8-23-2016 12:00 AM

Hidden Markov Model Based Intrusion Alert Prediction

Udaya Sampath Karunathilaka Perera Miriya Thanthrige
The University of Western Ontario

Supervisor
Dr. Jagath Samarabandu
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science

Abstract

Intrusion detection is only a starting step in securing IT infrastructure. Prediction of intrusions is the next step to provide an active defense against incoming attacks.

Most of the existing intrusion prediction methods mainly focus on prediction of either intrusion type or intrusion category. Also, most of them are built based on domain knowledge and specific scenario knowledge. This thesis proposes an alert prediction framework which provides more detailed information than just the intrusion type or category to initiate possible defensive measures. The proposed algorithm is based on hidden Markov model and it does not depend on specific domain knowledge. Instead, it depends on a training process. Hence the proposed algorithm is adaptable to different conditions. Also, it is based on prediction of the next alert cluster, which contains source IP address, destination IP range, alert type and alert category. Hence, prediction of next alert cluster provides more information about future strategies of the attacker.

Experiments were conducted using a public data set generated over 2500 alert predictions. Proposed alert prediction framework achieved accuracy of 81% and 77% for single step and five step predictions respectively for prediction of the next alert cluster. It also achieved an accuracy of prediction of 95% and 92% for single step and five step predictions respectively for prediction of the next alert category. The proposed methods achieved 5% prediction accuracy improvement for alert category over variable length Markov based alert prediction method, while providing more information for a possible defense.

Keywords: Feature Reduction, Hidden Markov Model, Intrusion Alerts, Intrusion Detection, Intrusion Response, Intrusion Prediction

Acknowledgement

First, I am sincerely grateful to my main supervisor, Dr. Jagath Samarabandu for his immense support, motivation, immense knowledge and guidance given to me throughout my studies.
I would like to thank my co-supervisor Dr. Xianbin Wang for his support and guidance.
I would also like to thank my supervisors for giving me the opportunity to work with them in a great research environment.

This work could not have been completed without the support and encouragement of my loving family members although we lived far away from each other. I take this opportunity to express my gratitude to my father, late Karunathilaka Perera, mother, Gunasinha Manike, father-in-law, Gamage Karunathunge, mother-in-law late, Lalitha Abeysekara and my beloved wife, Nilusha Karunathunge who actively encouraged me during my studies.

I would like to thank my friends (in no particular order), Rasika, Asanka, Himasha, Uditha, Nirosh, Nadun, Sriyananda, Dimuthu, Sharath and my lab mates Marjan, John, Matt and Justin for their support and guidance.

I would like to express my gratitude to professors at Western including Dr. Vijay Parsa, Dr. Aleksander Essex, Dr. Abdallah Shami and Dr. Quazi Rahman for their knowledge and guidance given throughout my studies at Western University. Also, I would like to thank all the staff members of Western University for making a supportive and nice environment.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
ASG	Attack Session Graphs
AUC	Area under the curve
AWID	Aegean Wi-Fi Intrusion Dataset
BoW	Bag of Words
CH	Chi-Squared statistic
CPU	Central Processing Unit
DAG	Directed Acyclic Graph
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DMZ	DeMilitarized Zone
DNA	Deoxyribonucleic acid
DNS	Domain Name Server
DoS	Denial of Service
FL	Fuzzy Logic
FP	False Positive
FTP	File Transfer Protocol
GA	Genetic Algorithm
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ID	Identification
IDS	Intrusion detection systems
IG	Information Gain
IP	Internet Protocol
IRS	Intrusion Response Systems
KNN	K-Nearest Neighbor algorithm

LCS	Longest Common Subsequence
NIDS	Network Intrusion Detection System
NMAP	Network Mapper
PR	Probabilistic Reasoning
RDP	Remote Desktop Protocol
ROC	Receiver Operating Characteristic
RPC	Remote Procedure Call
RSH	Remote Shell
SDF	Sensitive Data Flow
SSH	Secure Shell
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TP	True Positive
UDP	User Datagram Protocol
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

1.1 Motivation

Human activities associated with computational devices are rapidly growing day by day. Nowadays most smart devices are capable of connecting to the internet. The world is becoming a more connected place and people are heavily relying on services offered based on computer networks. With such a high involvement with interconnected devices, sensitive and valuable information are rapidly exchanging between computer networks and devices. On the other hand, traditional networks, applications and protocols were developed in an era where security was not a critical factor. Thus most of the networks, protocols and applications are vulnerable to intrusions. In some cases, although systems were developed to work in a secure network environment, due to rapid development, these systems needed to interconnect with other networks to provide new on-line services.

Devices with interconnection capabilities are growing rapidly in numbers. McAfee labs estimated that there will be 24.4 billion IP connected devices and 4 billion users in 2019 [1]. On the other hand, network intrusions are growing rapidly and new attack vectors are continuously being used to overcome security systems. Hence, deployment and continuous improvement of network security systems are of paramount importance. Their presence attempts to secure the network from intrusions and attacks and keeps the system operational to provide continuous services to its users.

An action or set of actions that compromises confidentiality, availability and integrity of a computer system is defined as an intrusion [2]. Intrusion detection is defined as identification of [3]:

- Users who attempt to use a computer system without authorization,
- Authorized users who are abusing their privileges.

Initially, most of the research activities were focused on intrusion detection methods. Such intrusion detection systems identify the attacks that have already happened or currently happening in the network. To improve network security, advanced system models are required, which are able to detect impending or ongoing intrusions and take countermeasures to stop intrusion activities. Researchers are focusing on developing sophisticated Intrusion Response System (IRS) to fulfill this requirement.

The main objectives of an Intrusion Response Systems (IRSs) can be described as

1. Identification of ongoing and future intrusions,
2. Execute necessary prevention mechanisms to prevent ongoing and future intrusions,
3. Apply necessary precautions that make similar kind of intrusions less successful in the future.

Prediction of future intrusion activities plays a key role in intrusion response, which helps in identifying and executing response actions before an intrusion occurs. Intrusion detection systems (IDSs) generate intrusion alerts once they detect network intrusions. An intrusion alert is a record that contains important information about the intrusion, such as the origin, the victim, type of the intrusion and the time when the intrusion has been detected [4]. Multiple intrusion alerts may be generated for a particular intrusion due to slight variations. As an example, a “port scan” intrusion may produce two different types of intrusion alerts such as “TCP Portscan” and “UDP Portscan” based on execution of the port scan. These alerts can then be used as an input to intrusion response systems to understand attacker strategies and predict

future network intrusions.

The general system architecture of intrusion response system is shown in Figure 1.1. It is a modified version which was originally presented by Shameli et al. [5]. Alert Pre-processing and intrusion analysis module integration to the intrusion response selection process is additionally proposed to the architecture presented by Shameli et al. [5]. IRS utilizes IDS alerts as the main input source. IRS system architecture is mainly divided in to three modules: pre-processing, intrusion analysis and intrusion response.

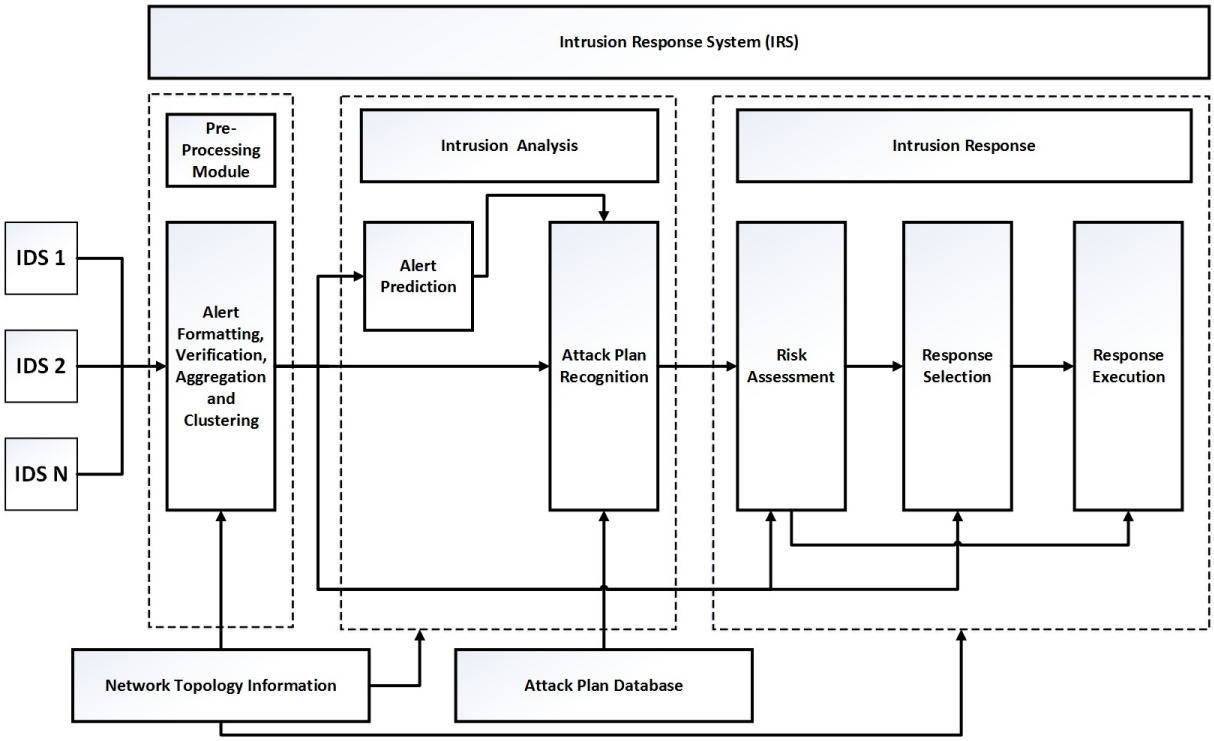


Figure 1.1: Architecture of intrusion response system.

1. Pre-processing module. Main input source of the IRS is IDS alerts. IRS received alerts from different IDSs. Different IDSs generates alert in different formats and may contain different attributes as well. In the first step, these alerts are formatted to a common format which contains important alert attributes. Since there are different input sources, redundant alerts may be present in the input. These redundant alerts are removed by the pre-processing module. Generally, IDSs generate a high volume of alerts. It is impossi-

ble to analyze these alerts individually and even if one were to do this, it may not produce useful information. Techniques such as alert aggregation, clustering and verification are employed to reduce large volume of alerts and group similar alerts together which makes analysis much easier.

2. Intrusion analysis module. The main objective of intrusion analysis module is to gain knowledge about current and future intrusion strategies of a would be attacker using intrusion alerts. Then that knowledge is used to execute responses to prevent ongoing and future intrusions. Intrusions Prediction methods can be mainly classified as,

   - Attack graph based intrusions prediction. An attack graph represents attack sequence of multi-stage attack. Attack graph based intrusion prediction employs an existing database of attack graphs for prediction. Based on the observed alert sequence the most suitable attack graph which matches observed intrusion sequence is selected by alert prediction module. Future intrusions are then predicted by the aid of the selected attack graph.
   - Sequence modeling based intrusions prediction. Alerts generated by IDS is treated as a sequential data series. Where current data point depends on previous data point or previous data points. Observed sequential data series is used to model a system in which that system represents the behavior of observed sequence. This model learns the properties of the sequential data series. Once a certain sequence of data points is observed, most probable next data point can be identified using this model. In the context of alert prediction, a model is developed based on observed alerts. This model represents the behavior of observed alert series. By using that model, possible future alerts are predicted after a sequence of historical alerts were observed.

IRS generates an attack plan based on detected intrusions and predicted intrusions. An attack plan consists of sequence of attack steps. This attack plan contains attack steps completed up to now and possible future attack steps. Based on that knowledge, IRS selects suitable response actions to prevent current and future intrusions.

3. Intrusion response module. Intrusion response module is the core of the intrusion response system. The main purpose of intrusion response module is to select and execute best response actions based on ongoing and predicted intrusions. Selecting a response is a very complicated process. Main components of response selection module can be categorized as risk assessment module, response selection module and response execution module [5].

    - Risk assessment module. Risk assessment module assesses risk of ongoing or predicted intrusions to the network services and assets. Output of risk assessment module is sent into response selection module.
    - Response selection module. Response selection module is responsible for selecting a response to an intrusion. The response selection methods can be mainly categorized as static mapping, dynamic mapping and cost-sensitive methods [5].
    - Response execution module. Response execution module is responsible for executing selected responses. Response execution can be categorized as proactive response, immediate response and delayed response based on response execution time [6], [7].

A detailed description about intrusion response systems is included in the section 2.2.

1.2 Alert Prediction in IDS

Most of existing intrusion response systems are operating in immediate response mode or delayed response mode. Where responses are executed immediately or after a specific time window of an intrusion detection. The major disadvantages of the immediate and delayed response methods are shown below [5].

- Response applied once an intrusion actually occurred, hence it may be difficult or not possible to revert back network to its original state.
- There is a possibility of spreading attacks through the network until it controls by the response system.

- It may take a time to revert back system to its original state (or healthy state) and system services can be affected during that period. Also, the attacker has sufficient time to execute more attacks until response is applied.

By considering these factors, it is beneficial to develop an IRS with proactive response capabilities. Alert prediction in IDS plays a key role in developing intrusion response system with proactive response capabilities. Let intrusion alerts observed in a specific time window be represented as $A = A_1, A_2, \ldots, A_t, \ldots, A_T$. Intrusion alert prediction is defined as predicting a next possible intrusion alert A_{T+1}, given that the previously generated intrusion alert sequence A is available.

Intrusion alert prediction can be mainly categorized as network attack graph based prediction [8], [9], [10] and sequence modeling techniques (such as Markov model, Hidden Markov model, Bayesian networks and Dynamic Programming) based prediction [11], [12], [13]. Sequence modeling techniques have been successfully implemented in fields such as biology (DNA sequence) [14], speech pattern identification [15] and financial data forecasting [16]. Hidden Markov Model (HMM) is one of the widely used sequential data modeling method. The hidden Markov model is a stochastic model which was introduced in the late 1960s by Baum and his colleagues [17], [18]. Due to rich mathematical structure of hidden Markov model, it widely applied in real world applications such as speech recognition, handwriting pattern recognition, gesture recognition, intrusion detection and speech tagging.

Intrusion alerts typically contains two fields that are used to identify an intrusion which caused the corresponding intrusion alert. These fields are called “alert type” and “alert category.” Alert type contains detailed information about an intrusion whereas alert category contains higher level information about an intrusion. As an example “UDP Filtered Portsweep” and “UDP Portsweep” alert types both belongs to “attempted-recon” alert category and the alert type “UDP Filtered Portsweep” provides much more specific information than the alert category “attempted-recon.”

Most of the previous intrusion prediction methods were mainly focused on prediction of ei-

ther alert type or alert category. A proper response action cannot be executed by only knowing future intrusion type or intrusion category. In order to execute useful response action, it is important to predict other important parameters such as who is the attacker (attacker IP address) and who is the victim of the attack (victim IP address). Then response action can be executed to specific network segment without applying response action to the whole network. Also, response action can be targeted specifically to the attacker. The behavior of attackers are depending on many factors such as network topology, operating systems running on hosts, services running on the network and geographical location of networks, etc. Therefore, prediction module should be able to adapt different conditions and networks. If prediction module heavily depends on domain knowledge and specific scenario knowledge (such as attack graphs based prediction methods) whereas such a kind of prediction modules will not be able to adapt to different conditions. Hence it is important to develop an alert prediction module which is capable of operating under different conditions and networks.

1.3 Contributions

The objective of this research is to develop an alert prediction module based on hidden Markov models.
Key contributions include:

1. Investigation of feature reduction on the performance of intrusion detection based on an existing data set containing network intrusions.

2. An algorithm for predicting the next intrusion alert, given a sequence of intrusion alerts from an existing IDS tool such as ”Snort”.

3. Comparison of the proposed alert prediction algorithm against existing prediction algorithms which shows comparable accuracy while providing additional information for a possible response.

4. A software module which implements the proposed alert prediction algorithm.

Block diagram of the proposed alert prediction framework is illustrated in Figure 1.2.

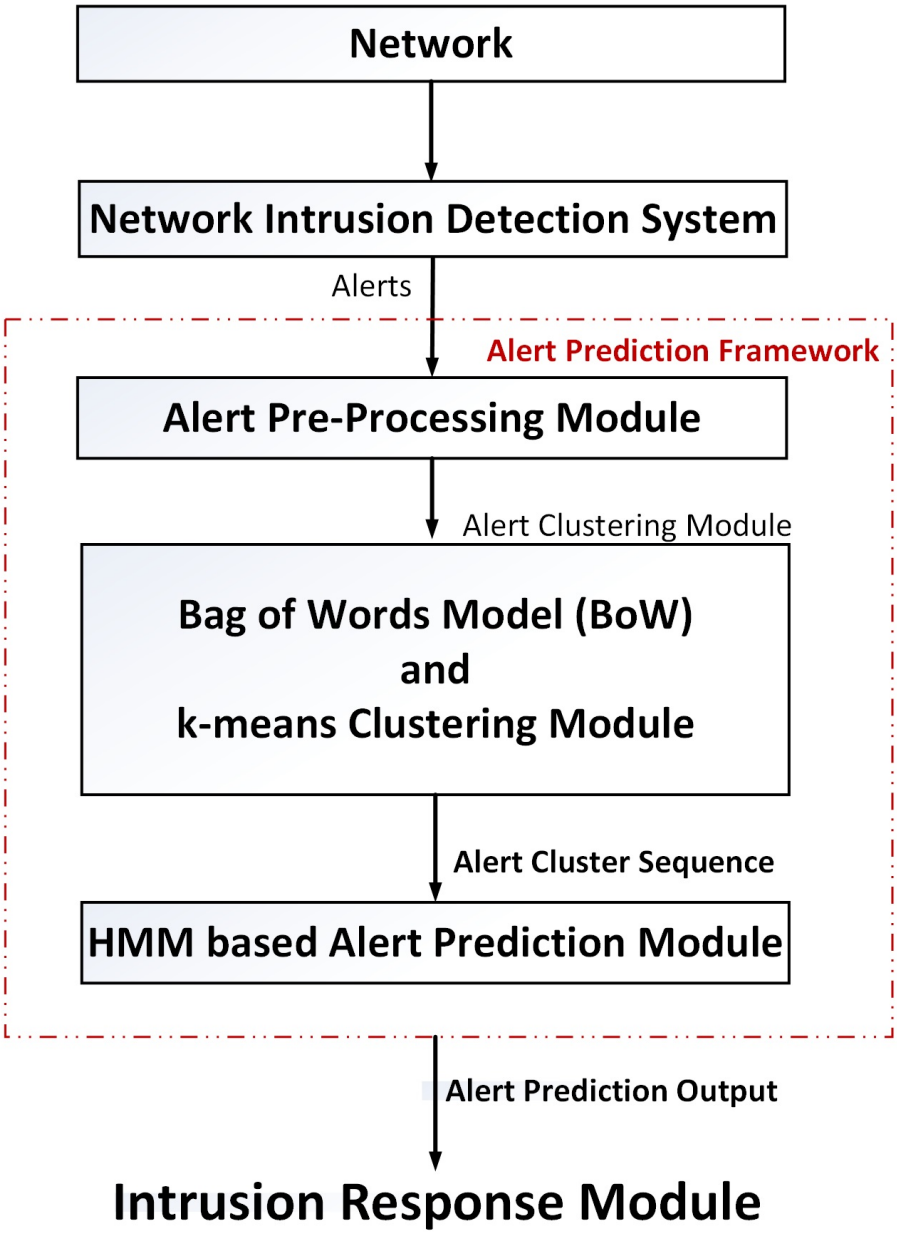


Figure 1.2: Block diagram of the proposed alert prediction framework.

1.4 Document Structure

The content of this thesis organized as follows:

Chapter 2 consists of a literature survey of intrusion alert prediction research area. It describes

an overview of intrusion detection methods and intrusion response methods. Also, this chapter includes a detailed description of alert prediction research activities.

Chapter 3 describes usage of feature reduction techniques such as Information Gain and Chi-Squared statistics to improve intrusion detection capabilities of Random Forest, Random Tree, Adaboost, J48 decision tree and OneR machine learning techniques.
Also, this chapter describes alert prediction process in detail, which includes the Bag of Words (BoW) model based k-means alert clustering process and implementation of hidden Markov model for alert prediction. Also, this chapter discusses software and programming language use in implementation of alert prediction module.

Chapter 4 includes the results of intrusion detection performance evaluation of Adaboost, Random Forest, Random Tree, J48 and OneR machine learning techniques with Information Gain and Chi-Squared statistics feature reduction techniques.
Also, this chapter discusses performance evaluation of proposed alert prediction method with respect to the number of clusters and hidden Markov model parameters. It also compares prediction accuracy of alert cluster based sequence modeling and alert category based sequence modeling.

Chapter 5 includes critical analysis of results and challenges involve in the alert prediction process. This chapter also discusses possible basic intrusion response actions that can be implemented based on output of proposed prediction system. It also includes recommended improvements for proposed system.

Chapter 2

Background

In the early days, when a network intrusion is detected by intrusion detection systems, it notifies the incident to the system administrator to take further actions. However, this process can be delayed due to reasons such as lack of availability of human experts in real time. As a solution for this, researchers are interested in developing automated Intrusion Response Systems (IRSs) which are capable executing responses automatically without human involvement. However, executing a response to an intrusion that has already happened, or currently happening in the network requires developing an IRS with prediction capabilities that is capable of preventing ongoing and future intrusions.

Within the broader area of network security research, there are many research activities that aim to improve intrusion detection, prediction and response. This chapter summarizes intrusion detection, prediction and response. Section 2.1 describes intrusion detection methods in general and section 2.2 describes intrusion response methods. Alert processing and prediction are described in more detail in section 2.3.

2.1 Intrusion Detection Systems

Intrusion detection systems (IDSs) generate alerts once they detect intrusion activities. These alerts can be used as an input to intrusion response systems to understand strategies of an attacker and predict future intrusion activities. Typical network security intrusion detection

methods can be classified as misuse based detection, anomaly based detection (behavior based detection) or a combination of misuse based and anomaly based [19], [20].

Misuse based detection mechanisms compare network packet flow with known malicious threats patterns. If a pattern is matched, it is identified as an intrusion [21], [22]. Advantages of misuse based intrusion detection are higher accuracy and easy implementation. Inability to detect unknown intrusions and requirement of regular pattern database updates are the main disadvantages of misuse based detection.

Anomaly detection is based on the behavior of the system/user. The behavior of the system is classified as a normal operation or abnormal operation mode based on measurements of system parameters and characteristics such as bandwidth utilization of services, protocols that used in the system, ports involved and transaction rates of services [23], [24]. The main advantage of behavior based detection is that it is capable of identifying unknown attacks. The main disadvantage is lower accuracy.

2.1.1 Classification of Intrusion Detection Techniques

Intrusion detection approaches can be categorized as below [25], [26]:

1. Knowledge based systems. Knowledge based intrusion detection system contains information about known attacks and vulnerabilities. The knowledge base is utilized to identify possible attacks on the network. The simplest knowledge based detection is pattern matching based intrusion detection [27], [28]. State transition based detection [29], [30], expert systems [31], [32] and logic based detection are other knowledge based intrusion detection systems.

2. Statistical methods. Network attribute values are measured over a time and behaviors of the system are classified based on statistical properties of these attributes, common attributes include CPU usage, memory usage, network usage, number of user logins, number of logout requests, number of active tcp connections, etc. During normal operation, these attributes will be in a certain range. During an abnormal operation they may

diverge from their normal range, which are used to identify intrusions [33], [34], [35]. The main advantage of statistical models is that it can identify unknown attacks. The main disadvantage of statistical methods is the need to monitor a system over long time period to accurately model the system.

3. Protocol analysis. Protocol analysis looks for misuse or incorrect use of protocols to detect security attacks [36], [37], [38]. Normal behaviors of the system are modeled based on protocol usage analysis. If the protocol usage is different from normal behavior, the state of the system is identified as abnormal.

4. Soft computing and classification based methods. Soft computing methods are used to solve problems when available information is insufficient. Different soft computing and machine learning methods have been proposed for intrusion detection. Artificial Neural Network based systems (ANNs) [39], [40], Fuzzy Logic (FL) based systems [41], [42], [43], Genetic Algorithms (GAs) based systems [42], [44], Artificial Immune Systems [45], [46], [47], Probabilistic Reasoning (PR) based systems and Ant Colony Algorithms based systems [48], [49] are most popular soft computing methods that are used in network security domain.

   Classification and clustering group similar kind of objects together and as a result of that, it is possible to identify different types of objects or behaviors. Classification methods such as Support Vector Machine (SVMs) and K-Nearest Neighbor algorithm (KNN) are most popular classification methods those are used in intrusion detection [50], [51], [52]. Advantages of soft computing and classification based methods are the ability to work with incomplete information as well as the ability of detecting unknown attacks. Over-fitting and more time consumption during the training process are the main disadvantages of soft computing and classification based methods.

5. Hybrid methods. Hybrid methods combine two or more intrusion detection methods or techniques. The intrusion detection system assigns weights for each method or techniques and combines the result of individual methods to calculate the final result. As an example, the detection system may employ two classification techniques to classify the

data and combines results of two classification technique to get the final result [53], [54]. Also, intrusion detection system may combine two different methods such as knowledge-based detection methods and behavior-based detection methods to develop a hybrid detection system [55], [56], [57]. The main disadvantage of hybrid methods is higher computational cost compared to individual methods.

There is a possibility that intrusion detection systems detect normal behavior of the system as abnormal and conversely abnormal as normal. These two events are referred as a false alarm and a missed detection respectively. Also, it is possible that IDSs fail to identify exact intrusion type even they identified it is as an intrusion activity, this is referred as a detected attack. An actual intrusion that correctly identified by IDS as an intrusion with correct intrusion type is referred as a true alarm. False alarms and missed detections should be minimized to accomplish better result in intrusion response process [7].

2.1.2 Data Reduction Techniques to Improve Intrusion Detection

IDSs collect network data captured by sensors and sniffers for intrusion detection purposes. Typically network sensors produce a larger amount of data even for a small network. Also, network packet capture contains many attributes (features) such as source IP address, destination IP address, packet frame length, packet data length, etc. It is impossible to analyze all these features and even if one were to do this, it increases computational cost of the IDS. Therefore, it is important to identify the best features that efficiently contribute to intrusion detection process. Sensor data reduction methods can be broadly categorized as following [54]:

- Data filtering. Data filtering is used to reduce the amount of sensor data that is processed by IDSs. Unwanted data captured by sensors are filtered by the filtering process. As an example, IDS can filter out some data based on human expert analysis (such as log files and partially captured data). Filtering process helps to improve detection accuracy and decrease storage requirement. Data filtering should be done with carefully, otherwise useful data can be filtered out.

- Feature selection. It is important to identify the relationship of features to intrusion detection. Some features may contain redundant information and some features may

actually not contain any useful information relevant to intrusion detection. With increase of features, processing time of IDS may increase as well. Therefore, it is important to identify features that mostly contribute to intrusion detection and it helps to develop efficient and accurate IDS.

- Data clustering. Data clustering groups similar kind of objects together and as a result of that, it can be used as a data reduction method. Also, it reduces processing and analyzing cost of data, because clustered data can be analyzed and processed instead of individual data.

Feature selection methods can be mainly classified as filter-based methods and wrapper-based methods [58]. Although filter-based methods have better efficiency when compared with wrapper-based methods, wrapper-based methods are shown to have better accuracy than filter-based methods [59].

1. Filter-based feature selection methods. Filter-based feature selection methods evaluate features by inspecting inherent properties of the data. Filter-based feature selection is a pre-processing step and it is independent from induction algorithm (classification algorithm) [58]. Filter based methods apply statistical tests to the features to identify which features are mostly correlated to the output. It calculates a score for each feature and features with a higher score are more correlated to the output. Advantages of filter-based feature selection methods are following [60]:

   - Scalable, can be applied to large feature set and large datasets,
   - Independent from classification algorithm. Therefore, selected features can be applied to different classification algorithms,
   - Computationally less expensive.

   Information Gain (IG) [61], Relief algorithm [62], Chi-squared test [61], Markov Blanket Filter (MBF) [63] and Correlation-based Feature Selection (CFS) [64] are some filter-based feature selection methods.

2. Wrapper-based feature selection methods. Wrapper-based feature selection methods evaluate different subset of features using induction algorithm [58]. Wrapper-based method prepares different combination of feature sets from original feature set. Then feature sets are evaluated by adding and removing features to find the best feature set using an induction algorithm. The main advantages of wrapper-based feature selection methods are higher accuracy and ability to select the best feature set [59]. Higher computational cost and being dependent on the induction algorithm are main disadvantages of this method [60].

2.1.3 Datasets for Training, Testing and Evaluation of Intrusion Detection

Although real time intrusion detection is an important feature of an intrusion detection system, offline mode operation provides many opportunities. For research, offline mode provides opportunity for in depth analysis of patterns and behaviors of intrusions. In addition to that, it provides opportunity for in depth testing of intrusion detection algorithms. In operational point of view, offline mode provides in depth analysis of past data and generates prevention methods for future occurrences.

Networks generate different traffic patterns based on their behavior and network elements generate alerts, logs, alarms and warnings if they detect an abnormal behavior. These two types of data are used to identify the abnormal behaviors of the networks. The network data can be generated using two main methods a) capturing real network data and b) generating data by simulation. These generated network data is used to develop intrusion datasets. Intrusion datasets provide an opportunity for in-depth analysis of patterns, behaviors of intrusions and testing of intrusion detection algorithms offline. Intrusion dataset can be mainly categorized as [25]:

1. Real network dataset. Real network dataset includes data captured from real networks over few days. The captured data include normal and abnormal behaviors of the network. Famous real network datasets are UNIBS dataset [65] and ISCX−UNB dataset [66].

2. Benchmark dataset. Benchmark dataset is generated by simulated environments in large network. Simulated environment generates different intrusion scenarios. The most popular and widely used benchmark dataset is KDDcup99 dataset [67]. Other famous datasets are NSL-KDD dataset [68], DARPA dataset [69], UNSW-NB15 dataset [70], LBNL-ICSI dataset [71] and CAIDA dataset [72] are famous benchmark datasets.

3. Synthetic dataset. Synthetic dataset is generated to meet a specific requirement. Synthetic dataset is usually used to evaluate the system prototype theoretically.

In this thesis, two datasets were used for the experiments. The Aegean Wi-Fi Intrusion Dataset (AWID) [73] was used for the performance evaluation of OneR, Adaboost, J48 decision tree, Random Forest and Random Tree machine learning techniques based intrusion detection with feature reduction. The DARPA (Defense Advanced Research Projects Agency) [69] dataset was used for the performance evaluation of proposed alert prediction framework.

2.2 Intrusion Response Systems

Intrusion Response Systems (IRSs) mainly utilize alerts generated by the Intrusion Detection Systems to gain knowledge about ongoing and impending intrusions. Typically, most of the IDSs generate a large amount of alerts during their operation. These alerts must be utilized efficiently to gain knowledge about intrusions. Mechanism such as alert aggregation [74], [75], [76], alert correlation [77], [78], [79], [80] and alert prediction [8], [11], [12], [81] are proposed to improve utilization of IDS alerts and gain knowledge about ongoing and impending intrusions using IDS alerts.

2.2.1 Intrusion Response System Classification

The following section describes classification of intrusion response systems which is mainly based on studies done by Shameli et al. [5] and Stakhanova et al. [82]. The classification of intrusion response systems is shown in Figure 2.1 which is generated based on studies done by Shameli et al. [5] and Stakhanova et al. [82].

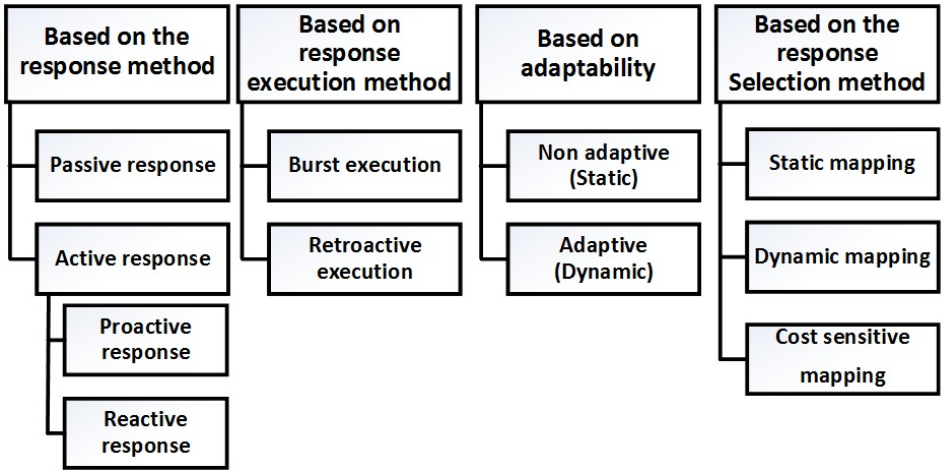


Figure 2.1: Classification of intrusion response systems which is generated based on studies done by Shameli et al. [5] and Stakhanova et al. [82]

1. Classification based on automation. Notification systems, manual response systems and automated response systems are the main categories of IRS when they are classified based on their degree of automation.

   - Notification response systems. The intrusion response systems identify intrusion activities and then IRS notifies to other systems or system administrator to perform further actions.
   - Manual response systems. Manual response systems have pre-configured action sets based on the type of the intrusion. When the system detects an intrusion, it prompts these pre-configured action sets to the system administrator to execute.
   - Automated response systems. Automated response systems operate without human intervention. The system applies necessary prevention actions automatically when it detects an intrusion or identifies an intrusion in progress or predicts an intrusion.

2. Classification based on response methods. Passive response methods and active response methods are the main categories of IRS when they are classified based on their on response methods.

   - Passive response methods. Once an intrusion is detected or predicted, the response system informs it to other parties and relies on their actions.

- Active response methods. Response system itself takes actions to stop or minimize impact of security threats.

Active response execution has a major impact on intrusion prevention and running condition of the system. Time of a response execution and response execution sequence are major characteristics of active response implementation. Response mechanism can be categorized as proactive response, immediate response and delayed response based on response execution time [6], [7].

- Proactive response mode. Response actions are taken before an intrusion happen.
- Immediate response mode. Response actions are taken immediately once an intrusion is detected.
- Delayed response mode. Response actions are taken after a specific time window of an intrusion detection.

Most of the existing intrusion response systems are operating in immediate response mode or delayed response mode. The major disadvantages of the immediate and delayed response methods are described below [5].

- Responses are applied once an intrusion actually occurred, hence it may be difficult or not possible to revert back network to its original state.
- There is a possibility of spreading an attack through the network until it is controlled by the response system.
- It may take a time to revert back system to its original state (or healthy state). System services can be affected during that period. Also, an attacker has sufficient time to execute more attacks until a response is applied.

3. Classification based on response execution methods. Response execution can be mainly categorized as burst executions or retroactive executions.

- Burst executions. All actions are executed once an intrusion is detected. IRS does not analyze the impact on each individual action. The main disadvantage of burst

executions is that it has a higher operation cost in some cases. Most of the times lesser number of actions may be sufficient to prevent an intrusion.

- Retroactive execution. An initial set of response actions are executed and risk of the intrusion to the network is evaluated before executing further response actions. Based on risk of the intrusion the need to execute any further actions are identified.

4. Classification based on adaptability. Intrusion response systems can be categorized as non-adaptive (static) and adaptive (dynamic) based on their response behavior.

5. Classification based on response selection methods. The response selection method is one key property that determines the adaptability of intrusion response systems. The response selection methods can be mainly categorized as static mapping, dynamic mapping and cost-sensitive methods.

   - The static response selection is the simplest response selection method where an intrusion is statically mapped to a particular response or set of responses [83]. Usually, these kinds of systems are easy to build and maintain. On the other hand, their responses are easy to predict and therefore, real-world intrusions may adapt to evade such preventive measures. Additionally, it does not consider risk level of the intrusion to the network in the response selection process, which is another weakness of static response selection.

   - Dynamic response mapping systems select responses dynamically based on characteristics of an intrusion. Each specific intrusion has a set of responses that are assigned to it. Responses are dynamically selected by the intrusion response system, based on intrusion matrices (which are built based on intrusion characteristics such as intrusion severity and intrusion frequency) [84], [85].

   - Cost sensitive response selection methods select responses based on the damage cost of intrusion and cost of the response [86], [87]. Response action also has a cost, in some scenarios cost of executing response action is higher than the cost of the intrusions. In such a situation it is worth to keep the system with intrusion than executing response action. As an example consider Daniel of Service attack (DoS)

where an attacker floods useless traffic to degrade network services. In some cases, response action used to mitigate DoS attack may degrade network performance than DoS attack.

2.3 Alert Processing

The main objective of alert processing in an IDS is to gain knowledge about the current security status of the system and predict possible intrusion activities that can happen in the future. This section describes the overview of alert processing which includes alert aggregation, alert correlation and alert prediction.

- Alert pre-processing. The main objective of alert pre-processing is to remove redundant alerts that are received. The response systems may receive alerts from multiple intrusion detection systems. Since there are different input sources, redundant alerts may be present in the input. The first step of pre-processing is to format raw alerts from different IDSs to a common format and then duplicated alerts are removed using selected features. Typical features that are used to eliminate duplicate records are timestamp of the alert, source IP address, source port, destination IP address, destination port, signature of the alert (alert type) and signature class of the alert (alert category). Once duplication removal process is completed, then these alerts are used in the analysis process.

- Alert fusion, aggregation and clustering. Typically, IDS produces a large number of alerts. It is impossible to analyze these alerts individually and even if one were to do this, it may not produce useful information. Alert fusion, aggregation and clustering is a process which combines similar type of alerts together which makes analysis much easier with much more useful information. Alert clustering can be done in many different ways such as combining all alerts with same attributes (except time stamp) [81], [88] or grouping alerts with same alert type within a predefined time period [89].

- Alert correlation and Alert prediction. Alert correlation provides the relationship of alerts which is generated due to intrusions. By analyzing alert correlation, it is possible to identify strategies and plans of the attacker which can be used to predict possible future in-

trusion actions. During last few years many alerts correlation techniques were proposed. These correlation techniques can be classified as following [90]:

1. Alert correlation based on feature similarity. Alert correlation is done based on feature similarities of the alerts. Features such as source address, destination address, source port, destination port and alert type are used for correlation.

2. Alert correlation based on known scenarios. Correlation of alerts is defined based on known intrusion scenarios. The knowledge about the intrusion scenarios are stored as an attack graph or specified in attack languages (such as LAMBDA [91], ADeLe [92]). But the limitation of this method is it can only correlate alerts based on knowledge base availability.

3. Alert correlation based on preconditions and consequence. Early stage activities of an attacker have an impact on his later activities. If early stage steps are successful then later stage steps are more probable to execute and succeed. By analyzing pre intrusion activities sequence it is possible to predict possible post activities.

2.3.1 Intrusion Activities Prediction

Prediction of future intrusion activities based on observed intrusion activities is very challenging. Observed IDS alerts can be used to predict possible future intrusion activities. Intrusion activity prediction can be mainly classified as:

- Attack graph based intrusion activity prediction,
- Sequence modeling based intrusion activity prediction.

Attack Graph Based Intrusion Activity Prediction

An attack graph is one of the most commonly used method in network security analysis. An attack graph represents possible intrusion sequence of multi-stage attack. Consider a network present in [93] which consists of file server (host 1) and database server (host 2). The host 1 offers File Transfer Protocol (ftp), Remote Shell (rsh) and Secure Shell (ssh) services. The host

2 offers ftp and rsh services and firewall allows ftp, rsh and ssh traffic between host 0 to host 1 and host 2. One possible attack sequence for this network can be described as [93]:

1. Step 1. Attacker executes ssh buffer overflow exploit from host 0 to host 1.
2. Step 2. Attacker gets access to the host 1 and executes arbitrary codes on host 1.
3. Step 3. Attacker uploads a list of trusted hosts to the host 2 using ftp vulnerability.
4. Step 4. Attacker remotely executes shell commands on host 2.
5. Step 5. Attacker gains the root privilege of host 2 by using a local buffer overflow exploit on host 2.

The attack graph represents possible attack sequence. Hence, these attack graphs can be used to predict possible future attack activities once initial steps were observed.

Attack Graph Based Alert Prediction Research Activities

Qin and Lee proposed attack projection scheme based on probabilistic reasoning method [81]. They used attack tree to develop Bayesian network. The Bayesian network is represented as a Directed Acyclic Graph (DAG) with each node of the graph represents a variable which has a certain set of states and directed edges represent the relationship between variables. They represented the root node as the main goal of the attacker and leaf nodes as sub-goals of the attacker. The prior probability of parent nodes status and a set of conditional probability associated with child nodes are two main parameters of the Bayesian networks. They evaluated the likelihood of the goal and sub-goals based on observed intrusion activities and used this knowledge to predict possible future intrusion activities. The main limitation of this process is prediction depends on the availability of an attack plan library. Also, it requires mapping best possible attack graph for a given observation scenario.

Similar to the method proposed by Qin and Lee, Ramki et al. proposed Bayesian network based directed acyclic graph (DAG) aided alert prediction method [89] where they evaluated the correlation between alert types. Based on correlation and generated attack graph, they predicted next possible alert type that can occur. They used the DARPA2000 dataset [69] for their

alert type prediction.

Li et al. [8] proposed attack graph based intrusion prediction method. They used association rule mining to generate attack graphs. They examined the relationship between alerts to generate the attack graph. This attack graph represents an ordered sequence of attack steps with a predictability score from one step to another. By observing the attack graph, it is possible to predict possible future steps once several initial steps are observed. They used the DARPA1999 [69] and DARPA2000 [69] datasets for their experiments.

Multi-stage attack forecasts based on probabilistic matching was proposed by Cheng et al. where prediction of attack steps were done by measuring the difference between the stored and the actual multi-stage attack session graphs (ASG) [94]. Their method was inspired by the generalized Hough transform and polygonal curves mapping concept was used to find similarities between stored attack graphs and ongoing attack. Probabilistic mapping does not require exact mapping, therefore, this method performs better when it compare with exact mapping techniques such as Longest Common Sub-sequence (LCS) algorithm. They used a mapping which assign a number to an alert attribute based on attribute value range or condition. Then these numeric values of attributes are used to convert an intrusion alert to an ID. Attack session graphs were generated by using alert ID as a y value and corresponding time stamp as x value. They used the DARPA2000 dataset [69] for their experiments. The major limitation of this method is that prediction depends on the availability of an attack graph library.

Lippmann et al. analyzed attack graph generation methods proposed by various researchers and illustrated that most of these graph generation involved network with less than 20 nodes and most of them had poor capability of scaling [95]. This finding indicates that intrusion activity prediction methods propose based on attack graphs are not very feasible to employ in practice due to scaling issues.

Sequence Modeling Based Intrusion Activity Prediction

Sequence modeling techniques are successfully implemented in fields such as biology (DNA sequence) [14], [96], speech pattern identification [15], [97] and financial data forecasting [16], [98]. Sequential data can be defined as finite set of symbols appear in a sequential manner, as an example, we can consider A, B, E, G, T as a set of symbols, then few sequence data pattern can be represented as

Pattern 1 = A, E, G, E, B, T, A, A, G.

Pattern 2 = G, E, B, T, B, A, B, A, B.

Pattern 3 = T, A, A, G, E, B, A, A, G.

By studying symbol generation patterns, a model can be developed. This model represents characteristics of observed symbol pattern. This model can be used to identify desired symbol output (i.e. validating observed output) and predict future outcomes. When we think about the cyber domain, we can apply above properties to detect intrusion activities and predict possible intrusion activities. As an example, by observing above three patterns it can be seen that G, E, B, T and T, A, A, G are common sequence of patterns. If G, E, B pattern is observed and then we can predict T as a next possible symbol with higher probability compared to the other symbols. Markov model and hidden Markov model are widely used models for sequence modeling [96], [99].

Markov Model

Consider a source that generates one symbol at a time from the alphabet S = {A, T, E, D, M} as shown in Figure 2.2. N^{th} order Markov model assigns a probability estimation of a symbols of the alphabet based on "N" number of previous symbols outcomes. Let outcome symbol at time = t represents as X_t then outcome at time = t+1 (X_{t+1}) depends on $X_{t+1-N}, \ldots, X_{t-1}, X_t$ previous observations.

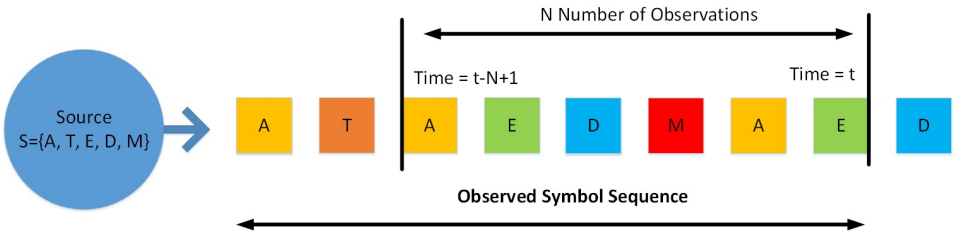


Figure 2.2: Symbol sequence generated by Markov model.

Hidden Markov Model

The Hidden Markov Model (HMM) is stochastic model which was introduced in late 1960s by Baum and his colleagues [17], [18]. Due to rich mathematical structure, hidden Markov models are widely applied in real world applications such as speech recognition, handwriting recognition, gesture recognition, intrusion detection, speech tagging and bioinformatics.

A hidden Markov model is used to model sequential of observations that can be observed over the time. Also there are underlying state sequences which are not observed, that produce these observations. These states are defined as hidden states. As an example, we can consider rainy, cloudy, stormy and sunny as observations and high, medium and low atmospheric pressure as hidden states which we cannot observe. Figure A.1 shows an example of hidden state and corresponding observation sequence for weather condition of six consecutive days.
A detailed description about the hidden Markov model is included in the appendix A.

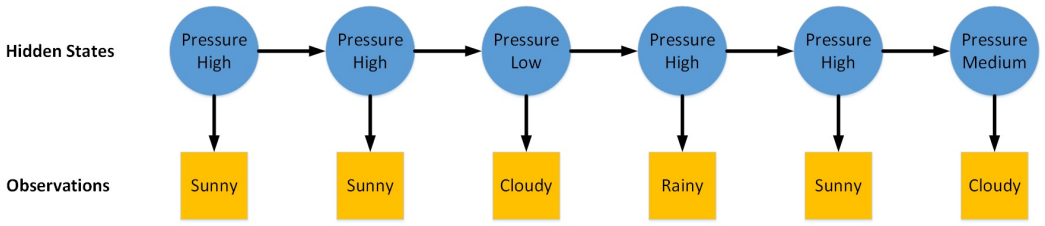


Figure 2.3: An example of hidden state and corresponding observation sequence for weather condition of six consecutive days.

Sequence Modeling Based Intrusion Activity Prediction Research Activities

The following section describes research activities for step prediction of attack based on sequential modeling techniques such as Markov model and hidden Markov model.
Fava et al. [11] proposed alert prediction method based on sequence modeling. Their method is based on Variable Length Markov Model. Markov model assigns a probability to a symbol based on their appearance in the sequence. In N^{th} order Markov model, probability of a symbol appearance depends on "N" number of previous observations. The main design challenge of applying sequence modeling for alert prediction is definition of symbols, because symbols are used to build data sequence. Individual alert has many attributes, therefore it is required to decide which attributes are used and how many attributes are selected? They have selected three alert attributes namely alert category, alert description and destination IP address of an alert as symbols in the Markov model and generated three different Markov models for alert category, alert description and destination IP address. Using these models, they predicted the next alert category, next alert description and destination IP address separately.

They defined prediction accuracy as the percentage of correctly predicted symbols over a total number of predictions. They used three accuracy values as top 1, top 2 and top 3. Top 1 accuracy means accuracy of most probable symbol predicted by the model. Top 2 accuracy means accuracy of occurring one of two most probable symbol predicted by the model as a next symbol. Top 3 accuracy means accuracy of occurrence of one of three most probable symbol predicted by the model as a next symbol. They evaluated their method using an in-house developed data set where 50% of data is used for Markov model training and other 50% is used for the testing. They have achieved 90% top3 accuracy for predicting next alert category and 70% for predicting next alert type.

One main limitation of their method is that they are predicting alert type, alert category and destination IP separately. So there are possibilities that combination of these three may not be a valid combination. As an example, the predicted alert category is "Intrusion Root" and predicted alert description is "ICMP PING NMAP". However, predicted alert description does

not belong to the predicted alert category. To address this limitation, Du et al. [12] presented an ensemble techniques based alert attribute combination method. Based on ensemble techniques, they have a proposed method to convert VMM alert prediction output to a score with respect to the host of the network. This score is used to predict next possible host that can be attacked. Major limitation of their method is that they have only interested to predict next host without predicting attack type.

Kholidy et al. proposed a hidden Markov model based online risk assessment and prediction models [100] with four system states. They defined the four system states as Hale(H): System operation is normal, Investigate (I): Malicious activities are attempted, Attack (A): Intrusion has been started and it is now progressing and Penetrate (P): Intrusion has successfully compromised the system. These four states are defined as hidden states of the system and alerts generated by the IDS are defined as the observation for hidden Markov model. They used DARPA 2000 dataset [69] to simulate their risk prediction algorithm which contains five major attack steps.

They have considered that, if the probability of the penetration state is higher than a threshold of 80% then there is a possibility of an intrusion activity in the future. Using this concept they were able to identify attack steps present in DARPA 2000 dataset before they actually occur.

Chapter 3

Methodology

Alerts generated by intrusion detection systems are used by intrusion response systems to gain knowledge about current and impending intrusions. There is a possibility that intrusion detection systems detect normal activities as intrusions and conversely intrusions as normal activities. These inaccurate detections should be minimized to accomplish better result in intrusion response process. Therefore, it is beneficial to develop accurate and efficient intrusion detection system. Network packet capture contains many attributes such as source IP address, destination IP address, packet frame length, packet data length, etc. In order to develop an efficient and accurate IDS, it is necessary to identify attributes which have a significant impact on identifying network intrusions. Section 3.1 presents the methodology used in identifying important attributes.

The ability to predict an intrusion is a key property for any intrusion response system that aims to block or impede an anticipated intrusion. Alert prediction plays a key role in developing intrusion response system with these capabilities. Sections 3.2 to 3.5 present the methodology used in the proposed alert prediction framework. Section 3.2 describes the overview of proposed alert prediction model. Section 3.3 talks about the alert pre-processing and the alert clustering process. Section 3.4 describes the implementation of hidden Markov model based alert prediction framework. Section 3.5 talks about the software and programming language use for the implementation.

3.1 Machine Learning Techniques for Intrusion Detection on Public Dataset with Feature Reduction

In this section, evaluation of different machine learning techniques based intrusion detection with feature reduction techniques is discussed. With better feature identification, it is possible to develop an efficient intrusion detection system. Feature reduction techniques such as Information Gain (IG) and Chi-Squared statistics (CH) were applied to evaluate intrusion detection performance of OneR, Adaboost, J48 decision tree, Random Forest and Random Tree machine learning techniques with feature reduction.

There are a number of studies that have used older datasets such as KDDCUP 99 [67], NSLKDD [68] and many researchers indicate that these datasets are outdated now [101], [102]. Hence, it is important to evaluate new dataset such as Aegean Wi-Fi Intrusion Dataset (AWID) [73].

3.1.1 Machine Learning Techniques for Intrusion Detection

Machine learning is a process of knowledge discovery of data without being explicitly programmed. Machine learning techniques are becoming popular in last decade and used in many day to day applications such as image recognition, natural language processing, spam detection, intrusion detection, search engine application, fault prediction and stock market analysis. Machine learning techniques can be mainly categorized based on their learning method as a) supervised learning, b) unsupervised learning and c) reinforcement learning.

1. Supervised learning. Learning process is done using labeled data. For a given input data entry desired output is known (output is labeled). Algorithm learns by comparing its output and desired output.

2. Unsupervised learning. This method is used when historical data labels are not available for training (desired output is not available). Algorithm finds structure of input data itself.

3. Reinforcement learning. Algorithm performs various actions to achieve certain goals and learns effectiveness of actions based on rewards received in a dynamic environment

For evaluation of machine learning based intrusion detection capabilities Adaboost, Random Forest, Random Tree, J48(C4.5) and OneR supervised machine learning techniques were used in this thesis.

1. Random Forest. Proper introduction of Random Forest algorithm is done by Leo Breiman in 2001 [103]. Random Forest is multiple decision tree based machine learning algorithm. It generates many classification trees and in order to classify new input, it applies input vector to each of the trees in the forest and then selects class that most of trees produced [103]. Random Forest algorithm has significant advantages such as it runs efficiently on large data sets and it can handle many input variables, furthermore it efficiently handles large percentage of missing data while maintaining accuracy [104].

2. Random Tree. Random Tree is a tree based classification algorithms. It employs a single tree for data classification and it uses randomly selected number of attributes (say K) at each node of the tree for the classification [105].

3. C4.5 (J48). C4.5 is a decision tree based classification algorithm introduced by Ross Quinlan [106]. It is a successor of Iterative Dichotomiser 3 (ID3) algorithm which was also developed by Ross Quinlan [107]. J48 is an open source Java implementation of the C4.5 algorithm. C4.5 algorithm employs normalized information gain to split attributes of the input data.

4. OneR. OneR is a rule based classification algorithm introduced by R.C. Holte in 1993 [108]. It is a very simple classification algorithm based on single rule (1-level decision tree). OneR ranks attributes of training dataset using error rate and selects most informative attribute to develop a rule that predicts class of the data. The OneR algorithm requires discrete attributes. If not, they are discretized.

5. Adaboost. Adaboost is a boosting machine learning algorithm introduced by Yoav Freund and Robert Schapire [109]. Adaboost algorithm is used to improve the performance of other learning algorithms by combining many relatively weak and inaccurate rules

generated by other learning algorithms. These other learning algorithms are referred to as weak learners or base learner components. The weak learner is used to generate a hypothesis for each sample in training dataset. Adaboost combines weak hypotheses generated by weak learners in many rounds to generate improved hypothesis [110].

3.1.2 Feature Reduction Process

In this thesis, Adaboost, Random Forest, Random Tree, J48(C4.5) and OneR supervised machine learning algorithms based intrusion detection techniques were evaluated. For training and testing, Aegean Wi-Fi Intrusion Dataset (AWID) [73] dataset was used. The Aegean Wi-Fi intrusion dataset includes separate datasets for training and testing. The training dataset was used to train machine learning techniques and corresponding testing dataset was used to evaluate performance of machine learning techniques.

Each record of the AWID dataset includes 155 attributes with the class attribute. The class attribute of a record is referred to a type of intrusion or normal network activity that corresponding record is belong to. Each attribute of a record except the class attribute is considered as a feature during the feature reduction process. The features include in the AWID datasets are shown in table B.2.

The experimental setup that was used in this thesis is shown in Figure 3.1. Experiments were conducted in two phases. In the first phase, performance evaluation of OneR, Adaboost, J48 decision tree, Random Forest and Random Tree machine learning techniques were evaluated without applying any feature reduction as shown in Figure 3.1 (a). In the second phase, information gain and chi-squared test feature reduction techniques were applied to identify important features. Based on that less important features were removed from testing and training datasets. These datasets with reduced feature set were used in the second phase to evaluate intrusion detection capabilities of machine learning techniques as shown in Figure 3.1 (b). Waikato Environment for Knowledge Analysis (WEKA) [111] machine learning toolkit was used to perform experiments.

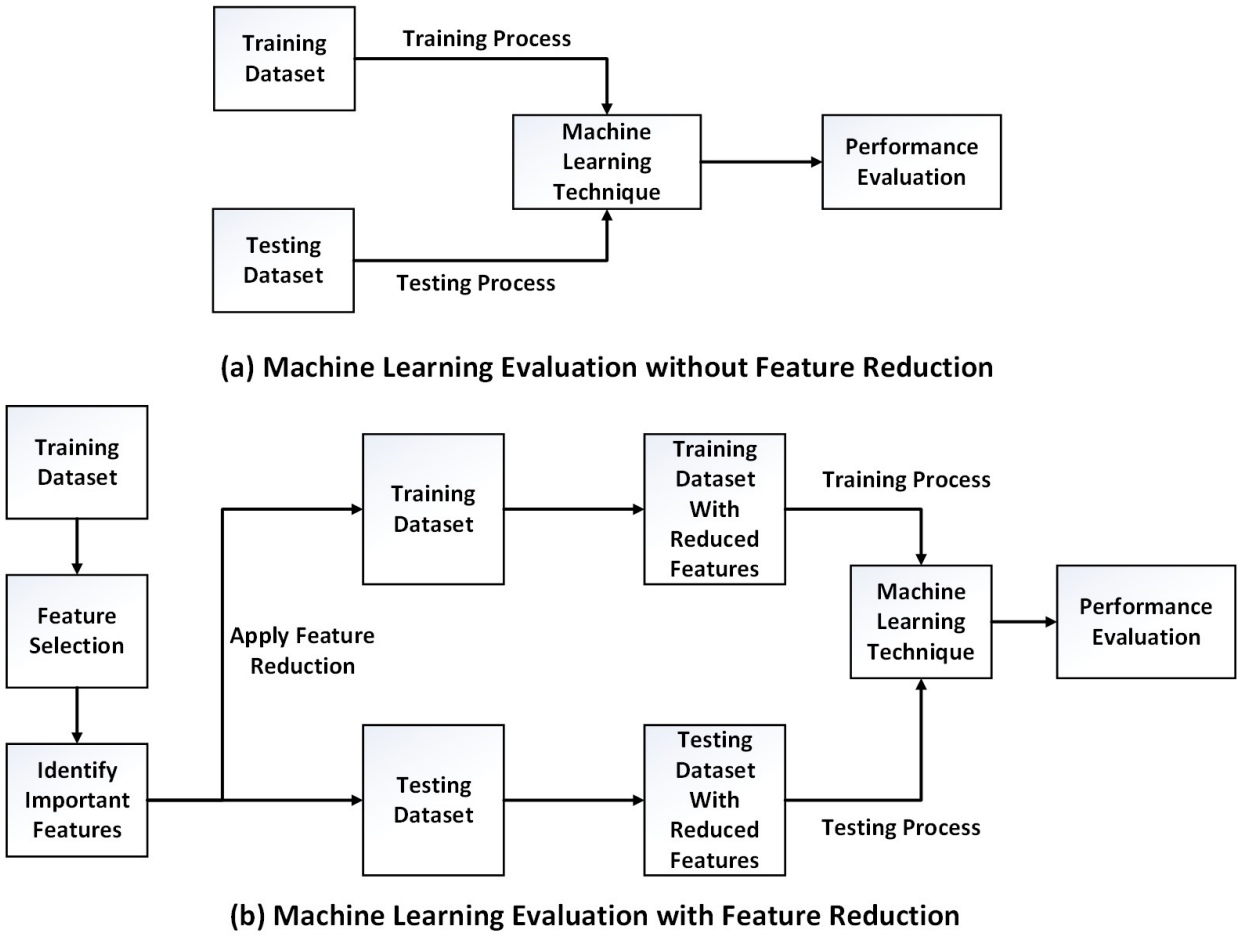


Figure 3.1: Machine learning techniques for intrusion detection with feature reduction.

The following sections describe the implementation of hidden Markov model based alert prediction framework in detail.

3.2 Alert Prediction Module Architecture

One of the objectives of this thesis is to investigate how alerts produced by intrusion detection system is used to understand attack strategies and predict possible future activities of the attacker. The proposed alert prediction module consists of three main modules namely alert pre-processing module, alert clustering module and alert prediction module as shown in Figure 3.2. Output of the alert prediction module is forwarded into intrusion response module. The response module is an external module from the proposed alert prediction framework.

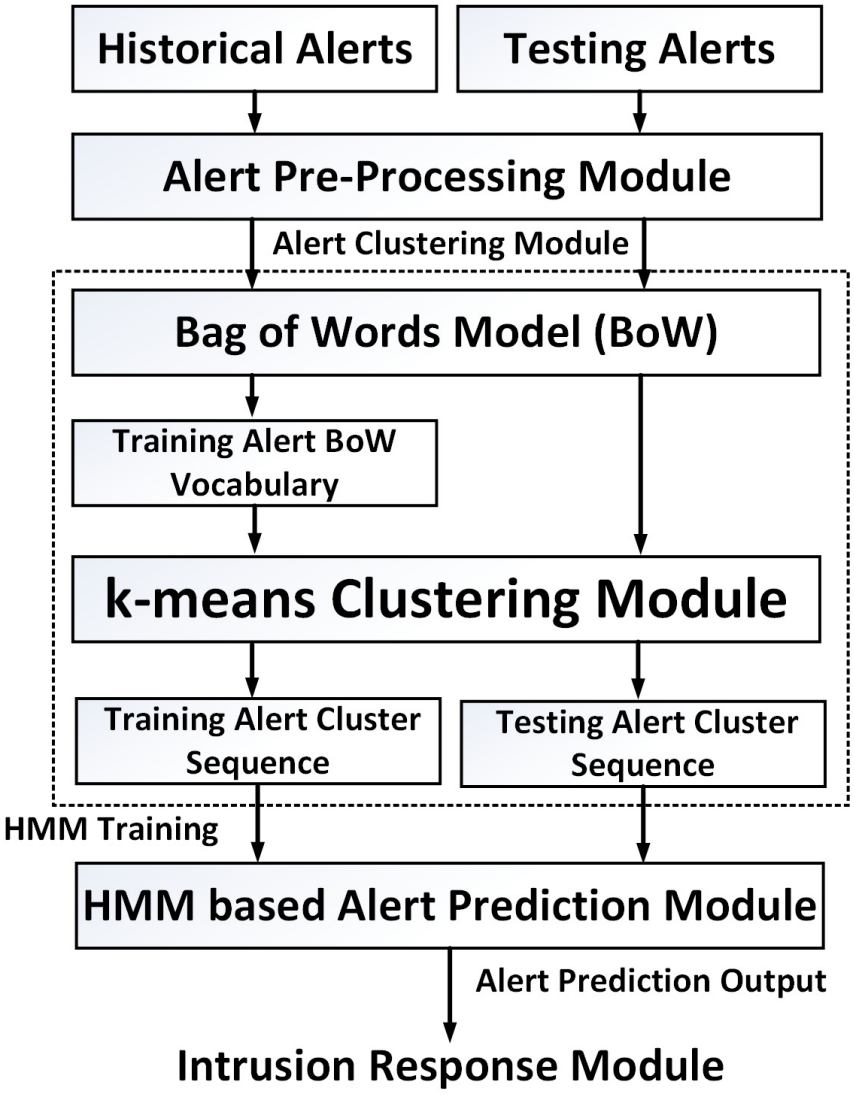


Figure 3.2: Proposed hidden Markov model based alert prediction module.

In this thesis, sequential data modeling concept was employed to predict next possible intrusion

alert for a given historical alert sequence. Hidden Markov model was used as sequential data modeling method. To build a sequential data series in network security domain, alert clustering concept was used. Intrusion alerts were clustered based on their feature similarities and then for a given sequence of alerts, alert clustering module produced a sequence of clusters. Those sequence of clusters were considered to build a sequential data series in our model. The hidden Markov model is used to predict next cluster using the sequence data series which was built in previous step. Since predicted cluster represents certain characteristics of alert attributes, that information can be used by intrusion response module to select and execute relevant response actions to mitigate future intrusions.

3.3 Alert Pre-Processing and Clustering

Intrusion alerts generated by intrusion detection systems (IDSs) was utilized by the proposed alert prediction framework. In this thesis, Snort was used as an intrusion detection system [112]. Snort is an open-source network intrusion detection system (NIDS) [112] which uses a signature based intrusion detection system and it employs rule set for intrusion detection. If the criteria mentioned in a rule is matched, then alert is produced for that condition by Snort. Snort rules consist of two main parts namely rule header and rule option. Rule header consists of a matching criteria together with an action that need to be performed when this criteria is matched [113]. Alerts generated by Snort were stored using a binary format (unified2 file format) and it is not human readable. Therefore, these alerts were converted to human readable format using Barnyard2 [114] interpreter and stored in a MySQL database for further processing.

Few alerts were produced by Snort for the DARPA 2000 dataset is shown in table 3.1. A Snort alert consists two major fields that are used to identify an intrusion which caused the corresponding intrusion alert. These fields are called “alert type” (also known as alert description or alert signature) and “alert category” (also known as alert signature class). Alert type contains detailed information about an intrusion whereas alert category contains higher level information about an intrusion. As an example “UDP Filtered Portsweep” and “UDP

Portsweep" alert types both belongs to "attempted-recon" alert category and the alert type "UDP Filtered Portsweep" provides much more specific information than the alert category "attempted-recon".

3.3.1 Alert Pre-Processing

The first step of the pre-processing was to format raw alerts from different IDSs to a common format and then duplicated alerts were removed using selected features. Timestamp, source IP, source port, destination IP, destination port, alert signature and alert signature class were used to identify duplicate alerts.

3.3.2 Alert Clustering

As described in section 2.3.1 most of the intrusion prediction methods are focused on prediction of next attack category only. However, only predicting next possible attack category is not sufficient to select an efficient response action. If we can predict next possible attack with victim and attacker host information, it would provide a better opportunity to execute a useful response action. The main objective of alert clustering is to group similar alerts together to maximize the information of alerts. Bag of Words (BoW) model, which is a popular concept in text document classification, was used to compare similarities between two alerts.

3.3.3 Bag of Words (BoW) Model

Bag of Words (BoW) model is a popular document analysis algorithm used in text and image classification. Bag of words model is based on following two assumptions.

- Documents are created by repetitively drawing one word from a bag of words which forms the vocabulary,
- Words in the bag may occur multiple times in a document.

Figure 3.3 shows documents generation process using a bag of words vocabulary.

Table 3.1: Sample Snort alerts for the DARPA 2000 dataset.

ID	Timestamp	Source_IP	Source tcp_port	Source udp_port	Destination_IP	Destination tcp_port	Destination udp_port	Signature ID	Signature_name (Alert type)	Signature_class_name (Alert Category)	Signature priority
1	3/7/2000 9:22	172.16.112.100			172.16.115.20			650	portscan: UDP Filtered Portsweep	attempted-recon	2
2	3/7/2000 9:28	172.16.114.50	21		172.16.113.50	1042		40842	ET POLICY FTP Login Successful	misc-activity	3
3	3/7/2000 9:32	172.16.112.50	33361		172.16.113.169	25		506	sensitive_data: sensitive data - eMail addresses	sdf	2
4	3/7/2000 10:12	202.77.162.213			172.16.113.50			655	portscan: UDP Portsweep	attempted-recon	2
5	3/7/2000 10:37	207.25.71.186	80		172.16.117.132	47154		684	http_inspect: SIMPLE REQUEST	unknown	3
6	3/7/2000 11:36	172.16.116.201	23337		206.83.105.134	80		36489	ET MALWARE User-Agent (Win95)	trojan-activity	1
.	.....	.....	....	....	.....	.....	.....	....	.....	.....	....
60	4/16/2000 14:45	207.25.71.200	80		172.16.113.207	2105		672	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown	3
61	4/16/2000 14:45	207.25.71.200			172.16.113.207			504	sensitive_data: sensitive data global threshold exceeded	sdf	2
62	4/16/2000 14:46							863	spp_arpspoof: Directed ARP Request	protocol-command-decode	3
63	4/16/2000 14:46	172.16.112.50	32777		195.115.218.108	25		506	sensitive_data: sensitive data - eMail addresses	sdf	2
64	4/16/2000 15:02	172.16.112.100	21		197.182.91.233	63621		40842	ET POLICY FTP Login Successful	misc-activity	3
65	4/16/2000 15:24	172.16.113.168	23		197.182.91.233	63729		26002	ET ATTACK_RESPONSE Output of id command from HTTP server	bad-unknown	2
66	4/16/2000 15:34	172.16.115.20		53	172.16.114.50		39288	325	PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt	attempted-user	1
67	4/16/2000 15:36	161.203.16.2	80		172.16.113.207	31196		29621	ET INFO PDF Using CCITTFax Filter	bad-unknown	2
68	4/16/2000 15:36	172.16.116.194	31369		207.25.71.141	80		36489	ET MALWARE User-Agent (Win95)	trojan-activity	1
69	4/16/2000 15:43	172.16.115.20			172.16.112.207			655	portscan: UDP Portsweep	attempted-recon	2
70	4/16/2000 15:44	172.16.115.20	32840		172.16.112.50	20		42427	ET POLICY Executable and linking format (ELF) file download	policy-violation	1
71	4/16/2000 15:55	172.16.116.194	22625		207.136.67.35	80		29067	ET INFO Exectuable Download from dotted-quad Host	trojan-activity	1
72	4/16/2000 16:09	172.16.112.100			172.16.115.20			650	portscan: UDP Filtered Portsweep	attempted-recon	2
73	4/16/2000 14:47	172.16.113.148	113		172.16.114.50	17836		587	stream5: Reset outside window	bad-unknown	2
74	4/16/2000 15:26	172.16.114.50	23		172.16.113.168	64792		586	stream5: FIN number is greater than prior FIN	bad-unknown	2
75	4/16/2000 16:06	172.16.115.20	23		202.77.162.213	2401		590	stream5: TCP Small Segment Threshold Exceeded	bad-unknown	2

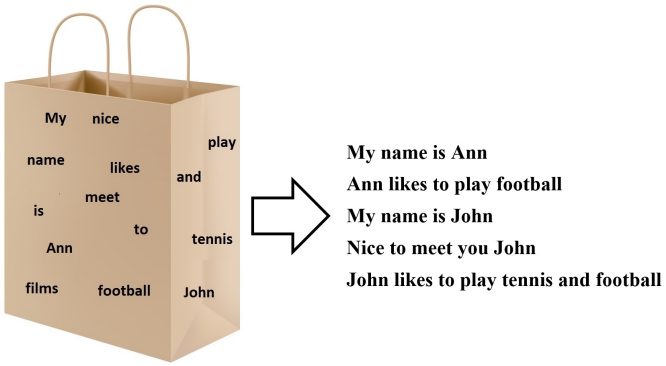


Figure 3.3: Documents Generation Using Bag of Words Model.

Consider following simple documents to understand how we can use BoW concept to compare documents.

Document 1. “Ann likes to play football.”

Document 2. “John likes to play tennis and football.”

Document 3. “Diana likes swimming.”

Vocabulary that used to compare these documents is shown in Figure 3.3. In practice high frequently used terms such as “is”, “are” and “to” are not included in the vocabulary.

BoW model generates a term frequency matrix for each document based on count of words in the vocabulary in each document as shown in table 3.2. Then by comparing term frequency matrix, we can identify similar documents.

Table 3.2: BoW model term frequency for documents.

Documents	**Vocabulary**											
	My	nice	name	likes	meet	Ann	films	football	tennis	john	nice	play
Document 1	0	0	0	1	0	1	0	1	0	0	0	1
Document 2	0	0	0	1	0	0	0	1	1	1	0	1
Document 3	0	0	0	1	0	0	0	0	0	0	0	0

By analyzing the term frequency matrix, it can be observed that document 1 and 2 are similar to each other while document 3 is different.

In above example only one global vocabulary is considered. However, different vocabular-

ies provide an opportunity to group similar documents together more precisely. Figure 3.4 illustrates concept of different BoW model classes. Words related to sports are grouped in a sports BoW class, words related to business are grouped in a business BoW class and words related to mathematics are grouped in a mathematics BoW class. When a new document is received, based on term frequency matrix of each BoW classes most suitable class for that document is identified.

Figure 3.4: Different Bag of Words vocabulary classes.

Consider following example to illustrate this idea further:

Sports vocabulary = { goal, foul, penalty, football, captain, player }

Mathematics vocabulary = { calculus, multiplication, subtraction, division, polynomials, curve }

Document 1. “Ann likes to read books.”

Document 2. “Football match between Brazil and Argentina is drawn. Each team scored 2 goals. Brazil captain scored two goals for their team, Argentina misses one penalty goal chance otherwise, they may have won the game.”

Document 3. “Student should be familiar with basic mathematics operations such as multiplication, subtraction and division. Calculus is part of secondary level education.”

Term frequency matrix of document 1, 2 and 3 for mathematical and sports BoW vocabularies are shown in table 3.3.

Table 3.3: Term frequency matrix of documents for mathematical and sports BoW vocabularies.

Documents	Sport Vocabulary						Mathematics Vocabulary					
	goal	foul	penalty	football	captain	player	calculus	multiplication	subtraction	division	polynomials	curve
Document 1	0	0	0	0	0	0	0	0	0	0	0	0
Document 2	3	0	1	1	1	0	0	0	0	0	0	0
Document 3	0	0	0	0	0	0	1	1	1	1	0	0

By observing term frequency matrices, it can be seen that document 1 does not belong to either sports or mathematics classes. Document 2 belongs to sports class and document 3 is belong to mathematics class.

This bag of words based text document classification concept was used to cluster alerts. Figure 3.5 illustrates alert clustering process. In the first step, important attributes of alert were selected to develop vocabulary. For this thesis, source IP address, source IP port, destination IP address, destination IP port, signature (alert type) and signature class (alert category) were selected to generate vocabulary.

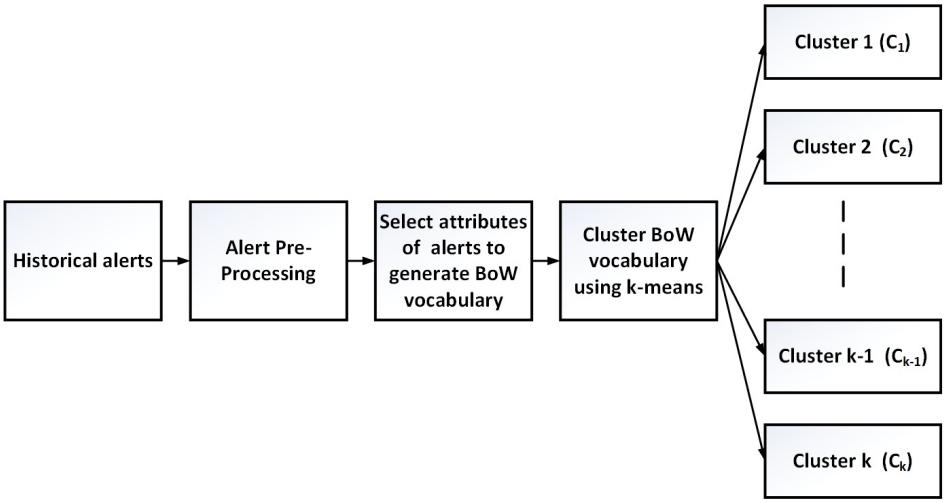


Figure 3.5: Alert clustering using Bag of Words model concept.

The next step is to cluster alerts based on their attributes. For the clustering process, k-means clustering algorithm was used. The k-means algorithm was first used by James MacQueen in 1967 [115]. Consider set of "n" data points X = $\{x_1, x_2, \ldots, x_n\} \in \mathbb{R}$, where each data point is a d-dimensional real vector. K-means clustering algorithm groups these features to k number of

groups (k ≤ n) where S = $\{s_1, s_2, \ldots, s_k\}$ represents the cluster set. Let mean of each clusters be $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}$. The aim of the k-means clustering is to minimize the sum of squared error over all clusters as shown in the equation (3.1) [116].

$$\underset{S}{\operatorname{argmin}} \sum_{k=1}^{k} \sum_{x_i \in s_k} \|x_i - \mu_k\|^2. \tag{3.1}$$

Historical alerts were used to generate set of alerts clusters. Once new alert was received, it was assigned to the best matching cluster using k-means algorithm.

The main objective of clustering process is to generate a sequence of symbols that can be modeled using sequential data modeling. Let k be the number of clusters was generated by k-means algorithm (represent cluster set C = $\{C_1, C_2, ..., C_i, ..., C_k\}$). Furthermore, T number of alerts ordered based on their time of origin (timestamp) was represented as $A_1, A_2, ..., A_i, ..., A_k$. The best matching cluster (from cluster set C) for each alert was assigned by the alert clustering module. Figure 3.6 illustrates this process.

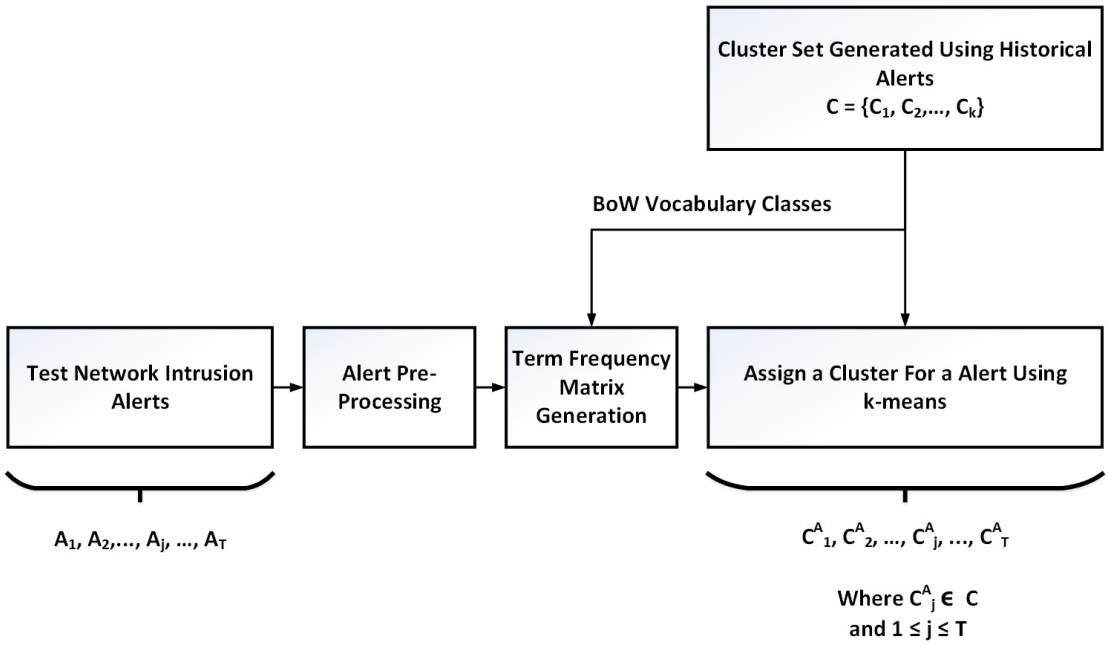


Figure 3.6: Alert clustering using k-means and generating symbol sequence.

3.4 Building Hidden Markov Models

A hidden Markov model was used to model sequential data produced from alert clustering module. As described in hidden Markov model in section 2.3.1, hidden Markov model consist of hidden states which produced the observations. Alert cluster sequence was considered as an observation sequence in this model. Parameters of hidden Markov model is defined as:

1. **N**, Number of hidden states in the HMM. Where individual states are denoted as $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$ and state at time t is denoted as q_t.

2. **M**, Number of distinct observation symbols per state. Where individual symbols are denoted as $\mathbf{V} = \{v_1, v_2, \ldots, v_M\}$.

3. State transition probability (**A**) NxN matrix. Where a_{ij} represents the state transition probability form state i to state j.
   $a_{ij} = P(q_{t+1} = S_j | q_t = S_i), where\ 1 \leq i \leq N and 1 \leq j \leq N.$

4. Observation emission probability (**B**) NxM matrix. Where k^{th} observation emission probability of the state j is represented by $\boldsymbol{b_j(k)}$.
   $\boldsymbol{b_j(k)} = P(v_k\ at\ t | q_t = S_j), where\ 1 \leq j \leq N, 1 \leq k \leq M.$

5. Initial state probability distribution ($\boldsymbol{\pi}$). Where $\boldsymbol{\pi}$ represents the initial states probabilities of the HMM.
   $\boldsymbol{\pi_i} = P(q_1 = S_i), 1 \leq i \leq N.$

6. Observation sequence (**O**). The observation sequence of length T is represented as $\mathbf{O} = O_1, O_2, \ldots, O_t, \ldots, O_T$, Where O_t is one of observation symbol from **V**.
   Hidden Markov model is typically represented by using **A**, **B** and $\boldsymbol{\pi}$ parameters and the model is denoted as $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$.

3.4.1 Hidden Markov Model Training

In this research, first order hidden Markov model was implemented where state transition from one state to next state only depends on the previous state. In our model, observations are distinct

alert cluster IDs generated from alert clustering module. The cluster ID sequence generated by historical alerts was used to train the hidden Markov model. Hidden Markov model was trained using Baum-Welch algorithm [97] as described in appendix A.

Notation:

- **T** = length of training alert sequence.
- **k** = number of clusters generated by k-means clustering module.
- Cluster set generated by k-means $\mathbf{C} = \{C_1, C_2, ..., C_t, ..., C_k\}$.
- Training alert sequence = $A_1, A_2, \ldots, A_t, \ldots, A_T$.
- Corresponding cluster sequence for training alert sequence $\boldsymbol{\theta} = C^A{}_1, C^A{}_2, \ldots, C^A{}_t, \ldots, C^A{}_T$ where $C^A{}_t \in C \quad for \quad 1 \leq t \leq T$.
- Observation sequence $\mathbf{O} = O_1, O_2, \ldots, O_t, \ldots, O_T$. In this model, observed alert cluster sequence is taken as observation sequence.
  Hence, $\mathbf{O} = \boldsymbol{\theta} = C^A{}_1, C^A{}_2, \ldots, C^A{}_t, \ldots, C^A{}_T$.
- The best hidden state sequence matching for the given observation sequence (**O**) is $\mathbf{Q} = q_1, q_2, \ldots, q_t, \ldots, q_T$.
- Initial states probabilities of the HMM ($\boldsymbol{\pi}$) matrix ($1xN$) is defined as:
  $\boldsymbol{\pi_i} \approx 1/N$, for $i = 1, 2, ..., N$.
- Initial state transition probability (**A**) matrix (NxN) is defined as:
  $\boldsymbol{a_{ij}} \approx 1/N$ for $i = 1, 2, ..., N$ and $j = 1, 2, ..., N$.
- Initial observation emission probability (**B**) matrix (NxM) is defined as:
  $\boldsymbol{b_j(k)} \approx 1/M$ for $j = 1, 2, ..., N$ and $k = 1, 2, ..., M$.

Hidden Markov model training process.

Forward probability, $\alpha_t(i)$ is defined as:

$$\boldsymbol{\alpha_t(i)} = P(O_1, O_2, \ldots, O_t, \ldots, O_T, q_t = S_i|\lambda). \tag{3.2}$$

Backward probability (β) is defined as:

$$\boldsymbol{\beta_t(i)} = P(O_{t+1}, \ldots, O_T | q_t = S_i, \lambda). \tag{3.3}$$

Define $\gamma_t(j)$ which represents probability of being in state S_j at time t as:

$$\boldsymbol{\gamma_t(j)} = P(q_t = S_j | O, \lambda); \quad for \quad 0 \leq t \leq T \, and \, 0 \leq j \leq N. \tag{3.4}$$

Probability of transfer from state S_i (at time t) to S_j (at time t+1) is defined as $\xi_t(i, j)$ as:

$$\boldsymbol{\xi_t(i,j)} = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}. \tag{3.5}$$

From the definition of $\gamma_t(i)$ and $\xi_t(i, j)$ we can find the relationship between them as:

$$\boldsymbol{\gamma_t(i)} = \sum_{j=1}^{N} \xi_t(i, j). \tag{3.6}$$

Parameters of hidden Markov model calculation process is expressed as:

Expected value of initial probability $\hat{\pi}_i = \gamma_1(i)$ for $i = 1, 2, ..., N$.

State transition probability matrix(A):

$$\hat{\boldsymbol{a}}_{\boldsymbol{ij}} = \frac{\text{Expected total number of transition from state } S_i \text{ to } S_j}{\text{Expected total number of transition from state } S_i \text{ to any state}}. \tag{3.7}$$

$$\hat{\boldsymbol{a}}_{\boldsymbol{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \; for \, i = 1, 2, ..., N \, and \, j = 1, 2, ..., N. \tag{3.8}$$

Observation Emission probability matrix(B):

$$\hat{\boldsymbol{b}}_{\boldsymbol{j}}(\boldsymbol{k}) = \frac{\text{Expected total number of time in state } S_j \text{ and observing symbol } v_k}{\text{Expected total number of times in state } S_j}. \tag{3.9}$$

$$\hat{\boldsymbol{b}}_{\boldsymbol{j}}(\boldsymbol{k}) = \sum_{\substack{t=1 \\ O_t = v_k}}^{T} \gamma_t(j) \Big/ \sum_{t=1}^{T} \gamma_t(j), \; for \, i = 1, 2, ..., N \, and \, k = 1, 2, ..., M. \tag{3.10}$$

Calculation of probability of observation sequence P($O|\lambda$):

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{3.11}$$

- **HMM parameters re-estimation process:**

HMM parameters calculation process is an optimization problem. The optimal values can be found using standard re-estimation process as shown in below [97].

Step 1. Initialize model with the best guess values for **A**, **B** and π.

Step 2. Compute $\alpha_t(i)$, $\beta_t(i)$, $\xi_t(i,j)$ and $\gamma_t(i)$.

Step 3. Re-estimate the model λ = (**A**, **B**, π).

Step 4. If $P(O|\lambda)$ increases (determined by improvement of **A**, **B** and π) above a given threshold value and then go to step 2; otherwise stop the process.

3.4.2 Hidden Markov Model Based Sequence Prediction

Trained hidden Markov model was used to predict next alert clusters based on observed alert cluster sequence.

Notation:

- Trained HMM λ = (**A**, **B**, π).
- **T** = Length of observed testing alert sequence.
- **k** = Number of clusters generated by k-means clustering module.
- Cluster set generated by k-means **C** = $\{C_1, C_2, ..., C_t, ..., C_k\}$.
- Observed testing alert sequence $\delta = A_1, A_2, \ldots, A_t, \ldots, A_T$.
- Corresponding cluster sequence for observed testing alert sequence $\omega = {C^A}_1, {C^A}_2, \ldots, {C^A}_t, \ldots, {C^A}_T$ where ${C^A}_t \in C \quad for \quad 1 \leq t \leq T$.
- Observation sequence **O** = $O_1, O_2, \ldots, O_t, \ldots, O_T$.

In this model, observed alert cluster sequence was taken as observation sequence.

Hence, $\mathbf{O} = \boldsymbol{\omega} = C^A{}_1, C^A{}_2, \ldots, C^A{}_t, \ldots, C^A{}_T$.

The main goal of this thesis is to predict next alert cluster ($C^A{}_{T+1}$) by assuming that previously generated alert cluster IDs based on attacker activities are available. By predicting next alert cluster, possible actions of the attacker can be identified.

The best hidden state sequence for the observed testing cluster ID sequence ($\boldsymbol{\omega} = C^A{}_1, C^A{}_2, \ldots, C^A{}_t, \ldots, C^A{}_T$) was identified using Viterbi algorithm.

Let $\mathbf{Q} = q_1, q_2, \ldots, q_t, \ldots, q_T$ is the best hidden alert sequence. Using state transition probability matrix ($\mathbf{A}$) and observation emission probability matrix ($\mathbf{B}$), probability of each cluster for the next position is calculated. Based on the probabilities of clusters, the best possible candidate for next cluster is selected. Let last element of $\mathbf{Q}$ was S_j. Probability of cluster C_i to be in next observation can be calculated using equation 3.12:

$$\text{Probability of cluster } C_i \text{ to be in next observation } (P_{T+1}(C_i)) = \sum_{r=1}^{N} a_{jr}\, b_r(i). \quad \text{Where } C_i \in C. \tag{3.12}$$

Using equation 3.12 probability of each cluster to be in next position was calculated. In this model, three possible clusters were identified for next observation based on their probability values. These three were labeled as level 1 prediction, level 2 prediction and level 3 prediction. Level 1 prediction includes cluster which has the highest probability. Level 2 prediction includes highest and second highest probability clusters. Level 3 prediction includes first, second and third highest probability clusters. Algorithm 1 illustrates pseudo code for next possible alert cluster prediction. For multiple step prediction (prediction of many data points in one prediction step), alert cluster which was predicted in previous prediction stage was appended to input observation sequence when predicting next point. Algorithm 2 illustrates pseudo code for multiple step alert cluster prediction.

Algorithm 1: Hidden Markov model based alert prediction.

Input : Parameters of Trained Hidden Markov Model (**A**, **B**, π, **N**, **M**),
Number of clusters = **k**, Where **M** = **k**
Observed testing alert cluster sequence $\omega = C^A{}_1, C^A{}_2, \ldots, C^A{}_j, \ldots, C^A{}_T$
Where $C^A{}_j \in \mathbf{C}\, and\, \mathbf{T} \geq j \geq 1$.

Output : Top three candidates for next possible alert cluster

```
Begin;
/* Finding best hidden sequence for given observation sequence */
Best_Hidden_State_Sequence(Q)= Viterbi Algorithm (A, B, π, ω)
Select last element of Q (say S_j)
Define Array P_cluster[M] = [P_cluster_1, ..., P_cluster_i, ...P_cluster_M]
for i ← 1 to M (number of distinct observations) do
    P ← 0
    for r ← 1 to N (number of hidden states) do
        P ← P + (a_jr x b_r(i))
    end
    P_cluster[i] ← P
end
Level 1 Prediction = Select cluster which has highest probability value from P_cluster
  Array
Level 2 Prediction = Select clusters which have first and second highest probability
  value from P_cluster Array
Level 3 Prediction = Select clusters which have first, second and third highest
  probability value from P_cluster Array
```

Algorithm 2: Hidden Markov model based alert prediction (multiple length).

Input : Parameters of Trained Hidden Markov Model (**A**, **B**, π, N, M),
Number of clusters = **k**, Where **M** = **k**
Observed testing alert cluster sequence $\omega = C^A{}_1, C^A{}_2, \ldots, C^A{}_j, \ldots, C^A{}_T$
Where $C^A{}_j \in \mathbf{C}$ *and* $\mathbf{T} \geq j \geq 1$. l (length of the prediction)

Output : Top three candidates for next possible alert cluster

1 Begin:
2 $counter \leftarrow 0$
3 $i \leftarrow 0$
4 **if** $counter == 0$ **then**
5 Input_cluster_ID_sequence = ω
6 **else**
7 Input_cluster_ID_sequence = New_alert_cluster_ID_sequence
8 /* Finding Level 1, 2, 3 Prediction using Algorithm 1 */
9 Level 1, 2, 3 Prediction = Algorithm1(Input_cluster_ID_sequence)
10 Output: Level 1 Prediction, Level 2 Prediction and Level 3 Prediction
11 $i \leftarrow i + 1$
12 $C^A{}_{T+i} \leftarrow$ Level 1 Prediction
13 $counter \leftarrow counter + 1$
14 ω = Append $C^A{}_{T+i}$ to ω
15 New_alert_cluster_ID_sequence = ω
16 **if** $counter \geq l$, **then**
17 stop.
18 **else**
19 Go to Line 4

3.5 Programming Language and Software tools

Alert pre-processing and prediction module was implemented using python programming language. Snort intrusion detection system (version 2.9.8.0 with rule set 2980) was used for intrusion alert generation. Alerts generated by the Snort were converted to human readable format using Barnyard2 [114] interpreter. MYSQL database was used as a database for the platform. Bag of words model implementation and k-means clustering implementation was done using python scikit-learn library [117]. Hidden Markov model was implemented using python programming language.

Chapter 4

Results

Experiments were performed in two phases. In the first phase, performance evaluation of OneR, Adaboost, J48 decision tree, Random Forest and Random Tree machine learning techniques based intrusion detection with feature reduction was conducted. Section 4.1 describes feature reduction process and performance evaluation of machine learning techniques based intrusion detection methods with feature reduction in detail.

In the second phase, performance evaluation of hidden Markov model based alert prediction framework was conducted. Experiments were mainly focused on identifying the effect of the number of clusters and HMM parameters on the accuracy of alert prediction. For the purpose of comparison with other intrusion prediction research activities, both alert cluster based prediction and alert category based prediction were evaluated. Section 4.2.1 describes the experimental process. Section 4.3 describes the performance evaluation of the alert prediction framework. Also, it talks about IP address and cluster distribution of generated alerts. Section 4.4 describes about the alert clusters formed by the proposed alert prediction framework.

4.1 Machine Learning Techniques for Intrusion Detection on Public Dataset with Feature Reduction

Intrusion detection capabilities of machine learning techniques namely OneR, Adaboost, J48 decision tree, Random Forest and Random Tree with feature reduction were evaluated. The publicly available Aegean Wi-Fi Intrusion Dataset (AWID) [73] was used for the evaluation. The Aegean Wi-Fi Intrusion Dataset (AWID) [73] is selected for the evaluation of different machine learning techniques based intrusion detection with feature reduction techniques. The main reason to select AWID dataset for this thesis is each record of the AWID dataset contains 155 attributes. This higher number of attributes provides opportunity to analyze impact on different level of attribute reduction to the intrusion detection. Also, this dataset contains wide range of intrusion types that can occur in an 802.11 Wi-Fi network and it provides opportunity to analyze the impact of attribute reduction on the detection of wide range of intrusions types. Filter-based feature reduction techniques namely Information Gain and Chi Squared test were used for the feature reduction process.

4.1.1 Aegean Wi-Fi Intrusion Dataset (AWID)

The Aegean Wi-Fi Intrusion Dataset (AWID) is a publicly available labeled dataset which was developed based on real traces of both normal and intrusion activities of an 802.11 Wi-Fi network under the supervision of University of the Aegean and George Mason University [73]. Single Access Point (AP) based network with WEP encryption was used to generate the AWID dataset. In order to maintain the diversity of network devices laptops, mobile devices, desktop computers, tablet devices and smart TVs were used in the AWID network setup. The main limitation of this dataset is the fact that it is a simulated attack instead of a real attack. All intrusions were generated by a laptop running Kali Linux 1.0.6 operating system. Therefore, the AWID dataset may not represent the behavior of a real network which contains actual users and attackers.

The AWID dataset consists of large and reduced datasets. It includes separate datasets for

training (denoted as Trn) and testing (denoted as Tst). Each record of the dataset is classified as either normal or a specific intrusion type (i.e., class attribute of a record is referred to a type of intrusion or normal network activity). The AWID datasets are mainly classified into two types based on their class distribution as high-level labeled dataset (AWID-CLS) and finer grained labeled dataset (AWID-ATK). The class distribution of AWID datasets are shown in table 4.1 [73]. The record distribution of AWID datasets are shown in table 4.2.

Table 4.1: Aegean Wi-Fi Intrusion Dataset (AWID) class distribution [73].

Dataset	**Class Lables**
AWID-CLS-F-Trn **AWID-CLS-F-Tst** **AWID-CLS-R-Trn** **AWID-CLS-R-Tst**	Flooding, Impersonation, Injection, Normal.
AWID-ATK-F-Trn **AWID-ATK-R-Trn**	Amok, Arp, Authentication request, Beacon, Cafe latte, Deauthentication, Evil twin, Fragmentation, Probe response, Normal.
AWID-ATK-F-Tst	Amok, Arp, Authentication request, Beacon, Cafe latte, Chop chop, Cts, Deauthentication, Disassociation, Evil twin, Fragmentation, Hirte, Power saving, Probe request, Probe response, Rts, Normal.
AWID-ATK-R-Tst	Amok, Arp, Beacon, Cafe latte, Chop chop, Cts, Deauthentication, Disassociation, Evil twin, Rts Fragmentation, Hirte, Power saving, Probe request, Normal.

A Description about AWID class labels presented in table 4.1 is included in table B.1.

Table 4.2: Aegean Wi-Fi Intrusion Dataset (AWID) record distribution [73].

Dataset	Classes	Type	Records	Normal Records %
AWID-ATK-F-Trn(full)	10	Train	162,375,247	97.151
AWID-ATK-F-Tst(full)	17	Test	48,524,866	97.528
AWID-CLS-F-Trn(full)	4	Train	162,375,247	97.151
AWID-CLS-F-Tst(full)	4	Test	48,524,866	97.528
AWID-ATK-R-Trn (reduced set)	10	Train	1,795,575	90.956
AWID-ATK-R-Tst (reduced set)	15	Test	575,643	92.207
AWID-CLS-R-Trn (reduced set)	4	Train	1,795,575	90.956
AWID-CLS-R-Tst (reduced set)	4	Test	530,643	92.207

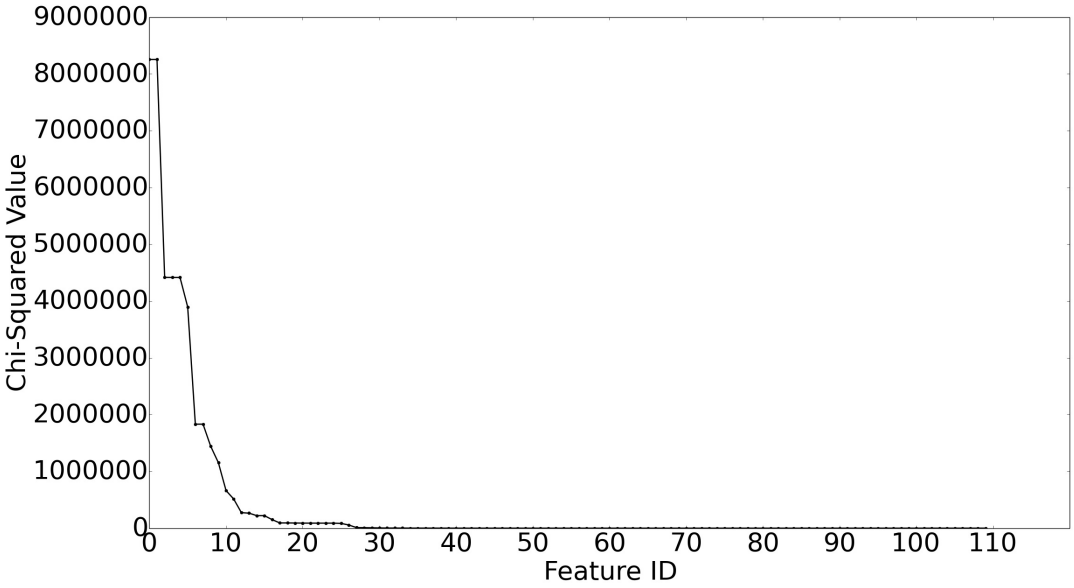
AWID reduced datasets were chosen for evaluation of machine learning techniques based intrusion detection with feature reduction techniques. Waikato Environment for Knowledge Analysis (WEKA) [111] was used as machine learning tool kit for the experiments. AWID-CLS-R-Trn, AWID-CLS-R-Tst, AWID-ATK-R-Trn and AWID-ATK-R-Tst datasets were selected for the experimental process. In order to reduce pre-processing complexity, the string attributes were removed from the datasets and 111 attributes out of 155 were selected for the experiments. Each attribute of a record of the AWID dataset except the class attribute is considered as a feature during the feature reduction process. The features included in the AWID datasets are shown in table B.2 and 111 features selected for the experiments are shown in table B.3. Each dataset of AWID datasets consists of separate dataset for training and testing. Training dataset was used to train the machine learning techniques and relevant testing dataset was used to evaluate performance of the machine learning techniques.

4.1.2 Evaluation of Feature Reduction Techniques

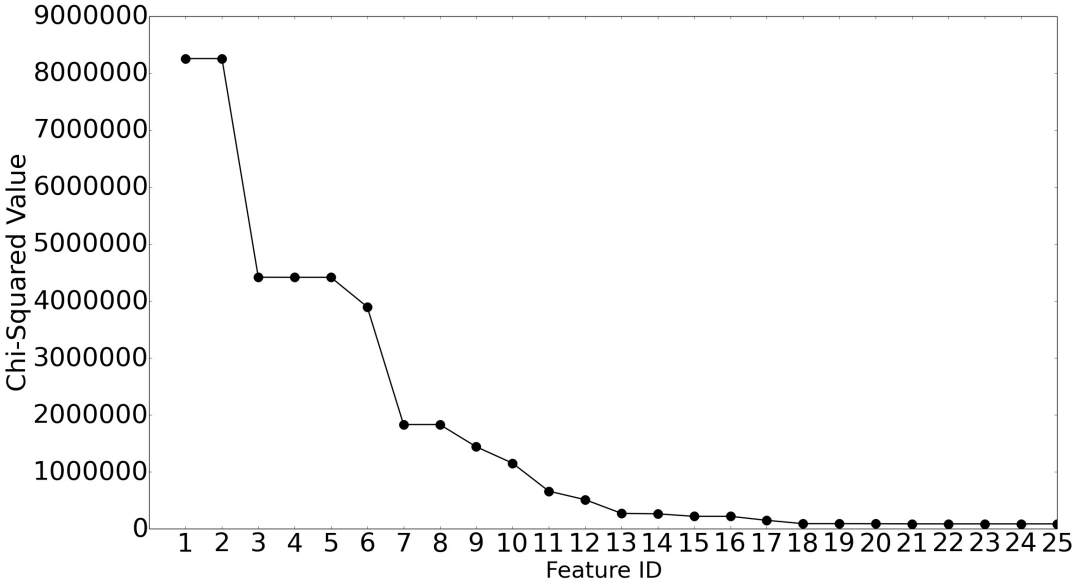
Absolute feature rank does not purely reflect influence of that feature to the final result. There can be some situations where all features may have higher or lower rank values without having a much variation. Also, rank of features may have continuous decline phase. Both these conditions do not highly influence to the final result [118]. The important fact that we can consider

is the decline of rank value of features. If we can observe a sharp decline, then we can identify the regions of features which are highly contributed to the final result [118].

To observe information gain and chi-squared statistic variation, features were sorted in descending order based on their information gain and chi-squared statistic values. Then feature IDs and rank of features was plotted with respect to each other as shown in Figure 4.1a, Figure 4.1b, Figure 4.2a and Figure 4.2b. It can be observed that, both information gain and chi-squared statistics values become zero after 41 features. As shown in Figures 4.1b and Figure 4.2b, sharp reduction of information gain after top ranked 5, 8, 10 and 15 features were observed.

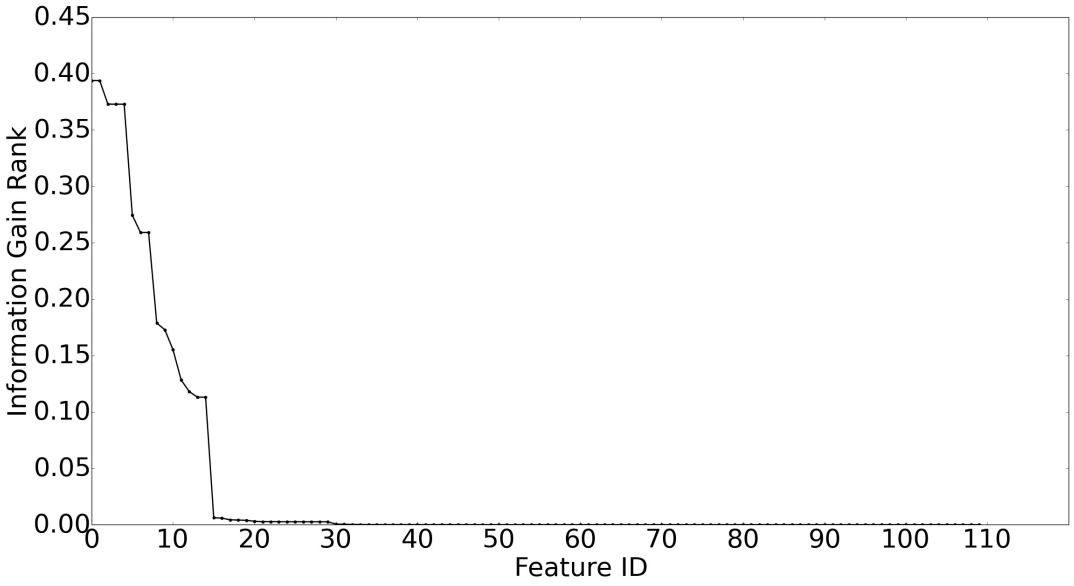


(a) Chi-squared feature evaluation for AWID-ATK-R dataset for 111 features.

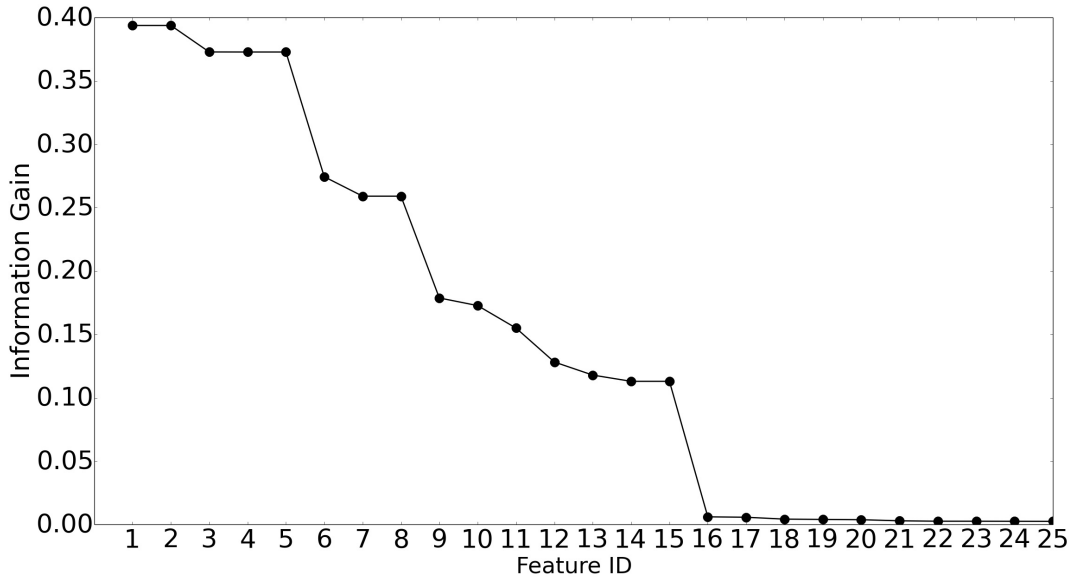


(b) Chi-squared feature evaluation of AWID-ATK-R dataset for top 25 features.

Figure 4.1: Chi-squared feature evaluation for AWID-ATK-R dataset.



(a) Information Gain feature evaluation of AWID-ATK-R-Trn dataset for 111 features.



(b) Information Gain feature evaluation of AWID-ATK-R-Trn for top 25 features.

Figure 4.2: Information Gain feature evaluation of AWID-ATK-R-Trn dataset.

4.1.3 Intrusion Detection Performance Evaluation with Feature Reduction Techniques

Based on feature ranking values, two major feature set regions were identified. Those were segments of 41 features and 10 features. Initially, evaluation was done without any feature

selection to evaluate intrusion detection performance of the machine learning techniques. In the second experiment, segments of 41 features were selected using information gain feature reduction technique. In the third experiment, segments of 10 features were selected using chi-squared statistic feature reduction technique. Accuracies of classification of Adaboost, J48, OneR, Random Forest and Random Tree machine learning techniques were recorded in each experiment. The experiments process is illustrated in Figure 4.3. The configurations of the machine learning techniques are illustrated in table 4.3. Those configurations were kept fixed during the experiments.

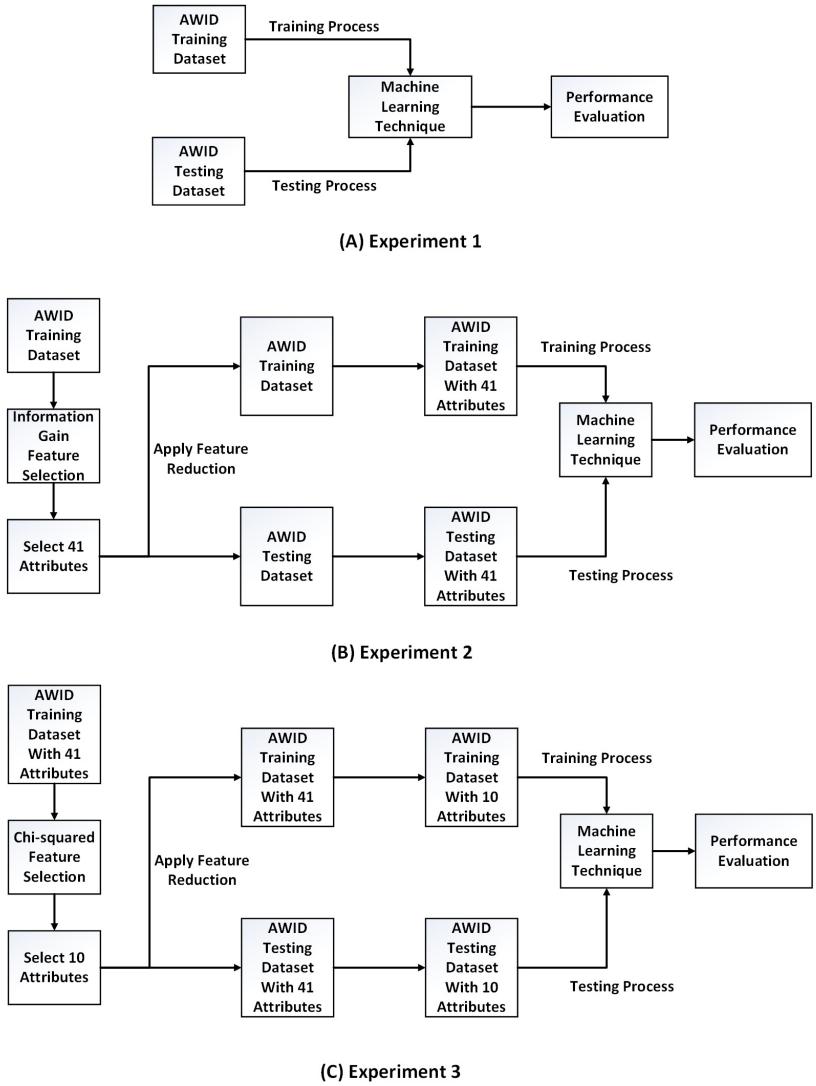


Figure 4.3: Intrusion detection performance evaluation with feature reduction techniques.

Table 4.3: Machine learning techniques configurations.

Machine Learning Technique	Parameters
Random Forest	Maximum depth of the tree = Unlimited, Number of trees = 10
Random Tree	Number of randomly chosen attributes (Kvalue) = log_2 x (number of attributes) + 1 Maximum depth of the tree = Unlimited
J48 (C4.5)	Confidence Factor = 0.25, Minimum number of instances per leaf = 2
Adaboost	Base classifier = Decision Stump Tree, Number of iterations = 10
OneR	Minimum bucket size used for discretizing numeric attributes = 6

The performance of machine learning techniques with 111, 41 and 10 features are listed in tables 4.4, 4.5 and 4.6 respectively. Over 90 % classification accuracy was achieved by the all five algorithms. The maximum accuracy of 95.12% was achieved by Random Tree algorithm for high-level labeled data with 41 features. The maximum accuracy of 94.97% was achieved by Random Forest algorithm for finer grained labeled with 41 features. It was observed that, accuracy of the classification was increased by the maximum of 2.4% for Random Tree algorithm and 1.8% for Random Forest algorithm for high-level labeled dataset and finer grained labeled dataset respectively with respect to feature reduction from 111 features to 41 features. However, accuracy of the classification was decreased by 1.44% for Random Forest algorithm and 2.24% for Random Tree algorithm for high-level labeled dataset and finer grained labeled dataset respectively.

Table 4.4: Machine learning evaluation of AWID with 111 features.

Dataset	**AWID-CLS-R-Trn and AWID-CLS-R-Tst Dataset (high-level class distribution)**					**AWID-ATK-R-Trn and AWID-ATK-R-Tst Dataset (finer grained class distribution)**				
Machine Learning Technique	**OneR**	**J48**	**Random Forest**	**Random Tree**	**Ada Boost**	**OneR**	**J48**	**Random Forest**	**Random Tree**	**Ada Boost**
Correctly Classified%	92.17	94.39	94.83	92.72	91.85	92.07	94.37	93.21	94.58	91.80
Incorrectly Classified%	7.83	5.61	5.17	7.28	8.15	7.93	5.63	6.79	5.42	8.20
TP Rate	0.922	0.944	0.948	0.927	0.918	0.921	0.944	0.932	0.946	0.918
FP Rate	0.898	0.141	0.445	0.616	0.255	0.898	0.117	0.735	0.418	0.254
Precision	0.861	0.969	0.942	0.876	0.909	0.853	0.944	0.894	0.92	0.907
F-Measure	0.888	0.938	0.933	0.901	0.91	0.885	0.944	0.906	0.93	0.908
ROC Area	0.512	0.884	0.974	0.954	0.946	0.512	0.915	0.962	0.964	0.932
Time	25.3	222.9	734.0	116.6	486.9	25.2	198.2	849.4	93.3	278.3

Table 4.5: Machine learning evaluation of AWID with 41 features.

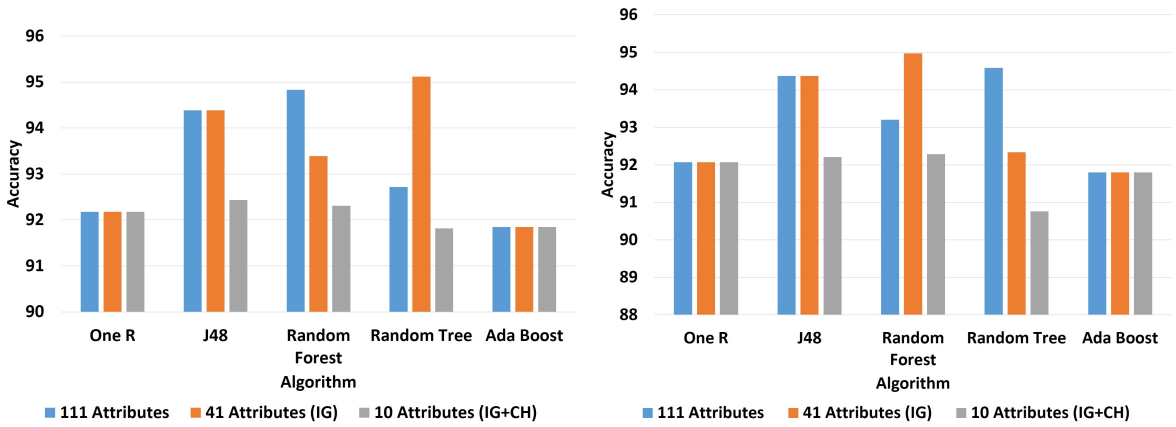
Feature Selection Method :- Information Gain Feature Evaluation										
Dataset	**AWID-CLS-R-Trn and AWID-CLS-R-Tst Dataset (high-level class distribution)**					**AWID-ATK-R-Trn and AWID-ATK-R-Tst Dataset (finer grained class distribution)**				
Machine Learning Technique	**OneR**	**J48**	**Random Forest**	**Random Tree**	**Ada Boost**	**OneR**	**J48**	**Random Forest**	**Random Tree**	**Ada Boost**
Correctly Classified%	92.17	94.39	93.39	95.12	91.85	92.07	94.37	94.97	92.34	91.80
Incorrectly Classified%	7.83	5.61	6.61	4.88	8.15	7.93	5.63	5.03	7.66	8.20
TP Rate	0.922	0.944	0.934	0.951	0.918	0.921	0.944	0.95	0.923	0.918
FP Rate	0.898	0.141	0.646	0.538	0.255	0.898	0.117	0.573	0.826	0.254
Precision	0.861	0.969	0.928	0.91	0.909	0.853	0.944	0.907	0.864	0.907
F-Measure	0.888	0.938	0.912	0.93	0.91	0.885	0.944	0.927	0.891	0.908
ROC Area	0.512	0.884	0.957	0.704	0.946	0.512	0.915	0.967	0.694	0.932
Time	14.0	188.8	353.4	58.3	305.3	15.4	160.9	280.3	26.7	232.3

Table 4.6: Machine learning evaluation of AWID with 10 features.

Feature Selection Method :- Information Gain Feature Evaluation and Chi-Square Feature Evaluation										
Dataset	**AWID-CLS-R-Trn and AWID-CLS-R-Tst Dataset (high-level class distribution)**					**AWID-ATK-R-Trn and AWID-ATK-R-Tst Dataset (finer grained class distribution)**				
Machine Learning Technique	**OneR**	**J48**	**Random Forest**	**Random Tree**	**Ada Boost**	**OneR**	**J48**	**Random Forest**	**Random Tree**	**Ada Boost**
Correctly Classified%	92.17	92.44	92.31	91.82	91.85	92.07	92.21	92.29	90.76	91.80
Incorrectly Classified%	7.83	7.56	7.69	8.18	8.15	7.93	7.79	7.71	9.24	8.20
TP Rate	0.922	0.924	0.923	0.918	0.918	0.921	0.922	0.923	0.908	0.918
FP Rate	0.898	0.888	0.893	0.791	0.255	0.898	0.922	0.899	0.915	0.254
Precision	0.861	0.909	0.895	0.893	0.909	0.853	0.85	0.872	0.85	0.907
F-Measure	0.888	0.89	0.889	0.896	0.91	0.885	0.885	0.887	0.878	0.908
ROC Area	0.512	0.84	0.936	0.578	0.913	0.512	0.814	0.926	0.535	0.932
Time	5.14	78.71	151.86	19.03	48.23	4.7	111.25	182.71	24.67	44.85

When number of features was reduced to 10, accuracy was decreased by the maximum of 2.5% for Random Forest algorithm and maximum of 3.8% for Random Tree algorithm for high-level labeled dataset and finer grained labeled dataset respectively. In the case of feature reduction from 111 features to 41 features and 111 features to 10 features, processing times were significantly reduced. The maximum processing time reduction of 51.85% was achieved by Random Forest algorithm for high-level labeled dataset with feature reduction from 111

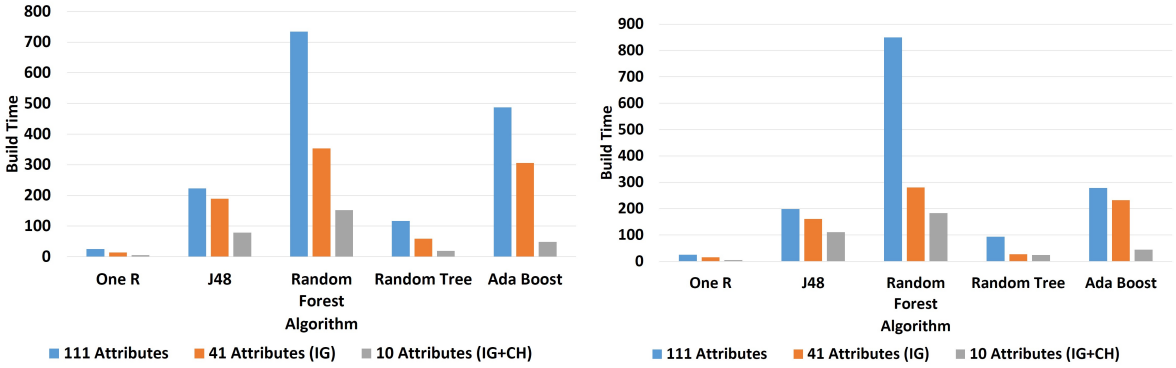
features to 41 features. The maximum processing time reduction of 79.70% was achieved by OneR algorithm for high-level labeled dataset with feature reduction from 111 features to 10 features. For finer grained labeled dataset, the maximum processing time reduction of 71.37% and 83.88% was achieved by Random Tree algorithm and Adaboost algorithm respectively. Accuracy of the classification and processing time is shown in Figures 4.4a, 4.4b, 4.5a and 4.5b respectively.



(a) Correctly classified % of high-level class distribution dataset.

(b) Correctly classified % of finer grained class distribution dataset.

Figure 4.4: Correctly classified % of AWID dataset.



(a) Build time of AWID dataset with high-level class distribution.

(b) Build time of AWID dataset with finer grained class distribution.

Figure 4.5: Build time of AWID dataset.

The AWID dataset was mainly divided in to two types based on its class distribution namely high-level labeling and finer grained labeling. It can be observed that with increase of classes, accuracy of the classification is slightly reduced. The maximum reduction of 2.78% for was

recorded for Random Tree algorithm.

The area under ROC curve (AUC) is a better measurement to compare performance of classifiers whereas a classifier with a higher AUC is better than a classifier with lower AUC. Very low AUC value was recorded for OneR algorithm compare to all the other algorithms in every cases. Over 0.9 values of AUC were achieved by Random Forest and Adaboost algorithms in all the experiments.

Nine common features were selected by both Information Gain feature evaluation (IG) and Chi-Square feature evaluation (CH) based on their ranked value when they were applied separately to the original datasets. These features are presented in table 4.7. Description about these features can be found in [119].

Table 4.7: Top 10 ranked features selection based on Information Gain and Chi-Square feature evaluation.

Rank	**Information Gain Feature Evaluation**	**Chi-Square Feature Evaluation**
1	Frame length on the wire (frame.len)	Frame length stored into the capture file (frame.cap.len)
2	Frame length stored into the capture file (frame.cap.len)	Frame length on the wire (frame.len)
3	MAC timestamp (radiotap.mactime)	MAC timestamp (radiotap.mactime)
4	Time since reference or first frame (frame.time.relative)	Time since reference or first frame (frame.time.relative)
5	Epoch Time (frame.time.epoch)	Epoch Time (frame.time.epoch)
6	Sequence number (wlan.seq)	Sequence number (wlan.seq)
7	Time delta from previous captured frame (frame.time.delta)	Time delta from previous captured frame (frame.time.delta)
8	Time delta from previous displayed frame (frame.time.delta.displayed)	Time delta from previous displayed frame (frame.time.delta.displayed)
9	Data Length (data.len)	Data Length (data.len)
10	Duration (wlan.duration)	Fragment number (wlan.frag)

4.1.4 Intrusion Detection Performance Evaluation with Different Levels of Feature Reduction

To further analyze the impact of features to the classification accuracy, the method employed by Gabrilovich et al was followed [118]. Features were sorted in descending order based on their information gain rank and chi-squared values. Then features with less rank value (i.e. referred as less informative features) were eliminated as percentages of 95%, 90%, 80%, 70%,..., 10% and 0% and then classification accuracies were recorded. By considering accuracy of intrusion detection in experiments conducted so far, Random Forest and J48 algorithms were selected for classification accuracy comparison. The classification accuracy of intrusion detection of Random Forest and J48 algorithms with different levels of feature reduction are illustrated in Figure 4.6, Figure 4.7 and table 4.8 respectively.

Table 4.8: Correctly classified % of finer grained class distribution dataset (AWID-ATK-R) with different feature reduction levels using information gain and chi-squared statistic.

Percentage of Less Informative	**Correctly Classified % Finer Grained Class Distribution Dataset(AWID-ATK-R)**			
Features Reduction	**Random Forest (CH)**	**J48 (CH)**	**Random Forest (IG)**	**J48 (IG)**
0	93.2055	94.3673	93.2055	94.3673
10	93.5934	94.3673	93.5934	94.3673
20	92.7743	94.3673	92.7743	94.3673
30	94.6409	94.3673	94.6409	94.3673
40	92.9713	94.3673	92.9835	94.3673
50	93.4428	94.3673	93.4428	94.3673
60	94.8355	94.3673	92.7625	94.3673
70	92.8158	94.3654	95.0763	94.3673
80	95.1640	94.3654	93.4867	94.3654
90	94.8633	94.4498	94.1467	94.4498
95	92.2606	92.2898	92.2606	92.2898

For J48 algorithm, with the increment of less informative features reduction percentage, accuracy of classification was remained almost the same at 94.36% until 80% of less informative features reduction level. Accuracy of classification was slightly increased by 0.09% at 90% of less informative feature reduction level and then decreased by 2.16% at 95% of less informative feature reduction level for both information gain and chi-squared statistic feature selection

methods.

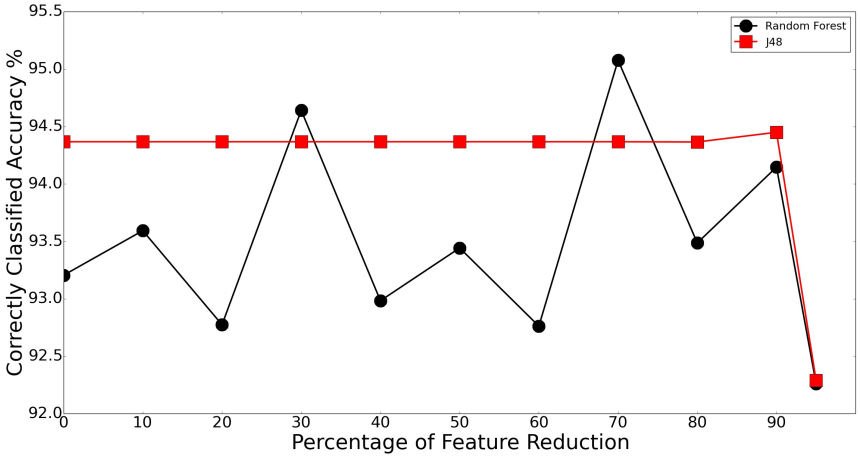


Figure 4.6: Correctly classified % of finer grained class distribution dataset(AWID-ATK-R) with different feature reduction levels using information gain feature selection method.

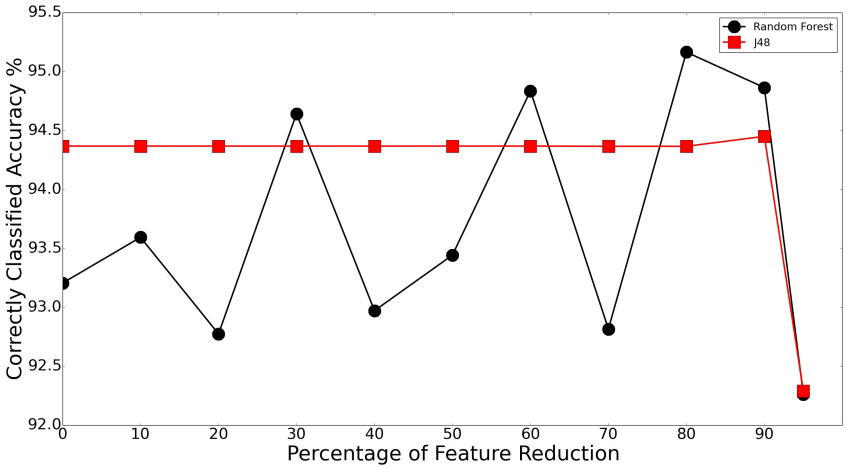


Figure 4.7: Correctly classified % of finer grained class distribution dataset(AWID-ATK-R) with different feature reduction levels using chi-squared statistic feature selection method.

For Random Forest algorithm, smooth transition of accuracy with the increment of less informative features reduction was not observed. With the increment of less informative features reduction percentage, the maximum accuracy of classification 95.0763% was recorded at 70% of feature reduction level for information gain and 95.1640% was recorded at 80% of feature reduction level for chi-squared statistics. With this results it was observed that, feature reduction improves the classification accuracy with proper selection of feature reduction level.

4.1.5 Summary

It was observed that, processing times were significantly decreased in the range of 15.29% to 51.85% and 16.54% to 71.37% with a maximum accuracy increment of 2.4% and 1.8% for high-level labeled dataset and finer grained labeled dataset respectively with respect to feature reduction from 111 features to 41 features. For feature reduction from 111 features to 10 features, processing times were significantly decreased in the range of 27.46% to 79.70% and 43.87% to 83.88% with the reduction of accuracy of the classification in the range of 0 to 2.5% and 0 to 3.8% for high-level labeled dataset and finer grained labeled dataset respectively.

Intrusion datasets are used to train, test and evaluate intrusion detection systems. It is challenging to identify relevant features that have a significant impact on the accuracy of intrusion detection. With a better feature identification, it is possible to develop an efficient IDS. The experimental results indicate that the feature reduction can improve the detection accuracy and the classification speed. However, simply removing low ranked features do not necessarily improve the classification accuracy. Different feature reduction levels should be inspected to identify the best feature reduction level. Frequent evaluations of datasets are important to identify weakness of datasets which helps in developing new datasets with better utility. Network intrusions are growing rapidly and new attack vectors are continuously being developed. Therefore, it is very challenging to generate a dataset that includes most of new intrusion types. Hence continuous evaluation and development of datasets are very important.

4.2 Hidden Markov Model Based Alert Prediction

In second phase of experiments, performance evaluation of hidden Markov model based alert prediction was conducted. Experiments were mainly focused on identifying the effect of the number of clusters and HMM parameters on the accuracy of alert prediction.

4.2.1 Experimental Setup

The DARPA (Defense Advanced Research Projects Agency) [69] dataset was used in the experimental process. The main reason to select DARPA as a dataset because it consists of two sets of multi-stage attack scenarios, which are quite similar to each other. As described in the chapter 3, Snort was used as an intrusion detection system. TCPdump files of the DARPA datasets were sent to Snort IDS to generate alerts. Alerts generated by Snort were sorted based on their time stamp and then sent into the alert pre-processing module followed by the alert clustering module. The corresponding cluster sequence for the input alerts sequence was generated by alert clustering module. Alert cluster sequential series was then sent into the hidden Markov model which was responsible for predicting future intrusion activities.

4.2.2 DARPA Intrusion Dataset

The DARPA data set includes three datasets: DARPA1998, DARPA1999 and DARPA2000. DARPA1998 and DARPA1999 data sets are related to intrusion detection evaluation. DARPA2000 data set was generated based on the specific scenarios. Two main segments namely DMZ Hosts and inside hosts were included in the network which was used to generate DARPA2000 dataset. DMZ consists of one IP segment 172.16.114.0/24 and inside hosts consist of five IP segments (172.16.112.0/24, 172.16.115.0/24, 172.16.116.0/24, 172.16.117.0/24 and 172.16.118.0/24). This dataset consists of data related to a complex network attack with several phases. Specifically, it consists of two multi-stage attacks labeled LLDOS 1.0 and LLDOS 2.0.2. Both these attacks have five main stages [69].

LLDOS 1.0 - Scenario One.

- Attacker probes the network.
- Detect node running “sadmin” demon.
- Attacker breaks in to a host by exploiting the Solaris “sadmind” vulnerability.
- Attacker installs trojan “mstream” DDoS software.
- Attacker launches a DDoS attack at an off site server.

LLDOS 2.0.2 - Scenario Two.

- Attacker probes public DNS server.
- Attacker breaks in to a host by exploiting the Solaris “sadmind” vulnerability.
- Attacker installs DDoS software and scripts via FTP.
- Attacker compromised other two hosts in the network.
- Attacker launches a DDoS attack using compromised hosts.

DARPA 2000 raw network packets were sent into Snort intrusion detection system to generate alerts. 11,264 and 10,468 raw alerts were generated by DARPA 2000 LLDOS 1.0 and DARPA 2000 LLDOS 2.0.2 respectively. These alerts were then sent into the alert pre-processing module. Redundant alerts with same attributes were filtered by the alert pre-processing module. After this process the total number of alerts were reduced to 5113 and 5645 respectively. During the next step, these alerts were forwarded to the alert clustering module.

A Snort alert consists two major fields that are used to identify an intrusion which caused the corresponding intrusion alert. These fields are called “alert type” (also refer as alert description or alert signature) and “alert category” (also refer as alert class). Alert type contains detailed information about an intrusion whereas alert category contains higher level information about an intrusion. As an example “UDP Filtered Portsweep” and “UDP Portsweep” alert types both

belongs to “attempted-recon” alert category and the alert type “UDP Filtered Portsweep” provides much more specific information than the alert category “attempted-recon”. Snort consists of 34 alert categories (alert classes) as shown in table 4.9. There are four levels in those categories based on their influence [120].

Nine alerts categories and 21 alerts descriptions were produced for DARPA 2000 LLDOS 1.0 while nine alerts categories and 19 alerts descriptions were produced for DARPA 2000 LLDOS 2.0.2 by Snort IDS as shown in table 4.10.

Table 4.9: Snort alert class (alert category) types [120].

Class Type (alert category)	Description	Priority
attempted-admin	Attempted Administrator Privilege Gain	high
attempted-user	Attempted User Privilege Gain	high
inappropriate-content	Inappropriate Content was Detected	high
policy-violation	Potential Corporate Privacy Violation	high
shellcode-detect	Executable code was detected	high
successful-admin	Successful Administrator Privilege Gain	high
successful-user	Successful User Privilege Gain	high
trojan-activity	A Network Trojan was detected	high
unsuccessful-user	Unsuccessful User Privilege Gain	high
web-application-attack	Web Application Attack	high
attempted-dos	Attempted Denial of Service	medium
attempted-recon	Attempted Information Leak	medium
bad-unknown	Potentially Bad Traffic	medium
default-login-attempt	Attempt to login by a default username and password	medium
denial-of-service	Detection of a Denial of Service Attack	medium
misc-attack	Misc Attack	medium
non-standard-protocol	Detection of a non-standard protocol or event	medium
rpc-portmap-decode	Decode of an RPC Query	medium
successful-dos	Denial of Service	medium
successful-recon-largescale	Large Scale Information Leak	medium
successful-recon-limited	Information Leak	medium
suspicious-filename-detect	A suspicious filename was detected	medium
suspicious-login	An attempted login using a suspicious username was detected	medium
system-call-detect	A system call was detected	medium
unusual-client-port-connection	A client was using an unusual port	medium
web-application-activity	Access to a potentially vulnerable web application	medium
icmp-event	Generic ICMP event	low
misc-activity	Misc activity	low
network-scan	Detection of a Network Scan	low
not-suspicious	Not Suspicious Traffic	low
protocol-command-decode	Generic Protocol Command Decode	low
string-detect	A suspicious string was detected	low
unknown	Unknown Traffic	low
tcp-connection	A TCP connection was detected	low

Table 4.10: Snort alert descriptions and categories generated for DARPA LLDOS 1.0 and LLDOS 2.0.2 by Snort.

Snort alert descriptions and categories generated for DARPA LLDOS 1.0		
No	**Signature Name (alert description)**	**Signature Class Name (alert category)**
1	portscan: UDP Filtered Portsweep	attempted-recon
2	portscan: ICMP Filtered Sweep	attempted-recon
3	portscan: UDP Portsweep	attempted-recon
4	portscan: TCP Distributed Portscan	attempted-recon
5	portscan: TCP Portsweep	attempted-recon
6	PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt	attempted-user
7	ET ATTACK_RESPONSE Output of id command from HTTP server	bad-unknown
8	stream5: TCP Small Segment Threshold Exceeded	bad-unknown
9	stream5: FIN number is greater than prior FIN	bad-unknown
10	ET POLICY FTP Login Successful	misc-activity
11	ET POLICY Inbound Frequent Emails - Possible Spambot Inbound	misc-activity
12	ET POLICY Executable and linking format (ELF) file download	policy-violation
13	spp_arpspoof: Directed ARP Request	protocol-command-decode
14	sensitive_data: sensitive data global threshold exceeded	sdf
15	sensitive_data: sensitive data - eMail addresses	sdf
16	ET MALWARE User-Agent (Win95)	trojan-activity
17	http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE	unknown
18	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown
19	http_inspect: UNKNOWN METHOD	unknown
20	http_inspect: SIMPLE REQUEST	unknown
21	http_inspect: UNESCAPED SPACE IN HTTP URI	unknown
Snort alert descriptions and categories generated for DARPA LLDOS 2.0.2		
No	**Signature Name (Alert Description)**	**Signature Class Name (Alert Category)**
1	portscan: UDP Portsweep	attempted-recon
2	portscan: UDP Filtered Portsweep	attempted-recon
3	PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt	attempted-user
4	ET ATTACK_RESPONSE Output of id command from HTTP server	bad-unknown
5	ET INFO PDF Using CCITTFax Filter	bad-unknown
6	http_inspect: LONG HEADER	bad-unknown
7	stream5: Reset outside window	bad-unknown
8	stream5: FIN number is greater than prior FIN	bad-unknown
9	stream5: TCP Small Segment Threshold Exceeded	bad-unknown
10	ET POLICY FTP Login Successful	misc-activity
11	ET POLICY Executable and linking format (ELF) file download	policy-violation
12	spp_arpspoof: Directed ARP Request	protocol-command-decode
13	sensitive_data: sensitive data global threshold exceeded	sdf
14	sensitive_data: sensitive data - eMail addresses	sdf
15	ET MALWARE User-Agent (Win95)	trojan-activity
16	ET INFO Exectuable Download from dotted-quad Host	trojan-activity
17	http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	unknown
18	http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE	unknown
19	http_inspect: UNESCAPED SPACE IN HTTP URI	unknown

4.2.3 Bag of Words Generation and Clustering Process

As described in the methodology section (section 3.3.3), first step was to generate the vocabulary for BoW model that represents alert attributes. For this purpose, DARPA 2000 LLDOS 1.0-scenario one was selected. Few sample alerts shown in table 4.11 were used to illustrate BoW vocabulary creation. Source IP address, destination IP address, alert type ID and alert category were used for vocabulary generation, where alert type was represented by alert type ID. Each unique word in alert attributes was added to the vocabulary. IP address was treated in a special way where four segments of an address was considered as four separate words. BoW vocabulary which was generated from sample alerts shown in table 4.11 is shown in table 4.12.

Table 4.11: Few Snort alerts generated for DARPA 2000 Dataset.

Source IP	**Destination IP**	**Alert Type ID**	**Alert Category**
172.16.112.100	172.16.112.20	650	attempted-recon
172.16.112.50	172.16.113.169	506	sdf
172.16.116.201	206.83.105.134	36489	trojan-activity
207.25.71.186	172.16.117.132	684	unknown
209.87.178.183	192.168.5.122	41297	web-application-attack

Table 4.12: BoW vocabulary generated from alerts shown in table 4.11.

Bag of Words Vocabulary								
activity	application	attack	attempted	recon	sdf	trojan	unknown	web
16	168	169	172	178	183	186	192	50
83	87	105	112	113	115	116	117	
684	71	207	209	25	506	650	20	
206	134	36489	201	100	122	132	41297	

The next task would be cluster this vocabulary to set of word cluster. As an example let total number of clusters was selected as five and then vocabulary (which was generated in previous step) was clustered to five clusters by the k-means clustering module. These clusters are shown in table 4.13.

Table 4.13: BoW vocabulary clusters generated from alerts shown in table 4.11.

Cluster Name	**Elements in Clusters**									
Cluster 0	87	183	178	209	41297	168	192	122	web	application
Cluster 1	172	16	112	113	169	50	506	178	168	sdf
Cluster 2	206	172	36489	16	134	83	201	116	105	trojan
Cluster 3	71	132	207	25	172	16	684	186	117	unknown
Cluster 4	16	172	100	112	115	20	650	186	recon	attempted

Based on these observations, it can be seen that cluster 0 represents “web application attack” between IP address 209.87.178.183 and IP address 192.168.5.122, cluster 2 represents “trojan activity attack” between IP address 172.16.116.201 and IP address 206.83.105.134 and so on. When a new alert was received, the best matching cluster for that alert was assigned by the alert clustering module based on alert attributes. The main objective of clustering is to combine alert attributes and generates a single unit that represents different alerts attributes. This single unit is the cluster ID. A Cluster ID sequence for given input alert sequence was generated by the alert clustering module and that sequence was used as a sequential data series in our model. This sequential data series was used to build HMM module and predict future alert clusters.

In this thesis, DARPA 2000 LLDOS 1.0-scenario one was used for vocabulary generation and then that vocabulary was clustered using the k-means algorithm. There were 243 unique words in the vocabulary, which was generated using DARPA 2000 LLDOS 1.0-scenario one. The DARPA 2000 LLDOS 2.0.2 - scenario two was used as a test network that must be monitored and predicted future activities. DARPA 2000 LLDOS 2.0.2 dataset TCPdump files were sent into Snort. Generated alerts were ordered based on their timestamps and then sent into the pre-processing module and the clustering module. The corresponding cluster IDs for the input alert sequence were assigned by the alert clustering module, which performed sequential data series. Part of that sequential data series was used to train HMM module and other segment was used to test prediction capabilities of HMM module. This process is illustrated in Figure 4.8.

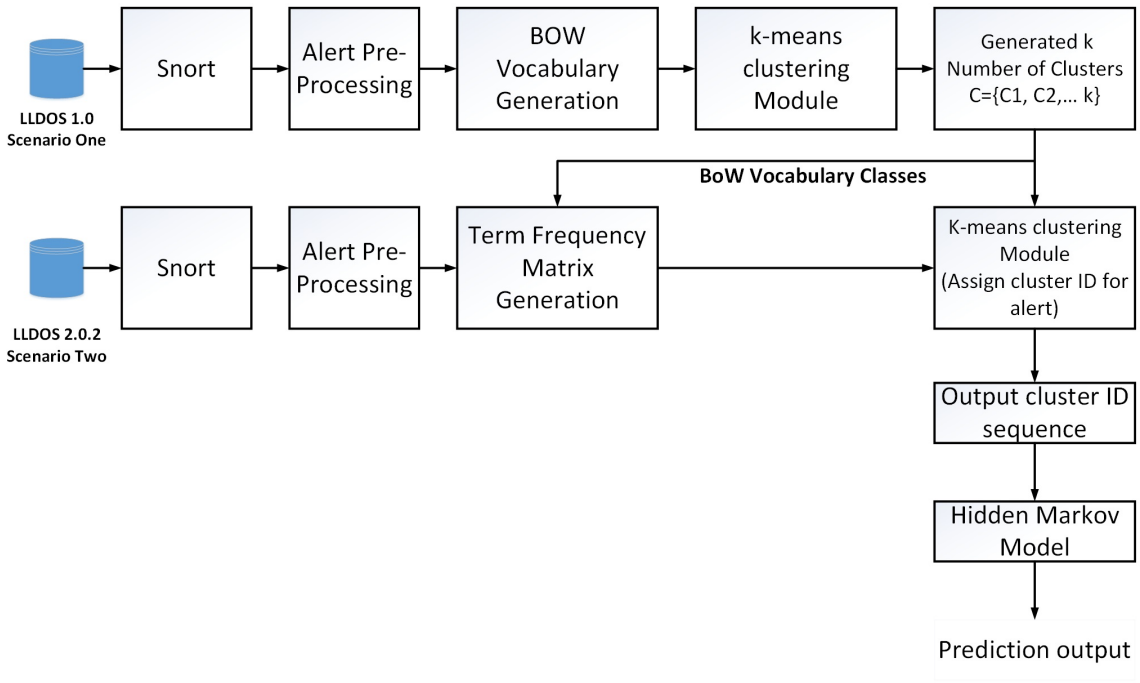


Figure 4.8: Generation of cluster ID for Snort alerts and alert cluster ID prediction process.

The objective of alert clustering is to group alerts which have similar attributes together. In experiments number of clusters were changed as 5, 10, 12, 14, 16, 18, 20, 30, 40 and 50 to observe the effect of number of clusters to the prediction output.

4.3 Alert Prediction

Performance evaluation of hidden Markov model based alert prediction module was performed using the alert cluster ID sequence generated by DARPA 2000 LLDOS 2.0.2 dataset. The length of the cluster ID sequence generated for DARPA 2000 LLDOS 2.0.2 was 5645. 2500 data points were used to train hidden Markov model and other segment was used to evaluate prediction capabilities of the model.

Following experiments were conducted to evaluate performance of proposed alert prediction framework. Detailed description about below mentioned experiments is included in sections 4.3.1 and 4.3.2.

- Performance evaluation of next alert cluster prediction.

    1. Effect of number of hidden states in hidden Markov model on the prediction accuracy.
    2. Effect of the training length on the prediction accuracy.
    3. Effect of number of clusters on the prediction accuracy.
    4. Performance evaluation of multiple intrusions predictions.

- Performance evaluation of next alert category (alert class) prediction.

    1. Effect of number of hidden states in hidden Markov model on the prediction accuracy.
    2. Performance evaluation of multiple intrusions predictions.

4.3.1 Performance Evaluation of Next Alert Cluster Prediction

Let the cluster ID sequence generated by the alert clustering module for DARPA 2000 LLDOS 2.0.2 dataset be denoted by ${C^A}_1, {C^A}_2, ...{C^A}_j...., {C^A}_N$ and N be the length of the sequence. Data series up to $N/2$ points ($\theta = {C^A}_1, {C^A}_2, ...{C^A}_j...., {C^A}_{N/2}$) was used to train HMM Module. Remaining segment ($\omega = {C^A}_{(N/2)+1}, ...{C^A}_j...., {C^A}_N$) was used to evaluate prediction capabilities of the model. Let testing series ω was denoted as $X = x_1, x_2, ...x_j...., x_T$ where T was the length of the testing sequence. In order to predict cluster at $j+1$ location, data series up to j ($x_1, x_2, ...x_j$) was sent to the HMM prediction model. During experiment process j was changed from 1 to $T-1$. Since the cluster at $j+1$ location was known, predicted cluster for $j+1$ location was compared with the actual cluster at $j+1$ location to evaluate prediction performance. Accuracy of the prediction was calculated using the equation as shown in 4.1

$$\text{Accuracy of the Prediction} = \frac{\text{Correctly Predicted Alert Clusters}}{\text{Total Number of Predictions}}. \tag{4.1}$$

In order to easily compare our results with existing alert prediction research activities, three accuracy values were calculated. Those were named as level 1 prediction accuracy (αL^1), level

2 prediction accuracy (αL^2) and level 3 prediction accuracy (αL^3). These notations were used throughout this thesis.

1. Level 1 prediction accuracy (αL^1). In level 1 prediction accuracy, a correct prediction means that, most probable prediction of the alert prediction framework is matched with next action of the attacker.

2. Level 2 prediction accuracy (αL^2). In level 2 prediction accuracy, a correct prediction means that, one of two most probable predictions of the alert prediction framework is matched with next action of the attacker.

3. Level 3 prediction accuracy (αL^3). In level 3 prediction accuracy, a correct prediction means that, one of three most probable predictions of alert prediction framework is matched with next action of the attacker.

- **Effect of number of hidden states in hidden Markov model on the prediction accuracy**

The number of hidden states is one of the major parameter of the hidden Markov model. There is no straightforward method to determine the best number of hidden states of a hidden Markov model [97]. To find out the best model, number of hidden states were changed from 2 to 10 with increment of 1 at a time. Accuracy of alert prediction are shown in Figure 4.9. The maximum αL^1 was observed as 67% for a HMM with 8 hidden states. The lowest αL^1 was observed as 43% for a HMM with 2 hidden states. It was observed that, with the increment of the number of states αL^1 was initially increased and then decreased. The maximum αL^3 was observed as 84% for a HMM with 7 hidden states. By considering this results, it can be observed that the number of hidden states has a significant impact on the accuracy of prediction. αL^1 was changed from 43% to 67% with the varying number of hidden states. During the experiments cluster size was kept at 10.

Variation of prediction accuracy (αL^1) with the length of running sequence is shown in Figure 4.10. Initially, accuracy of prediction was not very stable. It was observed that, after predicting 1500 alerts, accuracy of prediction was stabilized and converged to a steady value.

A HMM with 8 hidden states was achieved the maximum accuracy of prediction (αL^1). Therefore, that HMM module was selected for other experiments beyond this point.

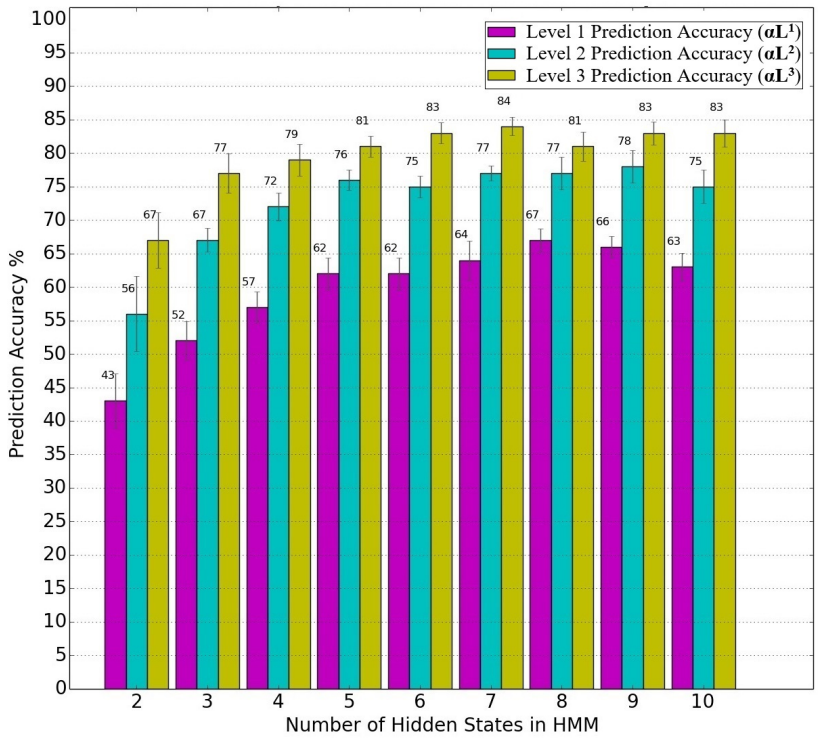


Figure 4.9: Prediction accuracy of alert clusters variation with number of states in hidden Markov model.

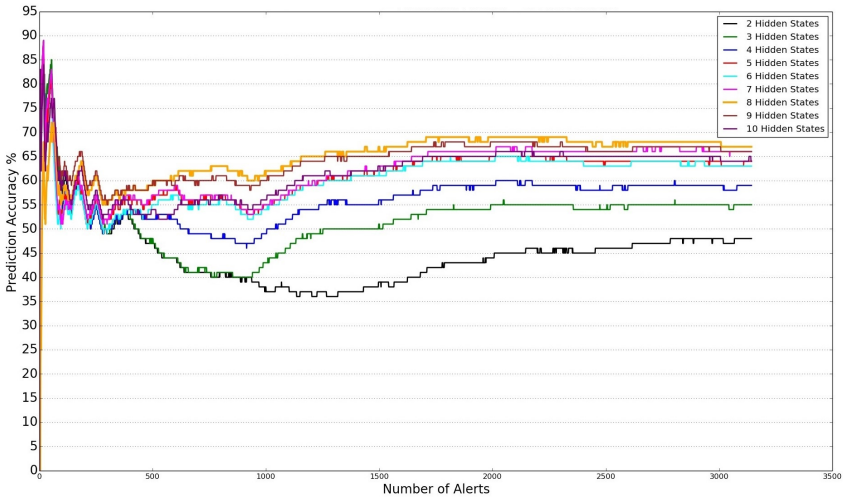


Figure 4.10: Level 1 prediction accuracy (αL^1) of alert clusters variation with length of running sequence.

- **Effect of the training length on the prediction accuracy**

The length of training data sequence has a significant impact on learning process of hidden Markov model parameters (A- state transition probability matrix, B- observation emission probability matrix, π - initial state probability). Variation of prediction accuracy (αL^1) with the length of the training sequence is shown in Figure 4.11. The length of the cluster ID sequence which was generated from DARPA 2000 LLDOS 2.0.2 was 5645. The length of training sequence was changed from 500 to 3500 with the increments of 500 at a time. The remaining segment was used to evaluate prediction performance. Prediction accuracy of next alert cluster ID was recorded for remaining length of the sequence (as an example let the length of training sequence to be 500 and the length of testing sequence to be 5145). Results indicate that, insufficient training lengths produce very low αL^1 (around 20%) for training length 500 and 1000. With the increment of training length αL^1 was increased and the maximum value for αL^1 was recorded as 73% for the training length of 3500.

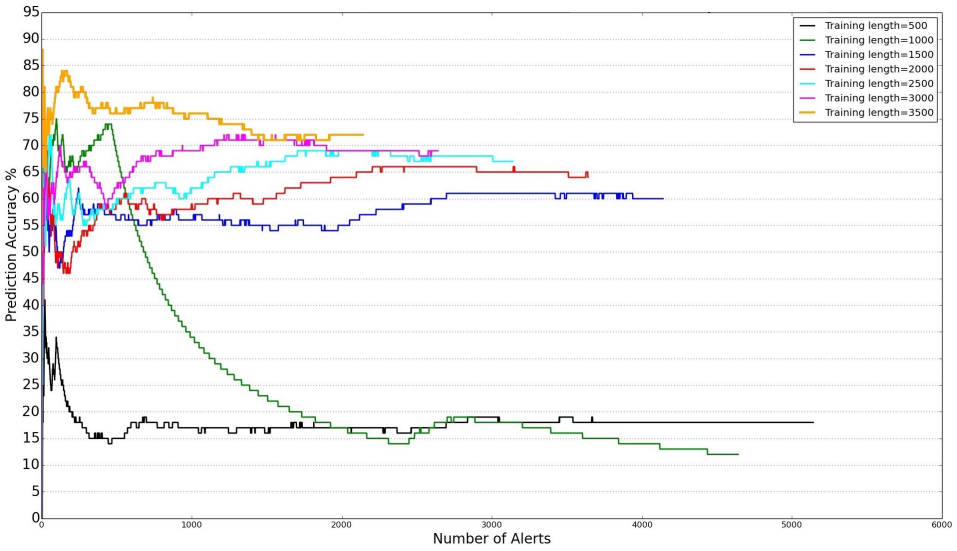


Figure 4.11: Level 1 prediction accuracy (αL^1) of alert clusters variation with length of training sequence.

- **Effect of number of clusters on the prediction accuracy**

The number of unique observation symbols (i.e. symbol size) has a significant impact on the prediction accuracy. To illustrate the impact on the size of unique observation symbols, number

of clusters were changed from 5 to 50. It was observed that HMM prediction fail to achieve better accuracy for higher observation symbol size. Alert perdition accuracy variation with the number of clusters is shown in Figure 4.12.

It is obvious that, a high number of clusters produced better separation among alerts and which result in better grouping. For DARPA dataset there were 9 different alert categories and 19 different alert types produced by Snort. Therefore, in order to group alerts by their corresponding alert category and corresponding alert type at least 9 and 19 clusters were required respectively. The maximum value for αL^1 was 88% recorded for 5 clusters and the lowest 31% was recorded for 50 clusters. However, up to 30 clusters αL^3 was remained over 70%.

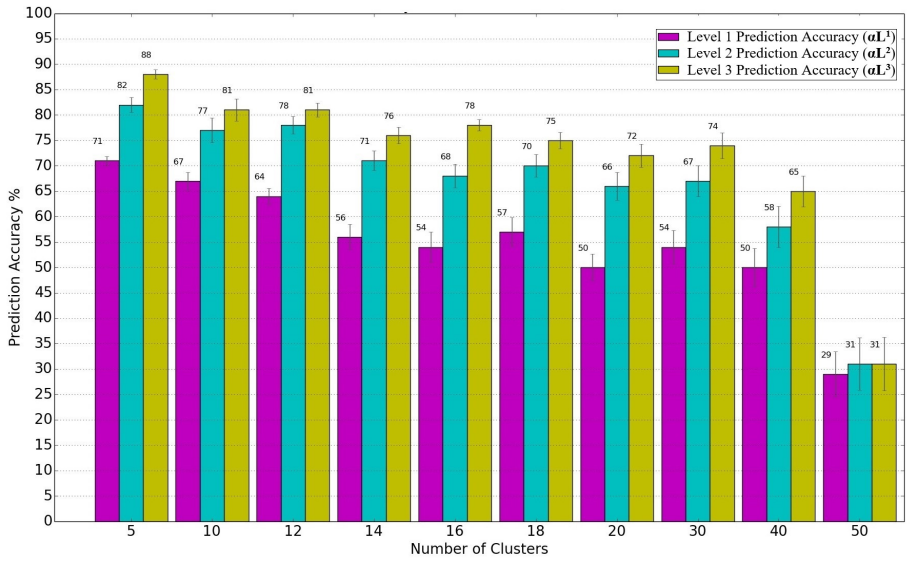


Figure 4.12: Prediction accuracy of alert clusters variation with number of clusters.

- **Performance evaluation of multiple intrusions predictions**

As described in sections 2.3.1 and 2.3.1 most of the research activities were focused on predicting next possible step only. In this thesis, multiple point prediction of length of l was executed as:

Let testing series was denoted as $X = x_1, x_2, ...x_j...., x_T$ where T was the length of the testing sequence. In order to predict the cluster at $j + 1$ location, data series up to j $(x_1, x_2, ...x_j)$ was sent to the HMM prediction model. The predicted cluster for $(j + 1)^{th}$ location was appended

to the input to predict cluster at $(j+2)^{th}$ location. This cycle continued until l number of clusters was predicted by HMM (i.e. data series up to j $(x_1, x_2, ...x_j)$ was used to predict l number of future clusters). After l number of clusters had been predicted by the HMM, next prediction cycle was started. For that next cycle, data series up to $j+l$ was selected and that cycle was continued until the system had predicted another l number of clusters. Likewise, system had continuously predicted multiple lengths of data points until it reached end of data sequence.

To examine the multiple data points prediction capability of HMM prediction module, the length of prediction (l) was changed from 1 to 5 and accuracy of prediction was recorded for 10 clusters. Accuracy of predictions is shown in Figure 4.13. It was observed that, with the increment of prediction length accuracy of prediction was decreased. One data point prediction was achieved the highest accuracy of 67%, 77% and 81% for αL^1, αL^2 and αL^3 correspondingly while five data point prediction was achieved lowest with 53%, 65% and 77% correspondingly.

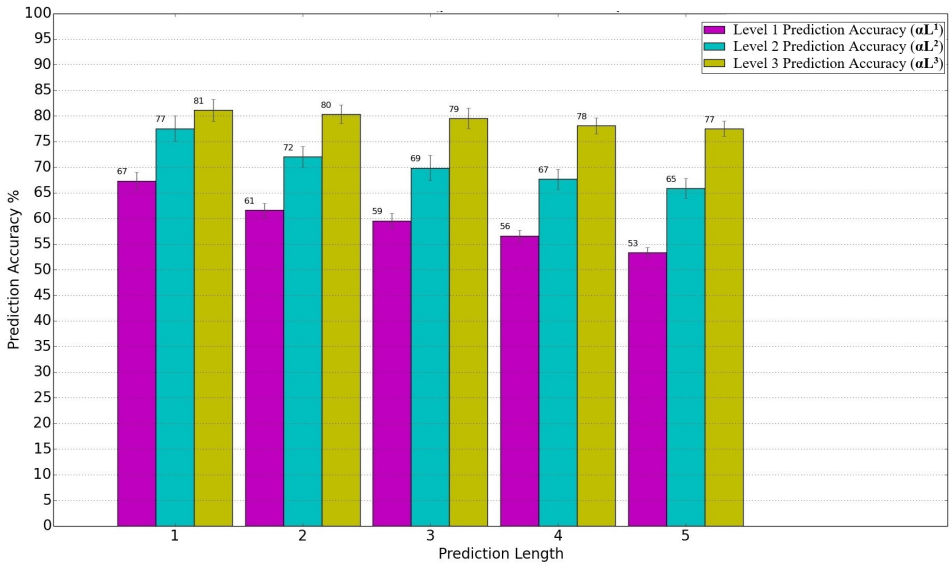


Figure 4.13: Prediction accuracy of alert clusters variation with multiple data point prediction.

To further analyze how the number of clusters were effected on the accuracy of prediction, the number of clusters were changed from 10 to 50 while keeping the length of prediction as 10. These results are shown in Figure 4.14. The maximum value 69% for αL^3 was achieved by the prediction module with 10 clusters configuration. The minimum value 13% for αL^3

was achieved by the prediction module with 50 clusters configuration. Up to 20 clusters, αL^3 remains higher than 60%. However, for 30, 40 and 50 clusters, accuracy of prediction was decreased significantly. In multiple length prediction, predicted symbol in the previous state was append to the input sequence to predict next symbol. Hence error of previous prediction was propagated to other predictions as well.

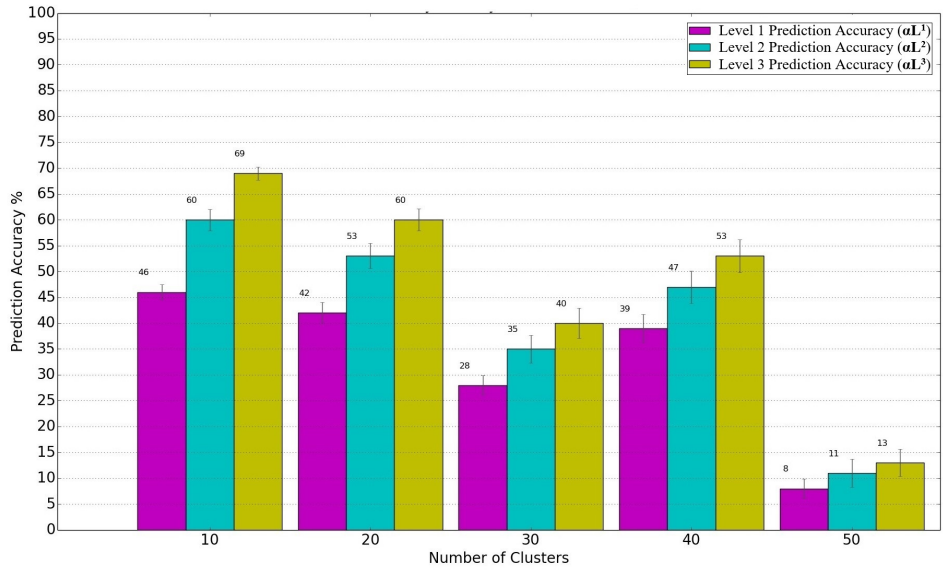


Figure 4.14: Prediction accuracy of alert clusters variation with number of clusters for 10 data point prediction.

4.3.2 Performance Evaluation of Next Alert Category (alert class) Prediction

As described in sections 2.3.1 and 2.3.1 most of alert prediction research activities were largely focused on predicting future alert category. In order to compare proposed hidden Markov model with other network intrusion prediction research activities, alert category based prediction was evaluated. For following experiments, sequential data series was generated by considering an alert category as an observation. For DARPA 2000 LLDOS 2.0.2 dataset nine different alert categories were generated by Snort. These alerts categories were mapped to the symbols as shown in table 4.14. As an example, let alert category sequence is generated by Snort as “Attempted-user”, “Policy-violation”, “Trojan-activity”, “Attempted-user” and “Misc-activity”. Then based on the mapping, corresponding symbol sequence for this five alerts is

B, E, H, B and D. By using this mapping, alert sequence generated for DARPA 2000 LLDOS 2.0.2 was converted to a symbol series. Similar to the other experiments 2500 data points were used to train hidden Markov model and the remaining segment was used to evaluate prediction performance.

Table 4.14: Snort alert categories generated for DARPA LLDOS 2.0.2 with symbol mapping.

Alert Category	**Symbol**
attempted-recon	A
attempted-user	B
bad-unknown	C
misc-activity	D
policy-violation	E
protocol-command-decode	F
sdf	G
trojan-activity	H
unknown	I

- **Effect of number of hidden states in hidden Markov model on the prediction accuracy**

To evaluate the accuracy of alert prediction with the number of hidden states, the number of hidden states were changed from 2 to 10 with increment of 2 at a time. The results of this experiment is shown in Figure 4.15. The maximum αL^1 of 76% was recorded for HMMs with six and eight number of hidden states. Also, it was observed that sudden drop of accuracy was recorded for a HMM with 10 hidden states. It was observed that, for this model number of hidden states were higher than the number of unique observations. Also, 10% of αL^3 accuracy improvement was observed for prediction of alert categories compared to prediction of alert clusters size of 10.

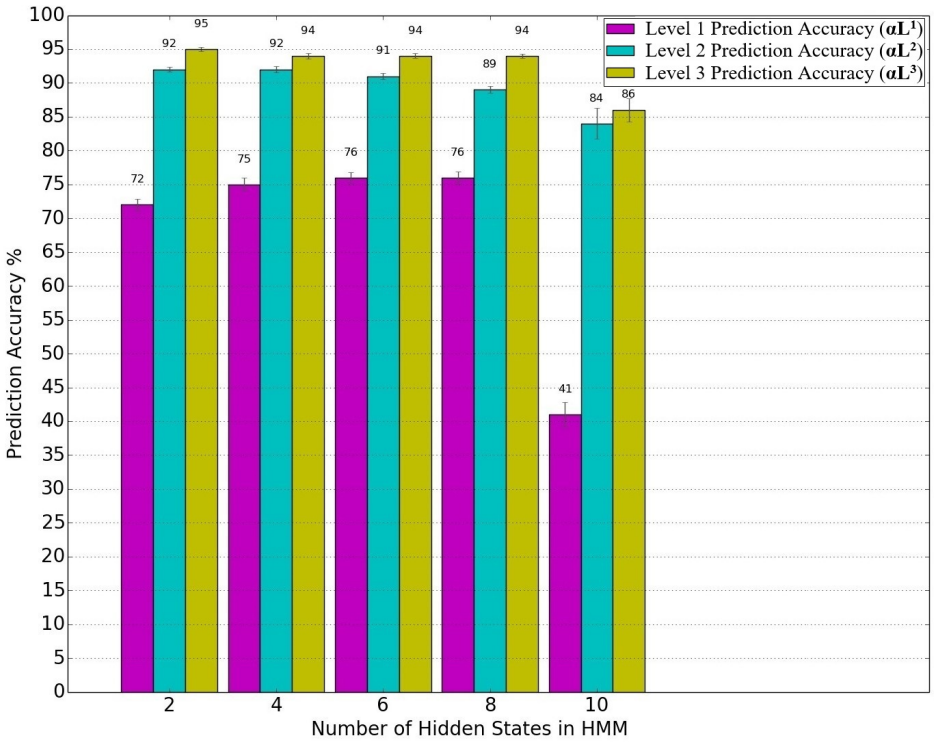


Figure 4.15: Prediction accuracy of alert category variation with number of hidden states in HMM.

- **Performance evaluation of multiple intrusions predictions**

To examine the multiple data points prediction capability of the HMM prediction module, the length of prediction (l) was changed from 1 to 5 and accuracy of prediction was recorded. Accuracy of future alert category predictions is shown in Figure 4.16. With the increment of prediction length, the prediction accuracy was decreased. The maximum prediction accuracies for αL^1, αL^2 and αL^3 were observed as 76%, 91% and 94% correspondingly for single intrusion prediction while the lowest was observed for five intrusions prediction as 72%, 88% and 92% respectively. By comparing these results, it was observed that, better prediction results were achieved for small observation symbol size.

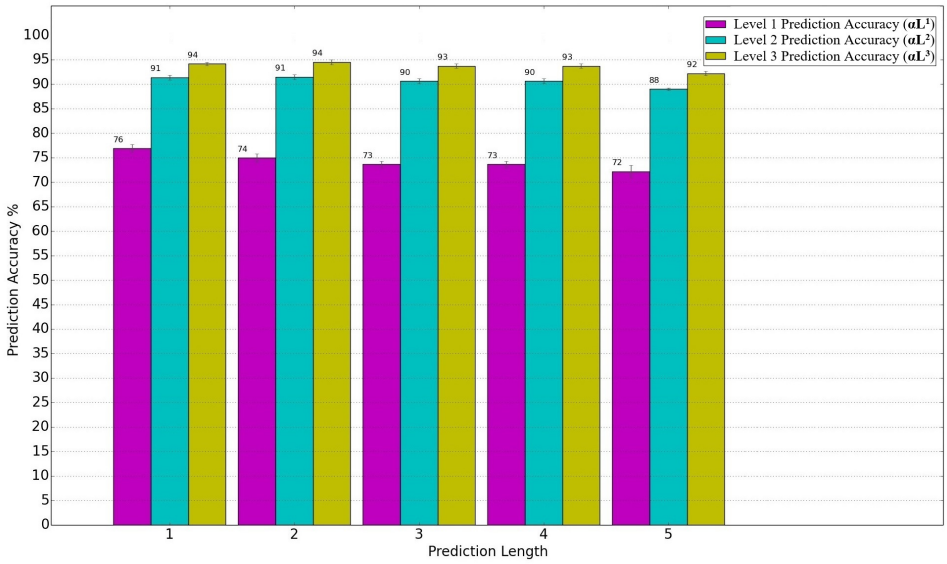


Figure 4.16: Prediction accuracy of alert category variation with multiple data point prediction.

Fava et al. [11] presented an alert prediction method based on variable length Markov model which is similar the proposed alert prediction framework. The major differences between their approach and the proposed algorithm is that they have generated three separate Markov models for alert category, alert description and destination IP address attributes of an alert. By using these models, they predicted the next alert category, next alert description and destination IP address separately. With this approach, they achieved an accuracy of 90% for αL^3 for predicting the next alert category. When the prediction algorithm proposed in this thesis was constrained to only predict the alert category without additional information, it obtained an accuracy of 95% for αL^3. Following are the key differences between the proposed algorithm and that of Fava et al. method:

1. Fava et al. used a data set developed in-house for their experiments whereas this thesis presents results from the DARPA data set.

2. The closest result that can be compared is the prediction of next alert category. The DARPA data set has nine unique alert categories whereas Fava et al. doesn't disclose the number of unique alert types that are present in their data set.

3. When the proposed alert prediction framework was used to predict the alert category as well as the source IP address, the destination IP address, and the alert type, it achieved

a level 3 prediction accuracy (αL^3) of 81%, 81%, 76%, 78%, 75% and 72% for cluster sizes of 10, 12, 14, 16, 18 and 20 respectively. Even thought the proposed alert cluster prediction achieved lower prediction accuracy compared to Fava et al. method, it provides critical information the intrusion which will be essential for any response.

4.3.3 Node IP and Alert Cluster Distribution

Alerts were generated by Snort for DARPA 2000 LLDOS 2.0.2 contains 187 different source IP addresses and 172 different destinations IP addresses for 5466 alerts. Those IP addresses were belonged to DARPA network and outside networks. The average number of alerts per destination host was 32. Distribution of destination IP addresses of alerts is shown in Figure 4.17. There were 34 destination IPs targeted more than 50 times. Host 172.16.116.194 were targeted in 262 times and hosts 172.16.115.87, 172.16.113.207, 172.16.116.44, 172.16.113.148, 172.16.113.168, 172.16.112.194, 172.16.117.132, 135.13.216.191 and 172.16.112.149 were targeted 199, 185 152, 127, 126, 99, 93, 92 and 54 times respectively. The average number of alerts generated by a host was 32 and there were 15 hosts generated more than 50 alerts. Distribution of source IP addresses of alerts is shown in Figure 4.18.

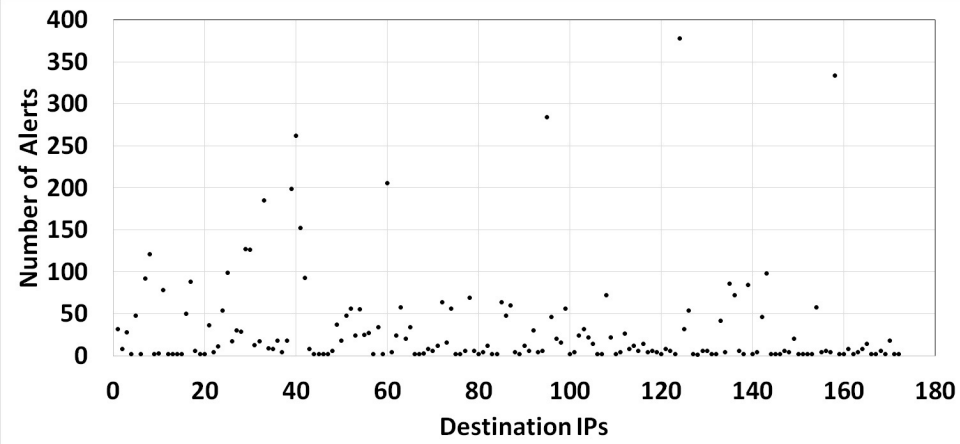


Figure 4.17: Destination IP distribution of alert generated for DARPA LLDOS 2.0.2.

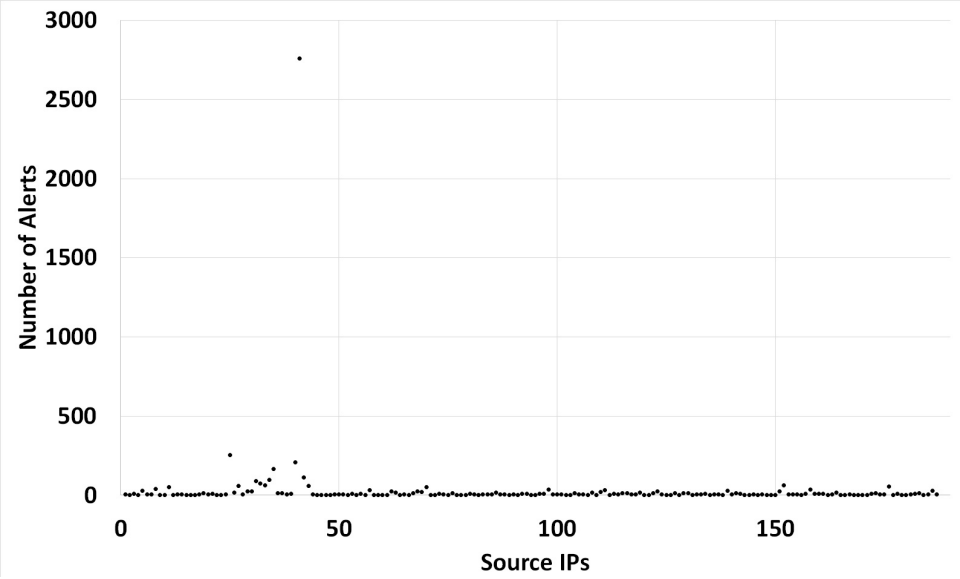


Figure 4.18: Source IP distribution of alert generated for DARPA LLDOS 2.0.2.

The objective of alert clustering is to group alerts which have similar attributes. The HMM prediction results indicated that, prediction accuracy had been improved with less number of clusters. But it is not possible to decrease number of clusters, because when number of clusters are reduced, it decreases the clear separation among alert clusters. To further illustrates this factor, number of different alert categories in each cluster were analyzed. Alert category distribution with the number of clusters is shown in table 4.15 and Figure 4.19.

Table 4.15: Alert cluster distribution of DARPA LLDOS 2.0.2.

Alert Category	5 Clusters	10 Clusters	20 Clusters	30 Clusters	40 Clusters	50 Clusters
unknown	2	4	9	11	18	22
sdf	1	2	4	8	11	10
trojan-activity	1	2	3	5	4	6
protocol-command-decode	1	1	1	1	1	1
attempted-user	0	0	1	1	1	1
attempted-recon	0	1	1	1	1	1
bad-unknown	0	0	0	0	1	1
misc-activity	0	0	0	0	1	1
policy-violation	0	0	0	0	1	1

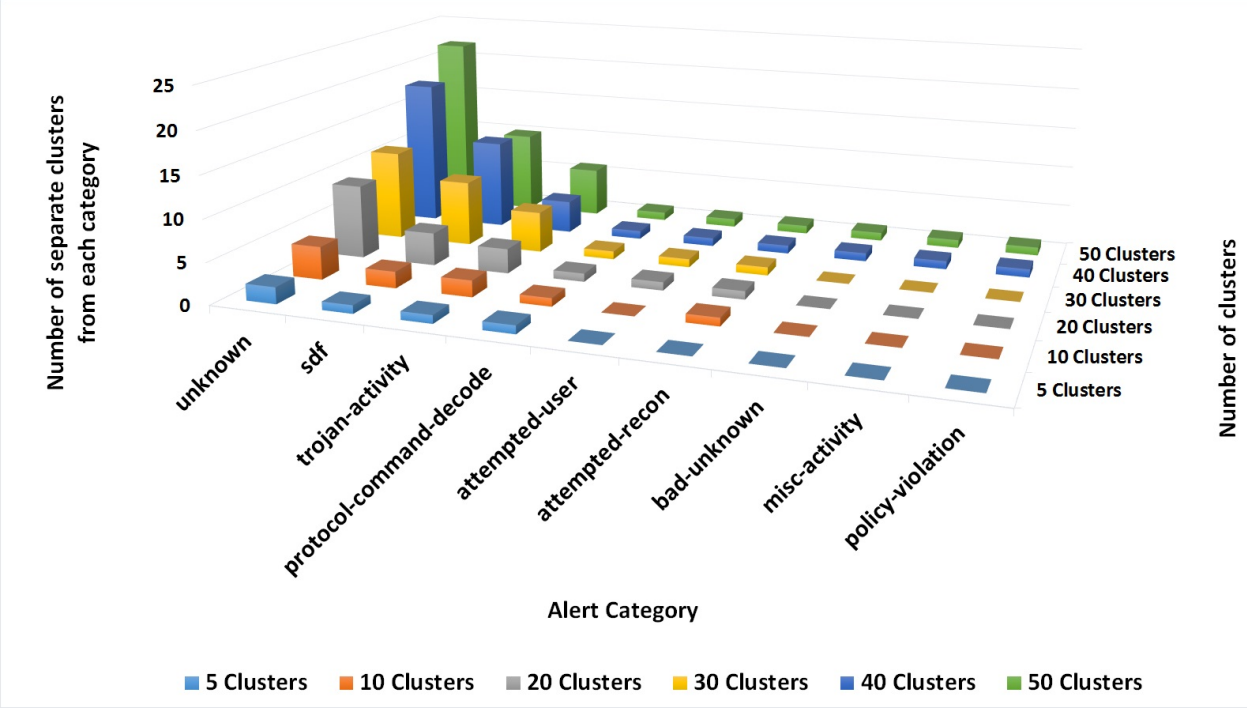


Figure 4.19: Alert cluster distribution of DARPA LLDOS 2.0.2.

For 5 clusters, there were two separate cluster groups for “unknown” alert category and one separate cluster group for “sdf”, “trojan-activity” and “protocol-command-decode” alert categories respectively. However, when the number of clusters increased to 10, there were four separate cluster groups for “unknown” alert category, two separate cluster groups for “sdf” and“trojan-activity” alerts categories, one cluster group for “protocol-command-decode” and

one cluster group for “attempted-recon” alert category. It can be observed that, with the increase of number of clusters clear separation between alert categories was witnessed. Also, alerts belong to same category were split into different alert clusters based on their IP address. As an example for 20 cluster configuration, there were nine clusters for “unknown” alert category, four clusters were generated for “sdf” alert category and three clusters were generated for “trojan-activity” alert category. These results indicate that, it is beneficial to increase the number of clusters which improves separation between alert clusters. As a result of that more similar alerts can be grouped together. On the other hand, the number of clusters was highly related to the accuracy of prediction. With the increase of the number of clusters prediction accuracy tends to decrease as described in section 4.3. Possible solution for this problem is further discussed in section 5.2.3.

4.4 Alert Cluster Output

It is important to find out what are the information that can be extracted from prediction output in order to determine possible responses. In this thesis, intrusion alerts attributes were used to cluster similar alerts together. Source IP address, destination IP address, alert type (alert signature) and alert category (alert signature class) were used to cluster intrusion alerts together. It impractical to consider exact source IP address and destination IP address in clustering process. It will eventually increase number of clusters as well. As a solution for this, exact source IP (such as 172.178.12.189) and destination IP range (such as 192.168.1.0/24) were considered. This selection can be justified since number of attackers are much less than number of users.

Some of the important clusters were formed by the proposed system is shown in table 4.16. Alert prediction module predicts future alert cluster that can be occur in the future. Then by analyzing alert attributes of the predicted cluster, response system can determine what is the type of the intrusion, from which source IP to which destination IP range it is going to happen. Based on this information, response system can execute response action for that network segment.

As an example, if prediction module predicts next possible cluster as cluster ID 1 where that represents alerts attributes:

Outside Network IP address: 199.95.209.99, **Inside Network IP range**: 172.16.116.0/24

Alert Class: trojan-activity, **Alert Type**: ET MALWARE User-Agent (Win95).

By looking at this information, response system can determine a possible future trojan activity (ET MALWARE User-Agent) that can occur from 199.95.209.99 (outside IP range) to 172.16.116.0/24 network segment of the network. This information is used by the response system to select a suitable response for predicted future intrusions. The possible responses that can be implemented using these cluster output is discussed in section 5.2.1.

Table 4.16: Alert cluster output of DARPA LLDOS 2.0.2.

	Outside Network IP Address	**Inside Network IP Range**	**Alert Category**	**Alert Type**
1	199.95.209.99	172.16.116.0/24	trojan-activity	ET MALWARE User-Agent (Win95)
2	207.25.71.141, 207.25.71.142, 207.25.71.30, 207.25.71.200	172.16.116.0/24	trojan-activity	ET MALWARE User-Agent (Win95)
3	172.16.115.20* (* inside host)	172.16.112.0/24	attempted-recon	portscan: UDP Portsweep , portscan: UDP Filtered Portsweep
4	172.16.115.20* (* inside host)	172.16.114.0/24	attempted-user	PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt
5	208.2.188.61, 204.71.242.42, 204.248.150.136, 139.72.190.50, 208.240.89.202, 151.193.131.161, 209.1.224.190, 199.173.162.18, 205.252.248.98, 209.67.29.11	172.16.116.0/24	trojan-activity	ET MALWARE User-Agent (Win95) or ET INFO Exectuable Download from dotted-quad Host
6	206.79.171.51, 206.39.184.2	172.16.116.194	trojan-activity	ET MALWARE User-Agent (Win95)
7	199.172.144.24, 209.1.224.15	172.16.113.0/24	unknown	http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE or http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
8	197.218.177.69	172.16.113.0/24, 172.16.112.0/24	sdf	sensitive_data: sensitive data - eMail addresses sensitive_data: sensitive data global threshold exceeded
9	192.254.26.2	172.16.112.0/24	unknown	http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE or http_inspect: NO CONTENT-LENGTH OR TRANSFER ENCODING IN HTTP RESPONSE
10	206.62.132.69	172.16.113.0/24	unknown	http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE
11	194.27.251.21	172.16.113.0/24, 172.16.112.0/24	sdf	sensitive_data: sensitive data - eMail addresses sensitive_data: sensitive data global threshold exceeded
12	199.172.144.24	172.16.116.194	trojan-activity	ET MALWARE User-Agent (Win95)
13	194.7.248.153, 204.248.150.136	172.16.113.0/24, 172.16.112.0/24, 172.16.116.0/24	sdf	sensitive_data: sensitive data - eMail addresses sensitive_data: sensitive data global threshold exceeded
14	134.205.131.26, 164.214.2.61, 192.225.36.9, 205.128.215.70, 205.181.112.114, 205.181.112.65, 205.181.112.72, 205.181.112.74, 205.252.248.98, 216.40.24.2	172.16.112.0/24, 172.16.113.0/24, 172.16.115.0/24, 172.16.116.0/24	unknown	http_inspect: MESSAGE WITH INVALID CONTENT-LENGTH OR CHUNK SIZE or http_inspect: NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
15	195.115.218.108	172.16.112.0/24, 172.16.113.0/24	sdf	sensitive_data: sensitive data - eMail addresses sensitive_data: sensitive data global threshold exceeded
16	197.182.91.233	172.16.112.0/24, 172.16.113.0/24, 172.16.114.0/24	sdf	sensitive_data: sensitive data - eMail addresses sensitive_data: sensitive data global threshold exceeded
17	209.185.151.128, 209.185.151.129 209.3.209.166	172.16.116.194	trojan-activity	ET MALWARE User-Agent (Win95)

Chapter 5

Discussion and Future Work

Usage of computer based applications is becoming increasingly necessary within the society. The rapid development of service automation and social networking increases human activities in cyberspace. Illegal activities such as unauthorized data access, data theft, data modification and various other intrusion activities have been growing rapidly during last decade. While intrusion detection techniques have emerged as a solution for network intrusions, the detection by itself is not sufficient. Automated advanced intrusion detection and response systems are required to provide real-time protection to current computer systems and networks

The objective of this thesis is to predict future network intrusions. Intrusion alerts generated by intrusion detection systems are used to learn strategies of attackers. This knowledge is utilized by the proposed prediction module for future intrusion prediction. Most of the previous intrusion prediction methods mainly focused on prediction of either alert type or alert category. However, the proposed module is focused on predicting a set of alerts attributes. Alert clustering concept is used to achieve this task. Source IP address, destination IP address, alert type and alert category attributes are used to cluster alerts. For a given alert sequence of $A_1, A_2, \ldots, A_t, \ldots, A_T$ sequence of clusters $C^A{}_1, C^A{}_2, \ldots, C^A{}_t, \ldots, C^A{}_T$ is generated by the alert clustering module. Then the alert prediction module predicts future cluster ($C^A{}_{T+1}$) based on the input sequence. Alert cluster prediction results are forwarded to an intrusion response module, which selects and executes suitable responses for future intrusions.

5.1 Conclusion

In this thesis, a hidden Markov method based alert prediction framework is proposed. Alert clustering is employed to group selected alert attributes together. A given sequence of alerts is converted to a sequence of alert clusters and then a hidden Markov model is used to predict future alert clusters based on the input. The proposed algorithm also provides the alert category as well as the source IP address, the destination IP address, and the alert type, which are critical in responding to the intrusion.

Most of the research activities related to intrusion activity prediction are focused on predicting next possible step only. In this thesis, multiple steps are predicted. Results indicate that, one step prediction achieved the highest prediction accuracy of 67%, 77% and 81% for αL^1, αL^2 and αL^3 respectively, while five step prediction achieved the lowest with 53%, 65% and 77% for αL^1, αL^2 and αL^3 respectively.

When comparing graphs based intrusion prediction methods with a sequential series based prediction algorithm such as the one proposed in this thesis, graph based method has following limitations:

- The prediction depends on the availability of an attack graph library.
- It requires matching the observed intrusion sequence with all attack graphs in the library to find a matching attack graph, which may increase computational cost of the prediction process.

Based on the results, it is observed that a smaller number of clusters tend to improve prediction accuracy. The maximum value of αL^3 was 88% for 5 clusters and the lowest 31% was recorded for 50 clusters. When the number of clusters are smaller, it results in a smaller set of unique symbols for the HMM model which improves the learning abilities of the HMM model compared to a larger symbol size. However, when the number of clusters are smaller, it will hinder the separation of unique alert types and cause merging of two or more alert types. As an example, in the DARPA data set, there are nine unique alerts categories generated by Snort

IDS. Therefore, if the number of clusters are less than nine, then one alert cluster may include more than one alert category which hinders the prediction.
Also, the experimental results indicated that when the number of hidden states are lower than the number of observations (i.e. when the number of observations are 10 and the number of hidden states are between 2 to 4), level 1 prediction accuracy (αL^1) is lower compared to higher number of hidden states (i.e. number of hidden states are between 5 to 8). It is observed that the maximum αL^1 difference was 24% for these two scenarios. This shows that that when the number of hidden states are low, it may not be possible to model the system states changes efficiently because not enough states are available to represent state transition during a multi-stage intrusion scenario.

There are still some challenges that need to be addressed in the proposed alert prediction framework. They include increasing the prediction accuracy with the increase of cluster size and predicting intrusion types that not present in the training data set. Currently it is not possible to predict an intrusion type if that not present in the training data set. Another challenge is identifying false alerts and misleading intrusion actions generated by the attacker in order to mislead intrusion detection systems. Some possible solution for these problems are discussed in the next section.

5.2 Future Work

5.2.1 Intrusion Response

Intrusion detection systems (IDSs) generate alerts once they detect network intrusions. These alerts can then be used as an input to intrusion prediction systems to understand strategies of attacker and predict future network intrusions. By identifying future steps of attacker, suitable response actions can be identified to mitigate future intrusions. Selecting a response is a complex process requiring assessment of risk of an intrusion, cost of a response and response execution procedure before selecting a response. Developing such as sophisticated response system is beyond the scope of this thesis. However, to illustrates the usage of alert prediction,

basic response actions that may be executed based on prediction results are discussed below.

Table 5.1 illustrates few response actions that may be executed for intrusions present in DARPA LLDOS 2.0.2. These response actions are proposed only to illustrate how prediction output can be utilized in the process of response action execution. In practice, response actions are much complex than this and response actions are not just selected only considering intrusion activity.

Table 5.1: Response actions for alert produced by DARPA LLDOS 2.0.2.

	Alert Type (alert signature)	**Alert Category (signature class)**	**Proposed Response Action**
1	portscan: UDP Filtered Portsweep	attempted-recon	Block traffic from originator or
2	portscan: UDP Portsweep	attempted-recon	Route originator traffic to honey pot for further analyze
3	portscan: TCP Portsweep	attempted-recon	of originator behavior
4	portscan: ICMP Filtered Sweep	attempted-recon	
5	portscan: TCP Distributed Portscan	attempted-recon	
6	PROTOCOL-DNS TMG Firewall Client long host entry exploit attempt	attempted-user	Block originator traffic, Temporally lock host table (to avoid modification)
7	ET POLICY Inbound Frequent Emails - Possible Spambot Inbound	misc-activity	Block email traffic originated from sender IP address
8	ET POLICY Executable and linking format (ELF) file download	policy-violation	Isolated Host from network, Run System virus scan process to remove agent that originates file download process
9	ET MALWARE User-Agent (Win95)	trojan-activity	Isolated Host from network, Improve host virus protection to improve security, Run System virus scan process to remove trojan agent
10	ET INFO Executable Download from dotted-quad Host	trojan-activity	Isolated Host from network, Improve host virus protection to improve security, Run System virus scan process to remove trojan agent

As an example, consider a case where the prediction module has predicted the next possible cluster as cluster ID 1 which has following alerts attributes as shown in table 4.16:
Outside Network IP address: 199.95.209.99, **Inside Network IP range**: 172.16.116.0/24
Alert Class: trojan-activity, **Alert Type**: ET MALWARE User-Agent (Win95).
By analyzing prediction output, originator IP address and destination IP address (or range) can be identified by the response system. Since next attack step of attacker is “trojan activity”, then attacker may download executable code or software to the destination hosts to start a trojan activity. Therefore, response actions such as “improve virus protection of host nodes”, “temporally isolated hosts nodes from network” and “run virus scan to identify trojan related software or codes” can be executed by the response system in order to prevent future intrusion activities.

5.2.2 HMM Training Process

Behavior of attackers depend on many factors such as network topology, operating systems running on hosts, services running on the network and geographical location of networks, etc.

Therefore, prediction module should be able to adapt to different conditions and networks. If the prediction module heavily depends on domain knowledge and specific scenario knowledge (such as attack graphs based prediction methods), it will not be able to adapt to different conditions. Hence it is important to develop an alert prediction module which is capable of operating under different conditions and networks.

Alert prediction module proposed in this thesis is not dependent on specific domain knowledge. Instead, it depends on a training process. Hence this module is capable of adapting to different environments. In order to gain efficient and best training process, a honey pot network may be used to train the proposed alert prediction module. A honey pot network is a network with decoy computer resources, which is set up for the purpose of being probed, attacked and potentially exploited [121].

Different intrusion scenarios have different intrusion patterns. Therefore, training a hidden Markov module which can adapt to multiple intrusion scenarios is a challenging task. In order to adapt to multiple scenarios, multiple HMM modules may be employed. Each HMM module may be trained based on different set of intrusion scenarios. Then, for a given input intrusion alerts sequence, suitable HMM module may be selected based on their likelihood value. A HMM model with the highest likelihood value will be the most suitable model that represents the behavior of the input intrusion alerts sequence.

5.2.3 Alert Clustering Process

The objective of alert clustering is to group alerts which have similar characteristics. HMM prediction results indicated that, lesser number of clusters improves prediction accuracy. The maximum value of αL^3 was 88% for 5 clusters and the lowest 31% was recorded for 50 clusters. However, αL^3 remains over 70% for up to 30 clusters. Even though a smaller number of clusters tend to improve prediction accuracy, this has a significant disadvantage. When number of clusters are reduced, it decreases the separation between alert clusters and that will result in meaning less clustering (i.e. alerts are clustered without clear separation). One possible

solution to this problem is to develop a distributed prediction model with multiple HMMs. Let number of clusters per one system to be x. If total number clusters are k and then k/x number of distributed system can be used. Clusters generated by alert clustering module will be divided to relevant distributed system based on cluster ID range, where each distributed segment is responsible for handling a range of clusters.

5.2.4 Effect of Noise and False Alerts

The performance of alert prediction framework depends on intrusion alerts generated by intrusion detection systems (IDSs). If intrusion detection system failed to detect some important intrusion activities, then it will affect the performance of prediction module. In addition, IDSs can also produce false alerts. Also, false activities (noise) can be generated by attackers to disturb intrusion detection and prediction process. The proposed HMM alert framework uses intrusion alerts as observations of HMM. In hidden Markov model, probability of an observation depends on corresponding hidden state. This makes the proposed algorithm more robust to noise and false alerts.

5.2.5 IRS Feedback Loop

Intrusion prediction output is sent into response module to assist response selection process. Future research on coupling the prediction module and response module may offer abilities to generate better responses. In some scenarios, a response action that is executed by a response system may be able to stop an intrusion. In such a case, parameters of the prediction module may be updated based on response actions. The proposed HMM based alert prediction module is capable of achieving this because in a HMM, every state has observation emission probability for each observation. If some observations are not possible, this probability distribution matrix may be updated to match new conditions.

Bibliography

[1] McAfee Labs. Mcafee labs report 2016 threats predictions. `http://www.mcafee.com/ca/resources/reports/rp-threats-predictions-2016.pdf`, 2015. Accessed: 2016-6-22.

[2] B. Sun, L. Osborne, Y. Xiao, and S. Guizani. Intrusion detection techniques in mobile ad hoc and wireless sensor networks. *IEEE Wireless Communications*, 14(5):56–63, October 2007. ISSN 1536-1284. doi: 10.1109/MWC.2007.4396943.

[3] B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, May 1994. ISSN 0890-8044. doi: 10.1109/65.283931.

[4] M. Marchetti, M. Colajanni, and F. Manganiello. Identification of correlated network intrusion alerts. In *Cyberspace Safety and Security (CSS), 2011 Third International Workshop on*, pages 15–20, Sept 2011. doi: 10.1109/CSS.2011.6058565.

[5] Alireza Shameli-Sendi, Naser Ezzati-Jivan, Masoume Jabbarifar, and Michel Dagenais. Intrusion response systems: survey and taxonomy. *Int. J. Comput. Sci. Netw. Secur*, 12 (1):1–14, 2012.

[6] N.B. Anuar, M. Papadaki, S. Furnell, and N. Clarke. An investigation and survey of response options for intrusion response systems (irss). In *Information Security for South Africa (ISSA), 2010*, pages 1–8, Aug 2010. doi: 10.1109/ISSA.2010.5588654.

[7] Sapon Tanachaiwiwat, Kai Hwang, and Yue Chen. Adaptive intrusion response to minimize risk over multiple network attacks. *ACM Trans on Information and System Security*, 19:1–30, 2002.

[8] Zhi-tang Li, Jie Lei, Li Wang, and Dong Li. A data mining approach to generating network attack graph for intrusion prediction. In *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, volume 4, pages 307–311. IEEE, 2007.

[9] Bo-Chao Cheng, Guo-Tan Liao, Chu-Chun Huang, and Ming-Tse Yu. A novel probabilistic matching algorithm for multi-stage attack forecasts. *Selected Areas in Communications, IEEE Journal on*, 29(7):1438–1448, 2011.

[10] Lingyu Wang, Anyi Liu, and Sushil Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer communications*, 29(15):2917–2933, 2006.

[11] Daniel S Fava, Stephen R Byers, and Shanchieh Jay Yang. Projecting cyberattacks through variable-length markov models. *Information Forensics and Security, IEEE Transactions on*, 3(3):359–369, 2008.

[12] Haitao Du, Daniel F Liu, Jared Holsopple, and Shanchieh Jay Yang. Toward ensemble characterization and projection of multistage cyber attacks. In *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*, pages 1–8. IEEE, 2010.

[13] A. A. Ramaki, M. Khosravi-Farmad, and A. G. Bafghi. Real time alert correlation and prediction using bayesian networks. In *Information Security and Cryptology (ISCISC), 2015 12th International Iranian Society of Cryptology Conference on*, pages 98–103, Sept 2015. doi: 10.1109/ISCISC.2015.7387905.

[14] David Kulp David Haussler and Martin G Reese Frank H Eeckman. A generalized hidden markov model for the recognition of human genes in dna. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology, St. Louis*, pages 134–142, 1996.

[15] Stephen E Levinson. Continuously variable duration hidden markov models for automatic speech recognition. *Computer Speech & Language*, 1(1):29–45, 1986.

[16] James D Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica: Journal of the Econometric Society*, pages 357–384, 1989.

[17] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[18] Leonard E Baum, John Alonzo Eagon, et al. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc*, 73(3):360–363, 1967.

[19] James Cannady and Jay Harrell. A comparative analysis of current intrusion detection technologies. In *Proceedings of the Fourth Technology for Information Security Conference*, volume 96. Citeseer, 1996.

[20] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Technical report Chalmers University of Technology, Goteborg, Sweden, 2000.

[21] M. Salour and Xiao Su. Dynamic two-layer signature-based ids with unequal databases. In *Information Technology, 2007. ITNG '07. Fourth International Conference on*, pages 77–82, April 2007. doi: 10.1109/ITNG.2007.80.

[22] P. Gupta, C. Raissi, G. Dray, P. Poncelet, and J. Brissaud. Ss-ids: Statistical signature based ids. In *Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on*, pages 407–412, May 2009. doi: 10.1109/ICIW.2009.67.

[23] O. Linda, T. Vollmer, and M. Manic. Neural network based intrusion detection system for critical infrastructures. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 1827–1834, June 2009. doi: 10.1109/IJCNN.2009.5178592.

[24] N.B. Aissa and M. Guerroumi. A genetic clustering technique for anomaly-based intrusion detection systems. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on*, pages 1–6, June 2015. doi: 10.1109/SNPD.2015.7176182.

[25] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014.

[26] Ismail Butun, Salvatore D Morgera, and Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 16(1): 266–282, 2014.

[27] Evangelos P Markatos, Spyros Antonatos, Michalis Polychronakis, and Kostas G Anagnostakis. Exclusion-based signature matching for intrusion detection. In *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN)*, pages 146–152, 2002.

[28] Sandeep Kumar and Eugene H Spafford. A pattern matching model for misuse intrusion detection. 1994.

[29] Koral Ilgun. Ustat: A real-time intrusion detection system for unix. In *Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on*, pages 16–28. IEEE, 1993.

[30] R. Rangadurai Karthick, V.P. Hattiwale, and B. Ravindran. Adaptive network intrusion detection system using a hybrid approach. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*, pages 1–7, Jan 2012. doi: 10.1109/COMSNETS.2012.6151345.

[31] Debra Anderson, Thane Frivold, and Alfonso Valdes. *Next-generation intrusion detection expert system (NIDES): A summary*. SRI International, Computer Science Laboratory Menio Park, CA, 1995.

[32] Xingchao Gong and Xin Guan. Intrusion detection model based on the improved neural network and expert system. In *Electrical Electronics Engineering (EEESYM), 2012 IEEE Symposium on*, pages 191–193, June 2012. doi: 10.1109/EEESym.2012.6258621.

[33] Paul Helman and Gunar Liepins. Statistical foundations of audit trail analysis for the

detection of computer misuse. *Software Engineering, IEEE Transactions on*, 19(9): 886–901, 1993.

[34] Zheng Zhang, Jun Li, CN Manikopoulos, Jay Jorgenson, and Jose Ucles. Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *Proc. IEEE Workshop on Information Assurance and Security*, pages 85–90, 2001.

[35] Federico Simmross-Wattenberg, Juan Ignacio Asensio-Pérez, Pablo Casaseca-de-la Higuera, Marcos Martin-Fernandez, Ioannis Dimitriadis, Carlos Alberola-Lopez, et al. Anomaly detection in network traffic based on statistical inference and alpha-stable modeling. *Dependable and Secure Computing, IEEE Transactions on*, 8(4):494–509, 2011.

[36] Holger Dreger, Anja Feldmann, Michael Mai, Vern Paxson, and Robin Sommer. Dynamic application-layer protocol analysis for network intrusion detection. In *USENIX Security*, 2006.

[37] Zhang Zhi. Ipv6 network intrusion detection protocol analysis techniques. In *Proceedings of the 2012 International Conference of Modern Computer Science and Applications*, pages 89–94. Springer, 2013.

[38] Hui Lin, Adam Slagell, Catello Di Martino, Zbigniew Kalbarczyk, and Ravishankar K Iyer. Adapting bro into scada: building a specification-based intrusion detection system for the dnp3 protocol. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, page 5. ACM, 2013.

[39] Guisong Liu, Zhang Yi, and Shangming Yang. A hierarchical intrusion detection model based on the pca neural networks. *Neurocomputing*, 70(7):1561–1568, 2007.

[40] Morteza Amini, Rasool Jalili, and Hamid Reza Shahriari. Rt-unnid: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. *Computers & Security*, 25(6):459–468, 2006.

[41] Jacinth Salome and Ramya Ravishankar. Fuzzy data mining and genetic algorithms applied to intrusion detection. *i-Manager's Journal on Software Engineering*, 1(4):23, 2007.

[42] Susan M Bridges and Rayford B Vaughn. Fuzzy data mining and genetic algorithms applied to intrusion detection. In *Proceedings of 12th Annual Canadian Information Technology Security Symposium*, pages 109–122, 2000.

[43] Farzaneh Geramiraz, Amir Saman Memaripour, and Maghsoud Abbaspour. Adaptive anomaly-based intrusion detection system using fuzzy controller. *IJ Network Security*, 14(6):352–361, 2012.

[44] Sevil Sen and John A Clark. Evolutionary computation techniques for intrusion detection in mobile ad hoc networks. *Computer Networks*, 55(15):3441–3457, 2011.

[45] Yaping Jiang and Junlin Chang. Intrusion prevention system base on immune vaccination. In *Intelligent Computation Technology and Automation, 2009. ICICTA'09. Second International Conference on*, volume 1, pages 350–353. IEEE, 2009.

[46] Yaping Jiang and Junwei Zhao. A dynamic model of intrusion prevention based on vaccine. In *Biomedical Engineering and Informatics (BMEI), 2013 6th International Conference on*, pages 575–579. IEEE, 2013.

[47] Haidong Fu and Chunxiang Zhang. Design of a danger signal detecting model based on fuzzy-set. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pages 1–3, Sept 2009. doi: 10.1109/WICOM.2009.5302569.

[48] Hai-Hua Gao, Hui hua Yang, and Xing-Yu Wang. Ant colony optimization based network intrusion feature selection and detection. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 6, pages 3871–3875 Vol. 6, Aug 2005. doi: 10.1109/ICMLC.2005.1527615.

[49] Mohammad Saniee Abadeh and Jafar Habibi. A hybridization of evolutionary fuzzy systems and ant colony optimization for intrusion detection. *The ISC International Journal of Information Security*, 2(1), 2015.

[50] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A new intrusion detection system using support vector machines and hierarchical clustering. *The VLDB JournalThe International Journal on Very Large Data Bases*, 16(4):507–521, 2007.

[51] B. Senthilnayaki, K. Venkatalakshmi, and A. Kannan. Intrusion detection using optimal genetic feature selection and svm based classifier. In *Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on*, pages 1–4, March 2015. doi: 10.1109/ICSCN.2015.7219890.

[52] Xiaokui Shu, Danfeng Daphne Yao, and Naren Ramakrishnan. Unearthing stealthy program attacks buried in extremely long execution paths. In *Proceedings of ACM CCS*, 2015.

[53] Ali Borji. Combining heterogeneous classifiers for network intrusion detection. In *Advances in Computer Science-ASIAN 2007. Computer and Network Security*, pages 254–260. Springer, 2007.

[54] Srilatha Chebrolu, Ajith Abraham, and Johnson P Thomas. Feature deduction and ensemble design of intrusion detection systems. *Computers & Security*, 24(4):295–307, 2005.

[55] S.O. Amin, M.S. Siddiqui, Choong Seon Hong, and Jongwon Choe. A novel coding scheme to implement signature based ids in ip based sensor networks. In *Integrated Network Management-Workshops, 2009. IM '09. IFIP/IEEE International Symposium on*, pages 269–274, June 2009. doi: 10.1109/INMW.2009.5195973.

[56] AS Aneetha, TS Indhu, and S Bose. Hybrid network intrusion detection system using expert rule based approach. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, pages 47–51. ACM, 2012.

[57] Gisung Kim, Seungmin Lee, and Sehun Kim. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4):1690–1700, 2014.

[58] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

[59] Hu Min and Wu Fangfang. Filter-wrapper hybrid method on feature selection. In *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, volume 3, pages 98–101, Dec 2010. doi: 10.1109/GCIS.2010.235.

[60] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[61] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

[62] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

[63] Daphne Koller and Mehran Sahami. Toward optimal feature selection. 1996.

[64] Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[65] UNIBS. University of brescia data set. `http://www.ing.unibs.it/ntw/tools/traces/`, 2009.

[66] UNB. University of new brunswick (unb) iscx data set. `http://www.unb.ca/research/iscx/dataset/iscx-dataset.html`, 2010.

[67] KDDCUP99. Kdd cup99 data set. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, 1999.

[68] NSL-KDD. Nsl-kdd data set for network-based intrusion detection systems. `http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html`, 2009.

[69] Massachusetts Institute of Technology. Defense advanced research projects agency dataset (darpa). http://www.ll.mit.edu/ideval/data, 2000.

[70] UNSW. University of new south wales data set. http://www.cybersecurity.unsw.adfa.edu.au/ADFA%20NB15%20Datasets.

[71] LBNL and ICSI. Lawrence berkeley national laboratory and international computer science institute data set. http://www.icir.org/enterprise-tracing/Overview.html, 2005.

[72] CAIDA. Center for applied internet data analysis data set. http://www.caida.org/data/overview/, 2015.

[73] AWID. Awid-wireless security datasets project data set. http://icsdweb.aegean.gr/awid/features.html, 2014.

[74] Hervé Debar and Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. In *Recent Advances in Intrusion Detection*, pages 85–103. Springer, 2001.

[75] A. Hofmann and B. Sick. Online intrusion alert aggregation with generative data stream modeling. *IEEE Transactions on Dependable and Secure Computing*, 8(2):282–294, March 2011. ISSN 1545-5971. doi: 10.1109/TDSC.2009.36.

[76] Guo Fan, Ye JiHua, and Yu Min. Design and implementation of a distributed ids alert aggregation model. In *Computer Science & Education, 2009. ICCSE'09. 4th International Conference on*, pages 975–980. IEEE, 2009.

[77] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A Kemmerer. Comprehensive approach to intrusion detection alert correlation. *Dependable and Secure Computing, IEEE Transactions on*, 1(3):146–169, 2004.

[78] Fu Xiao, Shi Jin, and Xie Li. A novel data mining-based method for alert reduction and analysis. *Journal of networks*, 5(1):88–97, 2010.

[79] Peng Ning, Yun Cui, and Douglas S Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 245–254. ACM, 2002.

[80] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer. Comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, July 2004. ISSN 1545-5971. doi: 10.1109/TDSC.2004.21.

[81] Xinzhou Qin and Wenke Lee. Attack plan recognition and prediction using causal networks. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 370–379. IEEE, 2004.

[82] Natalia Stakhanova, Samik Basu, and Johnny Wong. A taxonomy of intrusion response systems. *International Journal of Information and Computer Security*, 1(1-2):169–184, 2007.

[83] Laura Feinstein, Dan Schnackenberg, Ravindra Balupari, and Darrell Kindred. Statistical approaches to ddos attack detection and response. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 303–314. IEEE, 2003.

[84] D.J. Ragsdale, C.A.Jr. Carver, J.W. Humphries, and U.W. Pooch. Adaptation techniques for intrusion detection and intrusion response systems. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 4, pages 2344–2349 vol.4, 2000. doi: 10.1109/ICSMC.2000.884341.

[85] C Carver, JM Hill, John R Surdu, and Udo W Pooch. A methodology for using intelligent agents to provide automated intrusion response. In *Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, West Point, NY*, pages 110–116, 2000.

[86] Wenke Lee, Wei Fan, Matthew Miller, Salvatore Stolfo, and Erez Zadok. Toward cost-sensitive modeling for intrusion detection and response. 2000.

[87] Ivan Balepin, Sergei Maltsev, Jeff Rowe, and Karl Levitt. Using specification-based intrusion detection for automated response. In *Recent Advances in Intrusion Detection*, pages 136–154. Springer, 2003.

[88] Xinzhou Qin and Wenke Lee. Statistical causality analysis of infosec alert data. In *Recent Advances in Intrusion Detection*, pages 73–93. Springer, 2003.

[89] Ali Ahmadian Ramaki, Masoud Khosravi-Farmad, and Abbas Ghaemi Bafghi. Real time alert correlation and prediction using bayesian networks. In *Information Security and Cryptology (ISCISC), 2015 12th International Iranian Society of Cryptology Conference on*, pages 98–103. IEEE, 2015.

[90] Bin Zhu and Ali A Ghorbani. Alert correlation for extracting attack strategies. *IJ Network Security*, 3(3):244–258, 2006.

[91] Frédéric Cuppens and Rodolphe Ortalo. Lambda: A language to model a database for detection of attacks. In *Recent advances in intrusion detection*, pages 197–216. Springer, 2000.

[92] Cédric Michel and Ludovic Mé. Adele: an attack description language for knowledge-based intrusion detection. In *Trusted Information*, pages 353–368. Springer, 2002.

[93] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An attack graph-based probabilistic security metric. In *Data and applications security XXII*, pages 283–296. Springer, 2008.

[94] B. C. Cheng, G. T. Liao, C. C. Huang, and M. T. Yu. A novel probabilistic matching algorithm for multi-stage attack forecasts. *IEEE Journal on Selected Areas in Communications*, 29(7):1438–1448, August 2011. ISSN 0733-8716. doi: 10.1109/JSAC.2011.110809.

[95] Richard Paul Lippmann and Kyle William Ingols. An annotated review of past papers on attack graphs. Technical report, DTIC Document, 2005.

[96] Anders Krogh, Michael Brown, I Saira Mian, Kimmen Sjölander, and David Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of molecular biology*, 235(5):1501–1531, 1994.

[97] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989. ISSN 0018-9219. doi: 10.1109/5.18626.

[98] Rene Garcia and Pierre Perron. An analysis of the real interest rate under regime shifts. *The Review of Economics and Statistics*, pages 111–125, 1996.

[99] Antonello Panuccio, Manuele Bicego, and Vittorio Murino. A hidden markov model-based approach to sequential data clustering. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 734–743. Springer, 2002.

[100] Hisham A Kholidy, Abdelkarim Erradi, Sherif Abdelwahed, Ahmed M Yousof, and Hisham Arafat Ali. Online risk assessment and prediction models for autonomic cloud intrusion srevention systems. In *Computer Systems and Applications (AICCSA), 2014 IEEE/ACS 11th International Conference on*, pages 715–722. IEEE, 2014.

[101] M. Tavallaee, E. Bagheri, Wei Lu, and A.A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–6, July 2009. doi: 10.1109/CISDA.2009.5356528.

[102] Maheshkumar Sabhnani and Gursel Serpen. Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set. *Intelligent Data Analysis*, 8(4): 403–415, 2004.

[103] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[104] Leo Breiman and Adele Cutler. Random forests - classification description. `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`, 2015.

[105] Eibe Frank and Richard Kirkby. Random tree. http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/RandomTree.html, 2016.

[106] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[107] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[108] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[109] Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[110] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[111] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[112] Snort.org. The snort intrusion detection system. https://www.snort.org/, 2015.

[113] N. Khamphakdee, N. Benjamas, and S. Saiyod. Improving intrusion detection system based on snort rules for network probe attack detection. In *Information and Communication Technology (ICoICT), 2014 2nd International Conference on*, pages 69–74, May 2014. doi: 10.1109/ICoICT.2014.6914042.

[114] Noah Dietrich. Snort 2.9.8.x on ubuntu 12, 14, and 15 with barnyard2, pulledpork, and snorby. https://s3.amazonaws.com/snort-org-site/production/document_files/files/000/000/090/original/Snort_2.9.8.x_on_Ubuntu_12-14-15.pdf, 2015. Accessed: 2016-6-22.

[115] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[116] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31 (8):651–666, 2010.

[117] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[118] Evgeniy Gabrilovich and Shaul Markovitch. Text categorization with many redundant features: using aggressive feature selection to make svms competitive with c4. 5. In *Proceedings of the twenty-first international conference on Machine learning*, page 41. ACM, 2004.

[119] Wireshark filter field reference. `https://www.wireshark.org/docs/dfref/`, 2015.

[120] Martin Roesch and Chris Green. Snort users manual 2.9.8.2. `http://manual-snort-org.s3-website-us-east-1.amazonaws.com/snort_manual.html`, March 2016. Accessed: 2016-6-22.

[121] Lance Spitzner. *Honeypots: tracking hackers*, volume 1. Addison-Wesley Reading, 2003.

Appendix A

Hidden Markov Model

The Hidden Markov Model (HMM) is stochastic model which was introduced in late 1960s by Baum and his colleagues [17], [18]. Due to rich mathematical structure, hidden Markov models are widely applied in real world applications such as speech recognition, handwriting recognition, gesture recognition, intrusion detection, speech tagging and bioinformatics.

A hidden Markov model is used to model sequential of observations that can be observed over the time. Also there are underlying state sequences which are not observed, that produce these observations. These states are defined as hidden states. As an example, we can consider rainy, cloudy, stormy and sunny as observations and high, medium and low atmospheric pressure as hidden states which we cannot observe. Figure A.1 shows an example of hidden Markov model showing weather conditions with state transition probabilities and observation emission probabilities. As an example, state transition probability from pressure high to pressure low is 0.3. The sunny observation has 0.4 emission probability in pressure high state and 0.3 emission probability in pressure low state.

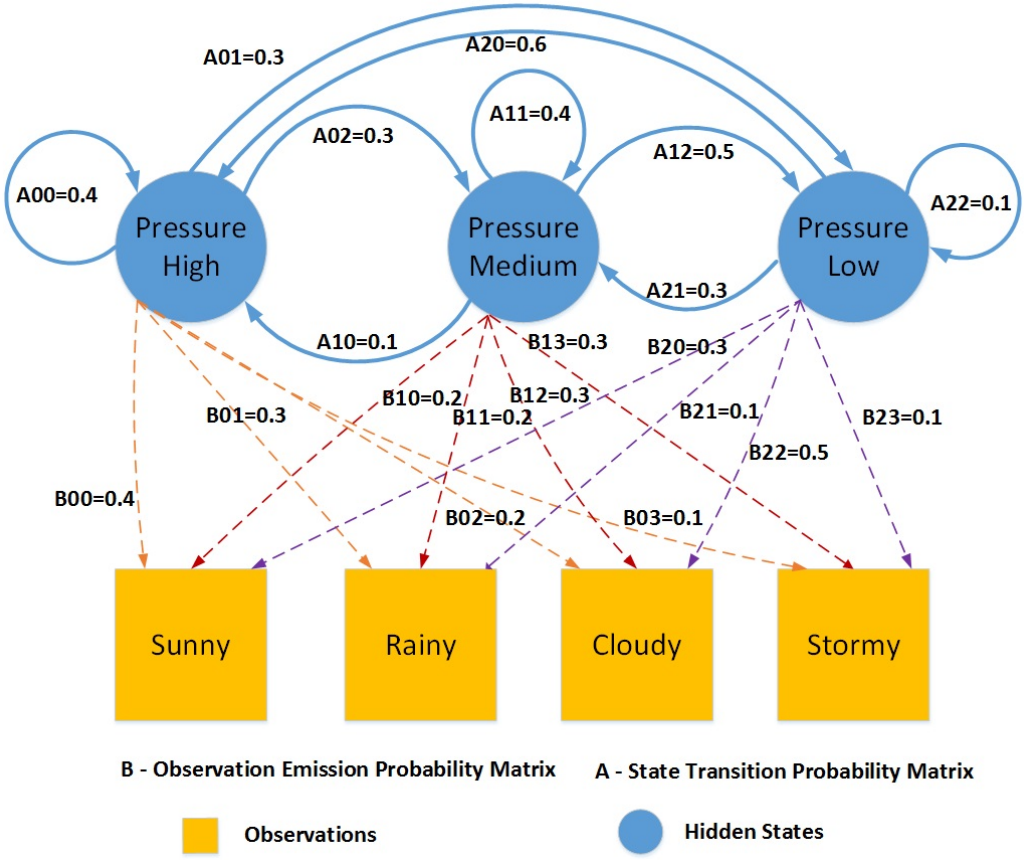


Figure A.1: An Example of Hidden Markov Model Showing Weather Condition.

Hidden Markov model holds following basic properties.

1. Markov chain property. There are finite number of states in the system where next state depends on the previous state of the system.

2. After state is changed, observation output is produced based on the observation probability distribution of that state.

- **Elements of the Hidden Markov Model**

Elements of the HMM can be described as:

1. N, Number of hidden states in the system.

   Where individual states are denoted as $\mathbf{S} = S_1, S_2, \ldots, S_N$ and state at time t denote as q_t.

2. M, Number of distinct observation symbols per state.

   Where individual symbols are denoted as $\mathbf{V} = v_1, v_2, \ldots, v_M$.

3. State Transition Probability (A) NxN matrix.

   Where a_{ij} represents the state transition probability form state i to state j.

   $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$, *where* $1 \leq i \leq N$ *and* $1 \leq j \leq N$.

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1N-1} & a_{1N} \\ a_{21} & a_{22} & \ldots & a_{2N-1} & a_{2N} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{N1} & a_{N2} & \ldots & a_{NN-1} & a_{NN} \end{bmatrix}.$$

4. Observation emission probability (B) NxM matrix.

   Where k^{th} observation emission probability of the state j is represented by $b_j(k)$.

   $b_j(k) = P(v_k \; at \; t | q_t = S_j)$, *where* $1 \leq j \leq N, 1 \leq k \leq M$.

$$B = \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1N-1} & b_{1M} \\ b_{21} & a_{22} & \ldots & b_{2N-1} & b_{2M} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ b_{N1} & b_{N2} & \ldots & b_{NN-1} & b_{NM} \end{bmatrix}.$$

5. Initial state probability distribution (π).

   Where π represents the initial states probability of the system.

   $\pi_i = P(q_1 = S_i), 1 \leq i \leq N$.

$$\pi = \begin{bmatrix} \pi_1 & \pi_2 & \ldots & \pi_{N-1} & \pi_N \end{bmatrix}.$$

6. Observation sequence (O).

   The observation sequence of length T is represented as $\mathbf{O} = O_1, O_2, \ldots, O_t, \ldots, O_T$,

   Where O_t is one of observation symbol from $\mathbf{V}$.

Hidden Markov Model is typically represented by using A, B and π parameters and model is denoted as $\lambda = (A, B, \pi)$.

A.1 Three Basic Problems of the Hidden Markov Model

There are three basic problems that associate with hidden Markov model [97].

1. Evaluation. When hidden Markov model (π) and observation sequence $(O_1, O_2, \ldots, O_t, \ldots, O_T)$ is given how we compute the probability of observation sequence $P(O|\lambda)$?

2. Decoding. When hidden Markov model and observation sequence $(O_1, O_2, \ldots, O_t, \ldots, O_T)$ is given how we find optimal hidden state sequence that explains the observation sequence ($\mathbf{Q} = q_1, q_2, \ldots, q_t, \ldots, q_T$)?

3. Hidden Markov Model Learning. When observation sequence is given how do we find hidden Markov model parameters that maximize probability of observation sequence $P(O|\lambda)$?

1. Solution to Problem 1. The Forward Algorithm.
   Consider HMM $\lambda = (A, B, \pi)$ and observation sequence $(O_1, O_2, \ldots, O_t, \ldots, O_T)$ is given, we want to calculate the probability of observation sequence. Let hidden state sequence is $\mathbf{Q} = q_1, q_2, \ldots, q_t, \ldots, q_T$.
   Forward algorithm defines forward variable $\alpha_t(i)$ as:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, \ldots, O_T, q_t = S_i|\lambda). \tag{A.1}$$

   $\alpha_t(i)$ can be compute recursively as:
   Step 1. Initialization- Calculate $\alpha_1(i)$:

$$\alpha_1(i) = \pi_i \, b_i(O_1), \quad 1 \leq i \leq N. \tag{A.2}$$

Step 2. Induction- Calculate $\alpha_{t+1}(j)$:

$$\alpha_{t+1}(j) = \Big[\sum_{i=1}^{N} \alpha_t(i)\, a_{ij} \Big] b_j\,(O_{t+1}), \quad 1 \le t \le T-1, 1 \le j \le N. \tag{A.3}$$

Step 3. Termination- Calculate P($O|\lambda$):

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{A.4}$$

The forward algorithm required only N^2T multiplications.

2. Solution to Problem 2. The Viterbi Algorithm The Viterbi algorithm is used to find the single best hidden sequence for given observation with a hidden Markov model. The other method of finding best hidden state sequence is forward-backward algorithm, it finds the best hidden state at each step, but the problem of this approach is in some cases this may not be the best hidden state sequence, since there are possibilities that transition from one state to another state is invalid (if $a_{ij} = 0$ then there is no transition from state i to state j). The Viterbi algorithm define value $v_t(j)$ as:

$$v_t(j) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1, q_2, \ldots, q_{t-1},\ q_t = j, O_1, O_2, \ldots, O_t|\lambda]. \tag{A.5}$$

$v_t(j)$ is the best score along the single path at time t, for a given state q_i at time t-1 the value of the $v_t(j)$ can be compute as:

$$v_t(j) = \max_{i=1,2,\ldots,N} [v_{t-1}(i)\, a_{ij}]\, b_j(O_t). \tag{A.6}$$

We need to maximize this value ($v_t(j)$) for each value of t and j, we can do it by finding the maximum value of $v_t(j)$ over all possible previous states.

Steps of the Viterbi algorithm defined as:

Step 1. Initialization:

$$v_1(j) = \pi_i\, b_i(O_1), \quad for \quad 1 \le i \le N. \tag{A.7}$$

$$\psi_1(j) = 0. \tag{A.8}$$

Step 2. Recursion:

$$v_t(j) = \max_{1 \le i \le N} \left[v_{t-1}(j)\, a_{ij} \right] b_j(O_t), \quad for \quad 2 \le t \le T, \quad 1 \le j \le N. \tag{A.9}$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N} \left[v_{t-1}(j)\, a_{ij} \right], \quad for \quad 2 \le t \le T, \quad 1 \le j \le N. \tag{A.10}$$

Step 3. Termination: The best score:

$$P^* = \max_{1 \le i \le N} [v_t(i)]. \tag{A.11}$$

The start of back trace:

$$q_T^* = \operatorname*{argmax}_{1 \le i \le N} [v_t(i)]. \tag{A.12}$$

The state sequence (backtracking):

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad for \quad t = T-1, T-2, ..., 1. \tag{A.13}$$

3. Solution to Problem 3. BaumWelch Algorithm.

The most difficult problem of hidden Markov model is the training process, where we need to find model parameters A, B, π for a given observation sequence. Assuming we know the hidden state sequence for a given observation, we can compute state change probability by considering the maximum likelihood as:

$$a_{ij} = \frac{\text{Total number of transition from state i to j}}{\text{Total number of transition from state i to any state}}.$$

But we cannot directly compute this because we don't know which hidden state that we are in. So Baum-Welch algorithm uses iteration and re-estimation process to find model parameters. We define variable backward probability (β) as:

$$\beta_t(i) = P(O_{t+1}, \ldots, O_T | q_t = S_i, \lambda). \tag{A.14}$$

β can be find using recursively using following steps,

Step 1. Let $\beta_T(j) = 1$, for $j = 0, 1, 2, ..., N$.

Step 2. Compute $\beta_t(j)$ as:

$$\beta_t(i) = \sum_{j=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j),\ For\, t = T-1, T-2, ..., 0\, and\, j = 0, 1, ..., N. \tag{A.15}$$

Define $\gamma_t(j)$ which represents the probability of being in a state S_i at time t,

$$\gamma_t(j) = P(q_t = S_j|O, \lambda); \quad for \quad 0 \leq t \leq T\, and\, 0 \leq j \leq N. \tag{A.16}$$

Where α gives the relevant probability up to time t and β gives the probability after time t. We can express γ as:

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}. \tag{A.17}$$

Probability of transfer from state S_i (at time t) to S_j (at time t+1) is defined as $\xi_t(i, j)$. Then $\xi_t(i, j)$ can be calculated as:

$$\xi_t(i, j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}. \tag{A.18}$$

From the definition of $\gamma_t(i)$ and $\xi_t(i, j)$, we can find the relationship between them as:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j). \tag{A.19}$$

From the definition of $\gamma_t(i)$ and $\xi_t(i, j)$, HMM model parameters (A, B, π) can be estimate as:

Let number of hidden states in the system = N and number of distinct observation symbols per state = M.

Expected value of initial probability($\hat{\pi}$).

$$\hat{\pi} = \begin{bmatrix} \hat{\pi}_1 & \hat{\pi}_2 & \dots & \hat{\pi}_{N-1} & \hat{\pi}_N \end{bmatrix}.$$

Where $\hat{\pi}_i = \gamma_1(i)$, for i = 1, 2, ..., N.

State transition probability matrix (A) can be calculated as:

$$\hat{a}_{ij} = \frac{\text{Expected total number of transition from state } S_i \text{to} S_j}{\text{Expected total number of transition from state } S_i \text{ to any state}}. \tag{A.20}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \; for\, i = 1, 2, ..., N \, and \, j = 1, 2, ..., N. \tag{A.21}$$

Observation emission probability (B) can be calculated as:

$$\hat{b_j}(k) = \frac{\text{Expected total number of time in state } S_j \text{ and observing symbol } v_k}{\text{Expected total number of times in state } S_j}. \tag{A.22}$$

$$\hat{b_j}(k) = \sum_{\substack{t=1 \\ O_t = v_k}}^{T} \gamma_t(j) \Big/ \sum_{t=1}^{T} \gamma_t(j), \; for\, i = 1, 2, ..., N \, and \, k = 1, 2, ..., M. \tag{A.23}$$

Re-estimation process:

Step 1. Initialized model with best guess values for A, B and π.

Step 2. Compute $\alpha_t(i)$, $\beta_t(i)$, $\xi_t(i,j)\, and\, \gamma_t(i)$.

Step 3. Re-estimate the model $\lambda = (A, B, \pi)$.

Step 4. If P($O|\lambda$) increases (higher than given threshold value), go to step 2, otherwise stop the process.

A.2 Hidden Markov Model Scaling Issues

It can be observed that with the increment of the T, $\alpha_t(i)$ tends to approach to zero exponentially (since both A and B contains probability values. So many numbers of multiplication of a_{ij} and b_j become zero), this result in underflow of HMM calculations. To avoid the underflow, a scaling factor is introduced for the calculation of $\alpha_t(i)$ and $\beta_t(i)$. Value of $\alpha_t(i)$ and $\beta_t(i)$ are normalized during the iteration process to avoid these values becoming zero. All equations involved in the process of calculating $\alpha_t(i)$ and $\beta_t(i)$ need to be modified with scaling factor. More information about this modification is available in the Rabiner HMM tutorial paper [97].

Appendix B

Aegean Wi-Fi Intrusion Dataset (AWID)

The Aegean Wi-Fi Intrusion Dataset (AWID) is a publicly available labeled dataset which was developed based on real traces of both normal and intrusion activities of an 802.11 Wi-Fi network under the supervision of University of the Aegean and George Mason University [73]. Each record of the AWID dataset is classified as either normal or a specific intrusion type (i.e., class attribute of a record is referred to a type of intrusion or normal network activity). The intrusion types include in the AWID datasets are shown in table B.1 [73]. Description about these attributes (features) can be found in [119]. The attributes (features) include in the AWID datasets are shown in table B.2 [73].

Table B.1: The intrusion types include in the AWID datasets [73].

No	Intrusion	Description
1	Amok	An Increased numbers of 802.11 Authentication Requests is noticed in Amok
2	Arp	It may be used as a first step for any of the Key cracking attacks
3	Authentication request	802.11 DoS Attack
4	Beacon	802.11 DoS Attack
5	Cafe latte	802.11 Keystream Retrieving Attacks
6	Chop chop	802.11 Keystream Retrieving Attacks
7	Cts	802.11 DoS Attack
8	Deauthentication	802.11 DoS Attack
9	Deauthentication	802.11 DoS Attack
10	Disassociation	802.11 DoS Attack
11	Evil twin	802.11 Man-in-the-Middle
12	Fragmentation	802.11 Keystream Retrieving Attacks
13	Hirte	802.11 Keystream Retrieving Attacks
14	Power saving	802.11 DoS Attack
15	Probe request	802.11 DoS Attack
16	Probe response	802.11 DoS Attack
17	Rts	802.11 DoS Attack

Table B.2: The attributes (features) in the AWID datasets [73].

ID	Attribute	ID	Attribute
1	frame.interface_id	29	radiotap.present.rxflags
2	frame.dlt	30	radiotap.present.xchannel
3	frame.offset_shift	31	radiotap.present.mcs
4	frame.time_epoch	32	radiotap.present.ampdu
5	frame.time_delta	33	radiotap.present.vht
6	frame.time_delta_displayed	34	radiotap.present.reserved
7	frame.time_relative	35	radiotap.present.rtap_ns
8	frame.len	36	radiotap.present.vendor_ns
9	frame.cap_len	37	radiotap.present.ext
10	frame.marked	38	radiotap.mactime
11	frame.ignored	39	radiotap.flags.cfp
12	radiotap.version	40	radiotap.flags.preamble
13	radiotap.pad	41	radiotap.flags.wep
14	radiotap.length	42	radiotap.flags.frag
15	radiotap.present.tsft	43	radiotap.flags.fcs
16	radiotap.present.flags	44	radiotap.flags.datapad
17	radiotap.present.rate	45	radiotap.flags.badfcs
18	radiotap.present.channel	46	radiotap.flags.shortgi
19	radiotap.present.fhss	47	radiotap.datarate
20	radiotap.present.dbm_antsignal	48	radiotap.channel.freq
21	radiotap.present.dbm_antnoise	49	radiotap.channel.type.turbo
22	radiotap.present.lock_quality	50	radiotap.channel.type.cck
23	radiotap.present.tx_attenuation	51	radiotap.channel.type.ofdm
24	radiotap.present.db_tx_attenuation	52	radiotap.channel.type.2ghz
25	radiotap.present.dbm_tx_power	53	radiotap.channel.type.5ghz
26	radiotap.present.antenna	54	radiotap.channel.type.passive
27	radiotap.present.db_antsignal	55	radiotap.channel.type.dynamic
28	radiotap.present.db_antnoise	56	radiotap.channel.type.gfsk

ID	Attribute	ID	Attribute
57	radiotap.channel.type.gsm	85	wlan.ba.control.multitid
58	radiotap.channel.type.sturbo	86	wlan.ba.control.cbitmap
59	radiotap.channel.type.half	87	wlan.bar.compressed.tidinfo
60	radiotap.channel.type.quarter	88	wlan.ba.bm
61	radiotap.dbm_antsignal	89	wlan.fcs_good
62	radiotap.antenna	90	wlan_mgt.fixed.capabilities.ess
63	radiotap.rxflags.badplcp	91	wlan_mgt.fixed.capabilities.ibss
64	wlan.fc.type_subtype	92	wlan_mgt.fixed.capabilities.cfpoll.ap
65	wlan.fc.version	93	wlan_mgt.fixed.capabilities.privacy
66	wlan.fc.type	94	wlan_mgt.fixed.capabilities.preamble
67	wlan.fc.subtype	95	wlan_mgt.fixed.capabilities.pbcc
68	wlan.fc.ds	96	wlan_mgt.fixed.capabilities.agility
69	wlan.fc.frag	97	wlan_mgt.fixed.capabilities.spec_man
70	wlan.fc.retry	98	wlan_mgt.fixed.capabilities.short_slot_time
71	wlan.fc.pwrmgt	99	wlan_mgt.fixed.capabilities.apsd
72	wlan.fc.moredata	100	wlan_mgt.fixed.capabilities.radio_measurement
73	wlan.fc.protected	101	wlan_mgt.fixed.capabilities.dsss_ofdm
74	wlan.fc.order	102	wlan_mgt.fixed.capabilities.del_blk_ack
75	wlan.duration	103	wlan_mgt.fixed.capabilities.imm_blk_ack
76	wlan.ra	104	wlan_mgt.fixed.listen_ival
77	wlan.da	105	wlan_mgt.fixed.current_ap
78	wlan.ta	106	wlan_mgt.fixed.status_code
79	wlan.sa	107	wlan_mgt.fixed.timestamp
80	wlan.bssid	108	wlan_mgt.fixed.beacon
81	wlan.frag	109	wlan_mgt.fixed.aid
82	wlan.seq	110	wlan_mgt.fixed.reason_code
83	wlan.bar.type	111	wlan_mgt.fixed.auth.alg
84	wlan.ba.control.ackpolicy	112	wlan_mgt.fixed.auth_seq

ID	Attribute
113	wlan_mgt.fixed.category_code
114	wlan_mgt.fixed.htact
115	wlan_mgt.fixed.chanwidth
116	wlan_mgt.fixed.fragment
117	wlan_mgt.fixed.sequence
118	wlan_mgt.tagged.all
119	wlan_mgt.ssid
120	wlan_mgt.ds.current_channel
121	wlan_mgt.tim.dtim_count
122	wlan_mgt.tim.dtim_period
123	wlan_mgt.tim.bmapctl.multicast
124	wlan_mgt.tim.bmapctl.offset
125	wlan_mgt.country_info.environment
126	wlan_mgt.rsn.version
127	wlan_mgt.rsn.gcs.type
128	wlan_mgt.rsn.pcs.count
129	wlan_mgt.rsn.akms.count
130	wlan_mgt.rsn.akms.type
131	wlan_mgt.rsn.capabilities.preauth
132	wlan_mgt.rsn.capabilities.no_pairwise
133	wlan_mgt.rsn.capabilities.ptksa_replay_counter
134	wlan_mgt.rsn.capabilities.gtksa_replay_counter
135	wlan_mgt.rsn.capabilities.mfpr
136	wlan_mgt.rsn.capabilities.mfpc
137	wlan_mgt.rsn.capabilities.peerkey
138	wlan_mgt.tcprep.trsmt_pow
139	wlan_mgt.tcprep.link_mrg
140	wlan.wep.iv
141	wlan.wep.key
142	wlan.wep.icv
143	wlan.tkip.extiv
144	wlan.ccmp.extiv
145	wlan.qos.tid
146	wlan.qos.priority
147	wlan.qos.eosp
148	wlan.qos.ack
149	wlan.qos.amsdupresent
150	wlan.qos.buf_state_indicated
151	wlan.qos.bit4
152	wlan.qos.txop_dur_req
153	wlan.qos.buf_state_indicated
154	data.len
155	class

Table B.3: The selected attributes (features) in the AWID datasets for the exeperiments.

ID	Attribute	ID	Attribute
1	data_len	41	radiotap_present_channel
2	frame_cap_len	42	radiotap_present_db_antnoise
3	frame_dlt	43	radiotap_present_db_antsignal
4	frame_interface_id	44	radiotap_present_db_tx_attenuation
5	frame_len	45	radiotap_present_dbm_antnoise
6	frame_marked	46	radiotap_present_dbm_antsignal
7	frame_offset_shift	47	radiotap_present_dbm_tx_power
8	frame_time_delta	48	radiotap_present_ext
9	frame_time_delta_displayed	49	radiotap_present_fhss
10	frame_time_epoch	50	radiotap_present_flags
11	frame_time_relative	51	radiotap_present_lock_quality
12	radiotap_antenna	52	radiotap_present_mcs
13	radiotap_channel_freq	53	radiotap_present_rate
14	radiotap_channel_type_2ghz	54	radiotap_present_rtap_ns
15	radiotap_channel_type_5ghz	55	radiotap_present_rxflags
16	radiotap_channel_type_cck	56	radiotap_present_tsft
17	radiotap_channel_type_dynamic	57	radiotap_present_tx_attenuation
18	radiotap_channel_type_gfsk	58	radiotap_present_vendor_ns
19	radiotap_channel_type_gsm	59	radiotap_present_vht
20	radiotap_channel_type_half	60	radiotap_present_xchannel
21	radiotap_channel_type_ofdm	61	radiotap_rxflags_badplcp
22	radiotap_channel_type_passive	62	radiotap_version
23	radiotap_channel_type_quarter	63	wlan_ba_control_ackpolicy
24	radiotap_channel_type_sturbo	64	wlan_duration
25	radiotap_channel_type_turbo	65	wlan_fc_order
26	radiotap_datarate	66	wlan_fc_type
27	radiotap_dbm_antsignal	67	wlan_fc_version
28	radiotap_flags_badfcs	68	wlan_frag
29	radiotap_flags_cfp	69	wlan_mgt_ds_current_channel
30	radiotap_flags_datapad	70	wlan_mgt_fixed_auth_alg
31	radiotap_flags_fcs	71	wlan_mgt_fixed_beacon
32	radiotap_flags_frag	72	wlan_mgt_fixed_capabilities_agility
33	radiotap_flags_preamble	73	wlan_mgt_fixed_capabilities_apsd
34	radiotap_flags_shortgi	74	wlan_mgt_fixed_capabilities_del_blk_ack
35	radiotap_flags_wep	75	wlan_mgt_fixed_capabilities_dsss_ofdm
36	radiotap_length	76	wlan_mgt_fixed_capabilities_ibss
37	radiotap_mactime	77	wlan_mgt_fixed_capabilities_imm_blk_ack
38	radiotap_pad	78	wlan_mgt_fixed_capabilities_pbcc
39	radiotap_present_ampdu	79	wlan_mgt_fixed_capabilities_preamble
40	radiotap_present_antenna	80	wlan_mgt_fixed_capabilities_privacy

ID	Attribute
81	wlan_mgt_fixed_capabilities_radio_measurement
82	wlan_mgt_fixed_capabilities_short_slot_time
83	wlan_mgt_fixed_capabilities_spec_man
84	wlan_mgt_fixed_category_code
85	wlan_mgt_fixed_fragment
86	wlan_mgt_fixed_sequence
87	wlan_mgt_rsn_akms_count
88	wlan_mgt_rsn_akms_type
89	wlan_mgt_rsn_capabilities_mfpc
90	wlan_mgt_rsn_capabilities_mfpr
91	wlan_mgt_rsn_capabilities_no_pairwise
92	wlan_mgt_rsn_capabilities_peerkey
93	wlan_mgt_rsn_capabilities_preauth
94	wlan_mgt_rsn_gcs_type
95	wlan_mgt_rsn_pcs_count
96	wlan_mgt_rsn_version
97	wlan_mgt_tcprep_link_mrg
98	wlan_mgt_tcprep_trsmt_pow
99	wlan_mgt_tim_dtim_count
100	wlan_mgt_tim_dtim_period
101	wlan_qos_amsdupresent
102	wlan_qos_bit4
103	wlan_qos_buf_state_indicated
104	wlan_qos_buf_state_indicated2
105	wlan_qos_eosp
106	wlan_qos_priority
107	wlan_qos_tid
108	wlan_qos_txop_dur_req
109	wlan_seq
110	wlan_wep_key
111	class

Curriculum Vitae

Name:	Udaya Sampath Karunathilaka Perera Miriya Thanthrige
Post-Secondary Education and Degrees:	University of Moratuwa Sri Lanka. Bachelor of Science in Engineering (1^{st} Class Honours) 2004-2008 University of Western Ontario London, ON, Canada. Masters in Engineering Sciences 2014 - 2016
Related Work Experience:	Teaching Assistant University of Western Ontario 2014 - 2016

Publications:

Machine Learning Techniques for Intrusion Detection on Public Dataset, *In Proceedings of the 29th Annual IEEE Canadian Conference on Electrical and Computer Engineering, 15−18 May, 2016 / Vancouver, Canada.*