Adaptive Observation-Centric Anomaly-Based Intrusion Detection: Modeling, Analysis and Evaluation

by

Zonghua Zhang

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Hong Shen

School of Information Science
Japan Advanced Institute of Science and Technology

March, 2006

Abstract

Anomaly-based intrusion detection is to discern malicious and legitimate patterns of behavior in the variables characterizing the normality of information systems. As the system normality is constructed only from an observed sample of normally occurring patterns, despite systems are dynamic in nature (or user-driven) and becoming increasingly complex, anomaly detectors often suffer from excessive false alerts. This dissertation presents our work on the development of effective and efficient models, methods and techniques for anomaly-based intrusion detection with the main concern of adaptability. Our work generally involves three parts, which can be summarized as follows:

The first part of our work is motivated by the observation that the fundamental understanding of the computing environments is an initial but essential step in the process of developing an effective anomaly detection model. Based on the similarity between anomaly detection and induction reference problems, we present a statistical framework for analyzing the general behavior of anomaly detectors from the perspective of their observable subjects. The objective is to lay a theoretical foundation for the modeling and developing of our specific anomaly detectors in the next stage of our work. To enrich the framework, we examine some challenging issues currently exist and present potential solutions, including host-based and network-based normality characterization, evaluation of anomaly detectors, etc.; The framework also involves some case studies and comparative analysis on several typical anomaly detectors, for the sake of presenting a formal way for the understanding and development of anomaly detectors' operational characteristics, and therefore bring them to broader applications.

Taking the constructed framework as starting point, and with the objective to capture the normality drifts of computer systems behavior driven by users or system itself, we develop three versions of SVM-based anomaly detectors, which employ three modified Support Vector Machines as the kernel detection scheme. Our modification aims to break the traditional assumption that anomaly detectors are always fed with training data that are readily available with desired quality in batch, and thus enable them to be trained online periodically for the sake of adapting to the new computing environments without triggering excessive false alerts. To validate those anomaly detector's performance, we implement the experiments by reforming 1998 DARPA BSM data set collected at MIT's Lincoln Labs, and conduct the comparative studies with the original algorithms. The experimental results verify that our new designed anomaly detectors outperform the original ones with fewer support vectors (SVs) and less training time without sacrificing detection accuracy.

Based on the observations and conclusions of the first part of work, we present another framework for the correlation of several observation-specific anomaly detectors. Our hope is that a collection of simple surrogates based on specific operating environments can cooperate well and evolve into generic models with broader anomaly detection coverage and less false alerts. As the specific implementation of the framework, we develop an integrated anomaly detection model named Autonomic Detection Coordinator (ADC) to defend against host-based intrusive anomalies. The cooperation between four host-based

anomaly detectors is formulated as a multi-agent partially observable markov decision process. A policy-gradient reinforcement learning algorithm is then employed to search in an optimal cooperation manner, with a set of parameters controlling individual anomaly detector's behavior. The generic behavior of the coordinator can be adjusted easily by setting a reward signal to adapt to changing system environments. A host-based experimental scenario is developed for implementation, and the experimental results show its satisfactory performance. The model is also extended as the basic framework for the modeling and analysis of multi-stage coordinated attacks in computer networks. The definitions and properties derived from the models (both defender-centric and attacker-centric) present us a formal way for the development of countermeasures to thwart or mitigate such attacks. Taking into account the specific concerns of attackers and defenders, two algorithms called Attackers Nondeterministic Trail Searching algorithm (ANTS) and Attacker's Pivots Discovery by Backward Searching algorithm (APD-BS) are developed respectively. The former one aims to search for the most efficient concurrent actions for attackers, and the latter one intends to discover the attacker's significant observations for defenders.

Acknowledgments

I wish to first express my sincere gratitude to my supervisor Professor Hong Shen for his constant encouragement and kind guidance during my work. His dedication, devotion to the research, and his passion and enthusiasm for the life deeply inspired me, not only for work, but also for life.

I would like to thank Associate Professor Xavier Defago, the advisor of my sub-theme, for his valuable suggestions and helpful comments on my sub-thesis research.

I devote my sincere thanks to all of the members of Shen Laboratory, especially to Dr. Xiaohong Jiang, Dr. Haibin Kan, and other friends from JAIST, who gave me a lot of help on my research and life. They are Dr. Keqiu Li, Dr. Gui Xie, Ms. Wenyu Qu, Ms. Hui Tian, Mr. Chao Peng, Mr. Yingpeng Sang, Ms. Wen He, Mr. Haibo Zhang, Ms. Yawen Chen.

My special thanks goes to the Graduate Research Program (GRP) offered by JAIST, which 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. Thanks also goes to the Foundation for C&C Promotion and the Telecommunications Advancement Foundation (TAF), which supported me to attend and present our work at some international conferences.

Finally, and importantly, I am very grateful to my parents, my brother, and my sister, for their full support and constant encouragement, without which, it is impossible for me to accomplish this work.

Contents

Chapter 1

Introduction

1.1 Background

As computer systems 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 their security, because a major attack can significantly reduce the capability of information systems, and 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, system authorization, encryption, etc., intrusion detection techniques are attracting increasingly attention. So far, many commercial intrusion detection systems (IDSs) have been applied in practice, however, their limited performance, the increasingly open and interconnected nature of today's networks, and more strict security concerns of people present us compelling mission to develop more effective and efficient intrusion detection techniques.

General speaking, intrusion detection is about discriminating malicious attacks (a more general term is anomalies) that might threaten the security from the normal activities of information systems. Existing intrusion detection techniques fall into two general categories: anomaly (or profile-based) intrusion detection and misuse (or signature-based) intrusion detection, and those two categories of intrusion detection techniques essentially have the nearly complementary concerns. Misuse intrusion detection aim to describe the behavior of known attacks, and then detects the patterns which match closely to activity that is typical of the known intrusion; anomaly intrusion detection relies on models of the intended behavior of users and applications in terms of specific observable variables and their operating environments, the deviation between an ongoing activity and the predefined normality patterns/rules interprets whether an intrusion is happening or already happened. Due to the inherent complexity of information systems, neither of those two kinds of techniques provides perfect intrusion detection performance. For those anomaly-based intrusion detectors (for simplicity, we call anomaly detector in this thesis), since the system/user normality is constructed only from an observed sample of occurring normal patterns, despite systems are dynamic in nature (or normality drift) and becoming increasingly complex, they often suffer from excessive false alerts. Whereas misuse intrusion detection methods are always criticized for their ineffective detection of novel attacks due to the incomplete characterized intrusive patterns, although they seldom report false alerts. From the technical standpoint, misuse intrusion detection is easier to implement thereby have been widely adopted in commercial IDSs, whereas

anomaly intrusion detection brings more challenges due to the difficult characterization of system normality. In addition to the taxonomy based on modeling methods, from the perspective of the sources of observable variables (observations latter for simplicity), intrusion detection can also be classified as host-based intrusion detection (HID) and network-based intrusion detection (NID). Since the hosts and communications links are always integrated as two major elements of networks, no matter what techniques are developed, in most cases, they are both of concern. In another sense, although detecting attacks against systems has, in practice, largely been delegated to sensors (such as NIDS), due to the inherent limits of those available NIDSs and the increasing application of encryption in communication, such as IPSec, SSL, intrusion detection and prevention have once again moved back to the host systems themselves. General speaking, most of our work presented in this thesis belongs to host-based anomaly intrusion detection.

Careful analysis on the available literature of intrusion detection shows us that two elements are essential to anomaly intrusion detection, namely, data models of the observable subjects or events, and the specific techniques employed to characterize and analyze the data model. Specifically, several concerns need to be considered well in the process of anomaly detection modeling: *what* observable subjects, and *what* attributes of those obtained subjects should be taken into account to characterize the normal activities; *what* existing approaches or novel methods can be employed to analyze and detect intrusive anomalies based on the detection targets and corresponding characterized observations? The selected observations (e.g., log files, system calls traces, network packets, command line strings) are used to characterize normal behavior and hence construct anomaly detectors' operating environment. Due to the diverse characteristics, different observations have different capabilities for describing system normality, and thus the formulated operating environment might limit their ability to discover some hidden intrusive attempts. For instance, some attacks might be detected in system call stacks, whilst escaping from system call traces [27], and these phenomena also exist even for the same anomaly detector. From this point of view, a preliminary analysis of the observation's property and fundamental understanding of the anomaly detector's operating environment would significantly facilitate their design and performance improvement [46]. Therefore, in our work, we always firstly conduct observation-centric analysis before the development of specific anomaly detection models.

1.2 Challenges and Contributions

A general recognized criterion for the evaluation of anomaly detection is the trade-off between the capability of detecting attacks and the ability of suppressing false alerts. In another word, the anomaly detectors are expected to detect malicious anomalies as accurate as possible, meanwhile suppress false positive rate (the ratio of normal patterns that is being detected as anomalous ones and the total number of normal ones) as low as possible. While accuracy is the essential requirement of an anomaly detector, its capability of handling false alerts attracts more concerns. Actually, evaluating anomaly detection alerts and conceiving an appropriate responses has posed as a challenging task in intrusion detection community for a long history. Both practitioners [12, 56] and researchers [3, 9, 41, 65] have observed that IDSs can easily trigger thousands of alarms per day, up to 99% of which are false alerts, and this flood of mostly false alarms might distract the intrusion

detection analyst from spotting real attack, which tend to be more subtle. The manual investigation of alerts just like the pick up of needles from shock, which has been found to be labor intensive and error prone. Tools and techniques to automate alert investigation are being developed, however, there is currently no silver-bullet solution to this problem.

So far, two general means have been developed to cope with the excessive false alerts. The first kind of approach is to design adaptive anomaly detectors that are capable of capturing *normality drifts*, i.e., concept drifts of normal behavior [4, 51, 55]. Another countermeasure is to conduct post-analysis on the alerts clusters by the abstraction and correlation of numerous detectors' reports [18, 40, 65, 80]. Usually, host-based anomaly detectors take the former means, while network-based anomaly detectors takes the latter one, since the post-analysis always needs some additional analytical models and extra computational overheads, such burdens can be shared by networks while not individual hosts. From another point of view, the former means attempts to eliminate the curses of false alerts, while the latter one intends to mitigate the aggregation effects of the false alerts flood that have been triggered by finding predominant and persistent root causes.

An initial but important stage for the modeling of anomaly detectors is to preprocess and analyze their operating environment in terms of specific observable variables, which is so called "Normality Characterization". However, due to the inherent nature of the anomaly detection, whose basic assumption is that attackers' behavior are significantly deviate from those of normal users, and because of the increasing complexity of modern computer systems and the diversity of the networks, it is generally agreed that there is no such thing as a typical and perfect "system normality description". A possible way, which is also the trend of current anomaly detection research, is to develop methods for characterizing a given operating environment sufficiently well so that optimal detectors for that environment can be designed. The cost must be paid of such work is to allow the limits of detectors, in terms of expected false alert rate, to be predicted. Along the line, most of the available anomaly detectors employ specific subjects with manageable properties as observation, and modeling the subjects as they need. Although many attacks can be identified using these models, unperfect description of the normality and the novel legitimate activities make them suffer from uncontrollable false alerts.

Promoted by the observations that we mentioned above, and motivated by the addressed problems, our work are mainly focused on the modeling and development of effective and efficient anomaly detectors, with the emphasis on their adaptable behavior in those changing computing environments. Our main contributions in the thesis are fourfold, which can be generalized as follows:

- To insight into the operational capabilities and limits of anomaly detectors, and evaluate them in convincing manners, we need analyze both anomaly detection models themselves and their operating environments. Rather than limit our attention to the analysis and design of specific anomaly detection techniques, we give a general investigation with perspective on the observable variables. Based on the similarity with induction problem, we cast anomaly-based intrusion detection in a statistical framework, which facilitates the analysis of their anticipated behavior at a high level. Existing problems and corresponding solutions about the characterization of system normality for the observable variables that from hosts and network have been discussed respectively, together with the case studies about the operational characteristics of several typical anomaly detection models. Moreover, the evaluation of anomaly detectors are also discussed based on some existing achievements.

Our studies show that the fundamental understanding of the observable subjects is the elementary but essential stage in the process of building an effective anomaly detection model, which therefore worth deep exploration, especially when we face the dilemma between anomaly detection performance and the computational cost.

- 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). In addition, 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 in Solaris OS. We developed three adaptive anomaly detectors by modifying conventional SVM, Robust SVM and One-Class SVM respectively. The modification is based on the idea from Online SVM, and thus enables those anomaly detectors to be trained online by breaking the strong traditional assumption that training data are readily available with high quality in batch. The main characteristic of those SVM-based anomaly detectors is able to capture the "normality drifts" of normal behavior traces so as to adapt to the changing computing environments. Both the theoretical analyse and experimental evaluation indicate that our SVM-based anomaly detectors can be trained online and outperform the original algorithms with fewer support vectors (SVs) and less training time without sacrificing detection accuracy.

- We developed an integrated anomaly detection model whose core component is an Autonomic Detection Coordinator (or ADC), with the objective to correlate a set of parametric anomaly detectors working in different computing environments. The model's formulation is prompted by two key observations, first, anomaly detectors work with different observations and have different detection coverage and blind spots; second, different observations might provide different information to reveal intrusive anomalies. In the model, the cooperation between individual detectors are formulated as a Partially Observable Markov Decision Process (POMDP). A policy-gradient reinforcement learning algorithm is applied to search in an optimal cooperation strategy, in order to achieve broader detection coverage with fewer false alerts. ADC's distributed architecture enables its scalability to more complex situations and the dependability to tolerate the failure of basic detectors. Moreover, ADC's behavior can be adjusted easily by setting a global reward signal function, to meet the diverse demands of changing system situations, allowing it to be trained periodically to capture the drifts of system normality. We have implemented this model as a multi-layered host-based defense system to defend against intrusive anomalies by correlating four parameterized host-based observation-specific anomaly detectors.

- As the multi-stage coordinated attacks bring many challenging issues to the security analysts due to their stealthy characteristics in temporal and spacial spans, based on the understanding of the basic properties of multi-stage coordinated attacks, we extend MPOMDP as a two-sided model for the characterization and analysis of both defender's and attacker's behavior. Firstly, users behavior (both defender and their adversaries) are cast in a general framework laying theoretical foundation for the latter modeling and analysis, behavior of both attackers and defenders are then specialized according to their particular concerns. From attacker's point of view, an ANTS algorithm is developed to search for such attack schemes (in terms of concur-

rent actions) with the minimum cost; from the defender's standpoint, a backward searching algorithm APD-BS is designed for discovering the key observations of the attackers in order to effectively countermine the attacking attempts by removing such key observations.

1.3 Thesis Organization

For the easy understanding of this thesis, chapters are organized in accordance with the claimed contributions, and presented as self-contained format, whereas sharing some common concepts, definitions, and notions.

Chapter 2 addresses the constructed framework for the analysis of anomaly-based intrusion detection. We first constructs a formal framework for the characterization of anomaly detectors' anticipated behavior; a general description of some particular observable subjects' normality is given; we then investigate several typical anomaly detectors' operating environments, with emphasis on their operational limits; some other issues associated with evaluation metrics are also discussed.

Chapter 3 is about the development of SVM-based anomaly detectors. First, we address the problem to be solved and describe the data source that is used in our work together with the data modeling; second, we introduce the effective binary classifier SVM, and modify three SVMs as online training detectors, which have different assumptions, for real time intrusion detection; Finally, experiments are implemented to validate the performance of our proposed methods, the comparative studies with the original algorithms are also conducted.

Chapter 4 presents our developed autonomic detection coordinator. We firstly construct a formal framework for the correlation of multiple anomaly detectors based on partially observable markov decision process, and cast the posed problems in the framework with detailed formulation, together with specific solutions. We then set a host-based experimental scenario conduct comparative studies to validate our model.

Chapter 5 extends the model of last chapter for the modeling and analysis of multi-stage coordinated attacks in computer networks. We first present the formal definition and basic properties of multi-stage coordinated attacks by casting the users' behavior in POMDP framework. Both attacker's and defender's behavior are then further characterized by taking into account their special observations and properties, and two algorithms drawn from ACO (Ant Colony Optimization) algorithm family are then developed with attacker's and defender's concerns respectively.

Chapter 6 summarizes our work that have been presented in this thesis, and discusses the future work.

Chapter 2

Observation-Centric Analysis on the Modeling of Anomaly-based Intrusion Detection

2.1 Introduction

The existing literature show that during the development of IDS, two elements worth careful consideration, namely, selection and modeling of the observable subjects and the techniques for characterizing and analyzing the data model. As we have known, network can be logically classified into two major components, hosts and communication links among the hosts. In those two kinds of computing environments, the behavior of a subject is observed via the available audit data log. For instance, network traffic data, which capture data packets traveling on the communicate links, and audit data, which record the sequence of events on the hosts can be selected as observable subjects. Those two domains actually 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 the techniques that are capable of distinguishing malicious and normal behaviors worth consideration.

As we know, the basic assumption for anomaly detection is that the intrinsic characteristic or regularity of the normal observable subjects deviate significantly from that of anomalies, therefore, the preprocess and analysis of the operating environment, which is composed of specific observations, is an initial but important stage for the modeling of anomaly detectors. In another intuitive explanation, characterization of the system normality is the key concerning the anomaly detector's trade-off between the capability of detecting anomalies and the ability of suppressing false alarm rate. A possible way, which is also the trend of current anomaly detection research, is to develop models for characterizing a given operating environment sufficiently well so that optimal detectors for that environment can be designed. However, due to the lack of the critical understandings and useful tools for characterizing observable subjects, most anomaly detectors are developed based solely on "expert" knowledge or intuition, which is often imprecise and incomplete given the increasing complexity of modern computer systems and the diverse nature of today's networking environments. The cost must be paid of such work is to allow the limits

of anomaly detectors, in terms of expected false alarm rate, to be predicted. More seriously, most existing anomaly detectors pay more attention to the technique itself, rather than the fundamental understanding of the working environments, which limits their contribution to this research field, and restricts them to a broader application. Another serious problem is the anomaly detectors' evaluation, which is deficient and unconvincing due to the limits of so-called benchmark data set, especially for those researches that have been focused on a specific method for a particular operating environment. To date, the most comprehensive evaluation of research on intrusion detection systems that has been performed is an ongoing effort by MIT's Lincoln Laboratory, performed under DARPA sponsorship. It does provide a basis of making a rough comparison of existing systems under a common set of circumstances and assumptions, however, many criticism and review have pointed out its shortcoming and flaws involving problems in determining appropriate units of analysis, bias towards possible unrealistic detection approaches, and questionable presentations of false alarm data [62]. Most anomaly detector's evaluation relatively little concerning some of the more critical aspects of their work, such as validation of their testing data, detailed characterization of their operational procedure and environments, etc. In this sense, concrete observation-centric analysis would facilitate the understanding and evaluation of diverse anomaly detectors's working mechanism.

With the introduced problems in mind, we intend to explore the fundamental attributes of some observable subjects, and analyze the operating environment of several typical anomaly detectors that drawn from different research fields, based on a general description of their anticipated behavior [92]. General speaking, this chapter presents our contributions as following:

- Casting the anomaly-based intrusion detection in a statistical modeling framework, and characterize the system normality in a general way by selecting several specific observable subjects that have been applied to some existing typical anomaly detectors;

- Conducting a careful analysis on the operating environments that some anomaly detectors work with (mainly the ordering property and frequency property), as well as the comparative studies in terms of their operational capabilities/limits;

- Concluding the current evaluation methodologies, and propose our idea for better measurement metrics based on some critical analysis.

The rest of this chapter is organized as follows. Section 2 establishes a statistical framework to describe the behavior of anomaly detectors from an overall view. In section 3, we give a general description of the selected observation's normality. Section 4 characterizes the operating environment of several typical anomaly detectors, together with the analysis of operational limits. In section 5, we propose our idea for better anomaly detection metrics based on some existing conclusions. Finally, we give a general discussion in section 6.

2.2 A General Statistical Description

A general statistical formulation of the computer misuse detection have been discussed in [33], which is generally regarded as a theoretical framework for the latter development of

intrusion detection models. With the similar formulation, while pay more attention to the anomaly detectors' operating environments, i.e., the properties of observable subjects, in this section, we give another statistical description for the anomaly detectors' anticipated behavior from a more general viewpoint. The description is based on the analogy between *anomaly detection* and *induction reference* problem. A general statistical framework can be utilized to describe the AD's behavior:

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 anomaly detectors.

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

O_t: observation that is captured at time interval t, it can represent a single event or a group of events according to the specific detection model, and its generation is governed by the hidden process H;

$Set(O_t, w)$: a set of observation 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., $H(t) = 0$;

$n(O_t)$: the probability that the subject to be generated by $N(t)$ at time step t is O_t, i.e., $Pr\{O_t|H(t) = 0\}$;

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

$m(O_t)$: the probability that generated malicious subject at time step t is O_t, i.e., $Pr\{O_t|H(t) = 1\}$;

$\phi_i, 0 \leq i \leq Num$: a pattern (or a probability measure) for legitimate activity that is stored in the normal dataset Φ with size Num;

$\tilde{AD}(\cdot)$: the probabilistic anomaly detector with input O_t or $Set(O_t)$ and output is the probability that input is determined as malicious;

$AD(\cdot)$: the deterministic anomaly detector with input O_t or $Set(O_t)$ and output is the binary determination whether input is malicious.

λ: *a priori* probability that current observable subject is normal, i.e., $\lambda = Pr\{H(t) = 0\}$, and λ is close to 1 due to the fact that the number of malicious process is much smaller than that of normal process.

As we know, the objective of anomaly detectors is to capture any malicious subjects that generated by the hidden stochastic process $H(t)$, and what they depend on is a collection of normality characterization of available subjects. Since Num, the size of the samples of the normal patterns Φ is limited, naturally, the most effective observations (or characterized patterns) are desirable. Generally, two properties of the observable subjects, that is, *ordering property* and *frequency property*, can be taken advantage of to construct the system normality according to the correlation of individual observed events O_t. Although some anomaly detectors drawn from machine learning (or specification-based techniques) do not take those two properties as their main concern, our analysis is mainly based on this basic taxonomy.

2.2.1 Frequency-based analysis

Property 1 *Assume that at time interval $[t-1, t]$, an AD observes an unordered set of events $e_1^t, e_2^t, ...e_n^t$, which is generated by H(t). The frequency of those events $F(e_1^t, e_2^t, ...e_n^t)$*

can be taken as a measurement to characterize the system normality, i.e., $O_t = F(e_1^t, e_2^t, ...e_n^t)$.

If O_t is taken independently (here O_t is considered as a unit of events), the available observation can be viewed as an unordered collection of subjects in a particular unit, and the consideration of temporal patterns that the observation may contain is excepted. Helman et al. [33] ever gave a thorough analysis for the statistical foundations of computers audit trail with such property, and in such cases, the probability that current subject O_t is malicious can be determined according to Bayes theorem,

$$\begin{aligned} Pr\{H(t)=1|O_t\} &= \frac{Pr\{O_t|H(t)=1\}\cdot Pr\{H(t)=1\}}{Pr\{O_t|H(t)=1\}\cdot Pr\{H(t)=1\}+Pr\{O_t|H(t)=0\}\cdot Pr\{H(t)=0\}} \\ &= \frac{Pr\{O_t|H(t)=1\}\cdot(1-\lambda)}{Pr\{O_t|H(t)=1\}\cdot(1-\lambda)+Pr\{O_t|H(t)=0\}\cdot\lambda} \\ &= \frac{m(O_t)\cdot(1-\lambda)}{m(O_t)\cdot(1-\lambda)+n(O_t)\cdot\lambda} \\ &= \frac{c(O_t)}{c(O_t)+\lambda/(1-\lambda)} \end{aligned}$$

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

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

A series of optimality conditions for the above detection model have been discussed in [33], and as they pointed, due to the lack of prior knowledge about λ, $m(O_t)$, and $n(O_t)$, it is almost impossible to carry it out into practice. Specifically, a good estimates of λ and a thorough understanding of distributions of the processes $N(t)$ and $M(t)$, which we call system normality, are not readily available, which thus make the detection task deem to be NP-hard.

Actually, anomaly detection can be regarded as an induction problem in some sense. Assume that we have an unordered set of n finite description of observable events (strings of symbols), $O_1, O_2, O_3, ..., O_n$. Given a new event at time t, O_t, what is the probability that it belongs to the set? A well fitting anomaly detector with good description for the known set of events is expected. The universal distribution [73] gives a criterion for goodness of fit of such description. According to our definition, the universal distribution $D_{\tilde{AD}}$ for anomaly detector $\tilde{AD}$ can be regarded as a weighted sum of all finitely describable probability measures on finite events:

$$D_{\tilde{AD}}([O_i]) = \sum_j \beta_j \prod_{i=1}^{t} p_j(O_i) \tag{2.1}$$

t is the time step representing the number of available observation set $[O_i]$, β_j can be taken as the weight of the j^{th} probability distribution on finite observations, and its definition based on the particular detection model, for example, for an anomaly detector using string match method, $\beta_j = 1$, if ongoing events match the exact pattern ϕ that

stored in normal pattern set Φ. Suppose that $[O_i], i = 1, 2, ...t$ is a set of t observations generated by stochastic process $h(x)$, the probability that $D_{\tilde{AD}}([O_i])$ assigns to a new observation O_{t+1} is

$$Pr(O_{t+1}) = D_{\tilde{AD}}([O_i] \bigcup O_{t+1}) / D_{\tilde{AD}}([O_i]) \tag{2.2}$$

The probability assigned to $[O_i]$ by stochastic generator $h(x)$ is

$$h([O_i]) = \prod_{i=1}^{t} h(O_i) \tag{2.3}$$

In an effective anomaly detection model, for a suitable set of observations $[O_i]$ that used for characterizing system normality, the probability assigned by $D_{\tilde{AD}}$ in (2.1) should be very close to those generated by hidden stochastic process $h(x)$ in (2.3), that is, a maximal prior information an anomaly detector can posses is the exact knowledge of λ, but in many cases the true generating process $h(\cdot)$ is not known, what we expect is that an anomaly detector based on $D(\cdot)$ performs well with small expected errors between $D(\cdot)$ and λ. For such two probability distributions on finite number of observations, a corollary derived from Hutter [38] can be given as:

Corollary 1 (Difference Bound) *The expected value of the sum of the squares of the differences in probabilities assigned by the stochastic generator $h(\cdot)$, and anomaly detector $D(\cdot)$ to the elements of the observation are less than a certain value, and the expected error in probability estimate might decrease rapidly with the growing size of the normal data set.*

The corollary guarantees theoretically that predictions based on $D(\cdot)$ are asymptotically as good as predictions based on λ with rapid convergence. Any *a priori* information that can be insert into $D(\cdot)$ to obtain less errors, and we believe that if all of the needed *a priori* information is put into $D(\cdot)$, then (1) is likely to be the best probability estimate possible to $h(\cdot)$, and thus anomaly detector could achieve one hundred percent accuracy. So far, neither modeling approaches, which aim to estimate c, $N(\cdot)$,$M(\cdot)$, nor nonmodeling approaches, which deduce and generate normal behavior rules using heuristic, clustering algorithms, data mining techniques and statistical measures, have given a thorough solution. Actually, the limited samples we can obtain, together with corresponding sampling errors, determine what we can do is just estimate and predict system normality in an approximate way.

2.2.2 Sequence-based analysis

Property 2 *Assume that at time instant $[t-1, t]$, an AD observes an ordered set of events $e_1^t, e_2^t, ...e_n^t$, which is generated by H(t). The ordering of this observed event sequence $S(e_1^t, e_2^t, ...e_n^t)$ can be taken as a measurement to characterize the system normality, i.e., $O_t = S(e_1^t, e_2^t, ...e_n^t)$.*

In many cases, the ordering property rather than the frequency property dominates the characteristic of observable subjects, the pattern of $Set(O_t, w)$ rather than the individual event O_t is thus of potential interest, and the ongoing events should be considered in

a consecutive manner instead of independently. Based on the assumption that current event O_t is related with previous events, hidden generation process, and time instant t, a pair of probability distribution can be given as following:

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

for most problems, the ultimate goal is just to identify a short temporal pattern of anomalous events, therefore, the sequence $O_{t-1}O_{t-2}...O_1$ can be replaced by $Set(O_t, w)$,

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

Similar to the analysis for unordered event set, a posterior probability of anomaly detection based on temporal-related events can be given as:

$$\begin{aligned} & Pr\{H(t)=1|O_t, Set(O_t, w), t\} \\ = & \frac{Pr\{O_t|H(t)=1, Set(O_t, w), t\} \cdot (1-\lambda^{'})}{Pr\{O_t|H(t)=1, Set(O_t, w), t\} \cdot (1-\lambda^{'}) + Pr\{O_t|H(t)=0, Set(O_t, w), t\} \cdot \lambda^{'}} \\ = & \frac{c \cdot (1-\lambda^{'})}{c \cdot (1-\lambda^{'}) + \lambda^{'}} \end{aligned}$$

Where $\lambda^{'} = Pr\{H(t)=0, Set(O_t, w), t\}$ is similar with λ, 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

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

From the above formulation, we do not know with certainty the generation of $Set(O_t, w)$ by mixture process $h(x)$, nor do we know the distribution of $M(t)$ and $N(t)$. The ongoing event O_t may depend on the current time step t, as well as the temporal pattern of events generated at time steps prior to t, which allows the possibility that $M(t)$ and $N(t)$ are non-stationary. Furthermore, instead of restricting our attention on $Set(O_t, w)$ whether and which its subsequence is generated by $M(t)$ or $N(t)$, we regard it as a whole dynamic temporal pattern, therefore, the detection problem of interest is to decide whether the appearance of ongoing event reveal the temporal pattern includes w events as anomalous, rather than concern the individual O_t, however, we do not exclude the possibility that the sudden appearance of anomalous event uncover any previous potential anomalies at once.

Similarly, the estimation of $Pr\{O_t|H(t)=1, Set(O_t, w), t\}$ and $Pr\{O_t|H(t)=0, Set(O_t, w), t\}$ can also be roughly considered as a simple inductive inference problem: *Given a string $O_{<t}$ (denote $O_1, O_2, ...O_{t-1}$), take a guess at its continuation O_t.* Specially, the generation of the event sequence $O_1, O_2, ...O_{t-1}$ is governed by a hidden stochastic process $h(\cdot)$, and μ is unknown probability distribution for taking O_t at particular time instant t based on the available event $O_1, O_2, ...O_{t-1}$, i.e. $\mu(O_t|O_{<t})$, while ρ is a guess probability distribution close to μ or converges, in a sense, to μ, and we expect that an anomaly detector based on ρ performs well. Taking into account the specific property of anomaly detection

and keeping the consistence with the former analysis, here we assume $\mu \approx \lambda$. Suppose $P := \{p_1, p_2, ...p_n\}$ is a countable set of candidate probability distributions on event sequences, a universal probability distribution π related to P (in essence O) hence can be defined as:

$$\pi(e^t_{1:n}) := \sum_p w_p p(e^t_{1:n}), \sum_{p \in P} w_p = 1, w_p > 0. \tag{2.4}$$

As the above notations, the normal observation set O or P is known and might contain the true distribution $\lambda = p_i$ if O or P is sufficiently large or with well characterization. Based on those assumptions, two corollaries therefore can be deduced from theorems of [38] in following to describe the general behavior of sequence-based anomaly detection models:

Corollary 2 (Convergence) *Assume a hidden stochastic process $h(\cdot)$ generates an event sequence $e^t_1, e^t_2...e^t_n$ over a finite space S with probability $\lambda(e^t_{1:n})$. An AD observers the first i events, the universal conditional probability $\pi(e^t_1|e^t_{<i})$ of the next symbol e^t_i given $e^t_{<i}$ is related to the true conditional probability $\lambda(e^t_i|e^t_{<i})$ in the following way:*

$$\sum_{i=1}^{n} E_{<i} \sum_{e^t_i} (\lambda(e^t_i|e^t_{<i}) - \pi(\lambda(e^t_i|e^t_{<i}))^2 \leq \ln w_\lambda^{-1}$$

where $E_{<i}[..] := \sum_{e^t_{<i} \in P^{i-1}} \lambda(e^t_{<i})[..]$ is the expectation and w_λ is the weight of λ in π.

which shows that the predication accuracy of anticipated anomaly detectors are asymptotically as good as predications based on the stochastic generator $h(\cdot)$ with rapid convergence. However, in practice, ongoing observation might not have exact matching pattern in P, i.e., $\lambda \notin P$, in such case, a "nearby" distribution $\hat{\lambda}$ with weight $w(\hat{\lambda})$ is expected, and the distance between $\hat{\lambda}$ and λ is bounded by a constant. The convergence of anomaly detectors determines the amount of training time or data required to have a stable model, and the detector converges well when most of the "anticipated" patterns appear repeatedly and are extracted well.

Corollary 3 (Error Bound) *Assume a hidden stochastic process $h(\cdot)$ generates an event sequence $e^t_1, e^t_2...e^t_n$ over a finite space S with probability $\lambda(e^t_{1:n})$ at time instant t. Θ_π is the universal prediction scheme (used by a probabilistic AD to determine the deviation between normal sequence and abnormal ones) based on the universal prior π, Θ_λ is the optimal prediction scheme based on the stochastic generator $h(\cdot)$. The total u-expected number of prediction errors $E_n^{\Theta_\pi}$ and $E_n^{\Theta_\lambda}$ of Θ_π and Θ_λ are bounded by:*

$$0 \leq E_n^{\Theta_\pi} - E_n^{\Theta_\lambda} \leq \sqrt{2Q_n S_n} \leq 2S_n + 2\sqrt{E_n^{\Theta_\lambda} S_n}$$

where $Q_n = \sum_{i=1}^n E_{<i}$ is the expected number of non-optimal predictions made by Θ_π, and $S_n := \sum_{i=1}^n E_{<i} \sum_{e^t_i} (\lambda(e^t_i|e^t_{<i}) - \pi(e^t_i|e^t_{<i}))^2$ is the squared Euclidian distance between λ and π.

The corollary actually gives the upper bound of the false alert rate of an ideal sequence-based anomaly detector. We usually pay our attention to the lower bound of the false alert rate of anomaly detectors, but in fact, all the possible detection schemes also have a upper bound to some extent. Although it makes little sense on designing an anomaly

detection system with near zero false alert rate, it really gives us an impression that any anomaly detection schemes based on sequence prediction would never perform too badly. And obviously, how to select a universal probability distribution π, specifically, $p_i \in P$ and w_i, is always the key to design an ideal sequence-based anomaly detection system.

Rather than considering the specific design of anomaly detectors, here we just attempt to show that anomaly detection problem essentially is also a prediction problem in some sense. Related proof of those two corollaries can be found in [38], which provides theoretic foundation for any anomaly detection scheme, and shows that probability distribution of the expected controllable process converge to that of the hidden stochastic process and limited by errors bound. Based on the historic data, the extent of the deviation between an expected event and ongoing event thus determines whether anomaly appears.

Generally, this section casts the anomaly detection problem in a statistical framework to describe the anticipated behavior of anomaly detectors from a high-level viewpoint, which facilitate us to construct a basic modeling for the further discussion in our latter work. Although the unrestricted assumption of the framework is quiet complex and general, it is nevertheless meaningful to provide an outline for our detailed analysis. As we know, many of subjects that anomaly detection schemes to examine are notoriously noisy, non-stationary, and defined on extremely large alphabets, while our framework extracts them to a comprehensible and manageable level, and based on which, we select several typical subjects that have been widely used for analysis.

2.3 Normality Characterization

Basically, two kinds of observable subjects from computer systems can be selected as the objects for monitoring and analyzing in order to capture the anomalous traces, namely, hosts in the network and the communication links among the hosts. From a general viewpoint, most of those subjects have the following characteristics:

- The amount of generated data is huge, with a large number of attributes, and each of those values may be in a complex form. Many of the data sources to be examined are defined on extremely large alphabets.

- Noisy data are generated randomly, mixed with normal data, which makes it difficult or even impossible to distinguish normal data and malicious data with high accuracy.

- The process that governs the generation of subjects is non-stationary, and the concept of system normality might be site-specific and drift with time.

- The profiles to be monitored are presented with unbounded data streams, being updated frequently, dynamic, and even transient.

Consequently, several criteria worth careful examination in order to select the most effective and efficient observable subjects for characterizing the system normality:

- *Availability*, the most basic condition, which means that the subject can be observed and captured directly or with some assist tools.

- *Tangibility*, which means that subjects can be recorded in a specific form, and can be recognized or dealt with in a particular way, such as user profiles or audit files.

- *Operability*, which means that although a subject might have a large number of attributes, it should be possible to be managed using some data processing techniques such as attribute projection, feature selection, or value aggregation.

- *Sensitivity*, which means that the subject is both robust to variations in system normality, and perturbed by intrusions, so as to reflect the normality drifts sufficiently well.

The first three criteria are essential to the observable subjects, and all the subjects that have been selected so far have these characteristics, such as the system calls of privilege processes in Solaris OS, system audit events, user command lines, CPU consumption, TCP/IP packets, etc. However, it is usually hard to define the "efficiency" of those subjects, namely, to what extent the subjects could characterize system normalities.

2.3.1 Host-based Normality

A great number of variables could be employed to characterize the state of a host, such as command line strings [58, 59], system call traces [28], resource consumption patterns [53], etc. The properties of all those variables could be encompassed into the framework that we established in the last section. However, in fact, the normal behavior of many variables does not have obvious pattern, which would be taken as "noise" of "normality". Burgess et al. [15] gave a careful analysis on the computer system normality, according to which, the system can be distinguished as three scales:

- *Microscopic,* details exact mechanisms at the level of atomic operations, such as the individual system calls and other atomic transactions in operating systems (in terms of *milliseconds*).

- *Mesoscopic,* looks at small conglomerations of microscopic processes and examines them in isolation, such as the individual process or session, or a group of processes executed by one program (in terms of *seconds*).

- *Macroscopic,* concerns the long-term average behavior of the whole system, such as the periodical activities of the users and their corresponding resources consuming patterns.

All the host subjects fall into these three categories, and can be taken as the objects for anomaly detectors, whether it aims to look for suspicious patterns or attempts to identify the values that deviate from the acceptable distribution of values. But actually, most of the available host-based anomaly detection methods take subjects at *mesoscopic* level due to its better controllable attributes to establish anomaly detection models. For instance, Forrest et al. [28,34] ever proposed an immunological detection model by analyzing system calls sequences, which focus on the *mesoscopic* level of UNIX operating system, and some subsequent independent works [48,50] also take system calls sequences as observable subjects. Consequently, the motivation to analyze the normality of the mesoscopic scale is obvious, that is, why system calls sequences can be selected as observation? What attributes these sequences have? Whether the regularity of such computing environment benefits the anomaly detection? Actually, Forrest et al. [28] has given an satisfied answer for the first question, but for the last two questions, there are still some problems need further exploration.

Most the work took the name of the system calls as the observable (other parameters passed to the system calls are ignored), after sequence is established, namely, $(s_1, s_2, ...s_l)$, detection methods such as Enumerating Sequences, Frequency-based methods, Data mining techniques, HMM, or some text categorization methods were applied to identify anomalies. The work of Lee et al. [48] showed that additional information to the sequence elements would improve detection performance without considering the trade-off between detection accuracy and computational cost. For instance, sequence can be established as $(s_1\_o_1, s_2\_o_2, ...s_l\_o_l)$ or $(s_1, o_1, s_2, o_2, ...s_l, o_l)$, where o_i represent the obname of system call i. Additionally, Lee et al. gave an analysis for the regularity of these objects using information-theoretic measures, such as entropy, conditional entropy, relative conditional entropy, information gain and information cost, which gives us a good clue for the characterization of the system normality. Specifically, for an audit data set X where each data item belongs to a class $x \in C_x, y \in C_y$, several information theoretic measures can be used to describe its characteristics, in order to built an appropriate anomaly detection model:

- *Entropy:*

$$H(X) = \sum_{x \in C_X} P(x) log \frac{1}{P(x)},$$

  where $H(X)$ is the entropy of X relative to C_X, and $P(x)$ is the probability of x in X. As we know, the amount of variability is most easily characterized by the entropy of the signal, if the variations in data are equally distributed about some preferred value, the the distribution over a sufficient number of instances would be normal. $H(X)$ thus can be used to measure the regularity of the record in audit data, and the data set with smaller entropy would improve the detection performance due to its purer nature and simpler structure.

- *Conditional Entropy:*

$$H(X|Y) = \sum_{x,y \in C_X, C_Y} P(x,y) log \frac{1}{P(x|y)},$$

  As we explained in the last section about sequence-based anomaly detection models, for two sequence sets,
  $X = (x_1, x_2, ...x_m)$, $x_i = (e_{i1}, e_{i2}, ...e_{in\_1}, e_{in})$,
  $Y = (y_1, y_2, ...y_m)$, $y_i = (e_{ik}, e_{ik}, ...e_{ik\_1}, e_{ik})$,
  where e_{ij} represent the event and $k < n$, $H(X|Y)$ thus can be used to measure the regularity of sequential dependencies, and the smaller the values is, the more deterministic of the sequence x after y is obtained, which therefore benefits the build of anomaly detection models.

- *Relative Conditional Entropy:*

$$E(p|q) = \sum_{x \in C_X} p(x) log \frac{p(x)}{q(x)},$$

where $p(x)$ and $q(x)$ are two probability distributions over the same $x \in C_x$, and $E(p|q)$ can be applied to measure the similarity of two datasets (e.g. training data and test data). The distance (similarity) between two audit datasets could provide us *a prior* knowledge to build and evaluate anomaly detection models.

- *Information Gain:*

$$Gain(X, A) = H(X) - \sum_{v \in Values(A)} \frac{|X_v|}{|X|} H(X_v),$$

where $Values(A)$ is the set of possible values of A and X_v is the subset of X where A has value v. $Gain(X, A)$ can be used as a criteria to select important attributes for achieving better classification, and thus prediction performance, essentially, it has the similar contribution as conditional entropy to measure regularity of sequential dependencies.

Although there are still some details about the data normality worth consideration, the proposed information-theoretic measures give us some fundamental understanding about the regularity of computing environment that the anomaly detectors work. Lee et al. [48] applied conditional entropy to determine the appropriate length used for sequencing the system calls to construct an anomaly detection model with the conclusion that there is a relationship between the fall of in entropy and the appropriate window size for probabilistically-based classifiers. But interestingly, Tan et al. [77] suggested that conditional entropy is not a universal sequence-length selection metric, and it almost has the same appearance in a general manner, independent of the particular datasets, which undermines its effectiveness. However, we still believe that those information theories can contribute to the characterization of the environment normality, and thus improve the performance of anomaly detectors to some extent. Moreover, we have already found out the intersection between those information-theoretic measures and the stochastic framework we have discussed in the last section, especially for those sequence-based anomaly detection models.

To measure the computer system normality from a macroscopic level, Burgess et al. [15] applied a scaling transformation to the measured data, and the distribution of fluctuations about the mean was approximated by a steady-state, maximum-entropy distribution with modulation by a periodic variation. The idea can be brief described as:

Motivation for Transformation: the entropy of the collected data are computed to gauge the variability of the signal, which indicates that signal is maximally; average and standard deviations are computed in terms of periodicity, and the periodogram standard deviation is itself a pseudo periodic functions of time, which shows that the system acts as a scale of activity that varies in time; each time is re-scaled by its local standard deviation, and the scaled distribution of measurements at a given periodic time is closely resembles a Planck distribution.

Transformation: As the entropy to be high, processes which have "fluctuation structure" can be written in exponential form $exp(-\beta E_i)$ as a Boltzmann distribution with some arbitrary set of parameters E_i, which satisfies the maximum entropy condition for fitting the data; The probability distribution is approximately written as

$$p[q] = exp(-\beta E[q]) / \int dq exp(-\beta E[q]).$$

To determine parameters $E[q]$, a stochastic model is used:

$$E[q] = \int dt[(\frac{dq}{dt}^2) + V(q)],$$

As the system is moderately loaded, two simple assumptions are based on: (a) maximal entropy of data and (b) fluctuations at no cost, therefore, $V(q) = 0$. Finally, Planck distribution, which is the form of the equivalent, transformed steady-state system is yielded through computing the fluctuation spectrum for the model on a periodogram.

Burgess et al. gave a method to characterize system normality from the point of view of macroscopic scale, which inspire us to detect host-based system anomalies from a macro perspective, however, due to its approximate nature, any attacks with normal pattern appearance are difficult to be identified based on such model, in addition, what information are required and effective for detecting anomalies need further exploration, and it heavily depends on what will we do once anomalies have been discovered. Intuitively, the normality of those observable subjects from mesoscopic and macroscopic scales could be combined to achieve better performance, macroscopic normality is used to monitoring the variant of system coarsely, while mesoscopic give doubtful activities further analysis and fine-grain characterization.

2.3.2 Network-based Normality

Due to the diverse nature of the computer network, it is almost impossible to establish an ideal mathematical model with perfect characterization of the normality of observable subjects, i.e. network packets, nor it is easy to design efficient intrusion detection techniques for networking. However, this does not only only for intrusion detection, but also more or less for other fields, such as traffic modeling and analysis. In this sense, the fundamental understanding of basic protocol behavior is a possible way to go. In addition, due to the inherent limits of the available IDSs and the increasing application of encryption in communication, such as IPSec, SSL, intrusion detection and prevention have once again moved back to the host systems. Here, we only propose some preliminary ideas to measure network normality, while further experimental analysis and verification are left to our later work.

So far, *tcpdump data* has been widely applied to detect attacks from the protocol scale (connection behavior). Generally, each record describes a connection using several features: timestamp, duration, source port, source host, source bytes (outbound bytes from source to destination), destination port, protocol type(TCP, UDP, ICMP or others), destination host, destination bytes (inbound bytes from source to destination), and flag. Due to the huge data amount generation everyday and the transient nature, it is really difficult to describe the system normality in details, and therefore simplification and preprocess is needed. Taking those features as various attributes, Lee at al. [48] used information gain as guiding principle to partition *tcpdump* data based on the assumption that the smaller the entropy is, the more regularity the dataset, and therefore benefit for modeling and characterizing anomaly detectors, and conditional entropy was applied to compute temporal and statistical features. Although it is true that such pre-analysis could facilitate anomaly detection modeling, huge amount of data and transient nature make it is time-consuming to determine the proper granularity of the subjects. Some techniques for online analysis of continuous stream give us some clues to capture the transient nature

of network subjects [21, 31]. Additionally, some network traffic modeling methods also give us some inspiration to monitor and obtain the necessary information for measuring network normality at a macroscopic level [54].

In order to develop a traffic model which can accurately characterize the diverse statistical properties with complex temporal correlation and non-Gaussian distributions of heterogeneous network, Ma et al. [54] proposed a wavelet domain-based models. In these models, correlation structures of wavelet coefficients for long/short-rang dependence processes are reduced to only a few key elements. For Gaussian traffic, Markov models can be implemented through a linear model on wavelet coefficients to capture the short-range dependence among wavelet coefficient, i.e.

$$d_s = \sum_{l=1}^{s-1} a_s(l) d_l + b_s w_s, 1 \leq l \leq N$$

where $a_s(l)$ and b_s are weighting factors depending on the one-dimensional index s, and w_s is i.i.d Gaussian noise with zero mean and a unit variance. The value of s and $a_s(l) = 0$ determines the model and the relations between wavelet coefficients, for example, when $s = 1$, and $a_s(l) = 0$ for all l, the model is the simplest one, i.e., an independent wavelet model.

For non-Gaussian distribution traffic, a shaping algorithm was derived using the relationships among wavelet coefficients, scale coefficients, and the cumulative process. Specifically, it includes two stages:

- *Traffic Modeling:* wavelet transform on a training sequence $\hat{x}$ to obtain wavelet coefficients and scaled coefficients, and then estimate the variance of wavelet coefficients and the cumulative probability function of scale coefficients at each time scale.

- *Synthetic Traffic Generation:* generating the background wavelet coefficients by Gaussian wavelet model and compute the shaped wavelet coefficients and scale coefficients recursively for all time scales, after wavelet inverse transformation, synthetic sequence $\tilde{x}$ is obtained.

Therefore, after wavelet transformation, whatever short- and long-range temporal dependence traffic are all "short-range" dependent on the wavelet-domain, which facilitates significantly the characterization of network normality and our analysis of anomalies at a macro level.

The countermeasure to deal with the transient nature of network observable subjects is online analysis, that is, process the data in a single pass, or a small number of passes. For instance, under some definition of "similarity", similar items can be clustered in the same partition, while different items are in different partitions. Based on the existing *facility location algorithm*, Guha et al. [31] modified it to produce exactly k clusters for solving k-Median problem in one pass, their experiment on KDD-CUP 99 intrusion detection data showed that raw tcpdump could be clustered into five clusters with 34 continuous attributes. In addition, Cormode et al. [21] ever proposed a novel algorithm for calculating a small summary for any data stream, i.e. l_o *sketch*, and employed *Hamming norm* to estimate the similarity of streams online, which also give us a rapid and ease method to analyze network regularity.

Based on the available techniques we have analyzed, a framework for measuring network normality can be concluded through a top-down procedure as follows (its skeleton is shown in figure 2.1):

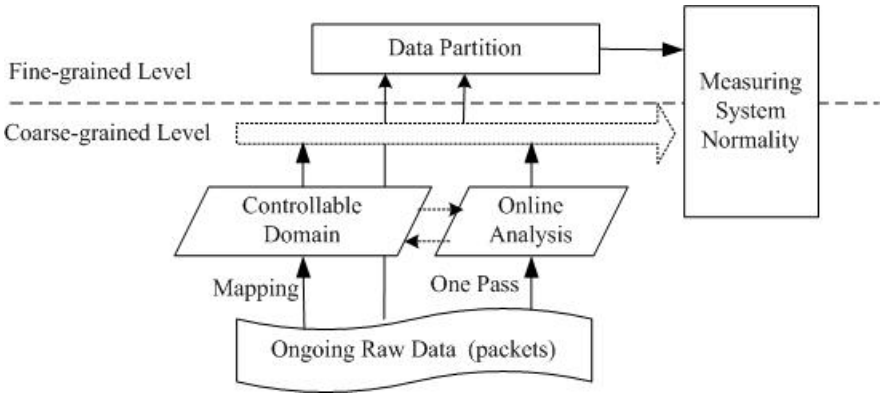

Figure 2.1: A simple framework for measuring network normality

1. *Coarse-grained Level:*

   - Mapping network traffic into wavelet domain to discover the periodicity of the specific network activities, which can disclose the sudden system collapse and unrhythmic activities;
   - Sketch-based techniques and clustering methods are applied to a certain doubtful time-scale (or a periodicity) to have further insightful investigation.

2. *Fine-grained Level:*

   - Information-theoretic measures are used to divide the processed network data from coarse-grained level into more "pure" data sets with higher regularity;
   - Building anomaly detection models based on the characterization of system normality.

Actually, collection and monitor of network observable subjects in a discrete way rather than a continuous way may not deteriorate the performance [15]. From the point of view of the observable subjects, we envision a framework in which several levels of data analysis are used as the basis to be combined to yield a single but effective system normality characterization. We envision further an approach in which anomaly detection models are built on the fundamental understanding of their operating environments, and have the adaptability in response to changing situation. The hope is that a collection of simple, elaborate surrogates based on specific observable subjects can evolve into generic models without performance deterioration. From the similar motivation, a host-based autonomic detection coordinator have been developed in [91].

2.4 Case Studies

Generally, operating environment means the working situation constructed by the observable subjects that anomaly detectors working with, and most of them can be cast in the framework we proposed in section 2. In the last section, we gave a general discussion of normality characterization for observable subjects from hosts and network. After a broad survey of the existing literature on anomaly detectors, we found that most work pay more attention to the design of the anomaly detection models themselves, rather than

the operating environment. Here, we take two kinds of anomaly detectors (frequency-based and sequence-based) as instances to insight their operational mechanisms from the perspective of operating environment.

2.4.1 STIDE Detector

The stide algorithm can be described as follows [28]:
Predefinition: for two sequence X and Y,
$X = (x_1, x_2, ...x_N)$, $Y = (y_1, y_2, ...y_N)$,
the similarity between them is defined as:

$$Sim(X,Y) = \begin{cases} 0 & \text{if } x_i = y_i, \text{ for all } i,\ 0 \le i \le (N-1) \\ 1 & \text{otherwise} \end{cases}$$

Given a set of sequences in the normal database,
$\{Y_1, Y_2, Y_3, ...Y_M\}$, $|Y_i| = N, 1 \le i \le M$, and a ordered set of sequences in test data, $\{X_1, X_2, X_3, ...X_{Z-(N-1)}\}$,
where $X_s = (x_s, x_{s+1}, ...x_{s+(N-1)})$ for $1 \le s \le (Z-(N-1))$, and the size of test data is Z, the similarity measure assigned the sequence X_S is:

$$\hat{Sim}(X_s) = \begin{cases} 1 & \text{if } Sim(X_s, Y_j) = 1, \text{ for all } j,\ 1 \le j \le M \\ 0 & \text{otherwise.} \end{cases}$$

Finally, locality frame count(LFC) with size L for each size N sequence in the test data is defined as:

$$LFC(X_s) = \begin{cases} \sum_{l=((s-L)+1)}^{s} \hat{Sim}(X_l) & \text{for } s \ge L \\ \sum_{l=1}^{s} \hat{Sim}(X_l) & \text{for } s < L \end{cases}$$

Based on this algorithm, a concise database containing normal sequences with length N can be generated for detecting anomalies. The algorithm is easy and effective, some more sophisticated models do not have significant performance improvement over the original model [81]. In the original work, the sliding window of the STIDE detector was set 6, Lee at al. [48] gave an analysis using *conditional entropy* to explain the selection of the "magic number", but Tan et al. [77] undermined the entropy-based analysis using a random data set. Furthermore, they gave a thorough analysis on the selection of detector window using a synthetic data set [77, 78]. Actually, this phenomena depends heavily on the STIDE's operating environment, and the detector essentially works in an exhaustive way, its performance therefore is effected by the normal data set, any foreign elements or sequences that unincorporated in the normal data set would be detected easily. As Maxion et al. [58] analyzed, STIDE has a blind region under $x = y$ in coordinate, where x-axle represents "size of foreign-sequence anomaly" and y-axle denotes "size of detector window". The existence of blind region cause the detector to suffer from simple exploits by a sophisticated attacker who have fundamental understanding with its operational limits. Therefore, the analysis and construction of normal sequence data set is essential to improve the performance of STIDE. The trad-off between the cost and accuracy is the variant detector window above six.

2.4.2 MCE: Cross Entropy-Based Anomaly Detector

Based on the assumption that the occurrence frequencies of different observable subjects can be measured during a certain time scale, a probability distribution can be used to represent the occurrence pattern during this period. In this model, the sequential property is out of consideration, which essentially is a kind of static method [89]. The method has not been widely used because of its unsatisfactorily performance in some situation. Its basic idea can be described as follows:

Assume $P(M)$ denotes the probability distribution characterizing the behavior of a normal model M and $P_i(M)$, $i = 1, 2, ...N$ denote the occurrence probability of event i among a set of N events, the similarity of two distributions P and Q can be measured using *cross entropy*:

$$\begin{cases} C(P,Q) = \sum_{i=1}^{N}(Q_i - P_i)log\frac{Q_i}{P_i}. \\ C(P,Q) \geq 0, \\ C(P,Q) = 0 \Leftrightarrow P = Q. \end{cases}$$

After determining a threshold for the similarity between P and Q using training data and validation data set, we can decide whether ongoing events set should be considered as intrusive with respective to the normal model. Actually, the performance of this method might be improved significant with the preprocess of data using *information-theoretic* measures that we discussed in last section.

Here, we do not intend to undermine the contribution of the work [89], and we only want to point out that a careful analysis of the operating environments that anomaly detectors work could also obtain the same conclusion as that from expensive *trial-and-prone-to-error* experiments. In their work, the anomaly detector operated with two kinds of observable subjects, one is program profiles based on Unix system calls, another is user profiles based on Unix shell commands. As we know, system calls executed by the same process have certain temporal pattern, namely, system calls from a specific process have the sequential correlation, at least the order between several system calls always keep unchanging. While for the shell command data, although individual user has particular pattern during his/her login session, that is, the token was recorded almost always keep the same entropy, the frequency of tokens rather than the sequential relations have more contribution to the characterization of user behavior. Under such cases, anomaly detectors which can capture temporal characteristics, such as HMM-based anomaly detector, obviously have better performance in the system calls data set than that of in the shell command data set. On the contrary, frequency distributions-based anomaly detector have the inverse performance due to the properties of operating environment. Therefore, after simply but effective analysis of the operating environments, we can get the same conclusion that [89] ever got easily.

2.4.3 Probabilistic Anomaly Detectors

Ye et al. [86] gave a nearly thorough analysis on the probabilistic techniques-based anomaly detectors with computer audit data, including decision tree, Hotelling's T^2 test, chi-square multivariate test and Markov chain. Part of conclusion they obtained was "*...unless the scalability problem of complex data models taking into account the ordering property of activity data is solved, intrusion detection techniques based on the frequency*

property provide a viable solution that produces good intrusion detection performance with low computational overhead."

Among the various probabilistic techniques-based intrusion detectors, expect Markov chain, all the others can be regarded as static intrusion detectors due to their statistical nature (although some ordering property of the observable subjects were also considered). Our analysis on their operating environment is motivated by following questions:

- *Whether* the property of the selected observable subjects have been explored thoroughly?
- *If not*, whether complex models could discover more?*otherwise*, whether frequency property is enough for their operational performance?
- Can we get a conclusion that some information will be lost when only event type of computer audit data are used to characterize system normality?

Here, we only consider the basic data model that all the probabilistic anomaly detectors applied. In the model, the observable subjects, namely, *audit data* are represented as frequency distribution $(X_1, X_2, X_3, ...X_N)$, where N denotes the number of different event in the audit set, and the exponentially weighted moving average method (EWMA) was applied to compute the value of X_i, specifically, if the current event t belongs to the ith event type,

$$X_i(t) = c * 1 + (1 - c) * X_i(t - 1),$$

if the current event t different from the ith event type,

$$X_i(t) = c * 0 + (1 - c) * X_i(t - 1),$$

where $X_i(t)$ is the observed value of the ith variable in the vector of an observation $(X_1, X_2, X_3, ...X_N)$ for the current event t, thus a $M \times N$ vector with M target values is constructed if the observation set has M data points; c is the smoothing constant that determines the decay rate; and $1 \leq i \leq N$. This model can convey not only the relative frequency distribution of N in a sequential events during a certain time scale, but also reflect the intensity of activities. However, from the point of view of the observable subjects, two aspects of the data modelling worth insightful consideration, i.e., the selection of parameter λ, and the correlation among data points. A figure below shows the decay effect of different smoothing constants.

We can see from Fig.2 that after a certain period, the weights drops close to zero, but the speed is different due to the various value of c. For example, when $c = 0.3$, the frequency value of $X_i(t)$ at the current event considers about the past 15 audit events (k=0,...,14), while past 22 events (k=0,...,21) are taken into account when $c = 0.2$. In work [87], c was set to 0.3−a commonly used value for the smoothing constant, other values were not tried and compared. Although we do not expect that some unknown c could improve the modelling performance dramatically, a comparative study should be carried out to insight the impacts of different values, and thus select one for better modelling. Furthermore, c might vary in different situation, due to the drifting of system normality, a constant value thus can hardly characterize all the normal activities well.

Both the normal and intrusive training data can be represented using the frequency distribution representation, and thus probabilistic techniques such as $Hltelling's\ T^2\ Test$,

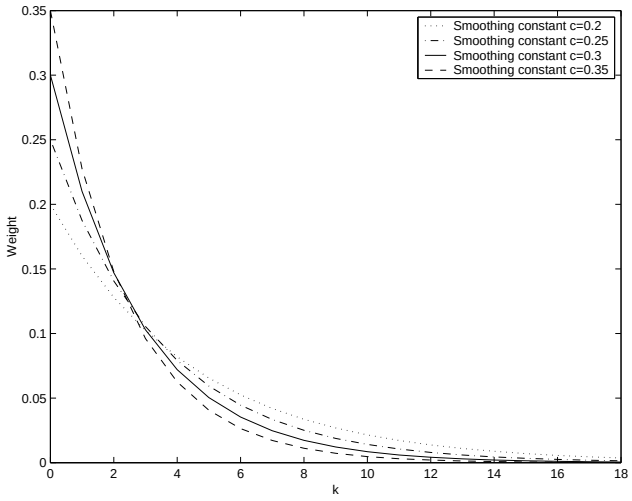

Figure 2.2: Decay effect with different smoothing constants

Chi − Square Multivariate Test can be used to calculate the distance between testing data and training data. An assumption to support this model is that testing data are taken as a whole collection of audit events. Although some ordering property is carried in the model, the knowledge of the unobservable process distribution is ignored. As we know, each process might generate a group of audit events, and there might exist some intervals between those groups, an underlying continuous measurement therefore should be considered in the data model, in order to capture the process shift. Based on this fact, a grouped data EWMA model [35], rather than variables based EWMA, might have more contribution to the characterization of computer audit events.

Additionally, in the original data model, only the audit event type was considered, while other attributes, such as user ID, process ID, session ID, the system object accessed, were omitted. To incorporate those necessary additional information, a multivariate EWMA can be used as follows:

$$X_i(t) = C * O_i(t) + (1 - C) * X_i(t - 1),$$

where $X_i(t)$ is the ith EWMA vector, $O_i(t)$ is the ith observation vector at time t, $i = 1, 2, 3...n$, C is the diag $(c_1, c_2, ...c_p)$ which is a diagonal matrix with $c_1, c_2, ...c_p$ on the main diagonal, and p is the number of variables, i.e., the number of attributes that we are considering. The MEWMA model takes into account all the necessary variables of audit events, and thus can be used to capture the process shift in multi-scales. Although it is much more complex than the univariate EWMA, a better performance is expected to be achieved if some scalability problems are solved well.

Preliminary analysis shows that the characterization of the operational situation has great effect on the anomaly detector's performance. Tracing back to the problems posed in the beginning of this subsection, we infer that more accurate/complex data models might benefit the improvement of anomaly detector's detection performance. However, scalability problem is another obstacle, which was claimed in [88]. The work also proved that the performance of first-order Markov chain is better than that of high-order stochastic models, although the latter one has more complex model (means more expensive computational cost) than the former one.

2.4.4 Comparative Analysis

As former analysis, all the anomaly detectors are specialized by their different detection coverage or blind spot, part of which attribute to operating environment. We hope that a thorough comparison analysis could provide us an approach to combine anomaly detectors together to achieve a broader detection coverage. In fact, the statistical modeling in section 2 facilitates the comparison between those anomaly detectors, in terms of detection capability and operational limits.

Table 2.1: A Comparison between Three Typical Anomaly Detectors

Anomaly Detector	Observation	Main Property		Detection Cost
		Frequency	Ordering	
STIDE [28]	System calls		$\surd$	$O(N \cdot L)$
MCE [89]	System calls/ Shell Command Lines	$\surd$		$O(N \cdot n^2)$
Markov Chain [86, 88]	Audit Events	$\surd$	$\surd$	$O(L \cdot n^2)$
Hotelling's T^2 [86, 87]				$O(L \cdot n^2)$
Chi-square [86, 87]				$O(L \cdot n^2)$

A brief compared results is shown in table 2.1 (Markov Chain, Hotelling's T^2, Chi-square are general called probabilistic anomaly detectors), where N is the size of normal profiles which has different meaning in different anomaly detectors, while L is the size of ongoing trace being detected. 'n' is the number of unique events/states. For STIDE, w is a predefined window size. What we compare here is only the detection cost, while the cost of models' construction are not considered. Note that the detection cost of STIDE can be reduced to $L * logN$, if normal data are stored in an effect form, i.e., forest of trees. The detection cost of probabilistic detectors are differ in specific techniques, for instance, $Hotelling's\ T^2$ requires a large memory to store the variance-covariance matrix and much time to compute the matrix multiplication and inverse, its time complexity for detection nearly $O(N^2)$ ($L << N$), while *Markov chains* or *chi-squre* multivariate test need less computational overhead, i.e., $O(N)$ or so.

Although the original detection models have their own operating environments. Careful analysis allow them to be extended to a broader application field. For example, STIDE was originally developed with system calls of privileged programs, but it can also be applied to audit events provided the scope of activities is not so wide, based on the similar properties of those two observations. Similarly, the probabilistic anomaly detectors that were originally operated with audit events and shell command lines can also be extended to system calls, if enough ordering property are included during the data modeling.

Among those detectors, STIDE has the highest detection capability in general case, because it stores all the unique system calls sequences in the normal profile. Any ongoing traces with system call sequences that never appeared in normal profile will be detected as anomalies (determined by LFC). According to Corollary 2, STIDE has a good convergence due to the high average value of w_μ. Generally, two elements contribute to the higher detection capability of STIDE:

- Observable subjects, i.e., system calls. As we know, system calls of privilege processes is a good level to reflect the user behaviors due to its limited range of actions,

sensitivity to changes, and stability over time. While shell user command lines and audit events have less characteristics compared with system calls.

- Nearly exhaustive searching mode. All the available unique system calls sequences are used to characterize system normality, which constructs a broad boundary to encompass normal behavior.

For frequency-based anomaly detectors, less characteristics of their operating environments, e.g., unpredictable range of activities, instabilities over time, cause them to suffer from low detection capability. According to the corollary 1, only the huge size of normal data set provides them an opportunity to decrease expected error between probability estimation and stochastic generator to a low level.

2.5 Evaluation of the Anomaly Detectors

Another hard stone in the anomaly detection research community is the anomaly detectors' evaluation. Most of existing IDSs take 1998 and 1999 DARPA Intrusion Detection System Evaluations Data Set [63] as benchmark for evaluating their performance, and most researches focus on tallying with detection accuracy and false positive rate of detection methods, rather than the fundamental understanding of evaluation environment. Therefore, the specific design of anomaly detectors based on particular situation, together with some strong assumptions limit their application to a broader application scope.

Mchugh [62] gave a thorough analysis of so-called benchmark data set, and proposed the essential conditions that ideal measurements should have. Briefly, it includes:

- The primary method, i.e ROC (Receiver Operating Curve), to present the results of the evaluation provides no insights into the root-causes for IDS performance, and the more helpful metrics should be developed.

- The curse of the false alarms generation has not been explained clearly, therefore, the useful description of the difference between activities that are identified correctly as an attack and those that provoke a false alarm needs more insightful investigation.

- To make sure that the false alarm rate for synthetic data has an obvious relationship to that of real data, background traffic data characterization is needed for calibrated artificial test data sets.

Up to now, we have not found such work that meet above requirements completely. With the problem that whether the environment regularity has effect on the probabilistic algorithms-based anomaly detectors, Maxion and Tan [59] provided an idea for successful data synthesis, and the result verified their hypothesis. But their model is too simple to interpret more complex anomaly detection models, and some additional observational work from real data is needed. In addition, only juxtapositional anomalies was considered in that model, while temporal anomaly detection was left.

Inspired by those former works, we have a primary idea to generate synthetic data for the general evaluation of anomaly detectors. Although it is still during the process of implementation and verification, we believe that it will contribute to the development of anomaly detection evaluation to some extent.

Firstly, collect pure real normal data source from a real environment, and mapping those collected data into controllable domain (for example, mapping network packets into wavelet domain and approximate host audit data as the Planck distribution respectively).

Secondly, apply some candidate anomaly detectors to the controllable data set, and analyze the data that ever provoked false alarms. This step should be done recursively to prune the data as pure normal data without confused false alarms.

Thirdly, in order to ensure the regularity of processed data, information-theoretic measures could be used to divide the data as smaller but purer ones.

Finally, artificial anomalies (such as foreign symbols or sequences, and rare sequences) are incorporated into the data. One way to make it more effective is to add predefined anomalies one by one, until to a determined amount.

2.6 Concluding Remarks

This work addressed the following issues and provided some potential solutions:

- The operational limits of some anomaly detectors are due to themselves or the particular operational environments they run.
- Whether a better characterization of system normality can improve the performance of anomaly detectors (sometimes obviously, sometimes may not).
- How to select proper anomaly detectors for a specific situation when we take into count the trade-off between performance and cost.
- It is usually hard to find a general way to evaluate existing anomaly detector's performance (including those state-of-the-art ones) in terms of admitted criteria (hits, misses, and false alerts). ROC is generally regarded as a typical but superficial analysis tool.

Those questions have been analyzed and discussed in a general way based on the available achievements, although there are still some problems worth further consideration, and some proposed ideas remains verification and implementation, we believe that future work along this way could contribute additional insight for the research and application of anomaly detectors. Someone may argue that our work are obvious and straightforward, we believe that it is important to develop a framework for the anomaly detection field, including characterization, identification and evaluation of their operating environment in order to guarantee their formal and rapid development, and it seems more important than just pruning detector itself regardless of its insightful understanding and broader application. Obviously, our future work includes the implementation of our proposed ideas, and the further analysis for the operating environment of several anomaly detectors from the view of observable subjects.

Chapter 3

Online Training of SVM-Based Anomaly Detectors for Adaptive Intrusion Detection

3.1 Introduction

As introduced in the first chapter, the available approaches for anomaly detection focus on tallying with detection accuracy and false alarms, the trade-off is always their main concern. Given enough time, most of those anomaly detectors can achieve satisfactory results in terms of the general recognized criteria. However, in practice, intrusion detection is a real-time critical mission, that is, malicious behavior should be detected as soon as possible or at least before the attacker eventually succeeds. In addition, there is usually an initial training period for an anomaly detector to characterize the observable subject's normal patterns, and most existing methods are based on the assumption that high quality labeled training data are readily available, which severely limits their application in practice. As the fact, the computer systems become increasingly complex and much more interconnected, whose collective behavior is intricate because of their interacting organisms, components, and systems. More, multiple users and remote network services are dominating external influences, make that more complicated. The lack of predictability in operating systems and network behavior is essentially due to this complexity, and therefore brings more challenges to the development of effective anomaly detection techniques. To cope with such complexity, anomaly detectors should undergo frequent retraining, to incorporate periodically new examples into the training data for classifying novel attacks, and more importantly, suppressing false alerts triggered by the drifts of those normal behaviors. In this sense, running time and training time should also be considered in addition to detection accuracy and false alarms when designing an adaptive anomaly detector.

In chapter 2 we have pointed out that the development of anomaly detectors usually contains two states. The first step is to carefully investigate the observable subjects and examine the computing environments; the second step is to design the specific detection schemes based on the understanding of the environments, and take fully advantage of the observation's properties. For the first stage, complex systems can be characterized by behavior at many levels or scales. In order to extract knowledge from a complex system, it is necessary to focus on an appropriate scale, or several levels if possible.

As introduced in the previous chapters, three scales are usually distinguished in many-component systems: the microscopic, mesoscopic, and macroscopic. Since our work in this chapter is focused on the system calls executed by privileged processes in Solaris Operating System, an illustrative scale separation can be examined as a case study: the individual system calls and some other atomic transactions (on the order of milliseconds) can be discerned as microscopic level; clusters and pattern of system calls, as well as other process behavior such as algorithms, procedures or even malicious codes can be abstracted as mesoscopic observations, which usually refer to a single process or to a group of processes owned by a single user (on the order of seconds); For those activities of a single user or a group of users in term of computing resources (Disk space, CPU, Memory, etc.) and on the order of minutes, hours, days, and even weeks, they are usually measured as macroscopic observations. In our work, we limit our attention to the strings of system calls, i.e.,privileged processes, over mesoscopic intervals. The analysis of strings of system calls originally was examined by Hofmeyr et al. [34], which intended to detect intrusion attempts from the bottom up, using immunological ideas.

The second stage of the development of anomaly detector, also the key stage, is to explore and utilize those properties of selected observations for the design of specific detection scheme. To date, various methods have been introduced to detect 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 [50], 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 chapter mainly involves:

- 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 [72] 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 [71] 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 [45]. 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 chapter 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 modeling of the data. In section 4, we introduce the effective classification method−Support Vector Machine (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 are implemented to evaluate the performance of our proposed methods, which is described in section 5. Finally, our conclusions are presented in section 6.

3.2 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 [87], Markov process [88], and discriminant analysis [2] from statistics; neural networks [29, 39] from pattern recognition; support vector machines [36, 64] from machine learning; and other clustering methods and classification methods from data mining [46, 47].

Forrest et al [28] 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 [30, 50, 81] to extend and improve the work with increasingly better performance. Warrender et al [81] 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 [86] 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) [88], 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 [50] 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 [36] applied Robust SVM [72], 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 [52]. 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 [26] proposed unsupervised anomaly detection methods with unlabelled data, and Nguyen [64] employed One-class SVM [71] 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.

3.3 Anomaly Detectors and Their Failure Curses

We have conducted an analysis on the anomaly detector's behavior based on the analogy between *anomaly detection* and *induction reference* problem. From the functional perspective, an anomaly detector can be roughly regarded as a simple kind of inductive inference system. In this system, an incoming observation O_i is regarded as a "question", while the normal behavior model M that stored in the memory is regarded as "answers". Given a new O_i, the system tries to find an appropriate answer M_i so that $\mathrm{AD}(O_i) \Rightarrow M_i$. Obviously, the aim of an AD design is to look for effective "answers" that have the highest *a priori* - that has "accurate descriptions". In generating such answers, some primitive normal behaviors have to be previously defined. From probabilistic prediction we can gradually to deterministic prediction, that is, whether the current "question" is an anomalous behavior. 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 AD 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, which have been studied well in

last chapter by formulating a statistical framework.

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 [3] gave an analysis of intrusion detector's base-rate fallacy using *Bayesian Theorem*, which points out the curses of the excessive false alerts generated by intrusion detectors. The main idea can be described as follows:

Assume that when process $M(t)$ is generated, intrusion alerts $\mathcal{A}$ is triggered. Suppose during any time interval Δt, $d$etection accuracy is the probability $Pr(\mathcal{A}|M(\Delta t))$, and $f$alse alarm rate is the probability $Pr(\mathcal{A}|\neg M(\Delta t))$. According to Bayes's theorem, we have,

$$Pr(M(\Delta t)|\mathcal{A}) = \frac{Pr(\mathcal{A}|M(\Delta t)) \cdot Pr(M(\Delta t))}{Pr(\mathcal{A}|M(\Delta t)) \cdot Pr(M(\Delta t)) + Pr(\mathcal{A}|N(\Delta t)) \cdot Pr(N(\Delta t))} \tag{3.1}$$

Derive from the notions from chapter 2, $M(\cdot)$ and $N(\cdot)$ are the stochastic processes of generating malicious operation and normal operation respectively. In the equation, the value of $Pr(M(\Delta t)|\mathcal{A})$ is determined by two terms which are related with $d$etection accuracy and $f$alse alerts rate respectively. In fact, the above equation show clearly that the factor governing the detection rate is dominated by the factor governing the false alert rate, due to the fact that $Pr(M(\Delta t)) << Pr(N(\Delta t))$ when Δt is very large. Therefore, false alert rate has more effect the equation's value than other items due to its larger coefficient, and thus it should be as low as possible in order to increase the value of the equation. Otherwise, many alerts from normal operations would make the system supervisor insensitive and intrusion detector inefficient, which is the motivation of our work to restrain false alerts to an acceptable level.

3.4 Observation-Centric Modelling

As we know, 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 [63] 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 [75] (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 [34] showed that privileged processes in UNIX are a good level to focus on because exploitation of vulnerabilities in privileged process can give an intruder super-user 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.

3.4.1 Original 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 system call is treated as a "word" and the set of system calls generated by a process is treated as the "document" [50]; all the training processes are treated as a set of documents.

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:

$$a_{ij} = f_{ij} \tag{3.2}$$

tf-idf weighting method is defined as:

$$a_{ij} = \frac{f_{ij}}{\sqrt{\sum_{l=1}^{M} f_{lj}^2}} \times log(\frac{N}{n_i}) \tag{3.3}$$

Based on the data model, several text categorization methods were proposed [36, 50] 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.

3.4.2 A New Data Model

In [50], 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) [10]. 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 θ, and is updated according to the equation:

$$\overrightarrow{p}\, f_{s,t}(\theta) = \overrightarrow{p}\, f_{s,t-1}(\theta) + \overrightarrow{p}\, f_{s,P_t}(\theta) \tag{3.4}$$

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 θ 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 θ in the processes p can be calculated as follows at time t:

$$w_t(\theta, \overrightarrow{p}) = \frac{(1 + log_2 f(\theta, \overrightarrow{p})) \times log_2(N_t/n_\theta)}{Z_{\overrightarrow{p}}} \tag{3.5}$$

where

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

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

n_θ is the number of processes that include system call θ;

$Z_{\overrightarrow{p}} = \sqrt{\sum_{\theta \in \overrightarrow{p}} w_t(\theta, \overrightarrow{p})^2}$ 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 θ 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 [34], 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 $< P_i >_{1 \leq i \leq w}$ is a sub-episode of $< P_i >_{1 \leq i \leq n}$, 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 $[P_i, P_{i+m}]$. 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 [85], which uses a time window on the historic data. We only consider the processes within the time window m:

$$N_{\overrightarrow{p}} = (1 - time/m) \cdot N_{\overrightarrow{p}} \tag{3.6}$$

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 [66], 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)* $\cup$ *subsequent(Eject, HG)* is intuitively shown in Figure 3.1.

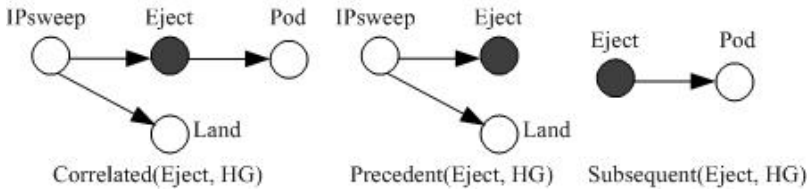

Figure 3.1: Attacks correlation graph

The intrusive session *Eject* is identified as an intrusion for the malicious process *eject-exploit*. Actually, when obviously malicious processes appear, such as *formatexpolit, ffb-exploit, 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.

3.5 SVM-Based Adaptive Anomaly Detectors

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.

3.5.1 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 [14]. In order to solve various problems effectively, several improved SVM such as Robust SVM [72], One-class SVM [71], Online SVM [45] 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 ith 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:

$$\begin{cases} min & \frac{1}{2}\|w\|^2 \\ s.t. & y_i(\langle w, x_i\rangle + b) \geq 1 \quad \forall i, \end{cases} \tag{3.7}$$

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

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \beta_i y_i(\langle w, x_i\rangle + b) + \sum_{i=1}^{l} \beta_i, \tag{3.8}$$

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

$$\begin{cases} min & L_D = \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \beta_i\beta_j y_i y_j \langle x_i, x_j\rangle - \sum_{i=1}^{l} \beta_i \\ s.t. & \sum_{i=1}^{l} \beta_i y_i = 0, \beta_i \geq 0, \forall i \end{cases} \tag{3.9}$$

Above equations only describe linear separable SVMs, and the general dual objective function can be rewritten in a matrix form as follows [45]:

$$L_D = \frac{1}{2}\beta^T K\beta - \langle c, \beta\rangle, \tag{3.10}$$

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 [14] conditions as follows:

$$\begin{cases} \beta_i = 0 \Leftrightarrow y_i f_i > 1, \\ 0 < \beta_i \leq C \Leftrightarrow y_i f_i \leq 1, \end{cases} \tag{3.11}$$

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

$$f(x) = sgn\left(\sum_{i \in SV} \beta_i y_i K(x, x_i) + b\right). \tag{3.12}$$

In order to solve the over-fitting problem of a soft margin SVM due to noisy training data, Robust SVM [72] 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:

$$\begin{cases} min & L_D = \frac{1}{2}\beta^T K\beta - \langle c, \theta\rangle \\ s.t. & \sum_{i=1}^{l} \beta_i y_i = 0, \beta_i \geq 0, \forall i, \end{cases} \tag{3.13}$$

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:

$$\begin{cases} min & L_D = \frac{1}{2}\beta^T K\beta \\ s.t. & 0 \leq \beta_i \leq \frac{1}{vl}, \sum_{i}^{l} \beta_i = 1, \end{cases} \tag{3.14}$$

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:

$$f(x) = sgn\left(\sum_{i=1}^{l} \beta_i k(x_i, x) - \rho\right). \tag{3.15}$$

If α_i meets the subject conditions, ρ can be recovered as:

$$\rho = \sum_{j=1}^{l} \beta_j k(x_j, x_i). \tag{3.16}$$

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.

3.5.2 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 [45], 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 [14], 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 kth 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):

$$f_k(x) = sgn\left(\sum_{i=1}^{|SV_i|} \beta_i^k Sy_i^k K(x, Sx_i^k) + b_k\right). \tag{3.17}$$

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 [45], 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 {
```

$W_k = \{Sx_i^{k-1}, Sy_i^{k-1}\}_{i=1}^{|SV_{k-1}|} \cup S_k$;
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 $W_l = SV_l \cup E_l$ 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 kth 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 [37] and decomposition algorithms are also worth considering in order to speed up the training phase of SVMs.

3.5.3 Convergence of the Modified SVMs

The convergence of the modified conventional SVM has been proved in Ref. [45] 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. [16] and [37] respectively. Similarly, the dual objective function $Eq.(17)$ of One-class SVM can be rewritten involving the working and correcting sets as follows:

$$\begin{cases} min & \frac{1}{2}[\beta_B^T K_{BB}\beta_B + \beta_B^T K_{BN}\beta_N + \beta_N^T K_{NB}\beta_B + \beta_N^T K_{NN}\beta_N] \\ s.t. & 0 \le \beta_B \le \frac{1}{vl}, \beta_B + \beta_N = 1. \end{cases} \tag{3.18}$$

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 β_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 [45] 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 4 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:

$$\begin{cases} min & L_D(\beta_{B'}, \beta_{N'}) \\ s.t. & < y_{B'}, \beta_{B'} > + < y_{N'}, \beta_{N'} >= 0, \\ & -\beta_{B'} \leq 0. \end{cases} \tag{3.19}$$

(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:

$$\begin{cases} min & L_D(\beta_{B'}, \beta_{N'}) \\ s.t. & 0 \le \beta_{B'} \le \frac{1}{vl}, \\ & \beta_{B'} + \beta_{N'} = 1. \end{cases} \tag{3.20}$$

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. □

3.6 Performance Evaluation and Comparison

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 3.2.

3.6.1 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 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

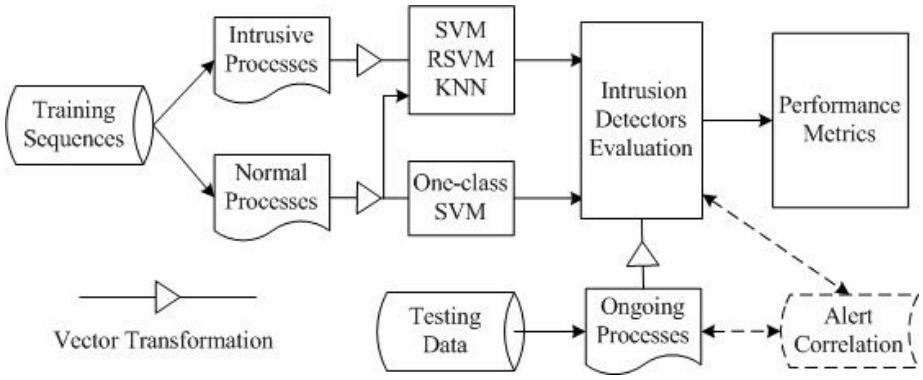

Figure 3.2: Evaluation procedure of the Proposed Intrusion Detectors

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 [62], 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 [50] 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 3.1: 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* [42] 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.

3.6.2 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 3.2). 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*.

Table 3.2: Regulation Parameters of Different Methods

Methods	Regulation Parameters
Traditional SVM	σ, C
Robust SVM	σ, λ
One-class SVM	σ, υ
KNN	k, τ
General parameters	sliding window w=60, time window T=10

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

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 3.3 by adjusting the related regulation parameters after training with clean data and noisy data respectively.

Following conclusions therefore can be derived from the results of table 3.3, which shows the Hits out of 24 attacks while no False Alerts (FA.), and FA. out of 5285 normal processes while 100% Hit:

- 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

Table 3.3: Two Samples of the Experiments with Training Data

	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.

(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 3.4: Comparison of the Detection Accuracy (%) when FP. less than 1%

Training Data	Methods	SVM	RSVM	KNN	One-Class SVM
	Parameters	σ=8, C=10	σ=4, λ=1.2	k=10 τ=0.85	σ=10 υ=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*

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 3.4, we recorded the number of SVs of different SVMs, and as illustrated in Table 3.5, 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 3.5: Comparison of the Number of SVs over Training Data Sets

Training Data	Methods	SVM	RSVM	One-Class 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 3.6 shows the ratio of the training time for original SVMs to the modified method with clean data and noisy data respectively.

Table 3.6: Ratio of the Training Time for the Modified/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 $8.3s$, 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.

3.7 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 data sets.
- 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.

Chapter 4

Constructing Multi-Layered Boundary to Defend Against Intrusive Anomalies: An Autonomic Detection Coordinator

4.1 Introduction

As we have analyzed in the first chapter, the procedure of building an anomaly detection model should involve first studying the intrinsic characteristic of the data (so-called observation) and then selecting a model that best utilizes the characteristic. The selected observations (e.g., system calls traces, network packet logs, shell command line strings) are mainly used to characterize behavioral normality and hence construct anomaly detector's operating environments. However, although a large number of variables could be used to characterize the stage of a host, cataloguing every possibility is impossible, and since the normal behavior of many variables has no obvious pattern, "normality", in most cases, is simply noise. The so-called effective anomaly detection models are mainly based on smoothed or coarse-grained data, which always contain less information than a fully detailed fine-grained data model. Therefore, how to pick out the most efficient observations for system normality characterization is the ever-lasting challenge in anomaly detection domain. One fact has to be realized is that different observations have different properties and characteristics, which might affect anomaly detector's capability of characterizing system normality. In some cases, observation might limit anomaly detector's ability to discover some hidden intrusive attempts. For example, Feng et al. [27] have shown that some attacks can be detected in system call stacks, whilst escaping from system call traces.

In our work, we pay more attention to the observations than the specific design of anomaly detection schemes. Taking the analysis on the anomaly detector's observations as a starting point, we attempt to construct a framework to correlate several observation-specific ADs in order to achieve better performance in terms of broader detection coverage, higher detection accuracy, and fewer false alerts. Specifically, the following questions prompt our work: Whether the combination or fusion of diverse observations or different properties of the same observation provide more information on revealing intrusive anomalies and, if it can, why and how? Whether the detection coverage can be broadened

while fewer false alerts are triggered by the complementary operation of some basic ADs, if it can, how? the complementary operation is deterministic or probabilistic? How to coordinate the observation-specific ADs to work together in an anticipated manner with provision of adaptability and reliability in order to capture the drifts of system normality? Also, an attack might include several stages, and leave traces in various manners for the same system vulnerability. The combination of different ADs is expected to extract concrete behaviors to a sufficiently high-level to detect families of attacks rather than individual instantiations, thereby allowing for the detection of all the attack variants that attempt to exploit the same weakness.

Additionally, the combination of ADs may seek some theoretical foundation from those state-of-the-art design methodologies and paradigms for fault-tolerant systems, which suggest that reliability or dependable computing environments can be achieved via a sequence of system partitioning, subsystem design, and system-wide integration [**?**]. In this sense, improved reliability (tolerate false positive and false negative) of each single AD within an information system (probably in different working environments) facilitates the construction of a multi-layered boundary to cope with attack-driven faults and react with more coordinated diagnosis and effective countermeasures, ultimately achieving improved performance as a whole [90]. In general, we envision a framework in which several levels of data analysis are used as the basis to be combined to yield a single but effective system normality characterization. We envision further an approach in which anomaly detection models are built on a fundamental understanding of their operating environments, and have the adaptability to respond to the diverse demands of various system situations. The hope is that a collection of simple surrogates based on specific observable subjects can cooperate and evolve into generic models with broader anomaly detection coverage and less false alerts.

With the objectives in mind, in this chapter, based on a brief observation-centric analysis on existing anomaly detectors in the first chapter, we formulate the correlation between different observation-specific anomaly detectors as a multi-agent model using Partially Observable Markov Decision Process (POMDP). The proposed detection model including a core component called an autonomic detection coordinator (ADC), is expected to work in a dynamic manner to find an optimal combination to adapt to the changing system situations with satisfactorily performance in terms of predefined evaluation metrics. A basic assumption to support the model is that various operating environment could provide different kinds of normal and anomalous information for system characterization, and thus different anomaly detectors could create a consensus on the identification of anomalies, while intersecting their judgement on false alerts. Moreover, the model could be easily extended to more complex situations, such as a network with distributed anomaly detectors, sensor networks, or wireless networks. Also, the probabilistic nature of the model guarantees it will work in a tolerant manner. Even though one of the individual anomaly detector might fail to work properly, ADC can still collect enough intelligence from the other anomaly detectors, and make a correct decision based on their consensus. Therefore, adaptability, scalability, and dependability enrich the functionality of ADC significantly, compared to an individual AD that works with a single operating environment. After the model formulation, we take a host with Solaris OS as experimental scenario to evaluate our integrated detection model, with emphasis on the attack-centric comparative studies between the detection performance of independent anomaly detectors and that of the model, based on the fundamental understanding of the employed anomaly detectors'

behavior and their respective operating environments.

The rest chapter is organized as follows: Section 2 gives a brief introduction about the related works. In Section 3, we construct a general framework for the correlation of multiple anomaly detectors based on POMDP, and cast the posed problems in the framework with detailed formulation, together with specific solutions. In section 4, we apply our formulated model to four host-based anomaly detectors, and take a host with solaris OS as experimental scenario to evaluate the model's performance. Section 5 reports the experimental results and presents some further discussions. Concluding remarks and future work are presented in the last section.

4.2 Related Work

The motivation for the correlation of different anomaly detection models is intuitive in essence, and many work have been done on this idea, although the research emphasis are not similar. Han et al. [32] combined multiple host-based detection models using a decision tree to lower false alert rate while keeping high detection accuracy. However, their detection models were established and combined on the same layer (i.e., audit events with some related parameters, arguments, and flags), although the utilized information were diverse. In addition, the decision tree essentially is a static combination approach, causing the model to be lack of adaptability, and the performance would deteriorate dramatically with the increasing number of elemental ADs.

Based on the observation that the human experts always attempt to design "root-cause" signatures that "combine" different attack characteristics in order to attain low false alarm rates and high attack detection rates, Giacinto et al. [29] proposed an approach to network intrusion detection based on the fusion of multiple classifiers, while each member of the classifier ensemble was trained on a distinct feature representation of patterns (in their work, each network connection refers to a sequence of data packets related to a particular service, which could be extracted as content features, intrinsic features and traffic features). The classification results of three neural networks were then combined using a number of "fixed" (Majority, Average) and "trainable" (Naive Bayes, A posteriori DCS) fusion rules. The reported results showed that the fused rules could achieve a better trade-off between generalization abilities and false alarm generation than that of individual classifiers. However, in this work, network-based intrusion detection essentially was formulated as a pattern recognition problem, and more efforts were paid to the comparative studies on the fusion approaches rather than the anomaly detection problem itself, specific analysis on the intrusion detection performance was not the emphasis either.

In addition, many collective intrusion detection models have been proposed to countermine distributed attacks by leveraging the information collected from distributed hosts, such as [68, 74, 83], or to improve the accuracy of alarms by correlating diverse observations of multiple heterogeneous sensors [40,65,80]. In those models, local monitors, agents or sensors are used to collect interesting events (from various sources, such as audit data, network packets, SNMP traffic, etc.) or alarm reports, and the distributed architectures provide various communication methods to exchange the local detection information. For instance, Graph-based IDS (GrIDS) [18] detect intrusive anomalies by building a graph representation of network activity based on the reports from hosts, and then infer the intrusive patterns or hostile activities based on predefined rules using TCP/IP traffic be-

tween hosts. Distributed Intrusion Detection System DIDS [74] also takes advantage of a collective approach to detect intrusive anomalies by deploying monitors to watch multiple network links and thus track user activity across multiple hosts. In [80], different measurement reports of the alarms are correlated in a probabilistic way to suppress false alerts, while [65] provides a tool TIAA for interactive intrusion analysis by correlating alerts on the basis of prerequisites and consequences of attacks.

Compared with the existing works, even though starting with similar motivation, our work is mainly focused on the anomaly detection model itself for correlating the anomaly measurements from independent anomaly surrogates, and searches the optimal correlation manners on the system state from learning instead of relying on a predefined set of rules or events. The emphasis of the work is on the analysis of the model's anticipated behavior from a high level standpoint, and the effects of the complementary correlation of different observations on revealing intrusive anomalies. The automatic detection with optimality, adaptability, dependability is also of the concern in our work.

4.3 ADC Modeling

Insightful analysis on the independent anomaly detector's behavior and the fundamental understanding of their detection coverage and blind spots is the preliminary but essential stage for their optimal correlation. With the motivation about the correlation of observation-specific anomaly detectors, and based on the analysis of their general behavior and special properties in the first chapter, an integrated anomaly detection model which mainly contains a core component−autonomic detection coordinator (ADC), is formulated in this section. After the formulation, a well-studied policy-gradient reinforcement learning algorithm is modified to serve as the specific approach to search in the optimal correlation strategy.

4.3.1 Technical Rational and Model Formulation

As described in the last section, the design challenge of ADs is the definition of system normality, and the *measurement distance* between ongoing activity and normal profiles is the key point of detection. During the operation, the AD triggers alert when ongoing activity produces a distance exceeding a specified *threshold*, or beyond a tolerance range, which thus introduces the problem of threshold selection. Intuitively, a larger threshold causes *missed detection* (or *false negative*), while a smaller threshold results in *false alarm* (or *false positive*). In most cases, the measurement threshold is closely related with the AD's detection coverage or blind spot.

Assume that each AD is an independent entity working in its own environment with uncertain perceptions, actions (or responses), and feedback, and each of them responses individually according to its local parameterized detection scheme. Our integrated detection model ADC attempts to combine those independent entities in an optimal way, with the anticipated behavior to suppress false alerts and achieve broader detection coverage. From the perspective of *control theory*, the design is to search in optimal decision strategies to minimize a cost function (or maximize a reward function), as the state of operational environment evolves over an extended period of time. Due to the system inherent uncertainty, either in the way the system state evolves or in the way the state is observed or both, it is not a well-posed problem to optimize the criterion directly. In this sense,

stochastic control formulations, which seek probabilistic strategies rather deterministic strategies, is a more suitable optimization method to meet our realistic objective.

It is worth noting here that our main concern is the behavior (response) of independent ADs, rather than their inner detection schemes. The independent AD decides whether the ongoing activity is legal or malicious, and since each of them only works in its own context, the true system state can only be indirectly observed through their respective detection measurement, and they have to maintain the estimates of the true system state, therefore, the detection problem is partially observable for the entire system. Furthermore, the decision process is a Markov process, because the next state of the system is dependent only upon the current state and the previous decision. Thus, a partially observable Markov decision process is formulated here.

Formally, a POMDP model is structurally characterized by four key elements [1]: a finite state space S of n distinct states, or $S = \{1, 2, ..., n\}$ of the system; a control space U of m distinct actions (or responses), or $U = \{1, 2, ..., m\}$ that are available to the detection policy; an observation space Z of q distinct observations, or $Z = \{1, 2, ..., q\}$; and, a (possibly stochastic) reward $r(i) \in \mathbb{R}$ for each state $s_i \in S$, or in another sense, cost $c_{i,j}(u)$ for state transition from s_i to s_j with a particular control u.

As a POMDP model, the interaction between an independent AD and its environment includes a sequence of decision stages as follows:

1. At time step i (discrete), the system is in a particular state $s_i \in S$, and the underlying state emits an observation $z_i \in Z$ to the AD according to a probability distribution $\nu(s_i)$ over observation vectors.

2. The AD responds $u_i \in U$ in accordance with a randomized policy, based on a probability distribution $\mu(z_i)$ over actions, with known z_i.

3. u_i determines a stochastic matrix $Pr(u_i) = [p_{ij}(u_i)]$, $p_{ij}(u_i)$ is the probability of making a transition from state s_i to state s_j under action u_i.

4. In every system state, the AD receives a reward signal r_i, while its aim is to choose a policy so as to maximize the long-term average of reward ($\mathbb{E}$ is the expectation operator),

$$\eta := \lim_{T \to \infty} \mathbb{E}[\frac{1}{T} \sum_{i=1}^{T} r_i]. \tag{4.1}$$

The above decision process shows that at each time step the AD sees only the observations z_i and the reward $r(i)$, while it has no knowledge of the underlying state space, how the actions affect the evolution of states, how the reward signals depend on the states, or even how the observations depend on the states. From another viewpoint, to each randomized policy $\mu(\cdot)$ and observation distribution $\nu(\cdot)$, the Markov chains for state transitions s_i and s_j are generated as follows:

$$s_i \in S \xrightarrow{\nu(s_i)} z_i \in Z \xrightarrow{\mu(z_i)} u_i \in U \xrightarrow{p_{ij}(u_i)} s_j \in S$$

In essence, all the above parameters can be organized into a family of action-dependent matrices: m $n \times n$ state transition probability matrices F, m $n \times q$ observation probability matrices H, m $n \times n$ transition reward matrices G. $\nu(s_i)$ is essentially a $m \cdot n \cdot q$

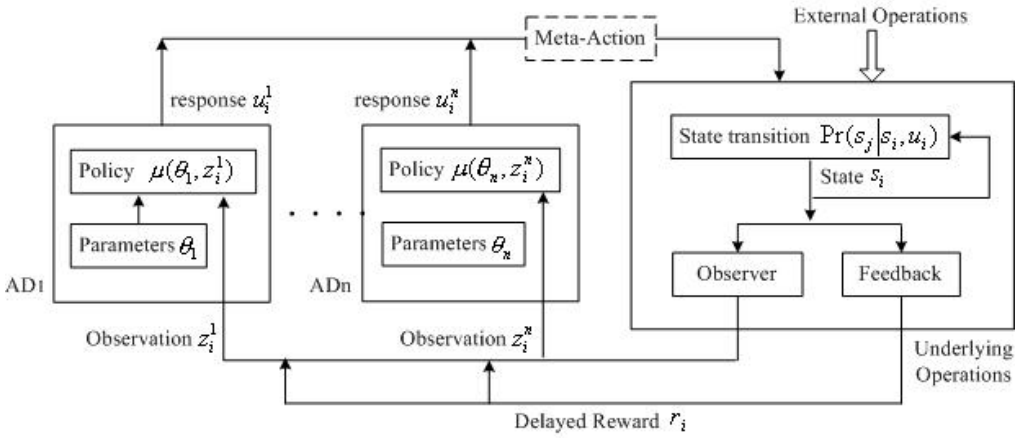

Figure 4.1: Architecture of the Autonomic Detection Coordinator

known observation probability $Pr(z_i|s_i, u_{i-1})$, while $\mu(z_i)$ is a $q \cdot m \cdot m$ action probability $Pr(u_i|z_i, u_{i-1})$. In order to parameterize these chains, we parameterize the policies, so that $\mu(\cdot)$ becomes a function $\mu(\theta, z_i)$ of a set of parameters $\theta \in \mathbb{R}^k$ as well as the observation z_i. The Markov chain corresponding to θ has state transition matrix $P(\theta) = p_{ij}(\theta)$ given by $p_{ij}(\theta) = E_{z_i \sim \nu(s_i)} E_{u_i \sim \mu(\theta, z_i)} p_{ij}(u_i)$. Therefore, equation (8) can be achieved by the parameterized policy with θ:

$$\eta(\theta) := \lim_{T \to \infty} \mathbb{E}_\theta [\frac{1}{T} \sum_{i=1}^{T} r_i]. \tag{4.2}$$

As the detection process of each AD can be formulated as partially markov decision process, the ADC naturally can be modeled as a multi-agent POMDP, as shown in figure 4.1. In the coordinator, several independent ADs with distinct operating environments are incorporated. Each of them sees a distinct observation vector, and has a distinct parameterized randomized policy that depends on its own set of parameters. If the collection of ADs is considered as a single AD, the individual observation vectors can be combined into a single observation vector, and similarly for the parameter vectors and action vectors, while the common goal of those ADs is to maximize the average reward, or minimize the on-average cost of the actions. Effectively, each AD treats the other ADs as a part of the system, and updates its own policy while remaining oblivious to the existence of the other ADs. The only communication between these cooperating ADs is via the globally distributed reward signal.

More formally, the meta-action of the coordinator U contains the cross product of all the responses available to each AD, that is, $U = \{u_1 \times u_2 \times \cdots \times u_n\}$. Because the AD parameters/policys are independent, each AD independently chooses actions that are combined to form the meta-action. For stochastic policies, the overall action distribution is the joint distribution of actions for each AD, $\mu(u_1, u_2, ...u_n|\theta_1, \theta_2, ...\theta_n, z_1, z_2, ...z_n)$. In another sense, ADC's meta-action U_i is essentially based on the estimate of the system state $\tilde{S}_i$, which contains the responses of independent ADs $(u_1, u_2...u_n)$, the proceeding meta-action U_{i-1}, and the proceeding system estimate $\tilde{S}_{i-1}$, or $U_i = \mu(\tilde{S}_i)$.

4.3.2 A Specific Solution

To establish the structure of model, several elements need to be specified via engineering assumptions, formal definitions, and even empirical training:

- A concatenation parameter setting $\theta = (\theta_1, \theta_2 ... \theta_n)$
- The system state space S of cardinality n, the observation space Z of cardinality q, and the action space U of cardinality m
- The state transition probability matrices F, observation probability matrices of ADs H, transition reward matrices G, and the definition of the reward signal r

As the meta-action of the ADC is affected by a concatenation parameter θ, while our aim is to find such a parameter setting (or control policy) for all the ADs that maximizes the expected long-term average reward in equation (8). This is essentially is a kind of direct reinforcement learning problem, which has been discussed in [7, 8].

In a brief word, the algorithm learns to adjust the parameters θ of a randomized policy with observation z_i, and chooses actions according to $\mu(z_i, \theta)$. It involves the computation of a vector q_t at time step t, and it updates according to:

$$q_{t+1} = \rho \cdot q_t + \frac{\nabla \mu_{u_t}(z_t, \theta)}{\mu_{u_t}(z_t, \theta)} \tag{4.3}$$

where $\rho \in (0, 1)$, $\mu_{u_t}(z_t, \theta)$ is the probability of the action u_t under the current policy, and ∇ denotes the gradient with respect to the parameters θ. The vector q_t is an eligibility trace of the same dimensionality as θ; it is used to update the parameters, and guides the policy to climb the gradient of the average reward. Here, we intend to apply a multi-agent variant of the OLPOMDP algorithm [6], which has been applied to solve a routing problem by Tao et al [79], and a multi-neurons learning problem in the brain by Bartlett et al [5]. The OLPOMDP gives a simple way to compute an appropriate direction to update the parameters:

$$\theta_t = \theta_{t-1} + \triangle\theta = \theta_t + \tau_t \cdot r_t \cdot q_t \tag{4.4}$$

where the long-term average of the updates $\triangle\theta$ lie in the gradient direction $\nabla\eta(\theta)$, r_t is the sum of the rewards, and τ_t is the suitable size of the steps taken in parameter space. The key feature of the algorithm is that the only non-local information each AD needs is a global reward signal; AD does not need to know any other information about the system state in order to climb the gradient of the global average reward.

More practically, considering the specific characteristics of the information systems (regardless of single hosts or computer networks), two assumptions need to be addressed to facilitate the algorithm's application:

Assumption 1 *For every given θ, the system converges to a unique steady state $s_0 \in S$ after a number of aperiodic normal operations.*

Specifically, although the system's underlying states are unknown, it will return to a steady state ultimately; that is, the right-hand-side of equation (9) is independent of the system starting state, and converges with probability 1 over all possible reward sequences $\{r_i\}$.

Assumption 2 *For the POMDP-based ADC, which coordinates the independent ADs, the updates of equations (10) and (11) for the ADC are equivalent to those that would be used by each AD.*

That is, if we let z_t^i denote the observation vector for the ith AD , u_t^i denote the action it takes, and θ^i denote its parameter vector, the update equation (11) is equivalent to the individual update equations,

$$\theta_t^i = \theta_{t-1}^i + \tau_t \cdot r_t \cdot q_t^i \tag{4.5}$$

where $\tau_1, \tau_2, \cdots > 0$, $\sum_{t=0}^{\infty} \tau_t = \infty$, and $\sum_{t=0}^{\infty} \tau_t^2 < \infty$, while the vectors $q_t^i \in \mathbb{R}^k$ are updated according to

$$q_{t+1}^i = \rho \cdot q_t^i + \frac{\nabla \mu_{u_t^i}(z_t^i, \theta^i)}{\mu_{u_t^i}(z_t^i, \theta^i)} \tag{4.6}$$

where ∇ denotes the gradient with respect to the AD's parameter θ^i.

In addition, several formal definitions need to be given in order to cast the independent AD's behavior in the POMDP model:

Definition 1 *All the ADs have no knowledge about the exact system states, in some sense, $|S|$ is infinite; the observation set Z=*{*Normal, Attack*}*, and the response/action set U=*{*Silence, Alert*}* according to their specific detection scheme.*

Due to the numerous and various operations, it is impossible for ADs to determine the exact system state especially those normal states. Generally, ADs consider two states *Normal* and *Attack*, while *Attack* state can be further specified according to the severity degree. Based on the the system estimate, ADs thus respond in a deterministic way according to their observations. In addition, to synchronize the responses to ADC from basic ADs, a staging scheme needs to be defined. Suppose an activity (local or remote) happens at time step t, ADs receive different observation streams independently in their own operating environments; let $\sigma(\cdot)$ denotes a general form of the ADs' decision rule, which partitions the infinite measurement space into discretely different decision regions, with each region corresponding to one of a finite number κ (according to the definition 1, $\kappa = 2$) of possible output observations. Given a single-stage measurement (AD-specific) $\chi_0 \in \mathbb{R}$ on stream ℓ, AD i makes a decision with the decision rule parameterized by a threshold value λ_ℓ^i, as follows:

Definition 2 *For every measurement stream ℓ with measurement χ_0, there is a decision rule $\sigma_\ell : \mathbb{R} \to \{0, 1\}$ of the form*

$$\sigma_{\chi_0}(\ell) = \begin{cases} 0, & \chi_0 \leq \lambda_\ell^i \\ 1, & \chi_0 > \lambda_\ell^i \end{cases}$$

where output "0" denotes the 'Normal' *observation and "1" the* 'Attack' *observation. Corresponding actions* 'Observe' *and* 'Alert' *are taken by AD according to their respective observation.*

As mentioned, the parameter θ of ADC is a concatenation of parameters θ^i, which is a row vector with form $\theta = (\theta^1, \theta^2, ...\theta^n)$. Essentially, θ^i is a column vector, and its dimensionality is determined by the number of the ith AD's parameters. It is worth

noting that λ^i in definition 2 is only the threshold determining the distance between normal and abnormal activities, while ADs's actions actually are also affected by other inner parameters. For the easy of discussion, we only consider the key parameter λ^i in the model.

Our next consideration is to to derive the second term of the right-hand side in equation (13) for every independent AD. Since it is difficult to parameterize the underlying detection schemes with λ^i, in order to make the ADs trainable and save computational cost, we assume a general probabilistic model for the ADs' behavior. Specifically, if we assume p is the *a prior* detection probability of AD, the probability of detecting n attacks among N activities is:

$$P_p(n|N) = \binom{N}{n} p^n (1-p)^{N-n} \tag{4.7}$$

taking the distribution as the function of the expected number of successful detections, $v = pN$, the equation becomes:

$$P_{v/N}(n|N) = \binom{N}{n} (\frac{v}{N})^n (1 - \frac{v}{N})^{N-n}$$

When $N \to \infty$, we have,

$$\begin{aligned} P_v(n) &= \lim_{N\to\infty} P_{v/N}(n|N) \\ &= \lim_{N\to\infty} \frac{N(N-1)\cdots(N-n+1)}{n!} \frac{v^n}{N^n} (1 - \frac{v}{N})^{N-n} \\ &= \frac{v^n e^{-v}}{n!} \end{aligned}$$

Obviously, $P_v(n)$ is a *Poisson distribution*. Hence, during a particular period, the probability of no anomaly appears is $P_v(0) = e^{-v}$, or for an AD, $Pr(\text{Slience}) = Pr(u_t = 0) = e^{-v}$. To make ADs' parameters trainable, we take v (which is essentially related with detection probability p) as a function of λ^i, i.e., $v = \varphi(\lambda^i)$. AD's action u_t thus generally obeys a rule $Pr(u_t = 0) = e^{-\varphi(\lambda^i)}$, while $\varphi(\lambda^i) \in (0, \infty)$ is defined as:

$$\varphi(\lambda^i) = \frac{M \cdot \lambda_t^i}{d_t^i(\ell)} \tag{4.8}$$

where M is a constant which can be determined during the training stage with knowledge of the number of injected attack operations, and λ_t^i is the threshold of ith AD at time instant t, while $d_t^i(\ell)$ denotes the measurement distance between ongoing observations ℓ and the normal patterns. Assumption 2 shows how to update the threshold λ_t^i in the direction that maximally increases the long-term average of the reward. From equation (13), we easily derive

$$\frac{\frac{\partial}{\partial \lambda_t^i}\mu_{u_t}}{\mu_{u_t}} = \begin{cases} -M/d_t^i(\ell) & \text{if } u_t = 0 \\ M \cdot e^{\frac{-M\cdot\lambda_t^i}{d_t^i(\ell)}} / \{d_t^i(\ell) \cdot (1 - e^{\frac{-M\cdot\lambda_t^i}{d_t^i(\ell)}})\} & \text{otherwise} \end{cases} \tag{4.9}$$

To complete the picture we need to define reward/cost functions for the ADC's behavior, which is used to guide the improvement of its detection performance. As we know, in the anomaly detection domain, following cases always happen, more or less:

- normal behavior is detected as normal (*correct*)
- normal behavior is detected as anomaly (*false positive*)
- abnormal behavior is detected as normal (*false negative*)
- abnormal behavior is detected as anomaly (*correct*)

Here we want to employe an expected cost function (regardless of AD-specific detection schemes), which is essentially equivalent to equation (8), to express a minimum probability-of-error criterion where errors (false positive & false negative) corresponding to ADC's action "Alert" while in state *Normal*, or "Silence" when not in state *Normal*,

$$C(\lambda_t) = \sum_{i=1}^{n} Pr(s_i)[R(0, s_i)Pr(d \leq \lambda_t | s_i) + R(1, s_i)Pr(d > \lambda_t | s_i)] \tag{4.10}$$

where d is a general form of measurement distance, λ_t is threshold at time step t, $Pr(s_i)$ is a *prior probability distribution* which can be obtained via training data, and, $R(A_0, s_i) : \{0, 1\} \times S \rightarrow \mathbb{R}$, represents rewards with action $A_0 \in \{0, 1\}$ when the true state is s_i, and it can be defined according to various system situation and security demands. Obviously, the ADC's expected optimal behavior can be achieved by minimizing $C(\lambda_t)$, or via selection of the optimal λ_t^*.

Based on our specific assumptions and formal definitions, a modified version of algorithm OLPOMDP [7] can be used to describe the AD's behavior:

Algorithm *Reinforcement Learning of ADC meta-action*

Given:
Coefficient $\rho \in [0, 1)$,
Step size τ_0,
Initial system state s_0,
Initial thresholds of independent ADs λ_0^i,
i.e., θ_t^i in concatenation vector θ.
Begin
for discrete time instant $t = 1, 2, \cdots$ **do**
Get ongoing observations and their corresponding measurement stream ℓ.
Generate action u_t^i according to the specific detection scheme and definition 2.
The ADC broadcasts the cost signal r_t.
Update q_{t+1}^i according to equation (13) and (16):
if the previous actions is "*Silence*" (i.e., $u_t = 0$)
$q_{t+1}^i = \rho \cdot q_t^i - M/d_t^i(\ell)$.
else
$q_{t+1}^i = \rho \cdot q_t^i + M \cdot e^{\frac{-M \cdot \lambda_t^i}{d_t^i(\ell)}} / \{d_t^i(\ell) \cdot (1 - e^{\frac{-M \cdot \lambda_t^i}{d_t^i(\ell)}})\}$.
end if
Update θ_{t+1}^i according to equation (12):
$\theta_{t+1}^i = \theta_t^i + \tau_t \cdot r_t \cdot q_{t+1}^i$.
end for
end

Note that r_t in equation (12) is the sum of costs that ADC have got so far, and q_t^i is a trace of the same dimensionality as θ_t^i, with $q_0^i = 0$; $\rho \in [0, 1)$ is a free parameter to control both the bias and the variance of the estimates produced by the algorithm. It has been shown that provided the bias is sufficiently small, it will converge to a region of near-zero gradient, which thus can be extended to the multi-detector environment, and the algorithm does not need access to the underlying state and does not make use of recurrent states. Except the global reward signal r_t, each AD only applies local information in its parameter updates. A multi-agent OLPOMDP is thus an ideal approach to optimize the cooperative responses of different anomaly detectors.

4.3.3 Practical Considerations

The modeling of our ADC and its anticipated behavior are supported by two assumptions. The first assumption is usually but not always strictly satisfied. During the online training phase, if the system is safe from attacks, no matter what the initial state $s_i \in S$, the system will return to a steady state s_0 after a number of normal operations. But once the system is attacked, the probability of achieving a steady state decreases dramatically. Even though such case could happen, equation (12) and (13) may still perform well when the process parameters vary slowly over a large number of stages. Strictly speaking, the second assumption can be regarded as a theorem deduced from equation (10) and (11), which is theoretically proved in [5,7]. So, it is always satisfied during the modeling process.

Table 4.1: A Simple Comparison between Four Basic Elements

Anomaly Detectors	Observation	Main Property		Parameters	Detection Cost
		Frequency	Ordering		
MCE [89]	Shell Command Lines	√		λ, L	$O(N \cdot n^2)$
Markov Chains [86, 87]	Audit Events	√	√	λ, L	$O(L \cdot n^2)$
STIDE [28]	System calls		√	λ, w, L	$O(N \cdot L)$
KNN [50]	System calls	√		λ, k	$O(N \cdot n^2)$

'L' is the size of ongoing observation, 'n' is the number of unique events/states, 'w' in STIDE is sliding window size, 'N' is the size of normal profiles which has different meaning in different ADs

During the modeling process, we limit our attention on the controllability of the ADC, some other considerations such as computational cost, independent ADs' underlying detection scheme, etc., are not of our main concerns, and essentially, the computational cost that the ADC needs is much less than that of the independent ADs. Several additional attractive properties in the following strengthen the model's advantages, and might broaden its application field:

- *Adaptability* In our model, reward signal is the only information shared by elemental ADs, and the cooperation between those ADs need not consider their explicit intercommunication; this allows the ADC to adapt to diverse system situations by setting different reward signals. Furthermore, with the adjustable reward signal, the ADC is expected to be capable of capturing the drifts of system normality in a dynamic manner, through periodical training.

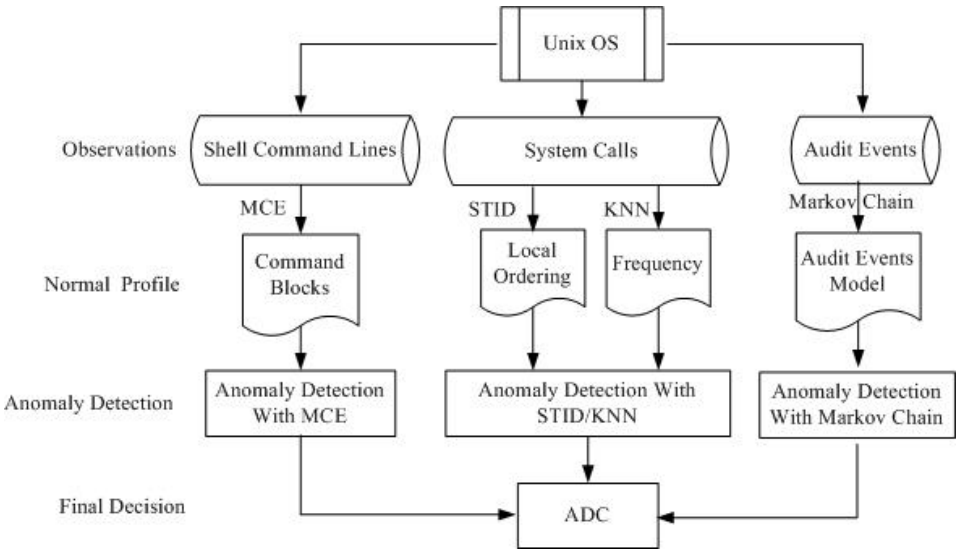

Figure 4.2: Architecture of the Multiple Observation-specific ADs Combination

- *Scalability* As shown in Fig. 1, the ADC's distributed architecture allows different numbers of independent ADs to be incorporated into the model without loss of performance. Furthermore, the ADC might achieve more intelligent controls based on the consensus with increased population of basic ADs. This flexibility thus benefits the extension of the ADC to more complex environments, such as a network with several dominated hosts/servers, sensor networks, and wireless networks.

- *Dependability* Because the ADC's action depends on the overall responses of a group of dependent ADs, even if one or some of them suffer from faults, the ADC still has the ability to provide reliable decisions with high confidence. Additionally, because the independent ADs are observation-specific, even though some sophisticated intrusive anomaly can escape from the detection of a particular AD with a knowledge of its blind detection region [78], other ADs in different layers still have the probability of revealing it.

- *Optimality* The key feature of our model. The underlying working manner is essentially formulated as an optimization problem with the objective function (8) or (17) and some constrained conditions. Assumption 2 shows that the simple update rule modifies the parameters of ADs in the direction that maximally increases the average reward, which leads to parameter values that locally optimize the performance of the independent ADs. ADC's general behavior is thus anticipated to be found as the optimal correlation strategy. The theoretical foundation about this optimization can be found in reference [6, 7].

4.4 Experimental Scenario−A Host-based ADC

This section describes the implementation and evaluation of a host-based ADC prototype, which intends to construct a multi-layered boundary to detect host-based intrusive operations.

Table 4.2: Statistics of the Data Source

Data Category\Source		No. of Command Lines	No. of Audit Events	No. of Processes
Training Set (Normal)		5,600	62,100	640
Testing Set	Normal Data	5,640	70,780	690
	Masquerader	2127	850	272
	Other Attacks	*no trail*	*uncounted*	35

4.4.1 ADC Setting

Based on the observation-centric analysis in section 3, and taking into account following considerations, we employed four typical ADs (shown in table 4.1) as basic elements to be incorporated into ADC, while their detailed descriptions can be found in their respective references.

1. the trade-off between the computational cost and detection performance is the main consideration,
2. since we take Solaris OS as the experimental platform, the selected ADs are host-based, and observation-specific,
3. every AD has sensitive parameters that can be controlled to impact observations,
4. for the easy of control and analysis, the number of independent ADs should not be too large

In table 4.1, λ, as defined in the definition 2, is the threshold determining the similarity between ongoing observations and those in normal profiles. Besides this common parameter, the observation of ADs are also affected by other inner parameters. For instance, for STIDE, the window size of system call sequences w, the locality frame count (LFC) L can also be adjusted to impact the observation. For MCE (Minimum Cross Entropy based on frequency distribution), the length of command blocks L is also an adjustable parameter (but usually according to the length of login sessions). For KNN, k, the number of nearest neighbors of the ongoing process, is also a key parameter for its observation. While for the Markov Chain detector, the size of the observation window L is often regarded as an important parametric variable which affects the difference between two sequences. However, because most of those inner parameters are closely related to the training phase, we only include λ^i into the concatenation parameter vector θ here, that is, $\theta^i = \{\lambda^i\}$, while other parameters are not included. In addition, figure 4.2 shows a general combination architecture for ADC's working environments. MCE works with the user shell command lines by extracting its frequency property; Markov Chains mainly takes advantage of the ordering property of audited events; both STIDE and KNN operate with system calls of privilege process, but the main property they exploit are different, i.e., local ordering and frequency respectively.

4.4.2 Scenario Description and Data Collection

An intrusion instance is exemplified in the following to show the operating scenario of our ADC. A keyboard masquerader or remote interloper takes control of a terminal/host, and

Table 4.3: Attacks List in the Experiments

Attack Category	Attack Description	No. of instances
Masquerader	access to programs and data as an imposter by controlling the keyboard	850 commands
Buffer Overflow	*xlock* heap buffer overflow vulnerability	2
	eject buffer overflow vulnerability	3
	lpset buffer overflow vulnerability	3
DoS	Exhausting Disk Space (with *dd*)	2
	Exhausting the Memory	1
	Consumption of process table	2

then takes advantage of the legitimate user's privileges and access to system programs and data. The intruder may attempt to read or write access to private data, acquire unauthorized system privileges (or even abuse of legitimate privileges), and install some softwares such as *Trojan* for further malicious behavior. For the sophisticated intruder with knowledge of AD installed in target terminal, he might take some seeming legal tricks to surpass the detection coverage. In such activity, the intruder leaves trace data, in various forms, to victim terminal, such as shell command lines (especially for keyboard masquerader) with corresponding audit events, privilege processes with system calls, etc. The ADC is thus expected to detect those anomalies during the malicious attacks based on the trace data.

To validate ADC's functionality and performance, we need a scenario as described to implement the prototype and conduct concerned evaluation. However, to the best of our knowledge, there is no true trace data in the open literature that meets our experimental demands. Therefore, we have to collect, combine and formulate the experimental data with our particular considerations. To simulate the scenario, we need two role-players, Alice and Bob, to operate in a same Solaris 8.0 operating system (SunOS release 5.0). Alice plays the role of a legitimate user, while Bob pretends to be a masquerader.

The following is a summary of the procedure that was performed in order to create simulation scenario and collect the experimental data,

1. Alice operates in the host as a legitimate user, and executes the system programs under normal conditions to obtain records of normal usage, to obtain normal data.

2. Alice executes some known exploits to obtain the data recording attacks as the part of training set.

3. Bob executes a series of system programs under normal conditions, and operates with objective to steal some privacy data that regarded to be confidential to obtain masquerader's trace data.

4. Bob downloads and executes some published exploits and determine the corresponding system programs that those exploits misuse, to obtain the data recording the occurrence of the attacks.

5. Using the normal data obtained from step 1, and the labeled attack data obtained from 2, train ADC and determine its behavior evolvement. Using the data obtained

from step 3, evaluate the ADC's capability to detect masquerader attack. Using the data obtained from step 4, evaluate the ADC's performance in terms of detection accuracy and false alarms.

During Alice's four-week operation, she usually used text editor (*vi*, *ed*, etc.), compiler(*gcc, cc, etc.*), and some system programs(*ps, lpr, sendmail etc.*) on the machine SunBlade 1500, meanwhile, she also executed several attack instances to train the ADC. Excluding wrong commands and some noisy data, while keeping repeated ones, a total of 132,886 records of BSM audit data and 11,240 shell command lines (using the shell *.history* file to log all truncated commands without additional information) were obtained, and these data were roughly averaged as part of pure training set and as testing set. Note that during the collection of shell command lines, we also recorded the related audit events and executed processes in terms of system calls, as BSM provides the monitor of the execution of system calls by all processes launched by the user. However, considering the processes in user mode usually cannot harm the system security, we only recorded those processes in kernel model that require services from system kernel. Additionally, to collect the attack data, Bob pretended to be a masquerader/attacker and operated in the same host. Besides some normal operations, he carried out a series of attacks, including 8 cases of *local buffer overflow* and 5 cases of *DoS*. A small batch of Bob's commands history (2127 audit events, 850 command lines, and 272 processes) were added into testing set as a masquerader trace data, together with the intended attack data. Table 4.2 and table 4.3 shows the experimental data we used in detail.

4.4.3 Structural Specification and Parameter Setting

To cast ADC model in the experimental scenario, its statistical structure and related parameters have to be specified. Obviously, a choice for S reflecting the true and fine-grained state of the information system, especially those normal states, is neither meaningful for security concerns nor feasible for practical implementation. In this sense, S must abstract the operating environment at a controllable level to reflect the coarse-grained state transitions, namely, the changes between models of anticipated normal activity and the stages of anticipated attack. Taking into account the specific experimental scenario, we assume following possible states:

- N, *normal* state without any attacks,
- A_m, local *masquerader* attack,
- A_b, local/remote *buffer overflow* attack,
- A_d, local/remote *DoS* attack.

The state space is thus $S = \{N, A_m, A_b, A_d\}$, or $n = 4$, and the possible state transitions that ADC intends to observe is $N \rightarrow (A_m) \rightarrow A_b(A_d)$.

As described in definition 2, each AD has two kinds of observations, and takes two corresponding response in a deterministic manner. ADC's observation space is thus a combination of responses from independent ADs, that is, Z={MCE:0, MCE:1, MK:0, MK:1, STIDE:0, STIDE:1, KNN:0, KNN:1}, or $q = 6$. ADC's actions, *Silence* or *Alert*, are thus evolved based on the combination of AD's observations and proceeding feedback rewards, which is essentially a probabilistic decision process. With the specified

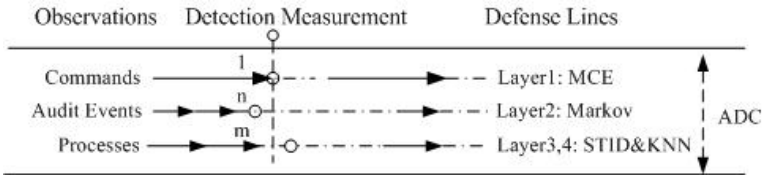

Figure 4.3: Correlation of Detection Measurements Among Different Layers

Table 4.4: Independent ADs' Default Parameters

	MCE	Markov Chain	STIDE	KNN
Sequence length(L)	30	15	6 (LFC=20)	variable
Threshold (λ)	0.45	0.80	0.6	0.72 (k=10)

'k' is the number of nearest neighbors of the ongoing observation

structure, another essential step is to derive a conditional probability chain, including $Pr(s_j|s_i, u_i), Pr(z_i|s_i, u_{i-1}), Pr(u_i|z_i, u_{i-1})$. In essence, these conditional probabilities are the relative frequency of each random event based on the specified POMDP structures S, Z, U, C. Practically, those probability distribution can be estimated, or empirically approximated by training. *Machine learning* theory shows that, the more samples provide in the training data, the more accurate the probability is estimated, and thus the more effective the ADC is applied to the testing data.

In addition, a reward function needs to be specified to guide the evolvement of ADC's anticipated behavior, which mainly considers two types of errors, that is, false positive and false negative,

$$r(s_j|s_i, u_i) = \begin{cases} r_1, & s_i \in \{A_m, A_b, A_d\}, u_k = \text{Silence} \\ r_2, & s_i \in N, u_k = \text{Alert} \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

Furthermore, to simply the training stage, the independent ADs' initial parameters ADC employed were directly derived from their original version, as shown in Table 4.4. Thus, the parameter vector θ is a matrix with size $N \times M$, where M is the number of basic ADs, N is the number of controllable parameters. The initial concatenation parameter θ_0 is therefore denoted as following, but in actual experiment, we only adjust the first row of θ, i.e., $N = 1, M = 4$.

$$\theta_0 = \begin{pmatrix} 0.45 & 0.80 & 0.60 & 0.72 \\ 30 & 10 & 6 & 0 \end{pmatrix}$$

4.5 Experimental Results and Analysis

In the simulation-based experiment, we intend to explore three problems: How ADC evolve its anticipated behavior during the training stage? How about the ADC's performance on suppressing false alerts? How about the ADC's detection coverage/blind spots on detecting known and novel attacks?

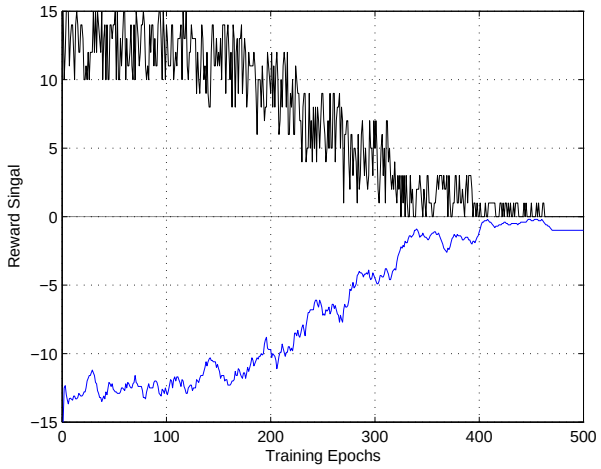

Figure 4.4: Reward Signal of the Coordinator During Training Phase

4.5.1 Training Procedure

The training procedure generally includes three steps: train independent AD's behavior to determine their threshold λ, and create normal profiles; train ADC's behavior to search for an optimal concatenation vector θ; train ADC with trained θ to obtain its probabilistic actions. Here the first step is omitted, and the procedure mainly focuses on the later two steps.

As shown in table 4.2, 5,600 command tokens with 62,100 audit events and 640 processes were taken as training data. Therefore, all 5,600 command tokens were used to create a distribution-based behavioral model. Corresponding audit events and processes were also used to create normal profiles for Markov Chains, STIDE, and KNN respectively. Since the amount of the available data are limited, we used joint sets, that is, half of training data were interleaved with half of testing data (altogether 5,620 command tokens, 66,400 audit events, and 660 processes) to train the ADC. As every login session (i.e., from login to logout) contains about 30 command tokens, for simplicity, we used a constant window to partition command tokens, together with corresponding audit events and system calls. Hence, a total $\lfloor \frac{5620}{30} \rfloor = 187$ commands blocks were available, meanwhile, corresponding audit events and system calls that executed by processes were also extracted as input to respective ADs. Since the independent ADs have different observations and detection measurements, a staging-scheme needs to be developed to synchronize their reports. Here, we take the most outer layer, namely user command lines, as the baseline of the ADC's decision stage, that is, ADC take actions at each command trace. Figure 4.3 shows the logic relationship between the detection stages of independent ADs.

Since the training data contains only normal data, we set the items in equation (18) as $r_1 = 0$, $r_2 = -1$. A total reward signal is then calculated after one pass through the sequence data concatenated by observation traces (the ideal value should be 0 if there is no false alarms). During the decision stage, to simplify the consensus strategy, any false alarm reported by any AD would led to ADC's action *Alert*, with penalty to all ADs. Figure 4.5 depicts the evolvement of the ADC's behavior during the training phase (with 500 training epochs, parameters ρ=0.90 and $\tau_1 = \tau_2 \cdots = 10^{-3}$, $M = 1$). The upper part of the figure shows the changing of the number of false alerts in the training phase, and

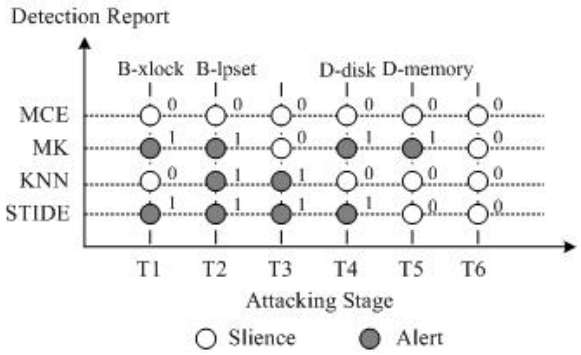

Figure 4.5: Reports of the Basic ADs During Attack Training

the lower part of the figure shows the average reward signal (to manifest the trend, ADC only considered the past 10 passes, i.e., T=10 in equation (2)). The figure shows clearly that the ADC had incrementally improved performance during the training phase, as the reward signal improves, on average, over time to an optimum. We found that after the 462th pass, there was no false alert triggered. After being trained, the parameter vector of coordinator θ is:

$$\theta = \begin{pmatrix} 0.42 & 0.84 & 0.69 & 0.79 \\ 30 & 10 & 6 & 0 \end{pmatrix}$$

Another training procedure is to achieve the ADC's probabilistic behavior, therefore, a batch of training data that contains both normal and attack data are needed. To obtain such data, during Alice's four-week-operation, she also executed 2 cases of `buffer overflow` (`xlock` & `lpset`) and 2 cases of `DoS` (`Attempts to Exhaust Disk Space` and `Memory`). Here, to save the testing data and training time, we only used artificial attack data (those related processes and audit events) to train the ADC's probabilistic action, and thus achieved the consensus strategy during the attack stages. Figure 4.5 simply shows the training results, where T_i means the ith attacking stage in terms of 30-command-block (executing scripts with exploit codes). Note that in T_3 no exploit code was executed, while KNN and STIDE still reported *Alert* due to operations in T_2.

4.5.2 Testing of False Alarms

To evaluate the capability of the ADC of suppressing false alerts, we tested the trained ADC using the normal testing set in table 4.2. Similar to the processing of training data, the testing data was also divided into 188 commands traces (each trace contains 30 command tokens), together with their underlying audit events and processes. Figure 4.6 shows the relationship between the average false alert rate (the number of false alerts over the number of command traces) and the number of command traces used for testing data. Since the ADC gives the report with the pace of each command trace, we compared its performance with that of MCE (with initial parameter), which also reports once on every command trace (computed with the command traces in training data).

The figure shows clearly that the ADC suppressed false alerts significantly compared with MCE. Specifically, ADC generated its first false alert at the 101th command trace (i.e., F.P.=0.99%, first 94 command traces has been used to train ADC, thereby no false alerts were triggered until the 101th command trace). At the 183th command trace,

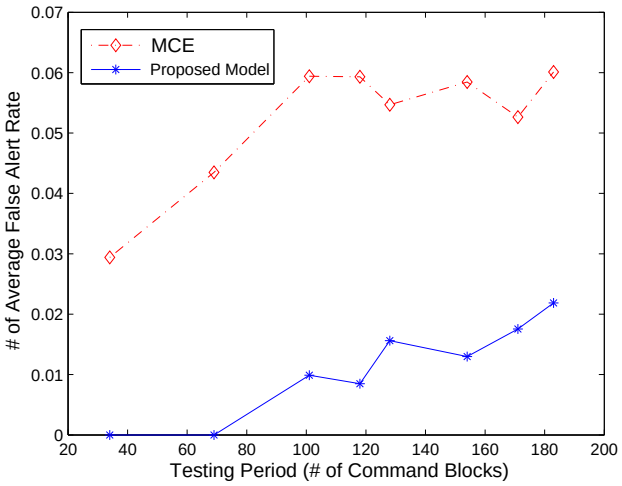

Figure 4.6: False Positive Rate on Testing Data

MCE has generated 11 alerts, i.e., F.P.=6.01%, while the ADC only generated 4 alerts, F.P.=2.19%. We found that at the 128th and 171th command traces, MCE did not report false alert, while ADC reported, which means that one of the other 3 ADs has made wrong reports. Although the analysis of other three detectors are helpful to insight into the story, we did not carry it out here due to the intractable data partition, and the lack of a compelling need to do. In addition, the parameter used by the MCE was directly derived from the ADC rather than by individual training, therefore, we can not rule out the possibility that the MCE might achieve better performance after being trained and parameterized carefully with another training dataset.

4.5.3 Detection of Common Exploits

In addition to the false alerts, detection accuracy is also our concern, which needs careful analysis.

Firstly, we evaluated the ADC's masquerade detection performance. 850 command tokens (with underlying 2127 audit events and 272 processes) operated by Bob were truncated into 28 command traces (each login session also contains 30 command tokens or so), and injected at randomly selected positions, without replacement, into the stream of original 188 command traces (a more complicated case is to inject the command traces into the command tokens instead of command traces; in such a case, the boundaries between the traces might generate uncontrollable false alerts). Meanwhile, the underlying audit events and processes that have been executed by the '*masquerader*' were also injected into the respective normal observation traces (as shown in Fig 3).

The results is shown in Table 4.5. Among total 216 command traces (188 normal + 28 anomalous), MCE detected 20 out of 28 anomalous command traces with a F.P. 11.17% by regulating the threshold to 0.38. After this detection spot, the F.P. raised sharply to 100% with a total 21 anomalous command traces being detected. While the trained ADC detected 23 anomalous command traces with a F.P. 9.57%. A careful comparative study showed us the reasons contributing to the better performance of the ADC:

- MCE only worked in the command traces layer, while the ADC also considered

Table 4.5: Comparison of masquerade detection results between MCE and ADC

Methods	MCE	ADC
# of normal command traces	188	188
# of anomalous Command traces	28	28
Traces size	30	30
Hits(%)	71.43	82.14
Misses(%)	28.57	17.86
F.P.(%)	11.17	9.57
Total Detected	21	23

Table 4.6: Comparison of detection performance between ADC and Basic ADs

ADs/Statistics	Buffer Overflow Attacks							
	$xlock(1)$	$eject(1)$	$lpset(1)$	$eject(2)$	$xlock(2)$	$lpset(2)$	$eject(3)$	$lpset(3)$
MCE	×	×	×	×	×	×	×	×
Markov Chain	✓	✓	✓	✓	✓	✓	✓	✓
STIDE	✓	✓	✓	✓	✓	✓	✓	✓
KNN	×	✓	✓	✓	×	✓	✓	✓
ADC	✓	✓	✓	✓	✓	✓	✓	✓

'✓' denotes hit, '×' denotes miss

ADs/Statistics	DoS Attacks					Metrics		Threshold
	$Disk(1)$	$Memory(1)$	$PT(1)$	$Disk(2)$	$PT(2)$	Hits(%)	F.P.(%)	
MCE	×	×	×	×	×	0	0	0.42
Markov Chain	✓	✓	×	✓	×	84.62	4.35	0.88
STIDE	✓	×	✓	✓	✓	92.30	3.48	0.75
KNN	×	×	✓	×	✓	76.92	5.36	0.95
ADC	✓	✓	✓	✓	✓	100.00	1.01	–

"PT" denotes process table attack

the reports from other three ADs. The most important one is Markov Chain, which works in audit events layer (actually, the STIDE and KNN have contributed nothing since there were no intrusive process appeared during the masquerader detection).

- An intruder always tries to hide the not-so-frequently-used commands essential for intrusion by adding some frequently used commands, while the additional audit event information can be used to extract out the redundant information about the particular command(s). Therefore, even if the malicious attempts could not be detected in the command traces, ADs work in audit events or executed processes might reveal those anomalies.

- 5 "masquerader" login sessions contain more than 30 command tokens, the truncation to the size of 30 might generate more false alerts. Another 4 login sessions have the same seeming with "Normal" ones, causing them to be free of being detected by MCE and ADC. This might also because "masquerader" Bob actually is a legitimate user, even though he has attempted to make some artificial deviation.

Secondly, the trained ADC was used to detect the injected attacks that shown in Table 4.3, and its performance was compared with that of independent ADs. In this test, *detection accuracy* is defined as the ratio of the detected attacks to all the injected attacks (hidden in 35 intrusive processes, any detected intrusive process represented the detection of the corresponding attack). *false alert rate* is the ratio of the misreports to all the normal processes (total 690). To simplify the experiment while keeping its validity, we assumed that false alerts would not be generated by those normal traces that have been used in the last experiment for testing false alerts, and the ADC takes probabilistic behavior according to the consensus strategies from training stage. The initial parameters used by the basic ADs were directly derived from the ADC, whereas in order to investigate the relationship between detection accuracy and F.P., we had to adjust them independently. Although ROC curve is a traditional method to describe the relationship between the detection accuracy and F.P., it can hardly provide us a fundamental and insightful understanding of the detection story. A careful attack-centric analysis with comparative studies is more helpful.

Table 4.6 shows the detection results of the ADC, and the best trad-off between the detection accuracy and F.P. of the basic ADs by adjusting their respective thresholds (a higher detection accuracy would cause a dramatic increase of false alerts). Specifically, we have following observations:

- since the intrusive processes were injected into the normal processes without corresponding command traces, MCE always took action '*Observe*'; Actually, since the exploit code is written and executed in the script, it leaves nothing in the command traces layer. But before that, the local intruder might reveal some unique behavior in commands to achieve his/her goal.

- the ADC detected all the injected attacks by combing the reports from elemental ADs, while its false alert rate was very low (i.e., 7 among 690 processes were misreported); The novel Buffer Overflow attack `eject` and DoS attack by exhausting `Process Table` were detected.

- for the Buffer Overflow attacks, which always attempt to obtain elevated privileges by gaining a *user→root* transition, most of detectors could them out with a low F.P..

- Both STIDE and KNN detected *DoS* attack by the consumption of process table, since this attack generated excessive amount of '*fork*' system calls. Although they did not show any abnormality in system call transitions (abuse of a perfectly legal action), which made Markov Chain fail to detect successfully, the high frequency of '*fork*' and the lack of the corresponding local ordering in our training processes revealed its anomaly.

- Compared with the ADC, the basic ADs had to trigger more false alerts in order to achieve a higher detection accuracy. For instance, the detection of the consumption of process table cause many normal processes to be misclassified by STIDE and KNN as abnormal ones.

Although the attacks in our experiments are not so sophisticated, they did help us to insight into the detection coverage of various detectors. For instance, Markov Chain performs well on detecting 'Buffer Overflow' attacks, based on the state transitions (micro), while STIDE can detect anomalies based on the local ordering of observations (macro states transition), and KNN is good at revealing those attacks showing high frequency of a particular observation. The complementary operations of those ADs might provide a broader detection coverage, and probably abstract some specific attack instances to a higher level to disclose the "root-cause" of anomalies and to conduct post-analysis of alert correlations.

4.5.4 Further Discussion

The correlation between the meta-actions are essential to the final decision of the ADC, which is based on the consensus of reports from basic ADs being parameterized by a policy gradient learning algorithm. Although the detection manner is probabilistic, it still can be analyzed from a deterministic viewpoint. In our model, each AD only has two action modes, i.e., *Observe* and *Alert* (denoted by $0, 1$ respectively). Thus, total $2^4 = 16$ kinds of report sequences would be generated, among which, '$0000'$ (definitely *'Observe'* mode) most occurs, while '$1111'$ (definitely *'Alert'* mode) rare occurs. After being trained, each combination might find its matching action mode, if it cannot be adjusted to achieve '$0000'$. Obviously, the greater of the number of ADs (M) and action modes (N), the more complex of the report sequence (N^M), and thus the more difficult for the ADC to make correct decision. In our experiment, to insight into the probabilistic behavior of ADC and the state transitions, we trained ADC using pure normal data and "malicious" data respectively. A more ideal case is to train ADC using true data trace mixed by normal and anomalous activities (usually unobtainable), and find probabilistic action modes (that is, the various combination manners of the basic ADs).

Another point worth addressing is the trade-off between detection accuracy and false alerts, which is an inherited problem of the anomaly-based intrusion detection. We have given some assumptions to support our model formulation, since it is truly hard to achieve the goal without any insightful correlation analysis of the false alerts and post-processing techniques [65]. In our model, the combination of the concerns from both observations and

detection schemes might facilitate it to find some solutions. In addition, the adaptability of the model enables ADC to adapt the changing environments and suppress F.P. to an acceptable level. Once the F.P. exceeds a given threshold, the ADC can be re-trained to learn new probabilistic action modes and capture the drifts of system normality.

4.6 Concluding Remarks and Future Work

In this paper, we constructed a framework for the correlation of anomaly detectors. The objective is to broaden detection coverage, suppress false alerts, and capture "root-cause" attack instances in an automatic and adaptive manner. In general, our work are threefold:

Firstly, anomaly detectors' general behavior and their failure curses were analyzed to show the feasibility and advantage of their complementary operation.

Secondly, and the major contribution, we have developed an integrated detection model ADC, which was formulated as a partially observable markov decision process. A policy-gradient reinforcement learning algorithm was applied to tackle the delayed reward, partially observable, multi-agent learning problem. The key features of the model includes,

- the ADs learn cooperative behavior to complement each other, under the guide of a global reward signal, without inter-communication and explicit underlying states
- The adjustable reward signal enable the ADC to be capable of adapting to changing system situations with various security concerns. Both of remote attacks and local attacks are expected to be detected based on the dynamic combination of underlying independent ADs
- the distributed architecture allows the ADC to tolerate some failure of basic ADs, and to be extended easily to more complex environments

Finally, a host-based experimental scenario was developed and utilized to implement and evaluate the ADC's performance. Four well-known host-based ADs were employed as basic elements of ADC to construct a multi-layered boundary to defend against intrusive anomalies. The experimental results and comparative analysis demonstrated that our ADC outperforms individual ADs in terms of several admitted criteria.

In general, the framework presented a formal and effective way for the modeling, analysis, evaluation and implementation of the anomaly detector's complementary operation, which might facilitate the development of more efficient cooperative detection models. As the subsequent work, we intend to collect more real trace data (and some artificial anomalies) to enrich the experiments, in meantime, reducing the computational cost of the proposed method by abstracting effective observations for each AD is also of our concern. Other consensus strategies, or combination methods from the data fusion domain are also worth consideration. Furthermore, we will extend our model to the computer networks, to evaluate the performance of ADC on detecting distributed attacks. In this case, individual ADs locates in different hosts, especially those dominated hosts.

Chapter 5

Janus: Modeling and Analysis of Multi-Stage Coordinated Attacks in Computer Networks

5.1 Introduction

Although people never stop seeking the measures to secure their information systems by developing various defense strategies, attackers always can find some crafts to achieve their malicious intentions. Among the attacks, the most destructive and difficult one to detect are those that occur in stages over time and cooperated by a group of attackers, which is beyond much the power of individual attackers. To accomplish such a compromise, attackers must undergoes a process of reconnaissance, penetration, attack, and exploit. Meanwhile, the attackers need to cooperate each other for their common goals by resource sharing, task allocation, information communication and synchronization. Most of the current intrusion detection techniques aim to identify the individual stages of an attack with a certain accuracy and some false alerts, in most cases, some post-analysis of alerts correlation are usually needed. While the detection of a sophisticated multi-stage coordinated attacks remains difficult, which may involve a complex set of steps like using hosts as stepping stones, exploiting trust relationships and vulnerabilities of hosts, and data theft under the control of several organized hackers with particular capabilities.

Due to the special characteristic of multi-stage coordinated attacks, the simple combination and correlation of some individual attack countermeasures can hardly provide effective safeguard for the computer networks with distributed potential holes. An intuitive way is to deploy a rich set of intrusion detection sensors to collect the information related with the network links and hosts log data, the attack scenario hence can be constructed through the alert abstraction and correlation. However, the huge volume of alerts (both positive and negative ones), the different types of sensors (signature-based or anomaly-based), the sources of the alerts (hosts or communication links) and their semantics (preconditions or consequences) always make it too complex to handle. Strategic models can also be expressed as attack tress, which represent goal-oriented attack behaviors in hierarchical data structures. Such approaches provide a foundation for abstractly describing multistage behavior based on the conditional causal relationships between events or states, but those paradigms are generally static, which cannot provide a comprehensive model for the analysis of organized attacker's dynamic and concurrent

behaviors. An ideal approach not only considers the state transitions (for multi-stage) but also handle interactions between attacker's joint actions (for coordinated). Taking into the concerns that are not involved into most of existing schemes, we construct the multi-stage coordinated attacks as a multi-agent partially observable markov decision process (MPOMDP). In addition, due to the inherent limits of the available IDSs and the increasing application of encryption in communication, such as IPSec, SSL, intrusion detection and prevention have once again moved back to the host systems, therefore, the observations in our model are mainly from hosts rather than communication links. Specifically, the state transitions of the computer network and the attacker's object are represented in terms of those observable variables from hosts, such as audit log, file systems, and some particular application ports.

Rather than focused on the design of specific countermeasures, this chapter sheds light on the modeling and analysis of multi-stage coordinated attacks from a hight-level viewpoint. We envision a framework in which the security-related information (key observations) of the dominant hosts in the computer networks can be utilized to characterize their trust relationships and causal vulnerabilities. We also envision that attacker capabilities and scenarios can be constructed and represented in terms of observed preconditions, so that cooperative behavior can be isolated and multi-stage attacks can be suspended in case of further compromise. The hope is that those two concerns, like two sides of one coin, from the standpoints of defender and attacker respectively, can be combined together to achieve a complementary perspective for the detection of multi-stage coordinated attacks. The analytical model we are going to utilize, which has already been analyzed in the last chapter, is named Janus[1] (a Joint ANalytical model with two United Shields). Janus essentially formulate the behavior of both attacker and defender as a Partially Observable Markov Decision Process (POM-DP), and then develop two algorithms with apparently opposite concerns.

The rest of this chapter is organized as follows. Section 2 describes the attacker's basic behavior. Section 3 casts the attacker's behavior into a formal framework, and defines some notions and basic properties. Section 4 analyzes the model from the perspective of both defender and attacker. Section 5 discuss some similar techniques that can be utilized to enrich our modeling and analysis. Section 6 concludes the chapter and points out the future work.

5.2 Attacker's Basic Behavior

Different from those traditional attacks, which are launched by a single attacker to a single victim in a short period, multi-stage coordinated attacks are usually conducted by a group of organized attackers sharing the same objective and attacking tools, and usually accomplish in a long period through number of operations. Obviously, two key features are *multi-stage*, and *coordinated*, which are analyzed respectively in following.

Multi-stage: Or stealthy attack, which means that each attack consisted of a sequence of distinct steps, while each step represents an atomic attacker activity. For example, an attack might involve a network surveillance step, followed by an intrusion step using a known vulnerability, followed by a privilege escalation step to improve access

[1]The god of gates and doorways in Roman Mythology, depicted with two faces looking in opposite directions.

Figure 5.1: Multi-Stage Attacks

to the target, and finally achieve some goals such as information theft or denial of some system service.

A typical example of multi-stage attack is vulnerability finding worms, whose life-cycle generally includes four steps [49, 82]:

- Propagation: port scanning and probing to identify potential victim systems and specific vulnerabilities present on those systems,
- Activation: user-to-root access exploitation by executing a set of commands,
- Infection: downloading and installing an attack payload or planting malicious codes, and,
- Replication: eventually, conversion of the victim into a new attacker and looking for the next victim

Such attacks are sequential chains composed by several atomic operations, as shown in figure 5.1, while any failure of any step would cause the attack fail to succeed. It is worth noting that most of worm-like attacks are automated and non-coordinated, and they might spread to a large-scale area in a very short time. In figure 5.1, assume S_0 is the initial state of the system, while S_n is the system state when attacker has achieved his/her goal. The final state transition from S_{n-1} to S_n depends on the former transitions, without which, S_n cannot be achieved.

Coordinated: A group of attackers simultaneously compromise a target host or network by joint actions. As mentioned, the attacker does not necessarily mean human, it also might be artificial agents/tools, malicious scripts/codes acting on behalf of human. A very simple form of coordinated attack is distributed attacks like DDoS in which a multitude of compromised systems attack a single target, thereby causing denial of service for users of the targeted system. The key characteristic of such attacks is that the compromising point is multiple, the aggregated effects rather than the individual compromise lead to the successful attack. Figure 5.2 illustrates a simple coordinated attack scenario, in which two attacker e_1 and attacker e_2 cooperate each other to move the system from state s_0 (initial state) to s_n (compromised state). From state s_0 to state s_3, e_1 and e_2 act independently without necessary cooperation, while from state s_3 to s_n, attacker e_2's action u_n^2 has to be concurrently applied with e_1's actions u_{n-1}^1 and u_n^1, that is, the transition from state s_3 to s_n must depend on the joint action of e_1 and e_2. Although there is no compelling need to discriminate the multi-stage coordinated attacks and coordinated attacks, it is worth noting that the coordinated attacks usually takes several stages, but sometimes the correlation among the independent attackers are not so close, and the sequential property of the actions are not very obvious, while our concern here are those coordinated attacks with observable stages.

So far, many effective measures have been developed to cope with stealthy attacks and coordinated attacks respectively, however, the integration of those two forms of attack bring the defender more challenges:

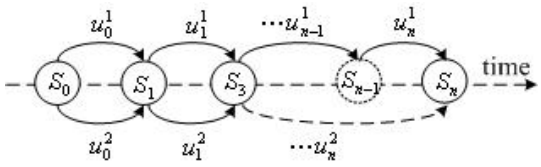

Figure 5.2: Coordinated Attacks

- **Escape from the detection**. The coordinated attack can avoid both misuse detection and anomaly detection, by breaking the attack pattern into many apparently innocent pieces. Although the individual activity appears normal, the aggregate effects might lead to successful compromise.

- **Countermine the detection**. Instead of managing to avoid being detected by defenders, the coordinated attacks might distract the intrusion detection systems by triggering IDS's alerts and consuming its resources. Although some minor or decoy attacks are detected, the major attack can still succeed, since the coordinated attack usually launch several simultaneous attacks.

- **Diversity**. Coordinated attacks can take various forms to achieve the same goal. At a particular stage, even one atomic attack has been detected, it can still take other actions to bypass the detection, or launch some similar attacks in parallel with the main attack.

We assume a network composing a set of potential victims $V = \{v_1, v_2..., v_m\}$ which can be individual hosts or the vulnerabilities in the hosts, and a group of attackers $E = \{e_1, e_2, ...e_n\}$ attempt to crack V during multiple stages T, where $T = \{t_1, t_2, ...t_l\}$. Another assumption is that the network undergoes the states $S = \{s_1, s_2, ...s_n\}$ under attacks during particular stages, while each attacker has his own action space u^i. The multi-stage coordinated attack scenario can be generalized with following properties:

- $Pr\{E_t(\hat{u})|s_i, v_i\}$ is assumed as the probability of a group of attackers E take joint actions $\hat{u} = u^1 \times u^2 \times \cdots \times u^n$ at stage t, with the knowledge of system state s_i and victims v_i. The elements have following specific meaning,

- It is the joint action $\hat{u}$ rather than the individual action u^i that moves the system state from s_i to s_j.

- The joint action $\hat{u}$ depends on the preconditions of V in state s_i, while the concurrent action list changes with the specific state.

- The attackers always intend to take the minimum set of actions (n of $\hat{u}$ is as small as possible) to achieve their goal (subgoal).

5.3 A Formal Framework

Based on the understanding of the multi-stage coordinated attacks in the last section, this section aims to further characterize the attacker's behavior by formulating it as a multi-agent partially Markov decision process.

5.3.1 Model Formulation

Formally, a POMDP model is structurally characterized by four key elements [1]: a finite state space S of n distinct states, or $S = \{1, 2, ..., n\}$ of the system; a control space U of m distinct actions (or responses), or $U = \{1, 2, ..., m\}$ that are available to the detection policy; an observation space Z of q distinct observations, or $Z = \{1, 2, ..., q\}$; and, a (possibly stochastic) reward $r(i) \in \mathbb{R}$ for each state $s_i \in S$, or in another sense, cost $c_{i,j}(u)$ for state transition from s_i to s_j with a particular control u.

To cast the multi-stage coordinated attacker's behavior in a POMDP framework, we have to specify the basic elements and notions as following,

1. $S = \{s_0, s_1, ...s_n\}$ is a set of system states, which cannot be observed directly by attackers. s_0 represents the initial states, and $s_i (i \neq 0)$ denotes the state when attacking goal or subgoal has been achieved.

2. $O = \{o_1, o_2, ...o_q\}$ is a set of observations to the attackers when system in state S.

3. $U = \{u^1, u^2, ...u^m\}$ is a set of actions taken by m attackers, where $u^i = \{u^i_1, u^i_2, ...u^i_l\}$ is the set of actions available to attacker $e_i (l$ is various with $u^i)$ and, $E_i(\hat{u}) \subset U$.

4. A reward signal r_t to show whether the goal (subgoals) of the attackers has been achieved at stage t, or $c_{i,j}(u)$ represents the cost moving state s_i to s_j with action $\hat{u}$.

Further, some additional constraints have to be set considering the multi-stage coordinated attacks' specific characteristic. Since the system states cannot be observed directly, they can only be represented as a set (or conjunction) of those observations o_i that are true in the state. For example, attacker e_1 successfully logged in v_1 as a superuser and e_3 accessed the log files in v_2, the observation o_i thus can be represented as $e_1(v_1) \wedge e_3(v_2)$, in most cases, the vulnerabilities combination $\hat{v}_i$ rather than the individual ones v_i contributes to the observation o_i. However, in essence, $o_i \subset \hat{v}_i$, since o_i is only those v_i that can be taken advantage by attackers; Attacker's action u^i are diverse due to their location, assigned jobs, assistance tools and capability, and we assume that each of them performs at most one action at a time (in a particular stage, some attackers could be idle); It also need to note that not only the single state s_i, but also the conjunction of states $\hat{s}_i$ represents the attacker's goal, since a complex goal might consist of several concurrent subgoals.

Based on the formal definitions and the specific concerns, the above parameters can be organized into a family of action-dependent matrices: $m \cdot n \times n$ state transition probability $Pr\{s_j | s_i, E_i(\hat{u})\}$ of matrices F, $m \cdot n \times q$ observation probability $Pr\{o_i | s_i, E_{i-1}(\hat{u})\}$ of matrices H, $m \cdot n \times n$ transition reward matrices G, and a $q \cdot m \times m$ action probability $Pr\{E_i(\hat{u}) | o_i, E_{i-1}(\hat{u})\}$ (which is essentially equals to $Pr\{E_t(\hat{u}) | s_i, v_i\}$). From the attacker's viewpoint, the reward signal should be maximized provided subgoals are achieved at each stage, i.e.,

$$min\{\lim_{T \to \infty} \mathbb{E}[\frac{1}{T} \sum_{i=1}^{T} r_i]\}, \tag{5.1}$$

in another sense, the attacking cost $c_{i,j}(E_i(\hat{u}))$ is always expected to be minimized from

state s_i to s_j, and the objective function is

$$min\{\lim_{T\to\infty}\frac{1}{T}\mathbb{E}[\sum_{k=1}^{T}c_{i,j}(E_i(\hat{u}))]\} \tag{5.2}$$

5.3.2 Basic Properties

Since the model is formulated from the perspective of attackers, their joint action is our main concern, since it directly related with the security of the system. In a more detail, each attacker's action can be specified as: which attacker e_i is performing the action u^i, what are the preconditions $\hat{v}_i$ of the action, who other attackers are involved at the current stage, what concurrent actions have to be taken, what is the system state (by observations) after the operation.

Property 3 (Preconditions) *The successful attack depends on the preconditions V that can be taken advantage by attackers E. At the initial state, V mainly denotes the set of exploitable vulnerabilities in the system, at other states, it also includes the conditions that have been generated by previous actions.*

Many techniques and tools, which are based on modeling network specifications, fault tress, graph models, and performance models, have been developed to analyze vulnerability by checking system logs and monitoring specific monitoring performance metrics. Attack trees [70] and graph-based network vulnerability analysis [67, 76] are two popular methodology. Attacker tress usually constructed in a given specific environment, and quantifies vulnerability by mapping known attack scenarios into tress; graph-based approach analyzes risks to specific network assets and examines the possible consequences of a successful attack. As the preconditions, the analysis system usually requires a database of common attacks (which might be broken into atomic steps), the information related with network configuration and topology, and even attackers' profile. The nodes of the graph identify an attack stage, and the graph thus enables us to identify the attack paths with the highest probability of success attack.

Property 4 (Concurrent Actions) *A group of attackers' joint action $\hat{u}_i$ keeps logically consistent in each underlying state s_i, which means u_i^j of e_j does not necessarily contradicts with u_i^k of e_k. The consistence of independent actions depends on that of preconditions and post-conditions.*

This property is essential to the attack's efficiency, suppose the goal ultimately will be achieved. For instance, if a group of sophisticated attackers cooperate well, and no member acts any negative effects on the current system state, their joint action at each stage would achieve their subgoals (at least generate the necessary preconditions for latter stages). For a group of amateur attacker, the same goal might cost more.

Property 5 (Consequence) *The consequence of the coordinated attackers' action $\hat{u}_t$ is a concurrent list of observations, which are jointly consistent, without any conflicts, in the state s_i. The consequential observation might be generated by an individual attacker, or a group of attackers.*

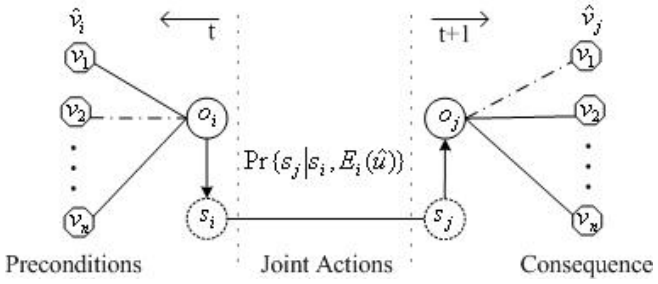

Figure 5.3: Stat Transition Under Attacks

Property 3 essentially seems to the property 2, both of which are focused on the aggregate affects of the concurrent actions. The consequence of joint action $\hat{u}_i$, in fact, is the precondition of joint action $\hat{u}_{i+1}$. In this sense, those three properties are consistent.

Taking into account those three properties, a Markov chain for state transitions are generated as:

$$s_i \in S \xrightarrow{\nu(s_i)} o_i \in O \xrightarrow{\mu(o_i)} E_i(\hat{u}) \in U \xrightarrow{p_{ij}(E_i(\hat{u}))} s_j \tag{5.3}$$

where $\nu(s_i) = Pr\{o_i|s_i, E_{i-1}(\hat{u})\}$, $\mu(o_i) = Pr\{E_i(\hat{u})|o_i, E_{i-1}(\hat{u})\}$, and $p_{ij}(u_i) = Pr\{s_j|s_i, E_i(\hat{u})\}$. A figure can be used to illustrate the relationships between the elements, which is shown in figure 3,

Note that the dash line connecting v_2 and o_i means v_2 does exist, but is not employed as o_i; the dash line connecting v_1 and o_j means v_1 does exist, and is not generated due to actions; The dash circle of s_i and s_j means the states are not observed directly. The consequence of $E_i(\hat{u})$ (only those v generated by previous actions), and the previously existing v combine as the preconditions $\hat{v}_j$ of the next action $E_{i+1}(\hat{u})$.

5.4 Janus: a Two-Sided Analytical Model

The last section presents a formal model incorporating both security analyst and the attackers' concerns, which therefore can serve both as a semantic model of computer networks (mainly from defender's standpoint) and as an analytical model for a group of attackers. This section firstly addresses the attacker's behavior based on the formulated model, and then proposes defender's countermeasures from the opposite viewpoint of the same model.

5.4.1 Attacker-Centric Analysis

In the above formal model, the system state S contains both normal system states S_N and the sates S_A when system are being compromised, which are not accessible for the observers while can be distinguishable through particular observations O by their knowledge and skills. However, for a group of attackers, there is no such a compelling need to differentiate all the possible state transitions, rather, they only need to discern those attack-relevant states by some distinctive features of the states. Therefore, the states in the attacker's analytical model derived from the general one only means those attack-relevant states, i.e., the system states undergoing attacks $S_A = \{s_0, s_1, \cdots s_a\}$, and $S_A \subseteq S$.

As the model shows, reward signal or attacking cost[2] can be used to evaluate attacks' efficiency. Assume a group of attackers $E = \{e_1, e_2...e_n\}$ successfully attack an object by T stages, and the final state is s_a, the most desirable reward signal should be $max\{\mathbb{E}[\frac{1}{T}\sum_{i=1}^{T} r_i]\}$. Suppose the initial system state is s_0, and the system states are transited in a sequential order, i.e., $s_0, s_1, \cdots, s_{a-1}, s_a$, the cost of transiting system states can be computed as,

$$\begin{aligned} C_{0,a}(E) &= c_{0,1}(E_0(\hat{u})) + c_{1,2}(E_1(\hat{u})) + \cdots + c_{a-1,a}(E_{a-1}(\hat{u})) \\ &= \sum_{i=1}^{a} c_{i-1,i}(E_{i-1}(\hat{u})) \end{aligned} \tag{5.4}$$

more generally, for $s_i, s_j \in S, i \neq j \in [0, a]$, $C_{0,a}(E) = \sum_{i,j} c_{i,j}(E_i(\hat{u}))$. Obviously, for coordinated attackers E, the smaller $C_{0,a}(E)$ the better, and a group of sophisticated attackers are always expected to achieve smaller $C_{0,a}(E)$ than that of amateur attackers. In most cases, the smaller attacking cost $C_{0,a}(E)$ means a smaller set of concurrent actions $|U|$, which are essential to the efficiency of multi-stage coordinated attacks for the sake of following observations,

- the smaller $|U|$ might require smaller $|V|$, which means that smaller concurrent action sets needs less necessarily preconditions for attack and,
- the smaller $|U|$ might require smaller $|E|$, which means that a smaller number of attackers are involved in the attack and,
- small $|U|$, $|V|$ and $|E|$ reduce the complexity of attacks, and thus suffer less probability of being detected.

No attacker prefer those cost-consuming attacks if they have options of cheaper ones for the same goal, in another word, they would not use a large set of concurrent actions if a smaller subset can achieve the desired effect. If so, all our analysis are in vain. The observation can be briefly described as follows,

Observation 1 *For a particular target, provided necessary preconditions V, there exist an optimal concurrent action set $\hat{U}$ by which a group of attackers E can achieve their goal with minimum attacking cost $C_{0,a}(E)$.*

More formally, we model the action-dependance state transition as a directed *state contact graph* $G =< S_A, W >$. The edges of the graph $W = V \times U$, where V is the set of preconditions for a particular joint action, and U is the collection of joint actions. Each directed edge represents a transition between two system states $s_i, s_j \in S_A$ at a certain stage. We represent each edge by a tuple $w =< s_i, s_j, E_i(\hat{u}), o_i >$ ($o_i \in \hat{v}_i$ is the exploitable preconditions) where s_i is the previous state and s_j is the target state with $E_i(\hat{u}) \times o_i$. Edge w is thus $s_i \xrightarrow{E_i(\hat{u}) \times o_i} s_j$. Based on the model and three properties derived from the general framework, observation 1 can be generalized as a following corollary,

Corollary 5 *In the directed state contact graph G, a group of attackers E always intend to search for the least-weight-path w_{min} from source node s_0 to the destination node s_a.*

[2]attacking cost means the cost that a group attackers have to pay during an attack stage, which can be measure by time, computational costs&resources, and some assistant tools, etc.

It is worth noting that the formulation of G is based on the assumption that attackers' actions have no loops, or excluding the wrong operations, recoveries, etc. In this sense, the attacking plans might be deterministic rather than probabilistic. Moreover, insights to individual attacker's behavior might benefits us better understanding of a group of attacker's joint actions,

Corollary 6 *For a particular attacker e_i, its action-related edges can only be regarded as a collection of discontinuous lines connecting two different system states.*

For a particular attacking scenario, e_i's operating traces can also be viewed as a sequence of state-related actions, namely, $\tau(e_i) = (u_0^i(s_0), u_1^i(s_1), \cdots, u_k^i(s_k))$, where $u_j^i(s_j)$ presents the action u_j is executed by e_i in state s_j, and it might be nothing if e_i is not involved in a particular attacking stage. This is based on an assumption that with the knowledge of G, attacker e_i always knows which action to be executed in every stage. The general attacking scheme is thus the combination of all the attackers' operating traces, i.e, $\tau(E) = (\hat{u_0}(s_0), \hat{u_1}(s_1), \cdots, \hat{u_a}(s_a))$. More generally, a group coordinated attackers' capacity can be measured by the tuple (U, N), where U is all the available actions (more specifically, the knowledge, skills, and assistant tools, etc.) available to the attackers, N is the total number of participators. Although the individual attackers in practice might have different capabilities and experiences, considering the common target and the internal sharing of the knowledge/skills/tools during the coordinated attacks, it is reasonable assume a group of attackers to be *homogeneous*. Based on the formulated model and the derived assumptions, also motivated by the attacker's intention, we attempt to develop an algorithm inspired by those ideas from ant colony optimization (ACO) algorithms family [20, 25], called *Attackers Nondeterministic Trail Search (ANTS)*, to search for such an attacking scheme as presented in corollary 1. In the algorithm, each attacker is viewed as a context-awareness ant searching for the food (subgoal), while the observations o_i at each stage construct covert channels for ants' action-dependent context.

Our formulated problem essentially belongs to the group of (constrained) shortest path problems that can be solved by ANTS algorithms, and can be specialized in following aspects,

- There is a set of constraints Ω for the concurrent actions of attackers.
- The available actions of attackers is a finite set $N = \{n_1, n_2, \cdots, n_l\}$.
- for every system state, a set of actions can be taken over N as $\delta =< n_r, n_s, \cdots, n_u, \cdots >$ ($< r, s, \cdots >$ to simply). Assume Δ is the set of all possible coordinated actions, we denote by $\widetilde{\Delta}$ the set of feasible sets with respect to the constraints Ω. The elements in $\widetilde{\Delta}$ define the feasible actions. $|\delta|$ is the size of a set δ, i.e., the number of actions in the set.
- There is a neighborhood structure among concurrent action sets defined as follows: δ_2 is an adjacent action set of δ_1 if (1) $\delta_1 \in \Delta$ and δ_2 $in\Delta$, (2) δ_2 can be reached from δ_1 by an additional logical action, i.e., $\delta_2 =< \delta_1, r >$($r \notin \delta_1$, while $r \in \delta_2$). The feasible adjunct action of δ_1 is the set containing all action sets $\delta_2 \in \widetilde{\Delta}$; if $\delta_2 \notin \widetilde{\Delta}$, δ_2 is viewed as the infeasible adjunct action set of δ_1.
- An attacking scheme AS is an element of $\widetilde{\Delta}$ verifying all the requirements.

- There is an attacking-specific cost $Cost$ to evaluate every AS.

Base on the problem-specific characterization, an ANTS algorithm can be developed as follows,

Attackers Nondeterministic Trail Search
void ANTS(U, N)
  Initialize (U, N, $minimum\_cost$, O_0); //O_0 is preconditions currently available
  while (termination_criteria_not_met)
    repeat in parallel for $k = 1$ **to** N
      initialize_ant(k);
      L = update_ant_memory(); //understanding of the current concurrent action
      while ($current\_system\_state \neq target\_system\_state$)
        compute_transition_probability (F, H);
        //F, H are matrix populated during training procedure
        take_next_action($\tilde{U}$); // $\tilde{U}$ is the feasible actions in U
        L = update_internal_state();
      end while
      if (state_transited)
        compute_reward_signal(r);//according to equation (1)
        $cost$ = compute_transition_cost(G); //according to equation (2)
        deposit_pheromone_update(r);
      end if
      minimum_cost = update_minimum_cost($cost$);
      get_ant_trail(k);
    end repeat in parallel
    get_concurrent_action_list();
    if (attacking_goal_not_achieved)
      return("Attack Failed!");
    end if
  end while

Note that criteria for the termination of ANTS can be set as required, it might be the goal has been achieved or the minimum attacking cost exceed a certain threshold, while the subgoals can be viewed as some specific preconditions for the next targets. F, H, G are three matrices of MPO-MDP characterizing the transition probability of system states, observations, and actions, which can be populated by a situation-specific training procedure. Considering the specific correlation among attackers (by constraints Ω), the searching process can be accelerated by their inter-communication (i.e., the procedure *update_internal_state()* and *take_next_action($\tilde{U}$)*), which also avoids the algorithm getting into local optima.

5.4.2 Defender-Centric Analysis

Taking the same analytical model as basis, defender can take advantage the inferred information for the prevention of multi-stage coordinated attacker's exploitation:

- The model facilitate us specific methods and techniques to defend against, mitigate or suspend the attacker's action in both temporal and spacial spans.

- If the attacker's graph G is well modelled by prior vulnerability correlation and analysis, the *pivot* of the attacking scheme thus can be figured out and removed easily.

- Corresponding cost-saving countermeasures can be taken based on the estimates of system state transition by seeking the tradeoff between *failure cost* caused by attacks and *maintenance cost* due to defence.

Vulnerability analysis has been taken as a kind of effective methodology to examine security-related properties for enhancing the dependability and security of computer systems, and many analytical models and tools [13, 17, 69] have been developed so far. However, we do not focus our attention on the development of our own analytical tools here, instead, we intend to figure out the key stages of attacking plans by assuming vulnerabilities have already been discovered and construing possible attack graphs. Intuitively, for the attack graph G, if a set of edges connecting essential exploits, namely, *backbone* of the attack graph are cut off, the attackers would never achieved their goal successfully. The observation can be briefly described as follows,

Observation 2 *An exploit might depend on a set of preconditions v and undergo multiple elementary actions u in order to take advantage of a single vulnerability (or subgoal), the vulnerabilities and corresponding examiners therefore allow us to derive a cause-consequence relationship for each basic actions.*

Similar with the construction of attacker graph G, the correlation among vulnerabilities can also be extracted as a directed *state contact graph* $G' =< S_N, W' >$, where S_N is the set of underlying system states representing the state transitions under known vulnerabilities, while W' is the collection of abstracted edges connecting $s \in S_N$. Note here, defenders not only concern those state transitions resulting in attacks, but also seemingly normal transitions, therefore, $S_A \subseteq S_N$ and $|S| \leq |S_A \cap S_N|$. Also, $W' = V \times U$, where V is the preconditions and, U is vulnerable operations, $w \in W'$ thus essentially denotes the multiple vulnerable operations u on several objects v that are involved in exploiting a vulnerability, in another word, changing system states. A corollary characterizing such property can be generalized as follows,

Corollary 7 *In the directed graph G', there exist at least one path, usually a collection of edges w', from the source node s_0 to the destination node s_n, without which, graph G' would turn to disconnected graph being cut into several parts.*

Obviously, if such edges w' do exist, and can be found out even approximately by heuristic methods in non-deterministic manners, defenders can easily understand adversary's possible attempts by figuring out the key observations. Since our basic assumption is that G' has been constructed by vulnerability analytical tools such as model-checking, a backward searching algorithm might be more feasible and efficient to explore the objective with desired properties. We intend to develop such an algorithm called *Attacker's Pivots Discovery by Backward Searching*, or APD-BS, which can also derived from the ACO algorithm family. The main idea is generalized as follows,

1. select the most significant vulnerabilities resulting in the system compromised state s_n, i.e., for those observations with a higher probability $Pr\{o_i|s_n, u\}$.

2. put the ants on those interested node, and trace back those *neighbors* meeting constraints in probabilistic manners,

3. rank the edges with the amount of pheromone left by those ants walking from source node to end node, based on which, the most significant *pivots* can be figured out for remove.

4. the above three steps are carried out iteratively until the termination criteria are met.

The main element of this metaheuristic algorithm is *ants*, which constructs candidate discoveries (a complete discovery is a solution) for the problem by individually and iteratively computation. The complete discovery contains a collection of correlated observations being generated from system state s_0 to s_N, while intermediate discoveries only contains parts of them. At each step, every ant k computes a set of feasible expansions to its current discovery and moves to one of these probabilistically according to a probability distribution p_{ab}^k (ant k from observation a to the next one b) by combining and specifying following two values,

- the attractiveness ε_{ab} of the move, as computed by some *a priori* desirability of that move;

- the trail level τ_{ab} of the move, indicating how proficient is has been in the past to make that particular move, which is essentially *a posteriori* indication of the desirability of that move.

Taking into account our specific concern, and based on the pre-knowledge that $H(a) = Pr\{a|s_i, u_{i-1}\}$ and $H(b) = Pr\{b|s_i, u_{i-1}\}$, we define the attractiveness of the ants' move as the correlation coefficient of those two probabilities,

$$\varepsilon_{ab} = \frac{Cov(H(a), H(b))}{\sqrt{D(H(a))}\sqrt{D(H(b))}} \tag{5.5}$$

where $Cov(H(a), (b))$ is the covariance of $H(a), H(b)$ and $\sqrt{D(H(a))}$ is the variance of $H(a)$, and the ant's pheromone trail update rule can be defined as,

$$\tau_{ab} = \rho\tau_{ab} + \tau_0(1 - \frac{c_i}{\overline{c}}) \tag{5.6}$$

where $\rho \in (0, 1]$ is a coefficient such that $1-\rho$ represents the decrease of trail between two generations of complete discoveries, and τ_0 is the initial trail level which is usually fixed to be an arbitrary small positive value. $\overline{c}$ is a moving average on the cost of the last l discoveries, and c_i is the cost of the ith ant's discovery. We can see that the cost $G(a)$ for system state transition essentially is taken as an underlying guidance for ant's movement, this is reasonable if we consider the fact, as we have discussed previously, attackers always manage to seek the *cheaper* actions for system state transition. Henceforth, the probability for ant k to make the move is given by

$$p_{a,b}^k = \begin{cases} \{\tau_{ab} \cdot \xi + \varepsilon_{ab} \cdot (1 - \xi)\} / \{\sum_{x \in tabu_k} \tau_{ax} \cdot \xi + \varepsilon_{ax} \cdot (1 - \xi)\}, \text{if } b \notin tabu_k \\ 0, \text{otherwise} \end{cases} \tag{5.7}$$

where the sum is over all the feasible moves and $\xi \in (0, 1]$ is a control parameter balancing the relative importance of the trail τ_{ab} and the attractiveness ε_{ab}, $tabu_k$ is an observation dependent set for kth ant's feasible moves. In this sense, p_{ab}^k actually is a tradeoff between the desirability of the current move and the past trails. With the definitions described above, the algorithm can be developed as follows in pseudocode,

Attacker's Pivots Discovery by Backward Searching
void APD-BS()
Initialize (p_o, *current_observation*, *minimum_cost*);
//*current_observation* is a set of observations $O_{initial}$ currently available
//p_o is *apriori* probability of observation selection, $minimum\_cost = \infty$
Put ants on those N selected nodes with $Pr\{O_{initial}|s_n, u\} > p_o$;
while (termination_criteria_not_met)
  repeat in parallel for $k = 1$ **to** N
    initialize_ant(k);
    L = update_ant_memory(); //get the information of the current context
    while ($current\_observation \neq \emptyset$)
      for($j = 1$ to $number\_of\_current\_observation$)
        a = get_current_observation[j];
        b = neighbor(a);
        //neighbor is defined as those observations resulting in the same system state
        ε_{ab} = attractiveness_compute(a, b);
      end for
      move_backward(); //make the move according to equation (7)
      compute_move_cost(); //update the *k-th* ant's discovery cost
      append_new_observation($tabu_k$);//update the *k-th* ant's new feasible move
    end while
    $cost$ = get_current_discovery_cost();
    if($cost < minimum\_cost$)
      $minimum\_cost = cost$; //update the *k-th* ant's moving costs
      update_discovery();//update the *k-th* ant's discovery
    end if
  end repeat in parallel
  for (each_move_of_ants())
    Update_trail_level();// update N ants' trail level by equation (6)
  end for
end while
Pivots_selection();
//select the key observations connecting by those edges
//with the most amount of pheromone

Note that since the development of our algorithm is based on the assumption that observatons/vulnerabilities have already been characterized by particular assistant tools, three matrices F, H, G can be populated with the preknowledge in advance.

In addition, the general MPO-MDP model formulated in the section 3 presents us a formal way to utilize the probabilistic state estimate as a sufficient statistic and optimally decompose the feedback controller into a recursive estimator and a response selector, which has been developed in [43] as a host-based autonomic defense system for the pro-

vision of the system survivability. Based on the functional decomposition, defender can select the proper countermeasures according to the feedback from system state estimator. The decomposition can be briefly described as follows,

Notations:

$\mathcal{I}_k$, all the information received by defender D prior to selecting the kth action u_k, including o_{k-1}, u_{k-1}, and $\mathcal{I}_{k-1}$.

$\mathcal{B}_k$, the system state estimate in stage k, which is a column vector with length n (state space), and the ith element $b_k(i) = Pr(s_i|\mathcal{I}_k)$ representing the relative confidence that state s_i is indeed the true system state in stage k. In essence, $\mathcal{B}_k \in \{\rho \in [0,1]^n | \sum_{l=1}^{n} [\rho]_l = 1\}$; when $[\rho]_i = 1$, defender surely knows system is in state s_i at current stage k; while when $\mathcal{B}_k$ is a uniform distribution, defender has no knowledge about system state estimate at all.

Intuitively, from the perspective of defender, an estimation policy φ that outputs each estimate $\mathcal{B}_k$ can be derived recursively as $\mathcal{B}_k = \varphi(o_k, u_{k-1}, \mathcal{B}_{k-1})$. Based on the probabilistic state estimate, defender can select the candidate countermeasures according to a response policy μ, namely, $u_k = \mu(\mathcal{B}_k)$. The objective of such a decomposition is to achieve the tradeoffs between the failure cost of a compromised information system and the maintenance cost of ongoing defensive countermeasures. As defined above, $c_{i,j}(u)$ is the cost for state transition from s_i to s_j with action u at stage k, while the defender's aim is to save cost during operation stages, i.e,

$$\lambda(\varphi, \mu|F, H, G) = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}[\sum_{k=1}^{T} c_{i,j}(\mu(\mathcal{B}_k))|F, H, G] \tag{5.8}$$

The equations essentially shows that the aim of defender's countermeasures is to minimize λ by simultaneously optimizing φ and μ. As for stationary POMDP models, probabilistic analysis can yield the optimal estimation φ^* via the closed-form recursion [43], with the fixed φ, the optimal response policy taken by defender can be expressed by

$$\mu^*(\mathcal{B}_k) = \arg\min_{u_k \in U} [\sum_{s_i \in S} c^*_{i,j}(u_k) b_k(s_i)] \tag{5.9}$$

where $c^*_{i,j}(u_k)b_k(s_i)$ is the expected cost obtained through an optimal selection of countermeasures at future decision stages, given current state s_i and action u_k. The countermeasures with minimum cost taken by defender thus is a set of values $c_{i,j}(u_k)$ so that the selected control $\mu(\mathcal{B})$ is obtained at each stage by applying the last equation. However, it it worth noting that the determination of the optimal countermeasures and the minimization of the given objective function is typically not possible, and thus approximation methods and heuristics must be applied to find near-optimal policies.

5.5 Implementation Issues and Concluding Remarks

The characterization and detection of multi-stage coordinated attacks pose as a difficult task in the intrusion detection domain due to its special temporal characteristics and wide spacial spans. From the temporal standpoint, a complete attack scheme can be accomplished through multiple steps whereas each step cannot be detected individually. In addition, such attacks usually span a wide area including multiple objects, which make it

harder to be discovered due to apparently normal pieces. To cope with this kind of attacks, in this chapter, we presented a two-sided model Janus for their representation and analysis. The model was cast in the Multi-agent Partially Observable Markov Decision Process (MPO-MDP) framework for both attacker's and defender's behavior characterization. Based on the model, three parts of work were conducted,

- the basic properties of multi-stage coordinated attacks were analyzed, and some key observations were figured out for attacks' more effective description. The analysis lays the theoretical foundation for the further modeling and analysis of users' (bother attacker and defender) behavior.

- attacker's behavioral characterization was specialized by taking account into their particular concerns. From attacker's point of view, an ANTS algorithm (Attacker Nondeterministic Trail Searching algorithm) derived from ACO algorithm family was developed to search for attack schemes with the minimum action cost;

- a defender-centric analysis was also carried out for the development of efficient countermeasures. Another searching algorithm which also derived from ACO algorithm family, called Attacker's Pivots Discovery via Backward Searching (APD-BS) was developed to capture the most significant observations/vulerabilities for a successful attack. By removing those key objects, attacks are expected to be thwarted or countermined. In addition, from the defender's standpoint, the MPO-MDP model was decomposed into two parts, i.e., system estimator and response controller, based on which, defender can select proper countermeasures concerning the tradeoffs between the system maintenance cost and the failure cost.

The future work is mainly focused on the implementation of our two designed algorithms, i.e, ANTS and ADP-BS, and meanwhile developing more efficient local search (LS) algorithms to accelerate those two algorithms' convergence speed. Those two algorithms are also expected to be applied to simulation-based testing environments with real trace data. For example, one specific application of ADP-BS is to identify the worms origin in the computer networks [84], that is, to determine both the host responsible for launching a propagating worm attack and the set of attack flows (can be viewed as edges w') that make up the initial stages of the attack tree via which the worm infect successive generations of victims. Since worm's generation and propagation can be essentially formulated as an attack graph we have analyzed, ADP-BS armed with efficient local search algorithms thus can be used to discover the most suspect origins.

Chapter 6

Conclusions

6.1 Summarization

This thesis contributes to developing effective and efficient models, methods and techniques for anomaly-based intrusion detection in hosts and networks with the concern of adaptability. Taking the observation-centric analysis as the theoretical foundation and starting point, we have developed three versions of SVM-based adaptive anomaly detectors that can be trained online for capturing the drifts of normal behavioral patterns; we also have developed an integrated anomaly detection model (core component is called autonomic detection coordinator) for correlating several individual observation-specific anomaly detectors; the modeling framework is also extended for the preliminary analysis of coordinated multi-stage attacks in computer networks. Generally, the work presented in this thesis, also our main contributions, can be summarized as follows:

1. The first part of work lays the foundation for the modeling and development of the specific anomaly detectors. Based on the similarity with the induction inference problem, we cast anomaly detection in a statistical framework, which gives a formal analysis on anomaly detector's anticipated behavior from a high level; The challenging issues and potential solutions are both presented, including the characterization of host-based and network-based normality, evaluation of anomaly detectors; Case studies on several typical anomaly detectors present us a formal way for the analysis of their operational capabilities, which might benefit their improvement and broader application.

2. The second part of work is about the development of three versions of SVM-based anomaly detectors. The kernel detection scheme are three modified Support Vector Machines, which reject the traditional assumption that training data for anomaly detectors are readily available with high quality in batch. Those SVM-based anomaly detectors aim to capture the normality drifts of normal behaviors by periodical training online, so that they can adapt to the new computing environments without triggering excessive false alerts. To validate those anomaly detector's performance, we implemented the experiments by reforming 1998 DARPA BSM data set collected at MIT's Lincoln Labs, and conducted the comparative studies with the original algorithms.

3. The third part of work contains two pieces of contributions. First, taking the formal observation-centric analysis of the individual anomaly detector's behavior as the

departure point, we developed an integrated detection model called ADC by correlating several individual parametric anomaly detectors. The complementary operations among the basic elements was formulated as a partially observable Markov decision process, and searching in an optimal cooperation manner, i.e., an optimal parameters setting for every anomaly detector. The anticipated behavior is to broaden overall detection coverage with fewer false alerts. A host-based experimental scenario was developed to implement ADC; Second, we extended the model as an approach for the modeling and analysis of multi-stage coordinated attacks in computer networks, the basic properties were given, together with the behavior analysis from the point of view of defenders and attackers. Moreover, two searching algorithms drawn from ACO algorithm family were developed, with the objective to search for attack schemes with the minimum action cost (from attacker's standpoint) and to capture the most significant observations/vulerabilities of a successful attack (from defender's standpoint), respectively.

6.2 Future Work

Our future work basically along the line currently undergoing, which is expected to be conducted in the following two mainstreams and thus make our already accomplished work more complete.

Firstly, to enrich the observation-centric analysis that we have carried out. We will extend our SVM-based anomaly detectors, which originally worked in the environments constructed by system calls (mainly frequency property), to some other operating environments such as shell command lines. Our main objective is to explore the relationship between the anomaly detector's performance and their computing environments.

Secondly, to extend the anomaly detection models that we have developed. We attempt to extend our integrated anomaly detection model to countermine those distributed attacks (e.g. worms, Dos, etc.). Although ADC's extended version, which is called *Janus*, has been applied to the modeling and analysis of multi-stage coordinated attacks in computer networks, its implementation (especially two algorithms ANTS and ADP-BS) worth further consideration with the specific computing environments. The model is also expected to dynamically preserve the desirable capability of information systems in the face of malicious intrusive attacks, i.e., survivability, and detect those attacks in their early stage so that cost-sensitive anti-attack strategies can be taken to mitigate threats in both scope and severity, mainly thwart attack's spread and prevent the further penetration.

Bibliography

[1] Douglas Aberdeen, "A Survey of Approximate Methods for Solving Partially Observable Markov Decision Processes", *National ICT Australia Report,* Canberra, Australia, December 8, 2003.

[2] M. Asaka, T. Onabuta, T. Inoue, S. Okazawa and S. Goto, "A New Intrusion Detection Method Based on Discriminat Analysis," *IEICE Trans. INF.and SYST.,* Vol.E84-D, No.5, pp.570-577, May 2001.

[3] 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.

[4] B. Balajinath, S.V. Raghavan, "Intrusion detection through learning behavior model," *Computer Communications*, pp.1202-1212, Elsevier Science, 2001.

[5] Peter L. Barlett and Jonathan Baxter, "Hebbian synaptic modifications in spiking neurons that learn", *Technical report, Computer Sciences Laboratory,* RSISE, ANU, November, 1999.

[6] Jonathan Baxter and Peter L. Barlett, "Stochastic Optimization of Controlled Partially Observable Markov Decision Processes", *Proceedings of the 39th IEEE Conference on Decision and Control(CDC00)*, pp.124-129, vol.1, Australia, Dec, 2000.

[7] Jonathan Baxter and Peter L. Barlett, "Direct Gradient-Based Reinforcement Learning: I.Gradiment Estimation Algorithms", *Technical report, Research School of Information Sciences and Engineering*, Australian National University, July 1999.

[8] J. Baxter, L. Weaver, and P.L. Bartlett, "Direct Gradient-Based Reinforcement Learning: II.Gradiment Descent Algorithms and Experiments", *Technical report, Research School of Information Sciences and Engineering,* Australian National University, September 1999.

[9] Bloedorn, E., Hill, B., Christiansen, et al., Data mining for improving intrusion detection, *http://www.mitre.org/support/papers/tech_papers99_00/.*

[10] T. Brants, F. Chen, A. Farahat, "A System for New Event Detection," *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'03)*, pp.330 - 337, July 28-August 1, 2003, Toronto, Canada.

[11] S. Braynov and M. Jadiwala, "Representation and Analaysis of Coordinated Attacks", *Proceedings of the 2003 ACM workshop on Formal methods in security engineering*, pp.43-51, Washington, D.C. 2003.

[12] Broderick, J., IBM outsourced solution, http://www.infoworld.com/cgi-bin/displayTC.pl/980504sb3-ibm.htm, 1998.

[13] H. K. Browne, W. A. Arbaugh, J. McHugh, W. L.Fithen, "A Trend Analysis of Exploitations," *Proceedings of 2001 IEEE Symposium on Security and Privacy (sP'2001)*, pp.214-229, May 14-16, 2001, Oakland, California, USA.

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

[15] M. Burgess, H. Haugerud, and S. Straumsnes, "Measuring System Normality," *ACM Transactions on Computer Systems,* Vol.20, No.2, Pages 125-160, May 2002.

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

[17] Shuo Chen, Zbigniew Kalbarczyk, Jun Xu, Ravishankar K.Iyer, "A Data-Driven Finite State Machine Model for Analyzing Security Vulnerabilities," *2003 International Conference on Dependable Systems and Networks(DSN'03)*, pp.605-614, San Francisco, California, USA.

[18] Cheung, S., Crawford, R., Dilger, M., Frank, etc., "The design of GrIDS: A Graph-based Intrusion Detection System," *Technical Report CSE-99-2,* U.C. Davis Computer Science Department, 1999.

[19] Sung-Bae Cho and Hyuk-Jang Park. "Efficient anomaly detection by modeling privilege flows using hidden Markov model", *Computer and Security,* Vol. 22, No.1, pp.45-55, 2003.

[20] O. Cordon, F. Herrera, T. Stutzle "A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends," *Mathware and Soft Computing* 9, 2002.

[21] G. Cormode, M. Datar, P. Lndyk, and S. Muthukrishnan, "Comparing Data Streams Using Hamming Norms(How to Zero In)" *IEEE Transaction on Knowledge and Data Engineering,* Vol.15, No.3, pp.529-540, May/June 2003.

[22] K. Daley, R. Larson, J. Dawkins, "A Structural Framework for Modeling Multi-Stage Network Attacks", *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)*, pp.5-10, Vancouver, Canada, 2002.

[23] 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, pp.201-21, Jan/April, 2002.

[24] X. Defago, P. Urban, N. Hayashibara, and T. Katayama, " Definition and Specification of Accrual Failure Detectors", *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2005)*, pp.206-216, Yokohama, June 28-July 1, 2005.

[25] M. Dorigo, V.Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man, Cyber. Part B,* vol.26, pp.29-41, 1996.

[26] 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 Academic Publishers, 2002.

[27] H. H. Feng, O. M. Kolesnikov, P. Fogla, W. Lee, and W. Gong, "Anomaly Detection Using Call Stack Information", *Proceedings of 2003 IEEE Symposium on Security and Privacy(SP'03)*, pp.62-75, Berkeley, CA, USA, May 2003.

[28] S. Forrest, S.A. Hofmeyr, and T.A. Longstaff, "A sense of self for UNIX processes," *Proceedings of 1996 IEEE Symposium on Security and Privacy(SP'1996)*, pp.120-128, Los Alamitos, CA, USA, May 1996.

[29] G. Giacinto, F. Roli, L. Didaci, "Fusion of multiple classifiers for intrsuion detection in computer networks," *Pattern Recognition Letters,* 24 (2003) 1795-1803.

[30] A.K.Ghosh, A.Schwartbard and A.M.Shatz. "em Learning Program Behavior Profiles for Intrusion Detection," *Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring,* pp.51-62, Santa Clara, CA, 1999.

[31] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, "Clustering Data Streams: Theory and Practice," *IEEE Transaction on Knowledge and Data Engineering,* Vol.15, No.3,pp. 515-528, May/June 2003.

[32] S. J. Han, and S. B. Cho, "Combining Multiple Host-Based Detectors Using Decision Tree," *Artificial Intelligence,* LNAI 2903, pp.208-220, 2003.

[33] P. Helman and G. Liepins, "Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse," *IEEE Transaction on Software Engineering,* Vol.19, No.9, pp.886-901, September 1993.

[34] Steven A. Hofmeyr, Stephanie Forrest, Anil Somayaji, "Intrusion Detection using Sequences of System Calls," *Journal of Computer Security,*Vol6, No.3, Page:151–180, 1998.

[35] S. H. Steiner, "Grouped Data Exponentially Weighted Moving Average Control Charts, " *Journal of the Royal Statistical Society: Series C (Applied Statistics),* Vol. 47, No. 2, pp.203-216, 1998.

[36] W. Hu, Y. Liao, V.R. Vemuri. "Robust Support Vector Machines for Anomaly Detection in Computer Security," *Proceedings of The 2003 International Conference on Machine Learning and Applications (ICMLA'03),* pp.168-174, Los Angeles, California, June 2003.

[37] W. Hu and Q. Song, "An Accelerated Training Algorithm for Robust Support Vector Machine", *IEEE Transaction on Circuits and Systems II: Fundamental Theory and Applications,* Vol. 51, No.5, 2004.

[38] M. Hutter, "Optimality of universal Bayesian sequence prediction for general loss and alphabet," *Journal of Machine Learning Research ,* 4 (2003): 971-1000.

[39] 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,* pp.69-75, 25, (2003).

[40] Joshua H., Dorene K.R., Larra T., Stephen T., "Validation of Sensor Alert Coorelators," *IEEE Security and Privacy,* pp.46-56, 2003.

[41] Julisch, K., "Clustering Intrusion Detection Alarms to Support Root Cause Anlaysis," *ACM Trans. on Information and System Security*, Vol.6, No.4, pp.443-471, Nov. 2003.

[42] Klinkenberg,R., and Joachims,R, "Detecting concept drift with support vector machines," *Proceedings the 17th International conference on Machine Learning (ICML-00)*, pp.487-494, Stanford, US: Morgan Kaufmann Publishers, San Francisco, US.

[43] O.Patrick Kreidl, Tiffany M. Frazier, "Feedback Control Applied to Survivability: A Host-Based Autonomic Defense System," *IEEE Trans. on Reliability,* Vol.53, No.1, pp. 148-166, March 2004.

[44] T. Lane, C. E.Brodley, "An Empirical Study of Two Approaches to Sequence Learning for Anomaly Detection," *Machine Learning,* vol.51, pp.73-107, 2003.

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

[46] W. Lee, S. J. Stolfo. "Data Mining Approaches for Intrusion Detection," *Proceedings of the 7th USENIX Security Symposium,* San Antonio, Texas, Jan.26-29, 1998.

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

[48] W. Lee and D.Xiang, "Information-theoretic meaasures for anomaly detection," In *IEEE Symposium on Security and Privacy (SP'2001)* pp. 130-143, 14-16 May 2001, Oakland, California.

[49] Elias Levy, "Worm Propagation and Generic Attacks", *IEEE Security and Privacy*, Vol.3, No.2, pp. 63-65, March/April 2005.

[50] Y. Liao and V. R. Vemuri, "Use of K-Nearest Neighbor Classifier for Intrusion Detection," *Computers & Security,* 21(5), pp.439-448, Oct., 2002.

[51] Terran Lane, Carla E. Brodley, "Temporal Sequence Learning and Data Reduction for Anomaly Detection," *ACM Trans. on Information and System Security*, Vol.2, No.3, pp. 295-331, August 1999.

[52] E. Lundin and E. Jonhsson, "Anomaly-based intrusion detection: privacy concern and other problems," *Computer Networks,* Vol.34, pp.623-640, 2000.

[53] T. F. Lunt, "IDES: An intelligent system for detecting intruders", In *Proceedings of the Symposium: Computer Security, Threat and Countermeasures,* pp.30-45, Rome, Italy, Nov., 1990.

[54] Sheng Ma, and Chuanyi Ji, "Modeling Heterogeneous Network Traffic in Wavelet Domain," *IEEE/ACM Transactions On Networking,* Vol.9, No.5, pp.634-649, October 2001.

[55] Marcus A. Maloof, Ryszard S. Michalski, "Incremental learning with partial instance memory," *Foundations of intelligent systems,* Lecture Notes in Artificial Intelligence, Vol. 2366, pp.16-27. Berlin: Springer-Verlag, 2004.

[56] Manganaris, S., Christensen, M., Zerkle, D., and Hermiz, K., "A data mining analysis of RTID alarms", *Computer Networks,* Vol.34, No.4, pp.262-294, 2000.

[57] Sunu Mathew, Chintan Shah, Shambhu Upadhyaya, "An alert Fusion Framework for Situation Awareness of Coordinated Multistage Attacks", *Proceedings of the Third IEEE International Workshop on Information Assurance (IWIA'05)*, pp.95-104, College Park, MD, USA, 2005.

[58] Roy A. Maxion, and Kymie M.C. Tan, "Anomaly Detection in Embedded Systems," *IEEE Transaction on Computers,* Vol.51, No.2, pp.108-120, Feb.,2002.

[59] Roy A. Maxion, and Kymie M.C. Tan, "Benchmarking anomaly-based detection systems", *Proceedings of International Conference on Dependable Systems and Networks (DSN2000)*, pp.623-630, 25-28 June 2000, New York, NY, USA.

[60] Roy A. Maxion, "Masquerade Detection Using Truncated Command Lines", *Proceedings of International Conference on Dependable Systems and Networks (DSN2002)*, pp.219-228, Washington, DC, 23-26 June 2002.

[61] Roy A. Maxion, "Masquerade Detection Using Enriched Command Lines", *Proceedings of International Conference on Dependable Systems and Networks (DSN2003)*, pp.5-14, San Francisco, CA, 22-25 June 2003.

[62] John 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, November 2000, Pages 262-294.

[63] MIT Lincoln Laboratory, *http://www.ll.mit.edu/IST/ideval/data/data_index.html*

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

[65] P. Ning, Y. Cui, D.S. Reeves, and X. Ding, "Techniques and Tools for Analyzing Intrusion Alerts," *ACM Transactions on Information and Systems Security,* Vol.7, No.2, May 2004, Pages 274-318.

[66] P. Ning, Y. Cui, D. S. Reeves, "Constructing Attacks Scenarios through Correlation of Intrusion Alters", *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, pp.245-254, November 18-22, 2002, Washington, DC, USA.

[67] C. Phillips, L. Swiler, "A graph-based system for network-vulnerability analysis", *Proceedings of the 1998 Worshp on New Security Paradigms,* pp.71-79, Charlottesville, VA, USA, 1998.

[68] Porras, P.A., Neumann, P.G., "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," *Proceedings of the 20th National Information Systems Security Conference*, pp.353-365, October 7-10, 1997, Baltimore, Maryland, USA.

[69] R. W. Ritchey, P. Ammann, "Using Model Checking to Analyze Network Vulnerabilities," *Proceedings of 2000 IEEE Symposium on Security and Privacy (SP'2000)*, pp.156-165, May 14-17, 2000, Oakland, California, USA.

[70] B. Schneier, "Attack Tress," *Dr. Dobb's J.*, vol. 12, Dec. 1999 (Computer Security), http://www.ddj.com/articles/1999/9912.

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

[72] 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, pp.440-448. Nov.2002.

[73] Ray J.Solomonoff, "Three Kinds of Probabilistic Induction: Universal Distributions and Convergence Theorems," *http://world.std.com/ rjs/pubs.html*, June 2003.

[74] Snapp, S. R., Smaha, S. E., Teal, D. M., Grance, T., "The DIDS (Distributed Intrusion Detection System) prototype", *the summer USENIX Conference,* pp.227-233, San Antonio, Texas, USENIX Association, 1992.

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

[76] L. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," *DARPA Information Survivability Conference and Exposition,* pp.146-161, Anaheim, California, 2001.

[77] Kymie M.C. Tan and Roy A.Maxion, ""Why 6" Defining the Operational Limites of stide, an Anomaly-Based Intrusion Detector," *Proceedings of the 2002 IEEE Symposium on Security and Privacy, SP'2002*, pp.188-201, Berkeley, California, USA, May 2002.

[78] Kymie M.C. Tan, Kevin S.Killourhy, and Roy A.Maxion, "Undermining an Anomaly-Based Intrusion Detection System Using Common Exploits," *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002),* LNCS, Vol.2516, pp.54-73, Springer-Verlag, 2002.

[79] Nigel Tao, Jonathan Baxter, Lex Weaver, "A Multi-Agent, Policy-Gradient approach to Network Routing", *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pp.553-560, MA, USA, July 2001.

[80] Valdes, A., Skinner, K., "Probabilistic Alert Correlation," *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001),* LNCS, Vol.2212, pp.54-68, Springer-Verlag, 2001.

[81] C. Warrender, S. Forrest, B. Pearlumtter, "Detecting Intrusions Using System Calls: Alternative Data Models," *Proceedings of 1999 IEEE Symposium on Security and Privacy (SP'1999)* , pp.133-145, Oakland, 1999.

[82] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms", *Proceedings of the 2003 ACM CCS workshop on Rapid Malcode (WORM'03)*, pp.11-18, Washington. DC, USA 2003.

[83] White, G., Fisch, E., Pocch, U, "Cooperating Security managers: A peer-based intrusion detection system," *IEEE Network*, Vol. 10, No. 1, pp.20-23, January/February 1996.

[84] Yinglian Xie, Vyas Sekar, David A. Maltz, Michael K. Re iter, Hui Zhang "Worm Origin Identification Using Random Moonwalks," *Proceedings of 2005 IEEE Symposium on Security and Privacy, (SP'2005)* pp.242-256, Oakland, CA, May 2005.

[85] Y.Yang, T.Pierce, and J.Carebonell. "A study on retrospective and on-line event detection," *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-98),* pp.28-36, Melbourne, Australia, Aug., 1998.

[86] 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, pp.266-274, July 2001.

[87] 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, pp.810 - 820, July 2002.

[88] Nong Ye, Timothy Ehiabor and Yebin Zhang, "First-order Versus High-Order Stochastic Models For Computer Intrusion Detection," *Quqlity and Reliability Engineering Internation,*2002(18): 243-250.

[89] Dit-Yan Yeung, Yuxin Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognition 36* (2003) 229-243.

[90] Zonghua Zhang, Hong Shen, "Application of Online-training SVMs for Real-time Intrusion Detection with Different Considerations", *Computer Communications*, Vol.28, No. 12, pp.1428-1442, Elsevier Science.

[91] Zonghua Zhang, Hong Shen, "Constructing Multi-Layer Boundary to Defend Against Intrusive Anomalies: An Autonomic Detection Coordinator" *Proceedings of the International Conference on Dependable Systems and Networks(DSN2005),* pp.118-127, Yokohama, Japan, June 28-July 1, 2005.

[92] Zonghua Zhang, Hong Shen, "A brief Observation-Centric Analysis on Anomaly-based Intrusion Detection", *Proc. of the First Information Security Practice and Experience Conference (ISPEC 2005)*, LNCS, pp.178-191, Singapore 11-14 April, 2005.

Publications

[1] Z. Zhang, H. Shen, "Application of Online-training SVMs for Real-time Intrusion Detection with Different Considerations," Journal of *Computer Communications*, Vol.28, No. 12, July 2005, pp.1428-1442, Elsevier Science.

[2] Z. Zhang, H. Shen, "A Brief Observation-Centric Analysis on Anomaly-based Intrusion Detection", to appear in the *International Journal of Network Security.*

[3] Z. Zhang, H. Shen, "Towards a Framework for the Correlation of Observation-Centric Anomaly Detectors: Modeling, Analysis, and Evaluation", *Submitted to IEEE Trans. on Reliability.*

[4] Z. Zhang, H. Shen,"A Brief Comparative Study on the Computer System Dependability and Security," *Proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT2005)*, pp.493-497, Dec., 2005, DaLian, China.

[5] Z. Zhang, H. Shen, "Constructing Multi-Layer Boundary to Defend Against Intrusive Anomalies: An Autonomic Detection Coordinator," *Proc. of the International Conference on Dependable Systems and Networks (DSN2005)*, pp.118-127, Jun., 2005, Yokohama, Japan.

[6] Z. Zhang, H. Shen, "A Brief Observation-Centric Analysis on Anomaly-based Intrusion Detection," *Proc. of the First Information Security Practice and Experience Conference (ISPEC2005)*, pp.178-191, Apr., 2005, Singapore.

[7] Z. Zhang, H. Shen, "Dynamic Combination of Multiple Host-based Anomaly Detectors with Broader Detection Coverage and Less False Alerts", *Proc. of the 4th International Conference on Networking (ICN'05)*, pp.989-996, Apr., 2005, Reunion Island, France.

[8] Z. Zhang, H. Shen,, "Capture the Drifting of Normal Behavior Traces for Adaptive Intrusion Detection Using Modified SVMs," *Proc. of the 3rd International Conference on Machine Learning and Cybernetics (ICMLC2004)*, pp.3045-3051, Aug., 2004, ShangHai, China.

[9] Z. Zhang, H. Shen, "Online Training of SVMs for Real-time Intrusion Detection," *Proc. of the 18th IEEE International Conference on Advanced Information Networking and Applications (AINA2004)*, Vol 1, pp.568-p573, Mar., 2004, Fukuoka, Japan.

[10] Z. Zhang, H. Shen, "Suppressing False Alarms of Intrusion Detection Using Improved Text Categorization Methods," *Proc. of the IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE2004)*, pp.163-p166, Mar., 2004, TaiWan, China.