A CONTROLLER-AGNOSTIC RANDOM ORACLE BASED INTRUSION DETECTION METHOD IN SOFTWARE DEFINED NETWORKS

ADNAN

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

A CONTROLLER-AGNOSTIC RANDOM ORACLE BASED INTRUSION DETECTION METHOD IN SOFTWARE DEFINED NETWORKS

ADNAN

THESIS SUBMITTED IN FULFILMENT
OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2016

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: ADNAN

Registration/Matrix No: WHA130022

Name of Degree: PH.D

Title of Thesis: A Controller-agnostic Random Oracle-based Intrusion Detection Method in Software Defined Networks

Field of Study: Information Security

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:
Designation:

ABSTRACT

The revolutionary concept of Software Defined Networks (SDNs) potentially provides flexible and well-managed next-generation networks. All the hype surrounding the SDNs is predominantly because of its centralized management functionality, the separation of the control plane from the data forwarding plane, and enabling innovation through network programmability. Such distinguishing features make SDNs flexible, vendor agnostic, programmable, cost effective, and create an innovative network environment. Despite the promising architecture, security was not considered as part of the initial SDN design. Moreover, security concerns are potentially augmented considering the logical centralization of network intelligence. The motivation of this dissertation is to address the defense space against the threat of attacks in SDNs that primarily target the control plane to wrest either full or partial control of the entire network. Additionally, this problem exacerbates in the context of SDNs unlike traditional networks. The SDN controller signifies a single point of failure and thus serves as a potential primary target for attackers. Consequently, the controller compromise in any way would certainly throw the entire network into chaos. Besides, the operational semantics of the OpenFlow mandates unmatched packets to be sent directly to the controller lower the barrier of mounting sophisticated attacks on the SDN controller. Moreover, at present, the control plane has no built-in security mechanism that prevents malicious SDN agents from sending authorized but forged flows to corrupt the controller state or bring the entire network down, in the worst case, even if the OpenFlow is Transport Layer Security (TLS) enabled. Likewise, the soft programmable switches that are directly connected to the controller running atop end host servers are attractive targets for attackers to initiate control plane flooding; apart from authorized but untrusted hosts. To preserve the correct functioning of the entire SDN architecture, an efficient detection of various distributed coordinated attacks and anomalies

triggered by large-scale malicious events that predominantly target the control plane is of paramount concern and an increasingly important research topic. As a result, developing an efficient controller-agnostic network intrusion-detection method is imperative. We propose a diverse fusion-selection approach that stands on Oracle to be applied to the classifier ensemble design, where the Oracle is a random linear function. We argue that the proposed method adds extra-diversity while promoting a higher level of intrusion-detection accuracy to effectively identify a wide variety of sophisticated network security attacks. We perform a rigorous evaluation of the proposed method by testing using Floodlight and Mininet to emulate SDN setting. We model the solution in the real setting of SDNs using High Level Petri Nets (HLPN), analyze the rules with Z language, and formally verified the correct functioning using Z3 SMT solver. To validate our proposed approach, we also carried simulation using a publicly available benchmark data-set with K-fold cross validation to exhibit the performance of the proposed method. The verification of the proposed approach is made with current state-of-the-art algorithms. Moreover, to show the resulting significant performance of the proposed approach to be optimistically unbiased, we employed a ten-fold cross-validation.

ABSTRAK

Konsep revolusioner dari Software Defined Networks (SDNs) berpotensi menyediakan rangkaian yang fleksibel dan diurus dengan baik untuk generasi akan datang. Semua gembar-gembur yang mengelilingi SDNs adalah sebahagian besarnya kerana fungsi pengurusan berpusat, pemisahan pesawat kawalan daripada data penghantaran kapal terbang, dan membolehkan inovasi melalui program menerusi rangkaian. Ciri yang membezakan tersebut membuat SDNs fleksibel, vendor agnostik, boleh aturcara, kos efektif, dan mewujudkan persekitaran rangkaian yang inovatif. Walaupun seni binanya menjanjikan masa depan yang cerah, keselamatan tidak dianggap sebagai sebahagian daripada reka bentuk awal SDN. Selain itu, masalah keselamatan berpotensi diperkukuhkan memandangkan pemusatan logik perisikan rangkaian. Motivasi disertasi ini adalah untuk menangani ruang pertahanan terhadap ancaman serangan di SDNs yang sebahagian besarnya menyasarkan pesawat kawalan untuk merampas kuasa sama ada sepenuhnya atau sebahagian daripada keseluruhan rangkaian. Selain itu, masalah ini lebih memburukkan dalam konteks SDNs tidak seperti rangkaian tradisional. Pengawal SDN menandakan satu titik kegagalan dan dengan itu berfungsi sebagai sasaran utama yang berpotensi untuk penyerang. Oleh itu, pengawal kompromi dalam apa-apa cara pasti akan membuang seluruh rangkaian ke dalam kekacauan. Selain itu, semantik operasi mandat OpenFlow paket tidak dapat ditandingi untuk dihantar terus kepada pengawal mengurangkan halangan pemasangan serangan canggih pada pengawal SDN. Selain itu, pada masa ini, kapal terbang kawalan tidak mempunyai terbina dalam mekanisme keselamatan yang menghalang ejen SDN berniat jahat daripada menghantar aliran kuasa tetapi palsu untuk merosakkan pengawal negeri atau membawa seluruh rangkaian ke bawah, dalam kes yang teruk, walaupun OpenFlow adalah Lapisan Pengangkutan Keselamatan (TLS) yang aktif. Begitu juga, suis diprogramkan lembut yang secara langsung disambungkan kepada

pengawal yang berjalan di atas pelayan hos akhir adalah sasaran menarik bagi penyerang untuk memulakan kawalan pesawat banjir; selain dari hos yang diberi kuasa tetapi tidak dipercayai. Untuk mengekalkan fungsi yang betul bagi keseluruhan seni bina SDN, pengesanan yang cekap pelbagai serangan terancang diedarkan dan anomali dicetuskan oleh peristiwa-peristiwa yang berniat jahat secara besar-besaran yang kebanyakannya mensasarkan pesawat kawalan adalah perkara yang paling utama dan topik penyelidikan yang semakin penting. Hasilnya, membangunkan kaedah pencerobohan pengesanan pelbagai dan dinamik untuk kapal terbang kawalan SDN adalah mustahak. Kami mencadangkan pendekatan pilihan pelbagai gabungan yang berdiri di atas Oracle untuk digunakan untuk reka bentuk pengelas ensemble, di mana Oracle adalah fungsi linear rawak. Kami berpendapat bahawa kaedah yang dicadangkan menambah tambahan -kepelbagaian di samping menggalakkan tahap yang lebih tinggi ketepatan pencerobohan pengesanan untuk mengenal pasti pelbagai jenis serangan keselamatan rangkaian canggih berkesan. Kami melakukan penilaian yang ketat untuk kaedah yang dicadangkan dengan menguji menggunakan lampu dan Mininet mencontohi tetapan SDN. Kami model penyelesaian dalam suasana sebenar SDNs menggunakan High Level Petri Nets (HLPN), menganalisis peraturan dengan bahasa Z, dan secara rasmi disahkan berfungsi dengan betul menggunakan Z3 SMT penyelesai. Untuk mengesahkan pendekatan yang kami cadangkan, kami juga menjalankan simulasi menggunakan umum penanda aras data-set dengan K-kali ganda pengesahan silang untuk mempamerkan prestasi kaedah yang dicadangkan. Pengesahan terhadap pendekatan yang dicadangkan dibuat dengan state-of -the -art algoritma semasa. Selain itu, untuk menunjukkan prestasi yang ketara yang terhasil daripada pendekatan yang dicadangkan itu supaya optimistik tidak berat sebelah, kita bekerja sepuluh kali ganda merentas pengesahan.

ACKNOWLEDGEMENTS

I begin by thanking my Benefactor, Allah Almighty of being indebted for His countless bounties and innumerable divine favors. I am beyond honored and deeply privileged to be grateful to Him for his richly blessings, grace, mercy, constant guidance and for always replenishing me when I need it the most throughout my career.

I would like to express my biggest gratitude to my supervisor, Prof. Abdullah, for his invaluable support and guidance throughout my Ph.D. My heartfelt appreciation goes to his commitment, encouragement, expertise, understanding, and patience that added largely to my graduate experience. I really felt thankful to have him as my supervisor and without his continued motivation; I would certainly not have considered my graduate career. I doubt that I will ever be able to convey my appreciation fully, but I owe him my eternal gratitude.

I would like to extend my special thanks to my co-supervisor Dr. Nor Badrul Anuar for his valuable assistance provided at all levels of the research project. He backed me with academic writing expertise, technical support and became more of a mentor and friend, than a supervisor. It was though his, persistence, understanding and kindness that I completed my thesis. I would also like to express my regards towards my friends and family for the support they provided me through my entire life and in particular; I must acknowledge my elder brother and best friend, Luqman Akhunzada, without whose love, constant motivation, and encouragement, I would not have a chance to pursue my dreams and make them come true.

Finally, I also recognize that this research would not have been possible without the financial assistance of the Bright Spark Unit, University of Malaya, Malaysia.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

This chapter begins with a background study to provide an overview of the research work. The chapter also presents the key motivations to value the area of research and significance of the proposed solution. Subsequently, the statement of the problem followed by the research objectives to be achieved are stated. The chapter also describes the proposed methodology of the research. Finally, the structure of the dissertation organization to show how the thesis will proceed are outlined.

The rest of the chapter is structured as follows: Section 1.1 presents the background study of the research work. Section 1.2 provides the key motivations. In Section 1.3, we present the statement of the problem followed by the statement of the research objectives in Section 1.4. Section 1.5 reports the methodology of the proposed research. Finally, we provide the layout of the thesis in section 1.6.

1.1 Background

The revolutionary concept of Software Defined Networks (SDNs) potentially provides flexible and well-managed next-generation networks. All the hype surrounding the SDNs is predominantly because of its centralized management functionality, the separation of the control plane from the data forwarding plane, and enabling innovation through network programmability. The emergence of the software-defined network (SDN) paradigm simplifies network management and enables innovation through network programmability. SDNs have given rise to radical changes in the traditional vertical integration model of a network by decoupling the forwarding hardware (data plane) from the control logic of the network (control plane). The data plane, which consists of switches and routers, is responsible only for forwarding traffic, whereas control logic and functionality are moved to an external entity known as the SDN controller. The network intelligence is logically centralized in trusted software-based SDN controllers that provide an abstract view of

underlying network resources. The abstraction of the flow broadly unifies the behavior of different SDN agents. Such distinguishing features make SDNs flexible, vendor agnostic, programmable, cost effective, and create an innovative network environment (Kreutz et al., 2015). Despite these remarkable features and the promising architecture of SDNs, market and industry observers are apprehensive about the security and dependability of SDNs. Today, the security of the SDN presents a challenge and a key concern. However, security was not considered in the initial SDN design.

The architecture of SDN poses new external and internal threats. The integrity and security of the SDN remain untested in the logical centralization of network intelligence. The entire network may be compromised through the SDN controller, which may be a single point of failure and primary target. Furthermore, SDNs are more programmable than traditional networks,thereby rendering SDNs more vulnerable in terms of security. The abstraction of available flows at the SDN controller helps significantly in harvesting the intelligence of underlying resources. This knowledge base can be used for further attacks, exploitations, and, in particular, reprogramming of the entire network. Likewise, the southbound interface of SDN can be targeted by using a diverse set of denial-of-service (DoS) and side-channel attacks.

Moreover, SDN agents can be potentially targeted and injected with false flows. Cyber-attacks launched through SDNs can result in more devastating effects than those launched through simple networks. The STRIDE threat analysis methodology demonstrates the strength and analysis of the OpenFlow (OF) protocol, which is the first viable SDN technology. However, this analysis focused on the exploitation of DoS attacks and execution of information disclosure Similarly, the lack of transport layer security (TLS) at the southbound interface can also lead to DoS attacks, rule modification, and malicious rule insertion.

Despite the fact that security was not considered as part of the initial design, each

layer/interface of the SDN has its own security implications and requirements (Kreutz et al., 2015; F. Hu, Hao, & Bao, 2014). Consequently, the SDN essentially necessitates dynamic forensic remediation and robust policy frameworks. Security must be built as part of the SDN architecture and delivered as a service to ensure the privacy and integrity of all connected resources. SDN necessitates a simple, scalable, cost-effective, and efficient secure environment.

1.2 Motivation

The revolutionary idea of Software Defined Networks (SDNs) potentially provides to redefine the future of next-generation networks. Indeed, all the hype surrounding the SDNs is predominantly because of its centralized control, the separation of the control plane from the data forwarding plane, flow abstraction and enabling innovation through network programmability. SDNs continuous to gain market traction and will continue to hold a long-term promising position in the networking industry. A report by the SDxCentral (SDxCentral, 2016) indicates that the SDN market is expected to rise from $1.5 billion in 2013 to $35.6 billion in 2018. Likewise, the International Data Corporation (IDC) (IDC, 2016) recently forecasts that the control layer/virtualization software market as a single segment of the overall SDN market will reach $2.4 billion in 2020. Moreover, the IDC also expects that the control layer/virtualization software and SDN applications will observe the fastest growth world-wide in these two software categories, which will be worth approximately $5.9 billion in 2020. Furthermore, SDN is the most rapidly evolving landscape and cloud computing is majorly driving the vast rise in SDN, which expects a market worth more than $12.5 billion in 2020. However, the market and industry observers are still apprehensive about the security of the SDNs. Moreover, security concerns are potentially augmented considering the logical centralization of network intelligence (Kreutz et al., 2015; Nunes, Mendonca, Nguyen, Obraczka, & Turletti, 2014).

Unlike traditional networks, the network intelligence is logically centralized in trusted software-based SDN controller that represents the core of the SDN architecture. SDNs are entirely dependent on the correct functioning of the controller and that must always be well-preserved for two main reasons: (i) The controller signifies a single point of failure and would always remain a hotspot and primary target for the attackers. The controller compromise in any way would certainly throw the entire network into chaos. Consequently, it might put the whole network under the control of the adversary. (ii) The controller is a central decision maker and the core of centralized network intelligence. On the contrary, traditional network attacks may also affect SDNs. However, adopting traditional defenses are not directly applicable for two major reasons: (i) Traditional network security solutions assume switches to be intelligent, whereas; in software defined networks (SDNs) switches are just dump forwarding entities; and (ii) SDNs to provide a comprehensive defense mechanism absolutely necessitates either patching the controller or essential redesign of the OpenFlow protocol (Kreutz et al., 2015).

Despite the promising architecture of SDNs, security was not considered as part of the initial design. Currently, there is no built-in control plane security mechanism to avoid sophisticated attacks triggered by large-scale malicious events that predominantly target the SDN controller to degrade the overall performance of the SDNs or bring the entire network down, in the worst case, even if the OpenFlow is Transport Layer Security (TLS) enabled. Transport Layer Security (TLS) cannot prevent malicious switches from sending authorized but forged flows to corrupt the SDN controller state. Hence, compromised end hosts can easily initiate control plane flooding to bring down the entire network. Moreover, authorized but untrusted hosts can also smoothly initiate attacks to corrupt the controller state apart from potentially malicious programmable soft switches (F. Hu et al., 2014).

1.3 Statement of The Problem

The distinguishing property of SDNs is the centralized control architecture, which results in significant managerial benefits. However, this property represents a single point of failure. The norm of SDNs is the centralized control network intelligence. Moreover, the SDN controller has a pivotal role in management, and it is a solo decision making entity, thus becoming a primary target. The logical centralization of network intelligence necessitates the correct functioning of the controller, and that must always be well-preserved. Moreover, security concerns are potentially augmented considering the centralized logical control architecture of the SDNs unlike traditional networks.

On the other hand, the operational semantics of the OpenFlow mandates unmatched packets to be sent directly to the controller lower the barrier of mounting sophisticated attacks that predominantly target the SDN controller. When a message that is received by an SDN switch with no match entry is directed to a controller by the OpenFlow protocol by default, such messages are called Packet-Ins. Manipulating Packet-Ins can easily launch diverse attacks to corrupt the controller state against all major available OpenFlow based SDN controllers. More importantly, at present, the control plane has no built-in security mechanism to avoid the manipulation of Packet-In messages.

Unlike traditional networks, soft programmable switches that are directly connected to the controller running atop end host servers are the attractive targets for the attackers. In traditional networks, the network hardware switches are relatively difficult to physically compromise and alter routing rules that govern network communication for further subsequent attacks. Hence, compromised soft programmable switches can easily initiate control plane flooding to bring down the entire network. Moreover, authorized but untrusted hosts can also smoothly initiate attacks to corrupt the controller state.

Currently, there is no built-in security mechanism that prevents malicious SDN agents

(i.e. soft switches, end hosts, etc.) from authorized but forged flows that predominantly target the SDN controller to degrade the overall performance of the SDNs or bring the entire network down, in the worst case, even if the OpenFlow is Transport Layer Security (TLS) enabled. Transport Layer Security (TLS) cannot prevent malicious soft switches from sending authorized but forged flows to corrupt the state of the controller.

In order to preserve the correct functioning of the entire SDN architecture, an efficient detection of various distributed coordinated attacks and anomalies triggered by large-scale malicious events that predominantly target the control plane is of paramount concern and an increasingly important research topic. As a result, developing a diverse, highly flexible intrusion-detection method that is capable to effectively identify a wide variety of sophisticated network attacks is imperative.

1.4 Statement of Objectives

We aim to effectively identify the threat of various distributed coordinated attacks in SDNs that primarily target the control plane by implementing a dynamic and efficient intrusion-detection method that adds extra-diversity while promote a remarkable detection accuracy with incredible low false-positive rates.

The objectives of this research are as follows.

1. To review the security vulnerabilities, attacks, and challenges of each SDN layer/interface and to establish the research gaps by analyzing the state-of-the-art SDN security techniques considering the earliest to the latest trends.

2. To investigate the defense space and formally analyze the threat of attacks that predominantly target the control plane in the real setting of SDNs.

3. To propose a diverse and highly flexible intrusion detection method capable of effectively identifying automatically and in real-time varied sophisticated network

attacks and large-scale malicious events that primarily target the control-plane of the SDNs.

4. To evaluate the proposed method by testing using Floodlight, a popular java based SDN controller and Mininet to emulate SDN setting. Moreover, modeling the proposed method in the real setting of SDNs using High Level Petri Nets (HLPN), analyze the rules with Z language, and formally verified using the Z3 constraint solver that is an efficient automated SMT (satisfiability modulo theories) solver by Microsoft Research Labs, mostly used in the verification and analysis of diverse software systems. To validate our proposed approach, we also carried simulation using a publicly available benchmark data-set with K-fold cross validation to exhibit the performance of the proposed method, where K is not a fixed parameter. The verification of the proposed approach is made with current state-of-the-art algorithms. Moreover, to show the resulting significant performance of the proposed approach to be optimistically unbiased, we employed a ten-fold (i.e., k=10) cross-validation.

1.5 Proposed Methodology

The whole research is carried-out in four main phases as shown in Figure 1.1. We studied the security implications of the entire SDN architecture with extant state-of-the-art security solutions in SDN considering the earliest to the latest trends. The security vulnerabilities, attacks, and challenges of the promising software defined network (SDN) architecture is reviewed and analyzed. A contemporary layered/interface taxonomy of the reported security vulnerabilities, attacks, and challenges of the SDN to illustrate the main categories of security implications that pertain to each SDN layer/interface is also devised. The possible threats that may affect and target a particular layer/interface alongside a suggested compact solution to help design secure SDNs is also highlighted. The

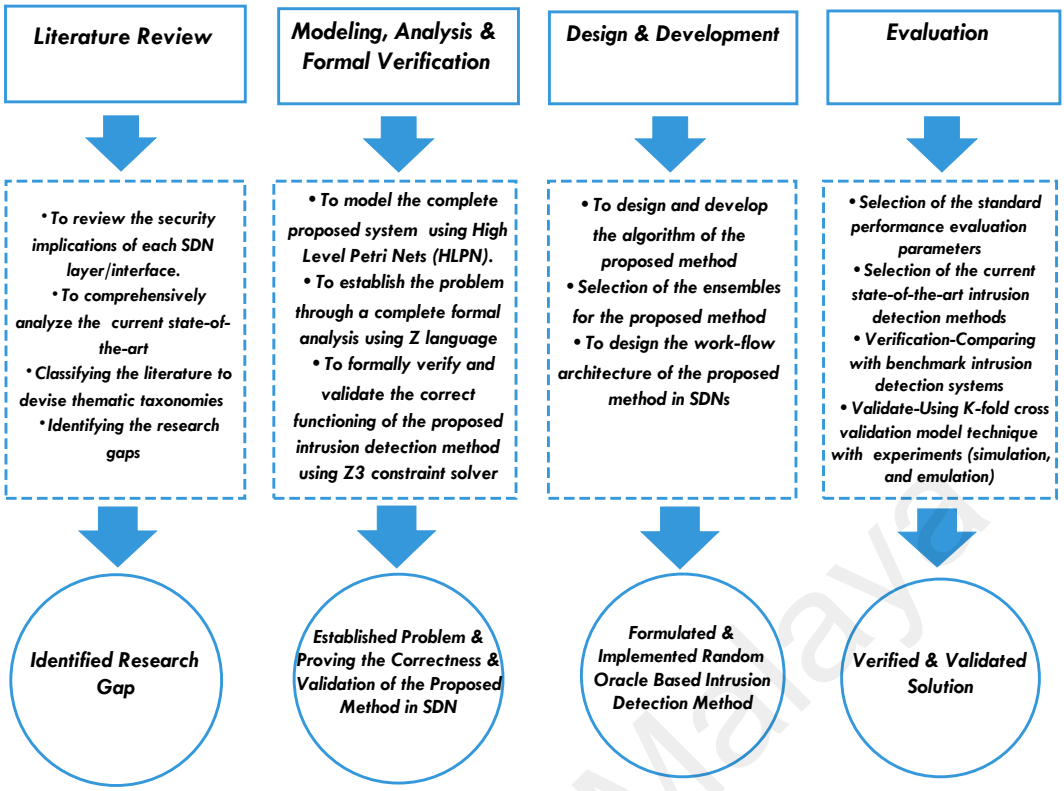

Figure 1.1: Proposed Research Methodology

extant state-of-the-art security solutions are also critically analyzed to devise a comprehensive thematic taxonomy. Moreover, we analyze each state-of-the-art security solution to identify the distinguishing SDN features utilized for each security mechanism, and the exact problem addressed by a particular technique together with the simulation or emulation environment of the corresponding technique. The distinguishing features of SDN represent the potential of emerging SDNs. Additionally, we also identify the potential effect of each state-of-the-art security solution on the corresponding SDN layers/interface. The critical discussion on the extant state-of-the-art security solutions extend the domain knowledge of the current security trends in the SDNs, the major strengths of potential SDNs, and the research gaps that need thorough investigations. We evidently noticed that security is still the key concern and is an equally striking challenge that reduces the growth of SDNs. Moreover, the deployment of novel entities and the introduction of several architectural components of SDNs poses new security threats and vulnerabilities.

The defense space against the threat of attacks in SDNs that primarily target the

control plane to wrest either full or partial control of the entire network is investigated and formally analyzed. To establish the problem, a detailed analysis is carried out in the real setting of SDNs. We carried out a formal analysis using Z language rules of the salient features that help identify diverse network attack patterns in the control plane of the SDNs. Few important OpenFlow (OF) messages such as Packet-In, Packet-Out, Flow-Mod and Flow-Removed were utilized to demonstrate that how our proposed method works identifying various network attack patterns in the real setting of SDNs. Moreover, we provide modeling of the proposed model using High Level Petri Nets (HLPN). We also formally verified the correct functioning of the proposed approach using Z3 constraint solver that is an efficient automated SMT (satisfiability modulo theories) solver by Microsoft Research Labs, mostly used in the verification and analysis of diverse software systems (Cok, Stump, & Weber, 2015). The key purpose as opposed to simulation and testing is that the system is verified by providing a formal proof on an abstract mathematical model of the system that exhaustively checks and proves the intended behavior of the proposed system. We also demonstrated the impact and analysis of attack on SDN controller and closely analyzed the detection accuracy behavior and analysis of some of the extant state-of-the-art classification based network anomaly detection systems.

A diverse fusion-selection approach that stands on Oracle to be applied to the classifier ensemble design, where the Oracle is a random linear function is proposed. We argue that the proposed method adds extra-diversity while promoting a higher level of intrusion-detection accuracy. Moreover, the approach is highly flexible and is capable to effectively detect a wide variety of sophisticated network security attacks. The method works as apparently the proposed model utilizes the tactic of the well-known divide-and-conquer strategy together-with the use of multiple random oracles. Moreover, our proposed method is different and diverse from the standard model of classifier selection (CS), whereby a single Oracle governs the whole feature space and the use of multiple

random oracles makes our proposed model diverse and unlike from the classifier fusion (CF) and the dynamic switching model (i.e. fusion-selection). Moreover, our proposed network anomaly detection system is likely to identify network attack patterns that mainly target the control plane of the SDNs without prior or having specific knowledge. We also argue that our proposed method is capable of real-time intrusion detection in the SDN environment.

The proposed method is evaluated by testing using Floodlight, a popular java based SDN controller and Mininet to emulate SDN setting. We model the proposed method in the real setting of SDNs using High Level Petri Nets (HLPN), analyze the rules with Z language, and formally verified using the Z3 constraint solver that is an efficient automated SMT (satisfiability modulo theories) solver by Microsoft Research Labs, mostly used in the verification and analysis of diverse software systems (Cok et al., 2015). To validate our proposed approach, we also carried simulation using a publicly available benchmark data-set with K-fold cross validation to exhibit the performance of the proposed method, where K is not a fixed parameter. The verification of the proposed approach is made with current state-of-the-art algorithms. Moreover, to show the resulting significant performance of the proposed approach to be optimistically unbiased, we employed a ten-fold (i.e., k=10) cross-validation.

1.6 Thesis Layout

This thesis mainly comprises of seven chapters. Every chapter of the thesis is primarily divided into three parts; Introduction-to states the key objectives; Body-to represent the relevant material; and Conclusion-to judge or evaluate the objectives to be achieved of the corresponding chapter with a linkage to the next chapter. The remainder of this dissertation is organized as follows. Figure 1.2 presents an overview of the thesis layout.

Chapter 2: Literature Review

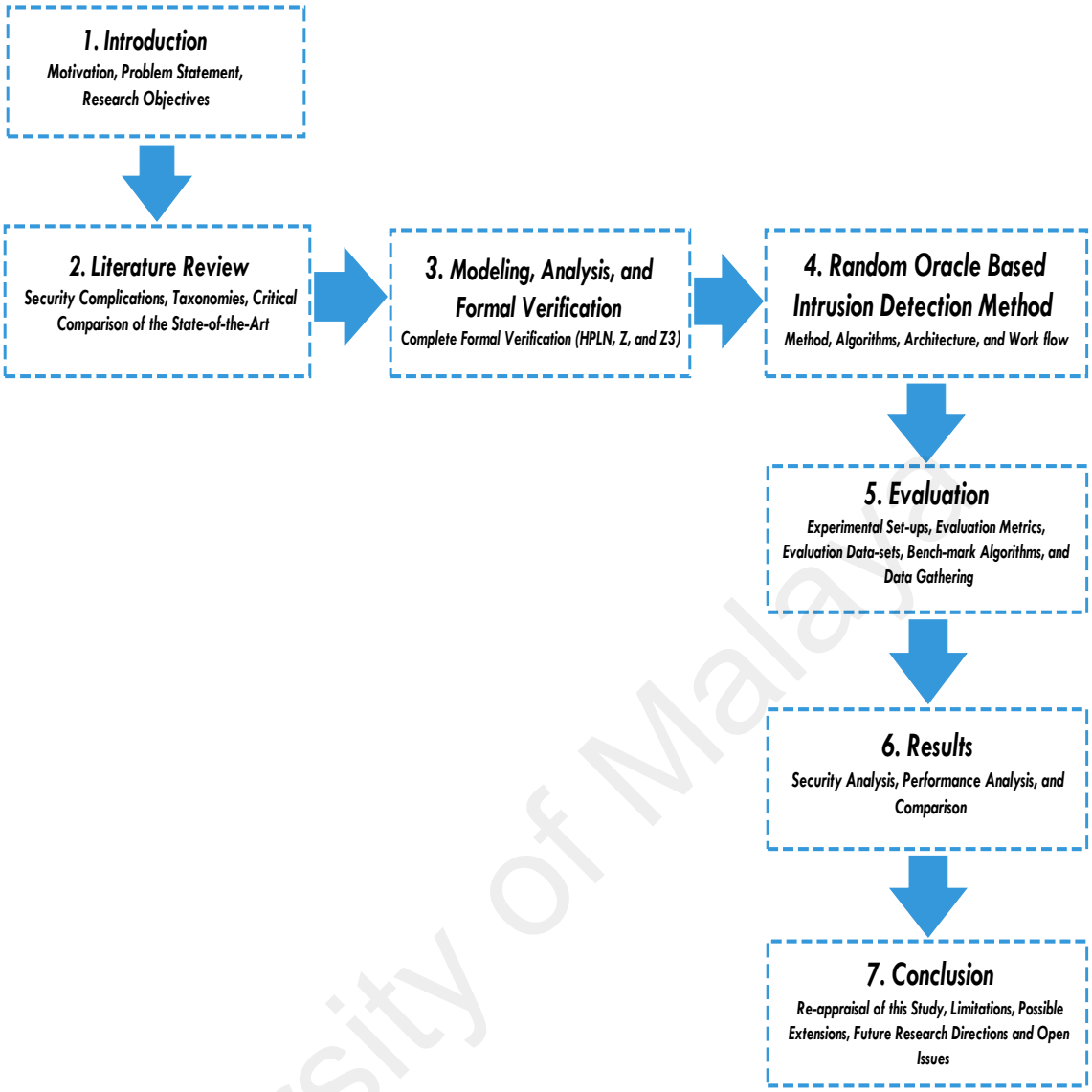

Figure 1.2: Thesis Layout

This chapter presents a thorough review and analyze the security vulnerabilities, attacks, and challenges of the promising software defined network (SDN) architecture. We devised a contemporary layered/interface taxonomy of the reported security vulnerabilities, attacks, and challenges of the SDN to illustrate the main categories of security implications that pertain to each SDN layer/interface. The chapter also presents a broad overview of the existing security implications and challenges on each SDN layer/interface. We also highlight and analyze the possible threats that may affect and target a particular layer/interface alongside a suggested compact solution to help design secure SDNs. The extant state-of-the-art security solutions in SDN considering the earliest to

the latest trends are also critically analyzed to devise a comprehensive thematic taxonomy. Moreover, the critical discussion extends the domain knowledge of the current security trends in the SDNs, the major strengths of potential SDNs, and the research gaps that need thorough investigations. The chapter clearly differentiates and presents two main schools of thought in the SDN security domain. Moreover, the chapter analyzes each state-of-the-art security solution to identify the distinguishing SDN features utilized for each security solution and the exact problem addressed by a particular technique together with the simulation or emulation environment of the corresponding technique. The distinguishing features of SDN represent the potential of emerging SDNs. Furthermore, the chapter also identifies the potential effect of each state-of-the-art security solution on the corresponding SDN layers/interface. Finally, We advocate the development of secure and dependable SDNs by presenting potential security requirements and their key enablers.

Chapter 3: Problem Analysis

The chapter presents a detailed problem analysis in the real setting of SDNs. We demonstrated various possible ways of sophisticated network security attacks to target the control plane of from within the SDNs (i.e. close model). As opposed to simulation and testing is that the system is verified by providing a formal proof on an abstract mathematical model of the system that exhaustively checks and proves the intended behavior of the system. Additionally, the chapter provides a critical analysis of the specified sophisticated attacks on the control plane of the SDN even if the control channel (southbound API) is TLS (Transport Layer Security) enabled. The chapter also briefly discusses the preliminaries to model and analyze a system. Moreover, verification of the system and results is provided. Finally, the chapter demonstrates briefly the impact and analysis of attack on the control plane and closely analyzes the detection accuracy behavior and analysis of some of the extant state-of-the-art classification based network anomaly detection

systems.

Chapter 4: A Random Oracle Based Intrusion Detection Method

The chapter elaborates our proposed diverse random oracle based intrusion-detection method to accurately identify large-scale malicious events that predominantly target the control plane to degrade the overall performance of the Software Defined Networks (SDNs) or bring the entire network down in the worst case. The chapter clearly elaborates our proposed diverse fusion-selection method that stands on random oracle and shows its diverse nature from the extant main approaches used to design classifier ensembles. The chapter provides a comprehensive understanding of the random linear oracle (RLO), which we consider the main contribution of our proposed solution followed by a detailed description of the RLO training and prediction phase algorithms. The chosen classifier ensemble employed for the proposed method. Moreover, the chapter also addresses the reason of the employed chosen classifier ensemble. The chapter also addresses the significance of the proposed solution and gives justification that why our proposed method works. Moreover, the work-flow of the proposed solution is also demonstrated in the real setting of the Software Defined Networks (SDNs). Moreover, we also reason out that the proposed solution is acceptable to be deployed for real applicable scenarios of diverse distributed environments and in particular, for securing the control plane of the software defined networks. The distinguishing features of programmability make SDNs flexible, and vendor agnostic. Once you design a module being part of the OpenFlow protocol, it is applicable to most of the commercial controller since OpenFlow is a de-facto standard southbound API of the SDNs. Consequently, our proposed solution becomes controller independent, which means controller agnostic. Finally, a summary and concluding remarks of the chapter are also provided.

Chapter 5: Evaluation

This chapter elaborates the evaluation of the proposed diverse fusion-selection intru-

sion detection method in terms of the over-all performance and effectiveness. The chapter also explains the tools, and the bench-mark data-set used for the complete evaluation of the proposed random linear oracle (RLO) model. The set-up environments, the pre-requisites, and the programming tools and language used for the final implementation. Moreover, the standard parameter used for assessing the performance of the proposed model with a model validation technique is also provided.

Chapter 6: Results and Discussions

The chapter presents the results of the experimental research (i.e. simulation, and emulation) of the proposed intrusion-detection method. The conducted experimental research demonstrates the promising results and better performance of the proposed intrusion detection method in identifying varied sophisticated network attacks. The main objective of the chapter is to demonstrate the outstanding performance of the random oracle based proposed intrusion detection method compared to the state-of-the-art classifier ensembles in this domain. Moreover, to verify the out-performance of the random oracle based intrusion-detection method compared to state-of-the-art base-classifiers in this domain. Additionally, to demonstrate that the proposed method is viable to identify diverse attacks triggered by large-scale malicious events in the control-plane of the SDNs in real-time.

Chapter 7: Conclusion

The Chapter concludes the dissertation by reporting on the re-examination of the objectives of the research. It summarizes the main findings of study, highlights the significance of the proposed method. The possible future extensions and limitations of the proposed work. Finally, in an effort to anticipate secure and dependable SDNs, the chapter presents the state-of-the-art security trends and cutting-edge future research directions to be tackled by young researchers and professional around the globe.

CHAPTER 2: LITERATURE REVIEW

The chapter begins with a simplified overview of the complete Software Defined Networks (SDNs) architecture alongside the underlying fundamental concepts is provided to help readers gain an easy and smooth understanding of the SDNs. This chapter reviews the security vulnerabilities, attacks, and challenges of the promising SDN architecture. A contemporary taxonomy of the reported security vulnerabilities, attacks, and challenges is devised to illustrate the main categories of security implications that pertain to each layer/interface of the SDN. Moreover, the chapter thoroughly analyzes the possible threats that may affect and target a particular layer/interface alongside a suggested compact solution to help design secure SDN.

The ensuing chapter presents the state-of-the-art security solutions in SDN considering the earliest to the latest trends. The main categorization of solutions is followed by a critical analysis and discussion on devising a comprehensive thematic taxonomy. We analyzes each state-of-the-art security solution to identify the distinguishing SDN features utilized for each security solution and the exact problem addressed by a particular technique together with the simulation or emulation environment of the corresponding technique. The distinguishing features of SDN represent the potential of emerging SDNs. The critical discussion extends the knowledge of the domain of the current security trends in the SDNs, the major strengths of potential SDNs, and the research gaps that need thorough investigations. Moreover, the analysis carried out clearly differentiates and presents two main schools of thought in the SDN security domain. We also advocate the production of secure and dependable SDNs by presenting potential requirements and key enablers.

The remainder of this chapter is structured as follows: Section 2.2 introduces a simplified overview of the SDN architecture. In Section 2.3, the paper provides a broad overview

of the security implications of each SDN layer/interface. We further classify broadly the reported security vulnerabilities, attacks, and challenges of SDN by devising a thematic layered/interface-based taxonomy. The possible threats that may affect and target a particular layer/interface with the suggested compact solution are highlighted and analyzed in Section 2.4. Section 2.5 presents state-of-the-art security solutions followed by a critical discussion to present a thematic classification. Moreover, the section contains the main categorization of state-of-the-art security mechanisms and identifies their potential effect on each SDN layer/interface. Furthermore, the section presents the employed approach to identify the distinguishing features of SDNs, addressed the exact problem with their implementation environment, and illustrates the two main schools of thought. Section 2.6 discusses the requirements and key enablers for dependable and secure SDNs. Finally, we provide the concluding remarks in Section 2.7.

2.1 A Comprehensive Overview of the SDN Architecture

The SDN architecture consists mainly of three planes, namely, application plane, control plane, and data plane, with their corresponding application programming interfaces (APIs) (Jarraya, Madi, & Debbabi, 2014; F. Hu et al., 2014; Lara, Kolasani, & Ramamurthy, 2014). Figure 2.1 depicts a simplified view of the SDN architecture, which is explained by using a top–down approach. The simplified overview is provided to help the readers gain an easy and smooth understanding of the analysis of the SDNs security vulnerabilities, attacks, challenges, and state-of-the-art solutions.

2.1.1 Application Plane

The application plane is also known as the application layer, which is responsible for providing a set of services and applications, such as intrusion detection system (IDS), intrusion prevention system (IPS), deep packet inspection, load balancers, security monitoring, and access controls (Nunes et al., 2014; Jarraya et al., 2014). The SDN applications are

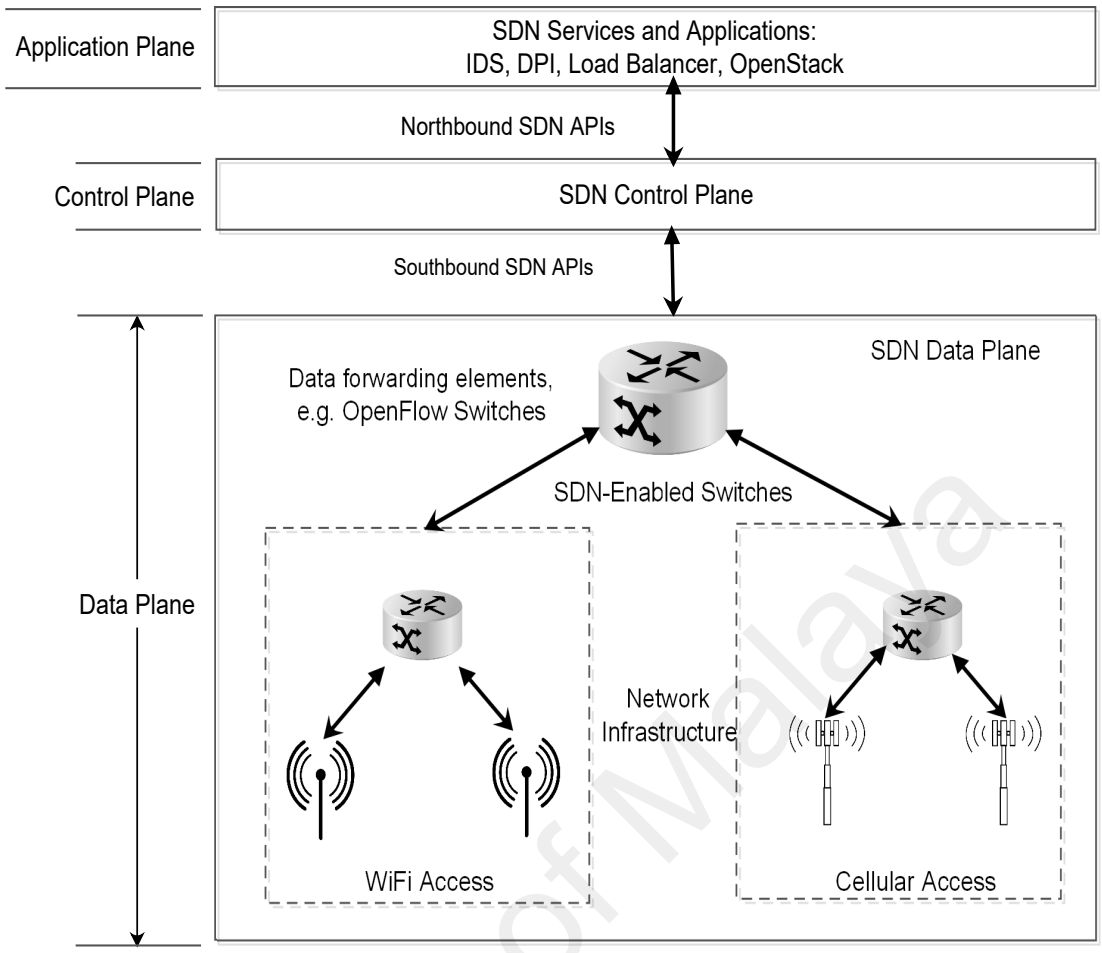

Figure 2.1: A Simplified View of the SDN Architecture

basically programs that directly, explicitly, and programmatically share the anticipated network behavior and requirements with the SDN controller via northbound APIs. The applications and services can extract information with regard to the policy or behavior of the underlying architecture of SDNs. Furthermore, application-to-control plane communication is carried out because of various reasons (Kreutz et al., 2015; Nunes et al., 2014).

2.1.2 Northbound SDN Interface

To support application or service orchestration, automation, and innovation, the SDNs employ open APIs, which are commonly known as northbound APIs. The northbound interface enables the application-to-control plane communication, and it is also recognized as the controller–service communication interface. Moreover, the interface facili-

tates in providing the abstract view of the underlying network. Likewise, the northbound APIs empower the direct expression of network behavior and requirements. However, the northbound SDN interface/APIs are more likely implemented on an ad hoc basis because no standard northbound SDN interface/APIs exist at present (Jarraya et al., 2014).

2.1.3 Control Plane

The control plane is also referred to as the control layer of the SDN (Jarraya et al., 2014). The control plane includes a special network component called the SDN controller, which is logically centralized but physically distributed in principle (F. Hu et al., 2014; Kreutz et al., 2015) . The controller is a software platform that is responsible for establishing and terminating flows and paths within SDNs. The SDN controller provides programmatic interfaces to the underlying network. The overall management functionality of SDN is simply entrusted in the SDN controller while it facilitates the programmability of the entire network. Likewise, the control layer also provides an abstraction of the underlying resources (Lara et al., 2014). Moreover, the SDN centralized logical control model can be applied to a wide variety of applications, underlying networks, and physical media, such as wired (e.g., Ethernet), wireless (e.g., 802.11 and 802.16), and optical networks (Yang et al., 2015). Some popular SDN controllers and their corresponding brief descriptions are shown in Table 2.1.

2.1.4 Southbound Interface/APIs

To support the overall programmatic control of the forwarding plane, event notification, capability advertisement, and statistics reporting, the SDN uses southbound interface/APIs (Farhady, Lee, & Nakao, 2015). The southbound interface/APIs provide a link between the control layer (control plane) and the infrastructure layer (data plane). Particularly, the southbound SDN interface/APIs enable communication between a controller and a switch. Thus, the interface is also known as a controller–switch communication

Table 2.1: The SDN Controllers and their Corresponding Brief Description

SDN Controllers	Open Source	Language	Description
NOX	Yes	C++ - Python	NOX is the first controller written in C++ / Python to support fast, asynchronous IO
POX	Yes	Python	Written in Python. Performs well as compared to NOX applications (especially when run under PyPy)
Maestro	Yes	Java	Maestro provides the *view* abstraction for grouping related network state into a subset
Floodlight	Yes	Java	Floodlight can manage both OpenFlow and non-OpenFlow networks
Beacon	Yes	Java	Beacon is a cross-platform, modular, Java-based controller that supports both event-based and threaded operation
OpenDayLight	Yes	Java	It uses OSGi framework and provide REST API having weak consistency.
Trema	Yes	Ruby/C	Trema is a full-stack, programming framework that allows users to develop and test OpenFlow controllers on a laptop
RouteFlow	Yes	C++	It is a special purpose controller and provides virtualized IP routing over OpenFlow hardware
Ryu	Yes	Python	It supports OpenFlow from version 1.0 to version 1.3 and integrates with Open-Stack, building virtual network without using VLAN
FlowVisor	Yes	C	It is a special purpose controller for OpenFlow network virtualization.
SNAC	No	C++	A NOX-0.4 based controller to manage the network, configure devices and monitor different events
Helios	No	C	It provides a programmatic shell for performing integrated experiments
ONOS	Yes	Java	Building networks for service provider with performance, scale-out design, and high availability

interface. The interface assists the administrators in handling traffic of the underlying switching hardware of the data plane by pushing out controller decisions.

Currently, OpenFlow (OF) is the most popular and common southbound interface. People consider OF and SDN synonymous, although this is a misconception. In reality, OpenFlow represents a part of the entire SDN architecture (Lara et al., 2014). OF is a control-to-data plane communication protocol, and it is not the only existing protocol. Some SDN proprietary southbound protocols include Cisco's Open Network Environment Platform Kit and Juniper's Contrail (Jarraya et al., 2014).

2.1.5 Data Plane

The data plane is composed of underlying network infrastructures and is also known as the infrastructure layer of the SDN (Jarraya et al., 2014). The data plane consists of forwarding hardware, such as switches and routers. The control function is entrusted to the controller; thus, the underlying hardware, such as switches and routers, is responsible only for data forwarding and is also acknowledged as the forwarding plane of the SDN (F. Hu et al., 2014). The data plane implements the management functionality of the

controller through SDN-enabled switches. Subsequently, the SDN-enabled switches are used to forward data, collect network information, and send the information back to the control plane via southbound interfaces (Govindarajan, Meng, & Ong, 2013).

2.2 SDN Security Vulnerabilities, Attacks and Challenges: A Layered/Interface Taxonomy

The section presents a comprehensive overview of each SDN layer/interface security vulnerabilities, attacks, and challenges. We further broadly classify the reported security vulnerabilities, attacks, and challenges of SDN by devising a thematic taxonomy based on each SDN layer/interface. The layered/interface taxonomy of SDN security vulnerabilities, attacks, and challenges is depicted in Figure 2.2. The taxonomy clearly illustrates the main categories of security implications of each layer/interface.

2.2.1 Application Plane Security Vulnerabilities, Attacks and Challenges

Controlling the network by using software is the principal property of SDNs. Thus, most implemented and deployed applications of SDN represent diverse network functions and can access the underlying network resources under certain privileges (Wen, Chen, Hu, Shi, & Wang, 2013). SDN applications have many advantages, and yet they cause serious security challenges. This section briefly yet extensively explores the application plane-related security vulnerabilities, attacks, and challenges, which are classified into eight main categories, namely, (1) nested applications, (2) applications abusing SDN internal storage, (3) applications abusing SDN control messages, (4) trust establishment, (5) third-party applications and open development environments, (6) authentication, authorization, and accountability, (7) exhaustion of resources, and (8) application executing system commands.

1. Nested Applications:Nested applications present a real challenge to deal with and are vulnerable to the following reported exploitations (Monsanto, Reich, Foster,

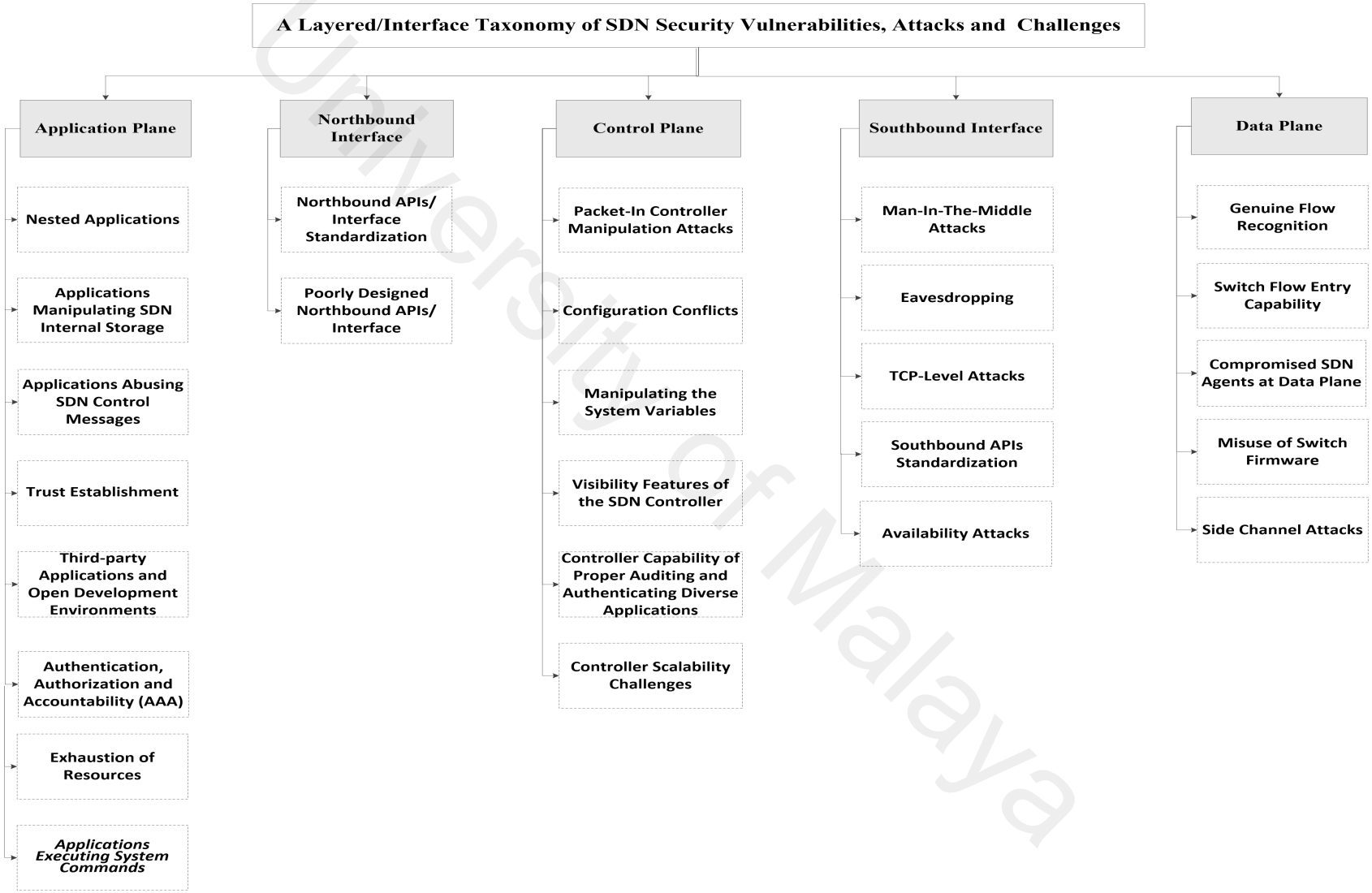

Figure 2.2: A Layered/Interface Taxonomy of SDN Security Vulnerabilities, Attacks and Challenges

Rexford, & Walker, 2013; H. Xie, Tsou, Lopez, Yin, & Gurbani, 2012).

a) Service Chain Interference: Applications with chained execution may cause serious interference and security challenges. For instance, malign applications that participate in a service chain can drop control messages before the awaited applications, thus causing extreme interference. Moreover, interference may occur when a malicious application falls in an infinite loop to stop applications with chained execution.

b) Gateway to Unauthorized Access: A malevolent nested application can sidestep the access control by issuing the instance of another class application and can be a gateway to unauthorized access.

2. Applications Manipulating SDN Internal Storage: The application in the application plane receives certain privileges to access the underlying resources; thus, the SDN controller shares internal storage among various SDN applications (Wen et al., 2013). Eventually, applications can access and manipulate the internal database of an SDN controller, which can further be used for many subsequent attacks, such as manipulating network behavior (Shin et al., 2014).

3. Applications Abusing SDN Control Messages: A control message is responsible for the two-way communication between the data plane and application plane. An arbitrarily issued control message of SDN by an application may lead to the following attacks (Dover, 2013).

a) A malicious application that overwrites an existing flow rule in the controller switch flow table may lead to unexpected network behavior; this phenomenon is known as flow rule modification.

b) A malicious application may block all communication by issuing a control message that clears the flow table entries of an SDN switch.

4. Trust Establishment: To establish trust between the SDN applications and the controller, compelling trust mechanisms must be present (Kreutz, Ramos, & Verissimo, 2013). An application server that stores sensitive user information can be compromised, and legitimate user credentials can subsequently be used to add forged but authorized flows to the network. Mechanisms to certify network devices exist; however, compelling mechanisms to establish trust to certify network applications do not exist. Moreover, the centralized control architecture of SDNs necessitates a centralized system to certify the multitude and diversity of network applications and presents an interesting area that is yet to be explored.

5. Third-party Applications and Open Development Environments: Third-party applications could also result in serious security vulnerabilities and challenges because of the lack of standard and consensus-based development environments, programming models, and paradigms, and the variety of vendors Use Cases for ALTO with Software Defined Networks. Importantly, third-party applications could cause serious issues of interoperability and collision in security policies. Moreover, dealing properly with the diversity and multitude of third-party applications and non-standard open software development environments is challenging.

6. Authentication, Authorization and Accountability (AAA): Authentication of nested applications is a major challenge in programmable networks, and the diversity of third-party applications makes this situation difficult. Authorization-related attacks can lead to illegal access to the controller, thereby affecting the lower three corresponding layers/interface (Kreutz et al., 2013). In-authentic applications can damage the application layer, northbound interface, and control layer. No compelling

authorization mechanisms exist for such application in the centralized control architecture of SDN with open software development environments. Likewise, accountability is another real challenge considering the various third-party and nested applications responsible for the consumption of network resources (Kreutz et al., 2013; H. Xie et al., 2012).

7. Exhaustion of Resources: Malicious applications can exclusively contribute to exhaust all the available system resources and seriously affect the performance of other applications, including the SDN controller. Such attacks have been verified (Wen et al., 2013).

    a) Memory consumption: A malicious application may be involved in continuous consumption of system memory or in memory allocation to exhaust all the available system memory.

    b) CPU consumption: A malicious application can seriously exhaust all the available CPU resources by simply creating useless working threads.

    c) A malicious application can execute a system exit command to dismiss the controller instance.

8. Application Executing System Commands: A malicious SDN application is capable of terminating the controller instance by executing a system exit command. The attack was demonstrated previously (Shin et al., 2014).

Critical Remarks: The security vulnerabilities, attacks, and challenges discussed above are critical and can target all the corresponding SDN layers/interface. Protection against diverse malign applications will remain a challenge for SDNs (Zhang, 2013). No compelling mechanism for distinguishing user or third-party or network service appli-

cations exists, and the accountability and access control of nested applications have not been demonstrated.

2.2.2 Northbound APIs/interface Security Vulnerabilities, Attacks and Challenges

Application plane has direct implications on the overall underlying SDN architecture. Hence, the northbound APIs are considered highly important targets for exploitation. The reported security vulnerabilities, attacks, and challenges of the northbound interface are stated below. Two main challenges cause the vulnerability of northbound APIs.

1. Northbound APIs/Interface Standardization: No standard northbound API exists, and working with various northbound open APIs is challenging (Nadeau & Pan, 2011). Moreover, open independent development environments without standard specifications are faced with increased risk from various security challenges posed by skilled adversaries.

2. Poorly Designed Northbound APIs/ Interface: A poorly designed northbound interface can be misused easily by SDN applications to manipulate the behavior of other applications (Kreutz et al., 2013). For instance, an SDN application may exploit a poorly designed northbound API to evict an ongoing application session. Moreover, an SDN application may use a poorly designed northbound interface to randomly unsubscribe a target application, thereby rendering it incapable of obtaining important subscribed control messages that can be easily carried out by the unsubscription of an event listener.

2.2.3 Control Plane Security Vulnerabilities, Attacks and Challenges

The distinguishing property of SDNs is centralized control architecture, which results in significant managerial benefits. However, this property represents a single point of failure. The norm of SDNs is the centralized control network intelligence. Moreover,

the SDN controller has a pivotal role in management, and it is a solo decision making entity, thus becoming a primary target. Furthermore, the visibility features of the SDN controller also pose serious security challenges. The section elaborates control plane-related security vulnerabilities, attacks, and challenges.

1. Packet-In Controller Manipulation Attacks: A packet-in essentially represents a packet that does not match any flow rules at the data plane, and the OF protocol mandates that such packets must be sent by the switch to the controller directly. When a message that is received by an SDN switch with no match entry is directed to a controller by the OF protocol by default, such messages are called packet-ins. At present, the control plane has no built-in security mechanism to avoid the manipulation of packet-in messages even if the OF is TLS enabled. Authorized switches can also send forged packet-in messages that can subsequently be used to corrupt the controller state by the following practical attacks that may occur on many SDN controllers.

    a) Service Chain Interference: Applications with chained execution may cause serious interference and security challenges. For instance, malign applications that participate in a service chain can drop control messages before the awaited applications, thus causing extreme interference. Moreover, interference may occur when a malicious application falls in an infinite loop to stop applications with chained execution.

    b) Directed DoS Attacks: An attacker may place the controller in an unpredictable state by flooding a target SDN controller with packet-ins; this attack was demonstrated in a previous work (Dhawan, Poddar, Mahajan, & Mann, 2015). One of the practical forms of launching a directed DoS attack is packet-in flooding, which places the SDN controller in an unpredictable

condition. Major SDN controllers are still a target of this attack. DoS attacks using packet-ins may be carried out in many ways. A DoS attack is a serious concern in the centralized control architecture of SDNs.

c) Poisoning the View of the Controller: An address resolution protocol (ARP) packet relayed as a packet-in message can be forged to adversely affect the view of the controller; this attack was demonstrated in a previous work (Hong, Xu, Wang, & Gu, 2015).

d) Poisoning the Network Topologies and Traffic Hijacking: Manipulating the link discovery service relayed as packet-in messages can be utilized to create fake network topologies. Moreover, the vulnerability of the host tracking service can be exploited to establish traffic hijacking.

e) Fabricated Links Creation: An LLDP packet relayed as a packet-in message may also be forged to create a fabricated link.

f) Side-channel Attacks: Manipulating packet-ins can also leverage side-channel attacks to obtain sensitive information.

g) Other Challenges and Vulnerabilities of Control Messages: Control messages can also be manipulated to corrupt the state of the controller in many ways. Some practical attacks and vulnerabilities of control messages include spoofing a target switch and the attack, and switch-table flooding attack using Data Path ID to place the controller in an unpredictable state.

2. Configuration Conflicts: The control plane enforces a network-wide policy. However, single-domain multiple SDN controllers, multi-tenant SDN controllers, and multiple OF architectures may lead to serious configuration conflicts and inter-federated configuration conflicts (Al-Shaer & Al-Haj, 2010). Moreover, stateful applications may not work properly because the controller may not be able to syn-

chronize the network updates, and it does not have the capability to recall past events.

3. Manipulating the System Variables: Manipulating a system variable may also corrupt the controller state. For instance, a system time alteration by an attacker may turn the controller practically off from linked switches.

4. Visibility Features of the SDN Controller: The visibility features of SDN can also be vulnerable to harvesting of network intelligence of the underlying architecture for further exploitation.

5. Controller Capability of Proper Auditing and Authenticating Diverse Applications: The security of the control plane is normally measured and challenged in terms of controller capabilities. For instance, a real challenge is enabling the controller to properly facilitate the authentication and authorization of network resources consumed by applications implemented on top of the control plane with appropriate tracking, auditing, and isolation (Hartman, Wasserman, & Zhang, 2013; Sezer et al., 2013).

6. Controller Scalability Challenges: The controller is the sole entity responsible for centralized decision making. Controllers nowadays become a bottleneck in a 10 Gbps link high-speed network. Moreover, the lack of scalability causes the following serious problems detailed in the literature (Sezer et al., 2013).

    a) Saturation attacks

    b) Experience delay constraints

    c) A single point of failure

2.2.4 Southbound API/Interface Security Vulnerabilities, Attacks and Challenges

Separating the control plane and data plane results in serious disasters; more importantly, control plane security has direct implications on the data plane (Kreutz et al., 2013). Thus, the controller–switch communication channel remains a favorable choice for attackers. The specification of Open-Flow (OF) datagram transport layer security and TLS for channel security is optional in the latest versions of OF. However, TLS is not secure from TCP-level security attacks (Liyanage & Gurtov, 2012) and is not reliable (Kreutz et al., 2013) in many cases. Subsequently, the diverse attacks described below occur.

1. Man-in-the-Middle Attacks: The southbound interface provides an opportunity to actively control the control channel, thereby causing man-in-the-middle attacks. In this type of attack, a skilled attacker modifies the control messages exchanged between the control plane and the data plane, such as flow rule messages, to corrupt network behavior (Benton, Camp, & Small, 2013).

2. Eavesdropping: The southbound interface can be targeted for both active and passive eavesdropping. For instance, an opponent may steal highly sensitive information by sniffing the control channel, i.e., learning about the advertised network topologies by sniffing ongoing control messages.

3. TCP-level Attacks: The southbound APIs can be targeted for TCP-level attacks because the optional TLS does not implement TCP-level protection; the attacks were demonstrated in a previous study (Liyanage & Gurtov, 2012).

4. Southbound API Standardization: Unlike the northbound APIs, no standard southbound API exists, which presents a real challenge for the SDN community. A compelling and consensus-based southbound interface is needed to address the security vulnerabilities, attacks, and challenges of the interface thoroughly and uniformly.

5. Availability Attacks: Availability-related attacks refer to DoS and distributed denial of service (DDoS) attacks. The south-bound interface can be targeted by communication flooding attacks, which specifically affect the control layer, the control data interface, and the data layer (Scott-Hayward, O'Callaghan, & Sezer, 2013; Benton et al., 2013).

2.2.5 Data Plane Security Vulnerabilities, Attacks and Challenges

The SDN switches and routers are dump forwarding devices. The capability of controller decisions is based on these data plane forwarding entities. This section explains some of the foremost data plane-specific security vulnerabilities, attacks, and challenges.

1. Genuine Flow Recognition: The controller decisions are based on the SDN switch flow rules. Hence, one of the key challenges for the controller is recognizing and differentiating whether the switch-generated flow rules are candid or fraudulent. A compromised switch that generates malicious flow rules can render the data plane practically offline.

2. Switch Flow Entry Capability: An SDN switch has a limited ability to maintain the number of flow entries, and an SDN switch can practically be targeted with saturation attacks. These attacks subsequently lead to switch DoS attacks that render the data plane in an unpredictable state. For instance, an attacker can simply install or flood a large number of flow entries to exhaust the switch-limited resources. Flow rule flooding was demonstrated in the literature (Sherwood et al., 2009; Monsanto et al., 2013; Shin et al., 2014).

3. Compromised SDN Agents at Data Plane: A compromised host or SDN switch can contribute to a variety of attacks. For instance, an attacker can fill up the compromised target switch flow table to advertise fake topologies (logical or physical) or

render the controller in an unpredictable state. These attacks were demonstrated in previous works (Farhady et al., 2015). Compromised SDN agents can also lead to a variety of man-at-the-end (MATE) attacks. Furthermore, a manipulated control message by an attacker can render the data plane in an unpredictable state and disconnect the control plane from the data plane. A compromised host or SDN switch may also trigger dynamic attacks, such as traffic rerouting, traffic hijacking, and network DoS attacks.

4. Misuse of Switch Firmware: A previous study indicated that certain SDN switch hardware tables cannot process crafted flow rules .

5. Side-channel Attacks: The data plane can be attacked with side-channel attacks. For example, an input buffer can be used to identify flow rules, and analyzing the packet processing time may determine the forwarding policy (Scott-Hayward et al., 2013).

2.3 Possible Security Threats Affecting each SDN Layer/Interface

This section presents and analyzes the possible security threats within SDNs. Table 2.2 illustrates the analysis on a particular security threat that affects each SDN layer/interface. Security threats that affect corresponding layers/interface must be addressed in emerging SDNs. The table also presents the protection mechanisms, security requirements with their corresponding affected functionalities, and SDN layers/interface against each possible threat. Operating system (OS) alteration represents the destruction or alteration of entire SDN elements, such as controllers (Li, Raghunathan, & Jha, 2009). A threat can be mitigated by ensuring that system integrity is protected by implementing trusted computing. OS alteration can target all the layers of SDN and affect the management of running services and applications.

Software framework alteration identifies the destruction or alteration of a middleware or its components. Similar to securing against OS alteration, software framework alteration can be prevented by ensuring system integrity while implementing trust computing (Baldini et al., 2012). Furthermore, software framework alteration can affect all the layers of SDNs. Likewise, the software failure threat represents a general failure in any of the components that constitute the software framework, application, and operating system. Software failure can be mitigated by employing high-assurance techniques while ensuring the robustness of a system. Unlike the software framework threat, software failure can affect all layers of SDNs.

Hardware failure represents generic failure of a hardware in any component (Baldini et al., 2012). Similar to software failure, hardware failure can be reduced by employing high-assurance techniques while ensuring the robustness of a system. The entire set of functionalities is affected by the hardware failure threat that targets the control and data layer.

The configuration data alteration threat represents the destruction or alteration of configuration data that are required by SDNs to perform different functions. Configuration data can be removed or modified. The threat can be mitigated by ensuring data integrity in the SDN middleware. Furthermore, the threat can target the control layer, control–data interface, and data layer while possibly affecting the resource and application management functionalities. Similarly, the configuration data extraction threat is an eavesdropping threat (Garnaev & Trappe, 2013) where the attacker gathers configuration information that can be utilized in subsequent attacks. The threat can target the control layer, control–data interface, and data layer. Unauthorized access to SDN services is identified as a security breach. The threat can be mitigated by deploying secure administration modules while ensuring system integrity. The security requirements for mitigating the threat are identification and verification. However, the threat can affect the entire set

Table 2.2: Possible Security Threats Affecting Each Layer/Interface of SDNs

Possible Security Threats	Protection Techniques	Security Requirements	Affected Functionalities	Application Layer	Northbound Interface	Control Layer	Southbound Interface	Data Layer
Operating System Alteration	trusted computing	system integrity protection	application management	✓		✓		✓
Software Framework Alteration	trusted computing	system integrity protection	application management	✓		✓		✓
Software Failure	high assurance	robustness, system integrity protection	all functionalities	✓		✓		✓
Hardware Failure	high assurance	robustness, system integrity protection	all functionalities			✓		✓
Configuration Data Alteration	data integrity functionality in SDN middleware	data integrity protection	resource management, application management.			✓	✓	✓
Configuration Data Extraction	data integrity functionality in SDN	confidentiality protection	data management			✓	✓	✓
Unauthorized Access to SDN Services	deploying secure administration module	identities verification, ensuring system integrity	all functionalities	✓	✓	✓	✓	✓
User Data Alteration	data integrity functionality in SDN	ensuring data integrity	data management					✓
Masquerading as Authorized SDN Controller	use of digital signatures for SDN software modules	ensuring system integrity, identities verification, accountability	application management			✓	✓	✓

of functionalities and target all the layers/interfaces of SDN.

User data alteration threat involves the destruction or alteration of user data, such as customized profiles of user traffic (Baldini et al., 2012; Li et al., 2009). The user data alteration threat can be mitigated by ensuring data integrity. The threat can affect data management and target the data layer. By masquerading as an authorized SDN controller, the threat identifies the activation of a malicious software on SDNs, such as a controller platform (Baldini et al., 2012). The threat can be mitigated by using the digital signatures for SDN software modules. Threat mitigation requires system integrity, identity verification, and accountability. Application management can be affected by the threat activation and target the control layer, control–data interface, and data layer of the

SDN.

2.4 State-of-the-art SDN Security Solutions: A Complete Analysis and Overview

In this section, we provide a complete analysis and overview of existing state-of-the-art SDN security solutions. Basically, we present the main classification of the innovative security solutions of SDNs, as shown in Figure 2.3. SDN security solutions are classified into six main categories, namely, (a) secure design of SDN, (b) implementation of satisfactory audit, (c) enforcement of security policy, (d) security monitoring and analysis, (e) security augmentation, and (f) fault tolerance. We further classify the surveyed solutions by devising a thematic taxonomy based on the SDN layers/interfaces, the distinguishing features of SDNs, implementation environment, and security objectives. The taxonomy is presented in Figure 2.3. This section also presents the identification of the distinguishing features and the effect of security solutions on each layer/interface of the SDN.

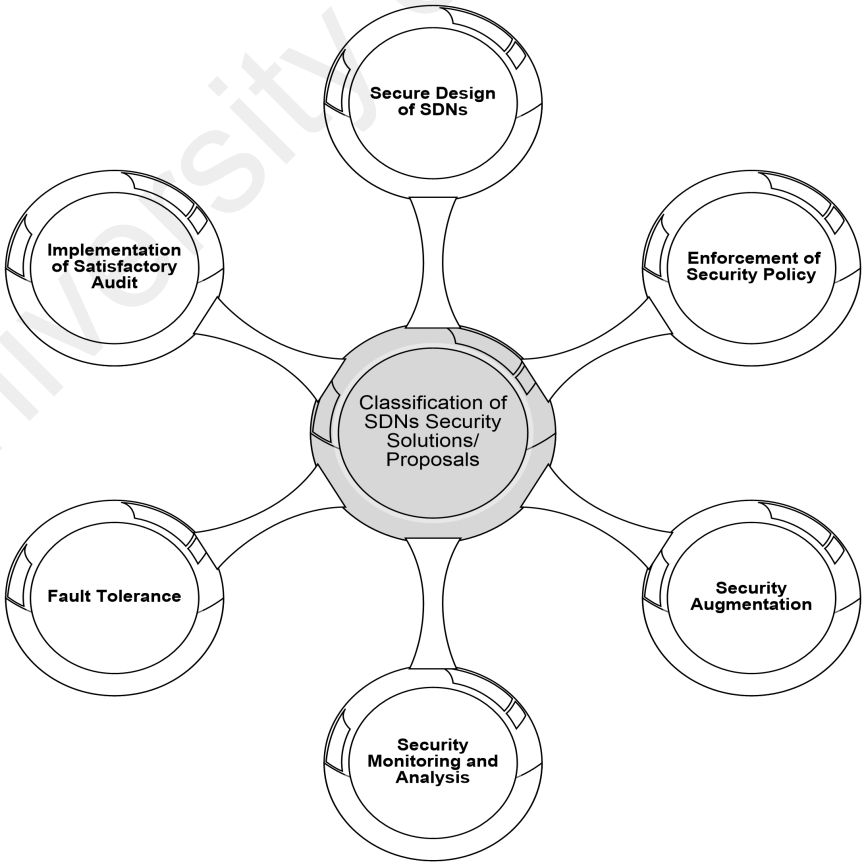

Figure 2.3: Main Classification of the State-of-the-Art SDNs Security Solutions

2.4.1 Secure Design of SDN

Efforts in this category are limited. FRESCO (Shin, Porras, et al., 2013) a security-specific application development framework for OF networks, was proposed in the literature. FRESCO is a security application development platform that facilitates the exportation of API scripts, which help security experts develop threat detection logic and security monitoring as programming libraries. Moreover, FRESCO is a click-inspired (Shin, Porras, et al., 2013) programming framework that facilitates the rapid design and modular composition of different security mitigation and detection modules using OF. The implementation and evaluation demonstration of FRESCO is performed through NOX, which is an open-source OF controller; however, the security constraints generated by these applications are controlled and enforced by FortNox (Porras et al., 2012). FRESCO work can be extended to different architectures (Ng, 2010; Mogul et al., 2010).

FortNox (Porras et al., 2012) employs a security enforcement kernel (SEK) to enforce flow constraints for active defense against different threats. FortNox is basically an enforcement engine that is responsible for avoiding rule conflicts from different security authorizations. FortNox uses two protection mechanisms: (1) rule prioritization, which ensures that any new rule that contradicts the rules produced by FRESCO applications are simply overridden because of the highest priority, and (2) the conflict detection algorithm is applied to each new rule, thus rejecting any new rule immediately when a conflict is detected. FortNox is merely a software extension that deals mainly with particular OF application policy violation, dynamic flow tunneling, and rule conflict detection. Furthermore, it addresses flow rule contradiction in real time and avoids any adversarial OF application attempts to inject flow rules to bypass SEK enforced rules. Work on FortNox is highly inspired by previous research (Al-Shaer & Al-Haj, 2010; Al-Shaer, Marrero, El-Atawy, & ElBadawi, 2009; El-Atawy et al., 2007; A. X. Liu, 2008; G. G. Xie et al.,

2005); the FRESCO work is an extension carried out by the authors of FortNox.

Remarks: The first proposal is a major contribution toward secure programming, which has a direct effect on the application layer, control layer, and interfaces between the two layers, except the data layer. The second proposal focuses on rule conflict and authorization, which primarily affect the control layer and south and northbound interfaces. However, this action does not improve the security of the application layer and infrastructure layer. FortNox is classified in this category mainly because of its SEK that is used for real-time verification.

2.4.2 Implementation of Satisfactory Audit

Security audit is an important area that must be addressed thoroughly. Verificare (R. W. Skowyra, Lapets, Bestavros, & Kfoury, 2013) was proposed as a design and modeling tool that satisfies all the requirements of a system design, and it is a key contribution in this category. Verificare is an audit design and modeling tool for verifying real-world system properties against a system model, which highlights and traces any property violation. Moreover, the tool guarantees network safety, correctness, and reliability. Verificare is designed using a complementary pair of SDNs and formal verification. The authors further provide an example for their work using OF-based learning switches to enable communication between mobile nodes. This particular proposal also considers the verification of network correctness and specification modeling while considering scalability issues. Some intersecting verification tools are available, such as PRISM (Kwiatkowska, Norman, & Parker, 2011) and Proverif (Blanchet, 2005); however, these tools are not completely comparable to Verificare. Another major contribution in this area (Handigol, Heller, Jeyakumar, Maziéres, & McKeown, 2012) allows software developers of SDNs to trace the root cause of bugs by reconstructing a series of events that cause that particular bug. Packet backtrace assists SDN programmers in resolving logical errors, helps implementers of the

switch to resolve protocol compatibility errors, and helps network operators in submitting a complete bug report to vendors. This tool also uses a network debugger, which is a programmatic control used to facilitate new ways of debugging networks. For instance, in debugging the network, servers could not connect to clients and no forwarding rule for packet matching is found in the middle of the network; this approach is similar to the detection of servers placed in the wrong location. Moreover, this tool effectively tracks the root cause of the bug.

The virtual source address validation edge (VAVE) (Yao, Bi, & Xiao, 2011) is a significant contribution in this area and is used to solve the problem of source address validation. VAVE also checks whether the source of the packet is valid or not based on the generated rules. The VAVE application is designed mainly to validate the source address. However, spoofing is still a problem on the Internet. Another standard called source address validation (SAVI) (Saucez, Bonaventure, & Iannone, 2015) does not provide complete protection against spoofing because of solution space constraints; VAVE can be employed to improve SAVI (Yao et al., 2011). Moreover, VAVE can be used to avoid attacks related to IP spoofing, SYN flooding, smurf attacks, and DNS amplification (Yao et al., 2011). Furthermore, VAVE uses NOX controller to regulate the validation rules from a global view on each SAVI device.

A more recent contribution in the area of security audit is Fleet (Matsumoto, Hitz, & Perrig, 2014). Fleet defends against a malicious administrator who is attempting to reprogram the entire network. Fleet is useful against damage to routing and forwarding and disturbance of network availability by misconfiguring the SDN controller. Table 2.3 summarizes the state-of-the-art SDN security solutions. The table demonstrates clearly the distinguishing features utilized for each state-of-the-art SDNs and the exact problem addressed by a particular technique together with the simulation/emulation environment of the corresponding technique. The table also shows that certain techniques with in-

common classification exist. By contrast, techniques of a similar class are not exclusively of the same nature. The identification of features in the table represents the potential of emerging SDNs.

2.4.3 Enforcement of Security Policy

Enforcing security policy is a fairly important and serious issue in SDNs. The security research community of SDNs has paid considerable attention to addressing the issue of policy conflict resolution and enforcement. Researchers (Son, Shin, Yegneswaran, Porras, & Gu, 2013) proposed FLOVER, which is a model-checking system that verifies the flow policies against the security policies of a network. FLOVER is a flow verification tool that detects inconsistencies that arise in a flow table against the security policy of the network by using satisfiability modulo theories solver. The performance of FLOVER is efficient. However, FLOVER is inspired by former research on modeling security policies of firewalls, and it does not address the dynamic flow rules in SDNs.

The other major contribution is No bugs In Controller Execution (NICE) (Canini, Venzano, Perešíni, Kostić, & Rexford, 2012). The NICE tool combines model checking with symbolic execution to determine inconsistencies in multiple OF applications. NICE proficiently uncovers bugs in OF programs while considering the correctness of the application installed on multiple devices. Another technique (Kothari, Mahajan, Millstein, Govindan, & Musuvathi, 2011) employed symbolic execution with the same principle as NICE to detect network protocol manipulation attacks. FlowChecker (Al-Shaer & Al-Haj, 2010) adopts binary decision diagram (BDD) to handle intra-switch misconfiguration in a particular flow table. FlowChecker is used to resolve the conflicts that may arise across several OF switches. It simply encodes the flow table configuration through BDD and utilizes different model checking techniques to model the interconnected network of OF switches.

Table 2.3: Classification and Comparison of the State-of-the-Art SDN Security Solutions

Classification	Techniques	SDNs Features	Problem Addressed	Environment
Secure Design	FRESCO	Centralized management	Secure application development and modular composition of detection and mitigation security modules.	NOX
	FortNOX	Flow based forwarding and dynamic updation of rules	The problem of rule conflict and provisioning of role-based authorization.	NOX
Security Audit	VERIFICARE	Programmable aspects at control plane	Verification of network correctness and specification modelling	OpenFlow
	SDN-Debugger (ndb)	Flow based forwarding and programmable aspects	Tracing the root cause of bugs in a Network	Mininet
	Fleet	Centralized management	Malicious administrator problem	Mininet and POX (Fleet Controller)
	VAVE	Traffic analysis and dynamic updation of rules	Source address validation	NOX
Security Enforcement Policy	FLOVER	Flow rules policies	Model checking for security policies to do real time verification	NOX
	Flow-based Security Language (FSL)	Flow control features of SDNs	Network security policy enforcement	NOX
	Splendid Isolation: language based security	Centralized management	Traffic isolation and slice abstraction	OpenFlow
	NICE (No bugs In Controller Execution)	Programmable Aspects (controller programs)	Policy for exclusion of bugs in Controller at application run time using finite-state model checking	NOX
	LiveSec	Flow control and an introduction of new Access-Switching layer	Interactive Policy Enforcement	NOX
	SANE	Centralized management	Central Policy Enforcement	SANE Controller
	Ethane	Centralized Control and dynamic updation of rules	Flow-rule Enforcement	Ethane Controller
	PermOF	Centralized management	Policy enforcement of Access Control	OpenFlow
	FLOWGUARD	Flow path analysis and dynamic updation of rules	Verification of Security Policy	Floodlight
	SDNs firewall using Header Space Analysis (HSA)	Flow space analysis and dynamic updation of rules	Developing a robust firewall application for SDNs	Floodlight
	VeriFlow	Centralized management, flow analysis and dynamic updation of rules	Verification of network-wide invariants in real time using equivalence classes and forwarding graphs	Mininet
	NetPlumber	Centralized management	Policy Compliance checking in real-time	GoogleŐs SDN, the Stanford backbone and Internet 2
	FlowChecker	Flow based forwarding and centralized management	Policy enforcement for flow-table configuration verification at application run time using binary decision graphs model checking.	GENI OpenFlow environment

Classification	Techniques	SDNs Features	Problem Addressed	Environment
Security Augmentation	Slick Architecture	Centralized control and flow analysis	Traffic steering based on prerequisite security criteria (Policy Enforcement)	Slick Controller
	FlowTags Architecture	Centralized control	An extended SDNs architecture for Correctness of systematic policy enforcement.	SDN controller
	OrchSec	Traffic monitoring and centralized management	Controller-agnostic security application development	Mininet, Floodlight and POX
	Covert channel protection using Filter	Centralized control and packet analysis	The problem of Covert Channel attacks	OpenFlow
	SIMPLE	Centralized control	middlebox-specific traffic steering	Extended POX (SIMPLE controller)
	OF-RHM	Centralized control and dynamic updation of rules	Moving target defense	NOX
	Self-Organizing Maps (SOM)	Traffic analysis and dynamic updation of rules	The problem of DDoS attacks	NOX
	OpenSAFE	Traffic analysis and centralized management	How to and where to route the traffic for network analysis and security monitoring applications	NOX
	ident++	Flow based forwarding and centralized management	Delegation of network security to end hosts for more flexible and powerful enforcement of policies.	ident++ controller (OpenFlow)
	Resonance	Centralized control with continuous monitoring	Access control enforcement for enterprise networks	
	FlowNAC	Traffic analysis and centralized management	Network access control	NOX
	MAPPER	Programmable aspects (Programmable switches and routers)	Fine grained access control policy enforcement	
	NetFPGA based solution	Programmable aspects of SDNs	Intrusion detection	
	ADS Revisit	Programmable aspects of SDNs	An extension for finding anomalies detection accuracy	NOX
	CloudWatcher	Flow monitoring and analysis	An extension for Cloud monitoring	NOX
	L-IDS	Centralized management and dynamic flow rules	Intrusion detection	L-IDS controller and OpenFlow
	NICE (Network Intrusion detection and Countermeasure sElection)	SDNs programmable switches and APIs	Intrusion detection and mitigation	OpenFlow
	SnortFlow	Flow based forwarding	Intrusion Prevention	Snort and OpenFlow
Security Monitoring and Analysis	OpenWatch	Flows monitoring and analysis (Adaptive flow counting)	Load balancing and achieving anomaly detection accuracy	
	FleXam	Centralized management	Deep packet inspection (DPI)	OpenFlow
	AVANT-GUARD	Data-to-control plane communication	The problem of DoS attack	POX
	NetFuse	Flow analysis and centralized management	Protection against traffic over load caused by internal or external threats	OpenFlow
	CONA	Traffic analysis and dynamic updation of rules	Resource Exhaustive Attacks	NetFPGA-OF (OpenFlow)
Fault Tolerance	SPARC	Centralized management	Fault tolerance	NOX
	Link failure	Centralized management	Fault tolerance	NOX

The VeriFlow scheme verifies real-time invariants (Khurshid, Zou, Zhou, Caesar, & Godfrey, 2013). VeriFlow monitors dynamic changes in the network by constructing a model of the network behavior. Moreover, VeriFlow employs custom algorithms to derive network errors automatically. NetPlumber is a scheme that is similar to VeriFlow (Kazemian et al., 2013). However, NetPlumber employs header space analysis (HSA) and dependency graphs for real policy checking. Both VeriFlow and NetPlumber are protocol independent and have the same runtime performance. Both schemes support verification of forwarding actions in real time. Unlike VeriFlow, however, NetPlumber verifies arbitrary header modifications, including encapsulation and rewriting.

An important contribution (Hinrichs, Gude, Casado, Mitchell, & Shenker, 2008) presented a flow-based policy enforcement grounded on flow security language (FSL). Implemented on NOX controller, FSL employs the concept of network flow and handles policy conflicts between administrators and conflicts that arise in the rule set of the administrators. Moreover, the decision with regard to policy is enforced on each packet and on network links to hasten enforcement decisions per second. Another close category of language-based security is termed splendid isolation (Schlesinger, Story, Gutz, Foster, & Walker, 2012). The authors discussed mainly the verification of the isolation of the program traffic. Moreover, the study addressed the problem of programming networks securely and reliably. Slice-based network programming abstraction and isolation protect the programs from outside interference with other important security benefits. In addition, it simplifies construction of programs. A key contribution is LiveSec (K. Wang, Qi, Yang, Xue, & Li, 2012), which is a flexible and scalable security management architecture (Koponen et al., 2010). The main purpose of LiveSec is interactive policy enforcement to ensure complete end-to-end traffic control, distributed load balancing of security workload, and application awareness monitoring through live traffic monitoring and historical traffic replay. SANE (Casado et al., 2006) protection architecture proposed

a logically centralized controller for all routing and access control decisions and policies. The SANE proposal was considered a radical change to the network infrastructure. A significant extension of SANE is Ethane (Casado et al., 2007). Ethane primarily proposes two components for controlling the network: (a) a centralized logical controller to enforce global policy and (b) switches that forward packets according to flow table rules. However, Ethane was proposed for flow rule enforcement. SANE and Ethane established the foundation for emerging SDNs. A significant contribution is PermOF (Wen et al., 2013), a fine-grained permission system that comprises a set of OF-specific permissions and runtime isolation mechanism for applying the permissions. The set of OF-specific permissions is designed considering four different aspects: (a) threat model, (b) controller implementation API set, (c) application functional requirements, and (d) control messages in OF. The proposed isolation mechanism isolates the controller and applications in a thread container. The applications cannot call controller procedures or directly refer to the memory of a kernel. Moreover, the application and OS are isolated by introducing a shim layer between them called an access control layer. The shim layer is controlled by the kernel of a controller. Basically, the design consideration is setting of permission and isolation mechanisms to enforce permission control.

FLOWGUARD (H. Hu, Han, Ahn, & Zhao, 2014) is a comprehensive framework responsible for the detection and resolution of policy violations of firewalls in the dynamic environment of SDNs. This technique helps in resolving policy conflicts automatically and in real time. The authors in (J. Wang et al., 2013) proposed a systematic approach to detect and resolve conflicts in SDN firewalls by checking the firewall authorization space and flow space. The main aim of the study was to build a robust SDN firewall. The proposed technique was used to implement a cross-check on both flow tables and firewall policies to detect conflicts efficiently. The approach searches the flow paths in the entire network, checks these paths against all firewalls, and denies rules to determine whether

conflict arises with the firewall deny rules. However, the conflict resolution strategies vary with the operations involved in the flow entries and flow rules. The effectiveness and efficiency of the proposed approach is examined through HSA.

2.4.4 Security Augmentation

This particular area addresses the promising SDNs that can be extended to enhance security for different applicable networks (Ding, Crowcroft, Tarkoma, & Flinck, 2014). The SDN control plane ensured that sophisticated network-wide policies can be deployed and implemented through simple programs. SDNs can be used to enable operators to implement network policies, such as performing deep packet inspection (Kreutz et al., 2015). Middleboxes can also be used to implement network policies (Sekar, Egi, Ratnasamy, Reiter, & Shi, 2012; Joseph, Tavakoli, & Stoica, 2008). Recent studies have proposed the integration of security middleboxes into SDNs to ensure that the programmability aspect is beneficial for security purposes. Two notable contributions and extensions to enhance security by using the distinguishing features of SDNs are Slick (Anwer, Benson, Feamster, Levin, & Rexford, 2013) and FlowTags architecture (Fayazbakhsh, Sekar, Yu, & Mogul, 2013). Slick is a prototypical control plane that allows middleboxes to assist network operators and deploy refined policies efficiently. Moreover, Slick architecture proposes a centralized controller that installs and migrates functions onto middleboxes. Subsequently, the applications can request the corresponding functions to route particular flows based on their prerequisite security criteria. FlowTags architecture is an extension of the SDN architecture and uses flow tracking capability to ensure consistent policy enforcement. The tags are added by middleboxes, which are subsequently used by switches to enforce policies systematically. The FlowTags architecture uses FlowTags APIs to communicate with the controller. FlowTags comprises flow information embedded in a packet header to track and enable supervised routing of the tagged packets. However, the

Slick and FlowTags architectures have a serious limitation: they work based mainly on pre-defined policies and cannot handle dynamic operations. In addition, directing traffic through the desired sequence in traditional networks is difficult and requires both operator expertise and manual efforts. Another major contribution is SIMPLE (Qazi et al., 2013), which is a policy enforcement layer and an introduction to an appropriate middlebox deployment that is managed thoroughly by the SDNs. SIMPLE facilitates programming of the entire network by directing a selected traffic through the appropriate middlebox. Realizing the benefits of SDNs for controlling the middlebox-specific traffic steering is the primary goal of the present research proposal. Furthermore, the feasibility of using SDNs along with industry concerns for integration with the existing infrastructure is addressed. A key contribution is OrchSec (Zaalouk, Khondoker, Marx, & Bayarou, 2014), which is an orchestrator-based architecture that enhances network security to develop reliable security applications. The architecture is based on the use of the SDN control function features and the network monitoring aspects. The proposed architecture decouples the application development process from the SDN controller, thereby making it controller agnostic. The proposed solution can better withstand DoS/DDoS and cache poisoning/ARP spoofing. Researchers in (X. Liu, Xue, Feng, & Dai, 2011) proposed an SDN-based architecture to restrict covert channel attacks in traditional networks. Tracing the information flow from a low-level host to a high-level host or the same level host, and vice versa, is difficult. The technique employs an OF-based module called "filter," which is responsible for controlling and checking the packet flow, the content of the packet, and the delay it experiences to avoid side-channel attacks.

The solutions based on the SDN frameworks are as follows: Primarily, an active and live IP address to be attacked should be found. Changing an IP address frequently and proactively is a novel defense mechanism called the Moving Target Defense (MTD). A prime contribution (Jafarian, Al-Shaer, & Duan, 2012) proposed an SDN-based MTD

architecture, where the controller frequently assigns a random virtual IP address to each host translated from the real IP address of the host. Moreover, IP mutation to the end hosts is transparent; however, real IP addresses can only be reached by an authorized access. The proposed mechanism is based on OpenFlow Random Host Mutation (OFRHM), which is used to manage a pool of IP addresses assigned to hosts in a particular network in which the actual IP addresses are hidden from outside the network; this approach represents a specific form of adaptive cybersecurity. The defense against stealthy scanning to discover vulnerable targets in the entire network was presented in (Jafarian et al., 2012). This work was inspired by previous research on proactive cyber defense (Atighetchi, Pal, Webber, & Jones, 2003; Kewley, Fink, Lowry, & Dean, 2001; Antonatos, Akritidis, Markatos, & Anagnostakis, 2007). Table 5 shows the main classification together with the effect of each of the state-of-the-art security mechanism on SDN layers/interface and its potential.

Although it is a well-understood security problem, DDoS attack detection is still a serious concern. A major contribution toward DDoS detection was presented in (Braga, Mota, & Passito, 2010). The technique utilizes self-organizing maps (SOMs) to identify abnormal/injected flows. The proposal was based on the SDN programmatic interface, which facilitates switch information handling. The DDoS attack detection is grounded on the flow features. A complementary work that uses SOM is presented in (Ramadas, Ostermann, & Tjaden, 2003; Min & Dongliang, 2009; Jiang, Yang, & Xia, 2009; Mitrokotsa & Douligeris, 2005). One unique feature is lightweight detection, which can be upgraded easily against new attacks. Moreover, the technique facilitates the addition or removal of switches from the detection loop. Security monitoring of the entire network is a tedious task; one notable contribution of SDN-enabled security monitoring extension is OpenSAFE (Ballard, Rae, & Akella, 2008), which monitors large-scale networks. OpenSAFE is used to manage traffic routing through network monitoring agents by using its

Table 2.4: Classification and Identification of the Impact of the State-of-the-Art Security Solutions on Different SDN Layers/Interfaces

Techniques Classification	**Techniques**	**SDN Layer/ Interface**				
		Application Layer	North-bound Interface	Control Layer	South-bound Interface	Data Layer
Secure Design	FRESCO	✓	✓	✓	✓	
	FortNOX		✓	✓	✓	
Security Audit	VERIFICARE		✓	✓	✓	
	SDN-Debugger (ndb)	✓			✓	
	Fleet			✓	✓	✓
	VAVE			✓	✓	✓
Security Enforcement Policy	FLOVER	✓	✓	✓	✓	
	Flow-based Security Language (FSL)	✓	✓	✓	✓	
	Splendid Isolation: language based security		✓	✓		
	NICE (No bugs In Controller Execution)	✓	✓		✓	
	LiveSec	✓	✓	✓	✓	✓
	SANE			✓	✓	✓
	Ethane			✓	✓	✓
	PermOF	✓		✓	✓	✓
	FLOWGUARD	✓	✓	✓	✓	✓
	SDNs firewall using Header Space Analysis (HAS)	✓	✓	✓	✓	✓
	VeriFlow	✓	✓	✓	✓	
	NetPlumber	✓	✓	✓	✓	
	FlowChecker	✓	✓	✓	✓	
Security Augmentation	Slick Architecture	✓	✓	✓	✓	✓
	FlowTags Architecture	✓	✓	✓	✓	✓
	OrchSec	✓	✓	✓		
	Covert channel protection using Filter		✓	✓	✓	✓
	SIMPLE	✓		✓		✓
	OF-RHM			✓	✓	✓
	Self-Organizing Maps (SOM)	✓		✓	✓	
	OpenSAFE	✓	✓	✓	✓	
	ident++	✓	✓	✓	✓	✓
	Resonance	✓		✓	✓	✓
	FlowNAC	✓		✓	✓	✓
	MAPPER	✓		✓	✓	
	NetFPGA based solution	✓				✓
	ADS Revisit	✓		✓	✓	
	CloudWatcher	✓	✓	✓	✓	
	L-IDS	✓		✓	✓	✓
	NICE (Network Intrusion detection and Countermeasure sElection)	✓		✓	✓	✓
	SnortFlow	✓		✓		✓
Security Monitoring and Analysis	OpenWatch			✓	✓	✓
	FleXam	✓		✓	✓	
	AVANT-GUARD			✓	✓	✓
	NetFuse	✓	✓	✓	✓	✓
	CONA	✓	✓	✓	✓	✓
Fault Tolerance	SPARC			✓	✓	✓
	Link failure			✓	✓	✓

ALARMS policy language. ALARMS is a flow specification language that simplifies dramatically the management of the tools used for network monitoring. A similar prominent solution for security monitoring solution in the cloud using SDN features is CloudWatcher (Shin & Gu, 2012), which is a practical and feasible framework for cloud environments and provides security monitoring to supervise the dynamic flows of the network smoothly and efficiently.

SDNs can be useful for network-wide access control. A major contribution is Resonance (Nayak, Reimers, Feamster, & Clark, 2009), which uses programmable switches and controllers to enable distributed network monitoring by using dynamic access control enforced by network devices themselves. Resonance is a dynamic access control system based on real-time alerts and flow level information; however, it does not follow the centralized architecture. Therefore, resonance is an interface-based policy control. Another recent major contribution for access control is FlowNAC (Matias, Garay, Mendiola, Toledo, & Jacob, 2014). FlowNAC is used to secure access to all available network resources and devices. However, its design principle is subjected to a centralized policy decision, and policy enforcement should be defined once to avoid collision. Another proposed solution is mobile application personal policy enforcement router (MAPPER) (Sapio et al., 2014), a fine-grained access control that guarantees network and information security. MAPPER is responsible for imposing user/role specific policies without obtaining access to the device of an end user. The framework can identify and differentiate traffic generated by diverse applications, platforms, and devices. The ident++ protocol solution, which depends on the distributed architecture, is presented in (Naous, Stutsman, Mazières, McKeown, & Zeldovich, 2009). The proposed protocol queries end users and hosts for information to ensure their active involvement in forwarding decisions and to simultaneously avoid bottleneck in the controller.

The SDNs can be better extended in the field of intrusion detection and prevention.

An SDN-based learning intrusion detection system (L-IDS) (R. Skowyra, Bahargam, & Bestavros, 2013) is used to protect embedded mobile devices in a particular location. The L-IDS detects a wide variety of attacks while reconfiguring the network in real time. The technique helps mitigate the attacks; however, this solution primarily involves delegating network security. A hardware-based solution for network IDS/IPS is presented in (Goodney, Narayan, Bhandwalkar, & Cho, 2010). This technique uses NetFPGA, an open-source field programmable gate array (FPGA) based network interface with OpenFlow modules. The technique is a co-design of hardware/software for developing a deep packet inspection engine. A major contribution in revisiting anomaly-based intrusion detection using SDNs is proposed in (Mehdi, Khalid, & Khayam, 2011). The proposed methodology provides high detection accuracy and is suitable small home/small office or home networks. Researchers proved that SDNs are effective for some prominent anomaly detection system algorithms. A significant contribution is NICE in virtual network systems (Chung, Khatkar, Xing, Lee, & Huang, 2013). NICE is an OF-based framework for the detection of vulnerable applications installed on virtual machines. Subsequently, these vulnerable applications can either be used to compromise VMs in the cloud, especially in infrastructure-as-a-service clouds, or may work as a zombie for further exploitation. NICE is a distributed vulnerability detection framework based on analytical models, programmable virtual switches (OF), and OF programming APIs to build an efficiently monitored control plane to detect and mitigate sophisticated attacks. Another efficient attempt is SnortFlow (Xing, Huang, Xu, Chung, & Khatkar, 2013), which is an IPS for the cloud environment. The technique simply combines the capabilities and features of Snort and OF to employ IPS. NICE and SnortFlow are implemented for the cloud environment with almost the same objectives. However, NICE is an attack graph-based IPS, whereas SnortFlow utilizes snort for intrusion detection and OF-based reconfigurable network features to prevent intrusion.

Remarks: A critical analysis of the literature on SDN security clearly shows that the security research community has two opinions about SDNs. One school of thought is focused on securing SDNs and making them dependable. By contrast, the second school of thought believes in the use of the remarkable features and capabilities of SDNs to enhance and improve the security of different applicable networks; the latter belongs to our main classification of SDN security augmentation. Security augmentation in our classification implies that SDNs are equivalent to "security defined networks." The SDNs contribute greatly in improving the security of many modern and existing cellular, mobile, and wired networks. Although SDNs have promising architecture, their security, reliability, and dependability has not been proven.

2.4.5 Security Monitoring and Analysis

Security monitoring and analysis is an essential part of the dynamic environment of the SDNs. The authors in (Shin, Yegneswaran, Porras, & Gu, 2013) proposed AVANT-GUARD, an implementation of two significant changes to SDNs. One extension to the data plane is called connection migration, which significantly minimizes the data-to-control plane interactions that increase during DoS attacks on a southbound interface. Another extension called actuating trigger expedites the responsiveness to the changing flow dynamics within the SDN data plane. Actuating triggers are introduced over the statistics collection services of the data plane. The proposed model is resilient against different security threats. However, AVANT-GUARD has several limitations. Although AVANT-GUARD works well on network scanning and on transport control protocol (TCP) SYN flooding attacks, it has no counter user datagram protocol, Internet control message protocol, or application layer DoS attacks. OpenWatch (Zhang, 2013) is an adaptive flow counting method that detects anomalies in the SDNs. Monitoring the network adds an overhead to the network; hence, the proposed model uses an adaptive flow-based count-

ing mechanism to ensure accuracy while anomalies are detected, thereby considerably reducing the overhead. The studies orthogonal to OpenWatch are (Yu, Jose, & Miao, 2013; Jose, Yu, & Rexford, 2011; Huici et al., 2012; Moshref, Yu, & Govindan, 2013). Unique to the method is its provision of input to the anomaly detectors, simultaneously considering the optimization of the network.

Another contribution is the FleXam (Shirali-Shahreza & Ganjali, 2013), a sampling extension that provides access to an OF controller to obtain packet-level information. The FleXam enables the controller to sample the packets stochastically or deterministically considering the application requirements. FleXam eliminates flow setup time and reduces the control plane load. Consequently, such applications can directly run on small networks. The authors in (Suh et al., 2010) recently proposed a content-oriented networking architecture (CONA) that introduced the content-centric communication model to resolve the issue of accountability. CONA is a content-aware monitoring and supervision method that can be used to detect resource-exhaustive attacks. Moreover, CONA is useful in the detection of a particular malicious host that generates a DDoS attack. Content extraction on the agent helps CONA in attaining accountability and simultaneously taking countermeasures against different DoS attacks. CONA uses NetFPGA and OF-enabled switches. Another major contribution in this particular area is NetFuse (Y. Wang, Zhang, Singh, Lumezanu, & Jiang, 2013), which is responsible for monitoring surges due to routing configuration errors, security breaches and attacks, and other operator errors. NetFuse is based on multi-dimensional flow aggregation and detects suspicious flow clusters in cloud and data-center networks. NetFuse also addresses traffic overloading, which has several serious financial and availability implications in cloud and data center environments.

2.4.6 Fault Tolerance

Fault tolerance against different attacks is a serious concern in the SDN dynamic environment, particularly in SDN centralized architecture. Few contributions have been made in this particular area. Researchers in (Staessens, Sharma, Colle, Pickavet, & Demeester, 2011) conducted different experiments of recovery from a link failure using OF. The technique uses OF-based switches to handle data-to-control plane failures. However, dependency on a centralized architecture may cause delay in restoration and recovery from failure in case of large-scale networks. Split architecture carrier grade networks (SPARK) are another major contribution to this field (Sharma, Staessens, Colle, Pickavet, & Demeester, 2011) and ensure fast failure recovery using OF. The SPARK project uses OF-based switches and controllers. Furthermore, this technique is used for fast data-to-control plane recovery failures. The SPARK project aims to develop a fast restoration and recovery mechanism in case of failure by any means. Another solution called automated protection switching (APS) is even faster than SPARK and may not require contacting the controller after failure. Both techniques are highly remarkable contributions; however, the technique in (Staessens et al., 2011) caused some delay in large-scale networks. We briefly illustrate the two main schools of thought followed by a thematic taxonomy.

2.4.7 Taxonomy of SDNs Security

The taxonomy shown in Figure 2.4 is based on state-of-the-art security solutions for SDN security. The devised taxonomy will help the researchers to understand the problem clearly and to consider all significant aspects of the emerging SDNs.

2.4.8 Secure Design of SDN

The existing solutions can be further classified based on the following parameters: solution categories, SDN layers/interfaces, security measures, implementation (simulation/emulation) environment, and security objectives. Researchers contributed toward auditing

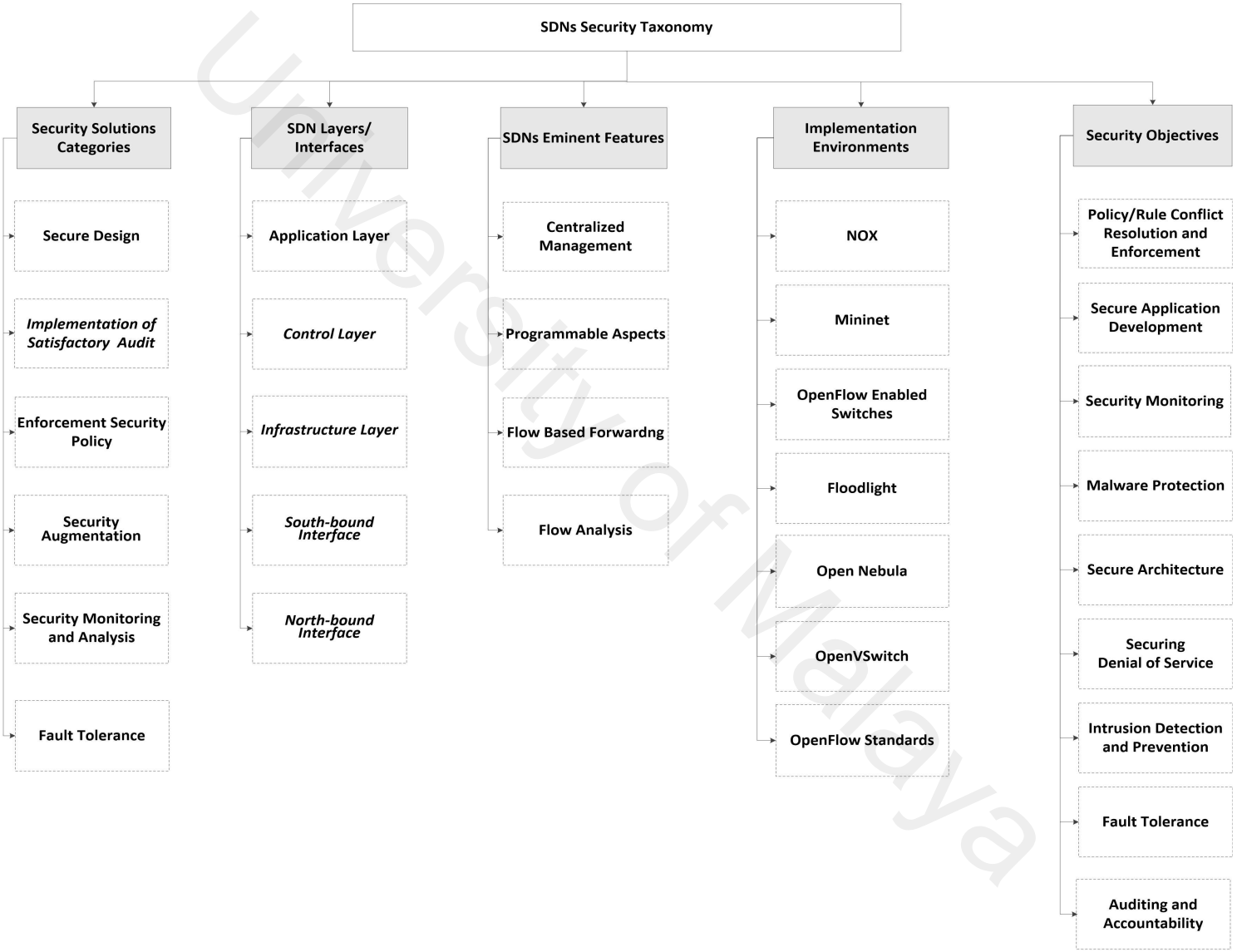

Figure 2.4: Taxonomy of SDNs Security Solutions

the dynamic environment of SDNs for security and accountability purposes. Major contributions were based on security policy enforcement and security enhancement of different applicable networks. Several studies focused on security monitoring and analysis, whereas other researchers contributed to fault tolerance.

All these innovative security solutions mainly target issues of a particular layer/interface of the entire SDN architecture. Addressing the issues of different layers/interfaces against various corresponding possible attacks and threats makes these solutions unique. However, these state-of-the-art solutions broadly consider the distinguishing features of SDNs, including centralized management, programmable aspects, flow-based forwarding, and flow analysis, as explained in the introduction. Each security solution of SDNs renders a particular eminent feature that makes these solutions even more exceptional. Moreover, the taxonomy is based on the implementation (simulation/emulation) environment. The majority of the security solutions are based on OF standards and popular OF controllers, as shown in Table 2.1. The extant security SDN solutions are based purely on a particular security objective. The taxonomy presents the major security objectives, including policy/rule conflict resolution, rapid designing and development of secure applications, monitoring for security purposes, and malware protection to prevent stealthy scanning and propagation. Moreover, intrusion detection and prevention secure the architecture and defend against different DOS attacks. Auditing and accountability are considered primary objectives for the dynamic environment of SDNs. Furthermore, a key objective is fault tolerance, with focus on the importance of the centralized management architecture of the SDNs.

2.5 Requirements and Key enablers for SDN security

Ensuring the security of every single component of the SDN is mandatory to build a secure SDN environment. The following are several essential requirements for securing

the SDN key components. Figure 2.5 depicts a situation in which an attacker searches for potential components of SDNs to be compromised. Moreover, 1 to 4 in Fig. 2.5 represent the requirements and their key enablers.

2.5.1 Securing the SDN Controller

Securing the SDN controller, which is the central decision point, is the foremost priority. The SDN controller is responsible for the overall management of the network. A simple compromise of the centralized SDN controller can affect the entire network (Metzler, 2012). Furthermore, as a single point of failure, the SDN controller serves as a potential target for attackers. The SDN controller as a software platform, if compromised, essentially allows a hacker to reconfigure the entire network. By spoofing the address of the controller, the attacker can take over the entire network easily through a fake controller.

This key component needs to be protected, and it can be protected in the following ways:

- High availability of the controller is ensured to protect against different DoS and DDoS attacks.
- The SDN controller must be protected with security policy enforcement, high availability, and minimum possible delay during incoming packets (Vissicchio, Vanbever, & Bonaventure, 2014).
- The OS that contains the controller must be secured against exploitable patches, backdoor accounts, and open doors, such as vulnerable open ports, services, and protocols.
- The protection of the system with the SDN controller must be secured against physical threats.

- The controller must have a mechanism that alerts an administrator to limit control communication during the attack or in case of a sudden attack (Jammal, Singh, Shami, Asal, & Li, 2014).

- An intelligent access control list must be implemented for packet filtration, and full isolation among the tenants that share the infrastructure must be ensured.

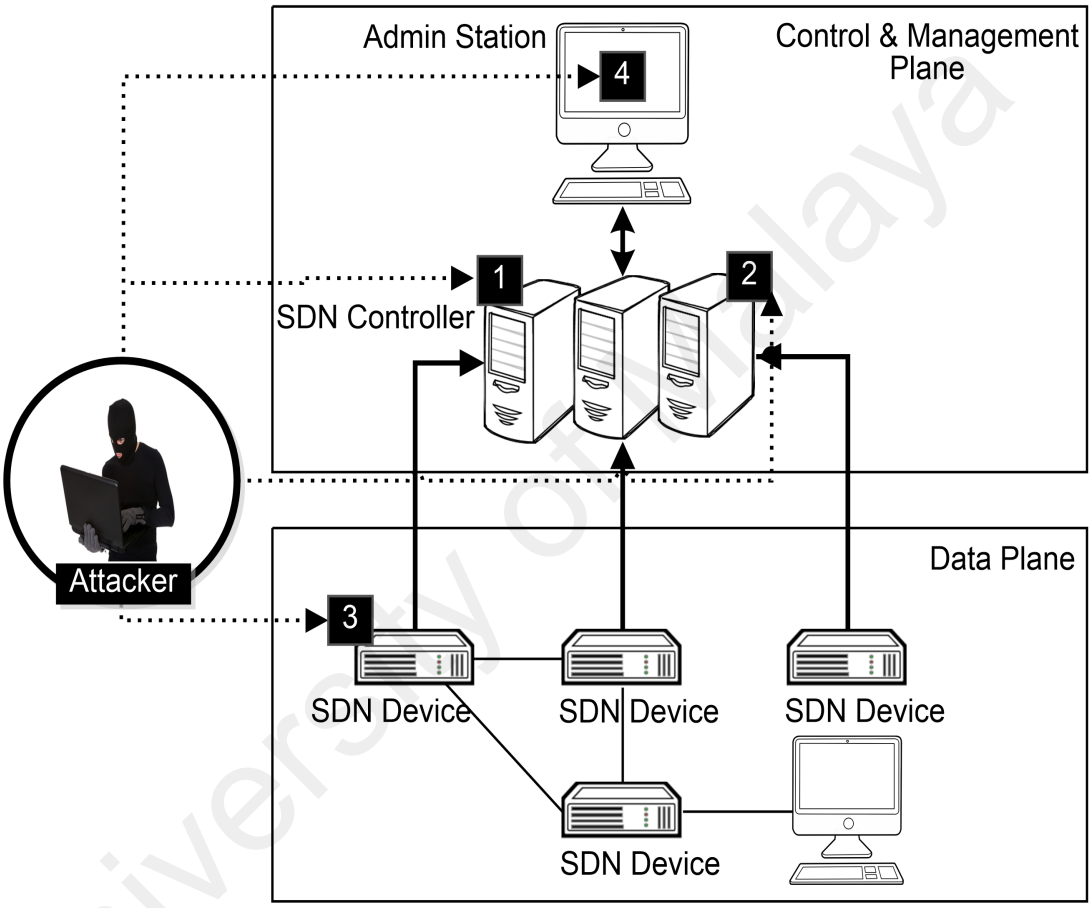

Figure 2.5: Attacker Searching for Potential Targets

2.5.2 Protecting Flow Paradigm of the SDN

SDN grounded on flow-based forwarding can ensure end-to-end communication security. The flow paradigm is the soul of SDN and must be protected. A successful influx of bogus flow may compromise the entire network (B. Wang, Zheng, Lou, & Hou, 2015). Flow abstraction of the controller may result in harvesting of the intelligence of the connected resources, and such harvested intelligence can be used in further attacks and exploitation

(Kreutz et al., 2015). An updated access control mechanism should be deployed in the network. Moreover, flows should be encrypted to prevent the injection of malicious flows. Proper authentication and authorization should be implemented to prevent side-channel attacks.

2.5.3 Fortifying SDN agents

The security of the SDN agent is important because it constitutes the data plane environment To compromise a strong entity, such as the SDN controller, an attacker may reach the target by compromising any vulnerable agent of SDN. For instance, link layer discovery protocol packets with forged source addresses can cause the SDN controller to install flow rules grounded on bogus information. Moreover, many existing switches, as part of the SDN infrastructure layer, are by default in listener mode, which may easily lead to the launch of malicious connections (Costa & Costa, 2013; Kreutz et al., 2013). Injecting false flow at any SDN agent can lead to its distribution to numerous agents who ultimately cause serious network disturbance. The security of the SDN agents requires deploying the latest identity management, threat isolation, and mitigation techniques. Moreover, the SDN agents require physical security. Further IPS, IDS, and firewalls should be actively deployed.

2.5.4 Hardening Application Programming Interfaces (APIs) and communication channels

APIs can be a potential target for attackers. Most importantly, the southbound APIs can be targeted easily for different DoS attacks to make the entire network unavailable. The creation of malicious APIs by skilled programmers is a critical issue; this trend already exists in the security research community. The communication channel between each layer must be well protected; security measures include secure coding, deployment of integrity checks, and digital signing of the code. Moreover, the communication channels

can be hardened by using TLS/SSL security or other cryptographic alternatives, such as threshold cryptography (Kreutz et al., 2013; Sookhak et al., 2015).

2.6 Conclusion

The emergence of SDNs has resulted in additional security requirements because of newly deployed infrastructural entities. Despite the promising architecture of SDNs, security was not considered part of the initial design. Significant work is in progress to develop state-of-the-art SDN security applications and solutions. However, research on SDN security is still in its infancy. Secure and dependable SDNs remains a distant goal.

To meet the newly imposed network security requirements, this study presented a broad overview of the security implications of each SDN layer/interface. We devised a contemporary layered/interface taxonomy of the reported security vulnerabilities, attacks, and challenges of SDNs to illustrate the main categories of security implications for each SDN layer/interface. Furthermore, we highlighted and analyzed the possible threats that may affect and target a particular layer/interface with a suggested corresponding compact solution. A discussion on state-of-the-art security solutions was also presented. A comprehensive survey, analysis, and classification of extant SDN security solutions promote ways to secure dependable SDNs. The surveyed solutions were further classified by devising a thematic taxonomy based on the SDN layers/interfaces, the SDNs' eminent features, implementation environment, and security objectives. This study identified the SDNs' distinguishing features for each state-of-the-art security solution, thereby making these security solutions unique. Moreover, the potential effects of each security solution on different SDN layers/interfaces were identified and presented. This study illustrated two main schools of thought in the SDN security research community. The potential security implications with their key enablers were elaborated for the development of secure and dependable SDNs.

CHAPTER 3: MODELING, ANALYSIS AND FORMAL VERIFICATION

This chapter investigates the defense space against the threat of attacks in SDNs due to both legitimate and compromised end hosts/soft switches that wrests either full or partial control of the entire network. Moreover, to preserve the correct functioning of the entire SDN architecture, an efficient detection of various distributed coordinated attacks and anomalies triggered by large-scale malicious events that predominantly target the control plane is of paramount concern. The main purpose of this chapter is to formally verify and validate the whole proposed system in the real-setting of SDNs rather than just problem analysis. The main contribution as opposed to simulation and testing is that the system is verified by providing a formal proof on an abstract mathematical model of the system that exhaustively checks and proves the intended behavior of the system. Additionally, the chapter provides a critical analysis of the specified sophisticated attacks on the control plane of the SDN even if the control channel (southbound API) is TLS (Transport Layer Security) enabled. The chapter also briefly discusses the preliminaries to model and analyze a system. Moreover, verification of the system and results is provided. Finally, the chapter demonstrates briefly the impact and analysis of attack on the control plane and closely analyzes the detection accuracy behavior and analysis of some of the extant state-of-the-art classification based network anomaly detection systems.

The remainder of the chapter is organized as follows. Section 3.1 discusses the motivation for the verification of the proposed system. Section 3.2 elaborates the tools and techniques used for the verification of the proposed research work. Section 3.3 presents the complete modeling and analysis of the proposed system. Section 3.4 provides the verification results. Section 3.5 presents a brief impact and analysis of the attack on SDN controller. The detection accuracy analysis and behavior of some of the extant state-of-the-art classification based network anomaly detection systems is provided in Section 3.6.

Finally, we provide the concluding remarks in Section 3.7.

3.1 Motivation for Verification of Proposed System in SDN

The formal verification of an intrusion-detection system that is capable to effectively identify a wide variety of sophisticated network attacks in the control plane of the SDNs is crucial and all-important. The early and efficient detection of the threat of attacks in the SDN control plane that subsequently help raising mitigation policies is of paramount concern and certainly necessitates to be formally verified to prove the correctness and consistency of the proposed system. The motivation behind complete formal verification is beyond just problem analysis, rather it formally verifies the validity of our proposed Intrusion-Detection System (IDS) being an extended module of the controller in SDNs. Besides, the problem is well-analyzed and established.

3.2 Preliminaries

In preliminaries section, we are discussing tools and techniques used in this research work for the formal verification.

3.2.1 High-Level Petri Nets (HLPN)

Petri Nets are very useful for mathematical and graphical modeling of wide range of systems, such as distributed, parallel, concurrent, stochastic, non-deterministic, and asynchronous systems. However, a tradeoff must be kept between modeling generality and analysis capability. Even a modest model can become too large for analysis process. In this work, we have used a variant of the conventional Petri Nets, termed as High-Level Petri Nets (HLPN) for the formal verification of the proposed migration technique. The HLPN is a set of 7-tuple $N = (P, T, F, \varphi, R, L, M_0)$, where:

1. P is a set of finite places,

2. T is a set of finite transitions such that P and T are two distinct sets $P \cap T = \phi$,

3. F represents the flow relation from place to transition or transition to place such that $F \subseteq (P \times T) \cup (T \times P)$,

4. ϕ represents the mapping function that maps places to data types, such that ϕ: P $\rightarrow$ Data types,

5. R defines the set of rules that maps T to logical formulae such that R: T $\rightarrow$ Formula,

6. L represents the labels that are mapped on each flow in F, such that L: F $\rightarrow$ Label,

7. M_0 represents the initial state/marking where flow can be initiated, such that M: P $\rightarrow$ Tokens.

The first three variables *(P, T, F)* provide information about the structure of the Petri Net. The next three variables *(ϕ, R, L)* provide the static semantics of the Petri Net, which means that the information does not change throughout the system. To build an understanding of a Petri Net, we demonstrate a small example. Figure 3.1 represents a

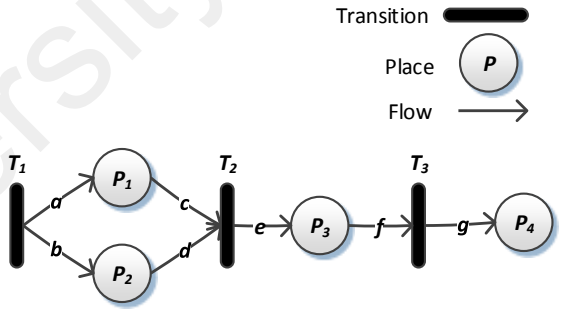

Figure 3.1: An Example of the HLPN

simple Petri Net, having 4 places *($P = P_1, P_2, P_3, P_4$)*, 3 transitions *($T = T_1, T_2, T_3$)*, and 7 flows *($F = a, b, c, d, e, f, g$)*. In HLPN, each place has some tokens to enable adjacent transitions, which means that the preconditions must hold for the transition to fire. The tokens can correspond to one type or a cross product of different data types. In Table 3.1, we have mapped places to the following data types.

Table 3.1: Places to Data-type Mapping

Places	Mapping	Description
$\varphi\ (P_1)$	(int, bool)	P_1 holds variables of type int and bool
$\varphi\ (P_2)$	(char)	P_2 holds variable of type char only
$\varphi\ (P_3)$	(string, int, bool)	P_3 holds variables of type string, int and bool
$\varphi\ (P_4)$	(bool)	P_4 holds variables of type bool only

Let α and β be the nodes of the HLPN N if and only if $\alpha, \beta \in P \cup T$. A node α is an input node of another node β if and only if there is a directed arc from α to β such that $(\alpha, \beta) \in F$. Node α is an output node of β if and only if $(\beta, \alpha) \in F$. The precondition is $\bullet P_1 = (P_2 \mid (P_2, P_1) \in F)$ and post condition is $\bullet P_1 = (P_2 \mid (P_1, P_2) \in F)$. The precondition must hold to enable the transition. For example in the Figure 3.1, precondition for T_2 will use c and d as input. Similarly, post-condition will take values from outgoing flow to enable further transitions.

3.2.2 SMT-Lib and Z3 solver

In the context of automated reasoning and formal verification, Boolean Satisfiability Solvers (SAT) are used. However, now the decision problems are encoded and solved as Satisfiability Modulo Theories (SMT). The SAT are propositional satisfiability solvers. The SMT takes the decidability problem as first order logic formula and decide for its satisfiability based on the decidable background theory. There are a number of theories supported by the SMT solvers, such as equality and un-interpreted functions, linear arithmetic over rationals, linear arithmetic over integers, non-linear arithmetic over reals, over arrays, bit vectors, and combinations. The SMT-Lib provides a common input platform for many solvers used for the verification of systems. Behavioral specifications of a system can be represented using abstract models. The SMT solvers are then used to perform bounded model checking to explore a bounded symbolic execution of the model. A number of solvers are available that support the SMT-Lib such as the Beaver, the Boolec-

tor, the CVC4, the MathSAT5, the Z3, and the OpenSMT. The differentiating feature of solvers can be the underlying logic (First Order Logic (FOL) or Temporal Logic), supported theories, input formulas, and interfaces (Cok et al., 2015).

We employed the Z3 constraint solver, which is an efficient automated SMT solver by Microsoft Research Labs. The Z3 solver is mostly used in the analysis and verification of software systems. The underlying verification theory for our system's model is the theory of array that is used to prove the satisfiability of our model's logical formulae. The array theory is frequently used in software modeling domain.

3.3 Modeling and Analysis

The complete HLPN model of the OpenFlow (OF)-enabled switch, an SDN controller and an intrusion detection application built as an extended module of the controller in software-defined network is illustrated in the Figure 3.3; whereas, the sequence diagram is shown in Figure 3.2.

As described in Definition 1 in Section 3.2.1, HLPN is a 7-tuple N = P, T, F, ϕ, R, L, M_0. Before modeling the system, we first need to state P and the associated data types. There are 15 places and 12 transitions in the model, as depicted in Figure 3.3. Table 3.2 contains the names and mappings of places. Table 3.3 contains types used in the model. In the next step, we are defining the set of rules, pre, and post-conditions to map to T. Before writing rules of transitions let us have a quick overview how the OpenFlow-enabled switch, the SDN controller and our proposed application works. The system comprises of an OF-enabled switch, an SDN controller and an intrusion detection application built as an extended module of the controller.

TCP handshaking connection (i.e. TCP three-way handshake) is an automated negotiation process that dynamically sets the communication channel parameters between two different network entities (i.e. OF-enabled switch, and controller) before normal commu-

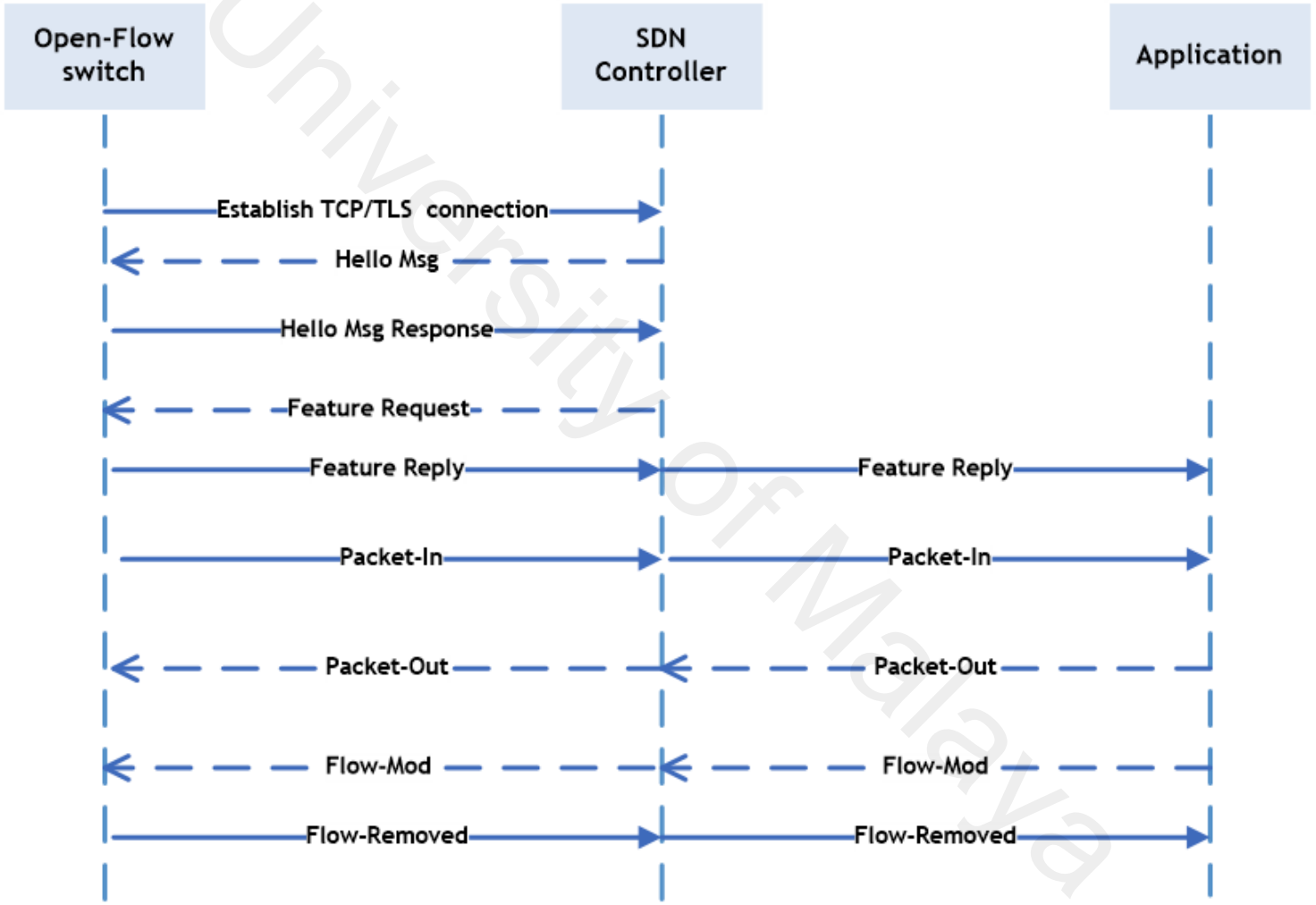

Figure 3.2: Sequence Diagram of the OpenFlow Specified Communication with the Switch, SDN Controller, and the Proposed Application

nication starts. However, the OpenFlow latest versions also supports the optional TLS (Transport Layer Security). We consider here a TLS enabled communication between the switch and the controller. After successful TCP/TLS connection establishment, following are some important messages that are exchanged between the OF-enabled switch and the controller. The first initiative after the TCP establishment is the Hello message exchanged between the switch and the controller. Hello message is a form of symmetric communication that can be sent in either direction, but mainly it is exchanged upon TCP/TLS connection start-up for version negotiation (i.e. OpenFlow version). In our sequence diagram, these messages are just shown for the initial start-up. The rest of the messages (i.e., Features-reply, Packet-In, Packet-Out, Flow-Mod, and Flow-Removed) exchanged among the three entities (switch, controller, and application) as shown in the sequence diagram in Figure 3.2. These messages are considered very important to demonstrate specific diverse sophisticated attacks to prove the correctness of the proposed model and to closely witness and analyze the attacks with TLS-enabled OpenFlow. Moreover, a detailed analysis of the complete OpenFlow protocol messages is beyond the scope of this thesis. Moreover a complete HPLN model for a specified communication between the switch and the controller with detailed transitions are shown in Figure 3.3.

New tokens can only enter in the model through transition ***R*** (Start-Connection). As there is no input arc joining the transition so pre-condition of this transition does not exist. TCP connection three-way handshaking and establishment is by default the first step to start communication between different network entities. We assume the connection establishment already done at this transition and can be shown at the transition ***R*** (Start-Connection).The rules for this transitions is listed as below:

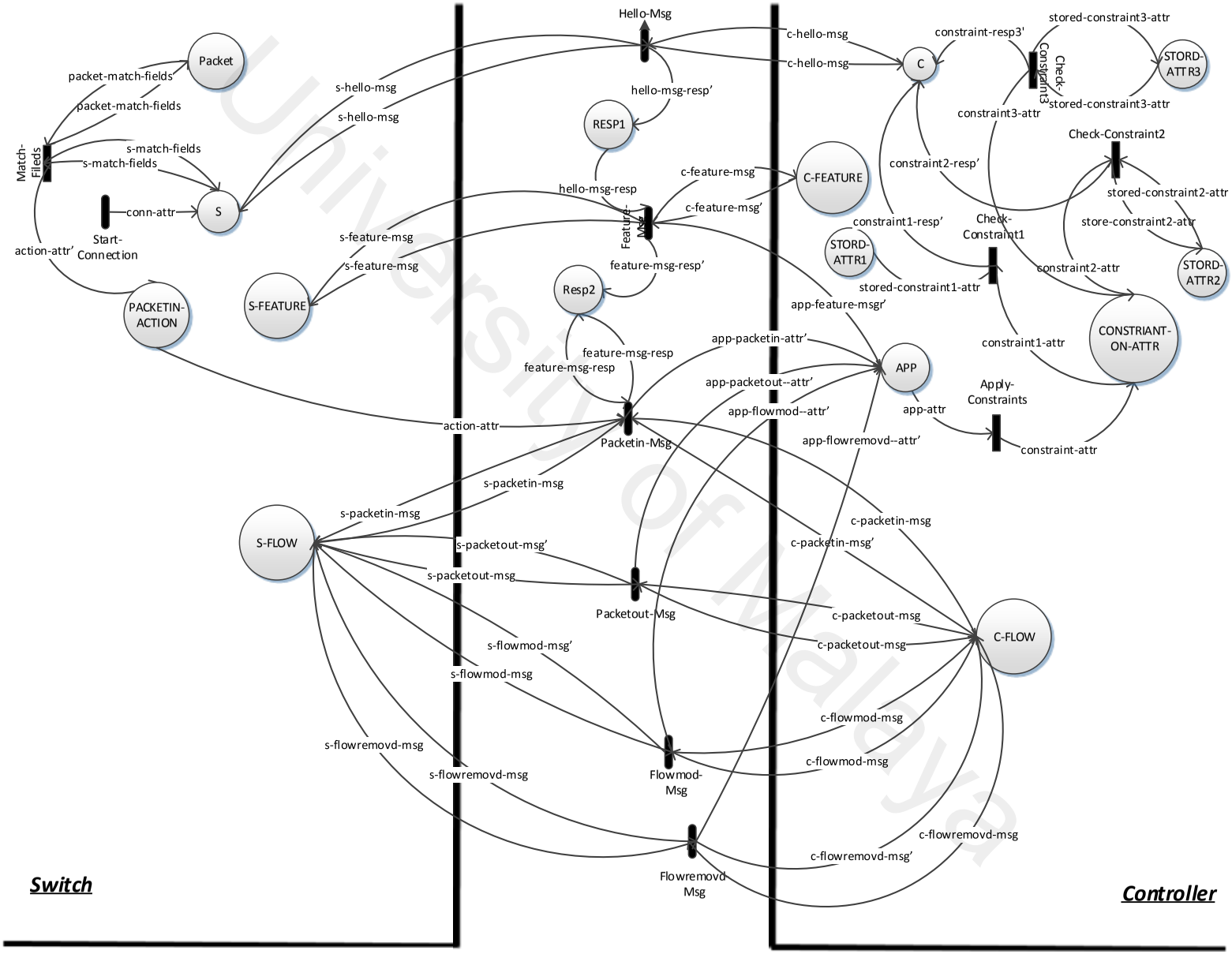

Figure 3.3: An HLPN Model of the OpenFlow Specified Communication with the Switch, SDN Controller, and the Proposed Application

$$= \quad \exists \textit{conn-attr} \quad \in \quad \textit{Conn} \quad - \quad \textit{Attr} \quad | \quad \bullet \textit{conn-attr} \quad = \quad \phi$$

Hello message is a form of symmetric messages and can be sent in either direction, but mainly it is exchanged upon connection start-up. The next transition ***R*** *(Hello-Msg)* occurs when exchanged hello message upon connection startup. The said transition is depicted in the formula below:

$$\begin{aligned}
&\textbf{\textit{R}}\ \textit{(Hello-Msg)} = \forall \textit{s-hello-msg} \in S - \textit{Hello} - \textit{Msg}, \\
&\forall \textit{c-hello-msg} \in C - \textit{Hello} - \textit{Msg}, \forall \textit{hello-msg-resp} \in \\
&\textit{Hello} - \textit{Msg} - \textit{Resp} \mid \textit{s-hello-msg}[4] = \textit{c-hello-msg}[1] \wedge \\
&\textit{s-hello-msg}[5] = \textit{c-hello-msg}[2] \wedge \textit{s-hello-msg}[6] = \textit{c-hello-msg}[3] \wedge \\
&\textit{s-hello-msg}[7] = \textit{c-hello-msg}[4] \rightarrow \\
&\textit{hello} - \textit{msg} - \textit{resp}[1] = \textit{true}, \textit{Hello} - \textit{Msg} - \textit{Resp}' = \textit{Hello} - \textit{Msg} - \textit{Resp} \cup (\textit{hello-msg-resp}[1])
\end{aligned} \tag{3.1}$$

The controller may request the identity and the basic capabilities of a switch by sending a features request message; the switch must respond with a features reply message that specifies the identity and basic capabilities of the switch. This is commonly performed upon establishment of the OpenFlow channel. Upon successful hello message response shown as RESP1 place, the transition ***R*** *(Feature-Msg)* occurs. The transition ***R***

(Feature-Msg) is shown in the rule below:

$$
\begin{aligned}
&\boldsymbol{R}\ \textit{(Feature-Msg)} = \forall \textit{hello-msg-resp} \in Hello - Msg - Resp,\\
&\forall \textit{s-feature-msg} \in S - Feature - Msg,\\
&\forall \textit{c-feature-msg} \in C - Feature - Msg, \forall \textit{feature-msg-resp} \in Feature - Msg - Resp,\\
&\forall \textit{app-feature-msg} \in App - Feature - Msg, |\\
&\textit{hello} - \textit{msg} - \textit{resp}[1] = true \wedge \textit{s-feature-msg}[1] = \textit{c-feature-msg}[1]\\
&\wedge \textit{s-feature-msg}[2] = \textit{c-feature-msg}[2]\\
&\wedge \textit{s-feature-msg}[3] = \textit{c-feature-msg}[3] \wedge\\
&\textit{s-feature-msg}[4] = \textit{c-feature-msg}[4] \rightarrow\\
&\textit{c-feature-msg}[5] := \textit{s-feature-msg}[5] \wedge\\
&\textit{c-feature-msg}[6] := \textit{s-feature-msg}[6] \wedge \textit{c-feature-msg}[7] := \textit{s-feature-msg}[7] \wedge\\
&\textit{c-feature-msg}[8] := \textit{s-feature-msg}[8] \wedge \textit{c-feature-msg}[9] := \textit{s-feature-msg}[9] \wedge\\
&\textit{c-feature-msg}[10] := \textit{s-feature-msg}[10] \wedge \textit{c-feature-msg}[11] := \textit{s-feature-msg}[11] \wedge\\
&\textit{feature-msg-resp}[1] := true \wedge\\
&\textit{app-feature-msg}[1] := \textit{s-feature-msg}[1] \wedge\\
&\textit{app-feature-msg}[2] := \textit{s-feature-msg}[2] \wedge \textit{app-feature-msg}[3] := \textit{s-feature-msg}[3] \wedge\\
&\textit{app-feature-msg}[4] := \textit{s-feature-msg}[4] \wedge \textit{app-feature-msg}[5] := \textit{s-feature-msg}[5] \wedge\\
&\textit{app-feature-msg}[6] := \textit{s-feature-msg}[6] \wedge \textit{app-feature-msg}[7] := \textit{s-feature-msg}[7] \wedge\\
&\textit{app-feature-msg}[8] := \textit{s-feature-msg}[8] \wedge\\
&\textit{app-feature-msg}[9] := \textit{s-feature-msg}[9] \wedge
\end{aligned}
\tag{3.2}
$$

$$\textit{app-feature-msg}[10] := \textit{s-feature-msg}[10] \wedge$$

$$\textit{app-feature-msg}[11] := \textit{s-feature-msg}[11]$$

$$C - Feature - Msg' = C - Feature - Msg \cup (\textit{c-feature-msg}[5], \textit{c-feature-msg}[6],$$

$$\textit{c-feature-msg}[7], \textit{c-feature-msg}[8], \textit{c-feature-msg}[9], \textit{c-feature-msg}[10], \textit{c-feature-msg}[11])$$

$$Feature - Msg - Resp' = Feature - Msg - Resp \cup (\textit{feature-msg-resp}[1]),$$

$$App - Feature - Msg' = App - Feature - Msg \cup (\textit{app-feature-msg}[1],$$

$$\textit{app-feature-msg}[2], \textit{app-feature-msg}[3], \textit{app-feature-msg}[4], \textit{app-feature-msg}[5],$$

$$\textit{app-feature-msg}[6], \textit{app-feature-msg}[7], \textit{app-feature-msg}[8], \textit{app-feature-msg}[9],$$

$$\textit{app-feature-msg}[10], \textit{app-feature-msg}[11]) \quad (3.3)$$

A header/match field is used to describe and compare to which incoming packet this entry is applicable. The rule for the transition ***R*** *(Match-Fields)* is depicted below.

$$\textbf{\textit{R}}\ \textit{(Match-Fields)} = \forall \textit{s-match-fields} \in S - Match - Fields,$$

$$\forall \textit{packet-match-fields} \in Packet - Match - Fields, \forall \textit{action-attr} \in Action - Attr \mid$$

$$\textit{s-match-fields}[8] \neq \textit{packet-match-fields}[2] \vee$$

$$\textit{s-match-fields}[9] \neq \textit{packet-match-fields}[3] \vee$$

$$\textit{s-match-fields}[10] \neq \textit{packet-match-fields}[4] \vee$$

$$\textit{s-match-fields}[11] \neq \textit{packet-match-fields}[5] \vee$$

$$\textit{s-match-fields}[12] \neq \textit{packet-match-fields}[6] \vee$$

$$\textit{s-match-fields}[13] \neq \textit{packet-match-fields}[7] \vee$$

$$\textit{s-match-fields}[14] \neq \textit{packet-match-fields}[8] \rightarrow$$

(3.4)

$$action - attr[1] := packetin(true),$$

$$Action - Attr' = Action - Attr \cup (action\text{-}attr[1]) \quad (3.5)$$

A Packet-In essentially represents a packet that does not match any flow rules at the data plane, and the OF protocol mandates that such packets must be sent by the switch to the controller directly. When a message that is received by an SDN switch with no match entry is directed to a controller by the OF protocol by default, such messages are called Packet-Ins. At present, the control plane has no built-in security mechanism to avoid the manipulation of Packet-In messages even if the OF is TLS enabled. Authorized switches can also send forged Packet-In messages that can subsequently be used to corrupt the controller state by a wide variety of network attacks. Moreover, the Packet-In operational semantics of the Open-Flow lower the barrier of mounting sophisticated attacks on the control plane of the SDNs. Upon failure of match fields response shown as RESP place and successful feature message response, the transition ***R*** *(Packetin-Msg)* occurs. The transition ***R*** *(Packetin-Msg)* is shown in the rule below:

$$\begin{aligned} \textbf{R}\ (Packetin\text{-}Msg) = \forall &feature\text{-}msg\text{-}resp \in Feature - Msg - Resp, \\ &\forall packetin\text{-}action \in Packetin - Action, \\ &\forall s\text{-}packetin\text{-}msg \in S - Packetin - Msg, \\ &\forall c\text{-}packetin\text{-}msg \in C - Packetin - Msg, \forall app\text{-}packetin\text{-}attr \in \\ &App - Packetin - Attr \mid \\ &feature - msg - resp[1] = true \wedge packetin - action[1] = true \rightarrow \end{aligned} \quad (3.6)$$

$$c\text{-}packetin\text{-}msg[1] := s\text{-}packetin\text{-}msg[1] \wedge c\text{-}packetin\text{-}msg[2] := s\text{-}packetin\text{-}msg[2] \wedge$$

$$c\text{-}packetin\text{-}msg[3] := s\text{-}packetin\text{-}msg[3] \wedge c\text{-}packetin\text{-}msg[4] := s\text{-}packetin\text{-}msg[4] \wedge$$

$$c\text{-}packetin\text{-}msg[5] := s\text{-}packetin\text{-}msg[5] \wedge c\text{-}packetin\text{-}msg[6] := s\text{-}packetin\text{-}msg[6] \wedge$$

$$app\text{-}packetin\text{-}attr[11] := s\text{-}packetin\text{-}msg[1] \wedge$$

$$app\text{-}packetin\text{-}attr[12] := s\text{-}packetin\text{-}msg[2] \wedge$$

$$app\text{-}packetin\text{-}attr[2] := s\text{-}packetin\text{-}msg[3] \wedge$$

$$app\text{-}packetin\text{-}attr[13] := s\text{-}packetin\text{-}msg[4] \wedge$$

$$app\text{-}packetin\text{-}attr[14] := s\text{-}packetin\text{-}msg[5] \wedge$$

$$app\text{-}packetin\text{-}attr[15] := s\text{-}packetin\text{-}msg[6] \wedge$$

$$C - Packetin - Msg' = C - Packetin - Msg \cup (c\text{-}packetin\text{-}msg[1], c\text{-}packetin\text{-}msg$$

$$[2], c\text{-}packetin\text{-}msg[3], c\text{-}packetin\text{-}msg[4], c\text{-}packetin\text{-}msg[5], c\text{-}packetin\text{-}msg[6])$$

$$App - Packetin - Attr' = App - Packetin - Attr \cup (app\text{-}packetin\text{-}attr[11], app\text{-}packetin\text{-}attr$$

$$[12], app\text{-}packetin\text{-}attr[2], app\text{-}packetin\text{-}attr[13], app\text{-}packetin\text{-}attr$$

$$[14], app\text{-}packetin\text{-}attr[15]) \quad (3.7)$$

Packet-Out is a controller to switch communication message, where the controller is subject to direct packets out to a particular port on the switch, and also forward the number of packets received via Packet-Ins. Packet-Out messages generally comprises of a full packet or a buffer ID referencing a particular packet, which is stored in the switch. Packet-Out also contain action's list to be exactly followed and an empty action list simply represents packet to be dropped. The resultant and final evaluation outcome of Packet-In message can be a Packet-Out, Flow-Mod or Flow-Removed messages that are stored in C-FLOW place. The transition ***R*** *(Packetout-Msg)* depicts the successful evaluation of the Packet-Out message when the C-FLOW place receives the Packet-Out response. The transition

is illustrated below:

$$\begin{aligned}
&\boldsymbol{R}\ (\textit{Packetout-Msg}) = \forall \textit{c-packetout-msg} \in C - \textit{Packetout} - \textit{Msg},\\
&\forall \textit{s-flowmod-msg} \in S - \textit{Packetout} - \textit{Msg}, \forall \textit{app-packetout-attr} \in\\
&\textit{App} - \textit{Packetout} - \textit{Attr} \mid\\
&c - \textit{packetout} - \textit{msg}[7] = \textit{packetoutresponse} \rightarrow\\
&\textit{s-packetout-msg}[1] := \textit{c-packetout-msg}[1] \wedge \textit{s-packetout-msg}[2] := \textit{c-packetout-msg}[2] \wedge\\
&\textit{s-packetout-msg}[3] := \textit{c-packetout-msg}[3] \wedge \textit{s-packetout-msg}[7] := \textit{c-packetout-msg}[8] \wedge\\
&\textit{s-packetout-msg}[8] := \textit{c-packetout-msg}[9] \wedge \textit{s-packetout-msg}[9] := \textit{c-packetout-msg}[10] \wedge\\
&\textit{s-packetout-msg}[10] := \textit{c-packetout-msg}[11] \wedge\\
&\textit{s-packetout-msg}[11] := \textit{c-packetout-msg}[12] \wedge\\
&\textit{s-packetout-msg}[12] := \textit{c-packetout-msg}[13] \wedge\\
&\textit{s-packetout-msg}[13] := \textit{c-packetout-msg}[14] \wedge\\
&\textit{s-packetout-msg}[14] := \textit{c-packetout-msg}[15] \wedge\\
&\textit{s-packetout-msg}[15] := \textit{c-packetout-msg}[16] \wedge\\
&\textit{app-packetout-attr}[11] := \textit{c-packetout-msg}[1] \wedge \textit{app-packetout-attr}[12] := \textit{c-packetout-msg}[2] \wedge\\
&\textit{app-packetout-attr}[2] := \textit{c-packetout-msg}[3] \wedge \textit{app-packetout-attr}[16] := \textit{c-packetout-msg}[8] \wedge\\
&\textit{app-packetout-attr}[17] := \textit{c-packetout-msg}[9] \wedge\\
&\textit{app-packetout-attr}[18] := \textit{c-packetout-msg}[10] \wedge\\
&\textit{app-packetout-attr}[19] := \textit{c-packetout-msg}[11] \wedge\\
&\textit{app-packetout-attr}[20] := \textit{c-packetout-msg}[12] \wedge\\
&\textit{app-packetout-attr}[21] := \textit{c-packetout-msg}[13] \wedge\\
&\textit{app-packetout-attr}[22] := \textit{c-packetout-msg}[14] \wedge
\end{aligned} \tag{3.8}$$

$$app\text{-}packetout\text{-}attr[23] := c\text{-}packetout\text{-}msg[15] \wedge$$

$$app\text{-}packetout\text{-}attr[24] := c\text{-}packetout\text{-}msg[16]$$

$$S - Packetout - Msg' = s - Packetout - Msg \cup (s\text{-}packetout\text{-}msg[1], s\text{-}packetout\text{-}msg$$

$$[2], s\text{-}packetout\text{-}msg[3], s\text{-}packetout\text{-}msg[7], s\text{-}packetout\text{-}msg[8],$$

$$s\text{-}packetout\text{-}msg[9], s\text{-}packetout\text{-}msg[10], s\text{-}packetout\text{-}msg[11],$$

$$s\text{-}packetout\text{-}msg[12], s\text{-}packetout\text{-}msg[13], s\text{-}packetout\text{-}msg[14], s\text{-}packetout\text{-}msg[15])$$

$$App - Packetout - Attr' = App - Packetout - Attr \cup (app\text{-}packetout\text{-}attr[11],$$

$$app\text{-}packetout\text{-}attr[12], app\text{-}packetout\text{-}attr[2], app\text{-}packetout\text{-}attr[16],$$

$$app\text{-}packetout\text{-}attr[17], app\text{-}packetout\text{-}attr[18], app\text{-}packetout\text{-}attr$$

$$[19], app\text{-}packetout\text{-}attr[20], app\text{-}packetout\text{-}attr[21],$$

$$app\text{-}packetout\text{-}attr[22], app\text{-}packetout\text{-}attr[23], app\text{-}packetout\text{-}attr[24]) \quad (3.9)$$

Flow-Mod is one of the main messages that allows the controller to modify the state of an OF-enabled switch. The transition ***R*** *(Flow-Mod-Msg)* depicts the successful evaluation of the flow-Mod message when the C-FLOW place receives the flow-Mod response. The transition is illustrated below:

$$\boldsymbol{R}\ (Flowmod\text{-}Msg) = \forall c\text{-}flowmod\text{-}msg \in C - Flowmod - Msg,$$

$$\forall s\text{-}flowmod\text{-}msg \in S - Flowmod - Msg, \forall app\text{-}flowmod\text{-}attr \in$$

$$App - Flowmod - Attr \mid$$

$$c - flowmod - msg[17] = flowmodresponse \rightarrow$$

(3.10)

$$\textit{s-flowmod-msg}[1] := \textit{c-flowmod-msg}[1] \wedge \textit{s-flowmod-msg}[2] := \textit{c-flowmod-msg}[2] \wedge$$

$$\textit{s-flowmod-msg}[3] := \textit{c-flowmod-msg}[3] \wedge \textit{s-flowmod-msg}[16] := \textit{c-flowmod-msg}[18] \wedge$$

$$\textit{s-flowmod-msg}[17] := \textit{c-flowmod-msg}[19] \wedge \textit{s-flowmod-msg}[18] := \textit{c-flowmod-msg}[20] \wedge$$

$$\textit{s-flowmod-msg}[8] := \textit{c-flowmod-msg}[9] \wedge$$

$$\textit{s-flowmod-msg}[5] := \textit{c-flowmod-msg}[5] \wedge$$

$$\textit{app-flowmod-attr}[11] := \textit{c-flowmod-msg}[1] \wedge \textit{app-flowmod-attr}[12] := \textit{c-flowmod-msg}[2] \wedge$$

$$\textit{app-flowmod-attr}[2] := \textit{c-flowmod-msg}[3] \wedge \textit{app-flowmod-attr}[25] := \textit{c-flowmod-msg}[8] \wedge$$

$$\textit{app-flowmod-attr}[26] := \textit{c-flowmod-msg}[9] \wedge \textit{app-flowmod-attr}[27] := \textit{c-flowmod-msg}[10] \wedge$$

$$\textit{app-flowmod-attr}[17] := \textit{c-flowmod-msg}[11] \wedge$$

$$\textit{app-flowmod-attr}[14] := \textit{c-flowmod-msg}[12] \wedge$$

$$S - Flowmod - Msg' = S - Flowmod - Msg \cup (\textit{s-flowmod-msg}[1], \textit{s-flowmod-msg}$$

$$[2], \textit{s-flowmod-msg}[3], \textit{s-flowmod-msg}[18], \textit{s-flowmod-msg}[19],$$

$$\textit{s-flowmod-msg}[20], \textit{s-flowmod-msg}[9], \textit{s-flowmod-msg}[5])$$

$$App - Flowmod - Attr' = App - Flowmod - Attr \cup (\textit{app-flowmod-attr}[11],$$

$$\textit{app-flowmod-attr}[12], \textit{app-flowmod-attr}[2], \textit{app-flowmod-attr}[25],$$

$$\textit{app-flowmod-attr}[26], \textit{app-flowmod-attr}[27], \textit{app-flowmod-attr}[17],$$

$$\textit{app-flowmod-attr}[14]) \quad (3.11)$$

The Flow-Removed message is used to inform the controller about the removal of a flow-entry from a flow table. These messages are subject to be generated, when the SDN controller asks to delete a flow-entry They are generated as the result of a controller flow delete requests or when the flow timeout exceeds of a flow-entry (i.e. switch flow expiry process starts). The transition ***R*** *(Flow-Removed-Msg)* shows the successful evaluation of the flow removed message when the C-FLOW place evaluates and receives the Flow-

Removed response. The transition is illustrated below:

$$
\begin{aligned}
&\boldsymbol{R}\ (\textit{Flowremovd-Msg}) = \forall \textit{s-flowremovd-msg} \in S - Flowremovd - Msg, \\
&\forall \textit{c-flowremovd-msg} \in S - Flowremovd - Msg, \forall \textit{app-flowremovd-attr} \in \\
&App - Flowremovd - Attr \mid \\
&s - flowremovd - msg[19] = flowmodresponse \rightarrow \\
&\textit{c-flowremovd-msg}[1] := \textit{s-flowremovd-msg}[1] \wedge \\
&\textit{c-flowremovd-msg}[2] := \textit{s-flowremovd-msg}[2] \wedge \\
&\textit{c-flowremovd-msg}[3] := \textit{s-flowremovd-msg}[3] \wedge \\
&\textit{c-flowremovd-msg}[19] := \textit{s-flowremovd-msg}[17] \wedge \\
&\textit{c-flowremovd-msg}[21] := \textit{s-flowremovd-msg}[20] \wedge \\
&\textit{c-flowremovd-msg}[4] := \textit{s-flowremovd-msg}[4] \wedge \\
&\textit{c-flowremovd-msg}[22] := \textit{s-flowremovd-msg}[21] \wedge \\
&\textit{c-flowremovd-msg}[23] := \textit{s-flowremovd-msg}[22] \wedge \\
&\textit{c-flowremovd-msg}[24] := \textit{s-flowremovd-msg}[23] \wedge \\
&\textit{app-flowremovd-attr}[11] := \textit{s-flowremovd-msg}[1] \wedge \\
&\textit{app-flowremovd-attr}[12] := \textit{s-flowremovd-msg}[2] \wedge \\
&\textit{app-flowremovd-attr}[2] := \textit{s-flowremovd-msg}[3] \wedge \\
&\textit{app-flowremovd-attr}[26] := \textit{s-flowremovd-msg}[17] \wedge \\
&\textit{app-flowremovd-attr}[28] := \textit{s-flowremovd-msg}[20] \\
&\textit{app-flowremovd-attr}[13] := \textit{s-flowremovd-msg}[4] \\
&\textit{app-flowremovd-attr}[29] := \textit{s-flowremovd-msg}[21]
\end{aligned}
\tag{3.12}
$$

$$app\text{-}flowremovd\text{-}attr[30] := s\text{-}flowremovd\text{-}msg[22]$$

$$app\text{-}flowremovd\text{-}attr[31] := s\text{-}flowremovd\text{-}msg[23]$$

$$C-Flowremovd-Msg' = C-Flowremovd-Msg \cup (c\text{-}flowremovd\text{-}msg[1],$$

$$c\text{-}flowremovd\text{-}msg[2], c\text{-}flowremovd\text{-}msg[3], c\text{-}flowremovd\text{-}msg[18],$$

$$c\text{-}flowremovd\text{-}msg[19], c\text{-}flowremovd\text{-}msg[20],$$

$$c\text{-}flowremovd\text{-}msg[9], c\text{-}flowremovd\text{-}msg[5])$$

$$App-Flowmod-Attr' = App-Flowmod-Attr \cup (app\text{-}flowremovd\text{-}attr[11],$$

$$app\text{-}flowremovd\text{-}attr[12],$$

$$app\text{-}flowremovd\text{-}attr[2], app\text{-}flowremovd\text{-}attr[25],$$

$$app\text{-}flowremovd\text{-}attr[26], app\text{-}flowremovd\text{-}attr[27],$$

$$app\text{-}flowremovd\text{-}attr[17],$$

$$app\text{-}flowremovd\text{-}attr[14]) \quad (3.13)$$

Application pre-processes the relevant attributes from the messages. However, finding the relevant attributes (feature selection) for prediction of the varied network attack patterns is done through Weka. The application pre-processes these attributes in an offline fashion while testing in real-time. The transition ***R*** *(Apply-Constraints)* takes the relevant attributes of messages and apply constraints on them and store them in CONSTRAINT-ON-ATTR place.The transition is illustrated below:

$$\textbf{\textit{R}}\ (Apply\text{-}Constraints) = \forall app\text{-}attr \in App-Attr,$$

$$\forall constraint\text{-}attr \in Constraint-Attr \mid \quad (3.14)$$

$$\textit{constraint-attr}[1] := Apply-Constraint(\textit{app-attr}[19], \textit{app-attr}[32]),$$

$$\textit{constraint-attr}[2] := Apply-Constraint(\textit{app-attr}[20], \textit{app-attr}[32]),$$

$$\textit{constraint-attr}[3] := Apply-Constraint(\textit{app-attr}[29], \textit{app-attr}[32]),$$

$$\textit{constraint-attr}[4] := Apply-Constraint(\textit{app-attr}[31], \textit{app-attr}[32]),$$

$$\textit{constraint-attr}[5] := Apply-Constraint(\textit{app-attr}[19], \textit{app-attr}[33]),$$

$$\textit{constraint-attr}[6] := Apply-Constraint(\textit{app-attr}[20], \textit{app-attr}[33]),$$

$$\textit{constraint-attr}[7] := Apply-Constraint(\textit{app-attr}[29], \textit{app-attr}[33]),$$

$$\textit{constraint-attr}[8] := Apply-Constraint(\textit{app-attr}[30], \textit{app-attr}[33]),$$

$$\textit{constraint-attr}[9] := Apply-Constraint(\textit{app-attr}[31], \textit{app-attr}[33]),$$

$$\textit{constraint-attr}[10] := Apply-Constraint(\textit{app-attr}[19], \textit{app-attr}[34]),$$

$$\textit{constraint-attr}[11] := Apply-Constraint(\textit{app-attr}[20], \textit{app-attr}[34]),$$

$$\textit{constraint-attr}[12] := Apply-Constraint(\textit{app-attr}[28], \textit{app-attr}[34]),$$

$$\textit{constraint-attr}[13] := Apply-Constraint(\textit{app-attr}[29], \textit{app-attr}[34]),$$

$$\textit{constraint-attr}[14] := Apply-Constraint(\textit{app-attr}[30], \textit{app-attr}[34]),$$

$$\textit{constraint-attr}[15] := Apply-Constraint(\textit{app-attr}[31], \textit{app-attr}[34])$$

$$Constraint-Attr' = Constraint-Attr \cup (\textit{constraint-attr}[1],$$

$$\textit{constraint-attr}[2], \textit{constraint-attr}[3], \textit{constraint-attr}[4], \textit{constraint-attr}[5],$$

$$\textit{constraint-attr}[6], \textit{constraint-attr}[7], \textit{constraint-attr}[8], \textit{constraint-attr}[9],$$

$$\textit{constraint-attr}[10], \textit{constraint-attr}[11], \textit{constraint-attr}[12], \textit{constraint-attr}[13],$$

$$\textit{constraint-attr}[14], \textit{constraint-attr}[15]) \quad (3.15)$$

The transition ***R*** *(Check-Constraint1)* compares the constrained applied attributes of the

messages to the offline stored attributes of the messages and evaluates a response that is sent to the controller. The response helps the controller to analyzes thoroughly the network attack patterns extracted triggered by large scale malicious events that predominantly target the control plane to degrade the overall performance of the SDNs or bring the entire network down in the worst case. The transition is illustrated below:

$$\begin{aligned}
\boldsymbol{R}\ \textit{(Check-Constraint1)} = \forall &\textit{stored-constraint1-attr} \in \textit{Stored} - \textit{Constraint}1 - \textit{Attr}, \\
&\forall \textit{constraint1-attr} \in \textit{Constraint}1 - \textit{Attr}, \\
&\forall \textit{constraint1-resp} \in \textit{Constraint}1 - \textit{Resp} \mid \\
&\textit{constraint1-attr}[1] \geq \textit{stored-constraint1-attr}[1] \wedge \\
&\textit{constraint1-attr}[2] \geq \textit{stored-constraint1-attr}[2] \wedge \\
&\textit{constraint1-attr}[3] \geq \textit{stored-constraint1-attr}[3] \wedge \\
&\textit{constraint1-attr}[4] \geq \textit{stored-constraint1-attr}[4] \rightarrow \\
&\textit{constraint1-resp}[1] = \textit{Resp}, \\
\textit{Constraint}1 - \textit{Resp}' = \textit{Constraint}1 - \textit{Resp} &\cup (\textit{constraint1-resp}[1]) \quad (3.16)
\end{aligned}$$

The transition ***R*** *(Check-Constraint2)* compares the constrained applied attributes of the messages to the offline stored attributes of the messages and evaluates a response that is sent to the controller. The response helps the controller to analyzes thoroughly the network attack patterns extracted triggered by large scale malicious events that predominantly target the control plane to degrade the overall performance of the SDNs or bring

the entire network down in the worst case. The transition is illustrated below:

$$\begin{aligned}
&\boldsymbol{R}\ \textit{(Check-Constraint2)} = \forall \textit{stored-constraint2-attr} \in \textit{Stored} - \textit{Constraint2} - \textit{Attr},\\
&\forall \textit{constraint2-attr} \in \textit{Constraint2} - \textit{Attr},\\
&\forall \textit{constraint2-resp} \in \textit{Constraint2} - \textit{Resp} \mid\\
&\textit{constraint2-attr}[5] \geq \textit{stored-constraint2-attr}[1] \wedge\\
&\textit{constraint2-attr}[6] \geq \textit{stored-constraint2-attr}[2] \wedge\\
&\textit{constraint2-attr}[7] \geq \textit{stored-constraint2-attr}[3] \wedge\\
&\textit{constraint2-attr}[8] \geq \textit{stored-constraint2-attr}[4] \wedge\\
&\textit{constraint2-attr}[9] \geq \textit{stored-constraint2-attr}[5] \rightarrow\\
&\textit{constraint2-resp}[2] = \textit{Resp},\\
&\textit{Constraint2} - \textit{Resp}' = \textit{Constraint2} - \textit{Resp} \cup (\textit{constraint2-resp}[1])
\end{aligned} \tag{3.17}$$

The transition ***R*** *(Check-Constraint3)* compares the constrained applied attributes of the messages to the offline stored attributes of the messages and evaluates a response that is sent to the controller. The response helps the controller to analyzes thoroughly the network attack patterns extracted triggered by large scale malicious events that predominantly target the control plane to degrade the overall performance of the SDNs or bring

the entire network down in the worst case. The transition is illustrated below:

$$\begin{aligned}
&\boldsymbol{R}\ \textit{(Check-Constraint3)} = \forall \textit{stored-constraint3-attr} \in \textit{Stored} - \textit{Constraint3} - \textit{Attr},\\
&\forall \textit{constraint3-attr} \in \textit{Constraint3} - \textit{Attr},\\
&\forall \textit{constraint3-resp} \in \textit{Constraint3} - \textit{Resp} \mid\\
&\textit{constraint3-attr}[10] \geq \textit{stored-constraint3-attr}[1] \wedge\\
&\textit{constraint3-attr}[11] \geq \textit{stored-constraint3-attr}[2] \wedge\\
&\textit{constraint3-attr}[12] \geq \textit{stored-constraint3-attr}[3] \wedge\\
&\textit{constraint3-attr}[13] \geq \textit{stored-constraint3-attr}[4] \wedge\\
&\textit{constraint3-attr}[14] \geq \textit{stored-constraint3-attr}[5] \wedge\\
&\textit{constraint3-attr}[15] \geq \textit{stored-constraint3-attr}[6] \rightarrow\\
&\textit{constraint3-resp}[3] = \textit{Resp},\\
&\textit{Constraint3} - \textit{Resp}' = \textit{Constraint3} - \textit{Resp} \cup (\textit{constraint3-resp}[1]) \quad (3.18)
\end{aligned}$$

Table 3.2: Places and Mapping of the OpenFlow Based SDN Verification

Places	**Mapping**
$\varphi(S)$	(syn×ack×syn-ack×version×type×length× xid)
$\varphi(C)$	(syn×ack×syn-ack×version×type×length× xid×respcostraint1× respconstraint2× respconstraint3)
$\varphi(S-FEATURE)$	(version×type× length×xid× datapath_id× n_buffers× n_tables×pad× feature capabilities× feature actions × ports)
$\varphi(C-FEATURE)$	(version×type× length×xid× datapath_id× n_buffers× n_tables×pad× feature capabilities× feature actions × ports)
$\varphi(S-FLOW)$	(Dpid×Time ×Type× Reason× Buffer ID × Payload× Payload (Pout Payload) ×actions× bufferIDSet× ×similarPacketcounts× PacketInexist× EtherType× length× Protocol× Type_s× command× match× idle timeout× Flowremovedresponse × FlowModsBefore× duration× Byte-count× Packet_count)
$\varphi(C-FLOW)$	(Dpid×Time ×Type× Reason× Buffer ID × Payload× Packetoutresponse× Payload (Pout Payload) ×actions× bufferIDSet× similarPacketcounts× PacketInexist× EtherType× length× Protocol× Type_s× Flowmodresponse× command× match× idle timeout× FlowModsBefore× duration× Byte-count× Packet_count)
$\varphi(APP)$	(version×type× length×xid× n_buffers× n_tables×pad× feature capabilities× feature actions × ports×Dpid×Time ×Type× Reason× Buffer ID × Payload× Payload (Pout Payload) ×actions× bufferIDSet× ×similarPacketcounts× PacketInexist× EtherType× length× Protocol× Type_s× command× match× idle timeout× FlowModsBefore× duration×Byte-count× Packet_count×Constraint1×Constraint2 ×Constraint3)

$\varphi(Constraint-Attributes)$	(Constraint1similarPacketcounts× Constraint1PacketInexist× Constraint1duration× Constraint1Packet_count× Constriant2similarPacketcounts× Constraitn2PacketInexist× Constraint2Duration× Constriant2Byte-count× Constriant2Packet_count× Constriant3similarPacketcounts× Constraint3PacketInexist× Constriaint3flowmodsbefore× Constriaint3Duration× Constriant3Byte-count× Constraint3Packet_count)
$\varphi(Stored-constraint1)$	(Constraint1similarPacketcounts× Constraint1PacketInexist× Constraint1duration× Con-straint1Packet_count)
$\varphi(Stored-constraint2)$	(Constriant2similarPacketcounts× Constraitn2PacketInexist× Constraint2Duration× Constriant2Byte-count× Constriant2Packet_count)
$\varphi(Stored-constraint3)$	(Constriant3similarPacketcounts× Constraint3PacketInexist× Constriaint3flowmodsbefore× Constriaint3Duration× Constriant3Byte-count× Constraint3Packet_count)
$\varphi(Packet)$	(Protocol-type × Src-Mac-Add× Dest-Mac-Add× Src-ip × Dest-ip × Src-port × Dest-port × Service × num-bytes-src-dst × num-bytes-dst-src × Fr-no × Fr-len × Cap-len × Head-len × Frag-off × TTL × Seq-no × CWR × ECN × URG × ACK × PSH × RST × SYN × FIN × Land × Mss-src-dest-requested × Mss-dest-src-requested × Ttt-len-src-dst × Ttt-len-dst-src × Conn-status × count-fr-dest × count-fr-src × count-serv-src × count-serv-dest× num-pushed-src-dst × num-pushed-dst-src × num-SYN-FIN-src-dst × num-SYN-FIN-dst-src × num-FIN-src-dst × num-FIN-dst-src × count-dest-conn × count-src-conn × count-serv-srcconn × count-serv-destconn × num-packets-src-dst × num-packets-dst-src × num-acks-src-dst × num-acks-dst-src × num-retransmit-src-dst× num-retransmit-dst-src)

Table 3.3: Data-types Used in the Algorithms

Data types	**Description**
luser	A string type for the entity
Syn	A string type for the entity
Ack	A string type for the entity
Syn-Ack	A string type for the entity
version	An integer type for the entity
type	A string type for the entity
length	An integer type for the entity
xid	A string type for the entity
respconstraint1	A string type for the entity
respconstraint2	A string type for the entity
respconstraint3	A string type for the entity
datapath_id	A string type for the entity
n_buffers	An integer type for the entity
n_tables	An integer type for the entity
pad	A string type for the entity
feature capabilities	A string type for the entity
feature actions	A string type for the entity
ports	A string type for the entity
Dpid	A string type for the entity
Time	A integer for the entity
Type	A string type for the entity
Reason	A string type for the entity
Buffer ID	A string type for the entity
Payload	A Boolean type for the entity
Payload (Pout Payload)	A Boolean type for the entity
actions	A string type for the entity
bufferIDSet	A Boolean type for the entity
similarPacketcounts	An integer type for the entity
PacketInexist	A Boolean type for the entity
EtherType	A string type for the entity
length	An integer type for the entity
Protocol	A string type for the entity
Type_s	A string type for the entity
command	A string type for the entity
match	A string type for the entity
idle timeout	An integer type for the entity
Flowremovedresponse	A string type for the entity
FlowModsBefore	An integer type for the entity
duration	An integer type for the entity

Byte-count	An integer type for the entity
Packet_count	A integer type for the entity
Packetoutresponse	A string type for the entity
Flowmodresponse	A string type for the entity
Constraint1	A string type for the entity
Constraint2	A string type for the entity
Constraint3	A string type for the entity
Constraint1similarPacketcounts	An integer type for the entity
Constraint1PacketInexist	A Boolean type for the entity
Constraint1duration	An integer type for the entity
Constraint1Packet_count	An integer type for the entity
Constriant2similarPacketcounts	An integer type for the entity
Constraitn2PacketInexist	A Boolean type for the entity
Constraint2Duration	An integer type for the entity
Constriant2Byte-count	An integer type for the entity
Constriant2Packet_count	An integer type for the entity
Constriant3similarPacketcounts	An integer type for the entity
Constraint3PacketInexist	A Boolean type for the entity
Constriaint3flowmodsbefore	An integer type for the entity
Constriaint3Duration	An integer type for the entity
Constriant3Byte-count	An integer type for the entity
Constraint3Packet_count	An integer type for the entity
Protocol-type	A string type for the entity
Src-Mac-Add	A string type for the entity
Dest-Mac-Add	A string type for the entity
Src-ip	A string type for the entity
Dest-ip	A string type for the entity
Src-port	A string type for the entity
Dest-port	A string type for the entity
Service	A string type for the entity
num-bytes-src-dst	An integer type for the entity
num-bytes-dst-src	An integer type for the entity
Fr-no	An integer type for the entity
Fr-len	An integer type for the entity
Cap-len	An integer type for the entity
Head-len	An integer type for the entity
Frag-off	A Boolean type for the entity
TTL	An integer type for the entity
Seq-no	An integer type for the entity
CWR	A string type for the entity
ECN	A string type for the entity
URG	A string type for the entity

ACK	A string type for the entity
PSH	A string type for the entity
RST	A string type for the entity
SYN	A string type for the entity
FIN	A string type for the entity
Land	A string type for the entity
Mss-src-dest-requested	A string type for the entity
Mss-dest-src-requested	A string type for the entity
Ttt-len-src-dst	An integer type for the entity
Ttt-len-dst-src	An integer type for the entity
Conn-status	An integer type for the entity
count-fr-dest	An integer type for the entity
count-fr-src	An integer type for the entity
count-serv-src	An integer type for the entity
count-serv-dest	An integer type for the entity
num-pushed-src-dst	An integer type for the entity
num-pushed-dst-src	An integer type for the entity
num-SYN-FIN-src-dst	An integer type for the entity
num-SYN-FIN-dst-src	An integer type for the entity
num-FIN-src-dst	An integer type for the entity
num-FIN-dst-src	An integer type for the entity
count-dest-conn	An integer type for the entity
count-src-conn	An integer type for the entity
count-serv-srcconn	An integer type for the entity
count-serv-destconn	An integer type for the entity
num-packets-src-dst	An integer type for the entity
num-packets-dst-src	An integer type for the entity
num-acks-src-dst	An integer type for the entity
num-acks-dst-src	An integer type for the entity
num-retransmit-src-dst	An integer type for the entity
num-retransmit-dst-src	An integer type for the entity

3.4 Verification

This section states the verification process using SMT-Lib Z3 solver. We used a model checker to automatically verify if the specifications have been satisfied by the model. We verified property 1, property 2, and property 3 as correctness properties. The correctness properties are illustrated in Section 3.4.1 and the results are provided in Section 3.4.2.

3.4.1 Properties

Property 1: Directed Denial of Service (DoS) Attack

The four essential attributes *(i.e., similarPacketcounts, PacketInexist, duration, and Packet_count)* can help identify directed denial of service (DoS) attack using Packet-Ins in the control plane that wrests either partial control or put the entire SDN network down. Packet-In flooding is one of the practical forms of launching a directed DoS attack against all major OpenFlow based SDN controllers, which places the controller in an unpredictable state. Major SDN controllers are still a target of this attack. A directed DoS attack is a more serious concern in the centralized control architecture of the SDNs. The PacketInexist is a Boolean variable as shown in the data-types used for the proposed model. If PacketInexist equals to true, and the controller is targeted with huge number of similar Packets for a specified duration will place the controller in an un-predictable condition throwing the entire network into chaos. Moreover, directed DoS attacks using Packet-Ins can be carried out in many ways. Particularly, we meant here to continuously target the control plane of the SDN with valid and un-forged Packet-Ins for a certain period of time and the aforementioned attributes can help identify the direct DoS attacks. Moreover, the TLS-enabled control channel cannot even prevent such attacks. The result that we obtained against this assertion is unsat. The code snippet for property 1 is shown below.

```
(assert(not(and(>=(select Constriant1similarPacketcounts 1)(select Constriant1similarPacketcounts 1)) (>=(select Constraint1PacketInexist 2)(select Constraint1PacketInexist 2))(>=(select Constriaint1Duration 3)(select Constriaint1Duration 3)) (>=(select Constraint1Packet_count 4)(select Constraint1Packet_count 4)))))(check-sat)
```

Property 2: Distributed Denial of Service (DDoS) Attack

The essential attributes *(i.e., similarPacketcounts, flowmodsbefore, PacketInexist, Byte-count, duration, Packet_count)* can help identify network sophisticated attacks triggered by large-scale malicious events that predominantly target the SDN controller to degrade the overall performance of the SDNs or bring the entire network down, in the worst case, even if the Open-Flow is Transport Layer Security (TLS) enabled. The flowmodsbefore attribute shows the number of similar flows from a source that target the control plane for a specified time. These attacks typically represent distributed denial of service attacks (DDoS), which is a serious threat in the Internet and particularly becomes even more serious when targeting the centralized architecture of the SDN (i.e., Control Plane Controller). There are many ways to launch DDoS attacks; however, a DDoS attacks can easily be identified using the aforementioned attributes. Moreover, the PacketInexist attribute shows the possibility of manipulating Packet-Ins for diverse DDoS of attacks. The result that we obtained against this assertion is unsat.The code snippet for property 2 is shown below.

```
(assert(not(and(>=(select Constriant2similarPacketcounts 5)(select Constriant2similarPacketcounts 1))(>= (select Constraint2PacketInexist 6)(select Constraint2PacketInexist 2))(>=(select Constriaint2Duration 7)(select Constriaint2Duration 3))(>= (select Constriant2Bytecount 8)(select Constriant2Bytecount 4))(>=(select Constraint2Packet_count 9)(select Constraint2Packet_count 5)))))(check-sat)
```

Property 3: Denial of Service (DoS) Attack Using Forged Packet-Ins

The five essential attributes *(i.e., similarPacketcounts, PacketInexist, Byte-count, duration, Packet_count)* can help identify denial of service (DoS) attack using authorized but forget Packet-Ins in the control plane that wrests either partial or full control of the SDN controller. This is also a practical forms of launching a DoS attack against major OpenFlow based SDN controllers, which places the controller in an unpredictable state. The additional attribute byte-count can precisely help in identifying such attacks. The rest of the description of the attack is almost similar. The only difference here is that the authorized switches send forged Packet-In messages that can subsequently be used to corrupt the controller state while throwing the entire network into chaos. The result that we obtained against this assertion is unsat. The code snippet for property 3 is shown below.

```
(assert(not(and(>=(select Constriant3similarPacketcounts 10)(select Constriant3similarPacketcounts 1)) (>=(select Constraint3PacketInexist 11)(select Constraint3PacketInexist 2)) (>=(select Constriaint3flowmodsbefore 12)(select Constriaint3flowmodsbefore 3)) (>=(select Constriaint3Duration 13)(select Constriaint3Duration 4)) (>=(select Constriant3Bytecount 14)(select Constriant3Bytecount 5)) (>=(select Constraint3Packet_count 15)(select Constraint3Packet_count 6)))))(check-sat)
```

3.4.2 Results

Executing the SDN model in *Z*3 solver along with asserted properties reveal the results that the model is functioning correctly. Note that our goal in this research is to verify the correctness of the models without measuring or analyzing the performance of the system.

Fig 3.4 depicts the execution time taken by *Z*3 solver on each security property of SDN model. The values in Table 3.4 illustrate that the verification of the proposed model in the real setting of the SDN ends up in finite time. The solver's execution time of proposed model is presented in Table 3.4.

Table 3.4: Execution Time of Diverse Security Properties

Security Property	**Execution Time**
property1	0.34 sec
Property2	0.37 sec
property3	0.22 sec

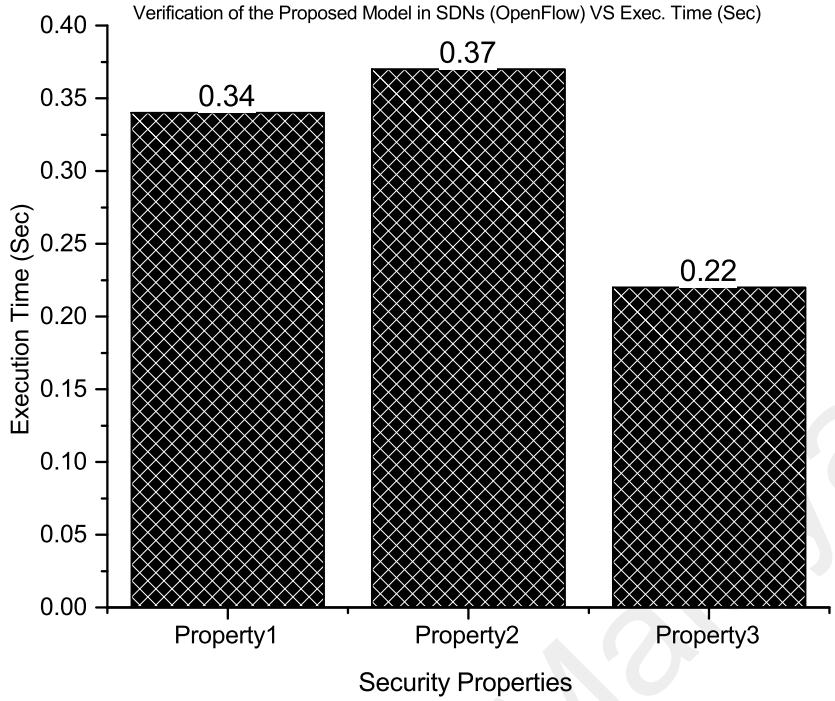

Figure 3.4: Verification Results of the Proposed Model

3.5 Impact of Attack Analysis on SDN Controller

The section presents the impact and analysis of flooding attack in the control plane of the SDNs. We already have a detailed discussion on diverse practical attacks on the control plane of the SDNs in the literature review. However, this section is dedicated to show briefly the impact and analysis of control plane flooding attack in different scenarios. Moreover, the analysis carried out here is simply to demonstrate the severity of attack in the control plane. The impact and analysis of the attacks are evaluated in terms of connection loss and connection set-up latency. The exact reason behind the potentially augmented severity of the attacks is nothing but the centralized architecture of the SDNs. The saturation of attack simply means throwing the entire network into chaos or putting the complete network down, in the worst case.

For the experimental set-up, we use OpenDaylight v1.1.0 (ODL) (OpenDaylight, 2015), a popular SDN controller. Moreover, we employ Mininet (mininet, 2015) (Lantz,

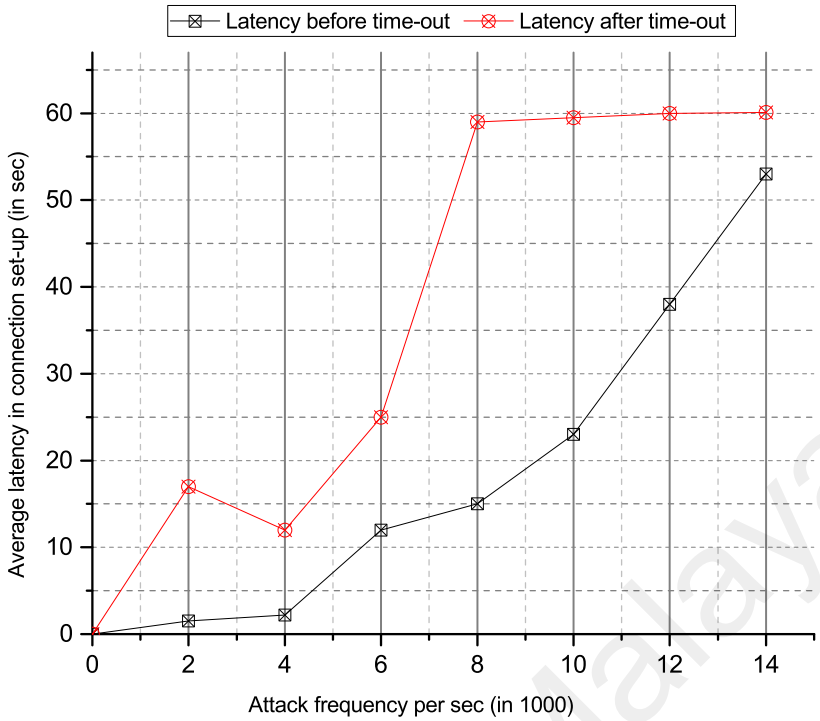

Figure 3.5: Average Latency When the Attack is Launched from a Remote Host on Different Network

Heller, & McKeown, 2010), which realistically creates a virtual network on a computer in-order to emulate the SDN setting. The experimental set-up is implemented on a Lab system Intel ® core i5- 2500M CPU at 3.30 GHz, and 8 GB RAM. Furthermore, for emulating the attacks and analysis, we employed tools such as Scapy (Scapy, 2015) and Cbench (Cbench, 2015; Tootoonchian, Gorbunov, Ganjali, Casado, & Sherwood, 2012). The flow time-out is set to 15s. Moreover, the maximum connection timeout is set to 60s, and after the maximum limit; the requests are considered as lost. Consequently, we take the average of all runs.

Figure 3.5 and Figure 3.6 represents the results of a scenario, where we launch a controller DoS (denial of service) attack using Packet-Ins from a remote host on different network. Figure 3.5 evidently shows the extreme severity of the Packet-In flooding attack in the control plane. The connection set-up latencies reach to 53s even before time-out. The attack consumes the maximum controller resources because the flow rules are

disposed due to a large number of Packet-Ins. However, in case of after time-out; the connection set-up latencies saturation starts at 8kbps. On the other hand, Figure 3.6 shows the loss fraction in connection set-up. Figure 3.6 clearly shows that the loss fraction in connection saturation starts just at 6kbps, in case the requests are sent after time-out.

Figure 3.7 and Figure 3.8 represents a scenario, where we launch an a controller DoS (denial of service) attack using Packet-Ins from a host on the same network. We observe quite high connection set-up latency, when the requests are sent after the time-out period. The connection set-up latencies reach to 24s at 14kbps in case of after time-out. However, in case of before timeout, it reaches 3s at 14kbps. On the other contrary, Figure 3.8 shows the loss fraction in connection set-up. We observe the average loss in connection set-up increases with the gradual increase in attack frequency. However, Figure 3.6 clearly shows that the loss fraction in connection saturation starts at 12kbps, in case of after time-out.

3.6 Detection of Diverse Attacks Analysis

The section presents the simulation results of detection and behavioral analysis of four diverse attacks. The objective of this section is to closely analyze the detection accuracy analysis and behavior of the extant state-of-the-art classification based network anomaly detection systems. The analysis is carried out to show that some of the attack's detection accuracy of different network anomaly detection systems is very low and largely lacks its applicability, particularly in the control plane of the SDNs. Consequently, the overall detection accuracy certainly necessitates to be improved to timely preserve the correct functioning of the SDN controller.

We employed the benchmark NSL-KDD data-set (Tavallaee, Bagheri, Lu, & Ghorbani, 2012) and MATLAB R2013a for the implementation of the algorithms. We use four diverse class of attacks (DoS, Probe, U2R, and R2l). Denial of service (DoS) attack rep-

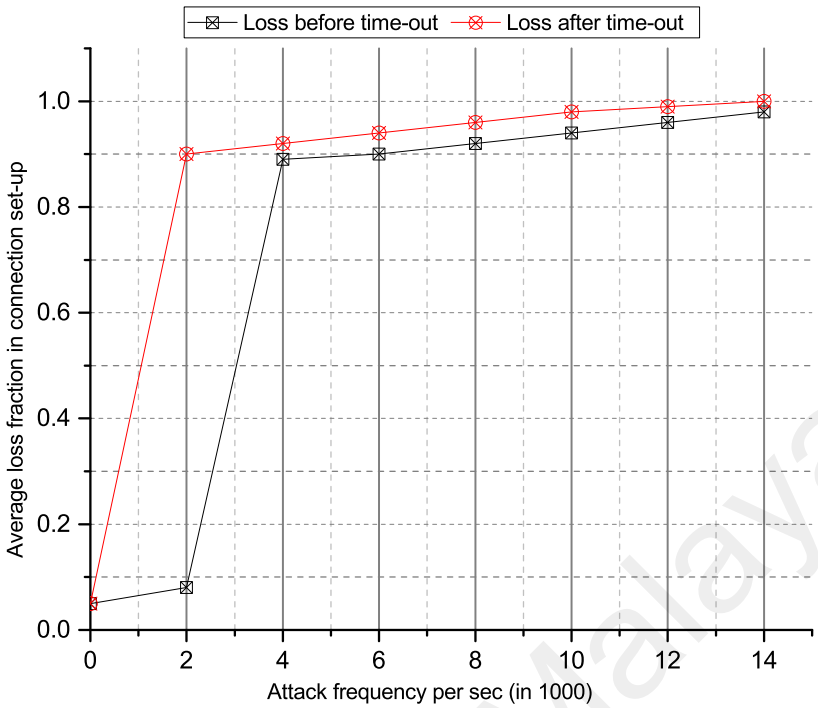

Figure 3.6: Average Loss When the Attack is Launched from a Remote Host on Different Network

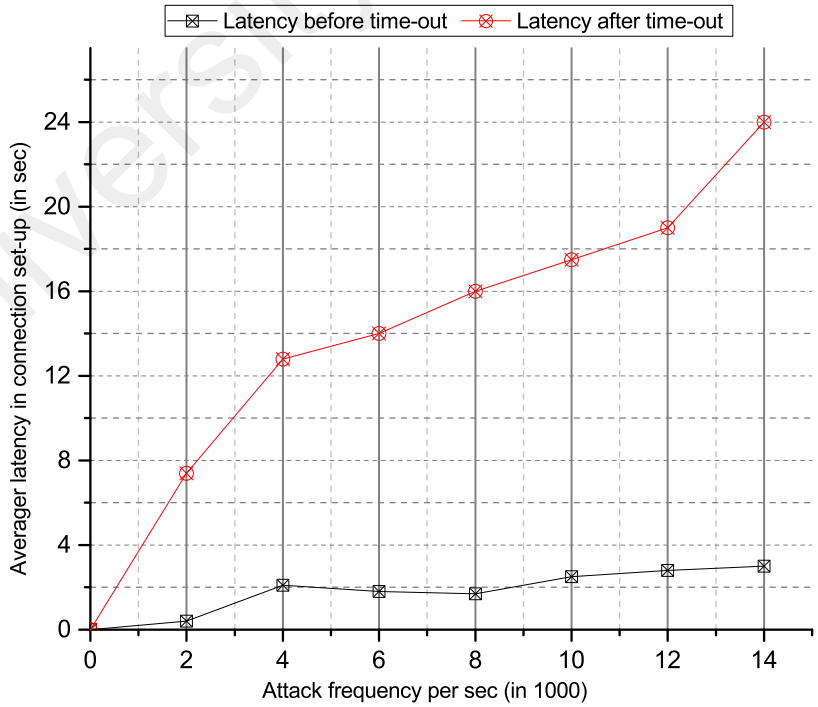

Figure 3.7: Average Latency When the Attack is Launched from a Host on the Same Network

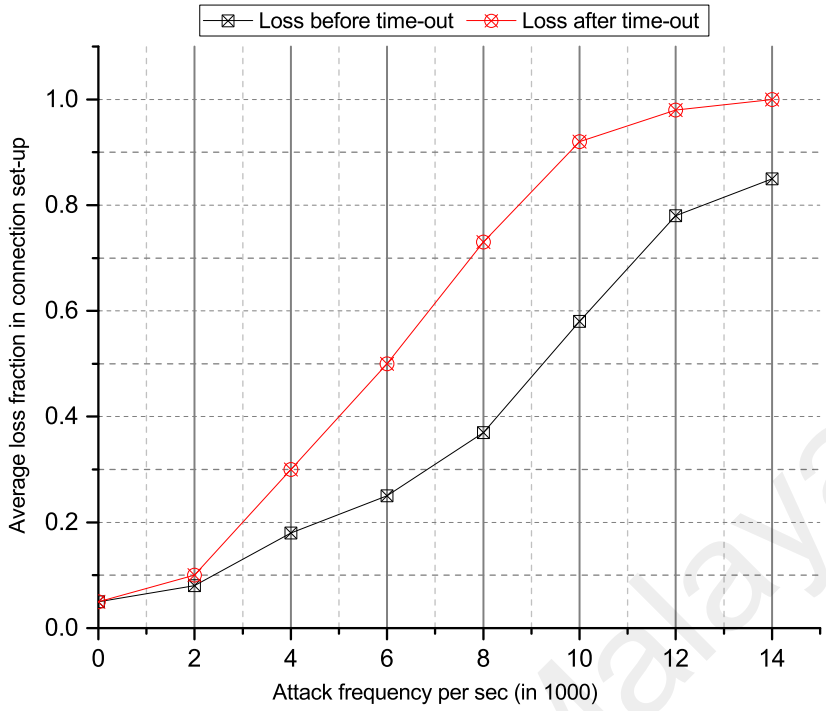

Figure 3.8: Average Loss When the Attack is Launched from a Host on the Same Network

resents an attempt to make the target networks and computing resources unavailable for further legitimate requests, whereas; probe attacks indicate scanning and surveillance of the victim network resources merely to gather sensitive information or to find diverse vulnerabilities of the attack target. However, U2R denotes an attempt to gain unauthorized access to the root privileges of the victim machine and R2L attack essentially attempts of gaining unauthorized local access particularly form a remote machine on the network. This is also known as remote log-in attack. Moreover, the detailed description of the NSL-KDD is given in Chapter 5. Furthermore, the state-of-the-art implemented algorithms are also elaborated in Chapter 5.

Figure 3.9 presents the detection accuracy of state-of-the-art classifier ensembles that is Bagging and Random Subspace (Oza & Tumer, 2008). Both of the algorithms outperforms in detecting accurately the Denial of service (DoS) and U2R attack, which is above 90%. However, the R2L detection accuracy of both the algorithms is below

90%, which is considered low and is not feasible in critical situations. Particularly, the R2L detection accuracy rate of the Random Subspace below 50% is extremely low and unacceptable. Moreover, the probe attack detection accuracy is also observed below 90%. Since the control plane represents a single point of failure and the centralized core of network intelligence, low detection accuracy rates of any sophisticated attack can easily throw the entire network into chaos. The detailed comparison and analysis of each attack for state-of-the-art classifier ensembles are shown in Figure 3.11, Figure 3.13, Figure 3.15, and Figure 3.17.

Figure 3.10 presents the detection accuracy of state-of-the-art base/single classifiers (i.e, Decision-stump, Naive-Bayes, Multilayer Perceptron (MLP), and Linear discriminant analysis (LDA)) (Chandola, Banerjee, & Kumar, 2009). Naive-Bayes, MLP, and LDA classification based anomaly detection algorithms outperforms in detecting accurately the Denial of service (DoS) . However, the DoS detection accuracy rate of the Decision-stump algorithm is observed below 85%, which is considered very low detection accuracy rate. All the for algorithm performs well in detecting U2R attack, the Naive-Bayes outperforms; whereas, the LDA performs relatively low. However, the R2L detection accuracy of both the MLP and LDA algorithms is below 80%, which is considered extremely low and is not feasible in critical situations. Particularly, the R2L detection accuracy rate of the LDA below 50% is unacceptable. Although, the other two algorithms performs well in this category. Moreover, the probe attack detection accuracy rate for LDA is also observed below 70%, which is also not acceptable in centralized architectures. Although, the other three algorithms also performs well in this category. The detailed comparison and analysis of each attack for state-of-the-art base/single classifiers are shown in Figure 3.12, Figure 3.14, Figure 3.16, and Figure 3.18.

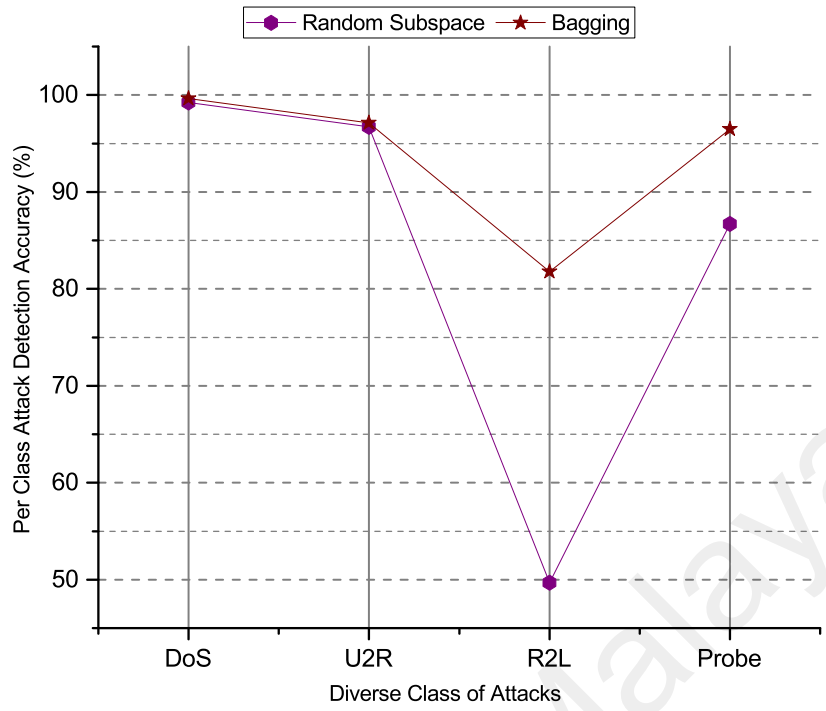

Figure 3.9: State-of-the-Art Classifier Ensembles Detection Accuracy Analysis of Diverse Attacks

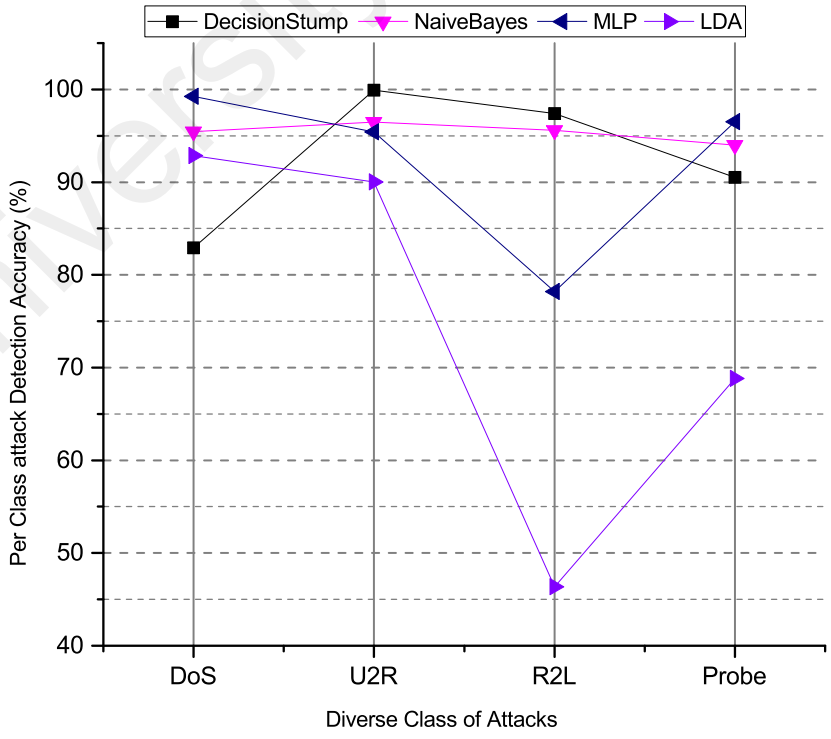

Figure 3.10: State-of-the-Art Base classifiers Detection Accuracy Analysis of Diverse Attacks

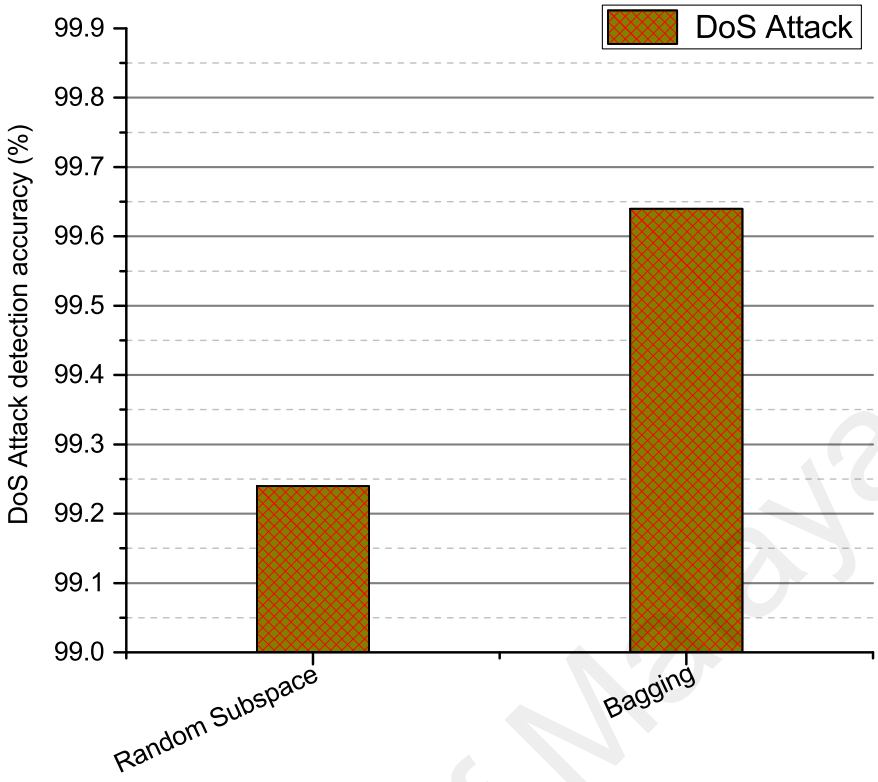

Figure 3.11: Classifier Ensembles Detection Accuracy Aalysis of Denial of Service (DoS) Attack

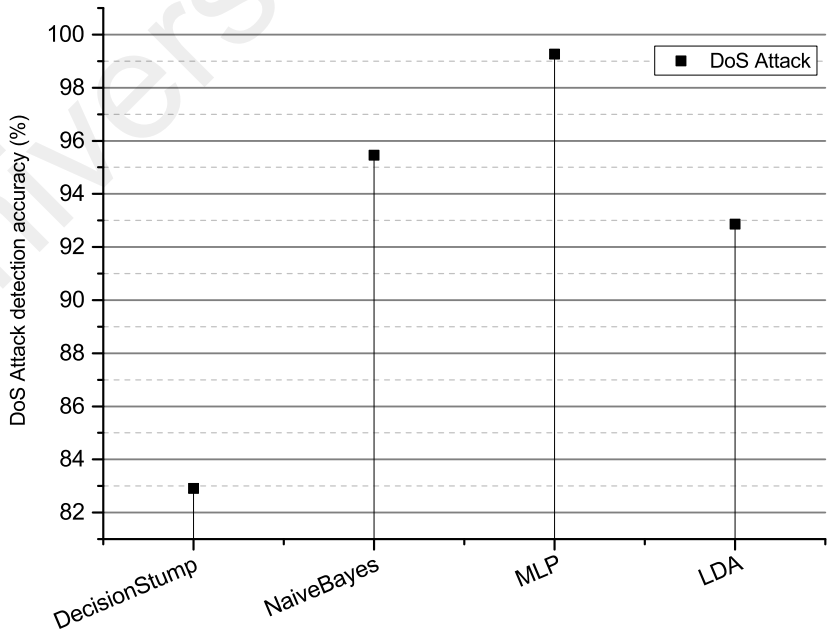

Figure 3.12: Base Classifiers Detection Accuracy Analysis of Denial of Service (DoS) Attack

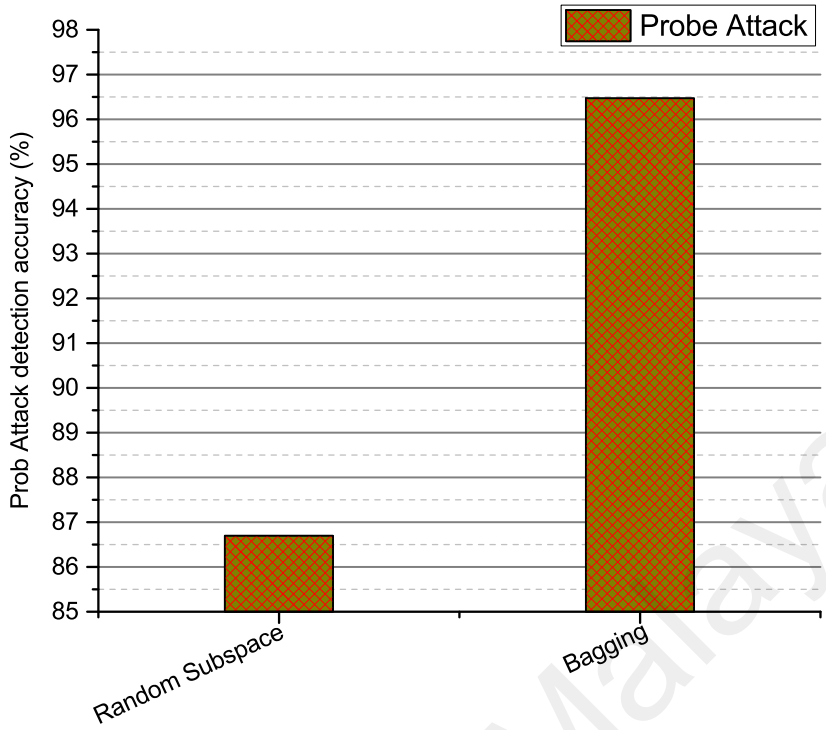

Figure 3.13: Classifier Ensembles Detection Accuracy Analysis of Probe Attack

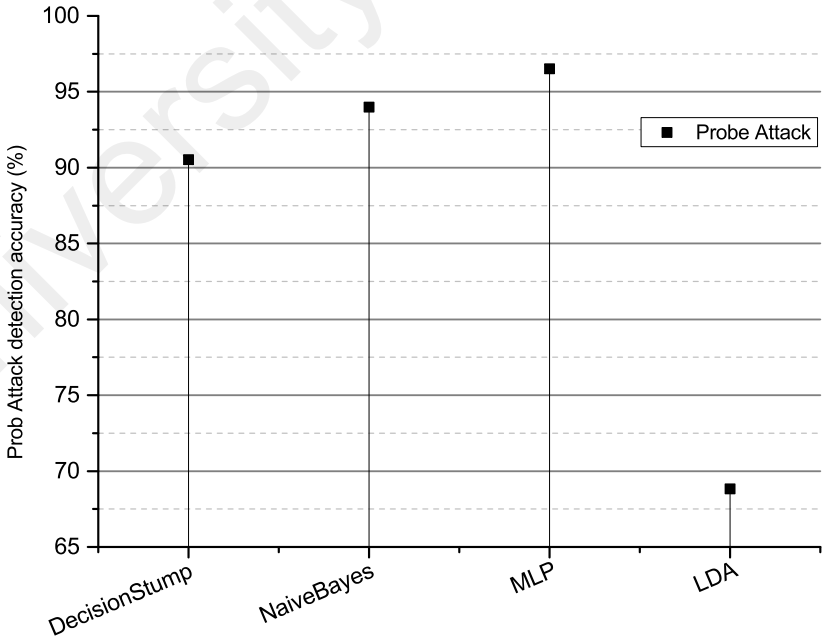

Figure 3.14: Base Classifiers Detection Accuracy Analysis of Probe Attack

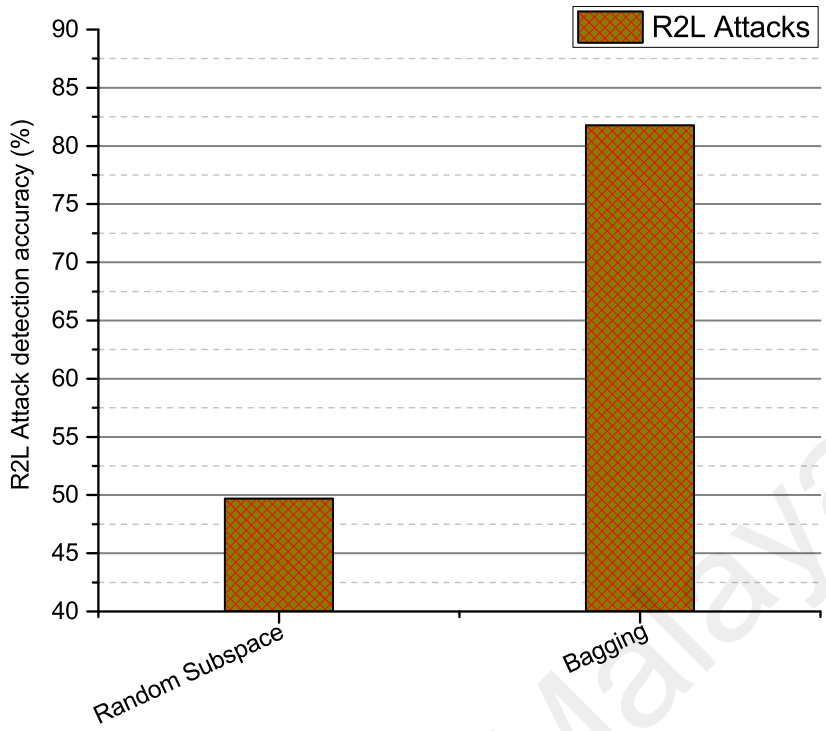

Figure 3.15: Classifier Ensembles Detection Accuracy Analysis of R2L Attack

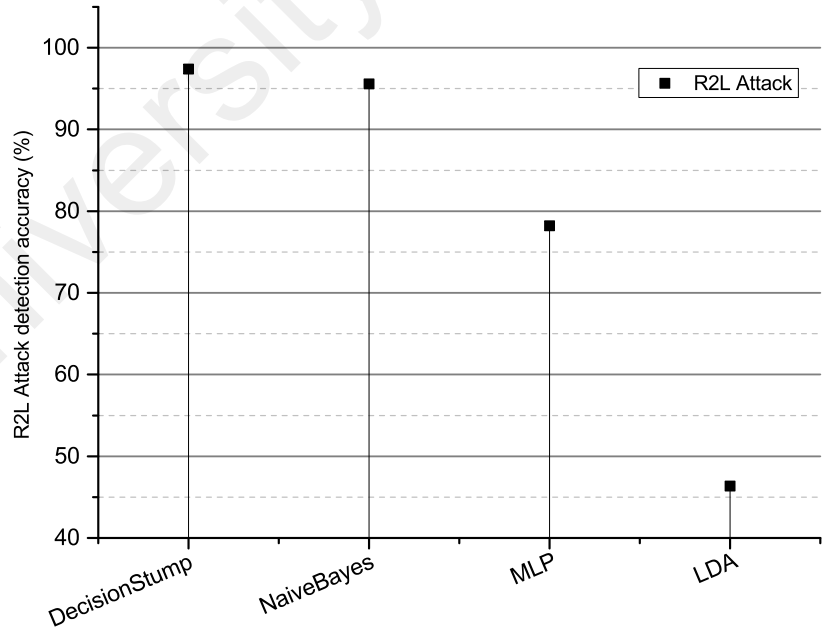

Figure 3.16: Base Classifiers Detection Accuracy Analysis of Probe Attack

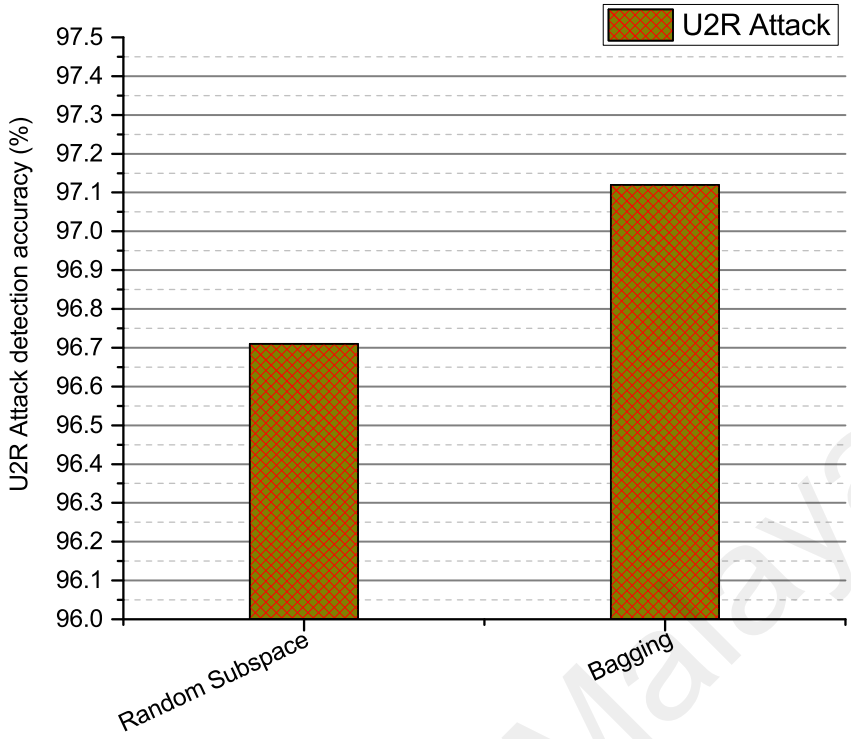

Figure 3.17: Classifier Ensembles Detection Accuracy Analysis of U2R Attack

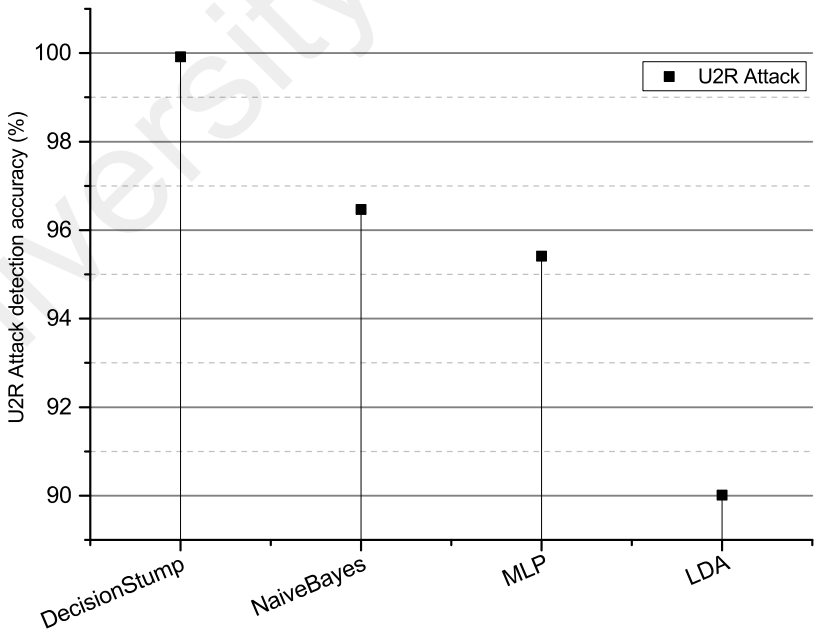

Figure 3.18: Base Classifiers Detection Accuracy Analysis of U2R Attack

3.7 Conclusion

The defense space against the threat of various sophisticated attacks in SDNs that primarily target the control plane to wrest either full or partial control of the entire network in Open-Flow based SDNs is formally analyzed. Moreover, a complete modeling, analysis and verification of the proposed system in the real-setting of SDNs is provided. Furthermore, the proposed system is verified by providing a formal proof on an abstract mathematical model of the system that exhaustively checks and proves the intended behavior of the proposed system.

Executing our proposed SDN model in Z3 solver along with asserted security properties reveals the results, which proves the correctness of the proposed model. Moreover, the execution time taken by Z3 solver on each security property of the proposed models illustrate that the verification ends up in the finite time. A brief impact analysis of the attack on SDN controller is also demonstrated. Finally, we closely analyze the detection accuracy behavior and analysis of the extant state-of-the-art classification based network anomaly detection systems. The analysis is carried out to show that some of the attack's detection accuracy of different classification based network anomaly detection systems is very low, which is certainly not applicable in centralized networks; particularly in the control plane of the SDNs. Consequently, the overall detection accuracy certainly necessitates to be improved to timely preserve the correct functioning of the SDN controller.

CHAPTER 4: A RANDOM ORACLE BASED INTRUSION DETECTION METHOD

The chapter elaborates our proposed dynamic and robust intrusion-detection method to accurately identify large-scale malicious events that predominantly target the control plane to degrade the overall performance of the Software Defined Networks (SDNs) or bring the entire network down in the worst case. We propose a diverse fusion-selection approach that stands on Oracle to be applied to the classifier ensemble design, where the Oracle is a random linear function. We argue that the proposed method adds extra-diversity while promoting a higher level of intrusion-detection accuracy. Moreover, the approach is highly dynamic, flexible and is capable to effectively detect a wide variety of sophisticated security attacks. The method works as apparently the proposed model utilizes the tactic of the well-known divide-and-conquer strategy together-with the use of multiple random oracles.

The key objectives of the chapter are as follows.

- The chapter clearly elaborates our proposed diverse fusion-selection method that stands on Random Oracle. Moreover, the chapter presents the extant main approaches used to design classifier ensembles.
- The chapter provides an understanding of the Random Linear Oracle (RLO), which we consider the main contribution of our proposed dynamic and robust solution followed by a detailed description of the RLO training and prediction phase algorithms. The chosen classifier ensemble employed for the proposed method. Moreover, the chapter also addresses the reason of the employed chosen classifier ensemble.
- The chapter also addresses the significance of the proposed solution and gives justification that why our proposed method works.

- To demonstrate the work-flow of the proposed solution in the real setting of the Software Defined Networks (SDNs). Moreover, we also reason out that the proposed solution is acceptable to be deployed for real applicable scenarios of diverse distributed environments and in particular, for securing the control plane of the software defined networks.

- Finally, a summary and concluding remarks of the chapter is provided.

The remainder of this chapter is structured as follows: Section 4.2 gives a necessary background knowledge of the existing major proposed approaches to design classifier ensembles. Section 4.3 comprehensively elaborates our diverse fusion-selection method that stands on a RLO. The section also presents the RLO model with a detailed description of the corresponding training and prediction phase algorithms. Moreover, the section briefly discusses the chosen classifiers used for the proposed RLO based classifier ensembles method. In Section 4.4, we briefly give justification that why our proposed method works. Section 4.5 presents the work-flow of the proposed method in the real setting of SDNs. Finally, we provide the concluding remarks in Section 4.5.

4.1 Proposed Approaches to Classifier Ensemble Design

We propose a dynamic and robust diverse fusion-selection method that stands on Oracle, where the Oracle is a random linear function. The proposed Random Oracle/ Random Linear Oracle (RLO) based method encapsulates the chosen classifier ensemble to accurately identify large-scale malicious events that predominantly target the control plane of the Software Defined Networks (SDNs). Moreover, our proposed method comprises of a diverse fusion-selection method to classifier ensembles design that certainly necessitates clearly elaborating the underlying key concepts.

The classifier ensembles simply represent a combination of base-classifiers. Subse-

quently, the idea is merely not to rely on the solo decision of a single classifier; instead, a combination of base-classifiers information is gathered to take the final decision; which is widely recognized as classifier ensembles or multiple classifier system (MCS). Normally, the ensembles approach yields better performances. At the same time, the effectiveness of the ensemble method also depends on the diversity and accuracy of the individual base classifiers. Two complementary approaches (i.e. classifier fusion and classifier selection) and switching between the two, which is also known as fusion-selection have been primarily proposed to design different classifier ensembles strategies. Before going to comprehensively elaborate our proposed method, we need to briefly discuss the concept of classifier selection (CS), classifier fusion (CF), and fusion-selection respectively. For a more clear understanding, Figure 4.1, depicts a thematic taxonomy of the major proposed extant approaches to classifier ensembles design.

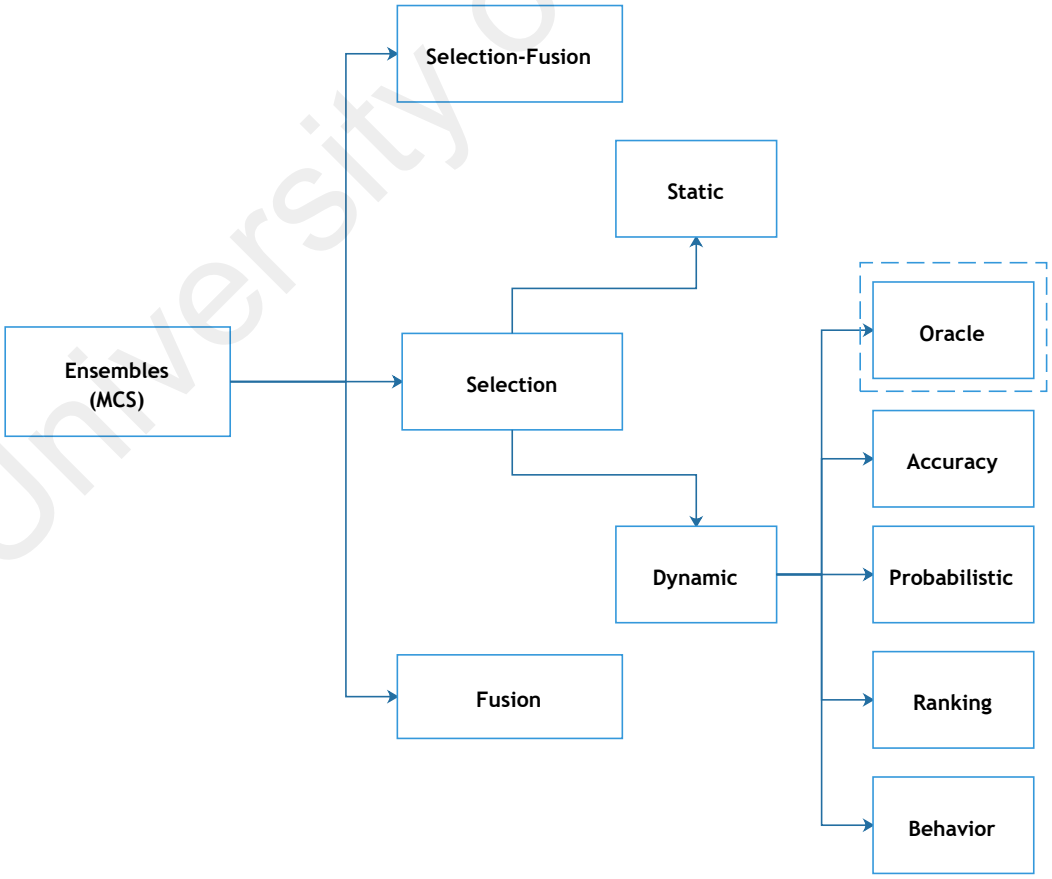

Figure 4.1: A Thematic Taxonomy of the Proposed Approaches to Classifier Ensemble Design

In order to design classifier ensembles or multiple classifier systems (MCS) (Roli, 2015), we have two main complementary approaches (i.e. classifier selection, classifier fusion) and the third approach is sort of a hybrid approach that simply follows switching between selection and fusion according to the given circumstances (Britto, Sabourin, & Oliveira, 2014; Kuncheva, 2002; Kuncheva & Rodriguez, 2007). Moreover, the classifier selection strategy is further divided into two sub-categories (i.e. dynamic selection, and static selection). The dynamic selection (DS) is further divided into five sub-categories, specifically when the classifiers individual performance is the main source of information. The taxonomy clearly depicts the five sub-categories of the dynamic selection (DS); however, we here are only concerned with the oracle based competence evaluation of an individual classifier in a given ensemble designed strategy (Britto et al., 2014). The explanation of the rest of the individual-based dynamic selection (DS) classification and group-based dynamic selection classification is beyond the scope of this thesis.

4.1.1 Classifier Selection (CS)

The classifier selection method, utilizes each base classifier to be responsible for the classification problem. However, the final decision is based on a single most proficient base classifier, whereas; the selection of the competent base classifier is done by an expert called "oracle" for a given input X. Basically, the classifier selection works on the assumption of the presence of a single oracle that nominates the most competent classifier. Moreover, the idea of the CS resurfaced numerous times over the past three decades. Nevertheless, it can be mainly classified into two main categories (i.e. static classifier selection and dynamic classifier selection) (Britto et al., 2014; Burduk & Walkowiak, 2015; Kuncheva & Rodriguez, 2007).

Static classifier selection: In case of static classifier selection, the region of competence for each base-classifier is done during the training phase, prior to actual classifica-

tion (Burduk & Walkowiak, 2015).

Dynamic classifier selection: Compared to static classifier selection, the region of competence in the dynamic classifier selection is done during the classification where the oracle exercises estimating diverse accuracies to announce the winner. Moreover, the dynamic classifier selection method is viewed as the faster version compared to static classifier selection (Britto et al., 2014).

4.1.2 Classifier Fusion (CF)

The classifier fusion (CF) approach, on the other hand, governs the whole classification boundary and the decision is purely based on the combined results of the ensemble classifiers. Moreover, the CF governs the whole feature space and is probable to mis-classify certain objects (Visentini, Snidaro, & Foresti, 2016; Kuncheva & Rodriguez, 2007).

4.1.3 Fusion-selection

The fusion-selection was initially proposed in (Kuncheva, 2002), whereby the statistically significant nominated classifier becomes a solo decision maker over the remaining classifiers. On the contrary, the entire ensemble is summoned, and the classifier decisions are fused; if and only if the nominated classifier is not statistically significant over the remaining classifiers. The oracle and the classifiers are trained together to particularly experience certain regions in the feature space to differentiate as part of the training. Subsequently, the Oracle learns the most trusted classifier for a given X. literally; the classifiers have been assigned weights of competence by the oracle for a given input X, despite choosing the single most proficient classifier. Consequently, the classifier ensemble decision is obtained from the fusion of weighted opinions (Kuncheva, 2002; Britto et al., 2014; Burduk & Walkowiak, 2015).

4.2 Proposed Random Oracle Based Method to Classifier Ensemble Design

We propose a dynamic and robust diverse fusion-selection method that stands on Oracle, where the Oracle is a random linear function. The proposed Random Oracle/ Random Linear Oracle (RLO) method encapsulates a chosen classifier ensemble to accurately identify large-scale malicious events and abnormal network behavior that predominantly target the control plane of the Software Defined Networks (SDNs). The idea is the introduction of a random oracle and replacing the chosen classifier ensemble with a mini-ensemble of two classifiers. The Oracle embeds a hyperplane and randomly split the classification data into two parts. Subsequently, a classifier is assigned to each half; however, during the classification phase, the Oracle for each classifier is applied and the respective sub classifier makes the decision to be fused further at the ensemble level.

Our proposed method is different and diverse from the aforementioned standard model of classifier selection (CS), whereby a single Oracle governs the whole feature space of the given classifier ensemble. Moreover, the use of multiple random oracles makes our proposed model diverse and unlike from the classifier fusion (CF) and the aforesaid dynamic switching model (i.e. fusion-selection). The proposed method is an anomaly based network intrusion/attack detection system, where the primary goal is the automation of intrusion detection that accurately classify large scale malicious events that mainly target the control plane of the software defined networks. Furthermore, automation systems employed for discriminating between anomalous and normal behavior often use machine learning algorithms such as classification or clustering. However, our proposed diverse fusion-selection that stands on random oracle represents a classification based network anomaly detection system, which subsequently belongs to the main category of supervised network anomaly detection systems. Moreover, our proposed network anomaly detection system is likely to identify abnormal network attacks that mainly tar-

get the control plane of the SDNs without prior or having specific knowledge, and that is why the anomaly based detection systems are widely applicable in diverse fields.

The key aspect of our proposed supervised anomaly based network intrusion detection system is computational complexity to be applicable to the real setting of Software Defined Networks (SDNs). No doubt, the classification based anomaly detection systems more often require expensive training times with usual fast testing processing time. However, it is often acceptable in real applicable scenarios to train the corresponding model in an off-line fashion/mode while testing certainly requires being in real time. We follow exactly the same fashion for our proposed model, which will be explained later in the work-flow of the proposed model. On the contrary, we argue that un-supervised learning techniques despite many advantages need no training phase, testing phase is expensive and that can be a limitation to be applicable in the real domain of the SDNs. Moreover, we also argue that signature based intrusion detection systems may work, but they are not effectual for the very dynamic and distributed SDN environments. The reason is very simple, generating a packet signature responsibility that moves from a switch or a middle-box in SDNs must be directed to a remote control program, where not only the processing time is slower than the hardware but also necessitates every packet to be redirected to it. Despite many other disadvantages, the process becomes time intensive and complex to be applicable to the real setting of SDNs distributed environments.

We also argue that our proposed diverse fusion-selection method that stands on a random oracle adds extra-diversity while promoting a higher level of intrusion detection accuracy, which is a real challenge to meet largely when high volume network traffic is involved. Moreover, the approach is highly dynamic, flexible and is capable to effectively and accurately detect a wide variety of large-scale malicious events and sophisticated network security attacks. Furthermore, the method works as apparently the proposed model utilizes the tactic of the well-known divide-and-conquer strategy together-with the

use of multiple random oracles. Additionally, we majorly consider our novel contribution as the introduction of random linear oracle to the classifier ensembles design in the field. The random oracle; however, has the capability to encapsulate any classifier ensembles strategy (i.e. any combination of base classifiers). Nonetheless, the chosen classifier ensemble employed for our proposed method comprises of classification and regression Tree (C&RT/CART) and Multilayer Perceptrons (MLP), which is also widely known as feed-forward artificial neural network (ANN).

Since we consider our major contribution as the introduction of Random Oracle/ Random Linear Oracle (RLO) in our proposed method, a detailed description of the RLO with their corresponding algorithms is given below. We also discuss briefly the employed chosen base-classifier models (i.e., CART, and MLP) for the proposed method.

4.2.1 Random Linear Oracle (RLO)

The Random Linear Oracle model represents a random discriminant function that splits the data into two subsets regardless of structure or class label. The main idea is the replacement of the chosen classifier ensemble by a mini-ensemble of the two classifiers and an Oracle, where the Oracle is a random linear function. The oracle creates a random hyperplane to divide the space into two subspaces drawn in the feature space of the dataset.The chosen classifier ensemble comprises of a pair of base-classifiers (i.e., CART, and MLP) that learns on diverse subspaces to be decided by random oracle. Subsequently, each half-space data is utilized to train the classifier within our chosen ensemble strategy. The oracle is applied to each classifier during the classification, and one of the sub-classifier from each ensemble pair decides to be fused further at the level of the chosen ensemble. For instance, the classification of a new object X, the respective classifier's oracle from the chosen ensemble decides which sub-classifier to use. Finally, the ensemble combination rule is used to combine the labels issued by the sub-classifiers. We employed

the majority voting scheme for the ensemble combination rule in our proposed method.

The RLO whole process is based on training and prediction phase. During the training phase, the classifier ensemble and the oracle are trained together to push the classifiers into specializing in diverse regions of the feature space of the training data-set. Subsequently, it makes the oracle capable of learning which classifier to trust most for a given input X. Moreover, the oracle here assign weights of competence to the corresponding classifiers rather than just selecting the most proficient classifier. Consequently, the classifier ensemble final decision is derived as a fusion of weighted opinions (i.e., majority votes in our case).

Two RLO model are built and trained during the training phase, whereas; in prediction phase merely one of the two models is used. Therefore, the computational complexity of the RLO is almost similar to the base method complexity, if and only if the training and prediction complexity of the oracle is low. The training time as usual depends linearly on the number of training examples. The complete and clear descriptions of both training and prediction along with their corresponding algorithms are as follows.

4.2.1.1 Building The Training Phase With Random Linear Oracle (RLO)

The random linear oracle (RLO) here comprises of a single classifier ensemble (pair of base classifiers (i.e. CART, MLP)) and an oracle. It is worth mentioning that during the training phase, the classifier ensemble (pair of chosen base-classifiers) are built and trained with a disjoint partition of the training data. Training phase of the random linear oracle model mainly consists of the following steps respectively.

1. Select randomly the oracle.

2. Split the training data set into two subsets using random linear oracle, which means the random oracle creates a hyperplane and divides the data space into two sub-

spaces without considering the cluster structures and class labels of the data. The hyperplane is created through the points having same distance from the two objects by the random discriminant function, where the distance is calculated through the Euclidean distance formula. The length of the line segment connecting two points normally represents the Euclidean distance between the corresponding two points. For instance, the distance of two points having coordinates (x, y) and (x1, y1) is given by the following formula.

$$Dist((x,y),(x1,y1)) = \sqrt{(x-x1)^2+(y-y1)^2}$$

Moreover, calculating the distance between the two points X= (x1, x2,....., xn) and Y= (y1, y2,....., yn) in Euclidean n-space can be done through the following formula.

$$Dist(X,Y) = \sqrt{(x1-y1)^2+(x2-y2)^2+.....+(xn-yn)^2}$$

$$Dist(X,Y) = \sqrt{\sum_{a=1}^{n}(xi-yi)^2)}$$

3. Building and training the the pair of base classifier models using random linear oracle (RLO) for each subset of the training data is done in the third step. Consequently, the RLO is added to the current ensemble. It is to be noted that during the training phase, the classifier ensemble (pair of base chosen classifiers) are built and trained with a disjoint partition of the training data, whereas the training time as usual depends linearly on the number of training examples. Moreover, building the oracle needs two different training objects to be selected at random and finally the pair of models and the oracle itself forms the trained random linear oracle (RLO) model.

Building the training phase with Random Linear Oracle (RLO) algorithm gives a more clear demonstration of the given steps.

Algorithm 1 Algorithm-1: Building Training Phase With RLO (Random Linear Oracle)
Input: Training Data-set T, Base classifier model M
Output: Random linear oracle model RLO

```
1:  function RANDOMLINEARORACLE
2:      T1 = 0                          ▷ T1 represents the training data-set of the 1st sub-model
3:      T2 = 0                          ▷ T2 represents the training data-set of the 2nd sub-model
4:      RLO.instance[1] ← [x|(x,y) is a random instance from T]
5:      RLO.instance[2] ← [x|(x,y) is a random instance from T]
6:      for each instance(x,y) ∈ T do   ▷ Split the training data-set into two subsets
7:          if distance(RLO.instance[1],x) > distance(RLO.instance[2],x) then
8:              T2 ← T2(x,y)                ▷ add the instance to the second subset
9:          else
10:             T1 ← T1(x,y)                ▷ add the instance to the first subset
11:         end if
12:     end for
13:     RLO.model[1] ← M(T1)            ▷ Train the first Sub-classifier model
14:     RLO.model[1] ← M(T1)            ▷ Train the second Sub-classifier model
15: end function
```

4.2.1.2 Building The Prediction Phase With RLO

The RLO prediction phase is very simple. Here in the prediction phase, one of the inputs is the already trained RLO and a given test instance X. Subsequently; the oracle decides and selects one of the two models. The prediction phase mainly consists of the following two steps respectively.

1. Select one of the two sub-classifier of each member of the chosen ensemble using random linear oracle (RLO).

2. Return the expert prediction (prediction with confidence depending on which side of the hyperplane X is). Consequently, X is assigned a class having the majority of votes. While applying the ensemble combination rules (i.e., combine all the decisions of the classifiers, we employed majority voting in this particular case to announce the final decision).

Algorithm 2 Algorithm-2: Building Prediction Phase With Trained RLO
Input: Trained Random Oracle RLO; Instance X
Output: Predicted value

```
1: function PREDICITION
2:     if distance(RLO.instance[1],x) > distance(RLO.instance[2],x) then
3:         returnRLO.model[2].predict(x)          ▷ Prediction with 2nd sub-model
4:     else
5:         ReturnRLO.model[1].predict(x)          ▷ Prediction with 1st sub-model
6:     end if
7: end function
```

4.2.2 The Chosen Classifier Ensemble Used for the Proposed Method

The chosen classifier ensemble employed for our proposed method essentially comprises of classification and Regression Tree (C&RT/CART) and Multilayer Perceptrons (MLP). Both the classifiers are very widely known algorithms used for classification with rich available literature. Therefore, a detailed description of both the classifiers is beyond the scope of this thesis. However, the following section briefly explains the two classifier. Moreover, the main reason of the employed chosen base-classifiers (current ensemble) is their individual performances, versatile nature, better extracting complex attack data patterns, and precise decision making during the classification. Furthermore, we already discussed that the effectiveness of the ensemble method also depends on the diversity and accuracy of the individual base classifiers.

4.2.2.1 Classification and Regression Tree (C&RT/CART)

The classic Classification and Regression Tree (C&RT/CART) algorithm was initially disseminated by (Breimanetal, 1984). C&RT builds decision trees to capture complex input data patterns by utilizing the concept of information theory. C&RT is a decision tree learning mechanism that generates trees (regression and classification) based on whether the input variable is numeric or categorical, respectively. It is a recursive partitioning approach that creates non-overlapping regions where continuous features are represented by rectangles and categorical features as a subset of values to identify the most likely

dependent variable within the corresponding particular region. The tree-building algorithms (classification and regression) mainly employ a bunch of if-then logical (split) conditions (tree nodes) to achieve high detection accuracy. C&RT is very well-known for its efficiency in terms of handling irrelevant input data, missing values, and input data of multiple types. The main advantage of C&RT approach is that trees interpretation of results is extremely simple. Moreover, tree-building algorithms not only boost new observation's classification but it is too easier to evaluate few if-then logical (split) conditions (tree nodes). For instance, simple if-then statements are easier to present to management for decision making rather than some elaborate equations.

4.2.2.2 Multilayer Perceptrons (MLP)

A Multilayer Perceptrons (MLP) is a feed-forward artificial neural network (ANN), where the information moves only in one direction, and that mean no formation of loops and cycles in the network. MLP is the simplest form of neural networks, which comprises of an input layer, one or more hidden layers, and an output layer. MLP layers having no direct connection to the outside world are termed as hidden layers. Computational units (Neuron/ Perceptrons) of multiple layers are connected in a directed graph, with each layer fully connected to the subsequent layer in a feed-ward way. MLP has the ability to remarkably process imprecise or complicated data precisely, extracting complex patterns, and detecting multifaceted trends that is too difficult to be noticed by other computer techniques. It's exceptional ability of adaptive learning from a set of given training data and mapping of any complexity, though the learning needs repetition with training samples. MLP outperforms and yields significant generalizations in situations where mapping and discovery derivation of relationship explicitly is almost impossible. MLP have been applied to solve a verity of diverse and complicated problems in different fields of pattern recognition, machine learning and information security.

4.2.3 Why Our Proposed RLO Based Approach Works

There are two main intuitive reasons that explain well why our proposed random linear oracle (RLO) based method may work.

1. Divide-and-conquer tactic: The first and quite obvious reason of the proposed method of working well is the use of divide-and-conquer tactic to solve the problem of an anomaly based intrusion detection. The RLO distribution of feature space into two parts makes the classification easier for the chosen classifier ensemble. Subsequently, the accuracy of the individual ensemble members is likely to be higher than or at least no worse than that of an ensemble classifier which operates on the whole feature space. The RLO exactly follows the spirit of the divide-and-conquer approach, whereby decomposing a large problem into sub-problems that are (supposedly) to be solved more conveniently and accurately.

2. The second obvious reason is that the classification of a data point X is done through one of the two sub-classifier of each classifier ensemble member. The diversity of the sub-classifier is expected to be large as the sub-classifiers have been trained with diverse data subsets already determined by the random oracle (RO). Consequently, with RLO classifiers are shaped.

4.3 Work-flow of the Proposed Solution

The section presents the work-flow of the proposed intrusion detection model for identifying large scale malicious events and abnormal and abnormal attacks in the control plane of the SDN environment. We assume here that the proposed detection module is already implemented in Floodlight, a popular java based SDN controller. Figure 4.3 presents the work-flow of the proposed solution in the real setting of the software defined networks. The rest of the explanation of the work-flow are as follows. To efficiently address the

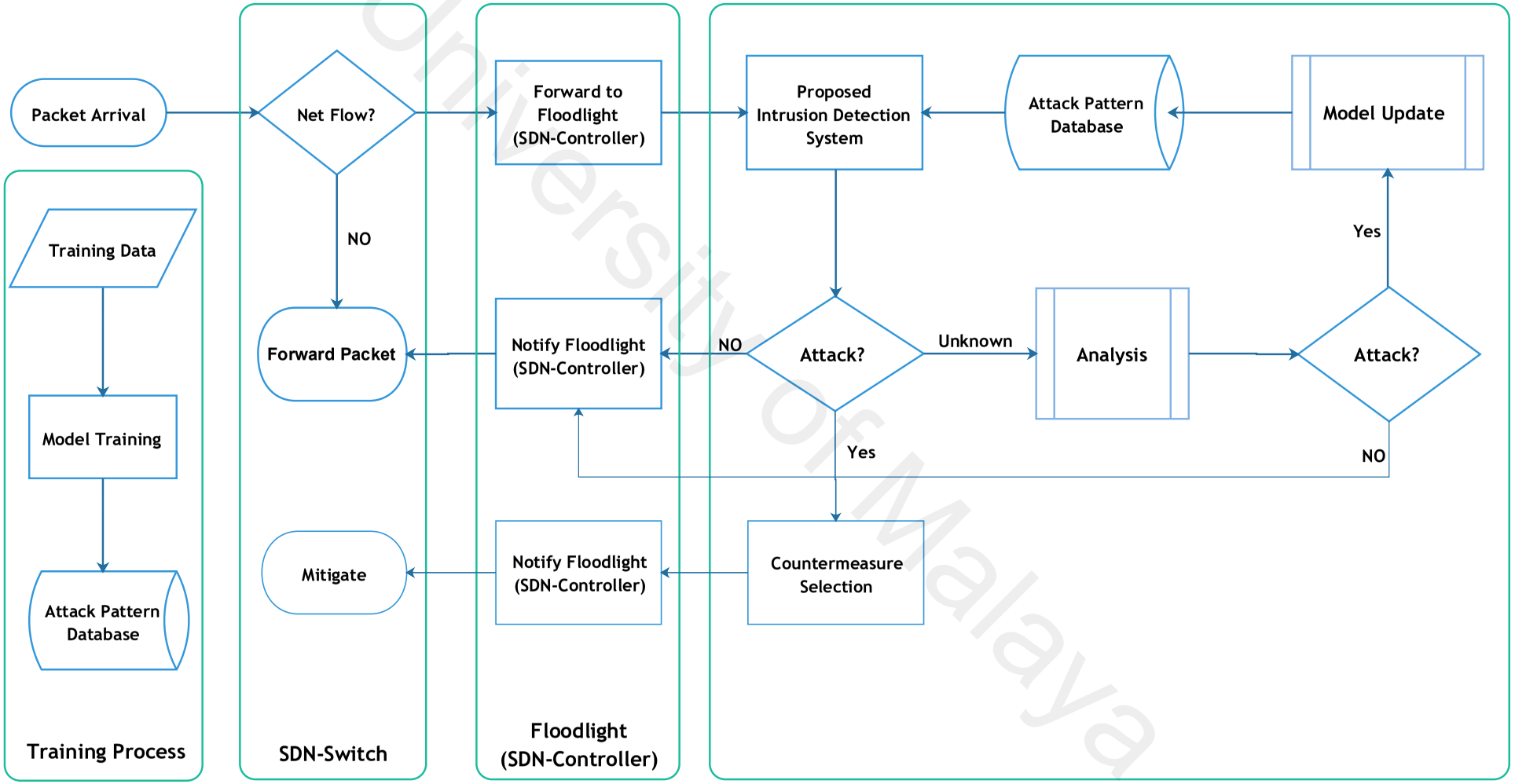

Figure 4.2: Work-flow of the Proposed Method in the Real Setting of SDNs

problem of expensive training time required by the classification based anomaly detection systems. The proposed RLO based detection model entire training process is carried out in an offline mode/fashion while testing being in real time, which is often acceptable in real applicable scenarios. In an online process, when a new packet arrives at a switch that belongs to an existing flow-rule is forwarded to the controller while updating the flow statistics. The packet needs to be processed by the proposed detection module to identify the abnormal behavior. The results may indicate whether the packet is malicious representing an attack or a normal packet. In case of a normal packet, the controller is notified and the packet is forwarded to the intended destination. In case of intrusion/attack, the attack is either known attack or unknown attack. The known attack simply generates the alert and notifies the controller for possible countermeasure selection. Drop the packet is the default action, if there is no pre-set policy for handling the attack alerts. However, a detailed discussion of the countermeasure selection and mitigation is out of the scope of this thesis. If the attack is un-known (zero day attack), it is further analyzed to train, model and update the system with new attack patterns for possible upcoming detections through a model updating process.

4.4 Conclusion

We propose a diverse fusion-selection method that stands on Oracle, where the Oracle is a random linear function. We also provide the underlying key concepts of classifier ensemble design strategies to plainly show that our proposed method is different and diverse from the standard model of classifier selection, classifier fusion, and fusion-selection. Moreover, we elaborate comprehensively the Random Linear Oracle (RLO) method, which we consider the main contribution of our proposed dynamic and robust solution followed by a detailed description of the RLO training and prediction phase algorithms. We briefly discuss the chosen classifier ensemble employed for the proposed

method along-with their reason of selection.

Moreover, we also present the significance of the proposed method and provide justifications that why the method works. The divide-and-conquer tactic and the use of multiple random oracles empowers the proposed method. Finally, we demonstrate the work-flow of the proposed solution in the real setting of the Software Defined Networks (SDNs) and reason out that the proposed method is acceptable to be deployed for securing the control plane of the software defined networks.

CHAPTER 5: EVALUATION

The chapter elaborates the evaluation of the proposed diverse fusion-selection intrusion detection method in terms of the overall performance and effectiveness. The chapter also explains the tools, and the bench-mark data-set used for the complete evaluation of the proposed random linear oracle (RLO) model. The setup environments, the pre-requisites, and the programming tools and language used for the final implementation. Moreover, the standard parameter used for assessing the performance of the proposed model with a model validation technique is also provided.

The remaining chapter is structured as follows. Section 5.2 explains the setup environment, the pre-requisites, and the programing tools used for the implementation and deployment of the proposed model. Moreover, the section also elaborates the standard parameter, the benchmark data-set employed, and comparison with the current state-of-the-art algorithms. Section 5.3 presents the classification performance data of the state-of-the-art ensembles and base classifiers. Section 5.4 concludes the evaluation chapter.

5.1 Evaluation of the Proposed Method

This section elaborates the complete methodology adopted for the evaluation of the proposed method. It presents the experimental setup environments (i.e. emulation,simulation), the standard parameters used, and the corresponding bench-mark data-sets employed for the evaluation.

5.1.1 Experimental Setup

For the emulation evaluation setting of the proposed method, we choose the Floodlight (Floodlight, 2014), as the SDN controller, and the reason behind is very simple that the module loading system of the Floodlight is easily extended and enhanced. Moreover, we employ Mininet (Lantz et al., 2010), which realistically creates a virtual network on a

computer in-order to emulate the setting of the SDN. The entire experimental set-up is implemented using Java on a Lab system Intel ® core i5- 2500M CPU at 3.30 GHz, and 8 GB RAM that runs a popular Java based SDN controller, Floodlight with our proposed attack detection system on it as an extended module. Furthermore, for emulating the attacks, we employed tools such as Scapy and Cbench (Cbench, 2015) to emulate diverse attacks. The overall emulation environment is depicted in Figure 5.1.

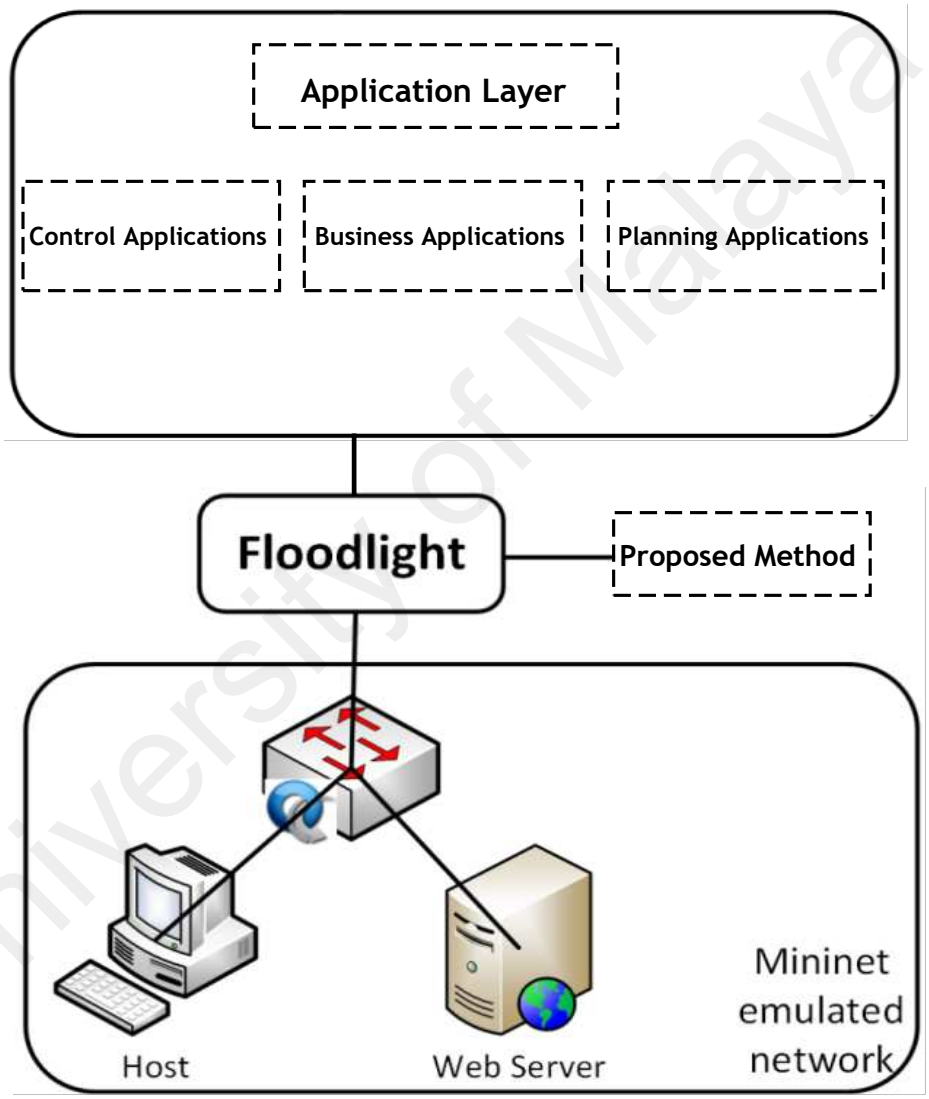

Figure 5.1: Emulation Set-up Environment

On the contrary, the algorithms were implemented in MATLAB R2013a for a comprehensive simulation to exhibit the effectiveness of the proposed method in detecting diverse class of attacks. To validate our proposed approach, we applied a K-fold cross validation using a benchmark publicly available data-set broadly known as NSL-KDD,

where K is not a fixed parameter. However, to show the resulting significant performance of the proposed approach to be optimistically unbiased, we employed a ten-fold (i.e., k=10) cross-validation. K-fold validation is the most widely used model validation technique for diverse intrusion detection systems. The verification of the proposed approach is made with state-of-the-art intrusion detection algorithms. The standard parameters and the corresponding bench-mark data-set are detailed in upcoming sections.

5.1.2 Performance Evaluation Parameters

We evaluate the performance of our proposed Intrusion Detection System (IDS) with the following standard parameters (i.e., accuracy, detection-rate/precision, and false alarms/False-Positive-Rate (FPR), True-Positive-Rate (TPR)/Recall,True-Negative-Rate (TNR)/Specificity and F-measure, which are employed globally by the majority of researchers in the literature to evaluate the performance of any IDS. The parameters selection usually varies for diverse circumstances; however, we consider all the parameters for rigorous evaluation of our proposed IDS. The accuracy is one of the most important and widely used evaluation parameter. The evaluation parameters are given below.

$$Accuracy = \frac{TN+TP}{TN+TP+FN+FP}$$

$$Precision(Detection-Rate(DR)) = \frac{TP}{TP+FP}$$

$$False-Alarm(False-Positive-Rate(FPR)) = \frac{FP}{FP+TN}$$

$$Recall(True-Positive-Rate(TPR)) = \frac{TP}{TP+FN}$$

$$Specificity(True-Negative-Rate(TNR)) = \frac{TN}{TN+FP}$$

$$False-Negative-Rate(FPR) = \frac{FN}{TP+FN}$$

$$F-Measure = 2*(\frac{Precision*Recall}{Precision+Recall})$$

Where *TP,TN,FP, and FN* represents the followings.

- True Positives (TP): TP classifies malicious activities correctly as malicious.
- True Negatives (TN): TN classifies non-malicious/benign activities as benign.
- False Positives (FP): FP classifies non-malicious/benign activities as malicious.
- False Negatives (FN): FN classifies malicious activities as benign/non-malicious.

5.1.3 Data-set Employed for the Performance Evaluation

The evaluation of the proposed approach is based on a publicly available data-set that is primarily known as NSL-KDD (NSL-KDD, 2014) (Ji, Jeong, Choi, & Jeong, 2016).

5.1.3.1 Description of Data

The section represents the necessary explanation of the data-set. The NSL-KDD data-set comprises testing set (22,544 records) and training set (125,973 records) that holds 41 attributes (six binary, three nominal, and thirty-two numeric attributes). The data-set includes both normal and attack data. However, the attacks in particular are subsequently grouped into four key categories as shown in Table 5.1. Our study combines the testing and training data (148,517 records) as input data to employ ten-fold cross-validation.

Denial of service (DoS) represents an attempt to make the target networks and computing resources unavailable for further legitimate requests. Probe attacks indicate scanning and surveillance of the victim network resources merely to gather sensitive information or to find diverse vulnerabilities of the attack target. U2R denotes an attempt to gain unauthorized access to the root privileges of the victim machine. R2L attack essentially attempts of gaining unauthorized local access particularly form a remote machine on the

Table 5.1: Four Main Intrusion Classes and their Corresponding Sub-types

Intrusion Class	Intrusion Types
R2L	Sendmail, snmpguess, xlock, xsnoop, named, ftp_write, imap, guess_passwd, multihop, phf, spy, ware-zclient, warezmaster, snmpgetattack, httptunnel
DoS	Processtable, smurf, teardrop, back, land, neptune, pod, ,mailbomb, worm, apache2
Probe	Portsweep, satan, saint, ipsweep, nmap, mscan
U2R	sqlattack, buffer_overflow, loadmodule, perl, rootkit, ps, xterm

network. The Four main intrusion classes with their corresponding types are shown in Table 5.1.

5.1.3.2 Detection of abnormal behavior

The data needs to be preprocessed to get the results. The three main types (i.e., nominal, protocol, and numeric) comprises the total number of 41 features having their sub categories. For instance, the three nominal variables of the NSL-KDD data-set as shown in Table 5.2 includes service, flag, and protocol type. Each variable contains numerous distinct attributes values such as flag contains 11 attributes (i.e., S2, S3, S1, REJ, and among others), protocol type holds (i.e., ICMP, TCP, and UDP) and service comprises 70 attributes (i.e. WHOIS, POP3, HTTP, SSH and among others).

Apparently, extracting information to detect the abnormal behavior from this huge set of distinctive attributes is difficult. In order to streamline the issue, a binary coding scheme is employed to the three nominal variables where "zero" represents the non-occurrence of a category of interest and "one" denotes the occurrence of that particular category. For instance, the protocol type attribute value for UDP either contains 1 or 0.

5.1.4 Comparison with the Current State-of-the-art

The verification and performance evaluation of our proposed method is carried-out by comparing the current state-of-the-art classification based network anomaly detection

Table 5.2: Data-types and their Corresponding Detailed Features

Type	Features	Accumulative Features
Nominal	Flag, Protocol_type, Service	3
Binary	is_host_login, root_shell, su_attempted, root_shell, Land, logged_in, is_guest_login	6
Numeric	src_bytes, urgent, duration, dst_bytes, wrong_fragment, hot, num_compromised, num_failed_logins, num_root, num_file_creations, num_access_files, num_shells, num_outbound_cmds, count, srv_count, srv_serror_rate, rerror_rate, serror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate,dst_host_srv_diff_host_rate, dst_host_serror_rate,dst_host_srv_serror_rate, dst_host_srv_serror_rate , dst_host_rerror_rate	32

mechanisms. However, there are two main categories of comparison made with the extant state-of-the-art: (i) classifier ensembles and (ii) base/single classifier based anomaly detection systems, which is detailed in the subsections.

5.1.4.1 State-of-the-art Classifier Ensembles

The section briefly describes the two customized state-of-the-art ensemble methods that are used for the comparison to evaluate the performance of the proposed method in this study. These approaches have two features in common: first it employs the re-sampling techniques in order to create new training sets to add extra diversity, and second these approaches can encapsulate any base-classifier. Table 5.3 clearly shows the state-of-the-art ensemble methods, the problem domain, and the data-set employed.

Bagging: The method manipulates the original training data-set to construct numerous versions of training sets known as bootstrap replicates of the original data set. The method aggregates the classifications decision drawn by the base-classifiers through voting. The predicted class is supposed to have accumulated the majority of votes. Consequently, this approach is known as “bootstrap aggregating”; however, it is better recog-

nized by its acronym "bagging" (Breiman, 1996).

Random Subspace: The random subspace selects randomly the subset of the original features to train the base classifier ensembles. However, the method aggregates the outputs of the ensemble members by employing the majority vote method to get the final output. Unlike bagging, the Random space can encapsulate any base classifier. Moreover, the random selection of diverse feature subsets of the data-set can create diversity that subsequently leads to improve the accuracy (Ho, 1998).

Table 5.3: List of Bench-marked Classifier Ensembles

State-of-the-art Classifier Ensembles	Problem Domain	Employed Data-set
Bagging	Anomaly Detection	NSL-KDD
Random Subspace	Anomaly Detection	NSL-KDD

5.1.4.2 State-of-the-art Base Classifiers

The state-of- the-art base classifier that we select for the comparison with our proposed method belongs to different classification families. The simple logistic regression belongs to the Logistic Model trees (Landwehr, Hall, & Frank, 2005). However, the Naïve Bayes (Chen, Huang, Tian, & Qu, 2009) belongs to Bayes theorem; whereas, the BayesNet (Murphy et al., 2001) goes to the Bayesian model. Heoffding also belongs to the tree family (i.e., an incremental any time decision tree induction algorithm) (Pfahringer, Holmes, & Kirkby, 2007). The multilayer perceptron (MLP) belongs to Neural Networks (Ruck, Rogers, Kabrisky, Oxley, & Suter, 1990); whereas, Linear Discriminant Analysis (LDA) is simply a classical statistical method (Izenman, 2013). Table 5.4 clearly shows the state-of-the-art base-classifiers methods and , the problem domain, and the data-set employed.

5.2 Data Gathering for the Classification Performance of the Ensemble Methods

This section presents the data gathering from the experiments carried out for the classification performance comparisons of the aforementioned bench-marked algorithms with our proposed method. Since, we compare the performance of the proposed method with

Table 5.4: List of Bench-marked Base Classifiers

State-of-the-art Base Classifier	Problem Domain	Employed Data-set
Simple Logistic	Anomaly Detection	NSL-KDD
Naive-Bayes	Anomaly Detection	NSL-KDD
Linear Discriminant Analysis (LDA)	Anomaly Detection	NSL-KDD
BayesNet	Anomaly Detection	NSL-KDD
Heoffding	Anomaly Detection	NSL-KDD
Multilayer Perceptron (MLP)	Anomaly Detection	NSL-KDD

the state-of-the-art classifier ensembles as well as base classifiers in the field of classification based network anomaly detection. For the classification performance comparison, we have already declared all the standard three parameters used for the evaluation of any intrusion detection method. Moreover, for the evaluation, mainly the testing performance is taken into consideration. Additionally, the training performance where-ever necessary is also provided. The data gathered from the varied experiments carried out to show the classification performance comparisons of the proposed method with the state-of-the-art classifier ensembles as well as base classifiers on the basis of these these standard parameters/metrics, which are exclusively shown in the following sub-sections.

5.2.1 Performance Comparison of the Proposed Method

The section presents the performance comparison of the proposed Random Oracle-based method. For the evaluation, mainly the testing performance is taken into consideration. Additionally, the training performance where-ever necessary is also provided. Moreover, the section presents the 10 fold cross-validation average training and testing accuracies, and the average training and testing per-class accuracies with corresponding 10-fold cross validations. Additionally, we present the average results of the required performance evaluation parameters with their corresponding 10-fold cross validations. The 10-fold cross validations of the confusion-matrix of both training testing phase of the proposed method is given in Appendix.

The average results of the required performance evaluation parameters with their

corresponding 10-fold cross validation are shown in Table 5.5. The average training and testing per-class attack detection accuracies with their corresponding 10-fold cross validation are shown below in Table 5.6, Table 5.7, and Table 5.8 respectively.

Table 5.5: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the Random Oracle-based Intrusion Detection Method

Random Linear Oracle (RLO)	Accuracy	DR (Precision)	FNR	TNR	FPR	TPR (Recall)
Fold-1	99.39%	98.95%	2.72%	99.80%	0.20%	97.28%
Fold-2	99.32%	99.16%	3.33%	99.84%	0.16%	96.67%
Fold-3	99.45%	99.20%	2.55%	99.85%	0.15%	97.45%
Fold-4	99.38%	99.03%	2.84%	99.81%	0.19%	97.16%
Fold-5	99.43%	99.37%	2.88%	99.88%	0.12%	97.12%
Fold-6	99.36%	99.12%	3.05%	99.83%	0.17%	96.95%
Fold-7	99.46%	99.16%	2.47%	99.84%	0.16%	97.53%
Fold-8	99.39%	98.99%	2.72%	99.81%	0.19%	97.28%
Fold-9	99.51%	99.79%	2.80%	99.96%	0.04%	97.20%
Fold-10	99.43%	99.24%	2.72%	99.86%	0.14%	97.28%
Average	**99.41%**	**99.20%**	**2.81%**	**99.85%**	**0.15%**	**97.19%**

Table 5.6: The Average Training Per-class Accuracies with 10-Fold Cross-validations of the Proposed Random Oracle-based Intrusion Detection Method

Random Linear Oracle (Training)	Normal	DoS	U2R	R2L	Probe
Fold1	0.999	0.999	0.986	0.912	0.989
Fold2	0.998	0.999	0.987	0.931	0.99
Fold3	0.998	0.999	0.987	0.924	0.989
Fold4	0.9987	0.999	0.985	0.929	0.989
Fold5	0.9986	0.999	0.985	0.926	0.989
Fold6	0.999	0.999	0.981	0.911	0.984
Fold7	0.998	0.999	0.984	0.929	0.99
Fold8	0.998	0.999	0.986	0.92	0.989
Fold9	0.998	0.999	0.984	0.921	0.99
Fold10	0.998	0.999	0.983	0.919	0.989
Average Training Accuracy(%)	**99.89**	**99.97**	**98.53**	**92.28**	**98.93**

Table 5.7: The Average Testing Per-class Accuracies with 10-Fold Cross-validations of the Proposed Random Oracle-based Intrusion Detection Method

Random Linear Oracle (Testing)	Normal	DoS	U2R	R2L	Probe
Test1	0.997	0.998	0.981	0.863	0.972
Test2	0.998	0.999	0.971	0.899	0.992
Test3	0.997	0.999	0.994	0.889	0.988
Test4	0.997	0.999	0.971	0.914	0.982
Test5	0.997	0.999	0.971	0.899	0.982
Test6	0.997	0.999	0.976	0.891	0.984
Test7	0.997	0.998	0.984	0.909	0.985
Test8	0.997	0.999	0.966	0.894	0.99
Test9	0.996	0.999	0.989	0.902	0.985
Test10	0.996	0.999	0.968	0.899	0.983
Average Testing Accuracy (%)	**99.74**	**99.93**	**97.77**	**89.64**	**98.48**

Table 5.8: The 10 fold Cross Validation Per-class Average Training and Testing Accuracy of the Proposed Random Oracle-based Intrusion Detection Method

Random Linear Oracle (RLO)	Normal	DoS	U2R	R2L	Probe
Per Class Avg. Training Accuracy (%)	99.89	99.97	98.53	92.28	98.93
Per Class Avg. Testing Accuracy (%)	99.74	99.93	97.77	89.64	98.48

5.2.2 Performance Comparisons of the State-of-the-Art Ensemble Methods

The section presents the classification performance comparisons of the state-of-the-art ensembles methods (i.e. Bagging,and Random Subspace). Moreover, the ensembles comprises of base classifiers (i.e. MLP, CART). Furthermore, for the evaluation, mainly the testing performance is taken into consideration. Additionally, the training performance where-ever necessary is also provided. Moreover, the section also presents the per-class average attack detection accuracies, the average 10 fold cross-validation training plus testing accuracies, and the average results of the required performance evaluation parameters with their corresponding 10-fold cross validation data of the corresponding state-of-the-art ensembles methods.

The Bagging,Random Subspace average results of the required performance evaluation parameters with their corresponding 10-fold cross validation are shown in Table 5.9 and Table 5.14. The average training and testing per-class attack detection accuracies with their corresponding ten-fold cross validation of Bagging and Random Subspace are

shown in Table 5.10, Table 5.11, Table 5.12, Table 5.13, Table 5.15, Table 5.16, Table 5.17, and Table 5.18 respectively. Moreover, the average results of the F-measure with 10-fold cross validation of the proposed method, Bagging, and Random Subspace are shown in 5.19. Additionally, the overall performance comparison of the state-of-the-art classifier ensembles methods of the required evaluation parameters and the overall average detection per class accuracy performance comparisons are shown in Table 5.20, and Table 5.21.

Table 5.9: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the Bagging Ensemble Intrusion Detection Method

Bagging	Accuracy	DR (Precision)	FNR	TNR	FPR	TPR (Recall)
Fold-1	98.90%	99.66%	1.96%	99.69%	0.31%	98.04%
Fold-2	98.91%	99.80%	2.07%	99.82%	0.18%	97.93%
Fold-3	99.06%	99.70%	1.67%	99.73%	0.27%	98.33%
Fold-4	98.99%	99.80%	1.90%	99.82%	0.18%	98.10%
Fold-5	99.17%	99.76%	1.50%	99.78%	0.22%	98.50%
Fold-6	98.95%	99.72%	1.90%	99.74%	0.26%	98.10%
Fold-7	98.26%	99.71%	3.33%	99.74%	0.26%	96.67%
Fold-8	99.25%	99.33%	0.88%	99.38%	0.62%	99.12%
Fold-9	98.85%	99.71%	2.11%	99.74%	0.26%	97.89%
Fold-10	98.94%	99.69%	1.90%	99.71%	0.29%	98.10%
Average	**98.93%**	**99.69%**	**1.92%**	**99.71%**	**0.29%**	**98.08%**

Table 5.10: The 10-Fold Cross-validation Average Training and Testing Accuracies of the Bagging Ensemble Intrusion Detection Method

Bagging	Training Accuracy (%)	Testing Accuracy (%)
Fold1	99.19	98.9
Fold2	99.02	98.91
Fold3	99.19	99.06
Fold4	99.13	98.99
Fold5	99.19	99.17
Fold6	99.23	98.95
Fold7	98.43	98.26
Fold8	99.61	99.25
Fold9	99.09	98.85
Fold10	99.04	98.94
Average Accuracy (%)	**99.11**	**98.93**

Table 5.11: The Average Training Per-class Accuracies with 10-Fold Cross-validations of the Bagging Ensemble Intrusion Detection Method

Bagging (Training)	Normal	DoS	U2R	R2L	Probe
Fold1	0.9982	0.9972	0.9773	0.8554	0.9746
Fold2	0.9986	0.9975	0.9778	0.8116	0.9621
Fold3	0.9986	0.998	0.9796	0.8351	0.9742
Fold4	0.9986	0.9971	0.9773	0.8333	0.9709
Fold5	0.9984	0.997	0.9813	0.8474	0.975
Fold6	0.9986	0.9987	0.977	0.8414	0.9747
Fold7	0.9986	0.992	0.9493	0.7199	0.9492
Fold8	0.9974	0.9992	0.9899	0.9562	0.988
Fold9	0.9987	0.9966	0.9741	0.8276	0.9713
Fold10	0.9987	0.9976	0.9758	0.8041	0.9665
Average Training Accuracies (%)	**99.84**	**99.71**	**97.59**	**83.32**	**97.06**

Table 5.12: The Average Testing Per-class Accuracies with 10-Fold Cross-validations of the Bagging Ensemble Intrusion Detection Method

Bagging (Testing)	Normal	DoS	U2R	R2L	Probe
Test1	0.9969	0.9968	0.9819	0.8093	0.9594
Test2	0.9982	0.9976	0.9637	0.7887	0.9623
Test3	0.9973	0.9979	0.9819	0.8273	0.9671
Test4	0.9982	0.9961	0.9714	0.817	0.9681
Test5	0.9978	0.9987	0.9663	0.8376	0.9768
Test6	0.9974	0.9974	0.9767	0.8067	0.9632
Test7	0.9974	0.9901	0.9534	0.7139	0.9458
Test8	0.9938	0.9981	0.9767	0.9356	0.9816
Test9	0.9974	0.9942	0.9793	0.8196	0.9594
Test10	0.9971	0.9974	0.9611	0.8222	0.9632
Average Testing Accuracies (%)	**99.71**	**99.64**	**97.12**	**81.77**	**96.47**

Table 5.13: The 10-Fold Cross-validations Per-class Average Training and Testing Accuracy of the Bagging Ensemble Intrusion Detection Method

Bagging	Normal	DoS	U2R	R2L	Probe
Average training	99.84	99.71	97.59	83.32	97.06
Average testing	99.71	99.64	97.12	81.78	96.47

Table 5.14: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the Random Subspace Ensemble Detection Method

Random Subspace	Accuracy	DR (Precision)	FNR	TNR	FPR	TPR (Recall)
Fold-1	96.69%	99.79%	6.69%	99.82%	0.18%	93.31%
Fold-2	95.56%	99.80%	9.04%	99.83%	0.17%	90.96%
Fold-3	97.60%	99.72%	4.73%	99.75%	0.25%	95.27%
Fold-4	96.98%	99.88%	6.17%	99.90%	0.10%	93.83%
Fold-5	97.89%	99.82%	4.21%	99.84%	0.16%	95.79%
Fold-6	96.92%	99.69%	6.10%	99.73%	0.27%	93.90%
Fold-7	95.92%	99.62%	8.13%	99.68%	0.32%	91.87%
Fold-8	99.37%	99.68%	0.98%	99.70%	0.30%	99.02%
Fold-9	97.15%	99.85%	5.79%	99.87%	0.13%	94.21%
Fold-10	95.17%	99.60%	9.67%	99.66%	0.34%	90.33%
Average	**96.93%**	**99.75%**	**6.15%**	**99.78%**	**0.22%**	**93.85%**

Table 5.15: The 10-Fold Cross-validation Average Training and Testing Accuracies of the Random Subspace Ensemble Detection Method

Random Subspace	Training Accuracy (%)	Testing Accuracy (%)
Fold1	96.91	96.69
Fold2	95.64	95.56
Fold3	97.62	97.6
Fold4	97.14	96.98
Fold5	98.04	97.89
Fold6	97.28	96.92
Fold7	96.06	95.92
Fold8	99.65	99.37
Fold9	97.3	97.15
Fold10	95.41	95.17
Average Accuracy (%)	**97.11**	**96.92**

Table 5.16: The Average Training Per-class Accuracies with 10-Fold Cross-validation of the Random Subspace Ensemble Detection Method

Random Subspace (Training)	Normal	DoS	U2R	R2L	Probe
Fold1	0.9987	0.9865	0.9732	0.4442	0.8543
Fold2	0.9989	0.9739	0.9732	0.2824	0.7963
Fold3	0.9991	0.9847	0.9695	0.502	0.9413
Fold4	0.9987	0.9856	0.9758	0.541	0.854
Fold5	0.999	0.987	0.9741	0.5965	0.9538
Fold6	0.9988	0.9871	0.9767	0.5676	0.8563
Fold7	0.9984	0.969	0.9709	0.356	0.8583
Fold8	0.9989	0.9988	0.987	0.9396	0.9917
Fold9	0.9988	0.9832	0.9603	0.6747	0.8453
Fold10	0.9989	0.9721	0.9695	0.2136	0.8001
Average Training Accuracy (%)	**99.88**	**98.28**	**97.3**	**51.17**	**87.52**

Table 5.17: The Average Testing Per-class Accuracies with 10-Fold Cross-validation of the Random Subspace Ensemble Detection Method

Random Subspace (Testing)	Normal	DoS	U2R	R2L	Probe
Fold1	0.9982	0.986	0.9715	0.4072	0.8433
Fold2	0.9983	0.9757	0.9585	0.2629	0.793
Fold3	0.9975	0.9871	0.9663	0.4897	0.9439
Fold4	0.999	0.9837	0.961	0.5129	0.8549
Fold5	0.9984	0.9852	0.9767	0.5825	0.9506
Fold6	0.9973	0.9839	0.9637	0.5619	0.8395
Fold7	0.9968	0.97	0.9715	0.3557	0.8453
Fold8	0.997	0.9978	0.9741	0.9175	0.9845
Fold9	0.9987	0.9813	0.9715	0.6572	0.8356
Fold10	0.9966	0.973	0.956	0.2216	0.7793
Average Testing Accuracy (%)	**99.78**	**98.24**	**96.71**	**49.69**	**86.7**

Table 5.18: The 10-Fold Cross-validations Per-class Average Training and Testing Accuracy of the Random Subspace Ensemble Detection Method

Random Subspace	Normal	DoS	U2R	R2L	Probe
Per Class Avg. Training Accuracy (%)	99.88	98.28	97.3	51.17	87.52
Per Class Avg. Testing Accuracy (%)	99.78	98.24	96.71	49.69	86.7

Table 5.19: The Average Results of the F-measure with 10-Fold Cross-validations of the Random Linear Oracle (RLO), Random Subspace and Bagging ensembles

F-measure	**RandomLinearOracle (RLO)**	**RandomSubspace**	**Bagging**
Fold1	98.11%	96.44%	98.84%
Fold2	97.90%	95.18%	98.86%
Fold3	98.32%	97.44%	99.01%
Fold4	98.09%	96.76%	98.94%
Fold5	98.23%	97.76%	99.13%
Fold6	98.02%	96.71%	98.90%
Fold7	98.34%	95.59%	98.17%
Fold8	98.13%	99.35%	99.22%
Fold9	98.48%	96.95%	98.79%
Fold10	98.25%	94.74%	98.89%
Average	**99.20%**	**96.71%**	**98.88%**

Table 5.20: Overall Average Performance of the Required Evaluation Parameters of the Proposed method with the Current State-of-the-art Classifier Ensembles using 10-Fold Cross-validation

Classifier Ensembles	Accuracy	DR (Precision)	FNR	TNR	FPR	TPR (Recall)
Random Linear Oracle (RLO)	99.41%	99.20%	2.81%	99.85%	0.15%	97.19%
Bagging	98.93%	99.69%	1.92%	99.71%	0.29%	98.08%
Random Subspace	96.93%	99.75%	6.15%	99.78%	0.22%	93.85%

Table 5.21: Overall Average Detection Per-class Accuracy Performance of the Proposed Method with the Current State-of-the-art Classifier Ensembles using 10-Fold Cross- validation

Classifier Ensembles	Normal	DoS	U2R	R2L	Probe
RLO Avg. Testing Accuracy (%)	99.74	99.93	97.77	89.64	98.48
Bagging Avg. Testing Accuracy (%)	99.71	99.64	97.12	81.78	96.47
Random Subspace Avg. Testing Accuracy (%)	99.78	98.24	96.71	49.69	86.7

5.3 Data Gathering for the Classification Performance Comparisons of the State-of-the-Art Base Classifiers

The section presents the classification performance comparisons of the state-of-the-art bench-marked base classifiers. For the evaluation, mainly the testing performance is taken into consideration. Additionally, the training performance where-ever necessary is also provided. Moreover, the section also presents the per-class average attack detection accuracies.

The average results of the required performance evaluation parameters of all the benchmark classifiers with their corresponding 10-fold cross validation are shown in Table 5.22, Table 5.23, Table 5.24, Table 5.25, Table 5.26 and Table 5.27. The average training and testing per-class attack detection accuracies with their corresponding ten-fold cross validation of the base-classifiers are shown in Table 5.28, Table 5.29, Table 5.30, Table 5.31, Table 5.32, and Table 5.33. Moreover, the average results of the F-measure with 10-fold cross validation of the all the base classifier methods are shown in 5.34. Additionally, the overall performance comparison of the state-of-the-art base clas-

sifier plus the given ensemble methods of the required evaluation parameters are shown in 5.35.

Table 5.22: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the BayesNet Intrusion Detection Method

BayesNet	Accuracy	DR (Precision)	FNR	TNR	FPR	TPR (Recall)
Fold-1	94.51%	95.59%	3.15%	96.85%	4.41%	97.04%
Fold-2	94.63%	95.59%	2.64%	97.36%	4.41%	97.50%
Fold-3	94.37%	95.77%	3.34%	96.66%	4.23%	96.86%
Fold-4	94.55%	95.81%	2.63%	97.37%	4.19%	97.52%
Fold-5	94.29%	95.51%	3.26%	96.74%	4.49%	96.93%
Fold-6	94.73%	96.25%	2.67%	97.33%	3.75%	97.49%
Fold-7	94.61%	95.78%	2.95%	97.05%	4.22%	97.22%
Fold-8	94.33%	95.57%	3.26%	96.74%	4.43%	96.93%
Fold-9	94.61%	95.51%	3.07%	96.93%	4.49%	97.11%
Fold-10	94.65%	95.81%	2.91%	97.09%	4.19%	97.26%
Average	**94.53%**	**95.72%**	**2.99%**	**97.01%**	**4.28%**	**97.19%**

Table 5.23: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the Simple Logistic Intrusion Detection Method

SimpleLogistice	Accuracy	DR(Precision)	FNR	FPR	TNR	TPR(Recall)
Fold-1	98.66%	98.69%	0.88%	1.31%	99.12%	99.18%
Fold-2	98.66%	98.40%	0.99%	1.60%	99.01%	99.07%
Fold-3	98.66%	98.64%	0.81%	1.36%	99.19%	99.24%
Fold-4	98.66%	98.30%	0.95%	1.70%	99.05%	99.11%
Fold-5	98.66%	98.51%	1.18%	1.49%	98.82%	98.91%
Fold-6	98.66%	98.86%	0.94%	1.14%	99.06%	99.13%
Fold-7	98.66%	98.60%	1.02%	1.40%	98.98%	99.05%
Fold-8	98.66%	98.43%	1.02%	1.57%	98.98%	99.05%
Fold-9	98.66%	98.65%	1.02%	1.35%	98.98%	99.05%
Fold-10	98.66%	98.61%	1.04%	1.39%	98.96%	99.04%
Average	**98.66%**	**98.57%**	**0.99%**	**1.43%**	**99.01%**	**99.08%**

Table 5.24: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the Hoeffding-tree Based Intrusion Detection Method

Heoffding	Accuracy	DR (Precision)	FNR	TNR	FPR	TPR (Recall)
Fold-1	93.74%	97.87%	9.11%	90.89%	2.13%	92.05%
Fold-2	93.85%	97.99%	8.74%	91.26%	2.01%	92.35%
Fold-3	94.30%	98.44%	9.36%	90.64%	1.56%	91.89%
Fold-4	94.69%	98.52%	8.56%	91.44%	1.48%	92.54%
Fold-5	93.77%	98.14%	9.12%	90.88%	1.86%	92.06%
Fold-6	93.37%	97.83%	9.49%	90.51%	2.17%	91.75%
Fold-7	96.14%	98.03%	4.90%	95.10%	1.97%	95.57%
Fold-8	93.72%	97.69%	8.87%	91.13%	2.31%	92.23%
Fold-9	94.85%	98.69%	8.69%	91.31%	1.31%	92.45%
Fold-10	93.50%	97.94%	9.57%	90.43%	2.06%	91.69%
Average	**94.19%**	**98.11%**	**8.64%**	**91.36%**	**1.89%**	**92.46%**

Table 5.25: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the NaiveBayes Intrusion Detection Method

NaiveBayes	Accuracy	DR (Precision)	FNR	TNR	FPR	TPR (Recall)
Fold-1	85.78%	85.13%	6.09%	93.91%	14.87%	93.78%
Fold-2	83.65%	81.26%	5.22%	94.78%	18.74%	94.38%
Fold-3	85.33%	84.23%	5.54%	94.46%	15.77%	94.25%
Fold-4	85.65%	83.97%	4.94%	95.06%	16.03%	94.83%
Fold-5	84.72%	83.36%	5.68%	94.32%	16.64%	94.05%
Fold-6	84.19%	81.66%	5.40%	94.60%	18.34%	94.22%
Fold-7	86.15%	84.88%	5.40%	94.60%	15.12%	94.43%
Fold-8	85.33%	83.31%	5.01%	94.99%	16.69%	94.72%
Fold-9	85.69%	84.64%	5.60%	94.40%	15.36%	94.22%
Fold-10	85.79%	84.91%	5.46%	94.54%	15.09%	94.37%
Average	**85.23%**	**83.73%**	**5.43%**	**94.57%**	**16.27%**	**94.32%**

Table 5.26: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the Multilayer Perceptron (MLP) Intrusion Detection Method

MLP	Accuracy	DR (Precision)	FNR	FPR	TNR	TPR (recall)
Fold-1	98.11%	98.35%	2.29%	1.52%	98.48%	97.71%
Fold-2	98.15%	98.49%	2.35%	1.39%	98.61%	97.65%
Fold-3	98.26%	98.49%	2.11%	1.39%	98.61%	97.89%
Fold-4	97.72%	98.26%	3.02%	1.60%	98.40%	96.98%
Fold-5	98.35%	98.62%	2.06%	1.27%	98.73%	97.94%
Fold-6	97.66%	98.10%	2.99%	1.74%	98.26%	97.01%
Fold-7	97.77%	98.01%	2.66%	1.83%	98.17%	97.34%
Fold-8	97.97%	98.29%	2.52%	1.57%	98.43%	97.48%
Fold-9	97.62%	98.28%	3.26%	1.57%	98.43%	96.74%
Fold-10	98.12%	98.28%	2.20%	1.58%	98.42%	97.80%
Average	**97.97%**	**98.32%**	**2.55%**	**1.55%**	**98.45%**	**97.45%**

Table 5.27: The Average Results of the Required Performance Evaluation Parameters with their Corresponding 10-Fold Cross-validation of the Linear Discriminant Analysis (LDA) Intrusion Detection Method

LDA	Accuracy	DR (Precision)	FNR	FPR	TNR	TPR(Recall)
Fold-1	89.90%	92.00%	13.46%	6.98%	93.02%	86.54%
Fold-2	90.51%	92.52%	12.66%	6.55%	93.45%	87.34%
Fold-3	89.97%	91.71%	12.97%	7.29%	92.71%	87.03%
Fold-4	89.85%	92.28%	13.88%	6.68%	93.32%	86.12%
Fold-5	90.32%	92.34%	12.90%	6.70%	93.30%	87.10%
Fold-6	89.94%	92.43%	13.85%	6.54%	93.46%	86.15%
Fold-7	90.06%	92.15%	13.27%	6.85%	93.15%	86.73%
Fold-8	90.37%	92.67%	13.14%	6.37%	93.63%	86.86%
Fold-9	90.27%	92.71%	13.40%	6.32%	93.68%	86.60%
Fold-10	89.91%	92.00%	13.43%	6.98%	93.02%	86.57%
Average	**90.11%**	**92.28%**	**13.30%**	**6.73%**	**93.27%**	**86.70%**

Table 5.28: Multilayer Perceptron (MLP) Average Testing Per-class Attack Detection Accuracies with 10-Fold Cross-validation

MLP	Normal	DoS	U2R	R2L	Probe
Test1	0.984815	0.995692	0.958549	0.778351	0.962282
Test2	0.986113	0.993257	0.950777	0.793814	0.968085
Test3	0.986113	0.99513	0.955959	0.798969	0.970958
Test4	0.984038	0.989698	0.955844	0.739691	0.958414
Test5	0.987283	0.995317	0.953368	0.814433	0.969022
Test6	0.982609	0.988575	0.935233	0.770619	0.962282
Test7	0.981703	0.991195	0.955959	0.783505	0.959381
Test8	0.984296	0.992319	0.940415	0.791237	0.966151
Test9	0.984296	0.988575	0.955959	0.71134	0.958414
Test10	0.984168	0.992133	0.96114	0.82732	0.968054
Avg. Testing Accuracy	98.45	99.22	95.23	78.09	96.43

Table 5.29: Linear Discriminant Analysis (LDA) Average Testing Per-class Attack Detection Accuracies with 10-Fold Cross-validation

LDA	Normal	DoS	U2R	R2L	Probe
Test1	0.930175	0.932572	0.893782	0.420103	0.675048
Test2	0.934458	0.935007	0.917098	0.43299	0.704062
Test3	0.92706	0.927515	0.909326	0.451031	0.717328
Test4	0.933169	0.925454	0.916883	0.42268	0.673114
Test5	0.933039	0.926377	0.906736	0.525773	0.700871
Test6	0.934588	0.924143	0.878238	0.445876	0.687621
Test7	0.931482	0.927876	0.911917	0.518041	0.669246
Test8	0.936275	0.929374	0.860104	0.479381	0.704062
Test9	0.936794	0.930137	0.909326	0.456186	0.672147
Test10	0.930184	0.927515	0.904145	0.481959	0.675702
Avg. Testing Accuracy	93.27	92.86	90.08	46.34	68.79

Table 5.30: The Average Testing Detection Accuracies of the State-of-the-art Base Classifiers (i.e., LDA, and MLP) with 10-Fold Cross-validation

Testing	LDA	MLP
Test1	89.90035	98.108
Test2	90.50633	98.1484
Test3	89.97374	98.26274
Test4	89.85322	97.71748
Test5	90.31715	98.35028
Test6	89.94075	97.65688
Test7	90.06194	97.77134
Test8	90.37102	97.9732
Test9	90.27067	97.61648
Test10	89.91382	98.12147
Average testing Accuracies	90.1109	97.97263

Table 5.31: The Average Training Detection Accuracies of the State-of-the-art Base classifiers (i.e., LDA, and MLP) with 10-Fold Cross-validation

Training	LDA	MLP
Fold1	90.10212	98.22541
Fold2	90.06845	98.02267
Fold3	90.10668	98.32119
Fold4	90.13728	97.97928
Fold5	90.0805	98.36458
Fold6	90.21659	97.69798
Fold7	90.08491	97.79598
Fold8	90.13885	98.14089
Fold9	90.0924	97.79972
Fold10	90.14177	98.28601
Average training accuracies	90.1169	98.0633

Table 5.32: Average Testing Per-class Attack Detection Accuracies of the State-of-the-art Base classifiers (i.e., LDA, and MLP) using 10-Fold Cross-validation

Avg. Testing Accuracies	Normal	DoS	U2R	R2L	Probe
LDA	93.27	92.86	90.08	46.34	68.79
MLP	98.45	99.22	95.23	78.09	96.43

Table 5.33: Average Testing Per-class Attack Detection Accuracies of the State-of-the-art Base classifiers (i.e., LDA, and MLP) 10-Fold Cross-validation

Avg. Training Accuracies	Normal	DoS	U2R	R2L	Probe
LDA	93.27	92.86	90.01	46.35	68.83
MLP	98.56	99.26	95.41	78.21	96.52

Table 5.34: The Average Results of the F-measure with 10-Fold Cross-validations of the Bench-marked Base Classifiers

F-measure	SimpleLogistic	MLP	LDA	NaiveBayes	BayesNet	Hoeffiding
Fold1	98.93%	98.03%	89.19%	89.25%	96.31%	94.87%
Fold2	98.74%	98.07%	89.85%	87.33%	96.53%	95.09%
Fold3	98.94%	98.19%	89.31%	88.96%	96.31%	95.06%
Fold4	98.70%	97.61%	89.09%	89.07%	96.65%	95.44%
Fold5	98.71%	98.28%	89.64%	88.39%	96.21%	95.01%
Fold6	98.99%	97.55%	89.18%	87.49%	96.87%	94.69%
Fold7	98.82%	97.68%	89.36%	89.40%	96.50%	96.78%
Fold8	98.74%	97.89%	89.67%	88.65%	96.25%	94.88%
Fold9	98.85%	97.50%	89.55%	89.17%	96.30%	95.47%
Fold10	98.82%	98.04%	89.20%	89.39%	96.53%	94.71%
Average	98.82%	97.88%	89.40%	88.71%	96.45%	95.20%

Table 5.35: Overall Performance of the Proposed Method with all the Bench-marked Classifiers of the Required Evaluation Parameters with 10-Fold Cross-validation

Intrusion Detection Mechanisims	**Accuracy**	**Precision**	**Recall**	**F-measure**
RandomLinearOracle (RLO)	99.41	99.2	97.19	99.2
SimpleLogistic	98.57	98.57	99.08	98.82
MultilayerPerceptron (MLP)	97.97	98.32	98.45	97.88
LinearDiscimenantAnalysis (LDA)	90.11	92.28	86.7	89.4
NaiveBayes	85.23	83.73	94.32	88.71
BayesNet	94.53	95.72	97.19	96.45
Tree-Hoeffiding	94.19	98.11	92.46	95.2
RandomSubspace	96.93	99.75	93.85	96.71
Bagging	98.93	99.69	98.08	98.88

5.4 Conclusion

The proposed random oracle based intrusion detection method is properly tested and evaluated. Moreover, a detailed explanation of the tools and the bench-mark data-sets used for the entire evaluation of the proposed random linear oracle (RLO) model is also provided. The complete set-up environment, the pre-requisites, and the programming tools and language used for the final implementation. Furthermore, the standard parameters used for assessing the performance of the proposed model are also presented. The detailed verification of the proposed method with the current state-of-the-art is also carried out. Finally, the chapter concludes with a 10-fold cross-validation of all the performance evaluation parameters in order to show a completely unbiased resulting performance.

CHAPTER 6: RESULTS AND DISCUSSION

The chapter presents the results of the experimental research (i.e., simulation, and emulation) of the proposed intrusion-detection method with their corresponding counterparts. Moreover, the discussion goes one step beyond by comparing the analysis of the results to demonstrate the validity of the proposed approach. The conducted experimental research demonstrates the promising results and better performance of the proposed intrusion detection method. The main objective of the chapter is to demonstrate the outstanding performance of the random oracle based proposed intrusion detection method compared to the state-of-the-art classifier ensembles (i.e, Bagging and Random Subspace) in this domain. Moreover, to verify the out-performance of the random oracle based intrusion-detection method compared to state-of-the-art base-classifiers (i.e, Simple Logistic, Multilayer Perceptron (MLP), Naive-Bayes, BayesNet, Linear Discriminant Analysis (LDA), and Hoeffding)) in this domain. Additionally, to demonstrate that the proposed method is viable to identify diverse attacks triggered by large-scale malicious events in the control-plane of the SDNs in real-time.

The remainder of this chapter is structured as follows: Section 6.1 presents the detailed performance analysis of the proposed method compared to the classifier ensembles on the standard evaluation parameters. Section 6.2 shows the detailed performance analysis of the proposed method compared to state-of-the-art single classifiers based intrusion detection mechanisms on the standard evaluation parameters. In Section 6.3, we have demonstrated the detailed analysis of the computation and communication cost to show the viability of the proposed method in real-time detection of attacks in the control-plane of the SDNs. Finally, we provide the concluding remarks in Section 6.4.

6.1 Performance Comparison Analysis of the Proposed Approach with Classifier Ensembles

In this section, we extensively elaborate the performance comparison of our proposed Random Linear Oracle (RLO) based intrusion detection method with the corresponding employed ensembles intrusion detection mechanisms. Moreover, the section also presents the detailed analysis of each and every performance evaluation parameter.

6.1.1 Detection Accuracy Performance

Accuracy parameter is considered one of the primary and powerful parameters to demonstrate the performance of an intrusion detection system. Figure 6.1 presents the overall average accuracy (in percent) for the benchmark NSL-KDD data set using 10-fold cross validation. Although the accuracy achieved above 90% by any intrusion detection technique is considered relatively good detection accuracy rate. However, the proposed Random Linear Oracle (RLO) based intrusion detection method outperforms the other tested classifier ensembles intrusion detection mechanisms. The proposed method correctly identifies diverse attacks with an average detection accuracy rate of 99.41% compares to Bagging and Random Sub-space ensembles detection mechanisms that achieve the accuracy rate of 98.93% and 96.93% respectively. Likewise, for more clarity; we also present the accuracy rates (in percent) using 10-fold cross validation. Figure 6.2 clearly depicts the outstanding performance in terms of detection accuracy from their corresponding counterpart ensembles detection mechanisms.

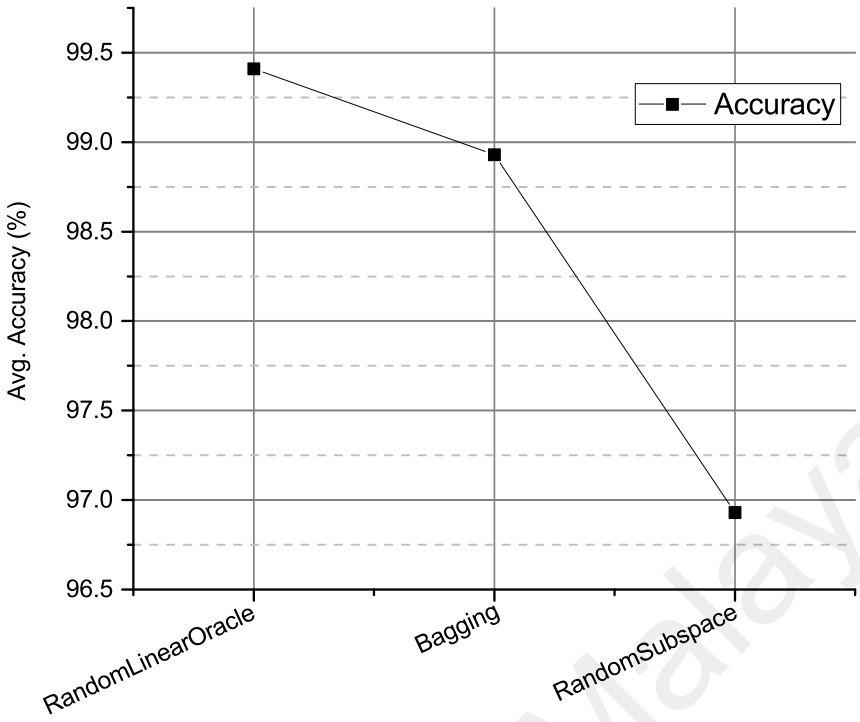

Figure 6.1: Average Detection Accuracy Performance Comparison of the 10-fold Cross-validation

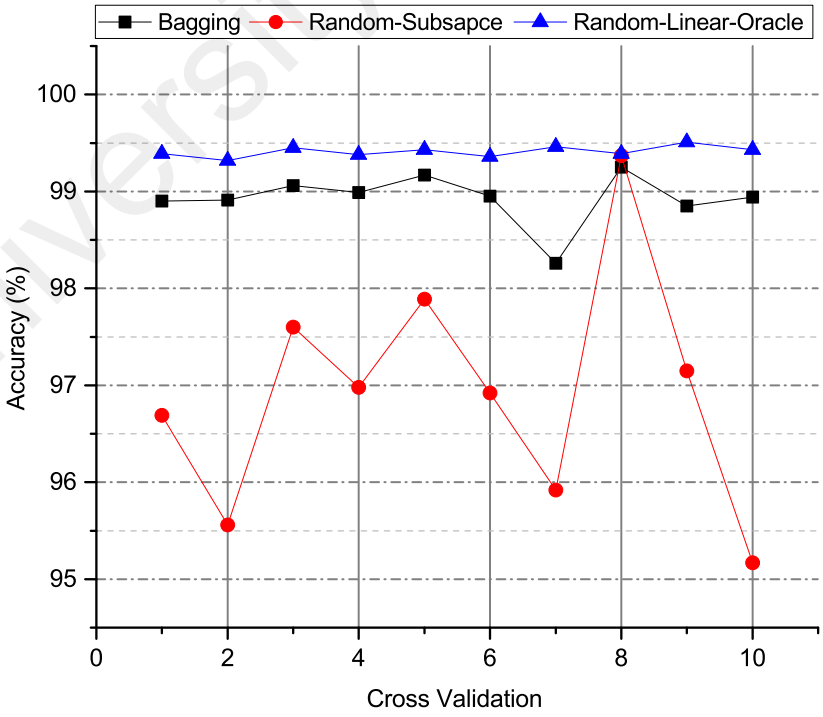

Figure 6.2: Detection Accuracy Performance Comparison of the 10-fold Cross-validation

6.1.2 Standard F-measure Performance

One of the significant parameter to measure the performance of an intrusion detection system is F-measure, which is the harmonic mean of recall and precision. We use the standard F-measure to evaluate the performance of Random Linear Oracle (RLO) based intrusion detection method. F-measure represents the tradeoff between the recall and precision. Figure 6.3 presents the overall average F-measure (in percent) for the benchmark NSL-KDD data-set using 10-fold cross validation. The F-measure rate of the proposed Random Linear Oracle (RLO) based intrusion detection method outperforms the other tested classifier ensembles. Similarly, for more clarity; we also present the F-measure rates (in percent) using 10-fold cross validation. Figure 6.4 also clearly depicts the significant performance in terms of F-measure rates (in percent) from their corresponding counterpart ensembles detection mechanisms.

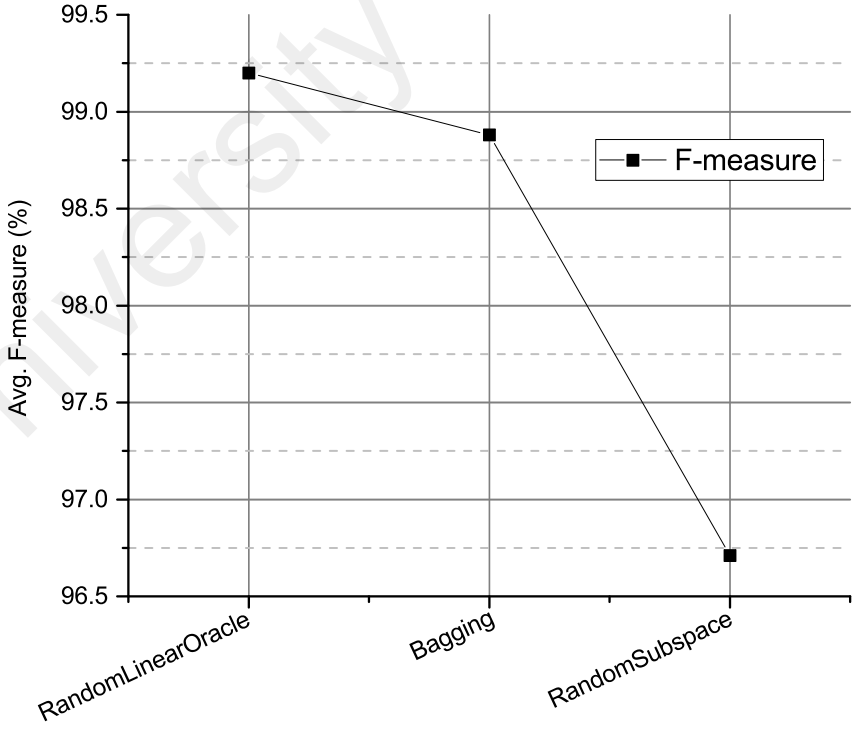

Figure 6.3: Average F-measure Performance Comparison of the 10-fold Cross-validation

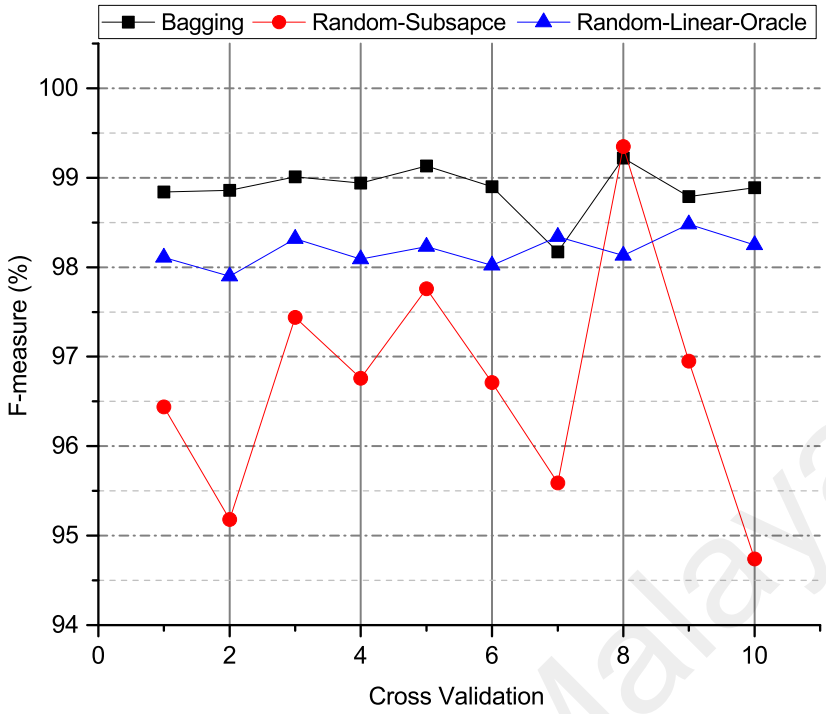

Figure 6.4: F-measure Performance Comparison of the 10-fold Cross-validation

6.1.3 Precision Performance

Precision parameter is simply used for measuring the relevance and supports the accuracy rate of an intrusion detection system. Figure 6.5 presents the overall average precision (in percent) for the benchmark NSL-KDD data-set using 10-fold cross validation. The average precision rate of the Random Subspace and Bagging algorithm is a bit high than the proposed Random Linear Oracle (RLO) based intrusion detection method as shown in Figure 6.5. However, the precision rates above 99% achieved by any intrusion detection algorithms is considered a significant performance. Similarly, the precision rates (in percent) using 10-fold cross validation in Figure 6.6 depicts the precision performance of all the ensemble detection mechanisms.

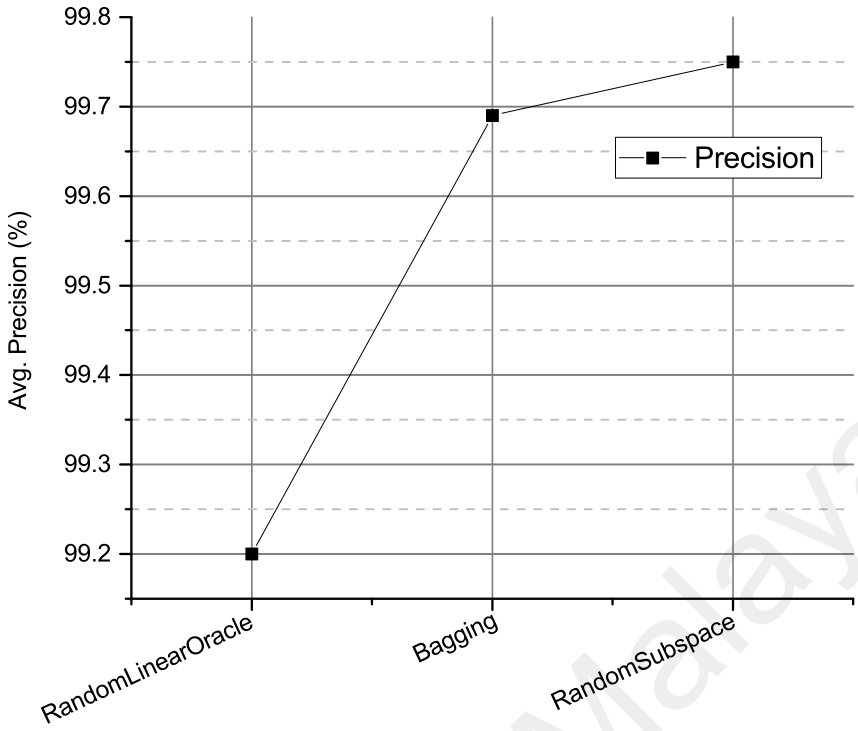

Figure 6.5: Average Precision Performance Comparison of the 10-fold Cross-validation

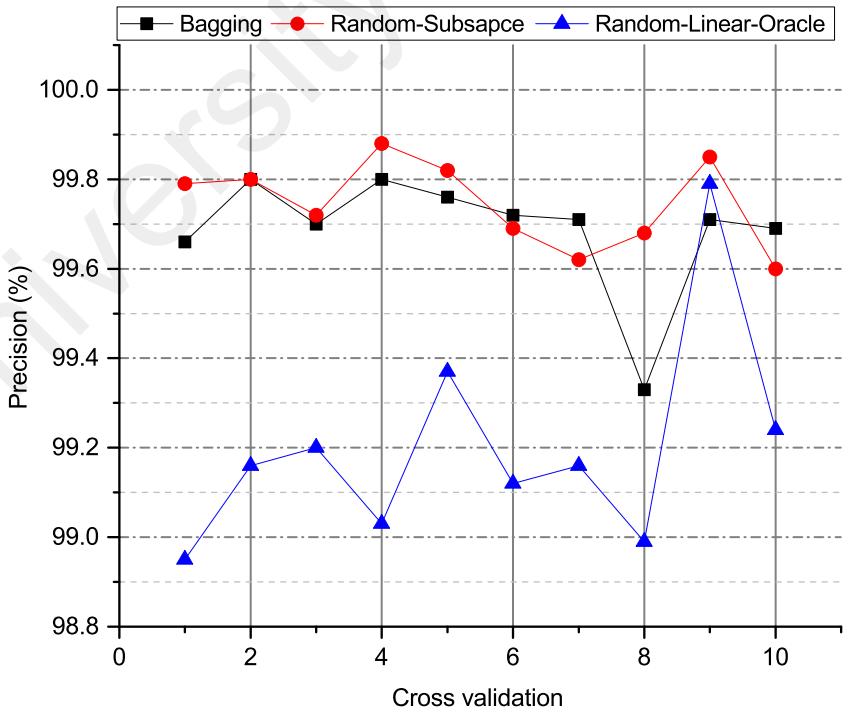

Figure 6.6: Precision Performance Comparison using 10-fold Cross-validation

6.1.4 Recall Performance

Recall parameter is also used for measuring the relevance and supports the accuracy rate of an intrusion detection system. Figure 6.7 presents the overall average recall (in percent) for the benchmark NSL-KDD data-set using 10-fold cross validation. The average recall rate of the Bagging algorithm outperforms than the other tested classifier ensembles intrusion detection mechanisms. However, the average recall rate of the proposed Random Linear Oracle (RLO) based intrusion detection method is higher than the Random Subspace detection algorithm. Similarly, the recall rates (in percent) using 10-fold cross validation in Figure 6.8 depicts the recall performance of all the ensemble detection mechanisms.

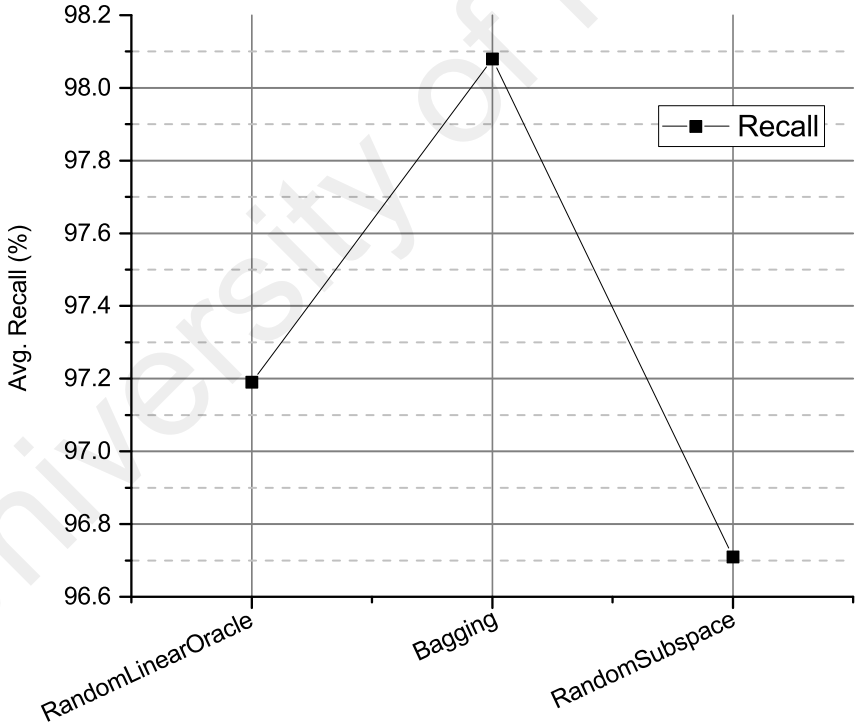

Figure 6.7: Average Recall Performance Comparison of the 10-fold Cross-validation

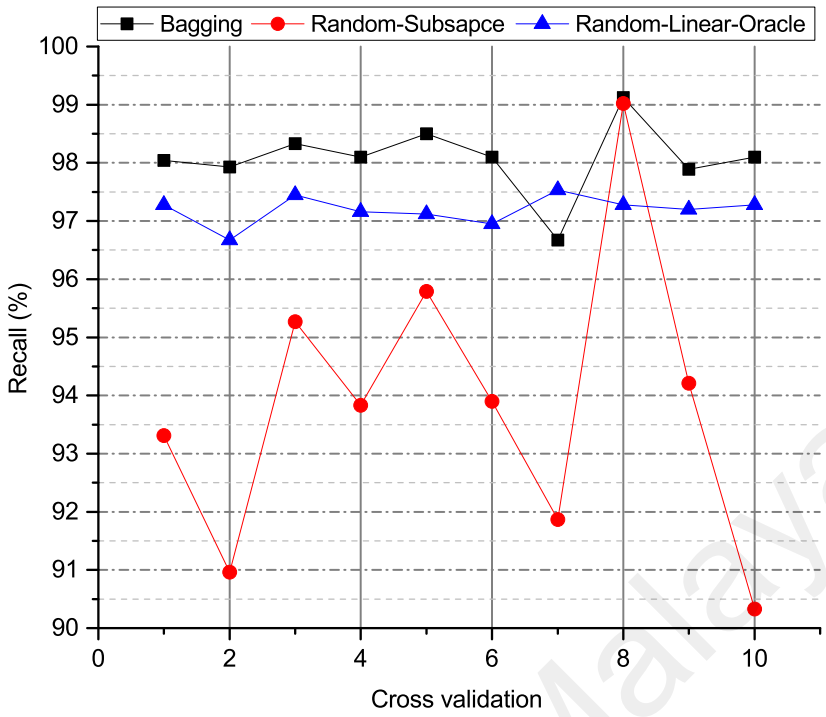

Figure 6.8: Recall Performance Comparison using 10-fold Cross-validation

6.1.5 Overall performance

The section presents the net performance of the classifier ensembles detection mechanisms as shown in Figure 6.9 and Figure 6.10, when applied to the benchmark NSL-KDD data-set. Figure 6.9 clearly demonstrates that the proposed Random Linear Oracle (RLO) based intrusion detection method outperforms the classifier ensembles in terms of detection accuracy and f-measure rates (in percent). Likewise, the proposed Random Linear Oracle (RLO) based intrusion detection method shows significant performance compare to its counterpart detection mechanism with the highest TNR values and the minimum FPR value. However, the TPR value of the bagging detection algorithm is slightly improved than the proposed Random Linear Oracle (RLO) based intrusion detection method. Figure 6.10 depicts the overall accuracy comparison of the classifier ensemble detection mechanisms.

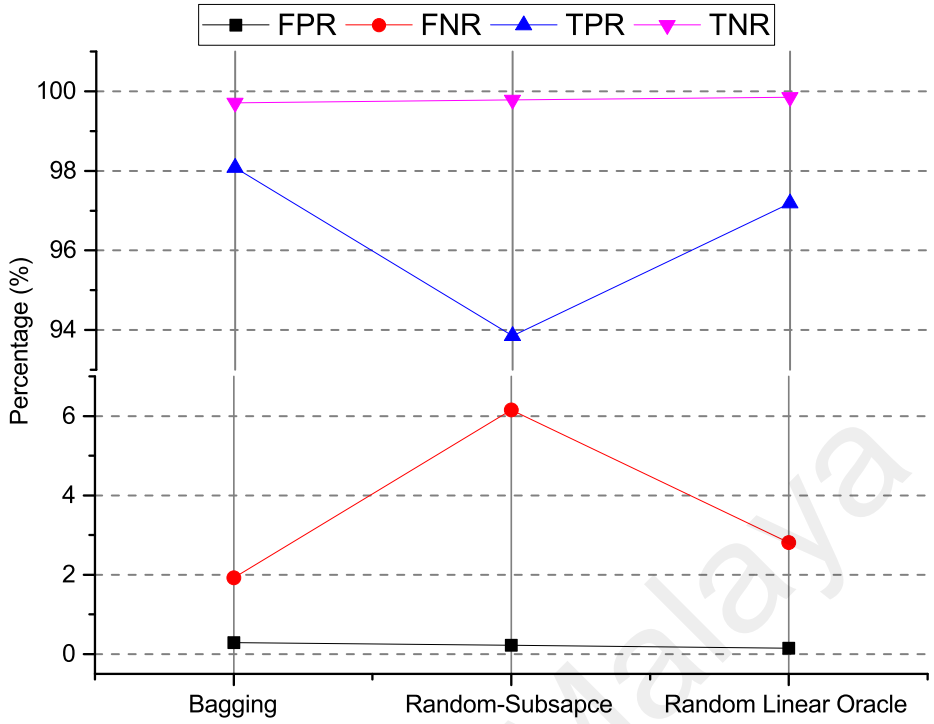

Figure 6.9: Net Performance Comparison of the Classifier Ensembles Detection Mechanisms

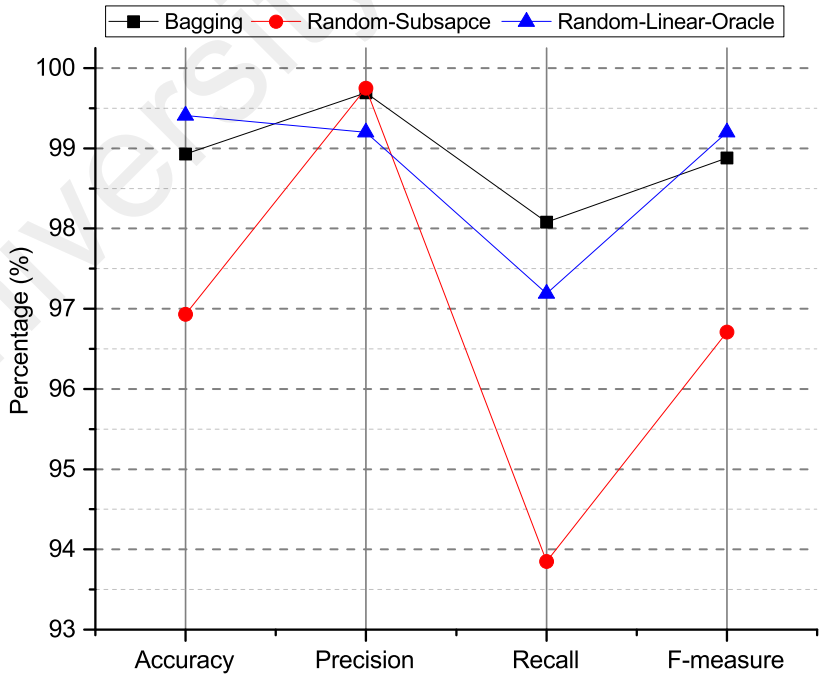

Figure 6.10: Overall Accuracy Comparison of the Classifier Ensembles Detection Mechanisms

6.2 Performance Comparison Analysis of the Proposed Approach with State-of-the-art Base-Classifiers

In this section, we extensively elaborate the performance comparison of our proposed Random Linear Oracle (RLO) based intrusion detection method with state-of-the-art single classifier based intrusion detection mechanisms. Moreover, the section also presents the detailed performance analysis of the employed evaluation parameter.

6.2.1 Detection Accuracy Performance

Accuracy parameter is considered one of the primary and powerful parameters to demonstrate the performance of an intrusion detection system. Figure 6.10 presents the overall average accuracy (in percent) for the benchmark NSL-KDD data set using 10-fold cross validation. Although the accuracy achieved above 90% by any intrusion detection technique is considered relatively good detection accuracy rate. However, the proposed Random Linear Oracle (RLO) based intrusion detection method outperforms the other tested single classifier based intrusion detection mechanisms. The proposed method correctly identifies diverse attacks with an average detection accuracy rate of 99.41%, which is the highest detection accuracy rate compared to their corresponding state-of-the-art base classifiers. Likewise, for more clarity; we also present the accuracy rates (in percent) using 10-fold cross validation, which clearly depicts the outstanding performance of the proposed method in terms of detection accuracy from their corresponding counterpart base classifiers.

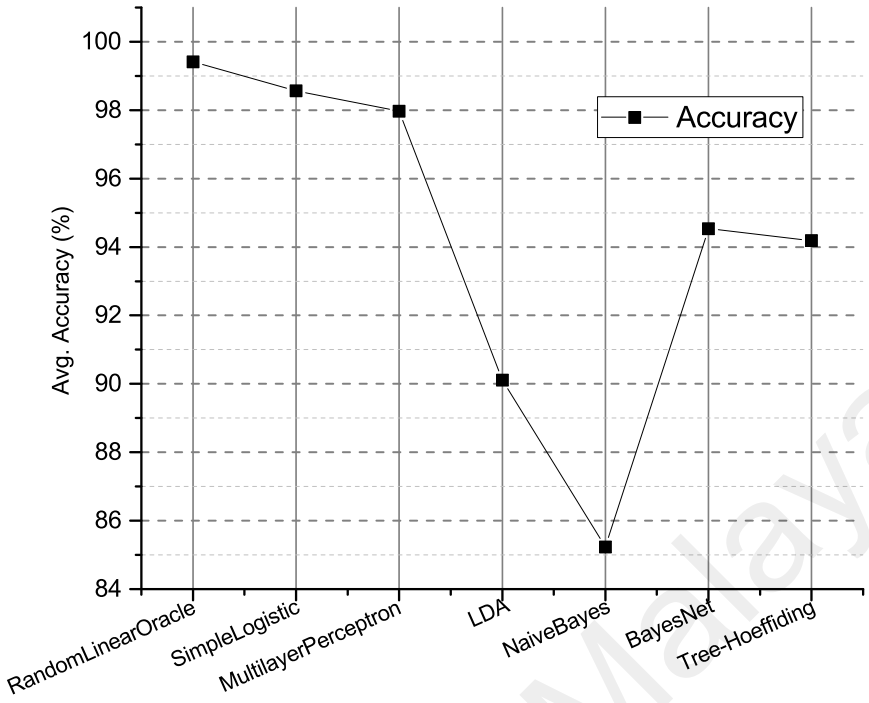

Figure 6.11: Average Detection Accuracy Performance Comparison of the 10-fold Cross-validation

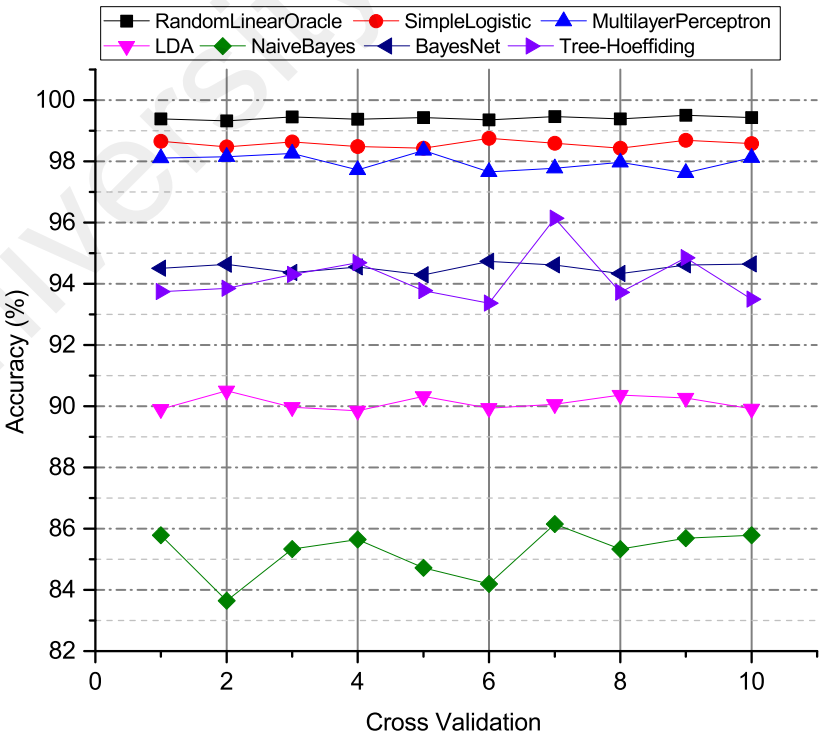

Figure 6.12: Detection Accuracy Performance Comparison using 10-fold Cross-validation

6.2.2 Standard F-measure Performance

One of the significant parameter to measure the performance of an intrusion detection system is F-measure, which is the harmonic mean of recall and precision. We use the standard F-measure to evaluate the performance of Random Linear Oracle (RLO) based intrusion detection method. F-measure represents the tradeoff between the recall and precision. Figure 6.3 presents the overall average F-measure (in percent) for the benchmark NSL-KDD data-set using 10-fold cross validation. The highest F-measure rate of the proposed Random Linear Oracle (RLO) based intrusion detection method outperforms the other tested classifier ensembles. The average F-measure rate of the proposed method is 99.4% as shown in Figure 6.13, the attained F-measure rate is the highest compared to the rest of the state-of-the-art base classifiers. Similarly, for more clarity; we also present the F-measure rates (in percent) using 10-fold cross validation as shown in Figure 6.14.

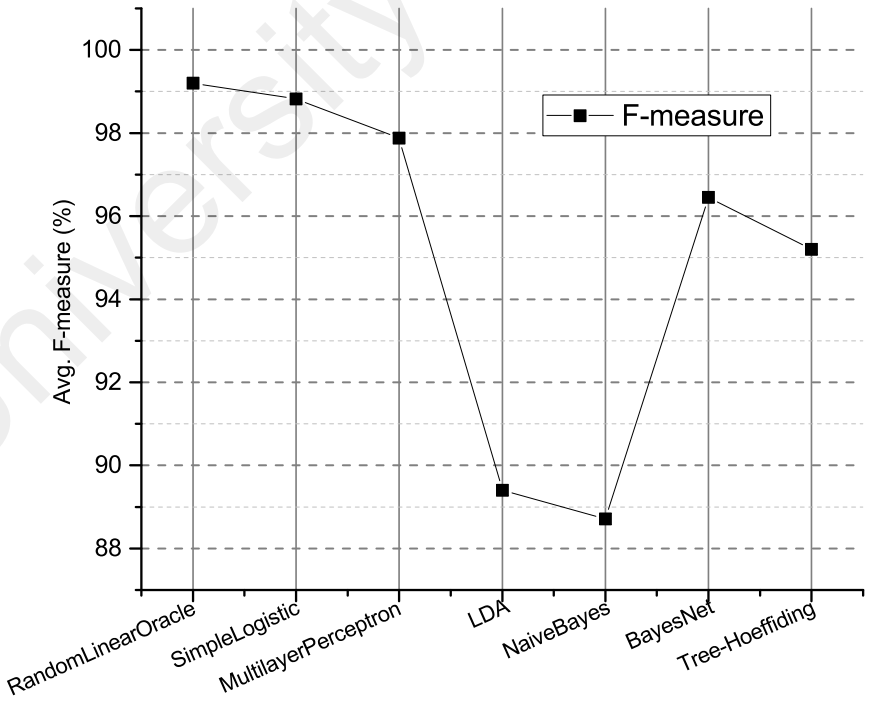

Figure 6.13: Average F-measure Performance Comparison using 10-fold Cross-validation

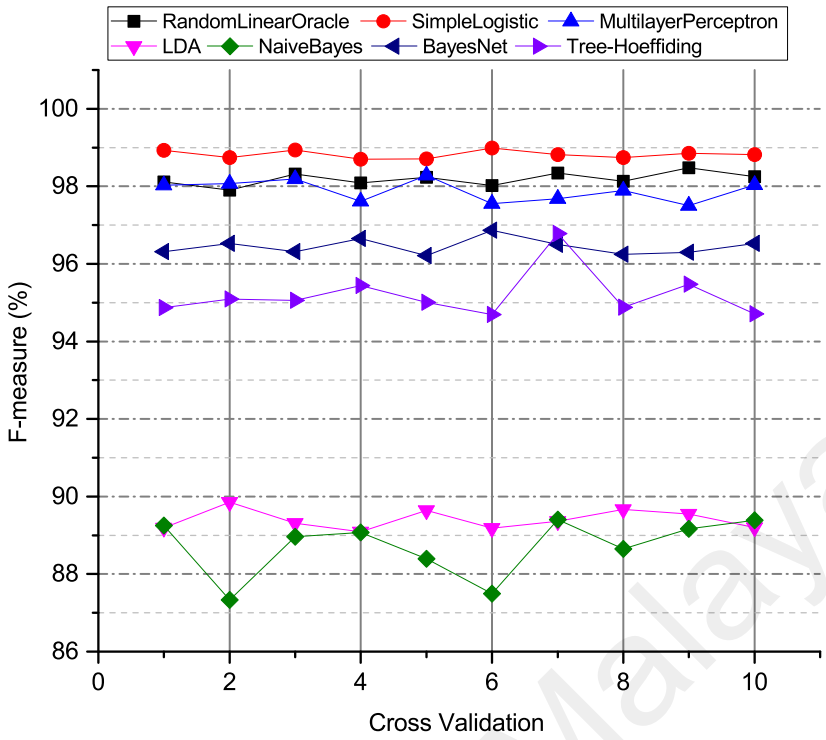

Figure 6.14: F-measure Performance Comparison using 10-fold Cross-validation

6.2.3 Precision Performance

Precision parameter is simply used for measuring the relevance and supports the accuracy rate of an intrusion detection system. Figure 6.15 presents the overall average precision (in percent) for the benchmark NSL-KDD data-set using 10-fold cross validation. The average precision rate of the proposed Random Linear Oracle (RLO) based intrusion detection method outperforms their counterparts. However, the precision rates (in percent) using 10-fold cross validation of the proposed intrusion detection algorithms also shows significant performance compared to their corresponding counterparts as shown in Figure 6.16.

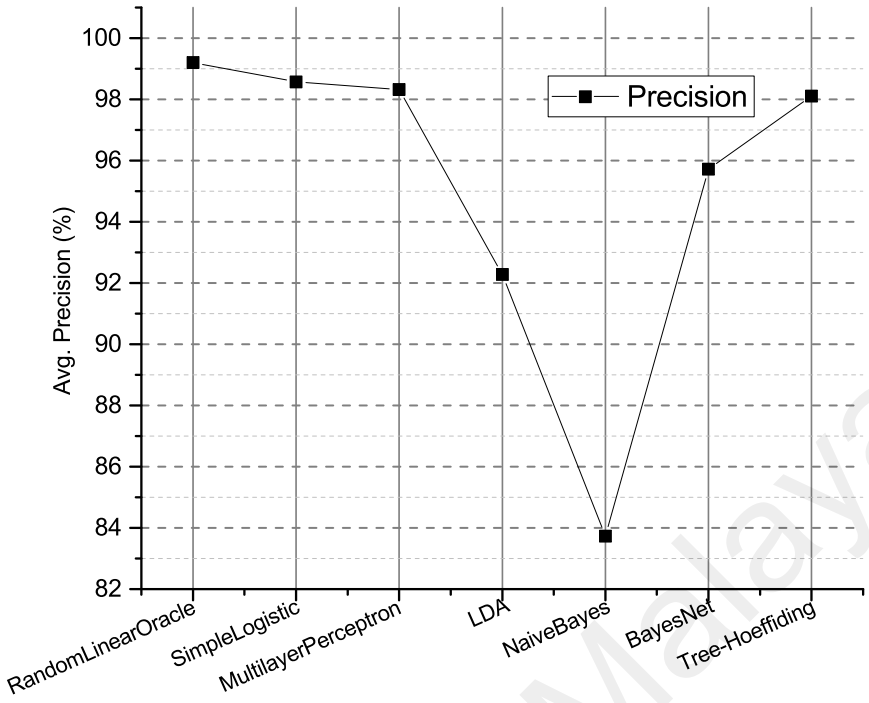

Figure 6.15: Average Precision Performance Comparison using 10-fold Cross-validation

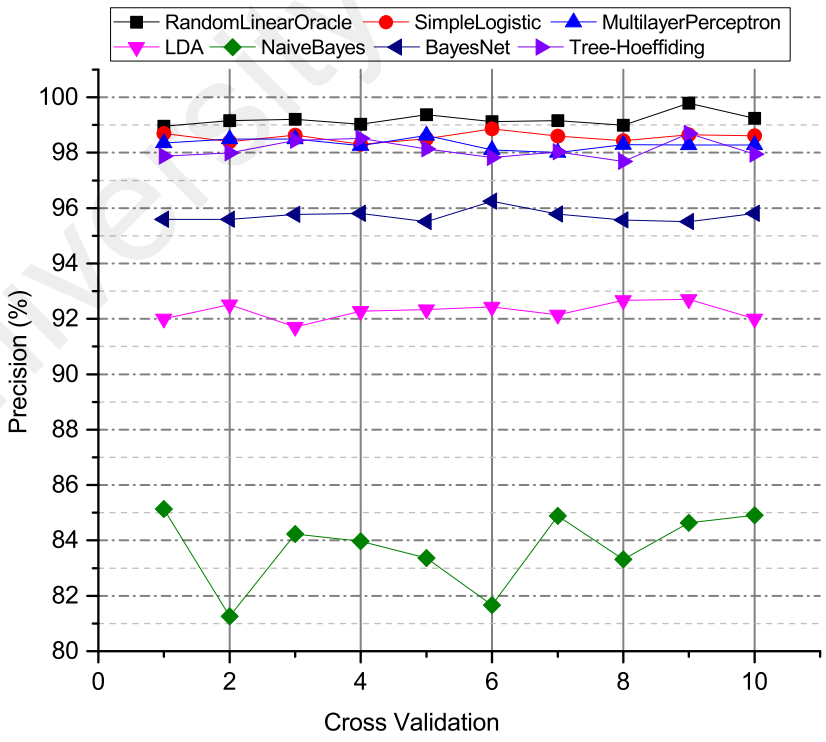

Figure 6.16: Precision Performance Comparison using 10-fold Cross-validation

6.2.4 Recall Performance

Recall parameter is also used for measuring the relevance and supports the accuracy rate of an intrusion detection system. Figure 6.17 presents the overall average recall (in percent) for the benchmark NSL-KDD data-set using 10-fold cross validation. The average recall rate of the Simple Logistic and MLP is a bit higher than the proposed intrusion detection approach. However, the average recall rate of the proposed Random Linear Oracle (RLO) based intrusion detection method is higher than the rest of their counterparts. Similarly, we also present the recall rates (in percent) using 10-fold cross validation. Moreover, Figure 6.18 depicts the recall rates (in percent) using 10-fold cross validation of all the classification based intrusion detection methods.

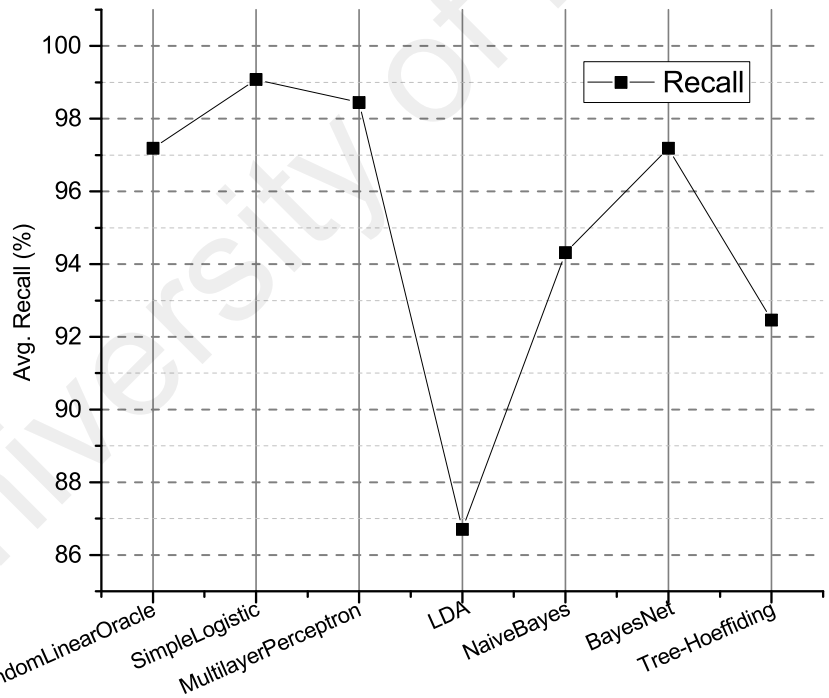

Figure 6.17: Average Recall Performance Comparison using 10-fold Cross-validation

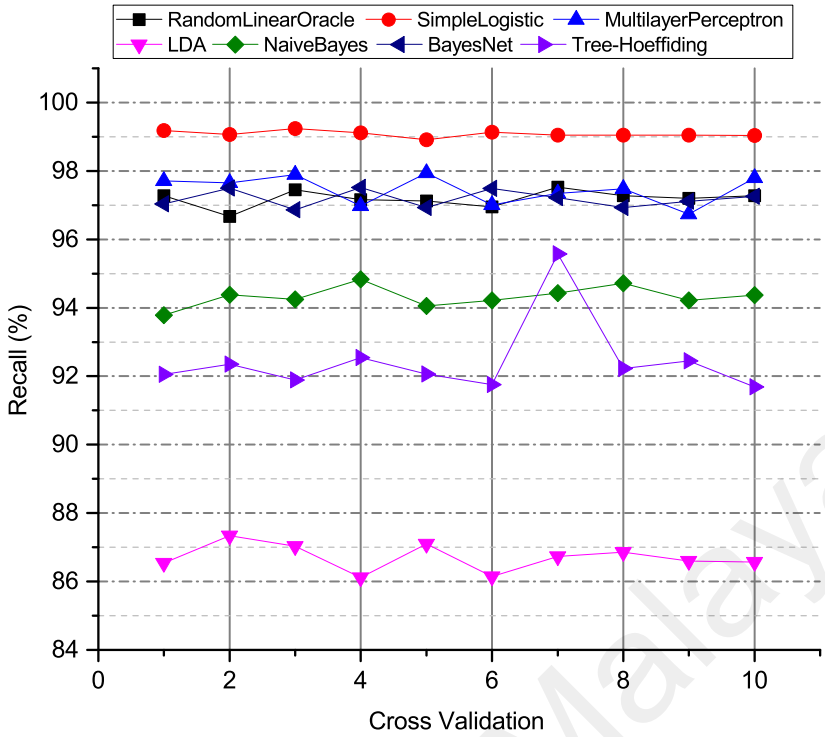

Figure 6.18: Recall Performance Comparison using 10-fold Cross-validation

6.2.5 Overall performance

The section presents the net performance of the classifier ensembles detection mechanisms of the proposed method with their corresponding counterparts when applied to the benchmark NSL-KDD data-set. From Figure 6.19, it can be easily concluded that the proposed Random Linear Oracle (RLO) based intrusion detection method outperforms the other tested base classifiers in terms of detection accuracy, precision, and F-measure rates (in percent). However, the recall rate (in percent) of the Simple Logistics is slightly higher than the proposed method. Likewise, the proposed Random Linear Oracle (RLO) based intrusion detection method outperforms compare to its counterpart detection mechanisms with the highest TNR value and the minimum FPR value. However, the TPR value of the Simple Logistic detection algorithm is slightly improved than the proposed Random Linear Oracle (RLO) based intrusion detection method. Figure 6.20 depicts the overall accuracy comparison of the classifier ensemble detection mechanisms.

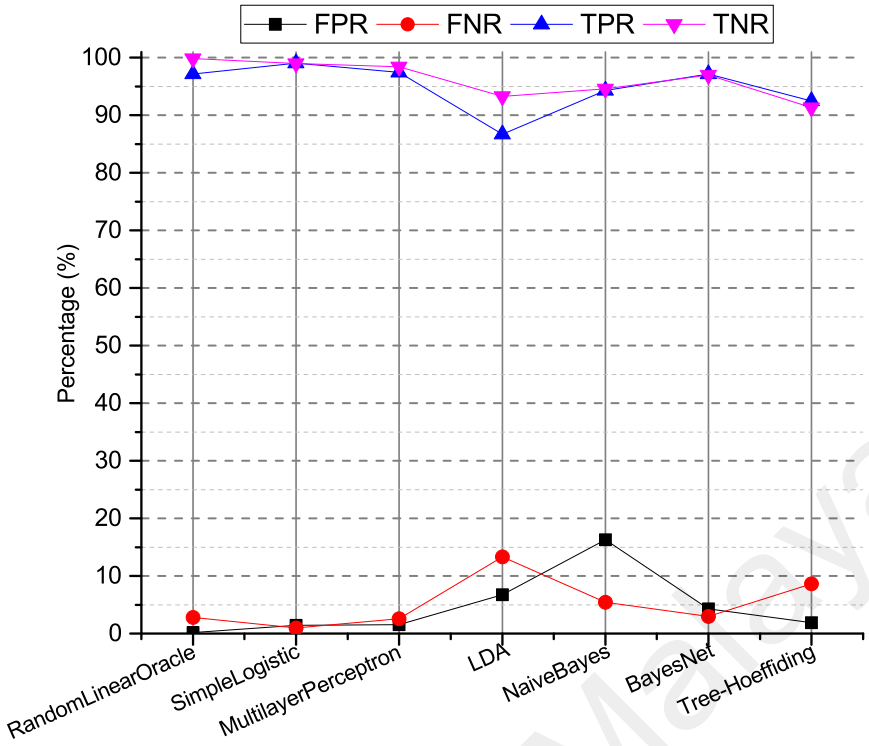

Figure 6.19: Net Performance Comparison of the Base Classifiers Intrusion Detection Mechanisms

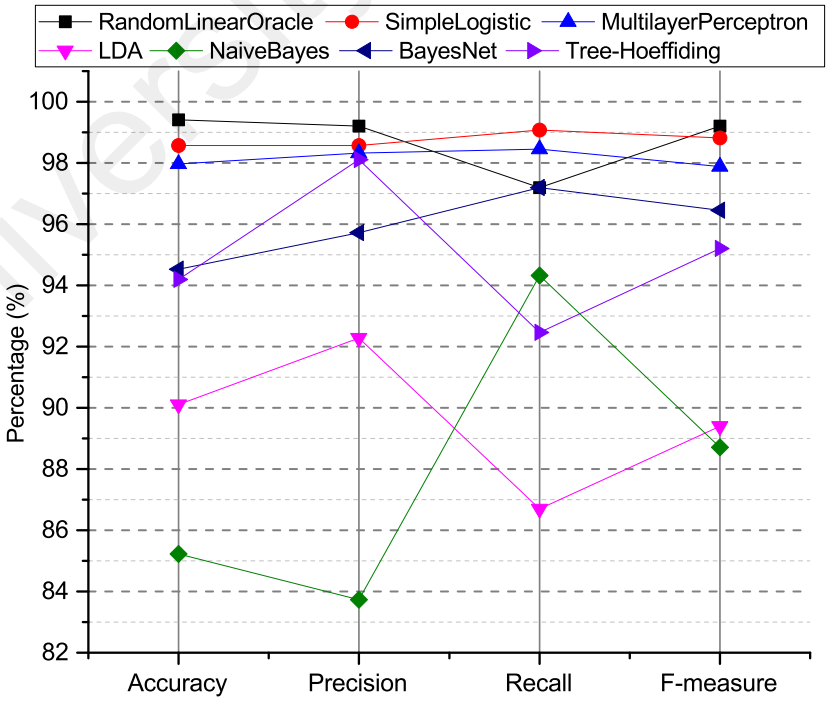

Figure 6.20: Overall Accuracy Performance Comparison of the Base Classifier Intrusion Detection Mechanisms

6.3 Results of Computation and Communication Cost

Since the controller represents the centralized core and network intelligence of the SDNs, the communication cost and computation overhead is a critical issue and must be taken into consideration. The newly designed and implemented software packet processing time in the controller may bring negative impact as long as the design is not carefully handled to cater for the computation and communication overhead. Long delays of processing packets may generate new attack surface areas in the control plane consequently leading to a single-point failure. In the following section, we presents the computation and communication cost.

6.3.1 Results of Computation Cost

The computation cost of our proposed solution mainly comes from three aspects: (1) the training process; (2) the testing process; and (3) the model update process. Since in our proposed architecture, we have considered the training in an offline fashion in order to have a careful SDN control plane design architecture of the proposed solution. Moreover, we have evaluated the overhead by employing 10% of data as the training data, and 10% of data as the testing data while the remaining data is divided into 8 update data-sets. The training data-set is used to generate the model and the iterative updates is done using the updating datasets. The computation cost of both the processes (i.e., partial model update and the complete model rebuild) is shown in Figure 6.21.

The Figure 6.21 clearly shows that the model generation time is linearly dependent on the number of data in the data-set (i.e., a linear function with respect to the total amount of data in the data-set). The simulation results shows that the complete model rebuilding process is very expensive. However, the partial model update time is dependent on the number of data in the new observations. The partial model update is much cheaper than the complete model rebuilding process.

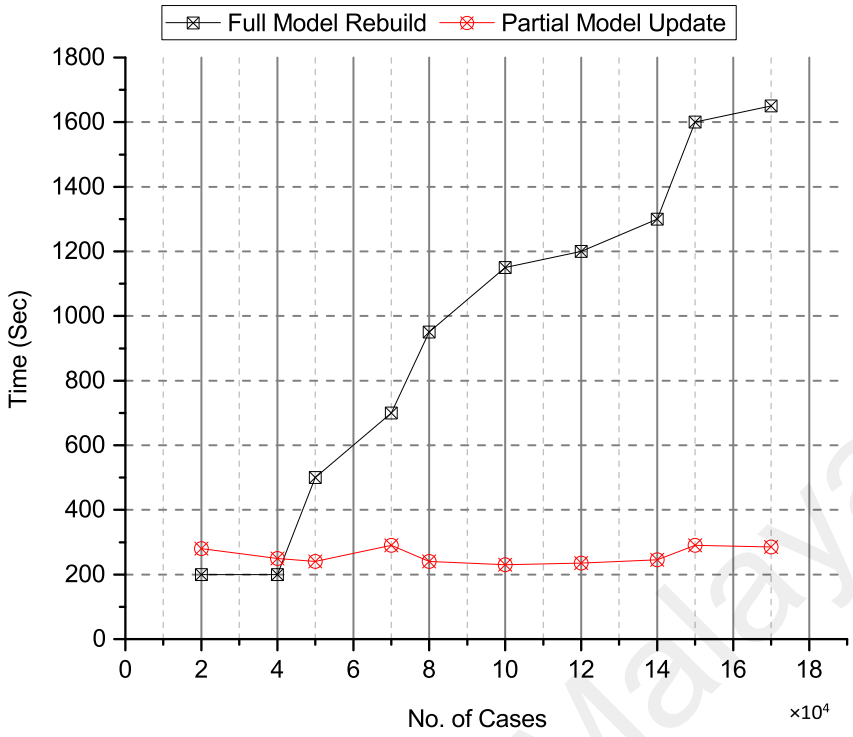

Figure 6.21: Computation Cost of the Proposed Method

6.3.2 Results of Communication Cost

We have conducted several experiments in order to evaluate the communication overhead. We have carried out the experiments in two different scenarios as shown in Figures 6.22, 6.23, 6.24, and 6.25 respectively. In the first scenario, we measured the end-user latency by observing RTTs for ping packets between two hosts separated by 3 hops of our own Lab. However, the second scenario is based on observing the RTTs of ping packets from a remote server located in a different place and the server running the SDN controller of our own Lab. We also measured the bandwidth of the network by using iperf 3 times an hour for 48h consecutively. We measured the average bandwidth as 83.5 MB/s for the first scenario and 29.6 MB/s for the second scenario receptively. In the first scenario, the connection is better compared to the second scenario because of the closely located set-up. Finally, we also test and observe the response time but having no significant effect. Consequently, the communication overhead is only related to RTT (round trip time).

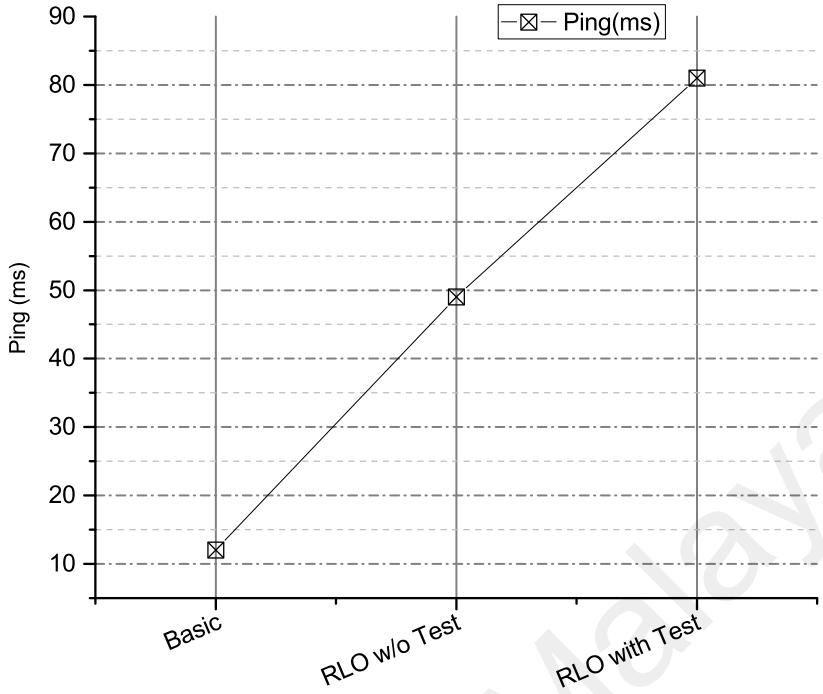

Figure 6.22: Comparison of Ping Latencies in the First Given Scenario

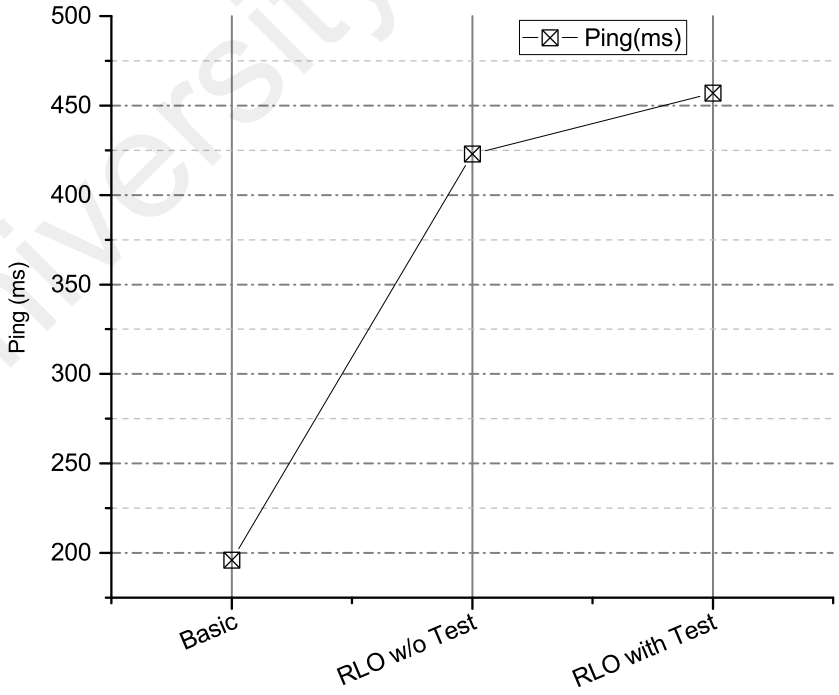

Figure 6.23: Comparison of Ping Latencies in the Second Given Scenario

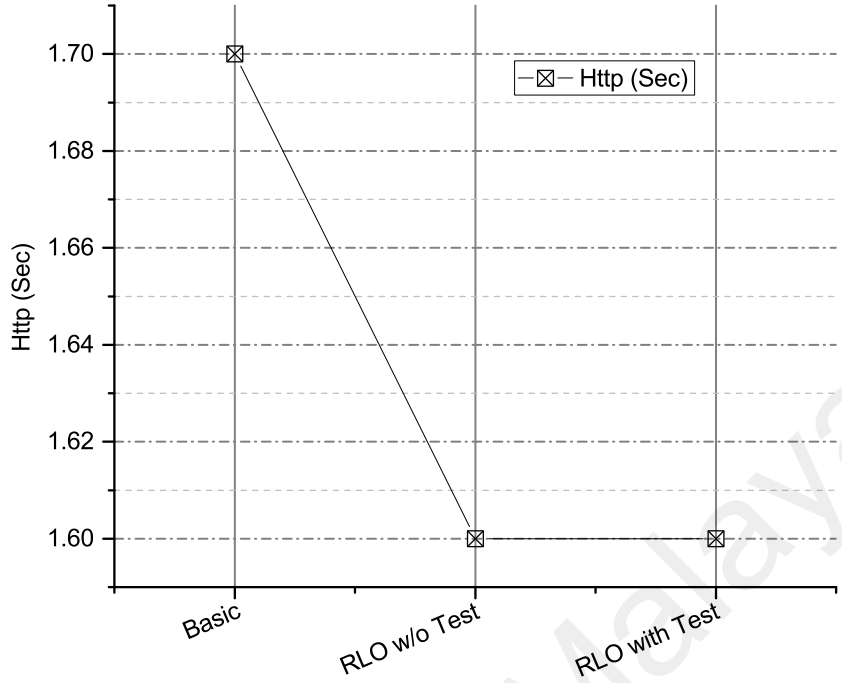

Figure 6.24: Comparison of the Response Time in the First Given Scenario

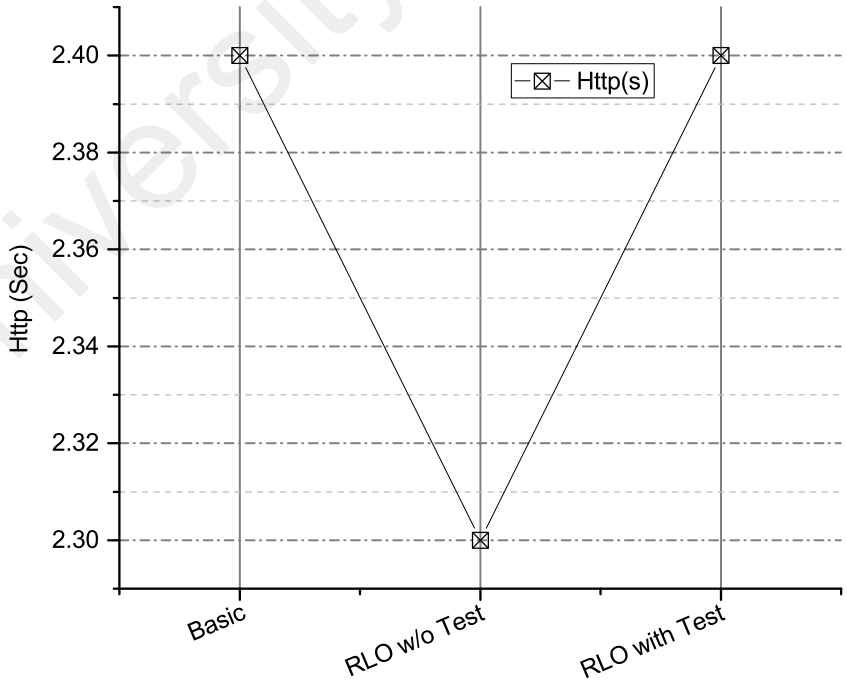

Figure 6.25: Comparison of the Response Time in the Second Given Scenario

6.3.3 End Users Latency with Varying Hosts

We compute the communication overhead of the proposed solution that end-users have experience, by observing RTTs for ping packets between two hosts separated by 6 hops. The Floodlight was modified to install rules with 1 sec idle timeout and Cbench was used to know the effect of the varying number of hosts for the observed ping latencies. We just plot the scenarios with 1 and 1K hosts to achieve more clarity. The overall results as shown in Figure 6.26 and Figure 6.27 clearly shows that the latency increases with the gradual increase in number of hosts. However; the latency overhead even with 1K hosts is not surprisingly high. The latency with 7K is just 200 . We attribute the overall reduced latency for both the cases with and without RLO to SDN Floodlight controller that throttles messages at high throughput.

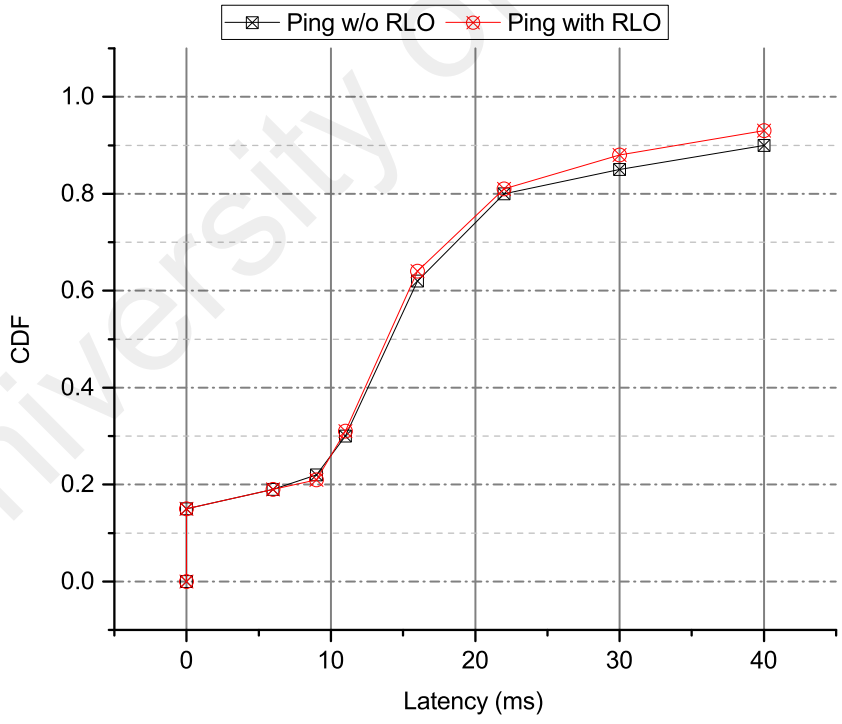

Figure 6.26: End-user Latencies with Varying Hosts

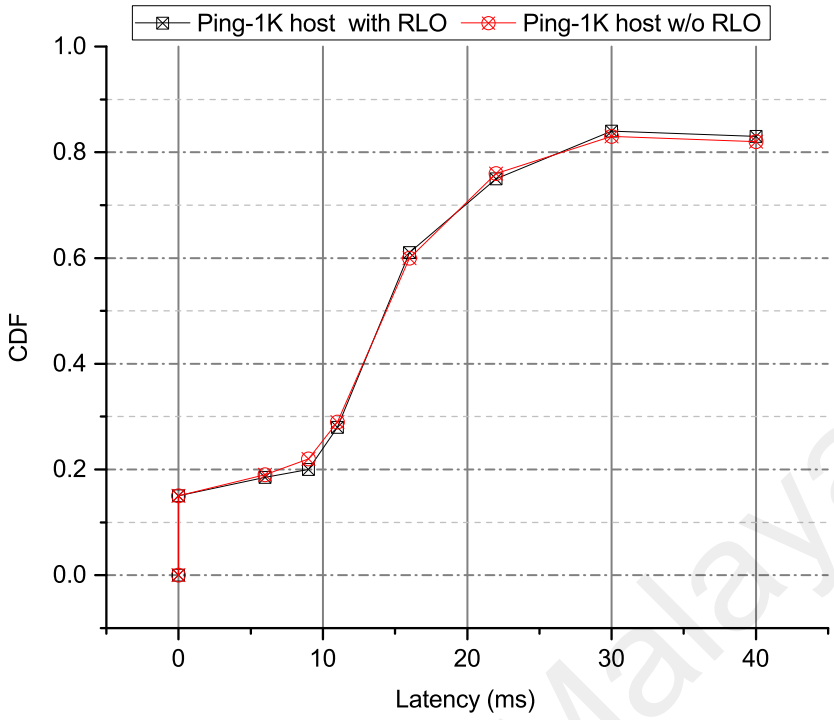

Figure 6.27: End-user Latencies with Varying 1K Hosts

6.4 Conclusion

The results of the experimental research (i.e., simulation, and emulation) of the proposed intrusion-detection method with their corresponding counterparts are presented. We clearly demonstrated the out-performance of the proposed method compared to the state-of-the-art classifier ensembles (i.e, multiple classifier based network anomaly detection systems) on the standard evaluation parameters. Moreover, we also verified the outstanding performance of the proposed random oracle based intrusion detection method compared to state-of-the-art base classifiers (i.e, single classifier based network anomaly detection systems). Finally, we presented the communication and computation cost to clearly show that the proposed method is viable to identify diverse sophisticated network attacks in the control-plane of the SDNs in real-time.

CHAPTER 7: CONCLUSION

This chapter concludes the thesis to put-forward the summary and accomplishments of the study. It summarizes the main findings of study, the attained research objectives, and highlights the significance of the proposed method. The chapter also provides the possible future extensions, limitations, and and scope of this work. Finally, in an effort to anticipate secure and dependable SDNs, the chapter presents the ongoing open security issues, the state-of-the-art security trends and cutting-edge future research directions.

The remainder of this chapter is organized as follows. Section 7.1 discusses the reassessment of the main findings and research objectives. Section 7.2 highlights contribution of the research work. Section 7.3 examines the scope and limitation of the research work. Section 7.4 provides the future plans and possible extensions of this work together with state-of-the-art on-going open security issues, challenges, and cutting-edge future research directions of the SDNs.

7.1 Reappraisal of the Main Findings and Research Objectives

The revolutionary idea of Software Defined Networks (SDNs) potentially provides to redefine the future of next-generation networks. Indeed, all the hype surrounding the SDNs is predominantly because of its centralized control, the separation of the control plane from the data forwarding plane, flow abstraction and enabling innovation through network programmability. Despite the promising architecture of SDNs, security was not considered as part of the initial design. Moreover, security concerns are potentially augmented considering the logical centralization of network intelligence.

The motivation of this dissertation is to address the defense space against the threat of attacks in SDNs due to compromised as well legitimate end hosts or programmable soft switches that wrest either full or partial control of the entire network. Additionally, this problem exacerbates in the context of SDNs for various crucial reasons. We aim

to effectively identify the threat of various distributed coordinated attacks in SDNs that primarily target the control plane by implementing a highly dynamic and robust intrusion-detection method that adds extra-diversity while promote a remarkable detection accuracy with incredible low false-positive rates.

The first objective was mainly based to conduct a thorough review of the security of SDNs. We studied the security implications of the entire SDN architecture with extant state-of-the-art security solutions in SDN considering the earliest to the latest trends. We reviewed and analyzed the security vulnerabilities, attacks, and challenges of the promising SDN architecture. We devised a contemporary layered/interface taxonomy of the reported security vulnerabilities, attacks, and challenges of the SDN to illustrate the main categories of security implications that pertain to each SDN layer/interface. We also highlighted and analyzed the possible threats that may affect and target a particular layer/interface alongside a suggested compact solution to help design secure SDNs. The extant state-of-the-art security solutions are also critically analyzed to devise a comprehensive thematic taxonomy. Moreover, we analyzed each state-of-the-art security solution to identify the distinguishing SDN features utilized for each security mechanism, and the exact problem addressed by a particular technique together with the simulation or emulation environment of the corresponding technique. The distinguishing features of SDN represent the potential of emerging SDNs. Additionally, we also identified the potential effect of each state-of-the-art security solution on the corresponding SDN layers/interface. The critical discussion on the extant state-of-the-art security solutions extend the domain knowledge of the current security trends in the SDNs, the major strengths of potential SDNs, and the research gaps that need thorough investigations. We evidently noticed that security is still the key concern and is an equally striking challenge that reduces the growth of SDNs. Moreover, the deployment of novel entities and the introduction of several architectural components of SDNs poses new security threats and vulnerabilities.

We identified that investigating the defense space against the threat of attacks in SDNs that wrest either full or partial control of the entire network is of paramount concern and increasingly important research topic.

The second objective was to closely analyze and establish the problem. A detailed problem analysis using Floodlight, a popular java based SDN controller was carried-out. Various possible ways of sophisticated network security attacks to target the control plane of the SDN architecture followed by the overall major impact of the attacks on SDNs were demonstrated. A complete formal analysis using Z language rules of the salient features that help identify diverse network attack patterns in the control plane of the SDNs to establish the problem was also carried-out. A few important OpenFLow (OF) messages such as Packet-in, Packet-Out, Flow-Mod and Flow-Removed were utilized to demonstrate that how our proposed method works identifying various network attack patterns in the real setting of SDNs. Moreover, to present a more clear analysis, we provide modeling of the proposed method using High Level Petri Nets (HLPN). We also formally verified the correct functioning of identifying varied network attack patterns.

The third objective was to propose a dynamic and robust intrusion detection method capable of effectively identifying varied sophisticated network attacks and large-scale malicious event automatically and in real-time that target the controller of the SDNs. A diverse fusion-selection approach that stands on Oracle to be applied to the classifier ensemble design, where the Oracle is a random linear function was proposed, implemented and tested. We argued that the proposed method adds extra-diversity while promoting a higher level of intrusion-detection accuracy, and that is achieved through the promising performance shown by the proposed method in chapter 6. Moreover, We also argued that our proposed method is capable of real-time intrusion detection in the SDN environment and that is also demonstrated.

The fourth objective was to perform a rigorous evaluation of the proposed method.

We performed a rigorous evaluation of the proposed method by testing using Floodlight, a popular java based SDN controller and Mininet to emulate SDN setting. We modeled the proposed method in the real setting of SDNs using High Level Petri Nets (HLPN), analyze the rules with Z language, and formally verified the correct functioning of our proposed method using Z3 SMT solver. We validated our proposed approach using a bench-marked publicly available data-set broadly known as NSL-KDD. The verification of the proposed approach is made with benchmark algorithms. Moreover, to show the resulting significant performance of the proposed approach to be optimistically unbiased, we employed a ten-fold cross-validation.

7.2 Contributions of The Research

The key contributions of this dissertation to the body of knowledge are as follows. Moreover, the contributions in terms of scholarly articles are separately stated in Appendix A.

The first contribution of this dissertation is a short tutorial paper on securing software defined networks: taxonomy, requirements, and open issues, published in a special issue on security and privacy in emerging networks of the IEEE Communications Magazine online journal. To the best of our knowledge, this work was the first short tutorial paper on research efforts made in this direction.

The second contribution of this dissertation is a comprehensive survey on secure and dependable software defined networks, published in the Journal of Network and Computer Applications. To the best of our knowledge, this work is the first to advocate an approach to achieve secure and dependable SDNs while comprehensively surveying, analyzing, and classifying the security vulnerabilities, attacks, and challenges of each SDN layer/interface together with state-of-the-art security mechanisms of SDN. Moreover, this paper presents a thematic layered/interface taxonomy of SDN security vulnerabilities, at-

tacks, and challenges. Although two survey papers were presented in the past, these works lack comprehensive thematic classifications and focus more on utilizing SDNs to secure different networks. The paper also complements our tutorial paper.

The third contribution of this dissertation is addressing diverse Man-at-The-End Attacks in SDNs as our future plan, which also results a full survey paper on Man-At-The-End attacks: analysis, taxonomy, human aspects, motivation and future directions published in the Journal of Network and Computer Applications. This is the first comprehensive literature review available till date on Man-at-The-End Attacks.

The fourth contribution of this dissertation is a detailed problem analysis in the real setting of Floodlight, a popular java based SDN controller. We demonstrated various possible ways of sophisticated network security attacks to target the control plane of the SDN architecture followed by the overall major impact of the attacks on SDNs. We also carried out a formal analysis using Z language rules of the salient features that help identify diverse network attack patterns in the control plane of the SDNs. We utilized few important OpenFlow (OF) messages such as Packet-in, Packet-Out, Flow-Mod and Flow-Removed to demonstrate that how our proposed intrusion detection method work identifying various network attack patterns in the real setting of SDNs. Moreover, to present a more clear analysis, we provide modeling of the proposed method using High Level Petri Nets (HLPN). We also formally verified the correct functioning of identifying varied network attack patterns. The outcomes of the problem analysis also results a journal paper. My recent work on formal verification just recently published in IEEE Transaction on Information Forensics Security.

The fifth contribution of this dissertation is the implementation of a diverse fusion-selection method that stands on Oracle. The proposed method adds extra-diversity while promoting a higher level of intrusion-detection accuracy. Moreover, the approach is highly dynamic, flexible and is capable to effectively detect a wide variety of sophisti-

cated network security attacks.

Finally, the sixth contribution of this dissertation is the rigorous evaluation of the proposed method by testing in the environment of Floodlight, a popular java based SDN controller with Mininet to emulate the SDN setting, to model the proposed method in the real setting of SDNs using High Level Petri Nets (HLPN), analyze the rules with Z language, and formally verified the correct functioning of our proposed method using Z3 SMT solver. Moreover, to validate our proposed approach, a benchmark data-set broadly known as NSL-KDD is used. The verification of the proposed approach is made with benchmark algorithms and to show the resulting significant performance of the proposed approach to be optimistically unbiased, a ten-fold cross-validation is employed. The outcomes of the verification with the current state-of-the-art also results two journal papers.

7.3 Research Scope and Limitations

The scope of this study is restricted to three main parts: (1) Analyzing the problem in the real setting of Floodlight, a popular java based SDN controller and modeling and formal analysis using High Level Petri Nets (HLPN) and Z language, (2) proposing a dynamic and robust intrusion-detection method that stands on Oracle, where the)Oracle is a random linear function. The method promotes a higher level of intrusion-detection accuracy with incredible low false positive rates. Moreover, the approach is highly dynamic, flexible and is capable to effectively detect a wide variety of sophisticated security attacks that primarily target the control plane of the SDN architecture, and (3) rigorous evaluation and validation of the proposed method to be applicable in the real-setting of SDNs (control-plane). The limitations of the are listed below.

- The method is highly dynamic, and works well with incredible performance in the SDNs, it also incurs limited communication and computation overhead on the SDN controller.

- The proposed method is likely to identify abnormal network attacks without prior or having specific knowledge but does not guarantee to accurately discover Zero-day attacks.

7.4 Future Works

The future works of this study mainly comprises of two sections. First, we elaborate the future plans and possible extension of our research work. Subsequently, we present on-going open security issues, the state-of-the-art security trends, and cutting-edge future research directions that might facilitate the wide acceptance of SDNs.

7.4.1 Possible Extensions of This Work

The future plans and possible extensions of this work are as follows

- The proposed research can be successfully extended to identify anomalies and diverse threat of attacks that mainly target the data-plane of the SDNs.
- We also plane to extend the proposed method for a hybrid model of Cloud Computing and Software Defined Networks to protect valuable computing infrastructure.
- Finally, we plan to extend this dynamic approach to ensure the security and privacy of the emerging networks and in diverse applicable areas of network security attacks identification.

7.4.2 SDNs Open Security Issues, Challenges and Future Research Directions

Security plays a vital role in deploying SDNs across different applicable networks. SDNs are receiving attention because of their diverse applicability. However, security is one of the key obstacles that hinder the growth and overall adoption of SDNs. This section elaborates open security issues, challenges, and foreseeable directions that might facilitate the wide acceptance of SDNs. Figure 7.1 presents the future security challenges, trends,

and direction of SDNs.

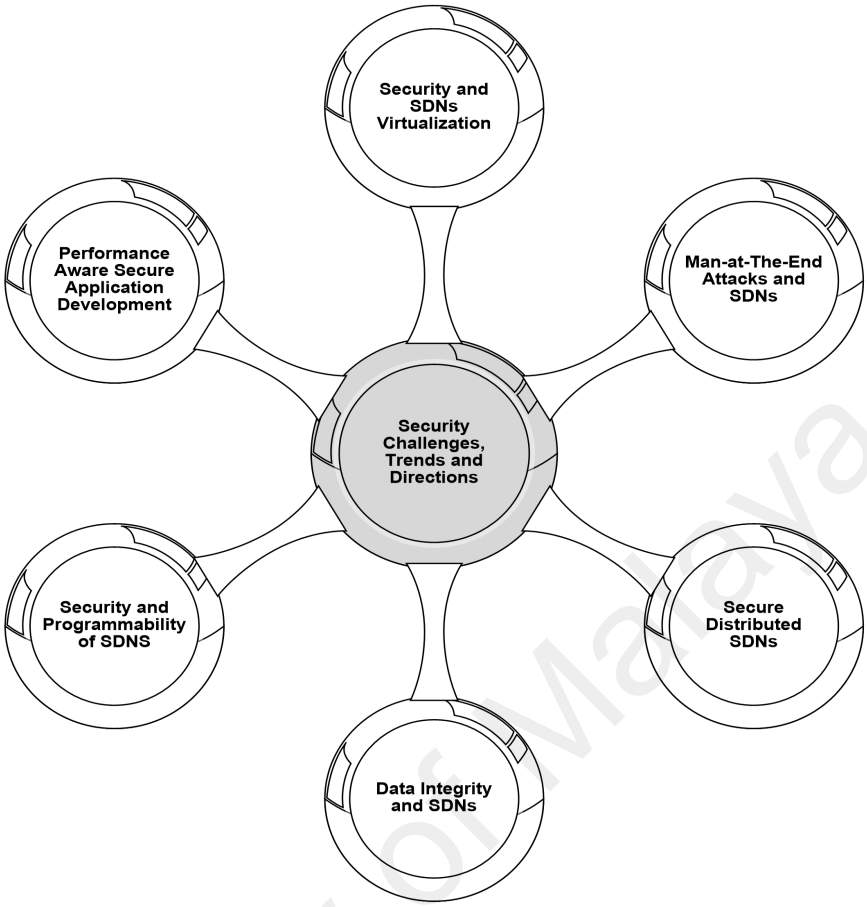

Figure 7.1: The SDNs Security Challenges, Trends and Directions

7.4.2.1 Security and SDNs Virtualization

The use of SDNs in network virtualization has introduced many security issues that need to be addressed. Several unaddressed issues related to SDNs deployed for network virtualization are presented. These issues were identified when the OF protocol was used for network virtualization. FlowVisor, a special-purpose controller, is used to ensure isolation between multiple virtual networks. In addition, it acts as a transparent proxy between a controller and a switch. FlowVisor configures the entire network into different slices. Moreover, it facilitates rewriting control messages according to user-defined policies to guarantee isolation. Furthermore, FlowVisor acts as a slicer and a controller in an SDN environment. Subsequently, it becomes a potential target for attackers because the entire

network is compromised once it is down. The issue of confidentiality, integrity, and availability to ensure network security must be addressed to achieve secure and dependable SDNs. FlowVisor does not implement action isolation; consequently, a controller can set any type of action on a flow entry without any control of the FlowVisor.

Victor proposed three possible threats: a) VLAN ID access problem: This problem occurs when a controller is denied access to any VLAN ID, whereas FlowVisor is allowed to create flow entries whose action can change the VLAN ID of a packet. This situation creates an opportunity for a nasty controller to inject packets into another slice or can be utilized to steal packets; b) Field rewrite problem: This problem occurs when a controller is given access to a particular VLAN ID tag to create a flow entry whose action can change the VLAN ID of its own packet, thereby creating an opportunity for a malicious controller to inject packets into another slice or to steal packets. Essentially, in a virtual environment, the header fields involved in creating a slice can experience the same problem. It can be a modification of any header field, such as IP/MAC source/destination address and transport source/destination numbers; c) Wildcard rewrite problem: This problem occurs when a controller is given access to a transport source port A, and the controller wants to create a flow entry with an unspecified transport source port (wildcard). FlowVisor should rewrite the valid transport source (A) to the wild-card value; however, in reality, this rewriting valid transport source does not occur, and it simply rewrites the wildcard value with any matching transport source port and may raise serious security concerns. The problem is the same for other fields, such as protocol type and transport destination. Malicious injections may occur by simply following the existing field rewrite problem, which allows the end user to change the VLAN ID tag in particular circumstances. Table 7.1 summarizes the security problems of the SDN virtual environment.

Table 7.1: Security Problems of SDNs Virtual Environment

SDNs Security Problems in Virtual Environment	Malicious Injection	Stealing Packets	Denial of Service (DoS)	Spoofing Attack	Status
VLAN ID Access Problem	✓		✓		Unexplored
Field Rewrite Problem	✓				Unexplored
Wildcard Rewrite Problem		✓			Unexplored
Implementation of Action Isolation					Unexplored
Denial of Service (DoS)			✓		Unexplored
Spoofing Attack				✓	Unexplored

7.4.2.2 SDN Controller-Specific Security Issues in Virtual Environments

The controller remains a potential target for attackers and is the most likely target of the first line of attack. The following are several unaddressed scenarios when the controller is targeted in a virtual environment. Figure 7.2 illustrates the particular scenario of using OpenVirteX, a special controller used to create virtual networks. When using OpenVirteX, several controllers, such as POX controller, are placed on the end user side, as shown in Fig. 5. Although POX has many advantages, it is also exploitable. Using OpenVirteX instead of FlowVisor addresses space isolation. However, it does not implement action isolation, that is, a controller can set any type of action on a flow entry without any control by OpenVirteX. Figure 7.2 shows the following attacks:

1. Denial of Service (DoS) Attack: A POX controller is placed on the end user side to create a particular virtual network using OpenVirteX. POX controller possesses critical knowledge of the network and is prone to many attacks, particularly DoS attacks. An attacker can generate a large number of flows to bring the network down or make it work improperly by targeting OpenVirteX.

2. Spoofing Attack: The spoofing attack can be illustrated considering the same scenario discussed above. For example, the floodlight controller is already aware of the IP address of OpenVirteX and can merely forge the IP address of OpenVirteX

to launch a simple and easy spoofing attack.

3. Malicious injection: Malicious injections may be performed by using the existing field rewrite problem, which allows the end user to change the VLAN ID tag in particular circumstances. Malicious injection creates an opportunity for a nasty controller to inject packets into another slice. OpenVirteX does not implement action isolation, that is, a controller can set any type of action in the flow entry without any control of the controller in this particular case.

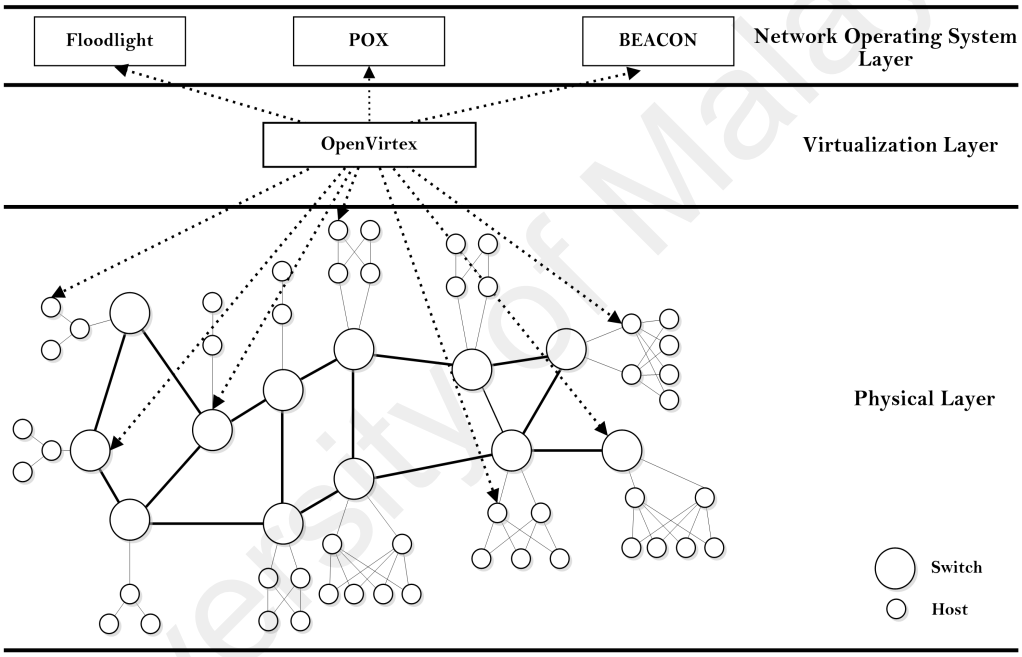

Figure 7.2: Controller-specific Attack Scenarios of Creating Virtual Environment using OpenVirtex

7.4.2.3 Man-at-The-End Attacks and SDNs

Any type of SDN, regardless of its architecture, design, configuration, and maintenance, relies on people. Traditional computer and network security for home networks, the Internet, the cloud, SDNs, and the Internet of Things are inadequate to address MATE attacks. MATE attacks are fundamentally difficult to resolve under general circumstances. The problematic part is that humans have become the edge, and auditing the human mind is

a complex task. Moreover, we cannot install antivirus on a system administrator's cerebral cortex. We still rely on perimeter defense even though the reality is that in digital world boundaries are non-existent. We continue to imagine that we have control over devices when we do not. We expect users to adhere to policies; however, we have no control over policy enforcement. SDNs may be targeted by MATE attacks in many ways. MATE attackers can primarily target the centralized control management architecture of SDNs, which is a single point of failure that may lead to the breakdown of a network. Furthermore, MATE attackers may penetrate and bypass the programmable aspects of SDNs, as these attackers are highly skilled and can compromise any SDN agent for further exploitation. A detailed description of possible MATE attack scenarios is beyond the scope of this study. MATE attacks need to be addressed to ensure secure and dependable SDNs. Figure 7.3 depicts a situation in which a MATE attacker has full access to his or her capabilities to enable him or her to bypass SDN protection mechanisms.

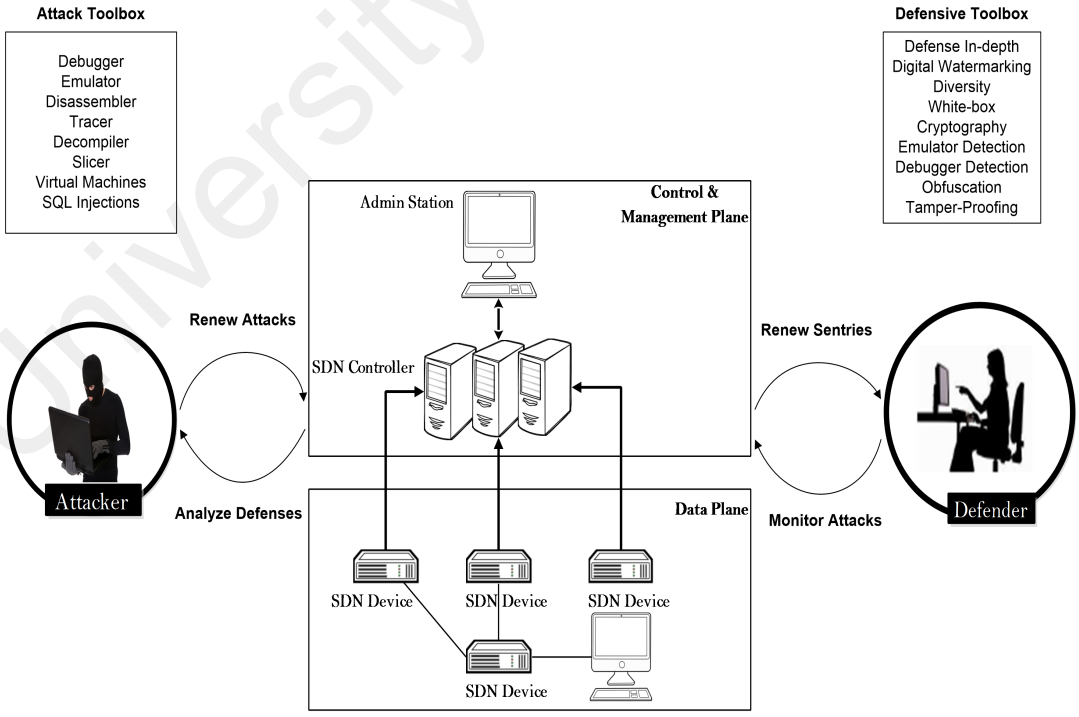

Figure 7.3: Man-at-The-End Attacker Having Full Access to Analyze and Utilize his or her Capabilities to Bypass the SDNs Protections

7.4.2.4 Performance Aware Secure Applications Development

The SDN centralized architecture does not bear long latencies unlike traditional networks. The northbound API of the SDN facilitates the development of a set of desired security applications at the application layer of the SDN. These security applications need to access packet-level information at different levels to be effective, particularly for digital forensics, intrusion detection, and prevention. Moreover, accessing the payload is obligatory where DPI is required. Subsequently, obtaining the required information has considerable latency, which causes the entire traffic to behave abnormally. Furthermore, in a special case of the first packet with unknown flow and having no buffer availability in the switch, the current OF version may send the entire packet to the SDN controller. Thus, a security application that manipulates deep packet inspection cannot benefit from the current version of OF in this particular case, thereby degrading the overall performance. Some studies were conducted to address this issue. Serious effort should be made to develop performance-aware security applications. Studies should also focus on developing security applications with good trade-offs among security, performance, and usability.

7.4.2.5 Secure and Dependable SDNs

SDN security and dependability are open issues. Several studies on security have focused on enhancing the security of different networks by using SDNs. SDNs have a promising architecture, yet they might pose potential security risks and possible threats to a network. Literature on SDNs indicates that security and dependability have not yet been achieved. Several potential threat vectors and vulnerabilities during the deployment of SDNs using OF were investigated. Unlike traditional network security attacks, SDNs introduce new threat vectors, particularly those related to the SDN controller and the southbound and the northbound communication interfaces. These three threat vectors, which are identified in, are predominantly related to SDNs, whereas other threat vectors are common. However,

these threat vectors may have a more devastating and augmented effect on SDNs than on traditional networks.

7.4.2.6 Programmability and SDNs

The SDN has to cope with the potential set of complex problems more related to the programmability aspects of the network. The tendency of launching sophisticated DDoS, phishing, spam and malware attacks is expected to increase massively. Subsequently, it will change the dynamics surrounding the infrastructures of secure SDNs. Moreover, if we see SDNs in the context of mobile wireless networks, then the possibility of injection and eavesdropping (active, passive) is much higher because of broadcast and extant vulnerabilities of the wireless channels. Furthermore, the case of mobile ad-hoc networks where typical security solutions are entirely infeasible to implement owing to their lack of infrastructure (e.g., security servers) becomes increasingly serious and complex. Classical security solutions necessitate downtime to orchestrate topological changes during reconfiguration, and configuration, when security services are debugged and turned on.

7.4.2.7 Data Integrity and SDNs

Security was not considered as part of the initial SDN design; thus, the certificate format to ensure data integrity is not well described by Open-Flow(OF) specifications. Sophisticated authentication and encryption mechanisms are needed to recover from packet failure and to prevent hackers from violating data integrity. However, the current OF specification recommends the use of TLS, which provides encrypted secure channels. Subsequently, eavesdropping is prevented. Yet TLS is unreliable in many cases. The OF specifications provide no details on the use of interoperable versions. Moreover, establishing a secure connection and checking the certificates between the switches or the controllers are not specified. For example, mutual authentication among multiple controllers is performed through an exchange of certificates signed by a private key of a third

party. The difference between these keys results in serious security vulnerabilities, and the controllers may not be able to inter-operate. To have more sophisticated cryptographic alternatives, this particular area needs to be well addressed.

7.4.2.8 Distributed SDNs and Security

SDNs introduce promising monitoring and management abilities in small to medium-sized network topologies. However, security compliance and policy enforcement, troubleshooting, debugging, and monitoring are problematic in distributed SDNs. Moreover, the management decisions of SDNs are based on the behavior and information of the network. Furthermore, the OF SDN session is subject to different actions specified in the flow table. Responding to a wide range of diverse events, such as intrusions, necessitates the enforcement of high-level policies by corresponding network operators. However, SDNs offer little to no automatic response mechanism for such events. Event-driven programming interfaces for service providers are needed to manage diverse asynchronous events linked with the corresponding network. The configuration of a carrier-grade network in particular remains a tedious task, given that administrators have to cope with vendor-specific and low-level interfaces to implement high-level functions. Consequently, low-level configurations remain a difficult task in distributed SDNs. These issues in relation to distributed SDNs need to be closely addressed.

REFERENCES

Al-Shaer, E., & Al-Haj, S. (2010). Flowchecker: Configuration analysis and verification of federated openflow infrastructures. In *Proceedings of the 3rd acm workshop on assurable and usable security configuration* (pp. 37–44).

Al-Shaer, E., Marrero, W., El-Atawy, A., & ElBadawi, K. (2009). Network configuration in a box: Towards end-to-end verification of network reachability and security. In *Network protocols, 2009. icnp 2009. 17th ieee international conference on* (pp. 123–132).

Antonatos, S., Akritidis, P., Markatos, E. P., & Anagnostakis, K. G. (2007). Defending against hitlist worms using network address space randomization. *Computer Networks*, *51*(12), 3471–3490.

Anwer, B., Benson, T., Feamster, N., Levin, D., & Rexford, J. (2013). A slick control plane for network middleboxes. In *Proceedings of the second acm sigcomm workshop on hot topics in software defined networking* (pp. 147–148).

Atighetchi, M., Pal, P., Webber, F., & Jones, C. (2003). Adaptive use of network-centric mechanisms in cyber-defense. In *Object-oriented real-time distributed computing, 2003. sixth ieee international symposium on* (pp. 183–192).

Baldini, G., Sturman, T., Biswas, A. R., Leschhorn, R., Gódor, G., & Street, M. (2012). Security aspects in software defined radio and cognitive radio networks: a survey and a way ahead. *Communications Surveys & Tutorials, IEEE*, *14*(2), 355–379.

Ballard, J. R., Rae, I., & Akella, A. (2008). Extensible and scalable network monitoring using opensafe. In *Inm/wren.*

Benton, K., Camp, L. J., & Small, C. (2013). Openflow vulnerability assessment. In *Proceedings of the second acm sigcomm workshop on hot topics in software defined networking* (pp. 151–152).

Blanchet, B. (2005). Proverif automatic cryptographic protocol verifier user manual. *CNRS, Departement dInformatique, Ecole Normale Superieure, Paris*.

Braga, R., Mota, E., & Passito, A. (2010). Lightweight ddos flooding attack detection using nox/openflow. In *Local computer networks (lcn), 2010 ieee 35th conference on* (pp. 408–415).

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Britto, A. S., Sabourin, R., & Oliveira, L. E. (2014). Dynamic selection of classifiers—a comprehensive review. *Pattern Recognition*, *47*(11), 3665–3680.

Burduk, R., & Walkowiak, K. (2015). Static classifier selection with interval weights of base classifiers. In *Intelligent information and database systems* (pp. 494–502). Springer.

Canini, M., Venzano, D., Perešíni, P., Kostić, D., & Rexford, J. (2012). A nice way to test openflow applications. In *Presented as part of the 9th usenix symposium on networked systems design and implementation (nsdi 12)* (pp. 127–140).

Casado, M., Freedman, M. J., Pettit, J., Luo, J., McKeown, N., & Shenker, S. (2007). Ethane: taking control of the enterprise. In *Acm sigcomm computer communication review* (Vol. 37, pp. 1–12).

Casado, M., Garfinkel, T., Akella, A., Freedman, M. J., Boneh, D., McKeown, N., & Shenker, S. (2006). Sane: A protection architecture for enterprise networks. In *Usenix security.*

Cbench. (2015). *cbench scalable cluster benchmarking.* Retrieved 2016-05-30, from `https://sourceforge.net/projects/cbench/`

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3), 15.

Chen, J., Huang, H., Tian, S., & Qu, Y. (2009). Feature selection for text classification with naïve bayes. *Expert Systems with Applications*, *36*(3), 5432–5435.

Chung, C.-J., Khatkar, P., Xing, T., Lee, J., & Huang, D. (2013). Nice: Network intrusion detection and countermeasure selection in virtual network systems. *Dependable and Secure Computing, IEEE Transactions on*, *10*(4), 198–211.

Cok, D. R., Stump, A., & Weber, T. (2015). The 2013 evaluation of smt-comp and smt-lib. *Journal of Automated Reasoning*, *55*(1), 61–90.

Costa, V. T., & Costa, L. (2013). Vulnerability study of flowvisor-based virtualized network environments. In *2nd workshop network virtualization intelligence future internet, rio de janeiro, brazil.*

Dhawan, M., Poddar, R., Mahajan, K., & Mann, V. (2015). Sphinx: Detecting security attacks in software-defined networks. In *Ndss.*

Ding, A. Y., Crowcroft, J., Tarkoma, S., & Flinck, H. (2014). Software defined networking for security enhancement in wireless mobile networks. *Computer Networks*, *66*, 94–101.

Dover, J. M. (2013). *A denial of service attack against the open floodlight sdn controller.* Dover Networks LCC.

El-Atawy, A., Samak, T., Wali, Z., Al-Shaer, E., Lin, F., Pham, C., & Li, S. (2007). An automated framework for validating firewall policy enforcement. In *null* (pp. 151–160).

Farhady, H., Lee, H., & Nakao, A. (2015). Software-defined networking: A survey. *Computer Networks*, *81*, 79–95.

Fayazbakhsh, S. K., Sekar, V., Yu, M., & Mogul, J. C. (2013). Flowtags: Enforcing network-wide policies in the presence of dynamic middlebox actions. In *Proceedings*

of the second acm sigcomm workshop on hot topics in software defined networking (pp. 19–24).

Floodlight. (2014). *SDN Controller java based controller.* Retrieved 2016-05-30, from http://www.projectfloodlight.org/floodlight/

Garnaev, A., & Trappe, W. (2013). To eavesdrop or jam, that is the question. In *Ad hoc networks* (pp. 146–161). Springer.

Goodney, A., Narayan, S., Bhandwalkar, V., & Cho, Y. H. (2010). Pattern based packet filtering using netfpga in deter infrastructure. In *1st asia netfpga developers workshop, daejeon, korea.*

Govindarajan, K., Meng, K. C., & Ong, H. (2013). A literature review on software-defined networking (sdn) research topics, challenges and solutions. In *Advanced computing (icoac), 2013 fifth international conference on* (pp. 293–299).

Handigol, N., Heller, B., Jeyakumar, V., Maziéres, D., & McKeown, N. (2012). Where is the debugger for my software-defined network? In *Proceedings of the first workshop on hot topics in software defined networks* (pp. 55–60).

Hartman, S., Wasserman, M., & Zhang, D. (2013). Security requirements in the software defined networking model. *Internet Engineering Task Force, Internet-Draft draft-hartman-sdnsec-requirements-01*.

Hinrichs, T., Gude, N., Casado, M., Mitchell, J., & Shenker, S. (2008). Expressing and enforcing flow-based network security policies. *University of Chicago, Tech. Rep*.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, *20*(8), 832–844.

Hong, S., Xu, L., Wang, H., & Gu, G. (2015). Poisoning network visibility in software-defined networks: New attacks and countermeasures. In *Ndss.*

Hu, F., Hao, Q., & Bao, K. (2014). A survey on software-defined network and openflow: from concept to implementation. *Communications Surveys & Tutorials, IEEE*, *16*(4), 2181–2206.

Hu, H., Han, W., Ahn, G.-J., & Zhao, Z. (2014). Flowguard: building robust firewalls for software-defined networks. In *Proceedings of the third workshop on hot topics in software defined networking* (pp. 97–102).

Huici, F., Di Pietro, A., Trammell, B., Gomez Hidalgo, J. M., Martinez Ruiz, D., & d'Heureuse, N. (2012). Blockmon: A high-performance composable network traffic measurement system. *ACM SIGCOMM Computer Communication Review*, *42*(4), 79–80.

IDC. (2016). *IDC members.* Retrieved 2016-05-30, from https://www.idc.com/getdoc.jsp?containerId=prUS41005016

Izenman, A. J. (2013). Linear discriminant analysis. In *Modern multivariate statistical*

techniques (pp. 237–280). Springer.

Jafarian, J. H., Al-Shaer, E., & Duan, Q. (2012). Openflow random host mutation: transparent moving target defense using software defined networking. In *Proceedings of the first workshop on hot topics in software defined networks* (pp. 127–132).

Jammal, M., Singh, T., Shami, A., Asal, R., & Li, Y. (2014). Software defined networking: State of the art and research challenges. *Computer Networks*, *72*, 74–98.

Jarraya, Y., Madi, T., & Debbabi, M. (2014). A survey and a layered taxonomy of software-defined networking. *Communications Surveys & Tutorials, IEEE*, *16*(4), 1955–1980.

Ji, S.-Y., Jeong, B.-K., Choi, S., & Jeong, D. H. (2016). A multi-level intrusion detection method for abnormal network behaviors. *Journal of Network and Computer Applications*, *62*, 9–17.

Jiang, D., Yang, Y., & Xia, M. (2009). Research on intrusion detection based on an improved som neural network. In *Information assurance and security, 2009. ias'09. fifth international conference on* (Vol. 1, pp. 400–403).

Jose, L., Yu, M., & Rexford, J. (2011). Online measurement of large traffic aggregates on commodity switches. In *Hot-ice.*

Joseph, D. A., Tavakoli, A., & Stoica, I. (2008). A policy-aware switching layer for data centers. *ACM SIGCOMM Computer Communication Review*, *38*(4), 51–62.

Kazemian, P., Chang, M., Zeng, H., Varghese, G., McKeown, N., & Whyte, S. (2013). Real time network policy checking using header space analysis. In *Presented as part of the 10th usenix symposium on networked systems design and implementation (nsdi 13)* (pp. 99–111).

Kewley, D., Fink, R., Lowry, J., & Dean, M. (2001). Dynamic approaches to thwart adversary intelligence gathering. In *Darpa information survivability conference & exposition ii, 2001. discex'01. proceedings* (Vol. 1, pp. 176–185).

Khurshid, A., Zou, X., Zhou, W., Caesar, M., & Godfrey, P. B. (2013). Veriflow: Verifying network-wide invariants in real time. In *Presented as part of the 10th usenix symposium on networked systems design and implementation (nsdi 13)* (pp. 15–27).

Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., ... others (2010). Onix: A distributed control platform for large-scale production networks. In *Osdi* (Vol. 10, pp. 1–6).

Kothari, N., Mahajan, R., Millstein, T., Govindan, R., & Musuvathi, M. (2011). Finding protocol manipulation attacks. In *Acm sigcomm computer communication review* (Vol. 41, pp. 26–37).

Kreutz, D., Ramos, F., & Verissimo, P. (2013). Towards secure and dependable software-defined networks. In *Proceedings of the second acm sigcomm workshop on hot topics in software defined networking* (pp. 55–60).

Kreutz, D., Ramos, F. M., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., & Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14–76.

Kuncheva, L. I. (2002). Switching between selection and fusion in combining classifiers: an experiment. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *32*(2), 146–156.

Kuncheva, L. I., & Rodriguez, J. J. (2007). Classifier ensembles with a random linear oracle. *Knowledge and Data Engineering, IEEE Transactions on*, *19*(4), 500–508.

Kwiatkowska, M., Norman, G., & Parker, D. (2011). Prism 4.0: Verification of probabilistic real-time systems. In *Computer aided verification* (pp. 585–591).

Landwehr, N., Hall, M., & Frank, E. (2005). Logistic model trees. *Machine Learning*, *59*(1-2), 161–205.

Lantz, B., Heller, B., & McKeown, N. (2010). A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th acm sigcomm workshop on hot topics in networks* (p. 19).

Lara, A., Kolasani, A., & Ramamurthy, B. (2014). Network innovation using openflow: A survey. *Communications Surveys & Tutorials, IEEE*, *16*(1), 493–512.

Li, C., Raghunathan, A., & Jha, N. K. (2009). An architecture for secure software defined radio. In *Proceedings of the conference on design, automation and test in europe* (pp. 448–453).

Liu, A. X. (2008). Formal verification of firewall policies. In *Communications, 2008. icc'08. ieee international conference on* (pp. 1494–1498).

Liu, X., Xue, H., Feng, X., & Dai, Y. (2011). Design of the multi-level security network switch system which restricts covert channel. In *Communication software and networks (iccsn), 2011 ieee 3rd international conference on* (pp. 233–237).

Liyanage, M., & Gurtov, A. (2012). Secured vpn models for lte backhaul networks. In *Vehicular technology conference (vtc fall), 2012 ieee* (pp. 1–5).

Matias, J., Garay, J., Mendiola, A., Toledo, N., & Jacob, E. (2014). Flownac: Flow-based network access control. In *Software defined networks (ewsdn), 2014 third european workshop on* (pp. 79–84).

Matsumoto, S., Hitz, S., & Perrig, A. (2014). Fleet: defending sdns from malicious administrators. In *Proceedings of the third workshop on hot topics in software defined networking* (pp. 103–108).

Mehdi, S. A., Khalid, J., & Khayam, S. A. (2011). Revisiting traffic anomaly detection using software defined networking. In *Recent advances in intrusion detection* (pp. 161–180).

Metzler, J. (2012). Understanding software-defined networks. *Reports. information. com*,

9.

Min, L., & Dongliang, W. (2009). Anormaly intrusion detection based on som. In *Information engineering, 2009. icie'09. wase international conference on* (Vol. 1, pp. 40–43).

mininet. (2015). *mininet virtual network creator.* Retrieved 2016-05-30, from `http://mininet.org/`

Mitrokotsa, A., & Douligeris, C. (2005). Detecting denial of service attacks using emergent self-organizing maps. In *Signal processing and information technology, 2005. proceedings of the fifth ieee international symposium on* (pp. 375–380).

Mogul, J. C., Tourrilhes, J., Yalagandula, P., Sharma, P., Curtis, A. R., & Banerjee, S. (2010). Devoflow: Cost-effective flow management for high performance enterprise networks. In *Proceedings of the 9th acm sigcomm workshop on hot topics in networks* (p. 1).

Monsanto, C., Reich, J., Foster, N., Rexford, J., & Walker, D. (2013). Composing software defined networks. In *Presented as part of the 10th usenix symposium on networked systems design and implementation (nsdi 13)* (pp. 1–13).

Moshref, M., Yu, M., & Govindan, R. (2013). Resource/accuracy tradeoffs in software-defined measurement. In *Proceedings of the second acm sigcomm workshop on hot topics in software defined networking* (pp. 73–78).

Murphy, K., et al. (2001). The bayes net toolbox for matlab. *Computing science and statistics*, *33*(2), 1024–1034.

Nadeau, T., & Pan, P. (2011). Software driven networks problem statement. *draft-nadeau-sdn-problem-statement-01*.

Naous, J., Stutsman, R., Mazières, D., McKeown, N., & Zeldovich, N. (2009). Delegating network security with more information. In *Proceedings of the 1st acm workshop on research on enterprise networking* (pp. 19–26).

Nayak, A. K., Reimers, A., Feamster, N., & Clark, R. (2009). Resonance: dynamic access control for enterprise networks. In *Proceedings of the 1st acm workshop on research on enterprise networking* (pp. 11–18).

Ng, E. (2010). *Maestro: A system for scalable openflow control* (Tech. Rep.). TSEN Maestro-Technical Report TR10-08, Rice University.

NSL-KDD. (2014). *NSL-KDD benchmark dataset.* Retrieved 2016-05-30, from `http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD-dataset.html`

Nunes, B. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. *Communications Surveys & Tutorials, IEEE*, *16*(3), 1617–1634.

OpenDaylight. (2015). *SDN Controller controller.* Retrieved 2016-05-30, from `https://`

www.opendaylight.org/

Oza, N. C., & Tumer, K. (2008). Classifier ensembles: Select real-world applications. *Information Fusion*, *9*(1), 4–20.

Pfahringer, B., Holmes, G., & Kirkby, R. (2007). New options for hoeffding trees. In *Australasian joint conference on artificial intelligence* (pp. 90–99).

Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., & Gu, G. (2012). A security enforcement kernel for openflow networks. In *Proceedings of the first workshop on hot topics in software defined networks* (pp. 121–126).

Qazi, Z. A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., & Yu, M. (2013). Simple-fying middlebox policy enforcement using sdn. In *Acm sigcomm computer communication review* (Vol. 43, pp. 27–38).

Ramadas, M., Ostermann, S., & Tjaden, B. (2003). Detecting anomalous network traffic with self-organizing maps. In *Recent advances in intrusion detection* (pp. 36–54).

Roli, F. (2015). Multiple classifier systems. *Encyclopedia of Biometrics*, 1142–1147.

Ruck, D. W., Rogers, S. K., Kabrisky, M., Oxley, M. E., & Suter, B. W. (1990). The multilayer perceptron as an approximation to a bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, *1*(4), 296–298.

Sapio, A., Baldi, M., Liao, Y., Ranjan, G., Risso, F., Tongaonkar, A., ... Nucci, A. (2014). Mapper: A mobile application personal policy enforcement router for enterprise networks. In *Software defined networks (ewsdn), 2014 third european workshop on* (pp. 131–132).

Saucez, D., Bonaventure, O., & Iannone, L. (2015). Lisp threats analysis.

Scapy. (2015). *Security Tool security power tool.* Retrieved 2016-05-30, from http://www.secdev.org/projects/scapy/

Schlesinger, C., Story, A., Gutz, S., Foster, N., & Walker, D. (2012). Splendid isolation: Language-based security for software-defined networks. In *Proc. of workshop on hot topics in software defined networking.*

Scott-Hayward, S., O'Callaghan, G., & Sezer, S. (2013). Sdn security: A survey. In *Future networks and services (sdn4fns), 2013 ieee sdn for* (pp. 1–7).

SDxCentral. (2016). *SDxCentral members.* Retrieved 2016-05-30, from https://www.sdxcentral.com/reports/sdn-market-size-infographic-2013/

Sekar, V., Egi, N., Ratnasamy, S., Reiter, M. K., & Shi, G. (2012). Design and implementation of a consolidated middlebox architecture. In *Presented as part of the 9th usenix symposium on networked systems design and implementation (nsdi 12)* (pp. 323–336).

Sezer, S., Scott-Hayward, S., Chouhan, P.-K., Fraser, B., Lake, D., Finnegan, J., ... Rao,

N. (2013). Are we ready for sdn? implementation challenges for software-defined networks. *Communications Magazine, IEEE*, *51*(7), 36–43.

Sharma, S., Staessens, D., Colle, D., Pickavet, M., & Demeester, P. (2011). Enabling fast failure recovery in openflow networks. In *Design of reliable communication networks (drcn), 2011 8th international workshop on the* (pp. 164–171).

Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., & Parulkar, G. (2009). Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 1–13.

Shin, S., & Gu, G. (2012). Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In *Network protocols (icnp), 2012 20th ieee international conference on* (pp. 1–6).

Shin, S., Porras, P. A., Yegneswaran, V., Fong, M. W., Gu, G., & Tyson, M. (2013). Fresco: Modular composable security services for software-defined networks. In *Ndss.*

Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., ... Kang, B. B. (2014). Rosemary: A robust, secure, and high-performance network operating system. In *Proceedings of the 2014 acm sigsac conference on computer and communications security* (pp. 78–89).

Shin, S., Yegneswaran, V., Porras, P., & Gu, G. (2013). Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 2013 acm sigsac conference on computer & communications security* (pp. 413–424).

Shirali-Shahreza, S., & Ganjali, Y. (2013). Flexam: Flexible sampling extension for monitoring and security applications in openflow. In *Proceedings of the second acm sigcomm workshop on hot topics in software defined networking* (pp. 167–168).

Skowyra, R., Bahargam, S., & Bestavros, A. (2013). Software-defined ids for securing embedded mobile devices. In *High performance extreme computing conference (hpec), 2013 ieee* (pp. 1–7).

Skowyra, R. W., Lapets, A., Bestavros, A., & Kfoury, A. (2013). Verifiably-safe software-defined networks for cps. In *Proceedings of the 2nd acm international conference on high confidence networked systems* (pp. 101–110).

Son, S., Shin, S., Yegneswaran, V., Porras, P., & Gu, G. (2013). Model checking invariant security properties in openflow. In *Communications (icc), 2013 ieee international conference on* (pp. 1974–1979).

Sookhak, M., Gani, A., Talebian, H., Akhunzada, A., Khan, S. U., Buyya, R., & Zomaya, A. Y. (2015). Remote data auditing in cloud computing environments: a survey, taxonomy, and open issues. *ACM Computing Surveys (CSUR)*, *47*(4), 65.

Staessens, D., Sharma, S., Colle, D., Pickavet, M., & Demeester, P. (2011). Software

defined networking: Meeting carrier grade requirements. In *Local & metropolitan area networks (lanman), 2011 18th ieee workshop on* (pp. 1–6).

Suh, J., Choi, H.-g., Yoon, W., You, T., Kwon, T., & Choi, Y. (2010). Implementation of a content-oriented networking architecture (cona): a focus on ddos countermeasure. In *Proceedings of european netfpga developers workshop.*

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2012). Nsl-kdd dataset. *http://www. iscx. ca/NSL-KDD*.

Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., & Sherwood, R. (2012). On controller performance in software-defined networks. In *Presented as part of the 2nd usenix workshop on hot topics in management of internet, cloud, and enterprise networks and services.*

Visentini, I., Snidaro, L., & Foresti, G. L. (2016). Diversity-aware classifier ensemble selection via f-score. *Information Fusion*, *28*, 24–43.

Vissicchio, S., Vanbever, L., & Bonaventure, O. (2014). Opportunities and research challenges of hybrid software defined networks. *ACM SIGCOMM Computer Communication Review*, *44*(2), 70–75.

Wang, B., Zheng, Y., Lou, W., & Hou, Y. T. (2015). Ddos attack protection in the era of cloud computing and software-defined networking. *Computer Networks*, *81*, 308–319.

Wang, J., Wang, Y., Hu, H., Sun, Q., Shi, H., & Zeng, L. (2013). Towards a security-enhanced firewall application for openflow networks. In *Cyberspace safety and security* (pp. 92–103). Springer.

Wang, K., Qi, Y., Yang, B., Xue, Y., & Li, J. (2012). Livesec: Towards effective security management in large-scale production networks. In *Distributed computing systems workshops (icdcsw), 2012 32nd international conference on* (pp. 451–460).

Wang, Y., Zhang, Y., Singh, V., Lumezanu, C., & Jiang, G. (2013). Netfuse: Short-circuiting traffic surges in the cloud. In *Communications (icc), 2013 ieee international conference on* (pp. 3514–3518).

Wen, X., Chen, Y., Hu, C., Shi, C., & Wang, Y. (2013). Towards a secure controller platform for openflow applications. In *Proceedings of the second acm sigcomm workshop on hot topics in software defined networking* (pp. 171–172).

Xie, G. G., Zhan, J., Maltz, D. A., Zhang, H., Greenberg, A., Hjalmtysson, G., & Rexford, J. (2005). On static reachability analysis of ip networks. In *Infocom 2005. 24th annual joint conference of the ieee computer and communications societies. proceedings ieee* (Vol. 3, pp. 2170–2183).

Xie, H., Tsou, T., Lopez, D., Yin, H., & Gurbani, V. (2012). Use cases for alto with software defined networks. *Working Draft, IETF Secretariat, Internet-Draft draft-xie-alto-sdn-extension-use-cases-01. txt*.

Xing, T., Huang, D., Xu, L., Chung, C.-J., & Khatkar, P. (2013). Snortflow: A openflow-based intrusion prevention system in cloud environment. In *Research and educational experiment workshop (gree), 2013 second geni* (pp. 89–92).

Yang, M., Li, Y., Jin, D., Zeng, L., Wu, X., & Vasilakos, A. V. (2015). Software-defined and virtualized future mobile and wireless networks: A survey. *Mobile Networks and Applications*, *20*(1), 4–18.

Yao, G., Bi, J., & Xiao, P. (2011). Source address validation solution with openflow/nox architecture. In *Network protocols (icnp), 2011 19th ieee international conference on* (pp. 7–12).

Yu, M., Jose, L., & Miao, R. (2013). Software defined traffic measurement with opensketch. In *Presented as part of the 10th usenix symposium on networked systems design and implementation (nsdi 13)* (pp. 29–42).

Zaalouk, A., Khondoker, R., Marx, R., & Bayarou, K. (2014). Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and sdn control functions. In *Network operations and management symposium (noms), 2014 ieee* (pp. 1–9).

Zhang, Y. (2013). An adaptive flow counting method for anomaly detection in sdn. In *Proceedings of the ninth acm conference on emerging networking experiments and technologies* (pp. 25–30).

APPENDIX A: LIST OF PUBLICATIONS

Some of my publications are as follows.

List of Published Q1 Journal Publications

- **Adnan Akhunzada**, et al. "Securing software defined networks: taxonomy, requirements, and open issues." IEEE Communications Magazine 53.4 (2015): 36-44. (ISI Indexed Q1, Impact Factor 4.7, ERA Ranking A*)
- **Adnan Akhunzada**, et al. "Secure and dependable software defined networks." Journal of Network and Computer Applications 61 (2016): 199-221. (ISI Indexed Q1, Impact Factor 2.22, ERA Ranking A)
- **Adnan Akhunzada**, et al. "Man-At-The-End attacks: Analysis, taxonomy, human aspects, motivation and future directions." Journal of Network and Computer Applications 48 (2015): 44-57. (ISI Indexed Q1, Impact Factor 2.22, ERA Ranking A)
- Quratul-ain, **Adnan Akhunzada**, et al. Formal Verification of the xDAuth Protocol, IEEE Transactions on Information Forensics & Security (ISI Indexed Q1, Impact Factor 2.5, ERA Ranking A*)
- Sookhak, Mehdi, **Adnan Akhunzada**, et al. "Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues." ACM Computing Surveys (CSUR) 47.4 (2015): 65. (ISI Indexed Q1, Impact Factor 5.24, ERA Ranking A*)
- Sookhak, Mehdi,**Adnan Akhunzada**, et al. "Geographic wormhole detection in wireless sensor networks." (2015): e0115324. (ISI Indexed Q1, Impact Factor 3.37, ERA Ranking A)
- Sookhak, Mehdi, **Adnan Akhunzada**, et al. "Towards dynamic remote data auditing in computational clouds." 2014 (2014). (ISI Indexed Q1, Impact Factor 1.7, ERA Ranking A)

List of Journal Papers Accepted for Publication

- Anam, Adnan Akhunzada, et al. "A High-Level Domain-specific Language for SIEM: Design, Development and Formal Verification", Accepted for publication in Cluster Computing (CLUS). (ISI Indexed Q1, ERA Ranking A)
- Asif Ihsan, Adnan Akhunzada, et al." Optimizing SIEM Throughput on the Cloud Using Parallelization"', Accepted for publication (ISI Indexed Q1, ERA Ranking A)

- Ahmed, Adnan Akhunzada, et al. "Survey on Network Virtualization Using OpenFlow: Taxonomy, Opportunities, and Open Issues.", accepted for publication in Transaction on Information Systems. (ISI Indexed Q3, ERA Ranking A)

List of Journal Papers Under-Process

- **Adnan Akhunzada**, et al. "A Random Oracle Based Intrusion Detection Method". (ISI Indexed Q1, ERA Ranking A*)
- **Adnan Akhunzada**, et al. "A Dynamic and Robust Intrusion Detection Method in Software Defined Networks: Formal Specification & Verification ",
- **Adnan Akhunzada**, et al. "Information Fusion Using Random Oracles", (ISI Indexed Q1, ERA Ranking A*)
- **Adnan Akhunzada**, et al. "Near-Miss Situation Analysis Of The SIEM Rules Through Security Visualizations", Transactions on Visualization and Computer Graphics. (Under-Review, ISI Indexed Q1, ERA Ranking A*)
- **Adnan Akhunzada**, et al. "Formal Specification and Verification of Cross Tenant Access Control (CTAC) Model for Cloud", IEEE Transactions on Information Forensics & Security (ISI Indexed Q1, Impact Factor 2.5, ERA Ranking A*)
- **Adnan Akhunzada**, et al. "An Enhanced Shibboleth Protocol", Journal of Computer Standards & Interfaces (CSI). (Under-Review, ISI Indexed Q2, ERA Ranking A)

APPENDIX B: CONFUSION MATRIX

Table 1: List of 10-folds of Confusion Matrix (Training) for Random Oracle Based IDS

Confusion Matrix	10 Fold Cross Validation (Training RLO))				
Fold1	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111727	27	4	26	32
	17	5563	0	0	1
	43	0	3425	3	2
	249	1	1	3239	2
	55	4	23	1	9220
Fold2	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111749	26	4	13	24
	17	5562	0	0	1
	40	0	3426	4	3
	268	1	4	3217	2
	52	0	23	2	9226
Fold3	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111746	28	3	17	22
	14	5566	0	0	1
	43	0	3427	4	0
	300	0	2	3188	2
	64	1	28	2	9208
Fold4	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111730	28	5	26	27
	17	5564	0	0	0
	42	0	3428	0	3
	248	1	3	3236	4
	60	1	25	1	9216
Fold5	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111738	23	4	25	26
	16	5564	0	0	1
	40	0	3429	3	1
	285	1	3	3200	3
	72	1	29	1	9200
Fold6	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111746	24	4	18	24
	14	5567	0	0	0
	41	0	3428	2	2
	317	1	4	3166	4
	54	0	21	1	9228
Fold7	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111732	25	2	33	24
	16	5565	0	0	0
	47	0	3421	3	2
	272	0	2	3216	2
	67	0	24	2	9211
Fold8	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111750	24	2	18	22
	19	5561	0	0	1
	42	0	3423	7	1
	290	0	2	3198	2
	69	3	24	3	9204
Fold9	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111732	29	3	27	25
	14	5566	0	0	1
	36	0	3434	3	0
	277	0	3	3210	2
	47	0	26	2	9228
Fold10	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	111741	28	5	20	22
	14	5566	0	0	1
	47	0	3420	4	2
	274	1	4	3210	3
	57	0	27	1	9219

Table 2: List of 10-folds of Confusion Matrix (Testing) for Random Oracle Based IDS

Confusion Matrix	10 Fold Cross Validation (Testing RLO)				
Fold1	Normal	DoS	U2R	R2L	Probe
	12399	3	1	8	13
	3	617	0	0	0
	4	0	380	1	1
	43	0	1	341	3
	8	0	2	0	1024
Fold2	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12404	3	1	7	9
	5	616	0	0	0
	10	0	375	1	0
	45	0	0	342	1
	14	0	4	1	1015
Fold3	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12405	5	1	7	6
	3	617	0	0	0
	8	0	376	1	0
	41	0	0	347	0
	6	1	2	0	1025
Fold4	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12401	6	1	9	7
	0	620	0	0	0
	7	0	376	1	2
	36	0	0	352	0
	16	0	6	1	1011
Fold5	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12409	4	1	4	6
	5	615	0	0	0
	6	0	379	1	0
	43	0	2	343	0
	10	0	2	1	1021
Fold6	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12403	3	4	8	6
	7	613	0	0	0
	3	0	382	0	1
	45	0	0	343	0
	14	0	4	0	1015
Fold7	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12404	4	1	11	4
	2	617	0	0	1
	7	0	375	2	2
	33	1	1	353	0
	10	0	1	0	1022
Fold8	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12400	7	1	6	10
	1	619	0	0	0
	7	0	377	0	2
	46	0	0	342	0
	9	0	1	0	1024
Fold9	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12419	0	0	1	4
	5	615	0	0	0
	6	0	379	1	0
	39	0	1	348	0
	10	0	6	0	1018
Fold10	**Normal**	**DoS**	**U2R**	**R2L**	**Probe**
	12406	3	0	8	7
	4	615	0	0	1
	5	0	380	0	1
	38	0	0	350	0
	12	0	5	0	1016