AN INTRUSION DETECTION SYSTEM FOR SDN-BASED TACTICAL NETWORKS: A MACHINE LEARNING APPROACH

by

Skhumbuzo Goodwill Zwane

(201144122)

A dissertation submitted in fulfilment of the requirements for the degree of

Master of Science in Computer Science

Faculty of Science and Agriculture

Department of Computer Science

University of Zululand

KwaDlangezwa

RSA

Supervisor: Paul Tarwireyi

Co-supervisor: Prof Matthew Adigun

2020

Abstract

Network security is increasingly becoming a critical and continuous issue due to ongoing advancements in Information and Communication Technologies (ICT) and the concomitant rise in the number of security threats. This is especially true for military communication networks as security breaches may have detrimental effects. However, over the years, it has become increasingly difficult to attain high levels of detection accuracy in military tactical networks with conventional anomaly detection systems due to the dynamic nature of network traffic in the battlefield, special operations, and the harsh environment where they operate. Intrusion detection systems (IDS) have emerged as essential countermeasures to preserve network security. In addition, the introduction of software-defined networks (SDN) in tactical networks presents countless opportunities for security.

This study developed an IDS model for military tactical networks that utilizes Machine Learning (ML) techniques for high detection rates and SDN for network global view and centralised data collection. Following the Design Science methodology, the model was designed based on guidelines from related literature, and a proof-of-concept prototype of the model was implemented to assess its effectiveness. The experimental results indicated that Machine Learning using network flow data collected via SDN can improve intrusion detection rates in tactical networks. Among the machine learning techniques, ensemble learning methods utilising Decision Tree classification methods, namely Random Forest and Adaptive Boosting, obtained high recall and precision when detecting DDoS attacks, malicious, and misbehaving nodes in an SDN-enabled tactical network.

Declaration

I, Mr. Skhumbuzo Goodwill Zwane, hereby declare that the work presented in this dissertation is my own work and that this dissertation has not previously been submitted in full or partial fulfilment of requirements for an equivalent or higher qualification at any other recognised educational institution. All sources of information used in this work have been acknowledged.

Dedication

This dissertation is dedicated to my late parents

Acknowledgements

Primarily, I would like to thank God for being able to complete this research with success. Then, I would like to express my special thanks to my supervisors Mr. Paul Tarwireyi and Prof. Matthew. O. Adigun whose valuable guidance has helped me patch together this research and prove it successful. Their suggestions and instructions have served as the major contributor towards the completion of the research.

Then, I would like to thank my family and friends who have assisted me with their support and valuable advice which was very helpful in various phases of this work's journey.

Last but not least, I would like to extend my thanks to the staff and colleagues at the University, not in any order of importance, namely Dr. O. Oki, N. Sibeko, Dr. P. Mudali, S. Ndlovu, I. Adebayo, T. Akinola, and I. Mba for their various contributions and ideas which aided me in completing this study.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	**Stands For**
AAC	Authentication and Access Control
AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AP	Access Point
API	Application Programming Interface
ACCS	Australian Centre for Cyber Security
AUC	Area Under the Roc Curve
Bagging	Bootstrap Aggregation
CART	Classification and Regression Tree
CIA	Confidentiality, Integrity, and Availability
CLL	Conditional Log-Likelihood
COMSEC	Communication Security
CP	Control Plane
CPU	Central Processing Unit
CS	Computer Science
DARPA	Defense Advanced Research Project Agency
DDoS	Distributed Denial-of-Service
DE	Decision Engine
DL	Deep Learning
DM	Data Mining
DoD	Department of Defence
DoS	Denial-of-Service
DP	Data Plane
DS	Design Science
DT	Decision Tree
ELK	Elasticsearch, Logstash, and Kibana
ELM	Extreme Learning Machine
FIDS	Flow-based Intrusion Detection System

FPR	False-Positive Rate
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
ID	Intrusion Detection
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information Export
IS	Information System
ISR	Intelligence, Surveillance, and Reconnaissance
ISTAR	Intelligence, Surveillance, Target Acquisition, and Reconnaissance
KDD	Knowledge Discovery in Databases
LL	Log Likelihood
MAC	Medium Access Protocol
MANET	Mobile Ad Hoc Network
MAODV	Multicast Ad hoc On-Demand Distance Vector
ML	Machine Learning
MLP	Multi-Layer Perceptron
NB	Naïve Bayes
NCW	Network-Centric Warfare
NIDS	Network Intrusion Detection System
NIP	Network Infrastructure Protection
NN	Neural Network
NOS	Network Operating System
OHE	One Hot Encoder
ONF	Open Networking Foundation
OOB	Out-of-Bag
PCA	Principal Component Analysis
PHY	Physical Layer
QoS	Quality of Service
RCCTO	Rapid Capabilities and Critical Technologies Office
REST	Representational State Transfer

RF	Random Forest
ROC	Receiver Operating Characteristics
SDN	Software Defined Network
SFIDS	Software-defined Flow-based Intrusion Detection System
SLN	Semantic Link Network
SNARC	Stochastic Neural Analogy Reinforcement Computer
SMTP	Simple Mail Transfer Protocol
SMO	Sequential Minimal Optimization
Sta	Station
SVC	Support Vector Classifier
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TPR	True Positive Rate
TRANSEC	Transmission Security
TWN	Tactical Wireless Network
UAV	Unmanned Aerial Vehicles
UDP	User Datagram Protocol
UML	Unified Modelling Language
VM	Virtual Machine

List of publications

Zwane, S., Tarwireyi, P., & Adigun, M. (2018). Performance Evaluation of Machine Learning Classifiers for Intrusion Detection. *IEEE, International Conference on Intelligent and Innovative Computing Applications (ICONIC)*

Zwane, S., Tarwireyi, P., & Adigun, M. (2019). A Flow-based IDS for SDN-enabled Tactical Networks. *International Multidisciplinary Information Technology and Engineering Conference (IMITEC).*

Zwane, S., Tarwireyi, P., & Adigun, M. (2019). Ensemble learning approach for Flow based Intrusion Detection System. *IEEE AFRICON*.

Zwane, S., Tarwireyi, P., & Adigun, M. (2019). Ensemble Learning for Flow-Based Intrusion Detection in SDN. *Southern Africa Telecommunication Networks and Applications Conference (SATNAC).*

Chapter 1: Introduction

1.1. Introduction

Modern military communications are revolutionising the way war will be fought in the future by evolving towards a Network-Centric Warfare (NCW) paradigm (Wilson, 2004). In this paradigm, strength is achieved through communications networks and information sharing. This battle philosophy places the emphasis on the ability to achieve an internet-like capability in operational areas, providing ubiquitous network access to enable “anytime, anywhere” communication (Burbank *et al.*, 2006). Tactical networks are those deployed to support users and platforms within the tactical operational region, also known as the tactical environment.

Military tactical networks are divided into four important segments (Mishra et al, 2017). The first segment comprises the computing infrastructure residing at the military headquarters. Generally, the computing and communication resources at this segment are usually plentiful and tend to be static. The second segment consists of the infrastructure network that connects headquarters with the base environment used to support military operations and missions. It uses satellite communications and sometimes leverages available infrastructures such as cellular communication networks.

The third segment is the base environment that uses portable laptops, desktops computers, networking equipment, and storage devices. The base is usually set up and used for a temporary period, which sometimes ranges from a few days to a couple of months. It is used to provide logistical support to military personnel who may be deployed to operations covering a large area. Lastly, connecting to the base environment is the tactical segment.

The tactical segment consists of the portable devices and networks used by militaries’ personnel at the edge of the operations. It usually consists of various handhelds devices, unmanned aerial vehicles (UAVs), mobile networking devices, intelligence surveillance and reconnaissance (ISR) devices, and a computing environment to be carried onto different platforms including ships, vehicles, and tanks. An ad hoc network can be formed by these devices and platforms by themselves, which is called a tactical mobile ad hoc network (tactical MANET). However, this network is not completely ad hoc as it usually relies on a limited amount of infrastructural support, such as the other upper tactical environment segments for information (Mishra et al, 2017).

As opposed to commercial networks, tactical networks usually operate in extremely harsh environmental conditions such as desert, jungle, arctic, and maritime, where all those different environmental conditions may have different radio frequency propagation and signal characteristics that present a number of challenges. Figure 1.1. illustrates some of the difficulties imposed by the tactical environment on tactical communication technologies.

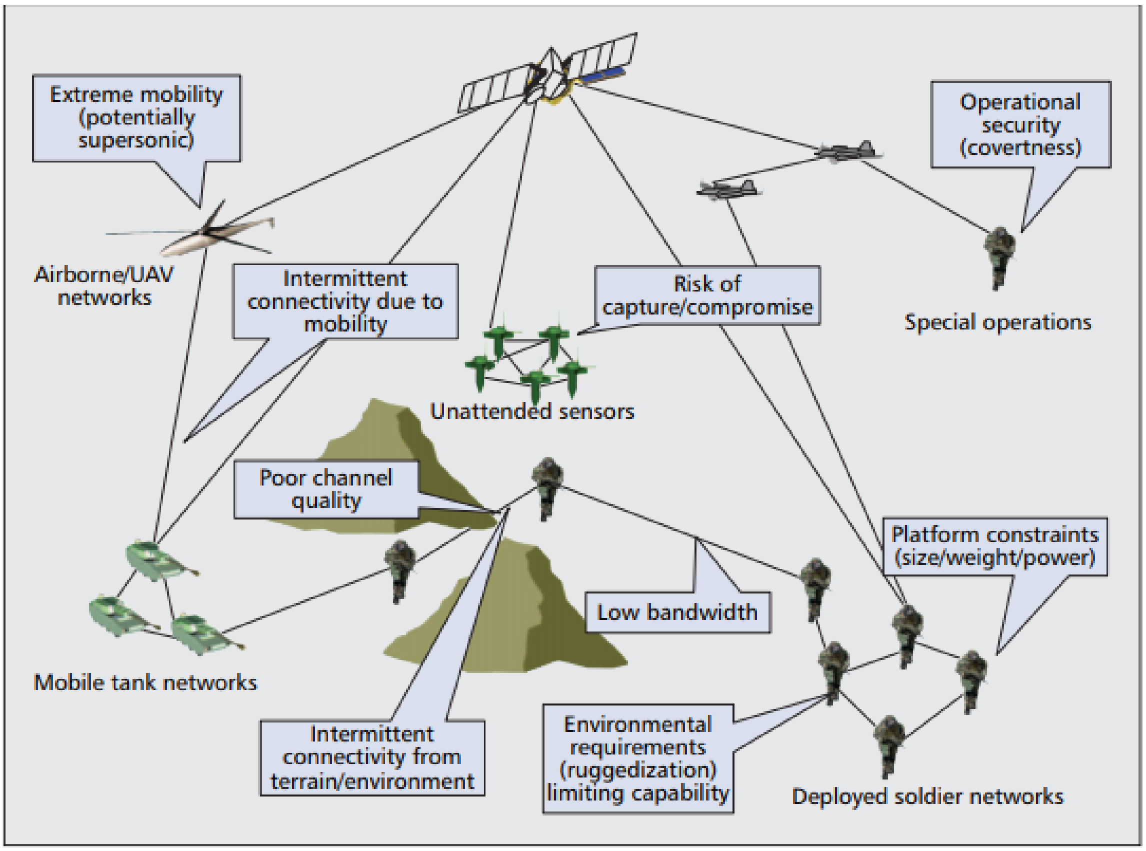

Extreme mobility (potentially supersonic)
Operational security (covertness)
Airborne/UAV networks
Intermittent connectivity due to mobility
Risk of capture/compromise
Special operations
Unattended sensors
Poor channel quality
Platform constraints (size/weight/power)
Low bandwidth
Mobile tank networks
Intermittent connectivity from terrain/environment
Environmental requirements (ruggedization) limiting capability
Deployed soldier networks

Figure 1.1 Limitations experienced in a military tactical environment (Burbank et al.*, 2006)*

The diversity of military tactical operations, the variety of equipment, the diverse speeds at which numerous parts of the tactical operation take place, the scale, and the environmental conditions all present difficult challenges to the full and smooth deployment and functioning of tactical networks (Burbank *et al.*, 2006). One of the major challenges to the tactical network is communication and information security. For example, in the tactical segment of the network, most of the devices are subject to capture by adversaries. Once captured, an adversary can compromise the captured device to infiltrate the network for malicious intent, such as lead troops into an ambush or gunfight with non-hostile forces. In addition, due to the wireless nature of the tactical network, it is possible for adversaries to listen to unprotected communications of their enemies or launch various kinds of cyber-attacks that can have detrimental effects, such as issuing denial-of-service attacks to shutdown nodes from communicating permanently or temporarily causing issues in the tactical network. As a result, security in tactical networks remains a big challenge for both Academia and Industry (Ken et al., 2017, Pawgasame et al., 2015, Little 2014).

In (Pawgasame and Wipusitwarakun, 2015) a survey discussing current issues and challenges in tactical wireless networks was presented. The authors argued that hostile environments where tactical wireless networks operate make it hard to protect information and data from being dropped, modified, or stolen by an intruder. For example, in a hostile environment network packets are regularly corrupted and dropped at nodes, and detection mechanisms might perceive such behaviour as a threat and raise a false-positive alert. They also argued that incorrect detection of hostile nodes in a hostile environment is one major research gap in the security of tactical wireless networks. They claimed that preventative methods such as cryptography are not sufficient for the protection of tactical networks, malicious nodes need to be detected early before any harm is done, hence they stated that a precise and reliable intrusion detection mechanism for the tactical wireless network is needed.

Also, Spencer *et al*. (2016) argued that there is no centralised management system by which networks and services can be globally configured and provisioned in many deployed military networks. This limits the flexibility of the network operator to make quick and urgent network security and management configurations. They also argued that tactical networks are deployed in different environments, hence, network movement and every modification to QoS requirement encompasses a time-consuming planning and configuration processes which degrades the speed of the network deployment, presenting challenges in terms of deploying, redeploying, and managing information and security services, limiting the swiftness and elasticity of a military force.

However, from those studies, it is evident that network management and security pose important technical challenges that need extensive innovation in order to realise an ideal tactical network solution (Burbank *et al.*, 2006). Thus, regarding that, this research focuses on analysing and addressing management and security challenges in tactical networks through the utilisation of new emerging technologies. While different security solutions have been proposed, such as authentication and cryptography, in (Atlam, Walters and Wills, 2018) the authors argued that cryptography sometimes failed to handle some attacks, for example, denial-of-service attacks. Atlam, Walters and Wills (2018) emphasised that it is critical to establish an Intrusion Detection System (IDS) that is capable of identifying and reacting to attacks efficiently in the network.

Intrusion Detection Systems (IDS) are needed in tactical networks to promptly and accurately recognise cyber-warfare attacks as soon as they are initiated and to respond to them before any harm is conducted. IDS are used to discover, determine, and identify illicit usage, access, and demolition of information systems. The different types of IDS techniques are; misuse-based, anomaly-based, and hybrid-based techniques. The misuse-based techniques, also known as signature-based techniques, are aimed to detect attacks using known signatures of those attacks. They are commonly known for their effectiveness in detecting known attack types without producing a large number of false alarms. Their drawback is that they are ineffective in detecting novel (zero-day) attacks, and they also require their database to be frequently updated with rules and signatures manually.

On the other hand, anomaly-based methods learn normal network behaviour and identify any deviations from the learned normal behaviour. Over the years they have been attractive since they have the ability to detect zero-day attacks. They are also attractive since the profiles of normal activities are customised for every application, network, or system, which makes it hard for attackers to know which activity they can conduct without being detected. The disadvantage of this method is a high false alarm rate, since new unseen legitimate system behaviours may be categorised as anomalies. The hybrid method combines both the misuse- and anomaly-based techniques. Hybrid methods are commonly used to improve detection rates of recognised intrusions and also reduce false-positive rates for unknown attacks.

IDS for tactical networks have a number of ideal goals and objectives they should meet. These include perfect accuracy and recognition totality (Little et al.., 2006). However, tactical networks present unique challenges to intrusion detection methods, since they are ad hoc, dynamic, and operate in harsh environments (Little et al.., 2006). As a result, the intrusion detection methods used usually fail to meet the ideal goal and objectives. How to reduce false alarms in tactical environments is one of the open research gaps that need to be addressed (Pawgasame et al, 2015). In addition, due to the lack of stable infrastructure which results in the absence of a centralised entity in the tactical segment (Spencer *et al.*, 2016), this limits the applicability of intrusion detection methods in these networks. Therefore, there is a need for an appropriate mechanism that can allow centralised control and a global view in a tactical network for successful IDS implementation.

More recently, Machine Learning (ML) techniques have shown promising results in intrusion detection (Ken et al, 2017, Haq et al, 2015) through high detection rates and efficiency. On the other hand, a newly emerged paradigm known as Software Defined Network (SDN) has shown great opportunities for the networking community. The SDN architecture is defined by the separation of the data plane from the control plane and consolidates the control plane functionality at a central location in the network (Mishra et al, 2017). The concept of SDN can offer a number of benefits to tactical military networks as it can allow improved traffic management abilities, meaning quick configurations for service delivery to support current operational priorities. SDN can also enable automation in military tactical networks, which will decrease preparation overheads as units and headquarters move. Lastly, tactical networks require skilled operators to support deployed military networks, SDN can reduce the burden on such operators, meaning fewer expert operators will be required to support deployed military networks (Spencer et al, 2017). It is, however, not clear what benefits SDN can provide for intrusion detection tasks in tactical networks.

Hence, this study presents an IDS that takes advantage of ML capability to learn from the past and SDN concepts to support service delivery in intrusion detection tasks to improve attack detection in mobile wireless networks.

1.2. Motivation

Tactical networks play a special role in networking, especially in the military battlefield where they support different kinds of devices in a very harsh environment. Network management and security are identified as the major concerns as the rate of network attacks has increased dramatically over the years and the tactics used by attackers continue to evolve. Given the low physical security in military tactical mobile nodes, a multi-level protection mechanism is required irrespective of the authentication used. Intrusion detection systems (IDS), acting as the second line of defence after authentication have the potential to improve the security of tactical networks by detecting different attacks and can help prevent other harmful attacks. Unfortunately, due to their dynamic nature, and lack of fixed infrastructure in tactical networks, IDS implementation in such networks is challenging and complex. Also, how to handle packet processing flows efficiently for huge amounts of data remains a research challenge. Therefore, there is a need to look at available options to implement IDS that address the current limitations of IDS in tactical networks.

Alternatively, with the promising emerging SDN technology, exploring the possibilities and opportunities this technology can provide in network security is important. Since network attacks increase daily, network security improvements need to be a major priority to catch up with these daily threats or attacks. SDN-based intrusion detection investigation will not only benefit tactical networks but would also benefit other kinds of network technologies. It is envisioned that this work will provide a Machine Learning IDS for SDN-based mobile tactical networks. This architecture will allow efficient data collection as well as detect and mitigate threats without any human intervention. It will also allow a mobile ad hoc network environment to be able to run IDS more efficiently and have less false detection rates since the controller will allow a global view of the network.

1.3. Problem Statement

Tactical networks are used in military operations so that airplanes, moving personnel, and tanks can communicate. Being that nodes communicate via wireless links, these wireless links among nodes are vulnerable to link attacks which comprise eavesdropping, leakage of secret information, active interference, data tampering, impersonation, denial-of-service (DoS), and message distortion. Additionally, in mobile ad hoc networks' routing protocols, nodes usually assume that other nodes would be reliable, trustworthy, and always cooperate to relay data (Kumar & Dutta, 2016). This assumption leaves vulnerabilities in the network because attackers can easily compromise the network by capturing and inserting malicious/non-cooperative nodes in the network. This is especially true for military communications, given the low physical security of mobile devices.

To achieve a secure network, detection measures such as intrusion detection systems (IDS) serving as a second line of defence in addition to traditional authentication measures have been investigated. To a certain extent, research work has previously been conducted in intrusion detection for traditional wired networks. However, due to key architectural differences, principal among them being the lack of fixed infrastructure and routing protocols used (Mishra et al, 2004), applying this research to wireless networks is not an easy plug-and-play task.

Thus, the lack of a single or centralised management system by which network services can be configured and provisioned limits the applicability of intrusion detection methods in tactical networks (Spencer et al, 2016). Other issues include false detection of nodes in tactical environments (Pawgasame et al, 2015), and performance and overhead concerns due to complex rules used to investigate network traffic (Alsmadi et al, 2016). Therefore, tactical MANETs require special IDS, designed specifically to benefit their needs and special characteristics, which include limited resources (Ken et al, 2017, Liu et al, 2008) before they can be safely deployed for industry use.

A technology paradigm that can provide a centralised and global view of the entire network, also allowing applications containing network rules to be applied without affecting the network functionalities is known as Software Defined Network (SDN). This work proposes an IDS that takes advantage of the network global view provided by SDN and the learning and predictive capabilities provided by Machine Learning (ML) to advance detection accuracy in Intrusion Detection Systems of tactical MANETs. This study evaluates readily available ML algorithms to select the most appropriate. These algorithms are then implemented for intrusion detection tasks in SDN-based tactical MANETs and evaluated.

1.4. Research Questions

This research aims to address the following research question:

- How can an Intrusion Detection System (IDS) that promptly and accurately recognises cyberwarfare attacks in tactical networks be designed and implemented?
    1. Which state-of-the-art techniques are the most suitable for intrusion detection in tactical networks?
    2. How can we design an effective IDS model for tactical networks?
    3. How can the designed IDS model be implemented and operationalised?
    4. What techniques and metrics can be used to evaluate this IDS model?

1.5. Goal and Objectives

1.5.1. Research Goal

The goal of this study is to investigate and implement the most suitable Machine Learning algorithm for intrusion detection in SDN-based tactical MANETs.

1.5.2. Research Objectives

The goal is broken down into the following achievable objectives:

1. To establish start-of-the-art network intrusion detection practices and the most suitable techniques.
2. To evaluate and compare machine learning algorithms using network intrusion datasets.
3. To design and implement an IDS model for tactical networks.
4. To evaluate the effectiveness of the proposed IDS.

1.6. Research Methodology

To achieve the goal and objectives of this study, the Design Science (DS) methodology was adopted and implemented as the main research. The DS method includes steps to create and evaluate IT artefacts designed to solve an identified organisational problem, and it involves a rigorous procedure to design an artefact to solve practical problems, make research contributions, evaluate designs, and communicate results to appropriate audiences (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). In this research, the goal was to develop an intrusion detection model for tactical MANETs. The IDS model was instantiated into a prototype and validated through implementation. Thus, the DS methodology was broken down into the DS research processes (see Figure 1.2) as outlined by (Peffers *et al.*, 2007), that is, (i) Problem analysis, (ii) Establishment of state-of-the-art, (iii) Solution design, (iv) and Solution evaluation. A brief overview of these phases is given in the following subsections: 1.6.1 to 1.6.4.

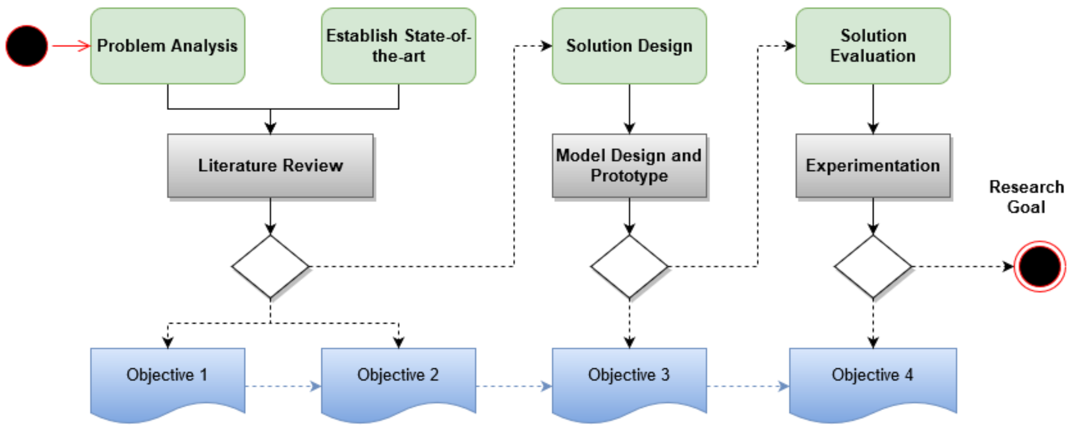


Figure 1.2 Overview of the research methodology

1.6.1. Problem Analysis

(Hevner *et al.*, 2004) stated that Design Science research must produce or develop a technology-based solution that addresses an important and relevant business problem. In military tactical scenarios, security is increasingly becoming an issue especially since more devices are being connected/networked. Therefore, the need to continuously monitor and tackle security issues is relevant. In that regard, this study addresses a relevant, real-world problem that can benefit organisations to achieve improved network security and incident-handling capabilities.

1.6.2. Establishment of state-of-the-art

To establish the state-of-the-art, this research used the literature review method. Recent conference papers and journals were reviewed to identify limitations and challenges in current network security approaches. The review also focused on the different security issues, attacks, and vulnerabilities experienced in tactical networks, challenges observed in the implementation of security mechanisms, and the performance of state-of-the-art network security techniques.

1.6.3. Solution Design

In this research, an IDS model for tactical networks was envisioned. By considering state-of-the-art computing paradigms, namely Software Defined Networks (SDN) and Machine Learning (ML), an IDS model was developed utilising SDN for network global view and data collection, while ML algorithms were used for effective classification of network data. The applicability of the model was demonstrated through the instantiation of a prototype based on the model. This solution intends to address network security issues and challenges that exist in military tactical networks.

1.6.4. Solution Evaluation

In this research, the proposed IDS (artefact) was deployed as a prototype to operationalise and evaluate it. Simulated network traffic was used to analyse and evaluate the proposed IDS to assess its quality and efficacy using experiments.

1.7. Research Contributions

In the recent era, the network structure itself is vulnerable to many different cyber-security issues. Therefore, network security can be considered a major concern. In that regard, this research explored how the existing attack methods can be detected and mitigated using recently proposed cybersecurity and networking trends.

Firstly, Software Defined Network (SDN) is noted as one of the few trends with a wide range of advantages to Tactical Networks. Thus, this work sets a foundation for further exploration into SDN as a security inhancer solution while also spurring future works and exploration of SDN potential in Tactical Netwoks. This study designed an IDS model that takes advantage of SDN for efficient flow sampling and preparation. This study demonstrated by designing a model for an intrusion detection system in SDN based tactical network, that SDN can be a useful asset to IDSs.

Further more, in the quest for effective network intrusion detection approaches, Machine Learning (ML) is also one of the fast growing fields and extensive research is being conducted. However, most studies conducted in the IDS domain only conduct a performance analysis of different Machine Learning algorithms using datasets without explicit demonstration of how they can then be used and implemented. This is a drawback as comparing ML methods using datasets obtained online doesn’t guarantee an effective IDS. Thus, this research followed the Design Science method to develop a protype of the proposed model. This method offers an important paradigm for conducting applicable and yet rigorous research. This approach allow the development, implementation and evaluation of a proposed systems, which can be adopted by other researchers in the field to evaluate their proposed ML based IDSs in addition to dataset analysis.

In this research, SDN combined with an efficient ML model produced an intelligent IDS that is capable of acquiring data from the tactical network devices. It then processes and analyze such the data to identify intrusive behaviours, while able to take countermeasures in real-time without any human intervention.

1.8. Organization of the Dissertation

The remainder of this dissertation is organised as follows:

Chapter 2 provides the theoretical background regarding the problem analysis and technologies concerned. The chapter discusses the concepts of security in tactical communication networks, intrusion detection, and the Software-defined networking paradigm. This chapter also provides the related work found in the literature focusing on the different approaches used for intrusion detection and methods of evaluation.

Chapter 3 presents the methodology adopted to fulfil and achieve the goal and objectives of the study. This chapter helped guide the research to ensure rigour and facilitate the research.

Chapter 4 provides a performance evaluation of popular Machine Learning techniques commonly used for intrusion detection. This chapter directly responds to the second research objective, which is to analyse and select the most suitable Machine Learning method that can be used for intrusion detection in tactical networks. From this chapter, two conference papers were presented, one based on Packet-based NIDS (Zwane, Tarwireyi and Adigun, 2019d) and the other on Flow-based NIDS datasets (Zwane, Tarwireyi and Adigun, 2019b).

Chapter 5 presents the proposed solutions for network security in resource-limited network environments. The chapter proposes a Flow-based IDS model and explains its corresponding components. Based on this chapter, a conference paper was published (Zwane, Tarwireyi and Adigun, 2019a).

Chapter 6 presents the operationalisation of the model proposed in Chapter 5. In this chapter, a proof-of-concept prototype of the model was implemented and evaluated for its effectiveness. Another conference paper was generated from this chapter, which outlines the proposed deployment architecture, test scenarios and preliminary results (Zwane, Tarwireyi and Adigun, 2019c).

Chapter 7 presents the results and discussions to help validate the proposed approach.

Chapter 8 concludes the study and the findings. The chapter summarises and further responds to each of the research questions outlined for the study. The conclusion, limitations, and future direction of this research are presented.

Chapter 2: Background and Literature Review

This chapter investigates the state-of-the-art and debates its relation to security issues in tactical networks. Conducting a detailed investigation of recent and innovative technologies could enable us to identify paradigms that can be used to design an ideal intrusion detection framework. This investigation also helps us to identify potential paradigms capable of addressing some of the network and communication security concerns in tactical networks. In accordance with the adopted methodology, Design Science Research, Step 1 is the problem analysis phase where the gaps in existing literature are identified.

In that regard, this chapter explores the different technologies that can be adopted and implemented, to achieve the aim of this research. This chapter is structured as follows: in Section 2.1 the notion of network security and the security requirements for tactical networks are presented. This aims to provide enough background and the area of interest for this research. Section 2.1 further discusses different popular intrusion detection approaches available in the network security research domain. This section gives a review of statistical, Machine Learning, and other intrusion detection approaches.

A review of the military tactical communication networks is presented in Section 2.3. The section discusses the challenges presented by other researchers in the deployment and usage of intrusion detection mechanisms on the military battlefield. In Section 2.4, the techniques recognised as suitable for adoption in intrusion detection system design for tactical networks are presented. Hence, ML and SDN are presented as the required solutions to mitigate the issues currently experienced in tactical network IDS deployment. Studies utilising ML for the task of intrusion detection are reviewed and their limitations are presented. Similarly, studies presenting intrusion detection techniques using ML and SDN are also reviewed and gaps are identified. Finally, Section 2.5 presents a summary of the issues, gaps, and future research presented in the reviewed works.

2.1. Tactical Networks Security Challenges and Implications

The magnitude of instability in a hostile environment is enormous, as many problems and challenges are encountered in tactical networks. Security is one of the most important issues to be addressed. Network security plays an important role in tactical communication networks as security breaches can directly affect mission success and cost personnel lives. Military tactical communication presents many different challenges to network security researchers. Hence the most appealing characteristics which influence such challenges are the nature of the network environment since network members are usually mobile, and topology changes over time as nodes randomly leave and join in an ad hoc manner. In addition, the devices or equipment are subject to capture, and enemies are considered well-skilled and motivated.

Over the past years, researchers have tried addressing the many issues encountered in tactical networks. For example, a protection mechanism that protects the physical layer from smart jammer attacks was proposed by Jeung *et al*, (2011). Usually, a smart jammer is used by an impostor to examine wireless channels for active channels and discharge a jamming signal on that channel. The method by Jeung *et al*, (2011) refers the smart jammer to the incorrect channel and prevents the actual channel from being jammed. Another example includes the efforts of (Madhu and Sreekumar, 2014), where they used a wireless sensor network to implement a secure unmanned vehicle navigation system. They used a cluster-based method where each of the clusters contained a set of armed and sealed motes in a specific area to prevent physical attacks, shown in Figure 2.1. In addition, a key management technique to avoid single key compromise which usually led to the entire network being compromised was proposed through using an improved version of LEAP. Their work resulted in a vehicle navigation system that was controlled by wireless sensor networks making the network more secure. The authors also claimed that their system was applicable to several applications, which include fire detection, and maintenance and monitoring applications.

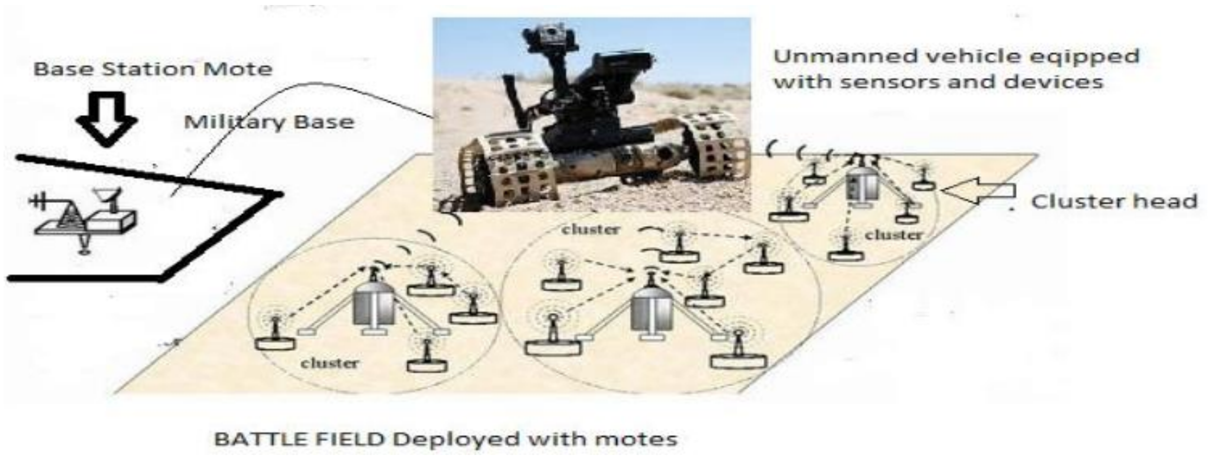

Base Station Mote
Military Base
Unmanned vehicle eqipped with sensors and devices
Cluster head
cluster
cluster
cluster
BATTLE FIELD Deployed with motes

Figure 2.1 Unmanned vehicle to protect the battlefield (Madhu and Sreekumar, 2014)

The stated work applies as evidence of the effort towards ensuring security in tactical networks. Unlike commercial wireless networks, tactical wireless networks operate in a hostile environment where the conditions are very unstable. Pawgasame and Wipusitwarakun, (2015) presented six issues that are encountered in tactical wireless networks as a result of the large magnitude of instabilities in the harsh environments where they are deployed. The first issue defined was the challenge of understanding the network’s behaviours. The great uncertainty in the network makes it hard to describe and predict network behaviours and outcomes. The ability of the network to cope with unstable changes was also reported as another issue experienced in tactical wireless networks. For example, real-time information sharing is required by some military applications, thus if the information transmission system cannot tolerate disruptions due to a harsh environment, defence systems may lose track of the hostile target. Another issue they reported in tactical wireless networks was network congestion. Congestion occurs when multiple mobile nodes are trying to communicate across the network gateway or access the same channel at the same time. Since instabilities may cause delays in packet transmission, packets may arrive at the gateway at the same time and cause congestion. In addition, congestion may also be caused during retransmissions due to unstable channels.

Also, in a hostile environment, packets may be corrupted or dropped due to the instability of the network, this may result in unreliability in the data delivery (Pawgasame and Wipusitwarakun, 2015). Intermittent interferences and hostile attacks may corrupt packet header and route packets to the wrong destination. Thus, reliability remains an essential issue that requires attention in tactical wireless networks. Also, the instability and dynamic movement of nodes in a tactical network present availability issues, since nodes may be out of range or links may be broken. As such, the ability of the network to provide network services is also a challenge in a tactical wireless network. Finally, security issues are commonly triggered by hostile attacks. For example, an adversary may insert malicious nodes into the network to drop, modify, or steal information to introduce information reliability problems. While sending huge numbers of packets could result in congestion and availability complications, likewise, network robustness and availability can be violated through communication disruption caused by an enemy jamming the communication signals. A summary of tactical wireless networks’ research interests, gaps, and challenges are presented in Table 2.1.

Table 2.1. Summary of research interest in tactical wireless networks (Pawgasame and Wipusitwarakun, 2015)

Research interests	**Solved Issues**	**Gaps**	**Challenges**

Wireless network modelling	Analysing the network	Hostile attacks capture models	Modelling network hostile attacks
Performance	Robustness, Reliability, Congestion	The trade-off on each technique	Performance improvement against instability for minimal trade-off effects
Routing	Robustness, Reliability, Availability, Security	Unstable reliable routes in a hostile environment	The predictable route is more attractive in a hostile environment
Security	Security, Availability, Reliability	False detection of hostile nodes in a hostile environment	Accurate detection of hostile nodes in the hostile environment
Management	Congestion and Reliability	Hard to achieve management in uncertain networks	Management with uncertain network parameters

In (Spencer *et al.*, 2016), the authors argued that the delivery of information services was hard to achieve due to the nature and structure of the deployment environment. They reported that the general characteristics of military networks, which include heavy reliance on wireless barriers, critical reliance of the commander on real-time access to information, and network installations and reinstallation at very short notice, introduces problems to network managers as they have to deliver vital information with the required quality of service in the face of changing operational priorities (Spencer *et al.*, 2016). They further argued that in various deployed military networks there is no single management system by which networks and services are configured and provisioned. Hence the creation of a hostile set of services in the tactical environment becomes a complex exercise in terms of planning and configuration, as there is a need for the manual configuration of a large number of devices, and this approach is also prone to human error.

The issues presented by (Spencer *et al.*, 2016) serve as an example of other issues in the military tactical network that affects or limits the swiftness and elasticity of a military force. This is because the operational tempo is degraded due to time-consuming planning and configuration processes undertaken each time the network, or part of the network, is moved. In addition, an expert signaller needs to be available at every networked site to support the delivery of information services which is usually a burden in terms of training and logistics in order to maintain and support such personnel (Spencer *et al.*, 2016). The mentioned studies illustrate a clear picture of the challenges and issues of experience in tactical networks. Some issues are introduced due to the environment in which such a network is deployed, and other issues are introduced by the general characteristics of the tactical network. We also observe that in addition to security challenges, the lack of centralised control and management where security services can be provisioned in tactical networks remains a gap that needs to be addressed. Sterbenz *et al.,* (2002) presented issues and challenges experienced in improving the survivability of mobile wireless networks and military networks' requirements. The authors presented six security and operational requirements for wireless networking technologies to support military operations:

- Transmission security (TRANSEC) – concerned with protecting wireless communication at the physical layer, medium access, and data link layers over wireless media.
- Communication Security (COMSEC) – concerned with protecting data and voice communications between designated endpoints, it is one of the most important security requirements that must be addressed.
- Network Infrastructure Protection (NIP) – defined as the protection of routing and network management infrastructure against both passive and active attacks.
- Authentication and Access Control (AAC) – defined as the support for multi-level security measures by implementing role-based access control on the application, application servers, and their proxies.
- Robustness – defined as the requirement for supporting hardware and software failure, asymmetric and unidirectional links, or limited connection range of wireless communication.
- Effectiveness – defined as efficiency in the use of electrical and computing power, silicon real estate, and communication bandwidth.

While those security requirements are equally important in securing networking technologies supporting military operations, in this study, we only focus on NIP through intrusion detection systems (IDS). Early efforts of IDS focused more on protecting the hosts from malicious intents. These systems were known as host-based intrusion detection systems (HIDS). However, with the birth of computer networking and its adoption by organisations, part of the IDS research shifted towards network intrusion detections (NIDS) which is the area of study in this research. The next section presents popular network intrusion detection techniques proposed by other researchers to address network security in communication networks.

2.2. Categorisation of NIDS Techniques

The purpose of network security is to serve as a mechanism to guard network resources and users against unauthorised and malicious intermediaries. Network security problems are intricate and valid for all types of computer networks regardless of whether they are for home users, commercial use, or military purposes (Karresand, 2004). Thus, network security plays a very important role in today's communication systems because they ensure confidentiality, integrity, and availability (CIA) of communication and network resources. However, in recent years different attacks have been formulated to attempt to compromise these CIA principles (Kidston *et al.*, 2010).

Intrusion detection systems (IDS) are an essential tool for protecting IP-based networks and to a certain extent maintain the CIA principles. IDS examine network traffic and computer system logs to identify attacks and raise alerts if attacks are detected. Traditional methods of intrusion detection employed deep packet inspection or stateful protocol analysis to detect attacks in network traffic (Fahad, Sher and Bi, 2017). Stateful protocol analysis checks complete semantics of protocols against a specified range and considers out-of-range values as intrusions. This method is regarded as computationally expensive (Fahad, Sher and Bi, 2017).

On the other hand, deep packet inspection presents challenges when the monitored network traffic is encrypted, additionally, it is computationally expensive to inspect complete packet payload and can cause performance bottlenecks in a high-speed IP network. (Karresand, 2004) argued that one of the most critical challenges in intrusion detection is encryption. They claimed that the main key in network intrusion detection systems is to inspect the packets sent over the network. This implies that the more information available the more efficient and correct the detection will be. However, when different parts of the packets or parts that approximately relate to the different layers in the OSI stack are encrypted, the efficiency of the network intrusion detection system decreases. Their claim is valid, especially in military networks, as very strong and thorough encryption policies are used to ensure the confidentiality of the information sent (Karresand, 2004).

As an alternative approach to those limitations, researchers such as Fahad, Sher and Bi (2017) proposed a flow-based intrusion detection system. This is a new solution to protect IP networks from unauthorised access. Flow-based Intrusion detection systems utilise network flow records as input and analyse them to discover whether the network traffic is either normal or malicious (Fahad, Sher, and Bi, 2017). Flow-based IDSs have become attractive to researchers due to the number of advantages they offer over traditional deep packet inspection techniques for intrusion detection. For example, 1) Flow-based inspection of packets has fewer privacy concerns than packet-based inspection since user information or payload is sheltered from any transitional scans. 2) Flow-based Intrusion detection can handle encrypted data as it only analyses packet header or flow information. And 3) Flow-based IDS are reported to have the ability to operate on high-speed backbone links, low deployment costs, and near real-time response, (Fahad et al., 2017). In this work, we adopted flow-based attributes for the proposed intrusion detection system, since it is more effective than using traditional packet-based attributes in terms of speed, resource usage, and real-time detection of attacks. Hence, we present a methodology for gathering network flow data in an SDN enabled network.

The adoption of flow-based intrusion detection has resulted in the proposal of different techniques for its design. A taxonomy of flow-based intrusion detection techniques was presented by (Fahad, Sher and Bi, 2017), shown in Figure 2.1. The taxonomy hierarchy classifies flow-based intrusion detection system approaches into statistical, Machine Learning, and other techniques.

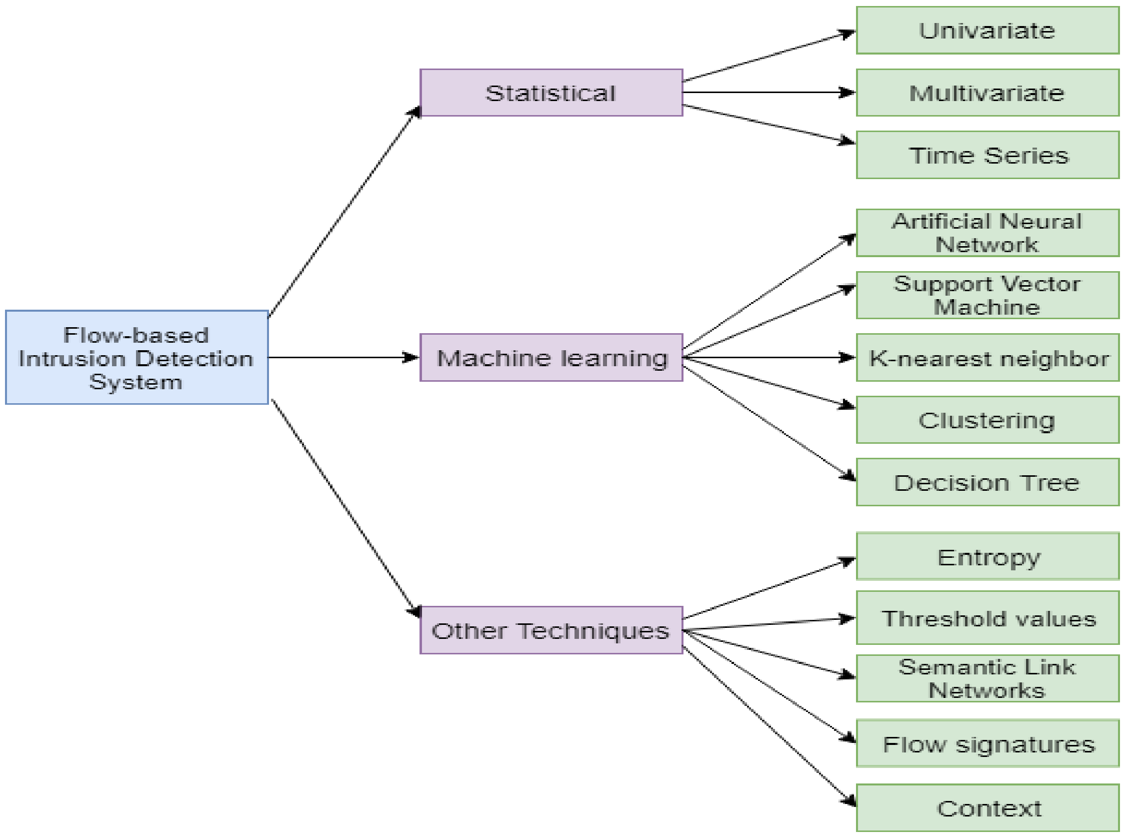


Figure 2.2 Taxonomy of flow-based intrusion detection techniques (Fahad, Sher and Bi, 2017)

In statistical techniques, the system builds a profile of the network traffic using a statistical function of the network traffic parameters (Fahad, Sher and Bi, 2017). The created profile is then used to compare incoming unseen traffic. The technique uses statistical measures to calculate the similarities between network traffic and the profile of normal network traffic. If the similarities are beyond a defined threshold the flow is marked malicious, otherwise normal. (Fahad, Sher and Bi, 2017) divided statistical methods into univariable, multivariable, and time-series statistical methods, see Figure 2.2.

Univariable statistical techniques (Ye *et al.*, 2002) are methods that analyse a single variable at a time, this can be the mean or standard deviation. Generally, they assume an underlying known distribution of the data. In a multivariable technique, the relationship between two or more variables is analysed. Lastly, time-series statistical methods (Viinikka *et al.*, 2009) use previously observed or seen values to predict new values.

Machine Learning (ML) is a field of computer science that trains computers to think like humans and make decisions when required (Haq, Onik and Shah, 2015). These methods try and copy human thinking practices which include logical reasoning, intuition, learning from past experiences, trial and error, and generalisation. ML techniques have been used extensively over the years, and they continue to be applicable in flow-based intrusion detection as well. A comparison of both statistical and machine learning techniques as presented in Table 2.2.

Table 2.2. IDS technique advantages and disadvantages

ID techniques	**Advantage**	**Disadvantage**
Statistical techniques	• Don't require previous knowledge of network attacks. • Can precisely detect attacks that cause abrupt and highly differed changes in network traffic.	• High dimensionality in network traffic affects performance. • Challenging to calculate the indicators of normal network traffic. • It can be bypassed by small and slow-ramped attacks that keep the effect of attack below statistical thresholds.
Machine learning techniques	• Can adjust themselves according to the traffic passing through. • Have a high detection rate. • Methods like ANN are able to generalise the model from limited information.	• Difficult to construct representative training datasets for supervised Machine Learning methods. • Can be computationally costly during training. • Have high false-positive alarm rates. • Unsupervised learning techniques need background information to determine the number of groups.

Other common techniques used for flow-based IDS are entropy, flow metric thresholds, flow signatures, and semantic link networks (SLNs) (Fahad, Sher and Bi, 2017). The idea behind entropy is to capture important characteristics of features in the traffic distribution and use these features to detect abnormalities and malicious behaviour in the network traffic. The commonly used entropy methods include Shannon (Zaidi *et al.*, 2017), Renyi, and variations of Tsallis entropy (Berezinski et al, 2014). The flow metric threshold is also used to detect an intrusion in the network flows. For example, one can specify an upper or a lower bound threshold that can send alerts if the specified threshold is being violated by any flow passing through the observation point. SLN mines the time, location, and other related information from the flow data which is used by the semantic links to detect suspicious flows on probabilistic semantic networks (Fahad, Sher, and Bi, 2017).

(Choudhary *et al.*, 2018) presented a survey based on IDS tools with particular focus on vulnerabilities and attacks directed towards a networked Unmanned Aerial Vehicles (UAV) environment. In their work, they presented the key component taxonomies of the UAV IDS. The components of the taxonomy are shown in Figure 2.3. The work also argued that to achieve a UAV-IDS system that ensures the necessary levels of effectiveness and efficiency, the IDS computational cost, threat and behaviour modelling, detection latency, implementation overhead, threat assessment, maximum network throughput, minimum resource consumption, and effective monitoring and response are challenges that need to be addressed to construct a secure cyber-physical UAV-IDS system (Choudhary *et al.*, 2018).

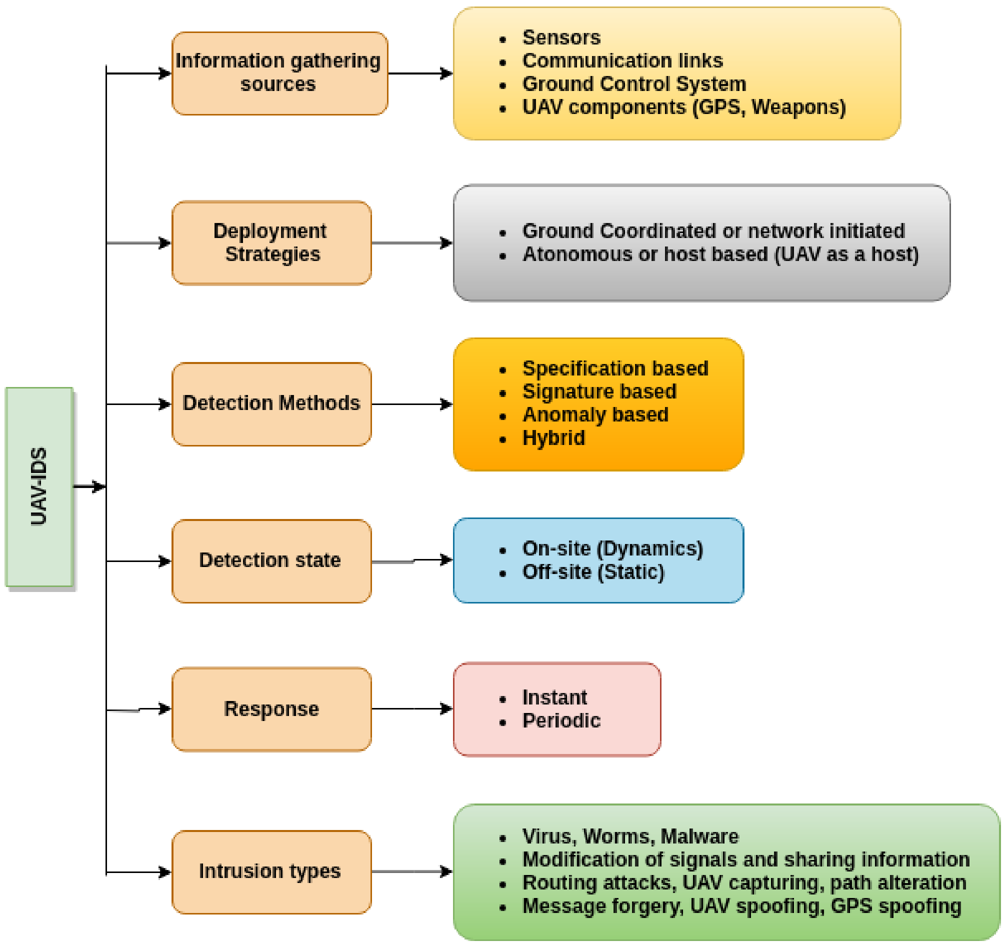

UAV-IDS
Information gathering sources
Sensors
Communication links
Ground Control System
UAV components (GPS, Weapons)
Deployment Strategies
Ground Coordinated or network initiated
Atonomous or host based (UAV as a host)
Detection Methods
Specification based
Signature based
Anomaly based
Hybrid
Detection state
On-site (Dynamics)
Off-site (Static)
Response
Instant
Periodic
Intrusion types
Virus, Worms, Malware
Modification of signals and sharing information
Routing attacks, UAV capturing, path alteration
Message forgery, UAV spoofing, GPS spoofing

Figure 2.3 Taxonomy of UAV-IDSs (Choudhary *et al.*, 2018)

In that regard, this study focused on state-of-the-art NIDS techniques regarded as most effective for intrusion detection. Another aspect that is usually ignored by researchers is how to handle data collection and processing in a timely and efficient manner to accommodate high volumes of data (Sultana *et al.*, 2019). In this study, the possibility of Software Defined Networks (SDN) to ensure efficient and real-time data collection from the network is considered. Hence in the next section, the state-of-the-art techniques and approaches that have the potential to address the challenges of intrusion detection systems in communication networks are presented.

2.3. State-of-the-art NIDS Techniques

Intrusion detection systems can be constructed using different techniques including statistical, time-series, ML, and others. Our study focusses on ML-based techniques as they are being continuously adopted due to their effectiveness when compared with other intrusion detection techniques (Fahad, Sher and Bi, 2017) (Khraisat *et al.*, 2019). Also, Machine Learning and Artificial Intelligence (AI) have many potential applications in the military context in different domains and all levels of warfare (Svenmarck *et al.*, 2018). For example, "ongoing advances in artificial intelligence (AI)" is planning to change society and eventually the character of war in accordance with the 2018 National Defense Strategy. The DoD has prioritised AI investment to retain multidomain supremacy over peer and near-peer opponents. The DoD AI approach calls for quickening the delivery and approval of AI to establish a common foundation to scale AI's impact across the department and enable decentralisation, development, and experimentation; developing partnerships with industry, academia, allies and partners to promote an AI workforce and to lead in military AI ethics and safety.

In addition, with the introduction of SDN, many researchers are now focusing on how SDN can help address limitations presented by tactical network deployment architecture. For example, in (Poularakis, Iosifidis and Tassiulas, 2018) an innovative architecture design for SDN-enabled mobile ad hoc networks was proposed. SDN has the potential to promote more advanced traffic management in the tactical boundary (Spencer *et al.*, 2016; Poularakis, Iosifidis and Tassiulas, 2018) and other domains. Also, Yan *et al.*, (2016) argued that SDN introduces opportunities for improved traffic management agility, as the complexity and volume of traffic from tactical systems grows. This study focuses on how state-of-the-art technologies such as SDN and ML can be utilised to improve network intrusion detection capabilities in tactical communication networks. The next subsections present works already done by other researchers in the field of intrusion detection using SDN- and ML-based approaches respectively.

2.3.1. Machine Learning-based IDS Approach

ML methods have recently been applied extensively in cybersecurity applications. ML classification algorithms commonly used for intrusion detection are applied in the form of Single classifiers, Hybrid classifiers, and Ensemble classifiers (Govindarajan and Chandrasekaran, 2012). Single classifiers are utilised when a single ML algorithm is used to construct an intrusion detection system. Hybrid classifiers offer a combination of more than one Machine Learning algorithm. Ensemble classifiers use multiple weak learners, such as classifiers performing somewhat superiorly to a random classifier. Hybrid classifiers are preferred over single classifiers, this is because one algorithm can be used for pre-processing the samples in a training set, removing non-representative training samples, then the results can be given to the second algorithm for pattern recognition to design a classifier, this method can vastly improve intrusion detection performance. Some of the most popular ML classification algorithms used in intrusion detection tasks include: 1) Decision Tree (DT) with Gini index (Breiman et al., 1984), Gain-ratio (Quinlan, 1993), and Chi-square (Mingers, 1989b), 2) Naïve Bayes (NB) (Division *et al.*, 1997), 3) Support Vector Machine (SVM) (Chen *et al.*, 2018), 4) Multi-Layer Perceptron (MLP) (Djuris, 2012), 5) Adaptive Boosting, 6) Bayesian Network (BN) (Division *et al.*, 1997), 7) Random forest (RF) (Jabbar *et al.*, 2017), and 8) Bootstrap Aggregation.

In (Buczak and Guven, 2016), an ML and DM methods used in the cybersecurity literature review was presented. The authors reported that the most effective method for cyber applications has not been established, they stated that due to the fertility and complexity of these techniques, it was difficult to make one recommendation for each task. Everything is based on the nature of an attack that the system was intended to detect. The authors further argued that when determining the effectiveness of ML/DM methods, there are several criteria that one needs to take into consideration; the accuracy, complexity, classifying speed, and understandability of the final solution of the ML/DM method. Their study also highlighted the importance of data sets in ML/DM for cyber intrusion detection. Buczak and Guven (2016) reported that for effective anomaly or misuse detection, it is beneficial for IDS to be able to reach network and kernel-level data, if possible, network data should be augmented by OS Kernel-level data. Their study however only focused on previously conducted works to analyse the usability of ML/DM in the context of intrusion detection. This study takes this work further by proposing an IDS method that utilises ML and conducts a performance evaluation of the system.

In (Yuill *et al.*, 2000) an intrusion detection technique to assist system administrators with intrusion detection problems encountered during incident response is proposed. The main goal was to identify the network devices that are likely to be compromised by an attacker. The authors proposed a solution based on the assumption that during an attack, the attacker reveals information about themselves and about the network vulnerabilities, which can be used to identify the networks likely compromised devices (Yuill *et al.*, 2000). Based on the US military battlefield intelligence process, the authors constructed models of the network as a battlespace. They constructed models of the attackers' capabilities, intentions, and course of action. They used the economies of crime, which is referred to as the theory behind criminal behaviour, to model the attackers' course-of-action. Finally, the models of network and attackers were used to identify the devices that are more likely to be compromised (Yuill *et al.*, 2000).

In a recent study by Pushpa and Kathiravan (2016), a cross-layer based multiclass intrusion detection system for secure multicast communication of MANET in a military network was presented. The authors introduced an indirect internal stealthy attack by skipping the collision avoidance mechanism against the unicast route discovery control packets of tree-based multicast routing protocol MAODV. They then analysed the robustness of the MAODV against indirect and direct internal stealthy attacks such as black hole and deny-to-forward, where they observed severe impact in PDR, throughput, and control overheads (Pushpa and Kathiravan, 2016). The authors then proposed a cross-layer based distributed Machine Learning anomaly detection system to protect against those stealthy attacks. They used MAC and routing layer integrated features instead of routing layer features alone to improve accuracy. The method presented high effectiveness for anomaly detection in military networks, however, their method is prone to high resource consumption and slow as it needs to collect data from different layers, such as the MAC and routing layer. Therefore, the present study envisions an intrusion detection approach that will only analyse flow-based data to reduce data collection and processing challenges.

In (Rhodes *et al.*, 2005), a maritime situation monitoring and awareness system using learning mechanisms was presented. Their system takes real-time tracking information and uses continuous on-the-fly learning which enables concurrent recognition of patterns of current states of single vessels in a local vicinity. Their learning system used a modified version of the

Fuzzy ARTMAP neural network classifier (Carpenter *et al.*, 1992). In essence, the approach consisted of an unsupervised clustering algorithm and a supervised mapping and labelling algorithm (Rhodes *et al.*, 2005). Their study illustrates the successful implementation of Machine Learning and AI in military settings. However, their implementation of Machine Learning techniques does not address communication security, instead, it focuses on vassal prediction. The present study applies Machine Learning to improve intrusion detection capabilities in military communication networks.

Liu *et al.* (2018) proposed an SVM-based weighted learning limit learning machine based on extreme learning machine (ELM) to address the problems of large amounts of data, high-security, and intrusion detection requirements in civil-military integration. The authors adopted an ELM mechanism since it is a fast learning method of a single hidden layer feedforward neural network, where the whole learning process is completed only once without iteration and resulting in extremely fast learning speed. In their approach, they first clarify the hidden layer of responsibility for each node, instead of tentatively setting the number of nodes required for the hidden layer like the original ELM, the number of nodes needed for a hidden layer is determined according to the classification purpose. Then SVM weight is used to optimise the weight and offset of each node. This ensures that each node has a better ability to complete the task of generalisation. The results obtained from their experiments indicate that SVM-ELM has higher detection accuracy and can quickly complete the training with superiority and stability compared to BP algorithm (Liu *et al.*, 2018). The authors claimed that for data-based civil-military integration equipment support system construction, they recommend implementing their method of intrusion detection. However, their study used the DARPA 1999 KDD dataset to assess their intrusion detection approach. Recent studies have argued that these datasets should no longer be used to evaluate intrusion detection methods because they are old and not a correct representation of modern network patterns (Gogoi, Bhuyan and Bhattacharyya, 2012). The present study will utilise a more recent intrusion detection evaluation dataset and network data sampled from a simulated tactical environment.

Revathi and Malathi (2013) compared the effectiveness of five Machine Learning models, such as Random Forest, J48, Support Vector Machine, CART, and Naïve Bayes. They observed that Random Forest outperformed the other algorithms by achieving higher test accuracy than the others. In a survey (Haq, Onik and Shah, 2015), the authors argued that SVM and ANN algorithms are usually the most popular approaches proposed for single learning classification. In addition, AdaBoost and majority voting are the most popular ensemble classifiers for intrusion detection. More recently, (Ertam, Õ and Yaman, 2017) compared Naïve Bayes, Bayesian Network, Random Forest, Multi-Layer Perceptron, and SOM for intrusion detection in computer networks. The authors found that MLP achieved high accuracy followed by Random Forest and Bayesian Network. The authors also reported that even though MLP achieved higher detection accuracy, the process of building the model for MLP took about 12 hours, which is undesirable, so they recognised Random Forest and Bayesian Network as best-performing in their study.

ML has been successfully deployed in a number of military applications, for example, surveillance and underwater mine warfare (Svenmarck *et al.*, 2018). However, for security applications, ML-based IDSs have been proposed and studies analysed their performance using datasets. Evidence of increased classification accuracy of attacks and automated model construction was reported. With those attributes, the adoption of ML techniques to enforce security in military tactical networks becomes an obvious choice since it can ensure perfect detection and recognition totality in such networks. However, most studies propose ML for intrusion detection without demonstrating how such systems can be operationalised. This study focuses on finding the most suitable ML algorithm then designing and implementing an IDS utilising such an algorithm to demonstrate the operationalisation of such a system. This will ensure that ML algorithms are not only recommended as better-performing using datasets but also consider their performance from a deployment point of view.

2.3.2. Software-Defined Network-based NIDS

Since SDN can facilitate dynamic policy control of a network, (Spencer *et al.*, 2016) argued that this can help establish services automatically based on policy, reduce planning overheads, reduce the operator burden and configuration errors. They also reported that network resources allocated to each service and the path taken through the network by the service's traffic can be dynamically controlled through SDN. This implies that network resources can be focused on mission goals even as mission priorities change(Spencer *et al.*, 2016). For example, (Ji Qing *et al.*, 2015) Proposed a flattening military network resource management framework that was based on Software-defined Networking (SDN) technologies, Figure 2.4.

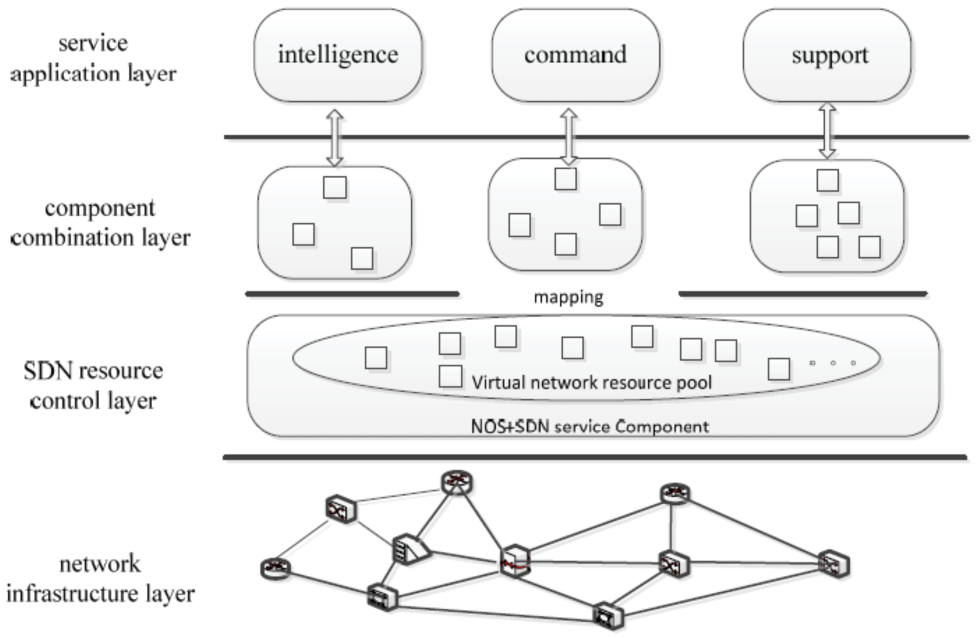

service application layer
intelligence
command
support
component combination layer
mapping
SDN resource control layer
Virtual network resource pool
NOS+SDN service Component
network infrastructure layer

Figure 2.4 Resource management framework with SDN control (Ji Qing et al.*, 2015)*

The authors modelled the relationship between military operations and further proposed a pre-combination service component-based resource optimisation method. Their simulation results indicated that their method improves the average response time for service requests. In their study, the authors focused on addressing network management issues in tactical networks (Spencer *et al.*, 2016). Their work further demonstrated the effectiveness of SDN in terms of logical controlling, where network services can be centrally managed and configured (Spencer *et al.*, 2016). This study extends the presented work by utilising SDN central logical controlling capabilities for network intrusion detection and mitigation as opposed to resource management.

Researchers in security have also investigated the use of SDN-enabled techniques for intrusion detection and mitigation, (Alsmadi and AlEroud, 2017) (Chang *et al.*, 2013) (Monshizadeh, Khatri and Kantola, 2017) (Yoon *et al.*, 2015) (Boero, Marchese, and Zappatore, 2017).

Bhunia and Gurusamy. (2017) proposed an SDN-based framework called SofThings for the detection of anomalies and mitigation of anomalies in IoT traffic. The objective of the framework was to achieve early detection of traffic anomalies closer to the edge of the network instead of detection at the core or higher levels of the network. This enabled fast identification of attacks on IoT devices and the initiation of mitigation procedures as appropriate. In the study, the Support Vector Machine (SVM) Machine Learning algorithm was used to detect anomalous traffic. Precision and recall were the two performance metrics used to measure the performance of the framework. From the results of the study, the authors observed a few false-positives when linear SVM was used, hence obtaining lower precision of detection. They further observed that when using non-linear SVM they obtained better precision. This was because non-linear SVM uses the Kernel trick and consequently reduces wrong detection. In addition, they also reported that their method can quickly restore the throughput loss, and is hence able to mitigate different attacks within a few seconds. In this study, we employed a similar approach; that is, adopt both SDN and machine learning to design an IDS. However, instead of focusing on IoT scenarios, our study took into consideration the tactical military network scenarios.

The work of (Kidston *et al.*, 2010) proposed a cross-layer framework to help solve network security issues in tactical networks. The framework supports automation and efficiency which is very useful in tactical networks. The framework proposed by the authors promotes the coordination of security services across the different communication layers. It was influenced by the fact that by taking metrics from the security services at one layer, for example, the authentication system and IDS, operations at other layers can be made more secure or optimised (Kidston *et al.*, 2010). For example, authentication and IDS operating at the application layer can provide real-time attack profiles into an integrated cross-layer security service. These results can then be passed to the other lower layers to improve their efficiency and robustness. However, this method increases the complexity and internal processing within a node, also increasing the communication requirements between nodes. The authors argued that security services that can be integrated using their framework include IDS, frequency hopping, and distributed authentication.

Wrona and Szwaczyk. (2017) argued that SDN networks offer a promising framework for the implementation of cross-layer data-centric security policies in military communication systems. They argued that one of the most important aspects of designing advanced security solutions is thorough experimental assessment and validation of proposed technical concepts prior to deployment in operational military systems. In their work, they proposed an OpenFlow based testbed for validating SDN security mechanisms. Their method can handle both mechanisms for protecting the SDN layer and data-centric security policies. The results obtained in their study confirmed their method's ability to validate simulation and analytic predictions.

In a study by Giotis *et al.* (2014) a scalable mechanism for performing anomaly detection and mitigation in SDN architectures is proposed. The mechanism is comprised of a) reduced data gathering with sampling being handled by the sFlow protocol, b) Anomaly detection, which was implemented by an entropy-based algorithm, and c) network-wide anomaly mitigation using OpenFlow. The authors further demonstrated that OpenFlow statistics collection and processing introduced scalability issues since it overloads the centralised control plane. They also argued that in low traffic environments, the performance of their mechanism was comparable to the native OpenFlow implementation. Moreover, they stated that in terms of resource usage, the proposed sFlow based approach presented superior behaviour compared with the native OpenFlow mechanism. Taking into consideration the lack of resources in tactical devices, this study will use sFlow sampling to ensure efficiency instead of the native OpenFlow sampling mechanism.

In (Sultana *et al.*, 2018) a survey with the overview of programmable networks, such as SDN, and various Machine Learning (ML)/ Deep Learning DL approaches were presented. The survey presented different challenges experienced while developing a flexible and efficient NIDS using ML/DL based techniques (Sultana *et al.*, 2018). The authors reported that one of the most predominant challenges is choosing the appropriate feature selection methods that can precisely determine relevant features for the IDS. They also argued that the existing dataset is not accurate for research and academic predictions. This issue makes it essential for researchers to create datasets to ensure consistent and accurate evaluation of NIDS. Another fundamental challenge of SDN-based NIDS presented by the authors was how to handle the processing of a high volume of data.

The study conducted by Sultana *et al.* (2018) also argued that to design a centralised SDN controller that can monitor and implement real-time intrusion detection in high-speed networks is a desired future goal that will be challenging to address. However, since this work proposes a flow-based intrusion detection method, it is capable of providing real-time network security solutions in high-speed networks. This is because the proposed method analyses network flows, which contains the summary of the packet header and not the packet payload as traditional deep packet inspection approaches do.

The work of Giotis *et al.* (2014) proposed a mechanism utilising sFlow for data gathering and sampling instead of the native OpenFlow protocols. This work adopts a similar approach. We employ the sFlow protocol for data gathering and sampling on top of SDN. This reduces processing overload and scalability issues in the control plane (Giotis *et al.*, 2014). This approach also has the potential to address the challenge of handling packet processing flows, as mentioned by (Sultana *et al.*, 2018).

2.4. Summary

Security requirements for tactical networks range from transmission security, communication security, authentication and access control, network infrastructure protection, and others. This study focusses on network infrastructure protection as a second line of defence in addition to encryption techniques used to address transmission, authentication, and communication security. To address NIP, one of the most effective solutions is network intrusion detection systems. Over the years, different intrusion detection techniques have been proposed which include the use of rule-based, statistical, and ML-based techniques. However, the environment and the nature of tactical networks present challenges to intrusion detection techniques. The implementation of intelligent security solutions in such networks requires add-on mechanisms that can allow logically centralised management for effective provisioning of security services (Spencer *et al.*, 2016) and an intelligent intrusion detection method that can detect malicious activities with high detection rates and presents minimal false detections.

In this chapter, an overview of network security, intrusion detection, and different challenges experienced in tactical network security was presented. This chapter further reviewed different works proposing intrusion detection systems that use ML as the detection method. The works reviewed indicated that it remains unclear which ML algorithm can perform better or can achieve higher accuracy in detecting hostile attacks in tactical networks. Different approaches have been proposed and evaluated by other researchers, however, most of their evaluations are conducted using outdated datasets (very old) and there is no evidence of actual deployment of these techniques to validate their performance in their respective production environments.

Furthermore, the lack of centralised management for effective provision of security services led to the investigation of SDN and its advantages in addressing IDS requirements in tactical networks. Different works proposing intrusion detection techniques that use SDN for intrusion detection were also reviewed. The review indicated that SDN can effectively and efficiently gather network data in real-time and globally for IDS to ensure timely and quick identification of network security breaches. From this chapter, we find evidence that an effective IDS for tactical MANETs can be constructed using SDN capabilities and ML techniques integrated to develop an intelligent IDS that can monitor and detect intrusions effortlessly without any human intervention.

Chapter 3: Research Design and Methodology

As mentioned in Chapter 1, this study aims to address the problem of intrusion detection in tactical military networks. Military tactical networks are usually resource-constrained and deployed in harsh environments where network communications are not stable due to different terrains types. In that regard, security approaches such as Intrusion Detection Systems (IDS) deployed in this network usually suffer from high false detection rates due to constant changes and mobility in the network. In addition, detection and handling of security incidents doesn't happen as quickly and efficiently as possible, which has a negative impact on the functioning and availability of the network. The goal of the study is, therefore, to investigate and implement the most suitable Machine Learning (ML) algorithm for intrusion detection while utilising SDN to improve network security in tactical mobile ad hoc networks.

In this Chapter, different Computer Science and Information Systems (CS/IS) methodologies that can be adopted by a researcher for their study are presented. A review of theoretical, Simulation, Experimentation, Case Study, and Design Science was conducted in order to determine the most effective method for this study. Due to the objectives and goal of this study, Design Science and Experimentation were the most appropriate methods to use. This study required that experiments be conducted to determine the most effective ML technique in detecting intrusions, which influenced the selection of experimentation methodology. Also, in accordance with the third research objective, the Design Science method was the most appropriate to facilitate the design and development of the IDS for tactical networks as it ensures a rigorous process to design an artefact intended to solve an observed problem (Hevner *et al.*, 2004), which is intrusion detection in this study.

Furthermore, the use of ML techniques requires that ML models be constructed and validated before they can be integrated and deployed. To facilitate this process, the data science project life-cycle was used. The data science project life-cycle contains steps and guidelines to ensure the development of quality and effective ML models, discussed in Subsection 3.2.3. To evaluate the IDS, instead of accuracy, the most-used metric for evaluating IDS models, the precision, recall, f-score, and AUC metrics were used since they are independent of the "accuracy paradox" (Boutaba *et al.*, 2018). These metrics are described in Subsection 3.2.4.

3.1. Research Methods Overview

Good quality research mainly focuses on the approach taken to address the problem under investigation, and the ability to document and demonstrate that the research findings can be reproduced. Moreover, research, in general, must have a sound basis in existing knowledge and theory to ensure that research was conducted.

"...research is used to refer to the activity of a diligent and systematic inquiry or investigation in an area, with the objective of discovering or revising facts, theories, applications, etc. the goal is to discover and disseminate new knowledge." (Ayash, 2014)

Research in the field of Computer Science (CS) and Information Systems (IS) is usually conducted through the application of one or more research methods. (Ayash, 2014) recognised three popular research methods for CS and IS namely: theoretical, simulation, and experimental methods. Other methods include Design Science (Herver et al, 2004), case study (Demeyer, 2011)(Yin, 1994), and surveys (Pfleger and Kitchenham, 2001).

1. Theoretical Simulation

Theoretical research methods are based on classical methodologies since they are related to mathematics and logic. This method is devoted to the algorithm analysis and design to discover solutions or to improve existing solutions. Within all the fields in CS, this method attempts to explain the limitations of computation and the computational paradigms. For example, theoretical methods are used to model a new system, and it can help in the discovery of theories and new mathematical models. However, theoretical methods may still use other methods to demonstrate the efficiency of new theories or models (Ayash, 2014).

2. Simulation

Simulation methods are commonly used in the CS domain since they offer the opportunity to explore systems that are external to the experimental domain or system that is under development or construction. This may include complex occurrences that cannot be realised in a laboratory. Domains that usually adopt simulation include, Astronomy, Physics, Economics, and specialised areas such as, the study of artificial life, virtual reality, or non-linear systems (Ayash, 2014).

3. Experimentation

The experimentation method refers to the task of conducting experiments that will occur in order to acquire results from real-world implementation. Experiments are commonly used to test veracity and theories. CS fields that usually adopt experimentation methods include, ANN, natural languages, automating theorem proving, and analysing performance and behaviours (Ayash, 2014).

4. Case Study

A case study is an empirical inquiry that investigates a contemporary occurrence within a real-life context. It is commonly used when the limitations between the occurrence and context are not clearly evident (Demeyer, 2011).

5. Design Science

Design Science (DS) is a method that strives to extend the limitations of human and organisational abilities by creating new and innovative artefacts (Hevner *et al.*, 2004). The Design Science research methodology incorporates, practices, procedures, and principles necessary to carry out the research with three main objectives: it is consistent with past literature, it offers a nominal procedure model for undertaking research, and it provides a mental model to present and evaluate the research. The frame work of DS as presented by Herver et al.(2004) is hwon in Figure 3.1.

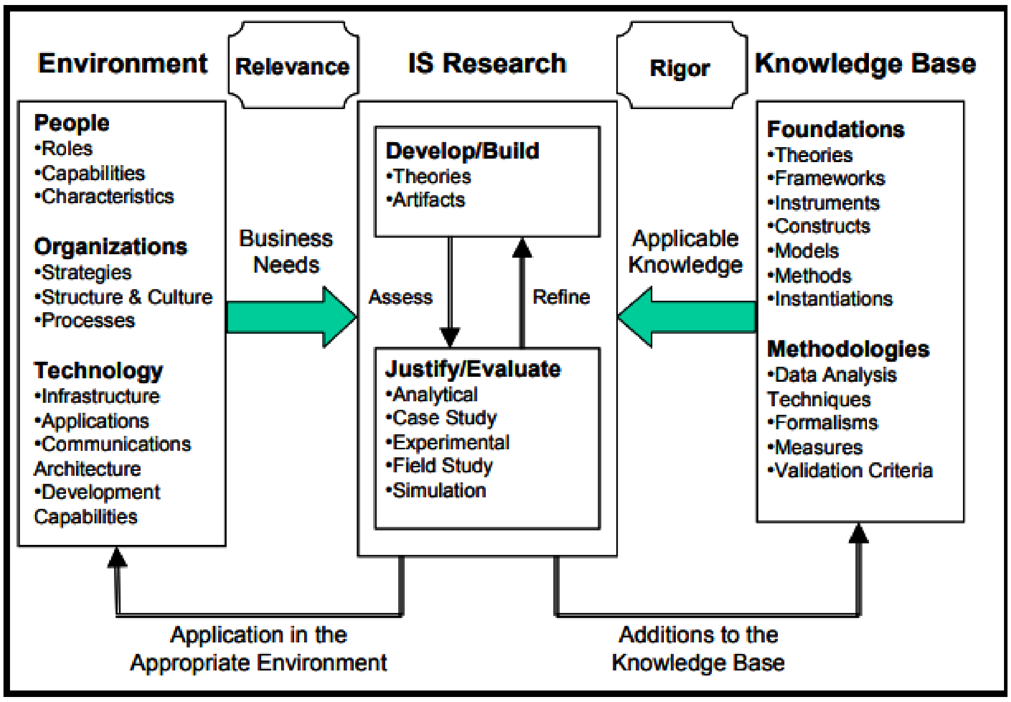


Figure 3.1. Information System DS Research Framework (Hevner et al., 2004)

3.2. Method Selection

In this research, the aim is to investigate and implement an ML-based intrusion detection system in an SDN-based tactical MANET. While conducting research it is important to ensure that the select methods align with the objective of the study. This research adopted two research methods in accordance with the research objectives described in Chapter 1. The first method is the Experimental method. This method was the most appropriate for evaluating different ML algorithms as it provides quantitative properties which can be processed and analysed. This also make it easier to compare the ML performance for suitability in intrusion detection tasks. The second method selected was the Design Science (DS) methodology. DS was adopted because it offers an important paradigm for conducting applicable and yet rigorous research. This approach allow the development, implementation and evaluation of a proposed system. In this study, DS research methods were used to facilitate the development and evaluation of the proposed IDS model for tactical networks. In the following Subsections, how the two selected methods will be used in this study is described.

3.2.1. Experimental Method

This study aims to design a Machine Learning-based IDS for SDN-based tactical MANETs. Machine Learning (ML) techniques have demonstrated applicability in solving intrusion detection problems. Due to a large number of available ML algorithms, it remains unclear which algorithms are suitable for intrusion detection. Hence, it is important to investigate and find a better-performing machine learning algorithm that can be adopted for classifying network traffic as normal or malicious. In this study, performance evaluation of ML classification algorithms is required in order to identify the most suitable machine learning technique for intrusion detection in tactical MANETs. Considering this objective, the experimental method stands out as the best-fitting option.

3.2.2. Design Science Method

To accomplish the goal and objectives of this study, the Design Science methodology was adopted. Design Science was suitable because this work intends to develop a model, therefore DS was a natural choice. DS provides the researcher with an overarching, guiding framework for addressing a complex problem since it involves a rigorous process to design an artefact to solve observed problems, to make research contributions, to evaluate the designs, and to communicate the results to the appropriate audiences (Peffers, Tuunanen, Rothenberger, &

Chatterjee, 2007). To ensure that all the DS criteria are met, seven guidelines were presented by Hevner *et al.* (2004). This study applied these guidelines as depicted in Figure 3.2.

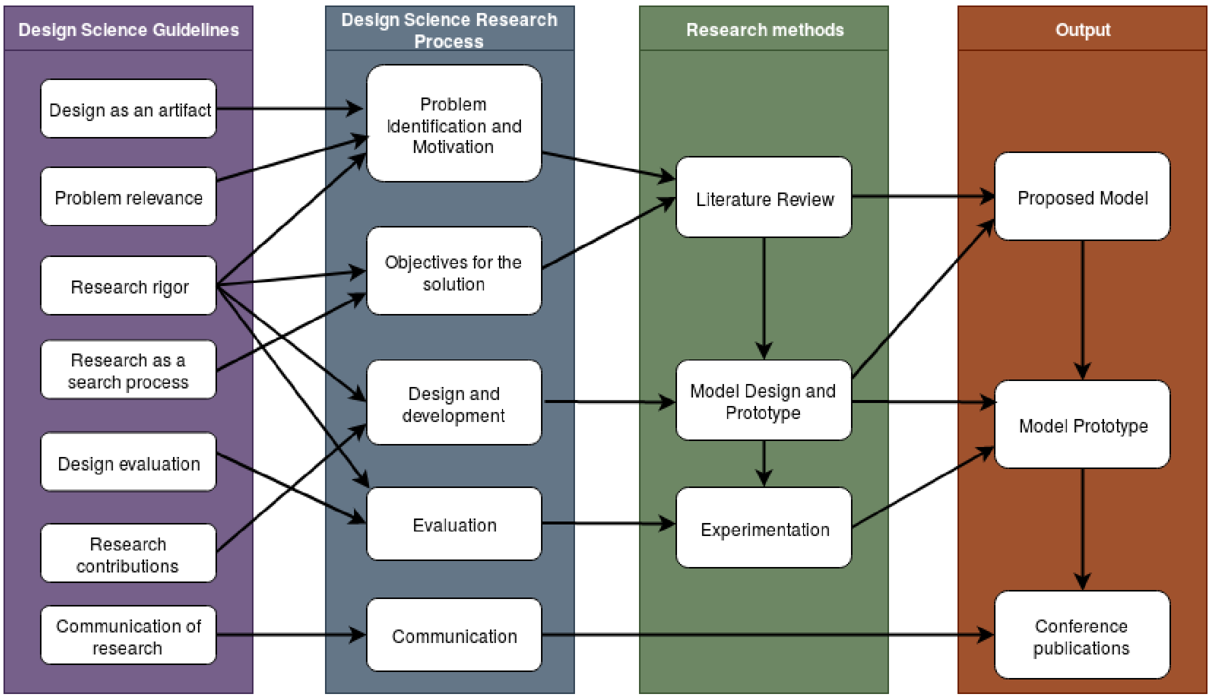


Figure 3.2. Application of Design Science guidelines and research processes

3.2.3. Data Science Method

In Design Science, the produced artefact should be a product of rigorous methods (Hevner *et al.*, 2004). In this study, the data science methodology, also known as the data science life-cycle shown in Figure 3.3, was used to facilitate the construction of the machine learning classification models used to classify network flow data in the proposed IDS. The data science method was chosen because the study seeks to address a data science problem. However, it was used as a sub-method under the DS methodology.

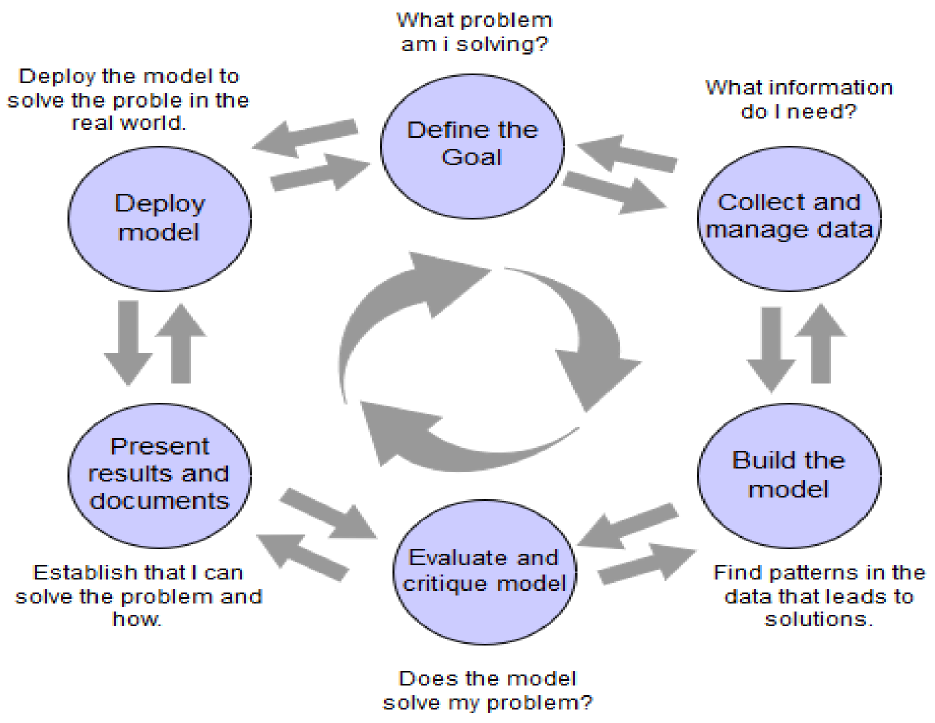

What problem am i solving?
Define the Goal
What information do I need?
Collect and manage data
Deploy the model to solve the proble in the real world.
Deploy model
Build the model
Find patterns in the data that leads to solutions.
Evaluate and critique model
Does the model solve my problem?
Present results and documents
Establish that I can solve the problem and how.

Figure 3.3 Data Science work cycle (Zumel and Mount, 2014)

The data science methodology is composed of different steps that need to be executed to ensure the delivery of an effective Machine Learning predictive or classification model. The steps involved are presented below;

1. **Defining the goal**

Defining the goal of the project forms the most important phase in any project. It is important to define a quantifiable and measurable goal. This phase is concerned with defining state-of-the-art methods used to solve the problem and limitations in the applied methods.

2. **Collecting and Managing Data**

After defining the goal of the project, the next important phase is data collection. The data collection encompasses the identification of the required data, exploring it, and preparing it to be appropriate for analysis. Reports suggest that this step is the most time-consuming over the other processes. In order to achieve our goal, network data is required for building a classifier capable of classifying normal and malicious activities. in this study network flow data is used for intrusion detection. As mentioned in the previous sections, monitoring network flow data for intrusion detection tends to be more effective than the analysis of complete network packets due to a number of reasons. the network flow data can take two possible labels: Normal and malicious. Labelling of the data is important for classifying and learning the different characteristics of each flow type.

3. **Build Model**

This stage employs statistics and Machine Learning for modelling, this is where useful insights from the data are extracted to achieve our goal. Usually, there is an overlap and back-and-forth between this modelling stage and the data cleaning stage. This is due to trying to find the best form and way to represent the data to model so it can produce better results.

4. **Evaluate and Critique Model**

At this stage, we have a model, yet it is time to determine if it meets our goal. The important questions to consider are:

1. Is the model precise and sufficient for our needs? Does it generalise well?
2. Does the model perform better than random guessing? Is it better than the method currently in use?
3. Do the obtained results make sense from the perspective of the investigated problem domain?

It is important that one answers yes to all these questions. If not, one should loop back to the modelling step or decide which of the different data attributes to check and choose more appropriate attributes. To evaluate our classifier the evaluation metrics presented in Subsection 3.2.4 and the confusion matrix, which tabulates the actual classifications against the predicted classifications, were used.

5. **Present Results**

After a back-and-forth loop of model development and evaluation is completed, and the envisioned model is realised, then the documentation of the model is conducted. Such documentation is dedicated to helping those who will deploy the model, hence be liable for using, running, and maintaining the model. Basically, the important things to document are how the model detects network intrusions, and how the end-users will use the model to achieve secure networks.

6. **Deploy Model**

The last stage of the cycle is putting the model into operation. This study implements the intrusion detection model in a wireless SDN-based tactical network, upon deployment, evaluating the effectiveness of the model using evaluation metrics presented in the next section.

3.2.4. Evaluation Metrics

Evaluation metrics allow one to evaluate and validate prediction or classification models. The objective is usually to determine how well a model performs in terms of classification or predicting a particular variable. The most common evaluation metric used to validate classification models is Accuracy. Accuracy is defined as the proportion or ratio of correct predictions among the total number of predictions.

$$Accuracy = \frac{\sum_{i=1}^{N} T_p}{Total\ Predictions}$$

Where T_p is the true predictions for each class, N is the total number of predictions. While accuracy can be a useful metric for ML model validation, in some cases it has its limitations (Boutaba *et al.*, 2018). This issue is known as the "Accuracy Paradox". The Accuracy Paradox states that a "predictive model with a given level of accuracy may have greater predictive power than models with high accuracy".

Another important measure to be considered for model performance evaluation is the model's capabilities in identifying positive cases. This metric is known as the True Positive Rate (TPR) or Recall and is defined as the number of true positives divided by the number of true positives plus the number of false negatives.

$$Recall = \frac{TP}{TP + FN}$$

Where true positives (TP) are the positive data points predicted as positive by the model and false negative are the data points the model identified as negative that are actually positive. Recall can also be regarded as the model's ability to find all the data points of interest in a dataset.

The Precision, which is defined by the number of true positives divided by the number of true positives plus the number of false-positives, will also be used.

$$Precision = \frac{TP}{TP + FP}$$

Precision can be considered as a measure of the ability of the classification model to identify only the relevant data points. While Recall measures the ability to find all relevant instances in the dataset, Precision expresses the proportion of the data points regarded by the model as relevant that are actually relevant, the F1 score defines the harmonic mean of both Precision and Recall, with the equation:

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Another important technique for evaluating a classification model is the Receiver Operating Characteristics (ROC) curve. The ROC curve plots the true positive rate (TPR) on the y-axis and the FPR on the x-axis, where the TPR is the recall, and FPR is the probability of false detection. However, since the ROC curve aids in visual analysis, it can be quantified by calculating the total Area Under the Curve (AUC). AUC is a metric that falls between 0 and 1 and is defined as the measure of the probability of confidence in the model to accurately predict positive outcomes for positive instances. The presented evaluation metrics are used to measure the effectiveness of the proposed IDS technique.

3.3. Summary

To achieve the aim of this study, the Design Science (DS) and Experimentation research methodologies were used. DS was used because it necessitates the application of rigorous methods in both the construction and evaluation of the designed artefact (Hevner *et al.*, 2004). This chapter presented an overview of different IS/CS research methodologies and the most applicable methods to this research were selected and further described. Thus, the study utilised the Design Science (DS) and Experimentation methods. The DS methodology was chosen because it provides guidelines to facilitate an artefact construction and evaluation in addressing an important business or organisation problem. In addition, the Experimentation method was selected due to the nature of the evaluation approach used to evaluate NIDS. Moreover, because the solution proposed in this study uses ML techniques, the Data Science project life-cycle process was adopted to facilitate the development and evaluation of the ML classification models used in the proposed IDS. Finally, the chapter presented the evaluation metrics that will be used to evaluate the efficiency of ML algorithms and the IDS proposed in this study.

Chapter 4: Performance Evaluation of ML Techniques for ID

Although ML techniques have been around for a long time, finding a way to use them efficiently and in real-time is a new trend (Zaman, 2018). Recent approaches in intrusion detection have applied Machine Learning to improve detection rates (Buczak and Guven, 2016) (Vijayanand, Devaraj and Kannapiran, 2018) (Dunning and Friedman, 2014). However, due to the large number of available ML algorithms, it remains unclear which algorithms are suitable for intrusion detection.

In that regard, this chapter explores and conducts a performance evaluation of different ML algorithms for the task of intrusion detection to find the better-performing in terms of detection rate and detection speed. Hence, this chapter addresses the second research objective, which is to find the most suitable ML method for intrusion detection in tactical networks. The first section presents the evaluation datasets used to carry out the experiments. To ensure that the results are accurate and usable, recent datasets such as UNSW-NB15 and CIDDS-001 were used instead of the KDD-CUP benchmark dataset. The choice of datasets was influenced by the number of issues and limitations of the KDD-CUP dataset presented by other researchers (Gogoi, Bhuyan and Bhattacharyya, 2012) (Moustafa, Slay and Technology, 2015). Section 4.2 presents the experimentation and performance evaluation of popular supervised ML algorithms in intrusion detection tasks. The first experiment uses packet-based datasets while the second experiment uses flow-based datasets, this is important to determine which technique yields better performance in packet and flow-based data. The result analysis (Section 4.3) indicates that ML algorithms perform better in flow-based data, achieving minimum model build and test time. In addition, it was also observed that ensemble learning techniques using Decision Tree as their base methods perform better than methods utilising probabilistic and non-probabilistic techniques. Finally, the concluding remarks are presented in the last section (Section 4.4).

4.1 Evaluation Datasets

ML techniques require data to train and test their efficiency. All algorithms used in this work are supervised Machine Learning algorithms, which means they do not only require data but data that is labelled. The most dominant among them is the KDDCUP99 dataset. The KDDCUP99 is a derivative of the DARPA98 network traffic dataset, which is a popular benchmark dataset used in the International Knowledge Discovery in Databases (KDD) competition.

From literature, the most used datasets for evaluating IDS performance are the benchmark network intrusion KDD-CP 99 and NSL-KDD datasets (Ertam, Õ, and Yaman, 2017), (Revathi and Malathi, 2013), (Haq, Onik and Shah, 2015). Those datasets originate from the Lincoln laboratories at MIT University. However, recent studies perceived that using those datasets does not reflect realistic output performance. The works of (Gogoi, Bhuyan and Bhattacharyya, 2012) and (vasudevan2011ssenet) argued that the KDD-CP 99 datasets contain a large number of redundant records in the training set. There are also reports of multiple missing records which are a factor in changing the nature of the data. In addition, in the NSL-KDD datasets which are an improved version of the KDD-CP 99 dataset, (Moustafa, Slay, and Technology, 2015) claimed that the datasets do not comprehensively represent a modern low footprint environment.

Recently, other datasets have been proposed as a benchmark dataset for IDS evaluation (Markus Ring *et al.*, 2017) (Gogoi, Bhuyan and Bhattacharyya, 2012). These datasets ameliorate the shortcomings of KDD-CP 99 datasets because they represent modern network behaviours. In (Moustafa, Slay and Technology, 2015) the UNSW-NB15 network intrusion detection dataset is proposed. The UNSW-BN15 network dataset is a hybrid of modern normal and abnormal network traffic created using the IXIA PerfectStorm tool at the Cyber range lab of the Australian Centre for Cyber Security (ACCS). The UNSW-NB15 network dataset is made up of nine different families of attack instances. In (Markus Ring *et al.*, 2017) the authors proposed the CIDDS-001 dataset which is a labelled flow-based dataset containing unidirectional NetFlow data (Verma and Ranga, 2018). The dataset consists of data extracted from an OpenStack environment with internal servers and an external server that is deployed on the internet to capture real and up-to-date traffic.

In this study, the UNSW-NB15 and CIDDS-001 datasets were used to evaluate and determine the better performing ML algorithm. The UNSW-NB15 dataset was used since it is one of the most recent packet-based datasets, containing a broad range of network packet attributes. To evaluate flow-based prediction on the algorithms, the CIDDS-001 dataset was also used as it is recent, easily accessible, and contains a range of attacks.

4.1.1. UNSW-NB15 Dataset (Moustafa, Slay and Technology, 2015)

The UNSW-NB15 dataset is a fairly recent packet-based dataset that contains a hybrid of real modern normal and synthetical abnormal network traffic from a synthetical environment at the UNSW cybersecurity lab. The UNSW-NB15 dataset represents nine major families of attacks achieved by utilising the IXIA PerfectStorm tool. It contains 49 features that were developed using Argus and Bro-IDS tools and twelve algorithms that cover characteristics of network traffic. The dataset was adopted because it reflects modern traffic patterns since it was generated more recently when compared to the benchmark dataset, KDD-CUP 99., the UNSW-NB15 network dataset distribution is presented in Table 4.1. see (Moustafa, Slay and Technology, 2015).

Table 4.1. UNSW-BN15 network dataset distribution

Type	**Number of Instances**	
	Training	**Testing**
Normal	37 000	56 000
Fuzzers	6 062	18 184
Backdoors	583	1 746
Analysis	677	2 000
DoS	4 089	12 264
Exploits	11 132	33 393
Generic	18 871	40 000
Reconnaissance	3 496	10 491
Shellcode	378	1 133
Worms	44	130
Total	82 332	175 341

4.1.2. CIDDS-001 Dataset (M Ring *et al.*, 2017)

The CIDDS-001 dataset is a labelled flow-based dataset generated by emulating a small business environment in the OpenStack Software platform and capturing the generated network traffic of virtual machines in unidirectional NetFlow format over a period of 4 weeks. The Network flow traffic was recorded in flow-based format instead of packet-based format to bypass the problem of encrypted connections. The normal traffic was generated by executing Python scripts on the clients which follow some self-defined guidelines, while the malicious traffic was generated by explicitly executing Ping-scans, Port-scans, Brute-force, and DoS attacks within the OpenStack environment. Other malicious traffic was captured from a server that was exposed to real and up-to-date attacks from the internet. containing unidirectional NetFlow data (Verma and Ranga, 2018). Table 4.2. presents the overview of attacks within a specific week within the CIDDS-001 dataset.

Table 4.2 Overview of attacks within the CIDDS-001 Dataset (Markus Ring *et al.*, 2017)

	OpenStack				**External Server**			
	PortScan	PingScan	DoS	BruteForc	PortScan	PingScan	DoS	BruteForc
Week 1	16	10	11	5	0	0	0	0
Week 2	8	6	7	7	2	0	0	4
Week 3	0	0	0	0	5	0	0	7
Week 4	0	0	0	0	1	0	0	3

4.2. ML Techniques Performance Evaluation

This section presents a performance analysis of different ML techniques to determine the most effective in terms of its ability to find all the data points of interest and its ability to identify only the relevant data point in the dataset.

4.2.1. ML Classifiers in Packet-based dataset

Classification is defined as a simple task in data analysis and pattern recognition that necessitates the creation of a classifier, which is, a function that assigns a class label to instances described by a set of attributes (Friedman et al, 1997). In the experiment, seven machine learning classifiers are considered. Multi-Layer Perceptron, Bayesian Network, Support Vector Machine (SMO), AdaBoost, Random Forest, Bootstrap Aggregation, and Decision Tree (J48).

To conduct a performance analysis of the classification algorithms, WEKA version 3.8 was installed on a Windows 10, Intel(R) Core i7-6700 CPU @ 3.40GHz machine, with 8 GB RAM. The dataset used for the evaluation is the UNW-NB15 network dataset proposed by (Moustafa, Slay and Technology, 2015). The classifiers are used to train intrusion detection models using a training set with 82,332 instances and tested using 175,341 unknown instances. The dataset is loaded to the WEKA tool. Data pre-processing is used to restructure and remove features that may cause overfitting or no information gain. The training and testing sets contain 45 attributes each, which include two types of labels; original *Class* labels *such as* 0 for normal and 1 for abnormal and *attack_Category* labels where each abnormal instance is labelled based on attack type, see Table 4.1. Thus, in this experiment, we use pre-processing tools to remove the ***attack_Category*** label and the instance ID to avoid data overfitting. Therefore, we remain with 43 attributes with 0 and 1 as our labels in our final evaluation dataset. The experiment was repeated 6 times for each classifier, and the average was used to ensure consistency.

4.2.2. Results

Table 4.3. presents the True Positive Rate (TPR), False-Positive Rate (FPR), and Area Under ROC Curve (AUC) of each classifier model using two-class labels (Normal and Attack). It is important to note that the higher the TPR, the fewer positive data points missed. Therefore, classifiers that achieve higher TPR are preferred over the ones that generate lower TPR. From the results above, MLP, AdaBoost, Random Forest, and Bagging generated higher TPR. For intrusion detection in TWN, a classifier with high TPR is required so as to correctly classify malicious traffic in the network. From Table 4.3 we observe that multilayer perceptron missed fewer positive data points, achieving TPR of 0,933. The second is AdaBoost with 0,903, random forest at 0,902 and bootstrap aggregation at 0,901. The remainder of the classifiers achieved TPR of less than 90 %.

Table 4.3. Classifier TPR, FPR, AUC

Classifier	Experimentation values		
	TPR	FPR	AUC
Multilayer-Perceptron	0.933	0.142	0.951
AdaBoost	0.903	0.087	0.968
Random Forest	0.902	0.057	0.981
Bagging	0.901	0.057	0.952
J48	0.887	0.069	0.952
SMO	0.882	0.073	0.905
Bayesian Network	0.809	0.123	0.965

The second important performance metric is the FPR. A higher FPR indicates that more negative data points are misclassified by the classifier. Multilayer perceptron has higher FPR followed by the Bayesian Network, this demonstrates that those two classifiers can produce more false-positive predictions than the other classifiers. Therefore, using those classifiers for intrusion detection in TWN is not recommended. For example, consider a dataset with two classes, positive and negative. If the data has one category overwhelming the majority of the data points, then the data is skewed class-wise. In this case, the data is imbalanced as one class has many data points from the other. Thus, accuracy is not reliable in such cases because if a dataset has 15 positive points and 5 negative points, correctly predicting all the positive points and failing to correctly predict any negative point yields accuracy of 75%. In this case, accuracy is not a good measure to assess model performance. This issue is usually referred to as the Accuracy Paradox.

When we consider the FPR, Multilayer Perceptron has a higher FPR followed by the Bayesian Network, with 0.142 and 0.123 respectively. We observe that the other classifiers have an FPR of less than 0,1. AUC is one of the most important metrics to measure classifier performance.

The last detection metric that can be used to measure detection performance is the AUC. As observed in the above subsection, Bootstrap, random forest, and AdaBoost achieved higher AUC. Hence these classifiers are suitable for implementation for intrusion detection. While TPR and FPR are important, it is important to consider the AUC as the best measure for selecting an ML classifier for TWN. Table 4.3 shows that all the models achieved AUC above 0.9. However, Bootstrap aggregation has the highest AUC (0.986), followed by Random forest at 0.981, and AdaBoost at 0.968.

The time for training a model is an essential factor due to the ever-changing cyber-attack types and features. Hence, anomaly detectors need to be trained frequently or incrementally, with fresh malware signature updates.. Multilayer-Perceptron (MLP) took longer to build the model, followed by SMO. MLP took about 24 hours to train, followed by SMO with an average of 9 minutes. Random forest and Bootstrap Aggregation with a build time of 28,68 sec and 18,86 secs follow. The remainder of the list is all bellow 10 seconds, as shown in Table 4.4.

Table 4.4 Classifier Build and Test time in seconds

Classifier	Time in seconds	
	Build (sec)	Test (sec)
Multilayer-Perceptron	86303.22	14.35

Classifier	Time in seconds	
	Build (sec)	Test (sec)
AdaBoost	9.02	1.06
Random Forest	28.68	4.91
Bagging	18.86	1.19
J48	8.08	1.02
SMO	541.88	1.44
Bayesian Network	2.56	1.63

When we consider test time, the Multilayer Perceptron takes more time, 14,35 seconds, followed by random forest, 4,91 secs. The remainder of the algorithms were able to test their models in below 2 secs, shown in Table 4.4.

On the other hand, the time to classify a new instance is an important factor that reflects the reaction time and the packet processing power of the intrusion detection system. Hence from the results, we observe that MLP can present challenges for real-time intrusion detection in TWN. This is due to a large amount of time required to build and test the MLP classifier.

From the obtained results, Random Forest, Bagging, AdaBoost, and J48 are the best performers in terms of TPR, FPR, and AUC. It should be noted that 3 of these classifiers are ensemble classifiers. In Addition, AdaBoost, Bagging, J48, and Bayesian Network classifiers, are much faster in building and testing their model.

4.2.3. ML Classifiers in Flow-based dataset

This subsection presents a performance evaluation of ensemble learning methods using flow-based data. This is because ensemble Machine Learning methods have distinguished themselves as exceptional detectors of malicious and anomalous actions in intrusion detection in both computer systems and networks. Ensemble methods demonstrated better performance when compared to single and hybrid ML methods in terms of detection accuracy, as observed from the performance analysis conducted in the previous subsection. Thus, three representative learning methods that are widely used for classification problems are used as the base machine learning methods for the ensemble methods. The base methods utilised are; Decision Tree (DT), Support Vector Machine (SVM), and Naïve Bayes (NB).

The experiment is conducted on a 1.80GHz Intel (R) Core i7 processor with 8 GB RAM. Python 2.7 and Scikit-learn version 0.20 installed on Ubuntu 16.04 mate Operating system. The performance analysis is conducted using the CIDDS-001 flow-based network intrusion detection datasets (Markus Ring *et al.*, 2017). The dataset consists of 14 attributes, where 1-10 are the NetFlow default attributes, and attributes 11 – 14 are attributes added during the labelling process. This study utilised 11 attributes for the ensemble learning evaluation since those attributes can be sampled from any simulation tool with ease. Also, only week 1 of the dataset (see Table 4.2) was used since it contained more attacks than the other weeks. CIDDS-001 dataset attributes shown in Table 4.5.

Table 4.5 CIDDS-001 network intrusion datasets attributes

No	**Attributes**	**Attribute Description Subhead**
1	**Src_IP_Addr**	Source IP address
2	**Dst_IP_Addr**	Destination IP address
3	**Src_port**	Source port
4	**Dst_port**	Destination port
5	**Proto**	Transport protocol
6	**Date_first_seen**	Start time flow first seen
7	**Duration**	Duration of flow
8	**Bytes**	Number of transmitted bytes
9	**Packets**	Number of transmitted packets
10	**Flags**	OR concatenation of all TCP flags
11	**Class**	Class label
12	**Attack_type**	Type of attack
13	**AttackID**	Same attack class carry same attack id
14	**Attack_description**	Additional information about attack parameters

4.2.4. Results

Table 4.6 shows the measures of the base classifiers and when used in the ensemble methods. The better-performing classifier in terms of detection accuracy is the Decision Tree (DT) classifier and random forest (RF) classifier, with an accuracy of 99.09 % and 99.14 % respectively. The probabilistic algorithm, Naïve Bayes (NB), obtained the worst performance, with an accuracy of 60.56 %. Majority voting performed slightly better than SVC, with an accuracy of 63.44 %. While considering the base classifiers in terms of accuracy, the DT-based classification method demonstrates better performance when compared to the other methods.

Table 4.6. Evaluation results for ensemble learning techniques

No	**Algorithm**	**Performance of ML**			
		Accuracy	***Precision***	***Recall***	***f-score***
1	DecisionTree	99.09 %	0.99	0.99	0.99
2	NaiveBayes	60.56 %	0.47	0.61	0.48
3	SVC	62.89 %	0.75	0.63	0.49
4	Bagging(DT)	99.08 %	0.99	0.99	0.99
5	Bagging (NB)	60.57 %	0.47	0.61	0.48
6	Bagging (SVC)	62.89 %	0.75	0.63	0.49
7	AdaBoost (DT)	99.15 %	0.99	0.99	0.99
8	AdaBoost (NB)	70.74 %	0.84	0.71	0.75
9	AdaBoost (SVC)	62.35 %	0.39	0.62	0.48
10	RandomForest	99.14 %	0.99	0.99	0.99
11	MajoritytVoting	63.44 %	0.76	0.63	0.50

The three base classifiers; DT, NB, and SVC were utilised to construct the bootstrap aggregation (Bagging) ensemble learning method, a bar graph representing their performance is shown in Figure 4.1.

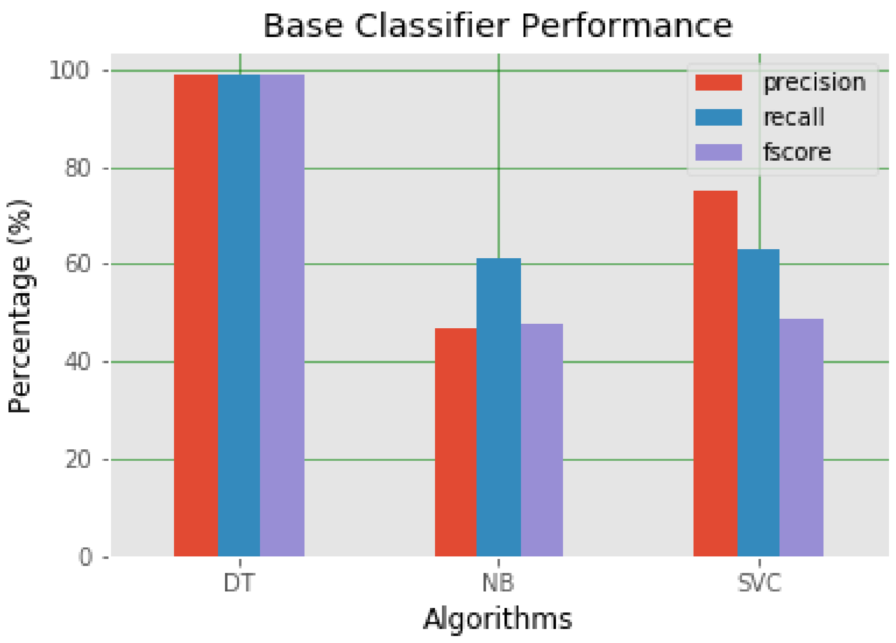


Figure 4.1 Base method accuracy

In Figure 4.2, a bar graph representing Bagging (DT), Bagging (NB), and Bagging (SVC). From the figure, it is clear that Bagging (DT) outperforms the other two ensemble methods in terms of detection accuracy. Bagging (SVC) performs slightly better than Bagging (NB)

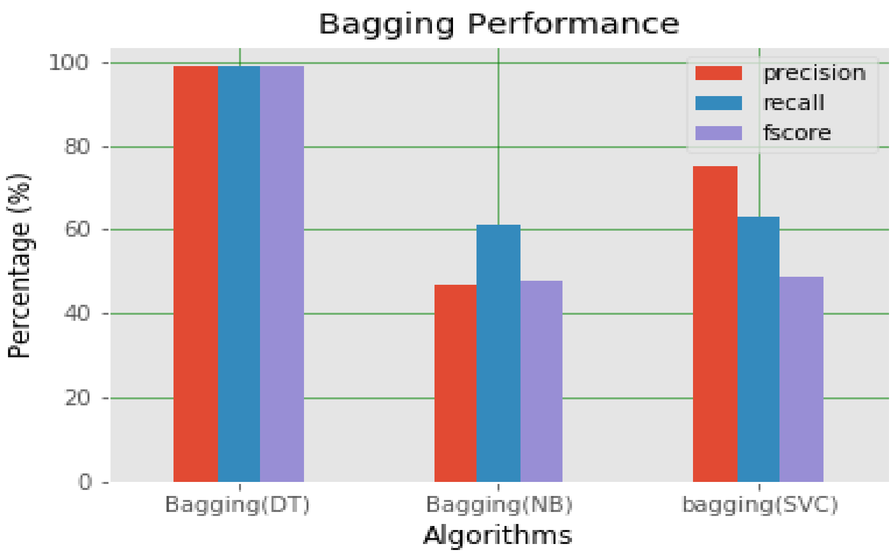


Figure 4.2 Bagging DT, NB, and SVC accuracy

In terms of AdaBoost ensemble learning the three base classifiers were implemented as AdaBoost (DT), AdaBoost (NB), and AdaBoost (SVC), as shown in Figure 4.3. The results indicate that methods utilising the Decision Tree algorithm tend to have better detection accuracy, such as 99 % accuracy obtained by AdaBoost (DT). AdaBoost (NB) outperforms AdaBoost (SVC) with accuracy of 70.74% over AdaBoost (SVC) accuracy less than 64 %.

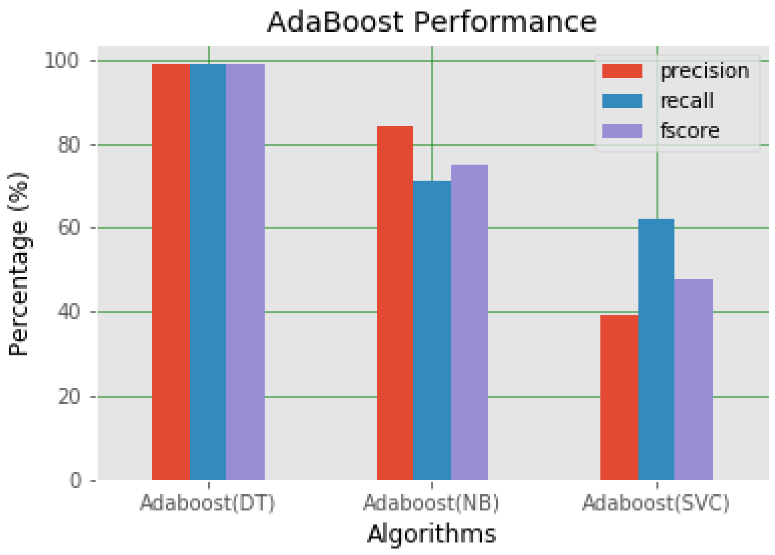


Figure 4.3 AdaBoost (DT), (NB), (SVC) accuracy

The results obtained from ensemble learning also show that Decision Tree-based classification methods before better than probabilistic and non-probabilistic methods. Non-probabilistic achieve slightly better results than the probabilistic-based classification method. In both Experiments, ensemble learning techniques demonstrated high detection rates and high AUC. Among the ensembles, the highest accuracy was obtained by Random Forest and AdaBoost with Decision Tree as the based learner. Figure 4.4 illustrates the ROC curve of different ensemble methods, namely: Random Forest (RF), Decision Tree (DT), gradient boosting, and AdaBoost. The methods obtained ROC values above 90 % which indicate that ensemble methods are powerful and can help detect network anomalies and for intrusion detection.

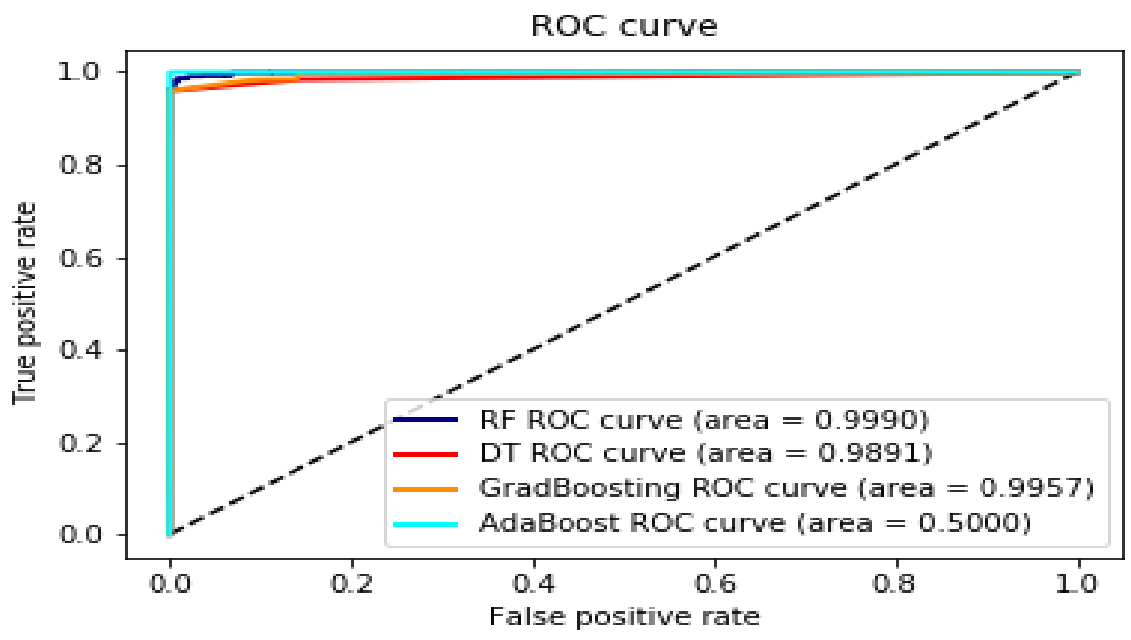


Figure 4.4 Ensemble learning techniques ROC Curve Plot

As presented by Figure 4.5, a closer look at the ROC curve illustrates that AdaBoost and Random Forest outperform the other methods by obtaining higher AUC values.

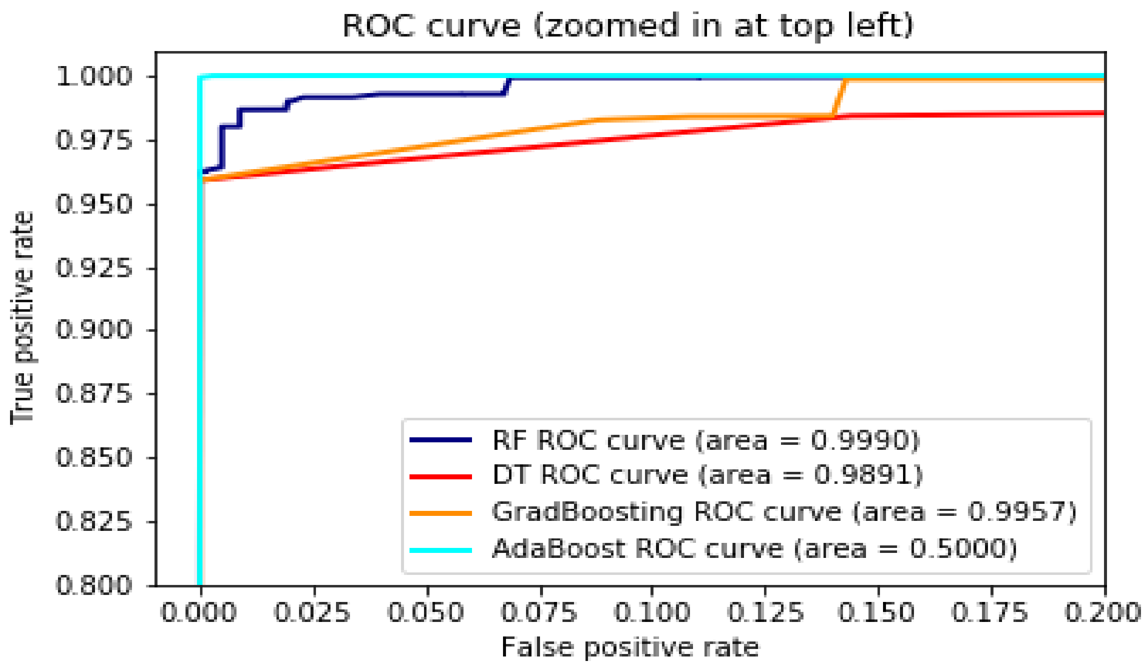


Figure 4.5 Zoomed image of Figure 4.4

Below the results for time taken for testing and bulding the machine learning models is discussed.

Decision Tree (DT) based methods: Methods utilising the Decision Tree algorithm tend to have better detection accuracy and demonstrate better performance in terms of correctly classifying malicious flows. On the other hand, DT methods take the least time when both building and testing their models. DT approaches outperform both probabilistic and non-probabilistic approaches. See Figure 4.6 below.

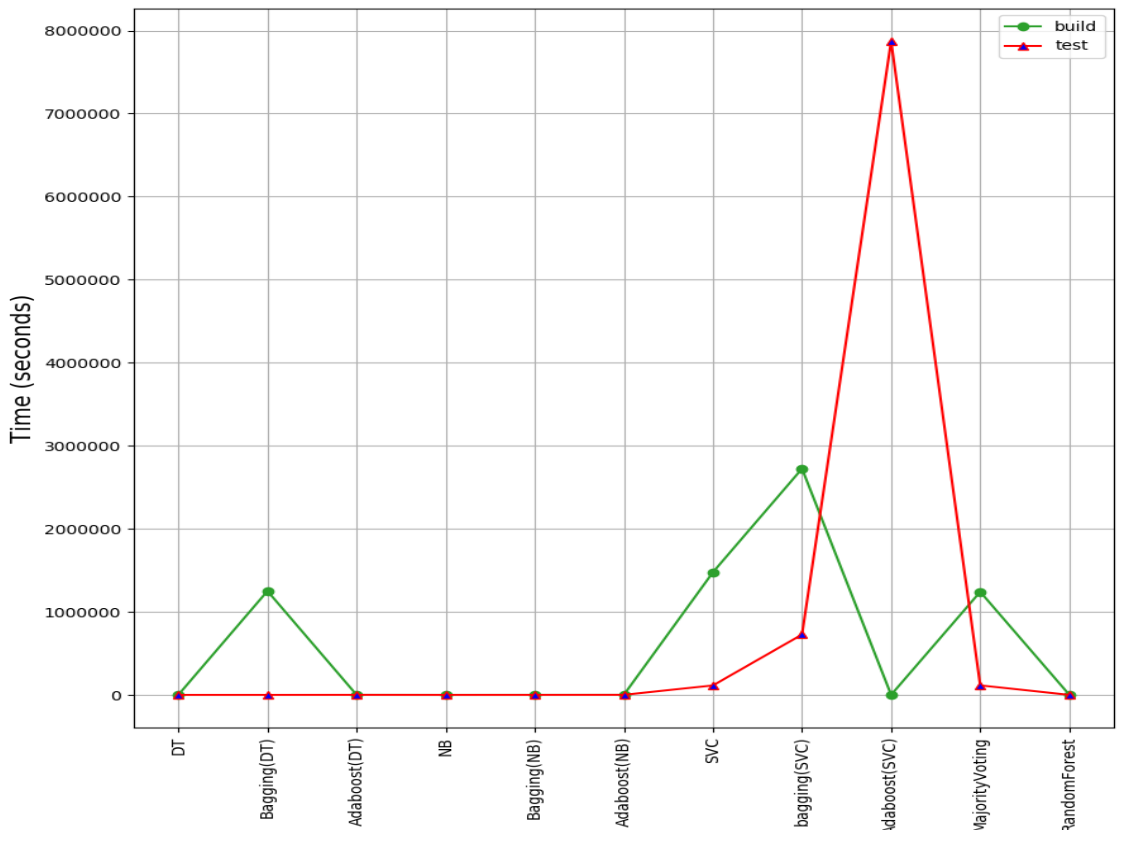


Figure 4.6 Classifier build and test time

Probabilistic (Naïve Bayes) based methods: Methods utilising NB demonstrated the worst performance in terms of detecting malicious flows. However, the time to train and test the model is much more reasonable than for the non-probabilistic approach utilising SVC.

Non-probabilistic (SVC) based methods: The results indicate that methods associated with Support Vector Machines perform better than probabilistic approaches in terms of flow classification. In addition, it takes longer to build and to test them, hence they are the worst methods to employ when the time is a critical issue.

4.3. Result Analysis

A summary of the results obtained from both experiments is illustrated in Figure 4.7. where the red bar indicates the accuracy of a classifier in packet-based datasets, and the blue bar the accuracy in flow-based datasets. From the Figure, it is apparent the algorithms obtained higher accuracy in the flow-based datasets. This is because flow data contains a minimum number of feature attributes compared to packet-based datasets. This means they are more lightweight for processing which fulfils the desired goal of efficiency in processing power and storage.

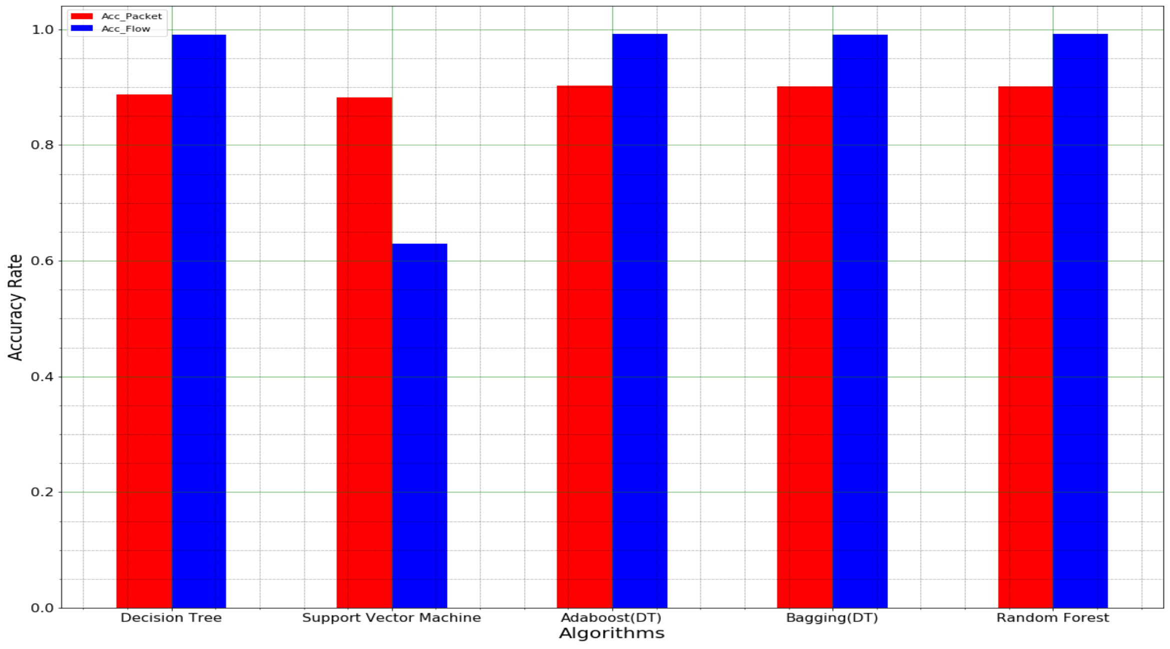


Figure 4.7 Packet and flow-based machine learning classifier performance

The time each model takes to learn from the data, and the time it takes to classify an instance are very important metrics to consider in fast-changing networks with real-time security requirements. Flow-based data analysis using ensemble learning methods demonstrated fast performances in build and test time, Figure 4.8, and Figure 4.9. In addition, random Forest and AdaBoost have less build time in flow-based data compared to the Bagging ensemble classifier.

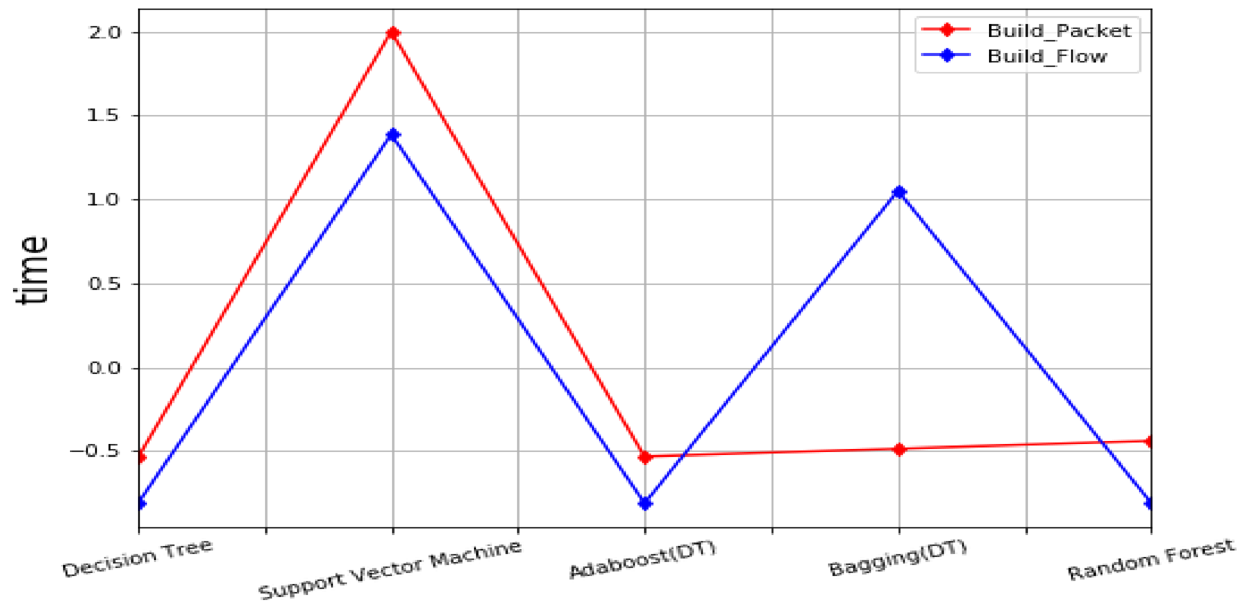


Figure 4.8 Model build time for packet and flow-based data

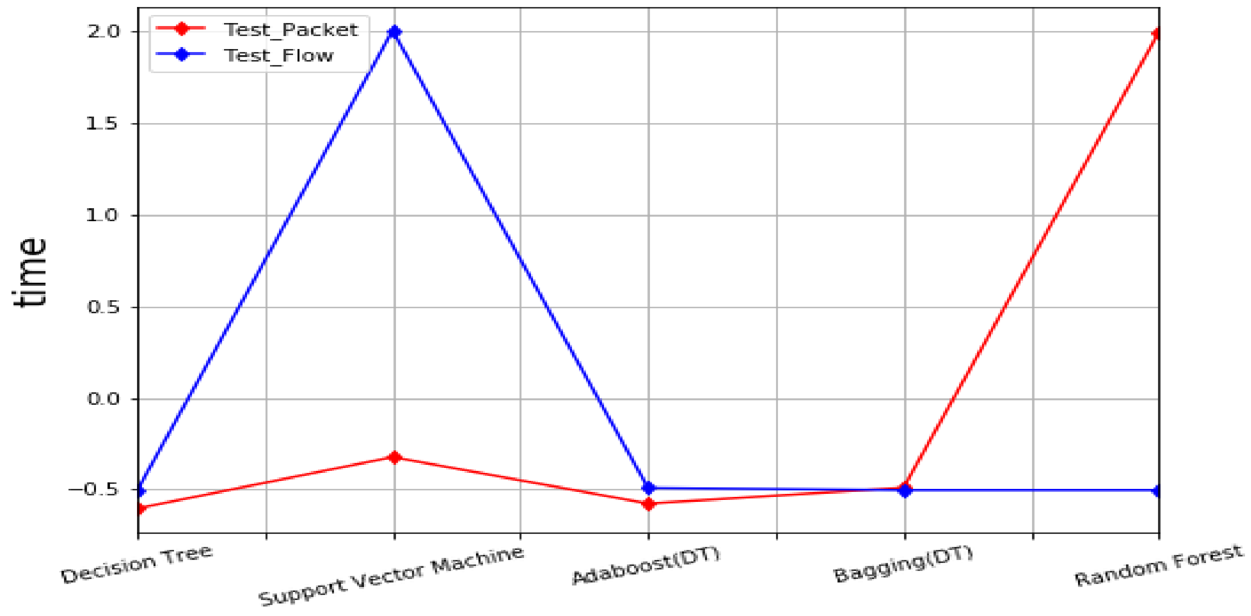


Figure 4.9 Model test time in packet and flow-based data

Conversely, when we consider the test time, presented in Figure 4.9, which is the amount of time taken by the method to classify new instances, Random Forest and AdaBoost ensemble techniques demonstrated better performance. Both the better identified best performing models, that is, AdaBoost and Random Forest were further implemented for classifying network data in a simulated SDN based tactical mobile network (presented in Chapter 7).

4.4. Concluding Remarks

To develop an IDS using Machine Learning, an ML algorithm capable of detection totality and allowed to train and retrain in a minimal amount of time is ideal. In this chapter, a performance evaluation of different machine learning methods was conducted. Our approach evaluated the algorithms using two datasets, namely: 1) the UNSW-NB15 network datasets, which are packet-based, and 2) the CIDDS-001 flow-based datasets. The packet-based evaluation approach indicated that Decision Tree-based methods outperformed the probabilistic and non-probabilistic techniques in terms of accuracy, False-positives, and AUC. Taking into consideration the model build time, the AdaBoost using Decision Tree demonstrated better performance, while Random Forest performed worst in test time.

The flow-based evaluation implemented Decision Tree (DT), probabilistic (Naïve Bayes), and non-probabilistic (Support Vector Classifier) methods using Bagging and AdaBoost ensemble methods. The results obtained indicated that DT-based methods also performed better for flow-based intrusion detection systems. They performed better in terms of Accuracy, Precision, Recall, and F-Score. The time taken by the Decision Tree algorithm to build and test a model was also reasonably small when compared with the probabilistic and non-probabilistic methods.

From the experiments, we observed that ensemble learning-based techniques performed better than single ML techniques. Comparison of the ensemble techniques, the Random Forest, and AdaBoost ensemble methods with Decision Tree as the base estimator demonstrated suitability for intrusion detection in both the flow-based and packet-based network datasets.

This chapter demonstrated that better results can be obtained when using a flow-based dataset, hence making them a suitable choice for IDS in TWN. The next chapter provides the proposed SDN enabled flow-based IDS.

Chapter 5: Software-Defined Flow-based Intrusion Detection System (SFIDS)

The brief description of tactical networks and their limitations provided in Chapter 2 indicate that there is a need for an innovative networking paradigm for tactical network scenarios. Currently, Software Defined Networks (SDN) can provide many benefits such as network global view, programmability, and centralised control and management. Those capabilities can be used to enhance tactical networks' limitations, for example, network global view can be used for data collection which presents the opportunity for better analysis of the network traffic in a tactical network for intrusion detection. In addition, due to the velocity, variety, and volume of the flow-data that can be acquired, big data methods can be used to power data-driven applications dedicated to different aspects of the network, including security solutions and IDS (Fahad, Sher and Bi, 2017). This Chapter proposes a Software-defined Flow-based Intrusion Detection System (SFIDS) model that uses SDN for data collection and ML techniques for intrusion detection.

The presented SFIDS method is directly driven by the high and growing demand for network security in tactical networks as existing solutions experience challenges due to the hostile environment, limited power budget, and lack of centralised control and management units where security services and mitigation strategies can be provisioned on demand. The core components of the SFIDS include packet observation, flow metering and exporting, data collection and preparation, finally data analysis. These components utilise the SDN architecture to fulfil their respective tasks. For example, packet observation, metering, and data exportation are handled in the SDN data layer. Data collection and preparation is handled by a centralised collector residing in the SDN control layer. The ML models (Classifiers) are then developed into SDN applications taking prepared data from the collector as input to analyse and classify each data instance as either malicious or normal.

5.1. Design Criteria

This study proposes a Software-defined Flow-based IDS (SFIDS) that uses Machine Learning (ML) techniques to address the high demand for network security in military tactical networks. The motivation for this approach is the demand for an intrusion detection technique that can precisely detect hostile nodes in a hostile environment (Pawgasame and Wipusitwarakun, 2015). In addition, because tactical network nodes have low processing and power budgets, a light-weight mechanism that can achieve a high detection rate using minimum network resources is required. The proposed approach adopts the SDN paradigm for data acquisition, network global view, and real-time data analysis without compromising network lifetime and network resources. The approach can be extended to support incident handling and containment, which is regarded as important after network intrusions are detected. This section presents the design criteria or functional requirements of the proposed system. Based on (Metcalf and Lapadula, 2000), the functional requirements necessary for intrusion detection systems in a military context include; collecting, processing, analysing, reporting, warning, displaying, controlling, reacting, storing, and interacting. In that regard, the design criteria that were laid out for the SFIDS to meet these functional requirements are as follows:

- The framework must be able to acquire or sample network flow-stats data in real-time. This can be achieved by using SDN and network flow sampling tools, such as sFlow, where a network flow is sampled by a sampling agent embedded in the SDN forwarding devices. Each network forwarding device samples flow-stat information and sends it to a logically centralised collector for processing and analysis.
- The process of data gathering should be lightweight so as to not affect network processes, functions, and lifetime. The framework should be able to gather network flow stats while using minimal or limited network resources.
- The framework must be able to perform reduction and pre-processing of sampled flow data for classification. This forms one of the most important features of the framework, as all the acquired data needs to be processed and presented in a specific format which will allow the Machine Learning algorithm to effortlessly learn and detect any anomalies.
- The framework must be able to classify network flow instances as normal or malicious with high detection rates and recognition totality. Hence, it must be able to achieve high detection accuracy with low or no false-positive rates. This will be achieved by using Machine Learning techniques with high detection rates such as ensemble learning techniques, as established in Chapter 4. Each flow instance is sent to the ML model for classification and labelling as normal or malicious.
- The framework should be able to generate timeless alerts if a malicious flow is detected. Generating alerts will help network administrators to react fast in terms of incident handling and incident containment. This can be done by deploying an SDN application on the

controller that can directly update flow table rules in the forwarding devices to block, drop or forward malicious traffic to a sandbox to monitor the malicious node.

- The alerts generated by the framework must be easy to understand and implemented so that it does not overload the operator and system.
- The alerts generated by the framework must be exportable to data visualisation frameworks, such as the ELK stack and databases for long term storage. Each alert must be accompanied by the malicious node's information such as MAC address, IP addresses, and protocol for effortless analysis.

5.2. SFIDS Model Architecture

This section presents the proposed Software-defined Flow-based Intrusion Detection System (SFIDS) design. Design Science propounds that artefacts have both inner and outer environments. The inner environment deals with components that make up the artefact. While the outer environment refers to forces external to the artefact (Michalos and Simon, 1970). This section describes the SFIDS model and its components as applicable in a tactical network scenario.

In this research, we proposed an intrusion detection system (SFIDS) which uses network flow data to detect anomalies, Figure 5.1. In general, SFIDS employs network flow sampling techniques to acquire network flow data from the tactical network devices and ML techniques to analyse the flow data and generate alerts if intrusive flows are detected. SFIDS is made up of four essential components, namely; Packet Observation component, Flow Metering and Export component, Data Collection component, and Data Analysis component.

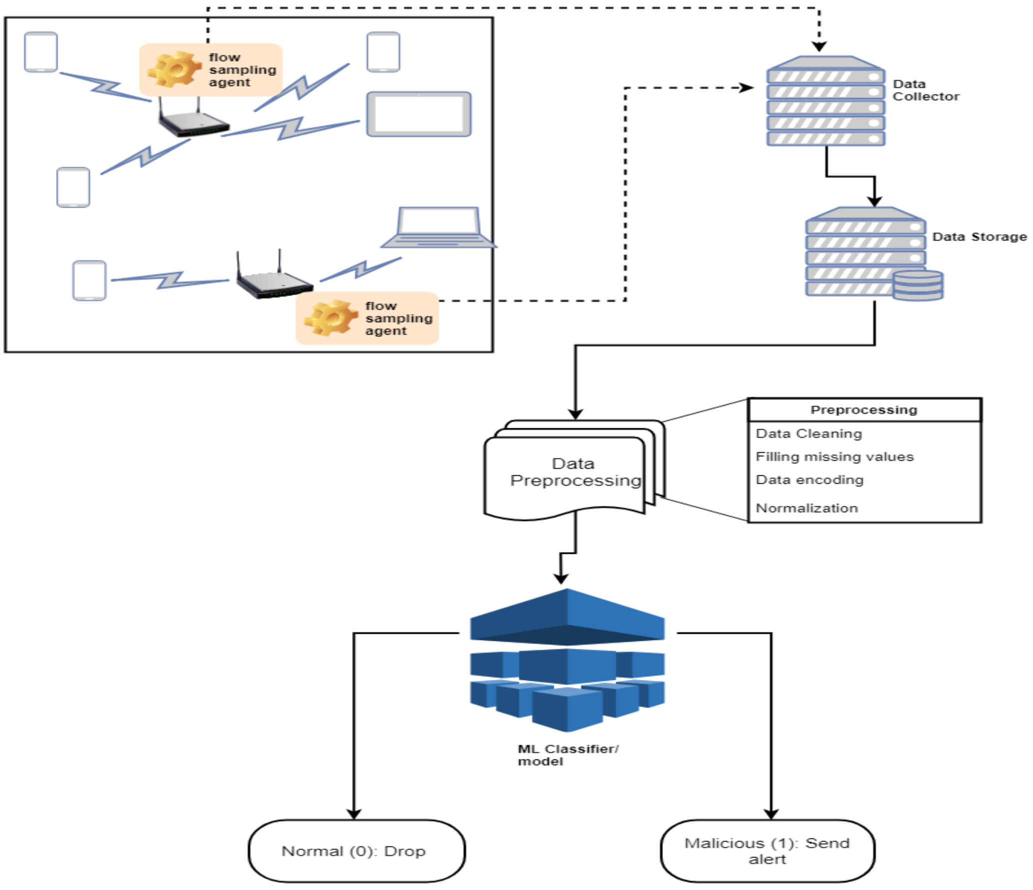


Figure 5.1 Proposed Software-defined Flow-based Intrusion Detection System (Zwane, Tarwireyi and Adigun, 2019a)

5.2.1. Packet Observation

The first step in the proposed model is packet capture which is carried out by a Network Interface Card (NIC) in the network devices. Each network device is embedded with a flow-sampling agent. During this stage, packets pass several checks, which include checksum error checks. The packet is then timestamped, which is important for processing functions and analysis application. After packet capture and timestamping, packet truncation is applied to reduce the volume of data received and handled by the capture application. The last step involves packet sampling and filtering. Sampling is applied to reduce the load for subsequent stages and to moderate the consumption of bandwidth, memory, and computation cycles. Similarly, filtering is used to lessen the amount of data to be processed at later stages.

5.2.2. Flow Metering and Export

During this stage, packets are aggregated into flow records. The flow records are then exported. Packet aggregation is accomplished through a metering process which is based on Information Elements (IE) that define the layout of flows. IE are fields that can be exported in flow records, for example, flow attributes, such as an IP address. After the metering process flow record sampling and filtering functions are performed. In contrast to packet sampling and filtering performed in the packet observation stage, flow sampling and filtering work on flow records instead of packets. Flow records are packaged into a specific message format depending on the protocol used. For example, the IPFIX message format or NetFlow format. After constructing the message, it is then exported to the logically centralised flow collector. The most implemented and deployed transport protocol for exporting such flows is UDP (Hofstede *et al.*, 2014).

5.2.3. Data Collection and Preparation

Flow records are exported to a logically centralised flow collector, which receives, stores, and pre-processes flow data from one or more flow exporters in the network. The flow collector conducts feature extraction which includes the task of picking the optimal features that will be used by the model to successfully classify the records. Data pre-processing is the process of converting flow records into a specific format that is acceptable to the detection algorithm used. This phase can include data cleaning, fixing missing values, data encoding, and normalisation. In this component, all the features of each flow record from the data collector are encoded and scaled. This allows the data analysis to be consistent while using less processing power, as required in power-constrained environments. Finally, the Collector exports the data for storage and for pre-processing, see Figure 5.1 above.

5.2.4. Data Analysis

At this stage, the results of all the previous stages come together. In the data analysis stage different data analysis methods can be applied, for example, flow analysis and reporting, threat detection, and performance monitoring (Hofstede *et al.*, 2014). The task of intrusion detection is considered a classification problem since the goal is to classify network data as normal or malicious. One way to achieve this is by directly predicting the qualitative response for the observations (James, 2014). Since the task at hand requires the analysis and prediction of data instances, it can be regarded as a data science problem.

However, from the experiments conducted in Chapter 3, we have observed that Decision Tree-based methods usually outperformed both probabilistic and non-probabilistic methods in terms of detection accuracy, model build, and test time. In other cases, Machine Learning models can yield unsatisfactory results due to a number of issues, for example, a single ML method is only capable of learning only some parts of patterns in the data. Thus, by suitably combining multiple learning techniques, such as every single learner being trained using a subset of the data, and their predictions being combined using an ensemble method for the final prediction, high detection rates can be achieved. This ensemble approach is presented in Figure 5.2.

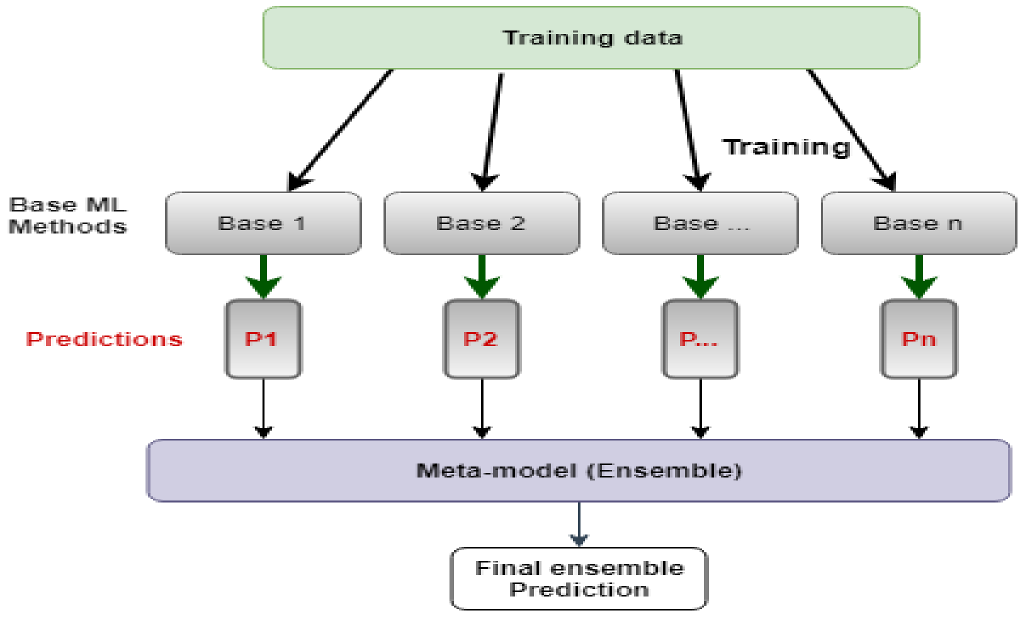


Figure 5.2. Generalized procedure for creating an Ensemble or a Meta- Model

To improve the classification accuracy, this research proposes combining single ML algorithms to build an ensemble model. This approach requires that two conditions are met (Illy *et al.*, 2019);

- The first condition requires that we have base learners or learning algorithms that perform better than random guessing or are reasonably accurate in their domain of proficiency. In this study, the Decision Tree classifier is used as the base learner.
- The second condition defines how to combine the output of several base estimators to produce the final result. The methods of combining the base methods’ results are divided into multi-expert and multistage combinations (Illy *et al.*, 2019). During a multi-expert combination, the base estimators work in parallel, and all of the selected outputs are used to generate the final result. For example, the Majority Voting and Bagging methods used a multi-expert combination. Conversely, the multistage combination employs a serial approach where each supplementary learner works on the limitation of the previous base learners, such as, trained or tested only on the instances that previous base learners were not capable of achieving satisfactory accuracy. Boosting is an example of a multistage combination.

Additionally, this stage can log all the instances and decisions taken into a log file. Such log files can be exported to a Log Management and Analysis tool to further analyse and visualise generated alerts.

As mentioned, the SFIDS adopts Machine Learning classification techniques to analyse and classify the network flow instances as normal or malicious. The algorithm shown in Figure 5.2 illustrates the steps adopted to create the ML classification model. The first step reads network flow datasets which consist of normal and malicious network traffic. This labelled data is used to construct the supervised Machine Learning classification model. After reading the data, we employ the pre-processing methods to prepare the data to build the model. The prepared labelled dataset is fitted to the ML classification algorithm, and this builds and returns an ML classification model capable of classifying network flow data as normal or malicious. The model is then exported to the application data analysis engine to analyse new network flow instances gathered from the network in near real-time.

The model presented in Figure 5.1 can be summarised by combining the stages into three main components, namely; **Data Acquisition**, **Data Pre-processing**, and **Decision Engine**. Where Packet observation, flow metering, and export resides at the Data Acquisition Component, data collection and preparation in the Data Pre-processing component, and the data analysis and classification in the Decision Engine component. Figure 5.3 presents an overview of SFIDS with the three main components of the system.

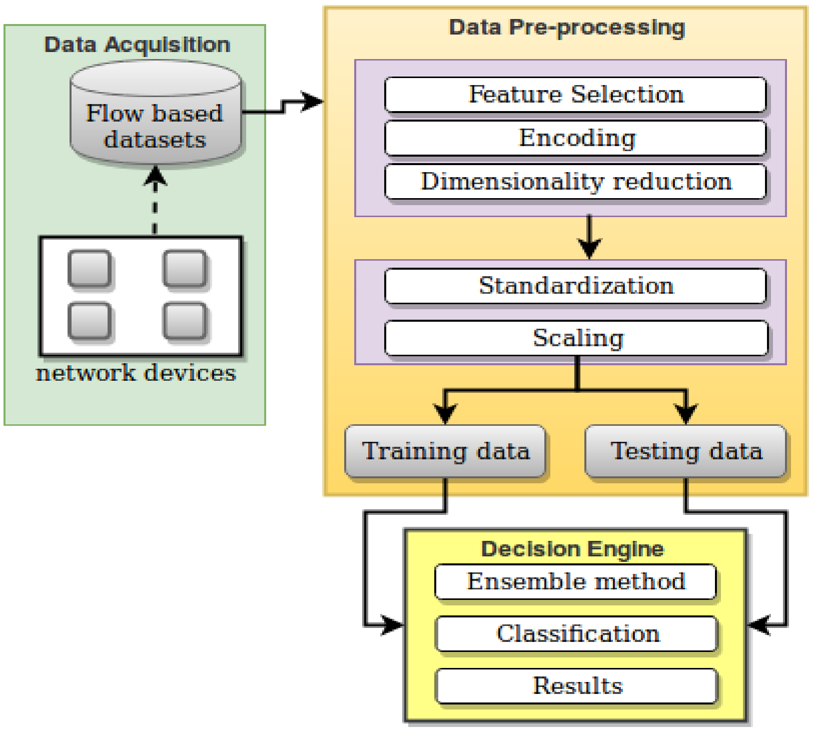


Figure 5.3 SFIDS using an ensemble learning method

5.3. Integration with SDN architecture

As mentioned in the previous chapters, SDN plays a significant role in SFIDS. The three components illustrated in Figure 5.3 can be easily integrated with the SDN architecture, as data acquisition can take place at the Data Plane, data pre-processing at the Control Plane, while the decision engine component resides at the Application Plane. The following section discusses how the SFIDS leverages SDN to accomplish its tasks. Figure. 5.4 presents how SFIDS is integrated with SDN for military tactical networks.

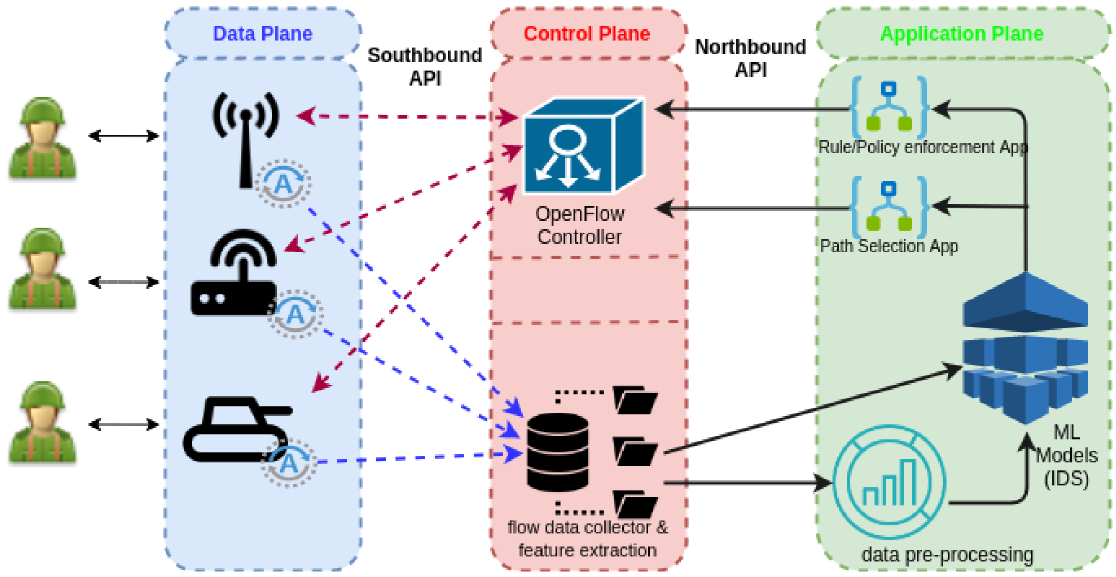


Figure 5.4 SFIDS and SDN (Zwane, Tarwireyi and Adigun, 2019a)

5.3.1. Data Plane

Data Acquisition Component **(**Packet Observation Stage and Flow Metering and Export Stage**):** All the network devices in the Data Plane are embedded with collector agents, as shown in Figure 5.4, the agents sample and send flow records to the centralised collector. The devices are configured to collect specific flow metrics and export them to the collector. Today built-in flow collection and export support are already offered by major vendors, for example, Cisco.

5.3.2. Control Plane

Data Pre-processing Component **(**Data Collection and Preparation Stage**):** The data collector residing in the CP module collects network flow records. The process of filtering is employed during feature extraction at this stage. The data collector generates and creates different datasets that are important for the adopted ML technique. Data sources are all network devices capable of communicating with the OpenFlow controller.

5.3.3. Application Plane

Decision Engine Component **(**Data Analysis Stage**):** The machine learning model is constructed and implemented as an SDN application. Different classifiers and regression models can be applied for different purposes as SDN applications using different datasets generated by the flow collector. Various applications can be constructed that are powered by ML models to influence the functioning of the network. Examples include incident handling applications, such as Rule or policy enforcement, and path selection applications. In our case, an ML model is built and used as an SDN intrusion detection application. This application will be capable of classifying network flows as malicious or normal. Figure 5.5 presents the SFIDS deployment architecture.

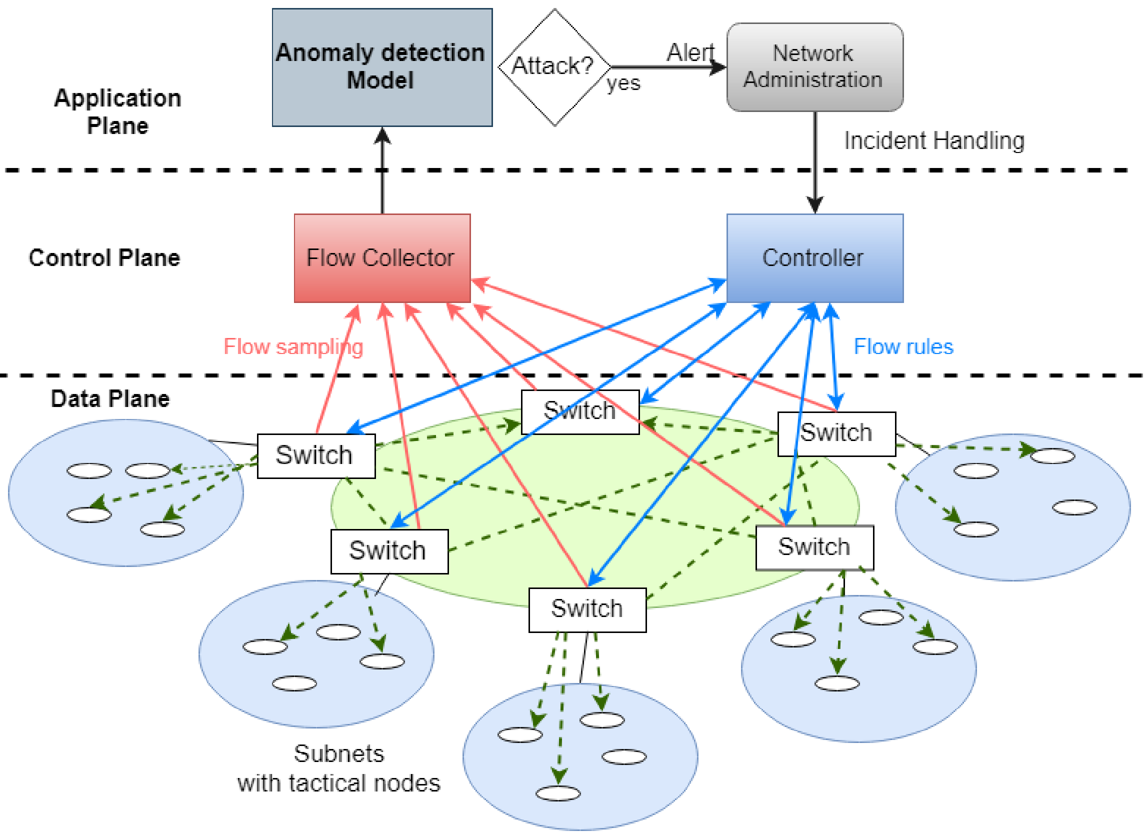


Figure 5.5 Proposed deployment architecture(Zwane, Tarwireyi and Adigun, 2019c)

The SFIDS uses a centralised network intrusion detection architecture, where network flow data is collected from the nodes and sent to a centralised collection point for analysis. This is made possible by the adoption of the SDN paradigm, which provides scalable data collection, centralised control, and global network view which allow data extraction, processing, analysis using minimum network resources (Amaral *et al.*, 2016). Thus, modern tactical military missions include a large number of actors which may consist of soldiers and vehicles operating independently or in coordination with each other or in sync with command centres (Poularakis, Iosifidis and Tassiulas, 2018). These actors are usually organised based on different commands or tactical goals. Actors are illustrated in Figure 5.5 as the subnets, consisting of a group of actors with similar commands. These actors may use fixed or deployable infrastructures to connect to high-level commands, for example, satellites, drones, and vehicles. This is reflected as switches in Figure 5.5 presented earlier, which can be any network device responsible for providing connectivity to the subnets.

5.4. Overall SFIDS Overview

In order to build an SFIDS, we need a method to extract flow data from the network. This method of extracting data should not affect or degrade the network's functionality at any point in time. This is an important stage since each of the instances created by this function represents the network flow statistics to be analysed. Note that after getting these values, they are sent for pre-processing. The pre-processed data plays a big role since it is sent to the trained machine learning classifier for analysis and classification.

5.4.1. Overall System Functionalities

SFIDS gather and analyse network flow data from the network for anomaly detection. To achieve this functionality, first, the approach applies flow sampling using network sampling tools. After the system has effectively collected/sampled a flow instance, the second phase of the application then applies pre-processing and preparation techniques to the instance, this converts the flow instance to a suitable format for ML analysis. The prepared flow instance is then classified as normal or malicious using an ML classifier or model which is trained using labelled network flow data with both normal and malicious traffic. As seen in Figure 5.6, the infrastructure layer is composed of networking devices that are regarded as data sources. The data collector resides in the Control Layer acting as a centralised data collector, and finally, the intrusion detection module with the classification model is contained in the Application Plane.

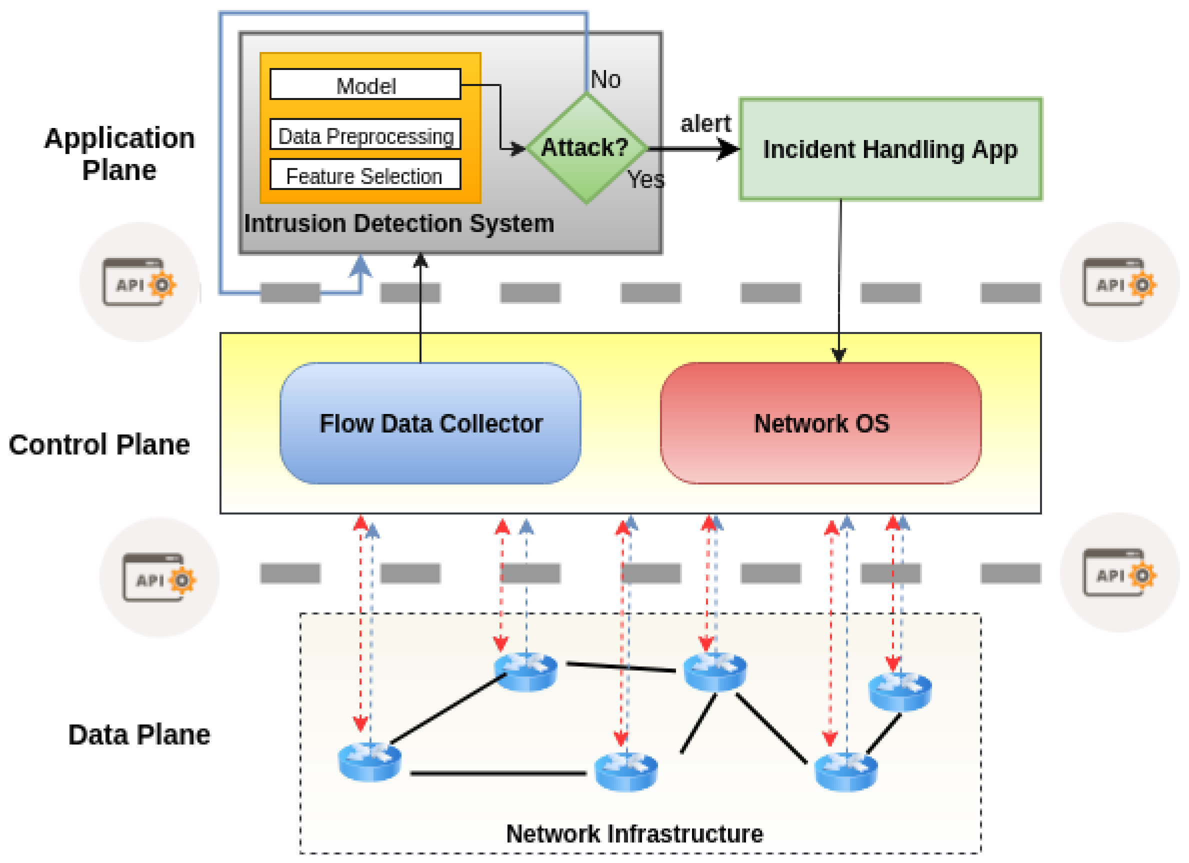


Figure 5.6 Overall FIDS model utilising SDN architecture

The steps of the algorithms from the SDN Data Plane to the Application Plane as shown in Figure 5.6 are discussed below:

Step 1: Data acquisition and sampling

This step plays an important role since it gathers network flow data from the network at multiple points, and presents it to the application for analysis and anomaly detection. As presented in Figure 5.7. Common approaches that can be utilised for this step include sFlow, NetFlow, and IPFIX tools (Hofstede *et al.*, 2014).

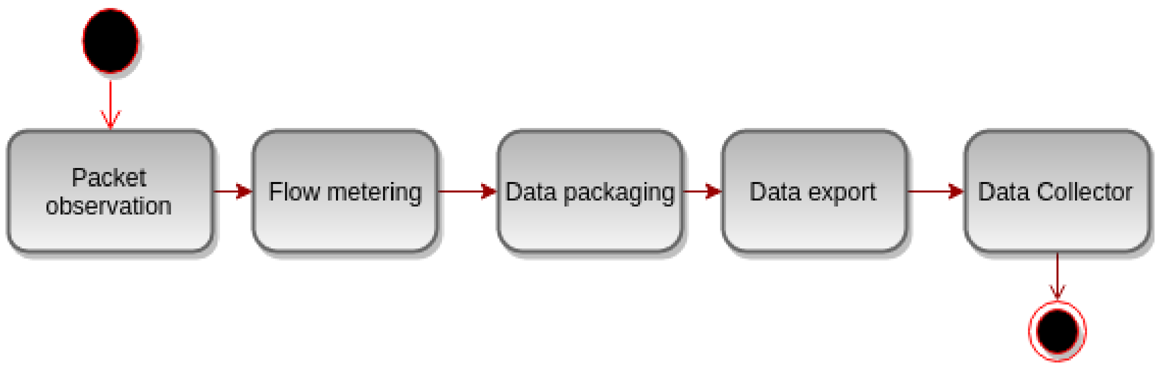


Figure 5.7 Data acquisition and sampling

Step 2: Data pre-processing and preparation

As mentioned above the second step applies pre-processing and preparation to each network flow instance sampled by the network sampling tools. Data pre-processing is applied to each of the instances, pre-processing done includes; encoding instance transport protocol, encoding categorical data, encoding numerical data, applying dimensionality reduction, and data scaling. These functions can be implemented using data analysis and pre-processing packages, for example, python pandas, and Scikit-learn (Pedregosa, Weiss and Brucher, 2011). The process is shown in Figure 5.8.

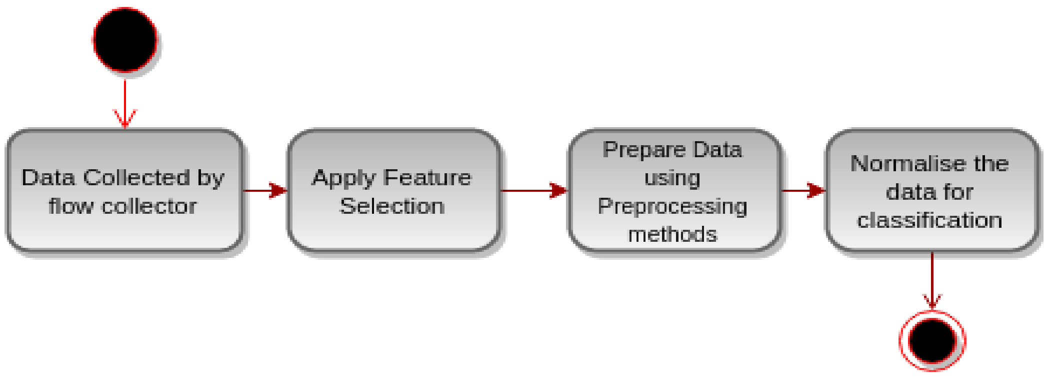

Data Collected by flow collector
Apply Feature Selection
Prepare Data using Preprocessing methods
Normalise the data for classification

Figure 5.8 Data pre-processing and preparation

Step 3: Data Analysis

The third step is concerned with analysing each flow instance gathered and prepared by the first and second steps. For data analysis, a Machine Learning model is constructed and the model is then used to fit a classifier, and use it to classify each instance to detect network intrusions. Thus, each flow instance is analysed and classified as either normal or malicious using an ML classifier trained with labelled data. The classification model generates alerts if a malicious flow is detected, else the process of network flow acquisition, preparation, and analysis repeats itself, as demonstrated in Figure 5.9.

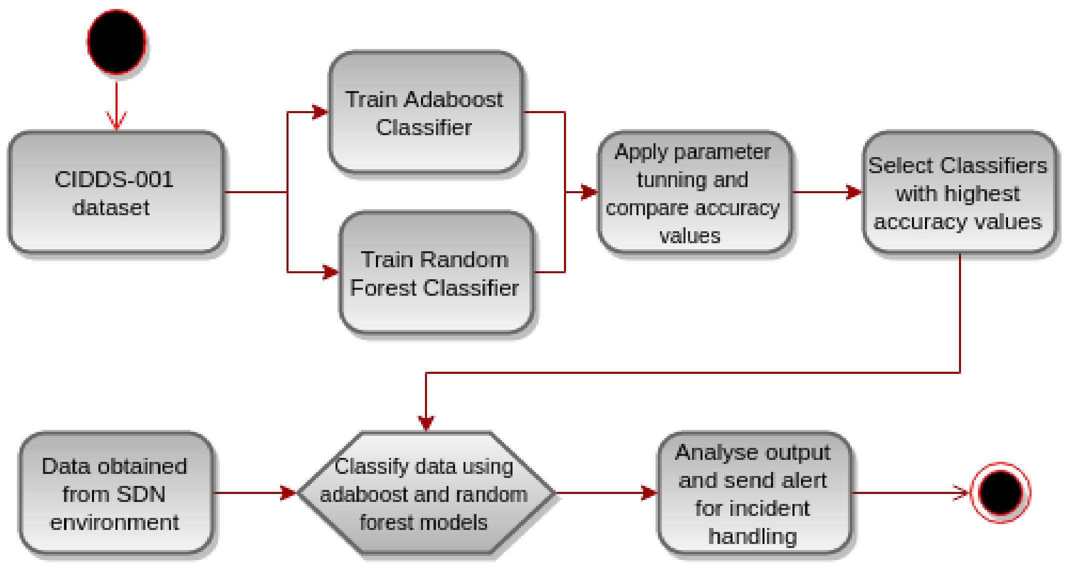

CIDDS-001 dataset
Train Adaboost Classifier
Train Random Forest Classifier
Apply parameter tunning and compare accuracy values
Select Classifiers with highest accuracy values
Data obtained from SDN environment
Classify data using adaboost and random forest models
Analyse output and send alert for incident handling

Figure 5.9 Data Analysis

Step 4: Incident Handling /Mitigation

The SFIDS method is capable of alerting network administrators by generating alerts if intrusions are detected. In conjunction with generating alerts, the study further proposes a system capable of handling incidents by employing mitigation strategies, such as blocking and dropping network traffic from an identified malicious node in the network. From the algorithm described in Figure 5.10, first, it gathers the ML classification model output. If the output indicates malicious behaviour, it extracts the IP address, MAC address, and the AP address providing network connectivity to the malicious node, else it returns to step 1. In step 3 the system sends network control flows to disable the malicious node (block node) from the network. Hence malicious nodes can be dropped or blocked from the network using this approach.

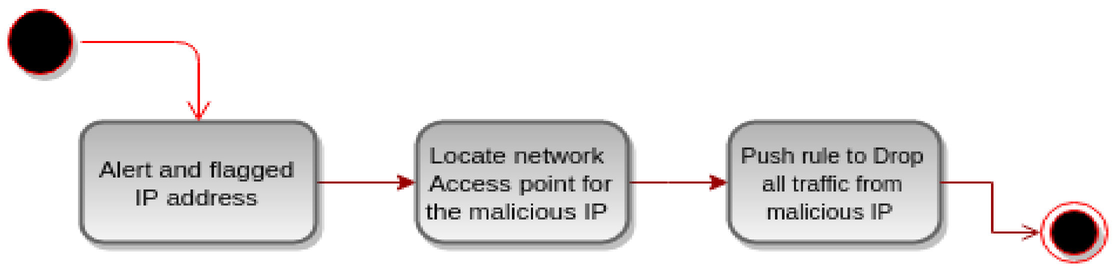

Alert and flagged IP address
Locate network Access point for the malicious IP
Push rule to Drop all traffic from malicious IP

Figure 5.10. Incident Handling

5.4.2. Unified Modelling Language (UML)

The SFIDS is composed of three modules that work together to extract, analyse, and take action, namely: flow sampling module, data collector module, and IDS application module, which is operated by the ML model. The flow of events in each of these components is described in Figure 5.11.

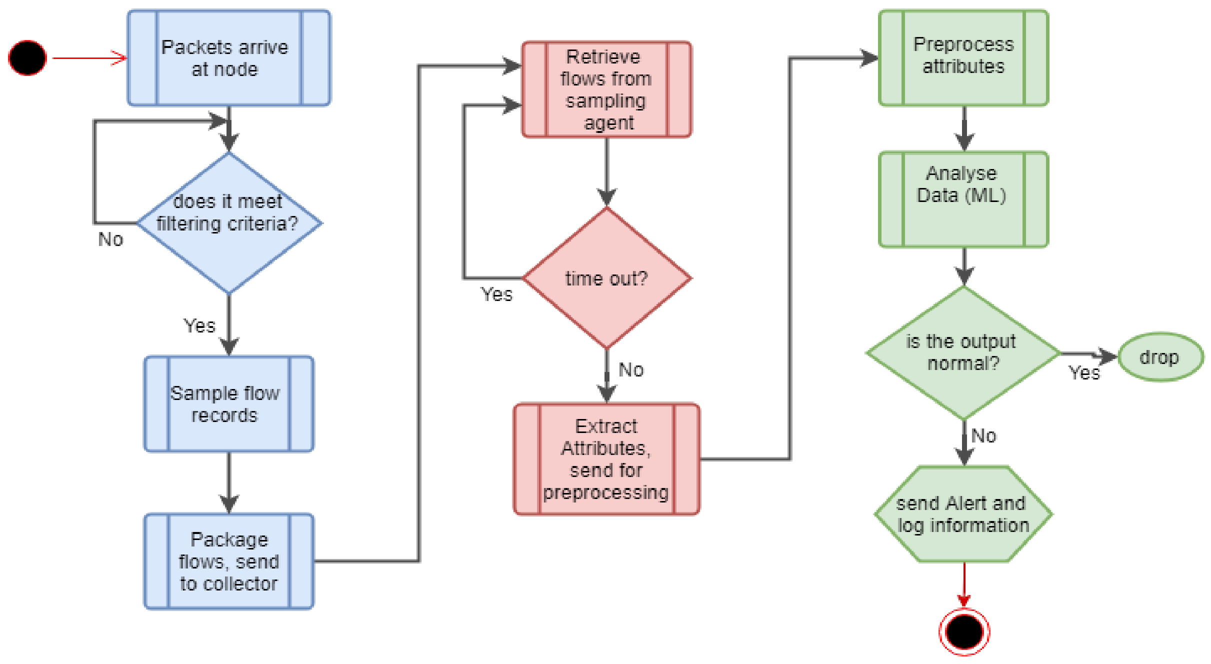

Packets arrive at node
does it meet filtering criteria?
No
Yes
Sample flow records
Package flows, send to collector
Retrieve flows from sampling agent
time out?
Yes
No
Extract Attributes, send for preprocessing
Preprocess attributes
Analyse Data (ML)
is the output normal?
Yes
drop
No
send Alert and log information

Figure 5.11 Flow of event in the SFIDS Model

Sampling Agents

The sampling agents wait for packets to be transmitted and collect flow information. The agent verifies if the packets meet specified criteria, which is vital for filtering network control messages, as depicted in Figure 5.11. If the packets meet specified criteria, filtering is conducted by defining a threshold. The threshold specifies and manages flows by inspecting packet headers. Flows are then packaged and sent to the collector.

Collector Modules

The collector module is responsible for collecting the data from the different sampling agents embedded in each network device. Its task is to collect the flow data, employ feature extraction then pass the data with appropriate features or attributes to the SDN application for data pre-processing and cleaning.

IDS Application

The IDS application periodically queries or retrieves flow records from the collector. It waits with time out and repeats the process. For each new flow record, pre-processing methods are applied to the flow record. After converting the flow record into ML acceptable input format, the ML model is used to classify it as normal or malicious. If the record is normal, then the IDS application moves on to the next flow record retrieved. However, if the record is malicious, the application takes a snapshot of the flow record's data and inserts the information in a log file. Logging detected malicious incidents could then be helpful for visualisation and security incident handling. A sequence diagram of the flow-based intrusion detection method using an ensemble learning technique is shown in Figure 5.12.

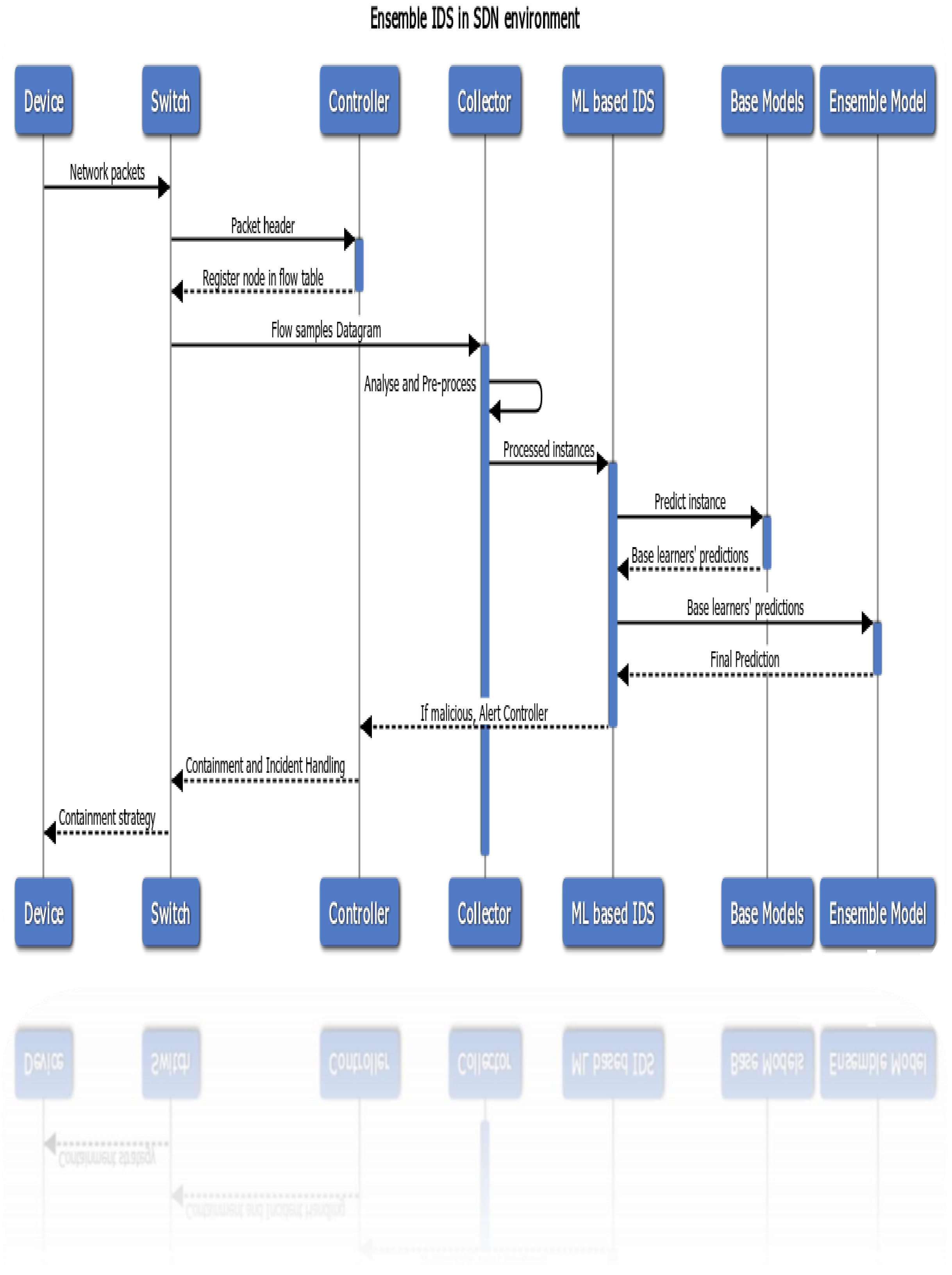


Figure 5.12 Sequence diagram

5.5. Summary

The integration of Machine Learning and SDN for intrusion detection in military tactical networks will result in a number of benefits which include high detection rates, efficiency, and real-time data collection. This will help ensure security while introducing timeless and real-time attack detection and alerting capabilities. Furthermore, the study proposes the use of network flow data for intrusion detection since it is computationally cheap, independent from encrypted data, lightweight, and faster compared to the traditional packet-based intrusion detection approach (Fahad, Sher and Bi, 2017). Accordingly, this chapter presented the Flow-based IDS model, functional requirements, and the integration with SDN architecture.

Chapter 6: SFIDS Implementation

To address the fourth objective of this research (as stated in Chapter 1), which is to implement SFIDS, this chapter outlines the implementation of the Software-defined Flow-based Intrusion Detection System (SFIDS) prototype to operationalise it using Mininet-Wifi. The chapter presents the solution implementation and evaluations guided by the Design Science methodology mentioned in Chapter 3. This chapter presents the technical feasibility of the SFIDS method. In particular, the chapter details the experimental setup and technologies used to realise and implement SFIDS. It further presents the data collection and pre-processing procedure in a wireless SDN environment to mimic an SDN-based tactical MANET deployment. This chapter further elaborates on Chapter 5 in its response to the second sub-research question, by focusing on the implementation SFIDS proposed in Chapter 5.

6.1. Environment Setup and Tools

This section presents the Software-Defined Network (SDN) environment setup and tools used to implement the proof-of-concept prototype. In particular, the section presents the topology used, emulation tool, OpenFlow controller, packet generation, and sampling tools.

6.1.1. Mininet-Wifi

Mininet-Wifi (Fontes *et al.*, 2015) is a wireless network emulation tool used in our experiments. Mininet-Wifi is an extension of the Mininet software popularly used in SDN research. Mininet is an emulation tool used to prototype a network on a laptop or PC by using the Kernel namespace feature. The network namespace is used by Mininet to provide individual processes with their network interfaces, ARP tables, and routing tables. It uses process-based virtualisation to run switches and hosts on the Kernel, which allow large networks with different topologies to be emulated and evaluated. Mininet-Wifi augments Mininet with virtual wireless stations and access points while maintaining the original SDN capabilities and lightweight virtualisation software architecture. Mininet-Wifi is a tool that allows the emulation of OpenFlow/SDN scenarios that enable high fidelity experiments that replicate real networking environments.

6.1.2. Network topology

Military tactical networks deployed in the battlefield have to support critical mission requirements in harsh operational environments. Such networks are forced to deal with the effects of frequent mobility, irregular link state, and inconstant bandwidth in hostile territory (Marcus *et al.*, 2019). This subsection presents the proposed tactical MANET topology used in this work. Figure 6.1. Shows a tactical mobile ad-hoc network topology with three Access Points (APs) and stations connected to them.

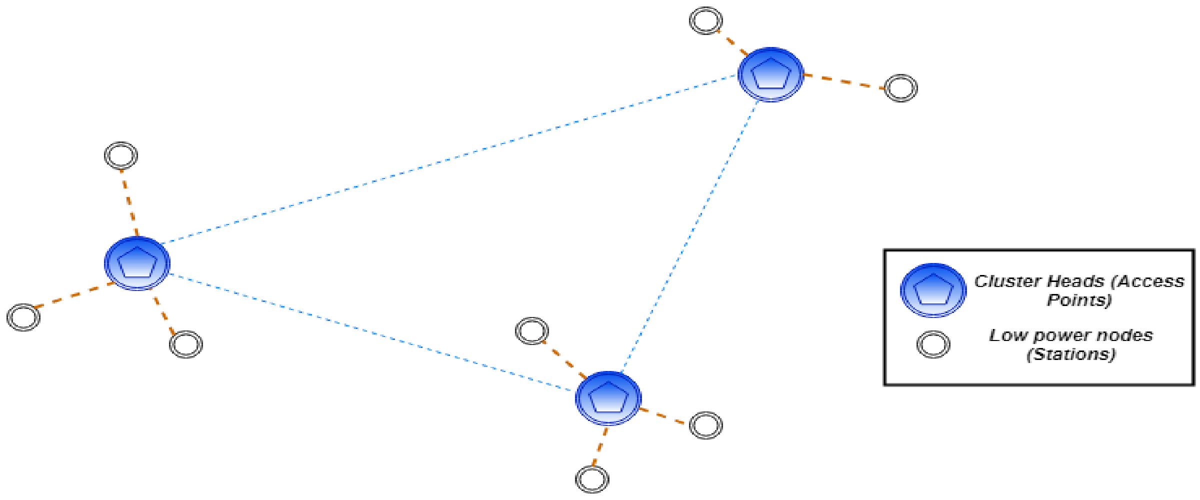


Figure 6.1 Tactical MANET topology

They are assumed to have a strong power supply, for example, they may be devices running on the troop vehicles, or temporary command and control centre. They provide network connectivity to the stations connected to them. The low- power devices, such as light devices carried around by the troops, connect to the cluster heads to get network access. The cluster heads are used to handle intrusion detection tasks since they possess a high power budget. To address the problems of dynamic configuration and network global view Software Defined Networking (SDN) is introduced to this topology.

6.1.3. OpenFlow Controller

The increasing popularity of SDN resulted in several OpenFlow controllers being developed, these controllers are classified into centralised and distributed controllers. In a centralised controller, a single server is responsible for all the control plane activities. The benefit of this approach is simplified management as it makes a single point of control available. Examples of popular centralised controllers include; Beacon (Erickson, 2013), Rosemary (Shin *et al.*, 2014), Maestro (Cai, Cox and Ng, 2010), NOX-MT (Tootoonchian *et al.*, 2012), and

OpenDayLight. However, centralised controllers suffer from scalability issues as each server has restricted capacity in handling data plane devices.

In contrast, distributed controllers have advantages in terms of scalability and high performance during increased demand for requests. Popular distributed controllers include, Hypervisor (Tootoonchian, 2010), SMaRtLight (Botelho *et al.*, 2014), ONOS (Berde *et al.,* 2014), ONIX (Koponen *et al.*, 2010), and Floodlight. In this work, the Floodlight controller is used. Floodlight is open-source software written in Java. The main advantage of Floodlight is that it is designed to allow third parties to modify the software and develop applications. The approach uses the Representational state transfer (REST) APIs to simplify the application interfaces to the product.

6.1.4. Flow Sampling

The most crucial stage of any intrusion detection is the data gathering stage. Intrusion detection methods are data-driven which means they can fulfil their purpose through analysing data. This is more than enough to emphasise the importance of data in such systems. The proposed FIDS system makes use of network flow data for intrusion detection. This is because the task of acquiring flow records from networks has been simplified recently (Fahad, 2017), with major vendors nowadays offering incorporated flow gathering and export support in their hardware. Examples of these flow collection and export protocols include Cisco's NetFlow, IPFIX, and sFlow. This process is packaged under the Packet Observation stage. In this work, sFlow was used for packet observation because it is the process of capturing packets from the communication line and pre-processing it for further use.

sFlow is a multi-vendor sampling technology that is embedded within network forwarding devices, in our case SDN data plane devices. It offers the ability to continuously monitor application-level traffic flows at wire speed on all interfaces simultaneously (Visible and Packard, 2003). The sFlow mechanism is made up of two components; the sFlow Agent and sFlow Collector.

The sFlow Agent is a software process that runs as part of network management software within the devices. It combines both interface counters and flow samples into sFlow Datagrams. The sFlow Datagrams are then immediately sent to the sFlow Collector**,** where they are analysed to produce a rich, real-time, network-wide view of the traffic flows. Figure 6.2 shows the sFlow mechanism, with the sFlow agent, collector, and datagrams.

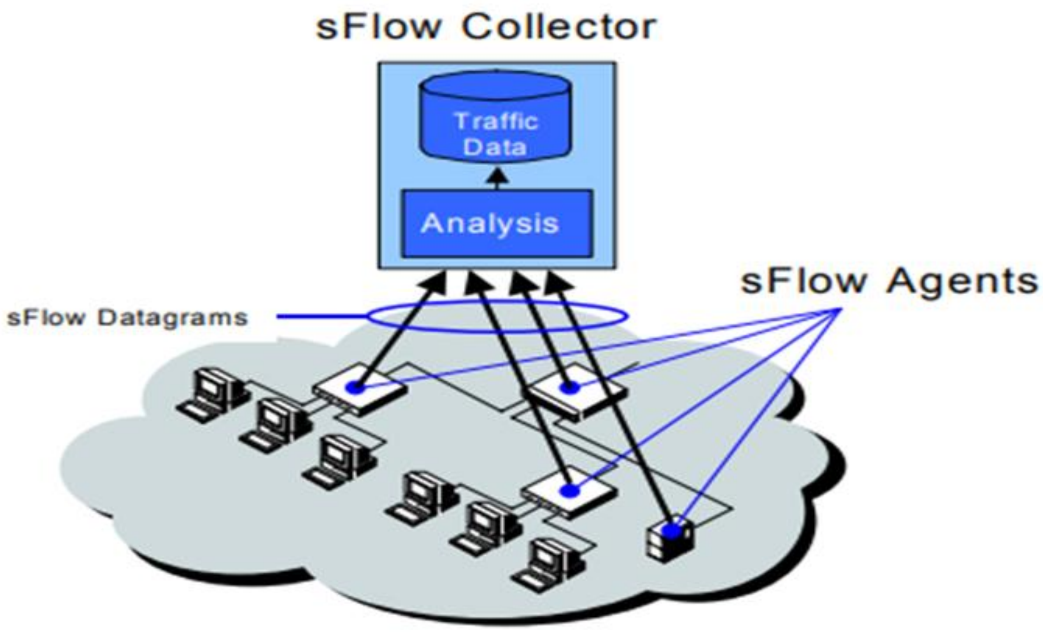


Figure 6.2 sFlow agent and Collector (Visible and Packard, 2003)

The sFlow sampling technology is characterised by the following network traffic monitoring requirements:

- sFlow makes network-wide view of usage and active routes available, enabling tens of thousands of interfaces to be monitored from a single location.
- sFlow achieves scalability, meaning it has the capacity for handling or monitoring links of up to 10GB/s and beyond without adding major network load.
- sFlow can be deployed at a very low cost, applicable in simple Layer 2 switches to high-end core routers without needing extra memory and CPU.
- sFlow is industry recognised, as such an increasing number of vendors offer sFlow support in their network devices.

6.2. Simulation and Data Collection

The primary focus of this section is to present the implementation of SFIDS in SDN-based wireless network. Figure 6.3 illustrates the proposed model embedded with the technologies discussed in Section 5.4 that are used to accomplish each task.

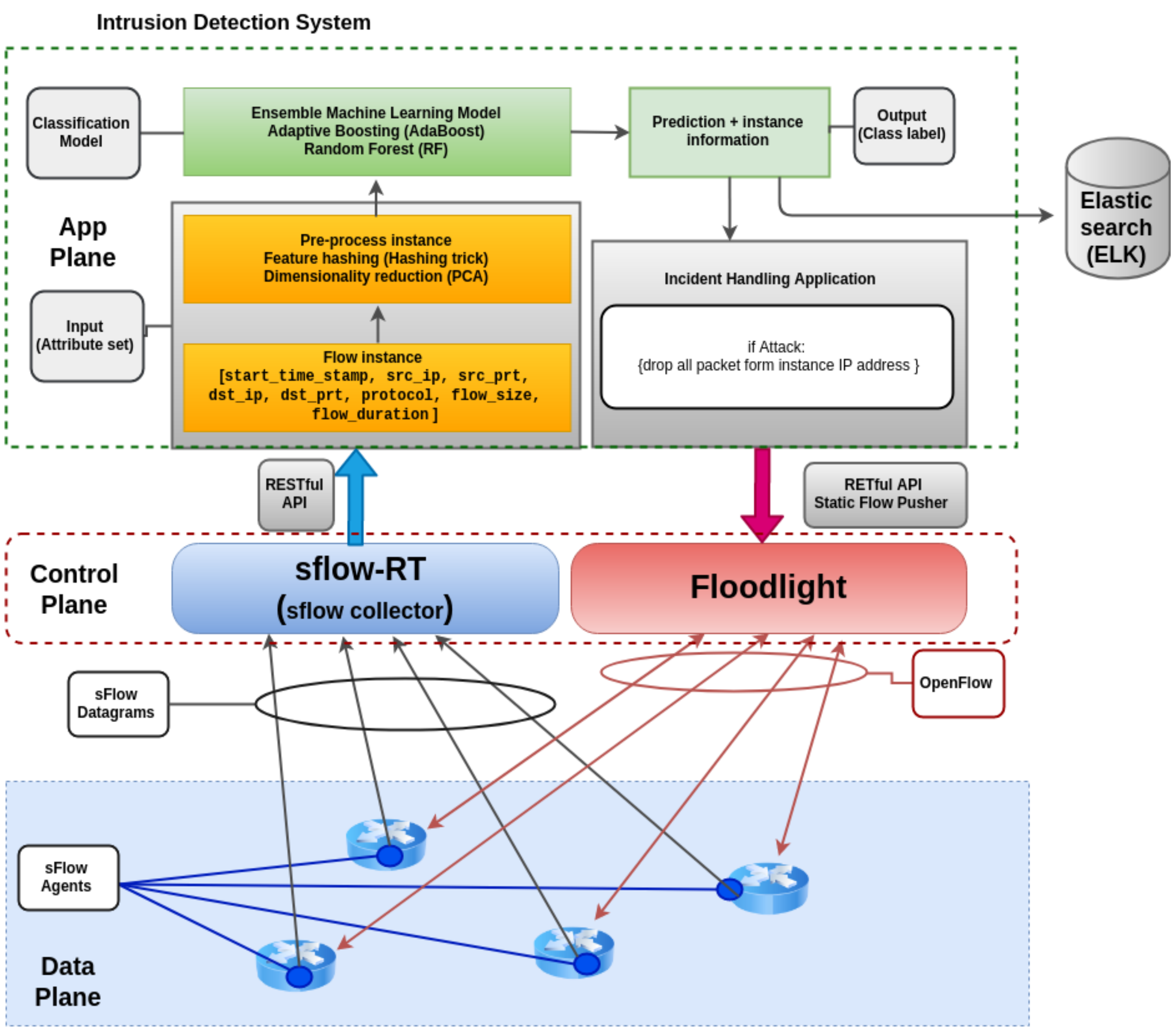
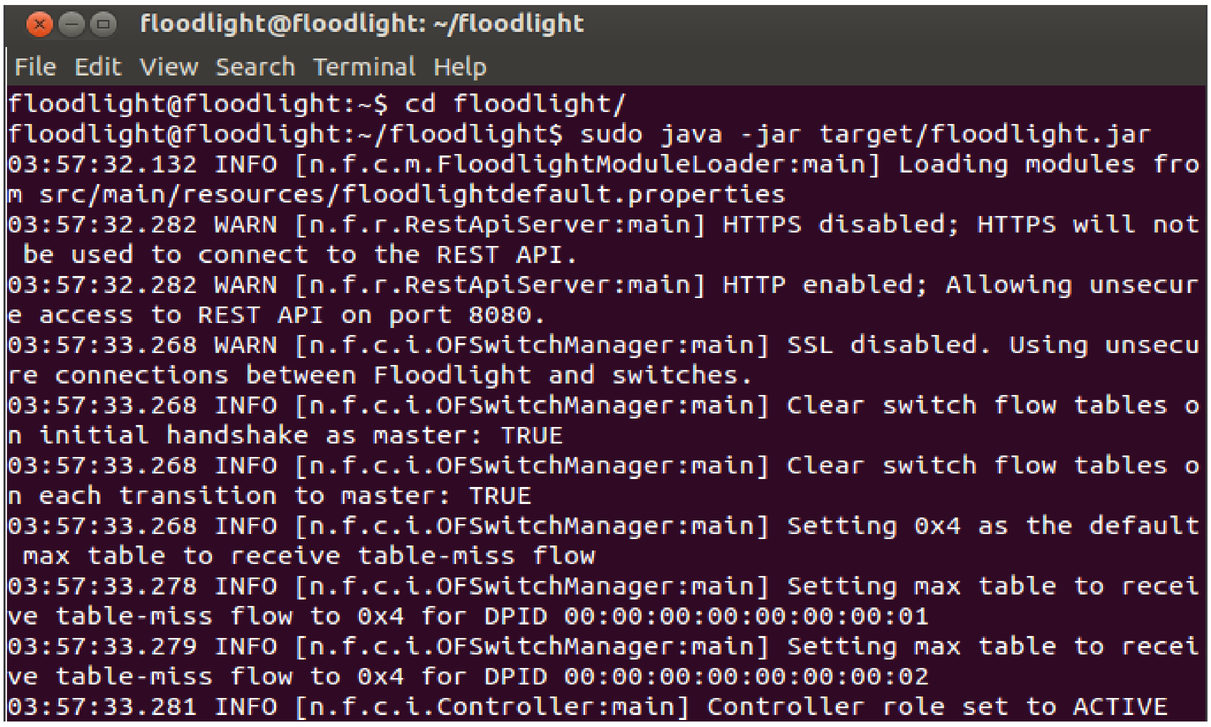

Intrusion Detection System
Classification Model
Ensemble Machine Learning Model
Adaptive Boosting (AdaBoost)
Random Forest (RF)
Prediction + instance information
Output (Class label)
Elastic search (ELK)
App Plane
Pre-process instance
Feature hashing (Hashing trick)
Dimensionality reduction (PCA)
Input (Attribute set)
Flow instance
[start_time_stamp, src_ip, src_prt, dst_ip, dst_prt, protocol, flow_size, flow_duration]
Incident Handling Application
if Attack:
{drop all packet form instance IP address }
RESTful API
RETful API Static Flow Pusher
Control Plane
sflow-RT (sflow collector)
Floodlight
sFlow Datagrams
OpenFlow
sFlow Agents
Data Plane

Figure 6.3 Implementation of FIDS model utilising SDN architecture

6.2.1. Network Simulation

In this study, using Mininet-Wifi, a wireless network with three access points (APs) and eight wireless stations (Sta) as shown in Figure 6.1 was created. The OVSKernelAP was used for the network Access points. The Floodlight OpenFlow Controller was used as the Network Operating System (NOS). Using VirtualBox a new Virtual Machine (VM) was created to host the Floodlight controller, hence the controller was running in a VM running on VirtualBox in the host computer containing Mininet-Wifi. In essence, the floodlight controller and Mininet-Wifi reside on logically separate machines, and this is done to mimic a realistic SDN deployment scenario where the controller may be located in a remote server. The floodlight controller is started using the command below and the output is shown in Figure 6.4.

sudo java –jar targer/floodlight.jar

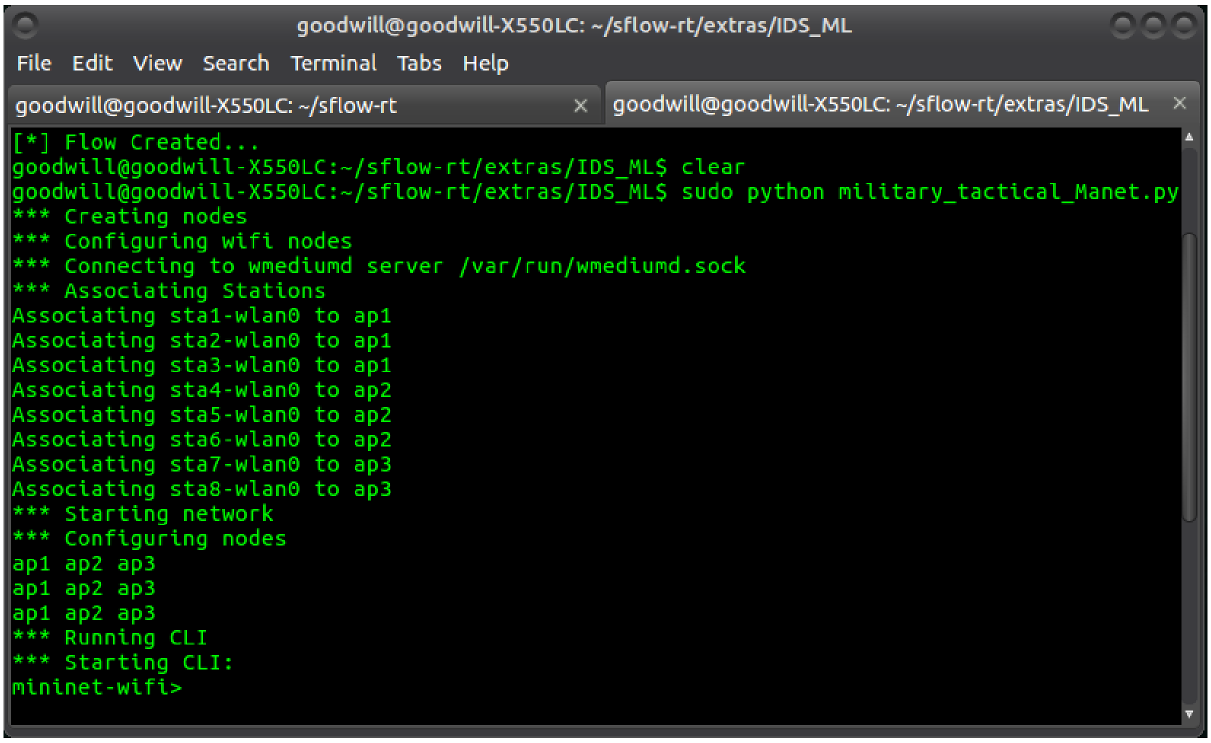

floodlight@floodlight: ~/floodlight
File Edit View Search Terminal Help

```
floodlight@floodlight:~$ cd floodlight/
floodlight@floodlight:~/floodlight$ sudo java -jar target/floodlight.jar
03:57:32.132 INFO [n.f.c.m.FloodlightModuleLoader:main] Loading modules fro
m src/main/resources/floodlightdefault.properties
03:57:32.282 WARN [n.f.r.RestApiServer:main] HTTPS disabled; HTTPS will not
 be used to connect to the REST API.
03:57:32.282 WARN [n.f.r.RestApiServer:main] HTTP enabled; Allowing unsecur
e access to REST API on port 8080.
03:57:33.268 WARN [n.f.c.i.OFSwitchManager:main] SSL disabled. Using unsecu
re connections between Floodlight and switches.
03:57:33.268 INFO [n.f.c.i.OFSwitchManager:main] Clear switch flow tables o
n initial handshake as master: TRUE
03:57:33.268 INFO [n.f.c.i.OFSwitchManager:main] Clear switch flow tables o
n each transition to master: TRUE
03:57:33.268 INFO [n.f.c.i.OFSwitchManager:main] Setting 0x4 as the default
 max table to receive table-miss flow
03:57:33.278 INFO [n.f.c.i.OFSwitchManager:main] Setting max table to recei
ve table-miss flow to 0x4 for DPID 00:00:00:00:00:00:00:01
03:57:33.279 INFO [n.f.c.i.OFSwitchManager:main] Setting max table to recei
ve table-miss flow to 0x4 for DPID 00:00:00:00:00:00:00:02
03:57:33.281 INFO [n.f.c.i.Controller:main] Controller role set to ACTIVE
```

Figure 6.4 Starting Floodlight Controller in terminal

After running the Floodlight controller, we open a new terminal in the host machine where Mininet-Wifi, sFlow, and Scapy are installed. As mentioned, a custom topology with three access points and eight wireless stations was created. The topology is created through running the python script as;

sudo python military_tactical_Manet.py


goodwill@goodwill-X550LC: ~/sflow-rt/extras/IDS_ML
File Edit View Search Terminal Tabs Help
goodwill@goodwill-X550LC: ~/sflow-rt
goodwill@goodwill-X550LC: ~/sflow-rt/extras/IDS_ML

```
[*] Flow Created...
goodwill@goodwill-X550LC:~/sflow-rt/extras/IDS_ML$ clear
goodwill@goodwill-X550LC:~/sflow-rt/extras/IDS_ML$ sudo python military_tactical_Manet.py
*** Creating nodes
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Associating Stations
Associating sta1-wlan0 to ap1
Associating sta2-wlan0 to ap1
Associating sta3-wlan0 to ap1
Associating sta4-wlan0 to ap2
Associating sta5-wlan0 to ap2
Associating sta6-wlan0 to ap2
Associating sta7-wlan0 to ap3
Associating sta8-wlan0 to ap3
*** Starting network
*** Configuring nodes
ap1 ap2 ap3
ap1 ap2 ap3
ap1 ap2 ap3
*** Running CLI
*** Starting CLI:
mininet-wifi>
```

Figure 6.5 Simulation of SDN based tactical MANET in Mininet-Wifi

From running the topology script, the resulting topology is used in our experiments. Figure 6.6 shows the network nodes and access points. While Figure 6.7 and Figure 6.8 shows the Floodlight GUI and different network nodes' terminals respectively.

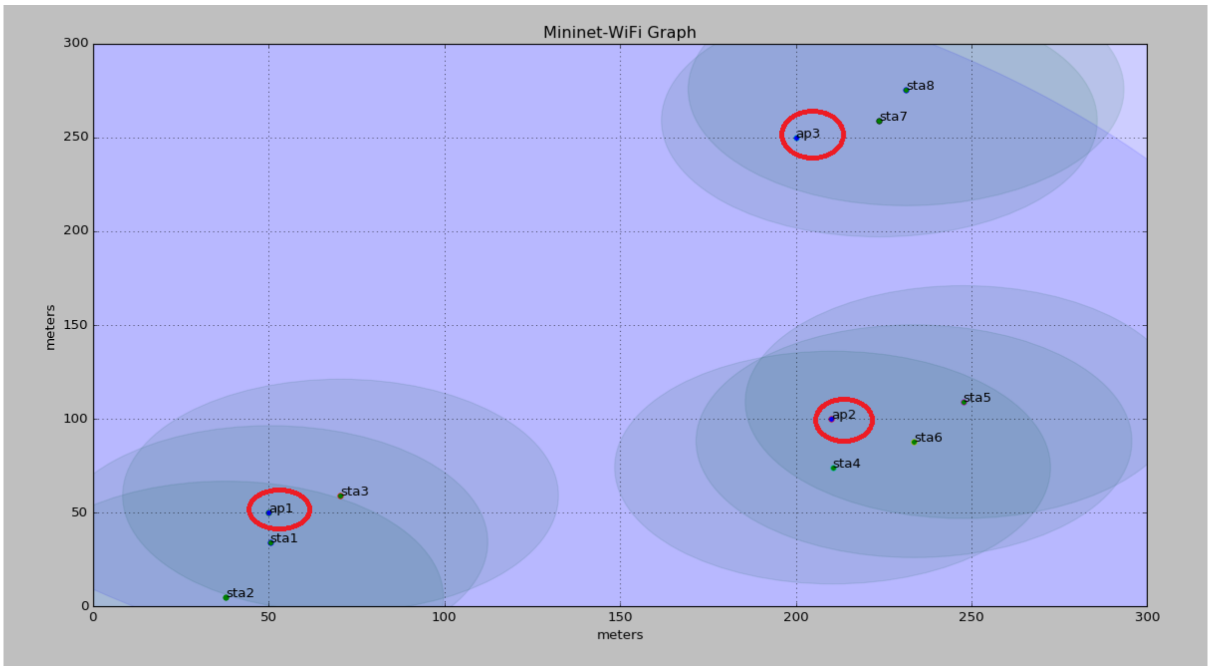
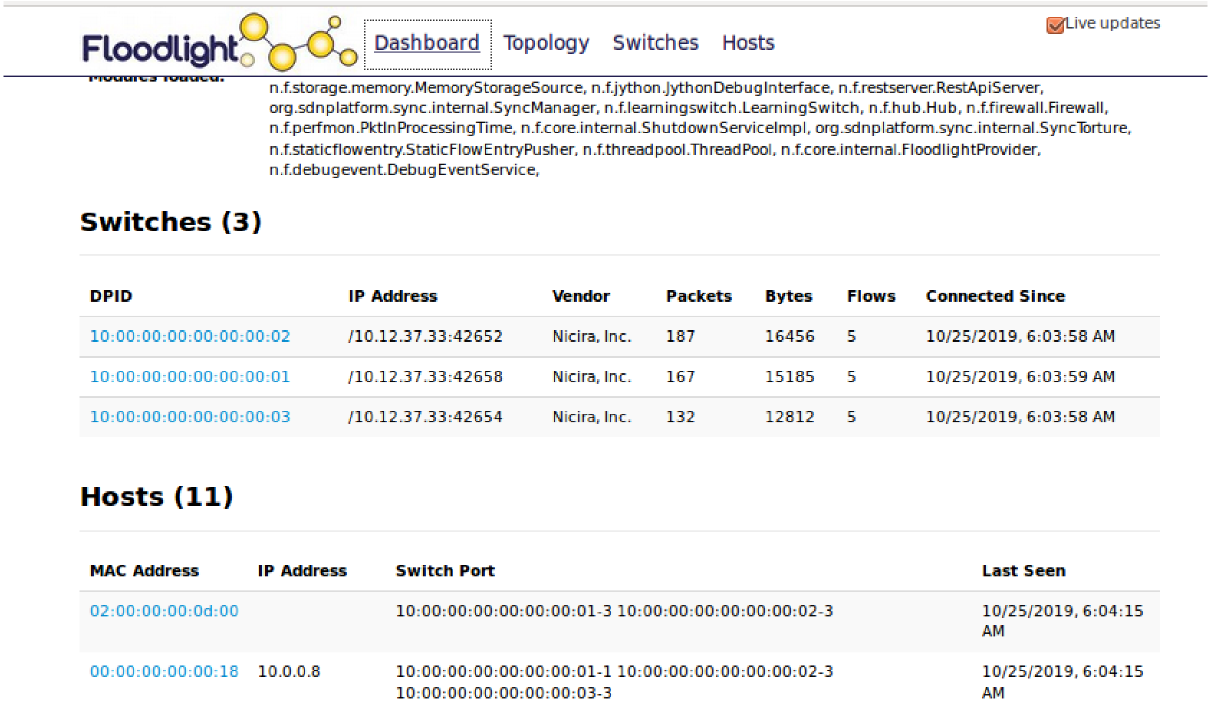


Figure 6.6 Wireless SDN topology with three APs and eight stations



Modules loaded: n.f.storage.memory.MemoryStorageSource, n.f.jython.JythonDebugInterface, n.f.restserver.RestApiServer, org.sdnplatform.sync.internal.SyncManager, n.f.learningswitch.LearningSwitch, n.f.hub.Hub, n.f.firewall.Firewall, n.f.perfmon.PktInProcessingTime, n.f.core.internal.ShutdownServiceImpl, org.sdnplatform.sync.internal.SyncTorture, n.f.staticflowentry.StaticFlowEntryPusher, n.f.threadpool.ThreadPool, n.f.core.internal.FloodlightProvider, n.f.debugevent.DebugEventService,

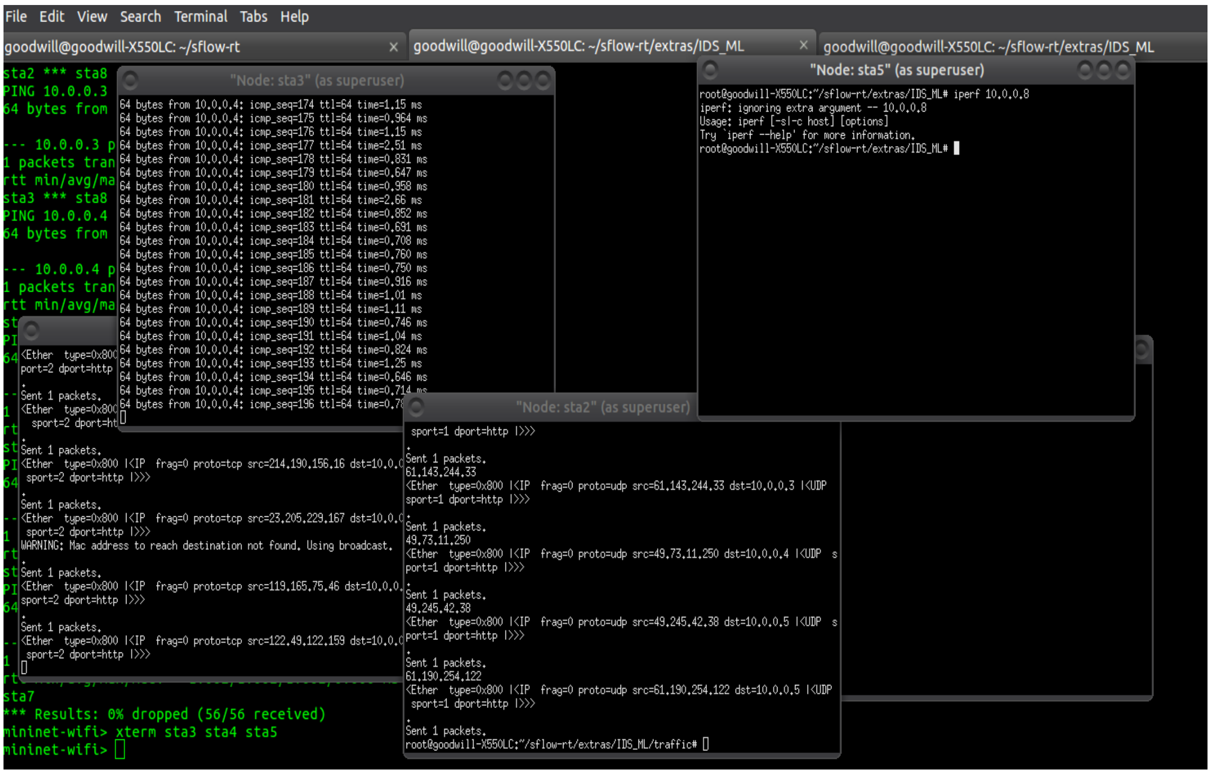
Switches (3)

DPID	IP Address	Vendor	Packets	Bytes	Flows	Connected Since
10:00:00:00:00:00:00:02	/10.12.37.33:42652	Nicira, Inc.	187	16456	5	10/25/2019, 6:03:58 AM
10:00:00:00:00:00:00:01	/10.12.37.33:42658	Nicira, Inc.	167	15185	5	10/25/2019, 6:03:59 AM
10:00:00:00:00:00:00:03	/10.12.37.33:42654	Nicira, Inc.	132	12812	5	10/25/2019, 6:03:58 AM

Hosts (11)

MAC Address	IP Address	Switch Port	Last Seen
02:00:00:00:0d:00		10:00:00:00:00:00:00:01-3 10:00:00:00:00:00:00:02-3	10/25/2019, 6:04:15 AM
00:00:00:00:00:18	10.0.0.8	10:00:00:00:00:00:00:01-1 10:00:00:00:00:00:00:02-3 10:00:00:00:00:00:00:03-3	10/25/2019, 6:04:15 AM

Figure 6.7 Floodlight GUI with 3 Access points and 8 Stations connected


File Edit View Search Terminal Tabs Help
goodwill@goodwill-X550LC: ~/sflow-rt
goodwill@goodwill-X550LC: ~/sflow-rt/extras/IDS_ML
goodwill@goodwill-X550LC: ~/sflow-rt/extras/IDS_ML
"Node: sta3" (as superuser)
"Node: sta5" (as superuser)
"Node: sta2" (as superuser)

```
sta2 *** sta8
PING 10.0.0.3
64 bytes from

--- 10.0.0.3 p
1 packets tran
rtt min/avg/ma
sta3 *** sta8
PING 10.0.0.4
64 bytes from

--- 10.0.0.4 p
1 packets tran
rtt min/avg/ma
sta7
*** Results: 0% dropped (56/56 received)
mininet-wifi> xterm sta3 sta4 sta5
mininet-wifi> 
```

```
64 bytes from 10.0.0.4: icmp_seq=174 ttl=64 time=1.15 ms
64 bytes from 10.0.0.4: icmp_seq=175 ttl=64 time=0.964 ms
64 bytes from 10.0.0.4: icmp_seq=176 ttl=64 time=1.15 ms
64 bytes from 10.0.0.4: icmp_seq=177 ttl=64 time=2.51 ms
64 bytes from 10.0.0.4: icmp_seq=178 ttl=64 time=0.831 ms
64 bytes from 10.0.0.4: icmp_seq=179 ttl=64 time=0.647 ms
64 bytes from 10.0.0.4: icmp_seq=180 ttl=64 time=0.958 ms
64 bytes from 10.0.0.4: icmp_seq=181 ttl=64 time=2.66 ms
64 bytes from 10.0.0.4: icmp_seq=182 ttl=64 time=0.852 ms
64 bytes from 10.0.0.4: icmp_seq=183 ttl=64 time=0.691 ms
64 bytes from 10.0.0.4: icmp_seq=184 ttl=64 time=0.708 ms
64 bytes from 10.0.0.4: icmp_seq=185 ttl=64 time=0.760 ms
64 bytes from 10.0.0.4: icmp_seq=186 ttl=64 time=0.750 ms
64 bytes from 10.0.0.4: icmp_seq=187 ttl=64 time=0.916 ms
64 bytes from 10.0.0.4: icmp_seq=188 ttl=64 time=1.01 ms
64 bytes from 10.0.0.4: icmp_seq=189 ttl=64 time=1.11 ms
64 bytes from 10.0.0.4: icmp_seq=190 ttl=64 time=0.746 ms
64 bytes from 10.0.0.4: icmp_seq=191 ttl=64 time=1.04 ms
64 bytes from 10.0.0.4: icmp_seq=192 ttl=64 time=0.824 ms
64 bytes from 10.0.0.4: icmp_seq=193 ttl=64 time=1.25 ms
64 bytes from 10.0.0.4: icmp_seq=194 ttl=64 time=0.646 ms
64 bytes from 10.0.0.4: icmp_seq=195 ttl=64 time=0.714 ms
64 bytes from 10.0.0.4: icmp_seq=196 ttl=64 time=0.78
```

```
<Ether  type=0x800
port=2 dport=http
.
Sent 1 packets.
<Ether  type=0x800
 sport=2 dport=ht
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=214.190.156.16 dst=10.0.0
 sport=2 dport=http |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=23.205.229.167 dst=10.0.0
 sport=2 dport=http |>>>
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=119.165.75.46 dst=10.0.0.
sport=2 dport=http |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=122.49.122.159 dst=10.0.0
 sport=2 dport=http |>>>
```

```
root@goodwill-X550LC:~/sflow-rt/extras/IDS_ML# iperf 10.0.0.8
iperf: ignoring extra argument -- 10.0.0.8
Usage: iperf [-s|-c host] [options]
Try `iperf --help' for more information.
root@goodwill-X550LC:~/sflow-rt/extras/IDS_ML# 
```

```
 sport=1 dport=http |>>>
.
Sent 1 packets.
61.143.244.33
<Ether  type=0x800 |<IP  frag=0 proto=udp src=61.143.244.33 dst=10.0.0.3 |<UDP
sport=1 dport=http |>>>
.
Sent 1 packets.
49.73.11.250
<Ether  type=0x800 |<IP  frag=0 proto=udp src=49.73.11.250 dst=10.0.0.4 |<UDP  s
port=1 dport=http |>>>
.
Sent 1 packets.
49.245.42.38
<Ether  type=0x800 |<IP  frag=0 proto=udp src=49.245.42.38 dst=10.0.0.5 |<UDP  s
port=1 dport=http |>>>
.
Sent 1 packets.
61.190.254.122
<Ether  type=0x800 |<IP  frag=0 proto=udp src=61.190.254.122 dst=10.0.0.5 |<UDP
 sport=1 dport=http |>>>
.
Sent 1 packets.
root@goodwill-X550LC:~/sflow-rt/extras/IDS_ML/traffic# 
```

Figure 6.8 Network nodes or Mininet-Wifi node instances (Containers)

6.2.2. Data Collection Scenarios

Military tactical networks with fixed infrastructure have comparable vulnerabilities with their commercial equivalents, except that they must be secure against adversaries that possess greater capabilities, resources, and motivation (Kurdziel, 2014). In addition, networks with mobile infrastructure components have extra vulnerabilities that must be considered. For example, devices or equipment are subject to capture by adversaries and an intruder may add a node to the network by replicating its *id* to get access to cryptographic keys and secret messages passing through the network.

The evaluation scenarios used in this study focus on adversaries taking advantage of a Mobile Ad hoc tactical network, as there is a possibility of an adversary gaining access to the network, and being able not only to eavesdrop sensitive information but also to mislead the users or harm the network (Carvalho and Costa, 2016). In that regard, the study focusses on internal network attacks rather than external attacks which are common in fixed infrastructure networks. The next subsection presents three test scenarios containing simulated internal attacks and assumptions made in each scenario.

1) Scenario 1: TCP flood attack detection

The first test assumes an intruder has successfully infiltrated the network and launched a TCP flood attack on a node in the network. The malicious node issues a TCP flood attack to a specific node they have already identified in the network. The target node is then denied service by a malicious node connected to the network (see Figure 6.9). The purpose of this test is to validate the effectiveness of SFIDS in terms of recognising and detecting malicious nodes in the network.

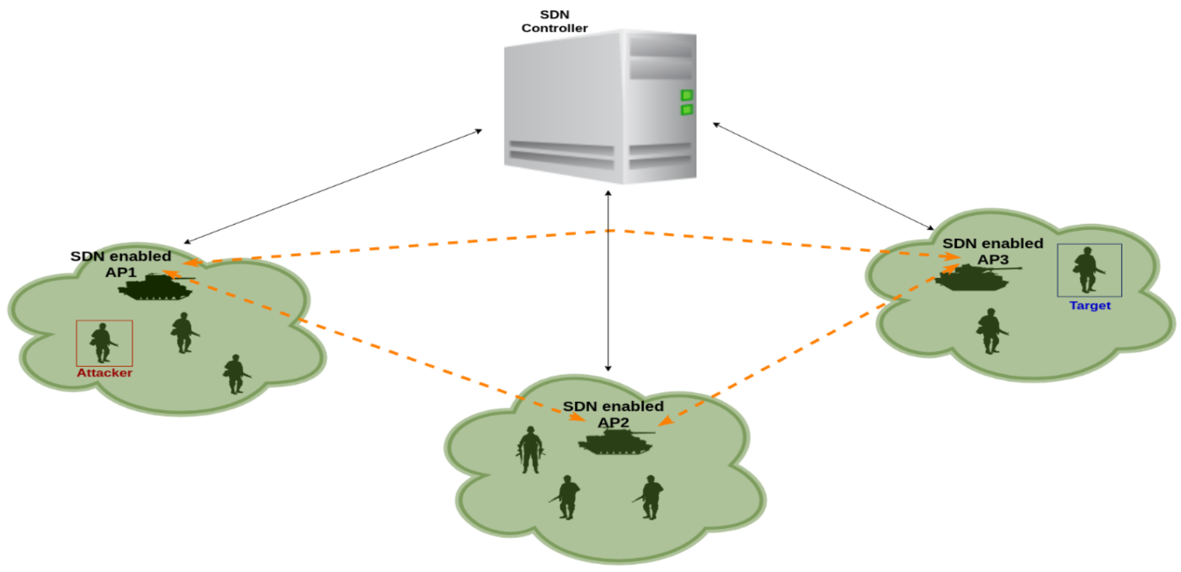


Figure 6.9 TCP flood attack scenario

- **Test 1 procedure**

The topology of the first test consists of eight nodes each connected to an Access Point (AP) as described in Chapter 5. All the nodes are connected to the network by wireless links, through Software Defined Networking (SDN) enabled APs. The attacker node attacks the target node connected to another AP using a TCP flooding attack.

2) Scenario 2: ICMP flooding attack detection

In the second test, we assume a node was successfully compromised and is used to launch an ICMP flooding attack on a specific target node. Since in tactical networks nodes are simple portable devices, it is sufficient to assume that such nodes can be captured and used to impersonate authorised nodes with hopes to disrupt network functions and interrupt communications. The purpose of this test is to evaluate the efficiency of the SFIDS in terms of detecting internal attacks or attacks arising from within the network.

- **Test 2 Procedure**

The second test consists of eight nodes connected to three access points respectively. Assuming that one of the network nodes is compromised and starts to act differently by producing a high volume of traffic flow, shown in Figure 6.10. The compromised node is used to launch an ICMP flooding attack on a target node. The compromised node (attacker) acts as the source of the attack.

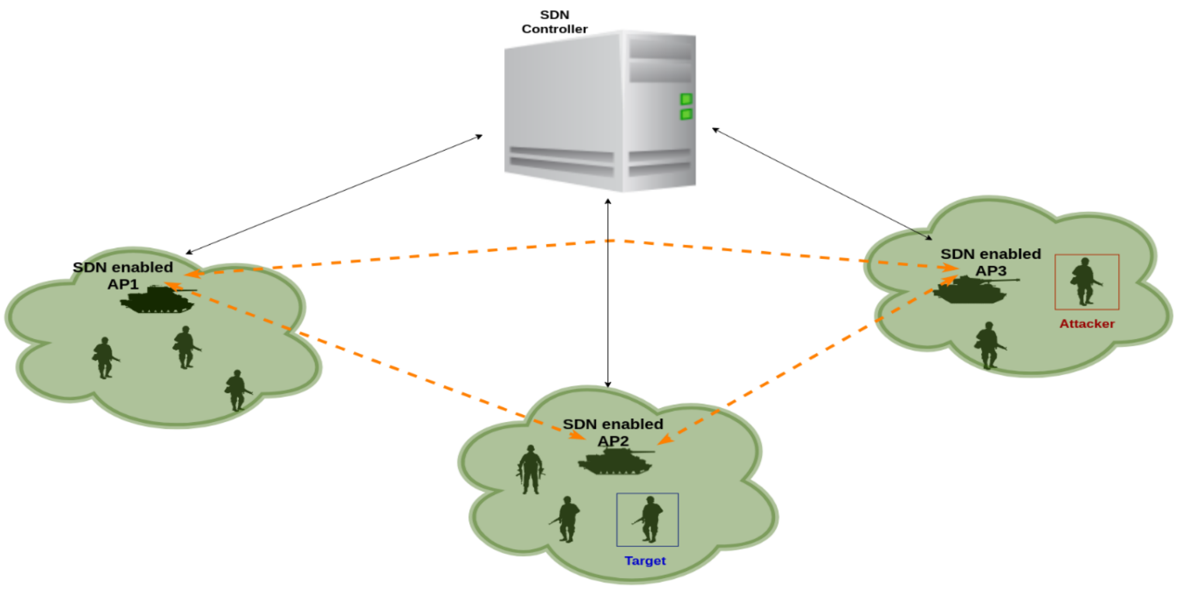


Figure 6.10 ICMP flood attack scenario

3) Scenario 3: DDoS attack detection

The last test assumes two network nodes are compromised and are used to launch a DDoS attack on a target node. A Distributed Denial-of-Service (DDoS) attack is usually used by an attacker to disable the availability of services, which can result in tactical network nodes losing network connectivity and becoming isolated from the network. The purpose of this test is to measure the effectiveness of the IDS to detect DDoS attacks originating from the internal network, as shown in Figure 6.11.

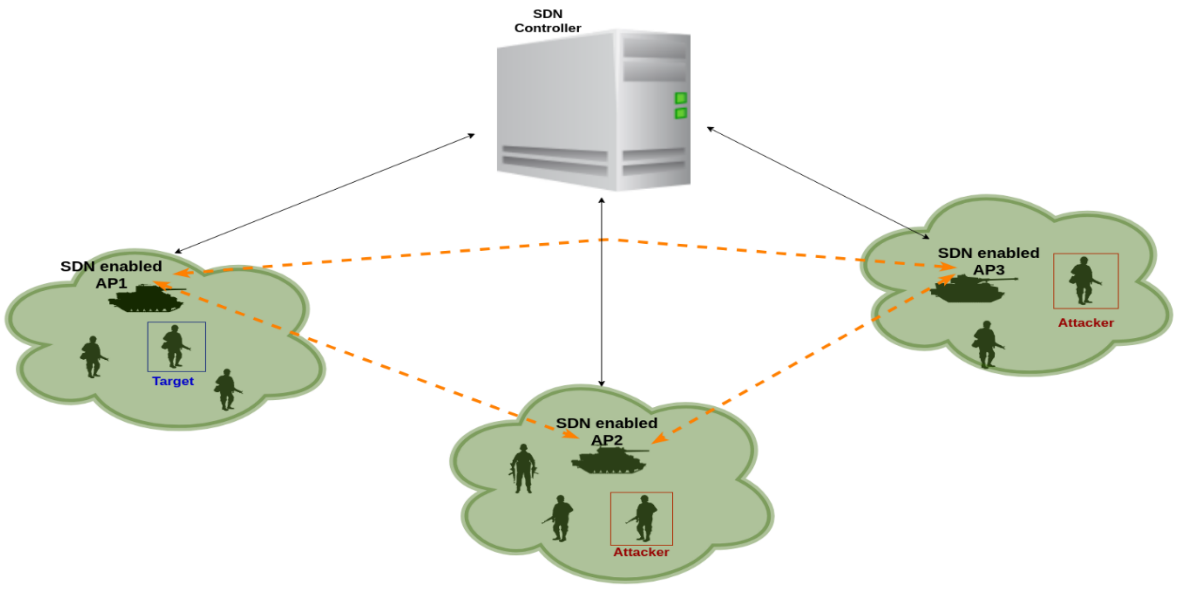


Figure 6.11 DDoS attack scenario

- **Test 3 Procedure**

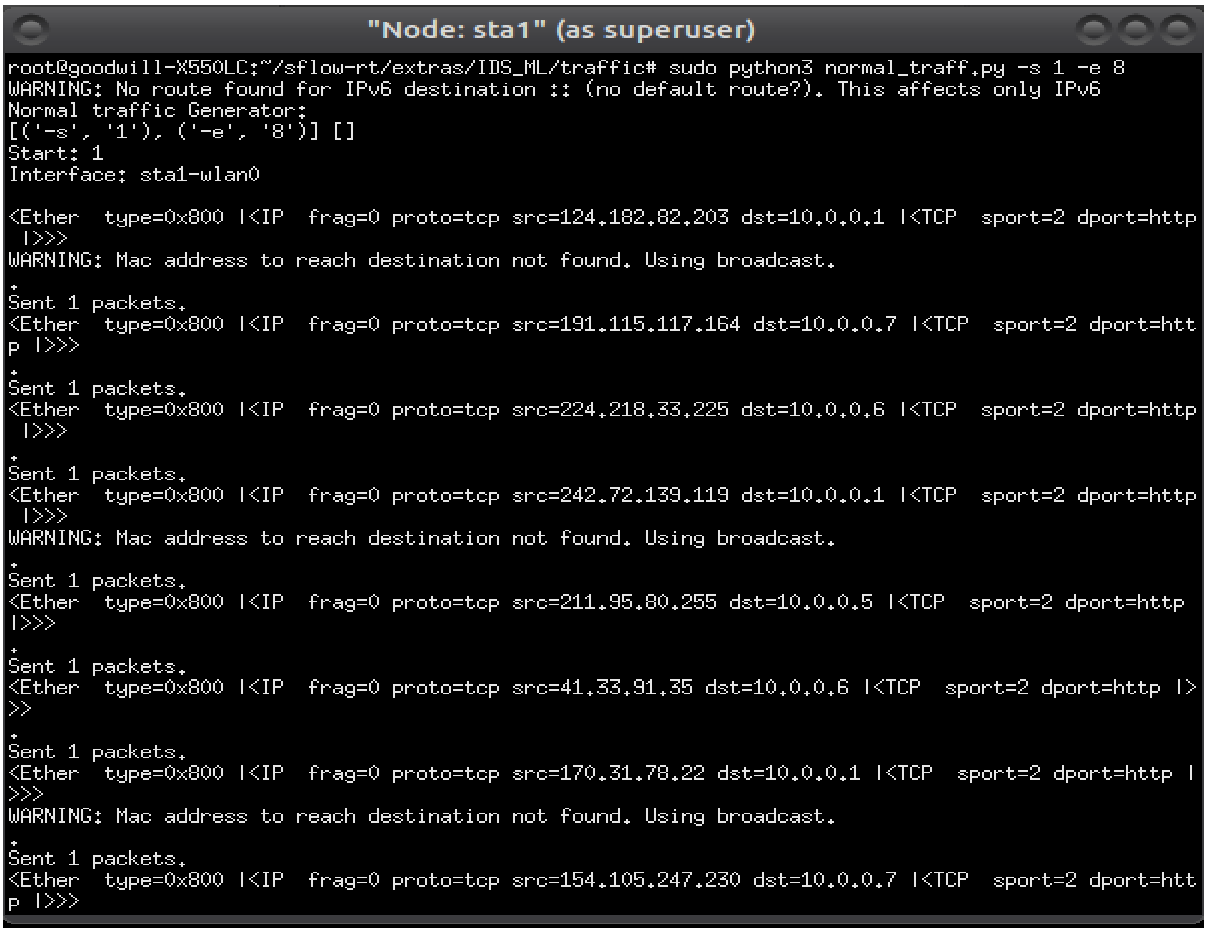
In the third test, the topology remains the same as the previous test cases. We assume two are compromised, these two compromised nodes are used to generate a high volume of traffic which amounts to a DDoS attack on a target node. At the same time, normal traffic is running between all the nodes.

6.2.3. Traffic Generation

In this work, packet generation is done using Scapy (Biondi, 2017). Scapy is a popular python packet manipulation tool used for packet generation, traffic sniffing, scanning, trace routing, probing, attacking, and packet forging. This work used Scapy to generate UDP and TCP normal packets. Also, hping3 was used to generate attack traffic for the three scenarios discussed in the previous subsection. Hping3 is a free packet generator and analyser used for security auditing of firewalls and networks.

Python version 3.5 is used to write the code for generating a random source IP address and host IP address. The "randrange" function is used to produce a uniform random float in the range [0.0, 0.1]. These numbers are then joined together to form source IP addresses. In the packet generation process, two additional parameters are set; the packet type and interval of packet generation. A combination of both UDP and TCP packets are used for the traffic. The interval was set at 0.1 seconds for the normal traffic. The adopted scripts for the normal traffic are from the work of (Stoyanova Todorova and Todorova, 2016). The packets are sent using the interface wlan0 of the respective station. Two parameters are provided to the script, which specify the range of last numbers of the destination IP addresses, and it generates traffic with those destination IP addresses of the nodes in the network. A station is randomly selected and the script to generate the normal traffic is executed. The nodes and traffic being generated is shown in Figure 6.12 and Figure 6.13.

```
sudo python3 normal_traffic.py –s 1 –e 8
```



```
"Node: sta1" (as superuser)
root@goodwill-X550LC:~/sflow-rt/extras/IDS_ML/traffic# sudo python3 normal_traff.py -s 1 -e 8
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
Normal traffic Generator:
[('-s', '1'), ('-e', '8')] []
Start: 1
Interface: sta1-wlan0

<Ether  type=0x800 |<IP  frag=0 proto=tcp src=124.182.82.203 dst=10.0.0.1 |<TCP  sport=2 dport=http
 |>>>
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=191.115.117.164 dst=10.0.0.7 |<TCP  sport=2 dport=htt
p |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=224.218.33.225 dst=10.0.0.6 |<TCP  sport=2 dport=http
 |>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=242.72.139.119 dst=10.0.0.1 |<TCP  sport=2 dport=http
 |>>>
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=211.95.80.255 dst=10.0.0.5 |<TCP  sport=2 dport=http
|>>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=41.33.91.35 dst=10.0.0.6 |<TCP  sport=2 dport=http |>
>>
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=170.31.78.22 dst=10.0.0.1 |<TCP  sport=2 dport=http |
>>>
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
<Ether  type=0x800 |<IP  frag=0 proto=tcp src=154.105.247.230 dst=10.0.0.7 |<TCP  sport=2 dport=htt
p |>>>
```

Figure 6.12 Normal traffic generator using station 1

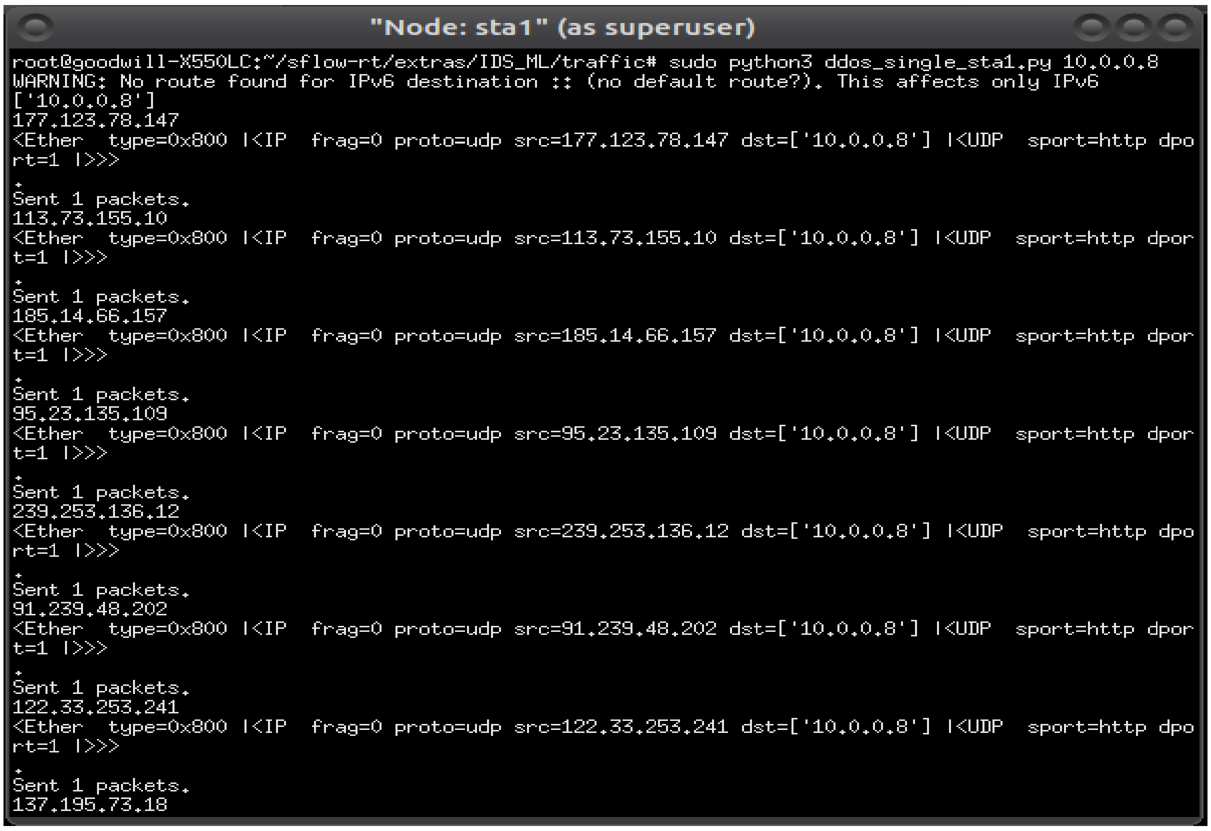
Scenario 1 TCP flooding attack traffic generation using hping3 commands

Scenario 2 ICMP flooding attack traffic generation using hping3 commands

Hping3 –c [number of packets] –d [packet-size] –w [TCP window size] –p [destination port] -flood

For Scenario 3, DDoS attacks traffic generation, a python script using Scapy was used to generate denial-of-service attacks targeted as a single node in the network, hence denying service to the target node.

sudo python3 ddos_single.py 10.0.0.8



```
"Node: sta1" (as superuser)
root@goodwill-X550LC:~/sflow-rt/extras/IDS_ML/traffic# sudo python3 ddos_single_sta1.py 10.0.0.8
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
['10.0.0.8']
177.123.78.147
<Ether  type=0x800 |<IP  frag=0 proto=udp src=177.123.78.147 dst=['10.0.0.8'] |<UDP  sport=http dpo
rt=1 |>>>
.
Sent 1 packets.
113.73.155.10
<Ether  type=0x800 |<IP  frag=0 proto=udp src=113.73.155.10 dst=['10.0.0.8'] |<UDP  sport=http dpor
t=1 |>>>
.
Sent 1 packets.
185.14.66.157
<Ether  type=0x800 |<IP  frag=0 proto=udp src=185.14.66.157 dst=['10.0.0.8'] |<UDP  sport=http dpor
t=1 |>>>
.
Sent 1 packets.
95.23.135.109
<Ether  type=0x800 |<IP  frag=0 proto=udp src=95.23.135.109 dst=['10.0.0.8'] |<UDP  sport=http dpor
t=1 |>>>
.
Sent 1 packets.
239.253.136.12
<Ether  type=0x800 |<IP  frag=0 proto=udp src=239.253.136.12 dst=['10.0.0.8'] |<UDP  sport=http dpo
rt=1 |>>>
.
Sent 1 packets.
91.239.48.202
<Ether  type=0x800 |<IP  frag=0 proto=udp src=91.239.48.202 dst=['10.0.0.8'] |<UDP  sport=http dpor
t=1 |>>>
.
Sent 1 packets.
122.33.253.241
<Ether  type=0x800 |<IP  frag=0 proto=udp src=122.33.253.241 dst=['10.0.0.8'] |<UDP  sport=http dpo
rt=1 |>>>
.
Sent 1 packets.
137.195.73.18
```

Figure 6.13 node (Station 1) generating DDoS traffic to target station with IP address 10.0.0.8

The data is labelled separately, as such the normal traffic is generated independently then given the label, such as 0, and the attack traffic is also sampled independently and labelled as 1. The methods used for sampling and pre-processing are described in the following section.

6.2.4. Flow Sampling in SDN

The next step involves data sampling, collection, and pre-processing for the IDS. The two most common protocols used for flow-stats collection in SDN environments are OpenFlow and sFlow (Giotis *et al.*, 2014). Nevertheless, in (Giotis *et al.*, 2014) the authors argued that using the native OpenFlow for periodic processing of flow-stats requests/replies affects the network performance. They observed that the approach requires a massive amount of CPU processing power from the controller. In this study, the sFlow-based approach is adopted.

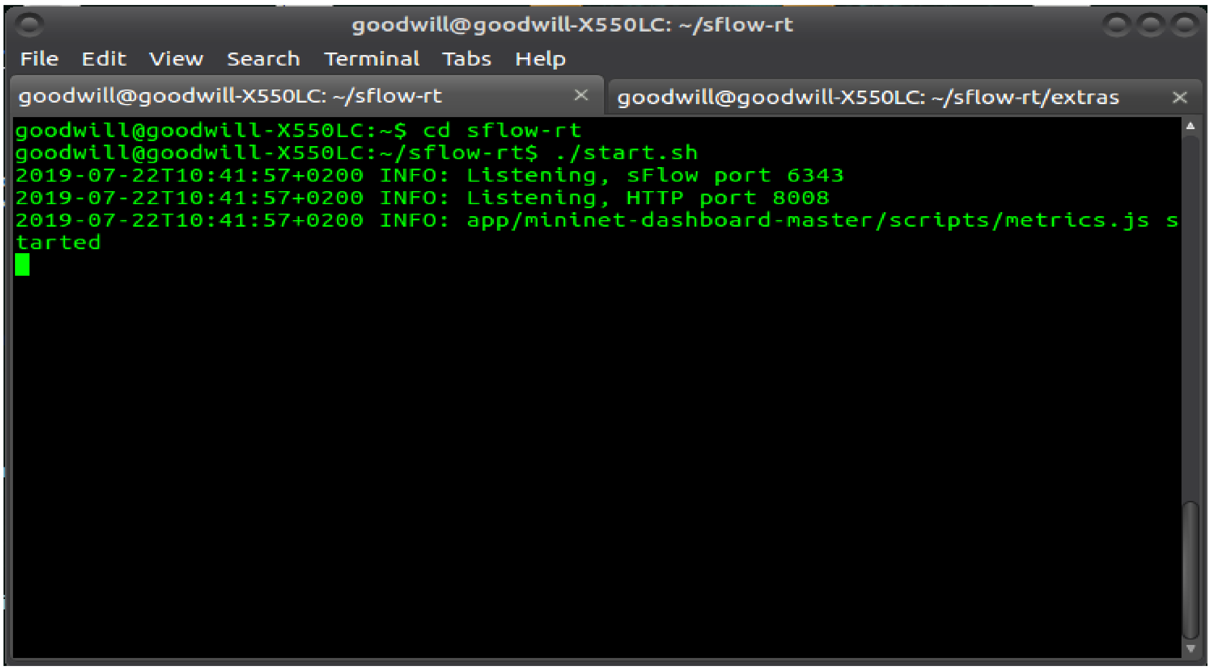
Due to the above-mentioned reasons, sFlow fits our deployment requirements as it can be deployed to power, memory, and CPU constrained devices and yield great results without requiring additional memory and processing power. In this work, the sFlow agents are installed in each of the SDN data plane network devices, in our case the Access Points (APs), see Figure 6.8. The sFlow Collector resides in the node housing the SDN controller, that is, an enormous power-budget device, possibly located at the military base or headquarters. This is done to simplify the task of passing and sharing information between the sFlow collector and the SDN controller.

All the APs in the topology are responsible for flow data sampling using the embedded sFlow agents. The APs send all the gathered data to the central sFlow collector. This method samples flow-stats data in the SDN environment without affecting network performance and consuming high processing power.

Each network device, such as AP, collects a variety of attributes for each flow in the network. However, since not all attributes are useful for intrusion detection, feature extraction methods are applied to select and extract only the useful attributes for the task of intrusion detection. Pre-processing of the data is then performed to convert the data into an acceptable format for analysis.

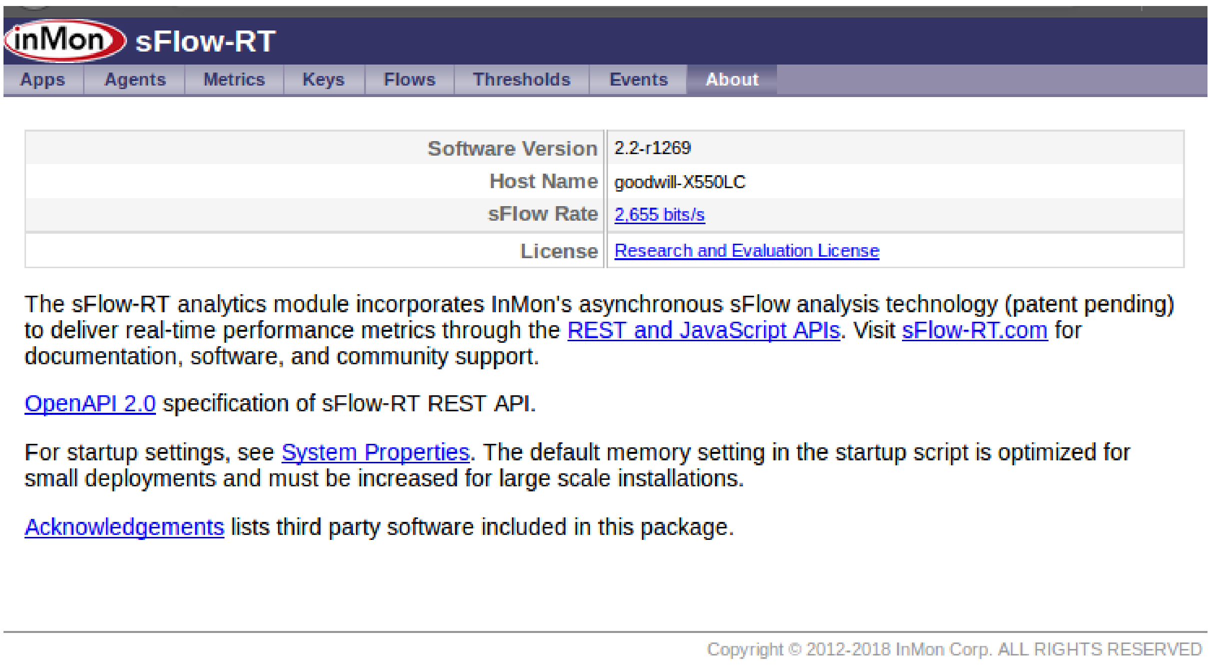
We run the sFlow-RT flow sampling tool and the CLI output is shown in Figure 6.14, while the GUI is presented by Figure 6.15.

./start.sh


goodwill@goodwill-X550LC: ~/sflow-rt
File Edit View Search Terminal Tabs Help
goodwill@goodwill-X550LC: ~/sflow-rt
goodwill@goodwill-X550LC: ~/sflow-rt/extras

```
goodwill@goodwill-X550LC:~$ cd sflow-rt
goodwill@goodwill-X550LC:~/sflow-rt$ ./start.sh
2019-07-22T10:41:57+0200 INFO: Listening, sFlow port 6343
2019-07-22T10:41:57+0200 INFO: Listening, HTTP port 8008
2019-07-22T10:41:57+0200 INFO: app/mininet-dashboard-master/scripts/metrics.js s
tarted
```

Figure 6.14 Starting sflow tool for flow stat collection in terminal



Software Version	2.2-r1269
Host Name	goodwill-X550LC
sFlow Rate	2,655 bits/s
License	Research and Evaluation License

The sFlow-RT analytics module incorporates InMon's asynchronous sFlow analysis technology (patent pending) to deliver real-time performance metrics through the REST and JavaScript APIs. Visit sFlow-RT.com for documentation, software, and community support.

OpenAPI 2.0 specification of sFlow-RT REST API.

For startup settings, see System Properties. The default memory setting in the startup script is optimized for small deployments and must be increased for large scale installations.

Acknowledgements lists third party software included in this package.

Copyright © 2012-2018 InMon Corp. ALL RIGHTS RESERVED

Figure 6.15 sFlow-RT running in the browser

6.2.5. Data Preparation and Labelling

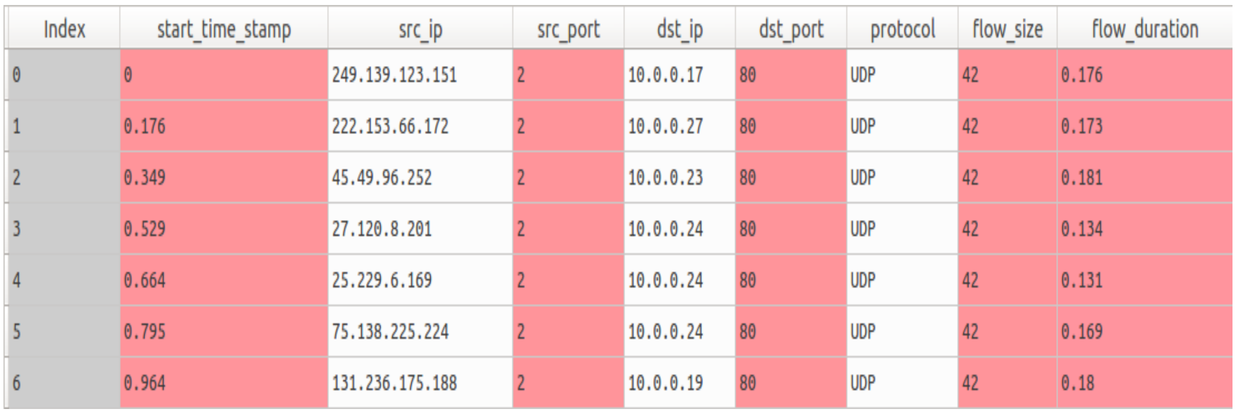
In this study, we extracted eight attributes that are considered important for classifying flow data in reference to the CIDDS-001 datasets (Markus Ring *et al.*, 2017). The attributes are presented in Table 6.1.

Table 6.1: Extracted Attributes

Attribute	**Description**
start_time_stamp	The start time flow is first seen
source_ip_addr	Flow Source IP address
source_port	Flow Source port
destination_ip_addr	Flow Destination IP address
destination_port	Flow Destination port
transport_protocol	Transport Protocols used (e.g. TCP, UDP, or ICMP)
flow_size	Flow size, average size of the packets seen in the flow
flow_duration	The duration of the flow

Hence, a combination of multiple instances extracted from the network results in a dataset composed of categorical and string attributes. After selecting these features, we apply pre-processing steps to convert the generated dataset to a format acceptable to classification algorithms.

Usually, Machine Learning algorithms are trained using prepared data that is in the form of numerical vectors which are called feature vectors. Therefore, before this data(refer to Figure 6.16) can be used to train a Machine Learning model, the data must be transformed into a feature vector of numeric values. The steps taken to label and pre-process the generated data are explained below.



Index	start_time_stamp	src_ip	src_port	dst_ip	dst_port	protocol	flow_size	flow_duration
0	0	249.139.123.151	2	10.0.0.17	80	UDP	42	0.176
1	0.176	222.153.66.172	2	10.0.0.27	80	UDP	42	0.173
2	0.349	45.49.96.252	2	10.0.0.23	80	UDP	42	0.181
3	0.529	27.120.8.201	2	10.0.0.24	80	UDP	42	0.134
4	0.664	25.229.6.169	2	10.0.0.24	80	UDP	42	0.131
5	0.795	75.138.225.224	2	10.0.0.24	80	UDP	42	0.169
6	0.964	131.236.175.188	2	10.0.0.19	80	UDP	42	0.18

Figure 6.16 Flow data sampled from SDN network

A. Observations Distribution

After finalising the experimental setup and making sure everything works well, a python script using Scapy version 3.0.0 to generate normal network and attack traffic is executed. The generated traffic is sampled using the defined sFlow mechanism. The normal traffic is given the label **0,** and the attack traffic is labelled **1**. In total the dataset used for the machine learning classifiers for all three scenarios is shown in Table 6.2. While Figure 6.17 shows a graphical representation of the data.

Table 6.2 Data Distribution for three scenario

	Normal	**Malicious**	**Total**
Scenario 1	95 993	33 802	129 795
Scenario 2	91 269	38 526	129 795
Scenario 3	95 993	33 802	129 795

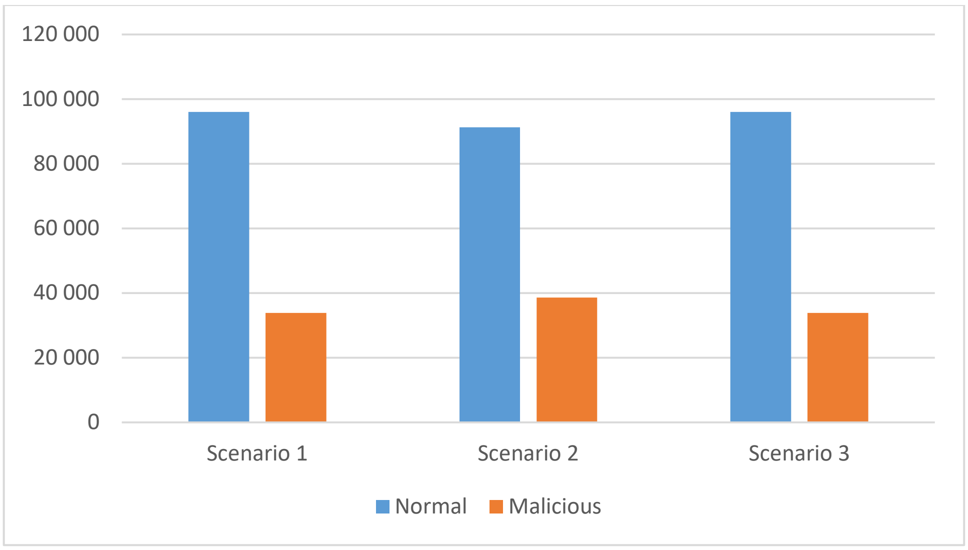


Figure 6.17 Distribution of Normal and Malicious network flow from SDN

i. Encoding Categorical features

Categorical features present challenges to machine learning algorithms when building models. While other ML packages can transform categorical data into numeric data automatically using default embedded methods, many other Machine Learning packages do not support such input, for example, the adopted Scikit-learn library (Pedregosa, Weiss and Brucher, 2011). However, the Scikit-learn library contains classes that can be used to encode categorical data. For example, OneHotEncoder and FeatureHashing.

- **OneHotEncoder** (OHE) is a representation method that takes each category value and turns it into a binary vector of size |i|, where "i" is the number of categories, and all columns are assumed to equal zero besides the category column. The limitations of OHE are that its representation produces high dimensionality, which causes an increase in the model's training time, serving time, and memory consumption. This method can easily cause the models to overfit the data.
- **FeatureHashing** also known as the hashing trick, is a method for turning arbitrary features into a sparse binary vector. It has a standalone hash function that does not require a pre-built dictionary of possible categories to function, and this makes it extremely efficient. In a simple implementation, the user specifies the desired output dimensionality, the method then hashes the input value into a number, then divides it by the specified dimensionality and returns the remainder, **R**. The final output is a vector of zeros with a one in the index **R**. FeatureHashing has low dimensionality since the user specifies it, making it efficient in processing time and memory. The dimesion used in this study are shown in Table 6.3

Table 6.3: FeatureHashing dimensionalities applied to each of the categorical features

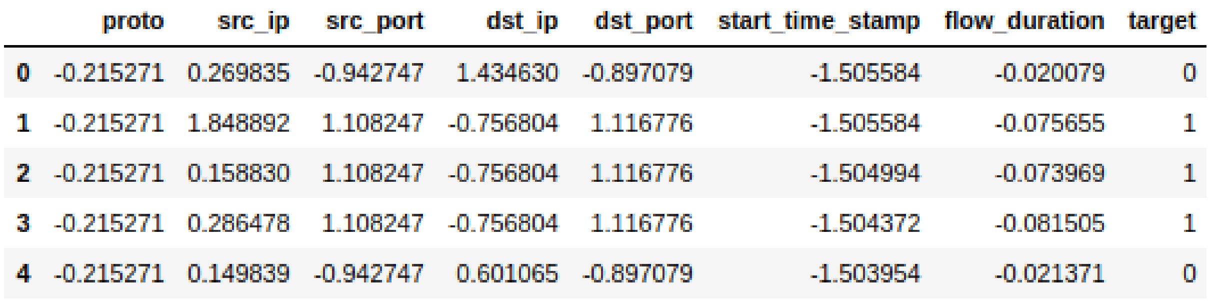
Feature	**Dimensionality**
source_ip_addr	10
source_port	10
destination_ip_addr	10
destination_port	10
transport_protocol	4

This work adopted FeatureHashing as the encoding method for all the categorical features of the dataset. Each categorical feature is hashed on its own, and the resulting hash function is later used to hash each new instance's categorical features. For example, each ***source_ip_addr*** of an instance is hashed using the same hash function used to hash the training data.

6. Dimensionality Reduction and Scaling

After encoding the mentioned features from the above step, the FeatureHashing method returns each feature with the specific dimensions, see Table 6.3 above. We then apply dimensionality reduction using the Principal Component Analysis (PCA) technique. PCA is an unsupervised linear transformation technique widely used for feature extraction and dimensionality reduction across different fields. PCA finds the direction of maximum variance in high-dimensional data and projects it into a new subspace with equal or fewer dimensions than the original data. The final output from FeatureHashing and PCA returns a single value for each feature. Finally, all the features are brought together to form a dataset in an appropriate format for machine learning algorithms in Scikit-learn.

In addition, if a feature has a variance magnitude greater than the variance of other features, that particular feature might dominate the other features in the dataset, which will introduce a drawback to the Machine Learning model built using the data. Therefore, we set the variance of the features to the same range using **StanderdScaler** class found in Scikit-learn. This scales the features to a range centred around zero. The final output dataset is shown in Figure 6.18



	proto	src_ip	src_port	dst_ip	dst_port	start_time_stamp	flow_duration	target
0	-0.215271	0.269835	-0.942747	1.434630	-0.897079	-1.505584	-0.020079	0
1	-0.215271	1.848892	1.108247	-0.756804	1.116776	-1.505584	-0.075655	1
2	-0.215271	0.158830	1.108247	-0.756804	1.116776	-1.504994	-0.073969	1
3	-0.215271	0.286478	1.108247	-0.756804	1.116776	-1.504372	-0.081505	1
4	-0.215271	0.149839	-0.942747	0.601065	-0.897079	-1.503954	-0.021371	0

Figure 6.18 Prepared data, after employing FeatureHashing, PCA, and scaling

6.3. Dataset Exploration and Visualisation

The next step after data collection in the data science methodology (Zumel and Mount, 2014) is data visualisation and analysis, which focuses on identifying relationships and characteristics of the collected data. In order to determine the relationship between flow-based data features and the output variable, feature selection, correlation heatmap, and data distribution techniques were used.

6.3.1. Feature importance

Feature selection is the task of identifying the most related features from the dataset. This can result in a reduction in the number of features to achieve better accuracy. Feature selection can help improve model accuracy, reduce overfitting, and reduce training time. In this study Feature importance was calculated as the score for each feature of the data, the higher the score, the more relevant is the feature towards the output variable. Feature importance was calculated using tree Extra Tree Classifier from the Scikit-learn library (Pedregosa, Weiss and Brucher, 2011). Figure 6.19 illustrates the most important features in the flow-based dataset collected from the simulated SDN-based tactical MANET. The created visuals suggest that ***src_port*** is the most highly-ranked feature in the dataset.

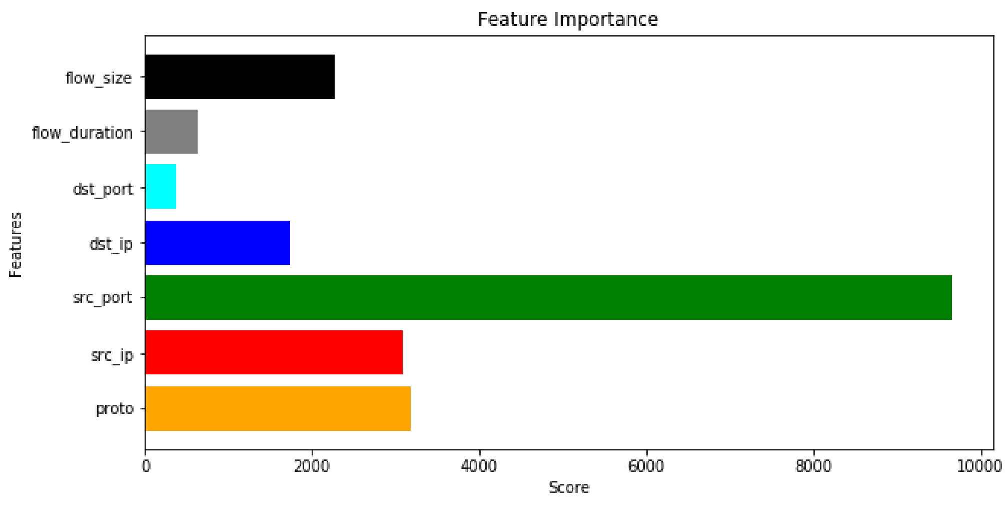


Figure 6.19 Scenario1 dataset feature importance

6.3.2. Correlation Matrix Heatmap

Correlation defines how each of the features is related to each other and or the target variable. The correlations can be negative or positive. A negative correlation means an increase in one value of the feature decreases the other value, while positive correlation means an increase in one value of the feature increases the value of the other feature. The feature with more relation to the target variable can help improve model accuracy, while the ones with a high relation to each other can affect model performance. The correlation between the features for the first scenario is presented in Figure 6.20.

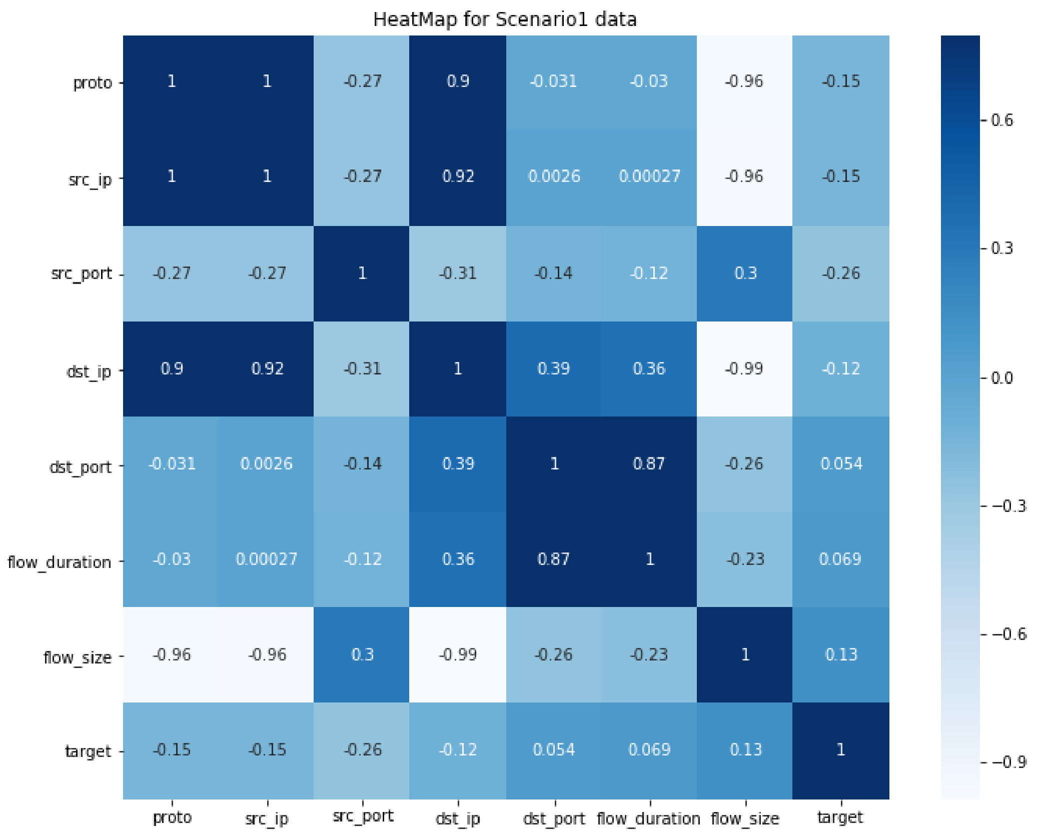


Figure 6.20 Scenario 1 correlation heatmap of features

6.3.3. Dataset distribution

Data distribution visualisation helps in understanding the characteristics of the data as one can learn the difference between the data and the target variables, such as normal flow data and malicious flow data. Data distribution visualisations, using the most relevant features identified using feature importance above, were plotted to help understand the relations between them. For example, the distribution of source port numbers is shown in Figure 6.21.

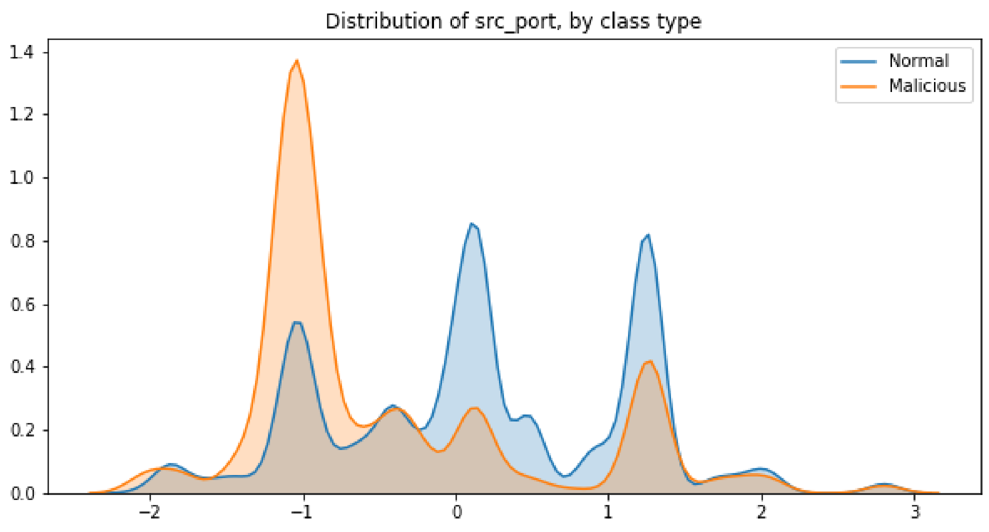


Figure 6.21 Distribution of source port by class type for scenario1 dataset

6.4. Summary

This chapter presented how to operationalise SFIDS, an SDN-based IDS that uses ML for data classification (presented in Chapter 5). The model is realised through a proof-of-concept prototype that can be deployed in an SDN environment using Mininet-Wifi. To evaluate the effectiveness of the model, three test scenarios were simulated and network flow data was collected and pre-processed. The data was sampled and collected from the network using sFlow (a flow sampling tool). Each of the normal and attack traffic data was collected independently and labelled accordingly. During pre-processing, FeatureHashing was used to encode categorical features while PCA was used for feature reduction. Furthermore, this chapter also presented the observed relations between the features of the data, which include feature importance, and correlations. Thus, after the collection and preparation of the data, the next chapter (Chapter 7) evaluates the effectiveness of the SFIDS approach using two ensemble learning techniques, namely; AdaBoost and RF.

Chapter 7: Results and Discussions

For the designed artefact (SFIDS) to be rigorously assessed, methodologies from the knowledge base must be used (Hevner *et al.*, 2004). In addition, the method must match properly with the artefact and the nominated evaluation metrics. Examples of design evaluation methods include; observational, analytical, experimental, testing, and descriptive methods. The observational method can be the study of an artefact in-depth in the deployment environment, examples include Case Study and Field Study. Analytical methods study the structure of the artefact for static qualities, such as complexity. Experimental methods examine the artefact in a controlled environment for quality and usability, including executing artefacts using artificial data, known as Simulation. In this study, the experimental method is used, as the artefact is evaluated using artificial flow data, gathered from a simulated SDN-based tactical MANET environment. Thus, this chapter strives to provide and present evidence of the effectiveness of an ML-based FIDS model for tactical mobile networks called SFIDS, described in Chapter 6.

7.1. SFIDS Review

SFIDS uses the SDN paradigm capabilities, which include network global view and real-time flow data extraction, to gather and process network data from nodes connected to the network. As in Section 5.3, each network device or node in the SDN infrastructure layer is embedded with a flow sampling agent that periodically samples flow data passing through the network. The data is then exported using the UDP protocol to a centralised collector residing in the SDN control layer. The data collector then prepares the data for analysis and classification in the SFIDS's Decision Engine (DE).

The DE is made up of an ensemble ML model which is located in the SDN application layer. In this study, two models were tested for their effectiveness in detecting intrusive activities in the tactical network, presented in Section 7.2. The decision engine takes each sample prepared by the collector and classifies it as either normal or malicious. If the sample is malicious then the decision engine generates an alert and notifies the SDN controller to take appropriate actions. For example, incident handling and enforcing rules to mitigate the attack. This process takes place in real-time allowing timely detection of attacks and mitigation as soon as possible before further harm is done to the network and its users.

A closer look at the SFIDS system and the different operations that occur within it for intrusion detection in tactical networks is presented in Figure 7.1 and the different colour arrows are explained below:

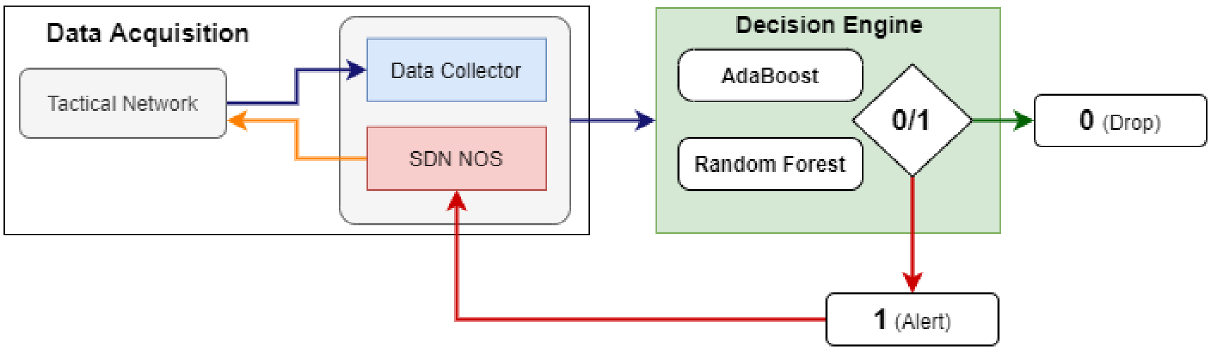


Figure 7.1 SFIDS model for tactical networks

- Blue: Flow data is sampled from the Tactical network and exported to a centralised collector that pre-processes and prepares each sample in real-time for the Decision engine.
- Green: The decision engine accepts the prepared sample as input and classifies it as 0: normal or 1: malicious. If the sample is classified as normal, it is dropped and a new sample is classified.
- Red: The decision engine classifies the input sample as 1: malicious, it sends an alert to the SDN Network Operating System (the controller) which can send a flow to either block all traffic originating from the device owning the sample or redirect all from that device to a sandbox to monitor its activities.

7.2. ML Models Architecture and Parameters

To ensure rigour in the development of the machine learning classification model (Hevner *et al.*, 2004), this study adopted and used the data science life cycle to develop and evaluate the ensemble ML models, as discussed in Chapter 3, Subsection 3.2.3. The ML classification models play a very important role in the proposed FIDS since they are responsible for the data analysis and intrusion detection functionality. For the evaluation of the SFIDS, ensemble algorithms Random Forest and AdaBoost were considered since they were recognised as better performing in Chapter 4. This Section presents the architecture and choice of parameters used for each of those ensemble algorithms used for SFIDS's DE.

7.2.1. Adaptive Boosting

AdaBoost is commonly known as a meta-estimator that starts by training a classifier on the original dataset and then trains additional copies of the classifier on the same dataset while adjusting the weights of the incorrectly classified instances. In this study, an AdaBoost model using Decision Tree as the base estimator is trained and tested for its effectiveness in detecting intrusions in SDN-based tactical MANET

In AdaBoost, the main parameter to tune to obtain good performance is the n_estimator parameter which controls the number of base estimators. To determine the most optimal number of estimators for good performance, a comparison of the classification error of a boosted Decision Tree using Real AdaBoost and Discrete AdaBoost was conducted. The results indicate that Real AdaBoost performed better than Discrete AdaBoost as it obtained a smaller train and test error. The results also indicated that after 50 trees, both the algorithms converge with 18.8 % error for the real AdaBoost and 18.9 % error for the Discrete AdaBoost respectively, as shown in Figure 7.2.

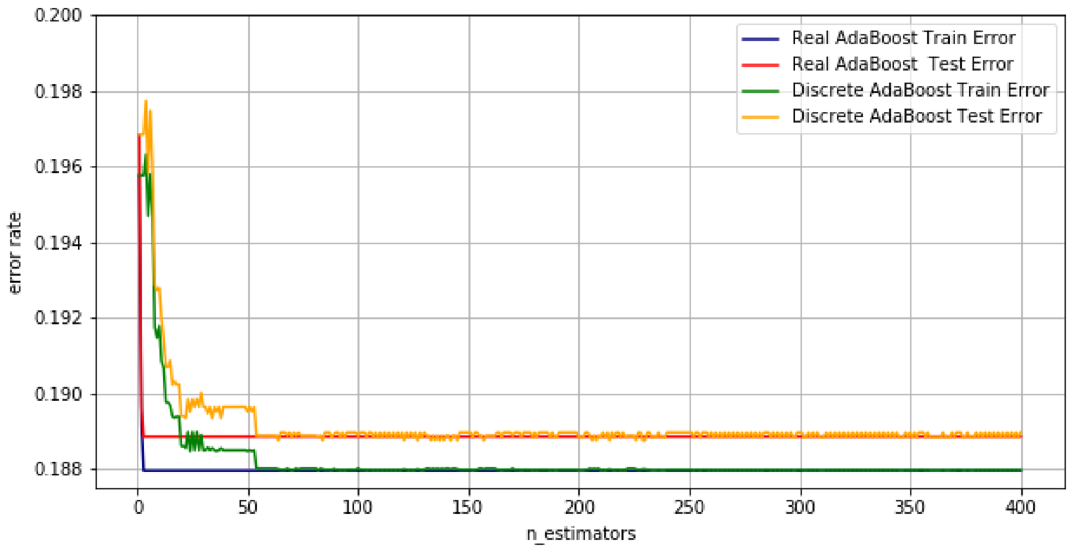


Figure 7.2 Real and Discrete AdaBoost error rate per number of estimators

In the AdaBoost classification model construction, The Real AdaBoost of Scikit-learn with 100 estimators was used, as the setup demonstrated minimum error which implies the capability of a high detection rate. Besides using 100 estimators to ensure a high detection rate. The parameters used to ensure model simplicity and less complexity are shown in Table 7.1.

Table 7.1 AdaBoost configuration parameters

AdaBoost parameters	
Base estimator	Decision Tree
N_estimators	100
Learning rate	1
Algorithm	SAMME.R

7.2.2. Random Forest

Random Forest is an ensemble model based on Bagging as the ensemble method and Decision Tree as the base method. In this subsection, the Random Forest algorithm or steps are presented to help understand how it can be fitted to the training dataset and tuned for effective performance.

Step 1: Select n random subset from the training set

Step 2: Train n Decision Trees

One random subset of the data is used to train one Decision Tree

Optimal splits for each Decision Tree are based on a random subset of the features. For example, if there are 10 features, the method randomly selects 5 out of the 10 to split.

Step 3: Each tree predicts the records in the test set independently

Step 4: Make the final prediction

For each record in the test data, use the class with the majority vote as the record's final prediction.

To achieve optimal performance from a Random Forest model, it needs to be thoroughly tuned to properly fit the dataset to ensure better performance. As mentioned in the section above, Random Forest is trained using bootstrap aggregation, where each new tree is fit from a bootstrap sample of the observations (Scikit-learn, 2019, v0.21.3). In that sense, we let this training observation be; $z_i = (x_i, y_i)$. Thus, we can calculate the average error for each z_i using predictions from the trees that do not contain z_i in their respective bootstrap sample. This error is called the out-of-bag (OOB) error. The OOB error is measured at the addition of each new tree during the training process. Hence this can allow practitioners to approximate a suitable value of n_estimators at which the error stabilises.

To identify the most appropriate configuration of the Random Forest Model, the OOB error was measured while incrementing the number of estimators. In the setup, three random forest models considering a different number of features, such as “sqrt”, “log2”, and “all the features” were compared to determine the best split and appropriate number of trees. The results indicated all the feature splits considered resulted in the same performance. Also, optimal performance from the model is obtained when it has 25 or more estimators. Figure 7.3. presents the performance of the Random Forest model with a different number of estimators.

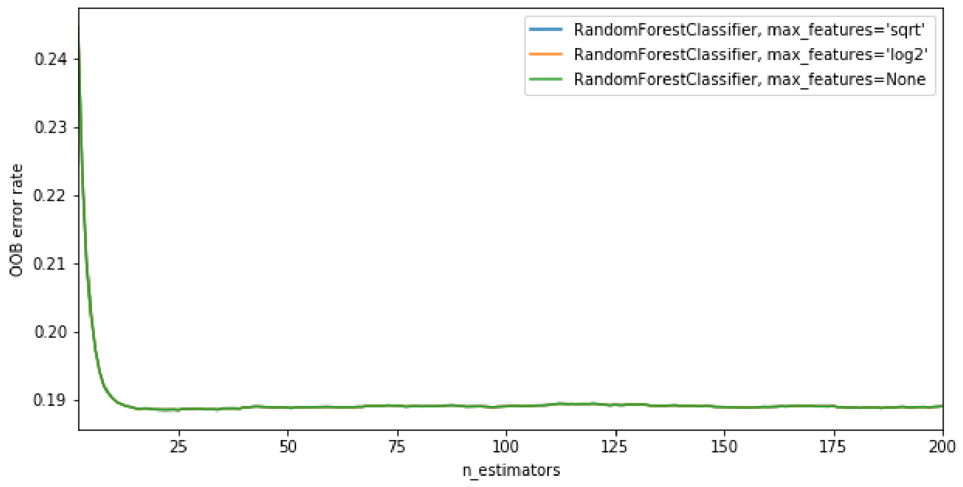


Figure 7.3 Random Forest OOB error

The insights obtained from the results presented in Figure 7.3 were used to devise the configuration of an optimal Random Forest model for the presented scenario. Table 7.2 presents the configuration used for the random forest model.

Table 7.2 Random Forest configuration parameters

Random Forest Parameters	
N_estimator	100
Criterion	Gini
Max_depth	None
Min_sample_split	2
Min_sample_leaf	0
Max_features	None

As with the AdaBoost classification model presented above, three scenarios are used to validate and evaluate the model. The test cases and performance results of the Random Forest classification model are presented next.

7.3. Performance Evaluation

The evaluation of the FIDS for tactical MANETs was conducted using three scenarios, see Chapter 6 Section 6.2. The results are presented based on those three scenarios, namely; TCP flood detection, ICMP flood detection, and DDoS attack detection.

7.3.1. TCP flood attack detection

The first scenario is concerned with evaluating the model's effectiveness in detecting TCP flood attacks issued by a legitimate network node, assuming it was physically captured and used to launch the attack. Table 7.3 shows the configuration for the training of the model, such as the dataset distribution for both training and testing of the model. From Table 7.3, 86 962 records were used for training and 42 833 instances for testing, with a total of 129 795 instances.

Table 7.3 Scenario 1 dataset configuration

Scenario 1	**Normal**	**Malicious**	**Total**
Train	64 416	22 546	**86 962**
Test	31 577	11 256	**42 833**
Total	**95 993**	**33 802**	**129 795**

Given the configuration and the parameters presented in Table 7.2 and Table 7.3, and the dataset configuration in Table 7.4, the confusion matrix of the resultant models after training with Scenario1 data is shown in Figure 7.4 and Figure 7.5. The AdaBoost model was able to correctly classify 6092 malicious instances as malicious and 5164 malicious instances as normal, while correctly classifying 29223 normal instances as normal, and incorrectly classifying 2354 normal instances as malicious. The Random Forest model, correctly classified 7081 malicious instances as malicious and 4175 malicious instances as normal. This suggests that the Random Forest model has a higher true positive rate than the AdaBoost model. However, the Random Forest model has a lower true negative rate than the AdaBoost model.

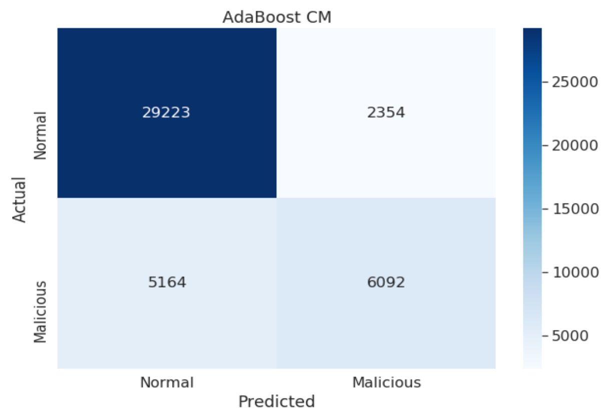


Figure 7.4 AdaBoost Confusion Matrix

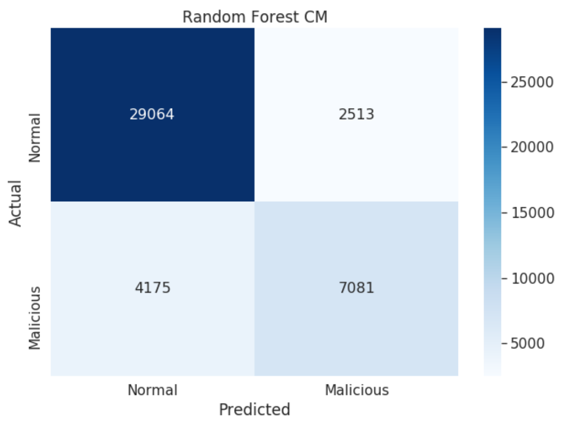


Figure 7.5 Random Forest Confusion Matrix

Breaking down these figures, the accuracy of the AdaBoost and Random Forest models can be calculated using the accuracy formula as,

$$Accuracy_{AdaBoost} = \frac{(TP+TN)}{(TP+TN+FP+FN)} = \frac{6092+29223}{6092+29223+5164+2354} = \frac{35315}{42833} = 0.8241 \approx 82\%,$$

$$Accuracy_{RF} = \frac{(TP+TN)}{(TP+TN+FP+FN)} = \frac{7081+29064}{7081+29064+4175+2513} = \frac{36145}{42833} = 0.8438 \approx 84\%$$

While both the accuracies are reasonable, Random Forest indicates higher detection accuracy. Further analysis of the results was conducted to ensure the models are independent of the Accuracy Paradox. The precision and recall are calculated for both the normal and malicious instances. For example, the values for malicious detection are calculated from the confusion matrix as;

$$Recall\_adaBoost = \frac{TP}{(TP+FN)} = \frac{6092}{6092+5164} = \frac{6092}{11256} = 0.541 \approx 54\%$$

And,

$$Precision_{adaBoost} = \frac{TP}{(TP+FP)} = \frac{6092}{(6092+2354)} = \frac{6092}{8446} = 0.721 \approx 72\%$$

The precision and recall for the normal traffic data can also be calculated using a similar approach. All the precision, recall, f1-score, and cross-validation score values for both AdaBoost and Random Forest are shown in Table 7.4 and Table 7.5 respectively.

Table 7.4 AdaBoost results for scenario 1

AdaBoost	Normal	Malicious	Weighted Avg
Precision	85%	72%	82%

Recall	93%	54%	82%
F1-score	89%	62%	82%
Cross Val Score	82.42%		

Table 7.5 Random Forest results for scenario 1

Random Forest	Normal	Malicious	Weighted Avg
Precision	87%	74%	84%
Recall	92%	63%	84%
F1-score	90%	68%	84%
Cross Val Score	84.42%		

Although the AdaBoost model predicted 11256 instances as malicious instead of 8446 instances, it was able to predict malicious instances correctly 54% of the time, while the RF model predicted malicious instances correctly 63% of the time. In addition, out of 8446 malicious instances, AdaBoost was able to correctly identify only malicious instances 72 % of the time, while RF only identified relevant instances 74% of the time. However, the cross-validation score for AdaBoost is 82.4% while RF obtained a validation score of 84.4%. Graphically the difference of obtained precision values for both models is illustrated in Figure 7.6. From Figure 7.7, it is evident that random forest is more precise in detecting normal and malicious traffic than AdaBoost when detecting TCP flood attacks in the network. The poor performance of AdaBoost can be due to the fact that AdaBoost is very sensitive to noisy data and outliers while in RF, outliers can easily be detected and don't affect the performance.

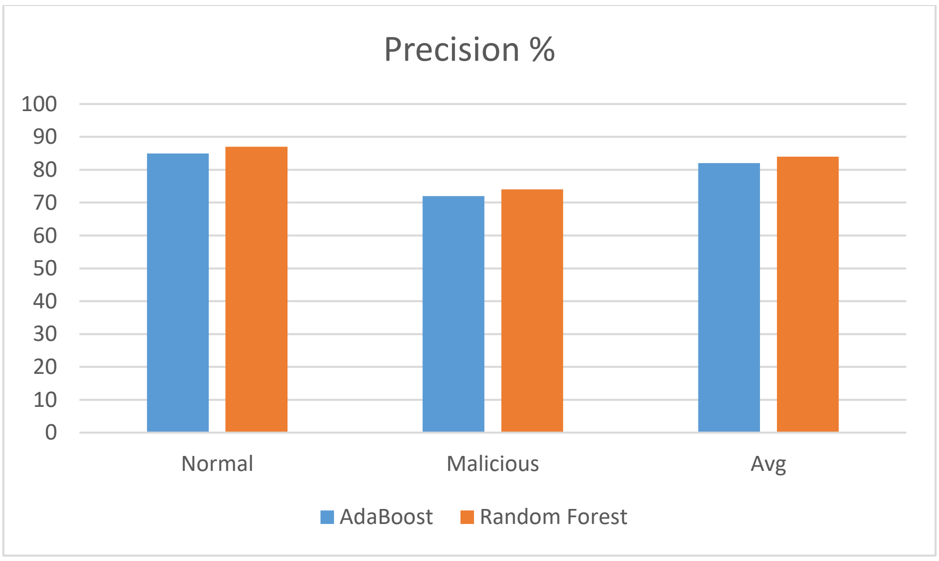


Figure 7.6 Comparison of precision values in TCP flood detection

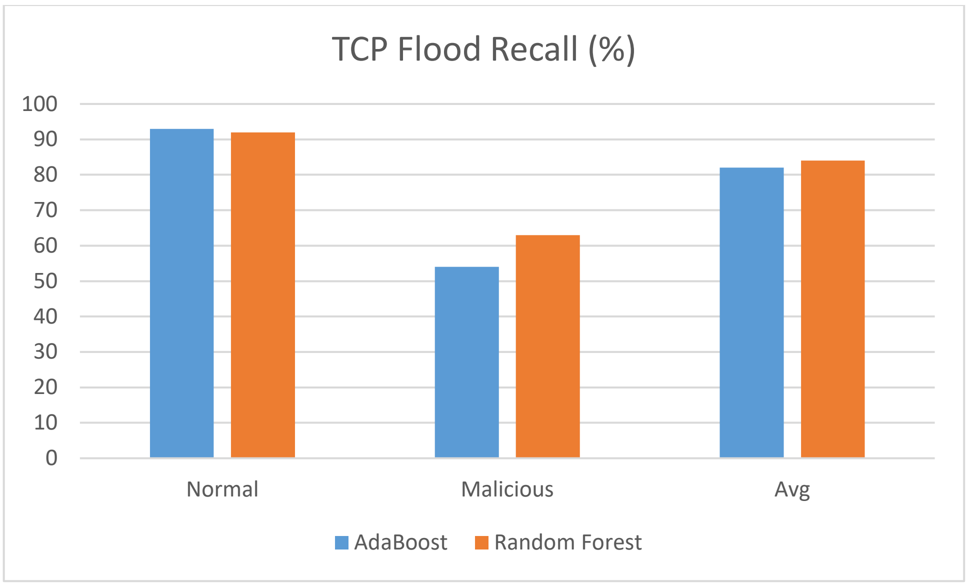


Figure 7.7 AdaBoost and Random Forest Recall results for TCP flood attack

When the recall is considered for both models, the results obtained indicate that Random Forest obtained higher precision than AdaBoost. However, for the normal traffic, little difference is observed compared to the recall of malicious traffic. The f1-score, which is the harmonic mean between precision and recall, is also presented in Figure 7.8 for both normal and malicious traffic. Considering that Random Forest indicated better performance in both precision and recall, the f1-score also emphasises Random Forest's supremeness in detecting TCP flood attacks in the SDN environment.

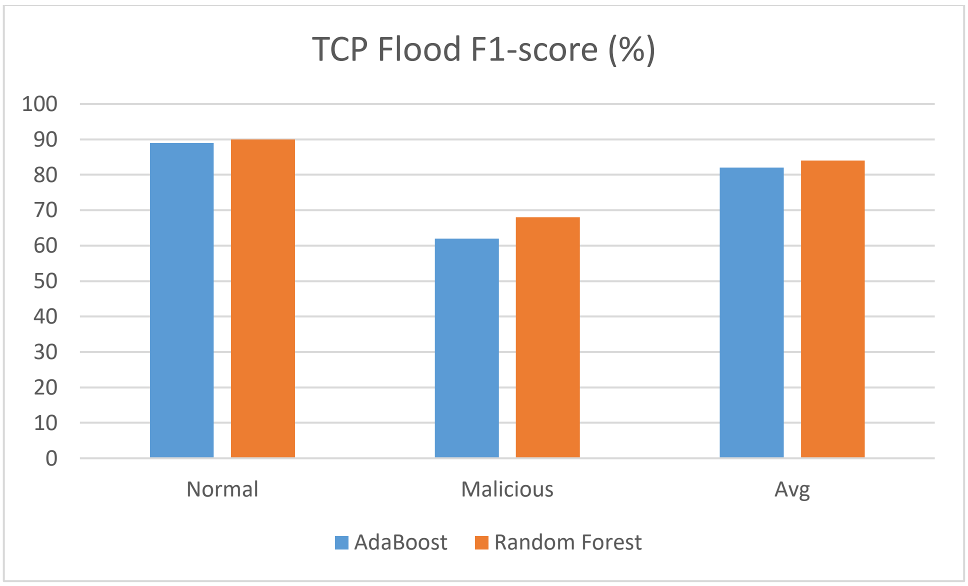


Figure 7.8 AdaBoost and Random Forest F1-score for TCP flood detection

To further show that the Random Forest model performed better than the AdaBoost model, it obtained an 87 % area under the ROC curve, while the AdaBoost model obtains 85%, as shown in Figure 7.9. Although Random Forest outperforms AdaBoost in this scenario, the difference is minimal, thus both models are effective.

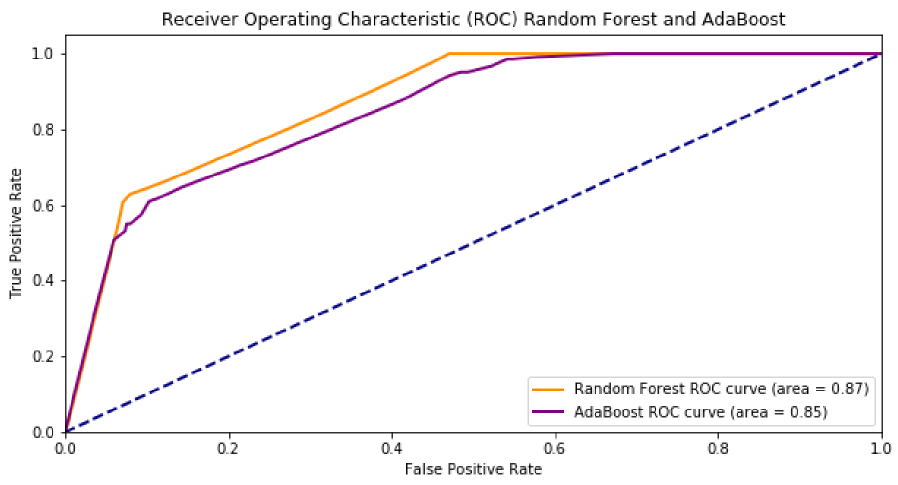


Figure 7.9 AdaBoost and Random Forest ROC curves

7.3.2. ICMP flooding attack detection

The second scenario also consists of a total of 129 795 samples, where 91 269 are normal instances, and the remaining 38 526 instances are malicious. The models, in this case, were tested using 42 833 instances containing 30 078 normal instances and 12 755 malicious instances. The training and testing data configurations are shown in Table 7.6. The resultant AdaBoost model based on those configurations is evaluated.

Table 7.6 Scenario 2 dataset configuration

Scenario 2	**Normal**	**Malicious**	**Total**
Train	61 191	25 771	**86 962**
Test	30 078	12 755	**42 833**
Total	**91 269**	**38 526**	**129 795**

The dataset used was generated based on Scenario 2, which simulated an ICMP flood attack from a node in the SDN tactical network. The AdaBoost and Random Forest models were created and tested based on the configuration presented in Table 7.7. The confusion matrix of the resultant models is presented in Figure 7.10 and Figure 7.11. The AdaBoost model misclassified 4124 normal instances as malicious and 4346 malicious instances as normal. In this regard AdaBoost classified less malicious instances.

The Random Forest model misclassified 2057 normal instances as malicious and 6475 malicious instances as normal. More than half of the malicious instances were classified as normal by the Random Forest classifier. From the confusion matrix presented, Figure 7.10 and Figure 7.11, each model's accuracy can be calculated using the same approach used in the first scenario.

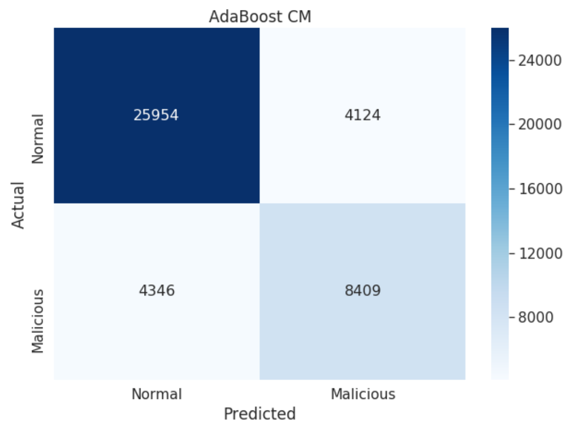


Figure 7.10 AdaBoost Confusion Matrix

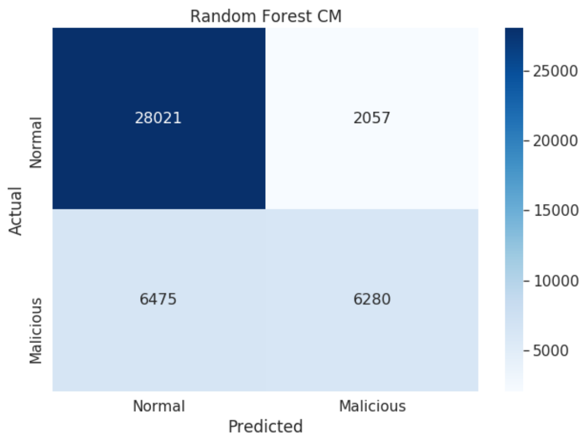


Figure 7.11 Random Forest Confusion Matrix

The precision, recall, f1-score, and cross-validation scores for AdaBoost and Random Forest are presented in Table 7.7 and Table 7.8.

Table 7.7 AdaBoost results for ICMP flood detection

AdaBoost	Normal	Malicious	Weighted Avg
Precision	86%	67%	80%
Recall	86%	66%	80%
F1-score	86%	67%	80%
Cross Val Score	80.23%		

Table 7.8 Random Forest results for ICMP flood detection

Random Forest	Normal	Malicious	Weighted Avg
Precision	81%	75%	79%
Recall	93%	49%	80%
F1-score	87%	60%	79%
Cross Val Score	80.23%		

The results obtained for scenario 2, which aims to validate the effectiveness of the models in detecting an ICMP flood attack indicate that Random Forest can precisely detect both normal and malicious traffic compared to the AdaBoost model, this is illustrated by the visualisation in Figure 7.12.

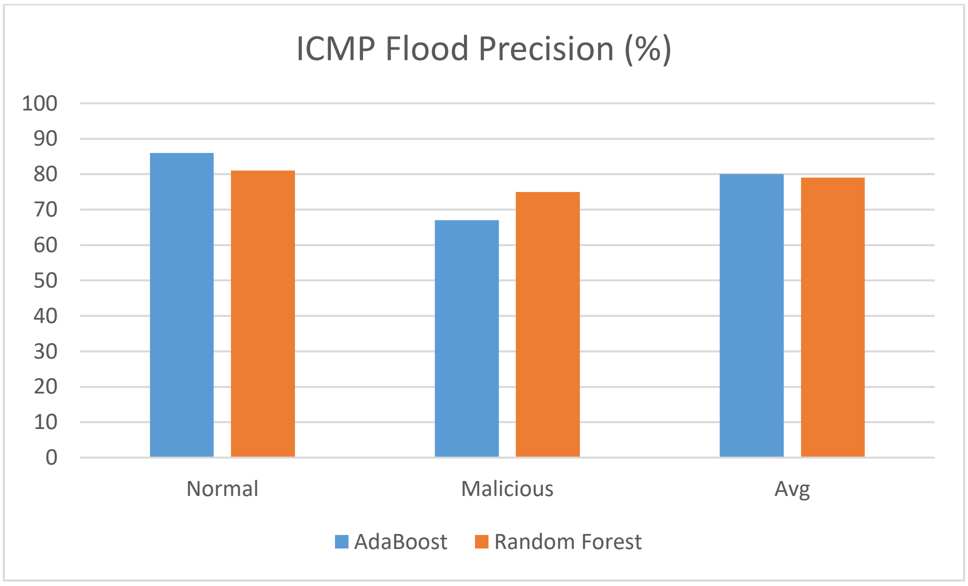


Figure 7.12 AdaBoost and Random Forest precision in ICMP flood detection

When the recall is considered, Figure 7.13, Random Forest also demonstrates superior performance compared to AdaBoost. However, the difference in recall for Random Forest and AdaBoost is minimal compared to the precision value difference of the models.

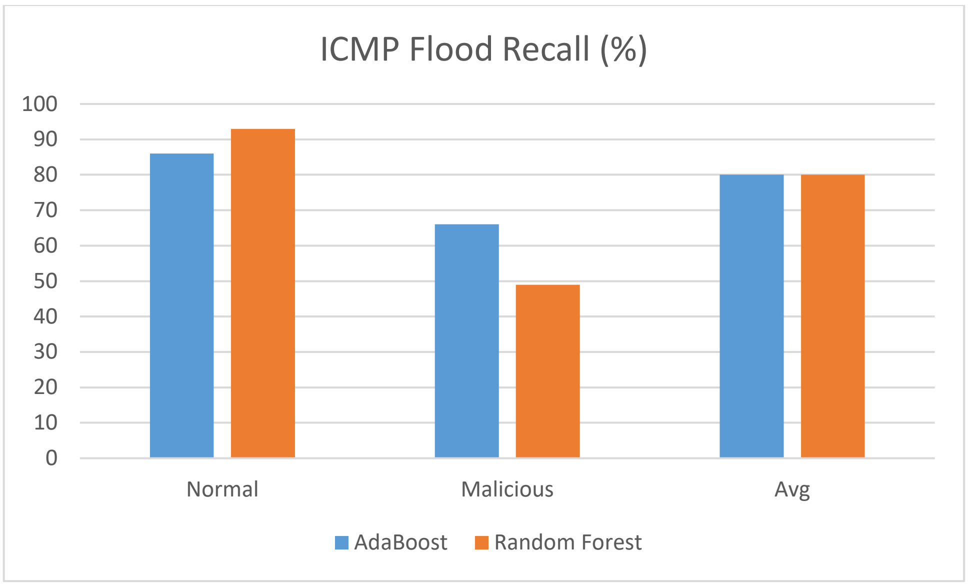


Figure 7.13 AdaBoost and Random Forest recall in ICMP flood detection

The f1-score calculated using both the precision and recall for the models is graphically presented in Figure 7.14.

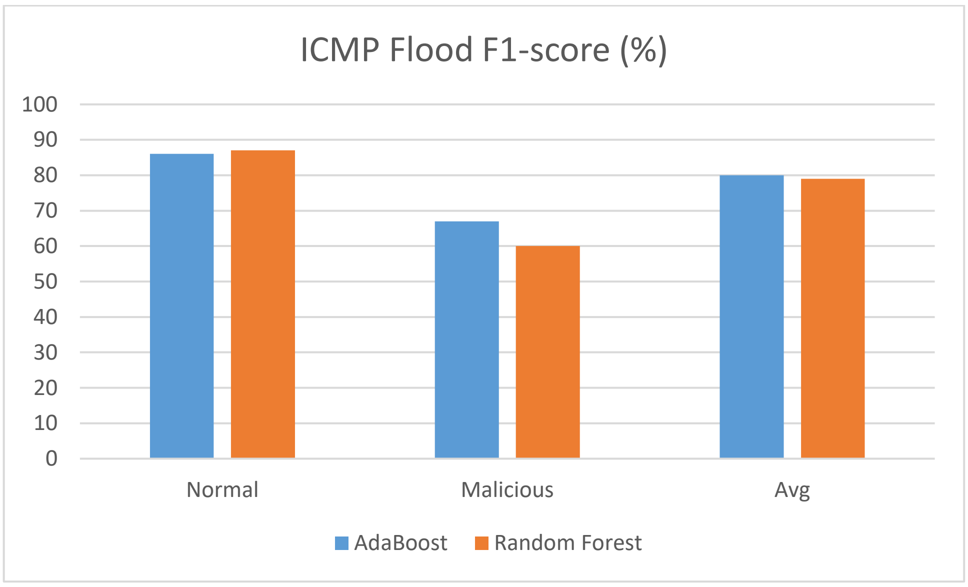


Figure 7.14 AdaBoost and Random Forest f1-score in ICMP flood detection

A model with a high f1-score is defined better-performing since it demonstrates effectiveness in classifying instances into their respective categories correctly. Figure 7.14 shows that Random Forest is more effective than AdaBoost in classifying normal and malicious instances. However, the difference is minimal for all the data points in the set used. The ROC curve for both models in an ICMP flood attack is presented in Figure 7.15.

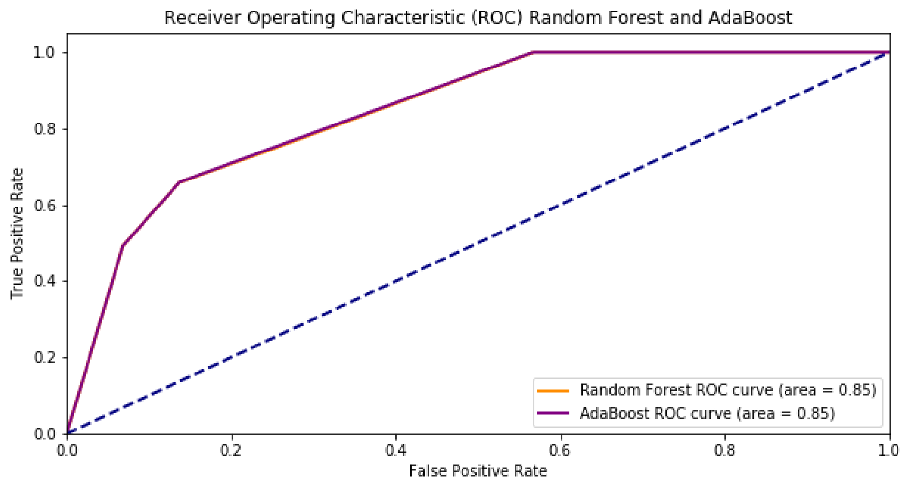


Figure 7.15 ROC curve for AdaBoost and Random Forest in ICMP flood detection

While Random Forest demonstrated superior performance in some cases of precision, recall, and f1-score, plotting the true false-positive rate and true positive rate suggests that both models yield the same level of effectiveness. As in the case of their detection accuracy where they obtain the same accuracy of 80 %, they obtained the same value of 85% AUC as well. Thus, for ICMP flood detection both models achieved the same level of performance, which may be the result of the data containing no outliers and both models were able to generalise very well.

7.3.3. DDoS attack detection

The third test case validates the ensemble models' effectiveness in detecting a DDoS attack issued by two nodes to a single target node in the SDN network. The configuration of the dataset used to build and evaluate the ML models for this scenario is presented in Table 7.9.

Table 7.9 Scenario 3 dataset configuration

Scenario 3	**Normal**	**Malicious**	**Total**
Train	49 659	37 303	**86 962**
Test	24 494	18 339	**42 833**
Total	**95 993**	**33 802**	**129 795**

The results of the ensemble models built using the configuration in Table 7.9 indicated acceptable performance. The AdaBoost model misclassified 5188 normal instances as malicious and 6112 malicious instances as normal, while in Random Forest 4685 normal instances were misclassified as malicious and 4280 malicious instances were classified as normal. The confusion matrix of both models is presented in Figure 7.16 and Figure 7.17, respectively.

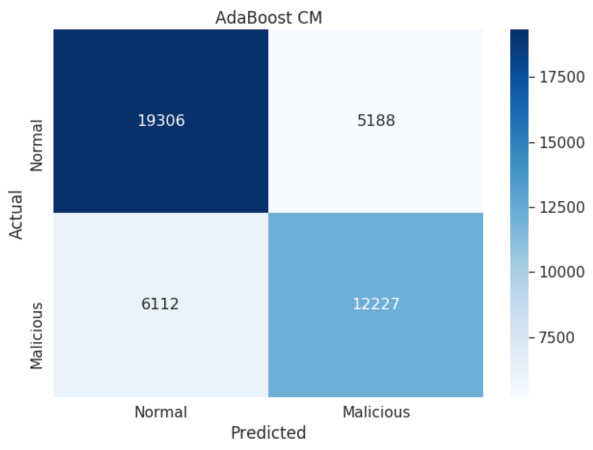


Figure 7.16 AdaBoost Confusion Matrix

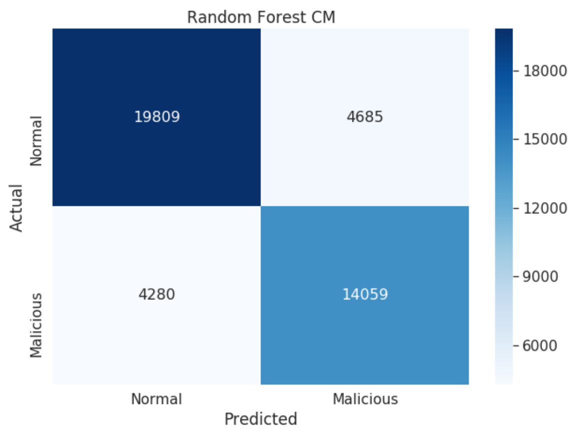


Figure 7.17 Random Forest Confusion Matrix

Both the obtained accuracies are fair considering that they are above 70%. However, Random Forest achieved 79% which indicates a 21% error, while AdaBoost obtains accuracy of 74% with an error of 26%.

From the confusion matrix values, the precision, recall, and F1-score were calculated. The figures are shown in Table 7.10 and Table 7.11. The results indicate that Random Forest can successfully classify flow data instances more correctly than AdaBoost. For example, AdaBoost correctly classifies malicious instances 70 % of the time, while Random Forest can classify malicious instances correctly 75% of the time. The cross-validation score for Random Forest is also higher than AdaBoost.

Table 7.10 AdaBoost result in DDoS attack detection

AdaBoost	Normal	Malicious	Weighted Avg
Precision	76%	70%	73%
Recall	79%	67%	74%
F1-score	77%	68%	74%
Cross Val Score	74.13%		

Table 7.11 Random Forest result in DDoS attack detection

Random Forest	Normal	Malicious	Weighted Avg
Precision	82%	75%	79%
Recall	81%	77%	79%
F1-score	82%	76%	79%
Cross Val Score	79.05%		

Figure 7.18 presents the precision values for normal, malicious, and average for both AdaBoost and Random Forest. The graph clearly shows that Random forest outperforms AdaBoost in classifying instances correctly most of the time since the obtained precision values are higher than AdaBoost.

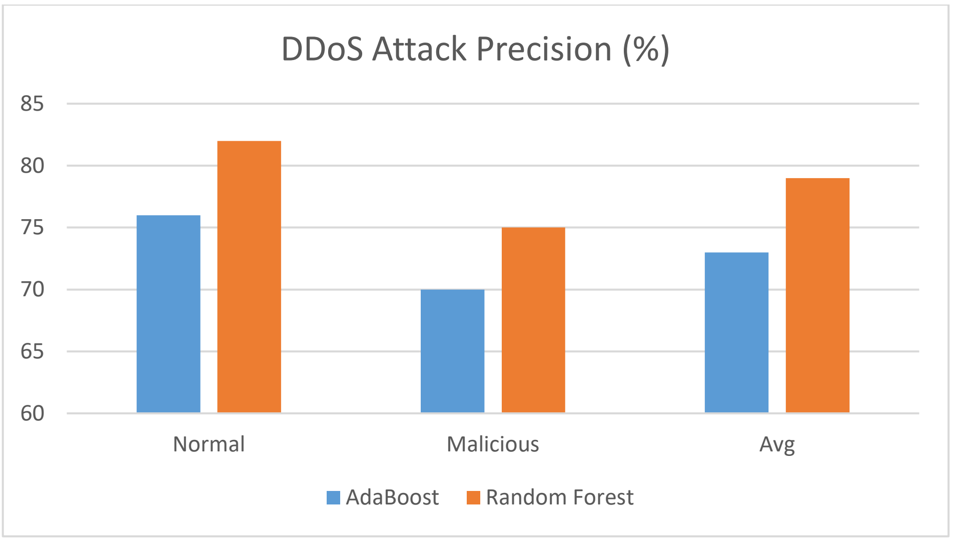


Figure 7.18 AdaBoost and Random Forest precision in DDoS attack detection

Between the AdaBoost and Random Forest models, Random Forest demonstrated high recall rates compared with AdaBoost, shown in Figure 7.19. Random Forest also demonstrated superior performance when the f1-score measure is considered, shown in Figure 7.20, as it is the harmonic mean between the precision and recall of the model.

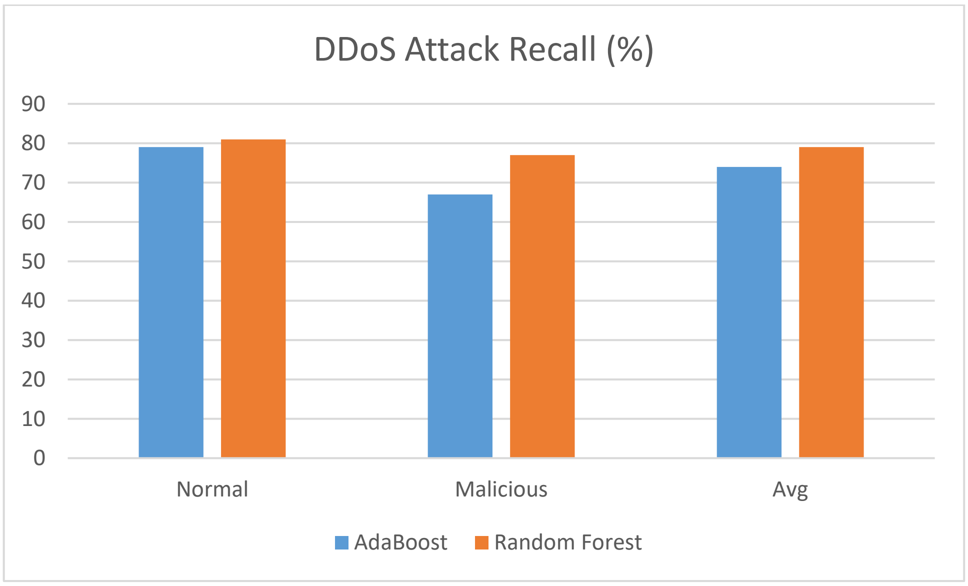


Figure 7.19 AdaBoost and Random Forest recall in DDoS attack detection

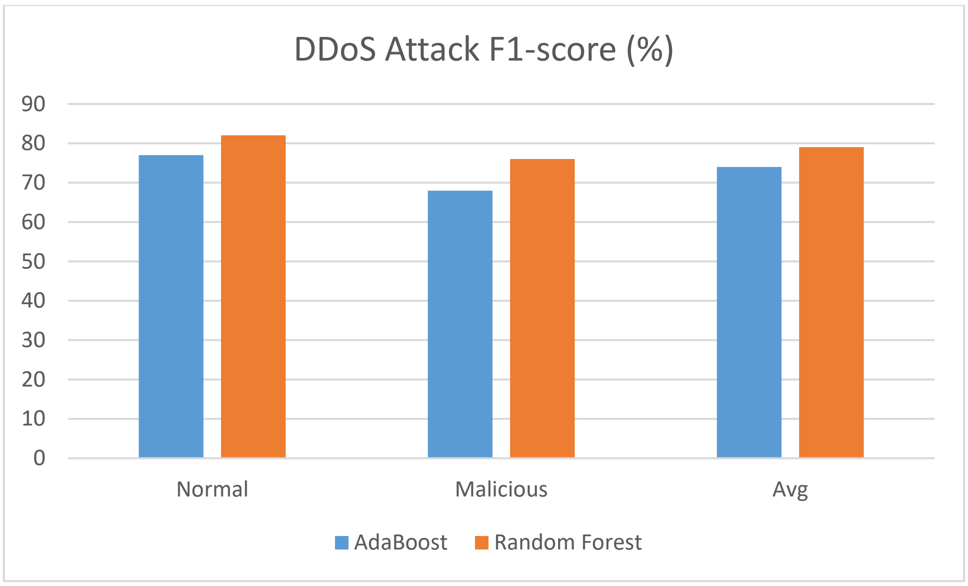


Figure 7.20 AdaBoost and Random Forest f1-score in DDoS attack detection

Another important metric is the AUC obtained from the ROC curve plotted for false-positive rate and true positive rate, as illustrated by Figure 7.21. AdaBoost obtained an AUC of 80% while Random Forest obtained 82%. This indicates that Random Forest outperformed AdaBoost in DDoS attack detection. RF is capable of conducting unsupervised clustering and outlier detection in the training dataset, which is effective in the detection of DDoS attacks since the traffic generated by a DDoS attack contains many different types of packets with different source addresses. This becomes a limitation to AdaBoost as the presence of outliers affects its performance.

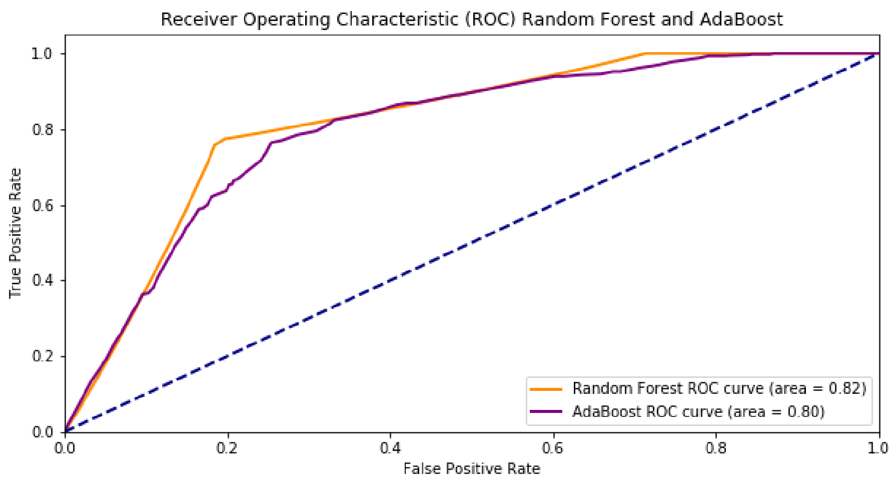


Figure 7.21 AdaBoost and Random Forest ROC AUC for DDoS attack detection

7.4. Discussion and Analysis

As presented in Chapter 4, ensemble learning methods using Decision Tree as their base estimator illustrated high detection performance compared to the other Machine Learning algorithms. This is due to the fact that an ensemble method can learn more than one pattern present in the data which improves detection rates compared to a single learning classifier which can only learn a single pattern from the data.

Among the ensemble learners, RF and AdaBoost were chosen for implementation in the proposed SFIDS. Their implementation was evaluated in three test cases. These test cases evaluate SFIDS's effectiveness in detecting TCP flooding attacks, ICMP flooding attacks, and DDoS attacks.

The first observation from the results is the difference in performance before and after parameter tuning. Parameter tuning plays a critical role in ensuring best-performance in ML algorithms as it describes the optimal architecture and parameters for high detection performances. Nevertheless, some algorithms can yield high performances without any tuning, while others may require tuning before achieving acceptable detection performances. Table 7.12 presents the AUC scores before and after parameter tuning. Before tuning, the RF algorithm was only able to achieve an AUC score of 77% and 87% after tuning, in the first test case. The same behaviour is also applicable to the AdaBoost algorithm. Therefore, after tuning, RF outperformed AdaBoost when detecting TCP flood attacks (test case 1). For ICMP flood detection (test case 2), both classifiers obtained the same level of performance. Finally, Random Forest demonstrated superior performance in detecting DDoS attacks (test case 3) when compared with AdaBoost. Hence, parameter tuning becomes a necessity in ensuring optimal performance from the chosen algorithms.

Table 7.12 AUC score before and after parameter tuning

AUC scores (Before and after tuning)	**AdaBoost** (Zwane, Tarwireyi and Adigun, 2019c)	**AdaBoost (Tuned)**	**Random Forest** (Zwane, Tarwireyi and Adigun, 2019c)	**Random Forest (tuned)**
Test Case 1	77%	**85%**	77%	**87%**
Test Case 2	71%	**85%**	71%	**85%**
Test Case 3	84%	**80%**	59%	**82%**

Conversely, since IDS are designed to be practical defence tools, their performance should ideally be tested in the real world. Yet, research studies that reveal this boundary are rare, while most studies in the research community studying IDS evaluate their IDS tools using only data sets (Haq, Onik and Shah, 2015; Khan *et al.*, 2018; Tang *et al.*, 2018; Khraisat *et al.*, 2019). In this study, we exposed the models that have been trained and evaluated on two data sets to real-world deployment. It was expected that the classification performance obtained from the data set evaluation would hold. However, the experiments revealed that this assumption is not correct. As it can be seen From Table 7.13, the difference in performance can be severe, with up to 15% - 25% difference in accuracy and AUC scores.

Table 7.13 Comparison of ML using data sets and Implementation

Metric	**Algorithm**	**CIDDS-001**	**UNSW-NB15**	**Implementation (SDN)**
Accuracy	AdaBoost	99.15%	90.3%	74.13% - 82.42%
	RF	99.14%	90.2%	79.05% - 84.42%
AUC	AdaBoost	100%	96.8%	80% - 87%
	RF	99.9%	98.1%	82% - 85%

As presented in Table 7.13, SFIDS utilising a Random Forest model obtained 99% precision and recall in the CIDDS-001 datasets, and was only able to obtain an average of 84% in the first deployment scenario, 79% in the second scenario, and 79% in the third scenario. This applies as evidence that better performance in dataset analysis does not imply better performance in production, hence the performance may depreciate. These results were also confirmed by utilising an AdaBoost model in the IDS, as it also obtained 99% in the CIDDS-001 dataset but in deployment, it achieved an average of 82% in the first scenario, 80% in the second scenario, and 74% in the third scenario. Therefore, we recommend that network security researchers may consider a test-bed or an actual implementation for the evaluation of IDS that uses ML techniques in addition to the traditional method of only using datasets.

Chapter 8: Conclusion and Future Work

This chapter aims to summarise and conclude this study. The chapter summaries the research problem introduced in Chapter 1, followed by the research questions and how they were answered in this study. Finally, the recommendations for future work and the limitations are discussed.

8.1. Problem Summary

The nature of the environment in which tactical networks are deployed introduces several issues for intrusion detection security mechanisms. This is because they are deployed in different terrain types, which include: urban areas, forests, hills, and the sea. All these deployment areas introduce different interference characteristics, which cause instabilities in the network, as a result, network packets get corrupted and dropped. Intrusion detection methods see such behaviours as malicious and result in high false detection rates. Hence, this clearly presents the need for an IDS capable of precise detection of hostile nodes in a hostile environment.

With recent innovation in technology, the goal of this study was to investigate and implement the most suitable Machine Learning algorithm using SDN for intrusion detection in tactical Mobile Ad-hoc Networks.

8.2. Research Questions

To address the problem, the author took into consideration one main research question:

How can an intrusion detection system (IDS) that promptly and accurately recognises cyberwarfare attacks in tactical networks be designed and implemented?

From this research question, four sub-research questions arose. The questions and how the research addressed each of them are presented below.

1. **Which ML algorithms are more suitable for intrusion detection in tactical networks?**

To address this sub-question, first, it was necessary to understand network intrusion detection techniques and issues in the tactical network domain. This was achieved through background study and literature review in Chapter 2. Secondly, in Chapter 4 experiments were conducted to evaluate and compare the algorithms using network intrusion detection system evaluation datasets. The experiments used both packet-based and flow-based network datasets to evaluate different Decision Tree-based, probabilistic-based, and non-probabilistic-based machine learning algorithms. From the experiments, it was observed that the Decision Tree-based methods demonstrated outstanding performance in terms of detection rate and classification time when compared to the probabilistic and non-probabilistic Machine Learning algorithms, such as Naïve Bayes and Support Vector Machines. Furthermore, ensemble learning techniques using Decision Tree as the base estimator demonstrated even better detection rates than the normal single learner Decision Tree. Specifically, Random Forest and AdaBoost were the top two to achieve a high detection rate with an acceptable training and classification time when compared to other ensemble methods, such as Bagging, and Majority Voting utilising probabilistic and then non-probabilistic algorithms as their base learner methods. Thus, it was concluded that Random Forest and AdaBoost using Decision Tree as the base estimator are the most suitable machine learning algorithms for intrusion detection in military hostile environment deployment scenarios.

2. How can we design an effective IDS model for tactical networks?

This research question was addressed using the Design Science (DS) methodology. DS was adopted because it offers an important paradigm for conducting applicable and yet rigorous research. Hence the study applied the DS methodology by following the DS research processes introduced by Peffers *et al.*, (2007). To design an IDS model for tactical battlefield networks, technologies, and state-of-the-art paradigms with the potential to help achieve low false detection and recognition totality in tactical battlefield networks were investigated. At first, the better performing ML algorithms were determined using network intrusion datasets, in Chater 4. SDN was then identified as the best solution which can allow efficient data collection and real-time global view of all the network devices connected in the network. Hence, in Chapter 5 the study designed an IDS model that takes advantage of SDN for efficient flow sampling and preparation. The model further uses the better performing ML algorithm to achieve high detection rates and accuracy in military tactical MANETs. In that regard, SDN combined with an efficient ML model can produce an intelligent IDS that is capable of acquiring data from the tactical network devices, processing and analysing such data to identify intrusive behaviours and be able to take countermeasures in real-time without any human intervention.

3. How can the designed IDS model be implemented and operationalised?

The tactical MANETs usually contain many different handhelds; unmanned aerial vehicles (UAVs); intelligence, surveillance, and reconnaissance (ISR) devices; mobile networking; and computing environments to be carried onto various platforms such as tanks, ships, or vehicles. To imitate this environment and demonstrate the operationalisation of the proposed IDS model, a network emulation tool known as Mininet-Wifi was used. Mininet-Wifi allows researchers to implement and evaluate SDN networks. In tactical battlefield networks, flow monitoring and sampling can be conducted in the UAV nodes and other network nodes capable of transmitting packets from one node to another. Hence in our implementation in Chapter 6, SDN forwarding devices are embedded with sFlow sampling techniques, which sample and export network flow data from the network to a logically centralised collector node for data cleaning and pre-processing support intrusion detection tasks. sFlow was adopted for flow collection as it is more efficient and a less resource-intensive alternative to the native OpenFlow sampling approach used in an SDN environment. The logically centralised flow collector node is responsible for preparing the data for analysis using a machine learning classification model train with both malicious and normal network flow data. The tools and simulations presented in Chapter 6 allowed the replication of the environment and the operationalisation of the model designed in Chapter 5. In that regard, we were able to design and implement an IDS model for tactical battlefield networks.

4. **What techniques and metrics can be used to evaluate this IDS system?**

To answer this sub-research question, in Chapter 7 three scenarios were simulated and network flow data was collected. For each scenario, normal and malicious network traffic was generated, such as a TCP flood attack in scenario 1, ICMP flood attack in scenario 2, and a DDoS attack in the scenario 3. Experiments were conducted to evaluate the effectiveness of the proposed IDS model. The aim was to evaluate the effectiveness of the proposed IDS when using either Random Forest and AdaBoost as the classifier to identify malicious nodes, recognise internal attacks, and identify and detect DDoS attacks. However, to determine the effectiveness of an IDS one has to measure the quality of the IDS in being exact and accurate through its predictions. Similarly, the capability of the IDS to remember from its previous experience is also important since it has to achieve a high detection rate and recognition totality. Therefore, considering the tactical network simulated scenarios and the prototype IDS developed, the precision, recall, f-score, and ROC curve metrics were used to evaluate the effectiveness of the proposed IDS model.

8.3. Conclusion

This work presented a flow-based IDS model also regarded as the SFIDS. The study used Design Science as the research method for SFIDS development and evaluation. The work demonstrated the utility and applicability of the SFIDS through a proof-of-concept implementation using SDN, network flows, and Machine Learning techniques for detection. The obtained results show that an intrusion detection system (IDS) utilising SDN for data collection and ML for data analysis achieves pleasing performance in timeously detecting malicious nodes in a hostile environment with high detection rates. However, it is also observed that the performance of an ML model varies based on the data presented to it. That is, an ML model can obtain a high detection rate when evaluated and analysed in a specific dataset but may not obtain the same performance when integrated and deployed in the production environment.

8.4. Recommendation for Future Research

Network security is steadily becoming a critical factor for businesses, organisations, and institutions. While researchers and the academic community are continuously proposing security methods and solutions to ensure security in networks. The recommended solutions are usually evaluated and validated using network datasets generated a long time ago, such as KDD-Cup dataset. Most of the evaluations conducted in such a manner don't present realistic results and can sometimes be misleading. Therefore, researchers in the IDS field should not only recommend a classifier as better-performing in the evaluation datasets but also in the deployment scenarios. It is necessary that researchers evaluate their IDSs using recent datasets and real-life deployment scenarios to obtain more realistic and accurate results.

Also, the work proposed here demonstrated the effectiveness of just a few ensemble algorithms. The work can be extended by evaluating the effectiveness of other Machine Learning algorithms such as deep learning. The creation of military-specific datasets and simulation topologies in which these datasets can be replayed is a critical factor that can improve research done for tactical networks' security.

8.5. Limitations

One of the major drawbacks of this research was the lack of tactical network evaluation datasets that can be used to validate and evaluate intrusion detection systems designed for tactical scenarios. Also, while Mininet-Wifi presents the platform for realising SDN- based research, the simulated tactical network lacks characteristics of a military hostile environment. In addition, due to the storage and simulation environment, a limited number of instances were sampled from the network, which we suspect could negatively affect the performance.

References

Alsmadi, I. M. and AlEroud, A. (2017) ‘SDN-based real-time IDS/IPS alerting system’, in *Studies in Computational Intelligence*, pp. 297–306. doi: 10.1007/978-3-319-44257-0_12.

Amaral, P. *et al.* (2016) ‘Machine learning in software defined networks: Data collection and traffic classification’, *Proceedings - International Conference on Network Protocols, ICNP*, 2016-Decem(NetworkML), pp. 1–5. doi: 10.1109/ICNP.2016.7785327.

Atlam, H. F., Walters, R. J. and Wills, G. B. (2018) ‘Internet of Nano Things’, (October), pp. 71–77. doi: 10.1145/3264560.3264570.

Ayash, M. (2014) ‘Research Methodologies in Computer Science and Information Systems’, *Computer Science*, 2014, pp. 1–4.

Berde, P. *et al.* (2014) ‘ONOS : Towards an Open , Distributed SDN OS’, *HotSDN ’14 Proceedings of the third workshop on Hot topics in software defined networking*, pp. 1–6. doi: 10.1145/2620728.2620744.

Bhunia, S. S. and Gurusamy, M. (2017) ‘Dynamic Attack Detection and Mitigation in IoT using SDN’.

Biondi, P. (2017) ‘Scapy Documentation’. doi: 10.1016/j.physrep.2008.09.003.

Boero, L., Marchese, M. and Zappatore, S. (2017) ‘Support Vector Machine Meets Software Defined Networking in IDS Domain’, in *Proceedings of the 29th International Teletraffic Congress, ITC 2017*. doi: 10.23919/ITC.2017.8065806.

Botelho, F. *et al.* (2014) ‘SMaRtLight: A Practical Fault-Tolerant SDN Controller’, pp. 1–7. Available at: http://arxiv.org/abs/1407.6062.

Boutaba, R. *et al.* (2018) ‘Open Access A comprehensive survey on machine learning for networking : evolution , applications and research opportunities’. Journal of Internet Services and Applications.

Buczak, A. L. and Guven, E. (2016) ‘A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection’, 18(2), pp. 1153–1176.

Burbank, J. L. *et al.* (2006) ‘Key challenges of military tactical networking and the elusive promise of MANET technology’, *IEEE Communications Magazine*. IEEE, 44(11), pp. 39–45. doi: 10.1109/COM-M.2006.248156.

Cai, Z., Cox, A. L. and Ng, T. S. E. (2010) ‘Maestro : A System for Scalable OpenFlow Control - Technical Report TR10-08’.

Carpenter, G. A. *et al.* (1992) ‘Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps’, *IEEE Transactions on Neural Networks*, 3(5), pp. 698–713. doi: 10.1109/72.159059.

Carvalho, J. M. A. and Costa, P. C. G. (2016) ‘Collaborative Approach for a MANET Intrusion Detection System using Multilateration’, *2016 11th International Conference on Computer Engineering & Systems (ICCES)*. IEEE, pp. 59–65. doi: 10.1109/ICCES.2016.7821976.

Chang, R. J. *et al.* (2013) ‘Extremely Lightweight Intrusion Detection (ELIDe)’, (December).

Chen, X. *et al.* (2018) ‘Ensemble Learning Methods for Power System Cyber-Attack Detection’, *2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. IEEE, pp. 613–616. doi: 10.1109/ICCCBDA.2018.8386588.

Choudhary, G. *et al.* (2018) ‘Intrusion Detection Systems for Networked Unmanned Aerial Vehicles : A Survey’.

Demeyer, S. (2011) ‘Research methods in computer science’, *IEEE International Conference on Software Maintenance, ICSM*. IEEE, p. 600. doi: 10.1109/ICSM.2011.6080841.

Division, C. S. *et al.* (1997) ‘Bayesian Network Classifiers *’, 163, pp. 131–163.

Djuris, J. (2012) ‘Design Space Approach in Optimization of Fluid Bed Granulation and Tablets The cientific WorldJOURNAL Research Article Design Space Approach in Optimization of Fluid Bed’, (May 2014). doi: 10.1100/2012/185085.

Dunning, T. and Friedman, E. (2014) *Practical Machine Learning: A New Look At Anomaly Detection*.

Erickson, D. (2013) ‘The beacon openflow controller’, p. 13. doi: 10.1145/2491185.2491189.

Ertam, F., Õ, F. Õ. K. and Yaman, O. (2017) ‘Intrusion Detection in Computer Networks via Machine Learning Algorithms’.

Fahad, M., Sher, M. and Bi, Y. (2017) ‘Flow-based intrusion detection : Techniques and challenges’, *Computers & Security*. Elsevier Ltd, 70, pp. 238–254. doi:

10.1016/j.cose.2017.05.009.

Fontes, R. R. *et al.* (2015) ‘Mininet-WiFi : Emulating Software-Defined Wireless Networks’.

Giotis, K. *et al.* (2014) ‘Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments’, (February 2016). doi: 10.1016/j.bjp.2013.10.014.

Gogoi, P., Bhuyan, M. H. and Bhattacharyya, D. K. (2012) *Packet and Flow Based Network Intrusion Dataset*.

Govindarajan, M. and Chandrasekaran, R. (2012) ‘Intrusion Detection using an Ensemble of Classification Methods’, *Proceedings of the World Congress on Engineering and Computer Science*, I(October).

Haq, N. F., Onik, A. R. and Shah, F. M. (2015) ‘An Ensemble Framework of Anomaly Detection using Hybridized Feature Selection Approach (HFSA)’, *2015 SAI Intelligent Systems Conference (IntelliSys)*. IEEE, pp. 989–995. doi: 10.1109/IntelliSys.2015.7361264.

Hebb, D. O. (1949) *The Organization of Behavior*.

Hevner, A. R. *et al.* (2004) ‘Research Essay Design Science in Information’, *MIS Quarterly*, 28(1), pp. 75–105.

Hofstede, R. *et al.* (2014) ‘Flow Monitoring Explained : From Packet Capture to Data Analysis With NetFlow and IPFIX’, 16(4), pp. 2037–2064.

Illy, P. *et al.* (2019) ‘Securing Fog-to-Things Environment Using Intrusion Detection System Based On Ensemble Learning’, (April), pp. 15–18.

Jabbar, M. A. *et al.* (2017) ‘ScienceDirect ScienceDirect RFAODE : A Novel Ensemble Intrusion Detection System’, *Procedia Computer Science*. Elsevier B.V., 115, pp. 226–234. doi: 10.1016/j.procs.2017.09.129.

Jeung, J., Jeong, S. and Lim, J. (2011) ‘Adaptive Rapid Channel-hopping Scheme Mitigating Smart Jammer Attacks in Secure WLAN’, pp. 1231–1236.

Ji Qing *et al.* (2015) ‘An SDN-based resource pre-combination dispatching strategy in military network’, in, pp. 6 .-6 . doi: 10.1049/cp.2015.0834.

Karresand, M. (2004) ‘Intrusion Analysis in Military Networks – An Introduction Technical report Intrusion Analysis in Military Networks – An Introduction’, (December).

Khan, S. *et al.* (2018) ‘Feature Selection of Denial-of-Service Attacks Using Entropy and Granular Computing’, *Arabian Journal for Science and Engineering*. Springer Berlin Heidelberg, 43(2), pp. 499–508. doi: 10.1007/s13369-017-2634-8.

Khraisat, A. *et al.* (2019) ‘Survey of intrusion detection systems: techniques, datasets and challenges’, *Cybersecurity*. Cybersecurity, 2(1). doi: 10.1186/s42400-019-0038-7.

Kidston, D. *et al.* (2010) ‘Mitigating security threats in tactical networks’, *... Communication and Networks, ...*, pp. 1–14. Available at: http://ftp.rta.nato.int/public/PubFullText/RTO/MP/RTO-MP-IST-092/MP-IST-092-20.doc.

Koponen, T. *et al.* (2010) ‘【8】Onix A Distributed Control Platform for Large.pdf’.

Kurdziel, M. T. (2014) ‘Cyber threat model for tactical radio networks’, *Wireless Sensing, Localization, and Processing IX*, 9103(June), p. 910305. doi: 10.1117/12.2047582.

Liu, T. *et al.* (2018) ‘Intrusion Detection of Data Platform Based on Extreme Learning Machine in Civil and Military Integration’, (Csse), pp. 296–306.

Madhu, A. and Sreekumar, A. (2014) ‘Wireless Sensor Network Security in Military Application using Unmanned Vehicle’, *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, pp. 8–51.

Marcus, K. M. *et al.* (2019) ‘An Environment for Tactical SDN Experimentation’, *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*. IEEE, pp. 1–9. doi: 10.1109/milcom.2018.8599775.

McCulloch, W. S. and Pitts, W. H. (1943) ‘originally published in: Bulletin of Mathematical Biophysics, Vol. 5, 1943, p. 115-133’, 5, pp. 115–133.

Metcalf, T. R. and Lapadula, L. J. (2000) ‘Intrusion Detection System Requirements A Capabilities Description in Terms of the Network Monitoring and Assessment Module of’.

Michalos, A. C. and Simon, H. A. (1970) *The Sciences of the Artificial, Technology and Culture*. doi: 10.2307/3102825.

Monshizadeh, M., Khatri, V. and Kantola, R. (2017) ‘An adaptive detection and prevention architecture for unsafe traffic in SDN enabled mobile networks’, in *Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management*. doi: 10.23919/INM.2017.7987395.

Moustafa, N., Slay, J. and Technology, I. (2015) ‘Intrusion Detection systems’.

Nguyen, T. N. (2018) ‘The Challenges in SDN/ML Based Network Security : A Survey’. Available at: http://arxiv.org/abs/1804.03539.

Pawgasame, W. and Wipusitwarakun, K. (2015) ‘Tactical wireless networks: A survey for issues and challenges’, *2015 Asian Conference on Defence Technology (ACDT)*, pp. 97–102. doi: 10.1109/ACDT.2015.7111592.

Pedregosa, F., Weiss, R. and Brucher, M. (2011) ‘Scikit-learn : Machine Learning in Python’, 12, pp. 2825–2830.

Peffers, K. *et al.* (2007) ‘A Design Science Research Methodology for Information Systems Research’, *Journal of Management Information Systems*, 24(3), pp. 45–77. doi: 10.2753/MIS0742-1222240302.

Poularakis, K., Iosifidis, G. and Tassiulas, L. (2018) ‘SDN-enabled Tactical Ad Hoc Networks : Extending Programmable Control to the Edge’.

Pushpa, M. and Kathiravan, A. (2016) ‘Cross-layer based multiclass intrusion detection system for secure multicast communication of MANET in military networks’, *Wireless Networks*. Springer US, 22(3), pp. 1035–1059. doi: 10.1007/s11276-015-1065-2.

Revathi, S. and Malathi, A. (2013) ‘A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection’, 2(12), pp. 1848–1853.

Rhodes, B. J. *et al.* (2005) ‘Maritime situation monitoring and awareness using learning mechanisms’, *Proceedings - IEEE Military Communications Conference MILCOM*. IEEE, 2005, pp. 646-652 Vol. 1. doi: 10.1109/MILCOM.2005.1605756.

Ring, M *et al.* (2017) ‘Flow-based benchmark data sets for intrusion detection’, *European Conference on Information Warfare and Security, ECCWS*, pp. 361–369. Available at: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85028023805&partnerID=40&md5=1e95f767994dde4a33199aa24418b078.

Ring, Markus *et al.* (2017) ‘Technical Report CIDDS-001 data set’, 001, pp. 1–13.

Shin, S. *et al.* (2014) ‘Rosemary : A Robust , Secure , and High-Performance Network Operating System Categories and Subject Descriptors’, *Ccs*, pp. 78–89.

Spencer, J. *et al.* (2016) ‘Towards a tactical software defined network’, *2016 International*

Conference on Military Communications and Information Systems (ICMCIS), pp. 1–7. doi: 10.1109/ICMCIS.2016.7496552.

Sterbenz, J. P. G. *et al.* (2002) ‘Survivable Mobile Wireless Networks : Issues , Challenges , and Research Directions’.

Stoyanova Todorova, M. and Todorova, S. T. (2016) *DDoS Attack Detection in SDN-based VANET Architectures*.

Sultana, N. *et al.* (2018) ‘Survey on SDN based network intrusion detection system using machine learning approaches Survey on SDN based network intrusion detection system using machine learning approaches’. Peer-to-Peer Networking and Applications, (January). doi: 10.1007/s12083-017-0630-0.

Sultana, N. *et al.* (2019) ‘Survey on SDN based network intrusion detection system using machine learning approaches’, *Peer-to-Peer Networking and Applications*. Peer-to-Peer Networking and Applications, 12(2), pp. 493–501. doi: 10.1007/s12083-017-0630-0.

Svenmarck, P. *et al.* (no date) ‘Possibilities and Challenges for Artificial Intelligence in Military Applications’, pp. 1–16.

Tang, T. A. *et al.* (2018) ‘Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks’, *2018 4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018*. IEEE, (NetSoft), pp. 462–469. doi: 10.1109/NETSOFT.2018.8460090.

Tootoonchian, A. (2010) ‘Hyperflow.Pdf’.

Tootoonchian, A. *et al.* (2012) ‘On Controller Performance in Software-Defined Networks’, *Presented as part of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, p. 55. doi: 10.1145/2491185.2491199.

Verma, A. and Ranga, V. (2018) ‘Statistical analysis of CIDDS-001 dataset for Network Intrusion Detection Systems using Distance-based Machine Learning’, *Procedia Computer Science*. Elsevier B.V., 125, pp. 709–716. doi: 10.1016/j.procs.2017.12.091.

Viinikka, J. *et al.* (2009) ‘Processing intrusion detection alert aggregates with time series modeling’, *Information Fusion*. Elsevier B.V., 10(4), pp. 312–324. doi: 10.1016/j.inffus.2009.01.003.

Vijayanand, R., Devaraj, D. and Kannapiran, B. (2018) ‘Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature', *Computers & Security*. Elsevier Ltd, 77, pp. 304–314. doi: 10.1016/j.cose.2018.04.010.

Visible, N. and Packard, H.- (2003) 'Switch / Router'.

Wilson, C. (2004) 'CRS Report for Congress Received through the CRS Web Network Centric Warfare : Background and'.

Wrona, K. and Szwaczyk, S. (2017) 'SDN testbed for validation of cross-layer data-centric security policies', pp. 1–6. doi: 10.1109/ICMCIS.2017.7956483.

Yan, Q. *et al.* (2016) 'Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments : A Survey , Some Research Issues , and Challenges', 18(1), pp. 602–622.

Ye, N. *et al.* (2002) 'Multivariate statistical analysis of audit trails for host-based intrusion detection', *IEEE Transactions on Computers*, 51(7), pp. 810–820. doi: 10.1109/TC.2002.1017701.

Yoon, C. *et al.* (2015) 'Enabling security functions with SDN: A feasibility study', *Computer Networks*, 85, pp. 19–35. doi: 10.1016/j.comnet.2015.05.005.

Yuill, J. *et al.* (2000) 'Intrusion-detection for incident-response , using a military battle ® eld-intelligence process', 34, pp. 671–697.

Zaidi, N. A. *et al.* (2017) 'Efficient parameter learning of Bayesian network classifiers', *Machine Learning*. Springer US, 106(9–10), pp. 1289–1329. doi: 10.1007/s10994-016-5619-z.

Zaman, M. (2018) 'Evaluation of Machine Learning Techniques for Network Intrusion Detection', *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, pp. 1–5.

Zumel, N. and Mount, J. (2014) *Practical Data Science with R*. Manning Publications.

Zwane, S., Tarwireyi, P. and Adigun, M. (2019a) 'A Flow-based IDS for SDN-enabled Tactical Networks', *International Multidisciplinary Information Technology and Engineering Conference (IMITEC).*

Zwane, S., Tarwireyi, P. and Adigun, M. (2019b) 'Ensemble learning approach for Flow based Intrusion Detection System', *IEEE AFRICON*, pp. 0–7.

Zwane, S., Tarwireyi, P. and Adigun, M. (2019c) 'Ensemble learning for Flow based IDS : A SDN Implementation', *Southern Africa Telecommunication Networks and Applications Conference (SATNAC).*

Zwane, S., Tarwireyi, P. and Adigun, M. (2019d) 'Performance Analysis of Machine Learning Classifiers for Intrusion Detection', *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*. IEEE, pp. 1–5. doi: 10.1109/iconic.2018.8601203.