2016

Intrusion detection using probabilistic graphical models

Liyuan Xiao
Iowa State University

Intrusion detection using probabilistic graphical models

by

Liyuan Xiao

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:

Carl K. Chang, Major Professor

Ying Cai

Hailiang Liu

Simanta Mitra

Johnny S. Wong

Iowa State University

Ames, Iowa

2016

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to my advisor and committee chair, Professor Carl K. Chang, and my committee members, Professor Ying Cai, Professor Hailiang Liu, Professor Johnny Wong, and Dr. Simanta Mitra, for their innovative guidance, tremendous patience, constructive suggestions and enormous support throughout this research and the writing of this thesis. Their insights and words of encouragement have often inspired me and renewed my hope for completing my graduate education.

I would also like to thank all my colleagues in the Software Engineering Lab for their suggestions and help on my research work. I am grateful for the support and assistance from IRB committee and more than 120 participants from Iowa State University, Nihon University in Japan, and Northeastern University in China on our experiment data collection. Without them, this thesis would not have been possible.

Finally, thanks to my families for their encouragement, support and love all the time.

CHAPTER 1. GENERAL INTRODUCTION

Modern computer systems are plagued by security vulnerabilities and flaws on many levels. Those vulnerabilities and flaws are discovered and exploited by attackers for their various intrusion purposes, such as eavesdropping, data modification, identity spoofing, password-based attack, and denial of service attack, etc. The security of our computer systems and data is always at risk because of the open society of the internet. Due to the rapid growth of the internet applications, intrusion detection and prevention have become increasingly important research topics, in order to protect networking systems, such as the Web servers, database servers, cloud servers and so on, from threats.

According to the definition in [10], a computer attack is the intelligence of evading or evading attempt of computer security policies, acceptable use policies, or standard security practices. Intrusion Detection can be seen just as a classification problem in which a given network traffic event is assigned as normal or malicious. The focus of this thesis is to build Intrusion Detection System, which is a mechanism designed to monitor and analyze network traffic information and users' activities in the target system in a given environment, and decide whether the activities are symptomatic of an attack or a legitimate use of the system. The process of intrusion detection includes the following phases: data collection, data pre-processing, intrusion recognition, and reporting and response. In order to fight against extraordinarily intelligent cyber-attacks in the era of rapidly growing information technology, effective and efficient intrusion detection systems are needed to promptly detect and prevent intrusion. Therefore, automatic intrusion detection is more demanding than ever. Various artificial intelligence and machine learning techniques, e.g., rule-based induction, classification, data clustering and data mining, have been widely used to obtain underlying models from training data. In this thesis, we aim to

build more accurate and efficient intrusion detection systems for classifying audited data into being intrusive or normal, using probabilistic graphical models.

According to different methodologies for training and predicting, there are mainly three categories [43] of intrusion detection systems: signature-based intrusion detection, anomaly-based intrusion detection, and hybrid intrusion detection. Signature-based intrusion detection identifies intrusions by matching audited data with pre-defined description of intrusion. This method is efficient in detecting well-known types of intrusions but usually fails to detect zero-day type intrusions. Anomaly-based intrusion detection methods establish models from normal behaviors and identify audited data by measuring the deviation between observed data and the built models. It is good at detecting new intrusions, but usually has high false positive rate. The hybrid intrusion detection is a combination of the former two approaches. The intrusion detection systems built in this thesis are hybrid intrusion detection systems that are obtained based on both normal and abnormal data records.

As summarized by Liao et al. [42], there are three main challenges in current intrusion detection researches:

1. Lower the false negative rate is one focus for signature-based intrusion detections, especially for some zero-day attacks. And lower the false positive rate is a focus for anomaly-based intrusion detection.

2. Collect training data set to build intrusion detection system. An intrusion may cause changes in some network traffic features. Those features could be collected from data packets in networks, command sequences from user input, low-level system information, e.g., system call sequences, log files, and CPU/memory usage, etc. A problem of great interest [8] in the training of intrusion detection systems is how to select key and effective features from a huge set of possible related features.

3. Enable intrusion detection systems to respond promptly and be real time.

In this thesis, we attempt to build more efficient Intrusion Detection System through three different approaches, from different perspectives and based on different situations. We cover those three approaches in Chapter 2, Chapter 3 and Chapter 4, respectively.

In Chapter 2, we propose Bayesian Model Averaging of Bayesian Network (BNMA) Classifiers for intrusion detection. In this work, we compare our BNMA classifier with Bayesian Network classifier and Naive Bayes classifier [26, 63], which were shown be good models for detecting intrusion with reasonable accuracy and efficiency in the literature. The main idea of BNMA [58] is that we choose k best Bayesian network models instead of using just one, and average those k selected models. When we have large amount of training data, many approaches are capable of producing models that fit the data well, and thus predicting future data accurately. However, large-size training dataset may be time consuming to collect in practice, we then very often have to rely on small-size dataset. In this case, there may exist more than one Bayesian networks that perform equally well in fitting the distribution of the training dataset, and the performance of using any single Bayesian Network classifier may not be satisfactory. The issue just mentioned was originally a very important motivation for us to think about using the BNMA on intruders and normal users classification. The BNMA first selects the k best Bayesian network models based on models' posterior probability out of all possible models given the training data. Then it predicts audited data by averaging all the k chosen models with weights proportional to their posterior probabilities. We conduct experiments over the KDD CUP 99 dataset [31], one of the most popular public datasets for evaluating intrusion detection systems. Our experiment results show that the BNMA classifier performs better than the Bayesian Network Classifier and Naive Bayesian Classifier in both accuracy and AUC (Area Under ROC). From the experiment results, we see that BNMA can be more efficient and reliable than its competitors, i.e., the Bayesian network classifier and Naive Bayesian Network classifier, for all different sizes of training dataset. The advantage of BNMA is more pronounced when the training dataset size is small. In fact, BNMA with smaller-size training dataset can work equally well, or even better than other models with larger-size training dataset. Therefore, the BNMA has the ability to accelerate the detection process as it could potentially save the time needed to collect more training data records.

In Chapter 3, we introduce the Situational Data Model as a method for collecting dataset to train intrusion detection models. Unlike previously discussed static features as in the KDD CUP 99 data [31], which were collected without time stamps, Situational Data are collected

in chronological sequence. Therefore, they can capture not only the dependency relationships among different features, but also relationships of values collected over time for the same features. The Situational Data Model is designed following the Situ framework [11], which was originally proposed for human-intention-driven service evolution in context-aware service environments. The Situ framework has a few advantages. For instance, Situ allows us to model and detect human intention by inferring human desires [64], and capture the corresponding context values through observation. Specifically, we collect our Situational Dataset following the structure of situation, consisting of desire, action and environmental context: The desire component describes a user's segmental thinking about the system at a specific time. This component makes the Situational data model to be more informative than other sequential data. On the other hand, the action component of the Situaiontal Dataset indicates a user's behaviors and operations on the system, and the environmental context indicates the system's status during the time when the actions are performed. In Situational Data Model, each data record is a sequence of situations collected at different time points. With Situational Dataset, we are able to train the relationships between actions, context and desires that happen at different time points, and build intrusion detection systems to classify the intention of a new sequence of situations, into being either intrusive or normal. In our research, we collected our Situational Dataset in Cooperative Research Envrionment(CoRE), which is a real web application. Through CoRE, the data set is generated from more than 120 invited participants. To compare the Situational Dataset and the traditional dataset consisting of only action sequences, we adopt the Hidden Markov Model (HMM) to build intrusion detection systems based on both datasets and then compare the two IDS. The experiment results show that the intrusion detection model trained by Situational Dataset outperforms that trained by action-only sequences.

In Chapter 4, we introduce the Situation Aware with Conditional Random Fields Intrusion Detection System (SA-CRF-IDS). The SA-CRF-IDS is trained by probabilistic graphical model Conditional Random Fields (CRF) [32] over the Situational Dataset proposed in Chapter 3. In SA-CRF-IDS, we hope to further improve the intrusion detection efficiency by both using a more informative training dataset, i.e., Situational Dataset, and adopting a more efficient classification model, i.e., CRF. In this chapter, we compare the Conditional Random Fields

and Hidden Markov Model for intrusion detection by both theoretic arguments and numerical experiments. For intrusion detection, CRFs can be more flexible and representative than other similar training methods such as the Hidden Markov Model, as often discussed in the literature. Our SA-CRF-IDS framework includes two layers: the desire layer and temporal intention layer. Both of the two layers are trained by CRF. The predicting processes of SA-CRF-IDS can be described as follows: Firstly, in the desire layer, SA-CRF-IDS labels a sequence of desires according to the sequence of actions and context. Secondly, in the temporal intention layer, SA-CRF-IDS labels a sequence of temporal intention value which quantifies the degree of attacking potential of the desires in numbers. Thirdly, the intention of situation sequences are classified to be either intrusive or normal based on the corresponding sequence of temporal intention values. A key idea of SA-CRF-IDS is that it predicts future audited data based on human's punctuated desires, instead of relying only on user's action and environmental context. In this chapter, our main interest is to compare the Conditional Random Field model and the Hidden Markov Model on each of the two datasets: the dataset with action-only sequences, and Situational Dataset proposed in Chapter 3. The results show that the CRF outperforms HMM with significantly better detection accuracy, and better ROC curve when we run the experiment on the non-Situational dataset. On the other hand, the two training methods have very similar performance when the Situational Dataset is adopted.

We conclude our work from Chapter 2 to Chapter 4 with a discussion in Chapter 5, including the accomplished work and potential future work.

CHAPTER 2. BAYESIAN MODEL AVERAGING OF BAYESIAN NETWORK CLASSIFIERS FOR INTRUSION DETECTION

Abstract

In order to defend against extraordinary intelligent attacks in the era of rapidly growing information and technology nowadays, effective and efficient intrusion detection models are needed to detect and prevent intrusion promptly. Bayesian network (BN) classifiers with powerful reasoning capabilities have been increasingly utilized to detect intrusion attacks with reasonable accuracy and efficiency. However, existing approaches using BN classifiers for intrusion detection face two problems. First, the structures of Bayesian network classifiers are either manually built with the help of domain knowledge or trained from data using heuristic methods that usually select suboptimal models. Second, the classifiers are trained using very large datasets which may be time consuming to obtain in practice. When the size of training dataset is small, the performance of a single Bayesian network classifier is significantly reduced due to its inability to represent the whole probability distribution. To alleviate these problems, we build a Bayesian classifier by Bayesian Model Averaging (BMA) over the k-best Bayesian network classifiers, called Bayesian Network Model Averaging (BNMA) classifier. We train and evaluate the classifier on the NSL-KDD dataset, which is less redundant, thus more judicial than the commonly used KDD Cup 99 dataset. We show that the BNMA classifier performs significantly better in terms of detection accuracy and Area Under ROC (AUC) than the Naive Bayes classifier and the Bayesian network classifier built with heuristic method. We also show that the BNMA classifier trained using a small dataset even outperforms two other classifiers trained using a very large dataset, thus BNMA is particularly effective when large training datasets are unavailable. This also implies that the BNMA is beneficial in accelerating the

detection process due to its less dependance on the potentially prolonged process of collecting large training datasets.

Key Words: Intrusion detection system, Bayesian network, Bayesian Model Averaging, Detection accuracy.

2.1 Introduction

An intrusion detection system is a mechanism used to monitor system and network situations, collect useful data such as suspicious activities and environmental context information, and analyze such data to predict and detect malicious intentions. As the amount of network throughput increases and security threat intensifies, intrusion detection systems have drawn much attention in recent years. In general, intrusion detection approaches are classified as either Signature-based Intrusion Detection (SD) or Anomaly-based Intrusion Detection (AD). SD is the process to compare signature patterns of known attacks or threats against captured events for recognizing possible intrusions. AD is the process to find deviation from a known behavior, and construct profiles representing the normal or expected behaviors derived from monitoring regular activities, network connections, hosts or users over a period of time [42].

Existing intrusion detection systems (IDS) are divided into five different types according to a survey paper by They are: Network-based IDS, which monitors network traffic data; Host-based IDS, which monitors and analyzes host activities like system calls, application logs and so on; Stack-based IDS, which examines the packets as they go through the TCP/IP stack; Protocol-Based IDS, which monitors the protocol in use of the computing system; and Graph-Based IDS, which is concerned with detecting intrusions that involve connections between many hosts or nodes.

Regardless of the types of systems, the challenge is to build effective predictive models with low error rates by utilizing and integrating various data resources. To achieve this goal, various approaches have been proposed. These include statistic-based, pattern-based, rule-based, state-based and heuristic-based approaches. As a statistic-based approach, Bayesian network (BN)

has been widely used in intrusion detection field due to its robustness in modeling the joint distribution of random variables and reasoning under uncertainty.

Sebyala *et al.* [49], Amor *et al.* [4], Vijayasarathy *et al.* [60], and Altwaijry & Algarny [3] built Naive Bayesian classifier, a type of simplified BN, to identify possible intrusions. Amor *et al.* [4] also compared Naive Bayes with the technique of decision tree and showed that Naive Bayes can reach a result almost as good as decision tree but with much faster computation. However, there is a very strong assumption in Naive Bayes that the feature nodes in the Naive Bayes model are independent from each other given the root node, which is not always the case in practice.

Kruegel *et al.* [38] proposed an event classification that makes full use of Bayesian networks and allows the modeling of inter-feature-node dependencies. They showed that these extensions improve the quality of the decision process and significantly reduce the number of false alarms. Lu *et al.* [26] gave a two-stratum Bayesian networks-based anomaly detection and decision model for IDS. Laskey *et al.* [41] created an innovative human behavior model to model user queries and detect situations and insider threats to information systems using multi-entity Bayesian networks. In [5], An *et al.* used dynamic Bayesian networks to model temporal environments and detect any privacy intrusions. In these applications, the network model structures were manually constructed with the help of domain knowledge without utilizing the training data that better reflects the real situation. To address this problem, Wee *et al.* [63] performed model selection by learning the BN structure from data. However, the model selection in [63] was conducted using heuristic methods, which usually select suboptimal models.

Further, most of the classifiers were trained and evaluated by utilizing the KDD Cup 99 dataset, which consists of about 0.5 million records [31]. A classifier trained with such a huge training dataset is usually capable of representing the probability distribution, thus achieves very good performance. However, obtaining such large-scale datasets can be challenging in practice, as it may take an unreasonably long time to collect the data resources. When the training dataset is small relative to the number of features considered, it is usually hard to select a single classifier model that properly represents the probability distribution of the model space. In such a situation, using a single model usually leads to poor classification on future data.

To address the problems raised above, we built a Bayesian classifier for intrusion detection by Bayesian Model Averaging (BMA) over the k-best Bayesian network classifiers. Instead of selecting a single Bayesian network classifier, we perform model selection to find the top k Bayesian network classifiers according to a certain scoring metric. When future data points are classified, the decision is made by averaging over the prediction results of the k-best Bayesian network classifiers. The motivation of doing this is that multiple Bayesian networks are better than one Bayesian network in representing the probability distribution of the model space, thus they offer better predictive power than one network, particularly in the domain where only small training datasets are available. To the best of our knowledge, this is the first attempt to employ BMA method in intrusion detection research.

The rest of the paper is organized as follows: In Section 2.2, we briefly introduce the concept of Bayesian network classifiers and BMA. We then introduce our BNMA classifier, which makes predictions by averaging over k-best Bayesian network classifiers. In Section 2.3, we describe the NSL-KDD dataset from which our training and testing datasets are drawn. In Section 2.4, we outline the construction and evaluation of our Bayesian Network Model Averaging (BNMA) classifier. In Section 2.5, we illustrate the details about the design of experiments and experimental results. In Section 2.6, we conclude with some discussions and ideas for future work.

2.2 Bayesian Networks and Bayesian Model Averaging

2.2.1 Bayesian Network Classifier

A Bayesian network G is a probabilistic graphical model that encodes a joint probability distribution over a set of variables $\mathbf{X} = \{X_1, X_2, ..., X_n\}$ based on conditional independencies [24]. It is a directed acyclic graph (DAG) where each node represents a random variable and an edge denotes a direct probabilistic dependency between the two connected nodes. For each node, there is a conditional probability distribution (CPD) containing the probabilities of the node taking different values given its parents' value. Formally, the DAG structure asserts that each node is conditionally independent of all non-descendants given its parent nodes. By these

assertions, the BN compactly represents the joint probability distribution as

$$p(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} p(X_i|\mathbf{Pa}_G(X_i)), \tag{2.1}$$

where $\mathbf{Pa}_G(X_i)$ denotes the set of parent nodes of X_i in G, and $p(X_i|\mathbf{Pa}_G(X_i))$ specifies the conditional probability distribution (CPD) of X_i given $\mathbf{Pa}_G(X_i)$.

Figure 2.1 gives a simple example of Bayesian network that portrays the probabilistic relationships among binary variables *Polution* (P), *Smoker* (S), *Cancer* (C), *XRay* (X) and *Dyspnoea* (D). The table associated with each variable is called Conditional Probability Table (CPT), encoding the conditional probability distribution of the variable given its parents. The joint probability distribution of the five variables can be written as

$$p(P, S, C, X, D) = p(P)P(S)p(C|P, S)p(X|C)p(D|C). \tag{2.2}$$

As the conditional probability distribution can be calculated from the joint probability, a Bayesian network consisting of a class variable and feature variables is readily applicable to the classification task. Take the lung cancer network as an example, if we choose *Cancer* (C) as the class variable (value unobserved), we can compute the probability of $C = T$ given any observed value set (p, s, x, d) as

$$p(C = T|p, s, x, d) = \frac{p(C = T, p, s, x, d)}{p(C = T, p, s, x, d) + p(C = F, p, s, x, d)},$$

where $p(C = T, p, s, x, d)$ and $p(C = F, p, s, x, d)$ can be computed efficiently using Eq.(2.2). Similarly, we can compute $p(C = F|p, s, x, d)$. Then we decide the value of C by comparing $p(C = F|p, s, x, d)$ and $p(C = T|p, s, x, d)$. Note that this is a binary classification, easily generalized to multi-class classification by comparing the conditional probabilities of all values of the class variable.

The Bayesian network structure and its associated CPDs can be specified with the help of domain knowledge, e.g., the lung cancer network. However, in most cases, the network structures and CPDs are unknown due to the lack of domain knowledge. In these cases, a Bayesian network classifier can be learned from training data. The learning process contains structural learning and conditional probability distribution estimation. In structural learning,

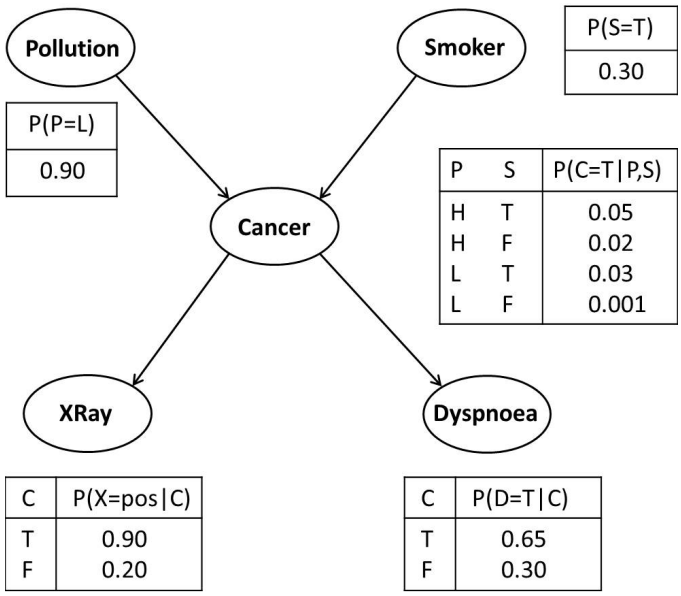


Figure 2.1: A simple example of Bayesian network: Lung cancer network.

a scoring metric is employed to evaluate the fitness of a structure in relation to the training data. Then, a search method is applied to find a good model [63] among possible structures. Since the number of possible structures is super-exponential with respect to the number of variables, finding the optimal structure is NP-hard [15]. Thus, some heuristic or approximate methods, such as greedy search, are used. However, the model structures selected in this way are often suboptimal. After the structure is constructed, the CPDs can be efficiently estimated using well-developed statistical methods such as Maximum Likelihood Estimation (MLE) or Bayesian Estimation [33].

2.2.2 Bayesian Model Averaging of Bayesian Network Classifiers

Regardless of types of the search methods used, these search methods suffer from the lack of distinguishability of scoring metrics when the training data is sparse, i.e., the size of the dataset is small relative to the number of variables. In this case, there can be many distinct Bayesian networks fitting the training data equally well. Thus, using a single Bayesian network potentially leads to poor predictions on future data.

A promising solution to alleviate this problem is to employ BMA, which provides a principled approach to the model-uncertainty problem by integrating all possible models weighted by their respective posterior probabilities. Formally, given a training dataset D and a future data point x (a realization of the variable set $\mathbf{X}$), we compute the posterior probability of observing x as

$$p(x|D) = \sum_{G} p(x|G, D)p(G|D), \tag{2.3}$$

where $p(G|D)$ specifies the posterior probability of a Bayesian network G given the training data D. $p(G|D)$ can be computed from commonly used scores such as BDe score [25]. Then, $p(x|G, D)$ can be computed by Eq. (2.2), as the network structure G is fixed in the conditional setup.

Since computing Eq.(2.3) requires enumerating all possible networks, which is super-exponential with respect to the number of variables, it is not of practical use. One solution is to approximate this exhaustive enumeration by using a selected set of model structures in $\mathcal{G}$, i.e.,

$$p(x|D) \approx \frac{\sum_{G\in\mathcal{G}} p(x|G, D)p(G|D)}{\sum_{G\in\mathcal{G}} p(G|D)}.$$

Dash and Cooper [16] described an efficient solution to BMA for prediction over the set of Bayesian network structures consistent with a partial ordering and with bounded in-degree. However, this approach is of limited applicability as it performs model averaging over only a restricted class of BNs consistent with a particular partial ordering. Thus, only a small portion of probability density can be accounted for. Tian *et al.* [58] proposed to find the k-best Bayesian network structures and use them to approximately compute $p(h|D)$, i.e., the posterior probability of any hyperthesis h. They implemented this idea to address the problem of structure discovery in BNs, i.e., computing $p(f|D)$, the posterior probability of the presence of any structural feature f. (e.g., an edge, in BN structures). They showed that the approximation achieved reasonable accuracy and outperformed the classical sampling methods such as MCMC [18] for structure discovery in BNs. In this study, we employ this idea to address the problem of model averaging for prediction (classification). We select the k-best Bayesian networks $G^1, ..., G^k$, and use them to approximately compute $p(x|D)$ as shown in Eq.(2.4),

$$p(x|D) \approx \frac{\sum_{i=1}^{k} p(x|G^i, D)p(G^i|D)}{\sum_{i=1}^{k} p(G^i|D)}. \tag{2.4}$$

Once $p(x|D)$ is computed, we could build a classifier to predict the value of any class variable as shown in Section 2.2.1. When $k = 1$, we select the best Bayesian network and use this single network to build a classifier. Thus, it is a special case of BMA.

2.2.3 Finding the k-best Bayesian Network Structures

In previous sections, we mentioned that the optimal model selection is an NP-hard problem, as the number of possible model structures is super-exponential with respect to the number of variables. Thus, in existing applications of Bayesian network classifiers, heuristic or approximate methods are employed to find the models which are usually suboptimal. Silander *et al.* proposed a dynamic programming (DP) algorithm which is capable of finding the globally optimal Bayesian network in $O(n2^n)$ time [53]. Tian *et al.* [58] extended the DP algorithm to find the top k Bayesian network structures. They demonstrated the applicability of the algorithm on networks with up to 20 variables. One nice feature of their method is that the estimation accuracy can be improved monotonically by spending more time to compute for larger k.

In this study, we employ this algorithm to select the k-best Bayesian network structures. We then estimate the CPDs using Bayesian Estimation for each of the k-best network structures that result in k discrete Bayesian network classifiers. Afterwards, we build our BNMA classifier by averaging the prediction results over the k-best Bayesian network classifiers.

In detail, we use 12 observed feature variables from the KDD Cup 99 dataset and 1 unobserved variable representing intrusion or not intrusion. We get the k-best Bayesian Network structures by running the software tool called KBest [58] which is used to compute the posterior probabilities of features by Bayesian model averaging over the k-best Bayesian networks. Inside of this tool package, it computes the local scores for all the families of each variable. With a selected k value, the software tool of KBest takes a file of data records as input and outputs the k best network structures and lists the estimated posterior probabilisties for each of the k best networks based on the input data.

2.3 Description of NSL-KDD Dataset

In previous IDS research, KDD Cup 99 dataset has been widely used to help build and evaluate these systems [4] [63] [3] [17]. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. However, KDD Cup 99 has two major issues that highly affect the assessment of the performance of evaluated systems [57]. The first deficiency in KDD Cup 99 dataset is the huge number of redundant records in the training dataset. This deficiency will cause learning algorithms to be biased towards more frequent records. The second deficiency is that the existence of repeated records in the test set will cause the evaluation results to be biased towards favoring the methods with better detection rates on frequent records.

In our experiment, we use the NSL-KDD [57] dataset, a new version of KDD Cup 99 dataset consisting of selected records of the complete KDD Cup 99 dataset with redundant and repeated records removed. As can be seen from the literature [57], the original KDD Cup 99 dataset is skewed and unproportionately distributed, training and testing directly on the KDD Cup 99 dataset can result in relatively high accuracy rate for different methods, making it difficult to effectively compare different classifiers. Using this NSL-KDD dataset for evaluation is more objective and judicial as it does not suffer from either of the two problems mentioned above. The NSL-KDD dataset contains a training set with 125,973 records and a testing set with 22,544 records. Each of the datasets contain 41 attributes describing different features of the connection and a class label assigned to each either as attack or as normal.

2.4 Construction and Evaluation of BNMA Classifier

In this section, we introduce how we construct and evaluate the BNMA classifier. Figure 2.2 illustrates the process from data processing to classifier training and evaluation. The whole process is elaborated in the following steps:

1. Download NSL-KDD dataset and select a subset out of a total 41 features as the variables for classifier building.

2. Randomly sample partial datasets of varying sizes from the overall NSL-KDD training dataset as the training sets. The whole NSL-KDD testing dataset is then used as the testing set.

3. Perform data discretization on the continuous features in training and testing datasets using the information-preserving discretization method.

4. Find the k-best Bayesian network structures using the training dataset, and estimate the CPDs for each networks using Bayesian Estimation. This results in k independent Bayesian network classifiers.

5. Combine the k Bayesian network classifiers into a Bayesian classifier using BMA.

6. Apply the Bayesian classifier to the testing dataset, calculate the accuracy and Area Under ROC (AUC).

7. Conduct four groups of experiments by repeating steps 2-6 using different training sets. The results for each classifier and each configuration of different size are then averaged over those four groups of experiments.

In the upcoming subsections, we give details on the process of feature selection, data discretization, and classifier training and evaluation.

2.4.1 Feature Selection

Feature selection is an indispensable pre-processing step when training a huge dataset with many features. Extraneous features not only add burden to the computation but also confound the detection process. The NSL-KDD dataset contains 41 features, some of which may be redundant and contribute less than the others to the detection process. Feature selection and feature deduction have been a very popular topic in intrusion detection field for identifying important input features to build computationally efficient and effective IDS. Singh & Silakari [54] proposed an ensemble approach for feature selection of the Cyber Attack dataset. Chebrolu *et al.* [12], Kayacik *et al.* [30] and Olusola *et al.* [46] specifically analyzed the feature relevance on the KDD Cup 99 dataset. In [12], Markov blanket model was used to select the feature

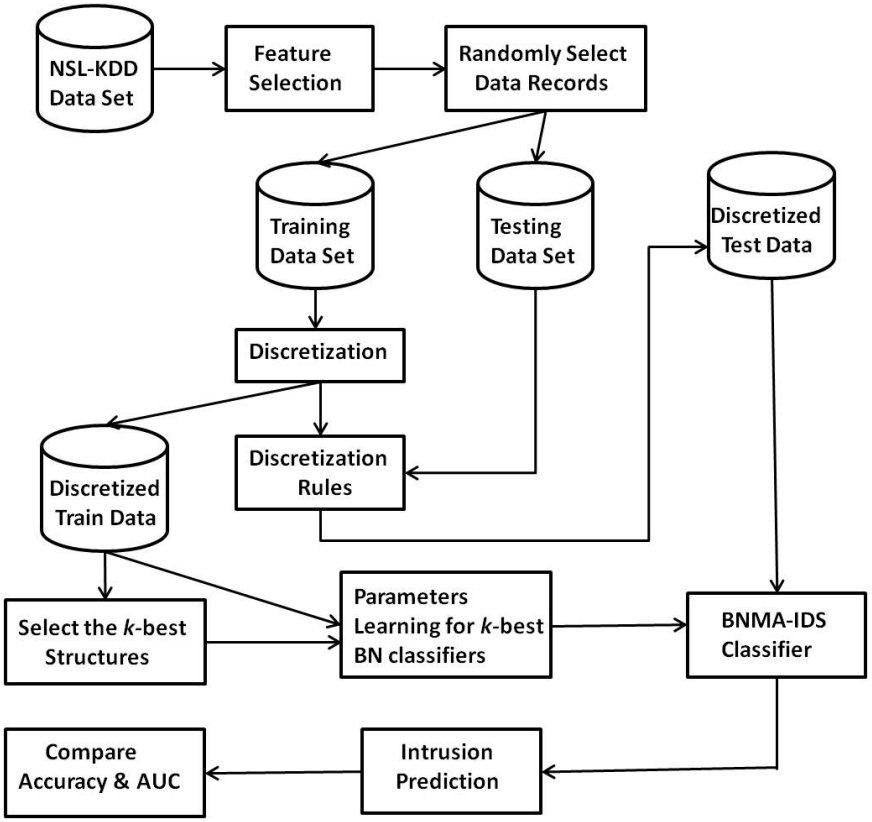


Figure 2.2: Flowchart illusrating the training and evaluation of BNMA classifier

set and it was shown that a selected set of 12 features can achieve better predictive accuracy than when the whole set of 41 features is used. Our main focus in this paper is to compare the methods based on the same datasets with the same feature set, rather than to study the KDD data. And we are also particularly interested in studying their performance using a carefully selected representative subset rather than the full features. Therefore, in our experiments we used the 12 features suggested in [12]. Those features are described in the following Table 2.1.

2.4.2 Data Discretization

As shown in Table 2.1, some of the selected features take continuous values. However, current implementation of Bayesian network classifiers can only handle discrete values. Thus, continuous features need to be discretized before being used to build a classifier. On the other hand, discretization can often make continuous features easier to understand and interpret, and produce faster learning models. Many learning models have been shown to perform better by

Table 2.1: List of Selected Features

FEATURE NAME	**DESCRIPTION**	**TYPE**
service	network service on the destination, e.g., http, telnet, etc.	Discrete
src_bytes	number of data bytes from source to destination	Continuous
dst_bytes	number of data bytes from destination to source	Continuous
logged_in	1 if successfully logged in; 0 otherwise	Discrete
count	number of connections to the same host as the current connection in the past two seconds	Continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	Continuous
serror_rate	% of connections with errors (refer to the same-host connection)	Continuous
srv_serror_rate	% of connections with errors (refer to the same-service connection)	Continuous
srv_diff_host_rate	% of connections to different hosts	Continuous
dst_host_count	sum of connections to the same destination IP address	Continuous
dst_host_srv_count	sum of connections to the same destination port number	Continuous
dst_host_diff_srv_rate	the percentage of connections to different services, among the connections aggregated in dst_host_count (32)	Continuous

discretizing continuous features [36]. Based on our knowledge, there are two types of commonly used discretization methods in many of the IDS [37] [46] [63], that is, the unsupervised discretization algorithms, e.g., equal intervals, equal frequencies, and the supervised discretization algorithms, e.g., maximum entropy discretization, χ^2 discretization, CAIM, etc. In our experiment, we adopted a discretization algorithm named CACC [59], which was a a static, global, incremental, supervised and top-down discretization algorithm. This information-theoretic algorithm extended the idea of contingency coefficient, combined with the greedy method, and was empirically shown to be promising in terms of accuracy, execution time, etc. Data discretized using such discretization scheme have much less information loss, thus better represent the distribution of original data compared to the ones discretized using other unsupervised discretization methods.

2.4.3 Classifier Training and Evaluation

One purpose of this study is to evaluate the performance of various classifiers with respect to varying sizes of the training dataset. Thus, we prepare several training datasets containing 500, 1000, 2000, 5000, 10000, 20000, 30000, 40000 records, respectively. We train and build a Bayesian classifier from each of the training sets by BMA over the k-best Bayesian network classifiers as described in Section 2.2. For each training dataset, we build the Bayesian classifier by setting k to various values. We then evaluate the performance of the classifier with respect to these different k values. When $k = 1$, the classifier is equivalent to a single Bayesian network classifier. The larger k is, the more models are employed for model averaging, which potentially leads to better predictive power.

We evaluate all classifiers on the same testing dataset. We compute the accuracy as the percentage of correctly classified records. Note that this is a binary classification problem, i.e. an attack or normal. A record is classified as an attack if the conditional probability of being an attack given the observation of other features is greater than 0.5; it is classified as normal otherwise. In addition to accuracy, we compute AUC as the area under the Receiver Operating Characteristic (ROC) curve, which is an estimate of the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Since

AUC does not depend on the classification threshold used, it is widely recognized as a better measure than accuracy, which is based upon a single classification threshold.

For comparison, we also build Naive Bayes classifiers and Bayesian network classifiers, which are selected by using the greedy hill-climbing search method. The training and testing processes of this two classifiers are executed in the open source software of Weka which is a collection of machine learning algorithms for solving real-world data mining problems. Weka is written in Java and runs on almost any platform. The algorithms can either be applied directly to a dataset or called from user's own Java code. When we use Weka to train and test the Naive Bayes classifiers and Bayesian network classifiers, we just need to pass a file of training records and a file of testing records separately to it and Weka gives detection accuracy and AUC result based on the input training and testing dataset. We implemented our algorithm of BNMA classifier and added it to Weka as a new algorithm. With the training dataset, testing dataset and k-best network structures given by KBest software as input, our implemented classifier can output the detection accuracy and AUC of using BNMA over the input dataset.

As described before, we repeated the experiment four times on different training and testing data sets with each training size and k value. We reported the average accuracy and AUC over the four trainings for each training size and k value. Considering the records of training dataset and testing dataset are selected randomly from NSL-KDD dataset, the averaged result over four experiments is more reliable and objective than the result based on one experiment.

2.5 Experimental Results

Figure 2.3 compares the detection accuracy of Naive Bayes (NB), Bayes Network built using greedy search (BN-Greedy) and the BNMA ($k = 1$) by size of the training dataset. First, we observe that the accuracy is approximately a non-decreasing function of training sample size. This is understandable since a larger training sample usually produces a classifier with better predictive power. The accuracies of NB and BN-Greedy are comparable to each other, while the BNMA classifier built using the best Bayesian network ($k = 1$) is significantly better than the two classifiers. Further, the BNMA ($k = 1$) trained classifiers using a small training set (2000) even outperforms the NB and BN-Greedy classifiers trained using a very large training

set (40000). The improvement is also significant when AUCs are compared (see Figure 2.4). This indicates that the BNMA classifier can achieve reasonably good predictive power even when a small training dataset is used.

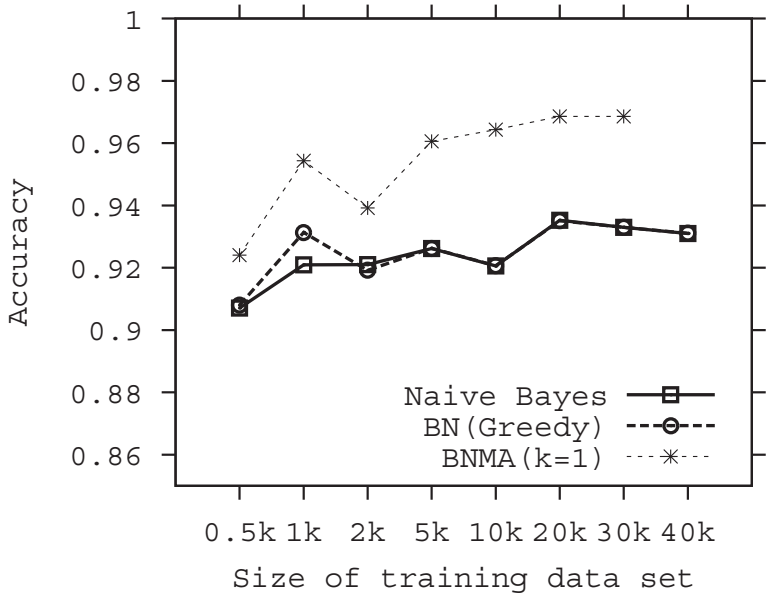


Figure 2.3: Comparison of detection accuracy by size of training set

In another set of experiments, we evaluate the BNMA classifier with respect to various k values. Table 2.2 compares the detection accuracy by k value and training set size. Table 2.3 compares the AUC by k value and training set size. It is shown that with the increase of k, both accuracy and AUC increase. AUC has a more obvious increase than accuracy. However, this improvement is not as significant as that in comparing BNMA ($k = 1$) with NB and BN-Greedy. The most significant improvement is in Table 2.3, for sample size 500, where the AUC jumps from 0.9615 for $k = 1$ to 0.9733 for $k = 200$. With the increase of sample size, the improvement decreases. This demonstrates that the BNMA is particularly effective on small sample sizes.

To investigate why the predictive power does not change very much with respect to the variation of the k value, we examine the structures of the top 10 networks produced using a training set with size 10000. Figure 2.5 illustrates the consensus structure for the 10 structures.

Table 2.2: Accuracy Comparison by k Value and Size of Training set

Training Set Size	$k = 1$	$k = 10$	$k = 50$	$k = 100$	$k = 200$
500	92.40%	92.40%	92.40%	92.40%	92.40%
1000	95.43%	95.43%	95.43%	95.56%	95.43%
2000	93.92%	93.97%	93.97%	93.97%	93.97%
5000	96.06%	96.14%	96.11%	96.14%	96.16%
10000	96.43%	96.43%	96.44%	96.47%	96.46%
20000	96.86%	96.86%	96.87%	96.87%	96.87%
30000	96.86%	96.86%	96.85%	96.86%	96.92%

Table 2.3: AUC Comparison by k and Size of Training set

Training Set Size	$k = 1$	$k = 10$	$k = 50$	$k = 100$	$k = 200$
500	0.9615	0.9706	0.9733	0.9733	0.9733
1000	0.9705	0.9718	0.9728	0.9728	0.9730
2000	0.9738	0.9738	0.9740	0.9744	0.9745
5000	0.9895	0.9898	0.9898	0.9900	0.9900
10000	0.9905	0.9905	0.9905	0.9908	0.9908
20000	0.9928	0.9928	0.9928	0.9928	0.9930
30000	0.9933	0.9933	0.9933	0.9933	0.9933

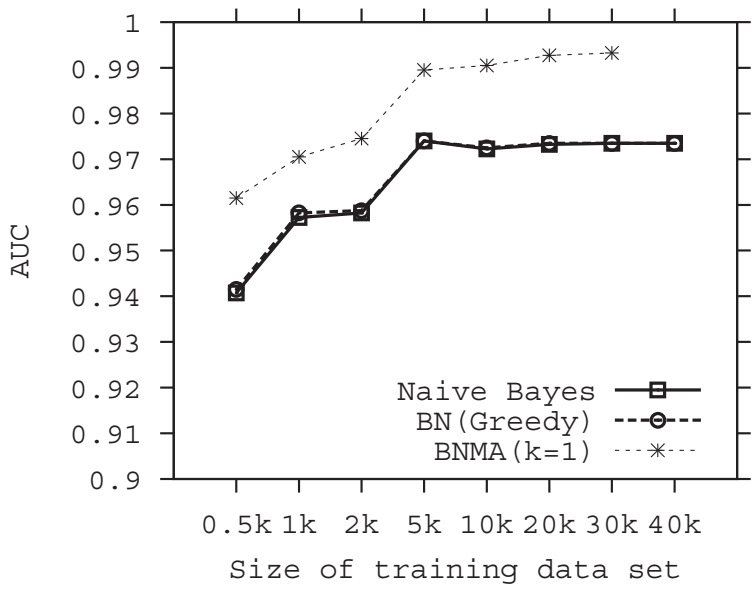


Figure 2.4: Comparison of AUC by size of training set

It is surprising that all 10 structures share the same skeleton with minor differences in the direction of the edges. This indicates that the top structures represent similar distribution, and thus make similar predictions for new data points. We can speculate that the top 200 structures may have very similar structure. For comparison, we also depict the Bayesian network structure learned using greedy hill climbing search method in Figure 2.6. It is easily observed that this structure is significantly sparser (fewer edges) than the consensus structure in Figure 2.5. This explains why the structure selected with heuristic method is suboptimal, because it fails to identify many important dependencies among the feature nodes which can be captured by our method of using BNMA classifier. It also explains why the best Bayesian network ($k = 1$) has significantly better predictive power than the BN-Greedy classifier.

2.6 Conclusion and Future Work

In this study we proposed a Bayesian classifier using BMA of k-best Bayesian network classifiers, called BNMA classifier, for intrusion detection. Previous IDS using Bayesian network classifier has two problems. First, the Bayesian network structure is selected using heuristic

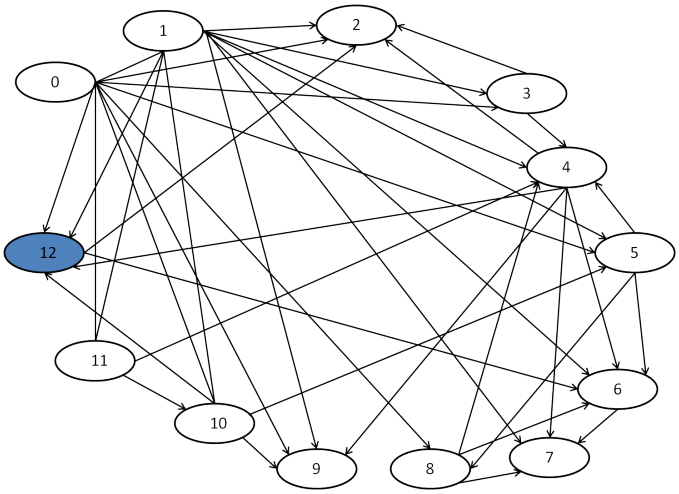


Figure 2.5: A consensus network built from the top 10 networks trained on sample size 10000. The correspondences between the nodes and the features are : 0-service; 1-src_bytes; 2-dst_bytes; 3-logged_in; 4-count; 5-srv_count; 6-serror_rate; 7-srv_serror_rate; 8-srv_diff_host_rate; 9-dst_host_count; 10-dst_host_srv_count; 11-dst_host_diff_srv_rate; 12-class (intrusion or not intrusion). Note that the class variable is shadowed. Directed edges existing in all 10 structures are depicted as solid arrows. The set of edges that exist in all structures but with various directions are depicted as solid lines.

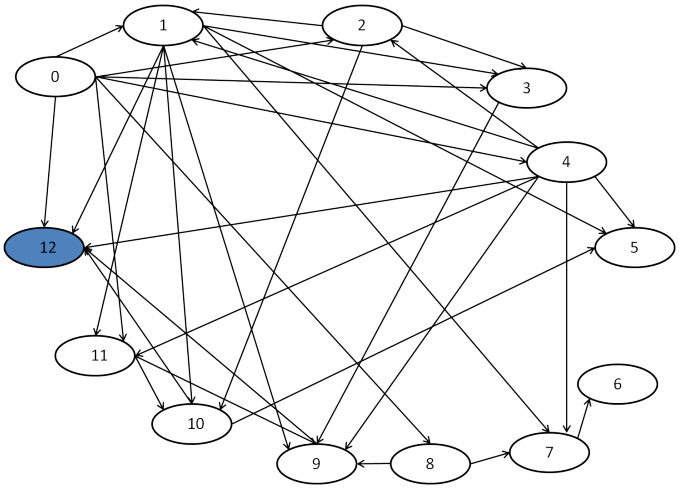


Figure 2.6: The Bayesian network trained on sample size 10000 using greedy hill climbing search method. The correspondences between the nodes and the features are the same as those in Figure 2.5.

methods, which usually return suboptimal models. Second, previous classifiers are trained and evaluated using a very large training dataset, which is usually hard to collect within a short time period. In this study, we used a DP algorithm to find the globally k-best structures and used them to build a Bayesian classifier by BMA. We showed that the BNMA classifier has significantly better predictive power than Naive Bayes and the Bayesian network classifier built using heuristic method. Even the classifier trained using a very small dataset outperforms the other two classifiers trained using a very large dataset. We then conclude that our BNMA classifier is particularly effective in detecting intrusions when only a few training records are available. This is very valuable since prompt detection of intrusion is of significant importance in such an era of rapidly growing Internet activities.

We also show that with the increase of k, i.e., more Bayesian network classifiers are used for model averaging, the better predictive power it can achieve. However, this improvement is not that significant, since the top structures actually share a very similar structure. This means the problem size (12 feature variables) is still not that large compared to the sample sizes examined. One question that users may ask is, what is the k value we should use? The answer is the larger, the better. However, it takes more time to train and integrate over larger number of classifiers. In this study, we consider 12 feature variables and $k = 100$ is already enough. The k value that should be selected depends on the problem size, i.e., the number of feature variables used to build the model. Thus, our future work is to select a larger set of feature variables for model building. Since Bayesian network is able to inherently do feature selection through its conditional dependency assertions, using a larger set of features should not significantly impact the performance. However, with a larger set of feature variables, it may need larger k, i.e., integrating over more models to achieve reasonably good predictive power.

Another area for future work is based on the observation that intrusions happen in dynamic environments, thus they themselves could be time-series data. An *et al.* [5] proposed to use dynamic Bayesian networks to model the temporal environment. However, the problems faced by the static Bayesian network classifier persist in dynamic Bayesian networks. Thus, it is a challenge to perform model averaging in the temporal environment setup.

In summary, since it uses less data while still achieving comparable or better predictive power, our BNMA classifier can save a huge amount of time on collecting training data records so that it can catch the intrusion more promptly and more accurately to avoid loss due to intrusion.

CHAPTER 3. SITUATIONAL DATA FOR INTRUSION DETECTION SYSTEM

Abstract

Intrusion detection is a research topic of great importance, especially for web-based applications, whose broad usage at the same time makes themselves attractive targets for malicious attackers. Meanwhile, applications of many emerging new web techniques (e.g., Web 2.0, HTML5, and cloud computing, etc.) add more challenges to intrusion detection. One significant issue of building Intrusion Detection System (IDS) is to find an efficient and informative training data set. In this paper, we proposed a Situational Data Model which is represented by a sequence of observed situations. Our main contributions can be summarized as follows. First, we designed Situational Data Model that has better data structure and is more informative. Each data record is a sequence of situations collected in chronological order. At each time point, the situation contains a series of information including the context and action for the possible intrusion, and potential intruder's desire. Different from action sequences data and state sequence data, Situational Data makes the training process be able to be performed over the sequence of punctuated desires which can be inferred from the sequence of actions and contexts. The valuable footprints of transition from one situation to another are well captured by this data collection mechanism. Second, we collected the training data from a real web application system Cooperative Research Environment(CoRE) which is developed for research use by Software Engineer Research Group in the Department of Computer Science at ISU. Our training data set provides valuable evaluation reference for dynamically intrusion detection system such as Hidden Markov Model based IDS. Third, we compared our Situational Data against the data comprised of action sequences data by employing the Hidden Markov Model, and the

result approves that Situational Data with desires can bring better classification accuracy and ROC curve.

Key Words: Intrusion detection system, Situ, Hidden Markov Model, detection accuracy, ROC

3.1 Introduction

In this chapter we introduce the Situational Data Model for building dynamic intrusion detection system. This Situational Data Model is based on the Situ framework introduced by Chang et al. [11], and is defined by situational data consisting of environmental context information, activities, and inferred desires.

Traditionally, there are two categories of intrusion detection systems, i.e., either misuse-based or anomaly-based. A misused-based intrusion detection system includes a number of attack descriptions or signatures trained from abnormal data. It gives intrusion alert whenever a stream of audited data matches with an attack model. On the other hand, an anomaly-based intrusion detection system relies on models trained from the normal data. Deviations of audited data from established models are interpreted as potential attacks [42]. Training data are usually collected from network, operating systems, or application log files. Various data sets are used in building all different kinds of intrusion detection systems. Through training on historical data, a model explaining the relationship between intrusion and selected features can be built. The established model is then used to predict outcomes for future audited data. For instance, Kruegel and Toth [39] created a decision tree to detect malicious events, based on a set of signatures of data constraints. Altwaijry [2] developed a naïve Bayesian classifier to identify possible intrusions. To evaluate an intrusion detection system, the public KDD Cup 1999 dataset [31] has been widely used by the researchers for experiments in the literature [14, 63]. This dataset contains 41 feature variables and up to 4 million staggering records. Also, the feature variables of the KDD Cup 1999 dataset include network traffic information such as src_bytes (number of data bytes from source to destination), dst_bytes (number of data bytes from destination to source), and protocol_type (type of protocol), etc. The classical approaches

typically first train a model based on selective features using a certain amount of data, then predict a future data record to be intrusive or normal based on the trained model. However, one common limitation of all those training data is that they only have static network traffic data, and do not take into account the dynamic nature of many features that may change over time. This may be partly due to that their feature values were collected statically without any time stamp. Therefore, they treat all different features being time-independent and ignore potential temporal relationships between the same features at different time points. In fact, the dynamic transitions of features' value in chronical order often carry important information, and can be very valuable for detecting the intrusions. For example, under statistic data structure, a potential anomalous behavior that repeatedly have massive amount of data bytes flowing from source to a destination at different time periods may not be distinguishable from a normal behavior with similar situation at a single time point. In this case, it may be very challenging for us to catch the intrusion if we just model the static context information in spite of the features being time dependent.

Dynamic features have been shown to be more efficient than static features. In fact, some researchers have been using sequential data for training intrusion detection models. For instance, Ariu et al. [7] designed the intrusion detection system HMMPayl, which uses the Hidden Markov Model to analyze the HTTP payload, to detect attacks against Web applications. Those authors expressed the payload as a sequence of bytes and showed that the HMMPayl is very effective against attacks like Cross Site Scripting [22] and SQL-Injection [35], whose payload would not be significantly different from that of normal traffics. On the other hand, the detection system proposed by Chen et al. [13] analyzed multiple logs from cloud to extract the intention of the actions recorded in the logs. In their work, the Hidden Markov Model was adopted to model the sequence of actions performed by hackers. Such stealthy events in a long time frame is significantly more useful for training the state-aware model. Besides, in [27], sequences of systems calls were used to build the classification schema. However, all those sequential features and data used in their IDS training do not have the human internal mental states (desires) involved. Human desires which could be driven from other features are more correlated to or more representative of human's intention than those features. In this thesis, we

propose a situational data model with actions, contexts of target web application system and desires of human. We collect and parse the data following the Situ framework [11]. That is, we represent each data record as a sequence of situations, of which each contains three components – desire, action, and context. We then compare Situational Data Set with action-only sequences dataset using the Hidden Markov Model. Our experiment result shows that situational dataset is more effective by having the training process guided by human desires.

The rest of this chapter is organized as follows: Section 3.2 introduces the related work Hidden Markov Model and Situ Framework. Section 3.3 describes the Situational Data Model and the process of Situational dataset collection. Section 3.4 gives our experiment design and results. Finally, we conclude in Section 3.5 with a summary of the current work and a discussion of potential future work.

3.2 Related Work

3.2.1 Hidden Markov Model

A hidden markov model (HMM) is a statistical model representing probability distributions over a sequence of observations, say $\{x_1, \ldots, x_T\}$, accompanied by hidden states $\{y_1, \ldots, y_T\}$ [21]. As a simple dynamic Bayesian network model, a HMM can be illustrated by Figure 3.1, and there are two defining properties: hidden, in the sense that all observations $\{x_t, t = 1, \ldots, T\}$ are assumed to be generate by some process whose states $\{y_1, \ldots, y_T\}$ are hidden; Markov property, in the sense that the distribution of those hidden states have markov property, i.e., at any time point t, the distribution of y_t given all its history $y_1, \ldots, y_{t-1}$ is equivalent to that of y_t given y_{t-1} only. In addition, the conditional distribution of any the observations x_t given all other nodes of the graph is the same as its conditional distribution given its immediate parent y_t. Bringing those markov properties together, the joint distribution of $(X, Y) = (x_1, \ldots, x_t, y_1, \ldots, y_t)$ can be written as

$$p(X, Y) = \prod_{t=1}^{T} p(x_t \mid y_t) p(y_t \mid y_{t-1}),$$

where we adopt the convention that y_0 represents null status with no information.

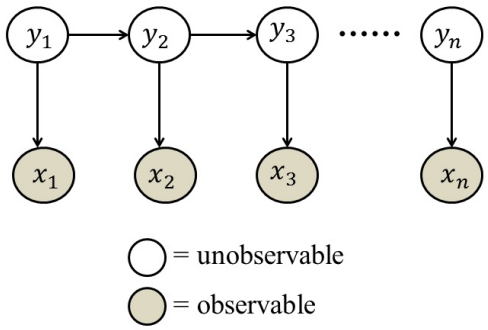


Figure 3.1: Hidden Markov Model

HMMs have been widely used in the literature for researches in different disciplines, e.g., economics, computer science, etc. HMM is also very often used in intrusion detection [7, 13, 27]. In applications, typically a parametric model is assumed for the joint distribution of $p(X, Y)$ such that $p(X, Y) = p(X, Y; \theta)$, and the goal of learning task is to obtain an estimator for θ based on observed outcomes $X = (x_1, \ldots, x_T)$. Intuitively, because $Y = (y_1, \ldots, y_T)$ are hidden, i.e., unobserved, θ may be estimated by the marginal distribution of X, i.e., $p(X; \theta) = \sum_{y_1,\ldots,y_T} p(X, Y; \theta)$. However, obtaining the marginal distribution may be challenging in practice and computation can be intensive using this brute-force approach. Nevertheless, in the literature efficient computational algorithms are available, e.g., E-M algorithm based Baum-Welch method [45], among others. In our discussion here, we use HMM model as a candidate method for comparing the effectiveness of Situational Dataset with another dataset in training Intrusion Detection System and classifying intruders and normal users. Details regarding how HMM is applied in our setting and how HMM is to be combined with the Situ framwork will be introduced in the upcoming discussions and section.

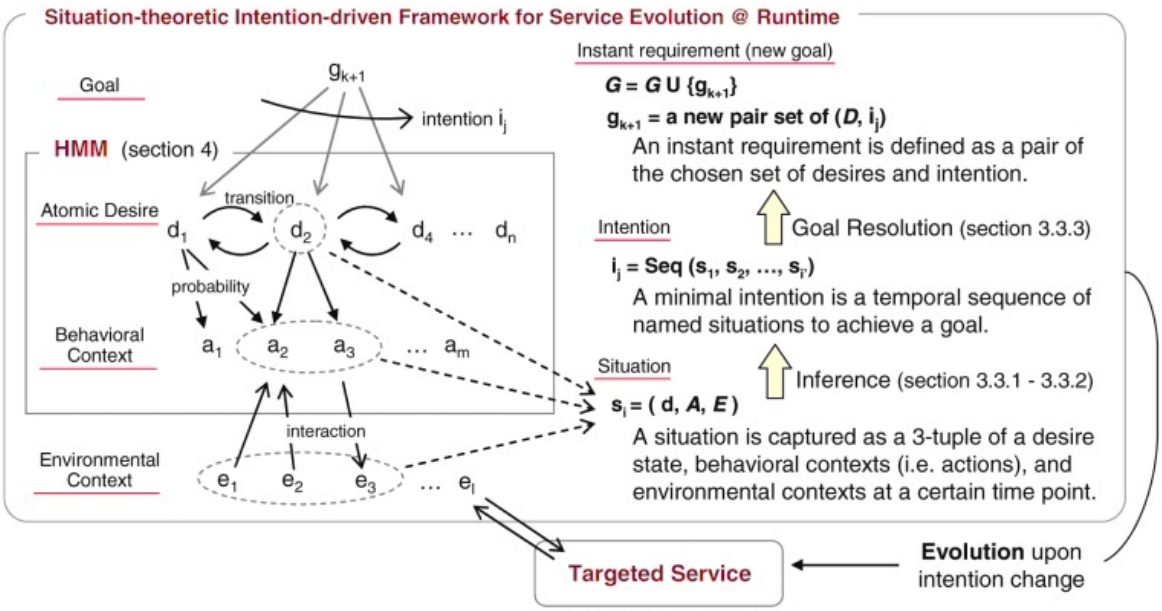


Figure 3.2: Situ Framework

3.2.2 Situ Framework

The data records used for training and prediction is modeled following the Situ Framework. Situ Framework [11] originally provides a situation-theoretic approach to human-intention-drive service evolution in context-aware service environments. The situation defined in this framework is rich in semantics and useful for capturing human thinking and behavioral patterns, which, in turn, help developers to construct the intention specification. Figure 3.2 from [11] shows the situation-theoretic intention-driven framework for service evolution at runtime.

In Situ framework [11], a human intention is defined as temporal sequence of situations to achieve a goal, which is described as $I = \text{seq}(S_1, S_2, \ldots, S_k)$ where $S_1, \ldots, S_k$ are goal-directed situations for the goal g. A situation at time t is expressed as situation (t). It is a triple $\{d, A, E\}$ in which d is the predicted users desire, A is a set of the users actions to achieve a goal which d corresponds to, and E is a set of environment context values with respect to a subset of the context variables at time t. As shown in Figure 3.2, an intention plays a proactive role in interpreting human actions, which are the key component connecting desires and specific contexts. The contexts are derived from sensing the entire environment surrounded by a user

being observed in a service system. This type of contexts such as time, location and so on is helpful in understanding the triggers and effects of human's actions. The actions are derived from human behaviors which interact with contexts bidirectionally. Situ is a higher level formal mechanism to understand human intention change by the situations.

3.3 Situational Data for Intrusion Detection

3.3.1 Definition of Situational Data Model for Intrusion Detection

In previous intrusion detection researches, the KDD Cup 1999 dataset has been widely used to build and evaluate the Intrusion Detection System [14, 63]. The KDD Cup 1999 dataset [57] include many relevant features such as connection, traffic, and content, etc. However, all of those features are static, in that no time stamps are associated with those features that could potentially change over time. Therefore, a potential layer of correlation or connection between those features is missing. In other words, features collected at two different time points, say t_1 and t_2, are treated as two entirely independent records regardless of their time stamp. We call those type of data as static data. In a static data model, the over-time trajectory of features are not collected thus cannot be used in training or utilized to predict audited data in the future.

When only static data are used for training, it is very difficult to detect malicious attacks that have similar network traffic features as normal behaviours, e.g., SQL injections. This naturally motivates making use of training data that can capture not just statistic features but also dynamic characteristics. For instance, sequential data that are collected with time stamps would help counter this challenge. In fact, sequential data have already been frequently used in the literature. For example, Ariu et al. [7] defined each data record as a sequence of payload bytes length collected at different time points. While this data model captures the payload bytes length in a sequential manner for intrusion detection, it is, however, not effective for detecting attacks that do not cause significant payload changes over different time points. For illustration, we can think of a simple example where a brute-force type attack is considered. Specifically, suppose an attacker repeatedly tries to log-in by guessing passwords without being

successful. In this case, the payload bytes length at each of those attempts will not differ much from each other, thus the system will probably fail to detect this attack.

Given reasons above, we believe that it can be more effective for intrusion detection if we consider not only the change of general context, e.g., payload byte length, but also other factors such as sequences of users' behaviors and thinking patterns. Following this thought, we build our Situational Data model based on the Situ framework [11]. According to the correlations of intention, desires, context, and actions described in Situ, we design our Situational Data model with three layers as shown in Figure 3.3. Specifically, let $t_1, \ldots, t_n$ be data collection time points in natural chronical order. The first layer represents action and context sequences, denoted by $[A_1, C_1], \ldots, [A_n, C_n]$. Then the second layer captures the desire sequence, $d_1, \ldots, d_n$. The third layer gives the temporal intention sequence $tempI_1, \ldots, tempI_n$. A user's footprint left in the system is expressed as a sequence of situations, say, $S_1, \ldots, S_n$. Each situation S_t at time point t consists of $[A_t, C_t], d_t$ and $tempI_t$, where $[A_t, C_t]$ is the set of actions and context observed at time t, d_t is the user's desire on the system at time t, and $tempI_t$ is a quantitative value indicating the degree of the segmental attacking intention for the desire at time t. A user's intention is then composed of a sequence of situations.

For intrusion detection purposes, one of the key components of the Situational Data Model is the desire layer which plays a central role as a bridge in the process of inferences in classification. It helps find hidden relationships among those trivial context and behavior information, then help correlate them with users' temporal intentions. It guides the predictive model with clearer direction on classifying intruders and normal users. When attackers have malicious intentions, they tend to plan and execute the attack step by step. Using a metaphor to illustrate the relationship among actions, environment context, human desires, and intention in a system, we can think of the preparations a smart robbery might do after he decides to rob. The intention of rob drives him to have some temporal desires such as avoiding to be recognized, avoiding to be recorded by camera, confirming no polices around, and so on. Those temporal desires then drive him to have some actions and context information like wearing sun glasses, wearing a mask, observing possible cameras, and so on, leading to a abnormal sequences of actions and environment context. The same for attacking computer systems, a user with malicious intention

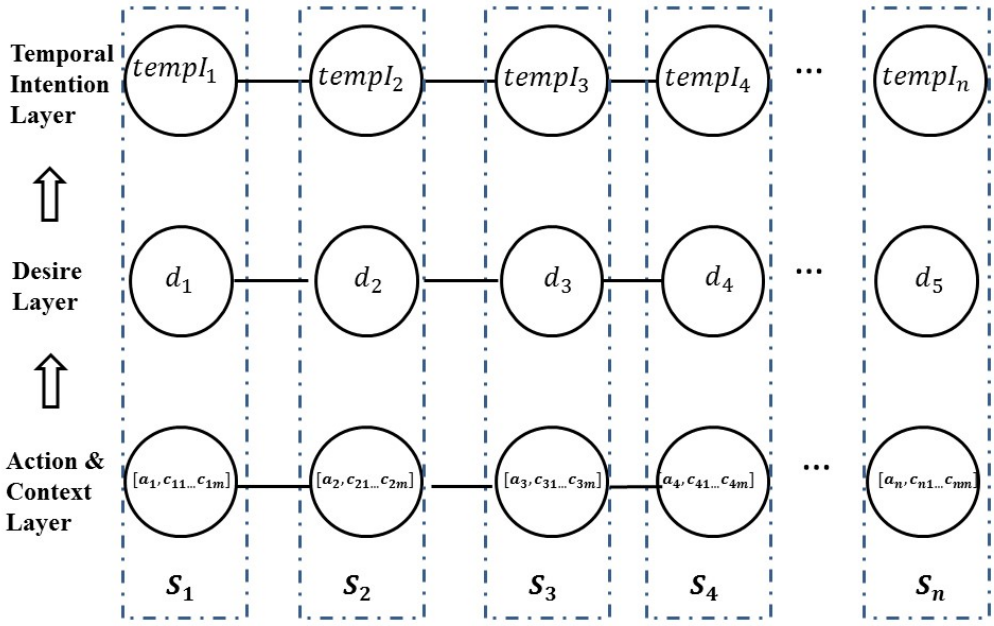


Figure 3.3: Situational Data Model

must have attacking-driven desires at some of the time points in the situation sequence. Those desires will then drive the attackers to learn vulnerabilities and information about the target system before he is able to intrude eventually. During the process of learning vulnerabilities and trying attack, the attackers would have abnormal behaviors, which make the environment context different from that of normal users'.

To use this Situational Data Model, we need to train two prediction models. The first model is a desire model for predicting the desire sequence based on action and context sequences. The other model is an intention model for predicting temporal intention by desire sequences. When IDS is used to predict and detect intrusion, firstly we treat action and context sequences, i.e., [A, C], as observed data and desire sequences as hidden state data to be predicted. Secondly, the predicted desire sequences from the desire model will be used as input of the intention model for predicting the temporal intention sequences. Finally, the trained IDS can classify an intention to be normal or malicious according to the values of the temporal intention sequence $tempI_1, \ldots, tempI_n$.

3.3.2 SQL Injection

In this subsection, we briefly describe one of the intrusion techniques used by assumed intruders in our data collection platform, i.e., CoRE. We explain how SQL injection happen since it is most frequently used by our assumed intruders.

In web based applications, when normal users submit their input information on the client side, the input value will be combined by a SQL statement written by programmers. For instance, suppose a normal user enters *marcus* as the username and *secret* as the password in the log-in page from the client side. The web-based application will interpret it in the way that is described by the following SQL statement:

SELECT * FROM users WHERE username = '*marcus*' and password = '*secret*'

Subsequently, this statement will search the user database and retrieve all relevant information that matches the user name *marcus* and password *secret*.

When a malicious user intrudes the system using the SQL injection technique, the main idea is to bypass the above user name and password check by injecting some input values that can turn the above concatenated SQL statement into an always-true statement regardless of the actual user name and password provided. This is because the website log-in program is written in such a way that it interprets log-in information by combining the phrases from "username" and "password" fields in the SQL statement, in the same way as the example shown in the previous paragraph. As a result, because the SQL statement is true, the log-in system will then grant access to the intruder. For example, suppose an attacker knows the database administrator's user name, *admin*. Then the attacker may capitalize on this and use this information in the log-in page by entering the username *admin'–*. This input will then generate a SQL statement as follows:

SELECT * FROM users WHERE username = '*admin*'- -' and password = '*anything*'

Note the sign $'--'$ is an annotation mark in SQL syntax, thus all the following statement after the $'--'$ will be regarded as potential annotations of the SQL statement. As a result,

the above SQL statement is actually equivalent to the following statement:

$$\text{SELECT * FROM users WHERE username} = \text{'}admin\text{'},$$

which is true as *admin* is an admin's username as aforementioned. In other words, no matter what value of password is provided in this case, the SQL statement is always true and the program will then respond with information associated with the username *admin*. Meanwhile, this also means that an attacker can use any username from the database and apply this technique to bypass the log-in page.

According to this SQL injection attack technique, the actions users did on the web site usually cause some database operations like $select, delete$, and so on, in the back end side of the application. The input information are actually used as parameters for the back end operation. The response type of URL request is the feedback that we got after the back end operation is done. We gathered those information on time stamps for our Situational Data Set which helps us to find the real intention of the users.

3.3.3 Collection of Situational Data for Intrusion Detection

We collect our Situational Data Set based on the real dynamic web application system, Co-operative Research Environment (CoRE), which is used for sharing published and unpublished internal research papers, comments and ideas, etc. This system is modified from an open-source web application called MyReview [48] by the Software Engineer Lab in the Department of Computer Science at Iowa State University. The CoRE is used as an online library system by scholars at Iowa State University. Upon logging in the system by assigned user name and password, a user is able to upload papers, download papers, write comments, view comments, and search by key words, etc. A sketch of different functionalities of this system is displayed in Figure 3.4.

Before the experiment was conducted, an approval was obtained from our Institutional Review Board (IRB) due to its involvement of human subjects. Our experiment complies with the federal regulations set forth by the Department of Health and Human Services and the Food and Drug Administration. In addition, all principal investigators in this experiment

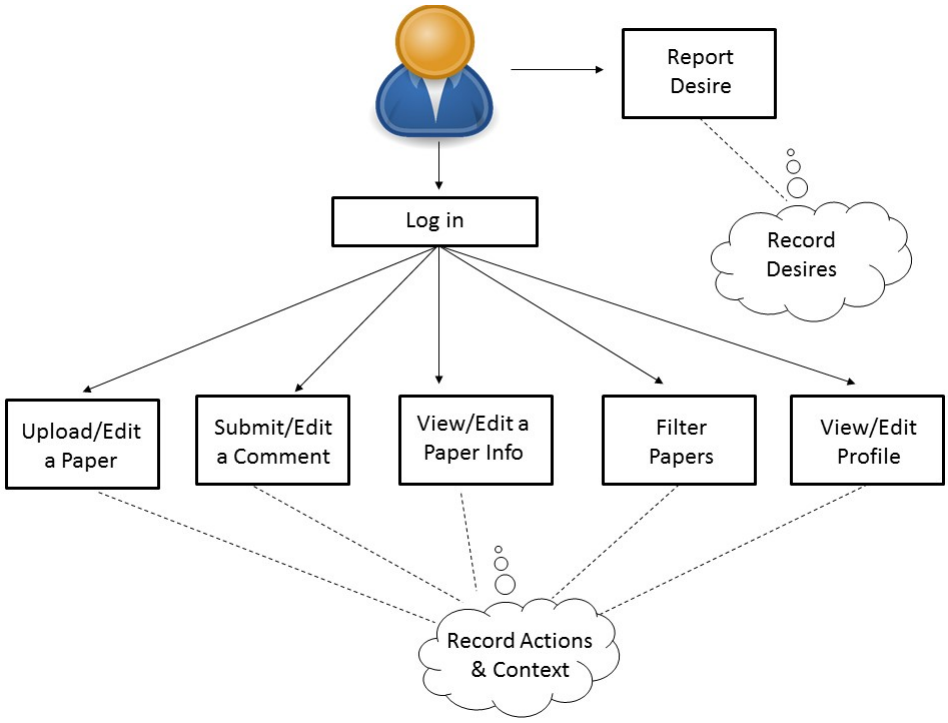


Figure 3.4: Cooperative Research Environment System

took the National Institutes of Health (NIH) Web-based training course Protecting Human Research Participants before their participation in the experiment. A total of approximately 120 participants, including normal users and assumed intruders, have contributed to this experiment study, where normal users used the CoRE system normally, and assumed intruders performed as intruders to attack the system using different attacking skills.

We collect the participants' actions, e.g., clicking different functional buttons and URLs, and the context information of the system, e.g., input information in text box of the web site and HTTP response type after sending a URL request. To capture users' actions and related context information of the system, a monitor program is embedded in the CoRE system as a sensor. We also collect users' desires each time when they have actions on the system. The participants' desires are collected by the "think-aloud" method, which is a questionnaire embedded in the system. Participants choose or enter their desires at different time points during the process of using the system. When the Situational Data Model is used in practice,

those desires cannot be collected from users thus they need to be manually labeled. In this data collection, we manually label temporal intentions based on the actions and context information.

As shown in the Table 3.1, we collected data records, including actions, contexts, and desires, at 8645 different time stamps, of which 6880 were generated by normal users and 1765 were generated by assumed intruders. Meanwhile, if we look at those records from situation sequence point of view, there are a total of 462 sequences, consisting of 365 normal sequences and 97 malicious sequences.

Table 3.1: Data summary

	Normal	Malicious	Total
Number of Records	6880	1765	8645
Number of Sequences	365	97	462

On the other hand, in terms of numbers of distinct values, our data contain 66 possible actions, 3 types of URL request errors or exceptions, 8 types of information inputs, 33 types of desires, and 7 possible temporal intention values. The malicious intention set includes SQL-injection attack, unauthorized url retries, authentication attack, etc. All possible values of intention, temporal intention, and contexts are given in the following:

$$I = \{\text{malicousI, normalI}\}$$

$$tempI = \{0, 1, 2, 4, 6, 7, 10\}$$

$$\text{Context} = \{\text{No Error, Error Remind, Error Report, Legal Input, No Input,}$$
$$\text{Legal Input, Sql Input, Inserted Link, Url Action, UrlGuessInput, Url PaperID}\}.$$

All possible values of actions and desires are shown in Table 3.2.

For illustration purpose, let us consider a simple example of situation sequences presented in Table 3.3, which is a subset of a user's records in our collected data from CoRE. Each row of this table is a situation, including values of action, context, and desire at a time point $t\,(t = 1, \ldots, 10)$. The user behaved normally in situations S_1 and S_2. From situation S_3 to situation S_6, the user made two attempts logging into the system illegally using SQL injection techniques. Following that, starting from situation S_7 until situation S_{10}, the user tried to

Table 3.2: Possible values of action and desire

Action		Desire
clickMenuMyProfile	clickSendInstruction	Test
clickMenuHome	clickMenuConfigSystem	ViewAllPapers
clickMenuSignin	clickUploadPaper	Log In
clickSignup	clickDownloadAllPapers	ViewAPaperInfo
clickLogin	clickSubmitEditPaper	DownloadPaper
clickMenuListUsers	clickLinkPrintMyComments	ViewMyPapers
clickLinkModify	clickSubmitPreQuestion	Get User Names
clickModify	clickCancelMyProfile	UploadPaper
clickEditMyProfile	clickHideSelection	ViewProfile
clickUpdateMyProfile	clickShowSelection	ViewHomepage
Null	clickCancelSubmitComment	SubmitComment
clickSkipPreQuestion	clickCancelUploadPaper	View All Papers
clickMenuAllPapers	clickFilter	FilterPapers
clickRemoveCommentMyPapers	clickPaperInfosMyComments	ViewMyComments
clickOkRemoveComment	clickPaperInfosMyPapers	ViewMyCommentInfo
clickMenuMyPapers	clickDownloadMyPapers	ViewMyPaperInfo
clickMenuMyComments	clickSubmitEditComment	EditPaper
clickEditPaperMyPapers	clickEditCommentMyPapers	EditComment
tryURL	clickCancelPostQuestion	Guess Password
clickPaperInfosAllPapers	clickCancelEditPaper	EditProfile
clickEndSession	clickCancelEditComment	RemoveComment
clickMenuSignout	clickEditCommentAllPapers	Sign Up
clickEditCommentMyComments	clickSubmitMyProfile	Register Illegal User
clickLinkRemoveMember	clickCommentMyPapers	Skip Login
clickOkRemoveMember	clickICSE	Modify User Information
clickMenuUploadPaper	clickFairUse	Access Page Unauthorized
clickCommentUploadPaper	clickEditPaperAllPapers	Search Administrator Content
clickSubmitComment	click(Link_EditPaper)	Access Paper Unauthorized
clickSubmitCommentAllPapers	clickEditPaperMyComments	Remove User
clickLinkAddNewAccount	clickDownloadMyComments	Create User
clickAddNewAccount	clickSendPsw	SendMaliciousLink
clickNoRemoveComment	clickRemoveCommentMyComments	clickLinkPrintMyComments
clickLinkSendInstruction	clickContact	clickSubmitComment

modify the database by removing certain users and tried to send malicious links to the existing users in the database. This is an intrusion example with very obvious intrusion intention that could be easily detected. For many other smart intruders, we may not be able to collect situation sequences with such obvious patterns. For instance, they may switch frequently between behaving normally and abnormally alternatively to hide their intrusion intentions.

Table 3.3: An example sequence of Situational data set

Situ	Action	Input	Error Type	Desire	tempI
S_1	clickSkipPreQuestion	No Input	No Error	Test	0
S_2	clickMenuSignin	No Input	No Error	Login	0
S_3	clickLogin	Sql Input	Error Remind	Skip Login	1
S_4	clickLogin	Sql Input	Error Remind	Skip Login	2
S_5	clickLogin	Sql Input	Error Remind	Skip Login	4
S_6	clickLogin	Sql Input	No Error	Skip Login	7
S_7	clickLinkRemoveMmber	No Input	No Error	Remove User	1
S_8	clickOkRemoveMember	No Input	No Error	Remove User	2
S_7	clickLinkRemoveMmber	No Input	No Error	Remove User	4
S_8	clickOkRemoveMember	No Input	No Error	Remove User	7
S_9	clickLinkSendInstruction	No Input	No Error	SendMalicousLink	1
S_{10}	clickSendInstruction	No Input	No Error	SendMaliciousLink	2

Although the situational data set under our discussion is collected based on target system domains, the methodology we proposed can assist in building IDS for any systems. When researchers or practitioners collect data to build their IDS, actions and context can be captured automatically and desires and temporal intentions need to be labeled manually. On the other hand, data set may be collected from different web applications or platforms, nevertheless they can be shared and combined as long as they have similar components based on intrusion detection investigation features. For example, we can build one IDS that is applicable for some commercial web sites specially and build another one for some financial web sites specially. Because the collected data can be used across different platforms, it is relatively easy for us to maintain and update the training data set for improving detection performance and handling possible new attack patterns.

3.4 Evaluation of Situational Data for IDS by HMM

Recall that our objective in this chapter is to evaluate our Situational Data Model compared to conventional data with action sequences only. Specifically, we apply the HMM model on data consisting of only action sequences and data under the Situational Data Model. To proceed, we first provide details regarding our experiment setup and descriptions of software for training HMM in Section 3.4.1, then we discuss results in Section 3.4.2.

3.4.1 Description of Experiment

For simplicity, let us denote the data set corresponding to action sequences only as A, and the data set with sequences of situations as B. For each of the two data sets, we partition the data into training set and testing set first, then generate another validation set independently. Specifically, we use stratified sampling such that a simple random sampling framework is applied and is stratified by normal sequences and malicious sequences, such that similar portions of malicious records in each of the training and testing sets can be obtained. In our experiment, we use 50-50 split of the entire data, such that 50% of the sequences will be randomly selected to serve as the training set with the rest serving as the testing set. A random number generator seed is selected such that results are reproducible. The validation set is then independently sampled from the entire data containing similar number of sequences as the training or testing data, i.e., 50% of the sequences. In general, we use the training set to train an HMM model and obtain an IDS, the validation set to select decision thresholds and the test set to compare performance.

As we discussed in Section 3.2.1, a parametric model $p(X, Y; \theta)$ is often adopted for HMM model, where X are observed data and Y are hidden state variables. Let $\mathcal{X}$ and $\mathcal{Y}$ be sets of all possible distinct values of X and Y, respectively. To implement HMM, we adopt the jahmm package [20], which uses the Baum-Welch algorithm. Before it can be run, this java package needs to have observations and initial estimates of parameters in an HMM model. Those parameters and their estimates of initial values are described below.

- $\{\pi(y) \mid y \in \mathcal{Y}\}$: Marginal probability distributions of the hidden state variables at starting time point. For each potential value, say y, of a hidden state variable, its initial value can be estimated by evaluating the proportion of sequences whose initial hidden state value is y.

- $\{\pi_{y_i, y_j} \mid y_i, y_j \in \mathcal{Y}\}$: Transition probability matrix of all possible values of a hidden state variable. For the transition probability from y_1 to y_2, this initial value can be calculated by counting the proportion of transition records $y_1 \to y_2$ out of all transition records $y_1 \to$ anything.

- $\{\pi_{x|y} \mid x \in \mathcal{X},\, y \in \mathcal{Y}\}$: Conditional probabilities of an observed variable taking a particular value, say x, given each possible value of a hidden state variable, say y. The initial value of this parameter can be estimated by calculating the proportion of records that link hidden state variable's value y and observed variable's value x among all records that have hidden state value being y.

When an HMM model using jahmm finishes running, the output contains final estimates of all the aforementioned parameters. In addition, it also includes the most probable outcome for each observed data Y. In our cases, the hidden state is temporal intention, with values ranging from 0 to 10. Note that those values do not directly translate into whether an observation corresponds to a normal or malicious intention. There are many possible ways to utilize the predicted temporal intention values for classification purpose. We use the average value of the predicted sequence, that is, suppose a predicted sequence of temporal intentions are $tempI_1, \ldots, tempI_T$, then we use

$$\text{avg.tempI} = T^{-1} \sum_{t=1}^{T} tempI_t. \tag{3.1}$$

A large value of avg.tempI indicates high intention of being malicious. Therefore, our decision rule based on predicted temporal intension is simply

$$\psi(\gamma) = I\{\text{avg.tempI} > \gamma\},$$

where γ is a threshold value, and a sequence will be classified into being malicious if $\psi(\gamma) = 1$ and normal otherwise. For the choice of threshold value γ, we will use the validation data to

select a γ such that some pre-specified criteria will be satisfied with desirable performance. For instance, we can check the accuracy of prediction, i.e., the proportion of sequences, including both malicious and normal sequences, that are correctly classified, and select the γ^* that maximizes accuracy. Apply this selected γ^* to the test data, and compare performance of different methods based on accuracy or other similar criteria. We give more details of evaluation criteria and threshold determination in the upcoming section.

For data set A where only action sequences are used, it is straightforward to apply the above HMM model by using action sequences as observed data X, and temporal intensions as hidden state variables Y. When our Situational Data Model (see Section 3.3.3 for more details) is used, i.e., data set B is concerned, we apply the HMM twice through a two-step procedure. In the first step, in addition to the action sequence, we also consider context sequences. Meanwhile, we consider the desire as a hidden state, and in the first step, the goal is to train a model that predicts desires based on action and context sequences. Note that in our training data, we have desires provided by participants. Following this step, we train another HMM model by treating desire sequences as observed data and temporal intention sequences as hidden state data. When the trained IDS model is used on the testing data set, we first predict the desire based on the model we built in the first step, then use the predicted desire in our trained model obtained in the second step to further predict the temporal intention sequence. We leave details of their comparisons in the upcoming discussion.

3.4.2 Experiment Results

To compare performance of the HMM applied to sequence of actions and the HMM applied to sequence of situations, we introduce the following measurements. Specifically, we consider accuracy, i.e., proportion of correctly classified outcomes; false positive rate, i.e., proportion of misclassified normal sequences; true positive rate, i.e., proportion of correctly classified malicious sequences, and ROC curve. In mathematical notations, without loss of generality, we can assume that the first n sequences are normal and the rest $N - n$ sequences are malicious, and let δ_i be the classification rule of sequence i, $i = 1, \ldots, N$, such that $\delta_i = 1$ indicates that sequence i is classified as being malicious or normal otherwise ($\delta_i = 0$). Then the accuracy

(ACC), false positive rate (FPR), and true positive rate (TPR) are defined as follows:

$$\text{ACC} = \frac{\sum_{i=1}^{n}(1-\delta_i) + \sum_{i=n+1}^{N}\delta_i}{N}, \tag{3.2}$$

$$\text{FPR} = \frac{\sum_{i=1}^{n}\delta_i}{n}, \tag{3.3}$$

$$\text{TPR} = \frac{\sum_{i=n+1}^{N}\delta_i}{N-n}. \tag{3.4}$$

For any intrusion detection method, an associated threshold γ for a decision instrument, e.g., posterior probability of being malicious, is typically selected to form a decision rule. And for each decision rule, based on the results, ACC, FPR and TPR can then be evaluated. A ROC curve would be created by collecting all (FPR, TPF) pairs for all possible threshold choices. Also, for simplicity, we denote the two HMM methods as HMM1 and HMM2 in the following:

$$\text{HMM}_1 : \text{HMM method making use of only action sequences;}$$

$$\text{HMM}_2 : \text{HMM method making use of Situ sequences.}$$

In Figure 3.6, we display the accuracies of HMM_1 and HMM_2, where the x-axis is the threshold value γ. For each of the HMMs, accuracy curves for both validation and test are shown. As we can see, except minor difference for HMM_2, validation and test data exhibit very similar accuracy curve performance. If we use the accuracy measurement as a guide to select threshold value for each decision rule, then for HMM_1, a threshold $\gamma_{hmm,1}$ can be selected as approximately 2.75, leading to about 85% accuracy as its maximum. For HMM_2, a threshold $\gamma \in [0.5, 1]$ results in a maximum accuracy of approximately 95%. Those accuracy curves show that HMM_2 produces better accuracy in terms of both the maximum value and overall profile. Meanwhile, the shape of the curve with respect the threshold value also makes sense. In particular, for an extremely small threshold, δ_i is more likely to be classified into being malicious, thus majority of δ_i will be 1. Based on equation (3.2), the accuracy will be close to $1 - n/N$, corresponding to the accuracy under the case when threshold is 0. On the other hand, when the threshold is extremely large, very little to no sequences may be classified as being malicious, i.e., majority of δ_i's are 0, thus the accuracy measurement will be very close to n/N. Therefore, the accuracy should increase when the threshold value initially increases from

0, and reaches its maximum at a certain point then start to drop when the threshold continues to grow.

(a) HMM using Action Sequence Only

(b) HMM using Situ Sequence

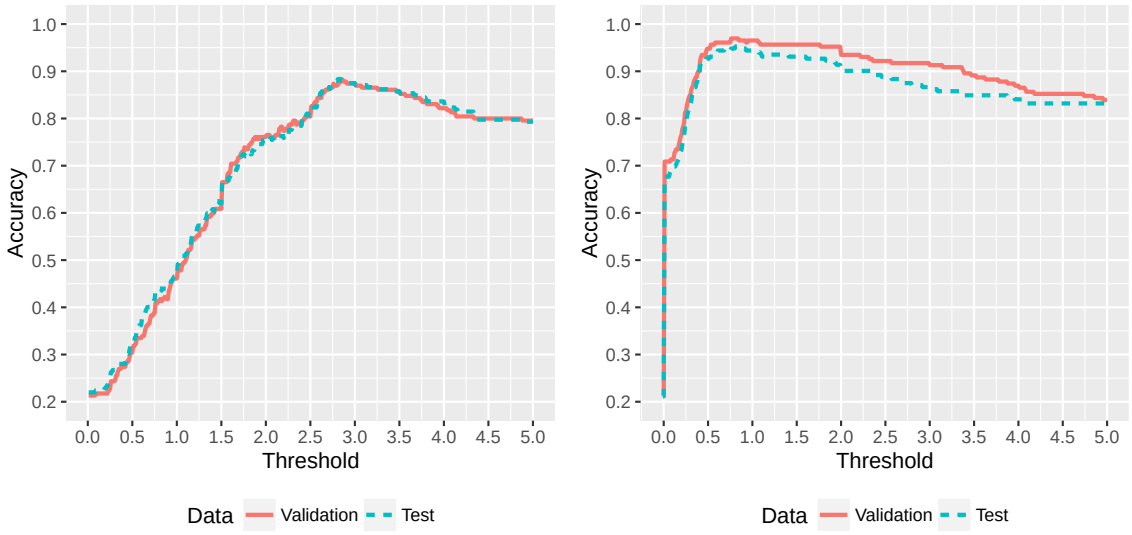


Figure 3.6: Accuracies of HMMs

An alternative approach to compare them is the ROC curve, which consists of all possible (FPR, TPR) pairs. We show ROC curves for HMM_1 and HMM_2 in subfigures (a) and (b) of Figure 3.8, respectively. In terms of ROC curves, the one that has larger area under the curve, or is closer to top left corner, is deemed to have better performance. This is because, if we fix the FPR and check the TPR, then the curve that has higher TPR value is better. This is essentially very similar to hypothesis testing where we control or fix one dimension, e.g., FPR or type I error, and look at the other TPR or power. Figure 3.8 shows that HMM_2 has better ROC curve property than HMM_1. That is, if we fix FPR to be around 5%, then HMM_1 gives about 50% FPR, while HMM_2 gives about 85% to 90% FPR. We do notice some visible difference between the ROC curves of validation and test datasets for each of the two HMM models. Specifically, if we fix FPR, and check the difference between TPRs of validation and test dataset, the maximum difference is about 6% for HMM_1 and about 10% for HMM_2. However, this may not be surprising due to the random variation of sampling.

Figure 3.8: ROC Curves of HMMs

To link the accuracy figures with the ROC curve figures, we also present the relationship between threshold values and their corresponding FPR and TPR in Figure 3.10. When the threshold value is about 2.75 for HMM_1, the FPR for both validation and test are controlled under 5%, however, the TPR is about 48%. That is, even that the accuracy for HMM_1 is about 85%, the TPR is still not very ideal. This is because the accuracy measure, i.e., ACC, in (3.2) can be regarded as a weighted average of FPR and TPR. In fact, ACC can be written as

$$\text{ACC} = \frac{n}{N}(1 - \text{FPR}) + \frac{N - n}{N}\text{TPR}. \tag{3.5}$$

In our example, because malicious events are relatively rare compared to normal events as it's mostly the case in practice, the accuracy measurement puts more weight, i.e., n/N, on FPR. Now let us examine FPRs and TPRs with respect to threshold values for HMM_2. As mentioned earlier, the accuracy performance is fairly well and stable for a range of threshold values between $[0.5, 1]$, as reflected in Figure 3.5b. If we take a threshold value of 0.5, then the associated FPR is controlled under 5%, and the TPR is about 90% for the validation data set and about 85% for the test data set. This makes the advantage of HMM_2 over HMM_1 more visible and pronounced.

From this exercise, we also see that accuracy may not be the only option to guide the determination of a threshold value. More broadly speaking, a general weighted version of accuracy can be proposed as

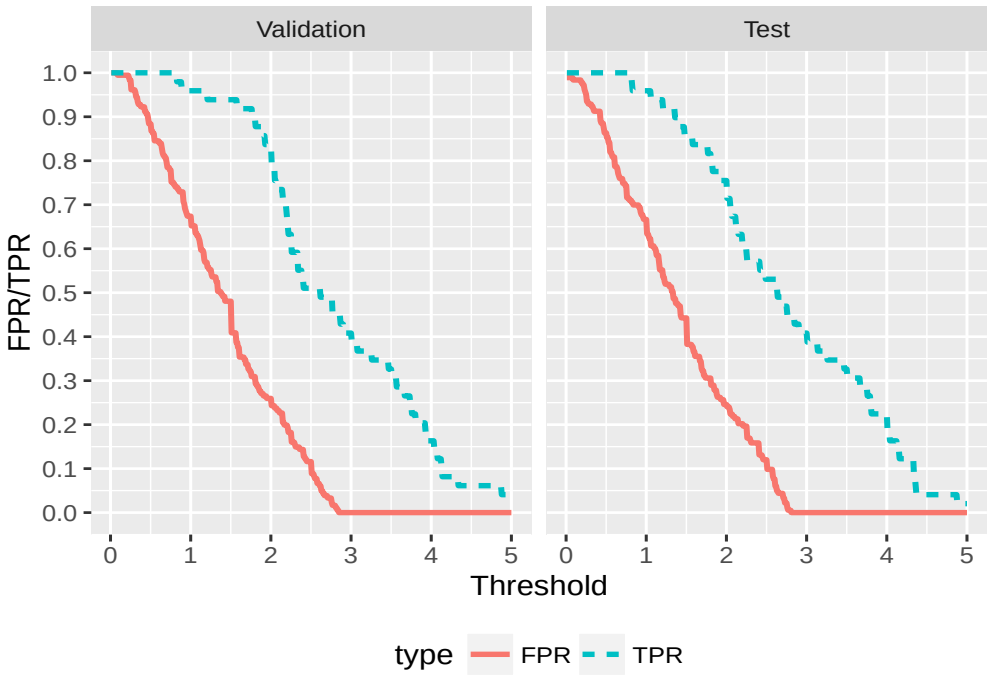
$$\mathrm{ACC}_w = w(1 - \mathrm{FPR}) + (1 - w)\mathrm{TPR}.$$

A threshold can then be selected using the validation data, based on a weighted version of accuracy measurement.

3.5 Conclusion and Future Work

In summary, we introduce a new data model, i.e., the Situational Data Model, to build an IDS. We give definition of this model, and elaborate on how this model can be applied in an HMM framework for intrusion detection. In particular, we describe how the situational data are collected in real applications, and we run experiments to explore the performance of the Situational Data Model when compared to the data model that uses only action sequences. We study with due diligence their accuracy, false discovery rates, false positive rates, and ROC curves. As we can see from the results, the Situational Data Model outperforms its comparator and can provide benefits in building more effective IDS. In the next chapter, we will investigate an alternative machine learning method other than HMM to build IDS, on top of our Situational Data Model.

(a) HMM using Action Sequence Only



(b) HMM using Situ Sequence

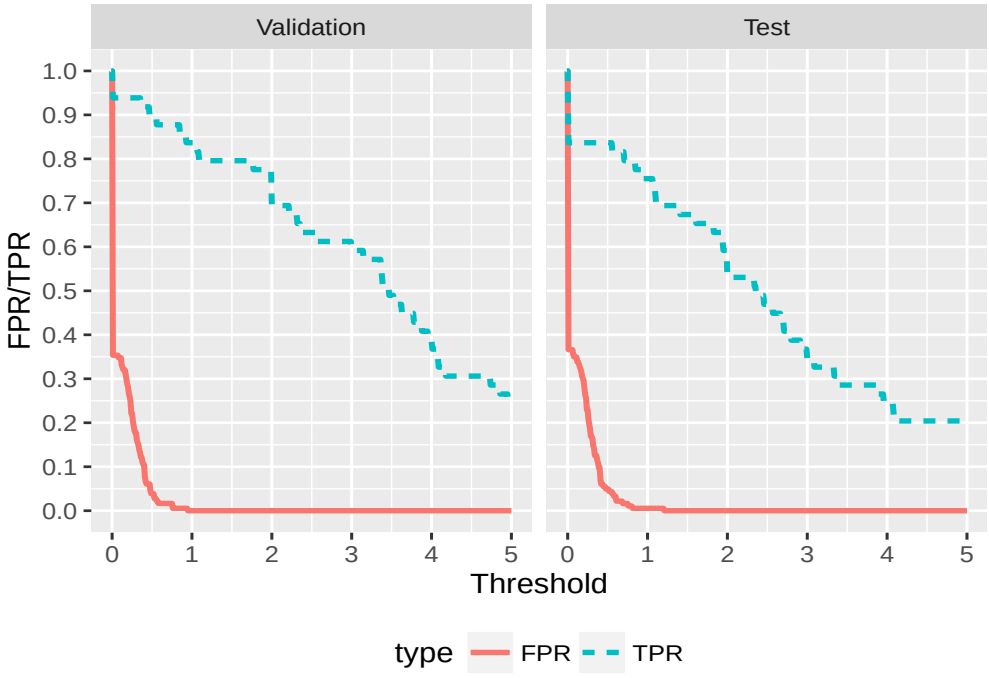


Figure 3.10: False Positive Rates and True Positive Rates of HMMs

CHAPTER 4. SITUATION AWARE INTRUSION DETECTION SYSTEM USING CONDITIONAL RANDOM FIELDS

Abstract

In this chapter, we introduce Situation Aware with Conditional Random Field based Intrusion Detection System (SA-CRF-IDS), where the intrusion analysis is performed by probabilistic graphical models of Conditional Random Fields (CRF) over a sequence of observed situations. Compared to other similar models, such as Hidden Markov Models (HMM), used in Intrusion Detection, CRF can represent intrusion problems more comprehensively. In an HMM, a Markovian assumption is adopted such that any hidden state only conditionally depends on the state from its immediate neighbor at the previous position or time point. This assumption is effectively relaxed in CRF models, where each hidden state can depend on all its neighbour hidden states and observations at any of the previous time points. This gives CRFs more freedom to detect crafty attackers that often hide their footprint by inserting lots of noisy information when attacking. We compare the CRF and the HMM model on both the sequences of actions data set and sequences of situation data set. From our experiment result, the CRF is more effective than HMM for the action-sequence data set. On the other hand, those two models are very competitive and comparable in detecting intrusion over the situational data set.

Key Words: Intrusion detection system, Conditional Random Field, Hidden Markov Model, Detection accuracy, ROC.

4.1 Introduction

This article introduces a paradigm for building Situation Aware with Conditional Random Field based Intrusion Detection System (SA-CRF-IDS). The SA-CRF-IDS is used for classifying malicious intention and normal intention by analyzing collected situational data such as environmental context information, activities, and inferred desires.

Many machine learning approaches have been routinely utilized in intrusion detection. For instance, Jonathon and Deepti [44] run data mining tools against a log file to detect intrusion. Chen et al. [14] used Rough Set Theory to select features and applied the Support Vector Machine method to learn and test based on selected features. Meanwhile, Feshki et al. [19] also utilized Support Vector Machine as the core component of their proposed system to classify alerts. In addition, Wee et al. [63] proposed a Bayesian network for the modeling of network intrusion domain. They also applied the powerful reasoning capabilities of Bayesian network to detect intrusions. Moreover, Al-Jarrah and Arafat [1] applied principal component neural networks for recognizing attacks and tested their system in real practice and the DARPA [31] datasets.

Among all those machine learning methods, Hidden Markov Models (see Section 3.2.1) are very popular statistical tools for intrusion detection because they are powerful for finding the hidden and underlying structure of given sequential data. For example, in [27], an HMM was used for training the intrusion detection model based on system calls sequences. Ariu et al. [7] designed an intrusion detection system named HMMPayl to detect attacks against web applications through the analysis of the HTTP payload and Hidden Markov Model. They represented the payload as a sequence of bytes. On the other hand, the detection system proposed by Chen et al. [13] analyzed multiple logs from cloud to extract the intension of the actions. In their work, a Hidden Markov Model was adopted to model sequences of attacks performed by hackers and such stealthy events in a long time frame will become significant in the state-aware model. Moreover, Jain and Abouzakhar [28] analyzed the performance of a Hidden Markov Model and Support Vector Machine for anomaly intrusion detection based on

the publicly available KDD Cup 1999 dataset. For sequential data sets, HMMs can represent and model intrusion detection problems better than many of other static approaches.

However, HMMs suffer from several assumptions that may not be realistic in many applications. Firstly, an HMM adopts the Markov assumption that the next state conditionally depends only on the current state. This assumption may be unreasonable in intrusion detection problems when we represent the attacking process as sequences of states and observations. A sophisticated attacker can intentionally perform normal actions and malicious actions alternately in a staggered fashion to confuse detection. Therefore, in this case it may not be sufficient to just check the current state to label its succeeding state, because observations at any earlier time point may also be relevant. Secondly, an HMM assumes that transition probabilities are independent of actual time points when the transitions take place. Third, an HMM assumes that observations are statistically independent from each other conditional on hidden states. This assumption may not be pragmatic in intrusion detection problems where the features are often associated with each other.

To address the problems mentioned above, we utilize Conditional Random Fields (CRFs) to build intrusion detection systems. CRFs have been used for intrusion detection in the literature [23, 51, 9, 47, 56, 62, 52]. However, most of the existing approaches aim to model the dependency relationships among different features, similarly to aforementioned classical methods. That is, they did not take full advantage of CRFs to train on sequences of features in chronical order to build intrusion detection models. Plus, most of the researches only worked on the KDD Cup 1999 dataset or DARPA 2000 data set, which may not be representative of many other practical intrusion settings. Therefore, in this paper we want to seize the opportunity to represent each data record as a sequence of situations, and develop intrusion detection models that use CRFs to build IDS using situation sequences. With situational data, CRFs can help us detect malicious intentions based on the footprints of situations.

Our SA-CRF-IDS has two major components. The first component is the situational data (refer to Section 3.3) collected and parsed following the Situ Framework [11]. The second one is the classifier based on Conditional Random Fields that can represent intrusion detection problems more generally than other similar approaches. The working process of the proposed

SA-CRF-IDS is as follows: Firstly, a sequence of desires is predicted based on the sequence of actions and context. Secondly, each of those desires is assigned value, i.e., $tempI$, that reflects the feasibility of attack at specified time points. Finally, the SA-CRF-IDS classifies each desire sequence into normal intention or malicious intention. In order to compare a CRF and an HMM, we run both models on each of the action-only sequences data set and the situational data set. We then compare their performance by studying their false positive rates, true positive rates, and ROC curves in intrusion detection, similar to our discussions in Chapter 4.3.2.

The rest of this paper is organized as follows: Section 2 introduces Conditional Random Fields with more details. Section 3 describes the construction of SA-CRF-IDS. Section 4 illustrates how we conduct our experiment and show what our results are. Section 5 concludes with a summary and a discussion about some potential future work.

4.2 Conditional Random Fields

As aforementioned, we plan to build our intrusion detection system using CRFs. A CRF is an undirected graph $\mathcal{H}$, whose nodes correspond to a node set $X \cup Y$, where X represents a set of selected features and Y stands for dependent variables of interest. This graph or network is annotated as a Gibbs distribution with a set of factors $\phi_1(D_1), \ldots, \phi_T(D_T)$ such that each D_t is not a subset of X (i.e., D_t contains information from both X and Y), where the index t denotes the position of CRF sequence. The network encodes a conditional distribution as follows [34]:

$$\begin{aligned} P(Y \mid X) &= \frac{1}{Z(X)} p(Y, X), \\ p(Y, X) &= \prod_{t=1}^{T} \phi_t(D_t), \\ Z(X) &= \sum_{Y} p(Y, X), \end{aligned} \tag{4.1}$$

where $Z(X)$ is the normalizing constant, or partition function, that makes $P(Y \mid X)$ a valid probability function. Conditional Random Fields are probabilistic models for computing the probability $P(Y \mid X)$ of a possible output $Y = (y_1, \ldots, y_T)$ given an input $X = (x_1, \ldots, x_T)$, which sometimes is also called an observation sequence.

CRFs have been broadly applied in many research areas and topics. For example, Angrosh et al. [6] used CRFs to build a supervised learning mechanism for context identification and sentence classification. Joder et al. [29] introduced the use of CRFs for the audio-to-score alignment task. Zhang and Gong [65] proposed a method for action categorization with modified hidden conditional random fields. Shen et al. [50] proposed a sparse hidden dynamic CRF model for user intent learning from their search session. Among those researches and applications, a particular CRF that is frequently used across various fields is the linear chain conditional random field, which can be represented graphically by Figure 4.1 [61, 55, 32]. A linear chain

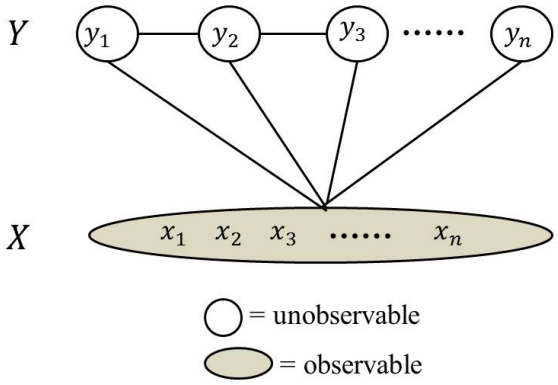


Figure 4.1: Linear Chain Conditional Random Fields

CRF often defines the following probability models

$$p(Y \mid X) = \frac{1}{Z(X)} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, X) \right\},$$
$$Z(X) = \sum_{Y} \prod_{t=1}^{T} \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, X) \right\}, \tag{4.2}$$

where X is a set of input variables that can be observed, Y is a set of output variables that needs to be predicted, the letter t denotes positions in observation sequence X and label sequence Y, and $f_1, \ldots, f_K$ denotes K feature functions defined either from domain knowledge or trained from the training dataset.

As a sequential model, a linear chain CRF has some similarities with an HMM, but also offers several advantages over the HMM. For illustration purpose, we display the graphical representative models of an HMM in Figure 4.2 (Note that the CRF is represented in Figure 4.1). First of all, a CRF allows rich and unconstrained feature representation that could overlap or refer arbitrary to the observation, because it does not model interdependence on the observations as a discriminative model. Secondly, the Markov assumption that a succeeding state is only dependent on the current state is relaxed in CRF. Thirdly, a CRF defines a conditional probability distribution that is not structured as a table, but rather induced by a small set of parameters $\theta_1, \ldots, \theta_K$, whose possible values are more general than just exponential positive values.

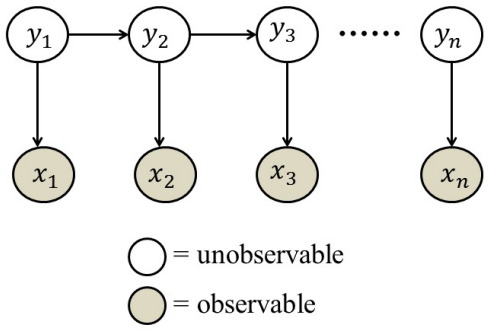


Figure 4.2: Hidden Markov Model

The goal in the training step of a CRF model is then to estimate the feature function weights $\theta_1, \ldots, \theta_K$ based on the data. In this training step, it aims at finding a sequence of weights such that the CRF model can represent the training dataset as accurately and likely as possible. The parameter estimation is usually performed by the penalized maximum likelihood method [55]. Once the parameters estimates are obtained and the model is built, for any observed feature value X^*, we can apply this model to obtain the conditional probability $P(Y \mid X^*)$ for all possible label sequence Y. Following this, we can then infer the label sequence as Y^* that

has the highest probability to occur, i.e., $Y^* = \arg\max_Y P(Y \mid X^*)$. The Forward-Backward and the Viterbi Algorithm [32], which are based on sending message along the chain in the only two possible directions, can be applied for the inference. We explain the details of parameter estimation and outcome inference later in Section 4.3 when we introduce the construction and evaluation of our SA-CRF-IDS.

4.3 Situation Aware Intrusion Detection using Conditional Random Fields

4.3.1 Framework of SA-CRF-IDS

In this section, we describe how an SA-CRF-IDS is constructed. In particular, Figure 4.3 illustrates the framework of an SA-CRF-IDS, starting from data processing to classifier training and evaluation. This process is elaborated in the following steps:

Step 1. Parse collected raw data into situational data following the model introduced in Section 3.3. Specifically, we format the data to sequences of actions and environmental contexts, and sequences of desires.

Step 2. Then we build two CRF models, by first establishing a CRF model, say desire CRF, from sequences of actions and contexts information to desire sequences, followed by building a CRF model, say tempICRF, from desire information to temporal intention. For training data, the temporal intention is manually labeled according to actions, contexts and desires.

Step 3. The inference process of an SA-CRF-IDS involves two steps. In the first step, a desire sequence is predicted based on the desire CRF model using input of a sequence of actions and contexts. Next, use the predicted desire sequence as the input in the tempICRF model, and obtain a sequence of temporal intention.

Step 4. The last step is to classify the observed data into being intrusion or not. This is achieved by comparing the average value of all individual temporal intention values over time in the predicted temporal intention sequence to a threshold γ. The selection of the threshold γ will be covered with more details in Section 4.4.

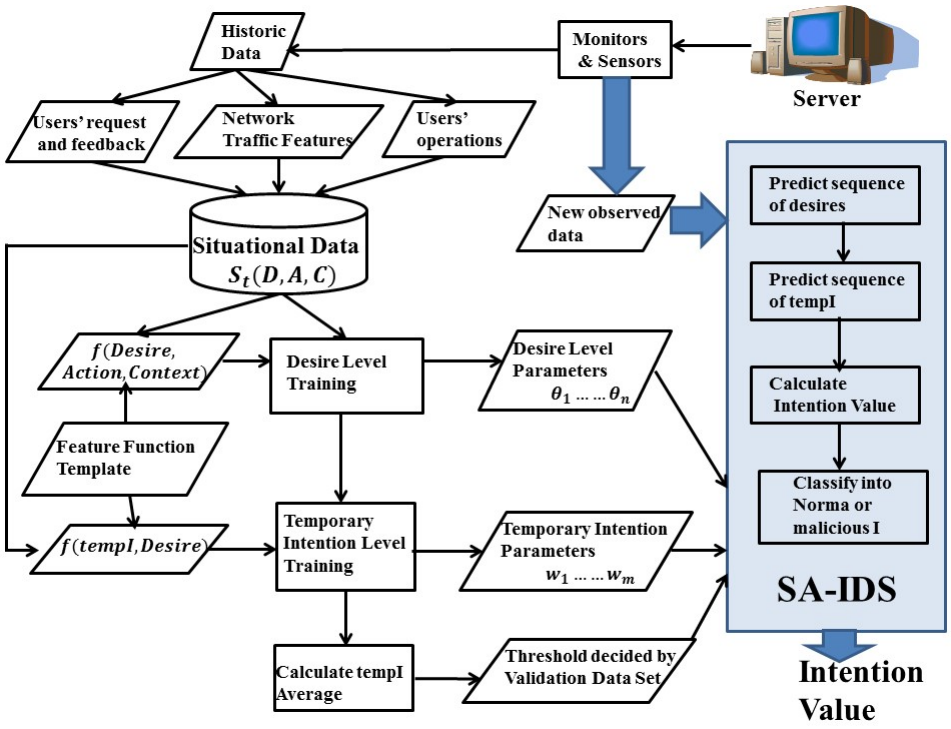


Figure 4.3: Flowchart illusrating the training and evaluation of SA-CRF-IDS

4.3.2 Parameters Training for SA-CRF-IDS

With the situational training dataset, we apply CRF to build the intrusion detection system. The process of building an IDS is essentially the process of parameter training. In our IDS, the parameter training includes $\theta_1, \dots, \theta_K$ training for desire CRF and $\omega_1, \dots, \omega_m$ training for tempICRF as shown in Figure 4.3. In this subsection, we discuss briefly the estimation of model parameters $\theta_1, \dots, \theta_K$ for the desire CRF model. The estimation of model parameters $\omega_1, \dots, \omega_m$ for the tempI CRF is similar. The parameters are usually trained by maximum likelihood estimation. Specifically, suppose the training dataset is $\mathcal{D} = \{(X_i, Y_i) \mid i = 1, \dots, N\}$, where X_i's are feature variables and Y_i's are outcomes of interest. The maximum likelihood principle is to find appropriate parameters such that the model can best represent the distribution of training dataset $\mathcal{D}$. When the training dataset contains N records, following equation (4.3), the logarithm of the joint conditional probabilities of $Y_1, \dots, Y_N$ given predictors

$X_1, \ldots, X_N$ can be written as

$$\begin{aligned} L(\theta, \mathcal{D}) &= \log \left(\prod_{i=1}^{N} P(Y_i \mid X_i, \theta) \right) \\ &= \sum_{i=1}^{N} \log \left[\frac{1}{Z(X_i)} \exp \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_k f_k(Y_{i,t-1}, Y_{i,t-1}, X_i) \right], \end{aligned} \tag{4.3}$$

where $Y_i = (Y_{i,1}, \ldots, Y_{i,T-1})$ is a sequence of response at time points $t = 1, \ldots, T$. Note that in the above model we did not write explicitly X_i and allow it to potentially contains all it components from different time points. Very often, for simplicity, we consider feature functions f_k such that $f_k(Y_t, Y_{t-1}, X) = f_k(Y_t, Y_{t-1}, X_t)$ for each k.

The parameters can be estimated by $\hat{\theta} = \arg\max_\theta L(\theta, \mathcal{D})$. In our discussion here, we consider exponential families such that the maximization of the the objective function is a convex optimization problem. More general CRF models are possible by adopting different parametric models, including varying feature functions f_k and probability mass functions beyond the exponential family.

Notice further that the above formula is generic. For instance, as we mentioned in the previous subsection, we have two CRF models, i.e., the desire CRF model that takes action and context sequences as X and desires sequences as Y, and the temporal intention tempICRF model that uses desire sequences as X and temporal intention sequences as Y. In desire model level of SA-CRF-IDS, we train the parameters $\theta_1, \ldots, \theta_K$ to maximize the corresponding objective function following (4.3). Similarly, in the temporal intention model level of SA-CRF-IDS, we train the parameters $\omega_1, \ldots, \omega_m$ to maximize the corresponding objective function following (4.4), where we allow potentially different feature functions g_j, $j = 1, \ldots, m$.

$$L(\omega, \mathcal{D}) = \sum_{i=1}^{N} \log \left[\frac{1}{Z(X_i)} \exp \sum_{t=1}^{T} \sum_{j=1}^{m} \omega_k g_j(Y_{i,t-1}, Y_{i,t-1}, X_i) \right]. \tag{4.4}$$

4.3.3 Inference of SA-CRF-IDS

Once parameter estimates $\hat{\theta}$ from the model in 4.3 and 4.4 are obtained, an SA-CRF-IDS model is built, and we can apply the SA-CRF-IDS for inference. In particular, for any new observed sequence X_{new}, the conditional probability of observing any potential response Y

given X_{new}, i.e., $P(Y \mid X^*, \hat{\theta})$, can be easily evaluated. Then the most probable sequence Y^* given X and based on the predicted model is

$$Y^* = \arg\max_{Y} P(Y|X_{new}; \hat{\theta}). \tag{4.5}$$

Note that the above step of finding the maximizer may not be as trivial as it looks like. The Viterbi algorithm is frequently used for finding the most probable sequence(s) [55].

As we described in Section 4.3.1, our SA-CRF-IDS involves two CRF models. Therefore, in the first step, we apply the desire CRF model and obtain the most probable desire sequence $desire^*$ based on the sequence of actions and contexts, i.e., $X_{new} = \{A_{new}, C_{new}\}$. Then using the temporal intention CRF model, we get the most probable temporal intention sequence $tempI^*$ given $desire^*$.

Based on the predicted temporal intention sequence $tempI^*$, we can calculate the average of $tempI^*$, say $avg.tempI^*$, over its components corresponding to temporal intention values at different time points. We then classify the whole sequence of situations to be normal or malicious by comparing $avg.tempI^*$ to a threshold γ. When $avg.tempI^*$ is equal or greater than γ, we classify it as intrusion, otherwise is normal.

4.4 Experiment Design and Evaluation

To validate our hypothesis of advantages, see, e.g., Section 4.1 and Section 4.2, of using CRFs in Intrusion Detection over HMMs, we conduct CRF-based experiments to compare the effectiveness of a CRF and an HMM when we apply them in the same data set. In this section, we describe details of our experiment design and the result.

4.4.1 Experiment Design

Here we conduct two experiments to compare a CRF and an HMM. In the first experiment, we compare them in the data set, say A, that includes only the action sequences. Then in the second experiment, we compare them in the Situational data set, say B, (see Section 3.3 for details). In other words, we will study the following four model+dataset combinations:

(1) the HMM model on data set consisting of only action sequences, i.e., A;

(2) the HMM model on the Situational data set, i.e., B;

(3) the CRF model on data set consisting of only action sequences, i.e., A;

(4) the CRF model on the Situational data set, i.e., B;

In Chapter 3, we have comprehensively discussed and compared (1) and (2). In this section, we focus mainly on (3) and (4) and the comparisons between (1) and (3), as well as (2) and (4). For each of those four experiments, we partition the dataset into a training set and a testing set first, then generate the validation set independently. Details on how we select them and the proportion of training, testing sets can be found in Section 3.4.1.

For implementation of CRF, we apply the open-source program CRF++ [40] to build our SA-CRF-IDS. As can be seen in the previous section, one important component of building a CRF model is the specification of feature functions $f_1, \ldots, f_K$ to be used in (4.3). In the CRF++ tool, this is achieved by providing a feature template file. In our experiments, we adopt the default feature template of CRF++, which is then automatically translated by CRF++ into feature functions for the CRF model. In terms of the selection of X and Y in (4.3), when we run CRF on the first data set, A, we use action sequences as X and temporal intention sequences as Y. When it comes to building our SA-CRF-IDS on the Situational data set, B, we first select action and context sequences as X and desire sequences as Y to build the desire CRF, then we use desire sequences as X and temporal intention as Y to build the tempI CRF.

In the previous section, we have discussed main procedure of the SA-CRF-IDS, e.g., the involvement of two CRF models. Here, we illustrate in details both experiments (3) and (4) in Figure 4.4 and Figure 4.5, respectively. Each run of CRF++ involves two processes, i.e., a training process and a testing process. In the training process, CRF++ takes a training data set and outputs a training model file. This training model file is then applied by CRF++ to the testing data set in the testing process. Take experiment (3) in Figure 4.4 for example, in steps ① and ②, CRF++ toolkit first trains a tempI CRF model based on the training data set consisting of action only sequences as X and temporal intention sequences as Y. Then using the trained model in steps ③ and ④, the CRF++ predicts temporal intention sequences for the test data set.

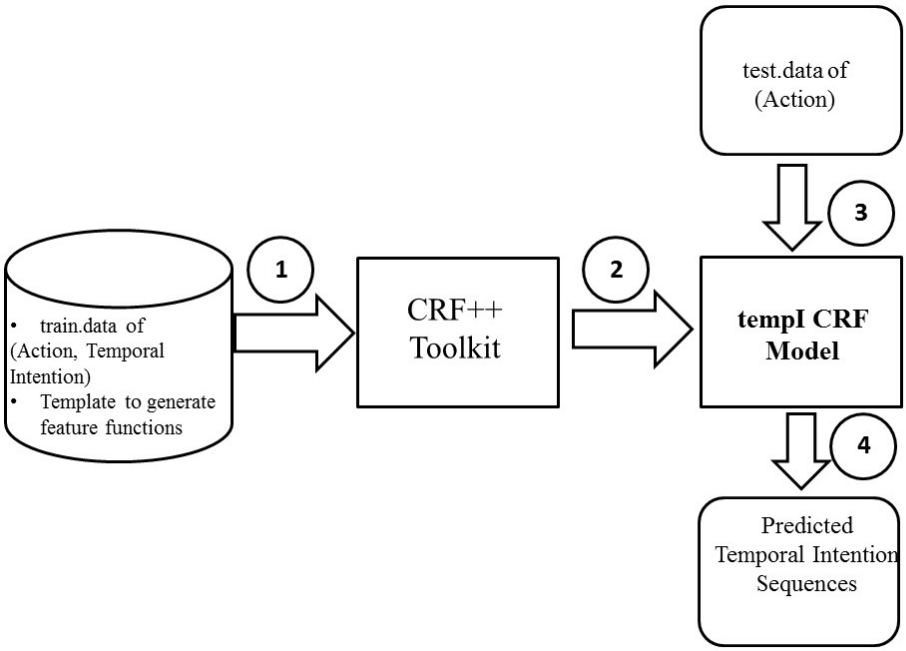


Figure 4.4: Process of CRF in Action-Only Sequence Data Set

For experiment (4), there are two CRF models as show in Figure 4.5. Specifically, first of all, the CRF++ toolkit trains a desire CRF model based on (Action, Context, Desire) components from the Situational training data set in steps ① and ②, and trains a tempI CRF model based on training data consisting of desire and temporal intention sequences in steps ⑤ and ⑥. Then in the testing step, the desire CRF model is first applied in steps ③ and ④ to predict a desire sequence, which is later used by the established tempI CRF model to predict the tempI sequence in steps ⑦ and ⑧.

The outputs from CRF++ include all information from the original testing data set, and additionally the most probable outcome Y^*, and the associated predicted conditional probabilities $p(Y^* \mid X_{new}; \hat{\theta})$. For instance, consider experiment (3) where we apply a CRF model on the data set A, the following sample output in Table 4.1 gives predicted outcome for one action sequence in the test data set. The first two columns include (Action, tempI) pairs in one sequence from the test data set, and the third column is the predicted most probable tempI sequence with associated probability value at each time point shown in the last column.

Table 4.1: Sample output from testing step by applying CRF++ on action-only sequences

Action	*tempI*	*tempI**	Prob
clickSkipPreQuestion	0	0	0.995
clickMenuHome	0	0	0.999
clickMenuSignin	0	0	0.993
clickLogin	1	1	0.882
clickLogin	4	2	0.772
clickLogin	7	4	0.755
clickLogin	10	7	0.670
clickMenuAllPapers	0	0	0.794
clickPaperInfosAllPapers	0	0	0.834
clickMenuAllPapers	1	0	0.854
clickRemoveCommentMyPapers	2	1	0.684
clickOkRemoveComment	4	2	0.648
clickMenuListUsers	0	0	0.510
clickLinkModify	1	1	0.862
clickModify	2	2	0.867
clickLinkModify	4	4	0.722
clickModify	7	7	0.665
clickMenuListUsers	0	0	0.802
clickLinkAddNewAccount	1	1	0.744
clickAddNewAccount	2	2	0.719
clickMenuAllPapers	0	0	0.649
clickRemoveCommentMyPapers	1	1	0.729
clickOkRemoveComment	2	2	0.704
clickEndSession	0	0	0.909

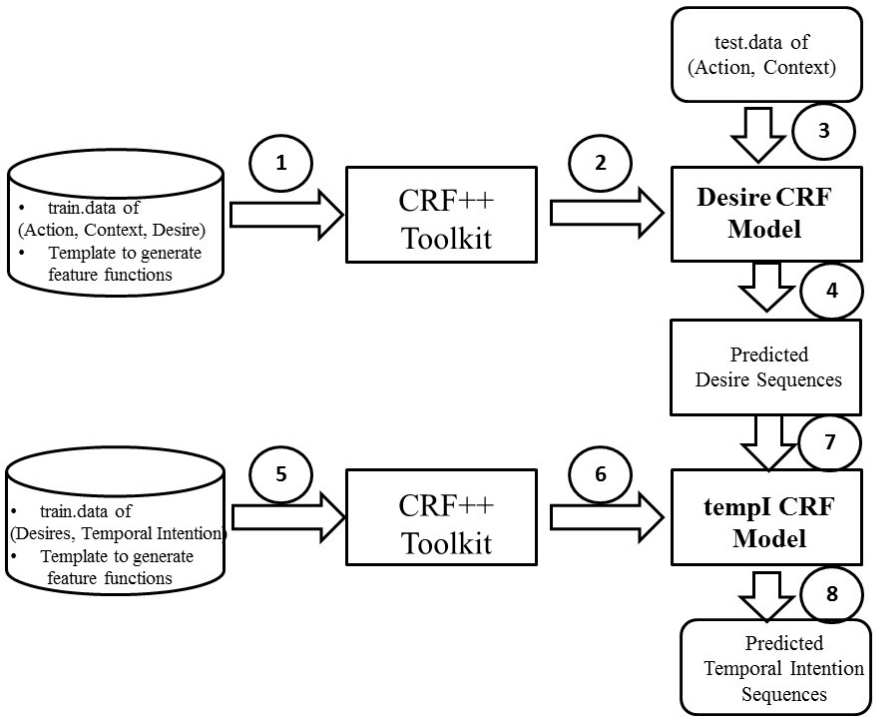


Figure 4.5: Process of CRF in Situational Data Set

For experiment (4), as aforementioned, it includes two CRF models, corresponding to the building of desire CRF and tempI CRF models. We exhibit part of the outputs corresponding to step ④ and step ⑧ of Figure 4.5 in Table 4.2 and Table 4.3, respectively. As can be seen in those two tables, in the output of the desire CRF model, sequences of action and context are taken as X, and desire as Y. Then the most probable desire sequence $desire^*$ is predicted with associated probabilities at each time point displayed in the last column. In the output of the tempI CRF model, the predicted desire sequences, i.e., $Desire^*$ are then used as X and tempI sequences are used as Y, leading to predicted $tempI^*$ sequences. As shown in Table 4.3, the predicted desire sequence $Desire^*$ in Table 4.2 is used as an input.

For inferences, as briefly covered in Section 4.3.3, we average each predicted Y^* sequence, which in our case is a $tempI^*$ sequence, to obtain $avg.tempI^*$. The $avg.tempI^*$ values are then compared to some threshold value γ and sequences with $avg.tempI^*$ larger than γ are classified as intrusions. In the upcoming section, we discuss the results of our experiments in details.

Table 4.2: Sample output from testing step of desire CRF by applying CRF++ on the Situational data set

Action	Context	
clickSkipPreQuestion	NoInput	NoError
clickMenuSignin	NoInput	NoError
clickSignup	NoInput	NoError
clickSignup	SqlInput	ErrorRemind
clickSignup	SqlInput	ErrorRemind
clickSignup	SqlInput	ErrorRemind
clickSignup	SqlInput	ErrorRemind
clickSignup	SqlInput	NoError
clickMenuSignin	NoInput	NoError
clickLogin	SqlInput	NoError
clickMenuMyPapers	NoInput	NoError
clickMenuUploadPaper	NoInput	NoError
clickMenuMyComments	NoInput	NoError
clickMenuAllPapers	NoInput	NoError
clickSubmitCommentAllPapers	InsertedLink	NoError
clickSubmitCommentAllPapers	InsertedLink	NoError

Desire	$Desire^*$	Prob
Test	Test	1.00
LogIn	LogIn	0.85
SignUp	SignUp	0.79
RegisterIllegalUser	RegisterIllegalUser	0.97
RegisterIllegalUser	RegisterIllegalUser	0.99
RegisterIllegalUser	RegisterIllegalUser	1.00
RegisterIllegalUser	RegisterIllegalUser	0.99
RegisterIllegalUser	RegisterIllegalUser	0.95
LogIn	LogIn	0.96
LogIn	LogIn	0.52
UploadPaper	ViewMyPapers	0.93
UploadPaper	UploadPaper	0.71
EditComment	ViewMyCommentInfo	0.45
EditComment	ViewAllPapers	0.75
EditComment	SubmitComment	0.98
EditComment	SubmitComment	0.97

Table 4.3: Sample output from testing step of tempI CRF by applying CRF++ on the Situational data set

*Desire**	tempI	*tempI**	Prob
Test	0	0	0.999
LogIn	0	0	0.998
SignUp	0	0	0.966
RegisterIllegalUser	1	1	0.914
RegisterIllegalUser	2	2	0.890
RegisterIllegalUser	4	4	0.830
RegisterIllegalUser	7	7	0.770
RegisterIllegalUser	10	10	0.892
LogIn	0	0	0.971
LogIn	0	0	0.996
ViewMyPapers	0	0	0.998
UploadPaper	0	0	0.996
ViewMyCommentInfo	1	0	0.991
ViewAllPapers	2	0	0.998
SubmitComment	4	1	0.966
SubmitComment	7	2	0.920

4.4.2 Experiment Evaluation

In this subsection, we compare CRF intrusion detection methods to HMM detection methods. Similar to the definition of HMM_i, $i = 1, 2$ in Chapter 3, we can define CRF_i, $i = 1, 2$ as the following:

CRF_1 : CRF method making use of action sequences only;

CRF_2 : CRF method making use of Situ sequences,

corresponding to experiment (3) and (4) (introduced in Section 4.4.1) respectively. Our focus here is to compare the performance of HMM_i and CRF_i for each $i = 1, 2$. That is, we want to compare the experiment results of (1) with (3), and (2) with (4). Again, we study their accuracy, false positive rate, and true positive rate under different threshold values, where ACC, FPR, TPR are defined in (3.2) - (3.4).

Firstly, we display the accuracy and ROC curve of CRF_1 in Figure 4.7. For ease of reference and comparison, we also display the accuracy and ROC curve of HMM_1 in the same figure. Those figures contain results for both the validation data set and test data set. For both HMM_1

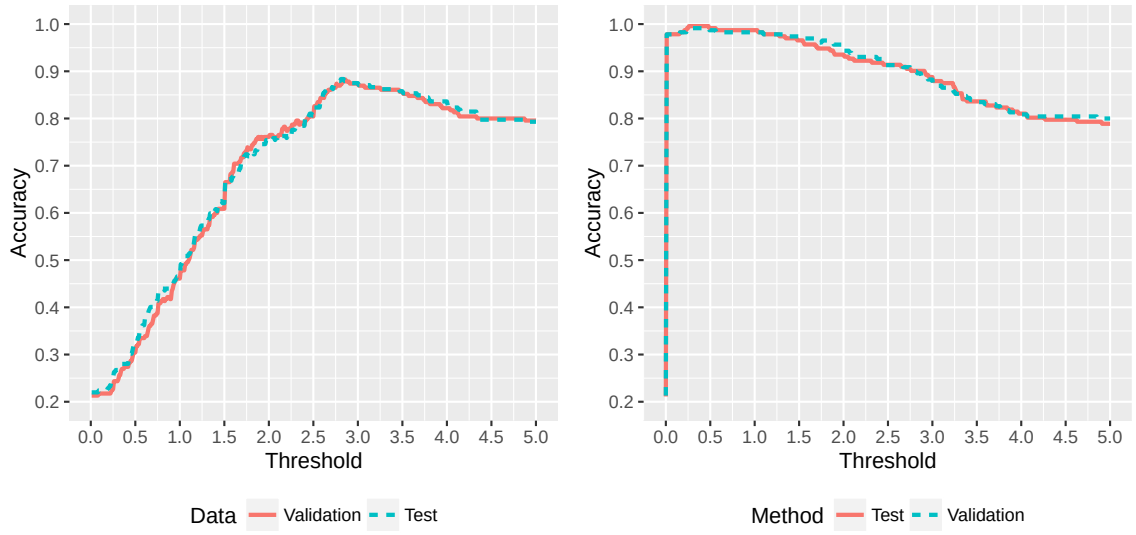
and CRF_1, we can see that validation and test data have very similar accuracy profile as their corresponding lines almost overlap with each other, except minor difference in the ROC curve for HMM_1. In terms of threshold selection guided by accuracy, a practical threshold for CRF_1 can be anywhere from 0.25 to 1 based on Figure 4.6b, leading to approximately $98\% \sim 99\%$ accuracy. On the other hand, based on Figure 4.6a, the threshold for HMM_1 can be selected as 2.75, giving about 85% accuracy. Thus, the accuracy of CRF_1 is significantly better than that of HMM_1, especially in terms of the maximum accuracy. In addition, as reflected in Figures 4.6c, 4.6d, the ROC curve of CRF_1 also clearly outperforms that of HMM_1.

Secondly, we compare the false positive and true positive rates of HMM_1 and CRF_1. We display their FPR and TPR profiles in Figure 4.9. Again, the overall trend for the validation and test data set behaves very similarly. If we compare Figure 4.8a and Figure 4.8b, we observe that the FPR for CRF_1 decreases very quickly towards 0 when the threshold value increases from 0. And for a threshold value that is slightly larger than 0 but no larger than 1, the FPR for CRF_1 is controlled under 5% and the associated TPR is maintained as high as 95%. However, the FPR for HMM_1 decays much slower than CRF_1 when the threshold increases. In particular, when the threshold is increased to a level, e.g., 2.75, such that FPR for HMM_1 is controlled under 5%, the corresponding TPR for HMM_1 is barely over 50%. The more dramatic difference (compared to accuracy) between TPRs of HMM_1 and CRF_1 when fixing FPR is understandable. This is because, as can be seen from (3.5), the accuracy is a weighted average of FPR and TPR, where the weights on TPR, i.e., $1-n/N$ (the proportion of malicious sequences) is roughly 20% in our case (see Table 3.1). Therefore, a roughly 50% difference in TPRs between HMM_1 and CRF_1 while with FPRs controlled under 5% for both of them, translates into approximately $20\% \times 50\% = 10\%$ difference in accuracy, which is consistent with the findings from Figure 4.6a and Figure 4.6b.

Now let us compare results from experiment (2) and (4) corresponding to HMM_2 and CRF_2. Similarly, we can look at accuracy, ROC curve, false positive rate, and true positive rate. We display their accuracy and ROC curves in Figure 4.11. First of all, the lines for validation data set and test data set align better for CRF_2, as visible difference can be seen between the curves of validation and test datasets for HMM_1. In terms of maximum accuracy values, CRF_2

(a) Accuracy for HMM_1

(b) Accuracy for CRF_1



(c) ROC Curve for HMM_1

(d) ROC Curve for CRF_1

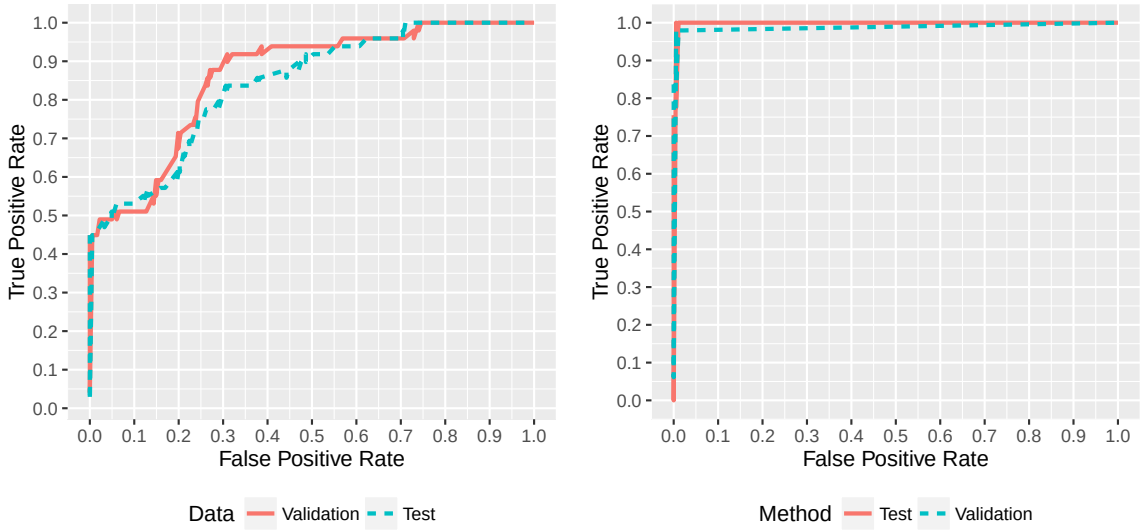
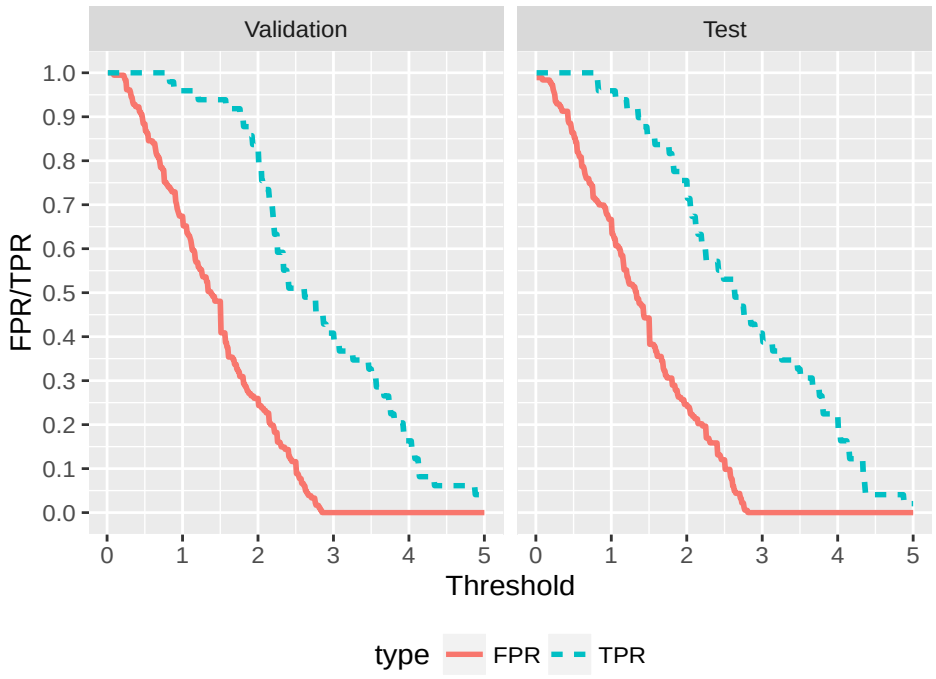


Figure 4.7: Accuracy and ROC curves of HMM_1 and CRF_1

(a) FPR and TPR of HMM_1



(b) FPR and TPR of CRF_1

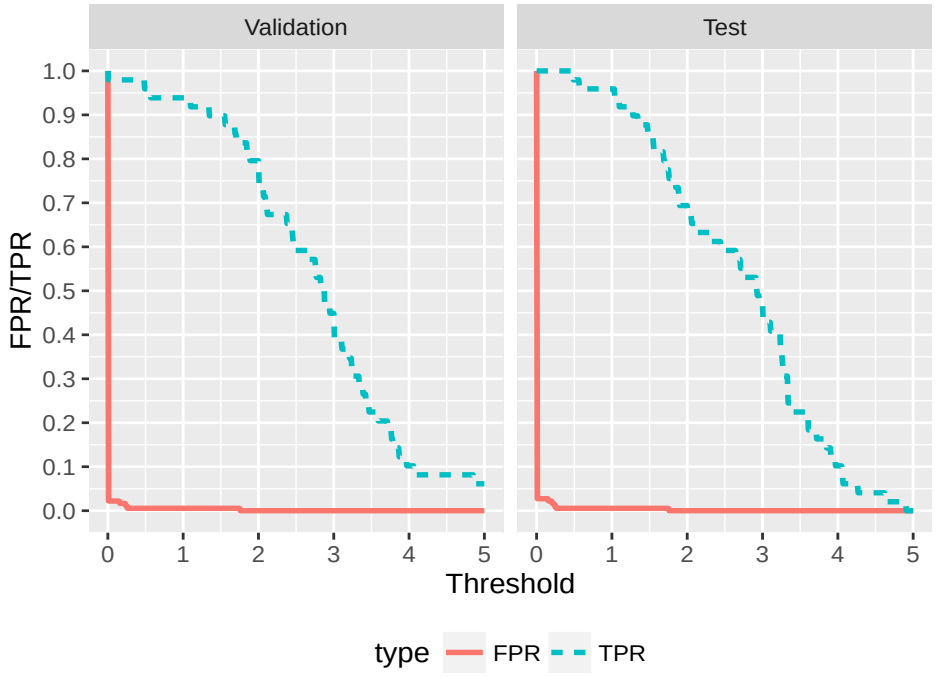
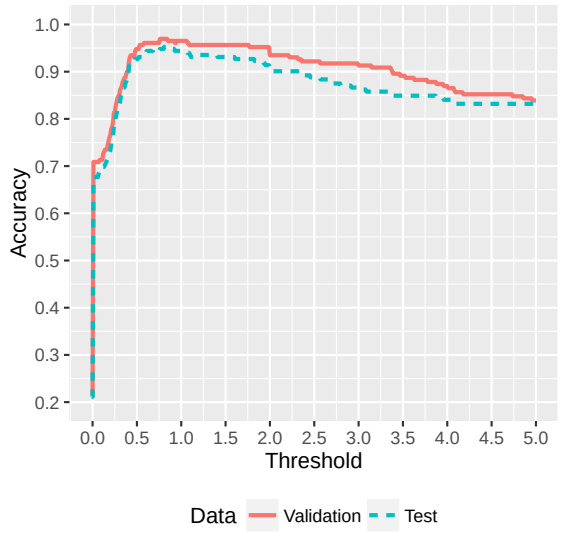


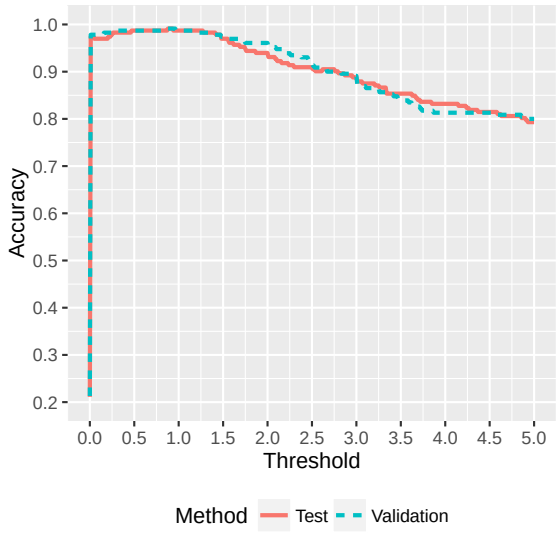
Figure 4.9: False Positive Rates and True Positive Rates of HMM_1 and CRF_1

has slightly larger maximum accuracy values (about 98% range)than that of HMM_2 (about 95% range). In addition, their overall trends with respect to the threshold value are similar, with CRF_2 climbing to the top sooner than HMM_2 when threshold increases from 0. On the other hand, the ROC curves in Figures 4.10c, 4.10d show that CRF_2 has better ROC curve performance than that of HMM_1.

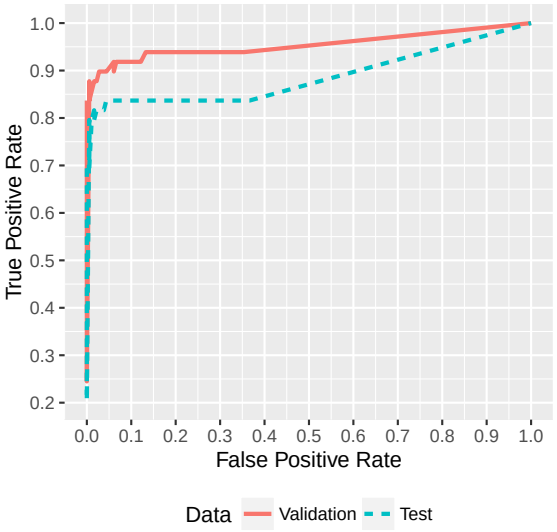
(a) Accuracy for HMM_2



(b) Accuracy for CRF_2



(c) ROC Curve for HMM_2



(d) ROC Curve for CRF_2

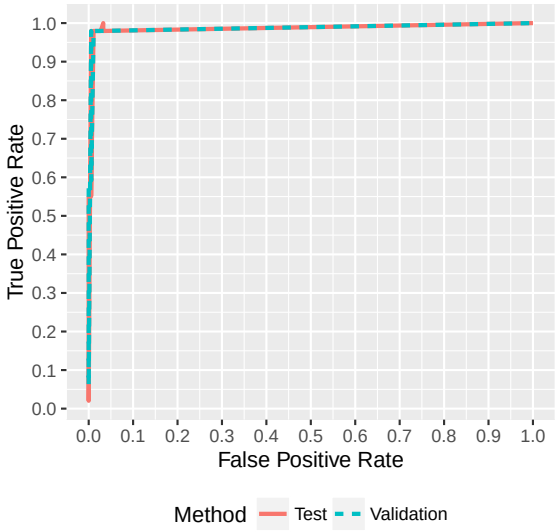


Figure 4.11: Accuracy and ROC curves of HMM2 and CRF2

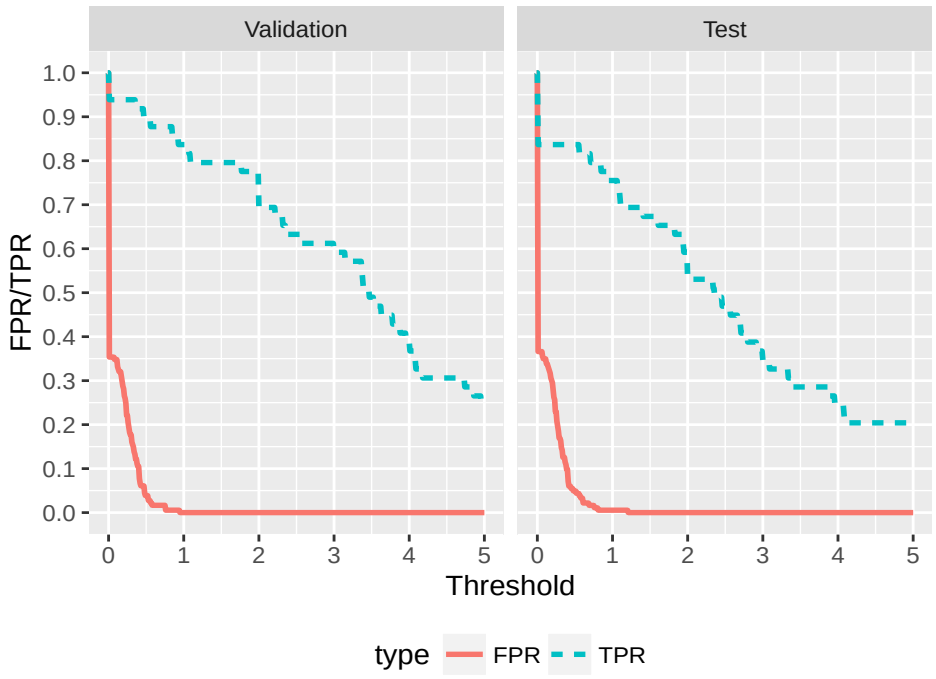
Lastly, if we examine the false positive rates and true positive rates displayed in Figure 4.13. For HMM_2, the TPR is in $85\% \sim 95\%$ range when the FPR is controlled under 5%. However, the TPR for CRF_2 is about 98% when its FPR is controlled under 5%.

In summary, from the above discussion, we can see all of HMM_2, CRF_1 and CRF_2 perform better than HMM_1 in our case study. That is, CRF models are more efficient than the HMM model using only action sequences. On the other hand, CRF_2 has slightly better performance than and HMM_2, and similar performance as CRF_1.

4.5 Conclusions

CRF models are more flexible than HMM models, and can be more adaptive in real applications. In our experiments, we found that CRFs can lead to more efficient intrusion detection systems than HMMs. On the other hand, an alternative approach to improve efficiency of an intrusion detection system is to build the system on more informative data. In our discussion here, we introduce the Situational data model, which provides more information than action only sequences. Therefore, our experiments also demonstrate that an IDS making use of more informative data can be more efficient, as can be seen in the advantage of HMM_2 over HMM_1. In fact, we see that by adopting either a CRF model or a Situational data model, it gives more efficient IDS than using HMM on action-only sequences. In addition, when HMM and CRF are applied on the Situational data set, CRF_2 performs slightly better and is also more robust than HMM_2, while both models have relatively satisfactory performance. For CRF_1 and CRF_2, their performance are very similar. However, we should acknowledge that any experiment on a single data set may be limited by the nature of the data itself. In our study, it may be that when both methods, i.e., CRF_1 and CRF_2, have very high accuracy, it is not possible to distinguish them over the current data. To explore in this direction further, a comparison between them on a different and larger situational data may be warranted.

(a) FPR and TPR of HMM_2



(b) FPR and TPR of CRF_2

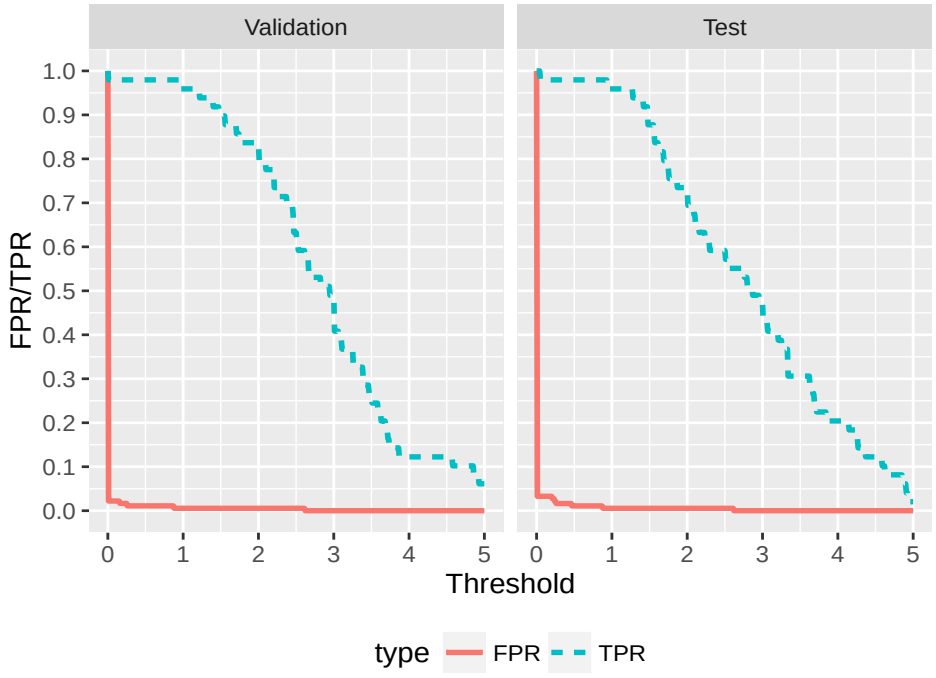


Figure 4.13: False Positive Rates and True Positive Rates of HMM_2 and CRF_2

CHAPTER 5. GENERAL CONCLUSION

This thesis contributes to improving the effectiveness of Intrusion Detection Systems built by probabilistic graphical models from following three different perspectives:

1. Firstly, we propose to apply the Bayesian Model Averaging of Bayesian Network Classifiers (BNMA) to build Intrusion Detection Systems. We compare the BNMA with Naive Bayes and Bayesian Network by using them to train IDS in the KDD data set. The result of our experiment shows that the BNMA is more efficient and more reliable than the other two models based on static data without time stamps, especially when only a small size of training data set is available.

2. Secondly, we build Situational Data Model as a mechanism for collecting dynamic sequential data set to train Intrusion Detection Systems. This Situational Data Model motivated by finding and utilizing the relationship between intrusion occurrences and the features' change over time. It is more informative by revealing footprints that a user left on the target system. We structure the Situational Data Model in time order and also add the human internal mental states, i.e., desires, to it. We define the quantitative measure (temporal intention) at each time point that describes the likelihood of being an intrusion at this specific time. Based on the Situational Data Model we designed, a Situational Data Set was collected on a real web application system named "CoRE" for training and evaluating Intrusion Detection Systems. We then applied Hidden Markov Models on both of the Situational sequences data set and action-only sequences data set to compare these two different data models. From our experiment results, we observe that our Situational Data Set is more effective than the action-only sequences data set.

3. Thirdly, based on the foundation of the dynamic sequential data (Situational Data Set and Action Only sequence data set) collected from the CoRE system, we propose to apply Conditional Random Fields for building Intrusion Detection Systems. Theoretically, CRFs are able to utilize information from the sequential data more flexibly and thoroughly. In addition, it is more representative and appropriate for intrusion detection than other dynamical training models such as HMMs. To test our hypothesis, we conducted experiments of applying CRF and HMM in action-only sequences data set and also in the Situational Data Set. From the experiment results, we see that the CRF outperforms the HMM significantly when we apply them in the actions only sequences. On the other hand, when it comes to the Situational Data Set, their performance are competitive with the CRF edging HMM slightly.

In summary, Bayesian Model Averaging of Bayesian Network Classifiers can benefit training Intrusion Detection Systems, especially for static data set of relatively small size. To detect intrusions before they eventually cause damage to the system, which typically leads to obvious abnormal changes for the static environmental context, we should build dynamical intrusion detection systems. Our Situational Data Model combined with Conditional Random Fields could be very valuable for building more effective dynamic intrusion detection systems, as evidenced from our experiment results. There are, however, still several limitations in this thesis:

1. The malicious data sets we collect from CoRE application are generated from assumed intruders but not real professional intruders. Our research could be more precise if we are able to get real malicious data from some real-life web applications to train our Intrusion Detection System. This limitation might also help explain that the effectiveness of HMM is comparable with CRF when both are applied in our Situational Data Set. Another possible reason for this phenomenon could be that the Situational Data Set has helped improve the effectiveness to a substantial degree that there is not much room left for further improvement in our Situational Data Set. Professional attackers could hide their malicious desires better in the observed situational sequences. In that case, the

advantages of CRF may be more obvious and we may be able to see that it outperforms HMM obviously. However, it is not easy to collect those sensitive data for research use due to credential concerns. Based on this limitation, we have tried our best to make our experiments as close as to real practice. We collected our data from a real web application CoRE, which provides a lot of functionalities that are frequently used in many other real web applications. Also, all the assumed intruders participate in our experiments are from Computer Science or related background and they are also trained with attacking skills before the experiments. Thus our assumed intruders can apply different attacking skills in their own way to attack the system and they closely mimic many kinds of attacking behaviors used by real attackers. Therefore, we believe that our research result is very convincing.

2. We have used supervised learning to train the Intrusion Detection Model. If we apply this method in real practice, we need to manually label both the desires and temporal intentions for our training data set. On the other hand, we don't need to label those for training data set when it comes to unsupervised learning, which usually performs clustering analysis on the training data set first, and then do the classification. However, unsupervised learning for intrusion detection usually has a very strong assumption that the amount of normal data is always much more than that of malicious data in the training data set. This assumption may not always be true in practice, thus it limits the application scope of the IDS in the real world. With our supervised learning methodology proposed in this thesis, we need the manual work for preparing and formatting the historic training data set. Nevertheless, we believe that our research is valuable since both the model training and classification process are automatic and can run in real time.

Our future work can focus on two aspects. The first aspect is to optimize the Situational Data Model. Currently, we collect the actions and environmental context information. All those collected information is only related to functional features. In the future, we may consider collecting some non-functional features in our sequences of situations to train Intrusion Detection System, such as throughput of the network, response time, etc. Also, we can work on

collecting another data set based on optimized Situational Data Model, and test the reliability of our methodology in data sets collected in other applications or platforms. The other aspect is to conduct some research on how to use unsupervised learning on this issue without the assumption that the amount of malicious data is always less than the normal data.

REFERENCES

[1] Al-Jarrah, O. and Arafat, A. (2015). Network intrusion detection system using neural network classification of attack behavior. *Journal of Advances in Information Technology Vol*, 6(1).

[2] Altwaijry, H. (2013). Bayesian based intrusion detection system. In *IAENG Transactions on Engineering Technologies*, pages 29–44. Springer.

[3] Altwaijry, H. and Algarny, S. (2012). Bayesian based intrusion detection system. *Journal of King Saud University - Computer and Information Sciences*, 24(1):1–6.

[4] Amor, N. B., Benferhat, S., and Elouedi, Z. (2004). Naive bayes vs decision trees in intrusion detection systems. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 420–424. ACM.

[5] An, X., Jutla, D. N., and Cercone, N. (2006). Privacy intrusion detection using dynamic Bayesian networks. In *International Conference on Electronic Commerce*, pages 208–215.

[6] Angrosh, M., Cranefield, S., and Stanger, N. (2010). Context identification of sentences in related work sections using a conditional random field: towards intelligent digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 293–302. ACM.

[7] Ariu, D., Tronci, R., and Giacinto, G. (2011). Hmmpayl: An intrusion detection system based on hidden markov models. *computers & security*, 30(4):221–241.

[8] Bahl, S. and Sharma, S. K. (2016). A minimal subset of features using correlation feature selection model for intrusion detection system. In *Proceedings of the Second International Conference on Computer and Communication Technologies*, pages 337–346. Springer.

[9] Bande, V. and Prasan, U. (2011). Robust intrusion detection system using layered approach with conditional random fields. 1.

[10] Catania, C. A. and Garino, C. G. (2012). Automatic network intrusion detection: Current techniques and open issues. *Computers & Electrical Engineering*, 38(5):1062–1072.

[11] Chang, C. K., Jiang, H.-y., Ming, H., and Oyama, K. (2009). Situ: A situation-theoretic approach to context-aware service evolution. *Services Computing, IEEE Transactions on*, 2(3):261–275.

[12] Chebrolu, S., Abraham, A., and Thomas, J. P. (2005). Feature deduction and ensemble design of intrusion detection systems. *Computers & Security*, 24(4):295–307.

[13] Chen, C.-M., Guan, D., Huang, Y.-Z., and Ou, Y.-H. (2012). Attack sequence detection in cloud using hidden markov model. In *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on*, pages 100–103. IEEE.

[14] Chen, R.-C., Cheng, K.-F., and Hsieh, C.-F. (2010). Using rough set and support vector machine for network intrusion detection. *arXiv preprint arXiv:1004.0567*.

[15] Chickering, D. M., Geiger, D., and Heckerman, D. (1995). Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128, Ft. Lauderdale, FL. Society for Artificial Intelligence and Statistics.

[16] Dash, D. and Cooper, G. F. (2004). Model averaging for prediction with discrete bayesian networks. *The Journal of Machine Learning Research*, 5:1177–1203.

[17] Devarakonda, N., Pamidi, S., Kumari, V. V., and Govardhan, A. (2012). Intrusion detection system using bayesian network and hidden markov model. *Procedia Technology*, 4(0):506–514.

[18] Eaton, D. and Murphy, K. (2007). Bayesian structure learning using dynamic programming and mcmc. In *Proceedings of Conference on Uncertainty in Artificial Intelligence*.

[19] Feshki, M. G., Sojoodi, O., and Anvary, M. D. (2015). Managing intrusion detection alerts using support vector machines. *no*, 9:266–273.

[20] Francois, J.-M. (2010). Jahmm: An implementation of hidden markov models in java.

[21] Ghahramani, Z. (2001). An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42.

[22] Grossman, J. (2007). *XSS Attacks: Cross-site scripting exploits and defense*. Syngress.

[23] Gupta, K. K., Nath, B., and Kotagiri, R. (2010). Layered approach using conditional random fields for intrusion detection. *Dependable and Secure Computing, IEEE Transactions on*, 7(1):35–49.

[24] Heckerman, D. (1996). A tutorial on learning with bayesian networks. Technical report, Microsoft Research Advanced Technology Division.

[25] Heckerman, D., Geiger, D., and Chickering, D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243.

[26] Huijuan, L., Jianguo, C., and Wei, W. (2008). Two stratum bayesian network based anomaly detection model for intrusion detection system. In *Proceedings of the 2008 International Symposium on Electronic Commerce and Security*, ISECS '08, pages 482–487, Washington, DC, USA. IEEE Computer Society.

[27] Imran, M., Afzal, M. T., and Qadir, M. A. (2015). Using hidden markov model for dynamic malware analysis: First impressions. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*, pages 816–821. IEEE.

[28] Jain, R. and Abouzakhar, N. S. (2013). A comparative study of hidden markov model and support vector machine in anomaly intrusion detection. *Journal of Internet Technology and Secured Transactions*, 2.

[29] Joder, C., Essid, S., and Richard, G. (2011). A conditional random field framework for robust and scalable audio-to-score matching. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(8):2385–2397.

[30] Kayacik, H. G., Zincir-Heywood, A. N., and Heywood, M. I. (2005). Selecting features for intrusion detection: a feature relevance analysis on kdd 99 intrusion detection datasets. In *Proceedings of the third annual conference on privacy, security and trust*.

[31] KDD (1999). KDDCUP 1999 data. Available at: http://kdd.ics.uci.edu/databases/kddcup99.

[32] Klinger, R. and Tomanek, K. (2007). *Classical probabilistic models and conditional random fields*. TU, Algorithm Engineering.

[33] Koller, D. and Friedman, N. (2009a). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

[34] Koller, D. and Friedman, N. (2009b). *Probabilistic graphical models: principles and techniques*. MIT press.

[35] Kost, S. (2004). An introduction to sql injection attacks for oracle developers.

[36] Kotsiantis, S. and Kanellopoulos, D. (2006). Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58.

[37] Kou, G., Peng, Y., Chen, Z., and Shi, Y. (2009). Multiple criteria mathematical programming for multi-class classification and application in network intrusion detection. *Information Sciences*, 179(4):371–381.

[38] Kruegel, C., Mutz, D., Robertson, W., and Valeur, F. (2003). Bayesian event classification for intrusion detection. In *Proceedings of the 19th Computer Security Applications Conference*, pages 14–23.

[39] Kruegel, C. and Toth, T. (2003). Using decision trees to improve signature-based intrusion detection. In *Recent Advances in Intrusion Detection*, pages 173–191. Springer.

[40] Kudo, T. (2005). Crf++: Yet another crf toolkit. *Software available at http://crfpp.sourceforge. net*.

[41] Laskey, K., Alghamdi, G., Wang, X., Barbará, D., Shackelford, T., Wright, E., and Fitzgerald, J. (2004). Detecting threatening behavior using bayesian networks. In *Conference on Behavioral Representation in Modeling and Simulation – BRIMS*, Arlington, VA, USA.

[42] Liao, H.-J., Richard Lin, C.-H., Lin, Y. C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24.

[43] Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., and Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, 36(1):42–57.

[44] Ng, J., Joshi, D., and Banik, S. M. (2015). Applying data mining techniques to intrusion detection. In *Information Technology-New Generations (ITNG), 2015 12th International Conference on*, pages 800–801. IEEE.

[45] Nguyen, L. (2017). Tutorial on hidden markov model. *Applied and Computational Mathematics*, 6(4-1):16–38.

[46] Olusola, A. A., Oladele, A. S., and Abosede, D. O. (2010). Analysis of kdd99 intrusion detection dataset for selection of relevance features. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, pages 20–22.

[47] Patil, S. V. and Kulkarni, P. J. (2011). Intrusion detection using conditional random fields. *International Journal of Network Security (2152-5064)*, 2(3).

[48] P.Rigaux (2014). The myreview system. http://myreview.sourceforge.net.

[49] Sebyala, A. A., Olukemi, T., and Sacks, L. (2002). Active platform security through intrusion detection using naive bayesian network for anomaly detection. In *Proceedings of the 2002 London Communications Symposium*.

[50] Shen, Y., Yan, J., Yan, S., Ji, L., Liu, N., and Chen, Z. (2011). Sparse hidden-dynamics conditional random fields for user intent understanding. In *Proceedings of the 20th international conference on World wide web*, pages 7–16. ACM.

[51] Shivarkar, S. A. and Bendre, M. R. (2010). Hybrid approch for intrusion detection using conditional random fields. *International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume*, 1.

[52] Shunmughanathen, K. et al. (2016). An intelligent temporal adaptive genetic fuzzy classification algorithm for effective intrusion detection. *Transylvanian Review*, 24(8).

[53] Silander, T. and Myllymäki, P. (2006). A simple approach for finding the globally optimal bayesian network structure. In *Conference on Uncertainty in Artificial Intelligence*, pages 445–452. AUAI Press.

[54] Singh, S. and Silakari, S. (2009). An ensemble approach for feature selection of cyber attack dataset. *International Journal of Computer Science and Information Security*, 6(2):297–302.

[55] Sutton, C. and McCallum, A. (2006). *An introduction to conditional random fields for relational learning*, volume 2. Introduction to statistical relational learning. MIT Press.

[56] Tan, Y., Liao, S., and Zhu, C. (2011). Efficient intrusion detection method based on conditional random fields. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 1, pages 181–184. IEEE.

[57] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. (2009). A detailed analysis of the kdd cup 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications,(CISDA).*, pages 1–6.

[58] Tian, J., He, R., and Ram, L. (2010). Bayesian model averaging using the k-best bayesian network structures. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 589–597, Corvallis, Oregon. AUAI Press.

[59] Tsai, C.-J., Lee, C.-I., and Yang, W.-P. (2008). A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences*, 178(3):714–731.

[60] Vijayasarathy, R., Raghavan, S., and Ravindran, B. (2011). A system approach to network modeling for ddos detection using a naïve bayesian classifier. In *2011 Third International Conference on Communication Systems and Networks (COMSNETS)*, pages 1–10.

[61] Wallach, H. M. (2004). Conditional random fields: An introduction. *Technical Reports (CIS)*, page 22.

[62] Wang, Y., Jiang, H., Liu, Z., and Chen, S. (2015). A crf-based method for ddos attack detection. In *Proceedings of the 4th International Conference on Computer Engineering and Networks*, pages 81–87. Springer.

[63] Wee, Y. Y., Cheah, W. P., Tan, S. C., and Wee, K. (2011). Causal discovery and reasoning for intrusion detection using bayesian network. *International Journal of Machine Learning and Computing*, 1(2):185–192.

[64] Xie, H. and Chang, C. K. (2015). Detection of new intentions from users using the crf method for software service evolution in context-aware environments. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, volume 2, pages 71–76. IEEE.

[65] Zhang, J. and Gong, S. (2010). Action categorization with modified hidden conditional random field. *Pattern Recognition*, 43(1):197–203.