Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A decision support system for constructing an alert classification model

Nien-Yi Jan [a,c], Shun-Chieh Lin [b,*], Shian-Shyong Tseng [b], Nancy P. Lin [a]

[a] Dept. of Computer Science and Information Engineering, Tamkang University, Taiwan, ROC
[b] Dept. of Computer Science, National Chiao Tung University, Taiwan, ROC
[c] Business & Marketing Strategy Research Department, Telecommunication Lab., Chunghwa Telecom Co., Ltd., Taiwan, ROC

ARTICLE INFO

Keywords:
Decision support system
Alert classification
Sequential pattern mining
Intrusion detection
Model construction

ABSTRACT

As the rapid growth of network attacking tools, patterns of network intrusion events change gradually. Although many researches had been proposed to analyze network intrusion behaviors in accordance with low-level network data, they still suffer a large mount of false alerts and result in difficulties for network administrators to discover useful information from these alerts. To reduce the load of administrators, by collecting and analyzing unknown attack sequences systematically, administrators can do the duty of fixing the root causes. Due to the different characteristics of each intrusion, none of analysis methods can correlate IDS alerts precisely and discover all kinds of real intrusion patterns. Therefore, an alert-based decision support system is proposed in this paper to construct an alert classification model for on-line network behavior monitoring. The architecture of decision support system consists of three phases: Alert Preprocessing Phase, Model Constructing Phase and Rule Refining Phase. The Alert Processing Phase is used to transform IDS alerts into alert transactions with specific data format as alert subsequences, where an alert sequence is a kind of well-aggregated alert transaction format to discover intrusion behaviors. Besides, the Model Constructing Phase is used to construct three kinds of rule classes: normal rule classes, intrusion rule classes and suspicious rule classes, to filter false alert patterns and analyze each existing or unknown alert patterns; each rule class represents a set of classification rules. Normal rule class, a set of false alert classification rules, can be trained by using sequential pattern mining approach in an attack-free environment. Intrusion rule classes, a set of known intrusion classification rules, and suspicious rule classes, a set of novel intrusion classification rules, can be trained in a simulated attacking environment using several well-known rootkits and labeling by experts. Finally, the Rule Refining Phase is used to change the classification flags of alert sequence across different time intervals. According to the urgent situations of different levels, Network administrators can do event protecting or vulnerability repairing, even or cause tracing of attacks. Therefore, the decision support system can prevent attacks effectively, find novel attack patterns exactly and reduce the load of administrators efficiently.

1. Introduction

With the rapid development of network technology, the network security becomes one of the most important issues today, especially for distributed denial of service (DDoS) attacks. The DDoS attack is a simple but ferocious attack since it could be launched by someone who has DDoS attacking tools. In recent years, many e-commerce enterprises had been attacked by DDoS attacks, such as Yahoo, eBay, Amazon, Key Internet Computers and so on, which causes great damage on these enterprises (Bridis, 2002; CERT/CC, 2006; Xu & Lee, 2003). Various approaches and monitoring policies had then been developed to detect and filter the malicious traffic of DDoS attacks. Unfortunately, it is very difficult to keep up with the rapidly growing expertise or knowledge on these domains due to the lack of the common terminology. Therefore, an ontology, which could be acquired from domain experts, is needed for providing a sharing vocabulary to integrate and categorize the rapid growth of knowledge of DDoS intrusion tolerance (Lee, Kim, Kwon, Han, & Kim, 2008; Lin & Tseng, 2004).

In order to detect and prevent anomaly network behaviors, many intrusion detection systems (IDS) or Firewalls have been developed to focus on well-known intrusion patterns through packet-based information, connection-base information, or some statistical network information. Although these kinds of approaches can be useful to defend the obvious activity patterns of intrusions, many intrusions are still hard to be detected by IDSs to notice human experts because of numerous noises and insufficient information among different intrusions. In other words, existing IDS tools just tag suspicious network behaviors into IDS alerts, but can not avoid generating false alerts. For domain experts, it is time-consuming to classify IDS alerts into true or false alert sequences precisely due to numerous noisy IDS alerts; for junior administrators, it is difficult to generate aggregated IDS alert

* Corresponding author. Tel.: +886 3 5731966; fax: +886 3 5721490.
E-mail address: jielin@cis.nctu.edu.tw (S.-C. Lin).

0957-4174/$ - see front matter Crown Copyright © 2009 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2009.02.097

sequences according to the similarities of intrusion patterns due to the lack of domain knowledge.

Although several methods such as genetic algorithm, neural networks, and data mining approaches, have been used to discover either unknown or useful patterns for experts, lots of hidden and concealed intrusion patterns may still be escaped because of insufficient and dirty information. None of them discusses on discovering intrusion patterns thoroughly from IDS information data, called IDS alerts. Therefore, we are concerned with how to design a systematic framework to assist administrators discovering intrusion patterns with IDS alerts. Our idea is to construct a decision support system to help experts construct an alert classification model for on-line intrusion detection of IDS alerts. For domain experts, the built alert classification model will reduce experts' efforts on how to precisely and quickly fix the root causes; for junior administrators, alert classification model will help them reuse the embedded domain expertise as references, and the domain knowledge will be no longer a limitation for intrusion detection.

As we know, too many false alerts result from IDS tools in present network environments because IDS tools should be designed as powerful as possible not to miss any detection of real intrusions. This means IDS tools become more and more sensitive to generate false alerts which are noise to discover real intrusion patterns in some network environments. In this paper, we collect alert transactions in an attack-free environment, which is a virtual intranet with several hosts in laboratories to simulate real Internet behaviors in the training stage. Accordingly, all alerts are treated as false alerts in this virtual intranet, and the frequent patterns to be mined in this case are treated as patterns of normal behaviors, or called patterns of false alert. In other words, we can use these data to construct normal behavior patterns to remove false alerts. Besides, using some specific rootkits to simulate the real intrusions of networks, we can also construct known intrusion patterns in the training stage to classify existing intrusion patterns.

The proposed decision support system consists of three phases in the training stage: Alert Preprocessing Phase, Model Constructing Phase and Rule Refining Phase. In Alert Preprocessing Phase, the raw IDS alert transactions are stored in alert warehouse and transformed into alert subsequences in each time interval to reduce false alerts and increase the possibility of discovering frequent sequential patterns, where alert subsequences can be obtained according to three sequence partitioning policies. In Model Constructing Phase, filtering and analysis methods are proposed to assist administrators construct three kinds of rule classes (normal rule class, intrusion rule class and suspicious rule class) to remove false alert patterns and analyze each existing or unknown alert pattern, where each rule class represents a set of classification rules. The normal rule class of false alert patterns can be constructed by a sequential pattern mining algorithm in an attack-free environment. A simulated attack environment is also established in this paper to train suspicious/intrusion alert sequences, where the system interacts with administrators to assist them in flagging appropriate classifications of different alert sequences into intrusion rule class or suspicious rule class. These kinds of classification rules are used to help administrators identify the label of alerts generated by IDS sensors on-line to reduce the effort of administrators. After interviewing with experts, the normal behavior patterns are used to remove most part of false alerts and intrusion behavior patterns are then used to identify the true intrusions. Finally, the suspicious behavior patterns could be used to help administrators discover unknown suspicious behavior patterns from remaining alerts. In Rule Refining Phase, because flags of alert patterns may change between two consecutive time intervals, so the differences of specific patterns must be highlighted and refreshed again to experts in each time interval. A least recently used*(LRU)* rule replacement policy is used to replace the rules which are less used recently in each classification rule class to ensure the performance of our system.

Our experiment demonstrates that the accuracy of our decision support system is well. This decision support system will construct a classification model for on-line monitoring. The alert classification model is useful for experts to discover suspicious or intrusion patterns quickly and precisely, and lightens the load of on-line alert analysis for experts obviously.

2. Preliminaries

2.1. Traditional analysis approaches for network intrusion

Intrusion detection includes identifying a set of malicious actions that compromise the integrity, confidentiality, and availability of information resources. Traditional methods for intrusion detection are based on extensive knowledge of signatures of known attacks, where monitored events are matched against the signatures to detect intrusions. These methods extract features from various audit streams, and detect intrusions by comparing the feature values to a set of attack signatures provided by human experts. However, the signature database should be manually revised for each new type of intrusion that is discovered. The limitation of signature-based methods is hard to detect emerging cyber threats, since by their very nature these threats may be launched using previously unknown attacks. These limitations have led to an increasing interest in intrusion detection techniques based upon data mining.

Previous researchers (Agrwal & Srikant, 1995; Hsin, 2005; Morin & Debar, 2003; Ning, Cui, & Reeves, 2002) have developed systematic approaches to analyze network traffic, and the format of network traffic is usually pre-defined and hard to change. Continuous query systems (Cabrera et al., 2001; Sabhnani & Serpen, 2003) share many of the concerns of acquiring and filtering continuous streams of data from the database field, but do not have the ability to easily add new function over that data.

2.2. IDS alert aggregation

Intrusion detection systems (IDSs) considered as powerful security tools in computer systems environments are widely deployed in computer networks to stand against a wide variety of attacks. These systems collect activities within the protected network and analyze them in order to detect intrusions, where activities are usually collected from two main data sources, network packet streams and host log files. There are many researches discussing about information aggregation and correlation of different alerts. Once the information is collected, the detection algorithm starts looking for any evidence for intrusion existence. Since the number of alerts increases rapidly nowadays, it is more difficult for security experts to organize information of intrusions from these numerous alerts to define intrusions quickly.

A probabilistic-based reasoning method (Valdes & Skinner, 2001) was used to correlate alerts by measuring and evaluating the similarities of alert attributes. Alert aggregation and scenario construction are conducted by enhancing or relaxing the similarity requirements in some attributes fields. A correlation system based on Bayesian reasoning (Chen, DeWitt, Tian, & Wang, 2000) was proposed to pre-define the relationship between mission goals and corresponding security events for further inference and correlation.

A "mission-impact-based" correlation system focusing on the attack impacts on the protected domains (Ning, Xu, Healey, & Amant, 2004) used clustering algorithms to aggregate and correlate alerts. Security incidents are ranked based on the security interests and the relevance of attacks to the protected networks and systems. Backward and forward reasoning techniques (Debar & Wespi, 2001), which are applied to correlate alerts, are applied with duplicate and consequence relationship. They used clustering algorithms to detect attack scenarios and situations. This approach pre-defined consequences of attacks in a configuration file.

Chronicle formalism was applied to aggregate and correlate alerts (Shin, Kim, & Ryu, 2004). The approach performs attack scenario pattern recognition based on known malicious event sequences. Therefore, this approach is similar to misuse detection and cannot detect new attack sequences.

Many researchers built alert correlation systems based on matching the pre-/post-conditions of individual alerts (Erhard, Gutzmann, & Libati, 2000; Goldman et al., 2001; Park & Lee, 2001). The idea of this approach is that prior attack steps prepare for later ones. Therefore, the consequences of earlier attacks correspond to the prerequisites of later attacks. The correlation engine searches alert pairs that have a consequence and prerequisite matching. Further correlation graphs can be built with such alert pairs (Park & Lee, 2001). One challenge to this approach is that a new attack cannot be paired with any other attack because its prerequisites and consequences are not defined. Recently, this approach has been extended the pre/post-condition-based correlation technique to correlate some isolated attack scenarios by hypothesizing missed attack steps (Porras, Fong, & Valdes, 2002).

2.3. IDS alert reduction

Recently, IDSs deployment raised a serious problem, namely management of a large number of triggered alerts. This problem is becoming worse by the fact that some commercial IDSs may generate thousands of alerts per day. Furthermore, many researchers indicated that the designed IDSs for detecting intrusions become more powerful at the expense of the increase of false positive alerts. Identifying the real alerts from the huge volume of alerts is a frustrating task for security experts or network administrators. Thus, reducing false alerts becomes a critical issue in the aspect of IDSs efficiency and usability.

Only few researches had been done on reducing false alerts of IDSs. A filtration technique of mining historical alerts (Alharby & Imai, 2005) is proposed to reduce false alerts rate, which treats the frequent behaviors over an extended period of time as normal behaviors. First, an approach is proposed for characterizing the "normal" stream of alerts. In addition, an algorithm for detecting anomalies by using continuous and discontinuous sequential patterns is developed, and the results of preliminary experiments shows this research is indeed effective for some cases of real-world intrusions.

A false alert classification model (Madden, Shah, & Hellerstein, 2002) was proposed to reduce the false alert rate using classification analysis of data mining techniques and is implemented based on associative classification in the domain of DDoS attack. This research used decision tree to reduce false alerts from IDS and to improve the performance of IDSs for keeping important information. Else, a probabilistic approach (Lin & Tseng, 2004) was introduced for the coupled sensors to reduce the false alerts.

3. Architecture

Although many IDSs have been proposed to assist administrators in detecting intrusion, false alarms are still huge and result in the difficulty of analysis. Traditional intrusion pattern analysis methods use different data sources with their own data formats according to different alert analysis methods. These methods have different characteristics to get the desired intrusion patterns. However, most of these researches are conceptual deficient and mutually independent; some of them provide sufficient data formats, and some others are conspicuous on analysis performance. Each intrusion pattern analysis method has its own limitation, so integrating advantages of these methods seems better than redesigning a new analysis framework. Hence, we propose a decision support system for constructing alert classification behavior patterns to help experts easily construct an alert classification model through the huge amount alerts.

The main purposes of IDS alerts analysis are finding more meaningful alert information and discovering the relation between these real alerts to verify system vulnerabilities and to infer attack causes. Some issues are derived from these purposes:

(1) How to choose appropriate analysis targets and data formats.
(2) How to filter false alerts efficiently.
(3) How to discover attack patterns and display appropriate data types for administrators to make policies.

3.1. The framework of decision support system

In the training stage, a decision support system for constructing an alert classification model consists of alert preprocessing phase, model constructing phase, and rule refining phase as shown in Fig. 1. As we know, most attack patterns are a sequence of actions, which can be represented as a sequence of IDS alerts. In alert preprocessing phase, all alerts triggered by IDS sensors are stored in the alert warehouse and will be transformed into alert sequences for each IDS sensor during a period of time which could be set by experts for batch training. Without alert preprocessing, the time complexity of pattern analyzing will become huge and the accuracy of pattern discovering will low down due to the huge among of false alerts. In model constructing phase, filtering and analysis methods are proposed to assist experts construct different classification rule classes to remove false alert patterns and analyze each existing or novel alert patterns. The normal alert behavior patterns can be trained firstly in an attack-free environment by sequential miming algorithms and then used to reduce the affect of noise on intrusion patterns as more as possible because the normal behavior patterns are always occur periodically and frequently. Hence, the suspicious behavior patterns can be trained in a simulated attacking environment and the corresponding classification labels can be flagged by experts according to the alert patterns in model constructing phase. These obtained patterns could be used to help administrator identify intrusions in the on-line stage. If new alert behavior patterns are discovered, they will be integrated into the alert classification model; otherwise, the alert model could be refined in rule refining phase.

3.2. Alert preprocessing phase

Since most of present attacks are target-specific and stealthy intrusions instead of large-scale violence (Symantec Corp., 2006) and alert transactions in alert warehouse are large, it is necessary to transform raw alert transactions into pre-defined alert transaction formats for further analyzing. Some characteristics of attack tools or intrusions or malwares can be discovered by analyzing alert sequences (Julisch & Dacier, 2002). As shown in Fig. 2, assume a source host triggers the same sequence of alerts against different target hosts in a noise-free environment without false alerts. Generally, it can be easily seen these scenarios from an attacker try out his/her attack tool against different targets.

Alert sequences can be represented as specific characteristics of attack tools or attackers, we hence select alert sequence as our

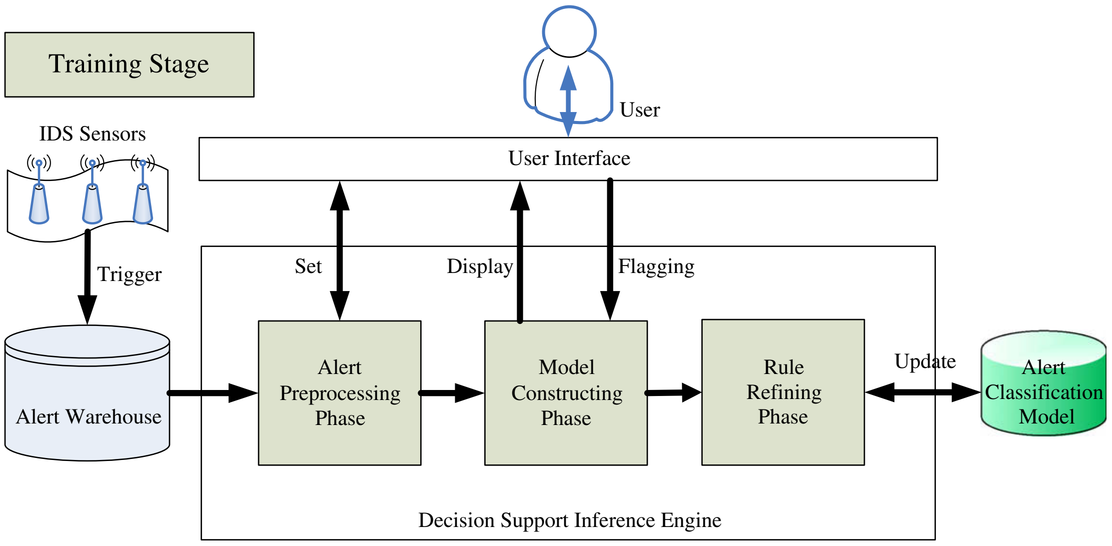

Fig. 1. The framework of decision support system.

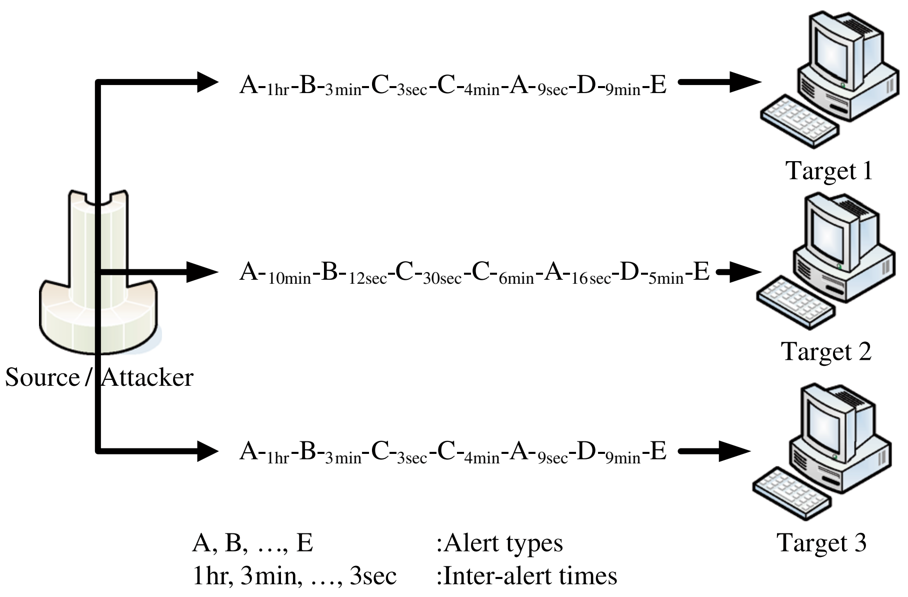
A-1hr-B-3min-C-3sec-C-4min-A-9sec-D-9min-E
Target 1
A-10min-B-12sec-C-30sec-C-6min-A-16sec-D-5min-E
Target 2
Source / Attacker
A-1hr-B-3min-C-3sec-C-4min-A-9sec-D-9min-E
Target 3
A, B, …, E :Alert types
1hr, 3min, …, 3sec :Inter-alert times

Fig. 2. An attack tool being run against three targets.

target data format. We collect IDS alert transactions from many IDS sensors as our training data sources, and set an appropriate time period *Batch_Time_Window* for batch preprocessing. The value of *Batch_Time_Window* may be an hour, a day, a week, or a month; but we suggest that to set the value of *Batch_Time_Window* as one day will be better in our experience. Besides, most intrusions will not continue for a very long time in its attack lifecycle, we set a short-term time period *Short_Time_Window* as the time period of our alert sequence transactions. We assume that each intrusion will finish its lifecycle in this short-term period. The value of *Short_Time_Window* may be 1 min, several minutes, or even an hour; but we suggest to set the value of *Short_Time_Window* as half of an hour will be better in our experience.

After explaining the definition of *Batch_Time_Window* and *Short_Time_Window*, there are still some issues for alert transaction construction. First, for a single alert transaction, it is difficult to classify it into true alert or false alert because some alerts are used to be triggered. Second, it is still not appropriate for alert sequences to be too long because long sequential patterns are not easy to be frequent. Our idea is to design proper policies corresponding to different environments, and some policies of alert sequence partition are proposed to construct alert sequence transactions for model constructing phase with different requirements. These policies are used to partition alert sequences into subsequences in each *Short_Time_Window*. According to different purposes, the corresponding policies are designed to obtain desired alert subsequence transactions. Three policies are shown as follows.

Case 1: Left-to-Right Non-repeat Policy

```
Step 1: Scan alerts from the first (left) to the end (right) of
        sequence.
Step 2: Partition sequence into scanned part and unscanned part.
Step 3: IF next alert in the unscanned part is equal to certain alert
        in the scanned part.
          Step 3.1  Partition the scanned part as a new subsequence.
          Step 3.2  Set the unscanned part as a new sequence.

Step 4: Else, Update the scanned and unscanned parts.
```

Step 5: IF there is still any element in the unscanned part, GOTO Step 1.

Case 2: Right-to-Left Non-repeat Policy

```
Step 1: Scan alerts from the end (right) to the first (left) of
        sequence.
Step 2: Partition sequence into scanned part and unscanned part.
Step 3: IF next alert in the unscanned part is equal to certain alert
        in the scanned part.
        Step 3.1  Partition the scanned part as a new subsequence.
        Step 3.2  Set the unscanned part as a new sequence.
Step 4: Else, update the scanned and unscanned parts.
Step 5: IF there is still any element in the unscanned part, GOTO
        Step 1.
```

Case 3: Equi-length Policy

```
Step 1: Set a value of the subsequence length.
Step 2: Partition each alert sequence into several subsequences
        with the fixed length.
```

Here 'Left-To-Right Non-Repeat Policy' is used as our partition policy for example. We suppose that there is an alert sequence of sensor H_1 in a *Short_Time_Window* t_1 as following:

SensorID	*Short_Time_Window* (t_1)
H_1	XABYXCYC

At first, let AS be alert sequence with length 8: XABYXCY; because AS[4] equals AS[0], so this sequence is divided into two alert subsequences: the scanned part XABY and the unscanned part XCYC, and executes partitioning again in the unscanned part XCYC as new alert sequence, AS with length 4. In new AS, we can find that AS[4] equals AS[2] again, hence this new alert sequence is divided into two alert subsequences: the scanned part XCY and the unscanned part C. After original alert sequence being scanned complete, we can get three alert subsequence transactions such as XABY, XCY and C as follows.

SensorID	*Short_Time_Window* (t_1)		
H_1	XABY	XCY	C

More sequence partition algorithms can be used to partition a long sequence into subsequences according to the application domain. In this article, three partition algorithms are discussed to our proposed decision support system.

3.3. Model constructing phase

Some researches discussed how to filter false alerts efficiently, different data characteristics and different filtering heuristics bring quite different filtering results. Generally speaking, most of them use specific analysis methods or compile expert experiences to construct filter models, and use them to discard those highly-possible false alerts to get clearer data.

Besides, finding specific patterns in these kinds of numerous sources like to discover meaningful patterns in distributed databases or data warehouses for decision support. Many methods have been proposed to analyze behavior models in databases, and different methods with different data source formats cause different outcomes. In our thought, none of these algorithms is powerful enough for correctly detecting intrusions.

There are two different purposes for alert pattern discovering obviously: false alert filtering and useful alert pattern discovering. It is common to filter false alerts before alert pattern discovering because noisy alerts will affect the accuracy of results. In other words, our approach tries to discover patterns of false alerts first, and then continue to discover useful patterns of suspicious or known alerts.

In the training stage, some construction methods are designed to build specific behavior rule classes. Three types of behavior classification rule classes with domain experts including normal behavior, intrusion behavior, and suspicious behavior rule classes are constructed. Each type of rule classes is constructed by individual methods, which uses different data sources as their data inputs.

For example, normal behavior rule class which is the collection of false alert patterns, can be constructed by collecting IDS alerts in an attack-free network environment and using specific normal behavior rule class construction method to discover false alert sequences. Besides, intrusion rule class and suspicious rule class can be constructed by collecting IDS alerts in a simulated network environment using some existing intrusion tools to generate simulated intrusions, to discover interim suspicious behavior rule class by specific rule class construction method. All interim classification rules of intrusion/suspicious behavior rule classes will be provided to experts, and the system will interact with them to flag these suspicious alert sequences with specific tags. If one suspicious pattern is a known attack, experts can flag it as a specific classification name in a kind of intrusion behavior classification rules; if one suspicious pattern is never verified, then it is an unknown pattern and is considered as a kind of suspicious behavior classification rules.

The procedure of behavior classification rule class construction in the training stage is shown as following:

1. Construct normal behavior classification rule class.
2. Construct suspicious/intrusion classification rule class.
3. Classify classification rules of suspicious/intrusion rule class into suspicious behavior rule class and intrusion behavior rule class.

These rule classes are used to monitor IDS alert behaviors in the on-line stage. IDS sensors trigger alerts at any moment, and it is not easy for experts to classify these consecutive alert sequences into normal behaviors or suspicious behaviors without our decision support system. With these types of rule classes, it is possible for real-time monitoring IDS alert sequences to discover most intrusion behaviors. The decision support system lightens the load of experts in analyzing IDS alerts and assist them focusing on how to explain the intrusion principles and how to fix the root causes. The rule class construction algorithms are detailedly described in next section.

The activation flow of each rule class is shown in Fig. 3. In the on-line stage, alert sequences are triggered by IDS tools and will be forwarded to alert warehouse. Before being stored into alert warehouse, each alert sequence will be matched by classification rule classes of classification model first. The normal behavior patterns will be firstly used to filter out most part of false alerts. Then, the intrusion behavior patterns will be used to detect the true intrusion alerts. Finally, the remaining alerts should be classified into known suspicious behavior patterns or unknown suspicious alerts.

If there is any alert subsequence matching the classification rules in normal behavior classification rule class, it will be filtered out immediately; if no alert subsequence is matched, classification rules of intrusion behavior classification rule class will be triggered to verify these on-line alert sequences. In the similar way, if any alert subsequence is matching by the classification rules of intrusion behavior classification rule class, system will highlight the alert subsequence to notice experts; if not, classification rules of suspicious behavior classification rule class will be triggered to verify these alert sequences again. If there is still no classification rule matched by alert subsequences, these alert sequences will be stored in alert warehouse as new batch data sources for rule class trainer in the next time window (see Fig. 4).

3.4. Rule refining phase

These obtained behavior rule classes could be used to help administrator identify intrusions in the on-line stage. If new alert behavior classification rules are discovered, they will be integrated into the alert classification rule classes; otherwise, the alert rule classes could be refined in the rule refining phase.

First, for the efficiency of rule inference, the maximum number of classification rule in each classification rule class needs to be limited. If new classification rules are incrementally added into rule classes, the inference performance of on-line monitoring will be decreased due to the huge of rules. The initial number of maximum rule number of each classification rule class in our experiment is set to 200.

If the quotas of classification rule class are full with classification rules, it is necessary to find replacement strategies for rule replacement. It is possible for each classification rule class to use different rule-replacement policies to implement rule-replacement according to individual data characteristics. In this paper, a *least recently used (LRU)* replacement policy is used in each classification rule class. LRU policy is often used as a page-replacement algorithm in operating system. LRU replacement associates with each rule and the time of that rule's last use. When a rule must be replaced, LRU chooses that rule and that has not been matched for the longest period of time. This strategy is the optimal rule-replacement policy looking backward in time, rather than forward.

For the rule contradiction checking issue, two classification rules with the same alert sequence may be flagged in different classification labels (e.g., normal or intrusion) in two consecutive batch interactions. For example, in one time interval, alert sequence ABC are flagged as "Normal", but ABC are flagged as "Suspicious" in the next time interval according to the different situations. In this case, system will interact with experts to make sure the flag of this alert sequence instead of simple and fixed replacement.

4. Rule class construction algorithms of model constructing phase

It is very difficult and cost for experts and administrators to monitor on-line IDS alerts to discover useful intrusion patterns. Nowadays, experts are still using their own knowledge and experience to defend intrusions, but the efficiency and effectiveness of intrusion detection is hard to improve. In this research, our idea is to construct a model to classify the collected alert patterns into

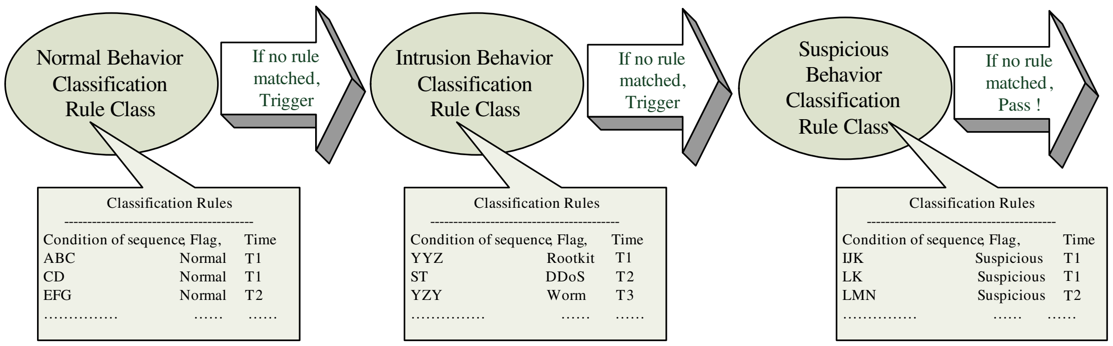
Normal Behavior Classification Rule Class
If no rule matched, Trigger
Intrusion Behavior Classification Rule Class
If no rule matched, Trigger
Suspicious Behavior Classification Rule Class
If no rule matched, Pass !
Classification Rules
Condition of sequence Flag, Time
ABC Normal T1
CD Normal T1
EFG Normal T2
Classification Rules
Condition of sequence Flag, Time
YYZ Rootkit T1
ST DDoS T2
YZY Worm T3
Classification Rules
Condition of sequence Flag, Time
IJK Suspicious T1
LK Suspicious T1
LMN Suspicious T2

Fig. 3. Meta rules of classification rule classes for on-line monitoring.

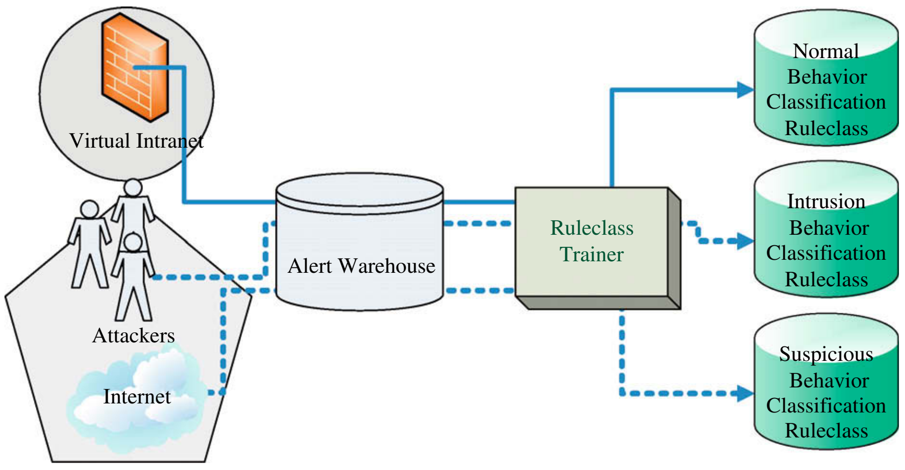
Virtual Intranet
Attackers
Internet
Alert Warehouse
Ruleclass Trainer
Normal Behavior Classification Ruleclass
Intrusion Behavior Classification Ruleclass
Suspicious Behavior Classification Ruleclass

Fig. 4. Three types of alert behavior classification rule classes.

several classifications with different flags, and experts can discover on-line suspicious or intrusion patterns easily and quickly. The model consists of normal, intrusion and suspicious behavior classification rule classes, where each rule class consists of hundreds to thousands classification rules depending on the computational ability of each detection sensor. Each rule class is constructed with individual construction method, which uses individual data source in specific network environments as their data input.

Normal behavior classification rule class represents the set of false alert patterns. Frequent behaviors of IDS alerts in an attack-free environment are the most alerts in experience, and these alerts are exactly false alerts. We use an AprioriAll-like sequential pattern mining algorithm (Valdes & Skinner, 2001) to discover frequent sequences of IDS alerts in an attack-free environment in laboratories, and these frequent sequences are used to construct classification rules of normal rule class.

As we know, some intrusions which have fixed patterns can be easily discovered by simple pattern matching approach on-line. However, varied intrusions become too sophisticated to detect, our suspicious/intrusion classification rule construction method is hence focusing on these kinds of intrusions. We design a score based method in this approach to discover varied alert subsequences as suspicious patterns. Some rootkit tools are used to simulate real-world attacks in a period of time, and these IDS alerts are collected as training data of suspicious/intrusion classification rule class construction. Some suspicious alert sequences will be discovered with this method, and system will interact with experts to classify these sequences into intrusion rule class or suspicious rule class; if one sequence is considered as a known intrusion, it is classified by intrusion rule class, and then used to update the classification rule of intrusion rule class; if one discovered alert subsequence is not considered as a known intrusion, it is highly possible to be a novel intrusion and should be classified into suspicious rule class to construct a new suspicious classification rule.

4.1. Normal behavior rule class construction

To achieve an objective of high detection rate without missing any intrusion, the design of IDS signature-based rules are asked to be as powerful as possible, but that makes IDS become more sensitive. Filtering of dirty alerts has two advantages including reducing the accuracy of alert analysis and the complex of data execution at the same time. Generally speaking, most of these researches use special analysis methods or compile expert experiences to construct filter models, and use this filter model to discard those highly-possible false alerts to get clear data. Before discussing the design of procedure, we must consider the characteristics of alerts and attacks.

(1) *Alert frequency*: According to different importance and bandwidth of hosts, their numbers of triggered alerts are very different obviously. Besides, the scale of subnets aggravate the difference of alert frequencies; the bigger scale a subnet is, the more total alert number is in that subnet. It is common to collect thousands of alerts in a busy subnet.
(2) *False alert frequency*: According to the results of most researches, it is indicated that false alert rates of different IDSs are between 60% and 90%. Some of these false alerts occurred with similar patterns in the same network, such as specific alert sequences or frequent source IP addresses, and those normal behaviors are triggered as alerts but they are not intrusions in fact. In other words, the idea of these researches is that if we can discover frequent behavior patterns of alerts, these frequent patterns are most likely to be false alerts.
(3) *Attack characteristic*: There are also some characteristics in attacks, so some filtering methods using specific characteristics to detect the corresponding intrusions efficiently. For example, some specific Rootkits will give rise to constant alert sequences, and is appropriate to use sequential pattern mining for filtering out these false alerts. For another example, worm is a kind of variable intrusions, so using Genetic Algorithm to filter false alerts seems better than others.

Our idea is to construct classification rules of normal behavior classification rule class, we can hence execute filtering by comparing all on-line alert sequences with the normal behavior classification rule class.

A normal behavior classification rule class construction (NBCRC) algorithm shown in Fig. 5 is proposed to discover attack sequences of normal network behaviors. This algorithm consists of two steps of sequential pattern mining and classification rule construction. At first, we assume frequent behavior, over an extended period of time, is likely to be normal. In other words, a modified *AprioriAll* sequential pattern mining method (Valdes & Skinner, 2001) is used to discover frequent sequences of alerts in single sensor, and these frequent sequences are treated as false alert patterns and collected as a rule class. At last, all frequent sequences are flagged with 'normal' and transformed to classification rules of normal behavior classification rule class. The normal behavior classification rule class is used to reduce false alert sequences in on-line stage. Our proposed specific algorithm is shown in Algorithm 1. To fit in with requirements of flexibility and robustness for administrators in such a decision support system, system interacts with administrators to decide the value of minimum support in AprioriAll algorithm dynamically. That makes it possible for administrators to make proper decisions according to different situations.

```
Algorithm 1: The normal behavior classification rule class construction algorithm.

  Input: Alert sequences of individual IDS sensors.
  Output: Normal rule class with classification rules.
  Step 1: Set a minimum support in each sensors.
  Step 2: Generate frequent sequences of one sensor with its
          support.
  Step 3: Flag each frequent alert sequences with 'normal'
          and use these flagged alert sequences to construct
          classification rules of normal behavior patterns.
```

4.2. Intrusion/suspicious behavior rule class construction

As shown in Fig. 6, the procedure of suspicious/intrusion behavior classification rule class construction, including alert sequence transformation, suspicious scoring and classification rule construction, is proposed to construct suspicious/intrusion behavior classification rule class, where each rule is represented as a suspicious/intrusion alert sequence.

Since the false alerts have been filtered out in this phase, all alert sequence transactions into 2-candidate alert subsequences (2-candidate means the length of this sequence is 2) to inspect each possible alert sequence strictly. Besides, we propose the specific suspicious scoring method to model possible behaviors of intrusions where two variables are designed to record locations and frequency of each 2-candidate alert subsequences respectively between several consecutive *Short_Time_Windows*. According to

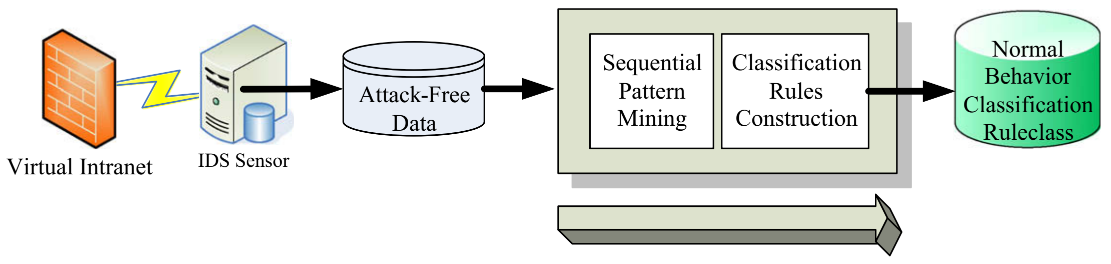
Virtual Intranet
IDS Sensor
Attack-Free Data
Sequential Pattern Mining
Classification Rules Construction
Normal Behavior Classification Ruleclass

Fig. 5. The procedure of normal behavior classification rule class construction.

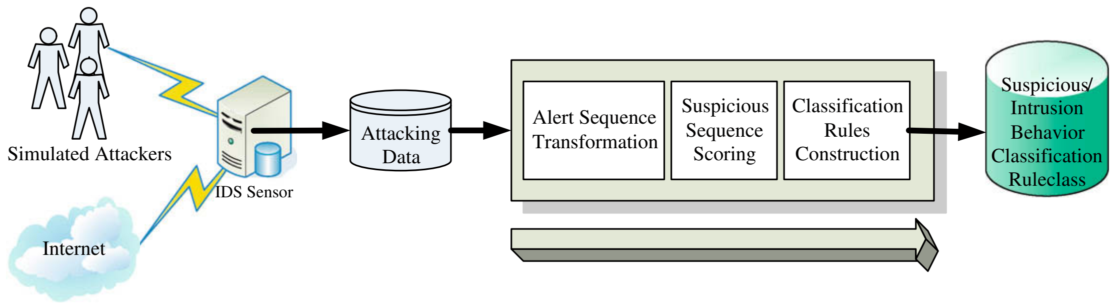
Simulated Attackers
Internet
IDS Sensor
Attacking Data
Alert Sequence Transformation
Suspicious Sequence Scoring
Classification Rules Construction
Suspicious/ Intrusion Behavior Classification Ruleclass

Fig. 6. The procedure of suspicious/intrusion classification rule class construction.

the characteristics of intrusions, higher the scoring value is, more suspicious the alert pattern is. If a subsequence is continuously repeated and discovered in the same host set, it will be treated as a suspicious behavior of intrusion.

For each 2-candidate alert subsequence, $Host(t_i)$, a set of hosts, represents the locations which discovered this subsequence in the time period t_i and the $|Host(t_i)|$, the number of hosts, represents the frequency which discovered this subsequence in t_i. Assume there are n hosts in the simulated environment. The scoring policy could be divided into three cases according to $Host(t_i)$ and $|Host(t_i)|$. Case 1 represents continuous attack from the same host set. Case 2 means that the attack between t_i and t_{i-1} is light variation on these hosts. Case 3 is much different from t_{i-1} and t_i since the attacking target is quite different. In case 2 and case 3, $\oplus$ is an exclusive-or operator.

Scoring policies

CASE 1: $Host(t_i) = Host(t_{i-1})$.
Set **Score** = 0 AND **Repeat**++.
CASE 2: $Host(t_i)! = Host(t_{i-1})$ & $|Host(t_i)| = |Host(t_{i-1})|$.
Set **Score** = **Score** + 1/2 $[|Host(t_i)$ Å $Host(t_{i-1})|*(1/n)]$
AND **Repeat** = 0.
CASE 3: $Host(t_i)! = Host(t_{i-1})$ & $|Host(t_i)|! = |Host(t_{i-1})|$.
Set **Score** = **Score** + $|Host(t_i)$ Å $Host(t_{i-1})|*(1/n)^*$
$[||Host(t_i)| - |Host(t_{i-1})||]$ AND **Repeat** = 0.

Finally, specific thresholds are set to flag some special situations as suspicious attack patterns, and then administrators are noticed to trace the causes of suspicious patterns and fix intruded hosts. Three flagging rules are proposed as follows to determine 2-candidate alert sequences with suspicious patterns if there is any 2-candidate alert sequence conforming to one of these rules. Therefore, all alert sequences of suspicious behaviors are constructed into classification rules of suspicious/intrusion behavior classification rule class.

Flagging rules

RULE 1: IF **Score** > **Score_TH,**
THEN flag as "Suspicious".
RULE 2: IF **(Repeat** > **Repeat_TH** & $|Host(t_i)|$**! = 0)**
& **Score** < **= Score_TH,**
THEN flag as "Suspicious".
RULE 3: IF $|Host(t_i)|$ = = n&Repeat < = Repeat_TH,
THEN flag as "Suspicious".

After the procedure of suspicious/intrusion behavior classification rule class construction, we can construct lots of suspicious alert sequences, and then experts are asked to flag these suspicious alert sequences with tags of specific intrusions as shown in Fig. 7.

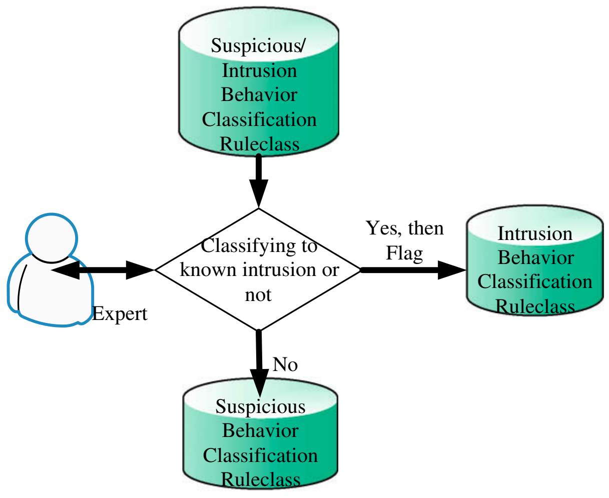
Suspicious/ Intrusion Behavior Classification Ruleclass
Expert
Classifying to known intrusion or not
Yes, then Flag
Intrusion Behavior Classification Ruleclass
No
Suspicious Behavior Classification Ruleclass

Fig. 7. The procedure of identifying intrusion rule classification.

System provides each suspicious alert sequence to experts, and experts flag these alert sequences by their own experiences and domain knowledge to construct classification rules of suspicious behavior classification rule class or intrusion behavior classification rule class. If one alert sequence is considered as a known intrusion pattern, it is necessary for experts to flag this alert sequence with specific intrusion signature tag which represents a name of known attack; if one alert sequence is not considered as a known intrusion by experts, it is perhaps a kind of novel intrusion patterns, and this alert sequence is still flagged as 'suspicious' to be constructed as a new classification rule of suspicious behavior classification rule class.

According to scoring policies and flagging rules, the algorithm of the suspicious/intrusion behavior classification rule class construction is shown in Algorithm 2. After calculating the $Host(t_i)$ and $|Host(t_i)|$ at time t_i in **Step 2**, the scoring policy is applied to calculate the values of variation score and repeat score of each 2-subsequences which are transformed in Step 1. Then, the scoring policies and flagging rules are used to help experts construct the classification rules in **Step 4** and **Step 5**. In order to discover complete intrusions patterns, we will string up 2-candidates alert subsequences into all k-subsequences in **Step 6** if two subsequences have the end to end relation. For example, if a set of 2-candidate subsequences is {AB, BC}, then the extended subsequences {ABC} will be generated. After all extended alert subsequences are discovered. Experts are asked to flag the classification label in each alert subsequence in **Step 7**.

```
Algorithm 2: The suspicious/intrusion behavior classification
rule class construction algorithm.

  Input:  Alert sequences, Score_TH and Repeat_TH.
  Output: Alert behavior classification rules of suspicious and
          intrusion behavior.
  Step 1: For each sensor, Transform the candidate alert
          sequences into 2-candidate subsequences.
  Step 2: For each 2-subsequence, calculate the $Host(t_i)$ and
          the $|Host(t_i)|$ values.
  Step 3: Store results of all 2-candidate subsequence
          transactions.
  Step 4: Apply scoring policies to calculate the values of
          score and repeat.
  Step 5: Flag classification label using the flagging rules.
  Step 6: Aggregate the extended alert subsequences.
  Step 7: Interact with experts to flag all suspicious alert
          sequences to classify these alert sequences into
          suspicious alert classification rules and intrusion
          alert classification rules.
```

5. Experiment

In order to evaluate the efficiency of the obtained behavior pattern, we setup an experimental environment equipped with one server and eight hosts to simulate real network.

5.1. The overview of the related tools

Snort and Intrusion Detection/Prevention System (2006) is a signature-based intrusion detection system and open source software. It represents a cost-effective and robust NIDS solution that fits the needs of many organizations. Snort is very flexible in the ways it can be deployed. Many security industry watchdogs use the Snort signatures as part of their security announcements (such as CERT). Intrusions are ravaging the Internet since they are constantly evolving to new variants by multiple updates weekly. The Snort mailing lists are fantastic resource for people who are trying to write their own signatures to develop the applications of central monitoring and alerting consoles.

Basic Analysis and Security Engine (BASE) (2005) is the basic analysis and security engine. It is based on the code from the analysis console for intrusion databases (ACID) project. This application provides a web front-end to query and analyze intrusions from the alerts coming generated by Snort.

To post processing of alert transactions requires commercial databases, e.g., the MS-SQL 2000 server, which is used to automatically extract, transform and load data from heterogeneous sources. The MS-SQL 2000 server analysis services includes OLAP, data mining and data warehouse tools, which makes better decisions, performs rapidly, and executes analysis on large and complex data sets using multi-dimensional storage.

The DRAMA (CORETECH Inc., 2006) is applied for building up the decision support inference engine. DRAMA is a rule-based, client–server tool/environment for knowledge-based system development. It can assist knowledge engineers in building up an rule-based decision support system. Using the client–server architecture of DRAMA, the rule base is maintained on a server and clients could access DRAMA server for inference services.

5.2. The design of the experimental environment

The knowledge-based architecture of collaborative discovering of suspicious network behaviors is implemented. All the related tools described above are one server, which plays the role of IDS alert analysis server, including IDS center for alert warehouse, web-based analysis console and alert analysis console. Besides, eight hosts all play the role of IDS sensors to trigger alerts as our data sources. The system and network profiles of these sensors are shown in Fig. 8.

We have conceptualized alerts according to Snort rule set, which is a network-based IDS whose alerts are triggered by a collection of signature-based rules. Each Snort rule is composed of a Snort identification number, a message that is included in the alert when the rule is triggered, an attack signature, and references to sources of information about the attack. Each alert is provided with an identifier, time and data, sensor identifier, triggered signature, IP and TCP headers and payload. These alerts will be stored in the relational database as our alert warehouse. Alerts in one period of time, e.g., 24 h for *Batch_Time_Window* (BTW), are collected by IDS center as data source in this experiment. For easy reading, we replace the original alert signature names with different capital letters.

The training data set collection in our experiment is shown as follows. The total number of IDS alerts is 164 collected in a day and the *Short_Time_Window* (STW) is set to 1 h. The total number of alert sequence transactions is 295.

Total number of IDS alerts	BTW	STW	Total number of alert sequence transactions
1640	24 h	1 h	295

5.3. The experimental results

To verify the feasibility of our decision support system, we use the data set to execute our alert classification model. We construct an alert classification model in the training stage, and use this alert classification model for on-line monitoring. After filtering false

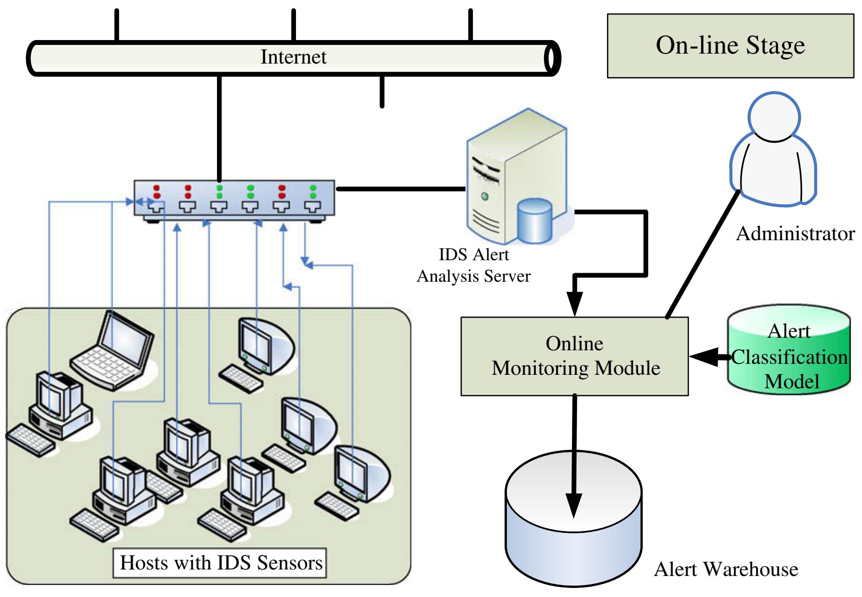

Fig. 8. System prototype in experiments.

alerts with normal behavior classification rule class, the result of alert reduction is shown in Fig. 9. The classification rules of normal behavior classification rule class which are used in this experiment would change their time flags for rule refining.

The filtered alert sequences are used to discover suspicious/intrusion alert sequences. We used pre-defined suspicious alert classification rule class and intrusion alert classification rule class to discover useful alert patterns for experts. Suspicious alert classification rule class and intrusion alert classification rule class are used to verify the on-line alert sequences. The result of suspicious/intrusion alert sequence verification is shown in Fig. 10. Twenty-nine percentage of candidate IDS alert sequences are known alert sequences, 42% of candidate IDS alert sequences are suspicious alert sequences, and 29% of candidate IDS alert sequences are unknown alert sequences. The alert sequences triggered as suspicious alert sequences or intrusion alert sequences would be highlighted to notice experts for on-line detection, and these used alert classification rules would change their time flags for rule refining. The part of unknown alert sequences would be forwarded to alert warehouse as the data sources of next batch model construction in the training stage.

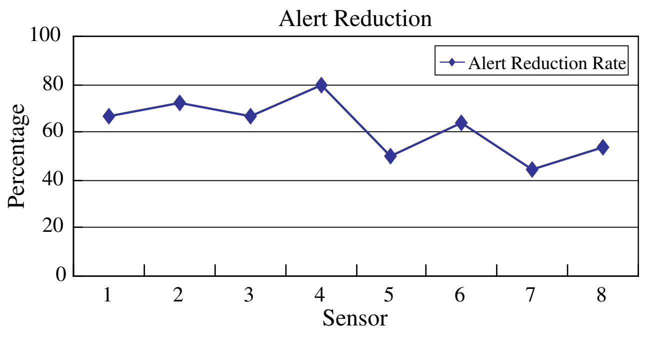

Fig. 9. Alert reduction rate of normal behavior classification model.

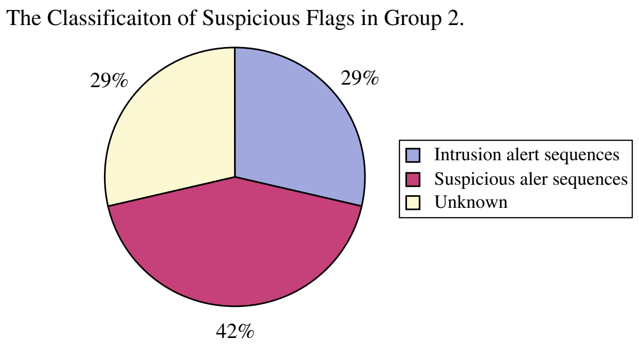

Fig. 10. Observations of percentages of different suspicious flags.

6. Conclusion

In this paper, we proposed a decision support system for constructing an alert classification model, which consists of three phases: alert preprocessing phase, model constructing phase and rule refining phase. In alert preprocessing phase, raw IDS alerts are collected into alert warehouse and will be transformed into alert transactions. Three kinds of alert classification rule classes including normal behavior classification rule class, intrusion behavior classification rule class and suspicious behavior classification rule class, are constructed in model constructing phase to filter normal alert patterns and then discover each known or novel alert pattern based upon the remaining alert transactions. Each classification rule class consists of fixed number of classification rules. An AprioriAll-like sequential pattern mining algorithm is proposed to construct classification rules of normal behavior classification rule class. A set of intrusion tools to collect data sources of suspicious/intrusion behavior classification rule class construction in the simulated attacking network environment. Our experiment demonstrates that the accuracy of our decision support system is well. This decision support system will construct a classification model for on-line monitoring. The alert classification model is useful for experts to discover suspicious or intrusion patterns quickly and precisely, and lightens the load of on-line alert analysis for experts obviously. The current implementation of our research constructs rule classes in the alert classification model. In the near future, more types of classification rule classes will be created for enhancing the performance of detection.

Acknowledgement

This work was partially supported by National Science Council of the Republic of China under contracts NSC95-2520-S009-008-MY3, NSC 96-2752-E-009-006-PAE and NSC 97-2511-S-009-001-MY3.

References

Agrwal, R., & Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the 11th international conference on data engineering* (pp. 3–14).

Alharby, A., & Imai, H. (2005). IDS false alarm reduction using continuous and discontinuous patterns. In *Proceedings of ACNS 2005* (pp. 192–205).

Basic Analysis and Security Engine (BASE) (2005). URL: <http://secureideas.sourceforge.net/>.

Bridis, T. (2002). Powerful attack cripples majority of key Internet computers. In *Yahoo! news*.

Cabrera, J. B. D., Lewis, L., Qin, X., Lee, W., Prasanth, R. K., Ravichandran, B., & Mehra, R. K. (2001). Proactive detection of distributed denial of service attacks using MIB traffic variables-A feasibility study. In *Proceedings of the seventh IFIP/IEEE international symposium on integrated network management* (pp. 609–622).

CERT Coordination Center (2006). URL: <http://www.cert.org/>.

Chen, J., DeWitt, D. J., Tian, F., & Wang, Y. (2000). NiagaraCQ: A scalable continuous query system for internet databases. In *Proceedings of ACM SIGMOD 2000* (pp. 379–390).

Debar, H., & Wespi, A. (2001). The intrusion-detection console correlation mechanism. In *Proceedings of the fourth international symposium on recent advances in intrusion detection (RAID 2001)* (pp. 235–240).

DRAMA Expert System, CORETECH Inc. (2006). URL: <http://www.coretech.com.tw/c_DRAMA.htm>.

Erhard, W., Gutzmann, M. M., & Libati, H. M. (2000). Network traffic analysis and security monitoring UniMon. In *Proceedings of the IEEE conference on high performance switching and routing* (pp. 439–46).

Goldman, R. P., Heimerdinger, W., Harp, S. A., Geib, C. W., Thomas, V., & Carter, R. L. (2001). Information modeling for intrusion report aggregation. In *DARPA information survivability conference and exposition II* (pp. 115–137).

Hsin, W. Y. (2005). *A study of alert-based collaborative defense*. Master thesis, National Chiao Tung University, Hsinchu, Taiwan, ROC.

Julisch, K., & Dacier, M. (2002). Mining intrusion detection alarms for actionable knowledge. In *Proceedings of the eighth ACM international conference on knowledge discovery and data mining* (pp. 366–375).

Lee, K., Kim, J., Kwon, K. H., Han, Y. G., & Kim, S. (2008). DDoS attack detection method using cluster analysis. *Expert Systems with Applications, 34*, 1659–1665.

Lin, S. C., & Tseng, S. S. (2004). Constructing detection knowledge for DDoS intrusion tolerance. *Expert Systems with Applications, 27*, 379–390.

Madden, S. R., Shah, M. A., & Hellerstein, J. M. (2002). Continuously adaptive continuous queries over streams. In *Proceedings of ACM SIGMOD 2002* (pp. 49–60).

Morin, B., & Debar, H. (2003). Correlation of intrusion symptoms: An application of chronicles. In *Proceedings of the sixth symposium on recent advances in intrusion detection (RAID 2003)* (pp. 92–112).

Ning, P., Cui, Y., & Reeves, D. S. (2002). Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the nineth ACM conference on computer and communications security* (pp. 245–154).

Ning, P., Xu, D., Healey, C. G., & Amant, R. A. St. (2004). Building attack scenarios through integration of complementary alert correlation methods. In *Proceedings of the 11th annual network and distributed system security symposium (NDSS'04)*.

Park, K., & Lee, H. (2001). On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications* (pp. 15–26).

Porras, P. A., Fong, M. W., & Valdes, A. (2002). A mission-impact-based approach to INFOSEC alarm correlation. In *Lecture notes in computer science, proceedings recent advances in intrusion detection* (pp. 95–114).

Sabhnani, M., & Serpen, G. (2003). Application of machine learning algorithms to KDD intrusion detection dataset within misuse detection. In *Proceedings of the international conference on machine learning; models, technologies and applications (MLMTA'03)* (pp. 209–215).

Shin, M. S., Kim, E. H., & Ryu, K. H. (2004). False alarm classification model for network-based intrusion detection system. In *Proceedings of IDEAL 2004* (pp. 259–265).

Snort, Intrusion Detection/Prevention System (2006). URL: <http://www.snort.org/>.

Symantec Corp. (2006). Symantec internet security threat report: Trends for July 05–December 05. In *Vol. IX*. URL: <http://www.symantec.com/index.htm>.

Valdes, A., & Skinner, K. (2001). Probabilistic alert correlation. In *Proceedings of the fourth international symposium on recent advances in intrusion detection* (pp. 54–68).

Xu, J., & Lee, W. (2003). Sustaining availability of web services under distributed denial of service attacks. In *IEEE transactions on computers* (Vol. 52(2)) (pp. 195–208).