tuning problem actually is a trade-off between reducing false alarms and maintaining system security.

1.3 An Overview and Basic Requirements

In this section, we propose an overview of our reduction method for IDSs false positive alarms that significantly reduces operators' workload. When the network is under attack, we believe that IDSs behave in a different way from the free-attack situation. This fact makes it clear that we can distinguish between the alarms sequences generated when the system is under attack from those sequences generated in a normal situation. In other words, the basic idea of our approach is simple: Frequent behaviour, over an extended period of time, is likely to be normal. A combination of specific alarm types occurring regularly in the same order, and within a certain time window, is most likely less suspicious than a sudden burst of alarms never seen before. We believe that any prospective technique for alarms reduction should satisfy the following requirements:

Scalability: Since IDSs can generate thousands of alarms per day, the reduction algorithm should be able to handle such huge amount of alarms.
Noisy data: Since IDSs trigger alarms subjected to be very noisy as detailed in [7, 8], the reduction model should be able to consider that property of IDSs alarms.
Alarm attributes: The model should support any type of attributes contained in the alarms, alarm attributes detailed by Julisch in [9].
Independency: Detection-Reduction processes should be independent, which allows the reduction algorithm to be applied to any type of IDSs.
Cost: To support automated reduction, the model should be efficient in terms of computation cost.

Below we will introduce our method to model normal alarms sequences and how to detect deviations from the norm.

2 An Approach

2.1 Methodology

Consider a finite set $U = \{X_1, X_2, ..., X_N\}$ of different sequences of a certain distinct alarm attribute. Each sequence X_i contains a different number of ordered alarms, the number of those alarms is called the length of the sequence. These sequences represent ordered alarms generated by IDS, furthermore, they

are defined as "normal" if they are generated under conditions not associated with intrusion attacks. We believe that every single element in the sequence has a value and contains important information. In addition to considering all possible subsequences, our intention is to make sure that each single alarm within the sequence is included in the normal model. In general, we have to expand each sequence to its basic components which is called sequential patterns. Formally, for any alarms sequence $X_i = \{x_1, x_2, ..., x_m\}$ of m elements within a time window of size W, a number n of sequential patterns can be extracted. In other words, sequential patterns: $X_i^{sq} = \{S_1, S_2, ..., S_n\}$, each S_i contains a number of ordered alarms. For any sequence X_i, the sequential patterns can be extracted as $X_i^{sq} = f(X_i)$ where $f(\cdot)$ functions as a sequential patterns generator. Normal sequential patterns are extracted from normal alarm sequences via $f(\cdot)$, which form the basis of a normal profile of the triggered alarms. Furthermore, $U^{sq} = \{X_1^{sq}, X_2^{sq}, ..., X_N^{sq}\}$ describes the normal alarms profile that can be used to classify the next coming sequence with a satisfactory threshold.

Sequential patterns data mining are popular and have been extensively studied because of their numerous applications, many algorithms were proposed [10–13]. In our experiments, and because of its ability to mine both types of patterns (continuous and discontinuous), we adopted the algorithm proposed by Chen to extract the sequential patterns, a brief description about the algorithm given in appendix A.

In the next sections more details are given about sequential patterns types, sequential patterns generation, and how a single sequence of alarms can be classified as normal or intrusive.

2.2 Alarm Sequences Classification

The alarms sequential pattern S_i can be classified into two classes, continuous and discontinuous. Precisely, suppose a sequential pattern S_i extracted from the alarms sequence $X_i = \{x_1, x_2, ..., x_m\}$ and contains a number of alarms, that is, $S_i = \{s_1, s_2, ..., s_l\}$. The pattern S_i is considered as a continuous pattern if all contained elements appear in consecutive positions of the sequence X_i, such that, there is an integer r such that; $s_1 = x_r,\ s_2 = x_{r+1}, ...,\ s_d = x_{r+l-1}$. For example, the continuous pattern (s_3, s_4) occurs in sequence: $X_1 = (s_1, s_2, s_3, s_4, s_5, s_6)$. The second category is the discontinuous patterns, we say that s_i is a discontinuous pattern if the contained elements don't appear in consecutive positions of the sequence X_i, that is, if there are existing integers $r_1 < r_2 < ... < r_l$ such that $s_1 = x_{r_1},\ s_2 = x_{r_2}, ...,\ s_l = x_{r_l}$. For example, the pattern (s_1, s_3) in sequence X_1 is a discontinuous pattern.

While continuous patterns reflect a clean alarm sequences, discontinuous patterns represent the sequences mixed with noisy data. Precisely, alarm streams can be mixed with undesirable data, thus, discontinuous patterns are very suitable to mine and then analyze noisy alarms. We believe that both of them - continuous and discontinuous - are generated by most of the IDSs. Any alarm sequences may consist of several continuous sub-sequences that are not adjacent. Ignoring such patterns leads to missing very meaningful patterns. For example, patterns:

(s_1, s_2) and (s_5, s_6) contained in sequence X_1, each one of them may have no meaning by itself, but if they are combined in one pattern like: $(s_1, s_2, *, s_5, s_6)$, they may represent a very useful meaningful pattern, where " $*$ " means a variable number of intermediate elements. In the next section, for a certain alarms sequence, we show how the two different types of sequential patterns can be extracted.

2.3 Extraction of Alarms Sequential Patterns

Let $X_i = \{x_1, x_2, ..., x_m\}$ denote all alarms in one sequence of length m, also, let the star " $*$ " denote a subsequence of any length of consecutive alarms $\{x_l, x_{l+1}, ...\}$, including zero length, which may be contained in X_i for $2 \leq l \leq m-1$. The sequence X_i can be expanded to a number of patterns called sequential patterns. In addition to the items themselves, sequential patterns are generated by extracting all possible forward combinations. Basically, forward combinations are sequences of items representing continuous and discontinuous patterns that may be contained in the sequence.

An alarm sequence of length m can be expanded to a number n of sequential patterns, including a number n_c as continuous, and n_d as discontinuous patterns. Two different methods are used to calculate n_d depending on the value of m whether it is an even or an odd value while using the same formula to calculate n_c. Table 1 shows the different formulas that were used to calculate the number of extracted sequential patterns from a certain sequence of length m.

Table 1. Different used formulas to calculate the number of sequential patterns $n = n_c + n_d$ that can be extracted from a sequence of length m.

Sequence length, m	Continuous (n_c)	Discontinuous (n_d)
Even	$0.5(m^2 + m)$	$1 + 2\sum_{i=1}^{0.5m-1}(m-1)i - i^2$
Odd	$0.5(m^2 + m)$	$1 + 2(m-2) + 0.25(m-1)^2 + 2\sum_{i=2}^{0.5(m-3)}(m-1)i - i^2$

Any extracted combination $S_i = \{s_1, s_2, ..., s_l\}$ from X_i can be considered as a sequential pattern if the following constraints are satisfied:

- $s_i \in X_i$
- $s_1, s_l \in X_i$
- If $s_i =$ " $*$ " then $s_{i+1}, s_{i-1} \in X_i$, for $1 < i < l$

By definition, the sequential pattern never starts or ends by " $*$ ". For example, if we have a sequence $X_i = ABCD$, we may have these continuous patterns AB, BCD, and ABC, or this discontinuous pattern $A*CD$. Because of the definition of the "$*$", the discontinuous one implicitly has two other patterns: ACD,

Table 2. Example of two sequences and their related Continuous & Discontinuous sequential patterns.

Sequence	Continuous	Discontinuous
ABC	A, B, C, AB, BC, ABC	A∗C
ABCD	A,B,C,D,AB,BC,CD,ABC,BCD,ABCD	A∗C,B∗D,A∗D,A∗CD,AB∗D

and *ABCD*. Table 2 shows an example of extracted continuous and discontinuous sequential patterns of two sequences. In the next section, we will show how deviation from the norm can be detected.

2.4 Alarms Deviation Detection

To classify a newly arrived sequence, corresponding sequential patterns are extracted in the same way described above. Then the similarity between sequential patterns of the new sequence and those of the normal sequences is calculated using the similarity algorithm. Each single extracted sequential pattern that is represented in the normal sequential patterns set is given a weight $w = 1/n$, where n is the total number of all extracted sequential patterns of that specific testing alarm sequence, and it can be obtained programmatically, or by using formulas given in table 1. The value of w falls in the range $0 \leq w \leq 1$. By calculating the total summation weights *sum* of all matches, strength of the deviation signal can be determined. If the total weights summation exceeds a certain threshold, the testing sequence is classified as normal. Otherwise, it is an anomalous sequence. An abstract of the pseudo code of the used similarity algorithm is given below:

```
1 Extract normal alarms sequential patterns;
2 for each new suspicious sequence X Do
3   extract all corresponding sequential patterns;
4   get n value; // number of extracted patterns
5   calculate sum; // all matched patterns
6       if sum/n ≥ threshold then
7         X is normal;
8       else then
9         X is abnormal;
```

3 Experiments

Experiments with DARPA 1999 data set have been conducted to evaluate the proposed approach. DARPA off-line data sets [14] were developed to evaluate any proposed techniques for intrusion detection. These data sets contain contents of every packet transmitted between hosts inside and outside a simulated military base. There were a collection of data including *tcpdump* and *BSM* audit data.

In our experiments we use the *tcpdump* data. There are three weeks of training data, week 1 and week 3 are free of attacks whereas week 2 is labelled with attacks.

3.1 The Detection System

In our experiments we examined the network traffic with a well-known IDS named *Snort* [15, 16]. *Snort* is a signature-based lightweight network intrusion detection system, it is available as an open source code on the internet. It can perform both protocol analysis, and content matching. *Snort* is capable of performing real-time traffic analysis and detect a variety of attacks. It looks for attacks signs by comparing the incoming packets with its own set of attacks signatures. Each detected attack is recorded in a database with relevant information called alarm attribute [9], such as: *alarm type*, *timestamp*, *source IP address*, *destination IP address*, and other information. In this work, we examine only the *alarm type*, the rest of the alarm attributes are planned to be our future work. *Snort* is capable of processing different types of protocols such as *TCP, UDP, ICMP, ARP, IPv6*, and others.

Our experiments were performed with *Snort 2.3.0RC2*. We processed the three weeks of DARPA 1999 training data set. The total generated alarms by *Snort* with its default setting were 97257 alarms with more than 6400 alarms per day on average. There were 76 different types of attacks. Experiments show that among the alarms, there are a limited number of alarms that have the most contribution. Table 3 shows the most frequent generated alarm types with the percentage of their occurrence. In the next section, we will show how we constructed our normal data set.

3.2 Normal Alarm Model

Since the first and third weeks of DARPA 1999 data set are free-attacks traffic, any triggered alarms of these two weeks are false positive alarms. After running *Snort*, we found 78972 alarms triggered within these two weeks which are considered as false alarms. For a certain time period, *Snort* generates hundreds of false alarms sequences. Obviously, a shorter time window would increase the total number of alarm sequences. Once alarm sequences are collected, the sequential patterns are extracted as described in the previous section.

Table 3. The most frequent triggered alarm types and their percentage of occurrence.

Alarm type	Occurrence
SNMP public access udp	72 %
ICMP PING NMAP	9.3 %
FTP CWD	7.2 %
TELNET access	2.3 %
SCAN FIN	1.3 %

The extracted sequential patterns of the first and third weeks false alarms represent the behaviour of the *Snort* as an IDS system in the free-attack environment, and we called them "normal" sequential patterns data. If a new sequential patterns data set is similar to this normal data set, it is considered as normal. Otherwise, abnormality appears and further investigations are needed. The experiment setup is shown in Figure 1. The next section shows the results of our experiments.

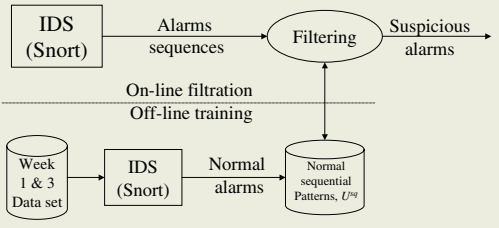

Fig. 1. Experiment setup shows the relation of the IDS and the proposed filtration process.

3.3 Experimental Results

This section assesses the proposed approach with respect to its ability to reduce the false alarms. To judge the effectiveness of applying continuous and discontinuous sequential patterns for alarms reduction, we ran two sets of experiments. In the first set, we fixed the parameter *threshold* to be 0.7, also, we fixed the sequence length m to be 15, and let the reduction algorithm run over all 15 days (i.e the three weeks). Figure 2 shows the alarms load reduction attained

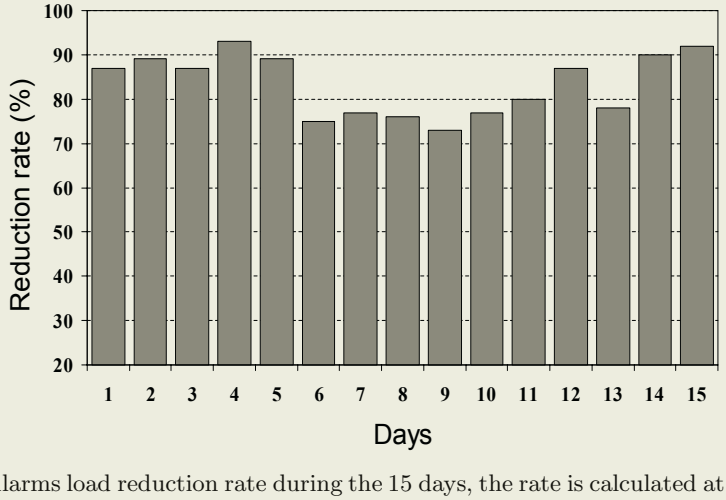

Fig. 2. Alarms load reduction rate during the 15 days, the rate is calculated at *threshold* 0.7 and sequence length m= 15.

in weeks 1, 2, and 3. Alarm reduction rate calculated as the ratio of triggered alarms with the filter to those triggered without the filter. Clearly, from the figure, approximately 80% of the alarms were discarded.

In the second set of the experiments, we ran the algorithm for the second week data set, and let m and the *threshold* vary. Figure 3 shows the alarms load reduction achieved for different sequence m length and *threshold*, the Results presented in the figure are encouraging. We found that the proposed approach is able to reduce the alarms up to 93% without filtering out any true alarms detected by the IDS. As the *threshold* goes up the reduction rates decrease because more sequential patterns are classified as abnormal and the alarm sequences corresponding to these sequential patterns haven't been filtered. For *threshold*=0.3 and during week 2, about 90% of false alarms are filtered when sequence length m is fixed at 20. The false alarms reduction rate drops to 62% when the *threshold* jumps to 0.85. Clearly, reducing the *threshold* leads to higher reduction rate, but to go further in decreasing the threshold leads to missing out some true alarms. The next section presents more discussion about the effects of the parameters m and *threshold* on the alarms load reduction rate.

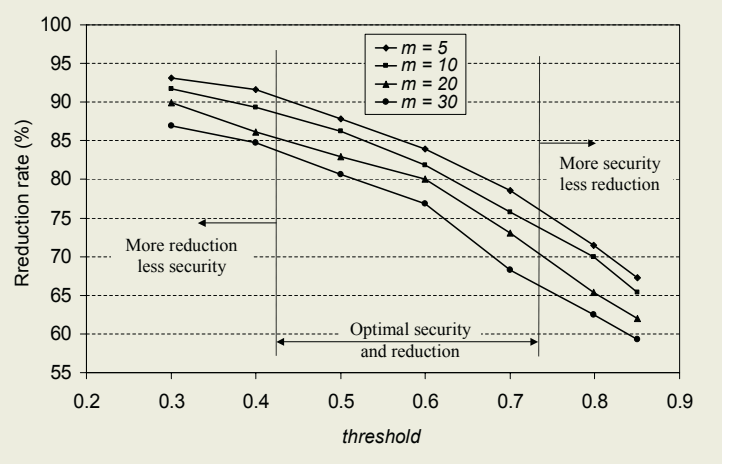

Fig. 3. False alarm reduction rate for different *threshold* and length m. The rate is calculated as the result of the division of the total number of triggered alarms during *week 2* with the filter and the total number of triggered alarms without the filter during the same period.

4 Discussions

For the proposed model to be efficient it must meet some basic operational requirements. In this section, we discuss some factors that we believe have apparent effects on the model efficiency, and form its main characteristics.

4.1 Critical Parameters

To achieve the most optimal parameters setting many preliminary experiments are performed. The three parameters values that have been tested are time

window W, alarm sequence length m, and similarity *threshold*. During the experiments, we noticed that for a certain time window we have different lengths of alarm sequences. Since the sequence length has a direct effect on the number of extracted patterns, consequently, the performance of the filter, we fixed the sequence length value for each experiment regardless of the time window. Different m values have been tested, from 5 to 30. For each value of m, we worked out the alarms reduction rates with different *thresholds*. The larger the value of m, the more sequential patterns are extracted and the higher the similarity between the tested sequential patterns set and the normal ones.

The second tested parameter (i.e. the *threshold*) plays an important role when judging whether a sequence of alarms is abnormal. When classifying new sequential patterns data set (new sequence), the average of the total matching weights summation is taken as the similarity score of the sequential patterns set. If the score is larger than the *threshold*, the new sequence is classified as normal. Obviously, high *threshold* values would lead to higher similarity, consequently, to higher normality, and more security would be achieved, but the classifier will suffer from high false alarms rate. On the other hand, lower *threshold* leads to lower security with lower false alarms rate. Since alarms reduction techniques are recommended only if the security level is maintained, we don't prefer very low *threshold*. However, very low *threshold* leads to misclassifying true alarms and it is not desirable and not accepted. Figure 3 shows three regions of possible operation based on *threshold* value. The optimal operation limits can be decided based on the security policy, which is a trade-off between security and reduction rate.

4.2 Model Efficiency

In section 1.3 we have listed some requirements that alarms reduction model should satisfy. In this section, we discuss how well the proposed model meets these requirements:

`Scalability`: It is something to think about whether you have processed one alarm or one million alarms. Our approach shows good performance in terms of scalability. For instance, our implementation of the model can process more than one million alarms and *execute* the similarity algorithm in less than 3 minutes when it is run on 1.5 GHz Pentium 4 CPU with 256MB RAM. Moreover, our approach is scalable in another sense. Since network traffics are very dynamic and changeable, and subsequently normal alarm streams, new extracted normal patterns can be added to the normal pattern data set without any need to reconstruct the whole data set.

`Noisy data`: Typical IDSs can generate very noisy alarms, where undesirable data are mixed with real alarms. Our approach can handle these noisy data efficiently by exploiting discontinuous sequential patterns to represent noisy alarm sequences.

`Alarm attributes`: In general, commercial IDSs alarms contain many attributes such as time, numerical, categorical, and free-text. The most efficient model has to support any type of alarm attributes. Since sequential patterns

extraction is never affected by the values of different attributes, the proposed approach is able to handle any attribute regardless of its type. In our current implementation we use *alarm type* (free-text) for alarms reduction.

`Independency`: Our proposed alarms reduction processes are totaly independent from the detection function. Therefore our model can be applied to most commercially-available IDSs without changing the existing security policy or system configuration. Since the normal model is constructed from alarms raised by IDSs under no attacks, and then used by the reduction algorithm in which incoming alarm streams are filtered, we believe that the entire reduction process can be seen as a plug-in to IDSs. Figure 1 shows the relationship of detection-reduction processes and how they are independent.

`Cost`: One of the critical issues in the reduction algorithm development is the cost. For extracting normal patterns of a number k of normal sequences of length m, we found that the suggested generator is an $O(km)$ cost. Since the normal model is built in off-line state this cost has no effect on the efficiency of the model. Similarity algorithm has a cost of an $O(n)$ where n is the total extracted patterns from the tested sequence.

Now, we will contrast the existing related alarms reduction methods with ours.

5 Related Works

There has been extensive considerable work in representing and reducing IDSs alarms. Here we only mention those efforts that contribute to the current proposal. In [17] Clifton and Gengo used episode data mining techniques to construct filtering rules. This work was extended by Julisch from IBM in [9], he introduced a clustering technique and used it in the design. In addition to discussing the main properties, Julisch observed that about 90% of false alarms are caused by a small number of root causes. Also related is the work by Manganaris, he applied association rules for false alarms reduction [18], he suggested an alarm filtration framework to handle IBM's network operation center based on extracted association rules. His approach classified incoming alarms based on frequent alarm set with minimum occurrence in bursts of alarms, non-frequent alarm sets flagged as suspicious. Our work, by contrast, uses different data mining techniques, it aspires to use continuous and discontinuous data mining for alarms filtering.

The other related work is done by Barbara *et al.* in [19] who enhanced anomalous patterns detection in computer networks by applying incremental data mining techniques. In [20] Lee *et al.* used data mining to construct the necessary features that help to detect the intrusions. In general, the last two works have overcome some limitations of existing detection systems by constructing more systematic models. By contrast, our approach is independent. It works with any IDSs and can handle any triggered alarms.

The works in [21, 22] focus on alarms grouping where each group contains the alarms of the same phenomenon. In that way, those works focus on the users features that have apparent effects on security issues raised by IDSs. In our work,

we advocate exploratory data mining that needs no prior knowledge about the users where users privacy can be partially maintained. In general, we employ sequential patterns data mining to handle IDSs alarms in a more efficient way.

6 Conclusion

We have presented a new approach for dealing with IDS false alarms. Some systems use pattern matching for *misuse* detection while others use *anomaly* detection. Both approaches have their advantages and disadvantages. In this paper, we have attempted to combine these approaches, by performing anomaly detection on the voluminous results produced by a misuse detection system "*Snort*". We have found our preliminary results with real-world data set encouraging: sequential patterns classification can be used to support investigation of a large number of alarms, the proposed approach employs historical alarms to build the norm and, more importantly, the viability of the proposed approach is demonstrated.

Our analysis of DARPA alarm traffic also produced interesting results. It showed that sequential patterns technique is suitable for false alarms reduction. Obviously, the basic value of using this technique is its ability to break the alarm sequence to its basic patterns. Moreover, the results have shown that alarm streams are very homogeneous, and contain very repetitive hidden useful patterns.

We have also investigated the suitability of using attribute *alarm type* for our purposes. It turned to work well in our framework. Our future work will focus on the validity of using other alarm attributes.

References

1. The 2004 E-Crime Watch survey, available at: http://www.csoonline.com/releases/ecrimewatch04.pdf
2. Kumar, S., Spafford E. H.: A Software Architecture to Support Misuse Intrusion Detection. In Proceedings of the 18th National Information Security Conference. (1995) 194-204
3. Forrest, S., Hofmeyr, S. A., Somayaji, A., Logstaff, T. A.: A Sense of Self for Unix process, Proceedings of 1996 IEEE Symposium on Computer Security and Privacy. (1996) 120-128
4. Ilgun, K., Kemmerer, R. A., Porras, P. A.: State Transition Analysis: A Rule-Based Intrusion Detection System. IEEE Transactions on Software Engineering, 21(3). (1995) 181-199
5. Javitz, H. S., Valdes, A.: The SRI IDES Statistical Anomaly Detector. In IEEE Symposium on Security and Privacy, Oakland, CA. SRI International. (May 1991)
6. Yihua, L., Vemuri, V. R.: Use of K-Nearest Neighbor classifier for intrusion detection. Computers & Security 21(5). (2002) 439-448
7. Bellovin, S. M.: Packets Found on an Internet. Computer Communications Review, 23(3). (1993) 26-31
8. Paxson, V., Bro,: A System for Detecting Network Intruders in Real-Time. Computer Networks, 31(23-24). (1999) 2435-2463

9. Julisch, K.: Mining Alarm Clusters to Improve Alarm Handling Effciency. In 17th Annual Computer Security Applications Conference (ACSAC). (December 2001) 12-21
10. Yen-Liang, C., Shih-Sheng, C., Ping-Yu H.: Mining hybrid sequential patterns and sequential rules. Information Systems V 27, 5. (2002) 345-362
11. Agrawal, R., Srikant, R.: Mining sequential patterns. Proceedings of the 7th International Conference on Data Engineering, Taipei, Taiwan, IEEE Computer Society. (1995) 3-14
12. Chen, M.S., Park, J.S., Yu, P.S.: Efficient data mining for path traversal patterns, IEEE Trans Knowledge Data Eng, 10(2). (1998) 209-221
13. Han, J., Pei, j., Yin, Y.: Mining frequent patterns without candidate. in proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Dallas, Texas, ACM Press, New York. (2000) 1-12
14. DARPA Dataset: http://www.ll.mit.edu/IST/ideval/
15. *Snort*: http://www.snort.org/
16. Roesch, M.: Snort – lightweight intrusion detection system for networks. In: Proceedings of USENIX LISA'99. (1999)
17. Clifton, C., Gengo, G.: Developing Custom Intrusion Detection Filters Using Data Mining. In Military Communications Int'l Symposium (MILCOM2000). (October 2000)
18. Manganaris, S., Christensen, M., Zerkle, D., Hermiz, K.: A Data Mining Analysis of RTID Alarms. Computer Networks, 34(4). (October 2000) 571-577
19. Barbara, D., Couto, J., Jajodia, S., Popyack, L., Wu, N.: ADAM: Detecting Intrusions by Data Mining. In IEEE Workshop on Information Assurance and Security. (2001)
20. Lee, W., Stolfo, S. J.: A Framework for Constructing Features and Models for Intrusion Detection Systems. ACM Transactions on Information and System Security, 3(4). (2000) 227-261
21. Valdes, A., Skinner, K.: Probabilistic Alert Correlation. In 4th Workshop on Recent Advances in Intrusion Detection (RAID), LNCS, Springer Verlag. (2001) 54-68
22. Staniford, S., Hoagland, J. A., McAlerney, J. M.: Practical Automated Detection of Stealthy Portscans. In ACM Computer and Communications Security IDS Workshop. (2000) 1-7

A GFP1

Yen-Liang et al. in [10] formulated the definition of sequential patterns, and proposed an algorithm GFP1 to mine hybrid (continuous and discontinuous) patterns.

In this work, we adopted GFP1 to discover and extract all continuous and discontinuous sequential patterns. In the following a brief description of this algorithm will be given, the details can be found in Yen-Liang paper.

Algorithm GFP1

1 Generate $C_{1,0}$
2 Determine $L_{1,0}$ by scanning the database

3 For $(k = 2; L_{k-1,k-2} \neq 0; k++)$ do
4 $C_{k,k-1} = GetJoin(L_{k-1,k-2})$
5 Determine $L_{k,k-1}$ by scanning the database
6 For $(j = 2; j = k; j++)$ do
7 $C_{k,k-1} = GetNextCandidate(L_{k,k-j+1})$
8 Determine $L_{k,k-j}$ by scanning the database

In the above algorithm, some steps need to be further explained. Each $C_{i,j}$ contains all candidate and possible sequential patterns, and each $L_{i,j}$ contains all frequent sequential patterns that exceed some support value, where i is the number of elements and j is the number of stars within each pattern. In the first phase, we need to find $C_{1,0}$, i.e, all candidate patterns with only one known element and zero " $*$ ".

GFP1 algorithm includes two functions *GetJoin* and *GetNextCandidate*.

GetJoin function is used to generate the candidate patterns $C_{k,k-1}$ of length k from the frequent pattern set $L_{k-1,k-2}$. By joining the patterns in $L_{k-1,k-2}$, we can find a set of candidate patterns which contains all frequent patterns in $L_{k,k-1}$ with a *support* value exceeding a certain threshold, *Support* parameter value required for GFP1, it describes the minimum frequent number required for a single sequential pattern to be considered as a useful pattern and then mined.

Function *GetNextCandidate* is used to generate $C_{k,k-j}$ from the frequent pattern set $L_{k,k-j+1}$. The candidate $C_{k,k-j}$ can be found by removing the star from all the frequent patterns in $L_{k,k-j+1}$.