Flow-based Intrusion Detection for Next-Generation Networks Using Unsupervised Learning

Ph.D Thesis

By

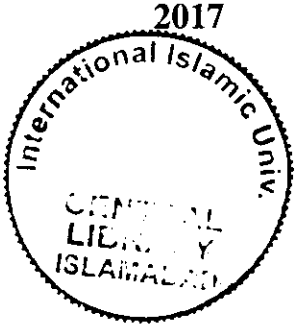
Muhammad Fahad Umer
63-FBAS/PHDCS/F10

Supervised By

Prof. Dr. Muahmmad Sher

Department of Computer Science & Software Engineering
Faculty of Basic & Applied Sciences
International Islamic University, Islamabad, Pakistan

2017



Computer networks

"

Wireless LANs

A dissertation submitted to the
Department of Computer Science & Software Engineering,
International Islamic University, Islamabad
as a partial fulfillment of the requirements
for the award of the degree of
Doctor of Philosophy in Computer Science.

Author's Declaration

I, **Muhammad Fahad Umer**, hereby state that my PhD thesis titled "Flow-based Intrusion Detection for Next-Generation Networks Using Unsupervised Learning" is my own work and has not been submitted previously by me for taking any degree from this International Islamic University, Islamabad or anywhere else in the country/world.

At any time if my statement is found to be incorrect even after my Graduate the university has the right to withdraw my PhD degree.

Name of Student: Muhammad Fahad Umer

Date:

Plagiarism Undertaking

I solemnly declare that research work presented in the thesis titled "Flow-based Intrusion Detection for Next-Generation Networks Using Unsupervised Learning" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and International Islamic University, Islamabad towards plagiarism. Therefore I as an Author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of PhD degree, the University reserves the rights to withdraw/revoke my PhD degree and that HEC and the University has the right to publish my name on the HEC/University Website on which names of students are placed who submitted plagiarized thesis.

Student /Author Signature: 

Name: Muhammad Fahad Umer

Reg # 63-FBAS/PHDCS/F10

INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD
FACULTY OF BASIC & APPLIED SCIENCES
DEPARTMENT OF COMPUTER SCIENCE & SOFTWARE ENGINEERING

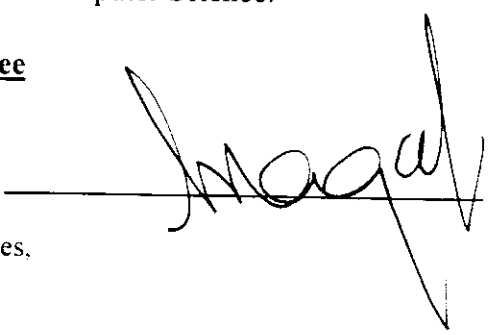
Date: _____

Final Approval

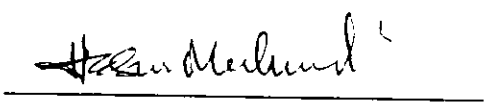
It is certified that we have read this thesis, entitled "**Flow-based Intrusion Detection for Next-Generation Networks using Unsupervised Learning**" submitted by **Mr. Muhammad Fahad Umer, Registration No.63-FBAS/PHDCS/F10**. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University Islamabad for the award of the degree of Doctor of Philosophy in Computer Science.

Committee

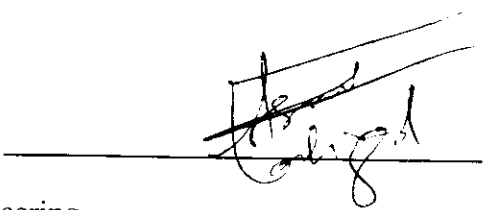
External Examiner:
Dr. Muhammad Abdul Qadir,
Associate Professor,
Professor/Dean Faculty of Engineering & Applied Sciences,
Mohammad Ali Jinnah University, Islamabad
Expressway, Kahuta Road, Zone-V,
Islamabad



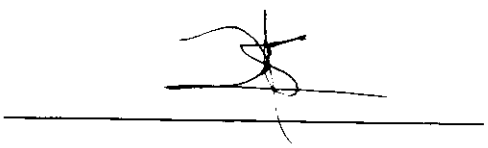
External Examiner:
Dr. Hassan Mahmood,
Associate Professor,
Department of Electronics
Quaid-i-Azam University, Islamabad.



Internal Examiner
Dr. Shahzad Ashraf Ch.,
Assistant Professor,
Department of Computer Science & Software Engineering
FBAS, IIUI



Supervisor
Dr. Muhammad Sher
Professor,
Department of Computer Science & Software Engineering
FBAS, IIUI



Chairman:
Dr. Syed Husnain Abbas Naqvi,
Chairman,
Department of Computer Science & Software Engineering
FBAS, IIUI

Dedication

Dedicated to My family.

Muhammad Fahad Umer

Acknowledgments

I am very grateful to *ALLAH* the *ALMIGHTY* for without His grace and blessing this study would not have been possible.

First of all, I would like to express my sincere gratitude to my supervisor *Prof. Dr. Muhammad Sher* for his continuous support throughout my degree. His guidance gave me the confidence to keep on moving in the extremely difficult period of my research.

I would like to extend immeasurable appreciation to *Dr. Yaxin Bi*, Ulster University, UK, who gave very useful suggestions in the research work.

I am also thankful to my friends and colleagues at the University, notably Syed Bilal Shah, Dr. Anwar Ghani, Dr. Imran Khan, and Dr. Attaullah, for keep on guiding and helping in a variety of aspects.

I am very grateful to my parents who keep on supporting me with their prayers. The daily encouragement from my father did not let me lose my concentration and interest in the degree.

In the end, I am extremely grateful to my wife and children who sacrificed the time that I was supposed to spend with them. They always remain inspiring. My warmest thanks to my wife for her everlasting love and sincere support throughout my studies and research.

My doctorate would not have been possible without the help of all these wonderful people. I pay my respect to all of them. Thank You.

List of Publications

- Muhammad Fahad Umer, Muhammad Sher & Yaxin Bi, Flow-based Intrusion Detection, Techniques and Challenges, Computers & Security, Volume 70, 2017 **Impact Factor : 2.849**
- Muhammad Fahad Umer and Muhammad Sher, A Two-stage Flow-based Intrusion Detection Model For Next-generation Networks, PLoS One (In Press), 2017, **Impact Factor: 2.806**
- Muhammad Fahad Umer, Muhammad Sher and Imran Khan, Towards Multi-Stage Intrusion Detection using IP Flow Records International Journal of Advanced Computer Science and Applications(IJACSA), 7(10), 2016. *(Indexed in Thomson Reuters)*
- Muhammad Fahad Umer, Muhammad Sher & Yaxin Bi, Applying One-Class Classification Techniques to IP Flow Records for Intrusion Detection. Baltic Journal of Modern Computing. 2017;5(1):70. (Indexed in Thomson Reuters)
- Muhammad Fahad Umer and Muhammad Sher., 2017. Automatic Clustering of Malicious IP Flow Records Using Unsupervised Learning. InEnterprise Security(pp. 97-119). Springer, Cham.

Abstract

Next generation networks provide voice, video and data services through a converged IP platform. Traditional intrusion detection systems using deep-packet or stateful protocol inspection are difficult to implement in next-generation networks due to slow throughput, low accuracy and inability to inspect encrypted payload. An alternate solution for securing next-generation networks from network attacks is to use flow-based intrusion detection. This thesis proposes a two stage flow-based intrusion detection architecture for next generation networks using unsupervised machine learning techniques.

Network flows provide a summarized view of network traffic. The use of flow records for intrusion detection involves less processing and does not require the knowledge of upper layer protocols. Due to these advantages, flow-based intrusion detection systems are a suitable choice for high-speed next-generation networks. The goal of this thesis is to develop a flow-based intrusion model for next-generation networks using unsupervised machine learning techniques. The thesis introduces a two-stage intrusion detection model for next-generation networks using flow records. In the two-stage intrusion detection model, the first stage detection process classifies the flows as normal or malicious. The normal flows are discarded, and malicious flows are forwarded to second stage detection process. The second stage detection process groups the similar malicious flows together in different attack clusters.

The two-stage intrusion detection model is implemented using unsupervised machine learning techniques. We propose one class classification for detection of malicious flows in the first stage. We evaluate different one-class classification techniques on a flow-based dataset and determine that SVM-based techniques outperform other techniques in detection of malicious flows. For the second stage, we suggest the use of a clustering technique to group similar malicious flow together. We evaluate k-NN, Self-Organizing Maps, and DBSCAN for clustering of malicious flow in different attack clusters. We select Self-Organizing Maps for implementation of second stage intrusion detection process due to better accuracy and flexibility of setting the number of attack clusters.

In next step, the thesis evaluates an integrated two-stage flow-based intrusion detection system using one-class SVM and Self-Organizing Maps at first and second stage respectively. Our results show that the two-stage flow-based intrusion detection system can detect a significant number of malicious flows and correctly places the majority of malicious flows in corresponding attack cluster.

Contents

List of Figures

List of Tables

Acronyms

APT Advanced Persistent Threat.

AUC Area under ROC Curve.

DTs Decision Trees.

IDS Intrusion Detection System.

IPFIX Internet Packet Flow Information Exchange.

NGN Next-generation Networks.

OC-SVM One-class Support Vector machine.

RDP Remote Desktop Protocol.

ROC Received Operator Characteristic.

SCADA Supervisory Control and Data Acquisition.

SOM Self-Organizing Map.

SSH Secure Shell.

SVDD Support Vector Data Descriptor.

SVM Support Vector Machine.

TPR True Positive Rate.

Chapter 1

Introduction

1.1 Overview

The advent of the internet has converted the world into a global public network. The need of global communication services for businesses and government cannot be over emphasized. Continuous efforts are made by technology companies and service providers to increase the capacity of network links and hardware. However, the traditional voice and data networks are under pressure due to increased demand for IP services such as Voice over IP (VoIP), IP Television (TV), Video on Demand (VOD) and other multimedia services.

The focus of network research community has shifted from enhancement of traditional network technologies to a modern architecture of communication termed as Next-generation Networks (*NGN*). *NGN* provide voice, video and data services through a converged IP platform. These services are available to user through single endpoint device regardless of any access network.

NGN separate service-related functions from the transport and access network layers. Figure 1.1 shows the architecture of next-generation network. The access layer connects subscribers of PSTN, ISDN, GSM and other access networks with transport layer. The transport layer performs the core IP operation and provides connectivity of all components to control layer. The control layer uses soft switching and performs call control, media gateway access and authentication functions. The *NGN* application layer provides session and non-session based services for end users.

For providers, *NGN* reduce the costs and provide avenue of introducing new services without deploying a new network infrastructure. For users, *NGN* provide IP-based Quality of Service and

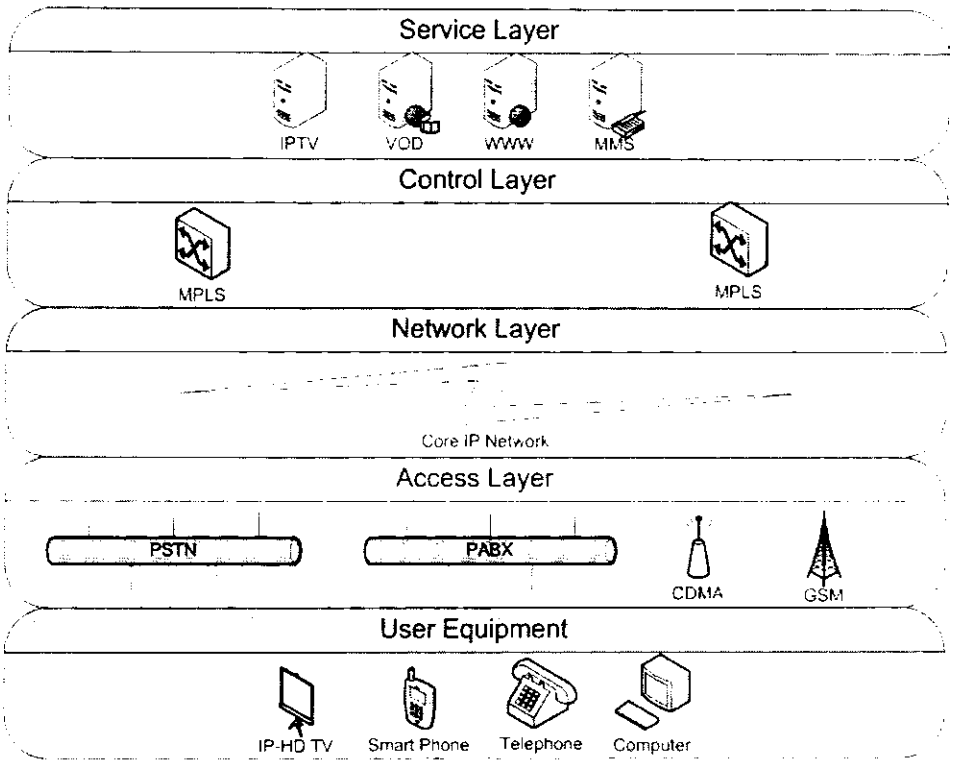


Figure 1.1: Next-generation network architecture

the ability to access all services from a single endpoint equipment.

1.2 Security in Next-Generation Networks

Today, the dependency on communication services has increased to a level that cyber security has become a dimension of national security. The global connectivity has resulted in effective access and economy growth, but at the same time also posed a greater security challenge. Recently, a large number of cyber security attacks over the government and corporate intranets and successes in intruding the restricted domains have been observed. The Global Information Security State Survey accumulates the total number of reported security incidents at 42.8 million in 2014. This number of detected security incidents is 48% greater as compared with 2013 [1]. The rate of detected security incidents has increased 66% since 2009. Although, the actual number of security incidents will be much higher because the survey only includes the number of attacks that were detected and reported. A number of security attacks may not be detected by the organizations or

would not have been reported due to commercial reasons.

The tremendous increase in data transfer rate, computation power, and expansion of computer networks has resulted in complex information security challenges which inspires research in securing networks. NGN encapsulate a variety of network architectures, services, and protocols in a layered architecture. Today, customers require end-to-end security against network attacks. This has to be accomplished without installing dedicated customer premises equipment (CPE). The current network security model is based on pervasive trust, where no packet, device or service can be trusted unless it is inspected. Therefore NGN require an integrated security model providing complete protection for the enterprise network. The integrated NGN security model should provide end-to-end protection without affecting overall network performance. The model should also be able to integrate with existing devices and services and can identify, classify, and trace back anomalous behavior across the network [2].

In the previous decade, having a firewall available for a network that is going to be connected to Internet was considered enough. The firewall was a passive way to look for any violation with predefined set of rules. But since long, firewalls are not able to detect intelligent attacks. To counter intelligent and more sophisticated attacks, Intrusion detection system came in to complement the firewall security and become an integral part of enterprise networks.

1.3 Intrusion Detection Systems

Intrusion Detection System (*IDS*) analyzes the log trails associated with the protected systems and decide whether these log trails contain traces of an attack or not. If the intrusion detection system detects an attack, it raises an alert. A human administrator or an automatic alert response system takes a corrective action based on the intrusion alert [3].

Figure 1.2 shows a generic intrusion detection model described in [4]. The model uses four types of function contained in event-boxes, database-boxes, analysis boxes and response boxes. The event-boxes can be sensors or agents, which collect the information about the network traffic activity. This information is stored in a database-box for subsequent processing by analysis-box. The analysis-box analyzes the gathered information to detect an attack. The decision of analysis-box is forwarded to response-box. The response-box alerts the human administrator or can also invoke an automatic correction or prevention system to secure the network.

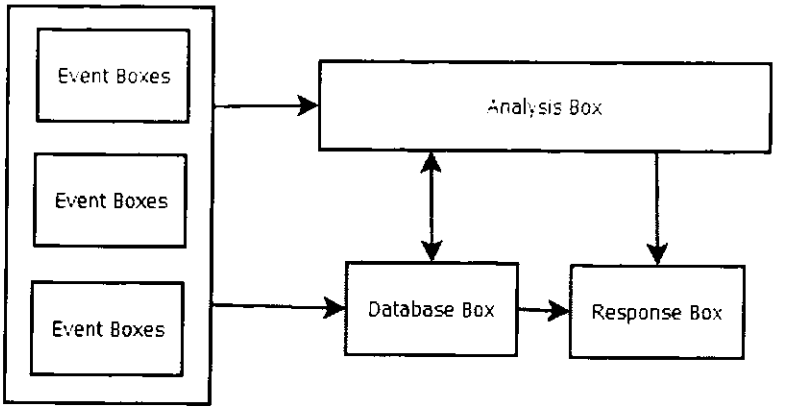


Figure 1.2: A Generic Intrusion Detection Model

There are a number of dimensions on which the intrusion detection systems can be classified. On the basis of detection methodology, the *IDS* are classified into misuse detection, anomaly detection and stateful protocol analysis [3]. Misuse detection techniques use an established signature database of malicious activities of known attacks and malware. The *IDS* compares a recorded system activity with the signatures database. If a match is found, an intrusion alert is raised. Misuse detection techniques are relatively accurate and distinctly detect an attack with a low false positive rate. Misuse detection systems already have detail information about all known attacks and no manual effort is required to determine the type and scale of attack. Most of the commercial intrusion detection products use misuse detection model [5].

Anomaly based intrusion detection systems try to define a model for normal network traffic behavior. Any deviation from the defined model is considered an intrusion [4]. This technique can give good detection results for novel attacks. The stateful protocol analysis (SPA) inspect the network traffic for conformance with protocol specification defined by the international standard organizations, e.g., IETF. The network is considered malicious An out of range value from the protocol specification parameters is considered intrusion.

Another dimension for classification of Intrusion Detection System (*IDS*) is the type of data processed by the IDS for intrusion detection. The input data to the IDS consist of application or/and system logs, network packets or packets headers. The IDS analyzing system or application logs are refereed as host-based IDS. Host-based IDS protects individual system or device and are not implemented at network level. The network-based IDS provides protection for the complete network. Traditional network-based intrusion detection systems use deep packet or stateful protocol inspection to detect an intrusion in the network traffic. Deep packet inspection (DPI) techniques

scan the packet beyond the protocol header and inspect its content. DPI techniques provide complete visibility of network traffic and filter the packet content for malware, virus or any other attack traces [6].

1.4 Limitation of IDS in Next-generation Networks

Information security was not a prime objective during the development of internet infrastructure [7]. Much of the security has been added as addons to the network protocols and technologies [8]. However, enterprises and service providers have now increased spending on the information security.

Intrusion detection system is an important tool in defense of computer networks. Intrusion detection systems have been a subject of tremendous research for increased defense against cyber attacks. The research has been done in all dimensions of intrusion detection systems. However, various approaches used to design intrusion detection have one or more limitations. The IDS using the misuse detection approach are unable to detect zero-day attacks for which no signatures are yet available. Maintaining the updated signatures database is also difficult. Whenever a new attack is discovered, all installations of the misuse based intrusion detection system need an immediate signature update. A number of sophisticated malware today use polymorphism which creates multiple variants of same malware. These variants easily bypass the signature comparison check in the misuse based intrusion detection system. To overcome these issues, intrusion detection systems use anomaly detection techniques for detection of novel and unseen attacks.

Anomaly-based intrusion detection systems have the advantage of detecting zero-day attacks over misuse based IDS. However, anomaly-based IDS require a profile of normal network traffic. Development of a network traffic profile that can represent all scenarios of normal network traffic is not a trivial task. Any unseen change in the network traffic, perfectly legitimate, can alert the IDS to raise an intrusion alert. Therefore, anomaly-based IDS have a high false-positive alarm rate.

The network-based intrusion detection system (NIDS) use deep packet inspection. Deep-packet inspection is not possible when packet content is encrypted. In stateful protocol inspection, the complete semantics of the protocol are checked against the specification and any out of range value is considered an anomaly. State-full protocol inspection techniques are protocol specific and cannot be generalized for unknown protocols. Also, both packet and stateful protocol inspection techniques are computationally costly and become a performance bottleneck [2, 3].

Another major challenges faced by the IDS is slow analysis and undesirable effect on throughput of network. The network traffic in medium sized network is transferred at a speed of 10Gbps. Network based IDS are unable to analyze the data at this data transfer speed [9].

The research in IDS is also effected due to unavailability of updated intrusion datasets. Most of the research in IDS is still being evaluated on KDD and DARPA datasets which are now 17 years old [10]. In our opinion, such datasets are not suitable for evaluation of IDS for detection of modern day attacks.

Privacy issues also limit the functionality of intrusion detection systems. The deep-packet inspection is considered a privacy concern because it also scan the packet payload which may contain sensitive corporate or personal information.

Some intrusion detection system are developed focusing on particular attack type or scenario. Such techniques are good in detecting the target attack but gives moderate performance during detection of other types of attack [8].

The analysis shows that there are number of open issues in existing approaches to intrusion detection. Further research is required in to address these open issues for development of an accurate, efficient and reliable in. The accuracy and efficiency of intrusion detection systems become more important in the context of Next-generation Networks (*NGN*). *NGN* architecture is open for all different types of networks and user services. Next-generation networks combine voice, video, and data services in a packet-based all IP network. This convergence of heterogeneous network architectures has serious security implications because *NGN*inherits the vulnerabilities of traditional access networks [11, 12].

1.5 Motivation

Next-generation networks(NGN) use flow records for various network management tasks including quality control, billing and intrusion detection. A promising solution for protection of next-generation networks is to apply flow-based intrusion detection. The flow records provide a unified way to access traffic information across the next-generation network. The flow-based intrusion detection systems use flow records as input and try to find out if network traffic is normal or malicious [13]. These flow records are collected from the network using specialized flow-enabled network devices and contain aggregated information of related network packets. The process of

flow collection is not dependent on high-level network architecture and user services. The flow records are on the average equal to 0.2% of the traffic [14] therefore the amount of data analyzed by the intrusion detection system is reduced. A comparison of flow-based and other intrusion detection techniques shows that only flow-based intrusion detection systems have the ability to work on high-speed links [15, 16].

Given the requirements by high-speed next-generation networks, we are certain that a two-stage flow-based intrusion detection will provide the best available defense against the network intrusions without effecting the network throughput. The two-stage architecture enables the intrusion detection system to implement different level of detection phases in increasing order of computation costs. We will focus on unsupervised machine learning techniques for implementation of our model. Unsupervised learning technique are because no labeled training set is require.

1.6 Problem Statement

The study of existing intrusion detection systems reveals that traditional approaches to intrusion detection has several limitation. The purpose of this research is to develop a flow-based intrusion detection model for next-generation networks against the network attacks. Our problem statement for this thesis is summarized as under:-

"How efficient flow-based intrusion detection can be implemented in next-generation networks using unsupervised machine learning techniques?"

1.7 Research Objectives

In this thesis, we try to solve the problem of attack detection in next-generation network using flow records. We will use unsupervised machine learning algorithms to detect malicious flows records. We have set following research objectives for our research:-

- Why traditional IDS are not suitable for protection of next-generation networks against network attacks? What is flow-based intrusion detection and why it is suitable for next-generation networks? What are the pros and cons of flow-based intrusion detection?
- What is the current state of art in flow-based intrusion detection? What are the advantages

and disadvantages of different approaches used for designing of flow-based IDS? What are the current research challenges for flow-based intrusion detection?

- What is an efficient technique of intrusion detection in next-generation networks? How can we use a two-stage intrusion detection model for flow-based detection in next-generation networks?
- How can we implement the two-stage flow-based intrusion detection model using machine learning techniques?
- Which machine learning technique is best suited for first stage detection process of the proposed two-stage flow-based intrusion detection model?
- Which machine learning technique is best suited for second stage detection process of the proposed two-stage flow-based intrusion detection model?
- How we can combine the best performing machine learning techniques in a two-stage flow-based intrusion detection system using unsupervised learning
- How the proposed two-stage flow-based intrusion model using unsupervised learning performs on the publicly available flow-based datasets?

1.8 Major contributions of this thesis

We propose a novel two-stage flow-based intrusion detection system for next-generation networks. The first stage uses an enhanced unsupervised one-class support vector machine which separate malicious flows from normal network traffic. The second stage uses a self-organizing map which automatically groups malicious flows into different alert clusters and give an attack label to every malicious flow. The techniques in both stages use unsupervised learning, therefore, no labeled training dataset is required. Experiment on a flow-based dataset generates promising results regarding accuracy and lower false alarm rate. Major contributions of the thesis are given below:-

- **State of the art review of flow-based intrusion detection system and existing challenges** An extensive survey available flow based intrusion detection techniques has been performed,
- **A two-stage flow-based intrusion detection model** We will propose a flow-based detection model for intrusion detection in next-generation networks. We will use a two-stage to improve the efficient and to decrease false positive alarm rate

- **Implementation of the proposed model using unsupervised machine learning techniques** Use of unsupervised learning as compared to supervised learning in intrusion detection is advantageous. We will determine that how unsupervised learning techniques can be used for anomaly detection in network flows.
- **Rigorous evaluation and validation on flow-based datasets.** We will use public flow-based datasets for evaluation of available technique and validation of our proposed technique. We will use multiple evaluation measure to compare the results.

1.9 Organization of the thesis

The thesis is organized in a logical sequence and every chapter moves a step closer towards conclusion of the research.

The first chapter gives an overview of the research discipline and also describe the motivation, problem domain, research objectives. Remainder of the thesis is organized as follows:-

- **Chapter 2** discusses the preliminary concepts of flow-based network analysis and flow-based intrusion detection. It also describes the IPFIX process to collect and export flow records from network routers. The general architecture describes the standard working of intrusion detection system.
- **Chapter 3** reviews the available flow-based intrusion detection system for next-generation environment. We have review the available flow-based intrusion system in context of next-generation network and discuss their strength and weaknesses.
- In **Chapter 4**, we propose a two-stage model for flow-based intrusion detection. We describe the architecture and flow diagram of our model. We also discuss implementation of our proposed model using machine learning techniques.
- In **Chapter 5**, we apply a one-class classification component for detection of malicious flows. We review various one-class classification techniques and evaluate them on a flow-based dataset to determine their performance for detection of malicious flows.
- In **Chapter 6**, we use unsupervised learning techniques for clustering of malicious flows in attack clusters. Every attack cluster contain malicious flows which are similar to each other. This approach helps in raising consolidated alerts for cluster of similar malicious flows.

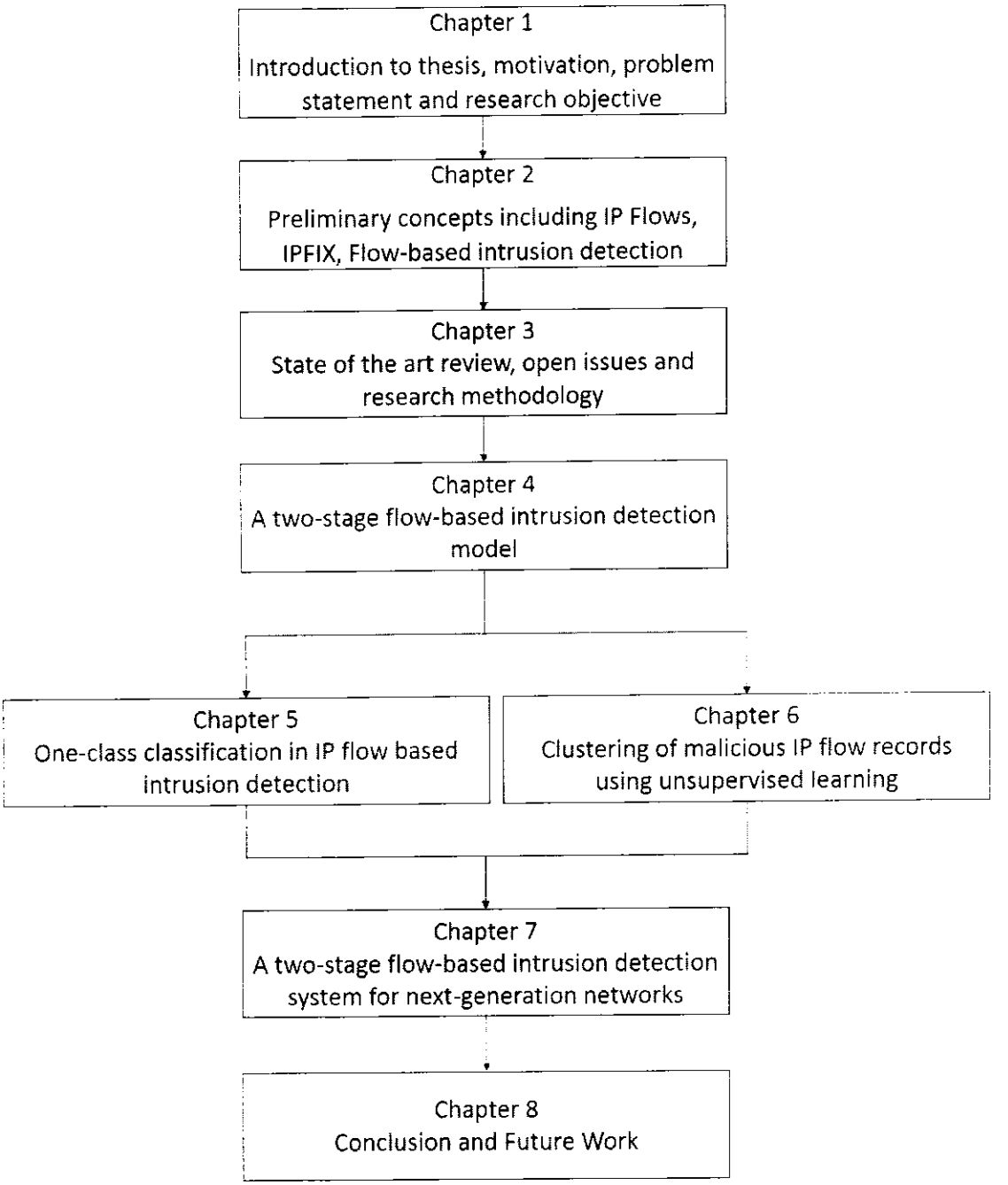


Figure 1.3: Thesis layout

- In **Chapter 7** we propose two-stage flow-based intrusion detection in next-generation networks using unsupervised learning. Evaluation on Flow-based data-sets. We have used a multiple flow-based dataset to evaluate our technique. The parameters of both detecting stages are optimized to obtain the best results. A two-stage model for flow-based intrusion detection in next-generation networks using unsupervised learning. Evaluation on Flow-based data-sets. We have used a two different flow-based dataset to evaluate our technique. The parameters of both detecting stages are optimized to obtain the best results.
- **Conclusion** Finally in chapter 8, we conclude our research and analyze the contributions of our work. We review the research objectives one by one and describe our achievements. We also present the impact of research. In the end, we identify directions for further research in the thesis topic.

Chapter 2

Preliminaries

2.1 Overview

Chapter 1 gave an introduction to the research topic and discussed the motivation for use of flow-based intrusion detection in next-generation networks. This chapter gives an overview of flow records and IPFIX flow export and collection processes. We also present the general architecture of a flow-based intrusion detection system.

2.2 Network traffic flows

In packet switching network, traffic flow is defined as a combination of packets having similar characteristics in a given time. Flow traffic is more general than packet traffic and provide a coarse-grain analysis of the network. In IP networks, an IP flow is described as "a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties" [17]. The key fields of flow record are Source IP , Destination IP, Source Port, Destination Port, Start time, End time, IP Protocol number and TCP Flags. There are several application where flow traffic is used e.g. congestion control, billing and intrusion detection.

2.3 Internet Packet Flow Information Exchange(IPFIX)

The flow data is collected from the network using flow enabled devices and exported in specific records using a flow export protocol. Since there are numerous application of flow data, all major vendors are now offering built-in flow collection and export support in their network hardware. The collection and processing of flow data in an IP network is controlled by a flow collection and export protocol. Different flow collection and export protocols exist but Cisco Netflow has been the most popular. Other vendors also supported the protocol under different names like Netstream for HP and Huawei and JFlow for Juniper.

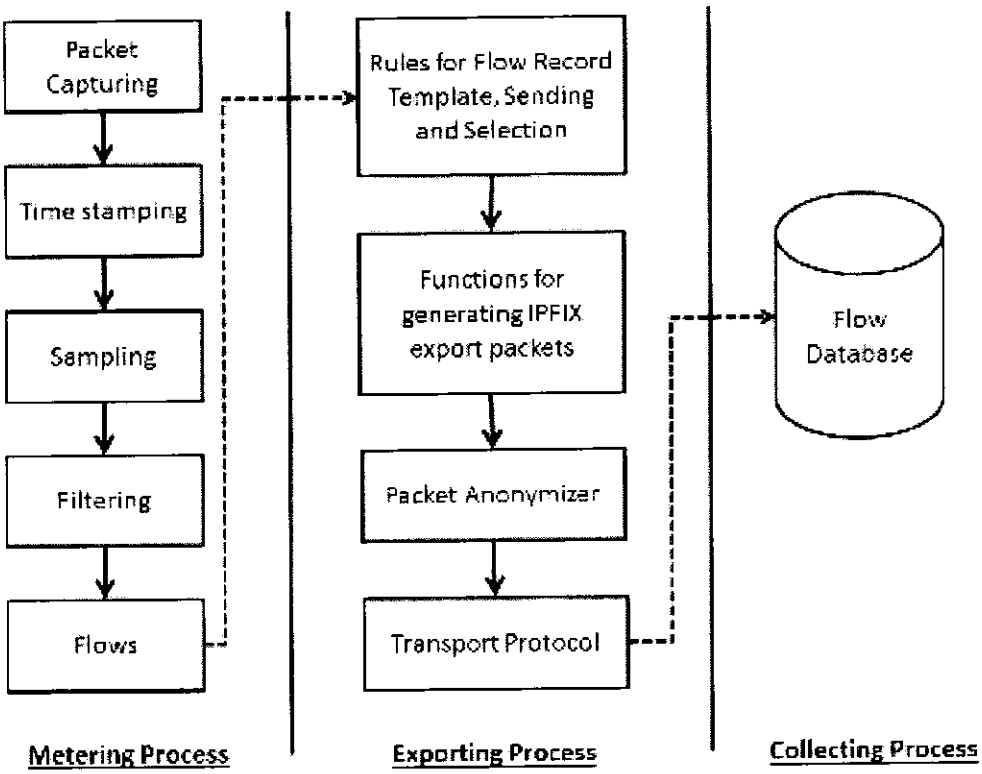


Figure 2.1: IPFIX Metering, Exporting and Collection Processes

Due to the popularity of Netflow, Netflow v9 has been adopted by Internet Engineering Task Force (IETF) for development of a standard flow protocol called IPFIX (RFC 3179). IPFIX is very flexible protocol with around 280 attributes. IPFIX allows export of flow records in any format defined by the export template. Unlike Netflow, IPFIX contain specific fields which can be used by vendors to store proprietary information. IPFIX architecture is explained in detail in RFC 5317.

Typical deployment of IPFIX, shown in Figure 2.1, consists of following three processes:-

- Observation point with metering process. Observation points collect the packets being passed through a specific interface. These packets are forwarded to a metering process where packets are time stamped. These time-stamped packets can be optionally sampled or filtered. These packets are then cached for specific interval such that all packets required for a specific flow are received. These packets are converted into flow records and forwarded to exporting process.
- Exporting Process. The rules for generating IPFIX flow records are defined in exporting process. The process generates the IPFIX records in the format defined by IPFIX template.
- Collecting Process. The collecting process collects IPFIX records from the exporting process and stores it in a flow database. The database is used by the user application for required purpose.

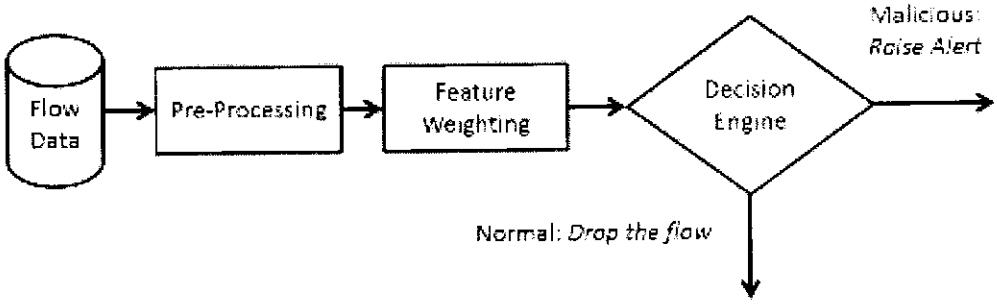


Figure 2.2: A simple flow-based intrusion detection model

2.4 Flow-based intrusion detection system

Flow-based intrusion detection systems use flow records as input and try to find out if the flow of traffic is normal or malicious [13]. Flow-based intrusion detection systems are also based on the generic intrusion detection model presented in [4]. The incoming flows are collected from the network using observation points (event-boxes). These flows can be optionally stored in a flow database (database-boxes). Then flows are forwarded to an analysis box for evaluation. The analysis box uses anomaly detection techniques for attack detection. If any attack is detected, a response is initiated through the response-boxes.

The flow records are able to characterize the communication behavior of different network attacks. This communication behavior can be analyzed by the detection engine to determine that if the the flow of traffic is normal or malicious. Since only the flow records are inspected, the intrusion detection system is relieved from the complex and time-consuming processing of packet content inspection. The flow packets analyzed on a network are one the average equal to 0.1% of the traffic and the overhead due to Netflow is on average 0.2% [13]. Therefore, flow-based intrusion detection has less data to process as compared to packet-based intrusion detection systems. However, flow-based intrusion detection is not matured enough. Flow-based techniques cannot detect attacks hidden in the packet payload. At best, flow-based intrusion detection system have near real-time response, low deployment cost and the ability to work in high-speed next-generation network environment [15].

Figure 2.2 shows the architecture of flow-based intrusion detection system. The system takes IPFIX/Netflow records as input. The flows records can have many attributes. All of these attributes may not be required in the classification decision and can become computational overhead. Also some important attributes like originating IP address, destination port need to play an important role in the detection decision. The feature selection phase only selects relevant attributes required for decision making. A pre-processing phase converts the flow records in a specific format which is acceptable to the anomaly detection algorithm. The anomaly detection algorithm uses the flows records for training and detection phases. In the detection phase, the algorithm marks the flow records as malicious or normal. If the flow is normal, it is considered safe and dropped with no subsequent action. Malicious flow can raise an alert and become the subject of further inspection.

2.5 Pros and Cons of Flow-based intrusion detection

Flow-based intrusion detection has a number of advantages over traditional intrusion detection systems. The flow-based IDS only analyze flow records. The flow records contain aggregated information of packet headers. The network traffic information is summarized in the form of flows and the amount of data processed by the IDS is reduced. Flow-based intrusion detection is, therefore, best suited for protection of backbone links where processing of complete network traffic is computationally difficult [13].

Many applications use end-to-end encryption. It is not possible to inspect the encrypted data at intermediate location by a packet-based intrusion detection. In this case, flow-based intrusion

detection is an appropriate choice because no packet data scanning is required. There are less privacy concerns user information is protected from any intermediate scan.

The collection of flow data from the network can easily be distributed among multiple flow collection points. Most of the latest hardware offers built-in flow-collection support. Thus flow data can be collected from multiple locations across the network with any additional cost [15].

An important feature of flow-based is the collection of flow data in a standard format using Netflow or IPFIX. The flow-based IDS do not need to incorporate any logic for collection of network data from variety of network architectures and protocols. The collection of flow data suing IPFIX also has additional benefits such billing, congestion control, and network behavioral analysis [14].

With some excellent benefits, flow-based intrusion detection also has some drawbacks. The flows used for intrusion detection contain generalized network information. It becomes difficult for the flow-based IDS to distinctly detect an attack using the generalized information. Flow-based techniques do not scan the packet payload. Therefore these techniques cannot detect the network attacks hidden in the packet payload. Overall, the flow-based techniques may not be as accurate as packet-based detection[13]. However, flow-based techniques can work in scenarios where packet-based intrusion detection cannot even work. At best, flow-based intrusion detection systems have a near real-time response, low deployment cost, and the ability to operate on high-speed backbone links[15]. These benefits motivate the use of flow-based intrusion for network attack detection in high-speed networks.

2.6 Chapter Summary

This chapter gives an introduction to flows and flow-based intrusion detection. It describes *IPFIX* protocol and also discuss the general architecture of flow-based intrusion detection system. In the next chapter, we review the state of the art in flow-based intrusion detection and identity the major open issues.

Chapter 3

State of the art in flow-based intrusion detection

3.1 Overview

Chapter 2 gave an introduction to network flows and flow-based intrusion detection system. In this chapter, we review the existing flow-based intrusion detection systems. We propose a taxonomy of flow-based intrusion detection system based on the technique used for. We review the available techniques under each class of methods and discuss their advantages and disadvantage. We also list the commercial application using flow-based intrusion detection. In the end, we identify open issues and research challenges for future research.

3.2 Related Work

Intrusion detection is a prominent research area and a number of survey and review articles exists. The articles include general-purpose, scenario-specific and technique-specific review of intrusion detection systems. The review articles describe the existing techniques and also identify important research challenges for future research. [18] discuss different intrusion detection techniques used for Mobile Ad-hoc Networks (MANETs). The authors reviewed and compared the existing systems and provided directions for future research. Another survey for intrusion detection and prevention system in MANETs is presented in [19].

A review of anomaly-based network intrusion detection systems (A-NIDS) is presented in [4].The article classifies the A-NIDS into statistical-based, knowledge based and machine learning based categorizes. The article reviews the available A-NIDS techniques and discusses their pros and cons. A list of commercially available A-NIDS is also given. In the end, the article also discusses the open issues and challenges posed by the anomaly detection systems. A survey of anomaly detection methods in network is presented in [20]. The survey organized anomaly detection techniques in four categories. The article describe every method and also give examples for its applications in networks

In [21], the authors give a detail survey of flow-based intrusion detection systems. The article provides an introduction of flow-based intrusion system and also describes the motivation behind the use of flow-based intrusion detection. A taxonomy of network attacks is created and flow-based techniques addressing the attack types are described. In the conclusion, the authors critically discuss the flow-based intrusion detection and identify future research directions.

A survey of intrusion detection systems using computational intelligence techniques is given in [5]. The survey reviewed the artificial neural networks, fuzzy systems, evolutionary computation, artificial immune systems, swarm intelligence, and soft computing algorithms.

A survey of intrusion detection and preventions systems is presented in [22]. In the survey, deficiencies in the existing systems are analyzed and use of intelligent techniques such as machine learning and autonomic computing is proposed for detection of known and unknown threats.

A comprehensive review of intrusion detection systems is presented in [3]. The article propose a taxonomy of intrusion detection systems based on the system deployment, data source, timeliness and detection strategy. Some future challenges for intrusion detection systems have also been presented.

In [23], the authors survey intrusion detection and prevention systems (IDPS) for cloud computing. The authors describe the characteristic of cloud computing, and discuss the challenges faced in development of IDPS for cloud. The article also identify the requirements for a cloud-based IDPS

A survey of intrusion detection system in Wireless Sensor Networks (WSNs) is given in [24]. The article includes a brief survey of IDSs and discuss their applicability to WSNs. The authors give a detailed review on IDSs devised for WSNs with their advantages and disadvantages. The survey also provide a general model for an IDS applicable to WSNs.

An extensive review of network anomaly detection methods, systems and tools is presented in

[25]. The article identify six different categorizes of network anomaly detection methods. The authors have described the advantages and disadvantages for every class of methods and discussed the related systems. The article also give detail of evaluation measures and datasets used for benchmarking of intrusion detection systems. In the end, an extensive discussion on open issues and challenges in network anomaly detection is also given.

In [26], a survey of collaborative intrusion detection system (CIDS) is presented. First, The authors define requirements for the successful deployment of CIDSs in large IT systems and critical infrastructures. The available CIDSs are classified into centralized, decentralized, and distributed classes a detail survey of techniques in every class is presented.

A survey of data mining and machine learning methods for intrusion detection is given in [27]. The article categorize the available systems under twelve different methods of data mining and machine learning. The Computational complexity of different methods has also been described. In the end, the article gives recommendations for use of data mining and machine learning techniques in intrusion detection.

The research in flow-based intrusion detection is gaining momentum. A number of intrusion detection techniques have been proposed in recent years that use flow records for attack detection. To the best of our knowledge, no effort exists which survey the current state of the art in flows-based intrusion detection after [21]. Consequently, we review the available flow-based intrusion detection systems and identify important research challenges. We also describe the different flow-based datasets used for performance evaluation of the flow-based system.

3.3 Flow-based intrusion datasets

The intrusion datasets are used for benchmarking the performance of intrusion detection systems (IDS).The datasets contain both normal and malicious network traffic.The IDS detects the malicious traffic present in the dataset. The performance of an IDS is evaluated by the actual number of attacks and the number of attacks detected by the IDS. Publicly available datasets make it easier to obtain symmetrical results for comparing different IDS. The intrusion datasets are generated in following two ways:-

- A laboratory environment is setup to simulate the different network scenarios. The attacks are artificially launched using scripts and traffic sample are collected from the network. This

type of dataset is easy to develop, and all attack types can be manually injected. However, such datasets do not represent the real-world network traffic scenario. The intrusion detection systems evaluated on such datasets are not guaranteed to give similar results in a real-world deployment.

- Another way to create intrusion datasets is to collect traffic samples from real-world networks. These datasets represent the actual nature of network traffic. However, these datasets may not contain all required types of attacks.The realistic datasets are difficult to build. Companies and enterprises do not permit collection of traffic samples from their network due to confidentiality and privacy issues.Also, legal laws do not allow publishing of actual data in public domain. Usually, user related information is removed from the dataset to address the privacy issues.

Although many datasets exist for packet-based intrusion detection, few datasets are available for flow-based intrusion detection. The packet based datasets have complete packet capture files whereas flow-based datasets contain IPFIX/Netflow records. We give brief information about the available flow-based datasets.

3.3.1 Sperotto intrusion dataset

The Sperotto dataset is the first publicly available flow-based dataset. It consist of 14.2M flow records collected through a "Honeypot" deployment in University of Twente network [28]. The honeypot virtual machine ran for 6 days and data collection resulted in a 24 GB dump file. Four standard services SSH, HTTP, FTP and AUTH/IDENT were run over the honeypot for six days. The SSH flows are result of automated brute-force dictionary attacks, where repeated tries are made for guessing user name and password from a list of dictionary words. The HTTP flows in the dataset were generated due to automated attacks carried out using tools like Nikto[1] and Whisker[2]. During the flow collection, one attacker installed an IRC proxy over the honeypot which has also generated some traffic. Both traffic dump and the services log file were downloaded and passed through a correlation process for the alert generation. The correlation process succeeded in labeling more that 98.5% of the flows and 99.99% alerts. Table 3.1 shows the number of flows for every alert in the dataset.

[1]https://cirt.net/Nikto2
[2]http://www.iss.net/security_center/reference/vulntemp/HTTP_WhiskerScan.htm

Table 3.1: Detail of flow records - Sperotto intrusion dataset

Traffic Type	Number of flows	Category
SSH	13942629	Malicious
FTP	13	Malicious
HTTP	9798	Malicious
AUTH-INDET	191339	Normal
IRC	7383	Normal
OTHERS	18970	Normal

Table 3.2: Sample of flow records - Sperotto intrusion dataset

Source IP	Destination IP	Packets	Octets	Source Port	Dest port	TCP flags	Prot
2463760020	3752951033	1	60	4534	22	2	6
2463760020	3752951032	1	60	1923	22	2	6
2463760020	3752951030	1	60	3185	22	2	6
2463760020	3752951031	1	60	2466	22	2	6
2463760020	3752951029	1	60	3056	22	2	6

The dataset is available in the form of Netflow v5 records. Table 3.2 shows a sample of the dataset. The source and destination IPs are converted to numeric form and anonymized for privacy reasons. The packet and octets indicate the total number of packets and octets in a flow. The four attributes, start time, start msec, end time and end msec give the flow start and end time. The TCP flags attribute contains the logical OR of TCP flags field of all packets in the flow. The Protocol field shows that underlying transport protocol.

3.3.2 TU intrusion dataset

Tezpur University Intrusion Dataset (TUIDS) is a packet and flow-based dataset developed by [29]. The dataset is generated in a laboratory environment in Tezpur University. The experimental setup used for generation of dataset consists of one router, one L3 switch, two L2 switches, one server, two workstations and forty nodes. The attacks are generated against different nodes. Another LAN of 350 nodes was also connected to the experimental setup. The attacks are launched from the LAN and also within the setup. Table 3.3 gives the detail of attacks. The dataset contains both packet and flow-based data. The flow-based dataset is in the form of Netflow v5 format. The flow

Table 3.3: Detail of attacks - TUIDS

No.	Attack	No.	Attack
1	Bonk	9	1234
2	Jolt	10	Saihyousen
3	land	11	Oshare
4	Netsea	12	Window
5	Newtear	13	Syn
6	Syndrop	14	Xmas
7	Teardrop	15	Fraggle
8	Winnuke	16	Smurf

records has 16 basic attributes four time-windows attributes and four connection based attributes. The detail of normal and malicious records in the flow-based dataset is given in Table 3.4.

Table 3.4: Detail of flow records - TUIDS

Category	Training dataset	Testing dataset
Normal flows	23120	16770
Attack flows	29723	23955
Total	52843	40725

3.3.3 CTU-13 dataset

The CTU-13 dataset was created in CTU University, Czech Republic [30]. The dataset consists of botnet traffic mixed with normal and background communication traffic. The traffic capture process consists of 13 different scenarios where a particular malware traffic was captured in each scenario. The environment for traffic capture consists of virtual machines running the Microsoft Windows XP SP2 operating system on a Linux Debian host. These virtual machines were bridged into the University network. The traffic was captured both on the Linux host and on the university network router connected to the Linux host. During the labeling process, all traffic was initially given the background label. The normal label was given to the traffic that was originated from switches, proxies, and legitimate computers. All traffic that came from the known infected machines was labeled botnet. Table 3.5 gives detail of the malware and traffic flow records in each scenario.

Table 3.5: Detail of flow records - CTU-13 intrusion dataset

Id.	Bot	Characteristic	Total Flows	Botnet Flows	Normal Flows	Background Flows
1	Neris	IRC, SPAM,Click Fraud	2824636	39933	30387	2753290
2	Neris	IRC, SPAM, Click Fraud, FTP	1808122	18839	9120	1778061
3	RBot	IRC, Port Scan, US	4710638	26759	116887	4566929
4	RBot	IRC, DDOS, US	1121076	1719	25268	1094040
5	Virut	SPAM, Port Scan, HTTP	129832	695	4679	124252
6	Mentri	Port Scan	585919	4431	7494	546795
7	Sogou	HTTP	144077	37	1677	112337
8	Merli	Port Scan	2954230	5052	72822	2875282
9	Neris	IRC, SPAM, Port Scan, Click Fraud	2753884	17880	43340	2525565
10	Rbot	IRC, DDOS, US	1309791	106315	15847	1187592
11	RBot	IRC, DDOS, US	107251	8161	2718	96369
12	NSIS.ay	PP	325471	2143	7628	315675
13	Virut	SPAM, PS, HTTP	1925149	38791	31939	1853217

Table 3.6: Important flow attributes - CTU-13 intrusion dataset

Duration	Prot.	Src. Addr	Src Port	Dst. Addr	Dst Port	Total Packets	Total Bytes	Src. Bytes
3550.1823	udp	212.50.71.179	39678	147.32.84.229	13363	12	875	413
0.0008	udp	84.13.246.132	28431	147.32.84.229	13363	2	135	75
0.0003	tcp	217.163.21.35	80	147.32.86.194	2063	2	120	60
0.0569	tcp	83.3.77.74	32882	147.32.85.5	21857	3	180	120
3427.7680	udp	74.89.223.204	21278	147.32.84.229	13363	42	2856	1596
3086.5473	tcp	66.169.184.207	49372	147.32.84.229	13363	591	45931	26480

The CTU dataset contains bidirectional Netflow records. Every flow record has 15 attributes. Table 3.6 shows a sample of the dataset with important attributes. The start time and duration fields are used to calculate the flow duration. The protocol value shows the transport layer protocol type. The direction field shows the direction of flow. It can be incoming, outgoing or bi-directional. The Total packets and total bytes fields contain the total number of packets and bytes transmitted in either direction. Another field source bytes also exists which can be used to extract the bytes received from the destination. The label field shows the type of flow.

3.3.4 SSHCure intrusion dataset

SSHCure is a SSH compromise detection dataset collected by [31]. It consists of all incoming and outgoing SSH traffic on a campus network. The dataset was exported in a Netflow v5 format from the four Cisco 6500 series routers. The dataset has two segments, D1 and D2. Both segments were collected over a period of one month on the campus network of the UT. The two segments reflect two different scenarios. The D1 segment consists of SSH traffic targeting honeypots. The D2 segment has the SSH data from normal servers. The D1 and D2 segments have 632 and 10716 attacks respectively. The ground truth for the dataset is obtained from the corresponding log files of servers and honeypots.

3.4 Techniques in flow-based intrusion detection

We have created a taxonomy of flow-based intrusion detection system on the basis of technique used for attack detection in flow records. Figure 3.1 shows the taxonomy hierarchy. It classifies the flow-based intrusion systems into *statistical*, *data mining*, *machine learning*, *hybrid* and *other* techniques. In the following section, we review the architecture, dataset and performance results of available flow-based intrusion detection systems in each category.

3.4.1 Statistical techniques

Statistical methods build a profile of the normal network traffic using a statistical function of network traffic parameters. This profile of the normal traffic is used to check the unseen incoming traffic. If the similarity of the traffic, calculated using statistical measures, is above the pre-defined threshold, the flow is marked normal or malicious otherwise [3, 32]. The statistical-based techniques have a strong mathematical foundation and work well if there is less diversity in network flows. High dimensionality and variation in network traffic can affect the performance of statistical intrusion detection systems [33]. We further categorize the statistical techniques into univariate and multivariate methods.

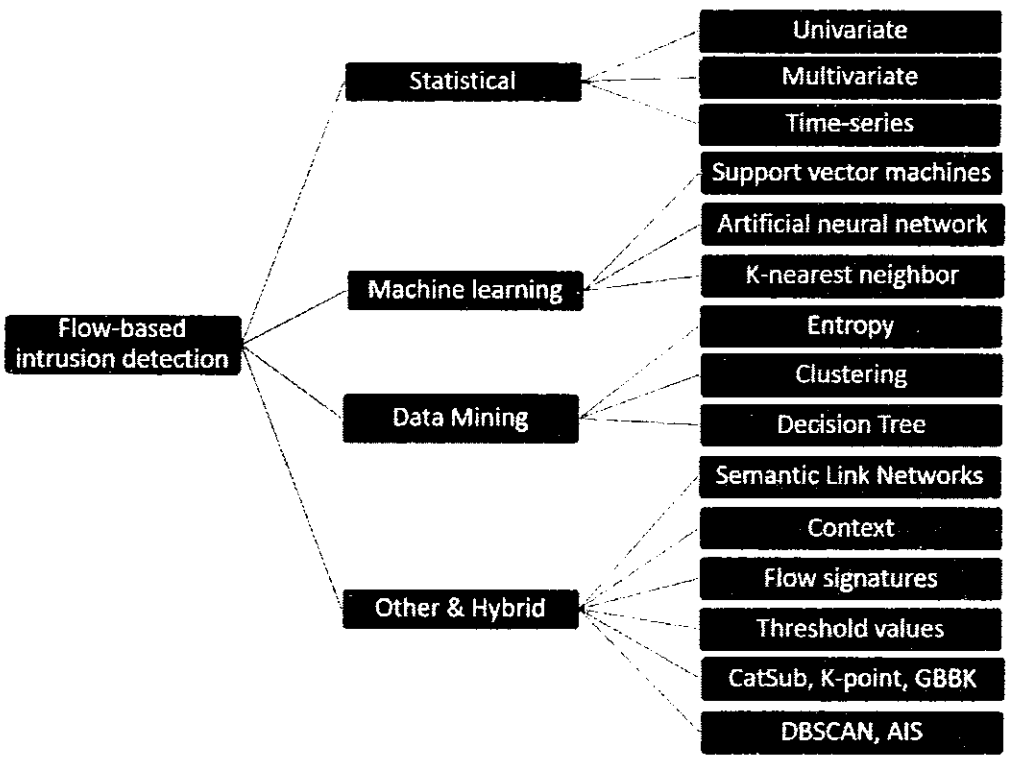


Figure 3.1: Taxonomy of flow-based intrusion detection system

3.4.1.1 Univariate statistical techniques

Univariate statistical techniques analyze a single variable at a time e.g., mean and standard deviation. These techniques assume an underlying known distribution of data. [34] presents a flow based system which analyzes the protocol behavior to detect TCP based slow and fast scans. TCP port scanning is the first step in launching an attack and attackers use TCP scans to determine the port numbers of critical user services. The authors construct a long-term and short-term profiles of TCP scans. The long-term and short-term profiles have different parameters of flows with their statistical mean and standard deviation. These parameters values are used as a threshold to detect a TCP scan. Authors compared the proposed system with a well known IDS Snort for detection of TCP scans in the incoming flows. The flow data for the evaluation of is generated from a live network connected over the Internet. The results show that the proposed system detects all 13 types of scans as compared to Snort, which can only identify 8 type of scans. The proposed technique is suitable for preliminary passive protection of a network. It cannot detect TCP scans which stealth themselves as legitimate network traffic and keep the mean and standard deviation values under the threshold.

A technique to detect flooding attacks in backbone network traffic is proposed in [35]. The traffic traces are aggregated into flow records using sketch data structure. Least Mean Square (LMS) filter and Pearson Chi-square deviation are used to detect changes in the flow records. The authors used MAWI dataset for evaluation [36]. The results show that the proposed approach outperforms other divergence techniques and achieve a detection rate of 100% with false alarm rate of 3.8%. However, these results are obtained by converting a packet-based dataset to custom flows and standard Netflow/IPFIX flow records are not used as input.

In [37], the authors propose a novel flow-based network traffic metric, Congestion Participation Rate (CPR). The CPR is used to detect low-rate DDoS attacks. The CPR of a flow F is defined as the ratio of incoming packets in congestion to the total incoming packets of that flow. A high CPR value means there is a greater probability that flow is malicious. All flows with CPR greater than the predefined threshold are classified as malicious and dropped. The authors have conducted validation experiments using NS2 simulations, test-bed experiments, and LBNL/ICSI enterprise traces. In all experiments, the calculated CPR under range for normal flows and become high in case of LDoS attack. The CPR has the advantage of detecting small scale slow-ramped attacks and can be used in integration with other intrusion detection techniques.

In [38], the authors propose a solution for detection of DNS tunnels using flow records. In DNS

tunneling, another protocol or payload is tunneled through DNS packets. DNS tunneling can pose a significant risk to the network. In this paper, eight flow-based variables are derived from the flow record which acts as indicative of DNS tunneling. Three different anomaly detection technique, Threshold method, Brodsky-Darkhovsky method and distribution based methods have been used to evaluate the traffic characteristic using the flow-based variables. The approach is validated using various datasets and results shows that the method can detect different tunnel usage scenarios with a high detection rate.

The IPFIX/Netflow export process exports flow records after certain time interval. During time interval, short-lived attacks can persist and are not detected until the flow records are exported and processed by the intrusion detection system. [39] proposed a solution for real-time intrusion detection of DDoS attacks using NetFlow and IPFIX. The authors extend the IPFIX/Netflow export process and directly connect it with a lightweight intrusion detection module. The intrusion detection module uses a time-series forecasting based on Exponentially weighted moving average (EWMA) mean calculation. Specific metrics related to DDoS attacks are measured and compared with the forecasted value. The traffic sample is considered malicious if the measured value does not lie within the range of the forecasted value. The characteristic of malicious flows are added to a blacklist which is then used to filter the malicious traffic. The technique is validated on a dataset captured from a service provider backbone network. The detection algorithm achieves 92% detection rate and

3.4.1.2 Multivariate statistical techniques

Multivariate techniques analyze the relationships between two or more variables. Multivariate techniques include Principle Component Analysis, Linear discriminate analysis, and discriminate analysis. [40] used Principle component analysis (PCA) for anomaly detection in flow data. PCA is an unsupervised learning dimensionality reduction technique. The authors used sketch structure with hashing of network traces. The hashed network traces are converted into entropy time-series and given as input to PCA classifier. The technique used three-step sketch structure which helps in attaining higher detection rate and lower false positive. The technique is evaluated over nine-year traces of MAWI dataset with different parameter tuning. The variants of proposed technique show improvement in results when compared to other PCA based anomaly detectors over the same dataset. The maximum accuracy obtained by the authors for F1-measure is 0.90 which is low if compared to other techniques discussed earlier. The reason for low F1-measure can be attributed

Table 3.7: Summary of flow-based intrusion detection system using statistical techniques

Article	Technique	Dataset	Performance Measure	Result
[34]	Mean and Standard deviation	Custom	No of TCP scans detected	100%
[35]	Chi Square Deviation	MAWI	Detection rate	100%
[37]	Change Detection	LBNL/ICSI	Comparison	Maximum
[38]	Change Detection	Custom	Detection rate	100%
[40]	PCA	MAWI	F1-Measure	0.0984
[41]	PCA	Real flow data	True Positive	94%
[42]	Holt-winter double exponential smoothing	MAWI	ROC Curve	Maximum
[43]	Holt-winter double exponential smoothing	LBNL/ICSI	Detection accuracy (False positive reduction)	99.99%
[39]	(EWMA) Forecasting	Detection rate	Custom	92%

to the real-time nature, variance and complexity of the dataset used.

A profile-based anomaly detection system using principal component analysis and flow analysis is proposed in [41]. This approach creates profiles for all types of normal traffic flow which are called Digital Signature of the Network Segments (DSNSFs). The process of anomaly detection is divided into two steps: traffic characterization and anomaly detection. The traffic characterization step extracts quantitative attributes from the flow records and creates the corresponding DSNSFS using principal component analysis. The anomaly detection step creates confidence bands using DSNSFs. These bands are matched with the normal traffic signatures, and any abnormality is notified to the system administrator. The proposed system in evaluated on a real network using sFlow flow export and collection protocol. The evaluation of anomaly detection phase achieves 94% True Positive Rate (*TPR*). However, this technique is difficult to implement in modern networks as it is not possible to generate the signatures of all networks traffic in advance.

3.4.1.3 Time-series statistical techniques

Time-series based statistical techniques use previously observed values to forecast new values. [42] apply Holt-Winters forecasting method to detect an anomaly in the flow traffic. The authors use four metrics: total bytes, total packets, the number of flows with similar volume to the same destination socket and the number of flows that have a similar volume and same source and destination address, but to different ports. These four metrics are used to detect three types of anomalies: flooding, TCP SYN, and port scan. Holt-Winters method keeps track of the normal metrics values and raises an anomaly flag if any value goes out of range. The technique is limited to only three anomalies and can be bypassed if the attacker keeps metrics values within range.

A high-speed flow level intrusion detection system (HiFIND) is presented by [43]. In this technique, a small set of packet headers fields including Source/Destination IP and Source/Destination ports is selected. It records information in efficient 2-D reversible sketches. Three types of attacks, i.e., SYN Flooding, Horizontal Scan and Vertical Scan are focused. The technique uses Holt-Winters double exponential smoothing and EWMA with season indices method for change detection in network traffic. The authors have applied three level of filters to reduce the number of false positives. The performance evolution of HiFind is carried out using both simulation and on-site deployment. A custom dataset of one-day traffic traces consisted of 900M flow records is used. HiFIND is applied in three phases and false-positives are reduced by separating the intrusion and network anomalies caused by misconfiguration. The authors compared the HiFIND with other statistical detection techniques for flow-based detection, and results show that HiFIND has similar accuracy but is memory efficient in worst case scenarios. HiFind is one of the few models to implement the intrusion detection systems security [44]. The use of Holt-winter double exponential smoothing and EWMA with season indices can have the drawback of statistical seasoning effect. The authors only used a 4-feature NetFlow record without including the protocol field. Therefore the system may not be able to detect the attacks sent on UDP packets. Another limitation of the HiFIND system is the inability to detect small and slow-ramped attacks.

3.4.2 Machine Learning

Machine learning techniques have been extensively used in packet-based and state-full intrusion detection system [45], [33], [3]. Machine learning techniques also remain in focus in flow-based intrusion detection. Intrusion detection models using machine learning techniques have low false

alarm rate and can adapt themselves in response to the traffic passing through. Machine learning techniques include artificial neural network, support vector machines and k-NN., In the following section, we discuss the flow-based intrusion detection systems which employ machine learning techniques to detect the anomalies in flows.

3.4.2.1 Artificial Neural Networks

Artificial neural network models human brain and uses small interconnected input units called neurons. Every neuron in neural network takes part in decision making, and the result is combined. Artificial Neural networks provide a solution to the anomaly detection problem by modeling the users behavior. Different neural networks used for anomaly based intrusion detection systems are discussed in [46].

A flow-based intrusion detection technique using the block based neural network is proposed in [47]. The authors use a hardware-based detection engine for real-time processing of high volume of data. A field-programmable gate array (FPGA) is used to construct a block based neural network (BBNN). The BBNN optimization is carried out using a genetic algorithm with the focus on to increase detection rate and decrease false alarm rate. For evaluation, the authors have compared the Sperotto dataset with DARPA. The authors have decided to use DARPA because Sperotto dataset contains fewer normal traffic samples then DARPA. The DARPA dataset is initially available in *tcpdump* format and is converted into NetFlow format use Softflowd and Flowd tools. The authors manually label These NetFlow records by reading the original DARPA dataset. The technique has been evaluated against Support Vector Machine (*SVM*), Radial basis function, and Nave Bayes methods. The detection rate of BBNN is same as of SVM, but the running time is quite good. The hardware-based BBNN took 0.005s as compared to 8.531s for SVM. Therefore, use of FPGA is promising for designing IDS for high-speed networks. More realistic results can be obtained by evaluating the technique on a flow-based data-set. Also, only a 4-feature Netflow record is used which may not be sufficient to capture the complete semantics of traffic flow.

In [48], the authors have used a two-stage neural network for intrusion detection using flow data. The first stage detects significant changes in the traffic that could be an attack. If an attack is detected in the first stage, the flow data is forwarded to a second stage that determines the type of attack. Authors have used a multi-layer feed-forward neural network (MLFF) for attack detection in stage-I and radial basis function network (RBFN) for attack classification in stage-II. The stage-I used six features while stage II 11 features. All stage I & II features are computed from a Netflow

v5 records. The Netflow records are generated from DARPA dataset using softflowd tool. Three different training algorithms, Resilient back-propagation, Levenberg-Marquardt and Radial Basis Function networks, have been used for training of both neural network stages. The first stage neural network gives 94.2% detection rate and 3.4% false positive rate with Levenberg-Marquardt network. For the second stage, the best detection rate of 99.42% is obtained with Levenberg-Marquardt network while Radial basis function gives the lowest false positive rate of 2.6%. Use of multiple stages helps in achieving higher efficiency since most of the input records are discarded in stage-I. The proposed system provides a comprehensive framework for all flow-based intrusion detection.

A multi-layer perceptron (MLP) with heuristic optimization algorithm to detect anomalies in flow-based traffic is suggested in [49]. The MLP interconnection weights are optimized using two heuristic techniques: Cuckoo and Particle Swan Optimization with Gravitational Search Algorithm (PSOGSA). Two datasets, DARPA and a subset of Sperotto dataset [50], have been used. The comparison of results shows that multi-layer perceptron with PSOGSA optimization gives the highest accuracy of 99.55% and 0.21% false alarm rate. However, the authors concluded that the proposed approach uses centralized processing and cannot detect distributed attacks such as DDoS.

3.4.2.2 Support Vector Machines

Support Vector Machines (SVM) is a classification technique which maps the dataset in an n-dimensional space. SVM uses vectors in the space as classes. If the data is not linearly separable, kernel functions are used to construct high-dimensional space. SVM gives accurate results and lower false positive rate in intrusion detection [3]. The SVM has also been used to flow-based intrusion detection.

A one class-SVM based model for intrusion detection using flow data is proposed in [50]. The one-class SVM learns the behavior of a single class type. The authors used the malicious dataset for the training of the one-class SVM. The learning on malicious records is fast since the ratio of malicious flows is low as compared to the normal flows. The dataset used for evaluation of the one class-SVM is extracted from the Sperotto dataset [28]. The dataset consists of 200 flow records and attributes. The accuracy results obtained with 98% accuracy and 0% false alarm rate. However, the accuracy results can drop down to 72% if a port attribute is missing. The technique has several weaknesses as explained in the author's public errata[3].

[3]https://www.cs.princeton.edu/ pwinter/ntms11-errata.html

Table 3.8: Summary of flow-based intrusion detection systems using Machine Learning techniques

Article	Technique	Dataset	Performance Measure	Result
[47]	ANN	DARPA	True Positive rate	99.92%
[49]	ANN	Sperotto, ADFA	Detection accuracy	99.55% for Sperotto Subset
[48]	ANN	DARPA	Detection Rate	1st stage - 94.2%, 2nd stage 99.42%
[50]	SVM	Sperotto	Class prediction	98% for Malicious traffic
[51]	SVM	Custom	Detection accuracy	92%
[52]	kNN, Fuzy Logic	Sperotto, custom	Correctness rate	99.34% for Sperotto
[53]	k-mean clustering	Sperotto	Purity measure	0.9577 for Sperotto

In [51], the authors present a technique for anomaly detection in large volumes of Netflow records using support vector machines. The technique takes into account both the contextual and the quantitative information of Netflow records. The approach applies kernel function on the netflow records and forward the computed values to a one-class SVM. The technique is evaluated on Netflow data volumes provided by an internet service provider. The authors used Flame tool to inject eight different attacks in the dataset. The results by using the *OC-SVM* are very promising and an average accuracy over all attack classes of 92% is obtained.

3.4.2.3 K-Nearest Neighbor(k-NN)

K-NN uses the knowledge of neighboring points to classify the input example. K-NN is extensively applied for packet-based intrusion detection systems [54] [55] [56] and also used for flow-based intrusion detection.

A flow-based intrusion detection system using k-NN technique with fuzzy logic is proposed in [52]. The work uses k-NN for selecting the best matching class. Least Mean Square technique is used for error reduction. The flow-based intrusion detection systems required additional computational intelligence as compared to packet-based systems as only the header information is available for decision. Fuzzy logic, therefore, seems to be a good choice for selecting the flow class label. Although the technique gives good results, the authors tested with only 200 training examples whereas the actual dataset contains around 14.2M records [28].

In [53], the authors proposed an improved nature-inspired technique for Optimum-path forest clustering (OPFC) and apply it to network intrusion detection. The OPFC is a k-NN graph which uses probability density function for weighting of nodes. The authors use Bat Algorithm, Gravitational Search, Harmony search and Particle Swarm Optimization techniques to determine the best value of k. The authors have also compared the performance of OPFC with K-mean clustering and self-organizing maps (SOM). All three techniques are applied to eight packet and flow-based datasets. The results of the evaluation are obtained in the form of Purity measure. The evaluation on the Netflow dataset gives purity measure of 0.9577, 0.75945 and 0.2145 for OPFC, K-mean and *SOM* respectively. Therefore OPFC outperforms the other two clustering techniques in flow-based detection.

3.4.3 Data mining Techniques

Data mining techniques identify novel and useful patterns in the data. Data mining techniques include Decision Trees (*DTs*), Entropy-based distribution, and clustering.

3.4.3.1 Decision Tree

Decision Trees (DTs) are supervised learning techniques used for classification. The decision tree creates a tree model by creating rules based on the attribute value for every tree node. The decision tree has also been applied to intrusion detection [57].

A flow-based solution to detect botnets is given in [58]. Botnets are a collection of compromised hosts controlled by a malicious user for various types of attacks and cyber crimes [59]. The authors argue that flow-based approaches are better than payload inspection since most of the botnets use encrypted communication channels. The proposed method uses a decision tree algorithm with Reduced Error Pruning algorithm to construct the botnet classifier. The flow records consist of 12 flow attributes. The classifier is evaluated on a dataset containing traces of two botnets. The technique gives a detection rate of 98.3% and 99.9% for malicious and non-malicious categories. The technique also successfully detects novel botnets. The proposed technique is simple and efficient, but it can be evaded by inflicting small changes in the attribute values. Also a flow analysis window of 300 seconds is too long, and the algorithm can miss small scale malicious flows [60].

3.4.3.2 Entropy

Entropy captures the important characteristic of features in traffic distribution. These features are used to detect the abnormal and malicious behavior in the network traffic. An algorithm for detection of abrupt changes in network entropy time series is proposed in [61]. The technique is based on the idea that any network attack will inflict significant changes in flow attributes which can be detected in entropy time series. The authors have proposed an abrupt change detection algorithm which uses a dynamic anomaly score to detect the attack. The algorithm is evaluated on a dataset obtained from an ISP's server. The dataset contains five days traffic of unidirectional network flows. Two synthetic anomalies HTTP DoS attack and horizontal network scan are manually injected at a specific time. The technique successfully detects the change in the traffic at the given time. As also indicated out by authors, the technique can be evaded with small scaled DDoS attacks. However, the technique does not need training data and can be straightforwardly used to monitor the network activity.

A technique to detect large-scale anomalies in the network traffic is proposed by [62]. The technique stores the profiles of normal traffic. All incoming flow records are first aggregated and subsequently compared with profiles of normal traffic. The deviation is then measured using the Shannons Entropy formula. For evaluation, a custom dataset is obtained from a commercial service provider and attacks are manually inserted using the Flame tool. A drawback of the technique is the requirement of normal network profiles which are quite difficult to generate in multi-service real-world networks.

An entropy-based Internet traffic anomaly detection system is discussed in [63]. The authors have used Shannon, Renyi and variants of Tsallis entropies combined with a set of feature distributions. The entropy techniques are employed in a flow-based framework. A variant of the Sperotto dataset is used for evaluation of the proposed methodology. In training mode, a profile is created for normal traffic using time-specific entropy values. Any value exceeding the upper limit of entropy limit is considered abnormal. The results show that Tsallis and Renyi's entropies performed best while Shannon entropy and counter-based methods performed poorly. This technique can generate false positives if there is a benign change in network traffic such as congestion.

In [64], authors describe a entropy-based approach for DDoS attack detection. The technique calculates the entropy values of the selected flow features. The features are used by the clustering algorithm to construct a normal flow profile. This normal flow profile is used to detect DOS attack in the incoming traffic. The technique is evaluated on the DARPA dataset and results are obtained

in the form of DF rate. DF rate is defined as the ratio of detection rate and false positive rate. is ration. The technique obtains a best DF rate of 7

Table 3.9: Flow-based intrusion detection systems using data mining techniques

Article	Technique	Dataset	Performance Measure	Result
[58]	Decision Tree	hh	Detection rate	98.3% and 99.9% for malicious and non-malicious
[62]	Entropy	Custom	Graph Generation	Anomaly detected
[64]	Entropy	DARPA	DF Rate	7
[63]	Entropy	Sperotto	Anomaly type detection	100%
[65]	Clustering	Custom	Detection rate	99.99% & 92.4% for successful and unsuccessful attempts
[41]	Clustering	Custom	True Positive rate	99.99% & 92.4% for successful and unsuccessful attempts

3.4.3.3 Clustering

Clustering techniques identify hidden patterns in the data and group the similar instances together. [66] proposed a network anomaly detection system using multiple unsupervised clustering techniques. The system captures packets from the network and aggregates into flows at random time slot. A change detection algorithm based on time series analysis is used to separate the malicious flows. The technique uses sub-space clustering and density-based clustering to create partitions of data in each sub-space. The algorithm also rank the clusters in order of abnormality. All clusters above the detection threshold are considered anomalies. The detection threshold is unique for every type of attacks The technique is evaluated on MAWI dataset and results are obtained in the form of ROC curve. The proposed technique has a larger area under ROC curve when compared with other unsupervised learning methods. This technique has the advantage that it requires no signature or training and can be instantly used to monitor the network traffic.

A ward clustering approach to detect the dictionary attacks against Secure Shell (*SSH*) is presented in [65]. SSH is a common way to access the remote servers over Internet and remain a favorite attack target. The authors used two key innovations to detect an attack in SSH protocol. First, two criteria have been employed that are not available in original SSH protocol i.e. checking the

existence of connection protocols and inter-arrival time of an auth-packet and the next. Second, authors have identified transit point of each sub-protocol in SSH. The two criteria and flow of traffic during the sub-protocol transit point is inspected. The technique uses Ward's clustering method based on Euclidean distance for evaluation of flow data. The proposed technique is evaluated on a dataset generated through two observation points connected to a server to Internet. The best results include 99.90% detection rate for unsuccessful SSH attack attempts and 92.80% detection of successful SSH attempts. The technique is promising for detection of stealthy dictionary attacks in SSH and should be used at host-level detection.

3.4.4 Hybrid techniques

Hybrid techniques employ more than one anomaly detection techniques to improved the detection accuracy. The anomaly detection methods are connected in serial with each other and process the input data sequentially. [15] propose a model of multilayer intrusion detection system using both flow and packed-based detection. The authors acknowledge the importance of flow-based intrusion detection due to efficient processing and low deployment cost. However, it is also argued that flow-based detection can suffer from information loss when flow data is extracted from the network packets. This information loss can make the flow-based detection system to make less accurate decisions. On the other hand, packet-based approaches suffer from high computational cost because complete packet payload is inspected. The authors suggest that a flow-based detection should be used at the first layer and the second layer performs deep packet inspection.

[67] proposes a multi-level hybrid intrusion detection method which combines supervised, unsupervised and outlier-based methods for intrusion detection. Three algorithms, CatSub+, K-point and GBBK have been used for supervised, unsupervised and outlier detection respectively. The hybrid framework is evaluated on a number of datasets including a flow-based dataset, TUIDS. Although results for flow-based evaluation are good, the technique has a significant weakness. The selection of supervised, unsupervised or outlier-based classifier at a particular level for a given dataset is based on the classification accuracy of the individual classifier for a given dataset. This adjustment is difficult to implement in real-time scenarios and similar performance results might not be achieved.

[68] present a distributed intrusion detection system based on Artificial Immune System and unsupervised clustering. The authors have used DBSCAN clustering algorithm. The clustering engine labels the network traffic as malicious and non-malicious. The output of clustering engine is used

to prove online and real-time training data for training of primary immune response detectors. The immune response detectors are placed around networks hosts. The technique is evaluated on KDD99 dataset and achieves F1-measure of 0.738.

Table 3.10: Flow-based intrusion detection systems using hybrid and other techniques

Article	Technique	Dataset	Performance Measure	Result
[67]	CatSub+,K-point,GBBK	TUIDS	Recall	0.99, 0.98 and 0.98 for DoS, Probe and Normal
[68]	DBSCAN, AIS	KDD99	F1-score	0.738
[69]	Threshold values	Custom	True Postie rate	99%
[70]	Flow signature	Custom	Brute force Detection	Yes
[71]	Semantic Links	Sperotto	F1-score	0.97
[31]	Threshold value	SSHCure	Detection accuracy	83% and 99%

3.4.5 Other techniques

[69] propose a flow-based system, SSHCure, for detection of SSH attacks. The SSHCure employs an efficient algorithm for the real-time detection of ongoing attacks and allows identification of compromised attack targets. Three phases of the SSH attack scanning phase, brute force phase, and die-off phase are identified. During scanning phase, the attacker scans an IP to find the SSH daemon. In brute-force phase, the attacker tries to log in the SSH server. If the login is successful. The die-off phase shows the attack traffic between an attacker and target host. SSHCure uses two flow metrics to detect an attack in all three phases. The first metric is an upper bound of two packets per flow. The second metric defined a minimum number of flow records for every attack. Every attack phase uses a different threshold value for the two flow metrics. The detection performance of the system is validated with empirical traffic data. The real-time implementation of SSHcure shows that algorithm has various shortcomings that ultimately cause compromises to remain undetected or false alarms to be raised [31].

[70] present a flow-based detection technique for Remote Desktop Protocol (*RDP*) brute force attack detection. The technique analyzes the traffic signature of RDP clients, brute force tools and successful authentication events. The flow signatures are evaluated on campus network of Masaryk

University for two months. The authors used RdpMonitor, a publicly available NfSen plug-in with derived NetFlow signature to automate detection. The plug-in successfully detect the malicious traffic and reported that approximately 40% of all RDP related traffic in the campus network is malicious.

[71] propose a novel approach for detection of cyber attacks using Semantic Link Networks (SLNs). Semantic Link Networks mine time, location, and other contextual information from the flow data. The contextual information is used by the semantic links among the alerts for suspicious flows on probabilistic semantic networks (SLNs). These semantic links help in retrieving relevant suspicious activities that can be a part of multi-step attacks. The technique is evaluated on a mix of Sperotto and ICSX dataset and achieved F1 score of 0.97. The semantic link technique is also compared with other flow-based intrusion detection system, and results show that it outperforms other methods.

[31] enhance the SSHCure and propose two phase detection algorithm for SSH compromise detection. The first phase is brute-force phase. The brute-phase is detected by checking for the same number of packets per flow. Two unsuccessful connection attempts from the attacker will have the same number of packets per flow. The second phase is compromise phase. The technique transforms the compromise phase in six attacks scenarios. If the flow of SSH traffic matches a particular attack scenario, an attack is detected, and the connection is closed down. The technique is validated on the SSHCure dataset and obtains an accuracy of 83% and 99% for the two segments in the SSHCure dataset.

3.5 Commercial applications of flow-based intrusion detection

The flow-based intrusion has also gained attention of commercial vendors. The Lancope's StealthWatch System[4], available through Cisco, is an enterprise network monitoring and security intelligence solution completely based on NetFlow. The StealthWatch system performs collection, aggregation, and analysis of NetFlow and other contextual data. The system s able to detect malicious behaviors linked to APTs, insider threats, DDoS and malware. Cisco's Next-generation Intrusion Prevention System[5] with FireSIGHT Management Center[6] also have Netflow capabilities.

[4]https://www.lancope.com/products-services-lancope
[5]http://www.cisco.com/c/en/us/products/security/ngips/index.html
[6]http://www.cisco.com/c/en/us/products/security/firesight-management-center/index.html

The Scrutinizer System[7] developed by Plixer Inc. is a flow-based incident response and behavior analysis product. Scrutinizer performs the collection, threat detection, and reporting of network activities using a number of flow technologies. It also offers real-time situational awareness and historical behaviors of the network. The system used fixed algorithms to detect DoS attacks, network scans and other abnormal network behavior.

The IBM's QRadar Security Intelligence Platform[8] is a security information and event management (SIEM) system with anomaly detection, incident forensics and vulnerability management. The QRadar also include flow-based analysis support.

Juniper Networks JSA Series Secure Analytics[9] gives a complete set of network monitoring and surveillance tools for enterprise level. The JSA series products have support for NetFlow, J-Flow, sFlow, and IPFIX. It also includes Network Behavior Anomaly Detection (NBAD) to detect rough servers, and APTs flow-based network activity.

In open-source world, the Bro[10] is a comprehensive platform for general network traffic analysis. It also has intrusion detection capability. In addition to other network tapping tools, Bro also uses Netflow for network analysis and threat detection [72].

3.6 Challenges and Open issues

The comparison of the packet and flow-based techniques shows that flow-based techniques are a better choice for protection of high-speed networks [15, 69]. However, [21] suggested that flow-based intrusion detection is not matured enough to replace the traditional packet-based techniques. [15, 21] propose that flow-based detection should be applied in combination with packet-based detection. Flow-based intrusion detection should be implemented at an initial layer of intrusion detection while more in-depth intrusion detection can be performed at a second layer using packet-based inspection. In this survey, we review the available flow-based intrusion detection systems and discuss their important features. We have identified the following challenges for further research in flow-based intrusion detection:-

- Intrusion detection datasets are a valuable tool for evaluation of proposed techniques. There

[7]https://www.plixer.com/products/scrutinizer/
[8]http://www-03.ibm.com/software/products/en/qradar
[9]http://www.juniper.net/us/en/products-services/security/secure-analytics/
[10]https://www.bro.org/

are very few public flow-based datasets exist. [35, 47, 48, 64] use packet-based datasets to generate the flow records for evaluation of flow-based techniques. However, evaluation on such datasets is not guaranteed to give similar results in real-world flow-based detection. Some techniques [51, 69] use custom datasets which makes it difficult to compare the results with other techniques . There is an urgent need for the development of public flow-based datasets having a variety of attack traffic for evaluation and comparison of different flow-based intrusion detection techniques.

- IPFIX/Netflow specify around 280 attributes for flow record. Researchers use a different selection of flow attributes to evaluate the flow of traffic. [47] and [49] use a 4 and 7-tuple flow record respectively. [58] employ a 12-tuple flow record. However, there is no literature exists which establish the relationship between flow attributes and attack types. Increasing the number of flow attributes will also increase the computation whereas using few flow attributes can miss important network information. It is therefore important to explore the relation of flow attributes with attack types.

- Some techniques do not use a representative dataset to obtain validation results. [50, 52] use a dataset of 200 flow records for validation whereas original dataset has near 1.4 millions records. The techniques evaluated on such dataset will not perform better in real-world and will give many false-positives. [73] presents a solution for obtaining representative instances from large datasets. The approach is useful when input size is sufficiently large and can affect the space complexity of the algorithm.

- Flow-based techniques have relatively less information about network traffic as compared to packet-based techniques [74]. Flow-based intrusion detection systems cannot detect attacks which are embedded in packet payload and do not alter the flow of traffic such as SQL injection and cross-side scripting. Some techniques use flow records consisting of four or five flow attributes [49, 50] which are not sufficient to analyze the flow of traffic. [15, 21] suggest the use of packet-based intrusion detection with flow-based techniques. However, such techniques will still have the drawbacks of packet-based detection. A solution this problem is computation of additional flow metrics using basic flow attributes [37].

- Some techniques examined in the previous section only address specific attack types. [34] only detects TCP scan and [65, 69] give solution for protection against SSH attacks. [37, 39] address the issues of DOS attack. Such techniques give better results for the particular attack type or scenario but have not been evaluated against other attacks. It is difficult to integrate

such techniques into a comprehensive flow-based intrusion detection framework. Therefore research should be focused on all flow-based intrusion detection framework.

- [35] do not use Netflow/IPFIX flow record for analysis of flow. Use of custom flow-records can affect the portability of the intrusion detection system. All major hardware vendors support Netflow/IPFIX and no additional equipment is needed for flow collection and export. Also, use of Netflow/IPFIX ensures that the intrusion detection framework can be used in a different network environment without any changes
- The value of flow export interval critically affect the performance of flow-based intrusion detection system [74]. [58] use a flow duration of 300 seconds. If flow export interval is kept longer, a short-timed attack can occur, and intrusion detection system may not be able to detect it. On the other hand, a shorter flow-interval can overload the IDS and can affect the overall efficiency. This effect of flow export interval on IDS performance should be analyzed in detail [39].
- An important issue in flow-based intrusion detection is the use of flow sampling techniques. The Packet Sampling (PSAMP) Protocol [75] specifies different sampling techniques for flow export process. Sampling techniques which are a true representative of the original population can help in archiving better results. Therefore it is important to investigate the effect of sampling techniques with the accuracy and efficiency of flow-based intrusion detection.

The above discussion and the review of available techniques reveal that although flow-based detection is promising, it needs further research and exploration to replace the existing packet-based intrusion detection systems.

3.7 Chapter Summary

This chapter reviews the current state of the art in flow-based intrusion detection. We survey the research performed in flow-based intrusion since 2008. The chapter starts with a brief overview of flow-based datasets. There are many datasets available for intrusion dataset and but only a few datasets can be used for flow-based evaluation. We give a summary of every dataset and describe the methods used for collection of the dataset. We construct a taxonomy of flow-based intrusion detection systems the on the basis of the technique used detection of malicious flows. We described

the architecture, classification algorithm and important aspects of available flow-based techniques. We also discuss the strengths and weaknesses of the every technique. In the end, we identify existing challenges and open issues in flow-based intrusion detections.

Chapter 4

A two-stage model for flow-based intrusion detection

4.1 Overview

Traditional network-based intrusion detection systems using deep packet inspection are not feasible for modern high-speed networks due to slow processing and inability to scan encrypted packet content. As an alternative to packet-based intrusion detection, researchers have focused on flow-based intrusion detection techniques. Flow-based intrusion detection systems analyze flow records for attack detection. Flow records contain summarized traffic information. However, flow data is very large in high-speed networks and cannot be processed in real-time by the intrusion detection system. In this chapter, we propose a novel efficient two-stage model for intrusion detection using flows records. The first stage in the model classifies the traffic as normal or malicious. The malicious flows are further analyzed by a second stage. The second stage associates an attack type with malicious flows. The proposed two-stage model is efficient because the majority of flows are discarded in the first stage and only malicious flows are examined in detail. We also describe the implementation of our model using machine learning techniques.

4.2 Previous work

Two different approaches are in use for multi-stage detection of network attacks. The first approach considers a single attack type spanned over multiple stages. Various stages of an attack include vulnerability scan, weakness exploitation, invasion, control, and spread. Every stage of an attack corresponds to a detection stage in the multi-stage IDS. In [76], a technique for detection of a single type of attack using multi-stage traffic analysis was proposed. Similarly, a multi-stage IDS using Hidden Markov Model is presented in [77]. Every attack stage is analyzed by detection agents using predefined attack signals. The signals of all attack stages are estimated by a determination stage using Hidden Markov Model for final intrusion detection decision. The IDS is evaluated on DARPA dataset and achieved a detection rate of 90%.

The other method for multi-stage IDS detects a different type of attack in every stage. The first stage detects preliminary attacks while additional stage use detail inspection to detect more sophisticated attacks. In [78], a network intrusion detection technique using Learning Vector Quantization(LVQ) was proposed. The authors used multiple stages to detect different types of attack. The technique was evaluated on DARPA dataset and achieved very low error rate. A multi-stage filter using enhanced AdaBoost for network intrusion detection is presented in [79]. The technique is evaluated on DARPA dataset and achieved good results for some attack types. A malware prevention and detection system using a combination of signature and anomaly-based IDS is presented in [80]. The signature-based IDS uses general characteristics of attack for detection. The anomaly-based IDS is implemented using the RIPPER classifier. The signature and anomaly based IDS are implemented in three stages. The first stage classifies the traffic as normal or malicious. The second stage determines the attack type while the third stage determines the variant of a particular attack type. The technique is evaluated on the NSL-KDD99 dataset and achieved F1-measure of over 0.97 for different stages.

In [81], a multi-stage detection model using time-slot and flow-based detection. is proposed. The time-slot detection stage checks the incoming traffic for obvious traffic characteristic. This stage classifies the traffic into normal, suspicious and malicious categories. The traffic detected as suspicious is converted into flows and forwarded to the flow-based detection stage. The technique is evaluated in DARPA dataset and achieved a detection rate of 68.4%.

In [82], the authors proposed a real-time multi-stage intrusion detection system using unsupervised learning to improve the detection rate of unknown attacks. The system uses flow records for attack detection. The multi-stage model uses two detection engines. The first engines use sub-space

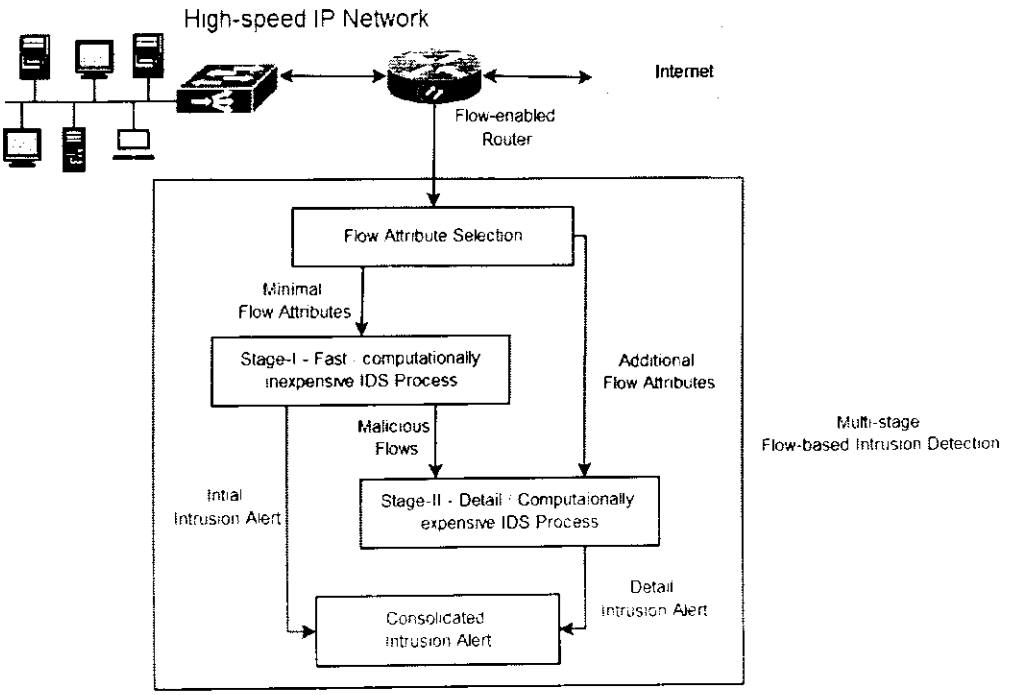


Figure 4.1: Architecture of two-stage intrusion detection model

clustering and to detect DoS, DDoS, and other attacks. The second detection engine analyzes the relation between attackers to detect Bot-master. The proposed technique focused on improving the detection rate of unknown attacks by additional flow features.

Our proposed approach differs from the existing work. Unlike most of the techniques, our model uses flow records instead of packets for intrusion detection. Our model separates the normal and malicious flow in the first stage and determines the attack type in the second stage. The implementation of our model uses a one-class and multi-class classification at the first and second stage. The use of one-class classification in a multi-stage model is a novel idea. The next section presents the architecture of our proposed model.

4.2.1 Architecture of the two-stage intrusion detection model

Although flow records contain summarized network traffic information, the flow data can be very large in high-speed networks [83]. Flow monitoring and analysis tools employ packet sampling techniques to obtain a subset of flow records [14, 84]. Furthermore, IPFIX defines around 280 flow attributes. Additional flow attributes can also be computed using the base flow attributes by

the IDS to detect different types of network attacks. Large input size and high feature space can overload the IDS. Also most of the traffic in the network is normal as compared to malicious traffic. Processing of malicious as well as normal traffic by the IDS will be performance bottleneck.

We propose a two-stage model for intrusion detection in high-speed networks using flow records. Figure 4.1 shows the architecture of our proposed approach. The flows are collected from the network using a flow-enabled device. These flows are passed through an attribute selection step. The two-stage model uses two stages for attack detection. The first stage selects a minimal set of attributes and determines whether incoming flows are normal or malicious. The first stage uses a fast and computationally inexpensive technique for detection. It discards the normal flows and forwards the malicious to the second stage detection process. An initial intrusion alert is also sent to the consolidated intrusion alert module.

The second stage process performs detail intrusion detection on the malicious flows. The size of malicious flows is very small in overall network traffic. Due to small input size, the second stage can commit additional resources for detailed and accurate detection of an attack type. The second stage analyzes the malicious flows and determines the attack type. The second stage can also use additional flow attributes for precise detection of an attack. If the flows do not belong to any attack class, they are marked as unknown in the detail intrusion alert. The unknown flows can belong to an unseen attack, or they can be false posties of the first stage. The second stage sends a detail intrusion alert to the alert module. The alert module raises a consolidated alert combining the alert information received from both detection stages.

Our proposed two-stage model discard normal flows in the first stage and ensures that only malicious flows are subject to detail intrusion detection. This increase the efficient of our model because no resources are consumed in the processing normal flows. Another benefit of our approach is the reduction of false positives. If the malicious flows detected in the first stage contain false positives, the second stage process does not associate a class type with such flows. In next section, we give implementation detail of our model using machine learning techniques.

4.2.2 Implementation of two-stage intrusion detection model using machine learning techniques

We propose the use of machine learning technique for implementation of the proposed model. Machine learning techniques have been extensively used in intrusion detection systems for adapt-

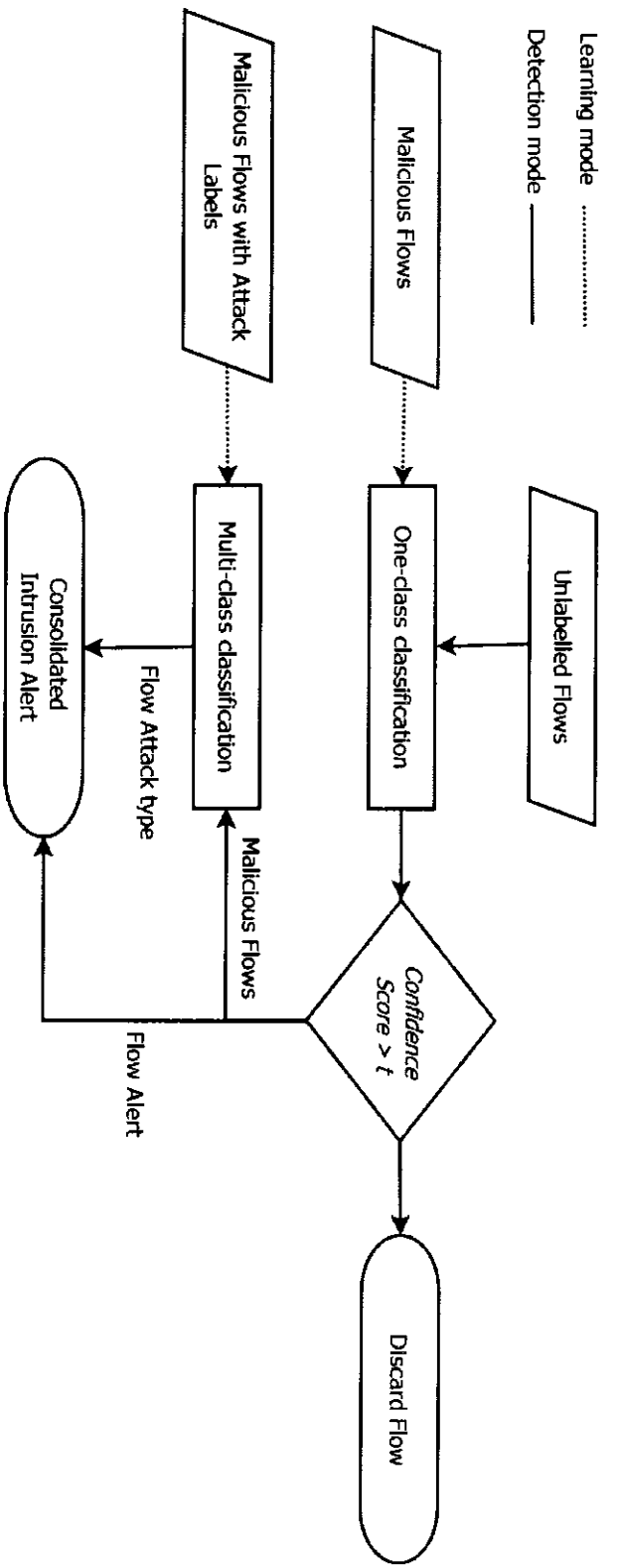


Figure 4.2: Two-stage intrusion detection model flow diagram

ability and improvement of detection rate. [27, 45].

The implementation of the two-stage model using machine learning techniques has two modes, learning mode and detection mode. In learning mode, the classification algorithms are trained using a labeled set of flows. Our model uses two training sets. The fist stage requires a training set consisting of only malicious flow. The second stage uses a training set of malicious flow with attack labels. In the detection mode, the IDS process unlabeled flows and raises consolidated intrusion alerts.

Figure 4.2 shows the implementation of our model using machine learning classification algorithms. The first stage detection process only detects malicious flows. There is only one target class in the first stage. We have proposed the use of the one-class classification for detection of malicious flows in the first stage. One class classification techniques learn the model for one target class. It only recognizes objects of target class and all other objects are rejected. The training set for one-class classification technique also consists of target class objects [85].

Mathematically, X is a training set consisting of only malicious flows. The one-class classifier learns an output function f_o using the optimized parameter set θ for a given flow x_i. The f_o gives a confidence score defining the membership of flow x_i with the malicious class.

$$f_o(x_i) = \theta_1 + \theta_2(x_i) \tag{4.1}$$

The value of f_o is used in a decision function h_o to obtain the classification result. For all unclassified flows $z_i \in Z$, if the value of $f_o(z_i)$ is higher then the maliciousness threshold t, the flow is classified as malicious or normal otherwise. The value of maliciousness threshold t is user-defined.

$$h_o(z_i) = \begin{cases} malicious. & \text{if} f_o(z_i) \geq t \\ normal. & \text{if} f_o(z_i) < t \end{cases} \tag{4.2}$$

The malicious flows recognized in the first stage are forwarded to the second stage. The second stage detection process associates an attack type with the malicious flows. Since the number of attack types can be more than one, we use multi-class classification technique to classify the flows into different attack types [86].

The training set Y contains labeled malicious flows for K attack types. The multi-class classifier learns an output function $f_{mk}(y_i)$ for all K attack types using the training set Y where $y_i \in Y$. The

function f_{mk} gives a confidence score for all attack types in K.

$$f_{mk}(y_i) = \theta_1 + \theta_2(y_i) \forall k \in K \tag{4.3}$$

For all unclassified flows $z_i \in Z$, The incoming flow is classified into the attack type for which the function $f_{mk}(z_i)$ gives the highest confidence score.

$$h_m(z_i) = \arg\max_{k \in K} f_{mk}(z_i) \tag{4.4}$$

The classification result of both stages is combined in a consolidated intrusion alert module. The alert module output the maliciousness of flow and the possible attack type in the alert. The information can be used by the security administrator to protect the integrity of the network.

4.3 Research Methodology

This thesis explores the application of machine learning techniques for development of a flow-based intrusion system for next-generation networks. In the first step, we have proposed an efficient flow-based intrusion detention model. In next step, we will use available machine learning techniques for implementation of our model and development of a flow-based intrusion detection system for next-generation network.

We will use publicly available flow-based datasets for intrusion detection system. Most of the datasets are available in Netflow v5 format. Therefore we will also use Netflow v5 compatible flow records for evaluation of our IDS. We plan to use Matlab and Weka for implementation of evaluation experiments. Detail of performance measures used for evaluation of intrusion detection processed is given in next section.

4.3.1 Performance measures

We plan to use precision, recall, F1-measure, ROC Curve and Area under ROC Curve to evaluate the performance of clustering algorithms [87]. Precision is the ratio of correctly clustered flows to

the number of clustered flows:

$$Precision = \frac{\text{number of true positives}}{\text{number of true positives + number of false positives}} \tag{4.5}$$

Recall is the ratio of correctly clustered flows to the number of actual number of flows:

$$Recall = \frac{\text{number of true positives}}{\text{number of true positive + number of false negatives}} \tag{4.6}$$

True positive refers to correctly clustered flows. False positives refers to the flows clustered but do not belong to the cluster. False negative refers to the number of flows incorrectly assigned to other clusters.

F1-measure is described as the harmonic means of precision and recall values:

$$F1 - measure = \frac{\text{2 x precision x recall}}{\text{precision + recall}} \tag{4.7}$$

Area under Received Operator Characteristic (*ROC*) curve (AUC) and F1 score [5]. The *ROC* curve plots the false alarm rate against true positive rate. In intrusion detection, ROC curve measures the detection rate as the false alarm threshold varies. The ROC curve is quantified by measuring the area under the curve (AUC). The value of *AUC* near 1 denotes a good intrusion detection process.

4.4 Chapter Summary

This chapter proposes a two-stage model for intrusion detection using Flow records. The first stage classifies the flow records into the normal and malicious classes. The second stage detection process performs detail analysis and classifies the flow into different attack types. We also give implementation detail of our model using one and multi-class classification. We conclude that our model is efficient because it discards the majority of the flows in the first stage using a computationally inexpensive algorithm. Only malicious flow are analyzed in detail. The two-stage detection model also reduces the false positive rate through the application of two different classification techniques.

Chapter 5

Malicious flows detection using one-class classification

5.1 Overview

In this chapter, we evaluate different one-class classification techniques for detection of malicious flows. One-class classification is employed when labeled training data is available for only one class. We have considered density estimation, reconstruction methods and boundary methods based techniques. The focus of our work is to evaluate various one-class classification techniques on a flow-based dataset for detection of malicious flows. The training dataset contains malicious flows and test dataset contains both normal and malicious flows. On the basis of results, we discuss the application of one-class classification techniques for flow-based detection of malicious traffic.

5.2 One-class classification

One-class classification is a particular case of binary classification which recognizes examples of only one class. The one-class classification is useful when training examples of only one class is available. Training sample for other classes is either not available or difficult to obtain [85]. The class for which training samples are available is called a target class. An important application of one-class classification is intrusion detection where target class is malicious category of network traffic. The one-class classification based intrusion detection only detects malicious traffic and

discards the normal traffic.

Mathematically, we assume that x_i is an training example from the target class dataset $X = \{x_1, x_2, x_3, \ldots, x_n\}$. The one-class classifier uses the training dataset X to learn a model with the optimized parameter set θ. After learning, the classifier can identify target class examples from unseen test dataset Z. The classifier defines an output function f using the optimized parameter set θ such that:

$$c_i = f(z_i, \theta) \tag{5.1}$$

Where z_i is an unseen example, and c_i is the class probability. The classifier uses a mapping function $h(z_i)$ over the output function $f(z_i)$ to map the target class probabilities into target or outlier classes. If the class probability is higher than a pre-defined threshold t, the target class is selected otherwise the example is declared outlier.

$$h(z_i, \theta) = \begin{cases} target. & \text{if} f(z_i, \theta) \geq t \\ outlier. & \text{if} f(z_i, \theta) < t \end{cases} \tag{5.2}$$

5.3 Previous work

Machine learning algorithms have remained in primary focus for designing intrusion detection systems [3]. Similarly, one-class classification has also been applied to solve the intrusion detection problem. An ensemble of one-class classifiers for intrusion detection is proposed in [88]. The authors have used a modular approach in which each module models a group of similar network protocols and services. Parzen density estimation, k-means, and v-SVM are used to construct the one-class classifier ensemble. The technique is evaluated on KDD99 dataset and results show that dividing the problem into different modules attains high detection rates with lower false alarm rates.

A flow-based intrusion detection system using one-class SVM classification is descried in [50]. The one class-SVM detects malicious flows and discards normal flows. A small subset of the UoT flow-based dataset [28], is used for evaluation.

A differential Support Vector Data Descriptor (*SVDD*) based one-class classification method to

detect more harmful attacks using host-based intrusion detection is proposed in [89]. Experimental results show that differentiated intrusion detection method performs better than existing techniques for detection of harmful attacks.

An application of one-class classification for intrusion detection in Supervisory Control and Data Acquisition (*SCADA*) networks is presented in [90]. *SCADA* networks monitor and control industrial and public service processes such as nuclear power plants, electrical power grids, gas pipelines and water distribution systems. The authors have employed two one-classification methods; Support Vector Data Descriptor (*SVDD*) [91] and Kernel Principal Component Analysis [92]. Both techniques used a SCADA network dataset for evaluation. Results indicate that both techniques tightly enclose the normal flow behavior in the SVM hypersphere and also detect intrusions.

Amer et al. [93] proposed two enhancements in one-class SVM for unsupervised anomaly detection. Both enhancements reduce the effect of outliers on the SVM model during training. Authors have compared the proposed techniques with nine other unsupervised anomaly detection algorithms and obtained promising results.

An industrial communication intrusion detection algorithm based on one-class SVM is presented in [94]. Authors have used particle swarm optimization [95] to tune the kernel parameters.

A robust one-class SVM for outlier detection is presented in [96]. The authors use dynamic weight assignment for training datasets for the smooth influence on one-class SVM. Experimental analysis shows the proposed weighted method has improved performance and robustness as compared to the conventional one-class SVM.

Survey of literature shows that although one-class classification has been in use for intrusion detection, there are many one-classification techniques yet to be explored in detail for flow-based intrusion detection.

5.4 One-class classification techniques

The one-class classification techniques are classified into density estimation methods, boundary methods and reconstruction methods [91, 97].

5.4.1 Density Estimation

The density estimation methods calculate the density of target class from training data using a density estimation function $f(z)$. The function $f(z)$ calculates the density of a test example and uses a threshold value to accept the example in the target class. If the density of test example is higher than the threshold, it is classified into target class. Otherwise, the example is considered an outlier [98]. We have evaluated three density estimation methods; simple Gaussian distribution using Mahalanobis distance, mixture of Gaussian and Parzen distribution.

5.4.1.1 Simple Gaussian distribution using Mahalanobis distance

This technique uses the Mahalanobis distance to calculate the density of test example z in the Gaussian distribution [91]:

$$f(z) = (z - \mu)^T \Sigma^{-1} (z - \mu) \tag{5.3}$$

where μ is the mean and Σ is the covariance matrix.

5.4.1.2 Mixture of Gaussian

The simple Gaussian distribution does not fit most data distributions. To model more complex data, a mixture of different Gaussian distribution is used. Density at point z for mixture of different Gaussian distributions is defined using the following equation [91]:

$$f(z) = \sum_{i}^{k} P_i e^{(z-\mu_i)^T \Sigma^{-1} (z-\mu_i)} \tag{5.4}$$

Where P_i is the probability that z belongs to ith Gaussian component, μ_i is the mean and Σ is the covariance matrix.

5.4.1.3 Parzen Distribution

Parzen density estimation uses a hypercube $\Re$ with dimension d and width h to calculate the density at point z. The hypercube $\Re$ is centered at z and density is calculated with respect to all examples x_i. We define a function o which gives a value of 1 if an example x_i is within the hypercube $\Re$ or 0 otherwise [91].

$$o(\frac{z-x_i}{h}) = \begin{cases} 1, & \text{if } x_i \text{ is inside the hypercube} \\ 0, & \text{otherwise} \end{cases} \tag{5.5}$$

$$f(z) = \sum_{i=1}^{N} o(\frac{z-x_i}{h}) \tag{5.6}$$

We use Gaussian estimate for the function $f(z)$ such that:

$$f(z) = \sum_{i=1}^{N} e^{(-(z-x_i)^T h^{-2}(z-x_i))} \tag{5.7}$$

5.4.2 Reconstruction methods

Reconstruction methods use training dataset to model the generating process for all examples. A reconstruction error is used to measure the fit of an actual example for the generating model. If the example does not fit the model and reconstruction error is high, the example is more likely an outlier [98]. Reconstruction methods include neural networks (Auto-encoder neural networks, self-organizing map) and subspace-based approaches (Principle Component Analysis).

5.4.2.1 Auto-encoder Neural Networks

The auto-encoder neural networks encodes the input to pass through a compact hidden stage. The input is reconstructed (decoded) at the output stage and matched with original input to calculate the reconstruction error [91]:

$$f(z) = ||z - z_{recons}||^2 \tag{5.8}$$

5.4.2.2 Self-organizing map

Self-organizing maps is a clustering technique where high dimension input space is mapped to low dimension output clusters. The self-organizing map has input and output layers which connect with each other through a competitive learning network. The output layer contains all clusters. An unseen example is placed at the output layer and the distance from the closest cluster center is calculated. The distance is reconstruction error defined by [91]:

$$f(z) = min_k ||z - \mu_k||^2 \tag{5.9}$$

where k is number of output clusters.

5.4.2.3 Principal Component Analysis

The Principal component analysis (PCA) is dimension reduction technique which maps the input space of size N to output space M such that $M < N$. The projection of input object z from N to M is defined by:

$$y = WW^T z \tag{5.10}$$

where W is $k \times d$ matrix storing eigenvector for output space, $d = N$.

PCA use the reconstruction error as a classification function. The reconstruction error is defined as the squared distance between the projection and the original example:

$$f(z) = ||z - y||^2 \tag{5.11}$$

5.4.3 Boundary Methods

Boundary methods construct a boundary around the target class examples such that most of the target examples lie within the boundary. These methods use a threshold value for acceptance of outliers within the boundary. An unseen example z belongs to target class if it lies within the boundary. We have used ν-Support Vector Machine(SVM) [99] and Support Vector Data Descriptors [91] for intrusion detection in flows records.

5.4.3.1 ν-SVM

The ν-SVM construct a boundary around the target class examples in the form of a hyperplane during training [99]. An unseen example z belong to target class if it falls within the hyperplane and considered outlier otherwise.

SVM uses a feature mapping function $\phi : X \rightarrow H$ which maps the input feature space X to a high dimension feature space H [100]. The similarity between an input x and its class prediction y in feature space H is be calculated using a simple kernel function:

$$K(x, y) = (\phi(x).\phi(y))_H \tag{5.12}$$

To separate the input examples from the origin with maximum margin using a hyperplane, following quadratic minimizing function is applied:

$$min_{w,\xi,\rho} \frac{1}{2} \|w\|^2 + \frac{1}{m\nu} \sum_{i=1}^{m} \xi_i - \rho \tag{5.13}$$

Subject to

$$(w.\phi(x_i)) \geq \rho - \xi_i, \xi_i \geq 0 \tag{5.14}$$

where

ξ_i : Slack variable to penalize the outliers

$\nu \in (0, 1)$: A user-defined error control parameter and sets an upper bound on the fraction of outliers and a lower bound on the number of support vectors.

ρ : The maximal margin for hyperplane from origin.

Using Lagrange multipliers and constructing a Lagrangian, the decision function for the classification of a test example z is defined as follows [99]:

$$f(z) = sgn(\sum_{i} \alpha_i k(z_i, z) - \rho) \tag{5.15}$$

5.4.3.2 Support Vector Data Descriptor(SVDD)

The support vector data descriptors construct a hypersphere around the target class training examples. The sphere has its center a and with radius $R > 0$. The volume of the sphere is minimized

such that it contains all training examples [91].

Following error function is minimized for SVDD:

$$F(R, a) = R^2 \tag{5.16}$$

Such that all target examples x_i lies within the sphere:

$$||x_i - a||^2 \leq R^2 \tag{5.17}$$

The strict data description of minimizing the sphere radius may not fit in all cases. To allow the possibility of outliers in the sphere, and to penalize the larger distances for x_i, slack variables are introduced. Thus, the minimization problem becomes:

$$F(R, a, \xi) = R^2 + C \sum_i \xi_i \tag{5.18}$$

Such that maximum target examples x_i lies within the sphere:

$$||x_i - a||^2 \leq R^2 + \xi_i, \xi_i \geq 0 \tag{5.19}$$

The C parameter is analogues to ν in ν-SVM and control the acceptable number of outliers.The parameters R, a, and ξ are optimized by using Lagrange multipliers and constructing a Lagrangian. We have following quadratic minimization problem [91]:

$$L(R, a, \xi, \alpha, \gamma) = \sum_i \alpha_i x_i . x_i - \sum_{i,j} \alpha_i \alpha_j . x_j \tag{5.20}$$

A test example z is classified into target class if its distance from center a is less than the radius of the sphere. The decision function for classification is defined as follows:

$$f(z) = \begin{cases} 1, & \text{if } ||z - a||^2 \leq R^2 \\ 0, & \text{otherwise} \end{cases} \tag{5.21}$$

Table 5.1: Features for flow records used for evaluation of one-class classification techniques

Feature	Description
Packets	Number of packets in flow
Octets	Number of bytes in flow
Duration	The duration of flow in milliseconds
Source Port	Source port number
Destination Port	Destination port number
TCP Flags	Cumulative OR of TCP flags
Protocol	The transport layer protocol

5.5 Experimental Results

The one-class classifier uses malicious flows for training. When the classifier is trained, it processes traffic containing both normal and malicious flows. The classifier separates malicious flows from normal network traffic. It forwards malicious flows to the second stage of intrusion detection for detail analysis and discards normal flows. The second stage fetches packet payload for the malicious flows and uses payload based inspection for detail intrusion detection. Detail intrusion detection can involve exploring of types attacks and victim services. The intrusion detection system raises an alert after the second stage detection process is completed. We consider various one-classification methods including density estimation, reconstruction methods and boundary methods for detection of malicious flows in first stage. We have evaluated all techniques on a realistic flow-based dataset and obtained results to determine the suitability of one-class classification for detection of malicious flows.

5.5.1 Dataset

We have used UoT flow-based dataset for evaluation of available one-class classification techniques [28]. The dataset contains a limited number of attacks. We have mixed flows of two malware, Sality and Asprox in the dataset to increase the type of attacks. Sality and Asprox are well-known malware and affected a large number of system. We have generated traffic flows for Sality and Asprox malware using the packet capture files available at Contagio Malware dump[1].

We have converted all four time attributes, T_{start}, M_{start}, T_{end} and M_{end} to a single value of duration

[1] http://contagiodump.blogspot.com/

Table 5.2: Malicious flow records in training data-set

Packets	Bytes	Duration(in msec)	Source Port	Dest. Port	TCP Flags	Protocol	Class
5.0	736.0	33.0	80.0	1413.0	43.0	6.0	Malicious
5.0	739.0	33.0	80.0	4416.0	43.0	6.0	Malicious
12.0	2244.0	2052.0	22.0	1555.0	27.0	6.0	Malicious
12.0	1152.0	3972.0	1084.0	22.0	27.0	6.0	Malicious
5.0	721.0	20.0	80.0	3339.0	43.0	6.0	Malicious
14.0	1969.0	2112.0	22.0	4165.0	27.0	6.0	Malicious
5.0	722.0	20.0	80.0	4626.0	43.0	6.0	Malicious
16.0	1739.0	4297.0	22.0	2042.0	27.0	6.0	Malicious
5.0	729.0	32.0	80.0	1296.0	43.0	6.0	Malicious
12.0	1152.0	2184.0	3316.0	22.0	27.0	6.0	Malicious

in milliseconds. Detail of flow attributes is given in Table 5.1

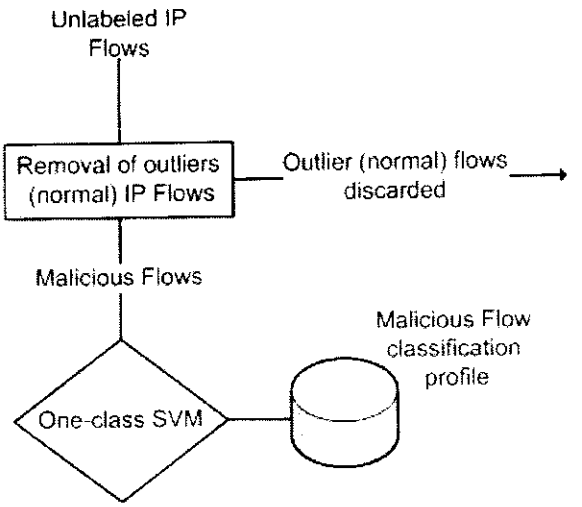


Figure 5.1: One-class SVM training

We have randomly sampled flows records for both normal and malicious classes. The training dataset contains 5000 malicious flow records because one-class classifiers use examples of the target class (malicious) for training 5.1. The test dataset contains 11421 records which include both normal and malicious flows. Table 5.2 and 5.3 shows sample of malicious and normal flows respectively. Table 5.4 gives detail of training and testing datasets.

Table 5.3: Normal flow records in test dataset

Packets	Bytes	Duration(msec)	Source Port	Destination Port	TCP Flags	Protocol
8.0	410.0	8819.0	3994.0	6667.0	24.0	6.0
21.0	1176.0	22.0	0.0	2816.0	0.0	1.0
34.0	1904.0	3004.0	0.0	2816.0	0.0	1.0
2.0	113.0	43.0	4416.0	6667.0	24.0	6.0
1.0	118.0	0.0	6667.0	4416.0	24.0	6.0
1.0	122.0	0.0	6667.0	3994.0	24.0	6.0
1.0	40.0	0.0	3994.0	6667.0	16.0	6.0
3.0	188.0	551.0	6667.0	1546.0	24.0	6.0
3.0	235.0	274.0	1546.0	6667.0	24.0	6.0
2.0	140.0	0.0	0.0	769.0	0.0	1.0

Table 5.4: Flow-based dataset for one-class classification

Training dataset		**Testing dataset**	
Malicious	Normal	Malicious	Normal
5000	0	2761	8660

5.5.2 Results

The one-class classification techniques are evaluated using two well-known performance measures; Area under Received Operator Characteristic (*ROC*) curve (AUC) and F1 score [5]. We have used Matlab, Weka, DDtools [101] and LibSVM [102] for performing the experiment. Table 5.5 gives detail results for various one-class classification techniques using multiple performance measures. During evaluation of density estimation methods, simple Gaussian gives a value of 0.6653 and 0.4310 for AUC and F1 score respectively. The ROC curve of simple Gaussian is shown in Figure 5.2. Mixture of Gaussian shows little improvement with AUC of 0.8374 (Figure 5.3) and F1 score of 0.4294. The Parzen density estimate has AUC of 0.6341 (Figure 5.4) and F1 score of 0.4283.

Reconstruction methods give average results for one-class classification. The auto-encoder neural network shows better performance with AUC of 0.9126 (Figure 5.5) and F1 score of 0.8208. The AUC for self-organizing map (SOM) is 0.5443 (Figure 5.6) and F1 score is 0.4146. The Principle component analysis (PCA) has AUC of 0.9258 (Figure 5.7) and F1 score of 0.4247.

One-class classification using boundary methods gives best results in detection of malicious Sflows.

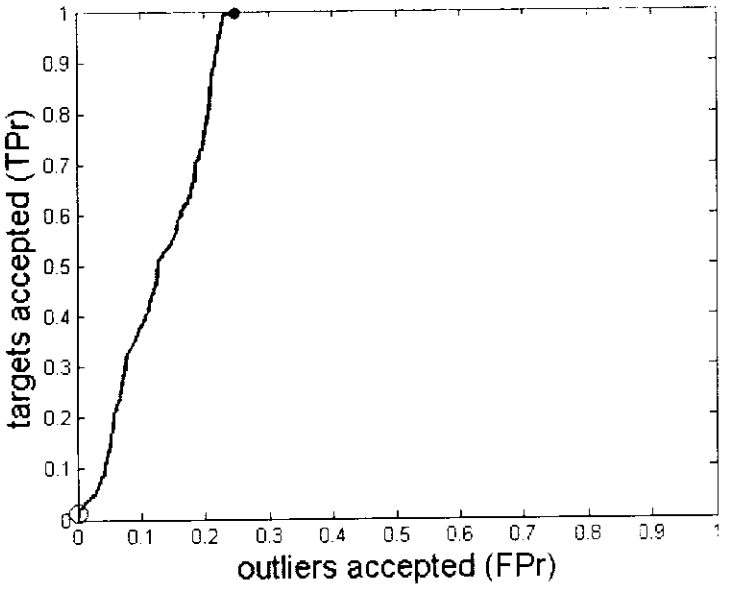


Figure 5.2: ROC curve for Simple Gaussian

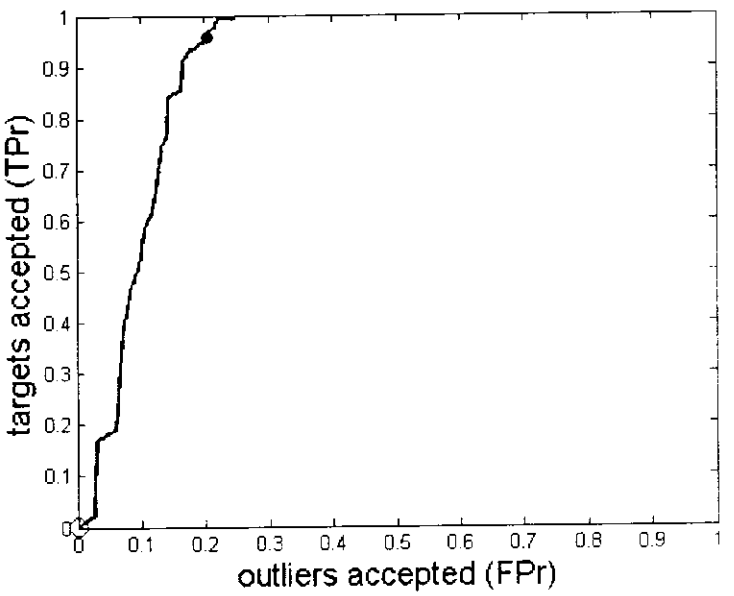


Figure 5.3: ROC curve for Mixture of Gaussian

The ν-SVM has AUC of 0.9297 (Figure 5.8) and F1 score of 0.9114. The support vector data descriptor (SVDD) also has similar results with AUC of 0.9335 (Figure 5.9) and F1 score of 0.9102.

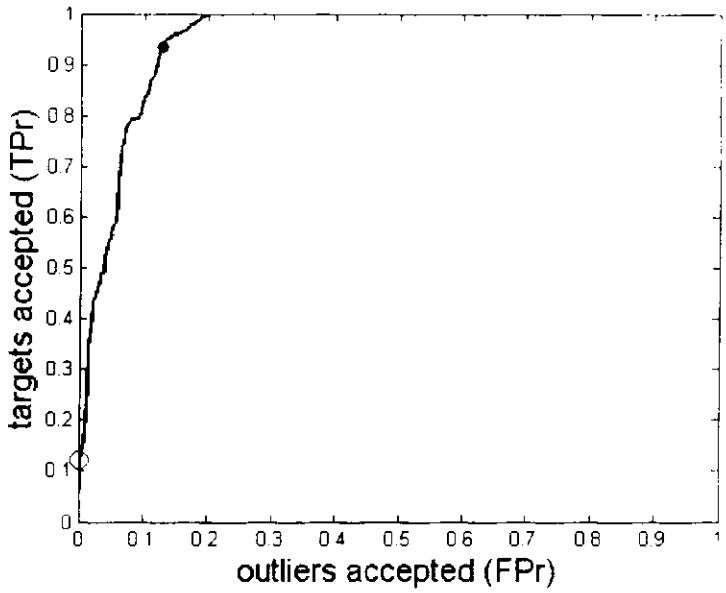


Figure 5.4: ROC curve for Parzen Distribution

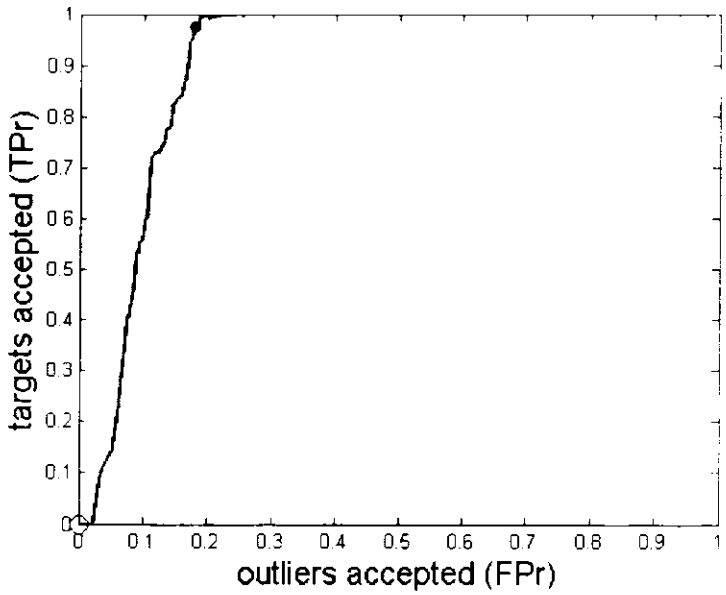


Figure 5.5: ROC curve for Auto-encoder Neural Network

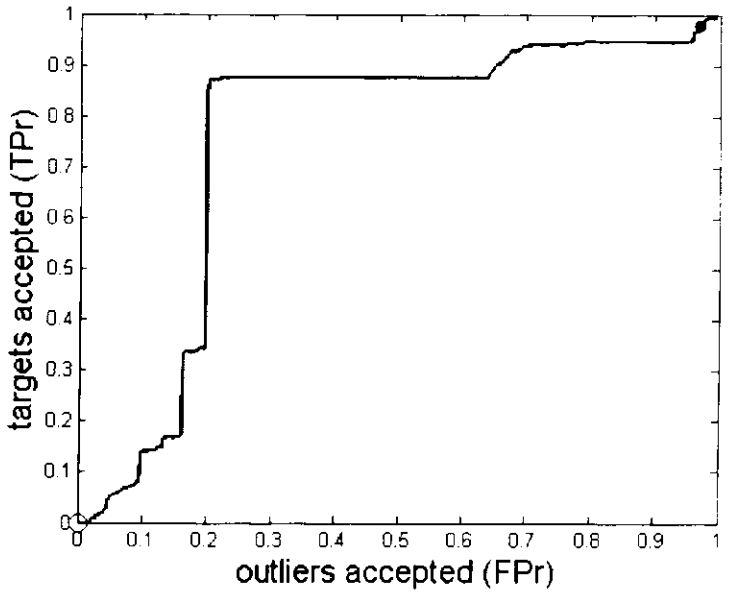


Figure 5.6: ROC curve for Self-organizing Maps

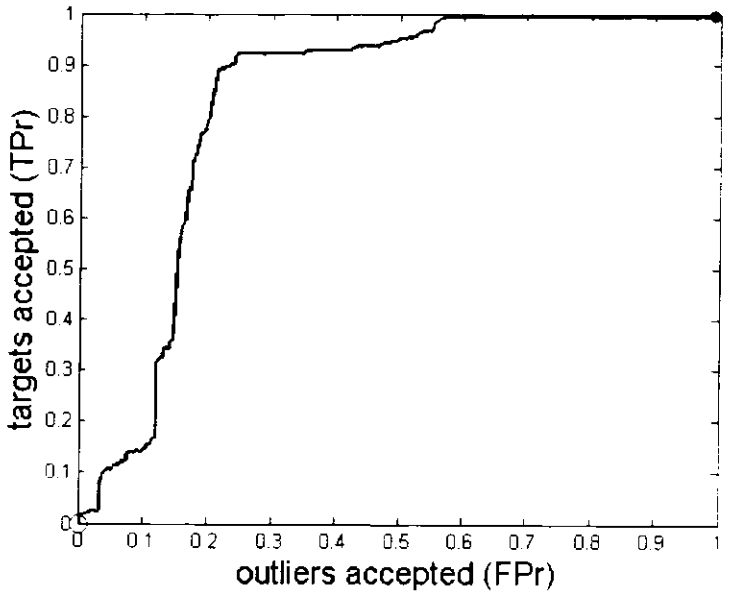


Figure 5.7: ROC curves for Principle Component Analysis

5.5.3 Discussion

In the first step, we have evaluated one-class classifiers techniques based on density estimation methods. The performance of density methods depends on the correct estimation of the density

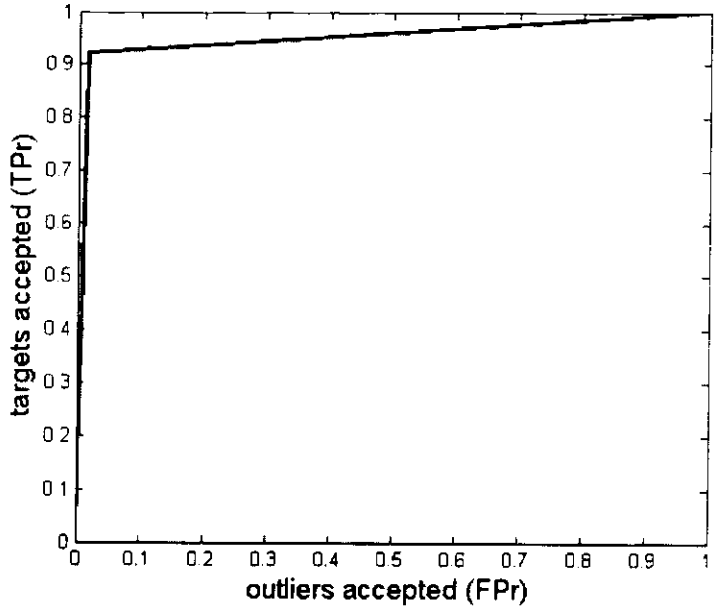


Figure 5.8: ROC curve for ν-SVM

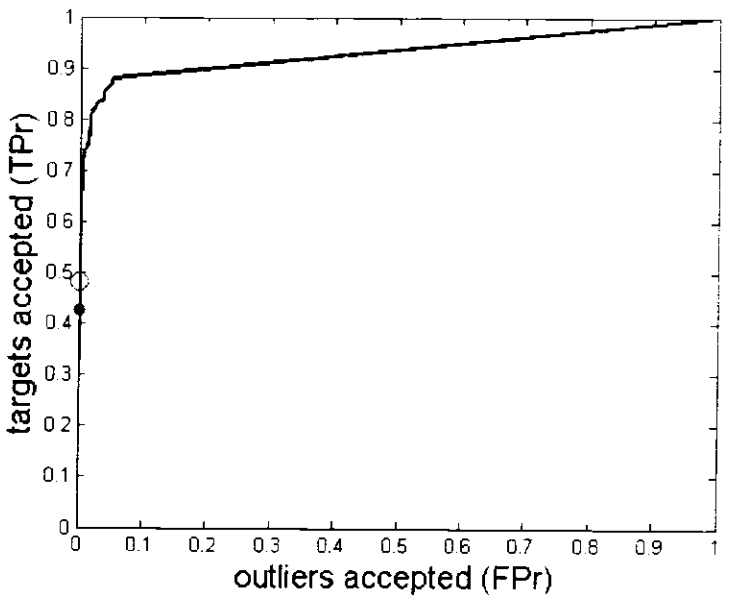


Figure 5.9: ROC curve for Support Vector Data Descriptor

function. These techniques give good results if the data is well-sampled and a large training dataset is available. However, the density functions do not give accurate results if data is sparsely plot-

Table 5.5: Results for one-class classification of malicious flows

One-class Classification Technique	Area under ROC Curve(AUC)	Precision	Recall	F1 score
Density Estimation				
Simple Gaussian	0.6653	0.2761	0.9826	0.4310
Mixture of Gaussian	0.8374	0.2789	0.9326	0.4294
Parzen density estimation	0.6341	0.2794	0.9178	0.4283
Reconstruction Methods				
Auto-encoder Neural Network	0.9126	0.7374	0.9239	0.8201
Self Organizing Maps	0.5443	0.2772	0.9362	0.4146
Principle Component Analysis	0.9258	0.1756	0.9001	0.4247
Boundary Methods				
ν-SVM	0.9297	0.9103	0.9125	0.9114
Support Vector Data Descriptor (SVDD)	0.9335	0.9258	0.8953	0.9102

ted in the feature space. Due to this limitation, these techniques do not have good performance for one-class classification. Our results also show that density based approaches have lower performance for one-class classification of malicious flows. The simple Gaussian method has lower accuracy because flow records and other data distributions do not fit in the bell curve of Gaussian representation. The constraint of a single bell curve is relaxed by using a mixture of Gaussian distributions. The mixture of Gaussian combine multiple Gaussian representations and also shows some improvement in results. However, still the mixture of Gaussian model is not an accurate representation of flows traffic distribution. The Parzen density method shows improved performance among all density estimation techniques. Parzen density estimation requires all training data to be made available during testing [97]. This is particularly challenging in the case of intrusion detection in network traffic where large training records are pushed on regular intervals.

In next step, the one-class classifier based on reconstruction methods have been evaluated. The reconstruction methods use a generating process to model the data. The parameters of the generating process are optimized for the correct representation of new objects. Although reconstruction methods can be applied for one-class classification problems, they are not primarily meant for this purpose [91, 103]. In case the data does not fit the model, a bias value is used to minimize the reconstruction error. The bias value destroys the important characteristics of the dataset [91]. Accurate modeling of flow records using reconstruction methods requires a large number of parameter to be optimized and can have high reconstruction error. These methods are computationally expensive for one class classification [97]. Another drawback of reconstruction methods is the difficulty

in training high-dimensional spaces [98]. This aspect limit the use of reconstruction methods for classification of flows because flow records can be very high dimensional as IPFIX includes around 280 attributes to define an flow record.

In next experiment, we have applied two neural network approaches which include auto-encoder neural network and self-organizing map. Results show that auto-encoder neural network has relatively good accuracy, but it requires some parameters to be specified by the user. These include the number of hidden layers, input units, and stopping criteria [97]. The auto-encoder neural network also needs a large training set for correct estimation of weights associated with hidden and input units. The self-organizing map (SOM) requires the user to specify the number of output clusters. Also SOM needs k^d neurons for d dimension dataset. It is computational expensive to estimate the weights of k^d neurons if both k and d are moderately higher [91]. Another type of reconstruction method is Principle component analysis (PCA). PCA also has lower accuracy for one-class classification of flows. The PCA does not perform well if the data has variance in all feature direction. In this case, PCA is unable to reduce the dimensionality of data [91].

One-class classification techniques using boundary methods give best results for identification of malicious flows. The boundary methods are specifically focused on one-class classification and also perform better if limited training sample is available [91]. These methods avoid the estimation of the complete probability density and can work efficiently if the dataset is high dimensional. In our experiment, we have used two SVM-based one class classification techniques; ν-SVM and Support Vector Data Descriptor (SVDD). The ν-SVM and SVDD use hyperplane and hypersphere respectively to enclose the target class examples. Finding the smallest sphere containing all target points is equivalent to find the segment containing the required points [99]. In our experiment, both methods show similar result and perform better than all reviewed classification methods. However, a drawback of SVM-based methods is the difficulty involve in the estimation of values for the parameter that controls the number of outliers within the boundary. The accuracy of SVM-based one-class classifiers is very sensitive to a slight change in the value of these parameters.

5.6 Chapter Summary

In this chapter, we evaluated available one-class classification techniques for detection of malicious flows. The one-class classification techniques include density estimation, reconstruction methods, and boundary methods. We used a flow-based dataset to train the one-class classifiers on malicious

flows and evaluated trained classifier on a test dataset containing both normal and malicious flow records. We have used multiple performance measures including Area under ROC Curve (*AUC*) and F1 score for comparison of results. On the basis of results, we discussed pros and cons of all techniques used for one-class classification. The results show that boundary methods i.e. SVM-based one-class classifiers give higher accuracy in identification of malicious flows. Therefore, we consider SVM-based one-classification techniques suitable for detection of malicious flows on the basis of experimental results. In next chapter, we will use unsupervised clustering techniques to classify malicious flows into different attack categories and provide insight into the malicious traffic.

Chapter 6

Clustering of malicious flows using unsupervised learning

6.1 Overview

Anomaly based intrusion detection systems, using either payload or flow level inspection, only classify network traffic as normal or malicious [104]. The intrusion detection system raises an alert when an anomaly is detected. However, the intrusion detection system does not provide additional information about the type of attack class. All these alerts are manually inspected to establish the validity of alert and to determine the attack type. On the basis of this information, the security administrator takes steps for any corrective action. Computer networks are a target of a wide range of attacks and every such attack requires different mitigation strategy [105]. Intrusion detection system raises a large number of alerts [106]. In the absence of an automatic attack clustering system, it is difficult for a security administrator to manually inspect every alert and employ countermeasures. This affects the overall usability of anomaly based intrusion detection system [104]. Therefore, structural organization of malicious traffic is required to reduce the number of alerts. A straight forward solution is to group similar malicious traffic together. A single alert can be generated for every set of malicious traffic. As a result, the overall number of alerts requiring manual inspection are reduced.

[104] has already proposed an automatic attack classification schemes for anomaly based intrusion detection systems. However, [104]'s technique used payload inspection and supervised learning

algorithms. In this chapter, we propose an automatic attack clustering scheme for flow-based intrusion detection systems using unsupervised learning. Automatic clustering of malicious flows using unsupervised learning is a novel idea. Our scheme receives malicious flows from a flow-based intrusion detection system. We group similar malicious flows into different attack clusters. The flow-based intrusion detection system raises a consolidated alert for every set of clustered flows instead of raising an alert for every malicious flow. We have used unsupervised learning for clustering and no labeled training set is required. Three unsupervised learning techniques k-means, self-organizing maps (SOM) and DBSCAN are considered. We have evaluated our technique on a realistic flow-based dataset. Experimental results show that our technique identifies all attack types in malicious IP and classify the majority of flows in the correct attack clusters.

6.2 Previous work

Numerous studies related to attack classification and alert management are presented in the literature. [104] proposed an automatic attack clustering technique, named Panacea, for packet-based intrusion detection systems. Panacea consists of two modules; Alert Information Extractor (AIE) and Attack Clustering Engine (ACE). The AIE processes alerts received from the intrusion detection system. It extracts byte sequences from the alert packet payload. This information is sent to Attack Clustering Engine (ACE) for clustering of attack packet. The clustering process has two main stages. First, the ACE is trained with several types of alert byte sequences to build a clustering model. The attack class information for training byte sequences is provided in manually an operator or automatically. After the training is completed, the ACE classify all incoming alerts automatically. The clustering engine uses Support Vector Machine (SVM) and RIPPER rule learner to group similar attack alerts together. Panacea is evaluated on alerts generated by Snort and promising results are obtained.

[107] have applied machine learning techniques for automatic clustering of malware. The proposed framework uses prototype-based clustering and clustering algorithms to identify novel classes of malware and assigns unknown malware to the discovered classes. In the first step, the framework monitors malware behavior in a sandbox environment and generates a report of malware behavior. The malware reports are converted into vector space model. Machine learning clustering and clustering techniques are applied to the vector data for identification of novel and known malware classes. Using both, clustering and classification, the behavior of malware can be analyzed incrementally reducing the run-time overhead. Two data sets of malware behavior have been used

to evaluate the proposed framework. The first dataset consist of 24 known malware classes with 3133 reports of malware behavior. The second dataset contains 33,698 behavior reports with an unknown number of malware classes. Results show that clustering methods have F1-measure of 0.950, and classification clustering methods have F1-measure of 0.981 and 0.997 for known and unknown malware classes respectively.

An alert correlation technique using Fuzzy Logic and Artificial Immune System (AIS) is proposed by [108]. The system calculates the degree of correlation between two alerts and uses this knowledge to extract the attack scenarios. For each pair of input alerts, the system creates a feature vector which is matched with a set of Fuzzy rules. If it finds a matching rule with the value higher than the matching threshold, then simply it uses the probability in that rule as the correlation probability of two alerts. If no rule is matched with the required threshold, it uses the AIRS algorithm. AIRS algorithm is a supervised learning algorithm that uses a set of fuzzy rules as training set and discovers the relationships between the values of the features in the rules. The system can be converted to complete supervised learning system by setting the value of rule matching threshold to 1 and vice versa by setting it to 0. The system is evaluated on two datasets, DARPA 2000 and netForensics honeynet data. Results show that average completeness and false alert rate for LLDoS1.0 attacks are 0.957 and 0.053 respectively. Average completeness and false alert rate for LLDoS2.0 attacks are 0.745 and 0.245 respectively. An alert correlation technique using Fuzzy Logic and Artificial Immune System (AIS) is proposed by [108]. The system calculates the degree of correlation between two alerts and uses this knowledge to extract the attack scenarios. For each pair of input alerts, the system creates a feature vector which is matched with a set of Fuzzy rules. If it finds a matching rule with the value higher than the matching threshold, then simply it uses the probability in that rule as the correlation probability of two alerts. If no rule is matched with the required threshold, it uses the AIRS algorithm. AIRS algorithm is a supervised learning algorithm that uses a set of fuzzy rules as training set and discovers the relationships between the values of the features in the rules. The system can be converted to complete supervised learning system by setting the value of rule matching threshold to 1 and vice versa by setting it to 0. The system is evaluated on two datasets, DARPA 2000 and netForensics honeynet data. Results show that average completeness and false alert rate for LLDoS1.0 attacks are 0.957 and 0.053 respectively. Average completeness and false alert rate for LLDoS2.0 attacks are 0.745 and 0.245 respectively.

[106] proposed the use of expert knowledge in probabilistic Bayesian network classifiers for intrusion detection and alerts correlation. In this technique, a model for intrusion detection and alert correlation is learned using a training set. This model is used to determine malicious activities and

validity of attack alerts. The authors argued that using expert knowledge has benefits when training data is insufficient or generates a large number of false negatives. Expert knowledge can be provided by a domain expert or is extracted from the probability distributions of the instances over attack class. A revised algorithm is proposed to modify the output of a probabilistic classifier with the use of available expert knowledge. The proposed technique is evaluated using two real network traffic datasets and DARPA dataset. Results showed that performance of intrusion detection and alert correlation is improved with the use of expert knowledge.

[109] presents a comprehensive evaluation of DPI and flow-based techniques for clustering of malware in network traffic. In flow-based detection, several supervised machine learning techniques including J48, Boosted J48, Nave Bayesian, Boosted Nave Bayesian, and SVM are used to identify malicious flows. Comparison between different algorithm results shows that J48 and boosted J48 perform better than other algorithms. Authors obtained a detection rate of 99% and a false positive rate of less than 1% for J48 and boosted J48 algorithms. In addition to detection of maliciousness of traffic, authors have used flows records for attribution of flow records into different malware families. The attribution process uses uni-directional flows through an iterative process to build Hidden Markov Modes for every malware family. In deep packet inspection (DPI) based approach, machine learning techniques are applied on complete packet captures. NLP and wavelets clustering of signals techniques to fingerprint maliciousness. The DPI techniques performed well for less noisy malicious traffic.

A DDoS detection method based on the online processing of traffic time series using higher order F-transform is proposed in [110]. This approach extracts the relationship between traffic data to identify traffic patterns and detection of anomalies. The technique also determines the intensity of DDoS attack. A simulated dataset was used for evaluation which contained nine-classes with three classes for normal, slow and rapid attack categories. The proposed technique achieves a success rate of 100% and is also 166 times faster than naive approaches.

In [111], the authors proposes a web attack clustering technique using the vector space model approach (VSMA). The vector space model converts a text document into real valued vectors based on the term frequencies [112]. Authors have converted HTTP requests into vector space data. The VSMA classifier has been evaluated on a dataset containing 250 malicious attacks including Redirecting URLs, script Injection, and XSS attacks. The VSMA classifier obtained 98% accuracy as compared to Naive bayes and Decision tree. Other approaches where authors have tried to determine the attack types are [113] and [114].

Review of literature shows that existing techniques suffer from a number of weaknesses such as use of payload inspection and requirement of labeled training set. Also the evaluation is carried out using non-representative old datasets (e.g. DARPA) which are not relevant today. Our proposed technique differs from the existing work in a number of aspects. Instead of using an alert correlation or management system, we propose an additional internal component for intrusion detection system that automatically creates attack clusters of malicious traffic. Our technique is based on flow-based intrusion detection model therefore no payload inspection is performed. Also we use unsupervised learning algorithms for attack clustering therefore no labeled training dataset is required. We have evaluated our technique on a realistic flow-based dataset combined with flows of two well-known malware.

6.3 Unsupervised learning techniques

Unsupervised learning is an important feature of machine learning. Unsupervised learning techniques differ from supervised learning in the way that unsupervised learning does not require a labeled training set [115]. Unsupervised learning techniques derive their results using the statistical properties of data. Unsupervised learning techniques group a set of data examples in same dimension if they have similar statistical properties. Unsupervised learning has numerous application in machine learning because it is not always possible to obtain a labeled dataset for the training of algorithm [116]. In the case of intrusion detection, availability of labeled training set is also difficult. Use of unsupervised learning in intrusion detection has several advantages over supervised learning [117].

Unsupervised learning techniques are divided into two categories; dimensionality reduction and clustering [118]. Dimensionality reduction techniques map a set of examples from a high dimension feature space to low dimension feature space. The new features in low dimension space not necessarily the same as those of original features in high dimension space. This process obtains a compact representation of data which is easy to process and analyze. Clustering techniques partition the data into different clusters such that distance between different clusters is maximized and the distance between a cluster boundary and its center is minimized. Clustering techniques discover new patterns in data which are not previously known.

Formally, clustering of input data X is defined as a set of clusters $C = \{c_1 \ldots . c_k\}$ such that $X = \cup_{i=1}^{k}$. Every example in the dataset belongs to only one cluster where $C_i \cap C_j = \emptyset$ for $i \neq j$

[119]. Clustering techniques use a distance function (Euclidean distance, Manhattan distance) to calculate the similarity between different examples

Main types of clustering algorithms include partitional, model and density based techniques [120]. Partitional clustering divides the data into different partitions where each partition represent a cluster [121]. Partitional clustering algorithms include k-means, k-medoids. Model base clustering technique use a particular model for every cluster. An example becomes a part of the cluster if it fits the model. Model clustering algorithms include self-organizing map (SOM) and adaptive resonance theory (ART). Density based clustering techniques mark the areas with higher densities as clusters while lower densities areas are considered noise. Density based clustering techniques include density-based spatial clustering of applications with noise (DBSCAN), ordering points to identify the clustering structure (OPTICS) and Mean-Shift [120]. We have considered three unsupervised learning algorithms: k-means, SOM and DBSCAN, for clustering of malicious flows.

6.3.1 K-means

K-means is an unsupervised learning algorithms based on partitional clustering technique. The k-means, randomly choose centroid vectors for k clusters in the data. In next step, the algorithm calculates squared distance between all data examples and k centroids vectors and assigns all them to their nearest cluster. The algorithm repeats the process and calculates k centroid vectors on the basis of new cluster membership. Again the membership of all example is calculated for new k centroid vectors. The process is repeated until no example moves the centroid vectors [122]. The k-mean algorithm minimize the following objective function:

$$\sum_{i=1}^{k}\sum_{j=1}^{n}||x_j - i|| \tag{6.1}$$

where k defines the total number of output clusters, i is the ith cluster center in k, n is the total number of examples, and j is the jth example in the n.

6.3.2 Self-organizing map

The self-organizing map (SOM) is an unsupervised artificial neural network[123]. The SOM network consists of a number of neuron units connected in multiple layers. SOM apply a particular

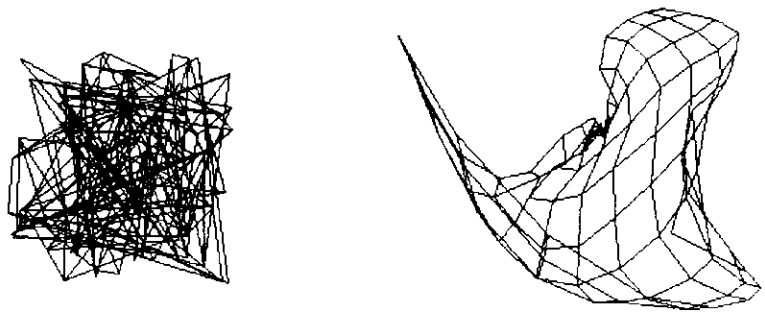


Figure 6.1: Self-organizing map learning process

type of unsupervised system called competitive learning. In competitive learning, the output neuron compete with each other and one neuron is active at a time. This approach is also called winner-take-all where the winning neuron supersedes all competing neurons. The competitive learning approach has three steps, competition, cooperation and adaptation.

Mathematically, Let $X = (x_1, x_2, ... x_l)$ is a set of n-dimensional input vectors which we want to classify in k different clusters. The SOM network for this clustering consists of k output neurons, and every neuron has n inputs. In competition, k neurons compete with each other to win an n-dimension input vector. Each competing neuron has a weight vector associated with it. The value of the weight vector is initialized to random small values. When an input vector $x \ni X$, is presented to the SOM network, it is compared with weight vectors of all competing neurons using following equation:

$$v = argmin_k ||w_k - x|| \tag{6.2}$$

A neuron v is declared winner if its weight vector is closest to the input example x. Winning neuron is also called BMU (best matching unit). In cooperation, the neurons closer to the BMU get excited more than distant neurons. This topological ordering is controlled using a neighborhood function. The effect of neighborhood function is maximum at the origin and decrease with distance and time. The range of the neighborhood is defined by a Gaussian function:

$$\sigma(t) = \sigma_0 e^{(-2\sigma_0 \frac{t}{t_m})} \tag{6.3}$$

where

σ_0 = Initial value of neighborhood range

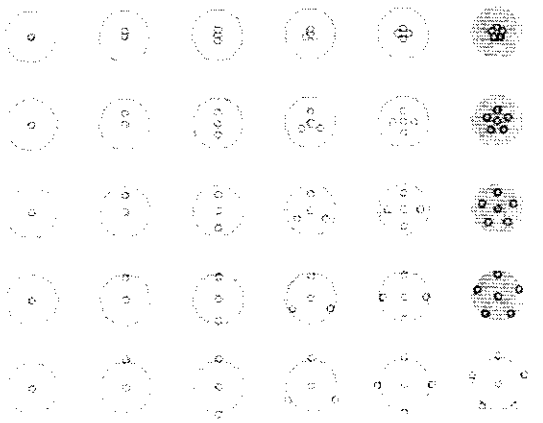


Figure 6.2: DBSCAN Clustering with minPoints = 6

t and t_m= The current and maximum iteration respectively
$\sigma(t)$ = The range of neighborhood at t stage.

The last step of adaptation defines a learning process and adjusts weights of neighboring neuron vectors to form a topographic self-organizing map:-

$$w_i(t+1) = w_i(t) + \eta(t)\sigma(t)(v - w_i(t)) \tag{6.4}$$

t and $\eta(t)$ represent the current iteration and learning rate. The effect of each learning iteration is to move the weight vectors w_i of the winning neuron and its neighboring neurons closer towards the input vector x_i. The continuous process of competition, cooperating and adaptation marks the cluster on the output layer as shown in Figure 6.1.

6.3.3 DBSCAN

DBSCAN is a widely used density based clustering algorithm [124]. DBSCAN clusters the points together which are located in high density areas. The points situated in the less dense areas are considered outliers or noise.

The DBSCAN clustering algorithm is based on two important input parameters: Epsilon(eps) neighborhood and Minimum number of points (minPoints). Epsilon-neighborhood of a point p contains all points such that the distance between p and all point q is less than the Epsilon. If the

number of points within the eps-neighborhood of p is greater than or equal to minPoints, then p is called a core point. All points within p's eps-neighborhood are directly density reachable from p if p is a core point. A point p is density reachable from the point q if there is a chain of objects $p_1, ..., p_n, p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i[125]. A point p is a border point if it is not a core point but it is density-reachable from another core point. The DBSCAN starts with the first point p in data and creates a cluster of all points which are directly density reachable or density reachable from point p. The same process is repeated for all points in the data. A point is marked outlier if it is not part of any cluster. A DBSCAN clustering with 6 minPoints is shown in Figure 6.2.

6.4 System Model

Intrusion detection systems classify network traffic into normal and malicious categories. In case of flow-based inspection, individual flows are classified as normal or malicious. These flows are in very large number and it is very difficult to inspect every individual flow manually. Our aim is to group malicious flows together into different attack clusters. Flow records consist of a number of attributes which show the general behavior of the traffic. All flow generated by single application generally follow a unique pattern. The key idea is that flows of an attack will also have similar values for flow attributes e.g. IP addresses, port numbers, protocol flags and packet sizes. Unsupervised learning algorithm uses the similarity in flow attributes for clustering of flows into different attack clusters.

The architecture of proposed approach is shown in Figure 6.3. The malicious flow clustering component is placed between the intrusion detection system and alert triggering component. An unlabeled training set of n malicious flows is extracted from the flow-based intrusion detection. These malicious flow records are passed through feature selection and normalization steps. The clustering algorithm creates different attack clusters using the unlabeled malicious flows in training set. We assign an attack label to every cluster. After the clustering and labeling process is complete, the attack clustering component compares every incoming flow with all clusters and assigns them the label of the closest cluster. The intrusion detection system raises a consolidated alert for a set of malicious flows having same attack label.

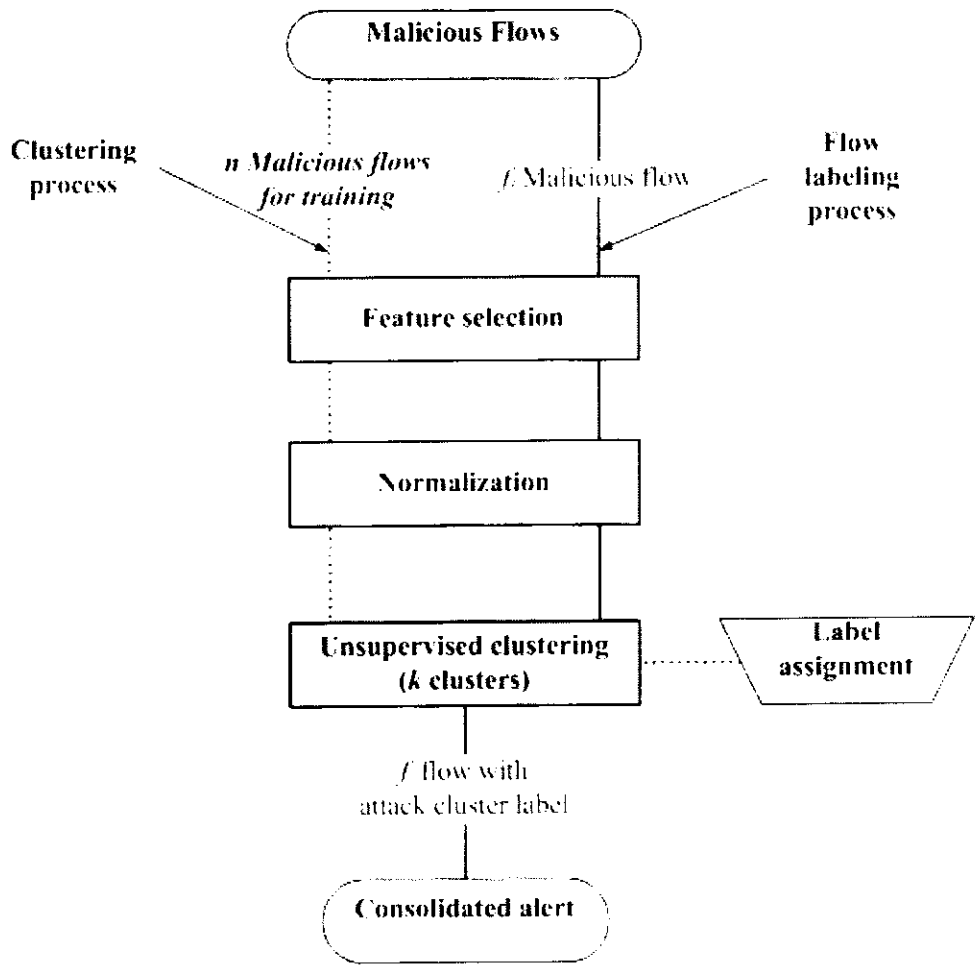


Figure 6.3: Automatic clustering of malicious flows

6.4.1 Flow collection

The attack classification component receives malicious flows from a flow-based intrusion detection system. The flow-based intrusion detection system separates malicious and normal flows. Normal flows are discarded while malicious flows are retained for further analysis. These malicious flows are forwarded to the attack clustering component. Initially, the flows are used to create attack clusters. After the clusters are created, all incoming flows are placed in different attack clusters. The flow-based intrusion detection system can use additional features computed using base flow attributes. However, these features may not be required for by the attack clustering component. The feature section step can add or remove additional features. In this paper, we have used standard Netflow v5 features[1].

We also normalize the flow records to control the variation in the flow data. Normalization process scale the values of flow attributes using a fixed range of values such that the larger value attribute cannot dominate smaller valued attributes. The normalization process only scales numeric attributes whereas nominal attributes are used as it. We have normalized flow data in the range of [-1,1] using the maximum and minimum value of a given attribute.

6.4.2 Flow clustering

In this step, we use a training set of unlabeled flows to create different attack clusters. The clustering algorithm process all flow records in the training set, until the clusters are converged.

We have used K-mean, SOM and DBSCAN algorithms for creation of clusters. Algorithm 1 outlines the process of k-mean clustering. The input to the algorithm is a set of unlabeled flows F and number of output clusters k. The algorithm normalizes the flow records and randomly select a flow for every cluster. This flow act as an initial cluster center for every cluster. In next step, membership of all examples is calculated and every example is assigned to the nearest cluster. In each iteration, the cluster centers are recalculated until the algorithm converges and cluster centers no longer change. The algorithm outputs a set of cluster centers C for k clusters.

Algorithm 2 shows the SOM clustering process. The input to the algorithm is a set of unlabeled flows F, number of output neurons(clusters) k, initial learning rate η and maximum number of iterations. The algorithm randomly initializes weight vectors for all neurons. In each iteration, a

[1]https://www.plixer.com/support/netflow_v5.html

wining neuron is selected for every flows and weight of its neighboring neurons are adjusted using a time decay function $\sigma(t)$. The algorithm outputs a set of cluster centers C for k clusters.

Algorithm 1 Clustering of flows using k-means

1: **procedure** CLUSTERFLOWKMEAN
   Input $F = \{f_1, f_2, f_3, ..., f_n\}$ set of malicious flows, k = number of output clusters
   Output $C = \{c_1, c_2, c_3, ..., c_k\}$ set of clusters
2: $F' = Normalize(F)$
3: **for** $c_i \in C$ **do**
4: $c_i = Random(F')$
5: **end for**
6: **repeat**
7: **for** $f' \ni F'$ **do**
8: $l_i = argmin_k ||c_k - f'||$
9: **for** k iterations **do**
10: $s_k = \{x_i | l[i] = k\}$
11: $c_k = \frac{\sum_{x_i \in s_k}}{s_k}$
12: **end for**
13: **end for**
14: **until** clusters are converged
15: **end procedure**

The DBSCAN clustering does not require the number of clusters to be set initially. However, two parameters eps and minPoints have to be set. Slight changes in these value affect the placement and number of clusters. Algorithm 3 describes the process of clustering using DBSCAN. The input to the algorithm is a set of unlabeled flows F, minPoints and eps value. DBSCAN processes all flows in F and creates a cluster for flow f enclosing all flows which are directly density reachable or density reachable from f. The algorithm outputs a set of cluster C.

6.4.3 Cluster and flow labeling

We label every cluster with the type of attack using domain knowledge of malicious traffic. The type of attack for every attack cluster is determined by analyzing flows at the center of every cluster. If the inspection of flows in an attack cluster does not reflect an attack type, the cluster number can be used as an attack label.

After the clusters are created, and labeling process is complete, the attack clustering component

Algorithm 2 Clustering of flows using Self-organizing maps

1: **procedure** CLUSTERFLOWSOM
   Input $F = \{f_1, f_2, f_3, ..., f_n\}$ set of malicious flows, k = number of output neurons (clusters), η = initial learning rate, max = maximum number of iterations
   Output $C = \{c_1, c_2, c_3, ..., c_k\}$ set of clusters
2: $F' = Normalize(F)$
3: Random initialize weight vector $W = \{w_1, w_2, w_3, ..., w_k\}$ for k neurons
4: **for** max iterations **do**
5: **for** $f' \ni F'$ **do**
6: Select wining neuron $c = argmin_k||w_k - f'||$
7: **for** i neurons in neighborhood **do**
8: $\sigma(t) = \sigma_0 e^{(-2\eta_0 \frac{t}{\eta_0})}$ /*Determine neighborhood range
9: $\eta(t-1) > \eta(t) > 0$
10: $w_i(t+1) = w_i(t) + \eta(t)\sigma(t)(-w_i(t))$ /*Update weights
11: **end for**
12: **end for**
13: **end for**
14: **end procedure**

processes all incoming flows. An incoming malicious flow f_i is compared with all clusters centroids. The f_i is assigned the attack label of cluster c, if the it is closest to f_i.

$$c = argmin_k||c_k - f_i|| \tag{6.5}$$

These labeled flows are forwarded to the alert triggering component of the intrusion detection system. The alert triggering component raise consolidated alerts for every set flows with same attack label. The intrusion detections system can be configured with the minimum and maximum number of flows required for a consolidated alert.

6.5 Experimental Results

In this section, we describe experimental results for clustering of malicious flows using k-means, self-organizing map (SOM) and DBSCAN algorithms on a flow-based dataset.

Algorithm 3 Clustering of flows using DBSCAN

1: **procedure** CLUSTERFLOWDBSCAN
    Input $F = \{f_1, f_2, f_3, ..., f_n\}$ set of malicious flows, eps = Epsilon value, $minFlows$ = minPoints value
    Output $C = \{c_1, c_2, c_3, ..., c_k\}$ set of clusters
2: Initialize i = 0
3: $F' = Normalize(F)$
4: **for** $f' \ni F'$ **do**
5: **if** f' is not visited **then**
6: Mark e' visited
7: **if** $epsNeigborhoodPoints(f') \geq minFlows$ **then**
8: $i = i + 1$
9: Assign f' to cluster c_i /* f' is a core point(flow)
10: **for** $e' \ni epsNeigborhoodPoints(f')$ **do**
11: **if** e' is not visited **then**
12: Mark e' visited
13: **if** $epsNeigborhoodPoints(e') >= minFlows$ **then**
14: Assign e' to cluster c_i if $e' \notin C$
15: **end if**
16: **end if**
17: **end for**
18: **end if**
19: **end if**
20: **end for**
21: **end procedure**
22: **procedure** EPSNEIGBORHOODPOINTS(p)
    output Returns the number of points in eps neighborhood of point p
23: **end procedure**

Table 6.1: Distribution of malicious flows in the dataset

Type	No. of Flows
HTTP Incoming traffic	2127
HTTP Outgoing traffic	2113
SSH Incoming traffic	4140
SSH Outgoing traffic	3360
Sality Incoming traffic	3952
Sality Outgoing traffic	2603
Asprox Outgoing traffic	5705
Total	24000

6.5.1 Dataset description

We have used Sperotto dataset for evaluation of unsupervised learning techniques [28]. We have extracted a random subset of the dataset for evaluation due to memory and computing limitations. To make the dataset more comprehensive, we have mixed flows generated by two well known malware; Sality[2] and Asprox[3]. Sality infects executable and DLL files. It also generates P2P botnet traffic and receives URLs of other malware and security risks for downloading. This virus then executes these malware on the infected computer system. Asprox malware uses SQL injections to spread itself and infect websites with malware containing redirects. Both malware have infected millions of computers worldwide.

We have obtained packet capture files for the malware from Contagio Malware Dump[4]. We converted packet capture files to uni-directional flows using nProbe[5]. Sality malware has both incoming and outgoing flows while Asprox malware dump has only outgoing flows. The resultant dataset contains malicious flows generated due to SSH brute force attack, HTTP server scanning and Sality malware. Distribution of malicious flows in dataset is given in Table 6.1.

The flows are uni-directional which means separate flow records are generated for incoming and outgoing traffic. Use of uni-directional flows is useful in identifying attacks classes because malicious incoming and outgoing traffic differs with each other [109].

[2]http://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/SALITY
[3]https://en.wikipedia.org/wiki/Asprox_botnet
[4]http://contagiodump.blogspot.com/
[5]http://www.ntop.org/products/netflow/nprobe/

Table 6.2: Malicious flows clustering results using K-mean

Attack cluster	Actual flows	Correctly clustered flows	Recall	Precision	F1-measures
HTTP Incoming	2127	1180	0.98	0.55	0.70
HTTP Outgoing	2113	2091	0.98	1.00	0.99
SSH Incoming	4140	3370	0.81	0.93	0.87
SSH Outgoing	3360	3360	1.00	0.78	0.88
Sality Incoming	3952	3400	0.86	0.66	0.75
Sality Outgoing	2603	960	0.36	0.41	0.39
Asprox Outgoing	5705	4503	0.78	0.99	0.87

6.5.2 Results

We have performed the experiments on Weka[6] using Weka's class to cluster evaluation mode. In this mode, Weka uses a labeled dataset to validate the clustering algorithm. In training phase, Weka ignores the class attribute and generates clusters from unlabeled dataset. In testing phase, every example in the dataset is compared with all clusters and assigned to the closest cluster. The class label attribute is used to check that if an example is assigned to correct cluster or otherwise. The class to cluster evaluation also determine additional performance measures.

6.5.2.0.1 Experiment I: K-means. In first experiment, we have used k-mean algorithm for clustering of malicious flows. The number of output clusters has been set to 8 because the dataset has 7 attack types. The 8th cluster is included to hold flows which cannot be clustered in any of the 7 clusters. Table 6.2 gives the results of k-mean clustering. K-mean's performance is best in clustering of HTTP outgoing flows with F1-measure of 0.99. It has average performance in clustering of HTTP incoming, SSH incoming, and SSH outgoing traffic with F1-measure of 0.70, 0.87, 0.88 and 0.75 respectively. The Sality incoming and Asprox outgoing clustering has F1-measure of 0.75 and 0.87 respectively. However, k-mean do not perform well in clustering of Sality outgoing traffic and gives F1-measure of 0.39.

6.5.2.0.2 Experiment II: Self-organizing map. We have used self-organizing map (SOM) in second experiment. The number of output clusters is set to 8. Table 6.3 gives the results for SOM clustering. The incoming and outgoing HTTP attack traffic is correctly clustered with F1-measure of 0.98 and 0.99 respectively. The incoming and outgoing SSH traffic has F1-measure of 0.70 and

[6]http://www.cs.waikato.ac.nz/ml/weka/

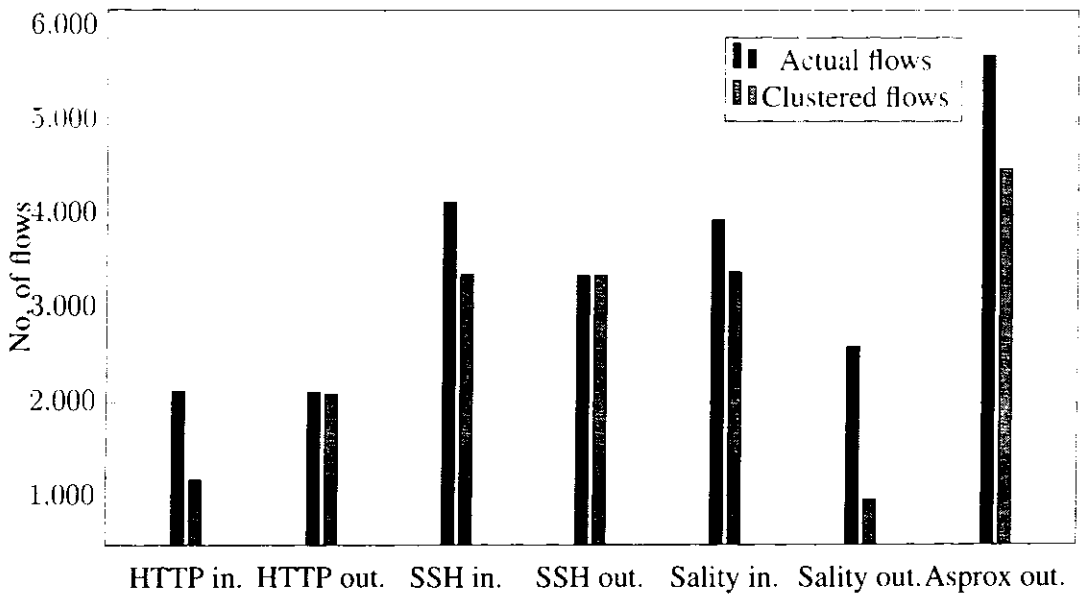


Figure 6.4: Comparison of flows in attack clusters with actual flows using k-means

0.98. SOM does not perform well in clustering of Sality and Asprox outgoing malware flows and gives F1-measure of 0.58 and 0.69 respectively. An additional benefit of SOM is generation of visual clustering map as shown in Figure 6.6.

6.5.2.0.3 Experiment III : DBSCAN. In third experiment, we have used DBSCAN algorithm for unsupervised clustering of malicious flows. DBSCAN requires two parameters to be set manually; epsilon and minPoint. The value of epsilon and minPoint is chosen empirically and set to 0.4 and 6 respectively. DBSCAN has an advantage over k-mean and SOM that it automatically

Table 6.3: Malicious flows clustering results using SOM

Attack cluster	Actual flows	Correctly clustered flows	Recall	Precision	F1-measures
HTTP Incoming	2127	2102	0.98	0.99	0.98
HTTP Outgoing	2113	2088	0.98	1.00	0.99
SSH Incoming	4140	3370	0.81	0.77	0.70
SSH Outgoing	3360	3360	1.00	0.96	0.98
Sality Incoming	3952	1879	0.86	0.76	0.80
Sality Outgoing	2603	961	0.63	0.55	0.58
Asprox Outgoing	5705	3036	0.53	0.99	0.69

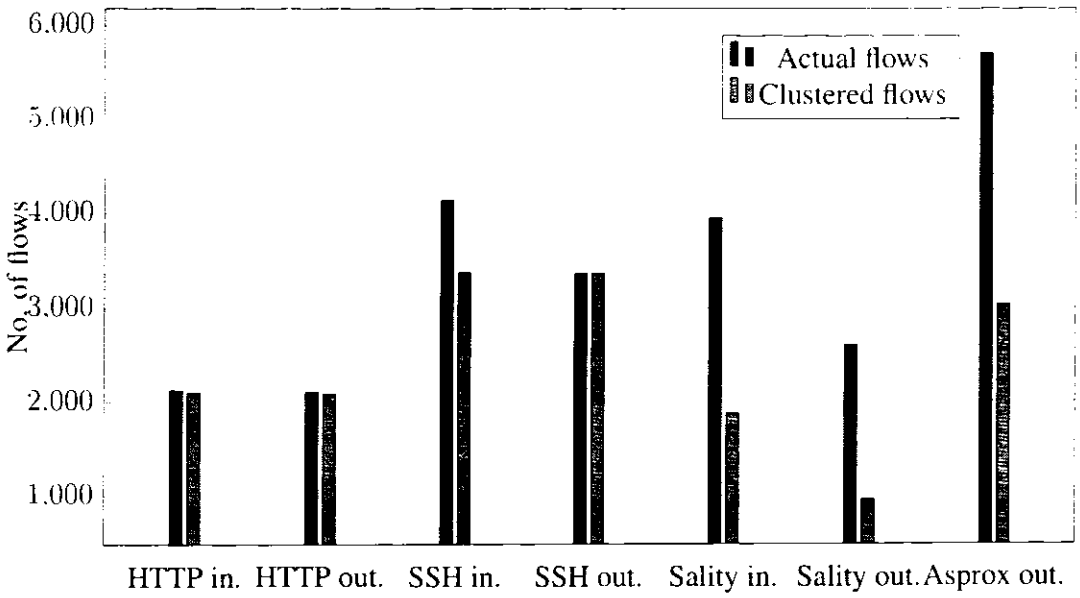


Figure 6.5: Comparison of flows in attack clusters with actual flows using SOM

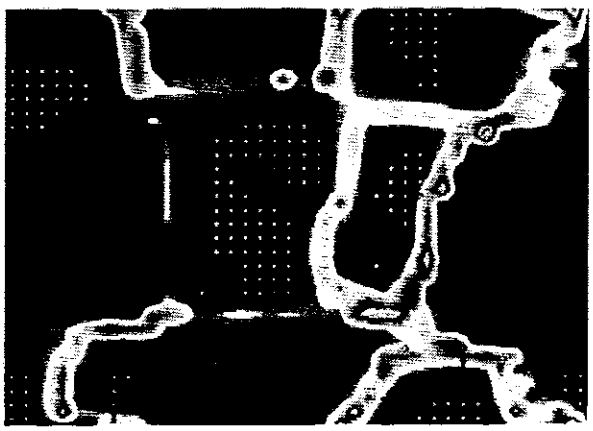


Figure 6.6: Self-organizing clustering map

Table 6.4: Malicious flows clustering results using DBSCAN

Attack cluster	Actual flows	Correctly clustered flows	Recall	Precision	F1-measures
HTTP Incoming	2127	2098	0.98	0.98	0.98
HTTP Outgoing	2113	2064	0.97	1.00	0.98
SSH Incoming	4140	3369	0.81	0.99	0.89
SSH Outgoing	3360	3360	1.00	1.00	1.00
Sality Incoming	3952	3407	0.86	0.78	0.82
Sality Outgoing	2603	1642	0.63	0.55	0.59
Asprox Outgoing	5705	5591	0.98	0.99	0.98

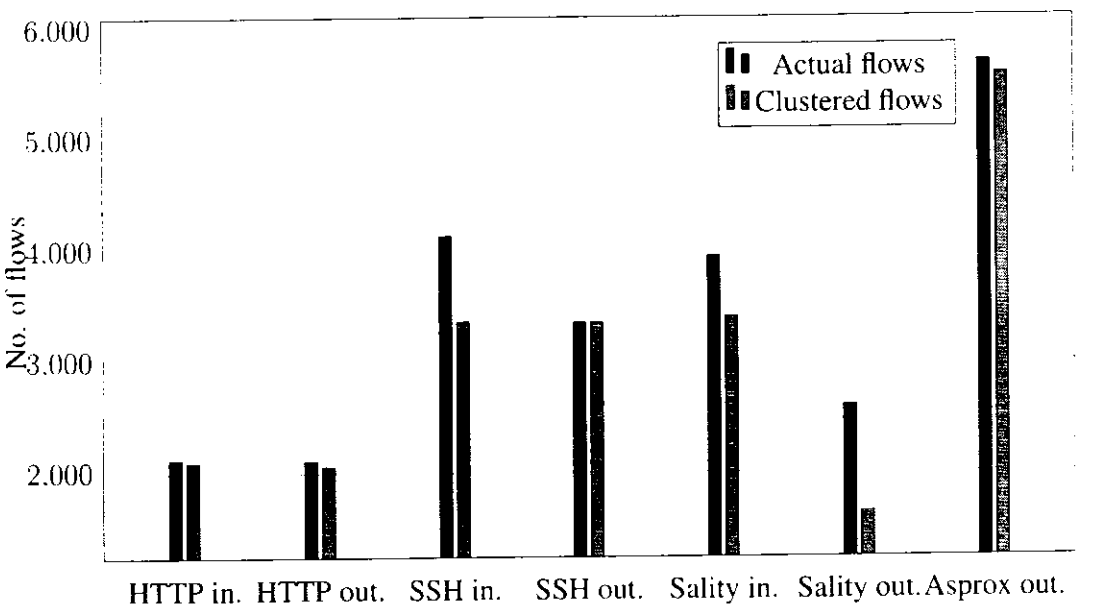


Figure 6.7: Comparison of flows in attack clusters with actual flows using DBSCAN

Table 6.5: Comparison of overall accuracy

Unsupervised learning algorithm	Overall accuracy
K-means	78%
Self-organizing map	80%
DBSCAN	89%

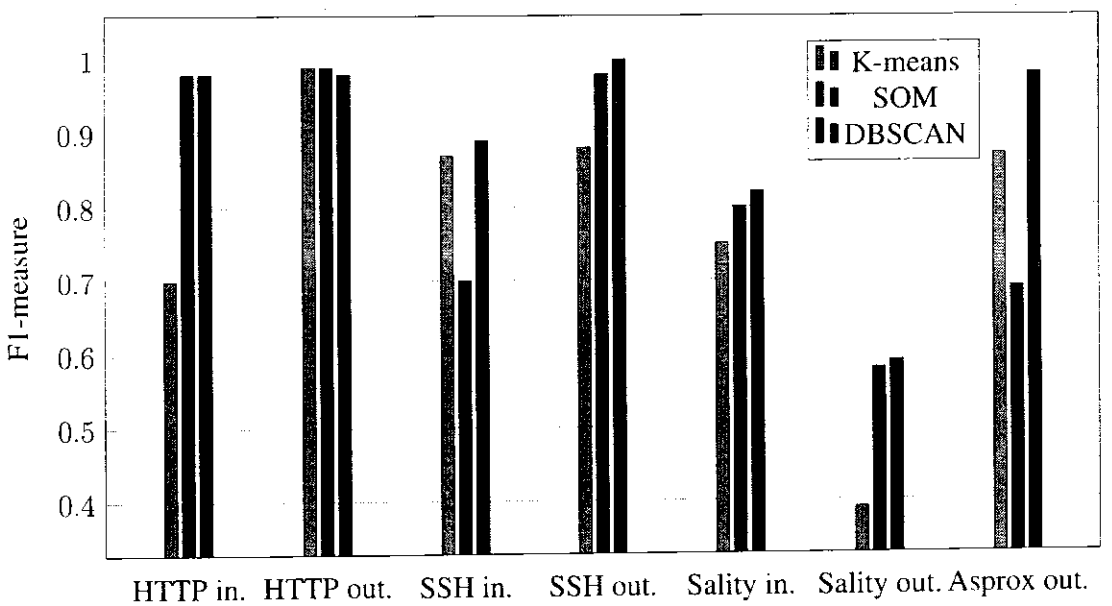


Figure 6.8: Comparison of malicious flow clustering results for K-means, SOM, and DBSCAN

determine the number of output clusters from the data. Table 6.4 gives the results of DBSCAN clustering. The incoming and outgoing HTTP attack traffic is correctly clustered with F1-measure of 0.98. All flows for incoming and outgoing SSH attack traffic are also correctly clustered with F1-measure of 0.98 and 1.00 respectively. The Sality incoming and Asprox outgoing traffic has an F1-measure of 0.82 and 0.98 respectively. However, clustering of outgoing Sality attack traffic has low performance and gives an F1-measure of 0.59.

6.5.3 Discussion

The experimental results show that unsupervised clustering algorithms successfully classify the majority of malicious flow records in different attack clusters. We have used three unsupervised learning algorithms, k-mean, SOM and DBSCAN for attack clustering. Figure 6.8 gives a compar-

ison of clustering of all three techniques. Table 6.5 lists the overall accuracy (OA) of all algorithms. The overall accuracy is calculated as the total number of correctly clustered flows divided by the total number of flows. The overall accuracy shows that DBSCAN gives the best performance as compared with k-means and SOM.

K-means clustering achieves an overall accuracy of 78%. K-means is a simple and efficient method of clustering and gives good performance when the flows are distinct. The k-mean clustering experiment shows that k-mean has very low accuracy in clustering of Sality malware traffic. The Sality malware's incoming and outgoing flows are similar to each other, therefore, k-mean is not able to cluster Sality flows accurately. Self-organizing map (SOM) gives better performance than k-mean and has an overall accuracy of 80%. SOM has good performance for all clusters but shows lower accuracy in clustering of Asprox flows. The Asprox malware traffic consists of variety of flows with different behavior and SOM is able to place them in a single attack cluster. SOM also has the capability of generating visual cluster map of malicious flows.

DBSCAN algorithm has the highest overall accuracy of 89% and places all types of flows in correct cluster. DBSCAN automatically determines the number of output clusters using the $espsilon$ and $minPoints$ values. However, these values have to be carefully set for best clustering performance. A slight change in these parameters affect the overall accuracy and also change the number of output clusters. DBSCAN has placed almost all types of flows in correct attack clusters. The performance of DBSCAN is also low for Asprox malware traffic due to variety in flows.

Both k-means and SOM algorithms require the number of clusters as an input parameter. The performance of clustering algorithms is largely dependent on correct estimation of number of clusters [126]. Generally it is assumed that this parameters is user supplied [127]. Our proposed scheme also requires that number of clusters for k-means and SOM are carefully set using domain knowledge of malicious traffic. Techniques such as [127], also exists which automatically determine the number of optimal clusters in the data.

Another important aspect is the labeling of flow clusters. We have manually labeled the clusters by inspecting a sample of flows in every cluster. However, the correct labeling is also dependent on the number of output clusters. If the number of clusters is low, a cluster will contain flows of different attack types and a single attack label may not be enough to describe all malicious flows. Similarly, if the number of clusters is too high, flows of a single attack type will be distributed in multiple clusters.

In unsupervised clustering, clusters are created from unlabeled flows and no ground truth is avail-

able. It is difficult to determine that an flow is placed in the correct cluster or not. This problem can be solved by allowing only those flows in a cluster which lie within a certain distance from the cluster centroid. We can define a threshold value on the probability or distance value of cluster membership.

Our proposed technique uses flows to cluster different attacks. Flow-based intrusion detection system does not detect attacks which are hidden in the packet payload. These attacks do not affect the normal flow behavior. Web-based attacks such as SQL Injection, Cross-side scripting do not cause any variation in the flow attributes and flow-based detection can not detect them [128] . However, if an attacker makes repeated attempts for SQL injection, it will be reflected in flows and can be detected. Our technique is not able to cluster malicious flows of web-based attacks due to the limitation of flow-based detection.

6.6 Chapter Summary

This chapter presented a novel technique for automatic clustering of malicious flows. Our technique uses unsupervised learning techniques and creates attack clusters from the unlabeled training set of malicious flows. Once the clusters are created, incoming flows are compared with all clusters and placed in the closet attack cluster. The intrusion detection system raises a consolidated alert for every set of flow which belong to same attack cluster. We have used three unsupervised learning techniques, k-mean, SOM and DBSCAN and evaluated them on a flow-based dataset. Results show that our technique successfully identifies various attacks in malicious flows and places the majority of the flows in correct attack clusters. Overall, our work has following important contributions; First, the automatic clustering of malicious flows is helpful in reducing the total number of alerts generated by the intrusion detection system. Second, we use flow records for clustering of malicious traffic and no payload inspection is performed. Flow records are capable of differentiating almost all type of attacks except those which are hidden in the packet payload. Third, we employ unsupervised learning techniques and no labeled training dataset is required. In next chapter, we combine the contributions of chapter 4 & 5 and proposed a two-stage flow-based intrusion detection.

Chapter 7

Implementation of two-stage flow-based IDS

7.1 Overview

In this chapter, we implement a flow-based intrusion detection system (IDS) to detect malicious traffic in next-generation networks using unsupervised learning techniques. We used the two-stage intrusion detection model presented in Chapter 4 to implement the IDS. The experiments conducted in Chapter 5 and Chapter 6 provided us with the knowledge about the accuracy and performance of different machine learning techniques. The experiments show that one-class Support Vector Machines (SVM) and Self-organizing Maps (SOM) are better techniques for the implementation of first and second stage detection stages in the two-stage model respectively. Therefore, we have used one-class SVM and SOM to designed first and second detection of the IDS. Our implementation support unsupervised learning therefore no labeled dataset is required. We have evaluated the IDS on multiple flow-based datasets and obtained promising results.

7.2 Two-stage flow-based IDS

The two-stage flow-based IDS is based on the two-stage intrusion detection model presented in Chapter 4. In the two-stage model, the first stage classifies the flows into normal and malicious while second stage groups the malicious flows into different attack types. We also proposed the use of machine learning techniques for implementation of the first and second stage. In chapter 5, we conducted an experiment to evaluate available one-class classification techniques for detection of

malicious flows. The results show that SVM-based one-class technique performs better than other techniques. In Chapter 6, we performed a second experiment to determine that how unsupervised learning clustering techniques can be used for automatics clustering of malicious flows according to attack types. We have evaluated K-means, SOM and DBSCAN clustering techniques. Although DBSCAN performs better than SOM, we have selected SOM for implementation of second stage due to following reasons:-

- It is difficult to specify the optimal value of DBSCAN parameters; epsilon and minPoints.
- The number of output clusters in DBSCAN depend on the value of epsilon and minPoints. Therefor it is difficult to specify a fixed number of output clusters.
- Although optimal value of epsilon and minPoints give better performance, the number of output clusters may not represent the actual types of attacks in the malicious traffic

Figure 6.3 shows the architecture of the proposed IDS. The IDS analyzes flow data to detect malicious network traffic. The intrusion detection model consists of two stages. The first stage detection process employs a one-class support vector machine (SVM). The one-class SVM classifier only identifies malicious flows while all other flows are discarded. The malicious flows are forwarded to the second stage which uses self-organizing map to group similar malicious flows into different attack clusters. Every attack cluster represents a specific type of network attack.

7.2.1 First stage intrusion detection

The first stage detection separates malicious and normal flows network traffic. Since we have only one class of interest i.e. malicious, the problem is solved by using a one-class classifier [85]. The one-class classifier recognizes objects of only one class. All input objects are either belong to a target (positive) class or considered outliers [129]. One-class classification is used when training dataset for only one class (target class) is available. The training dataset for other classes is either not available or difficult to obtain.

The one-class classification has already been in use for intrusion detection [89]. Available once class classification includes density estimation, reconstruction methods and support vector machines (SVM). We use SVM-based one-class classification techniques because SVM techniques give accurate results for intrusion detection [130]. One-class SVM constructs a boundary around the target class objects in the form of a hyperplane. The hyperplane is constructed in the feature space such that distance from the origin is maximum [99].

Mathematically, we assume that x_i is a training example from dataset $X = \{x_1, \ldots, x_m\}$ in the input space. Let ϕ is a mapping function which maps the input feature space X to a high dimensional feature space H. The dot product in H is computed using following simple kernel function:-

$$K(x, y) = (\phi(x).\phi(y))_H \tag{7.1}$$

To separate the input examples from the origin with maximum margin in a hyperplane, following objective function is applied [93]

$$min_{w,\xi,\rho} \frac{1}{2} \|w\|^2 + \frac{1}{m\nu} \sum_{i=1}^{m} \xi_i - \rho \tag{7.2}$$

Subject to

$$(w.\phi(x_i)) \geq \rho - \xi_i, \xi_i \geq 0 \tag{7.3}$$

The ξ_i is a slack variable used to penalize the outliers. The $\nu \in (0, 1)$ is a user-defined error control parameter and sets an upper bound on the number of outliers and a lower bound on the number of support vectors. A function $f(x)$ is defined which takes the value +1, if x falls within the hyperplane and -1 otherwise. Solving the above the minimization problem, the decision function for classification is defined:

$$f(x) = sgn((w.\phi(x)) - \rho) \tag{7.4}$$

The one-class SVM is a supervised learning algorithm and requires labeled training set for target class examples. To use one-class SVM with unsupervised learning, we employ an enhancement proposed by [93]. The enhancement considers the normal flows in the training dataset as outliers and removes them before training. The enhanced SVM introduce a variable η which represents an estimate that an instance in the unlabeled training set belongs to the target class (malicious flows) or is an outlier (normal flows). The η has value near to 0 for all outliers and eliminates the effect of outliers in the SVM training. Another variable β controls the maximum number of points that are

allowed to be outliers. Using enhancement proposed in [93], the Equation (2) can be written as:

$$\begin{gathered} min_{w,\rho} min_{\eta_i} \frac{1}{2} \|w\|^2 + \\ \frac{1}{m\nu} \sum_{i=1}^{m} \eta_i max_i(0, \rho - w\phi(x_i)) - \rho \\ \text{subject to } e^T \eta \geq m.\beta \end{gathered} \tag{7.5}$$

The minimization problem shown in Equation 7.5 is a non-convex problem which means that it is difficult to find a global minimum point. The problem is solved using the concave convex procedure [93]:

Let $g(h(w))$, where $h(w) = max(0, w\phi(x))$ and $g(u) = inf_{\beta \in 0.1}[\beta^T \mu]$, using concave duality, the objective is reformulated into following equation:

$$\begin{gathered} min_{w,\rho,\eta} E_{vex} + E_{cave} \\ E_{vex} = \frac{\|w\|}{2} + \eta h(w), E_{cave} = g^* \eta \end{gathered} \tag{7.6}$$

where $g*$ is the concave dual of g.

The enhanced one-class SVM requires that malicious flows in unlabeled training dataset should be in sufficiently large quantity then normal flows. We propose the use of honeypot-based flow collection architecture to generate the unlabeled dataset [131]. The flow records collected through honeypot are mostly malicious [132]. Figure 7.1 describes the flow collection process using honeypot. The honeypot is directly connected to the external routing interface. The flows collected through honeypot are stored in a flow database. The flow database containing both malicious and normal flows is directly accessibly by the one-class SVM.

Figure 7.2 shows one-class SVM training model. The unlabeled flows are passed through a outlier detection step which removes the normal flows. Remaining malicious flows are used to build a "malicious flow classification profile". After training, the one-class SVM is used to process the flows being extracted from the network as show in Figure 6.3. The malicious flows are forwarded to stage-II while normal flows are discarded.

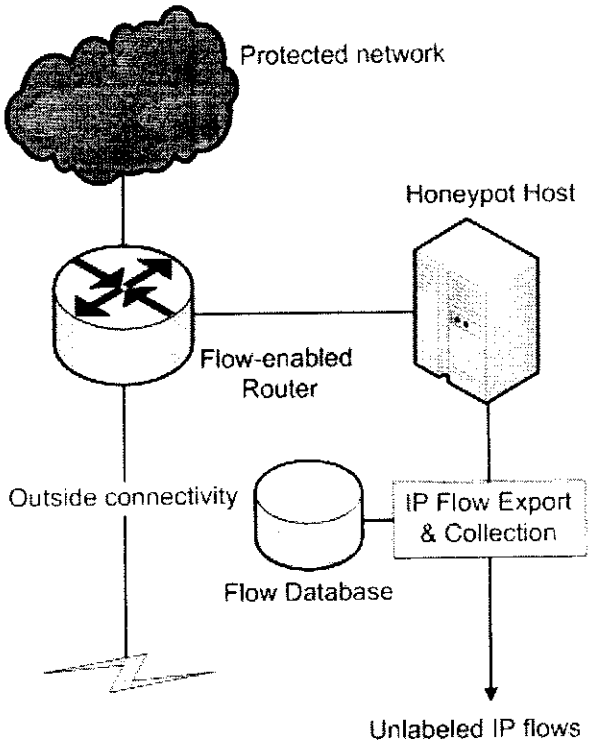


Figure 7.1: Flow collection process

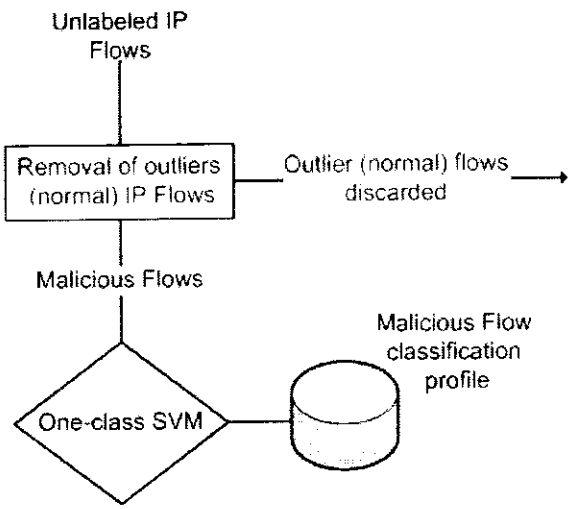


Figure 7.2: One-class SVM training

7.2.2 Second stage intrusion detection

The first stage detection process only classifies the network traffic into normal and malicious categories. It does not associate an attack class with malicious flows. These malicious flows require a manual inspection to determine the attack type and corrective actions. Although malicious flows are in a small fraction as compared to normal network traffic, these flow can still be in large numbers in NGN environment. Manual inspection of such large number of flow is a difficult task. The second stage detection process uses unsupervised learning to automatically group similar malicious flows into pre-defined attack clusters. We give an attack label to every cluster. All incoming flows are compared with all clusters and the label of the closest cluster is given to the malicious flow. The alert triggering component raises a consolidated anomaly alert for a set of similar flows having same attack labeled.

The second stage also uses unlabeled training dataset due to difficulty in obtaining labeled training set in a NGN environment. Unsupervised learning algorithm uses the similarity in flow attributes for clustering of flows into different attack clusters. We use self-organizing map (SOM) for classification of flows into a fixed number attack clusters. SOM is a neural network consisting of an input and output neuron layers. The neurons in the input layer interconnects with neurons in output layer through unsupervised competitive learning network [123]. The competitive learning is a winner-take-all approach and consists of two steps; competition and cooperation. In competition phase, a neuron in output layer is selected among all competing neurons using minimum Euclidean distance. The neuron whose weight vector comes closest to the input vector is declared winner.

Mathematically, for each input $v \ni V$, i^* neuron is declared winner if:

$$i^* = argmin_i ||w_i - v|| \tag{7.7}$$

In cooperation, the weights of the winning neuron and its neighboring neurons are adjusted using a time decay function. The effect of weight adjustment is high at the origin and decreases with the distance and time. The range of the neighborhood is defined by a Gaussian function:

$$\sigma(t) = \sigma_0 e^{(-2\sigma_0 \frac{t}{\tau_m})} \tag{7.8}$$

where
σ_0 = Initial value of neighborhood range

t and t_m= The current and maximum iteration respectively
$\sigma(t)$ = The range of neighborhood at t stage.

After a winning neuron is selected, the weights of neighboring neuron vectors are adjusted:

$$w_i(t+1) = w_i(t) + \eta(t)\sigma(t)(v - w_i(t)) \tag{7.9}$$

In above equation, t represents the current stage and $\eta(t)$ is the learning rate. The continuous process of competition and cooperating marks the cluster on topographic self-organizing map. Each neuron on the output layer denotes the resultant clusters. The number of output clusters has to be set before the classification by a user defined parameter k. The model uses one-class classification at first stage for detection of malicious flows. The one-class classifier only identifies malicious flows while all other flows are discarded. The malicious flows are forwarded to the second stage. The second stage uses self-organizing map for automatic clustering of similar malicious flows.

7.2.3 Deployment of two-stage IDS in Next-generation network

Figure 7.3 shows the implementation of flow-based intrusion detection system in NGN framework. The flow information is collected at the provider edge and forwarded to the intrusion detection system. Provider edge is a router installed at the boundary of the network. The intrusion detection system analyzes the flow records passing through the provider edge and raise an alarm if malicious flows are detected.

7.3 The datasets

We have used three flow-based datasets to obtain experimental results. The first dataset was developed in University of Twente and is publicly available [28]. We have created the second dataset ourselves by combining flows of various malware and Advanced Persistent Threat (*APT*) with normal flow traffic. The third dataset is a SIP dataset developed by Nassar et al. [133]. The flow records in all dataset are in Netflow v5 format. The attributes of flow records are described in Table 7.1.

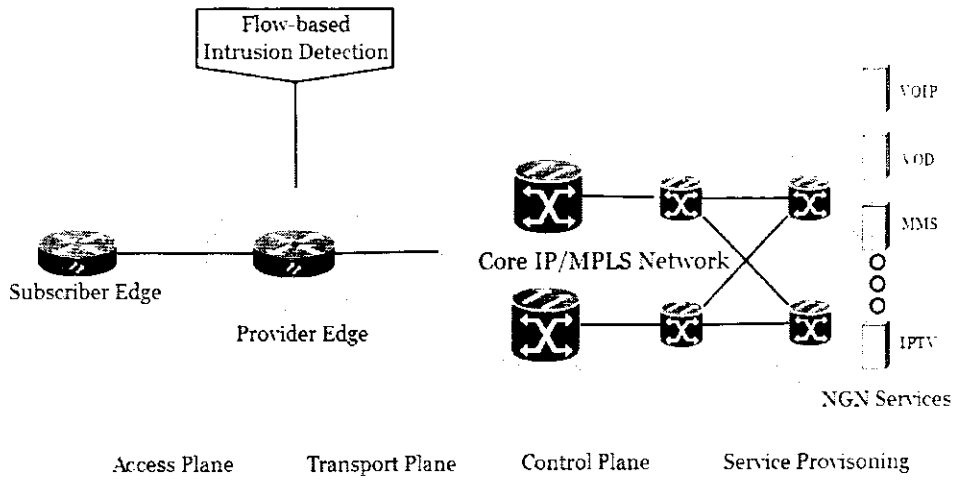


Figure 7.3: Deployment of two-stage IDS in Next-generation network

Table 7.1: Features for flow records in Sperotto dataset

Attribute	Description
Flow ID	The unique identifier of the flow (Not used in detection)
Source IP	The IP Address of the source.
Destination IP	The destination IP address
Packets	Number of packets in flow
Octets	Number of bytes in flow
Duration	The duration of flow in milliseconds
Source Port	Source port number
Destination Port	Destination port number
TCP Flags	Cumulative OR of TCP flags
Protocol	The transport layer protocol such 6=TCP, 17=UDP

Table 7.2: Detail of flows in Sperotto dataset

Alert Type	No. of Flows	Category
SSH	13942629	Malicious
FTP	13	Malicious
HTTP	9798	Malicious
AUTH-IDENT	191339	Normal
IRC	7383	Normal
OTHERS	18970	Normal

Table 7.3: Sampled flow dataset for one-class classification - Sperotto dataset

Training dataset		**Testing dataset**	
Malicious	Normal	Malicious	Normal
10000	500	11740	12500

7.3.1 Sperotto dataset

The Sperotto dataset consists of 14.2M flow records collected through a "Honeypot" deployment in University of Twente (UoT) network [28]. The honeypot was directly connected to internet to ensure maximum exposure to attacks. Three common services SSH, HTTP and FTP were run over the honeypot. The log files of the services were used for the labeling of flow records. Some traffic in the dataset is the side effect of alerts and is not considered malicious. During the flow collection, one hacker installed an IRC proxy over the honeypot. The traffic generated due to IRC is also non-malicious. We have used the non-malicious traffic as normal traffic. The alert types and number of flows corresponding to each alert type are shown in Table 7.2.

The four time related attributes *start-time, start-msec, end-time and end-msec* in the original dataset are computed to a single attribute of duration in milliseconds [50]. The dataset contains a very large number of flows which can be difficult to process due to computing limitations. Therefore, we have extracted a random subset of the dataset. We have created two datasets from the original datasets. The first dataset is a training dataset and contains malicious flows with a small amount of normal flows as outliers. The 10,500 flows from the dataset for training of one-class SVM in Stage-I detection process. After training, a test dataset consisting of 24240 flows is used for evaluation of one-class SVM. Table 7.3 gives detail of training and test dataset.

Table 7.4: Test and training dataset - APT and Malware dataset

Training dataset		**Testing dataset**	
Malicious	Normal	Malicious	Normal
3524	350	5286	24367

7.3.2 APT and Malware dataset

The Sperotto dataset [28] has a limited variety of malicious traffic. Most of the malicious traffic in the Sperotto dataset only consists of SSH attacks flows. Most of the modern networks attacks are carried out using malware and advance persistent threats (APTs). We have created a dataset of flows of traffic originating from different malwares. We have generated outgoing flow records for different malware and APTs using packet capture files obtained from Contagio Malware Dump[1]. We have considered Sality, Asprox, TBot and Nuclear malware. The flow records of these malware and APTs are combined with normal flow traffic. The normal traffic flows are collected from a campus network. These normal and malicious flows are used to create training and test dataset. The training dataset contains 3524 malicious and 350 normal flow records while test dataset has 5286 and 24387 flow records. Table 7.4 shows the detail of dataset

7.3.3 SIP dataset

The third dataset is a labeled VoIP dataset consisting of SIP packet traces [133]. The dataset has two sets of SIP traces collected from two different VoIP testbed networks. The first testbed uses Asterisk PBX server and the second testbed uses the OpenSIP proxy with RADIUS servers. We have only considered the OpenSIPs traces for evaluation in our experiment. The testbed configuration includes OpenSIP proxy and-and normal and malicious traffic generators. The normal traffic is emulated by groups of VoIP bots. Each group of bots connects with the internal and external interface of the SIP proxy respectively. The malicious traffic is generated using the Inviteflood and Splitter attack tools. The dataset is available in the form of SIP packet traces. We have used ntops' nProbes tool to extract Netflow v5 based flow records from the SIP packet traces. The detail of flow records in the dataset is given in Table 7.5.

[1] http://contagiodump.blogspot.com/

Table 7.5: Detail of flows - SIP dataset

Traffic Type	No. of flows	Category
InviteFlood SIP traffic	6496	Malicious
Splitter SIP traffic	3927	Malicious
Normal SIP traffic	7901	Normal

Table 7.6: Test and training dataset - SIP dataset

Training dataset		**Testing dataset**	
Malicious	Normal	Malicious	Normal
2083	300	10423	7901

7.4 Results

The proposed IDS is evaluated using the three datasets discussed in Section 7.3. We now discuss the results of each experiment.

7.4.1 Evaluation on Sperotto dataset

In first experiment, we evaluate the proposed IDS using the subset of UoT dataset [28]. Table 7.3 shows the detail of dataset. In first stage detection, enhanced one-class SVM uses the unlabeled training dataset for learning. The training dataset contain both normal and malicious flows but no flow is marked as malicious or normal. The enhanced one-class SVM detects the normal flows in the training dataset as outliers and does not use them for training. Table 7.7 shows the confusion matrix for outlier detection process. The one-class SVM successfully removes 98.40% normal flows from the training dataset. The remaining 9161 flows out of 10000 are used by the one-class SVM for learning the malicious behavior.

After training, the one-class SVM process the test dataset. We have used one-class SVM with

Table 7.7: Confusion Matrix for outlier detection during one-class SVM training - UoT dataset

Classified as	Malicious	Normal (Outliers)
Malicious	9161	839
Normal (Outliers)	8	492

Table 7.8: First stage detection results - UoT dataset

Recall	Precision	FPR	F1-Measure
0.9991	0.9001	0.1040	0.9470

sigmoid kernel. The one-class SVM correctly marks 11730 flows as malicious out of 11740 total flows. The false positive rate is also low and only 1301 normal flows have been classified as malicious. Table 7.8 shows the precision, recall, false positive rate and F1-score for one-class SVM.

The flows identified as malicious by one-class SVM in first stage are forwarded to second stage. These malicious flows are also used by the self-organizing map(SOM) to different attack clusters in second stage detection process. We manually set the number of attack cluster in SOM to six using domain knowledge of the evaluation environment. Each alert cluster relates to a specific attack types or a service that is under attack. The six attack clusters include the incoming and outgoing flow traffic for SSH and HTTP services and two additional clusters for placement of unknown alerts and miss-classified flows.

The second stage uses a self-organizing map (SOM) to categorizes the flows in different alert clusters. The total number of flows marked malicious by first stage are 1303 including 1301 false positives. The SOM process all malicious flows and places them in the closest attack cluster. The clustering results are given in Table 7.9. The HTTP IN, HTTP OUT and SSH IN categories remain consistent and similar number of flows are available in the output clusters. The actual number of flows in HTTP IN category are 2127 while the output cluster contains 2154 flows. Therefore only 27 flows are placed incorrectly. The HTTP OUT cluster has 2113 flows in input dataset and its output cluster contains 2085 with 28 flows placed in incorrect cluster. The SSH IN cluster has 4140 flows in the input set while output cluster contains 3992 flows. In this case, 148 flows have been incorrectly classified. The actual number flows for SSH OUT category is 3360 while the output cluster has 4006 flows. The SSH OUT category has highest number of incorrectly classified flows i.e. 646. Also 770 and 24 flows are placed in Other-I and II clusters respectively. This relatively high rate of miss classification is due to the 1301 false positives (normal flows) of first stage detection process. The comparison of clusters with actual flows in the input set is given in Figure 7.4.

Table 7.9: Clustering Results of SOM Classifier- UoT dataset

Alert Cluster	Actual Flows	Clustering results
HTTP IN	2127	2154
HTTP OUT	2113	2085
SSH IN	4140	3992
SSH OUT	3360	4006
Other-I	0	770
Other-II	0	24
Total	11740	13031

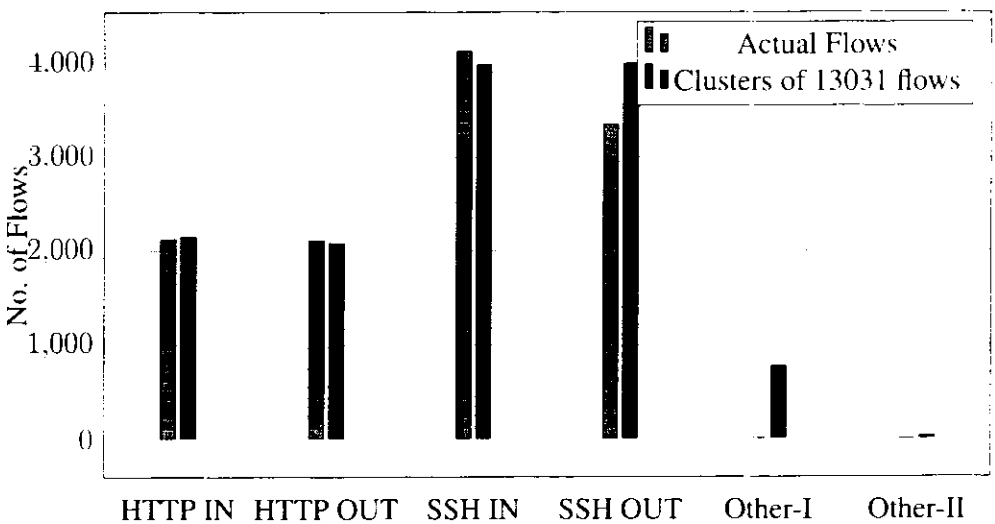


Figure 7.4: SOM Clustering results comparison - UoT dataset

Table 7.10: Confusion Matrix for Outlier detection during one-class SVM training - Malware and APT dataset

Classified as	Malicious	Normal (Outliers)
Malicious	2857	330
Normal (Outliers)	20	667

Table 7.11: First stage detection results - Malware and APT dataset

Recall	Precision	FPR	F1-Measure
0.9876	0.9170	0.017	0.9507

7.4.2 Evaluation on Malware and APT dataset

In second experiment, we use malware and APT flow dataset for evaluation. The enhanced one-class SVM uses unlabeled training dataset containing both normal and malicious flows. The enhanced one-class SVM removes the normal flows from the training dataset leaving only the malicious flows. Table 7.10 shows the confusion matrix for outlier detection in training flows. The one-class SVM successfully removes 94.28% normal flows from the training dataset. The remaining 2857 flows are used by the one-class SVM and SOM for learning the malicious behavior and creation of different attack clusters. We manually set the number of attack cluster in SOM to six which include four malware and APT cluster and two additional clusters to place the un-clustered flows.

The trained one-class SVM is presented with a test dataset of 29654 flows. It classifies 5226 flows as malicious. The detected malicious flows also include 434 normal flows as false positives. Table 7.8 shows the precision, recall, false positive rate and F1-score for one-class SVM. The 5226 malicious flows are forwarded to SOM clustering algorithm for placing into different attack clusters.

The SOM places the malicious flows into different attack clusters. Table 7.12 gives the clustering results. Some flows of Sality malware are placed Asprox cluster because Asprox malware traffic is not uniform. The false positives of first stage detection process are separated into Other-I and II clusters. Figure 7.5 compares the result of clustering with actual flows.

Table 7.12: SOM clustering results - Malware and APT dataset

Alert Cluster	Actual Flows	Clustering results
Sality outgoing	1669	1312
Asprox outgoing	3336	3649
TBot outgoing	133	200
Nuclear outgoing	88	64
Other-I	0	2
Other-II	0	59
Total	5286	5226

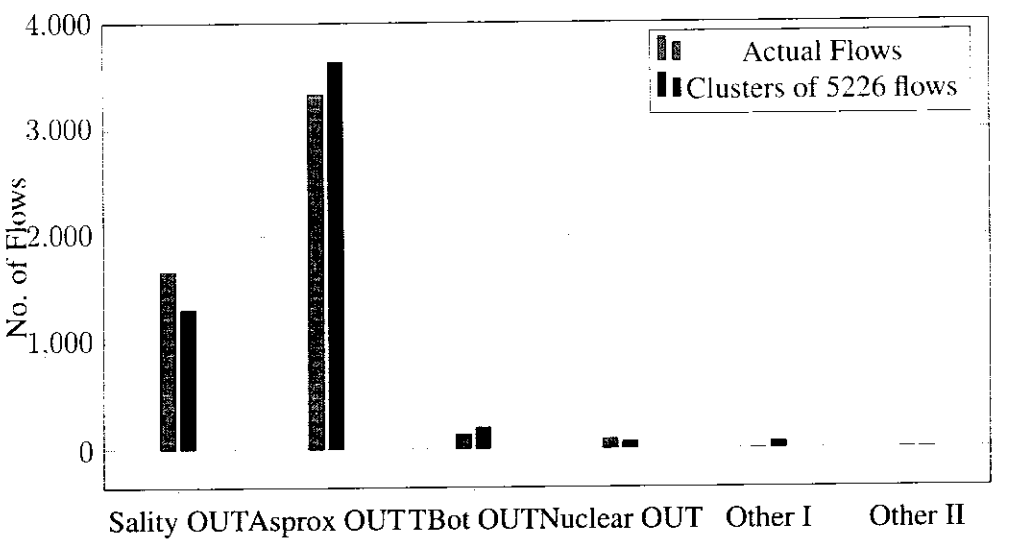


Figure 7.5: SOM Clustering results comparison - Malware and APT dataset

Table 7.13: Confusion Matrix for Outlier detection during one-class SVM training - SIP dataset

Classified as	Malicious	Normal (Outliers)
Malicious	1701	91
Normal (Outliers)	30	170

Table 7.14: First stage detection results - SIP dataset

Recall	Precision	FPR	F1-Measure
0.9920	0.9937	0.00	0.9928

7.4.3 Evaluation on SIP dataset

In the third experiment, we have used the SIP dataset, given in Table 7.6, for evaluation of IDS. In first stage detection, the enhanced one-class SVM uses the unlabeled training dataset for learning. The one-class SVM narks the normal flows as outliers and does not use them from learning. Table 7.13 shows the confusion matrix for outlier detection process. The one-class SVM successfully removes 94.05% normal flows from the training dataset. The remaining flows are used by the one-class SVM for learning the malicious behavior.

After training, the one-class SVM process the test dataset. The one-class SVM correctly marks 10339 flows as malicious out of 10456 total malicious flows. There is no normal flow marked as malicious thus the false positive rate is zero. Table 7.14 shows the precision, recall, false positive rate and F1-score values for the first stage detection.

The malicious flows identified in first stage detection are forwarded to the second stage. The second stage uses som for clustering of malicious flows according to attack type. We have used the same training dataset used in first stage for training of SOM. Since there are two types of malicious flows in the dataset, we have set the number of clusters to four. The two additional clusters, Other-I and Other-II, are included to contain the flows which SOM fails to associate with any attack type. The results of clustering process are given in Table 7.15. The first cluster consist of malicious flows belonging to SIP flood. The SOM is able to cluster 4848 flows out of total 6224 flows. The actual number of in second attack cluster are 4815. However the resulting cluster consist of 4834 flows which also includes some flows belonging to first attack cluster. The number of un-clustered flows are 657 which are placed in Other-I and Other-II cluster.

Table 7.15: SOM clustering results - SIP dataset

Alert Cluster	Actual Flows	Clustering results
SIP Flood	6224	4848
SIP Spitter	4815	4834
Other-I	0	162
Other-II	0	495
Total	10339	10339

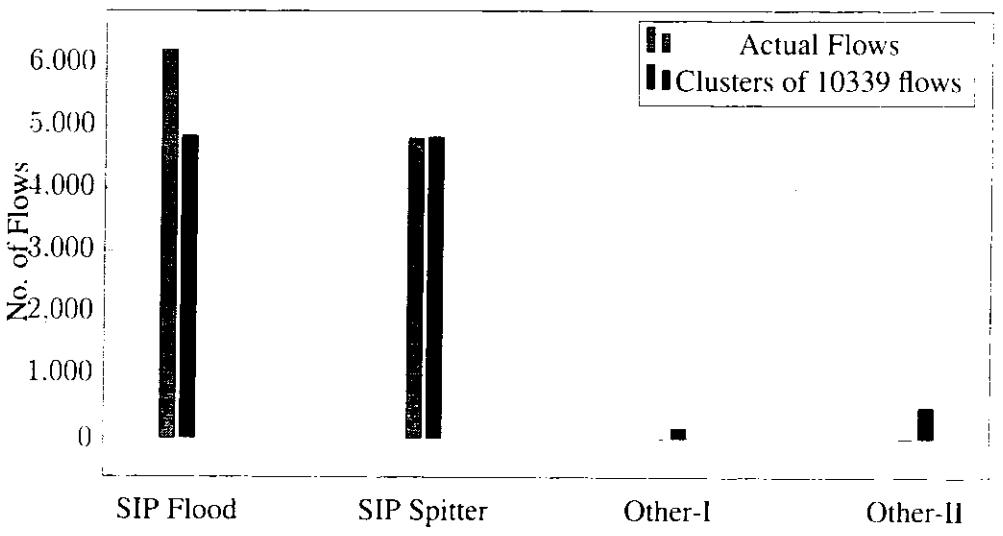


Figure 7.6: SOM Clustering results comparison - SIP dataset

7.5 Discussion

In this chapter, we proposed and evaluated a two-stage flow-based intrusion detection system for next-generation networks. Next-generation networks are all-IP communication infrastructures providing voice, video and data services. NGN encapsulate a variety of network architectures, services, and protocols. A straightforward way to monitor the NGN is to use the flows data. The IPFIX/Netflow provides a unified way to access traffic flow information from an all-IP next-generation network. We use flow data for intrusion detection process in NGN. Our proposed IDS process the flow data in a two-stage detection process. The first stage uses a one-class SVM for efficient detection of malicious flows. The one-class SVM discards all normal traffic flows and only retains malicious traffic for detail analysis. The low volume of flows during detail analysis enhances the system efficiency.

We use an enhanced version of one-class SVM which reduces the effect of outliers in SVM training. However, the algorithm proposed by is only valid if outliers (normal flows) in the training set are in sufficiently small quantity.

The accuracy of one-class SVM is very sensitive to the value of ν parameter. The ν is an upper bound on the number of outliers (normal flows) and lower bound on the number of support vectors. We have experimented with different values of ν to obtain best possible results. The optimization of ν parameter is also a promising research area and different techniques have been proposed to find out the optimal value of ν [134].

Our proposed technique uses flows for intrusion detection. Although flow-based detection is efficient, it does not have access to the important information about the viruses and malware in the traffic payload. Flow-based intrusion detection system is unable to detect attacks which are hidden in the packet payload such as SQL Injection, Cross-side scripting etc.

In some cases, network are built on proprietary protocols for which real-time training datasets are not already available or it is impossible to generate training datasets due to confidentiality and privacy of network traffic. In this case our model can be easily deployed since it uses unsupervised learning and does not required labeled class training.

Another important feature of our system is the use unsupervised learning. The unsupervised learning does not need need a labeled training datasets which is difficult to obtain for next generation networks.

The overall results demonstrate that our proposed technique detects almost all malicious flows in the dataset and correctly place them in the corresponding alert cluster. The experimental results also show that two-stage classification of flows gives better accuracy by reducing the false positives in second stage. The SOM classifier is also able to detect unknown alerts in the flows as an early warning. The experiment results confirms that proposed technique is effective and can be used in high-speed next-generation networks without effecting the network throughput. At service provider level, maintaining the QoS and SLA is a business requirement. As the proposed IDS model only processes the flow i.e. only a fraction of whole traffic, it can be easily deployed without affecting the network performance and user services.

7.6 Comparison with other techniques

The multi-stage intrusion detection using flow records is a novel idea. We have compared the results of our system with other two-stage flow-based approaches proposed in [48] and [135]. The comparison of results using F1-measure and false positive rate is given in Table 7.16. The comparison shows that our technique outperforms the other techniques in first stage detection. In second stage detection, our technique gives better results than [135]. However, the technique proposed in [48] has higher detection rate then our technique due to the use of supervised learning.

Table 7.16: Comparison of results with other mutlti-stage techniques

Detection stage	Measure	[48] (DARPA)	[135] (KDD99)	Our approach	
				Sperotto dataset	APT and malware dataset
First stage	Detection rate	0.9420	0.90	0.99	0.9876
	False positive rate	0.034	0.035	0.1040	0.017
Second stage	Detection rate	0.9942	0.90	0.95	0.93
	False positive rate	0.32	N/A	N/A	N/A

In another dimension, we have compared the results of our experiment with other technique evaluated on Sperotto's dataset. Table 7.17 give detail of the comparison. The results show that our technique outperforms other approaches.

Table 7.17: Comparison of results with other techniques evaluated on Sperotto's dataset

Technique	Performance Measure	Technique's Result	Our IDS Result
[49]	Detection Rate	99.55%	99.91%
[52]	Detection Rate	99.34%	99.91%
[50]	Recall	0.98	0.991

7.7 Chapter Summary

This chapter describe the architecture of two-stage flow-based intrusion detection system for next-generation networks. Our technique uses a two-stage classification scheme to detect malicious. The first stage detection consists of an enhanced one-class SVM that separates malicious flow from normal network traffic. The second stage detection uses a self-organizing map(SOM) which cluster the the malicious flows predefined attack classes. The flow collection and intrusion detection process in the our technique is independent of underlying network architecture. Therefore the proposed intrusion detection system can be deployed in any network with just enabling the IPFIX/Netflow support. The aspect is particularly useful in the context of next-generation networks (NGN) where different networks are converged to an all IP platform. we have evaluated the proposed model on two-flow based datasets. Results shows that flows contain sufficient information which can be used by the proposed model for intrusion detection next-generation networks. Also our technique gave promising results in both detection stages. We have compared the results with other approaches and our results outperforms the existing techniques.

Chapter 8

Conclusion and Future Work

8.1 Overview

Intrusion detection system is a useful tool for the protection of computer networks. We have proposed a flow-based IDS for next-generation networks using unsupervised learning techniques. We evaluated our proposed IDS on multiple flow-based IDS and obtained promising results. This chapter concludes our research and revisit the research objectives outlined in Chapter 1. We will also discuss the impact of our research and give directions for future research in flow-based intrusion detection systems.

8.2 Conclusion

Next-generation networks provide voice, video and data services in an IP-based network. Flow-based intrusion detection system is a suitable solution in NGN environment because the process of flow collection is not dependent on underlying network architecture and user services. This aspect is particularly useful in the context of next-generation networks (NGN) where different networks are converged to an all-IP platform. The problem statement of our research was to find out that how flow-based intrusion detection can be implemented in next-generation networks using unsupervised machine learning techniques. We proposed a two-stage flow-based intrusion detection model for next-generation networks. The first stage of our model classifies the flow records into the normal and malicious classes. The second stage detection process performs detail analysis and

classifies the flow into different attack types.

We now present the detail conclusion of our research concerning the research objectives formulated in Section 1.7. Our first objective was to identify the limitation of traditional intrusion detection systems when implemented in next-generations networks. Section 1.4 and 2.5 described that traditional approaches to intrusion detection are still struggling with high false-positive rate, slow throughout and inability to inspect encrypted payload. The next-generation network (NGN) architecture requires a novel solution for intrusion detection which is highly efficient and is independent of underlying network architecture and protocols. Flow-based intrusion detection system is a suitable solution in NGN environment because the process of flow collection is not dependent on underlying network architecture and user services. The aspect is particularly useful in the context of next-generation networks (NGN) where different networks are converged to an all-IP platform.We described that traditional intrusion detection system suffers from several weaknesses which include, slow throughout, inability to inspect encrypted packet payload, high deployment cost, and privacy concerns. Due to these weaknesses, implementation of traditional intrusion detection systems is difficult in next-generation networks. An innovative technique for design of intrusion detection system is the use of flow records.

Our second objective was to review the current state of the art in flow-based intrusion detection. A detailed review of available flow-based intrusion detection techniques in Chapter 3. We have created a taxonomy of flow-based intrusion detection systems based on the techniques used for the detection of malicious flows. We have reviewed 39 research articles which describe various flow-based intrusion detection techniques. Significant contributions of our review are as under:-

1. The focus of our review is flow-based intrusion detection. An earlier survey of flow-based techniques was [21], which is now six years old. A recent study of flow-based detection techniques is [136]. However, this study only focuses on similarity matching methods. Some techniques like [61], [37], [62] have not been discussed. Other surveys like [25], [27] discuss some flow-based techniques but with limited details.

2. We presented a summary of publicly available flow-based datasets. We also described the process used for generation of the dataset and gave details of important flow attributes.

3. We created a technique-based taxonomy of flow-based intrusion detection systems. Our work is different from the earlier survey papers which were organized on the basis of attack types. We review the available flow-based intrusion detection techniques in every class of methods and discuss their advantages and disadvantages.

4. We also provided a brief summary of important commercially available intrusion detection systems which use flow records for network attack detection.

5. We have identified important open issues and research challenges in flow-based intrusion detection.

Our next objective was to propose an efficient intrusion detection model for next-generation networks. We proposed a two-stage model for flow-based intrusion detection in next-generation networks (NGN). The first stage of the proposed model classifies the flow records into the normal and malicious classes. The second stage detection process performs detail analysis and classifies the flow into different attack types. For our next objective, we also give implementation detail of our model using one and multi-class classification techniques in Section 4.2.2. Our model discards the majority of the flows in the first stage using a computationally inexpensive algorithm. Only malicious flows are analyzed in detail. The two-stage detection model also reduces the false positive rate through the application of two different classification techniques.

Our next objective was to evaluate available one-class classification techniques for detection of malicious of flows in the first stage of our proposed model. We have considered density estimation, reconstruction methods and boundary methods based one-class classification techniques. We have performed the experiment on Scenario 4 & 5 of CTU-13 dataset [30] using Matlab. Chapter 5 gave detail about the experiment and discuss the results. We have obtained result in the form of Area under ROC curve, Precision, Recall and F1-measure. The best F1-score achieved for density estimation, reconstruction methods and boundary methods are 0.9114, 0.8201, and 0.4310 respectively. Our results showed that boundary based one-class classification techniques outperform destiny estimation and reconstruction methods. We, therefore, consider SVM-based one-classification techniques suitable for detection of malicious flows on the basis of experimental results.

In next step, we have evaluated unsupervised learning techniques for automatic creation of attack clusters from the unlabeled training set of malicious flows in the second stage of our proposed model. Once the clusters are created, incoming flows are compared with all clusters and placed in the closet attack cluster. We have evaluated three unsupervised learning techniques, k-mean, SOM and DBSCAN on a flow-based data-set. The results showed that unsupervised learning techniques successfully identified various attacks in malicious flows and placed the majority of the flow in correct attack clusters. Our results showed that DBSCAN gives the best performance. However, we were unable to fix the number of clusters in DBSCAN. Therefore, we have chosen SOM for

two-stage flow-based detection model.

To cover the next research objective, we have proposed a two-stage flow-based intrusion detection system for next-generation networks using the knowledge obtained from the experiments conducted in Chapter 5 and 6. Our proposed model processes the flow data in a two-stage detection process. The first stage uses a one-class SVM for efficient detection of malicious flows. The one-class SVM discards all normal traffics and forward the malicious traffic to second stage detection process. Due to two-stage intrusion detection process, only malicious flows are analyzed in detail. To enable unsupervised learning for one-class classification, we have used an enhanced version of one-class SVM. The one-class SVM removes the outlier from the training dataset, and only target class objects are used for learning the model. The second stage detection uses a self-organizing map(SOM) which cluster the the malicious flows predefined attack classes. The flow collection and intrusion detection process in the our technique is independent of underlying network architecture. Therefore the proposed intrusion detection system can be deployed in any network with just enabling the IPFIX/Netflow support. The aspect is particularly useful in the context of next-generation networks (NGN) where different networks are converged to an all IP platform. we have evaluated the proposed model on two-flow based datasets. Results shows that flows contain sufficient information which can be used by the proposed model for intrusion detection next-generation networks. Also our technique gave promising results in both detection stages. We have compared the results with other approaches and our results outperforms the existing techniques.

8.3 Impact of research

In the end, we discuss the potential impact of our research. The objective of the research was to develop a model that is capable of intrusion detection in next-generation network environment. Traditional intrusion detection techniques cannot be used for high-speed IP networks. Our proposed model has the ability to work any IP based environment with flow capability. Use of flow enables the intrusion detection system independent of underlying network architecture. We have focused on unsupervised learning techniques therefore our system do not require any labeled training dataset.

There are a number of scenarios where our proposed model will be useful. Many organization and enterprises use their own corporate intranets. Such flow based IDS can be deployed at the external interface to detect any possible attack without inflicting load on network resources. Also many

corporate intranets deploy end-to-end encryption device as network usually laid out of the physical boundary of the organization and transfer of plain data cannot be accepted. In this scenario, flow-based IDS can be deployed as it is independent of underlying encryption.

In many cases, network are built on proprietary protocols for which real-time training datasets are not previously available or it is impossible to generate training data sets due to confidentiality and privacy of data. In this case our model can be easily deployed since it uses unsupervised learning and does not required labeled class training. At service provider level, maintaining the QoS and SLA is a business requirement. As the proposed IDS model only processes the flow i.e. only a fraction of whole traffic, it can be easily deployed without affecting the network performance and user services.

8.4 Future Work

We have identified a number of directions in which our work can be extended. An important problem that we faced was the unavailability of standardized evaluation of intrusion detection systems. Very few intrusion datasets are available in public domain for evaluation of packet and flow-based intrusion detection systems. The already available datasets describe a particular attack scenario for some specific protocols. Classic datasets e.g DARPA and KDD99 are not very much relevant today. datasets are very There is need of intrusion datasets containing a variety of protocols and network attacks.

We have proposed the use of one-class classification for intrusion detection at first stage. We have used an enhanced version of one-class classification which supports unsupervised learning. However, there are other unsupervised one-class techniques available which can be evaluated for classification of malicious of flows. Similarly, a number of unsupervised learning techniques can also be explored for clustering of malicious flows according to attack types.

Another point of interest will be to combine multiple one-class classifiers for improvement in performance.

In this research, we have used a Netflow v5 based flow records. In future, the proposed intrusion detection model can be implemented using additional flow attributes. The IPFIX /Netflow v9 define around 280 flow attributes which provide in-depth information about the network traffic. These additional attributes can be used to build intrusion detection schemes for detection of novel

and stealth attacks.

We have used a two-stage model for intrusion detection. Further extension our model with increasing fine-grain attack detection at every stage. However, there will be a trade-off between the number of intrusion detection stages and computational complexity of the IDS.

Intrusion detection process involves the analysis of Big Data. There is a strong motivation to explore the relation of Big Data and flow-based network intrusion detection [137]. Use of Big Data analysis technique can better help understanding the large-scale network traffic.

Bibliography

[1] PWC, "Global information security practices: 2015 survey key findings and trends," http://www.pwc.com, 2015, accessed: 2016-01-20.

[2] R. Koch, "Towards next-generation intrusion detection," in *Cyber Conflict (ICCC), 2011 3rd International Conference on*. IEEE, 2011, pp. 1–18.

[3] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[4] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1, pp. 18–28, 2009.

[5] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied Soft Computing*, vol. 10, no. 1, pp. 1–35, 2010.

[6] T. AbuHmed, A. Mohaisen, and D. Nyang, "A survey on deep packet inspection for intrusion detection systems," *arXiv preprint arXiv:0803.0037*, 2008.

[7] D. F. Hsu and D. Marinucci, *Advances in Cyber Security: Technology, Operations, and Experiences*. Oxford University Press, 2012.

[8] C. Thomas, "Performance enhancement of intrusion detection systems using advances in sensor fusion," Ph.D. dissertation, Supercomputer Education and Research Centre, Indian Institute of Science, 2009.

[9] M. Husak, P. Velan, and J. Vykopal, "Security monitoring of http traffic using extended flows," in *Availability, Reliability and Security (ARES), 2015 10th International Conference on*. IEEE, 2015, pp. 258–265.

[10] S. Elhag, A. Fernández, A. Bawakid, S. Alshomrani, and F. Herrera, "On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems," *Expert Systems with Applications*, vol. 42, no. 1, pp. 193–202, 2015.

[11] M. H. Khyavi and M. Rahimi, "Conceptual model for security in next generation network," in *Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on*. IEEE, 2016, pp. 591–595.

[12] T. Bhattasali, R. Chaki, and N. Chaki, "Study of security issues in pervasive environment of next generation internet of things," in *Computer Information Systems and Industrial Management*. Springer, 2013, pp. 206–217.

[13] A. Sperotto and A. Pras, "Flow-based intrusion detection," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 2011, pp. 958–963.

[14] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: from packet capture to data analysis with netflow and ipfix," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 2037–2064, 2014.

[15] M. Golling, R. Hofstede, and R. Koch, "Towards multi-layered intrusion detection in high-speed networks," in *Cyber Conflict (CyCon 2014), 2014 6th International Conference On*. IEEE, 2014, pp. 191–206.

[16] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, and K. Singh, "Performance of flow-based anomaly detection in sampled traffic," *Journal of Networks*, vol. 10, no. 9, pp. 512–520, 2016.

[17] B. Trammell and B. Claise, "Specification of the ip flow information export (ipfix) protocol for the exchange of flow information," *RFC 7011 (Internet Standard), Internet Engineering Task Force, September 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7011.txt*, 2013.

[18] T. Anantvalee and J. Wu, "A survey on intrusion detection in mobile ad hoc networks," in *Wireless Network Security*. Springer, 2007, pp. 159–180.

[19] A. Nadeem and M. P. Howarth, "A survey of manet intrusion detection & prevention approaches for network layer attacks," *IEEE communications surveys & tutorials*, vol. 15, no. 4, pp. 2027–2045, 2013.

[20] W. Zhang, Q. Yang, and Y. Geng, "A survey of anomaly detection methods in networks," in

Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on. IEEE, 2009, pp. 1–3.

[21] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 3, pp. 343–356, 2010.

[22] A. Patel, Q. Qassim, and C. Wills, "A survey of intrusion detection and prevention systems," *Information Management & Computer Security*, vol. 18, no. 4, pp. 277–290, 2010.

[23] A. Patel, M. Taghavi, K. Bakhtiyari, and J. C. JúNior, "An intrusion detection and prevention system in cloud computing: A systematic review," *Journal of network and computer applications*, vol. 36, no. 1, pp. 25–41, 2013.

[24] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 266–282, 2014.

[25] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[26] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, p. 55, 2015.

[27] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[28] A. Sperotto, R. Sadre, F. Van Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection," in *IP Operations and Management*. Springer, 2009, pp. 39–50.

[29] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and flow based network intrusion dataset," in *Contemporary Computing*. Springer, 2012, pp. 322–334.

[30] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[31] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras, "Ssh compromise detection using net-

flow/ipfix," *ACM SIGCOMM computer communication review*, vol. 44, no. 5, pp. 20–26, 2014.

[32] A. Qayyum, M. Islam, and M. Jamil, "Taxonomy of statistical based anomaly detection techniques for intrusion detection," in *Emerging Technologies, 2005. Proceedings of the IEEE Symposium on*. IEEE, 2005, pp. 270–276.

[33] M. Gyanchandani, J. Rana, and R. Yadav, "Taxonomy of anomaly based intrusion detection system: a review," *Neural Netw*, vol. 2, no. 43, pp. 1–14, 2012.

[34] N. Muraleedharan and A. Parmar, "Adrisya: a flow based anomaly detection system for slow and fast scan," *Int J Netw Security Appl*, pp. 234–245, 2010.

[35] O. Salem, A. Makke, J. Tajer, and A. Mehaoua, "Flooding attacks detection in traffic of backbone networks," in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*. IEEE, 2011, pp. 441–449.

[36] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proceedings of the 6th International COnference*. ACM, 2010, p. 8.

[37] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin, "Flow level detection and filtering of low-rate ddos," *Computer Networks*, vol. 56, no. 15, pp. 3417–3431, 2012.

[38] W. Ellens, P. Żuraniewski, A. Sperotto, H. Schotanus, M. Mandjes, and E. Meeuwissen, *Emerging Management Mechanisms for the Future Internet: 7th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security, AIMS 2013, Barcelona, Spain, June 25-28, 2013. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ch. Flow-Based Detection of DNS Tunnels, pp. 124–135.

[39] R. Hofstede, V. Bartos, A. Sperotto, and A. Pras, "Towards real-time intrusion detection for netflow and ipfix," in *Network and Service Management (CNSM), 2013 9th International Conference on*. IEEE, 2013, pp. 227–234.

[40] Y. Kanda, R. Fontugne, K. Fukuda, and T. Sugawara, "Admire: Anomaly detection method using entropy-based pca with three-step sketches," *Computer Communications*, vol. 36, no. 5, pp. 575–588, 2013.

[41] G. Fernandes, J. J. Rodrigues, and M. L. Proença, "Autonomous profile-based anomaly

detection system using principal component analysis and flow analysis," *Applied Soft Computing*, 2015.

[42] H. A. Nguyen, T. V. Nguyen, D. I. Kim, and D. Choi, "Network traffic anomalies detection and identification with flow monitoring," in *Wireless and Optical Communications Networks, 2008. WOCN'08. 5th IFIP International Conference on.* IEEE, 2008, pp. 1–5.

[43] Z. Li, Y. Gao, and Y. Chen, "Hifind: A high-speed flow-level intrusion detection approach with dos resiliency," *Computer Networks*, vol. 54, no. 8, pp. 1282–1299, 2010.

[44] R. Sadre, A. Sperotto, and A. Pras, "The effects of ddos attacks on flow monitoring applications," in *Network Operations and Management Symposium (NOMS), 2012 IEEE.* IEEE, 2012, pp. 269–277.

[45] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[46] R. Beghdad, "Critical study of neural networks in detecting intrusions," *Computers & Security*, vol. 27, no. 5, pp. 168–175, 2008.

[47] Q. A. Tran, F. Jiang, and J. Hu, "A real-time netflow-based intrusion detection system with improved bbnn and high-frequency field programmable gate arrays," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on.* IEEE, 2012, pp. 201–208.

[48] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanovic, "Flow-based anomaly intrusion detection system using two neural network stages," *Computer Science and Information Systems*, vol. 11, no. 2, pp. 601–622, 2014.

[49] Z. Jadidi, V. Muthukkumarasamy, and E. Sithirasenan, "Metaheuristic algorithms based flow anomaly detector," in *Communications (APCC), 2013 19th Asia-Pacific Conference on.* IEEE, 2013, pp. 717–722.

[50] P. Winter, E. Hermann, and M. Zeilinger, "Inductive intrusion detection in flow-based network data using one-class support vector machines," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on.* IEEE, 2011, pp. 1–5.

[51] C. Wagner, J. François, T. Engel *et al.*, "Machine learning approach for ip-flow record anomaly detection," in *NETWORKING 2011.* Springer, 2011, pp. 28–39.

[52] A. Shubair, S. Ramadass, and A. A. Altyeb, "kenfis: knn-based evolving neuro-fuzzy inference system for computer worms detection," *Journal of Intelligent and Fuzzy Systems*, vol. 26, no. 4, pp. 1893–1908, 2014.

[53] K. A. Costa, L. A. Pereira, R. Y. Nakamura, C. R. Pereira, J. P. Papa, and A. X. Falcão, "A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks," *Information Sciences*, vol. 294, pp. 95–108, 2015.

[54] Y. Li and L. Guo, "An active learning based tcm-knn algorithm for supervised network intrusion detection," *Computers & security*, vol. 26, no. 7, pp. 459–467, 2007.

[55] M.-Y. Su, "Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3492–3498, 2011.

[56] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "Cann: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13–21, 2015.

[57] S. Thaseen and C. A. Kumar, "An analysis of supervised tree based classifiers for intrusion detection system," in *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on*. IEEE, 2013, pp. 294–299.

[58] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[59] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013.

[60] P. Casas, J. Mazel, and P. Owezarski, "Coping with 0-day attacks through unsupervised network intrusion detection," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*. IEEE, 2014, pp. 24–29.

[61] P. Winter, H. Lampesberger, M. Zeilinger, and E. Hermann, "On detecting abrupt changes in network entropy time series," in *Communications and Multimedia Security*. Springer, 2011, pp. 194–205.

[62] J. François, C. Wagner, R. State, and T. Engel, "Safem: Scalable analysis of flows with

entropic measures and svm," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 510–513.

[63] P. Bereziński, J. Pawelec, M. Małowidzki, and R. Piotrowski, "Entropy-based internet traffic anomaly detection: A case study," in *Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. June 30–July 4, 2014, Brunów, Poland*. Springer, 2014, pp. 47–58.

[64] X. Qin, T. Xu, and C. Wang, "Ddos attack detection using flow entropy and clustering technique," in *2015 11th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2015, pp. 412–415.

[65] A. Satoh, Y. Nakamura, and T. Ikenaga, "A flow-based detection method for stealthy dictionary attacks against secure shell," *Journal of Information Security and Applications*, vol. 21, pp. 31–41, 2015.

[66] P. Casas, J. Mazel, and P. Owezarski, "Unada: Unsupervised network anomaly detection using sub-space outliers ranking," in *NETWORKING 2011*. Springer, 2011, pp. 40–51.

[67] P. Gogoi, D. Bhattacharyya, B. Borah, and J. K. Kalita, "Mlh-ids: a multi-level hybrid intrusion detection method," *The Computer Journal*, vol. 57, no. 4, pp. 602–623, 2014.

[68] F. Hosseinpour, P. V. Amoli, F. Farahnakian, J. Plosila, and T. Hämäläinen, "Artificial immune system based intrusion detection: Innate immunity using an unsupervised learning approach," *International Journal of Digital Content Technology and its Applications*, vol. 8, no. 5, p. 1, 2014.

[69] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "Sshcure: A flow-based ssh intrusion detection system," in *Dependable Networks and Services*. Springer, 2012, pp. 86–97.

[70] M. Vizváry, J. Vykopal *et al.*, "Flow-based detection of rdp brute-force attacks," in *Proceedings of 7th International Conference on Security and Protection of Information (SPI 2013)*, 2013.

[71] A. AlEroud and G. Karabatis, "Context infusion in semantic link networks to detect cyber-attacks: A flow-based detection approach," in *Semantic Computing (ICSC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 175–182.

[72] A. Slagell. (2016) The bro monitoring platform. [Online]. Available: https://www.bro.org/current/slides/broverview-2015.pdf

[73] C. Guo, Y.-J. Zhou, Y. Ping, S.-S. Luo, Y.-P. Lai, and Z.-K. Zhang, "Efficient intrusion detection using representative instances," *Computers & Security*, vol. 39, pp. 255–267, 2013.

[74] J. Vykopal, M. Drašar, P. Winter *et al.*, "Flow-based brute-force attack detection," *Fraunhofer Research Institution AISEC*, pp. 41–50, 2013.

[75] B. Claise, "Packet sampling (psamp) protocol specifications," *IETF*, 2009.

[76] J. Dawkins and J. Hale, "A systematic approach to multi-stage network attack analysis," in *Information Assurance Workshop, 2004. Proceedings. Second IEEE International*. IEEE, 2004, pp. 48–56.

[77] D.-h. Lee, D.-y. Kim, and J.-i. Jung, "Multi-stage intrusion detection system using hidden markov model algorithm," in *Information Science and Security, 2008. ICISS. International Conference on*. IEEE, 2008, pp. 72–77.

[78] L. P. Cordella, A. Limongiello, and C. Sansone, "Network intrusion detection by a multi-stage classification system," in *International Workshop on Multiple Classifier Systems*. Springer, 2004, pp. 324–333.

[79] P. Natesan and P. Balasubramanie, "Multi stage filter using enhanced adaboost for network intrusion detection," *International Journal of Network Security & Its Applications*, vol. 4, no. 3, p. 121, 2012.

[80] A. Alazab, M. Hobbs, J. Abawajy, and A. Khraisat, "Malware detection and prevention system based on multi-stage rules," *International Journal of Information Security and Privacy (IJISP)*, vol. 7, no. 2, pp. 29–43, 2013.

[81] Y. Waizumi, H. Tsunoda, M. Tsuji, and Y. Nemoto, "A multi-stage network anomaly detection method for improving efficiency and accuracy," *Journal of Information Security*, vol. 3, no. 01, p. 18, 2011.

[82] P. V. Amoli and T. Hamalainen, "A real time unsupervised nids for detecting unknown and encrypted network attacks in high speed network," in *Measurements and Networking Proceedings (M&N), 2013 IEEE International Workshop on*. IEEE, 2013, pp. 149–154.

[83] J.-H. Jun, D. Lee, C.-W. Ahn, and S.-H. Kim, "Ddos attack detection using flow entropy and packet sampling on huge networks," *ICN 2014*, p. 196, 2014.

[84] K. Bartos and M. Rehak, "Ifs: Intelligent flow sampling for network security–an adaptive approach," *International Journal of Network Management*, vol. 25, no. 5, pp. 263–282, 2015.

[85] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *The Knowledge Engineering Review*, vol. 29, no. 03, pp. 345–374, 2014.

[86] C. Cortes, M. Mohri, and A. Rostamizadeh, "Multi-class classification with maximum margin multiple kernel." in *ICML (3)*, 2013, pp. 46–54.

[87] V. Chang and M. Ramachandran, "Towards achieving data security with the cloud computing adoption framework," *IEEE Transactions on Services Computing*, vol. 9, no. 1, pp. 138–151, Jan 2016.

[88] G. Giacinto, R. Perdisci, M. Del Rio, and F. Roli, "Intrusion detection in computer networks by a modular ensemble of one-class classifiers," *Information Fusion*, vol. 9, no. 1, pp. 69–82, 2008.

[89] I. Kang, M. K. Jeong, and D. Kong, "A differentiated one-class classification method with applications to intrusion detection," *Expert Systems with Applications*, vol. 39, no. 4, pp. 3899–3905, 2012.

[90] P. Nader, P. Honeine, and P. Beauseroy, "Intrusion detection in scada systems using one-class classification," in *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*. IEEE, 2013, pp. 1–5.

[91] D. M. Tax, *One-class classification*. TU Delft, Delft University of Technology, 2001.

[92] H. Hoffmann, "Kernel pca for novelty detection," *Pattern Recognition*, vol. 40, no. 3, pp. 863–874, 2007.

[93] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*. ACM, 2013, pp. 8–15.

[94] W. Shang, L. Li, M. Wan, and P. Zeng, "Industrial communication intrusion detection algorithm based on improved one-class svm," in *2015 World Congress on Industrial Control Systems Security (WCICSS)*. IEEE, 2015, pp. 21–25.

[95] M. Couceiro and P. Ghamisi, "Particle swarm optimization," in *Fractional Order Darwinian Particle Swarm Optimization*. Springer, 2016, pp. 1–10.

[96] J. Yang, T. Deng, and R. Sui, "An adaptive weighted one-class svm for robust outlier detection," in *Proceedings of the 2015 Chinese Intelligent Systems Conference*. Springer, 2016, pp. 475–484.

[97] O. Mazhelis, "One-class classifiers: a review and analysis of suitability in the context of mobile-masquerader detection." *South African Computer Journal*, vol. 36, pp. 29–48, 2006.

[98] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[99] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[100] L. Li, "Support vector machines," in *Selected Applications of Convex Optimization*. Springer, 2015, pp. 17–52.

[101] D. Tax, "Ddtools, the data description toolbox for matlab," June 2015, version 2.1.2.

[102] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 5 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[103] M. Wozniak, "Hybrid classifiers–methods of data, knowledge, and classifier combination, ser," *Studies in Computational Intelligence. Springer*, vol. 519, 2014.

[104] D. Bolzoni, S. Etalle, and P. H. Hartel, "Panacea: Automating attack classification for anomaly-based network intrusion detection systems," in *Recent Advances in Intrusion Detection*. Springer, 2009, pp. 1–20.

[105] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems," *Journal of Network and Computer Applications*, vol. 40, pp. 307–324, 2014.

[106] S. Benferhat, A. Boudjelida, K. Tabia, and H. Drias, "An intrusion detection and alert correlation approach based on revising probabilistic classifiers using expert knowledge," *Applied intelligence*, vol. 38, no. 4, pp. 520–540, 2013.

[107] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011.

[108] M. Bateni, A. Baraani, and A. Ghorbani, "Using artificial immune system and fuzzy logic for alert correlation." *IJ Network Security*, vol. 15, no. 3, pp. 190–204, 2013.

[109] A. Boukhtouta, S. A. Mokhov, N.-E. Lakhdari, M. Debbabi, and J. Paquet, "Network malware classification comparison using dpi and flow packet headers," *Journal of Computer Virology and Hacking Techniques*, pp. 1–32, 2015.

[110] P. Hurtik, P. Hodakova, I. Perfilieva, M. Liberts, and J. Asmuss, "Network attack detection and classification by the f-transform," in *Fuzzy Systems (FUZZ-IEEE), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1–6.

[111] B. R. N. Yadav, B. Satyanarayana, and D. Vasumathi, "A vector space model approach for web attack classification using machine learning technique," in *Proceedings of the Second International Conference on Computer and Communication Technologies*. Springer, 2016, pp. 363–373.

[112] M. F. Umer and M. S. H. Khiyal, "Classification of textual documents using learning vector quantization," *Information Technology Journal*, vol. 6, no. 1, pp. 154–159, 2007.

[113] F. Haddadi, S. Khanchi, M. Shetabi, and V. Derhami, "Intrusion detection and attack classification using feed-forward neural network," in *Computer and Network Technology (ICCNT), 2010 Second International Conference on*. IEEE, 2010, pp. 262–266.

[114] B. Dharamkar and R. R. Singh, "Cyber-attack classification using improved ensemble technique based on support vector machine and neural network," *International Journal of Computer Applications*, vol. 103, no. 11, 2014.

[115] J. Song, H. Takakura, Y. Okabe, and K. Nakao, "Toward a more practical unsupervised anomaly detection system," *Information Sciences*, vol. 231, pp. 4–14, 2013.

[116] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," *The Elements of Statistical Learning*, pp. 1–101, 2009.

[117] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: supervised or unsupervised?" in *Image Analysis and Processing–ICIAP 2005*. Springer, 2005, pp. 50–57.

[118] Z. Ghahramani, "Unsupervised learning," in *Advanced lectures on machine learning*. Springer, 2004, pp. 72–112.

[119] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 321–352.

[120] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.

[121] X. Jin and J. Han, *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010, ch. Partitional Clustering, pp. 766–766. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-30164-8_631

[122] K. A. Nazeer and M. Sebastian, "Improving the accuracy and efficiency of the k-means clustering algorithm," in *Proceedings of the World Congress on Engineering*, vol. 1, 2009, pp. 1–3.

[123] M. M. Van Hulle, "Self-organizing maps," in *Handbook of Natural Computing*. Springer, 2012, pp. 585–622.

[124] P. Shrivastava and H. Gupta, "A review of density-based clustering in spatial data," *International Journal of Advanced Computer Research (IJACR)*, vol. 2, 2012.

[125] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[126] L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek, "Automatically determining the number of clusters in unlabeled data sets," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 21, no. 3, pp. 335–350, 2009.

[127] M. K. Pakhira, "Finding number of clusters before finding clusters," *Procedia Technology*, vol. 4, pp. 27–37, 2012.

[128] H. Alaidaros, M. Mahmuddin, and A. Al Mazari, "An overview of flow-based and packet-based intrusion detection performance in high speed networks," in *Proceedings of the International Arab Conference on Information Technology*, 2011.

[129] S. S. Khan and M. G. Madden, "A survey of recent trends in one class classification," in *Artificial Intelligence and Cognitive Science*. Springer, 2010, pp. 188–197.

[130] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & Security*, vol. 24, no. 4, pp. 295–307, 2005.

[131] M. Husák, M. Drašar *et al.*, "Flow-based monitoring of honeypots," in *7th International Conference on Security and Protection of Information (SPI 2013)*, 2013.

[132] A. Mairh, D. Barik, K. Verma, and D. Jena, "Honeypot in network security: a survey," in *Proceedings of the 2011 International Conference on Communication, Computing & Security*. ACM, 2011, pp. 600–605.

[133] M. Nassar, O. Festor *et al.*, "Labeled voip data-set for intrusion detection evaluation," in *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*. Springer, 2010, pp. 97–106.

[134] Y. Xiao, H. Wang, L. Zhang, and W. Xu, "Two methods of selecting gaussian kernel parameters for one-class svm and their application to fault detection," *Knowledge-Based Systems*, vol. 59, pp. 75–84, 2014.

[135] P. Casas, J. Mazel, and P. Owezarski, "Unsupervised network intrusion detection systems: Detecting the unknown without knowledge," *Computer Communications*, vol. 35, no. 7, pp. 772–783, 2012.

[136] M. Drašar, M. Vizváry, and J. Vykopal, "Similarity as a central approach to flow-based anomaly detection," *International Journal of Network Management*, vol. 24, no. 4, pp. 318–336, 2014.

[137] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," *Journal of Big Data*, vol. 2, no. 1, p. 3, 2015. [Online]. Available: http://dx.doi.org/10.1186/s40537-015-0013-4