# Online Training of SVMs for Real-time Intrusion Detection Based on Improved Text Categorization Model

Zonghua Zhang, Hong Shen
Graduate School of Information Science,
Japan Advanced Institute of Science and Technology,
1-1 Tatsunokuchi, Ishikwa, 923-1292, Japan.
Tel: 81-761-51-1285, Fax: 81-761-51-1149,

*Abstract*— **As intrusion detection essentially can be formulated as a binary classification problem, it thus can be solved by an effective classification technique−Support Vector Machine(SVM). Additionally, some text processing techniques can also be employed for intrusion detection, based on the characterization of the frequencies of the system calls executed by the privileged programs. Based on the intersection of these two research domains, i.e. pattern recognition and text categorization, and breaking the strong traditional assumption that training data for intrusion detectors are readily available with high quality in batch, the conventional SVM, Robust SVM and one-class SVM have been modified respectively based on the idea from Online SVM in this paper, and their performances are compared with that of the original algorithms. After elaborate theoretical analysis, concrete experiments with 1998 DARPA BSM data set collected at MIT's Lincoln Labs are carried out. These experiments indicate that the modified SVMs can be trained online and the results outperform the original ones with fewer support vectors(SVs) and less training time without decreasing detection accuracy. Both of these achievements could significantly benefit an effective online intrusion detection system.**

*Keywords*— **computer security, intrusion detection, anomaly detection, support vector machines, text categorization**

## I. Introduction

As computer networks play an increasingly vital role in modern society with rapid increases in the functionality, connectivity and accessibility, more and more efforts are being put to its security, because a major attack can significantly reduce the capability of information systems. Any exploitable weakness of networks that can be used by hackers and criminals can potentially cause great losses to people. As backup measures for intrusion prevention, such as user authentication, authorization, encryption, etc., intrusion detection techniques are attracting increasing attention, and some achievements have been applied widely, with limited performance.

Briefly, the aim of intrusion detection is to identify malicious attacks that might threaten the security from the normal activities of information systems. Existing intrusion techniques fall into two general categories: anomaly detection and misuse detection. Anomaly detection techniques mainly focus on establishing normal activities pattern (set or rule) $\Omega$, and any current activity $\omega$ that deviates from $\Omega$ is treated as an intrusion. On the contrary, misuse detection techniques attempt to create a model of attack signatures $\Psi$, when a current signature $\psi$ matches $\Psi$, it is regarded as an intrusion. However, defects exist in both anomaly detection and misuse detection, false positives ($\psi$ is misclassified to $\Omega$) and false negatives (novel attack $\psi \in \Psi$ is ignored) often cause these techniques to fail. Due to the complementary nature of these two approaches, the intuitive approach to design an effective intrusion detection system is to combine them together. Generally, the criterion for evaluating the efficiency of an IDS is its ability to detect underlying attacks, while minimizing the false positive rate.

Based on the analysis of the available literature on intrusion detection, we found that two elements are essential to intrusion detection, namely, a data model of the observable subjects or events, and the corresponding techniques for characterizing and analyzing the data model. Specifically, several questions should be answered carefully: What observable subjects should be selected for monitoring and analyzing? What attributes should be considered for characterizing these related subjects? What existing approaches or novel methods can be employed to detect anomalies based on the characterized observation? It is well known that a computer and network system generally contains two components: hosts and communication links among hosts. Consequently, network traffic data, from captured data packets travelling on the communication links, and audit data, which record the sequence of events on the hosts, can be selected as observable subjects. Actually, those two domains can be further exploited for seeking more particular and effective observation, such as command line strings, system call traces, and resource consumption patterns in the host audit data, or the intrinsic features, traffic features, and content features of the network packets. Based on the characterization of the data model, all techniques that are effective for distinguishing intrusions from normal behaviors are worthy of consideration. Up to now, techniques drawn from statistics [11, 31, 32], data mining [18], pattern recognition [6], machine learning [13,25], and other research fields have effectively been applied to intrusion detection.

The available approaches for intrusion detection focus on improving detection accuracy and restraining false alarms, and given enough time, most of them can achieve satisfactory results in terms of these criteria. However, in practice, intrusion detection is a real-time critical mission, that is, intrusions should be detected as soon as possible or at least before the attack eventually succeeds. In addition, there is usually an initial training period for an intrusion detector to characterize the observable subject's behavior, and most existing methods are based on the assumption that high quality labelled training data are readily available. This assumption severely limits their application in practice. In fact, intrusion detectors must undergo frequent retraining, to incorporate periodically new examples into the training data for classifying novel attacks and changes from normal behavior. Therefore, running time and training time should also be considered in addition to detection accuracy and false alarms when designing an effective IDS.

Various methods have been introduced for detecting intrusions at the level of privileged processes in SUN OS, because of its special properties, such as sensitivity to intrusions, stability over time, and limited range of behaviors, hence any exploitation of vulnerabilities in privileged process can give an intruder super-user status and thus commit further attacks. In [20], intrusion detection was formulated as a text processing problem based on the analogy between "system calls/processes" and "words/documents". Here, we also take system calls executed by privileged processes as observable subjects for analysis. Generally, the contributions of our work presented in this paper mainly include:

- Based on the fact that original $tf$-$idf$ (term frequency inverse document frequency) weighting model in text categorization might cause high false alarm rate in anomaly detection, a new weighting model based on the $tf$-$idf$ method is established; this new model considers the special information between different processes and sessions of computer audit data.
- Based on the assumption that training data are noisy (normal data are mixed up with anomalies or errors we do not expect) , Robust SVM [27] is employed to discriminate anomalies and normal activities. Based on the assumption that anomalies in training data are hard to attain and the number of anomalies is much smaller than that of normal activities, One-class SVM [26] is applied to identify the few anomalies from training data.
- Rejecting the assumption that high quality labelled training data is always readily available, and based on the fact that training data should be frequently updated to adapt the new normal regularity, Robust SVM and One-class SVM are modified based on the idea from Online SVM [17]. That is, training data are provided in sequence online, rather than in a batch.

After an elaborate theoretical analysis, we evaluated our methods using reformulated 1998 DARPA BSM data and compared their performance with the original algorithms based on the original $tf$-$idf$ weighting model. The results show that our modified SVMs can significantly reduce training time with better generalization performance and fewer support vectors while maintaining high detection accuracy; They thus require less computational overhead and running time and so are more desirable for real time intrusion detection. Furthermore, our modified weighting model based on the $tf$-$idf$ weighting method suppresses the false alarm rate to an acceptable level, thus guaranteeing the proposed method to be applied in practice.

The rest of this paper is organized as follows. In section 2, we review some related work on the existing intrusion detection techniques that used host audit data as observable subjects. Section 3 formulates the problem we solved and describes the data source that was used in our work together with the modelling of the data. In section 4, we introduce the effective classification method−SVM, and modify three SVMs, which have different assumptions, for online training. After the analysis of the data model and the improvement of the candidate methods, experiments were implemented to evaluate the performance of our proposed methods, which is described in section 5. Finally, our conclusions are presented in section 6.

## II. Related Work

As we know, intrusion detection can be treated as a binary concept on a domain consisting of temporal sequences of discrete, unordered elements, such as system call traces, network packet traces, and resource consumption. So far, many effective techniques have been employed to this problem domain, including multivariate model [32], Markov process [33], and discriminant analysis [1] from statistics; neural networks [9, 15] from pattern recognition; support vector machines [13, 25] from machine learning; and other clustering methods and classification methods from data mining [18, 19].

Forrest et al [8] proposed to build program profiles with short sequences of system calls executed by running privileged programs for intrusion detection, based on the assumption that sequences of system calls in an intrusion are noticeably different from those of normal operations. The reason for selecting privileged programs as subjects is that it constitutes a natural boundary for a computer, and the range of behaviors of privileged processes is limited and relatively stable over time compared to user behavior. Subsequently, many researchers applied various techniques [10,20,29] to extend and improve the work with increasingly better performance. Warrender et al [29] even argued that the choice of data stream (short sequences of system calls) is more important than the particular method of analysis, but subsequent studies did not adequately support this conclusion. Ye et al [31] investigated the frequency and ordering properties of computer audit data, showing that the frequency property of multiple audit event types in a sequence of events is necessary for intrusion detection, and that the ordering property of multiple audit events can provide additional advantages to the frequency property. However, due to the scalability problem of complex data models (e.g. higher-order stochastic models) [33], intrusion detection techniques based on the ordering property can hardly provide a feasible solution that produces good

performance with low computational overhead, especially when the intrusive audit data are mixed with the white noise of normal audit data. The frequency property, on the other hand, can provide a viable tradeoff between computational complexity and intrusion detection performance. The motivation of our work heavily based on this conclusion.

Liao et al [20] used K-Nearest Neighbor(KNN) classifier to label program behavior as normal or intrusive. Specifically, each system call in the process was treated as a word, and the collection of system calls over each program execution was treated as a document; thus the system call frequencies were used as the main property to represent program behavior. This method can be easily implemented and in general has smaller computational overhead than other techniques from statistics, data mining, etc. Using the same data model, and based on the assumption that normal cases are mixed with anomalies in the training data, Hu et al [13] applied Robust SVM [27], which can solve the over-fitting problem effectively introduced by the noise in the training data set, to intrusion detection over noisy audit data; in this situation, if an attack occurs during the training process, the undesired intrusive behavior usually is regarded as normal one, undermining the intrusion detector's accuracy [21]. However, their experiments showed that intrusion detection based on the text processing model would generate an unacceptable false positive rate, so it could hardly be applied in practice. Additionally, based on the assumption that the number of normal instances is significantly larger than that of anomalies, Eskin et al [7] proposed unsupervised anomaly detection methods with unlabelled data, and Nguyen [25] employed One-class SVM [26] to identify "outliers" amongst positive examples (normal behaviors) by treating them as negative examples(abnormal behaviors). Although detection accuracy performance was comparable to some other intrusion detection techniques, the unchanged patterns which can not reflect *concept drift* limit its application. Moreover, all the intrusion detectors we listed above are based on the strict assumption that training data are readily available with high quality.

## III. Problem Definition and Objective

This section gives a general description of intrusion detection from our perspective, and describes the base-rate fallacy of intrusion detectors, which motivates us to suppress false alarm rate as low as possible. The data source that will be used in the experiments, along with its model for characterization are also presented.

### A. Intrusion Detectors and Their Failure Curses

Generally, the trade-off between the ability to detect novel attacks and the ability to generate a low rate of false alters is the main criterion for evaluating an intrusion detector. Actually, from the functional perspective, intrusion detectors can be roughly regarded as a simple kind of inductive inference system. In this system, an incoming process $Q_i$ is regarded as a "question", while the normal or anomaly models $M_i$ that stored in the memory are regarded as "answers". Then, given a new $Q_i$, the system tries to find an appropriate answer $M_i$ so that $\mathrm{ID}(Q_i) \Rightarrow M_i$. We look for effective IDs that have the highest *a priori* - that have "accurate descriptions". In generating such IDs, some primitive IDs have to be previously defined. From probabilistic prediction we can gradually to deterministic prediction, that is, whether the current "question" is an anomaly. Due to the fact that the sample size of "$M$" is limited but the number of questions "$Q$" are infinite and long-standing, the ken and adaptability of ID is a key to answer diverse questions successfully.

Therefore, the first objective is to train IDs to be capable of learning online to adapt the changing situations, and thus construct or update corresponding "$M$". For instance, in practice, training sequences are usually not readily available with labels and high quality, especially for a computer system with reconfiguration. In such case, the ID has to be trained online with training data provided in a sequence rather than in a batch.

Furthermore, the construction and characterization of training sequences "$M$" is the next objective needs to be considered well. Existing IDs mainly focus on the ordering property (sequential property) and frequency property. Similar with the description in [11], a statistical framework can be utilized to analyze those two properties.

*Notations:*

$H(t)$: a hidden stochastic process which maps the activities of legitimate users and attackers to a finite space $S$ in terms of discrete time step "$t$"; at time step $t$, if $H(t) = 0$, means legitimate user traces is generated, if $H(t) = 1$, means attacker traces is generated, and it is transparent to the intrusion detectors.

$h(x)$: a hidden stochastic process for generating event $x$.

$O_t$: an observable subject that is captured at time step $t$, it can represent a single event or a group of events according to the specific detection method, and its generation is governed by the hidden process $H$;

$Set(O_t, w)$: a set of available subjects $O_i$ ($i$ depends on the specific anomaly detection model) with window $w$ at time step $t$.

$N(t)$: a legitimate stochastic process that is generated at time unit $t$, i.e., $H(t) = 0$;

$M(t)$: a malicious stochastic process that is generated at time unit $t$, i.e., $H(t) = 1$;

What an ID cares is the current observation $O_t$, a pair of probability distribution therefore can be considered as follows:

$$Pr\{O_t|H(t) = 1, O_{t-1}O_{t-2}...O_1, t\}$$
$$Pr\{O_t|H(t) = 0, O_{t-1}O_{t-2}...O_1, t\}$$

if we do not consider the specific property of the observations $\{O_{t-1}O_{t-2}...O_1\}$, the above two probability distribution can be generalized as follows:

$$Pr\{O_t|H(t) = 1, Set(O_t, w), t\}$$
$$Pr\{O_t|H(t) = 0, Set(O_t, w), t\}$$

in such case, a posterior probability of intrusion detection can be given as:

$$\begin{aligned} &Pr\{H(t)=1|O_t, Set(O_t,w),t\} \\ &= \frac{Pr\{O_t|H(t)=1, Set(O_t,w),t\}\cdot(1-\lambda)}{Pr\{O_t, Set(O_t,w),t\}} \end{aligned} \quad (1)$$

as $Pr\{O_t, Set(O_t,w),t\} = Pr\{O_t|H(t)=0, Set(O_t,w),t\}\cdot \lambda + Pr\{O_t|H(t)=1, Set(O_t,w),t\}\cdot(1-\lambda)$, equation (1) can be simplified as:

$$Pr\{H(t)=1|O_t, Set(O_t,w),t\} = \frac{c\cdot(1-\lambda)}{c\cdot(1-\lambda)+\lambda} \quad (2)$$

where $\lambda = Pr\{H(t)=0, Set(O_t,w),t\}$, represents a *p*riori probability of the legitimate pattern which contains $w$ consecutive events that has been generated by $h(x)$, and an unknown constant

$$c = \frac{Pr\{O_t|H(t)=1, Set(O_t,w),t\}}{Pr\{O_t|H(t)=0, Set(O_t,w),t\}}$$

for equation (2), $Pr\{H(t)=1|O_t, Set(O_t,w),t\} > \alpha$ iff $c > \alpha\lambda/(1-\alpha)(1-\lambda)$. Thus it is easy to find that the performance of IDs is related directly with the value of $Pr\{H(t)=1|O_t, Set(O_t,w),t\}$, and it increases with the value of $c$. Based on the equation, a simple intrusion detection model can be defined as:

$$\tilde{ID}(O_t) = c, \quad or \quad ID(O_t) = \begin{cases} 0 & \text{if } \tilde{ID}(O_t) < \alpha \\ 1 & otherwise \end{cases} \quad (3)$$

Obviously, due to the lack of prior knowledge about $\lambda$, and $c$, it is almost impossible to carry the detection model into practice directly. Moreover, a good estimates of $\lambda$ and a thorough understanding of distributions of the processes $N(t)$ and $M(t)$, which we call system normality, are not readily available, which make the detection task deem to be $NP$-hard. From this point of view, no matter ordering property or frequency property of $O_t$, the ultimate goal is to characterize the observation normality as perfect as possible. For purposes of this paper, we explore and develop Support Vector Machine as an effective frequentist estimator to characterize and identify system anomalies.

In addition, based on the fact that the number of normal activities is several orders of magnitude larger than that of anomalies in our daily computer activities, Axelsson [2] gave an analysis of intrusion detector's base-rate fallacy using *Bayesian Theorem*. He pointed out that the false alarm rate is the limiting factor for the performance of an IDS, and thus the false alarms should be suppressed as few as possible in order to achieve substantial values of the Bayesian detection rate $P(Intrusion|Alarm)$. In practice, excessive alarms from normal activities would make the network supervisor insensitive and intrusion detector inefficient, which is a straightforward motivation for us to restrain false alerts to an acceptable level.

### *B. Attributes of Data Source*

As we mentioned in the last section, a computer network typically includes two kinds of objects−hosts, and communication links. Therefore, network traffic data and host audit trails are two main observations for capturing activities. In this study, we select the benchmark−1998 DARPA data set [23] as our experimental data. The data is provided by the 1998 DARPA Intrusion Detection System Evaluation Program, and it contains a large sample of computer attacks embedded in normal background traffic. TCPDUMP and BSM [28] (Basic Security Module) audit data were collected on a simulation network that simulated the traffic of an air force local area network, the set consists of seven weeks of training data and two weeks of testing data.

TCPDUMP contains data network packets travelling over communication nets, while BSM captures activities occurring on a host machine, based on the execution records of system calls by all processes launched by users. Most traces of attacks are revealed both in TCPDUMP and BSM audit data. In our study, BSM audit data from UNIX-based host machine (SUN Solaris OS) is selected as the subject for detecting anomalies. Based on the assumption that actions in the user space can not harm the security of the system and the security-related activities that can impact the system only happen when users request services from the kernel, BSM monitors the events related to the system security and records both the instructions executed by the processor in the user space and instructions executed in the system kernel. Actually, a full system call trace gives us overwhelming information, whereas the audit trial provides a limited abstraction of the same information, such information as memory allocation, internal semaphores, and consecutive files reads do not appear. And in fact, there is usually a straightforward mapping of audit events to system calls. BSM records the execution of system calls by all processes launched by users and it also contains other detailed information about events in the system, such as user and group login identification, file names with attributes and full path, command line arguments, return code etc. In our study, we only use the names of system calls and ignore other attributes. Former studies [12] showed that privileged processes in UNIX are a good level to focus on because exploitation of vulnerabilities in privileged process can give an intruder superuser status and thus commit further attacks, and the range of behaviors of privileged processes is limited compared to that of users. Therefore, we choose system calls executed by privileged processes rather than user profiles as the observable subject. Additionally, instead of establishing privileged process profiles by short sequences of system calls, we characterize the privileged processes using the frequencies of system calls. Due to the fact that the number of system calls is limited, and based on the assumption that intrusion detection can be considered as a binary categorization problem, models and methods from the text categorization domain can be employed in a straightforward manner.

### *C. Data Model*

When the connection is established between two hosts, several sessions are generated and then many processes are executed during the connection. The atomic element of our observation is system calls, which are executed by privilege programs. Using the text processing metaphor, each sys-

tem call is treated as a "word" and the set of system calls generated by a process is treated as the "document" [20]; all the training processes are treated as a set of documents.

### C.1 Analysis of the Original Data Model

Based on the analogy between program processes and documents, the simple frequency weighting method and *tf-idf* (term frequency inverse document frequency) weighting method can be applied to transfer a process into a vector. The simple model is established as follows:

Matrix $A = a_{ij}$, the collection of processes from different sessions, and $a_{ij}$ is the weight of system call $i$ in process $j$.

$f_{ij}$, the frequency of system call $i$ in process $j$.

$N$, the number of processes in the collection.

$M$, the number of distinct system calls in the collection.

$n_i$, the number of times that system call $i$ appears in the collection.

Thus, frequency weighting is defined as:

$$a_{ij} = f_{ij} \tag{4}$$

*tf-idf* weighting method is defined as:

$$a_{ij} = \frac{f_{ij}}{\sqrt{\sum_{l=1}^{M} f_{lj}^2}} \times log(\frac{N}{n_i}) \tag{5}$$

Based on the data model, several text categorization methods were proposed [13,20] for intrusion detection. Although these methods are easy to implement and effective for detecting intrusive processes with satisfactory accuracy, they are still far from ready for application in real life because of their unacceptably high false alarm rate. Careful analysis discloses the causes of generating excessive false alters: First, a session is hastily labelled as intrusive once one of its processes is detected as an anomaly; in such cases, any misclassified process would cause the whole session to be misjudged as an intrusion without discriminating other processes from the same session. Secondly, the correlations between the processes are ignored. Since most of attacks leave their traces in several processes and sessions, isolating processes might lose some essential information and thus decreases the detection accuracy and generates high false alarm rate. Additionally, some necessary time information are ignored, the incoming processes are dealt with independently, and the training data set is not updated in time. Thus it can not reflect current novel behavior in a timely fashion, leaving much space for intruders to commit attacks. With these problems in mind, we attempt to establish a new data model that considers all those aspects.

### C.2 New Data Modeling

In [20], an incoming process (new document) was compared with the training processes (existing documents) after being transformed to a vector by weighting techniques, and then KNN was used to cluster the processes according to their distance, based on the assumption that processes with similar properties will cluster together in the vector space. The applied weighting techniques are traditional *tf-idf* and simple frequency weighting. Due to the limited number of system calls, dimensionality reduction techniques are unnecessary. When a connection is established between two hosts, several sessions or processes will be generated, in order to reflect the source specific differences, we add the session information (such as Source Machine or session ID, which can be regarded as the topic of documents) [3]. Accordingly, the *tf-idf* model can be improved as follows: $\overrightarrow{p} f_{s,t}(\theta)$ represents the process $p$ from session $s$ at time $t$ which includes system call $\theta$, and is updated according to the equation:

$$\overrightarrow{p} f_{s,t}(\theta) = \overrightarrow{p} f_{s,t-1}(\theta) + \overrightarrow{p} f_{s,P_t}(\theta) \tag{6}$$

where, $\overrightarrow{p} f_{P_t}(\theta)$ denotes the process frequencies in the newly added set of processes $P_t$. The process frequencies can be used to calculate weights for the system calls $\theta$ in the process $p$. The model is based on the fact that different sessions include different processes, and various processes have various system calls, consequently it reflects session-specific differences. The same system call may have different weights because it belongs to different sessions. To specify the equation (6), the weight of the system call $\theta$ in the processes $p$ can be calculated as follows at time $t$:

$$w_t(\theta, \overrightarrow{p}) = \frac{(1 + log_2 f(\theta, \overrightarrow{p})) \times log_2(N_t/n_\theta)}{Z_{\overrightarrow{p}}} \tag{7}$$

where

$f(\theta, \overrightarrow{p})$, the frequency of system call $\theta$ in the process $p$;

$N_t$ is the number of processes in the current training set;

$n_\theta$ is the number of processes that include system call $\theta$;

$Z_{\overrightarrow{p}} = \sqrt{\sum_{\theta \in \overrightarrow{p}} w_t(\theta, \overrightarrow{p})^2}$ is the 2-norm of vector $\overrightarrow{p}$.

When calculating the weights of the system calls, we apply the session-specific $\overrightarrow{p} f_{s,\theta}$ instead of $\overrightarrow{p} f_\theta$. Therefore, information about the session could be included in our method. If no training data is available at $t = 0$ for a specific session, we can set $\overrightarrow{p} f_{s,0} = 0$ for its all $\theta$ or identify other similar sessions $s'$, that is, $\overrightarrow{p} f_{s,0}(\theta) = \sum_{s'} \overrightarrow{p} f_{s',0}(\theta)$, which happens when an intrusion detector is trained online.

Additionally, based on the fact that the number of system calls in the various processes might differ, and inspired by the work reported in [12], we divide one process into several segments by a sliding window of fixed length $w$, which advances with a step $s$, and can be determined experimentally. Here we note that only the process with a length longer than $w$ is divided into overlapping segments by the sliding window. Specifically, $< P_1, P_2, ...P_w > \Rightarrow < P_1, P_2, ...P_n > [s, w]$, where $< P_i >_{1 \le i \le w}$ is a sub-episode of $< P_i >_{1 \le i \le n}$, for a process with length $l$, $m = \lfloor \frac{l-w}{s} + 1 \rfloor$ segments can derive from it, and we assume that minimal occurrence of some attacks can be detected in $[P_i, P_{i+w}]$. We only take this step if the length of the process is much longer than that of the others. After dividing, $m$ segments from the same process are all transformed into vectors and treated as individual "documents".

In practice, normal processes and abnormal processes in the training data should be updated frequently for restraining false alarms and detecting novel attacks. Therefore,

some time information should also be considered. Here, we apply a linear time model [30], which uses a time window on the historic data. We only consider the processes within the time window $m$:

$$N_{\overrightarrow{p}} = (1 - time/m) \cdot N_{\overrightarrow{p}} \quad (8)$$

The processes outside the window are not considered. Actually, at the beginning of the training, time window $m$ should large enough to include all the processes; with the increase of the number of processes, $m$ can be adjusted manually or experimentally.

A simple example is given here to illustrate the measures we proposed. Intrusive session *Eject* is a buffer overflow using an eject program on Solaris OS, which might lead to a status transition from a common user to a super user. The session consists of a series of processes:

*telnetd−login−tcsh−quota−cat−mail−cat−gcc−cpp−ccl −as−ld −ejectexploit−pwd*

actually, in this session, only *ejectexploit* is the intrusive process, and if it executes successfully, an attack might happen. The process contains following system calls:

*close, close, close, close, open, close, close, execve, open, mmap, open, mmap, mmap, munmap, mmap, close, open, mmap, mmap, munmap, mmap, mmap, close, open, mmap, mmap, munmap, mmap, close, open, mmap, close, open, mmap, mmap, munmap, mmap, close, close, munmap, pathdonf, stat, stat, open, close, open, open, joctl, lstat, lstat, close, close, close, close, close, exit*

The weight of the system calls in the session *Eject* are only considered in the collection of the processes from the same source host. If we set the sliding window at fixed length 50, and left system calls *close, close, close, close, close, exit* advance with step 5, we can derive another two processes from the current process.

The final countermeasure to minimize the false positive rate is to consider the causal relationship between different attack attempts. With such consideration, when a process is identified as intrusive, we do not immediately treat the session it belongs to as an intrusion. As described in [24], in a series of attacks in which the intruder launches earlier attacks to prepare for later ones, there are usually strong connections between the consequences of the earlier attacks and the prerequisites of the later ones, especially in "stealthy" attacks with multi-stages. For instance, *format*, the buffer overflow using the *fdformat* UNIX system command leads to root shell, contains two stages: ftp over files and then *chmod* exploit files. Thus the correlation of the attacks is formulated as a connected DAG(directed acyclic graph), $HG = (N, E)$, in which the set $N$ of nodes is a set of attacks, and for each pair of nodes $n_1, n_2 \in N$, there is a edge from $n_1$ to $n_2$ in $E$ iff $n_1$ prepares for $n_2$. Therefore, the triple $(fact, prerequisite, consequence)$ holds for an attack happen in the multi-session scenario. Based on this assumption, when an intrusive process is detected, its neighbor processes or sessions are also considered carefully instead of immediately labelling the entire session as intrusive. Suppose in a sequence of attacks, we have 4 intrusive sessions *Ipsweep, Eject, Land, Pod*. *Ipsweep* performs either a port sweep or ping on multiple host addresses, *Land* and *Pod* are Dos attacks. Assuming that *Ipsweep* prepares for *Land* and *Eject*, *Eject* prepares for *Pod*, the relationship *correlated(Eject, HG)=precedent(Eject, HG)* ∪ *subsequent(Eject, HG)* is intuitively shown in Figure 1.

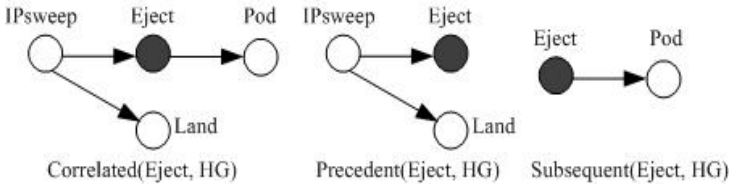


Fig. 1. Attacks correlation graph

The intrusive session *Eject* is identified as an intrusion for the malicious process *ejectexploit*. Actually, when obviously malicious processes appear, such as *formatexpolit, ffbexploit, ejectexploit*, the session should be interrupted as soon as possible. However, some intrusive processes are not obvious enough; for example, the denial of service attack *process table*, which consists of abuse of a legal activity, can hardly be identified because of its normal individual process. In order to detect such attacks effectively, the correlation between neighboring processes within a time window $T$ and the precedent attacks should also be considered.

## IV. Online Training of Support Vector Machines Based on Text Processing Model

Most of the available intrusion detection techniques were evaluated using a labeled high quality training data set, and the data set was unchanged once attained. However, in practice, training data is not readily available, and intrusion detectors must undergo frequent training for capturing novel attacks and adapting to changes in normal behaviors. After transforming ongoing processes into vectors based on the data model presented in the last section, we applied the effective binary classification method, support vector machine, to distinguish anomalies from normal activities. In this section, we first briefly introduce conventional SVM, RSVM, and One-class SVM that based on different assumptions, and then modify these methods by a general algorithm drawn from Online SVM. A theoretical analysis of the modified method is also given.

### A. Three SVMs with Different Assumptions

SVM is an approximate implementation of the Structure Risk Minimization principle based on statistical learning theory rather than the Empirical Risk Minimization method, in which the classification function is derived by minimizing the Mean Square Error over the training data set such that the maximum width of the margin between the classes can be achieved [4]. In order to solve various problems effectively, several improved SVM such as Robust SVM [27], One-class SVM [26], Online SVM [17] have been proposed. We give a brief mathematical description of these SVMs here; more detailed descriptions can be found in the corresponding reference.

Given a training sample: $D_l = \{x_i, y_i\}_{i=1}^{l}$, $x_i$ is the $i$th input vector, $x_i \in R^n$, $y_i \in [+1, -1]$, $l$ is the total number of input vectors and $n$ is the dimension of the input space. Suppose the relation between $x$ and $y$ is $y = sgn(f(x)+\varepsilon)$, where $sgn(x) = 1$, if $x \geq 0$ and $sgn(x) = -1$, if $x < 0$, the task uncovering function $f$ is called classification. SVC is a maximization(minimization) algorithm used to identify a set of linear separable hyperplanes in the feature space whose formula like $f(x) = \langle w, x\rangle + b$, and $2/\|w\|$ can be regarded as a canonical representation of the separating hyperplane. Maximization of the margin between the positive examples and negative examples can be transferred to the following problem:

$$\begin{cases} min & \frac{1}{2}\|w\|^2 \\ s.t. & y_i(\langle w, x_i\rangle + b) \geq 1 \quad \forall i, \end{cases} \tag{9}$$

By applying the Lagrangian multiplier, the problem can be formulated as:

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \beta_i y_i(\langle w, x_i\rangle + b) + \sum_{i=1}^{l} \beta_i, \tag{10}$$

The dual objective function is given below and the optimization problem is:

$$\begin{cases} min & L_D = \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \beta_i\beta_j y_i y_j \langle x_i, x_j\rangle - \sum_{i=1}^{l}\beta_i \\ s.t. & \sum_{i=1}^{l} \beta_i y_i = 0, \beta_i \geq 0, \forall i \end{cases} \tag{11}$$

Above equations only describe linear separable SVMs, and the general dual objective function can be rewritten in a matrix form as follows [17]:

$$L_D = \frac{1}{2}\beta^T K\beta - \langle c, \beta\rangle, \tag{12}$$

where $c$ is an $l \times l$ vector, $\beta = \{\beta_1, ..., \beta_l\}$ and $K = \{K_{ij}\}$, $K_{ij} = y_i y_j K(x_i, x_j)$, while $K(x_i, x_j)$ is called kernel function, which can be selected such formulas as $K(x_i, x_j) = \langle x_i, x_j\rangle^d$ or $K(x_i, x_j) = e^{\|x_i - x_j\|/\sigma}$. The feasible solution of Eq.(11) should satisfy the KTT [4] conditions as follows:

$$\begin{cases} \beta_i = 0 \Leftrightarrow y_i f_i > 1, \\ 0 < \beta_i \leq C \Leftrightarrow y_i f_i \leq 1, \end{cases} \tag{13}$$

The hyperplane $f(x)$ can be expanded from the kernel as follows:

$$f(x) = sgn\left(\sum_{i \in SV} \beta_i y_i K(x, x_i) + b\right). \tag{14}$$

In order to solve the over-fitting problem of a soft margin SVM due to noisy training data, Robust SVM [27] minimizes only the margin of the weight $w$ instead of minimizing the margin and the sum of misclassification errors. The objective function can be written as following:

$$\begin{cases} min & L_D = \frac{1}{2}\beta^T K\beta - \langle c, \theta\rangle \\ s.t. & \sum_{i=1}^{l} \beta_i y_i = 0, \beta_i \geq 0, \forall i, \end{cases} \tag{15}$$

where $\theta = \langle \gamma, \beta\rangle$, $\gamma = \{\gamma_1, ...\gamma_l\}$, and $\gamma_i = 1 - \lambda D^2(x_i, x^*_{y_i})$, $\lambda \geq 0$ is a pre-determined regularization parameter measuring the influence of averaged information(distance to the class center), and $D^2(x_i, x^*_{y_i})$ represents the normalized distance between data point $x_i$ and the center of the corresponding classes, $(x^*_{y_i}, y_i \in \{+1, -1\})$, in the feature space. The slack variable $\lambda D^2(x_i, x^*_{y_i})$ can be justified by considering it as part of the margin. Because of this term, the RSVM algorithm will have fewer support vectors and the decision boundary will be smoother.

Another adapted algorithm, called one-class SVM algorithm, identifies "outliers" amongst positive examples and uses them as negative examples. After mapping between input data space $X$ and high-dimensional feature space $H$ via a kernel, origin is treated as the only member of the second class. Then "relaxation parameters" is used to separate the point of the first class from the origin. As a comparison with the above algorithms, we can write the objective function as:

$$\begin{cases} min & L_D = \frac{1}{2}\beta^T K\beta \\ s.t. & 0 \leq \beta_i \leq \frac{1}{vl}, \sum_{i}^{l} \beta_i = 1, \end{cases} \tag{16}$$

where $v \in (0, 1)$ is a parameter that controls the trade-off between maximizing the margin from the origin and containing most of the data in the region generated by the hyperplane. The general decision function with kernel expansion is:

$$f(x) = sgn\left(\sum_{i=1}^{l} \beta_i k(x_i, x) - \rho\right). \tag{17}$$

If $\alpha_i$ meets the subject conditions, $\rho$ can be recovered as:

$$\rho = \sum_{j=1}^{l} \beta_j k(x_j, x_i). \tag{18}$$

Generally, two facts usually hold during the intrusion detection process. One is that training data is always mixed with noisy data, which thus decreases the capability of detectors to capture anomalies with high accuracy and increases the probability to generate false alarms. The second is that the number of anomalies is much smaller than that of normal activities, which thus motivate us to apply robust SVM and one-class SVM respectively.

### *B. Modified SVMs for Online Training*

The SVMs mentioned above are used for the classification of input data that are supplied and computed in batch.

It is time consuming to classify a large data set and thus these SVMs can not meet the demands of online applications, especially for intrusion detection, which needs periodical retraining. Online SVM [17], on the other hand, have input data supplied in sequence rather than in batch, and the experiments showed it has fewer support vectors and faster convergence than the conventional SVC. Here we would modify SVMs we discussed in the last section using the method from OSVM.

As we know, in the original SVMs a batch of training data are extracted as vectors and classified by solving the quadratic programming problems $Eq.(13, 16, 17)$. The number of elements determines the dimensionality of the vectors. A final hyperplane can be achieved after computing the objective functions. Now let us consider another case, that training data can not be acquired at one time or supplied in a sequence.

For supervised SVMs, i.e., conventional SVM and Robust SVM, we can give one example for each class, the hyperplane with a maximum margin for these two examples can be found by solving the objective function $Eq.(13, 16)$. When a new example becomes available, corresponding to the KKT conditions [4], two cases need to be considered, that is, whether or not the current optimal boundary can classify the new example correctly. If it can be classified, then the example is not a support vector, otherwise, a new hyperplane should be determined so that it can classify three examples. The new hyperplane can be found by minimizing the objective function with the SVs obtained from the current hyperplane and the new example. At the $k$th step, the set of SVs can be denoted as $SV_k$, and the existing examples are $\{Sx_i^k, Sy_i^k\}_{i=1}^{|SV_k|}$ respectively. The corresponding hyperplane is (conventional SVM):

$$f_k(x) = sgn\left(\sum_{i=1}^{|SV_i|} \beta_i^k Sy_i^k K(x, Sx_i^k) + b_k\right). \qquad (19)$$

Once the hyperplane is updated, the KKT conditions are checked for all $k$ examples, and the examples which violate the KKT conditions are fed to the algorithm as new examples. With reference to Online SVM [17], we rewrote a general algorithm for the three SVMs described above to improve the performance of their training phase:

**Algorithm of online training for SVMs**

```
void OnlineTraining( )
{
  set W_1 = {x_k, y_k}, for k = 1, 2, and |E_2| = 0,
  // or k = n, n+1 and |E_{n+1}| = 0
  Minimize Eq.(4, 7, 8) with W_1 to obtain an optimal
  boundary f_1.
  for (int k = 3; k <= l; k++) {
  // or k = [n+3, ..., n+l]
    Obtain a new element S_k = {x_k, y_k};
    if (S_k can be distinguished by f_{k-1}) {
      Add S_k to the corresponding class;
    }
    else {
      W_k = {Sx_i^{k-1}, Sy_i^{k-1}}_{i=1}^{|SV_{k-1}|} ∪ S_k;
      Minimize Eq.(4, 7, 8) with W_k to obtain a new
      optimal boundary f_k;
    }
    if (|E_k| = |{x_i, y_i | y_i f_k(x_i) violates the KKT
      conditions }_{i=1}^{k}| > 0){
      E_k be input next step as new elements;
    }
  }
  while (|E_l| > 0) {
    Minimize Eq.(4, 7, 8) with W_l = SV_l ∪ E_l to obtain
    an optimal boundary f_l;
  }
}
```

As described in the algorithm, we can give more than one example for each class (normal and anomaly) at first, that is, the process can start at any $k$th steps, thus some typical attacks can be kept, meanwhile the algorithm can learn to detect novel attacks. However, because of computational overhead, the number of SVs, $n$, for the existing examples should not be too large, otherwise, this algorithm can perform little better than the conventional training methods. Because of its unsupervised nature, One-class SVM only takes an original point and another normal behavior at its initial training stage for subsequent classification, while anomaly points are not necessary. Some other accelerated training algorithms [14] and decomposition algorithms are also worth considering in order to speed up the training phase of SVMs. The prove of the convergence of those modified SVMs is shown in the appendix.

## V. Performance Evaluation of Proposed Methods

In section 3, we briefly described our data source. To evaluate our proposed methods, we reformulated the training data and testing data based on the benchmark data set. We apply our three modified SVMs to the selected data, and compare the results with those of the original algorithms. The data are provided in sequence to SVMs as our need instead of in batch as the raw data format. Furthermore, the modified SVMs are based on our proposed weighting model, while the original SVMs are based on original $tf$-$idf$ weighting method. Actually, based on the same weighting method, original SVMs and modified SVMs can be compared, but we did not do that for it has little contribution to evaluate our new methods. The specific implementation procedure is shown in Figure 2.

### A. Training Data and Testing Data

According to the attributes of the data source, preprocessing of the DARPA data and feature(characteristics of system calls) extraction from those data sets are necessary before employing the data model and the techniques we proposed. Basic Security Module(BSM) audit data collected from a victim Solaris machine in a simulation network by DARPA Intrusion Detection System Evaluation

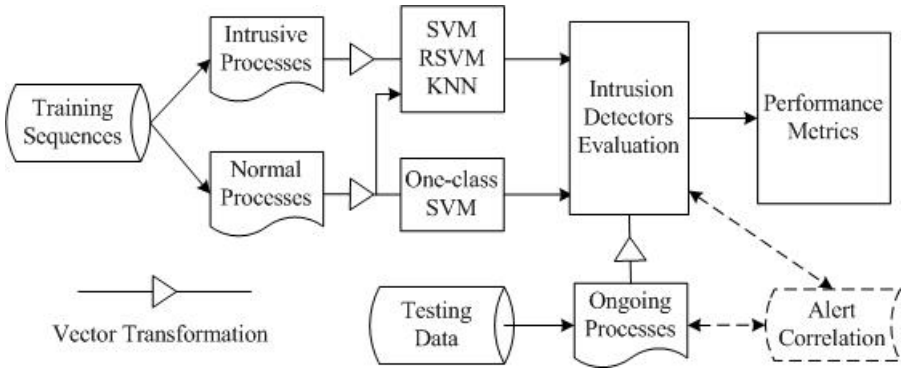


Fig. 2. Evaluation procedure of the Proposed Intrusion Detectors

System is used as the experiment data here, and only the name of system calls are considered; other attributes related to them are ignored. Each session, which consists of a number of processes, corresponds to a TCP/IP connection between two hosts, and each of them was labelled with numbers (session ID). A text categorization problem based on our weighting model is formulated, and the techniques we proposed in the previous section are applied to solve it. Any attacks or anomalies during the execution of processes attempted to detect them immediately, thus guaranteeing the intrusion detection in real time.

Actually, the 1998 DARPA data has been widely criticized for some hidden factors in it [22], which might have a direct effect on the quality of the results, and thus there is always the doubt the IDSs would well in real environments with diverse and dynamic traffic backgrounds. Following reasons need to be claimed for our application of this evaluation framework:

- It is usually hard to accumulate substantial intrusion detection data due to personal privacy, considerable recourse costs, and long-term period, which is also the reason that there are only several ID benchmark datasets are available.
- The existing 1998 DARPA data has been widely applied during past 6 years, its structure and attributes are well known in ID community, which thus greatly simplify the data preprocessing stage.
- Although the construction of synthetic data is a possible evaluation approach, the results can not be compared with other methods that use different data sets, and hence undermine its credibility.
- Although the amount of data is limited, after preprocessing and reformulation as experimental demands, it is general enough to evaluate our proposed methods, and actually, it is not such a good witness that can vindicate the merits of our methods adequately. On the other hand, however, as all the other methods, we cannot rule out the probability that our proposed algorithms tend to yield worse results under real conditions before they are really put into practice, in this sense, a real experimental prototype is desirable and helpful.

The benchmark data set provides 9 weeks audit data altogether (7 weeks are labelled training data, 2 weeks are unlabelled testing data). In addition, in order to compare our methods with KNN classifier, similar preprocessing steps of data as report in [20] is carried out:

1. There are 5 out of 35 (seven-week training) simulation days free of attacks; 4 out of these 5 days are picked arbitrarily as training data, the left one day data is taken as the normal part of testing data.
2. There are total 606 *distinct* processes drawn from more than 2000 sessions running on the victim Solaris machine during the selected four training days, all these processes are picked as the normal part of training data; There are total 40 attack instances (hidden in more than 55 sessions) in the seven-week training data, and 30 intrusive processes among those intrusive sessions (cover most of the attack types in the training data, such as "*eject, spy, ffb, ipsweep...*") are selected as anomaly part of training data.
3. The left one day data in step 1 (3rd day of week 7th) contains 412 sessions (total 5285 processes), and all these processes are taken as the normal part of testing data (session information are included); 24 attacks (16 are known, 8 are novel) from two-week DARPA testing data are taken as anomaly part of our testing data.

It is worth noting that an intrusive session may contain only a small part of intrusive processes or even only one, such as *eject, format, ffb, spy* and so on. Therefore, 55 intrusive sessions do not mean there are 55 attacks. In fact, there are 40 clear (components of the attacks are visible in $BSM$ data) or stealthy (components of the attack in the audit data are hide by encryption, by spreading the attack over multiple sessions or by other techniques) attack instances included in more than 55 intrusive sessions, representing all types of attacks and intrusion scenarios in the seven-week training data.

To evaluate our proposed methods that are based on different assumptions, two data sets are formulated here; one is taken as clean data, and another is taken as noisy data. As above reformulation, details of the training and testing data for those two data sets are illustrated in Table 1. Note that training data of the noisy data set takes only 15 out of the original 30 intrusive processes as anomalies, and the remaining 15 intrusive processes are disguised as normal processes and incorporated into the truly normal ones. The reason we formulated noisy data set is to verify the performance of Robust SVM. While for One-class

SVM, we only use normal training data (i.e., 606 normal processes). The testing data for clean data set and noisy data set are same.

TABLE I
EXPERIMENT DATA SETS

| | Clean data (processes) | | Noisy data (processes) | |
|---|---|---|---|---|
| | normal | intrusive | normal | intrusive |
| Training | 606 | 30 | 621 | 15 |
| Testing | 5285 | 24 attacks | 5285 | 24 attacks |

According to our definition, when a process is determined to be intrusive, the session with which the process is associated is classified as an attack session, and each attack counts as one detection, even with multiple sessions (for those stealthy attacks). Detection accuracy is then calculated as the rate of detected attacks, and the false positive probability is defined as the rate of misclassified normal processes (these two terms are not rigorously symmetrical here). A drawback of intrusion detection using original SVMs is that, as time passes, the old hyperplane can no longer accurately distinguish normal from abnormal activities, thus the detection accuracy decreases dramatically with an increasing false positive rate. The obvious way to handle this *concept drift* [16] is to periodically retrain the SVMs, therefore, training time is another important factor to consider. Because the running time of SVM is proportional to the number of support vectors (SVs), we prefer SVs rather than time counting to measure their performance. Additionally, we also have a comparative study on the performance between our proposed methods with that of existing method−KNN classifier.

### *B. Results and Discussion*

We did experiments over clean training data and noisy training data respectively using the SVMs we presented above. All the SVMs were implemented with the RBF kernel function(i.e. $K(x_i, x_j) = e^{\|x_i - x_j\|/\sigma}$), and the best hyperplanes were obtained by varying the related regulation parameters (table II). Moreover, for our modified SVMs, training data were provided in a sequence, namely, normal processes and intrusive processes were mixed up and provided one by one, while for original SVMs and KNN, training data were provided in a batch. A comparative study were carried on the performance of various SVMs, in terms of *detection accuracy, false positive rate, the number of support vectors* and *training time*.

Although the Receiver Operating Characteristic (ROC) curve is a typical method for measuring the performance of an intrusion detection technique, it provides little insight into the performance that we intend to evaluate, and we did not employ it here because the multi-variable would make it unclear. Actually, we care most about two points in the ROC, that is, detection accuracy when false positive rate is zero and the false positive rate when detection accuracy is 100%. These two terms of three different SVMs and KNN are shown in table III by adjusting the related regulation parameters after training with clean data and noisy data respectively.

TABLE II
REGULATION PARAMETERS OF DIFFERENT METHODS

| Methods | Regulation Parameters |
|---|---|
| Traditional SVM | $\sigma$, $C$ |
| Robust SVM | $\sigma$, $\lambda$ |
| One-class SVM | $\sigma$, $\upsilon$ |
| KNN | $k$, $\tau$ |
| General parameters | sliding window $w$=60, time window $T$=10 |

"$\tau$" denotes the threshold of KNN, "$T$" is the number of consecutive processes being considered, rather than the real time counting.

TABLE III
TWO SAMPLES OF THE EXPERIMENTS WITH TRAINING DATA
(Hits out of 24 attacks while no FA. and False Alerts (FA.) out of 5285 normal processes while 100% Hit)

| | Methods | Hits (%) | | FA. (%) | |
|---|---|---|---|---|---|
| | | Clean | Noisy | Clean | Noisy |
| SVM | Original | 54.2 | 58.3 | 12.3 | – |
| | Modified | 58.3 | 58.3 | 11.1 | – |
| RSVM | Original | 70.8 | 50.0 | 3.8 | 8.7 |
| | Modified | 70.8 | 54.2 | 3.8 | 8.7 |
| KNN | Original | 16.7 | 12.5 | 9.9 | – |
| | Modified | 20.8 | 12.5 | 9.9 | 11.1 |
| One-class SVM | Original | 70.8 | 70.8* | 10.1 | 10.1* |
| | Modified | 75.0 | 75.0* | 9.8 | 9.8* |

"–" means the value is unavailable, "*" means the value unchanged.

Following conclusions therefore can be derived from the results that shown in table III:

- Online training did not cause the detection accuracy deterioration of SVMs; instead, new weighting method sometimes improved the detection performance. For example, original KNN can not detect all the attacks hidden in testing set with noisy data (a DoS attack named *process table* cannot be detected due to its normal appearance), but it can do that with some false alerts using the new weighting method (considering the correlation between a particular time window). In addition, keeping zero false positive rate, a conventional SVM can detect 13 out of 24 attacks with clean training data (i.e. detection accuracy is 54.2%), while the modified SVM can detect 14 attacks; the modified RSVM and One-class SVM also had more hits than the original methods.
- All the methods experienced performance deterioration with noisy training data (One-class SVM had no change because it was trained with only normal data). To get 100 percent hits, the conventional SVM misclassified 12.3% of normal processes as intrusive ones with clean training data, while with noisy data, the conventional SVM could not detect all the intrusive processes until all the normal processes were misclassified as intrusive ones, indicating that conventional SVM has no ability to suppress the effect brought by noisy samples, neither does KNN.

• Compared with the conventional SVM, Robust SVM showed a slight decline of performance in the presence of noise, and were able to reach 100% detection accuracy while maintaining a low false positive rate.

TABLE IV
COMPARISON OF THE DETECTION ACCURACY (%) OF ALGORITHMS WHEN FP. LESS THAN 1%

| Training Data | Methods | SVM | RSVM | KNN | Oc-SVM |
|---|---|---|---|---|---|
| | Parameters | $\sigma$=8, C=10 | $\sigma$=4, $\lambda$=1.2 | $k$=10 $\tau$=0.85 | $\sigma$=10 $\upsilon$=0.005 |
| Clean Data | Original | 79.17 | 83.33 | 87.50 | 75.00 |
| | Modified | 87.50 | 83.33 | 91.67 | 83.33 |
| Noisy Data | Original | 50.00 | 70.83 | 66.67 | 75.00* |
| | Modified | 63.33 | 75.00 | 70.83 | 83.33* |

"One-class SVM is denoted by Oc-SVM, due to the limited margin."

Furthermore, as we mentioned above, intrusion detection systems usually require as low a false positive rate as possible due to the fact that too high false positive rate would make the systems ineffective. This is also the reason that most existing commercial IDSs prefer misuse ID techniques rather than anomaly ID techniques. To compare the performance of our proposed methods, the false positive rate was kept under 1% by regulating the parameters of the SVMs, and the detection accuracy of conventional SVM and RSVM were recorded over both clean training data and noisy training data (One-class SVM only was trained with normal data). The results in Table IV show that the performance of SVM, RSVM with clean training data and that of One-class SVM with normal data did not have much difference, but RSVM had better performance than traditional SVM over noisy data due to its ability to solve the overfitting problem. Additionally, the results showed that our modified methods suppress false positives effectively. For instance, the original conventional SVM achieved 79.17% detection accuracy with 0.99% false positive rate, while our modified SVM could achieve 87.50% detection accuracy with 0.75% false positive rate. Also shown in the Table IV, both modified RSVM and One-class SVM had better performance than the original ones. Surprisedly, KNN showed the best performance with cleaning data (91.67% hit with only 0.66% FA.), but it deteriorated greatly with noisy training data because of some misclassified intrusive processes.

Another factor worth considering is *the number of support vectors*. As we know, SVC classify new examples by solving a quadratic programming problem, and the computational complexity of SVCs has a linear relationship to the number of SVs, therefore, SVMs with less SVs require less running time, which significantly benefits online intrusion detection. When we derived table IV, we recorded the number of SVs of different SVMs, and as illustrated in Table V, traditional SVM and RSVM had more support vectors over the clean training data than over the noisy training data because of the misclassified elements, and the number of SVs of our proposed methods was generally less than that of the original ones. Original One-class SVM selected 48 out of 606 normal processes as its SVs, while the modified one reduce the number of SVs to 34. Unlike SVMs, KNN has to calculate the similarity distance between the ongoing process and all the processes in the training data (the size is usually huge in practice), in order to guarantee a high detection accuracy, which thus increase its running time and response time heavily.

TABLE V
COMPARISON OF THE NUMBER OF SVS OVER TRAINING DATA SETS

| Training Data | Methods | SVM | RSVM | Oc-SVM |
|---|---|---|---|---|
| Clean Data | Original | 46 | 34 | 48 |
| | Modified | 43 | 32 | 34 |
| Noisy Data | Original | 41 | 19 | 48* |
| | Modified | 36 | 19 | 34* |

Besides the detection accuracy and support vectors, another aspect that must be addressed is the *training time* of intrusion detectors. Available intrusion detection approaches rely too strongly on the assumption that high quality labeled training data can be readily obtained, which undermines their efficiency and limits their application. An ideal IDs should be trainable with any provided data, even online. Therefore, the ability to achieve satisfied detection accuracy during as short a certain training time as possible is very important for an IDS that works online. Table VI shows the ratio of the training time for original SVMs to the modified method with clean data and noisy data respectively.

TABLE VI
RATIO OF THE TRAINING TIME FOR THE MODIFIED SVMS/ORIGINAL SVMS

| Training Data | SVM(%) | RSVM(%) | One-class SVM(%) |
|---|---|---|---|
| Clean Data | 56.01 | 51.61 | 59.40 |
| Noisy Data | 65.12 | 66.67 | 60.20 |

The training time for the modified SVMs was much less than for the original ones; RSVM and One-class SVM need more training time than conventional SVM in order to get high detection accuracy with a false positive rate less than 1%. The training time for modified SVMs represents an average performance over 50 trials, and it greatly depends on the distribution of the SVs in the training sequence. During the experiment, we found that the modified algorithms converge faster to the optimal boundary if the SVs are provided to the algorithms earlier than the in other examples. However, modified algorithms deteriorated severely when abnormal points were provided after most of the normal points had been supplied, due to the sudden change of the nature of boundaries. Under such conditions, the modified algorithms perform narrowly better than the original algorithms. In our experiment, the fastest trial only takes 8.3$s$, when the normal processes and anomaly processes were provided alternately during the initial training phase,

in contrast to the worst trial, which takes $223.3s$, when some anomaly processes were supplied suddenly at the end of training stage, so we averaged the performance over 50 trials for comparison with the original algorithms.

## VI. Conclusion and Future Work

In this paper, intrusion detection was formulated as a text processing problem. Aiming to lower the high false positive rate, and based on the special characteristics of the observable subjects−system calls in privileged processes, we use a modified $tf$-$idf$ text processing model with considering the time information and the correlation between the processes, the prerequisites and consequences of the attacks, etc. In addition, we modified traditional SVM, RSVM and One-class SVM respectively, based on the method from OSVM. The preliminary experiments with the 1998 DARPA BSM audit data showed that our modified algorithms outperform conventional SVMs in terms of the number of support vectors and amount of required training time while keeping comparable detection accuracy. Specifically, the running time of the modified algorithms can be greatly reduced because of the fewer support vectors, and significant training time can be saved by the effective decomposition of the original algorithms for faster convergence. Both of these two aspects are essential to the design of an satisfactory online IDS. One significant discovery is that the modified One-class SVM can be trained online with unlabelled data sets because of its unsupervised nature, which contradicts the strong assumption that most existing methods are based on. It also inspires us to undertake further research about the application of online training with related effective unsupervised learning methods for intrusion detection, such as incremental learning methods. Although there may still be some reasons to doubt the performance of our proposed methods in practice, actually, we can not exclude causes from the limited sample of the experiment data. Moreover, we can conclude that the characterization of the observable subjects is more important than the specific method, so the effective description of the subjects is more meaningful for improving the performance of intrusion detection that uses text processing techniques. The following aspects are worth further consideration:

- Collecting more random samples particularly of intrusions from real environments, to evaluate our method. Some effective evaluation methods that offer insight into the mechanisms of anomaly detectors are worth further exploration, rather than just tallying detection accuracy with false positive rates using benchmark datasets.
- Comparing our method with other intrusion detection techniques from machine learning and pattern recognition.
- Some other effective incremental learning algorithms are worth considering to be applied to the realtime intrusion detection, and capture the system normality drift.

## Appendix

### Convergence of the Modified SVMs

The convergence of the modified conventional SVM has been proved in Ref. [17] by comparing it with the decomposition algorithm(DA). Here we only prove the convergence of modified Robust SVM and modified One-class SVM. As we know, the main idea of DA is that instead of immediately solving the large quadratic programming problem, small quadratic programming sub-problems are iteratively solved, and thus the iteration solution of the sub-problem brings the solution closer to the optimal solution. The training set is decomposed into two subsets, working subset $B$ and correcting subset $N$. At each step, $m$ elements exchange between the subset $B$ and $N$. With the elements exchanged, the sub-problem involving the new working set is solved. The exchange between $B$ and $N$ repeats until no example violates the KKT conditions. Note that $m$ is pre-determined as a constant, and the size of $B$ and $N$ are arbitrarily determined. The convergence of the DA for standard SVC and Robust SVC has been proved in Refs. [5] and [14] respectively. Similarly, the dual objective function $Eq.(17)$ of One-class SVM can be rewritten involving the working and correcting sets as follows:

$$\begin{cases} min & \frac{1}{2}[\beta_B^T K_{BB}\beta_B + \beta_B^T K_{BN}\beta_N + \beta_N^T K_{NB}\beta_B + \beta_N^T K_{NN}\beta_N] \\ s.t. & 0 \le \beta_B \le \frac{1}{vl}, \beta_B + \beta_N = 1. \end{cases} \tag{20}$$

where

$$\beta = \begin{pmatrix} \beta_B \\ \beta_N \end{pmatrix}, y = \begin{pmatrix} y_B \\ y_N \end{pmatrix}, K = \begin{pmatrix} K_{BB} & K_{BN} \\ K_{NB} & K_{NN} \end{pmatrix}.$$

All the DA of these different SVMs are based on the following two propositions.

**Proposition 1** *Moving a variable from $B$ to $N$ leaves the cost function unchanged, and the solution is feasible in the sub-problem.*

**Proposition 2** *Moving a variable that violates the KKT condition from $N$ to $B$ gives a strict improvement in the cost function when the sub-problem is re-optimized.*

Proposition 1 means that the objective functions of SVMs can be decomposed by subset $B$ and subset $N$, while the value of the cost function is unchanged. With proposition 2, the solution of sub-problem is improved when an element violating the KKT conditions is moved from $N$ to $B$. The difference between our modified SVMs and DAs is that modified SVMs keep $SVs$ in the working subset and move the other elements to the correcting subset, and thus $\beta_N$ is a zero column vector. In addition, modified SVMs move at least one element which violates the KKT conditions to the working subset at each step; the element can be either a new one just obtained or moved from the correcting set. Therefore, solving the sub-problem will make a improvement at each step. The following corollary given by Lau et al [17] shows that keeping the $SVs$ in the working set will not affect the optimal solution, and we attempt to prove that it also holds for both Robust SVM and One-class SVM.

**Corollary** *Moving an element which is not an SV from $B$ to $N$ leaves the cost function unchanged and the solution is feasible in the sub-problem.*

**Proof.** Suppose $B' = B - \{m\}, N' = N \cup \{m\}, \{m\} \in$

$B - SV$, where"$-$"denotes set substraction, $m$ represents an element which is not $SV$.

(1) For robust SVM, we have

$$\begin{aligned}
&L_D(\beta_{B'}, \beta_{N'}) \\
&= \tfrac{1}{2}[\beta_{B'}^T K_{B'B'}\beta_{B'} + \beta_{B'}^T K_{B'N'}\beta_{N'} + \beta_{N'}^T K_{N'B'}\beta_{B'} \\
&\quad + \beta_{N'}^T K_{N'N'}\beta_{N'}] - \gamma(\beta_{B'}^T + \beta_{N'}^T) \\
&= \tfrac{1}{2}[\beta_{B'}^T K_{B'B'}\beta_{B'} + 2\beta_{B'}^T K_{B'N'}\beta_{N'} + \beta_{N'}^T K_{N'N'}\beta_{N'}] \\
&\quad - \gamma(\beta_{B'}^T + \beta_{N'}^T)
\end{aligned}$$

The optimization problem can be formulated as follows:

$$\begin{cases} min & L_D(\beta_{B'}, \beta_{N'}) \\ s.t. & < y_{B'}, \beta_{B'} > + < y_{N'}, \beta_{N'} >= 0, \\ & -\beta_{B'} \leq 0. \end{cases} \tag{21}$$

(2) Similarly, for One-class SVM, we have

$$\begin{aligned}
&L_D(\beta_{B'}, \beta_{N'}) \\
&= \tfrac{1}{2}[\beta_{B'}^T K_{B'B'}\beta_{B'} + \beta_{B'}^T K_{B'N'}\beta_{N'} + \beta_{N'}^T K_{N'B'}\beta_{B'} \\
&\quad + \beta_{N'}^T K_{N'N'}\beta_{N'}] \\
&= \tfrac{1}{2}[\beta_{B'}^T K_{B'B'}\beta_{B'} + 2\beta_{B'}^T K_{B'N'}\beta_{N'} + \beta_{N'}^T K_{N'N'}\beta_{N'}]
\end{aligned}$$

The optimization problem can be formulated as follows:

$$\begin{cases} min & L_D(\beta_{B'}, \beta_{N'}) \\ s.t. & 0 \leq \beta_{B'} \leq \frac{1}{vl}, \\ & \beta_{B'} + \beta_{N'} = 1. \end{cases} \tag{22}$$

From Proposition 1, we know that the objective function $L_D(\beta_{B'}, \beta_{N'}) = L_D(\beta_B, \beta_N)$. We note that $N^{'}$ does not contain any $SV$. Hence, $\beta_{N'} = \mathbf{0}$, for its elements are not SVs, and thus $L_D(\beta_B, \mathbf{0}) = L_D(\beta_{B'}, \mathbf{0})$, where $\mathbf{0}$ is a column vector whose all elements are 0. In addition, we have $\beta_B^T y_B = \beta_{B'}^T y_{B'} = 0$ and the bound constraints of robust SVM are unaffected, and obviously, the bound constraints of one class SVM are unaffected also. Therefore, both the sub-problem of Robust SVM and One-class SVM have the same solution with their corresponding proposed algorithms which modify Proposition 1 but keep Proposition 2 of the DA. □

## Acknowledgements

This research is conducted as a program for the "Fostering Talent in Emergent Research Fields" in Special Coordination Funds for Promoting Science and Technology by Ministry of Education, Culture, Sports, Science and Technology.

## References

[1] Midori Asaka, Takefumi Onabuta, Tadashi Inoue, Shunji Okazawa and shigeke Goto, "A New Intrusion Detection Method Based on Discriminat Analysis," *IEICE Trans. INF.and SYST.,* Vol.E84-D, No.5 May 2001.

[2] Stefan Axelsson, "The Base-Rate Fallacy and the Difficulty of Intrusion Detection," *ACM Transaction on Information and System Security,* Vol.3, No.3, August 2000, Pages 186-205.

[3] Thorsten Brants, Francine Chen, Ayman Farahat, "A System for New Event Detection," *SIGIR'03, July 28-August 1, 2003,* Toronto, Canada.

[4] Christopher J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery,* 2, 121-167(1998).

[5] C.C.Chang, C.W.Hsu, C.J.Lin. "The analysis of decompostion methods for support vector machines," *IEEE Trans. Neural Networks,* 11(2000), 1003-1008.

[6] V.N.P.Dao and V.R.Vemuri. "A Performance Comparison of Different Back Propagation Neural Networks Methods in Computer Network Intrusion Detection," *Differential Equations and Dynamical Systems,* vol 10, No 1&2, pp201-21, Jan&April, 2002.

[7] E.Eskin, A.Arnold, M.Prerau, L.Portnoy and S.Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data," *Applications of Data Mining in Computer Security,* Kluwer, 2002.

[8] Forrest,S.,Hofmeyr,S.A.,&Longstaff,T.A, "A sense of self for UNIX processes," *In proceedings of 1996 IEEE Symposium on Security and Privacy,* Los Alamitos, CA: IEEE Computer Society Press.

[9] Giorgio Giacinto, Fabio Roli, Luca Didaci, "Fusion of multiple classifiers for intrsuion detection in computer networks," *Pattern Recognition Letters,* 24 (2003) 1795-1803.

[10] A.K.Ghosh, A.Schwartbard and A.M.Shatz. "em Learning Program Behavior Profiles for Intrusion Detection," *Proceedings of 1st USENIX Workshop on Intrusion Detection and Network Monitoring,* pp51-62, Santa Clara, CA, 1999.

[11] Paul Helman and Gunar Liepins, "Statistical Foundataions of Audit Trail Analysis for the Detection of Computer Misuse," *IEEE Transaction on Software Engineering,*Vol.19, No.9, September 1993.

[12] Steven A. Hofmeyr, Stephanie Forrest, Anil Somayaji, "Intrusion Detection using Sequences of System Calls," *Journal of Computer Security,* Page:151–180, 1998.

[13] Wenjie Hu, Yihua Liao, V.Rao Vemuri, "Robust Support Vetor Machines for Anomaly Detection in Computer Security," *The 2003 International Conference on Machine Learning and Applications (ICMLA'03)*, Los Angeles, California, June 2003.

[14] Wenjie Hu and Qing Song, "An Accelerated Training Algorithm for Robust Support Vector Machine, *IEEE Transaction on Circuits and Systems I: Fundamental Theory and Applications,* 2003.

[15] Daejoon Joo, Taeho Hong, Ingoo Han, "The neural network models for IDS based on the asymmetric costs of false negative errors and false positive errors," *Expert Systems with Applications,* 25, (2003) 69-75.

[16] Klinkenberg,R., and Joachims,R, "Detecting concept drift with support vector machines," *In Langley, P.,ed., Proceedings of ICML-00, 17th International conference on Machine Learning,*487-494. Stanford, US: Morgan Kaufmann Publishers, San Francisco, US.

[17] K.W.Lau, Q.H.Wu, "Online training of support vector classifier," *Pattern Recognition,* 36(2003), 1913-1920.

[18] Wenke Lee and Salvatore J.Stolfo, "A Framework for Constructing Features and Models for Intrusion Detection Systems," *ACM Transactions on Information and System Security,* Vol.3, No.4, Novermber 2000, Pages 227-261.

[19] Wenke Lee, Stolfo,S.J.,&Mok,K.W. "Mining audit data to build intrusion detection models," *In Proceedngs of the Fourth International Conference on Knowledge Discovery and Data Mining(pp.66-72),* Menlo Park, CA:AAAI Press.

[20] Y. Liao and V. R. Vemuri, "Use of K-Nearest Neighbor Classifier for Intrusion Detection," *Computers & Security,* 21(5), pp439-448, Oco, 2002.

[21] E. Lundin and E. Jonhsson, "Anomaly-based intrusion detection: privacy concern and other problems," *Computer Networks,* Vol.34, pp623-640,2000.

[22] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory," *ACM Transactions on Information and System Security,* Vol.3, No.4, Pages 262-294, Nov.2000.

[23] MIT Lincoln Laboratory, *http://www.ll.mit.edu/IST/ideval/.*

[24] Peng Ning, Yun Cui, Douglas S. Reeves, "Constructing Attacks

Scenarios through Correlation of Intrusion Alters," *CCS'02,* November 18-22, 2002, Washington, DC, USA.

[25] Binh Viet Nguyen, "An Application of Support Vector Machines to Anomaly Detection," *Research in Computer Science−Support Vector Machine, report,* Fall 2002.

[26] Scholkopf,B. Platt,J.C. Shawe-Taylor,J. Smola, A.J. and Williamson,R.C.2001, "Estimating the support of a high-dimensional distribution," *Neural Computation* 13(7):1443-1471.

[27] Qing Song, Wenjie Hu and Wenfan Xie, "Robust Support Vector Machine for Bullet Hole Image Classification," *IEEE Transaction on Systems, Man and Cybernetics Part C.* Vol 32. Issue 4, pp440-448. Nove.2002.

[28] Sun Microsystems, *SunShield Basic Security Module Guide, 1995.*

[29] C.Warrender, S.Forrest and B.Pearlmutter, "Detecting Intrusion Detection Using System Calls: Alternative Data Models," *In Proceedings of 1999 IEEE Symposium on Security and Privacy,* pp133-145, Oakland, 1999.

[30] Y.Yang, T.Pierce, and J.Carebonell. "A sudy on retrospective and on-line event detection," *In proceeding of SIGIR-98,* Melbourne, Australia, 1998.

[31] Nong Ye, Xiangyang Li, Qiang Chen, Syed Masum Emran, and Mingming Xu, "Probabilistic Techniques for Intrusion Detection Based on Computer Audit Data," *IEEE Transaction on Systems, Man, and Cybernetics-Part A:Systems and Humans,*Vol.31, No.4, July 2001.

[32] Nong Ye, Syed Masum Emran, Qiang Chen, and Sean Vilber, "Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection," *IEEE Transaction on Computers,*Vol.51, No.7, July 2002.

[33] Nong Ye, Timothy Ehiabor, and Yebing Zhang, "First-order versus high-order stochastic models for computer intrusion detection," *Quality and Reliablity Engineering International,* 2002, 18: 243-250.