University of Sheffield

The Detection of Advanced Persistent Threats in Software Defined Networks using Machine Learning

Abdullah Hamad Alqahtani

Supervisor:

Prof. John A Clark

A thesis submitted
for the degree of *Doctor of Philosophy*

in the

Department of Computer Science
The University of Sheffield

August 6, 2023

Declaration of Authorship

All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Signed:

Date:

Abstract

A Software-Defined Network (SDN) is a new type of network architecture that separates the control and network planes. The centralised controller can programmatically manage the underlying network devices. Although SDN provides many advantages, it raises new security challenges. A stealth attack is a particularly dangerous kind of attack adopted by adversaries who aim to avoid detection, typically by incurring lower levels of traffic during their activities than would arouse suspicion. Advanced Persistent Threats (APTs) are sophisticated attacks that implement stealth behaviour during their campaigns. They present major challenges to the security of systems. Little research has been carried out on detecting APTs in the context of SDNs. This is the focus of this thesis.

Initially, an enhancement of scanning capabilities in SDN is introduced and an open source scanner tool is adapted to operate more stealthily (allowing extended periods of time between operations it carries out). It has been made publicly and freely available to researchers. In this thesis, it is used to generate datasets (using Mininet) to train and evaluate detection models. Existing datasets do not adequately represent the presence of APTs, or do not do so in the context of SDNs. Thus, generating our own datasets was essential for the work in this thesis. However, we still make use of existing datasets in our evaluations, e.g. to show our approaches may still work effectively against non-APT threats. Of particular interest in this thesis is the use of stealth techniques as part of 'flow rule reconstruction' attacks, where attackers seek to infer aspects of packet handling policies that apply at targeted nodes. Inferring such information facilitates further attacks.

The most common Machine Learning (ML) techniques for signature-based detection (such as Decision Tree, K-Nearest Neighbour, Random Forest, XGBoost and Support Vector Machine) and for anomaly-based detection (such as Local Outlier Factor, Isolation Forest and One-class SVM) are evaluated. Consequently, XGBoost is proposed as a signature-based model to detect known stealth attacks in SDN and is shown to be highly effective.

Subsequently, a hybrid detection model is constructed by combining XGBoost (as a signature-based detection module) and a One-class SVM (as an anomaly-based detection module) leveraging the complementary aspects of these techniques to allow known and unknown attacks to be detected. This is the first demonstration of the effectiveness of a hybrid approach for APT detection in SDNs.

As systems evolve, the effectiveness of an ML-based classifier degrades because the distribution of the data it needs to handle increasingly deviates from that over which it was trained. This is known as *concept drift*. One cause of such drift is attackers changing their behaviour. A hybrid system (signature-based detection using an Adaptive Random Forest and anomaly-based detection using an Adaptive One-Class SVM) is presented that uses concept drift detection to instigate appropriate run-time model retraining. The approach can detect known and unknown attacks and adapt itself incrementally when concept drift happens. This is the first time concept drift has been considered in the context of intrusion detection for SDNs.

The validity of our IDS schemes is assessed using various datasets with different attacks and network sizes. ML-pipeline techniques commonly ignored in the IDS literature are employed as part of the work: hyperparameter tuning to generalised the model, imbalanced datasets are subject to resampling to prevent bias in predictions and feature reduction is employed to focus modelling on smaller numbers of highly informative features. Our proposed models are compared with available benchmark results in the field and also with competing approaches as part of our comprehensive empirical evaluation. Performance metrics such as Accuracy, Recall, Precision, and F1-score are used in the evaluation. These steps collectively ensure that our schemes are robust, accurate, and capable of generalising to new attack scenarios.

Overall, we show how machine learning can effectively detect APT stealth attacks under constant contextual conditions and under change. We address the detection of both known and unseen attacks. This is the first thesis to comprehensively address the effective detection of APTs in an SDN context and demonstrates that machine learning has a critical part to play in addressing the challenges APTs pose to SDNs.

Acknowledgements

I would like to express my deepest appreciation to my supervisor Professor John A. Clark, who supported me on the whole PhD journey. He provided me with knowledge, expertise, advice and guidance. I need to thank my panel members, Professor Hamish Cunningham and Dr. Prosanta Gope for their comments and feedback. I would like to express my gratitude to my viva examiners, Dr Andrei Popescu and Dr Vassilios G. Vassilakis, for their invaluable comments and suggestions in enhancing this thesis.

I am eternally grateful to all my family members for their support and wishes. I would like to thank all Security of Advanced Systems research group (at the University of Sheffield) for their help and being amazing friends and colleagues.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ANN **A**rtificial **N**eural **N**etwork

API **A**pplication **P**rogramming **I**nterface

APT **A**dvanced **P**ersistent **T**hreat

ARF **A**daptive **R**andom **F**orest

AUC **A**rea **U**nder the **C**urve

C&C **C**ommand and **C**ontrol

DNN **D**eep **N**eural **N**etwork

DoS **D**enial **o**f **S**ervice

DT **D**ecision **T**ree

FNR **F**alse **N**egative **R**ate

FN **F**alse **N**egative

FPR **F**alse **P**ositive **R**ate

FP **F**alse **P**ositive

HIDS **H**ost-based **I**ntrusion **D**etection **S**ystems

ICS **I**ndustrial **C**ontrol **S**ystems

IDS **I**ntrusion **D**etection **S**ystem

IF **I**solation **F**orest

IPS **I**ntrusion **P**revention **S**ystem

KNN **K** **N**earest **N**eighbour

LOF **L**cal **O**utlier **F**actor

LSTM **L**ong **S**hort **T**erm **M**emory

ML **M**achine **L**earning

MTD **M**oving **T**arget **D**efense

NBI **N**orthbound **I**nterface

NIDS **N**etwork-based **I**ntrusion **D**etection **S**ystems

OC-SVM **O**ne **C**lass **S**upport **V**ector **M**achine

P4 **P**rogramming **P**rotocol **I**ndependent **P**rocessors

RQ **R**esearch **Q**uestion

SBI **S**outhbound **I**nterface

SDN **S**oftware **D**efined **N**etwork

SMOTE **S**ynthetic **M**inority **O**versampling **T**echnique

SVM **S**upport **V**ector **M**achine

TN **T**rue **N**egative

TP **T**rue **P**ositive

XGBoost **e**Xtreme **G**radient **B**oosting

Chapter 1

Introduction

1.1 Motivation and Problem Statement

Software-Defined Networking (SDN) is being used in a large number of organisations due to the benefits the architecture offers in usability and efficiency. However, some security challenges are raised due to the decoupling of the network and control planes. A fundamental aspect of the approach is the maintenance of 'flow rule tables' in switches around the network. These rules dictate how a switch handles the packets that it receives. This is a critical aspect of internal network configuration.

Knowledge of what rules apply at switch nodes can be of considerable use to attackers. Consequently, preventing the malicious discovery of such rules in SDN switches is one of the main concerns of the SDN community and a vibrant research area [8]. When the attacker succeeds in determining the flow rules, some attacks can be launched, including some that leak data or poison the network. Gaining this traffic management data is usually referred to as 'flow rule reconstruction'. In this thesis, we will be concerned initially with such reconstruction attacks, particularly using advanced stealthy approaches, and how machine learning (ML) may be used to detect them. We will subsequently seek to widen the scope of attacks detected, targeting further stealth attacks and more traditional attacks.

Advanced Persistent Threats (APTs) are complex and sophisticated attacks. They are used not only for cybercriminal purposes but also for cyber warfare. Adversaries behind these types of attacks are well-funded, highly skilled and target-specific victims. The characteristics and sophistication of APTs make them far more dangerous than traditional threats and more difficult to detect. They are characterised by their

persistence, stealth and the level of resources and expertise employed by the attackers. They employ slow-and-low strategies and mimic normal user behaviour to remain under any thresholds of detection systems and to otherwise avoid suspicion [2]. This thesis focuses on APTs that target SDN networks. They behave extremely stealthily by keeping activity levels low and stretching attacks over extended periods (i.e., they employ a 'low and slow' strategy). Hence, the term 'stealth' in this thesis refers to the extreme stealth behaviour employed by APTs.

Monitoring and detecting stealth attacks in SDN is important. The concept of a Network Intrusion Detection System (NIDS) has been investigated to assist this endeavour. A NIDS is a software system or device that monitors network traffic to detect malicious activities. Intrusion detection approaches, in general, are categorised into two main types based on the detection approach: *signature-based* detection, where the system seeks to match current behaviour or data against predefined patterns (signatures) of attacks; and *anomaly-based* detection, which profiles normal behaviour and then highlights unusual behaviour as suspicious. Adopting machine learning within IDSs is now common. The capabilities of signature-based and anomaly-based approaches differ: the former is a highly appropriate tool for detecting *known* attacks (i.e. where a characteristic pattern of some form of the attack is available), whereas anomaly-based detection has better chances of detecting previously *unseen* attacks. Attacker behaviour will also change over time. Intrusion detection systems must adapt to such changes to maintain detection performance. This too raises challenges regarding how change is detected and how the intrusion detection system should respond.

A small number of researches have been proposed to detect APT attacks. The majority of these studies analyse a particular APT attack. Others evaluate their proposals on a generic dataset with no APT attacks or do not conduct evaluations on an SDN network. In general, there is a lack of literature proposing to detect APTs in an SDN environment.

This thesis aims to build a Network Intrusion Detection System to detect stealth attacks in an SDN environment. We seek excellent detection performance and a capability to evolve in the face of a changing adversary or network so that excellent

performance is maintained. Furthermore, we aim to detect known and unseen attacks. Our approach will also detect traditional attacks.

1.2 Research Questions

This thesis investigates the following three research questions:

1. **Research Question 1 (RQ1):** Can we use a machine-learning and signature-based approach to detect the stealthy reconstruction of flow rules in SDN networks?

2. **Research Question 2 (RQ2):** Can we use a machine-learning and hybrid (signature-based plus anomaly-based) approach to detect the stealthy reconstruction of flow rules in SDN networks?

3. **Research Question 3 (RQ3):** Can we continue to detect stealth attacks in SDN networks as adversaries change their behaviour?

1.3 Contributions

The contributions of this thesis are:

1. Proposal of a mechanism for making scans in SDN more stealthy.

2. Development of a customised scanning tool for stealthy reconstruction of SDN flow rules.

3. Production of a dataset including activities geared to the reconstruction of flow rules in SDN using some APT behavioural characteristics in the attack. This dataset has been made publicly available [1].

4. Investigation of the most popular machine learning techniques in the context of the two main detection approaches (supervised learning and unsupervised learning).

5. Proposal for a signature-based Network Intrusion Detection System (NIDS) to detect stealthy probing attacks in SDN networks.

[1] https://github.com/APT-SDNdataset

6. Proposal of a hybrid (signature-based and anomaly-based detection) NIDS that detects stealthy (known and unknown) SDN attacks.

7. Proposal of an incremental adaptive NIDS that adapts to maintain excellent detection performance in the face of behavioural change.

8. Incorporation of a richer ML pipeline than what is commonly seen in IDS work. Specifically, we demonstrate more sophisticated feature engineering and hyper-parameter tuning of all ML techniques used.

9. Evaluating the proposed models over different datasets.

1.4 Organisation of the Thesis

The rest of the thesis is presented as follows. The next chapter, Chapter 2, is a literature survey that presents background on the field of study and a review of the most related research. Chapter 3 presents the work on enhancing the stealth of flow rule reconstruction scans in SDN networks and shows how a signature-based supervised machine learning approach can detect them. Chapter 4 presents a hybrid NIDS to detect such scans in SDN networks. Chapter 5 shows our work on the incremental adaptive hybrid NIDS to detect APTs in SDN and overcome behaviour changes. Chapter 6 concludes the thesis and identifies future work.

1.5 Publications

The work presented in this thesis has been published as shown below:

1. **Enhanced Scanning in SDN Networks and its Detection using Machine Learning**. Abdullah H. Alqahtani and John A. Clark. The Fourth IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (2022).

2. **Detecting Stealthy Scans in SDN using a Hybrid Intrusion Detection System**. Abdullah H. Alqahtani and John A. Clark. WRIT – Workshop on Research for Insider Threats (2022)

Chapter 2

Literature Survey

2.1 Introduction

Computer networks are sets of devices connected to each other that communicate through pre-defined protocols. Traditional networks struggle to meet today's requirements of carriers, enterprises, and users. It is hard to manage and configure high-level policies on a large number of physical network devices. Network devices such as switches and routers are fixed and dedicated to managing network traffic based on predefined configurations and the complexity of legacy networks makes it difficult to reconfigure and respond to network events and changes.

The Software Defined Network (SDN) paradigm is a highly flexible means of providing distributed infrastructure and facilitating the means to manage it effectively and efficiently. It does so by separating various concerns, in particular separating the control plane of the architecture from the data plane (allowing management and control to be separated from routing). SDN has been adopted by a significant number of major companies such as Intel, Huawei, Cisco, Juniper, IBM, Brocade and Dell [9]. The SDN market is expected to reach US$ 37.24 billion by 2030 [10]. It is clearly of major significance.

The architecture, however, brings with it various security concerns. This thesis is concerned with one specific type of attack that could have SDN as a target. Advanced Persistent Threats (APTs) are stealthy and complex attacks that adopt strategies making them hard to detect. Reconstructing flow rules in SDN and other attacks will be considered. Below we review the literature concerned with SDN, threats to its security, and means of detecting malicious software and behaviours directed against

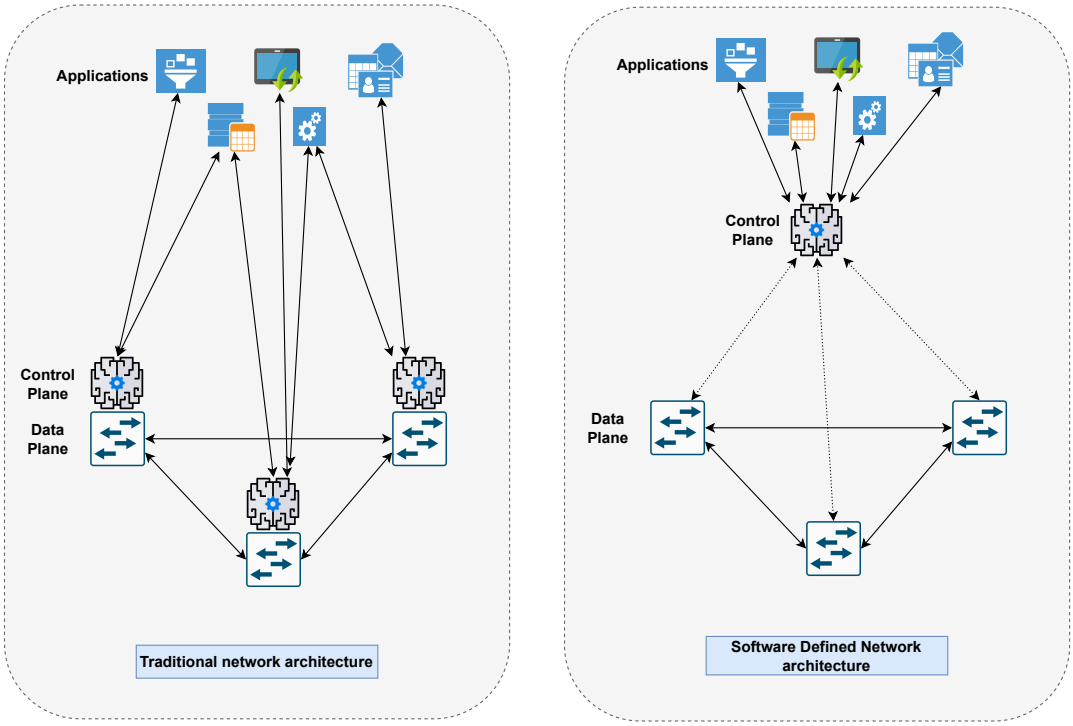

FIGURE 2.1: SDN vs Traditional Network

it.

2.2 Software-Defined Network

The Software-Defined Network (SDN) separates the control plane (SDN controller) from the data plane (SDN network devices such as switches and routers). In traditional networks, forwarding decisions happen in network devices, whereas in an SDN all forwarding decisions are defined in a centralised controller. The centralisation of intelligence and state of the network in the SDN controller provides more flexibility to the network [11]. To configure network hardware in the conventional network, a network operator must typically log in to every device in the network and configure it via the command line. Moreover, some vendors have their specific commands [12]. In the SDN network, centralising the controller and separating it from the network devices improved the network flexibility. Instead of static manual configuration, the SDN network configuration can now be more dynamic and programmable. The network operator can implement commands on any number of devices at the same time, regardless of their vendors, by leveraging the programmability offered by SDN. Through the controller, the operator can program switches by

sending commands through "southbound APIs". Figure 2.1 shows the difference in the location of the network brain in SDN and traditional networks. SDN has many advantages including:

1. The centralisation of the controller provides a global overview of the whole network. It can bring significant benefits to network security and performance.

2. Dynamic configuration of network infrastructures provides more flexibility to manage their features and functionalities.

3. The capability of the controller to update and modify network traffic as a response to changes.

2.2.1 SDN architecture

SDN has three planes (layers) and uses Application Programming Interfaces (APIs), or protocols, in the communications between these planes. The network/Infrastructure layer (data plane), Control layer (Control plane), and Application layer (Application plane) are the main three layers in an SDN network [13]. However, some literature [14; 11; 15] adds a fourth layer - a management plane - where the SDN manager performs the management functions. Figure 2.2 illustrates the main three layers of SDN. The Northbound Interface is the interface between the control plane and the application plane. The interface between the control plane and the network plane is called Southbound Interface.

1. Control plane: This is considered as the brain of the network, containing the network controller. The controller is centralised and moved to be an external entity running as software in the cloud or on a physical server. It has global visibility of the entire network and maintains the flow rules according to the policies derived from the services within the application plane. It provides the network path for data flow in SDN [16]. SDN networks can have multiple controllers logically centralised. The controller can monitor the network flows and request the network devices' status anytime. In addition, it manages the SDN applications in the application plane.

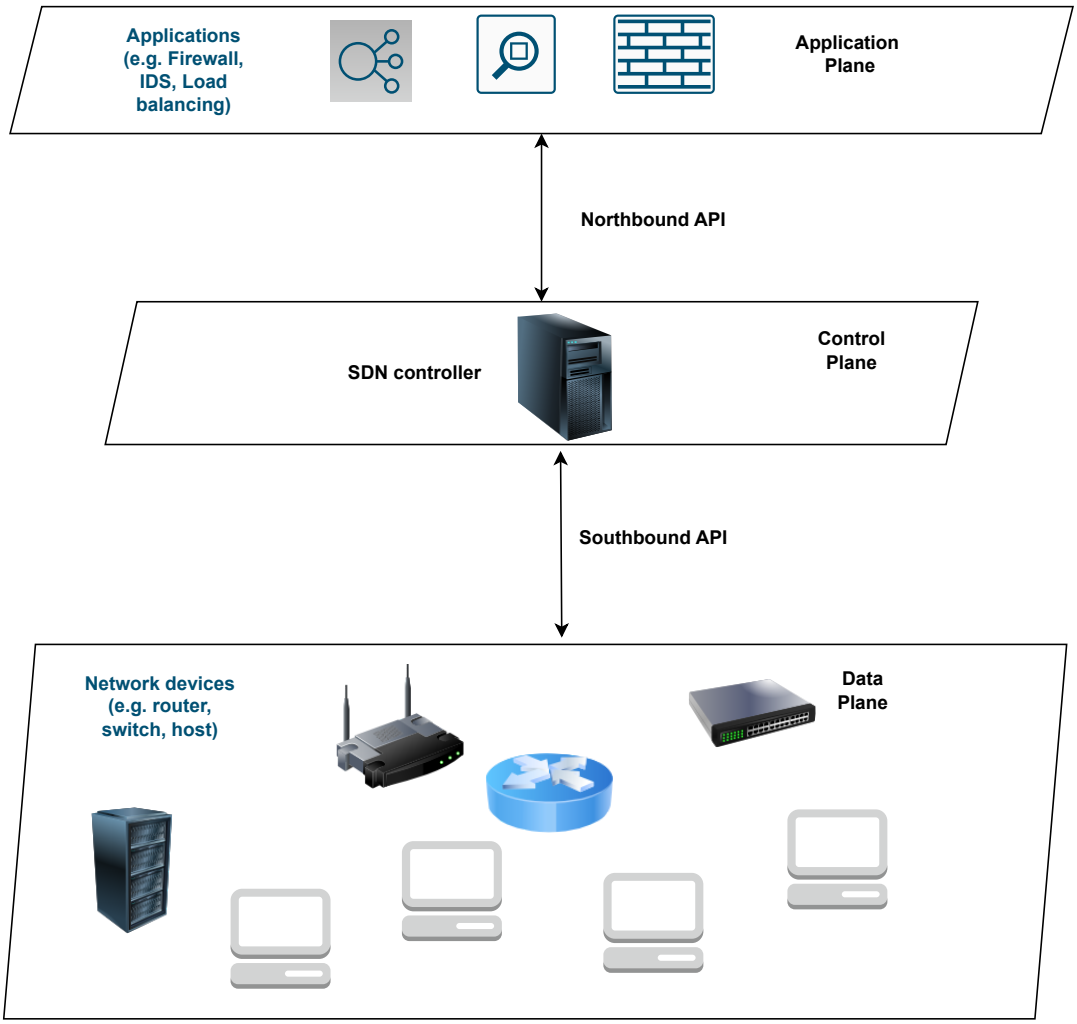

FIGURE 2.2: Three layers in SDN architecture

2. Data plane: In the traditional network, network devices are coupled with the controller in the same device. In SDN, the network devices are decoupled, making network infrastructure responsible for forwarding network packets [17]. Network elements (devices) such as switches, routers, hubs, modems and bridges lie in this layer. SDN devices can be in a virtual or physical model [18]. The SDN controller is considered the network brain. It communicates with the network plane using the southbound interface (SBI) APIs, e.g. using OpenFlow and Programming Protocol-independent Packet Processors (P4) [13]. Every switch has one or more flow tables consisting of rules to forward, drop, modify, etc. packets. These rules are inserted and updated by the controller. The network devices handle the incoming packet by checking the information in the packet header. This information is checked against the look-up table in the network device. The decision is made, for example, to forward the packet to a specific port, forward it to the controller, or drop the packet. If the information in the packet header does not match any of the rules in the flow table, the switch contacts the controller for further instruction, e.g. how to update the flow rule table to handle this packet [19].

3. Application plane: The application plane consists of one or more applications running on top of the SDN controller. Application developers can use APIs provided by the controller to program the network devices to meet enterprise requirements. SDN applications may include security applications (e.g. firewall and IDS/IPS), operator services, management applications, traffic monitoring, statistical applications, load balancing, and vendor applications, to name but a few.

4. Northbound Interface: The Northbound Interface (NBI) is the channel (API) used for communication between the application plane and the control plane. The network operator uses these APIs to issue commands and network configurations which transfer to the controller.

5. Southbound Interface:

   The Southbound Interface (SBI) is the API or protocol that works as a communication channel between the SDN controller and network devices (data

plane). In the SDN controller, the received requests from the SDN applications are translated into specific commands and then pushed to the network devices using this API. OpenFlow, Programming Protocol-independent Packet Processors (P4), Network Configuration (NETCONF), and Simple Network Management (SNMP) are examples of SDN Southbound APIs. In addition, a number of companies have produced their commercial APIs.

OpenFlow (OF) is a standard communication protocol used in SDN for communications between the controller and network devices. The SDN controller receives information from applications running on the application plane. This information is manipulated and fed into flow entries in SDN switches. It allows the SDN controller to determine how to deal with incoming packets, such as forwarding to a specific port, forwarding to the controller or dropping. OpenFlow can also be used to gather information from network devices.

Programming Protocol-independent Packet Processors (P4) is a programming language used in the communication between the SDN controller and the network devices. It is a programming language that describes the network behaviour in SDN devices. Compared with OpenFlow, it provides advantages by allowing the network administrator to program these devices at a much more granular level. The programmer can modify and define the fields in flow tables. It also allows the developer to define actions that can be applied when an instance of a packet matches a particular flow entry [20].

OpenFlow and P4 have different architecture paradigms. In OpenFlow, network devices such as switches or routers have fixed-function hardware pipelines. The chips in OpenFlow network devices, like all traditional network devices, are fixed-function and provide functionality to the device OS. This design is called “bottom-up”. OpenFlow uses flow-based information to define rules and matching mechanisms. The main limitation of OpenFlow is it is limited to a fixed set of supported header fields. For example, the matching process in OpenFlow is applied to specific (exact or wildcard) characteristics on packet header information. This limitation is overcome by P4 as it allows the network programmer to define the entire packet processing pipeline (Packet

TABLE 2.1: Comparison of aspects between OpenFlow and P4

Aspect	**OpenFlow**	**P4**
Packet Processing	Flow-based	The entire packet processing pipeline (flow-level and metadata)
Granularity of control	Flow-level (packet header)	Finer granularity by processing the entire packet (flow-level and metadata)
Flow Rules	Defined based on packet header fields	Defined based on packet header fields and metadata
Adoption	Widely adopted by network hardware vendors and has been dominant for a long time	Gaining traction and largely adopted in recent years by network device vendors
Support community	A large support in the SDN research community with a high number of tools and techniques	It has significant support from network devices vendors and Open Network Foundation (ONF) [22]. (SDN researchers however mostly focus on OpenFlow.)
Approach	Bottom-Up	Top-Down

header fields and metadata). They can define forwarding behaviour, packet parsing, and processing logic. Rather than applying exact or wildcard matching, the developer can customise matching logic allowing them to go beyond exact (or wildcard) matching. They can create complex matching conditions based on specific packet characteristics. These features provide more flexibility and adaptivity to programmable switches in P4 [21]. With special chips, P4 employs a "top-down" paradigm as the network programmer uses the P4 program to define a set of network features. The P4 program compiles the code and then injects the configuration into the network device. Figure 2.3 shows the difference in the architecture between OpenFlow and P4. Table 2.1 summaries the differences between OpenFlow and P4.

OpenFlow was the dominant protocol in SDN environments for many years. However, P4 has been the most widely used API in recent years. The evolution from OpenFlow to P4 gives the network operator more flexibility to implement more detailed commands to SDN network devices. P4 has many advantages for SDN; however, it is not a replacement for OpenFlow. OpenFlow

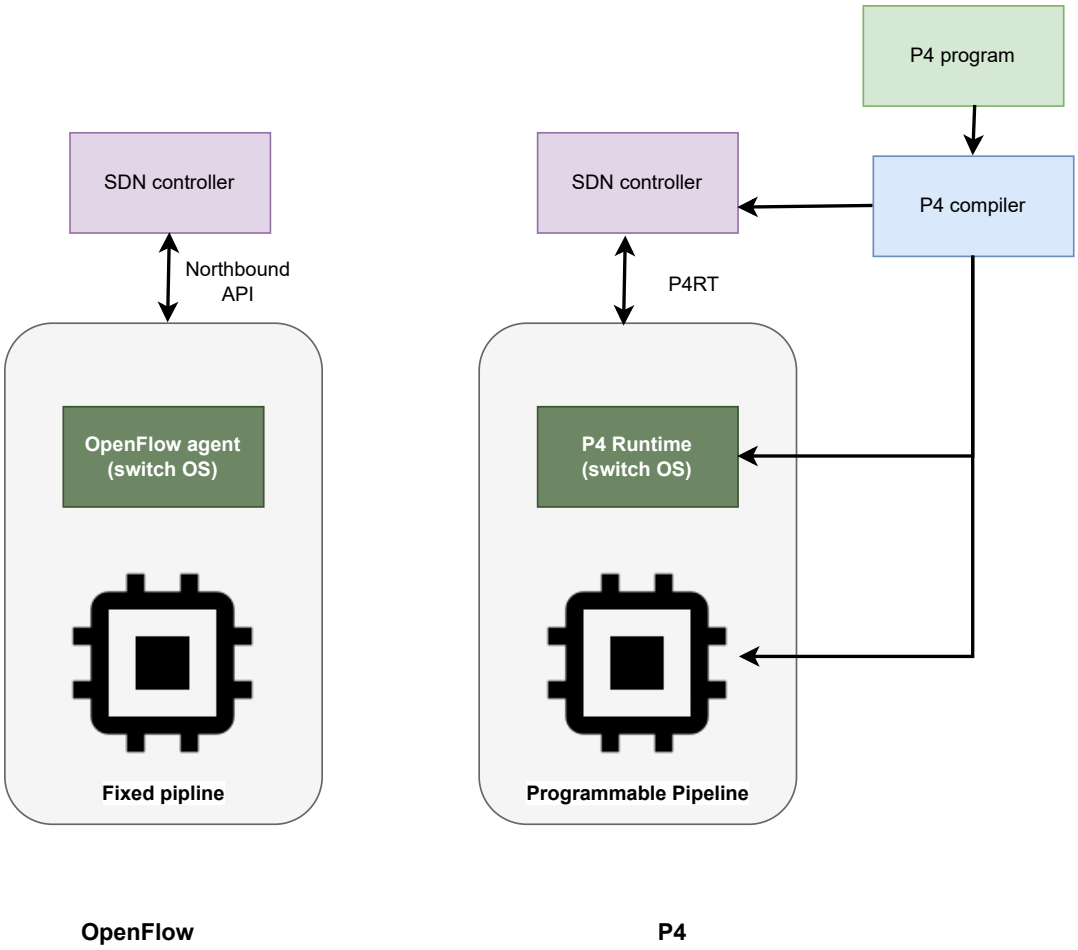

FIGURE 2.3: Evolution from OpenFlow to P4 [1]

can be a part of P4 and used in implementing a number of P4 programs such as P4Runtine [23].

2.2.2 Threats to SDN

The separation of the network plane from the control plane, together with its dynamic programming, brings improvements to the network. The global visibility of the controller makes the monitoring of network traffic easier than in a traditional network and allows easier implementation of security policies. Another benefit is the amount of information that can be gathered from the network flow. This is a challenge in conventional networks [22].

On the other hand, the new relationships between network elements in SDN networks give rise to security issues. The most commonly known attacks on conventional networks can be used against SDN, but the effects are different due to the

new architecture of SDN. Man-in-the-middle, spoofing, hijacking, DoS, tampering, repudiation, information leaks, and privilege escalation are examples of these attacks [12]. However, the centralisation of the controller creates a new set of attacks in SDN. A single controller (some SDN networks have multiple controllers) is an attractive target for attackers[24]. Moreover, the ease of deploying applications on the controller could allow adversaries to develop and deploy their malicious applications there. Reconstructing flow rules in SDN switches is a significant threat in SDN. The attacker can send probing packets to gather information about the flow rules in SDN devices. (This will be discussed in detail in 2.2.3.) The gathered information is a preliminary to further attacks. The following are examples of potential attacks in SDN:

1. Unauthorised access: An attacker can perform brute force or password-guessing attacks to get access to the SDN application or the management plane. In addition, the attacker can exploit software vulnerabilities to get access to SDN applications. Therefore, they can get access to different network resources without permission [25].

2. Data modification: One of the disadvantages of the centralisation of the controller is the possibility of attackers controlling the entire network if they succeed in hijacking the controller or spoofing its identity. As a result, the attackers can insert or modify flow rules in the network elements. They can route data based on their aims, e.g. they may route it to themselves [25].

3. Information disclosure: In an SDN network, adversaries can gather different information than in traditional networks. The flow rules in SDN switches are one of the attackers' targets in SDN. The adversaries can send probing packets to reconstruct flow rules. Reconstructing flow rules leads to further attacks, such as discovering the running applications and security policies. Another type of attack is the side-channel attack which is an attack that depends on the information that is gained from the physical implementation of the computer system, for example, the time of data processing in switches. In SDN, a switch applies certain forwarding policies on every packet: drop, forward, or send to the controller. The attacker can analyse the time of packet processing to do

that. For example, a packet sent to the controller will take longer than one immediately forwarded to another switch port. The gained information can be used to launch attacks on network elements or the SDN controller (such as DoS on the controller) [26; 27; 28; 12].

4. Malicious applications: A malicious or compromised application can launch several attacks on the controller, such as compromising or modifying configuration or management data, fetching flow table rules and contents, network topology and other sensitive information [12]. One SDN characteristic is the ability to integrate third-party applications - applications that are developed by a provider who is not the same provider of the SDN controller and network elements. In this case, the attacker can more easily get information about the SDN controller or other applications. Furthermore, it is more dangerous if the application itself is an attacker. A trusted connection and authentication are required to prevent malicious applications from connecting to the network [12].

5. Denial of Service (DoS): The separation of the control plane and the data plane causes new attack vectors. The centralised and global visibility of the controller makes it the most attractive element for attack by adversaries. One of the most difficult challenges in SDN architecture is the single point of failure when an attack on the SDN controller succeeds. Once an attacker launches a DoS attack on the controller, the entire network would fail. For example, an attacker can flood the controller with rule decision requests. When the number of requests is more than the size limit of handling requests by the controller, it is prone to miss a number of requests or go out of service. Similarly, an attacker can overflow flow tables in SDN devices, making them unable to handle new transmissions [29; 30].

6. Elevation of privilege: A number of attackers, such as APTs, often elevate their privileges once they get inside a victim's network and so gain unauthorised access to resources or applications. If an attacker gets higher privileges on the SDN controller, they could control the whole network. In addition, application services could be a target [13].

TABLE 2.2: General threats to SDN and their targeted interfaces

Type of attack	App Layer	App-Controller Interface	Control Layer	Control-Data Interface	Network Layer
Unauthorised access	✓	✓	✓	✓	✓
Data modification	×	×	✓	✓	✓
Data breach	×	×	✓	✓	✓
Timing attack	×	×	×	✓	✓
Malicious applications	✓	✓	✓	×	×
DoS/DDoS	×	×	✓	×	✓
Elevation Privilege	✓	×	✓	×	✓

App: Application;

Table 2.2 presents a summary of in which interface or layer the attack in SDN can happen.

2.2.3 Reconstructing flow rules in SDN

An OpenFlow switch has one or more flow tables in which every entry of a table has three main components:

- Match fields: these determine the criteria for matching incoming packets. Match fields consist of packet header fields, ingress port, and metadata (optional).
- Counters: these maintain counts of various occurrences in the flow data, such as the number of matched packets or bytes.
- Action: the action that would be executed when a packet matches a rule, for example, forward the packet to a specific port, forward it to the controller, or drop it.

When a new packet arrives at an SDN switch, the switch performs a lookup to determine how to handle the packet, e.g., where it should be forwarded. Each flow table contains a set of flow entries to manage the incoming traffic. The flow entries contain matched fields, counters, and a set of instructions that are to be applied to the incoming packets. If no appropriate (matching) entry for the packet can be found

in the flow table, the switch contacts the controller via the southbound interface to request updates for its flow table [31].

Figure 2.4 shows an example of an attacker sending probing packets to reconstruct flow rules in OpenFlow switches. They can send probing packets to a host to infer the required details to reconstruct flow rules in OpenFlow switches. This helps the attacker to discover which forwarding policy is applied when a new packet arrives. By reconstructing flow rules, the attacker can then launch further attacks such as bypassing Access Control Lists (ACLs), avoiding defensive tools such as Moving Target Defence (MTD) techniques, and learning what applications are running in the application plane (such as the load balancer) [32; 33; 8].

SDNmap [34] is an open-source SDN scanner that allows the user to reconstruct OpenFlow rules in SDN. It sends a sequence of probing packets to infer matching fields and applied actions. SDNmap can scan one or more hosts by specifying one or a range of IP addresses. The following are examples of the applied scanning steps taken by SDNmap to infer flow rules in OpenFlow switches.

- MAC address: To infer the MAC address, the attacker generates a spoofed MAC address and sends a prob request (TCP or ICMP) to the destination host (target). The host will lookup in its local ARP cache for the corresponding IP address. If the MAC address is founded in the entry, it will reply with the MAC address which is presented in its local ARP cache. But if the IP is not listed in the local ARP cache, the destination host will send an ARP request to resolve the MAC address of the sender. The host then will update its local ARP cache based on the reply from the sender. These techniques can reconstruct the flow rules of matching MAC source address and the field of matching destination MAC address [8].

- IP address: In the same way, the attacker generates a probe packet with a spoofed IP to infer the matching fields of source IP and destination IP [8].

- Protocols and ports: The adversary can send probing packets to determine if the entry in OpenFlow matches specific protocols (such as ARP, ICMP, TCP and UDP) and determine which ports are open. The attacker can send ARP requests and listen to reply messages in a manner similar to that adopted when

reconstructing a MAC address. For ICMP, it can send an ICMP Echo Resuest and list to the Echo Reply. Also, it can send a TCP SYN and wait for an ACK, if the destination port is open, it will receive an ACK. Otherwise, it will receive a TCP RST. To determine if the UDP is matching flow rules, the attacker can send a probing packet with a spoofed IP and listen for an ARP request. Based on the reply messages, the attacker can infer which ports can be reached by a specific source and which protocols are supported [8].

- Ingress port: This refers to the input port in the switches. An attacker can perform checks by sending probing packets and analysing ARP requests to deduce whether the ingress port is part of the matching flow rule or not. The attacker could impersonate another host in the sub-network to infer the ingress port field. The attacker spoofs the other host's ARP cache. If the attacker receives an ARP request from the recipient, it indicates that the ingress port is not checked in the flow rule matching criteria. If the attacker does not receive an ARP request, they can infer that the ingress port is a part of the matching flow rule criteria [8].

- Rewritten IP addresses: In this procedure, the attacker can infer the actions in the flow rule if that flow rule performs IP address rewriting (for source and destination). Initially, the attacker sends a UDP probing packet to a closed port. The destination should send back an ICMP error message (Destination Port Unreachable). This message reveals the actual IP addresses that it received that differ from IPs in the probing packet sent by the attacker. That indicates the action of rewriting IP is performed in the flow rule [8].

- Forwarding action: The attacker can infer the flow rule action of forwarding the received packet in the OpenFlow switch to a specific port or of dropping that packet. This rule can be inferred based on the analysis of the previous scanning steps (reconstructing the fields of MAC addresses, IP addresses, and protocols and ports). The forwarding action can be inferred by analysing the reply messages. The attacker infers that the action "drop" is performed when there is no reply message[8].

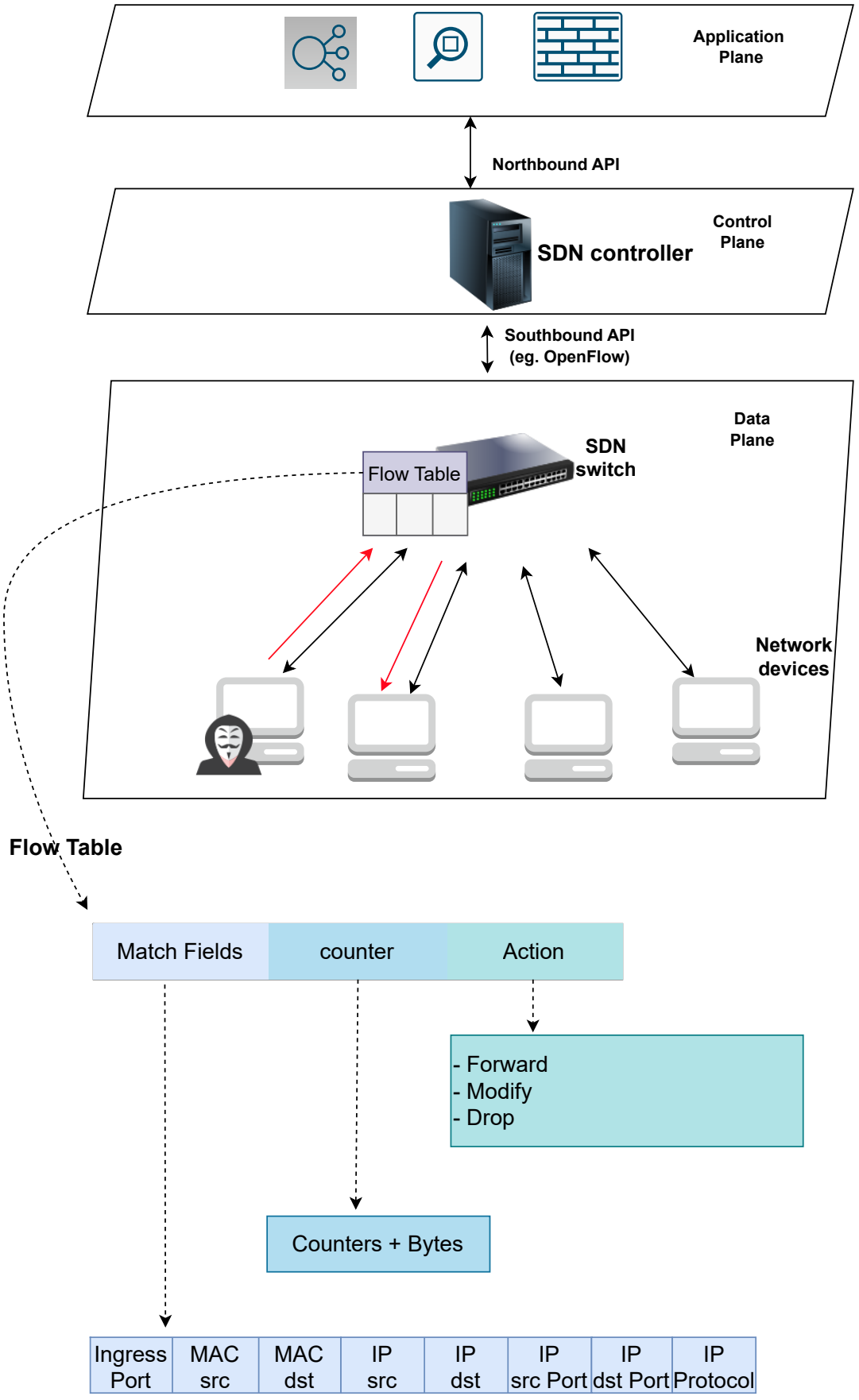

FIGURE 2.4: Reconstructing flow rules in SDN

TABLE 2.3: A comparison between APT and traditional attacks [7]

	Traditional Attacks	**APT attacks**
Attacker	Mostly single person	Highly organised, sophisticated, determined and well-resourced group
Target	Unspecified, mostly individual systems	Specific organisations, governmental institutions, commercial enterprises
Purpose	Financial benefits, demonstrating abilities	Competitive advantages, strategic benefits
Approach	Single-run "smash and grab", short period	Repeated attempts, slow-and-low strategy, adapts to resist defences, long term

2.3 Advanced Persistent Threats

Advanced Persistent Threat (APT) attacks target a specific system using sophisticated tools and approaches. In some cases, it is a set of continuous and stealthy attacking processes remaining in the target system for a long time [35]. Usually, these attacks target nation-state organisations or big companies, mostly for political, military or economic benefit. The impacts are usually far more insidious than traditional attacks. Table 2.3 compares APTs and traditional attacks. Attackers behind these attacks are usually highly skilled, well-funded and well-resourced. They are governments, activists or cyber-criminals. If they work for a government, they may have military or intelligence agency support. Despite a large number of security researchers saying that APT attacks require highly skilled attackers, they can often be launched with basic skills [7; 36]. In addition, the majority of APT attacks use known vulnerabilities [2]. There is no one behaviour of APTs; however, a number of characteristics are common. They are listed below:

1. *Low and slow*: attackers stay in the system for a long time without detection. Unlike a traditional "smash and grab", they move slowly and quietly inside the network, from one system to another, until they have accomplished their goals [37].

2. *Stealthiness*: refers to the ability of an attacker or their malicious activities to remain undetected by security measures and avoid raising suspicion. APTs employ techniques that minimize traffic volume, making their activities resemble those of regular users (by utilising the earlier strategy "low and slow"). If they intend to transmit the gathered data to the attacker (Command and Control C&C), which is outside the victim network, they use covert communication channels. In addition, some APTs clear any evidence of their presence that may be noticed by the incident response team.

3. *Adaptability*: people behind APT attacks adapt themselves to resist defenders' tools and efforts. Furthermore, they may use zero-day exploits in their attacks to avoid a victim's defences.

4. *Customised attacks*: in addition to traditional attacking tools, APT attackers often use tools and malware customised for their specific targets and goals. They spend as much time as possible gathering information about the target including the defence tools used, its environment and its vulnerabilities. They may then develop new tools exploiting vulnerabilities (which may be zero-day).

5. *Persistence*: APT attackers spend a great deal of effort to achieve their goal. If at first they fail, they adopt alternative approaches until they succeed, and stay as long as they can without being detected.

2.3.1 High profile APT examples

Five high-profile APTs are discussed below:

1. Flame: Also known as sKyWIper, this targeted many systems in Europe and the Middle East. The main aim of Flame was to collect secret and sensitive information on Microsoft Windows PCs. They use different techniques to collect data such as keystroke logging, screen capture, switching on microphones and cameras, and collecting data from external drives. The malware can switch on the Bluetooth of an infected device, if available, to collect data from devices within range that are also using Bluetooth. If a network connection is available, Flame sends the gathered information using the C&C server to the attackers.

If there is no connection, collected data is saved in a USB drive until that drive is connected to a device with a network connection. Consequently, the newly connected device is infected. The malware can propagate inside the network in one of two ways: by exploiting the print spooler and LNK; or by sending the malware to another computer in the same network as a Windows Update [38; 39].

2. Operation Aurora: Early in 2010, Google announced on its blog [40] that itself and more than twenty big companies had been targeted by an advanced attack. The operation targeted the infrastructure and resulted in stealing intellectual property from Google. In their report, they claim that the source of the attack was China. McAfee in their investigation report [41] noted that attackers exploited an unknown vulnerability in Microsoft Internet Explorer. Moreover, a vulnerability in Adobe Reader and Acrobat applications was exploited to get access into the network of some infected firms [2]. Interestingly, the later variants of this malware no longer use unknown vulnerabilities. Generally, the attack, which started in 2009 and lasted for months, gathered information about the system and network, including the usernames and passwords of its victims, and sent it to the command and control server. The IP and domain of the server were hardcoded in the malware [2].

3. Stuxnet: This is one of the widely recognised APT attacks to date and was discovered in mid-2010 [42; 43; 44; 45] although the first sample was known to date from 2007. Some instances were still active in 2012. The main aim was to damage or delay the Iranian nuclear program. Attackers behind that attack, widely thought to be state-sponsored [45; 46], targeted centrifuges in a uranium enrichment plant's network. In mid-2009, attackers started by infecting the SCADA system in the isolated network (most likely, using an infected USB drive). The malware then spread inside the network, exploiting a zero-day vulnerability in a print-spooler. As a result, about 1000 centrifuges were damaged. Although the malware had a specific target, the Iranian nuclear program, it spread around the world after one engineer connected a work laptop to his home network. Stuxnet also exhibited remarkable counter-analysis

measures, for example intercepting operator interactions with the system and faking responses to enquiries.

4. Shamoon: After Iranian nuclear facilities were attacked by Stuxnet [45], attackers learned from the experience and created Shamoon to attack one of the biggest oil companies in the world - the Saudi Arabian oil company called Saudi Aramco [47]. The attack happened in 2012, damaging about 30000 Windows based personal computers. Inserting the malware for the first time in a workstation within the network was not easy and may have required the involvement of someone with physical access. The malware was designed, after infecting the first machine, to propagate at a specific time over the network, overwriting the hard drives of infected computers. Although Shamoon was designed for a specific target, it spread to other oil companies such as the Qatari company RasGas and the American ExxonMobil [48].

5. SolarWinds attack: In December 2020, FireEye [49] reported that one of its penetration testing tools had been stolen by a national-based group (APT)[50]. The attackers successfully inserted a stealthy backdoor into a large number of networks in companies and organisations across the world. The attackers exploited a vulnerability in SolarWinds' Orion Platform (Orion) [51] to deliver their malware [52]. It is reported that the attack started on March 2020 and remained undetected until the end of the same year [53].

Table 2.4 summarises the most notable APTs examples.

2.3.2 APT stages

A successful APT attack goes through several stages. The number of stages differs from one attack to another. Although most stages of APT attacks are similar, there are some differences in the literature [2; 37; 54; 55; 56; 57; 58; 59]. Table 2.5 presents the different APT stages in the reported research. Based on these studies, the common APT stages can be identified:

TABLE 2.4: Example of some notable APT attacks

Attack	Period	Target	Results
Flame [38; 39]	2007 - 2013	Academia Research, Government entities, Specific individuals in Egypt, Europe, Iran, Israel, Lebanon, Palestine, Saudi Arabia, Sudan, Syria, Ukraine	Stealing information
Operation Aurora [2; 40; 41]	2009 - 2011	Commercial companies	Theft of sensitive information
Stuxnet [45]	2009 - 2012	Iran's uranium nuclear project	About 1000 centrifuges were damaged and the nuclear program was delayed
Shamoon [48]	2012 - 2013	Energy, oil and gas companies in Saudi Arabia	Destruction of about 30000 personal computers
SolarWind attack [50]	2020	Global private and public organisations	Stealing information

- Stage 1: Reconnaissance - The first step of any attack is gathering information about the target. Social Engineering, open-source intelligence (OSINT) tools and different scanning techniques can be used in this stage.

- Stage 2: Establish a Foothold - after gathering the necessary information, it is time to break into the target's network. Typically, attackers construct different vectors of attack using different tools. Exploiting vulnerabilities in the victim's system, installing malware and spear-phishing are examples of methods and techniques used by APT attackers to accomplish their foothold. APTs are persistent and spend a great amount of time trying to infect the victim network. APTs may target an isolated network, not connected to the internet, or one well protected against cyber-attack. In this case, the attackers may use social engineering methods to deceive someone inside the target network to download the malware. However, an employee inside the targeted organisation could easily install the malware deliberately. This is commonly referred as an 'insider' threat.

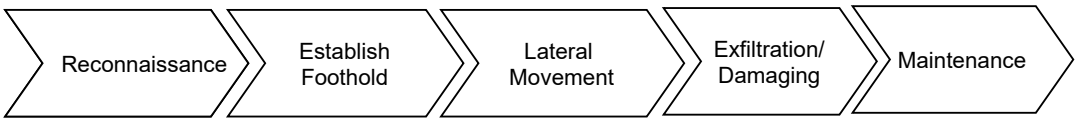

FIGURE 2.5: APT stages

- Stage 3: Lateral Movement - The attackers move inside the network from one component to another until they reach their targets. They may escalate their privileges during the movement. To avoid detection, APT attackers usually move stealthily using valid credentials. Therefore, previously stolen passwords may be used in this stage. Mimikatz [60] and Windows Credential Editor (WCE) [61] are popular examples of credential dumping tools that APT attackers could use. Extracting and analysing parts of the Windows Local Security Authority Subsystem Service (LSASS) process is another technique used for dumping credentials [62]. Attackers may keep scanning the network looking to get useful resources that help for dumping credentials.

- Stage 4: Exfiltration or damaging - If the attacker's goal is to get information or damage part or all of the system, it would happen in this phase. The attacker uses a command and control C&C server to communicate with the compromised system to extract stolen data or perform commands that the attacker would like to execute, such as updating malware, changing the C&C IP address and disabling systems. However, it is crucial that the rate of exfiltrated data is low during the communication to reduce the risk of it being considered anomalous traffic.

- Stage 5: Maintenance - The APT attackers usually maintain access using a back door open in the compromised network for future initiatives. They will typically test their accessibility periodically. If the access is lost, they may move back to the reconnaissance or foothold stages. Moreover, they may cover their tracks and clear any evidence, for example, by deleting logs.

2.3.3 APT defence methods

The sophisticated and advanced attacks in APTs' campaigns make the detection or prevention of these attacks challenging. APTs employ different tools or techniques

TABLE 2.5: APT stages in existing research

Ref.	stage 1	stage 2	stage 3	stage 4	stage 5	stage 6	stage 7
[2]	Recon	Establish foothold	Lateral movement	Exfiltration Impediment	Post-exfiltration Post-impediment	-	-
[37]	Incursion	Discovery	Capture	Exfiltration	-	-	-
[54]	Recon	Preparation	Targeting	Further access	Data gathering	Maintenance	-
[55]	Collect intelligence	Find a point of entry	Call home	Search for data assets	Move thought the network	Extract data	-
[56]	Recon, Launching and Infection	Control and management, Detection, Persistence	-	-	-	-	-
[57]	Recon	Compromise	Maintaining access	Lateral movement	Data exfiltration	-	-
[58]	Recon	Weaponisation	Delivery	Exploitation	Installation	C&C	Actions on objectives
[59]	Initial compromise	Establish foothold	Escalate privileges	Internal recon	Lateral movement	Maintain presence	Complete mission

in different stages. To detect (or prevent) this type of attack, It is recommended to implement different layers of defence. Correlating events, monitoring traffic, checking logs and pattern matching are known approaches to detecting traditional attacks, which would also help detect APTs. In [2], the authors classify defence methods into three major categories: monitoring, detection, and mitigation methods, as shown in Figure 2.6. More details about these methods are given below.

The first category comprises monitoring methods that are used to monitor suspicious activities. Monitoring methods include:

1. Disk monitoring: Monitoring all systems in the organisation's network is important to detect malicious activities. Monitoring the CPU usage can also help in detection. It is crucial to patch any vulnerabilities found in these systems [2].

2. Memory monitoring: Some types of malware, called fileless malware, execute not from a file, but from within memory. These malware are sophisticated and difficult to detect by anti-malware software. Monitoring memory usage helps in the detection of this type of malware.

3. Packet monitoring: Monitoring network traffic is a common practice in network security solutions. The investigator can monitor the network flow for abnormality or go deeper by inspecting the packets' payloads. A high number of packets, the amount of traffic, and the passing of some traffic over a port

that is not designed for it (for example, a non-http over port 80) are examples of suspicious behaviours.

4. code monitoring: Attackers usually seek vulnerabilities in systems and applications working in the network. It is important to check and analyse the software code for any vulnerabilities attackers could exploit. Moreover, monitoring the software behaviour on the network could help to identify possible vulnerabilities [63].

5. Log monitoring: Correlating different logs can help to detect even unknown attacks. Memory usage logs, CPU usage logs, applications logs and system logs are examples of these log files. The biggest challenge in detecting attacks using log monitoring is the large amount of data to be monitored and analysed [64].

Some APTs are previously known, but some are unknown or zero-day. Two main approaches to detecting APT attacks are:

1. Anomaly detection: APT attackers are persistent and usually implement evasion techniques in their attacks. Also, they sometimes use unknown malware to evade defence tools. Anomaly detection techniques profile normal behaviour and identify abnormal behaviour as deviating significantly from that profile [65]. However, not only do APTs apply evading techniques, but also sometimes behave in a 'normal' manner or close to it.

2. Pattern matching: Also known as signature-based detection. It is a way of detecting a predefined pattern (or signature) of attacks. Known attacks are identified then profiles of these attacks are constructed. When a pattern is encountered again, the classifier can detect it.

The following are the two main categories of mitigating attacks:

1. Reactive methods: The method is based on analysing the vulnerabilities in the system and the possible paths used during the attack. Graph analysis is one example of this mechanism to detect attacks [2; 66].

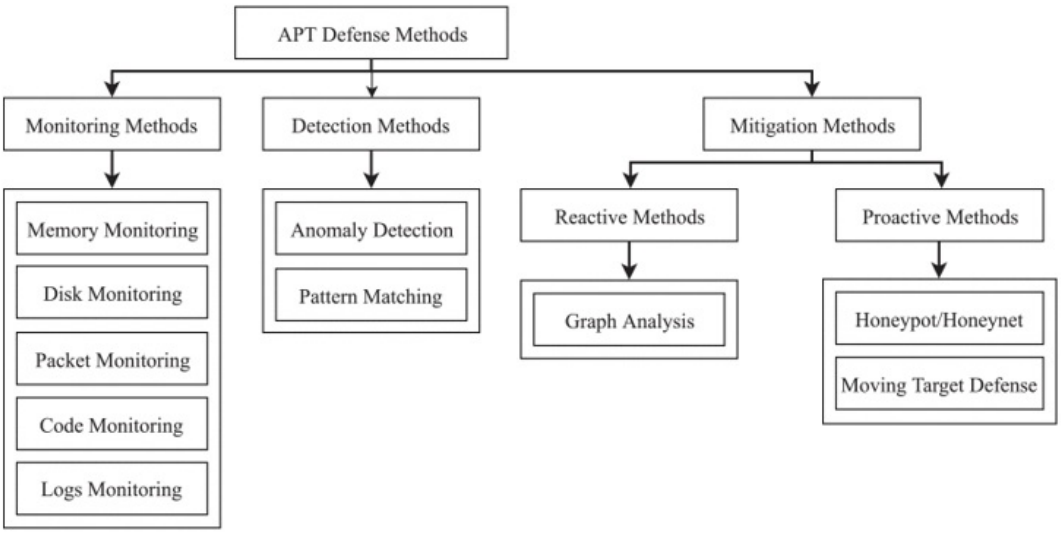

FIGURE 2.6: APT defence methods [2]

2. Proactive methods: One of the most used ways to prevent APTs is to misinform the attacker about the system. For example, honeypots, or honeynets, are hardware or software that work as a decoy presented to the attacker as a part of the production environment. Analysing accesses to the honeypot or the honeynet can lead to exposing the attack [67]. Moving Target Defence (MTD) is another example of confusing the attacker by reconfiguring the network, making it hard for attackers to collect information about the system [68].

2.4 Intrusion Detection Systems

An Intrusion Detection System (IDS) is a system or a device that monitors a network or a host to detect suspicious activities. When an attack is detected, it generates alerts to the administrator or the security team. Based on the alert, the admin can investigate the alarm and take the appropriate actions, such as updating the firewall (or flow rules in SDN) or blocking the sender. Figure 2.7 illustrates how IDS makes decisions based on predefined knowledge. Signature-based and anomaly-based detection are the two main detection techniques adopted by IDSs. A Network Intrusion Detection System (NIDS) is a type of IDS that monitors network traffic to detect suspicious attacks. Another type of IDS based on the source of data (location

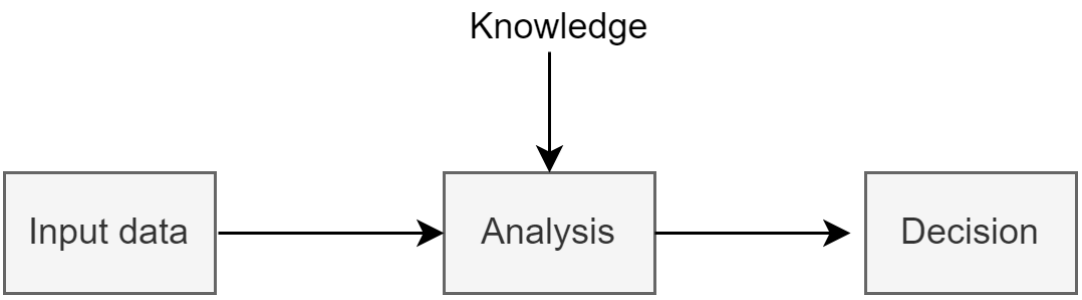

FIGURE 2.7: IDS making decisions based on predefined knowledge

of the IDS) is Host-based IDS (HIDS). An IDS taxonomy can be more detailed, as in Figure 2.8; however, the main categories are now discussed.

2.4.1 Detection approach

- Signature-based IDS: Signature-based approaches also called *misused-based*, detect abnormality based on predefined patterns or *signatures* of attacks. This is a significant means to detect known attacks and is known to exhibit low false alarm rates (i.e. it rarely classifies benign traffic as malicious). However, detecting unforeseen malicious activities that have different properties to known malware is a significant challenge for this approach.

- Anomaly-based IDS: In anomaly-based detection, the system is trained over normal traffic and builds a profile of it. When the behaviour of a user differs significantly from its normal profile, a potential attack is suspected.

2.4.2 Monitoring approach

As shown in figure 2.9, IDSs can be categorised based on the source of audit data as:

- Network-based IDS (NIDS): The NIDS (software or device) is placed on the network, monitoring network traffic inside, exiting or entering the network.

- Host-based IDS (HIDS): Here the system is installed inside one or more hosts that it is supposed to monitor. A HIDS monitors and analyses internal activities and files. In addition to analysing running software and local files (e.g. log files), it can inspect incoming packets to the host.

Table 2.6 summarises the strengths and drawbacks of each type.

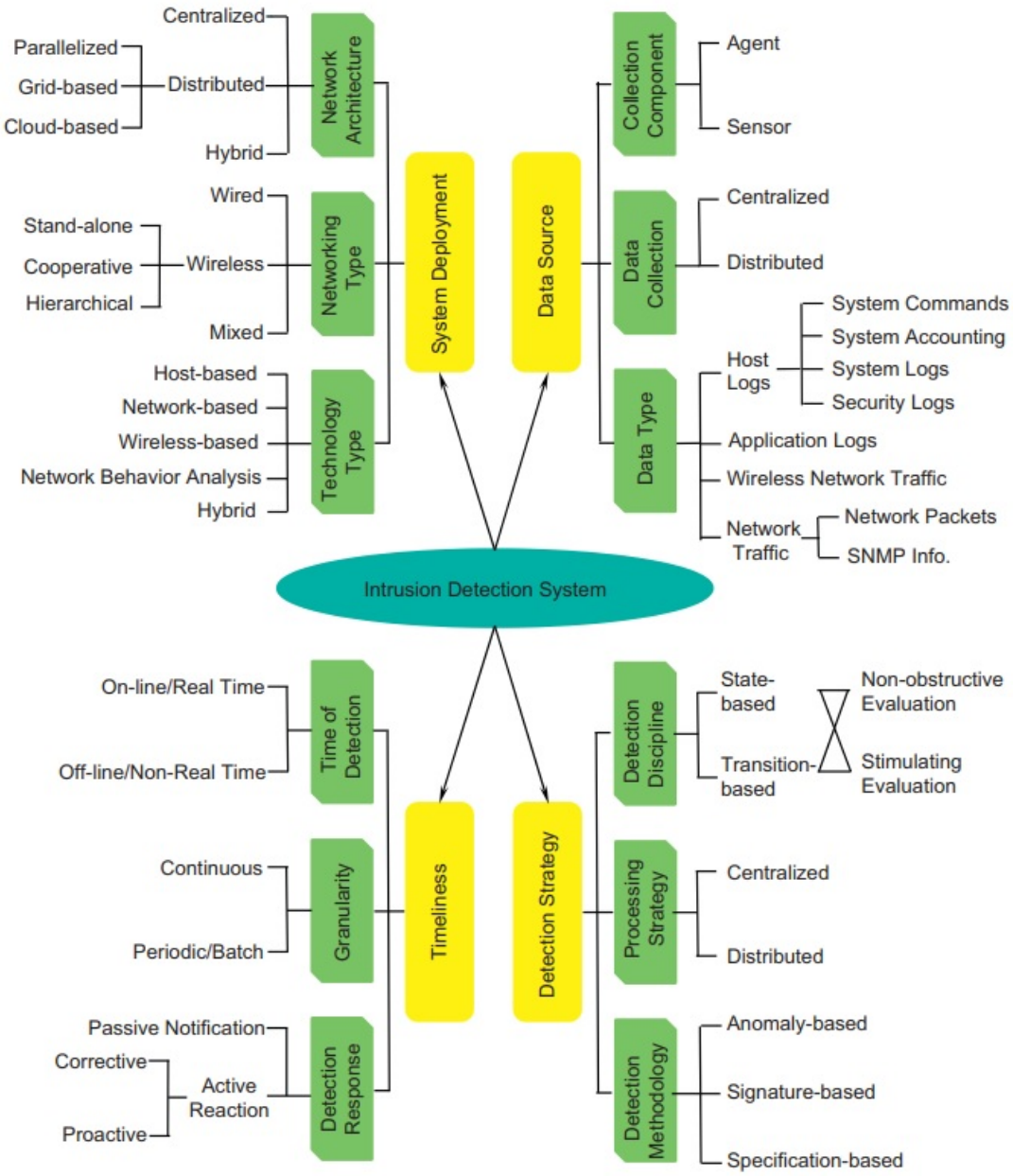

FIGURE 2.8: Detailed IDS taxonomy [3]

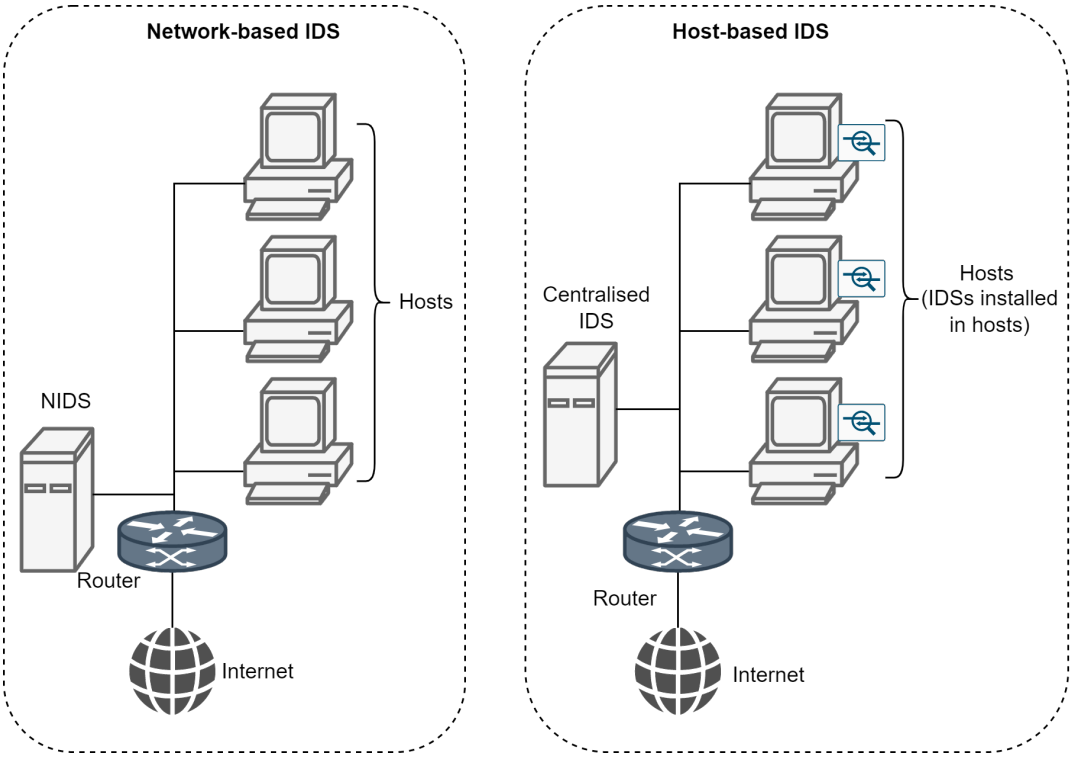

FIGURE 2.9: Types of IDS based on their source of data

TABLE 2.6: IDS location-based comparison

IDS	Strengths	Drawbacks
NIDS	Analyse the whole network communications	Difficult to analyse encrypted data
	One machine can cover the whole network	In some cases needs to use extra hardware
	Checks the traffic immediately	Little information is checked
HIDS	Easier to analyse encrypted data	Inspects only hosts that it installed in
	No extra hardware	Needs to be installed in every host that is considered to be protected
	Analyse a large number of files	Delay to report attacks

TABLE 2.7: Comparison between IDS and IPS

Scheme	Strength	Drawback
IDS	Detect attacks No risk of impeding legitimate traffic	Can not prevent attacks The action of preventing the attack may come late after the attack has succeeded
IPS	Prevent attacks	There is a possibility of preventing a legitimate transmission

2.4.3 Hybrid-IDS

Hybrid Intrusion Detection Systems combine two categories of systems. They might, for example, use two types of detection (signature-based and anomaly-based) or be based on their location as placed on the network and a host (NIDS and HIDS) at the same time.

2.5 Intrusion Prevention Systems

An Intrusion Prevention System (IPS) is a hardware device or a system that monitors network traffic (or a host) for suspicious activities. Rather than just reporting the attack, the IPS takes actions to prevent the attack. For example, blocking the source of an attack, dropping the suspicious transaction, updating the firewall with a new configuration and resetting the connection. Similar to an IDS, an IPS can be network-based, host-based or hybrid. Also, an IPS applies similar methods for the detection (i.e. signature-based or anomaly-based detection). As the detection techniques applied by IPSs have a high possibility of generating false positives, legitimate traffic can be prevented. This makes network administrators reluctant to implement IPSs, preferring IDSs instead. Table 2.7 summarises the strengths and weaknesses of both approaches.

2.6 Machine-Learning-based IDS

Machine Learning (ML) describes the ability of a machine to learn by practice on previous inputs and make decisions (predictions) on new samples. An ML-based

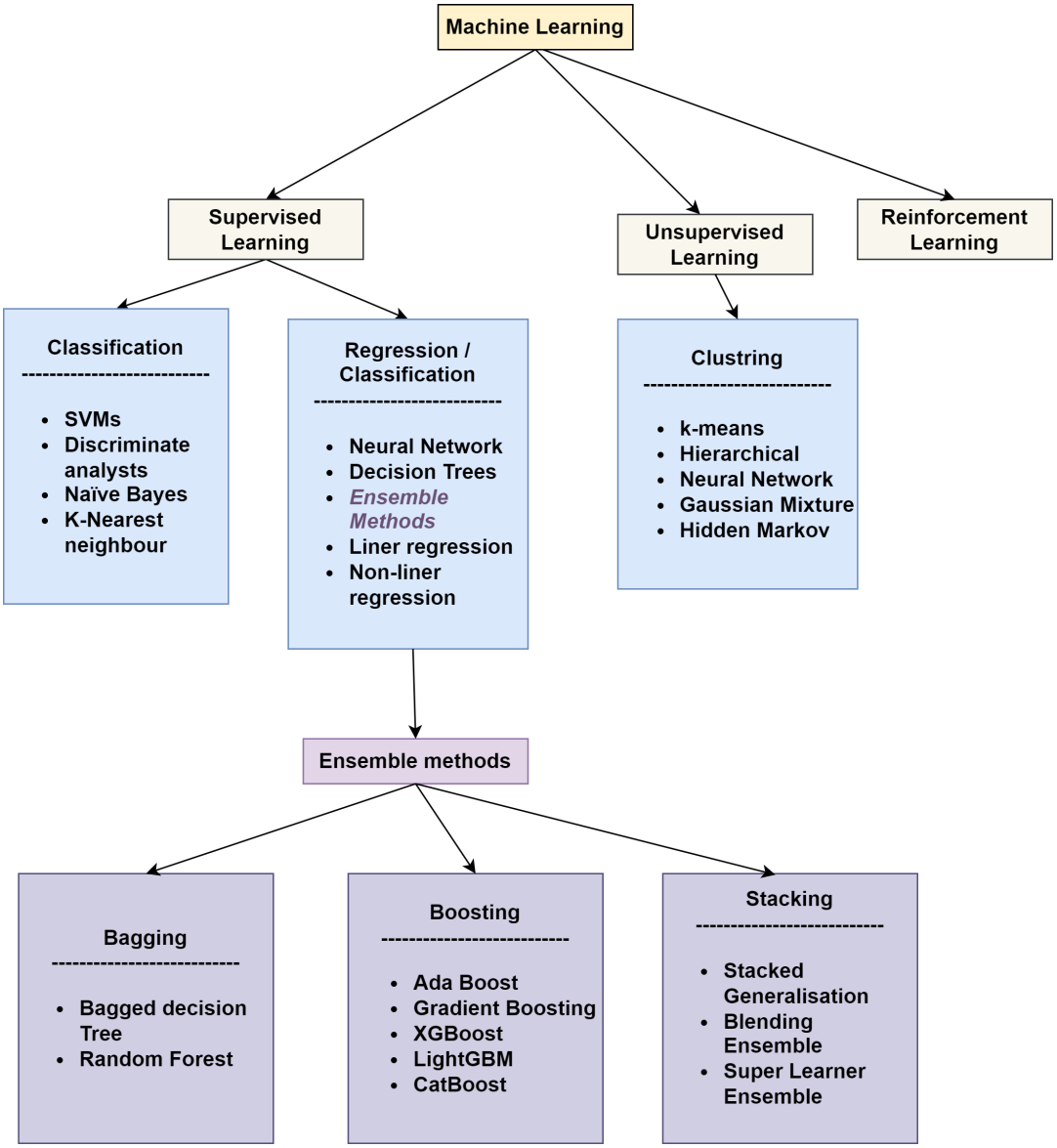

FIGURE 2.10: Machine Learning algorithms taxonomy

approach seeks to learn a prediction function f over a set of input features X. The value of $f(X)$ gives the target Y prediction (e.g. whether the inputs are malicious or normal).

$$Y = f(X) \tag{2.1}$$

Machine learning has been widely used for network security. It is the most used technology in Intrusion Detection Systems nowadays [69]. There are a large number of machine learning algorithms used by IDS researchers. Figure 2.10 shows examples of the most popular techniques based on their category. *Regression* and *Classification* are two approaches of machine learning. Regression algorithms predict a continuous value (quantity) such as price, salary and probability. On the other hand, the classification algorithms predict a discrete class or classes (category), such as True or False, benign or malicious, or the type of attack (e.g. DoS, probe or brute force). In classification problems, when the target has more than two classes, it is called multi-classification. Binary classification is the case when the target might be one of two classes (e.g. True and False, or, Malicious and benign).

2.6.1 Machine learning detection techniques

Machine learning can be categorised into three main categories based on the availability of labelled data (supervision of the developer):

1. Supervised learning: In this category, the model is trained over labelled data (i.e. data designated as normal or malicious). Supervised learning can be used for Classification and Regression. A reasonable number of ML techniques can do both based on the developer's needs. The following is a brief description of the most popular supervised algorithms for IDS researchers.

   Logistic Regression: This is easy to use and the go-to algorithm in binary classification [70]. Logistic Regression is a statistical model that estimates the probability of an instance being classified in one class (normal or attack). It has the ability to use discrete and continuous variables to classify new samples. Logistic Regression uses a logistic function, also called sigmoid, to draw an "S" shaped curve that maps any real number to a value between one and zero. If

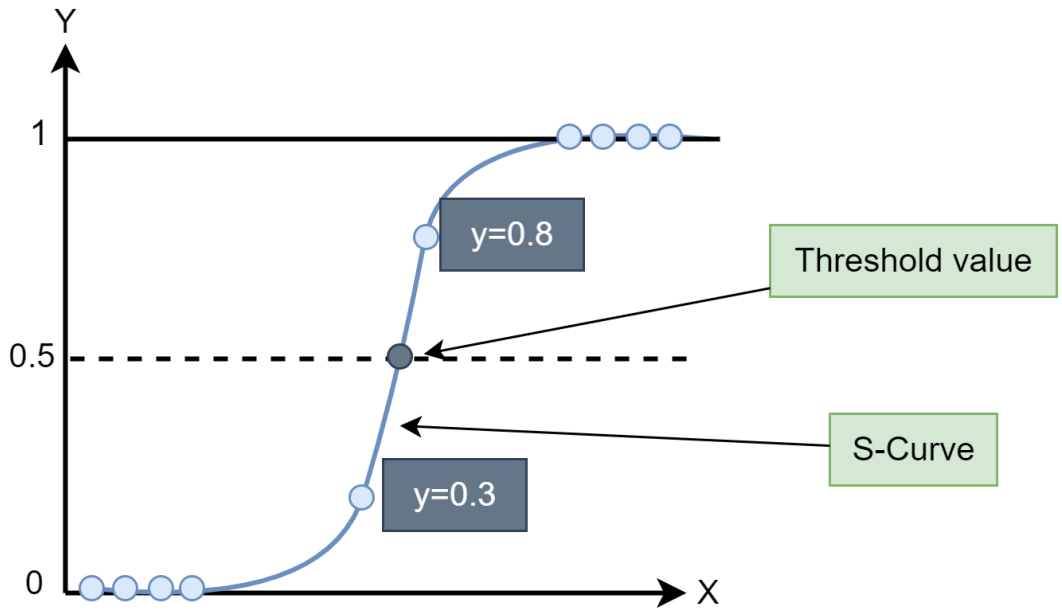

FIGURE 2.11: Logistic Regression in machine learning

the value is more than 0.5 then the predicted class is going to be 1, otherwise, it is 0. Figure 2.11 depicts the shape of Logistic Regression.

K-Nearest Neighbour (K-NN): K-NN is one of the most popular machine learning techniques used for regression and classification. For classification, the algorithm assumes that similar instances of one class are located near each other. When a new instance arrives it uses a distance function (metric) to measure the similarity of that instance with the nearest k instances (neighbours). An instance is assigned to the most frequently occurring class amongst the *k* nearest neighbours. Euclidean distance, Manhattan distance, Minkowski distance and Hamming distance are the most popular distance functions used by K-NN. k and the distance function are the most important parameters that play a significant role in k-NN performance. A drawback of the K-NN is the computational expense when the number of independent features is high; however, the prediction gets better when the number of features increases.

Decision Tree: This is a powerful machine learning technique that uses a tree-based model. Data is repeatedly split based on a set of rules until the end (leaf). The branches represent conditions (features) and the leaf is the class label. The

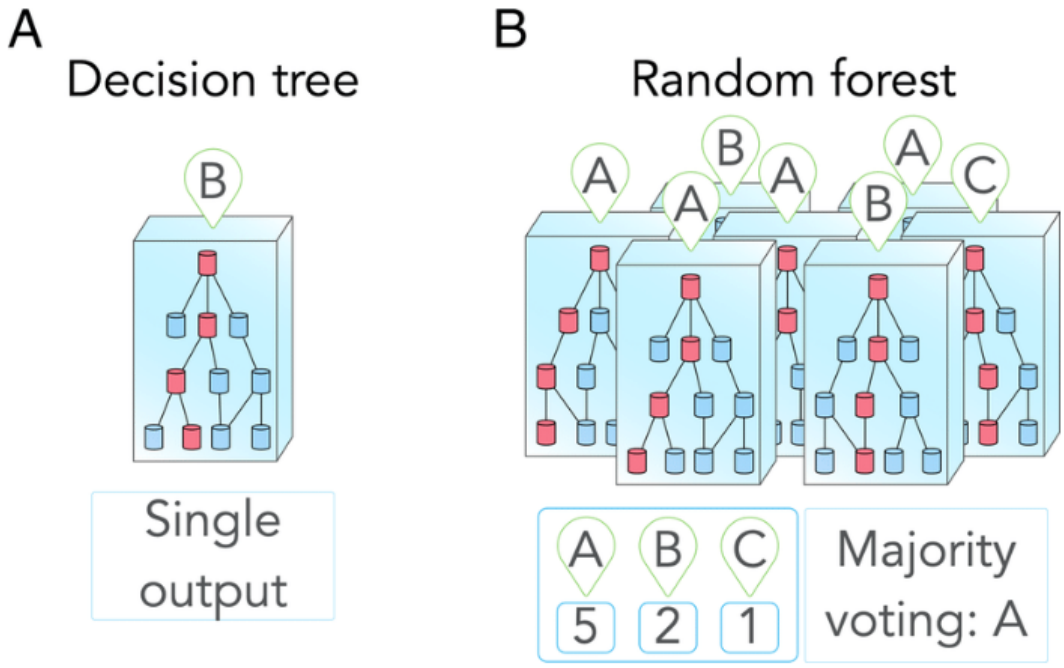

FIGURE 2.12: Difference between Decision Tree and Random Forest [4]

deeper the tree, the more accurate the decision, however, it may affect the generalisation of the model. The maximum depth is the parameter that represents the depth of the tree. Another important parameter is the 'criterion' which specifies the methods used to decide how the condition is constructed during split operation (in branches).

Random Forest: A Random Forest (RF) is an ensemble of Decision Trees. Every tree uses a subset of features for splitting. The decision is based on the outcome of the predictions of various decision trees. It is calculated by computing the means or average of the outcomes of participant trees. Random Forests overcome the limitations of Decision Trees by reducing overfitting and increasing the accuracy of decisions. Maximum features, maximum depth and the number of trees are the most popular parameters to be tuned when using Random Forest. The more trees, the more accurate the decision, however, this may require more computation and could affect the generalisation of the classifier. Figure 2.12 shows the differences between Decision Trees and Random Forests.

The eXtreme Gradient Boosting(XGBoost): XGBoost is a boosting ensemble method

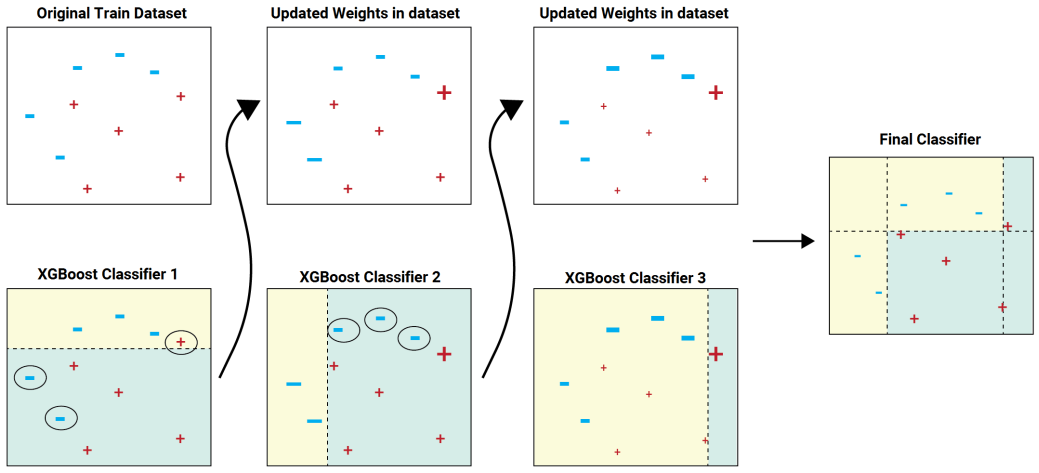

FIGURE 2.13: How XGBoost works [5]

where a sequence of classifiers are combined to provide more accurate decisions. Each model aims to correct wrong decisions of the model preceding it and provides input to the next model in the sequence [71]. Figure 2.13 illustrates how XBGoost works. The important parameters while using XGBoost are the maximum number of features to be used in a single run, the number of trees, and the maximum depth of every tree.

Support Vector Machine (SVM): The SVM is one of the strongest techniques for binary classification. It constructs a hyper-plane of N-dimensions between different instances based on their classes. The number of dimensions depends on the number of features. Choosing the optimal kernel, the type of regularisation and the kernel coefficient are the most important parameters to be considered while using SVMs.

2. Unsupervised-Learning: The model is trained over normal data to build a pattern for the behaviour of such traffic. In production, when a new instance of traffic has a different pattern than those trained on, it is flagged as malicious. Unsupervised learning is a well-known technique adopted by anomaly-based detection classifiers. One-class SVM, Isolation Forest and Local Outlier Factor are the most popular unsupervised-based anomaly/outlier detection technique.

One-class SVM (OC-SVM): An OC-SVM is an instance of SVMs used for anomaly

detection in unsupervised learning. The idea of classifying outliers is based on the majority of normal data being close to each other, and those far from them are anomalies. Rather than drawing a hyperplane, as SVM does, it uses a hypersphere to envelop most instances, with a few outliers remaining outside the created envelope. In the testing phase, when a new instance falls within the boundary of the hypersphere, it is classified as normal. Outliers are those that fall outside the created hypersphere (decision boundary). The kernel type used in the algorithm, gamma, and the percentage of outliers in the dataset are the most important parameters to be considered in One-class SVM [72].

Isolation Forest: A tree-based algorithm models normal examples by isolating outliers. These anomalies are typically few and have different characteristics of features. A random forest is generated randomly, and the features with thresholds are chosen randomly. It keeps isolating instances from each other. Anomalies are isolated very early as they are always far from other instances. The most important parameters to be tuned are the number of estimators (trees), max samples (the number of samples to be used in each tree for training), the percentage of anomalies in the dataset (called contamination), and the maximum number of features to be trained within every base estimator.

Local Outlier Factor (LOF): This calculates the local density of every point by measuring the distance between that point and its neighbour. Data points that have a significantly lower density compared to their neighbours are classified as anomalies.

3. Reinforcement Learning (RL): RL is a reward-based technique where the main idea is to improve the system based on the reward taken by an agent in an interactive environment.

Machine learning algorithms can be categorised based on their learning approach. Two types of techniques in this field:

1. Batch learning: Also called offline training, where the model is trained on the entire dataset at one time. It has the advantage of learning over a large amount

of data which improves the decision. The drawback of this model is that it does not adapt to network traffic changes. Such changes may be due to the natural evolution of the network architecture, or environment evolution, such as changes in attacker behaviour.

2. Incremental learning: In this case, the model does not require all data at the start to learn the model. Instead, it learns incrementally, adapting itself against changes in data distributions. Incremental learning has the advantage of learning over time with no need for updates by the operator.

2.6.2 Data preparation for machine learning

Data preparation (or preprocessing) is a vital phase in building machine learning. Figure 2.14 illustrates the phases involved in building a machine learning model. The collected data may have noises, errors or missing values which need to be dealt with. Some ML developers remove null data or replace it with 0s. Noise needs to be differentiated from anomalies. Thus exploring data helps to improve the performance of the model. Raw data comes from different sources have different structures. It needs to be transformed into a form that can be processed by ML techniques [73]. In machine learning categorical data needs to be transferred into numbers; however, these numbers have a different scale. To have a good model, it is important to transfer input variables into a specific scale. Normalisation and standardisation are the two techniques used to transform numerical data into a standard range. Normalisation is a technique to scale variables between 0 and 1. Standardisation is a transforming technique scale data into a standard Gaussian.

Feature engineering and selection is a vital process in the preparation of data. Feature selection is the process of variables that have an effect on the model. Delivering more features from the basic selected features is called feature engineering [74]. When data is prepared to be used by the ML model, optimal configuration gives a better model performance. Hyperparameter tuning is the process of finding the parameters for which the model gives its best performance. It should avoid over-fitting, where the model is tuned too tightly to the training data and performs well on it but does not perform well on new data. (The model does not generalise.)

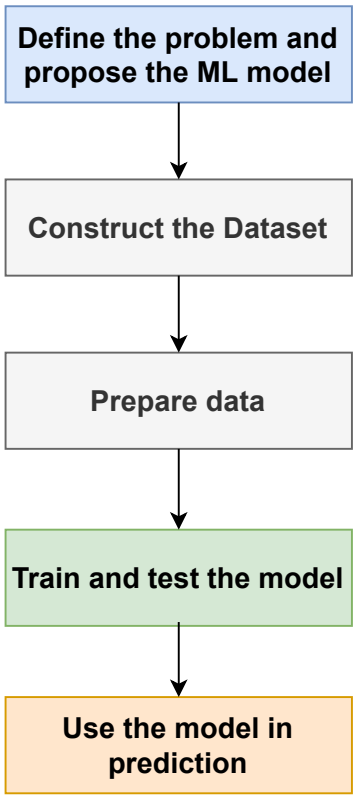

FIGURE 2.14: Building machine learning model phases

In addition, the developer seeks to avoid under-fitting, where the model performs poorly both on the training data and new data [70], because it fails to fully exploit information in both sets. Feature importance is a technique that analyses the available features and presents those that have more effect on the model [73; 74]. Principal Component Analysis (PCA) is used to synthesise new features. These new features are linear combinations of the current raw features and are information rich, i.e. they can inform classification more strongly than raw features. Such features do not exhibit the redundancy of the raw features and fewer of them are needed to effect high-performing classification. This also improves the amount of computation involved in classification [75].

2.7 Imbalanced Classification

Most machine learning classifiers assume the number of examples of each class is roughly equal. When the distribution of instances across the known classes is not equal, it is called an imbalanced classification problem. In some cases, one class

makes up a significant majority of the dataset and the other class is very rare. For example, one class has 95% of the entire data and the other one is just 5%. In a number of domains, the natural presence of one class is very often. For example, in stealth attacks, the majority of traffic is normal and just a few are malice [76]. The minority class, however, is more interested as the errors of classifying this class are more important to the user than those in the majority class. In Intrusion Detection Systems, when the attack is stealthy and forms the minority, classifying a normal sample as malicious (False Positive) is not as critical as misclassifying attacks as normal (False Negative). It is difficult to learn the characteristics of a class (or classes) with minorities as may no enough samples that give the classifier the ability to learn. Most machine learning algorithms require a modification to classify correctly and not classify all samples as the majority class. Selecting the most optimal evaluation metrics is essential when the dataset is imbalanced. Some metrics neglect the minority class effects.

Applying traditional algorithms and typical evaluation metrics can lead to poor performance of the model. An imbalanced classification requires different approaches than those applied to balanced datasets. This includes the preparation of the dataset, preprocessing of the classification techniques and the type of evaluation techniques used to measure the performance.

2.8 Concept Drift

The data distribution may change for a number of reasons, e.g. updating or replacing network devices, increasing or decreasing the number of users or devices in the network, or user behaviour changes. A classifier's performance may degrade if the data witnessed under training no longer adequately represents the current situation. The term concept drift is often used to describe this. Concept drift can be understood as changes in the relationship between the input and target outputs of a classifier [6].

The drift can be virtual or real. Virtual drift refers to a deviation in data distribution have no effect on the target prediction. The classification module will not be affected, and there is no need to update the module. A real concept drift is that

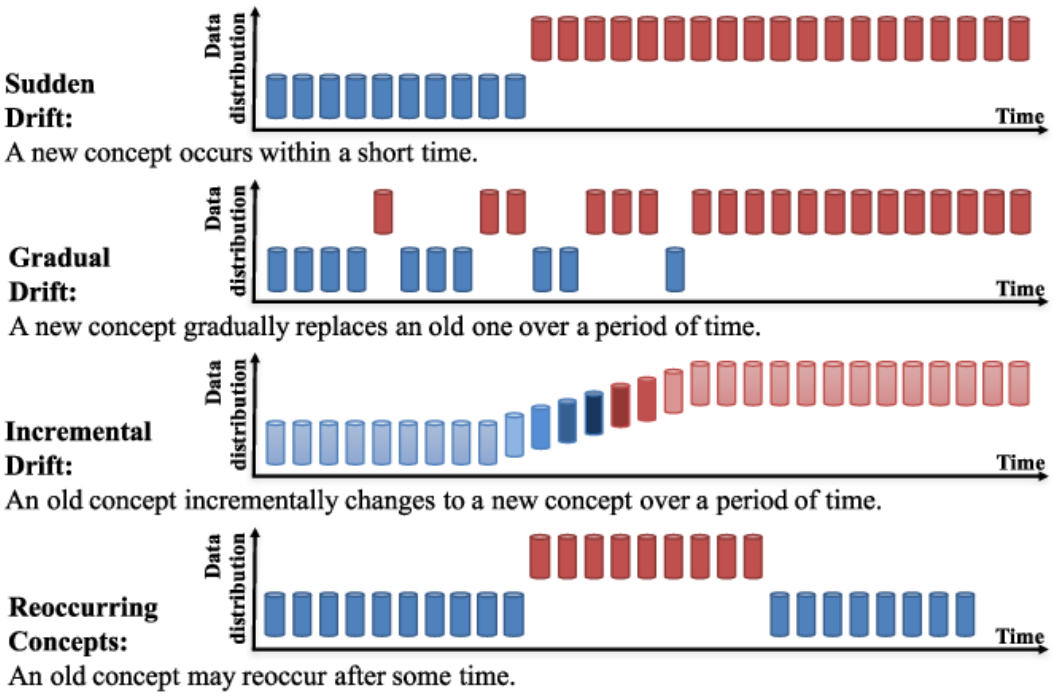

FIGURE 2.15: Concept drift types [6]

which makes a change in the target. This means that the classifier will lose its accuracy in classifying the network traffic. The model will need to be updated. Concept drift takes various forms (see Figure 2.15):

1. Sudden/abrupt: The drift happens suddenly at a specific time.
2. Incremental: The drift happens incrementally in a slow manner.
3. Gradual: The concept change duration is relatively large compared to sudden changes. Over time, the new concept gradually replaces the old one.
4. Reoccurring: This type is caused by, for example, seasonal events. A new concept occurs, but after some time, the old one occurs again.

2.8.1 Concept drift framework

The general framework of concept drift is composed of four stages: data retrieval - which retrieves enough data to help form a pattern that has a meaning; data modelling- which abstracts the gained data in the first stage and extracts the most important features; test statistics calculation - calculation of the dissimilarity; and;

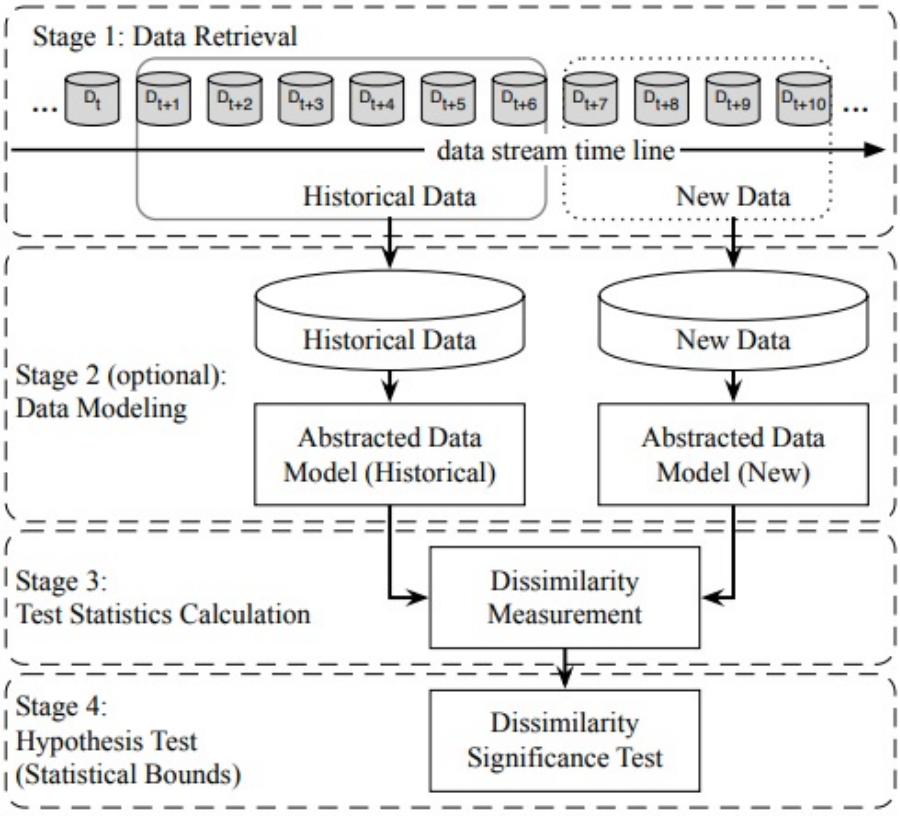

FIGURE 2.16: Concept drift stages [6]

hypothesis test: evaluate the statistical changes that are observed in the previous stage using a specific hypothesis test [6]. These stages are depicted in figure 2.16.

2.8.2 Concept drift detection techniques

In [6], the authors consider concept drift detection algorithms in three categories according to their implementation details: error rate-based drift detection, data distribution-based drift detection, and multiple hypothesis test drift detection.

2.8.2.1 Error rate-based drift detection

Techniques in this category calculate the error rates raised by the classifier. If the error rate is increased or decreased dramatically, a concept drift has happened, requiring the model to be adjusted. The most popular error rate-based drift detection algorithms are:

1. ADaptive WINdowing (ADWIN) [77]: This algorithm uses the technique of sliding windows. It calculates the statistics of every window. The window (W)

keeps sliding, and at some point, it is divided into two sub-windows (W_0, W_1). Then the distribution in every sub-window (W_0, W_1) is calculated. Concept drift is detected if there is a significant change in the average data distribution. The window size is variable as it can keep expanding and shrinking, making the sliding window mechanism more flexible than other error-based techniques. This makes ADWIN preferable, than other error-based techniques, by a number of researchers [77; 6].

2. Drift Detection Method (DDM) [78]: The technique calculates the prediction rate over the instances of incoming data. When the error rate exceeds a threshold (as drift happens), it starts using a new prediction learner. If it just reaches a warning level (warning zone), it warns the administrator and starts building a new learner while using the old one. DDM finds the detection of sudden drift challenging.

3. Early Drift Detection Method (EDDM) [79]: This works in a very similar manner to DDM. It overcomes the drawback in the detection of sudden drift in DDM. Rather than observing the error rate, it calculates the distance between two errors.

4. Hoeffding's bounds (HDDM_A and HDDM_W) [80]: Detection algorithms based on Hoeffding's bounds use moving averages. For the estimator, HDD_A uses the input average, and HDD_W uses Exponentially Weighted Moving Average (EWMA) statistics.

5. Kolmogorov-Smirnov Windowing (KSWIN) [81]: This is a detection algorithm based on the Kolmogorov-Smirnov (KS) statistical test. Using a sliding window, it compares the difference between empirical cumulative distributions over two windows.

6. Page-Hinkle [82]: The algorithm calculates the variance between the monitored values and their average until the present time. Drift is flagged when the change is significant. It can detect drifts but does not signal warnings.

2.8.2.2 Data distribution-based drift detection

In this type, the algorithms use distance functions to calculate the pattern changes between old and new data. Kdq-tree [83] is a data distribution-based detection technique that uses relative entropy, a statistical distance, that measures the distance between two sets of data distributions.

2.8.2.3 Multiple hypothesis test drift detection

In this category, multiple techniques (similar to those in the previous two categories) are applied. The detection algorithms apply different hypothesis tests. The multiple hypothesis test algorithms are applied in parallel or hierarchical ways.

2.8.3 Incremental learning model

In machine learning, the model is typically trained offline. All training data should be available at the time of training the model. The model can be deployed when the training is completed. The model can be trained later using a new batch of data. This is called offline (or batch) learning. Another type of learning is called incremental learning. This type does not require the whole data to be available initially. Instead, the data is processed sequentially, and the model updates itself while new data arrives [84].

2.9 Detecting attacks in SDN

Adopting anomaly detection techniques in SDNs is easier and more flexible than in traditional networks. For instance, in the traditional network, anomaly detection tools measure the network statistics by monitoring all network packets. However, in SDN these pieces of information can be retrieved easily from the data plane [85].

Packet-based detection and flow-based detection are the two methods that are used by IDS researchers. In a packet-based approach, the whole packet is analysed. In the flow-based method, just headers and similar information of packets are analysed. Combining the two methods has been suggested to overcome their drawbacks.

A reasonable number of studies leverage SDN to detect network attacks. The survey [86] has discussed different Machine Learning/Deep Learning algorithms

and their usage in network-based IDS (NIDS) over five studies. They found that using unsupervised learning algorithms for implementing NIDS in SDN is the best for previously unseen attacks, such as those often presented by APTs. They argued that most of the SDN-based NIDS are proposed for implementation on a SOHO (Small Office/ Home Office) network.

A flow-based Intrusion detection system is proposed [87] to detect attacks using an aggregator and a classifier system that gather statistical information from OpenFlow vSwitches and then compare it with predefined features using a supervised machine learning algorithm. The system works in parallel with the controller on the same layer. Therefore, there is no overhead on the controller and less communication is required. The system is misuse-based, and the classification model is updated offline.

The deployment of an Intrusion Prevention System (IPS) in conventional networks suffers from issues such as the high cost, being difficult to manage and low utilisation rate [88]. In the traditional network, the IPS is deployed in the ingress and egress points of data centres (DC). However, the authors in [88] proposed deploying only one IPS in each DC instead. The experiment results show that pinging the SDN network when not pre-configured consumes more than three times the resources required for the traditional network, but when the network is pre-configured it takes less for the SDN architecture.

In [89], a signature-based IDS is integrated with a flow-based neural network IDS. It is proposed to detect anomaly behaviour using a machine learning algorithm. A backpropagation algorithm has been used to train the network in a flow-based IDS model. In [90] a REST client communicates with the controller to get statistical flows that are stored for further classification using self-organized maps of Artificial Neural Networks. This architecture uses a star topology with a single switch. Moreover, the proposed method has performance drawbacks. In addition, using the measuring module on the controller causes a performance problem. Atlas [91] is a fine-grained mobile application detection using a machine learning (ML) algorithm for classification to collect information about active mobile applications that are running on the SDN network. However, the system is not proposed to detect malicious activities. An Intrusion Detection and Prevention System (IDPS) has been proposed that uses

SDN to detect and mitigate ARP spoofing and blacklisted MAC address attacks [92].

A reasonable number of research proposals have sought to detect unknown attacks in Software-Defined Networks using anomaly-based detection systems. A deep learning approach is proposed to detect unknown attacks [93]. The proposed model is evaluated over NSL-KDD, scoring 75% in accuracy. Stacked Auto-Encoders have been proposed to detect attacks in SDN [94]. The detection results (in accuracy) are 98% over NSL-KDD and 94% on CICIDS 2017. Various other approaches to detect attacks in SDN have been proposed and compared [95]. The decision Tree was found to be the best classifier among the nine techniques. An IDS is proposed to detect online DDoS and port scans using clustering techniques [96]. Several researchers utilise SDN to detect DoS and DDoS attacks [97; 98; 99]. Low-rate DoS attack is a type of DoS attack with low traffic volume, typically seeking only to keep services at or near 100% utilisation (enough to deny access to clients), rather than overwhelming such services with requests (the usual DoS approach). Using SDN, a number of researchers proposed solutions for this attack [100; 101].

A mean to detect a compromised switch in SDN is proposed in [102]. Any switch processing a packet in a way that does not follow the programmed rules is a compromised switch. SDN-RDCD [103] is a system proposed that implements extra controllers as a backup. They collect information from the main controller and SDN switches to detect compromised devices by analysing and detecting inconsistent behaviour through the main controller, backup controllers, and SDN switches.

2.9.1 Detecting APT in SDN environments

APTs have the characteristic of staying and moving from one device to another inside the victim network. Thus, the malware does stealthy scanning activities and moves slowly from one machine to another. The global view of the SDN controller can lead to detecting this type of attack more easily than in the traditional network. The SDN controller can react to any attack immediately as a result of the ability to manipulate each flow forwarded from network devices. Compromising the SDN controller can lead to easier spread within connected devices.

The literature focusing on APTs is limited despite APTs' obvious importance

[104]. Therefore, as the SDN is also still immature, little research has been conducted on detecting or preventing APTs in SDNs. MARS [105] is a proposed system intended to reconfigure the environment based on malware behaviour. Although the paper does not discuss APTs, they evaluated the system using 50 APT malware samples. An improvement in malware analysis can be shown in the results. A framework [106] was proposed to enhance the security in SDN-based data centres. The detection mechanism is based on collecting different logs from network devices besides monitoring traffic using signature-based techniques to detect APT patterns. However, there are no details about APTs and the detection mechanism does not consider unknown attacks. A hierarchical security framework [107] which benefited from SDN and NFV [108] is proposed to detect two types of APT, dynamic ongoing threats and unknown attacks, in WSN in smart cities. Chance discovery [109] and usage control (UCON) are the two technologies used in detection. In 2016, [110] claimed that by that time there were no evaluations or studies of stepping-stone attacks in the context of the software-defined network. However, they propose an architecture by adopting some stepping-stone detection techniques to detect this type of attack in SDN. Then an evaluation of different network topologies was carried out. However, the detection rate is decreased when the network size is large. PivotWall [111] is a security architecture to track information flow to detect stepping-stone attacks in APT using both the centralisation of the SDN controller and host-based information tracking. However, installing a customised kernel in every host is essential. In addition, it is only feasible for small networks.

A method to detect APTs in SDNs based on Hidden Markov Models (HMMs) is proposed in [112]. They use the relationship between the attacker's behaviour and the APT stage. However, the attack stages and attacker behaviours described there are the same as for a normal attack. A deception technique based on redirecting the traffic from the compromised operational network to a deception network is proposed in [113]. Although the main objective of this architecture, which uses SDN, is to observe APT attacks, there are no details about the APTs.

In the context of scanning SDN, some studies have been proposed to prevent fingerprinting attacks. In [114], a moving target defence (MTD) technique is used by randomising both the host IP and time to invalidate attacker reconnaissance. The

mutation in the randomisation of the source identity and time gives adversaries a false view of the system. Using a similar strategy, the same authors published another study [115] to slow down the process of scanning attacks in SDN by giving wrong information to the attacker while doing their scan. The idea is based on the IPs randomisation by assigning epheronal IPs to the original ones. In spite of the system being proposed to prevent stealth scanning, it is not proposed for APTs. In addition, there is no description of how to detect scanning activities. On top of that, there is a delay in the system during processing packets caused by the mechanism of IPs randomisation. Other work using MTD is proposed in [116] to make it difficult for the attacker to fingerprint the OS of hosts in SDN. However, this work is not proposed for APTs. The Reconnaissance Deception System (RDS) [117] is proposed to delay scanning activities that could be carried out by sophisticated attackers such as APTs. The system maps the network features to other features, therefore, the attacker misinforms the actual network features. A honeypot is employed to identify the compromised machine. The controller always checks the flow rules for all machines. If there is any transmission from one node to the honeypot that means this node is malicious. However, a honeypot is expensive and few systems deploy them. Also, sophisticated attackers usually can detect whether a honeypot exists in the system or not [118].

To detect APT C&C communications, [119] proposes to analyse traffic to identify malicious DNS. In their work, they identified 14 characteristics of malicious APT DNS and their ways of communication. The study in [120] found that APT C&C communications are accessing DNS records independently. On the other hand, accessing legal web domains is correlated. However, the previous two studies are not proposed or tested on SDNs.

2.10 Datasets

Intrusion detection systems use datasets to train and evaluate proposed models. There is a reasonable number of datasets in the IDS literature. They differ in structure and type. The majority are flow-based and include the metadata of the transmitted packets in addition to historical-based engineered features. These metadata

are connection-based information such as source IP, destination IP, protocol and port numbers. Other datasets are packet-based, usually pcap files that include metadata and payload. A small number of datasets are collections of files such as log files and malware hashes. The following shows the most popular used datasets in the field of IDS:

DARPA: The dataset is a collection of files, collected in 1999, including payloads, system events and log files. The dataset contains traces of various significant categories of attack: Denial of Service (DoS), User to Remote (U2R), Remote to Local (R2L), and probe attacks.

DEFCON [121]: This dataset was generated in 2000 and contains port scan and buffer overflow attacks. It is in packet format, generated during a capture the flag (CTF) competition at a DEFCON conference. It is made up almost entirely of attacks with few packets arising due to normal user behaviour.

KDD'99 [122]: KDD Cup 99 is a well-known benchmark dataset in the IDS research community. The dataset was created in 1999 by researchers at MIT. It comprises DARPA packet traces. The dataset has 41 features describing the basic content and traffic information. In addition to the normal traffic, it has four categories of attacks: DoS, R2L, U2R and probing attacks.

NSL-KDD [123]: This is an improved version of the KDD dataset. The NSL-KDD was produced to overcome a number of issues in the KDD dataset such as duplicate records. In spite of the wide use of KDD'99 and NSL-KDD between IDS researchers, its age means that it no longer represents current network architectures and attacks. This issue also applies to the DARPA dataset.

UNSW-NB15 [124]: This is a network-based dataset generated using simulation. The authors used Argus, Bro-IDs and additional scripts to generate 49 features. Different categories of attacks are involved including fussing, port scan, spam, backdoors, DoS, shellcode, and worms.

CICIDS 2017 [125]: This is one of the most used datasets in recent years in IDS work. It has various attacks such as probe, DoS, DDoS, Bot, Heart-bleed, Infiltration, Brute force, and Web attacks. It contains pcap files and more than 80 features generated using CICFlowMeter [126].

TABLE 2.8: CICIDS 2017 dataset attacks scenario

day	Benign	attack category	attack type	attack time
1	✓	×	-	-
2	✓	Brute Force	FTP-Patator SSH-Patator	9:20 – 10:20 a.m. 14:00 – 15:00 p.m.
3	✓	DoS / DDoS Heartbleed	DoS slowloris DoS Slowhttptest DoS Hulk DoS GoldenEye Heartbleed Port 444	9:47 – 10:10 a.m. 10:14 – 10:35 a.m. 10:43 – 11 a.m. 11:10 – 11:23 a.m. 15:12 - 15:32
4	✓	Web Attack Infiltration – Dropbox download Infiltration – Cool disk Infiltration – Dropbox download	Brute Force XSS Sql Injection Meta exploit Win Vista MAC Win Vista	9:20 – 10 a.m. 10:15 – 10:35 a.m. 10:40 – 10:42 a.m. 14:19 & 14:20-14:21 p.m. 14:33 -14:35 14:53 p.m. – 15:00 p.m 15:04 – 15:45 p.m.
5	✓	Botnet ARES probe DDoS	 port scan LOIT	10:02 a.m. – 11:02 a.m. 13:55 - 14:35 14:51 - 15:29 15:56 – 16:16

InSDN [127]: This SDN-based dataset, published in 2020, contains various attacks: DoS, DDoS, web attacks (XSS, and SQL injection), R2L, Malware (botnet), probe and U2R (exploitation) attacks. More than 80 features are generated using CICFlowMeter. The dataset is imbalanced, with more than 80% of the dataset being attacks. Its limitations are similar to those in CICIDS, caused by the use of CICFlowMeter.

DAPT 2020 [128]: The DAPT 2020 dataset contains various attacks and spans APT stages. It contains network traces (attack and normal) in addition to system log files. It is conducted over five days. Examination of the 85 features available in the dataset suggests it was generated using the tool CICFlowMeter. Table 2.9 shows details of attack scenario.

In general, except DAPT 2020, these generic datasets lack sophisticated attacks. Also, most available datasets are imbalanced. For example, malicious traffic makes up more than 80% of the InSDN dataset [128].

In our work, we require a network-based dataset that includes network traffic of

TABLE 2.9: DAPT 2020 dataset attacks scenario

day	Benign	attack stage	attack type	attack time
1	✓	×	-	-
2	✓	Reconnaissance	scanning	8:00 a.m. - 6:00 p.m.
3	✓	Foothold establishment	Sqli, XSS, authentication bypass	8:00 a.m. - 6:00 p.m.
4	✓	Lateral Movement	network scan (insider), authentication bypass, SQLi	8:00 a.m. - 6:00 p.m.
5	✓	Data exfiltration	Exfiltrating data to C&C	8:00 a.m. - 6:00 p.m.

TABLE 2.10: State-of-the-art datasets

Name	Year	Number of features	Attacks
DARPA	1999	41	DoS, Probe, R2L and U2R
KDD'99	1999	41	DoS, Probe, R2L and U2R
NSL-KDD	1999	41	DoS, Probe, R2L and U2R
Defcon	2000	-	port scan, and buffer overflow
UNSW-NB15	2015	49	DoS, fussing, port scan, spam, backdoors, reconnaissance, shell code and worms
CICIDS 2017	2017	80	Dos, DDoS, Brute force, port scan, botnet, web infiltration
InSDN	2020	80	DoS, DDoS, web attacks, botnet, probe, R2L, and U2R
DAPT 2020	2020	85	Web scan, port scan, account discovery, brute force, web attacks, backdoor, data exfiltration

stealth attacks (or APTs) against an SDN network. This is the main reason we have chosen to generate our own datasets. See section 3.2.2 for more details.

2.11 Evaluation Metrics

Evaluation criteria for a classifier are predominantly concerned with whether it classifies instances from each class correctly or incorrectly. (The classifier may be more successful in some classes than others.) In our (binary) case we are concerned with whether instances are malign or benign (normal). It is common to refer to malicious instances as 'positives' and benign or normal instances as 'negatives'. The four *fundamental* measures of performance are:

		Predicted	
		Negative	Positive
Actual	Negative	True Negatives (TN)	False Positives (FP)
	Positive	False Negatives (FN)	True Positives (TP)

FIGURE 2.17: Confusion Matrix

- **True Positive (TP)**: the number of instances of malicious transactions classified correctly (as malicious).
- **True Negative (TN)**: the number of instances of normal transactions classified correctly (as normal).
- **False Positive (FP)**: the number of instances of normal transactions classified incorrectly (as malicious).
- **False Negative (FN)**: the number of instances of malicious transactions classified incorrectly (as normal).

A *confusion matrix* is a matrix of numbers showing how many correct and incorrect predictions (i.e. true positives, false positives, true negatives and false negatives). It is very useful in imbalanced classification tasks as the user can see the actual number of correct and incorrect decisions by class, not just overall counts. From the confusion matrix, other important metrics can be calculated such as accuracy, recall, precision and F1-score. Figure 2.17 illustrates the structure of a confusion matrix.

A large number of evaluation metrics are used by machine learning researchers. The researcher selects the most appropriate metric based on the context (dataset) or model used. For imbalanced classification, [129] divided the evaluation metrics into threshold metrics, ranking metrics and probabilistic metrics. These are explained below.

2.11.1 Threshold metrics

These are metrics that calculate errors in the prediction. They represent an overview of the prediction rate when the predicted value is wrong. They are useful when the user is focusing on the errors in the model prediction. They are the most widely used in the evaluation of Intrusion Detection Systems. The score is a value between 0.0 to 1.0. The following are the most popular metrics in this category.

- *Accuracy*: This is the fraction of all predictions that are correct. *Accuracy = Number of correct predictions / Total number of predictions*, defined formally by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

- *Recall*: This is the fraction of actual attacks that are correctly detected as attacks, defined formally by:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

- *Precision*: This is the fraction of all predicted attacks that are genuine attacks, defined formally by:

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

- *F-measure*: The F1-score is a balance of Precision and Recall. It is the harmonic mean of Precision and Recall, defined formally by:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.5}$$

2.11.2 Ranking metrics

These metrics primarily focus on differentiating between the classes. They are useful when the user is focusing on how one class is identified correctly or wrongly. The score of the prediction, indicating whether an item belongs to that class, is a key aspect in this regard. The following are examples of widely used ranking metrics:

- *True Positive Rate (TPR)*: This is also called *Sensitivity* or *Recall*. It is the fraction of positive instances that are correctly predicted as positive, defined formally by:

$$TPR = \frac{TP}{TP + FN} \tag{2.6}$$

- *False Positive Rate (FPR):* This is the fraction of negative instances that are incorrectly predicted as positive, defined formally by:

$$FPR = \frac{FP}{FP + TN} \tag{2.7}$$

- *False Negative Rate (FNR):* This is the fraction of positive instances that are incorrectly predicted as negatives, defined formally by:

$$FNR = \frac{FN}{TP + FN} \tag{2.8}$$

- *True Negative Rate (TNR):* This is the fraction of negative instances that are correctly predicted as negative, defined formally by:

$$TNR = \frac{TN}{TN + FP} \tag{2.9}$$

- *A Receiver Operator Characteristic (ROC) curve:* ROC curve is a graphical plot that represents trade-offs between the true positive rate (TPR) and the false positive rate (FPR). Different parametrisations of machine learning approaches may produce different trade-offs.

  The RoC curve connects established trade-off points. Figure 2.18 shows how the regions around the RoC curve can be interpreted. The dotted diagonal line shows that the model is predicting the majority class under all thresholds which would show that no skill is provided by the model. If the curve goes under this line, it means that the technique performs worse than one exhibiting no skill. A point at the top left indicates a perfect model.

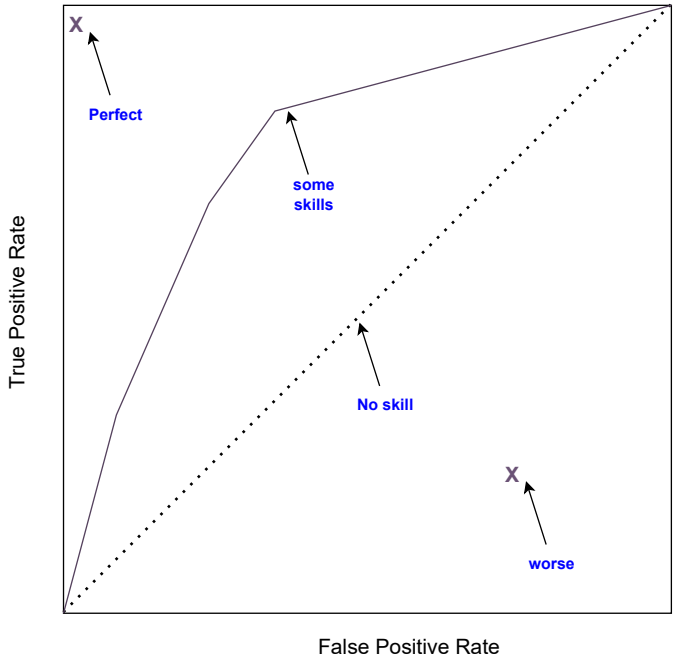

FIGURE 2.18: ROC curve

The ROC area under the curve (ROC AUC) is a score (between 0.0 and 1.0) describing all threshold values of the classifier. It is helpful to overcome the drawbacks of ROC in the ability to compare different classifiers based on their curves.

- *Precision-Recall Curve:* This metric represents a score as a result of the calculation of the area under the curve, which can be used to compare with another classifier. Figure 2.19 shows the details of the Precision-Recall Curve. As mentioned earlier, Precision and Recall focus on the minority class, which is of more concern in imbalanced datasets and stealth attacks. That makes the Precision-Recall curve more useful.

2.11.3 Probabilistic metrics

This type of metric focuses on the assessment of the reliability of the classifier. They are not concerned about the values of correct and incorrect predictions. They illustrate the uncertainty of a prediction by a classifier. Log loss and Brier score are examples of the most popular tools in this group.

Choosing proper evaluation metrics is essential in machine learning. For example, in imbalanced datasets, similar to those used in this thesis, *Accuracy*, one of the

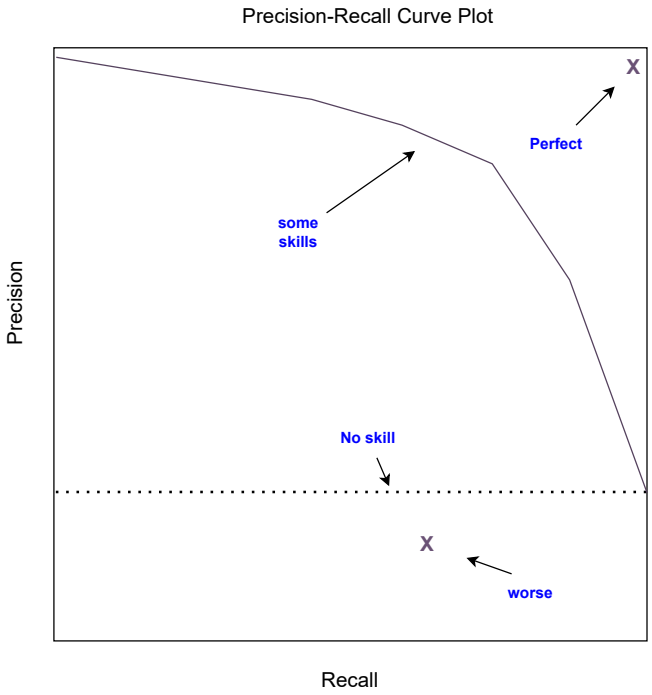

FIGURE 2.19: Precision-Recall Curve

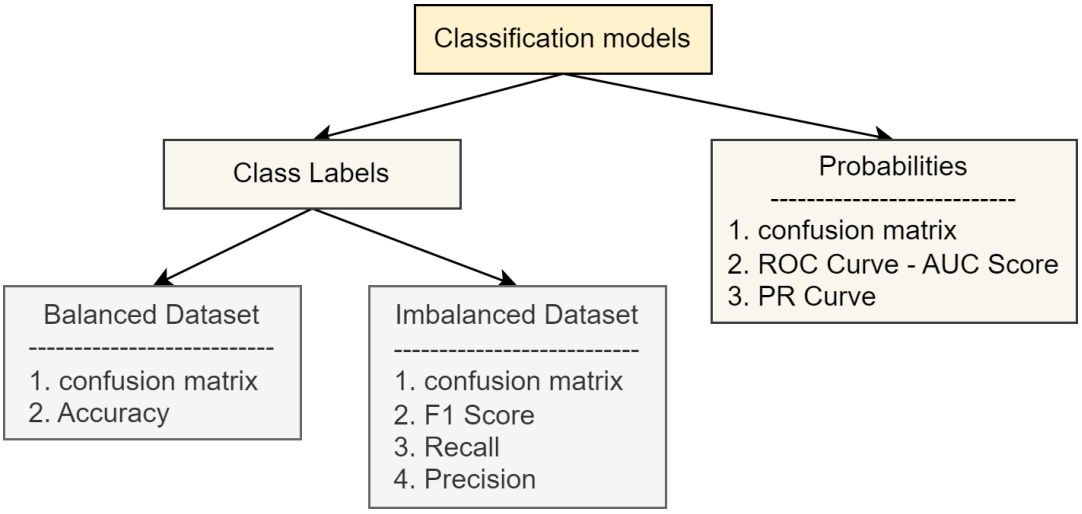

FIGURE 2.20: Machine Learning evaluation metrics

most used metrics in ML, may give a high score even when classifying all instances of the minor class incorrectly. For example, if the dataset contains 95% of normal traffic and just 5% as malicious, then if the model classifies all data (normal and malicious) as the major class (normal) it will give 0.95 accuracy (because the portion of True Negative (TN) is very high). This may deceive the user into believing that the classifier is performing very well [130]. *Recall, Precision* and *F1-measure* are alternatives to accuracy. They mainly focus on positives (which always present the minority in stealth attacks) [76]. Figure 2.20 presents a categorisation of machine learning evaluation metrics. In this thesis, we focus on *Recall, Precision* and *F1-measure*, however, *Accuracy* results are presented to allow comparison with existing benchmarks.

2.12 Proposed Area of Research

The literature review has revealed that 'flow rule reconstruction' attacks are a major current problem for SDNs. The literature review also revealed that very little research has been carried out on the topic of Advanced Persistent Threats (APTs) in the context of SDN. APTs are acknowledged as difficult to detect, typically adopting a 'low and slow' strategy. This suggests that current security issues could practically become much worse if the adversary incorporated APT aspects into their operations. Such stealth might significantly affect the security of SDNs and hamper their further uptake. Accordingly, we propose to investigate how an important means of attack on SDNs (flow rule reconstruction) can be made more stealthy and investigate how machine learning can be brought to bear for the detection of such attacks.

We propose first to use signature-based detection (Chapter 3) to establish a baseline. This will use a supervised learning approach. We will then extend our approach to incorporate anomaly-based detection, i.e. to develop a hybrid IDS (Chapter 4). As far as we are aware this is the first such hybrid IDS for stealth attacks in SDN. This provides prospects of being able to detect both known and unseen attacks.

Furthermore, current research on intrusion detection for SDN assumes a static adversary. But we know that attacker behaviours change over time. Indeed, for APTs such adaptivity is very much a deliberate strategy. Accordingly, we propose to investigate how the concept drift occasioned by changes in attacker behaviour may

be handled (Chapter 5). We know of no investigation of concept drift in the context of SDN intrusion detection.

Analysis of the literature reveals a significant opportunity to adopt a comprehensive ML pipeline in our work. Thus, we intend to adopt aspects such as feature engineering (seeking fewer and higher performing features) and hyper-parameter optimisation (getting the best out of adopted ML techniques via judicious parameter choices). This is driven by a desire to enhance performance but also to encourage fair future comparisons by other researchers. The current literature on IDS often ignores these, choosing to operate on raw features only, and not being fully transparent about the rational for parameter choices.

Stealthy flow rule reconstruction is an exemplary stealth attack. The incorporation of stealth in other aspects of malicious operations is clearly feasible. Our proposed programme of investigation offers immediate insight into how one current attack (i.e. flow rule reconstruction) can be 'stealthified' and how the detection challenges this poses can be met using ML. It also demonstrates the applicability of our proposed approach to a wider set of attacks.

Our work represents a focused, original, and important contribution to a little researched area. Along the way, the lack of available datasets for experimentation targeting APTs and SDN forces us to create our own and allows us to make available such resources for the research community.

The above provides the rationale for the specific research questions presented in section 1.2.

2.13 Summary

We have surveyed literature relevant to the proposed research, providing background and covering research in SDN, APTs, IDS, concept drift and ML. Open and important research questions have been identified. The next chapter begins to investigate the first of those questions.

Chapter 3

Enhanced SDN Scanning and its Detection using Machine Learning

In this chapter, an existing scanning tool is enhanced to demonstrate an increased level of stealth in scans targeting flow rule reconstruction. The tool has been made publicly available. A benchmark dataset is produced that includes APT attacks over exemplar SDN networks. It has also been made publicly available. Intrusion detectors based on common ML techniques, and using supervised learning, are evaluated.

This chapter investigates the first research question as stated in section 1.2. RQ1: *Can we use a machine-learning and signature-based approach to detect the stealthy reconstruction of flow rules in SDN networks?*

3.1 Introduction

In SDN, scanning is a fundamental part of the reconstruction of flow rules maintained at nodes and underpins many further attacks. It is more challenging when the attacker behaves more *stealthily*. This chapter presents a proposal for an enhanced approach to the reconstruction of flow rules within SDN network switches using *stealthy scanning regimes* and investigates the detection of the use of such regimes.

The SDNmap tool [34] is optimised to emulate stealthier scanning attacks with 'slow and low' movements. Such enhancements include extended waiting times between different activities with random values. In addition, there is an option to limit the number of scanned ports to the most common ports for attackers. These enhancements give rise to the free and open-source tool that we henceforth refer to as

APT-SDNmap [131]. APT-SDNmap now adopts a 'slow and low' scanning strategy which makes the detection of scans more difficult. A dataset containing APT activities is generated using Mininet [132] and APT-SDNmap. The dataset has been made publicly available and free to use [133]. The most popular supervised machine learning algorithms are applied to the detection of stealthy scanning attacks, establishing benchmarks for future research. Building the ML models includes comprehensive feature engineering and hyper-parameter optimisation to give the best chances of excellent results. A network-based Intrusion Detection System (NIDS) implementing XGBoost is shown to give the best detection performance on such attacks. The proposed detection model achieves at least 97.8% in Accuracy, Recall, Precision and F1 measures, using just five features.

3.1.1 Motivation

SDN is increasingly used in a large number of organisations. The decoupling of the control plane and the data plane raises a number of security challenges. Reconstructing flow rules in SDN switches is one of the main concerns of the SDN community and a vibrant research area. When the attacker succeeds in reconstructing the flow rules, some attacks can be launched, such as data leaks, network poisoning and DoS attacks.

Statistics show the adoption of APTs (Advanced Persistent Threats) in attacks has increased [134; 135]. The stealthiness and sophistication of APTs make them far more dangerous and difficult to detect than traditional threats. Little research has been carried out on detecting APTs in the context of SDNs (discussed in 2.9.1). Reconstructing flow rules in SDN using the strategies and behaviour of APTs is a significant attack whose detection is an important research area.

3.1.2 Contributions

The contributions in this chapter are:

1. The proposal of a stealthier scanning approach that adopts waiting times between various activities of an APT.

2. Adapting a scanning tool to incorporate the stealthier scanning approach so that it can be applied by APTs to reconstruct flow rules on an SDN. To the best of our knowledge, there is no available flow rule reconstruction tool to scan an SDN using APT behaviour.

3. Development of a publicly available dataset that includes APT activities in an SDN. To the best of our knowledge, this is the first publicly available dataset for APTs in an SDN.

4. A comparison of the most common supervised machine learning techniques. We indicate how to deal with an imbalanced dataset and use importance measures to inform feature reduction. The XGBoost model can detect APT scans in SDN with 0.97 in Accuracy, Recall, Precision and F1-score, and uses just a small number of features ($f = 5$)

3.1.3 Chapter organisation

The rest of this chapter is organised as follows. Section 3.2 presents related works. Section 3.3 shows the APT temporal behaviour. The proposed enhanced scanning is described in section 3.4. Section 3.5 gives a full description of the generated dataset. Section 3.6 presents the proposed hybrid model and the results of our experiments using ML to identify scanning attacks. Finally, section 3.7 provides a summary of the chapter.

3.2 Related Works

The following sub-sections discuss techniques for reconstructing flow rules, APT datasets in SDN, and the detection of APT scanning attacks.

3.2.1 Reconstructing flow rules techniques

DELTA [136] is a security assessment framework for SDN. It allows the user to establish different attack scenarios to assess the target. However, it does not consider APTs or stealth attacks. A side channel attack is used in some works to gather information about network configuration, e.g. security policies [137; 138]. Further side

TABLE 3.1: Reconstructing flow rules techniques comparison

tool	SDN	Reconstructing flow rule	Stealth	APT	Information
[139; 141; 140]	✓	✓	×	×	◔
[136]	✓	✓	×	×	●
[137]	✓	✓	✓	×	◓
[8]	✓	✓	✓	×	●
APT-SDNmap	✓	✓	✓	✓	●

[*] ◔ :few, ◓ : average, ● : high

channel attacks are proposed [139; 140] to gather information about the flow table, such as size and policy. This information would be useful for launching DoS attacks. In [141], the authors implement timing techniques to illustrate the ability of remote adversaries to determine whether a flow rule has already been installed in the host (by the controller). It does not consider stealth or APT attack scenarios. SDNmap is an open-source scanning tool for reconstructing flow rules in an SDN network. It generates, on average, at most one probe packet every second to evade new SDN defence techniques [8]. APTs are, however, usually even stealthier, as they wait for a longer time during their communications. Table 3.1 compares the aforementioned techniques and our approach, showing which techniques employ stealthy behaviour or APT characteristics and the amount of gained information from the target.

3.2.2 APT datasets in SDN

There is a reasonable number of datasets, discussed in section 2.10, available to evaluate intrusion detection systems against normal types of attacks on traditional networks. None targets stealth attacks over SDN. None contains data reflecting the reconstructing of flow rules in SDN. The only dataset conducted over an SDN is InSDN but its attack traffic greatly exceeds its normal traffic: attack traffic forms more than 80% of all traffic. DAPT 2020 is a dataset presenting APTs but is not collected over an SDN.

The lack of APT datasets has led researchers to collect data from different resources to build a dataset that represents APT attacks. For example, MARS [105] used 50 samples of APTs collected from the publicly available malware repository VirusShare [142]. Another example is [119], where the authors collected domains from malwaredomains.com [143], snort rule sets [144], and Targeted Cyberattacks

TABLE 3.2: Datasets comparison

DS	Year	Number of features	Generating type	SDN	Recon.	APT
KDD [122]	1999	41	Hybrid	×	×	×
UNSW-NB15 [124]	2015	49	Hybrid	×	×	×
UNB ISCX 2012 [151]	2010	20	Realistic	×	×	×
CICIDS2017 [152]	2017	80	Hybrid	×	×	×
ADFA-LD12 [153]	2013	49	Hybrid	×	×	×
InSDN [127]	2020	83	Hybrid	✓	×	×
DAPT 2020 [128]	2020	85	Hybrid	×	×	✓
Our dataset [133]	2022	25	Hybrid	✓	✓	✓

Recon.: Reconstruct flow rules

Logbook by Kaspersky [145]. The comparative study in [146] also used a collection of data gathered from different sources. They collected malware hashes from some reports by security companies such as Kaspersky and FireEye. Then they used VirusTotal [147] Private API to search for these hashes and download 1037 APT malware samples. Similarly, [148] built their dataset by collecting hashes of actual APT attacks from APT & CyberCriminal Campaign Collection [149] and the blog in [150]. Table 3.2 presents a comparison between the available datasets and the dataset generated in this chapter.

3.2.3 Detecting APTs in SDN using ML

Chapter 2 identifies a reasonable number of research works using SDN to detect malicious activities. A proactive approach is proposed to detect reconnaissance attacks by randomising network addresses [115]. A high number of failures caused by an adversary implies there is suspicious activity. An Intrusion Prevention System (IPS) [154] is proposed to prevent port scans using statistics collected from OpenFlow switches. A Deep Learning Neural Network (DNN) technique is applied to detect anomalies in an SDN using an anomaly-based approach and tested over the KDD dataset [93]. The detection exhibits only 75.75% accuracy. Further work by the same authors improves on this to achieve 89% accuracy using a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) [155]. The KDD dataset is legacy and includes no SDN traffic. Reconnaissance deception systems [117] [113] are proposed to defend against SDN scanning by mapping real network features to virtual ones,

TABLE 3.3: Related works: proposed techniques to detect APTs in SDN

scheme	SDN	Scan	Stealth	APT	ML	technique
[115]	✓	×	✓	×	×	statistics
[154]	✓	×	×	×	×	statistics
[93] [155]	✓	×	×	×	✓	Neural Network
[117]	✓	✓	×	✓	×	statistics
[113]	✓	×	✓	✓	×	Deception System
Our technique	✓	✓	✓	✓	✓	various ML tools

which could delay or prevent the targeted attacks. Employing extra components, however, could cause overheads in the system. Table 3.3 compares the capabilities of the approaches used in the above works with those of our proposed method.

3.3 APT temporal behaviour

Stealth malware differs from other malware by engaging in fewer interactions inside the victim network [156]. Scanning with a high rate can typically be detected by defence systems such as Intrusion Detection Systems (IDSs). It is accepted that one scan per minute could avoid most network detection techniques [157]. Therefore, stealth malware, specially APTs, reduce their visibility in the network by keeping movement low and applying longer times between their activities (i.e. using the 'slow and low' strategy) [158] [2].

One automated analysis technique to detect malicious software is analysing the behaviour of the software for a certain amount of time after its installation. If the results over that period of time are normal, then it is considered to be benign. Mostly, remaining idle for 10 minutes is sufficient for most sandboxes to conclude the software is not malicious [159; 146]. The analysis shows that a number of APTs, such as Duqu [39], Kilihos [160], and Nap [161], stay dormant for a certain period of time after installation, typically ten minutes, to avoid these detection techniques.

A large number of APTs employ waiting times between their activities inside a victim network. They may go to sleep for approximately two minutes between their activities. Flame is a complex APT with several components. In the startup stage, the main module mssecmgr.ocx is loaded. Then it waits two minutes before loading

another module called advnetcfg.ocx by services.exe. This action repeats three times every two to three minutes. After a further two minutes, the same service loads another module called nteps32.ocx. Then, a few components and processes are loaded and created. After waiting for two minutes, the services.exe writes ccalc32.sys which is loaded after one minute by winlogon.exe. It continues to employ waiting times for other activities, such as running a task every two minutes to retrieve data from the compromised machine [38] [162].

A similar strategy is adopted with Shamoon 2.0. It has an executable worm, called ntssrvr32.exe, that spreads inside the victim network. While this worm moves, it connects to a remote machine registry. When the remote system32 folder is found, it schedules a remote job to be run after one and a half minutes. Interestingly, after copying itself inside the network, it attempts to run a command and control module. The malware creates a Windows Task Scheduler job and waits around 90 seconds before executing the command and control module. It then waits for about 95 seconds before deleting the scheduled job. Moreover, when all tasks are completed, the system waits for two minutes before rebooting [163] [164].

Hydraq is a trojan used in the Operation Aurora cyber attack campaign. When the malware successfully installs, it starts to contact the command and control server using the configuration inside the backdoor by retrieving the hostname and alternate DNS. It checks if the IP address is valid or retrieves the hostname IP address using an available DNS. If the backdoor fails to resolve the hostname IP address, it sleeps for two minutes before attempting to resolve the IP address using an available DNS [165]. Pegasus, the OSN group spyware, is another example which waits for a few minutes between particular interactions [166].

In fact, waiting a short time such as 10 seconds is enough for some interactions. For example, Nap employs a 10 second waiting time between multiple attempts to resolve domains during their C&C communications. StoneDrill [163] is another APT that uses a wiper which is injected inside the running browser's memory. When the module is started, a script containing a sleep function is dropped and executed. This script employs a 10 second sleep function between a number of its activities. The same waiting time is also employed by Pegasus before and between some activities [166]. The backdoor in the SolarWinds attack also implements a number of sleep

functions [52].

3.4 Enhanced Scanning

As discussed in 2.2.3, SDNmap is an open-source scanner that can reconstruct flow rules in an SDN. We add functionality to SDNmap to enable it to carry out its work in a more stealthy manner (as might be expected of APTs). The generated tool, APT-SDNmap, allows a scanner (adversary or security penetration tester) to carry out a scan as part of flow rule reconstruction, but also to spread out that attack over time. The adversary may also choose to limit the number of ports that are to be scanned. These "slow and low" techniques, keeping rates of interaction and numbers of targets low, provide the adversary with an enhanced level of stealth. The implemented techniques in APT-SDNmap, i.e., added to SDNmap, are (with referenced variables taken from Algorithm 1):

- *Initial delay:* APT-SDNmap waits for 10 minutes ($T = 600$ seconds) after the installation in a victim machine before doing any scan or transmission.

- *Waiting between activities:* In APT-SDNmap, two time (T) intervals are employed during scans: a selected random time in the range 120 - 150 seconds from one type of scan to another (main scan S_μ); and a random time in the range 10 - 15 seconds as a waiting time when doing a scan inside a main type of scan (sub-scan S_v). For example, if the user executes a protocol and a port scan, it waits for around 120 to 150 seconds between the protocol scan and the port scan.

  Around two minutes is a reasonable waiting time between two main activities such as from one type of scan to another. But inside one scan, such as scanning a range of IPs (from x.x.x.0 to x.x.x.255), this would be excessively high. However, scans inside one type, such as a port scan, wait just 10 - 15 seconds between one port and another. Implementing a random value between 120 and 150 seconds, or 10 and 15 seconds, helps avoid being recognised by pattern matching algorithms.

- *Scan the most commonly attacked ports:* In addition to the two time intervals, APT-SDNmap offers an option to scan the most common ports for attackers, namely 25, 80, 443, 20, 21, 23, 143, 3389, 22, 53, 67, 68, and 110 [167] [154]. This option makes the level of interaction much lower than when interacting with a full range of ports. However, the user is free to scan all ports or select some.

Algorithm 1, shows the proposed time intervals of the attack. We suppose one host of the targeted network is already compromised and that APT-SDNmap has been installed. The adversary launches a scan inside the network using APT-SDNmap.

Algorithm 1: Sleep functions implementation algorithm

Initialisation
S_μ: main scan
S_v: sub-scan
S_ι: Instance scan
τ_1: lower value of waiting time
τ_2: upper value of waiting time
f_{rnd}: a random value generator
$Sleep(t)$: a sleep function where t time in seconds
T: time value $t_0 \leftarrow 600$
$Sleep(t_0)$
do
  $S_\iota \leftarrow S_\iota \in S_\mu$
  do
    $S_\iota \leftarrow S_\iota \in S_v$
    $T = f_{rnd}(\tau_1, \tau_2)$ // $\tau_1 = 10$, $\tau_2 = 15$
    $Sleep(T)$
  while S_v *is not finished*;
  $S_\iota \leftarrow S_\iota \in S_\mu$
  $T = f_{rnd}(\tau_1, \tau_2)$ // $\tau_1 = 120$, $\tau_2 = 150$
  $Sleep(T)$
while S_μ *is not finished*;

3.5 APT-SDN datasets

The lack of datasets that cover APTs in SDN, or even any dataset for reconstructing flow rules in SDN, means that a dataset must be developed reflecting realistic scenarios for stealth scans (reconstructing flow rules) in SDN. To plausibly reflect real-world operations, different approaches are considered. We implement different

network protocols (e.g. TCP, UDP, ICMP, IGMP, and ARP) reflecting the typical or widespread way in which computer networks are used. To make it more realistic, we generate malicious and normal traffic at the same time. The infected machine generates both attacks and normal traffic. Most available datasets generate malicious traffic and normal traffic at different times. The generated dataset has three different size scenarios (four hosts, eight hosts, and sixteen hosts). The datasets are cleaned, prepared, and made freely and publicly available [133].

3.5.1 Dataset generation methodology

We used a physical Windows 11 machine that hosts a virtual machine (VirtualBox) for the testbed. The virtual machine hosts Ubuntu 16.04 (64-bit) as VM, with the specifications including 2 GB of memory, a 30 GB virtual hard disk, and three processors. Within this VM (Ubuntu 16.04), we ran mininet, creating four hosts, one single switch (OVS switch) and one SDN controller (ovs-controller). All hosts were connected with the OVS switch. The network architecture and the SDN components involved in the experiments are depicted in Figure 3.1. One of these hosts is considered a server (IP: 10.0.0.2) receiving data from other hosts. The remaining three hosts were installed with Ostinato [168], enabling the generation of normal traffic. Ostinato is used to emulate typical user behaviour by transmitting varying numbers of normal packets across different protocols, such as TCP, UDP, ICMP, IGMP, and ARP. APT-SDNmap is used to implement scanning inside the network. It is assumed that one of the hosts has been compromised and that APT-SDNmap has been successfully installed in it. (In our experiment, this host had the IP address 10.0.0.1 .) Except for the server, all hosts send normal traffic with different protocols using Ostinato. This includes the infected machine, which simultaneously launches attacks. APT-SDNmap sends probing packets to the server (IP:10.0.0.2) over TCP to reconstruct flow rules. It limits the scanned ports to the most common ports for attackers (i.e. 5, 80, 443, 20, 21, 23, 143, 3389, 22, 53, 67, 68, and 116). Specifically, we executed the command: *python main.py 10.0.0.2/32 TCP h1-eth0['s']*. The traffic is processed using Wireshark to produce a .pcap file.

We conducted two more experiments with varying network sizes to evaluate the scalability of our proposed detection approach. We refer to the previous experiment

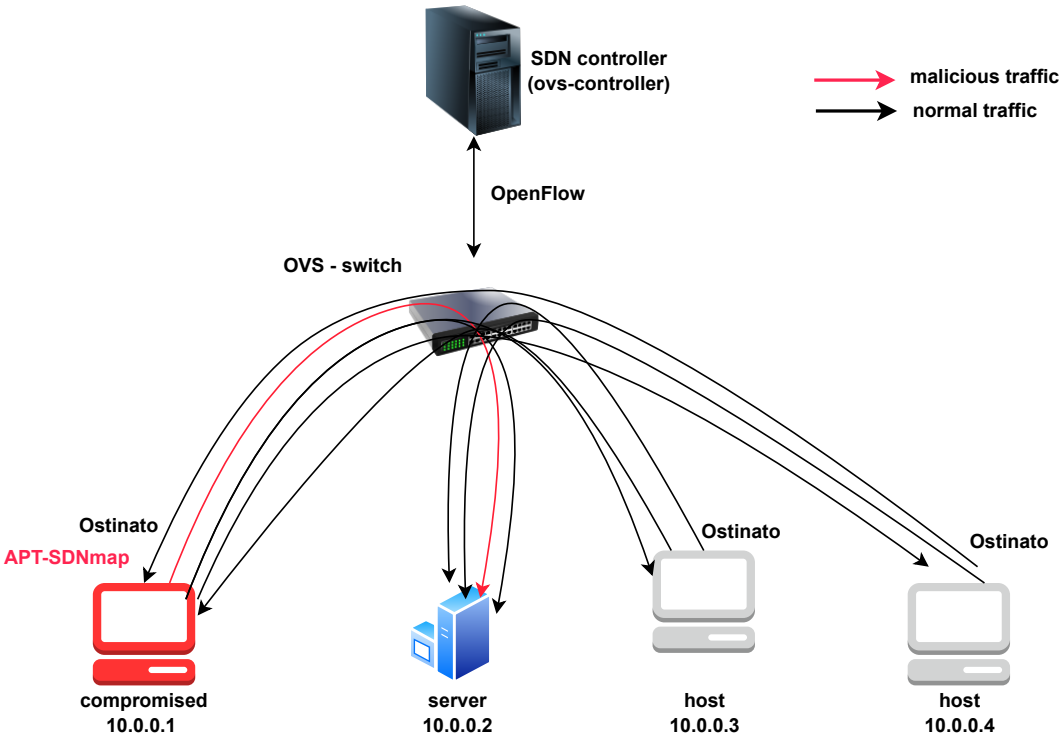

FIGURE 3.1: Network set up used to generate APT-SDNmap dataset

(with four hosts) as $h = 4$. The other two experiments maintained the same architecture (one switch and one controller) but with more hosts: eight hosts ($h = 8$) and sixteen hosts ($h = 16$). Both the malicious and normal transactions followed the same scenario, involving one compromised machine (launching a probing attack using APT-SDNmap), one server, and the remaining hosts (including the compromised one) generating normal activities using Ostinato.

3.5.2 Data preparation

The network protocol analyser t-shark is used to generate a CSV file by constructing the basic features from the pcap file. These features are frame number, frame date and time, source IP, destination IP, source MAC address, destination MAC address, protocol, source port, destination port, ethernet type, TCP flag and frame length. A cleaning data process is carried out. Every transmission is labelled manually (to be 0 or 1) based on different features such as time of transmission, source IP and port, and used protocol. Malicious transmissions are labelled as 1s, and normal transmissions as 0s. In each network size scenario, the abnormal traffic constitutes around 5%

TABLE 3.4: Experiment scenarios

scenario	number of hosts	normal	attack	total
1	4	13,944	816	14760
2	8	13,801	820	14621
3	16	16,440	803	17243

of the total. Table 3.4 shows the total number of transactions in each of the three network size scenarios.

3.5.3 Data pre-processing

This stage is essential before applying Machine Learning techniques. The widely used scikit-learn toolkit [169] provides the ML functionality used throughout our work. There are some null data dealt with by substituting with 0s. The data is split into a training set and a testing set, 70% and 30% respectively, using scikit-learn's *train_test_split*. SMOTE (synthetic minority oversampling technique) [170] is applied to the training set to balance data between the two classes by randomly replicating the minority class (1) samples to achieve balance with the majority class (0). Finally, the dataset features are scaled using the standard scalar scikit-learn technique StandardScaler[171].

3.5.4 Feature Engineering

In this section, we engineer and extract a set of features that can facilitate the detection of APT scanning attacks (discussed in section 3.4) in SDN. A large number of features (basic and historical-based) are examined to deliver features that have a significant effect on the detection model. 17 basic features (e.g. packet protocol) are extracted from the headers of packets. Another eight historical features (e.g. the average number of ICMP packets from a sender) are engineered based on the extracted basic features. Table 3.5 shows all extracted features and their descriptions. Figure 3.2 represents an overview of the derived historical-based features and their significant role in detection.

TABLE 3.5: Dataset features

Feature	Type	Description
Basic features		
No	Integer	frame number
time	Nominal	transition date and time
src_ip	Nominal	source IP
dst_ip	Nominal	destination IP
src_mac	Nominal	MAC source address
dst_mac	Nominal	MAC destination address
protocol	Nominal	protocol type
src_port	Integer	transmission source port
dst_port	Integer	transmission destination port
eth_type	Nominal	ethernet type
flag	Nominal	transmission flag
frame_len	Integer	frame length
TCP	Binary	TCP protocol
UDP	Binary	UDP protocol
ICMP	Binary	ICMP protocol
IGMP	Binary	IGMP protocol
ARP	Binary	ARP protocol
historical-based features (calculated for the identified host)		
src_host_count	Float	proportion of network packets (from any sender) sent with src as sender
src_ARP_count	Float	proportion of packets from src that are ARP packets
src_ICMP_count	Float	proportion of packets from src that are ICMP packets
src_IGMP_count	Float	proportion of packets from src that are IGMP packets
src_TCP_count	Float	proportion of packets from src that are TCP packets
src_UDP_count	Float	proportion of packets from src that are UDP packets
diff_proto_range	Float	proportion of available protocols used by src
diff_dst_port	Float	Count of unique ports accessed by the sender

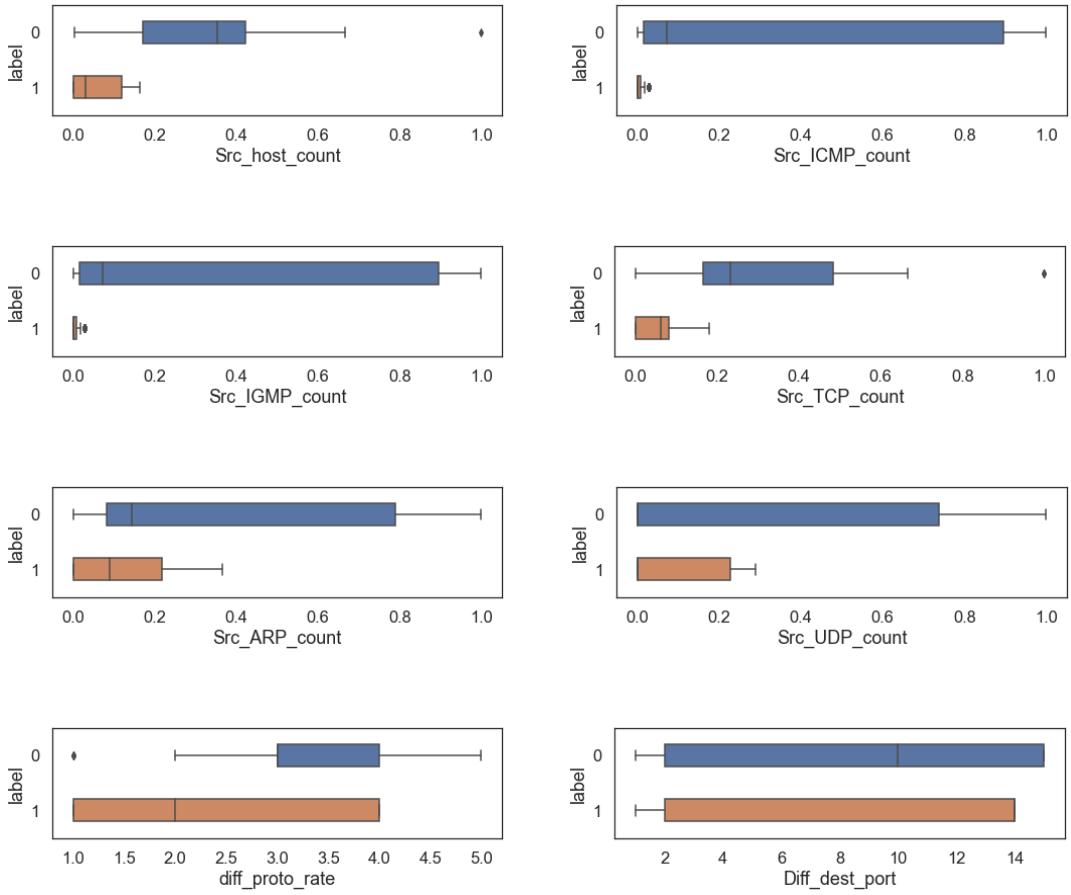

FIGURE 3.2: The impact of the extracted historical-based features on the proposed model

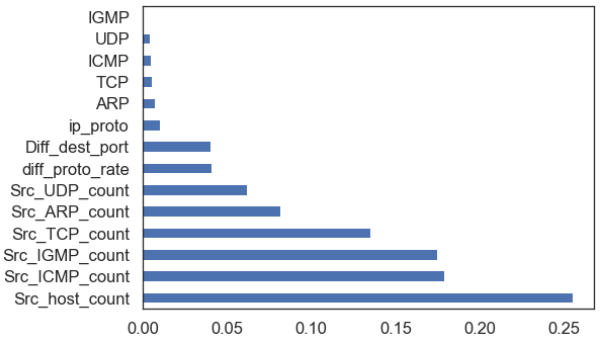

FIGURE 3.3: The most important features on APT-SDNmap dataset

In our experiments to develop the classifiers, we remove features that are overly specific: frame number, time, source IP, destination IP, source MAC address, destination MAC address, source port, and destination port. For example, in the training phase, when a specific source IP is flagged as benign, it would generally be predicted as normal in the testing phase. If the model learns to rely on specific values, it might perform well on the training data, but it could fail to perform over new data (i.e. it has been overfitted). All historical-based features are employed in the model, but from the set of basic features we use only Protocol, TCP, UDP, ICMP, IGMP and ARP. The omissions indicated are convenient but we would not assert that such omissions are *essential*. Rather, additional modelling functionality would have to be invoked to provide use at an appropriate level of abstraction. Thus, our approach is a simple and pragmatic one, though our results suggest that it is also an effective one.

Feature importance is a technique that measures the usefulness of given features in a model, indicating how every feature contributes to the model's prediction [74]. The Random Forest classifier [172] is used to evaluate the importance of all used features. The results are presented in Figure 3.3. In each network size scenario, ($h = 4$, $h = 8$ and $h = 16$), we conducted experiments with all features ($f = 14$), the top 9 features ($f = 9$) and the top 5 features ($f = 5$), selected after applying feature importance.

TABLE 3.6: Hyper-parameter tuning

parameter	description	optimal value
K-nearest neighbour		
n_neighbors	number of neighbour	8
metric	distance function of K	euclidean
XGBoost		
max_features	maximum number of features to be used in a single run	sqrt
en_estimators	number of trees	200
max_depth	maximum depth of every tree	7
Support Vector Machine		
kernel	transforming data function	rbf
C	regularization parameter	50
gamma	kernel coefficient	scale
Decision Tree		
Criterion	the function used to measure the quality of split	gini
max_depth	maximum tree depth	7
Random Forest		
max_features	maximum number of features used by individual tree	sqrt
max_dept	maximum tree depth	7
n_estimators	number of trees	200

3.6 Detecting scans using ML

In supervised learning, the model is trained over labelled data to predict unlabeled ones in the future. We have applied a number of the most popular supervised machine learning algorithms to the generated datasets. The best results came from K-nearest neighbour, Support Vector Machine, Decision Tree, Random Forest and XGBoost. Hyper-parameter tuning seeks to find parameter choices that give the best performance for the given ML techniques. GridSearchCV [173], which is supported by scikit-learn, is used for hyperparameter tuning. Further manual selection is employed to avoid over-fitting or under-fitting. Table 3.6 illustrates the selected parameters for the used algorithms. The following paragraphs summarise the usage of ML classifiers (discussed in detail in the previous chapter section 2.6) that are used in this work. The results of their application to the first network size scenario ($h = 4$) are presented here. The results of all scenarios are discussed in section 3.6.2.

The K-Nearest Neighbour (K-NN) algorithm assumes that similar instances of one class are located near each other. Choosing the right value of K is critical to increase the accuracy of the prediction rate. K=8 gives good prediction whilst avoiding over-fitting. Euclidean is the most popular distance function for K-NN and the one picked after applying hyperparameter tuning. K-NN performs very well over all experiments.

A Decision Tree (DT) is one of the most powerful classification tools in Machine Learning. In the experiment, the maximum tree depth (max_depth) is selected to be seven. During the experiments, a high number of features gives better results.

A Random Forest (RF) is an ensemble classifier consisting of multiple Decision Trees. A number of parameters are crucial to increase predictivity. *sqrt* is implemented for the maximum number of features. It takes the square root of the number of available features to fit in every tree in the model. The more trees ($n\_estimators$), the better performance of the model but this requires high computation and the model could not be generalised. Using 200 trees with a maximum depth of 7 produces promising results. The performance is mostly very high in all experiments, and the number of false positives is reduced when the number of features used is low.

The eXtreme Gradient Boosting(XGBoost) is an ensemble classifier that combines a set of Decision Trees to provide more accurate predictions in a fast way. 200 trees with a maximum depth of 7 give the best results. With just ($f = 5$) features, XGBoost outperforms all other classifiers in our experiment with at least 97.82% in all selected evaluation metrics used.

The Support Vector Machine (SVM) is one of the strongest techniques for binary classification. However, it shows the highest number of false positives among these techniques, especially when the number of features is reduced.

3.6.1 Proposed model

The SDN controller has global visibility over network devices. The network administrator can use any SDN controller (e.g. OpenDaylight [174], ONOS [175] and Ryu [176]) to provide the NIDS with batches of data. It can be configured to mirror network traffic received in SDN switches. Furthermore, the controller can request and receive network statistics from network devices using southbound APIs. For example, OpenFlow messages $ofp\_flow\_stats\_request$ and $ofp\_flow\_stats\_reply$ are used to request and receive, respectively. When the data are available, the feature extractor module extracts basic features and calculates the historical-based features (these features are discussed in 3.5.4). We created a Python script (for the APT-SDNmap dataset which used in experiment 1) to work as the feature extractor module. The ML classifier module uses the ensemble classifier XGBoost to classify (malicious or being) all instances. If the traffic is classified as suspicious, the administrator is notified to take further actions, such as blocking IPs or updating the flow rules table. Figure. 3.4 shows the proposed network architecture of the proposed model. If the classifier has been updated by training over new data, the admin can also update the current model through the northbound API.

3.6.2 Experimental results

A number of experiments are conducted and are described below:

1. Experiment 1:

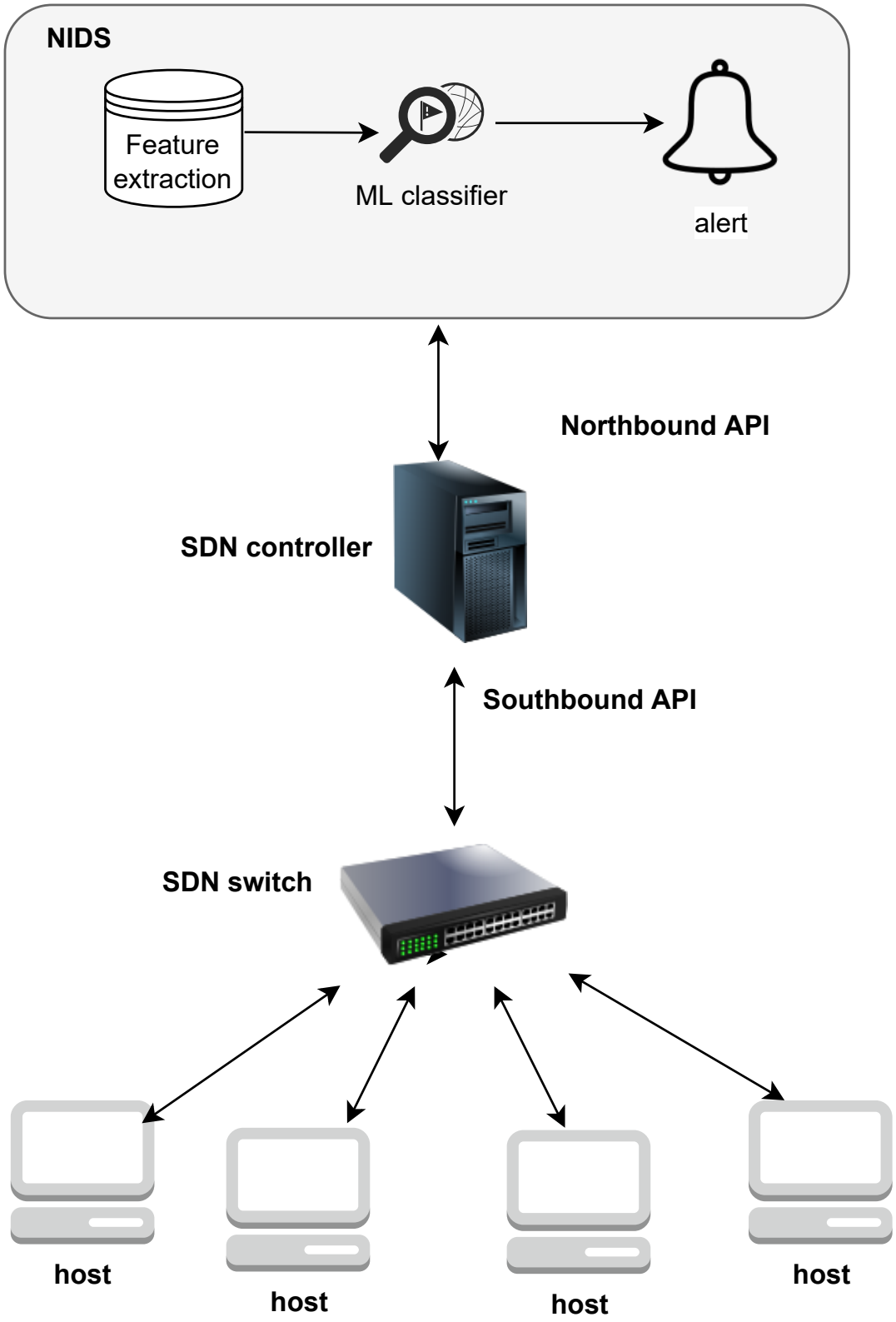

FIGURE 3.4: Network architecture of the proposed model

TABLE 3.7: Results on APT-SDNmap dataset expressed as percentages

	ML	f=14				f=9				f=5			
	tech	*Acc*	*Rec*	*Pre*	*F1*	*Acc*	*Rec*	*Pre*	*F1*	*Acc*	*Rec*	*Pre*	*F1*
h=4	KNN	99.86	98.26	99.12	98.69	99.88	98.69	99.12	98.91	99.70	98.26	96.17	97.20
	DT	99.43	97.82	91.83	94.73	99.27	97.82	89.28	93.36	98.05	98.69	73.22	84.07
	RF	99.36	99.13	89.76	94.21	99.54	98.26	93.38	95.76	99.72	97.39	97.39	97.39
	XGB	99.93	99.13	99.56	99.34	99.93	98.69	100	99.34	99.77	97.82	97.82	97.82
	SVM	98.64	98.69	79.93	88.32	98.71	99.13	80.56	88.88	97.29	97.39	66.27	78.87
h=8	KNN	99.63	96.34	96.34	96.34	99.59	95.43	96.31	95.87	99.72	98.63	96.00	97.29
	DT	99.72	98.17	96.41	97.28	99.70	98.17	95.98	97.07	99.47	93.60	95.79	94.68
	RF	99.77	99.08	96.44	97.74	99.79	98.63	97.29	97.95	99.65	97.26	95.94	96.59
	XGB	99.74	98.63	96.42	97.51	99.79	98.63	97.29	97.95	99.70	97.26	96.81	97.03
	SVM	98.95	99.08	83.14	90.41	98.95	99.08	83.14	90.41	97.85	80.82	77.29	79.01
h=16	KNN	99.67	95.41	97.44	96.42	99.67	95.41	97.44	.96.42	99.63	95.00	97.02	96.00
	DT	99.69	97.08	96.28	96.68	99.69	97.08	96.28	96.68	99.63	96.25	95.85	96.05
	RF	99.36	95.00	97.02	96.00	99.76	97.91	97.10	97.51	99.74	97.50	97.09	97.29
	XGB	99.78	98.33	97.11	97.72	99.82	99.16	97.14	98.14	99.76	97.91	97.10	97.51
	SVM	98.60	75.00	93.75	83.33	98.62	75.00	94.24	83.52	96.55	36.66	77.19	49.71

* **KNN**: k-Nearest Neighbors; **SVM**: Support Vector Machine; **DT**: Decision Tree; **RF**: Random Forest; **XGB**: XGBoost; **Acc**: Accuracy; **Rec**: Recall; **Pre**: Precision; **f**: number of features

Five machine learning classifiers were applied on the APT-SDNdataset. A different number of features are considered for every network size scenario. The use of all features ($f = 14$) is evaluated as the use of the most important 5 ($f = 5$) and the most important 9 ($f = 9$) features. The importance of all features is shown in Fig 3.3.

Table 3.7 shows the experiment results. On average, XGBoost outperforms all other classifiers, with any number of features, scoring at least 97% in all metrics (*Accuracy*, *Recall*, *Precision* and *F1 score*) except Precision in the second scenario $h = 8$ (which attains nearly the same value). K-NN has good results over any number of features, but it is practically not preferred because of computational costs and time to execute. Conversely, SVM is prone to obtaining a high number of false positives and false negatives. Overall, K-NN and DT perform better when the network size is small. Random Forest and XGBoost give fairly uniform results.

2. Experiment 2:

As there is no other dataset including APTs in SDN to do a further evaluation

TABLE 3.8: Results on InSDN dataset (probe attacks and normal instances) in percentage

ML	f=75				f=21				f=6			
tech	*Acc*	*Rec*	*Pre*	*F1*	*Acc*	*Rec*	*Pre*	*F1*	*Acc*	*Rec*	*Pre*	*F1*
KNN	99.96	99.96	99.96	99.96	99.96	99.98	99.97	99.98	99.81	99.90	99.95	99.93
DT	99.96	99.96	99.96	99.96	99.98	99.99	99.98	99.98	99.95	99.97	99.94	99.96
RF	99.79	99.98	99.66	99.82	99.99	99.98	99.95	99.96	99.94	99.95	99.95	99.95
XGB	99.98	99.99	99.98	99.98	99.98	99.99	99.97	99.98	99.96	99.97	99.96	99.96
SVM	99.94	99.96	99.93	99.95	99.51	99.86	99.31	99.58	99.16	99.94	98.64	99.29

* **KNN**: k-Nearest Neighbors; **DT**: Decision Tree; **RF**: Random Forest; **XGB**: XGBoost; **SVM**: Support Vector Machine; **Acc**: Accuracy; **Rec**: Recall; **Pre**: Precision; **tech**: Technique

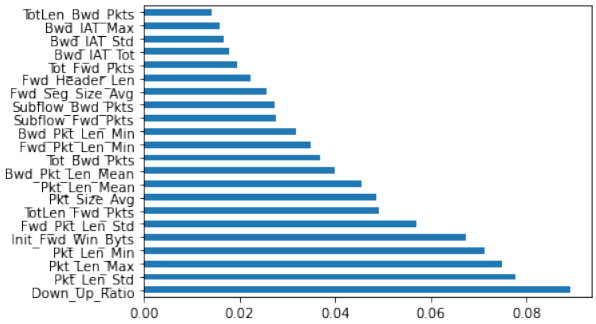

FIGURE 3.5: The most important features on probe InSDN

for the proposed model, it is evaluated over *probe* attacks in the InSDN dataset [127]. As mentioned in 2.10, InSDN has various attacks forming more than 80% of the entire dataset, which is considered very high. As this chapter focuses on scanning, the dataset is further prepared. Only instances representing probe attacks are kept. All other types of attacks (DoS, DDoS, web attacks, R2L, botnet and U2R attacks) are removed. The dataset now includes just probe attacks and normal samples, forming around 59% and 41% of the dataset, respectively. In this thesis, we refer to this dataset version as the *probe InSDN* dataset.

The results of evaluating the proposed model (and all classifiers used in Experiment 1) applying the same configurations (hyper-parameters) are shown in table 3.8. More than 99% is attained in all metrics over different numbers of features: (all features ($f = 75$), default feature importance ($f = 21$) and the five most important features($f = 5$)). Figure 3.5 shows the important features using a Random Forest classifier. Although the default features of InSDN are used in the evaluation, we removed the features that we believe could cause over-fitting on the training data. These are flow id, source ip, destination

TABLE 3.9: Results on InSDN dataset expressed as percentage

ML tech	f=5			
	Accuracy	*Recall*	*Precision*	*F1*
KNN	99.83	99.97	99.81	99.89
DT	99.82	99.97	99.81	99.89
RF	99.80	99.93	99.82	99.88
XGB	99.85	99.99	99.82	99.91
SVM	99.51	99.86	99.52	99.69

* **KNN**: k-Nearest Neighbors; **DT**: Decision Tree; **RF**: Random Forest; **XGB**: XGBoost; **SVM**: Support Vector Machine

ip, time stamps, source port and destination port. We argue that the model performance shows higher results over this dataset (probe InSDN) than those over APT-SDNdataset. This is because the probing attack in this dataset is not stealthy. In addition, the number of attacks (58.9% of the dataset) is higher than the number of attacks in the generated dataset (APT-SDNmapdataset), which forms around 5% of the whole dataset.

3. Experiment 3:

   In these experiments, the proposed classifiers are evaluated over the actual InSDN dataset. It is different from probe InSDN which is used in Experiment 2 by using the entire dataset, which includes various attacks (i.e. not just probe attacks). All attacks are mapped to be one class (malicious), and the binary classification methods are implemented. Attack traffic makes up 80% of the InSDN dataset and so it can not be said to represent stealth (e.g. slow-and-low) attacks. However, it is important to evaluate the system against different kinds of attacks launched on an SDN network. Feature importance is applied, see figure 3.6, to enhance the detection and performance. With just five features, the model still shows outstanding results scoring more than 99% in all evaluating metrics. All results are shown in table 3.9.

3.6.3 Discussion and challenges

Even though the stealth attacker tries to remain under detection thresholds, some scanning behaviours could arouse suspicion. Contacting a wide range of different

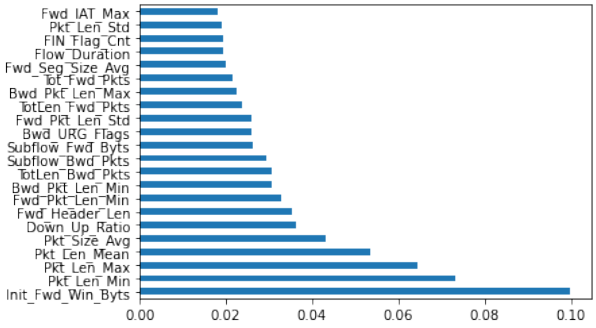

FIGURE 3.6: InSDN feature importance

IPs or using a high number of ports could identify a sender aiming to discover network configuration. Using a high number of ARPs may lead to suspicions that the initiator is discovering the active IP devices inside the local network. A specific user using a wide range of protocols could indicate that this user is trying to identify the supported protocols in the network.

In traditional attacks, a reasonable amount of all traffic is malicious. In DDoS attacks, the majority of sampled traffic may be malicious. In stealth attacks, malicious traffic forms a small fraction of all traffic. The datasets arising are significantly unbalanced. Developed approaches must be able to handle such unbalanced datasets. Nonlinear algorithms such as K-Nearest Neighbour and Decision Trees have the potential to detect these kinds of attacks. However, we found ensemble classifiers such as Random Forests and XGBoost are the best to detect APTs when attack traffic is much lower than normal traffic.

It is still a significant challenge to provide a comprehensive simulation of APT characteristics in one dataset, despite the emergence of research work concerned with simulation and benchmark datasets. This chapter addresses one particular APT characteristic (stealthy scanning of nodes). However, such scanning is an important activity since it generally acts as an enabler of future attacks. It is, therefore, a highly suitable target for enquiry in its own right.

3.7 Summary

In this chapter, an enhanced means of stealth is proposed for scanning in SDN. Such a strategy may be used by APTs to escape current detection. A dataset has been created and used to investigate how such enhanced scanning attacks can be detected by the application of Machine Learning techniques. We have also shown that significant feature reduction is possible whilst retaining robust performance and have applied hyper-parameter optimisation to derive maximum benefit from specific ML techniques. XGBoost is proposed for detection, which shows promising results. The system is evaluated over different datasets, showing consistency in performance.

The experiments show that suitably configured ML-based detectors can achieve excellent detection rates for stealthy scanning in SDNs. Feature engineering, part of such ML configuration, has an important role to play in the further development of efficient scanning detection (and more generally). The work used supervised learning only, and investigation of an anomaly-based approach would seem a natural next step, since the latter approach is typically better suited to an evolving threat environment. The APT-SDNmap tool assumes a single source for scanning. Allowing multiple nodes to collaborate could provide even stealthier scanning.

Chapter 4

A Hybrid NIDS for Detecting Stealthy Scans

Chapter 3 evaluated the most popular signature-based detection algorithms in machine learning for stealthy scan detection. A model to detect known stealthy scanning attacks in SDN using XGBoost was proposed. Although effective, signature-based detection is unlikely to perform well on unseen attacks. Anomaly-based detection is widely acknowledged as offering better prospects for detecting such unknown attacks. This chapter investigates how well a hybrid (signature-based and anomaly-based) IDS can detect known and unknown stealth attacks.

Thus it investigates the second research question, as stated in section 1.2, RQ2: *Can we use a machine-learning and hybrid (signature-based plus anomaly-based) approach to detect the stealthy reconstruction of flow rules in SDN networks?*

4.1 Introduction

APTs have multiple stages in their campaign. In an isolated or well-protected network, it can be a significant challenge for the attacker to get inside the targeted network. An insider can help in this case. The SDN architecture allows an insider to more easily get information about the network configuration than in traditional networks. Investigating the detection of insider scans and external attacks can help to cover the two types of networks (isolated and public).

4.1.1 The insider perspective and APTs

An insider within an SDN network operates with privileges external agents do not have. They are expected to use those privileges in a manner consistent with good behaviour within the system. However, they may use their privileges for malicious goals. Detection in part consists therefore in the identification of the end goals of such a user from their actions. This is acknowledged as very hard to do. Furthermore, their presence on the network is authorised and does not in and of itself arouse suspicion. An insider is, therefore, in a very powerful position. The insider can gather sensitive information or install required malicious software in the targeted network [177]. For an isolated or well-protected network, insider involvement may be essential for a successful attack. Gathered information helps the attacker to move inside the victim's network and compromise new devices. An SDN network provides the insider with a greater capability to gather information than in a traditional network. The dynamic configuration helps the attacker in gaining information. Another issue in SDN is that compromising the controller can lead to a compromise of the whole network.

An insider starts from a highly advantageous position. But insiders can also adopt stealth strategies to make detection even harder, carrying out scanning attacks over an extended period of time, with probes issued at an extremely low rate. This also presents an opportunity for defence. Detection of insiders can be regarded as the 'stretch goal' for intrusion detection in modern systems, such as those employing SDN architectures. SDN provides more flexibility to monitor the whole network and get useful information from network devices. If we can detect insider attacks, external attacks should not present much of a problem. Thus, we believe it is prudent to target insider attacks as a priority. Finally, an *insider perspective* for intrusion detection can provide defence-in-depth. Ultimately, a system's defences may fail to prevent the compromise of a node by an external agent. That comprised node is now an insider. In this chapter, we assume the insider is a user inside the network who has root access and so is able to install APT-SDNmap in a host and launch attacks. Performing scans using APT-SDNmap on the compromised host is enough to reconstruct flow rules in SDN switches. There is no need to move to other SDN

components to accomplish its goals. If movement is not essential, then there seems added incentive not to move: movement is not part of the normal operation of an insider's activities and so increases the chances of attracting attention.

4.1.2 Motivations and contributions

Stealth scans can be used to gather information that underpins further attacks (as discussed in section 2.2.3). Their adoption of some APT characteristics makes stealth scans hard to detect. With the help of an insider, the attackers can bypass the front-line defences and access the targeted network.

When adversaries create new attacks and adapt their behaviour (as APTs do) then employing anomaly-based IDS has better prospects for detecting such previously unseen malicious activity. However, this comes at a price. Anomaly-based approaches raise more false alarms. These alarms are often referred to as false positives (FPs). A further problem is that some malicious activity, even known malicious activity, may 'look like' normal activity. Consequently, exhibiting sets of measured properties consistent with experienced benign activity will usually not raise an alarm. This gives rise to so-called false negatives (FNs).

A hybrid approach can address the weaknesses of individual ML techniques. An anomaly detection approach can enhance the detection of unforeseen attacks (a weakness of the signature-based approach). On the other hand, the high rate of false positives (false alarms) by anomaly detection will be reduced by the use of a signature-based technique.

In this chapter, we propose a hybrid machine learning-based NIDS to detect stealth attacks, such as APTs, in an SDN. Incorporating the two detection techniques, signature-based and anomaly-based, improves detection. Two significant machine learning techniques are adopted to implement these approaches: XGBoost for signature-based detection and a One-class Support Vector Machine (one-class SVM or OC-SVM for short) for anomaly-based detection. The proposed model is evaluated over different datasets that include internal (insider) or both internal and external attacks. To improve the performance in the model, dimensionality reduction and hyper-parameter tuning are applied.

The contributions of this chapter are:

- The development of a hybrid NIDS that integrates two detection techniques is novel for this domain. To the best of our knowledge, it is the first IDS system using anomaly-based techniques to detect APTs in SDN networks.
- The evaluation of the proposed model over several datasets.
- A comparison between the proposed system and the other standard ML detection techniques and the most relevant works is given.

4.1.3 Chapter organisation

The remainder of this chapter is organised as follows. Section 4.2 reviews the most relevant proposed hybrid NIDS systems in the literature. Section 4.3 describes the proposed approach. Section 4.4 presents the implementation and evaluation of the proposed model. Section 4.5 gives a summary of the chapter.

4.2 Related Works

A number of researchers have proposed combining two Deep Learning classifiers to improve the detection in IDSs in SDN [155; 178]. The systems are evaluated over NSL-KDD, which has no SDN traffic. Integration of a One-class SVM and an LSTM-Autoencoder is proposed and tested over InSDN (an SDN-based dataset discussed in 2.10). The dataset is imbalanced and the majority of the traffic is attack traffic. Similarly, a hybrid system [179] using a Long short-term memory (LSTM) and Conventional Neural Network (CNN) is proposed for anomaly detection. However, it is evaluated over the CICIDS 2017 dataset, which contains only traditional attacks (discussed in detail in section 2.10). Nevertheless, none of the aforementioned works considers stealth attacks such as APTs. Table 4.1 presents a comparison of the proposed work and the most relevant works.

4.3 Proposed Scheme

The proposed hybrid system is composed of two main modules. The signature-based detection module uses the XGBoost classifier and the anomaly-based detection module uses a One-class SVM. The NIDS monitors streaming data through the

TABLE 4.1: Related works comparison

Scheme	SDN	Stealth	Accuracy*	Dataset Evaluation	Technique
[155]	×	×	89%	NSL-KDD	GRU-RNN
[178]	×	×	87%	NSL-KDD	GRU-LSTM
[180]	✓	×	90.5%	InSDN	OC-SVM-LSTM-Autoencoder
[179]	✓	×	98.60%	CICIDS2017	LSTM-CNN
Proposed model	✓	✓	98%	APT-SDN dataset	XGboost-OC-SVM

* **GRU**: Gated Recurrent Unit; **RNN**: Recurrent Neural Network; **LSTM**: Long Short Term Memory; **OC-SVM**: One-Class SVM; **CNN**: Convolutional Neural Network; * Accuracy column presents scores as given in the original sources (Scheme column).

SDN controller. The controller can request and receive the network statistics from SDN switches using southbound APIs. When a new packet arrives at the model, the signature-based detection module (XGBoost classifier) investigates whether the pattern of the traffic is malicious or normal. If the pattern of the attack is already defined, then the signature-based approach is best suited to detecting it; for that reason, we use XGBoost to perform the first check. If it is classified as malicious, then this transaction is deemed to be an attack. Otherwise, the transaction is inspected by the anomaly-based module (using a One-class SVM), as a further check for unknown attacks. If any one of the models (signature-based or anomaly-based detection), classifies the transaction as malicious, the administrator would be notified to take some action, for example, instructing the flow tables to be updated so that communications from the compromised machine are dropped, or requiring another mitigation action. If it is judged to be normal in both cases, then no further action is taken. The proposed system architecture is illustrated in Figure 4.1.

4.3.1 Signature-based detection module

A signature-based detection model can detect known attacks with low false positive rates. It can detect known attacks quickly and more accurately. In the proposed model, XGBoost is the classifier used for signature-based detection. It is well known for its execution speed and high performance. In the experiments, the model is trained over labelled training datasets, which include attacks and normal traffic, and is evaluated over the test sets.

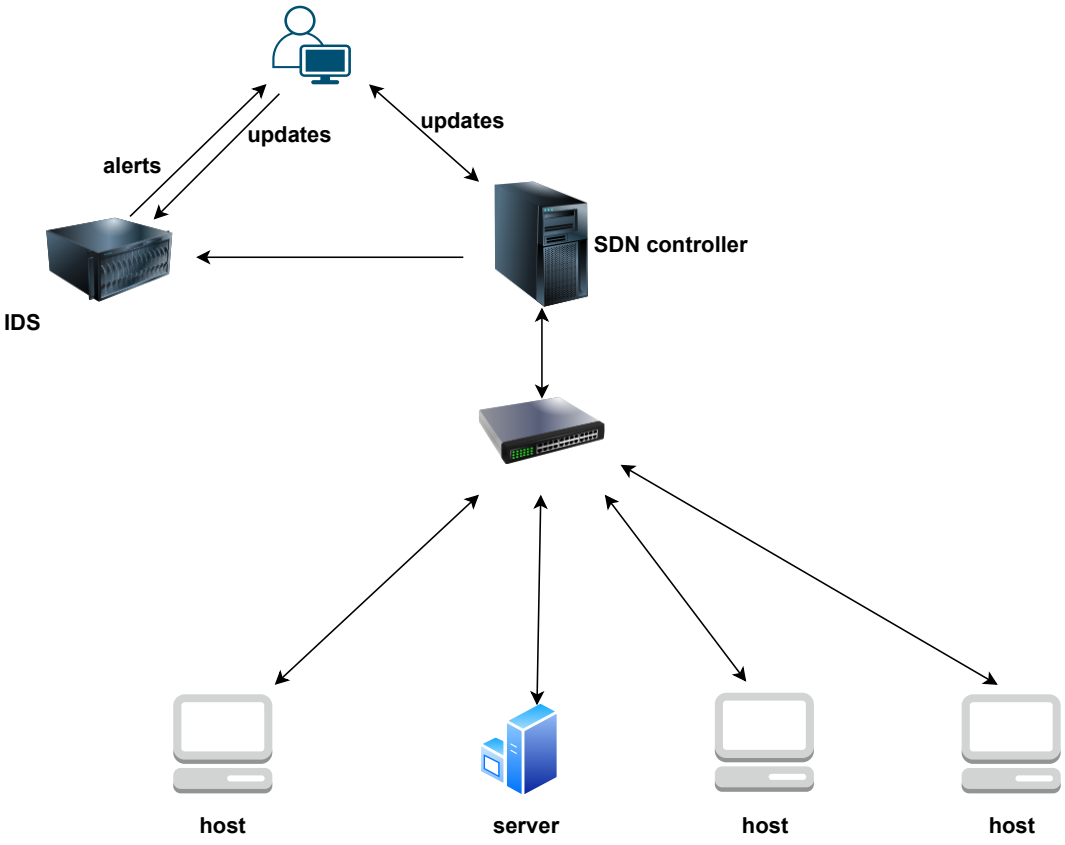

FIGURE 4.1: System Architecture

4.3.2 Anomaly-based detection module

An anomaly-detection algorithm profiles normal behaviour to form a model of that behaviour. Its model defines an envelope of acceptable behaviour and data outside this envelope are considered suspicious. The anomaly-based detection model is used here to help detect unknown attacks. It has a potential to detect zero-day exploits [181]. An OC-SVM is a significant option when non-malicious data dominates a dataset (as is the case with stealth scanning attacks). In this work, OC-SVM is trained over the normal portion of the training set. It is then evaluated over the whole testing set (containing normal and malicious behaviours).

4.3.3 Model parameters

Hyperparameter tuning is applied to find the best performance. The parameter selection considers avoiding over-fitting or under-fitting. In a One-Class SVM, $\nu \in (0, 1]$ plays a major role in the trade-off between generalisation and over-fitting. Another parameter in OC-SVM is $\gamma \in (0, 1]$. Figure 4.2 shows the effects of tuning the

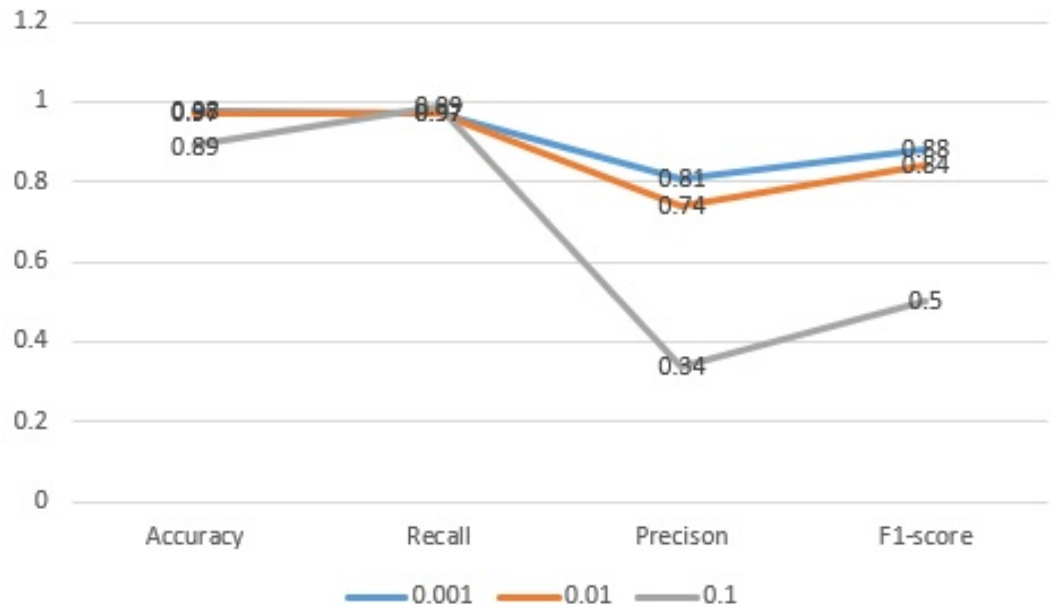

FIGURE 4.2: ν value tuning

ν value. In XGBoost the maximum number of features $max\_features$, the number of trees $en\_estimators$ and the maximum depth of every tree are the most important parameters. Employing a wide range of experiments for different parameters for every classifier, allows a high performing model to be identified. Table 4.2 presents the selected parameters to be used in the proposed model.

TABLE 4.2: Hyper-parameter tuning

parameter	description	optimal value
eXtreme Gradient Boosting(XGBoost)		
max_features	Maximum number of features in every single run	sqrt
en_estimators	Number of trees	200
max_depth	Maximum depth for any tree	7
One-Class SVM (OC-SVM)		
Kernel	Transforming data function	rbf
ν	This controls the fraction of outliers in the system	0.001
Gamma	Kernel coefficient	0.9

4.4 Implementation and Evaluation

The dataset generated in the previous chapter (section 3.2.2) is used to evaluate the proposed model. Further experiments are conducted over different datasets. This further evaluates the system on benchmark datasets, allowing comparison with works most similar to the scheme proposed here.

4.4.1 Preparation and preprocessing

APT-SDNdataset is used in the first experiments (against insider scan). Features that could expose the identity of devices (users) are eliminated. Fourteen features are selected from the dataset to be used in the experiments. These features are: Protocol, TCP, UDP, ICMP, IGMP, ARP, src_host_count, src_ARP_count, src_ICMP_count, src_IGMP _count, src_TCP_count, src_UDP_count, diff_proto_rage and diff_dst_port. Then we standardise the dataset features using the scikit-learn object *StandardScaler*. Principal Component Analysis (PCA) is applied to create fewer information-rich derived features. The experiments show that four such features (also referred to as principal components) are sufficient to represent the whole dataset for our detection purposes. Figure 4.3 shows the trend of how many components are enough for the dataset. Experiments with more than that number of components were attempted but the results were broadly similar. Thus, four components are used in all experiments in this chapter. Figure 4.4 shows the impact of these features (components) on the prediction capability of the model.

The dataset is split into 70% for training and 30% for testing. Because the dataset is imbalanced (malicious scanning transactions are rare), SMOTE (synthetic minority oversampling technique) [170] is applied on the training set to make the number of samples in each of the two classes (denoted by 0 and 1) relatively equal. Figure 4.5 illustrates the preparation and pre-processing phases and how the two ML modules (XGBoost as a signature-based and One-Class SVM as anomaly-based detection) work together to investigate attacks.

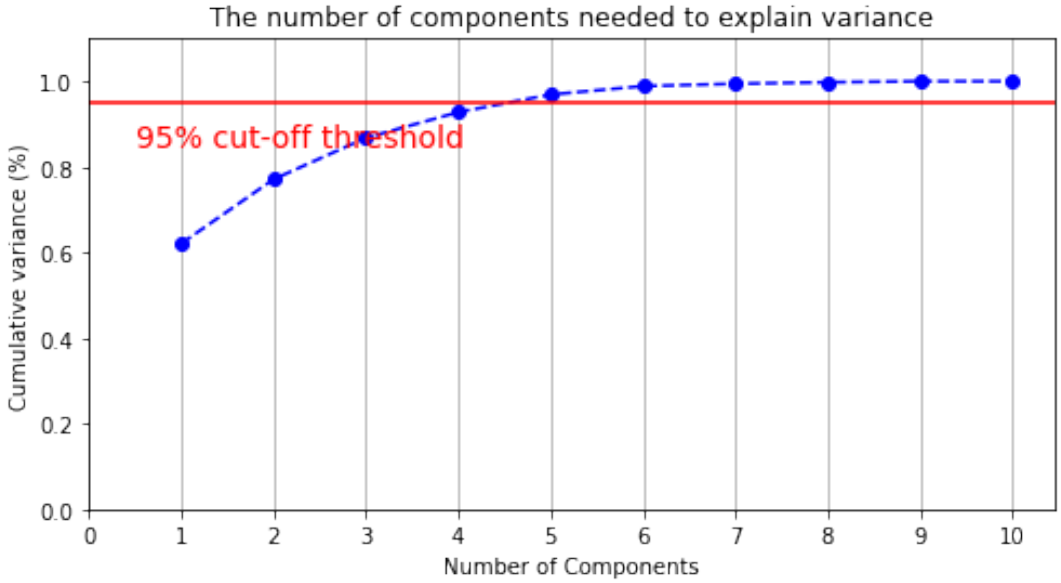

FIGURE 4.3: Number of components are enough to represent the dataset (APT-SDNmap dataset)

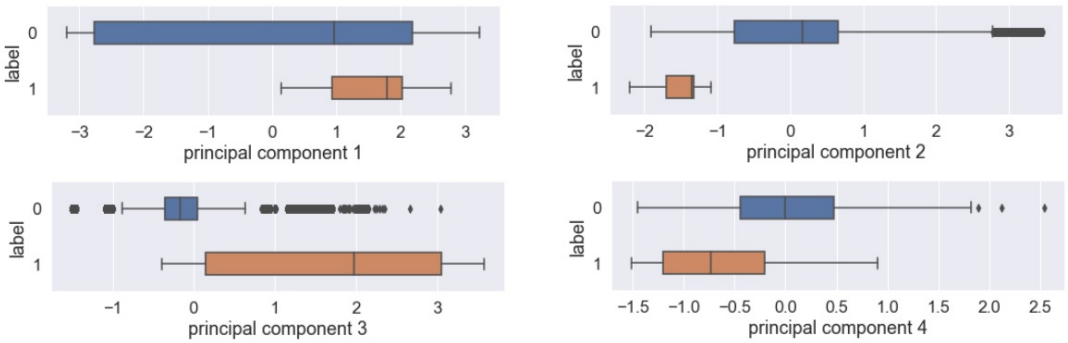

FIGURE 4.4: The selected components affect the model (using APT-SDNmap dataset)

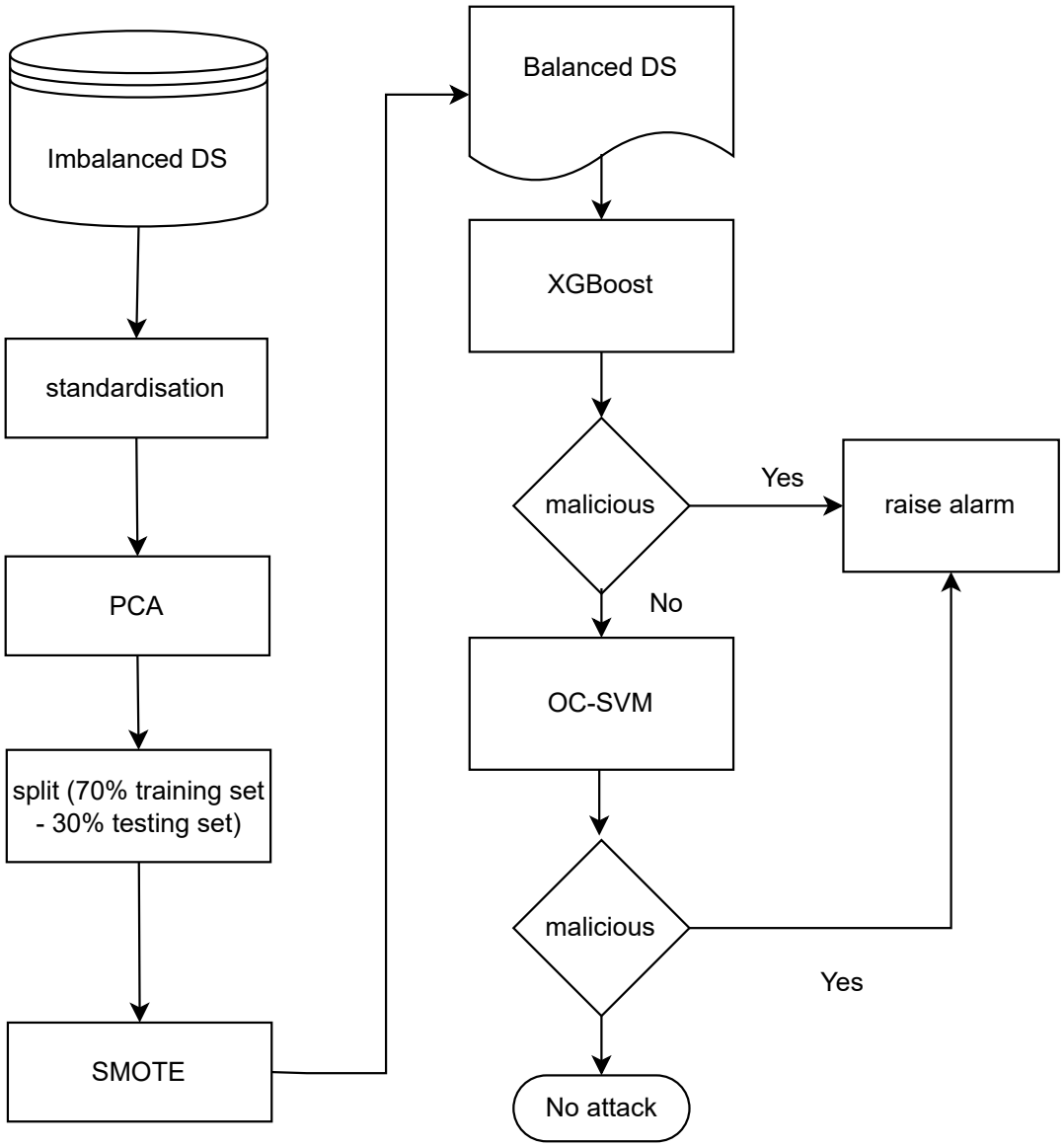

FIGURE 4.5: Hybrid IDS flowchart

4.4.2 Evaluation on insider scan

In this work, we conduct two separate experiments, using APT-SDNdataset, on the main two ML detection techniques to find the most optimal technique from every type to be employed in the proposed model. This is followed by experiments on the proposed hybrid model over different network sizes.

- **Signature-based detection technique:** The most popular ML supervised-learning algorithms, used in Chapter 3, are evaluated using PCA technique. Table 4.3, presents the results over Logistic Regression, K-nearest neighbour, Decision Tree, Random Forest, Support Vector Machine and XGBoost (eXtreme Gradient Boosting). All experiments use the $h = 4$ dataset. XGBoost outperforms all other techniques. Consequently, it is proposed as the classifier for the signature-based detection.

- **Anomaly-based detection techniques:** The most popular One-Class classification (ML-based anomaly detection technique when we have just two classes *normal* and *abnormal*) in Machine Learning are One-class SVM, Isolation Forest (IF) and Local Outlier Factor (LOF) [76]. These are evaluated over the dataset when $h = 4$. One-Class SVM has far better results than others. Although, the Isolation Forest shows better results than the Local Outlier Factor, both suffer from high numbers of false positives and false negatives compared to One-Class SVM, as shown in the confusion matrices in Figure 4.6. The highest results are from the OC-SVM (One-Class SVM) and so we propose it as the anomaly detector in the proposed scheme. Table 4.4 compares these techniques.

- **Hybrid Intrusion Detection System:** The proposed Hybrid NIDS employs XGBoost and OC-SVM and is evaluated over the three network sizes ($h = 4$, $h = 8$, and $h = 16$). The results are shown in Table 4.5. The proposed model scales well as the network size is increased. Compared to the results of the standard anomaly detection techniques, as shown in table 4.4, the proposed model outperforms all these standard techniques.

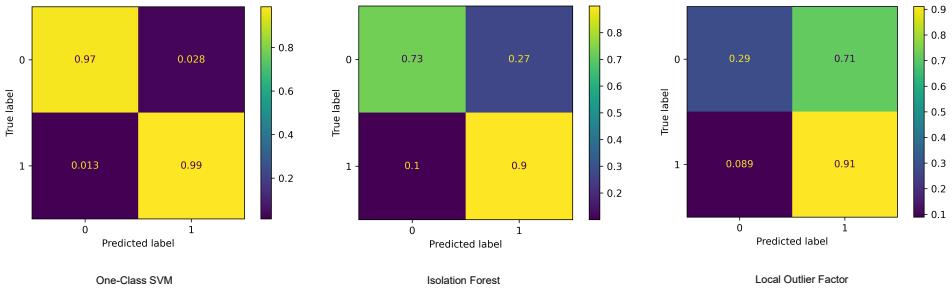

FIGURE 4.6: Confusion matrix for OC-SVM, IF, and LOF

TABLE 4.3: Supervised-learning results on APT-SDNmap dataset

Technique	Accuracy	Recall	Precision	F1-score
Logistic Regression	0.96	0.36	0.86	0.51
K-nearest neighbour	0.99	0.94	0.99	0.96
Decision Tree	0.99	0.85	0.98	0.91
Random Forest	0.99	0.90	0.99	0.94
Support Vector Machine	0.98	0.80	0.86	0.83
XGBoost	0.99	0.96	0.99	0.97

TABLE 4.4: Anomaly-based detection techniques comparison on APT-SDNmap dataset

Tech	Accuracy	Recall	Precision	F1
One-Class SVM	0.98	0.97	0.81	0.88
Isolation Forest	0.89	0.75	0.30	0.43
Local Outlier Factor	0.87	0.28	0.16	0.20

TABLE 4.5: Hybrid NIDS experimental results (using the APT-SDNmap dataset)

No. of hosts	Accuracy	Recall	Precision	F1
4	0.98	0.98	0.90	0.94
8	0.98	0.98	0.91	0.94
16	0.98	0.98	0.88	0.92

4.4.3 Detecting externals

APT attacks involve multiple stages, as discussed in section 2.3.2, starting from reconnaissance and scanning and ending with the aim of the attack, such as getting the gathered information or damaging the target or even maintaining access for further benefits. The communication between the attacker and the target is critical. In isolated networks, with the help of the insider, the attacker can deliver the malware and get the gathered information. When the network is accessible to the public, the attacker can use a number of tools for the attack. This way of conducting an attack is the most common because getting inside the victim's network (being an insider) is challenging for attackers. In this case, the attacker is described as external to distinguish them from the insider (internal).

In the previous section, an evaluation of the system is illustrated by using the APT-SDNmap dataset, which includes just insider attacks (scanning). In the following, the experiments were conducted over different datasets representing external attacks as well as internal attacks.

1. Experiment 1: The InSDN dataset has two sources of attacks: insiders launching attacks from two hosts inside the victim network and an external source (which forms the majority of the attacks in the dataset). The results show high scores, with 0.97, 0.99, 0.96, and 0.98 in Accuracy, Recall, Precision and F1, respectively.

2. Experiment 2: DAPT 2020 is a recent APT dataset that contains both internal and external threats. It mimics APTs and includes all its stages. The experiments demonstrated a score greater than 0.81 in Accuracy, Recall, Precision and F1.

3. Experiment 3: In this experiment, the proposed model is evaluated using one of the state-of-the-art datasets in the intrusion detection systems. CICIDS 2017 (discussed in detail in section 2.10) is used in the evaluation. It has various attacks (internal and external), carried out over five days. The results are 0.99 in Accuracy and Recall, 0.96 in Precision and 0.97 in F1.

TABLE 4.6: Hybrid NIDS experimental results over different datasets

Dataset	Accuracy	Recall	Precision	F1
InSDN (all attacks)	0.97	0.99	0.96	0.98
DAPT 2020	0.81	0.81	0.81	0.81
CICIDS 2017	0.99	0.99	0.96	0.97

Table 4.6 summarises the evaluation results of the hybrid model over different datasets.

4.5 Summary

Detecting stealth attacks in Software-Defined Networks is a critical security endeavour for the community. The insider can make it easier for stealth attackers to deliver malicious software. In this chapter, the internal and external perspectives of APTs are discussed. A hybrid Network Intrusion Detection System is proposed to detect stealth scans that aim to reconstruct flow rules in SDN. In addition, the detection of attacks carried out by externals is addressed. It can detect known attacks and those unseen previously. The proposed hybrid model overcomes the limitations of the individual techniques improving the detection methodology and reducing erroneous decisions. XGBoost and OC-SVM are implemented to detect known and unknown attacks, respectively. The system is evaluated over insider attacks and external attacks on different datasets, showing promising results.

Chapter 5

An Incremental Adaptive Network-based IDS

In Chapter 4, it is shown that adopting an anomaly-based detection technique delivers significant results in detecting unknown attacks. The combination of signature-based and anomaly-based detection techniques shows an improvement in the model's performance. However, ML classifiers are challenged when the data distribution changes. *Concept drift* is a term that describes the change in the relationship between the input data and the target value (label or class). The model is expected to degrade as certain forms of change occur. For us, the primary form of change will be in user behaviour (particularly changes in attacker behaviour). It is essential for a model to adapt itself to deviations in data distribution. SDN can help in monitoring changes in data distribution. This chapter discusses changes in stealth attacker behaviour and investigates the use of various concept drift detection algorithms. An incremental adaptive NIDS is proposed to tackle the issue of concept drift in SDN.

This chapter investigates the third research question, as stated in section 1.2, RQ3: *Can we continue to detect stealth attacks in SDN networks as adversaries change their behaviour?*

5.1 Introduction

One of the main challenges in machine learning-based intrusion detection systems is that the environment from which data is sampled changes over time. That is, the distribution of the data to be classified changes. Such change may occur, for

example, as a result of a change in the behaviour of both benign users and attackers. Concept drift is the change, for any reason, in the relationship between the input data and the predicted output over time. If the drift is not accounted for, the results of a predictive ML model will deteriorate [6]. In this case, there is a need for the IDS to adapt itself to any change by training over new data; it cannot simply depend on old data. More details are given in section 2.8.

5.1.1 Motivation and contributions

Traditional machine learning techniques are static and trained over specific batches of data. The model is built based on the behaviour of the users (normal or malicious) at that time of training. Over time, when the data distribution changes, the model performance will degrade unless it is trained again over newer data. The change can happen by changing the attack behaviour, such as employing a different attacking tool or launching a new type of attack. Updating the classifier regularly is costly and sometimes not feasible. An optimal approach, or one at least currently giving good chances of success, is to make the classifier adapt itself incrementally when a new intrusion emerges or when concept drift occurs for other reasons. The global visibility helps in monitoring changes in data distribution. This is because obtaining network statistics is easier in SDN than in legacy networks. Adaptive learning is the method of updating the predictive model as a reaction to concept drift [84]. In Chapter 4, the experiments show that combining two detection techniques helps to improve system performance. In this chapter, an incremental adaptive hybrid network intrusion detection system to detect previously unseen attacks is proposed.

Adaptive Random Forest (ARF) [182] and Adaptive One-class Support Vector Machine (Adaptive One-class SVM) [183] are implemented for incremental signature-based and incremental anomaly-based detection, respectively. An Adaptive Random Forest has the ability to detect a reasonable number of unseen data. However, stealth attacks are still difficult to detect. Anomaly-based detection classifiers are believed to be the most effective technique for detecting such attacks [181]. An Adaptive One-Class Support Vector Machine is employed in our proposed hybrid system to be the anomaly-based detection module. The contributions of this chapter are summarised as follows:

1. Proposal of the first incremental adaptive hybrid IDS to overcome concept drift caused by stealth attackers in SDN.

2. Determination of the optimal parameters using hyper-parameter tuning.

3. Evaluation of the most popular concept drift detection techniques.

4. Evaluation of the proposed system over different datasets, reflecting various attacks and scenarios (including known and unknown attacks).

5.1.2 Chapter organisation

The rest of this chapter is organised as follows. Section 5.2 summarises the capabilities of existing proposed approaches. Section 5.3 discusses an aspect of APT adaptivity. An explanation of the proposed model is given in section 5.4. Evaluation results are presented in section 5.5. A summary is given in section 5.6.

5.2 Related Works

Adaptive approaches have been brought to bear for intrusion detection in non-SDN environments. An ensemble classifier has been proposed to detect concept drift caused by the existence of new intrusions [184]. *HDDM* [80] is used as the concept drift detector and the system scores 94.91% accuracy. Another ensemble classifier [185] uses Hoeffding Adaptive Tree and Adaptive Random Forest to detect network attacks and uses ADWIN for concept drift. A network-based IDS uses a Page-Hinkley test (PHT) to detect concept drift to improve a deep learning classifier [186]. The classifier has high accuracy (99%) and is applied on the CICIDS2017 dataset which has just traditional attacks. (The dataset is described in section 2.10.) The model is not generalised as it is trained over the dataset without removing the features that could cause over-fitting. A stream learning IDS [187] is proposed for detecting concept drift. CIDD-ADODNN [188] uses Adadelta optimiser-based deep neural networks (ADODNN) to classify imbalanced data. The proposal used anomaly-based detection and was evaluated over three different datasets. Optimised Adaptive and Sliding Windowing (OASW) [189] is a window-based drift detection technique proposed for Internet of Things systems. The authors proposed

TABLE 5.1: Related works Comparison

Scheme	SDN	Stealth	Accuracy*	Technique	Dataset
[184]	×	×	0.94	HDDM	KDD Cup 99
[186]	×	×	0.99	PHT	CICIDS2017
[187]	×	×	0.96	bespoke	Fine-grained Intrusion Dataset (FGD) [191]
[188]	×	×	0.95 0.93 0.76	ADWIN	NSL-KDD Spam [192] Chess[193]
[185]	×	×	-	ADWIN	KDD Cup 99
[189]	×	×	0.99 0.98	ADWIN	IoTID20 [190] NSL-KDD
Proposed model	✓	✓	0.99 0.98 0.99 0.94	ADWIN	APT-SDNdataset DAPT 2020 CICIDS 2017 InSDN

* Accuracy column presents scores as given in the original sources (Scheme column)

an optimisation for LightGBM to detect anomalies in such systems. It is evaluated over the IoTID20 [190] and NSL-KDD datasets. The accuracy results are 0.99 on IoTID20 and 0.98 over the NSL-KDD dataset. These schemes are not proposed to detect stealth attacks or work on SDN. Table 5.1 compares the aforementioned studies and the proposed model.

5.3 APT Adaptivity

Two cases may cause concept drift when the aim is to detect APTs. These are described below.

5.3.1 Case 1: Natural changes in any network

Common system evolution may incur changes in the user population, the specific services produced, the platform such services are deployed on and the uses of such services, and so on. Thus, the data distribution of 'normal' behaviour may clearly change over time. The deviation in the data distribution (a.k.a. concept drift) causes the ML classifier prediction to degrade.

5.3.2 Case 2: Attack characteristics

APTs may continually adapt their behaviour so that they maintain sufficient similarities with normal behaviours to remain undetected [181]. They may stay dormant for a period of time or employ sleep functions between their activities (as discussed in 3.3). Moreover, the majority of these attacks go through multiple stages and at every stage, they have one or more tools that may create new classes of attack. New patterns may emerge as a result of these changes. [158; 2; 194]. It is essential to retrain the ML-based NIDS to track the concept drift and make the model more adaptive to any changes in data distribution.

5.4 Proposed Scheme

Given a period of time [0,t], streaming data arrives as a sequence of packets $K_{0,t} = \{s_0, s_1, s_2, ..., s_n\}$, where $s_i = (X_i, y_i)$ is one sample. X_i is a vector of independent features which can be labelled appropriately with $y_i \in \{0, 1\}$. Let $P_t(X, y)$ represent the *joint distribution* of X and y at time t. (It is a probability density function.) Concept drift happens when $P_t(X, y)$ changes over time. Thus, it occurs between two time points t_0 and t_1 if $\exists X \cdot P_{t0}(X, y) \neq P_{t1}(X, y)$ [84]. This is defined in terms of the *data* sampled. For example, if a new attack is developed after time t_0 and launched, corresponding to X' say, then $P_{t0}(X', 1) = 0$ (the attack never occurs in the data and so has zero density) but $P_{t1}(X', 1) \neq 0$ (the attack does occur and so has non-zero density). Technically, a specific vector of features X could in some circumstances correspond to malicious action but be benign in others, though often the assignation will be unequivocal. ML-based models that seek to predict y from X must adapt to ensure continued high performance.

The proposed hybrid system combines signature-based and anomaly-based detection techniques using Adaptive Random Forest (ARF) and adaptive One-Class SVM. The model monitors networks through the SDN controller. Figure 5.1 gives some details of the detection model. Every instance that arrives at the system is, firstly, examined by the signature-based ARF module (discussed in section 5.4.1). If the module classifies that instance as a known attack, then there is no need for further investigation by the anomaly-based detection module. If the packet is classified

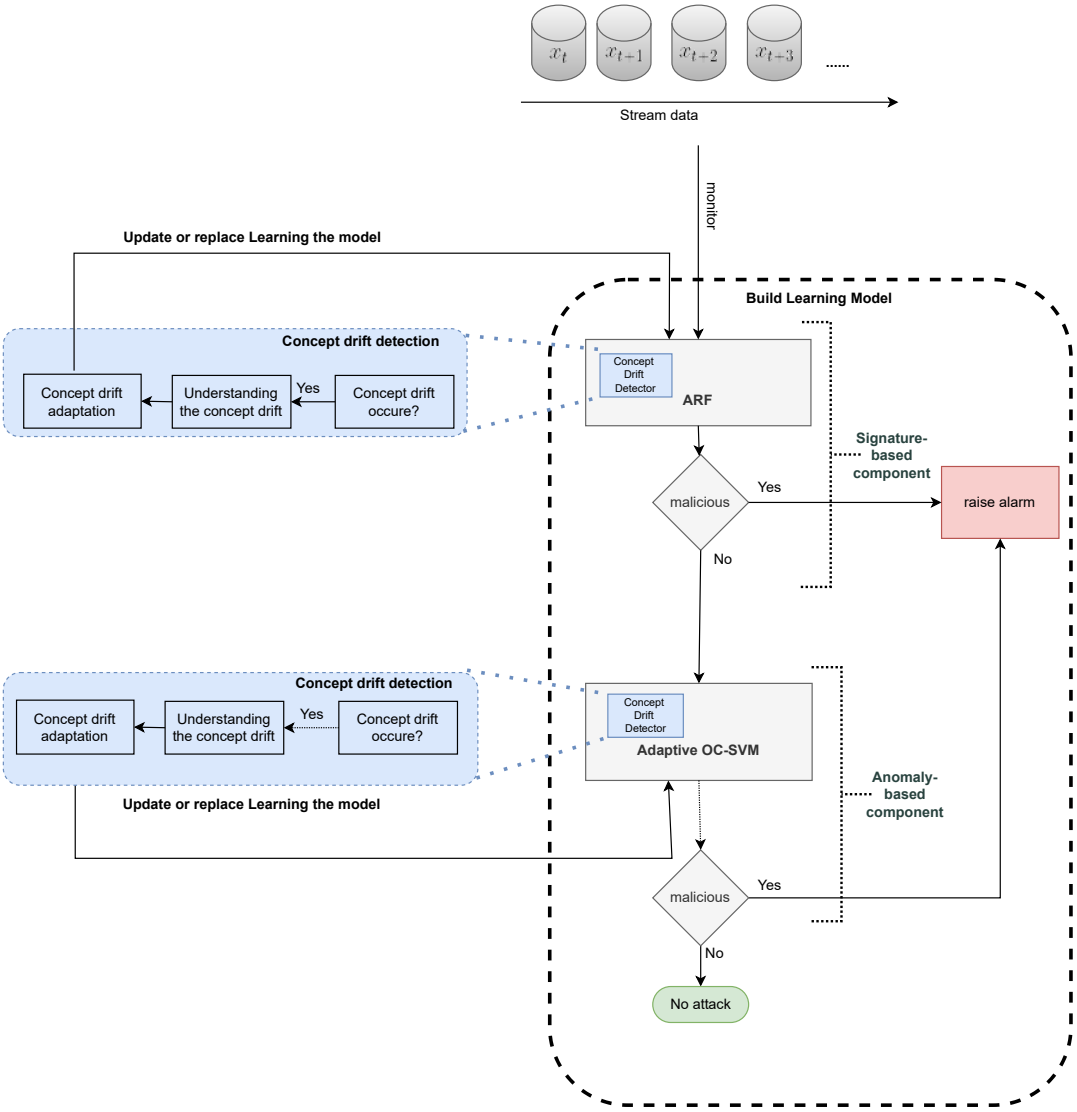

FIGURE 5.1: The proposed adaptive hybrid model

as benign by the signature-based module, it is passed to the anomaly-based detection module (Adaptive One-class SVM) (discussed in section 5.4.2) to check if it is an unknown attack. The administrator is notified if the signature or anomaly-based module flags an instance as an attack. The concept drift is detected using ADWIN, a concept drift detector, in both classifiers, and the model is updated. The model can detect known and unknown attacks. Both are incremental and can adapt themselves against concept drifts.

5.4.1 Signature-based detection module

The first phase of detection is the signature-based detection module. An Adaptive Random Forest (ARF) is the classifier used in this phase. ARF is a classification

TABLE 5.2: Hyperparameter tuning

parameter	description	optimal value
AdaptiveRandomForestClassifier (ARF)		
max_features	Maximum number of features in every single run	5
n_estimators	Number of trees	7
drift_detection_method	concept drift detection technique	ADWIN
Adaptive One-Class SVM (OC-SVM)		
ν	This controls the fraction of outliers in the system	0.2
Gamma	Kernel coefficient	0.9

algorithm for evolving data streams. The classifier is trained over labelled data to learn the pattern of malicious and normal activities. In addition, it is an incremental algorithm that can adapt itself against concept drift. The ARF uses a drift detector in every tree, which monitors drifts and warnings. In the case of a warning, the trees in the background will train new trees. If a drift is detected, the primary module will be replaced by the trained trees.

5.4.2 Anomaly-based detection module

Adaptive One-class SVM is an anomaly detection algorithm. The classifier is trained over the normal data and tested over both normal and malicious samples. During the training, it develops a profile of normal traffic. If the incoming traffic differs from the normal data that it is trained on, it is considered an attack. It also adapts itself incrementally when data distribution changes. The new data points are compared to the existing model. A change in the distribution of the data points indicates a concept drift is happening. SVMs and their variants have been shown to be highly effective classifiers across many domains.

5.5 Evaluation and Results

APT campaigns go through multiple stages and different kinds of attacks are involved in every stage. Moreover, there is no one pattern for APT attacks [2]. In our work, a number of experiments were conducted on different datasets to show the

effectiveness of the proposed model. These datasets include different attacks and scenarios. In all experiments, various evaluation measures are recorded: *Accuracy, Recall, Precision, and F1-score*.

1. **Model evaluation over stealth SDN-based dataset:** Two experiments are conducted to compare the concept drift detection techniques. APT-SDNdataset (more details are given in section 3.5), is used in both experiments. The dataset has a probe scan that reconstructs flow rules in a stealthy manner. The first experiment is a comparison between error rate-based drift detection techniques (discussed in section 2.8.2.1). These techniques are ADWIN, DDM, EDDM, HDDM_A, HDDM_W, PageHinkle, and KSWIN. The results are shown in Table 5.3. All error-based detection techniques give very similar results.

   The second experiment uses the data distribution-based drift detection technique (discussed in section 2.8.2.2). The kdq-tree detection is the algorithm implemented in the experiment. The results are 0.99 in Accuracy and 0.96 in Recall, Precision, and F1.

   Figure 5.2 presents how the error-based detection algorithm (using ADWIN in the comparison) is higher in all metrics compared to kdq-tree. Precision (which affects F1) is much lower in kdq-tree, compared to ADWIN results, due to the high percentage of false positives.

   ADWIN will be used on the model as the concept drift detection technique. Because most error-based concept drift techniques are very similar but it has more flexibility than others as discussed in 2.8.2.1. In addition, it gives better results than the distribution-based concept drift algorithm (e.i. kdq-tree).

2. **Model evaluation over SDN-based dataset:** For further evaluation on another dataset, the proposed model is evaluated over InSDN (further discussed in section 2.10). The dataset contains several attacks, making it suitable to evaluate the changes in the attacker's behaviour (concept drift). Another main advantage of using InSDN to evaluate this work is that the dataset is SDN-based traffic. Firstly, the model is evaluated using an error-based concept drift detection technique. ADWIN is used in the experiment, scoring 0.94, 1, 093, and 0.96 in Accuracy, Recall, Precision and F1 respectively. Another experiment is

TABLE 5.3: Results for error rate-based drift detection (over APT-SDNmap dataset)

Tech	Accuracy	Recall	Precision	F1
ADWIN	0.99	0.99	0.95	0.97
DDM	0.99	0.97	0.96	0.97
EDDM	0.99	0.99	0.94	0.96
HDDM_A	0.99	0.99	0.95	0.97
HDDM_W	0.99	0.99	0.95	0.97
PageHinkle	0.99	0.99	0.95	0.97
KSWIN	0.99	0.99	0.95	0.97

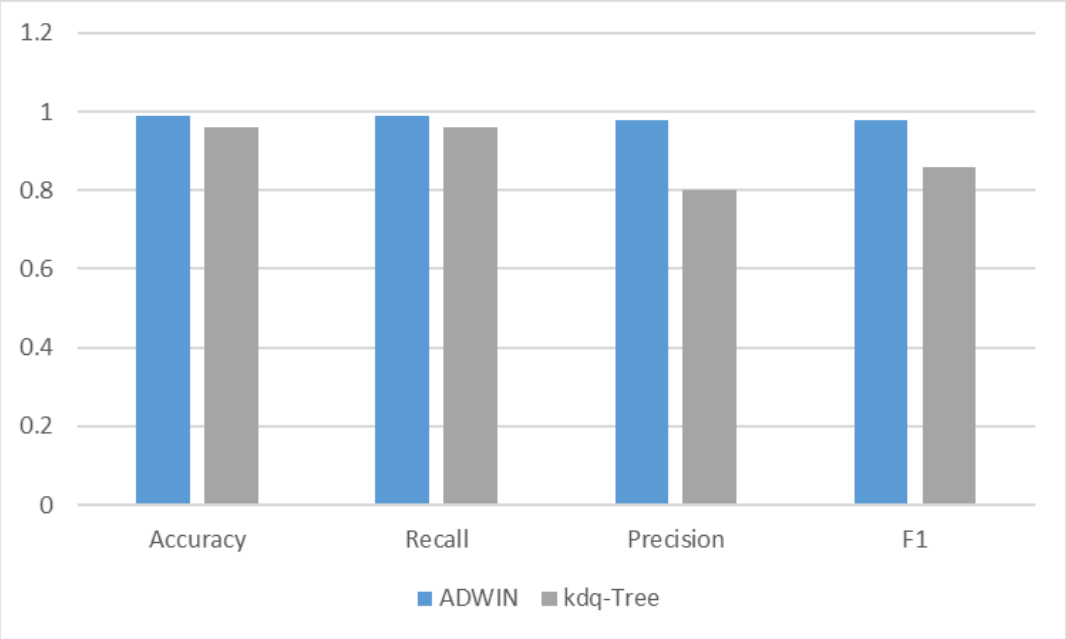

FIGURE 5.2: Error-based ADWIN and distribution-based kdq-tree comparison

conducted using the distribution-based technique. kdq-tree is the algorithm used in the experiment, and the results are 0.94 in Accuracy, 0.99 in Recall, 0.93 in Precision and 0.96 in F1. The results show significant performance in detecting actual attacks correctly (True Positives). But this came with a price, as it classified some normal instances as attacks (False Positives).

3. **Model evaluation over APT dataset:** The APT-based DAPT 2020 dataset is used to evaluate the proposed model. The approach of conducting normal traffic and various attacks over different days makes it very useful for evaluating the model against APTs and concept drift issues. More detail about the dataset is given in section 2.10. Figure 5.4 visualises the attack scenario. The dataset is prepared by removing columns that can affect the system's performance. Flow ID, Source IP, Source Port, Destination IP, Destination Port and Timestamp are the features that are removed from the dataset. Null values are replaced by zeros, and labels are converted to (0) for benign and (1) for malicious as the system is a binary classifier. The dataset is scaled using scikit-learn's StandardScaler [171]. The system gives significant results for detecting APTs, scoring 0.98 in Accuracy, Recall, Precision and F1. Another experiment is conducted to evaluate the implementation of the distribution-based detection technique. kdq-tree is the algorithm used in the experiment scoring 0.83, 0.80, 0.84, 0.82 in Accuracy, Recall, Precision and F1, respectively. The results are not significant, but the technique has advantages as it is unsupervised.

   Figure 5.3 shows a comparison between using the hybrid detection model in Chapter 4 and the model in this chapter after implementing the adaptivity.

4. **Model evaluation over traditional attacks:** The CICIDS 2017 dataset is used in this experiment. The dataset details and attack scenarios are discussed in section 2.10 (also visualised in Figure 5.5). It has a large amount of data recorded over five days with various attacks. As shown in Table 2.8, the first day is just normal traffic, but the following days have different attacks every day. This is a suitable scenario for concept drift as the attacker changes their behaviour over time. It is not as stealthy as DAPT 2020. Attacks make up only around

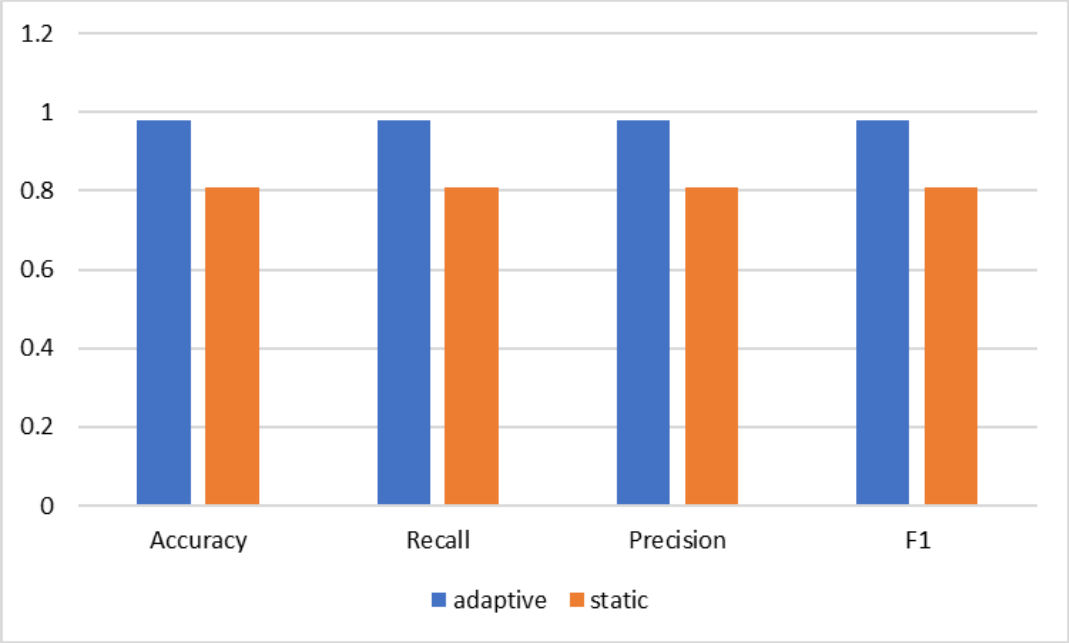

FIGURE 5.3: A comparison on the evaluation of static and dynamic hybrid models over DAPT 2020 dataset

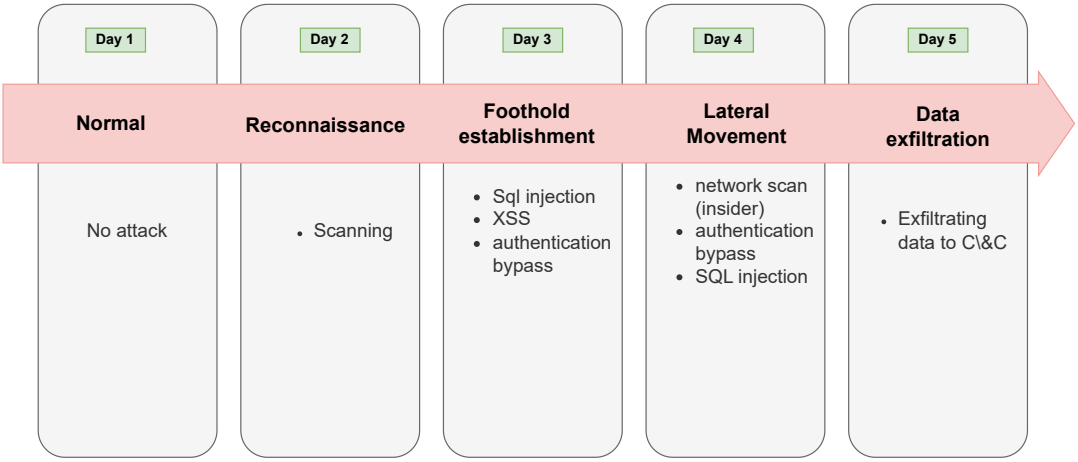

FIGURE 5.4: Attacks details on DAPT 2020 dataset

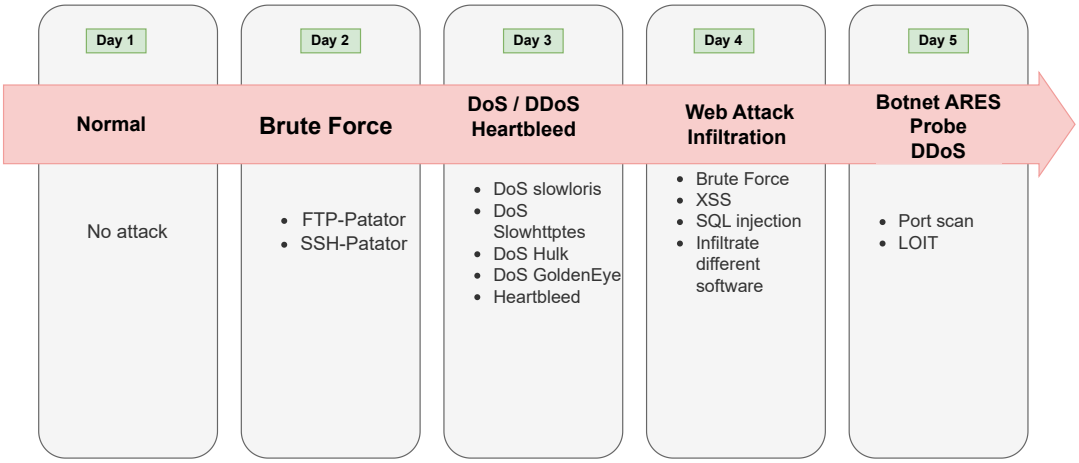

FIGURE 5.5: Attacks details on CICIDS 2017 dataset

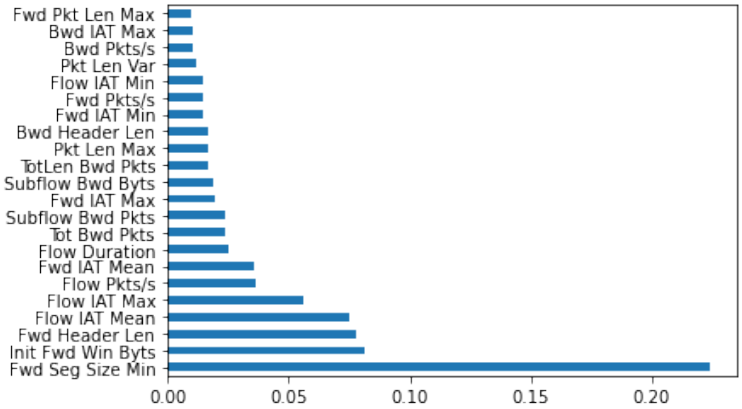

FIGURE 5.6: CICIDS 2017 feature importance

5% of the dataset. In this work, the dataset is prepared by removing all features that may expose the identity of the sender/receiver or cause over-fitting to the system. The removed features are Flow ID, Source IP, Source Port, Destination IP, Destination Port and Timestamp. Feature importance technique is applied using Random Forest technique [172]. Just seven features, the most important of which are shown in Figure 5.6, were selected and applied. The proposed system is a binary classifier. All attacks are converted to one class (1) and all normal records are assigned (0). In the first experiment, ADWIN is used as the concept drift detector. The second experiment uses the concept drift distribution-based detector (kdq-tree). Significant results are shown in both experiments. Using ADWIN it scores 0.99 in all metrics. With kdq-tree, it is 0.98 in Accuracy, Recall and F1 and 0.99 in Precision.

TABLE 5.4: A summary of the evaluation results of the proposed model over different datasets

Technique	Dataset	Accuracy	Recall	Precision	F1
ADWIN	APT-SDNdataset	0.99	0.99	0.95	0.97
kdq-Tree	APT-SDNdataset	0.99	0.96	0.96	0.96
ADWIN	InSDN	0.94	1	0.93	0.96
kdq-Tree	InSDN	0.94	0.99	0.93	0.96
ADWIN	CICIDS 2017	0.99	0.99	0.99	0.99
kdq-Tree	CICIDS 2017	0.99	1	0.86	0.92
ADWIN	DAPT 2020	0.98	0.98	0.98	0.98
kdq-Tree	DAPT 2020	0.83	0.80	0.84	0.82

A summary of all results (using ADWIN and kdq-tree) over all datasets is given in Table 5.4

5.6 Summary

Machine learning model performance will typically degrade when the data distribution is changed over time. In this chapter, two cases of user behaviour changes are discussed. APTs usually change their behaviour, causing a change in data distribution. An incremental adaptive hybrid intrusion detection system is proposed in this work. The signature-based detection (implemented by Adaptive Random Forest) and anomaly-based detection model (implemented using Adaptive One-Class SVM) are combined to detect anomalies. It can detect known and unknown stealth attacks. The system adapts itself incrementally to changes in data distribution (concept drift). ADWIN is adopted in the proposed model to detect concept drift and adapt the detection models. The following is a summary of the model experiments (using ADWIN):

1. APT-SDNmap dataset: This dataset is created to represent the scanning phase of the APT's campaign in the SDN network. The evaluation of the system over this dataset shows high scores, but the classifier flags some normal instances as malicious (False Positives).

2. InSDN dataset: The SDN-based dataset contains different attacks. The various attacks in the dataset are conducted at different times, showing how the dataset

can be useful for evaluating concept drift detection in SDN. The results show detecting actual attacks accurately but classify some normal as attack.

3. DAPT 2020 dataset: A recent APT dataset was recorded over different days during the working week (Monday to Saturday). The scenario of conducting different attacks daily makes it a good example of the change in attacker behaviour (causing concept drift). Despite the stealthiness of the attack, the scores from the evaluation of the proposed model are very high.

4. CICIDS 2017 dataset: A benchmark dataset in the IDS field. The dataset includes various traditional attacks, in addition to normal user activities, over five sequence days. The results of the application of the proposed model are very high.

The diversity of datasets used in the evaluation and the high scores obtained shows how the proposed model can detect known and unseen attacks and adapt against concept drift. We recommend adapting security solutions (e.g. intrusion detection systems) incrementally. This can maintain the performance of the system even when a new attack has emerged or the attacker changed his strategy of attack. SDN helps in this approach by allowing gathering and monitoring data to be easier.

Chapter 6

Conclusions and Future Works

This thesis proposes three NIDSs to detect stealth attacks in SDN: a signature-based NIDS, a hybrid (signature-based and anomaly-based) NIDS, and an incremental adaptive NIDS (also hybrid). Each NIDS was developed in order to answer a research question posed in Chapter 1. Below we discuss each of those research questions and consider the evidence generated by our experiments.

6.1 Investigation of the Research Questions

Three research questions were investigated:

1. **Research Question 1 (RQ1):** Can we use a machine-learning and signature-based approach to detect the stealthy reconstruction of flow rules in SDN networks?

   The work to address this research question is in Chapter 3. Reconstructing flow rules is a major challenge in SDN. It is more critical when the attacker behaves like an APT. Very little research had been conducted so far on this issue. We developed and presented a stealth scanner that reconstructs flow rules in SDN, implementing advanced scanning behaviour in victim networks. The enhancement of the scanner itself is an original contribution and has been made publicly available. We developed various candidate networks and emulated both normal and scanning behaviours. The resulting APT SDN focused datasets have also been made publicly available. Detecting stealth scans in SDN was investigated with the most common ML approaches, using supervised learning over various datasets. The proposed ensemble classifier XGBoost showed

the most promising results. A comprehensive ML pipeline was used throughout. Our model is the first application of ML to detect the stealthy scans of flow rule reconstruction attacks in SDN networks.

2. **Research Question 2 (RQ2):** Can we use a machine-learning and hybrid (signature-based plus anomaly-based) approach to detect the stealthy reconstruction of flow rules in SDN networks?

   This research question is addressed in Chapter 4. A hybrid model is proposed combining signature-based and anomaly-based approaches (XGBoost and One-Class SVM, respectively). The approach was evaluated over various datasets, including different attacks (insider and external) and known and unseen attacks. Combining the two detection approaches improves detection performance. This is the first demonstration of the power of a hybrid IDS for the detection of stealth attacks in SDN networks.

3. **Research Question 3 (RQ3):** Can we continue to detect stealth attacks in SDN networks as adversaries change their behaviour?

   An incremental adaptive network intrusion detection system is proposed in Chapter 5 to address this question. The proposed scheme can incrementally adapt itself when the attacker's behaviour changes. Error-based and distribution-based concept drift techniques are investigated and evaluated over different datasets. Adaptive Random Forest and Adaptive One-Class SVM are proposed to detect known and unknown attacks respectively. Both are employed in a hybrid system. Results show that adapting the detection models enhances the detection of APTs.

We used beneficially various ML pipeline components, particularly feature engineering (including the use of importance measures), re-sampling regimes to handle imbalanced datasets (which inevitably arise in intrusion detection), and hyper-parameterisation. These are generally accepted techniques that can improve model performance and enhance the generalisation of models. The current literature on IDS frequently omits these important techniques.

6.2 Future work

Further research might usefully be conducted into the following:

1. A comprehensive hybrid system: The proposed models, in this thesis, focus on network flow-based information. System events such as log files in network devices, however, can help in the detection. Building a hybrid detection system takes advantage of the network flow and local files and can improve detection results. Thus, we propose combining HIDS and NIDS.

2. Engineer new features from the available APTs datasets: DAPT 2020 is the only available APT dataset (other than our APT-SDNmap dataset). In DAPT 2020, the raw files (.pacp) that were used in the experiments to generate the dataset are available. The features in DAPT 2020 are engineered using the network traffic features generator CICFlowMeter. It would be useful to investigate APTs' specific features to enhance the contributions of the features to the detection models. Developing a feature extractor that considers stealth attacks can help to deliver similar features from any available (in the future) stealth dataset.

3. Reconstruct flow rules in P4 (Programming Protocol-independent Packet Processors): SDNmap and the extension APT-SDNmap are based on the OpenFlow protocol. Extending the work presented in this thesis to address corresponding flow rule reconstruction under the P4 regime would greatly widen applicability.

4. Real experiments: Our work is carried out on data generated under network emulation. Implementing our proposed models on real devices with different real networks is an obvious means to further validate the techniques we have investigated. In addition, we could test the system on actual APT malware to evaluate the approaches against unknown attacks in real environments.

5. Investigate the detection of APTs in Industrial Control Systems (ICS): APTs have the potential to cause significant damage in industrial control systems. Indeed, Stuxnet, targeting nuclear reprocessing systems, is an APT and is one

of the highest profile malware applications of all time [42]. The proposed models can be evaluated on industrial networks.

6.3 Finally

The detection of stealthy attacks in SDN networks is a critical area for research. We recommend it to the SDN, IDS, and ML communities.

Bibliography

[1] "What is p4." [Online]. Available: https://plvision.eu/expertise/sdn-nfv/p4

[2] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1851–1877, 2019.

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

[4] J. C. Pickles, T. J. Stone, and T. S. Jacques, "Methylation-based algorithms for diagnosis: experience from neuro-oncology," *The Journal of Pathology*, vol. 250, no. 5, pp. 510–517, 2020.

[5] "Introduction to xgboost in python." [Online]. Available: https://blog.quantinsti.com/xgboost-python/

[6] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[7] P. Chen, L. Desmet, and C. Huygens, "A study on advanced persistent threats," in *IFIP International Conference on Communications and Multimedia Security*. Springer, 2014, pp. 63–72.

[8] S. Achleitner, T. La Porta, T. Jaeger, and P. McDaniel, "Adversarial network forensics in software defined networking," in *Proceedings of the Symposium on SDN Research*, 2017, pp. 8–20.

[9] "Software-defined networking (sdn) market." [Online]. Available: https://www.marketresearchfuture.com/reports/software-defined-networking-market-1607

[10] "Software-defined networking (sdn) market." [Online]. Available: https://www.marketresearchfuture.com/

[11] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[12] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2015.

[13] I. Alsmadi and D. Xu, "Security of software defined networks: A survey," *Computers & security*, vol. 53, pp. 79–108, 2015.

[14] "Sdn architecture." [Online]. Available: https://www.opennetworking.org/wp\protect\discretionary{\char\hyphenchar\font}{}{}content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf

[15] A. Hakiri and P. Berthou, "Leveraging sdn for the 5g networks: Trends, prospects and challenges," *arXiv preprint arXiv:1506.02876*, 2015.

[16] B. Görkemli, S. Tatlıcıoğlu, A. M. Tekalp, S. Civanlar, and E. Lokman, "Dynamic control plane for sdn at scale," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2688–2701, 2018.

[17] A. Nayyar, B. Singla, and P. Nagrath, *Software Defined Networks: Architecture and Applications*. Scrivener Publishing LLC, 2022.

[18] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE communications surveys & tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014.

[19] I. Hassani and A. Moayeri, "Sdn security: A survey," *International Journal of Information, Security and Systems Management*, vol. 7, no. 1, pp. 785–792, 2018.

[20] "P4." [Online]. Available: https://opennetworking.org/p4/

[21] A. Liatifis, P. Sarigiannidis, V. Argyriou, and T. Lagkas, "Advancing sdn from openflow to p4: A survey," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–37, 2023.

[22] "Open networking foundation." [Online]. Available: https://www.opennetworking.org

[23] K. B. Nougnanke, "Towards ml-based management of software-defined networks," Ph.D. dissertation, Université Paul Sabatier-Toulouse III, 2021.

[24] S. Bian, P. Zhang, and Z. Yan, "A survey on software-defined networking security," in *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, 2016, pp. 190–198.

[25] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[26] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 165–166.

[27] A. Shaghaghi, M. A. Kaafar, R. Buyya, and S. Jha, "Software-defined network (sdn) data plane security: issues, solutions, and future directions," *Handbook of Computer Networks and Cyber Security*, pp. 341–387, 2020.

[28] S. Liu, M. K. Reiter, and V. Sekar, "Flow reconnaissance via timing attacks on sdn switches," in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017, pp. 196–206.

[29] R. Kandoi and M. Antikainen, "Denial-of-service attacks in openflow sdn networks," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2015, pp. 1322–1326.

[30] D. Hu, P. Hong, and Y. Chen, "Fadm: Ddos flooding attack detection and mitigation system in software-defined networking," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–7.

[31] "Openflowswitch," Sep 2022. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[32] Y. Liu, B. Zhao, P. Zhao, P. Fan, and H. Liu, "A survey: Typical security issues of software-defined networking," *China Communications*, vol. 16, no. 7, pp. 13–31, 2019.

[33] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[34] "Sdnmap." [Online]. Available: https://github.com/sdnmap

[35] D. S. Musa, "Advanced persistent threat-apt," 2014.

[36] "the non-advanced persistent threat - imperva." [Online]. Available: https://www.imperva.com/docs/HII_The_Non-Advanced_Persistent_Threat.pdf

[37] W. Symantec, "Advanced persistent threats: A symantec perspective," *Symantec World Headquarters*, 2011.

[38] S. A. TEAM *et al.*, "skywiper: A complex malware for targeted attacks," Budapest University of Technology and Economics Department of Telecommunications, Tech. Rep., 2012. [Online]. Available: https://www.crysys.hu/publications/files/skywiper.pdf

[39] B. Bencsáth, G. Pék, L. Buttyán, and M. Felegyhazi, "The cousins of stuxnet: Duqu, flame, and gauss," *Future Internet*, vol. 4, no. 4, pp. 971–1003, 2012.

[40] "A new approach to china." [Online]. Available: https://googleblog.blogspot.com/2010/01/new-approach-to-china.html

[41] R. Varma, "Mcafee labs: combating aurora," 2010.

[42] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, "Stuxnet under the microscope," *ESET LLC (September 2010)*, 2010.

[43] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2011, pp. 4490–4494.

[44] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, symantec corp., security response*, vol. 5, no. 6, p. 29, 2011.

[45] P. Mueller and B. Yadegari, "The stuxnet worm," *Département des sciences de linformatique, Université de lArizona. Recuperado de: https://www2. cs. arizona. edu/˜ collberg/Teaching/466-566/2012/Resources/presentations/topic9-final/report. pdf*, 2012.

[46] M. De Falco, "Stuxnet facts report: A technical and strategic analysis," *NATO Cooperative Cyber Defense Centre of Excellence*, 2012.

[47] Z. Dehlawi and N. Abokhodair, "Saudi arabia's response to cyber conflict: A case study of the shamoon malware incident," in *2013 IEEE International Conference on Intelligence and Security Informatics*. IEEE, 2013, pp. 73–75.

[48] C. Bronk and E. Tikk-Ringas, "Hack or attack? shamoon and the evolution of cyber conflict," *Online url: https://scholarship.rice.edu/handle/1911/92672*, 2013.

[49] "Fireeye." [Online]. Available: https://www.fireeye.com/

[50] "Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor." [Online]. Available: https://www.mandiant.com/

[51] "Solarwinds." [Online]. Available: https://www.solarwinds.com

[52] "Sunburst additional technical details." [Online]. Available: https://www.mandiant.com/resources/blog/sunburst-additional-technical-details

[53] "Suspected russian hack said to have gone undetected for months." [Online]. Available: https://www.wsj.com/articles/suspected-russian-hack-said-to-have-gone-undetected-for-months-11607974376?mod=tech_lead_pos3

[54] C. Five, "Advanced persistent threats: A decade in review," *Command Five PTY LTD*, pp. 1–13, 2011.

[55] B. Hudson, "Advanced persistent threats: Detection, protection and prevention," *Sophos Ltd., US February*, 2014.

[56] M. Ask, P. Bondarenko, J. E. Rekdal, A. Nordbø, P. Bloemerus, and D. Piatkivskyi, "Advanced persistent threat (apt) beyond the hype," *Project Report in IMT4582 Network Security at GjoviN University College*, vol. 2013, 2013.

[57] R. Brewer, "Advanced persistent threats: minimising the damage," *Network security*, vol. 2014, no. 4, pp. 5–9, 2014.

[58] L. Ertaul and M. Mousa, "Applying the kill chain and diamond models to microsoft advanced threat analytics," in *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018, pp. 252–258.

[59] R. Jasek, M. Kolarik, and T. Vymola, "Apt detection system using honeypots," in *Proceedings of the 13th International Conference on Applied Informatics and Communications (AIC'13), WSEAS Press*, 2013, pp. 25–29.

[60] "Mimikatz." [Online]. Available: https://github.com/gentilkiwi/mimikatz

[61] "Windows credential editor (wce)." [Online]. Available: https://www.ampliasecurity.com/research/windows-credentials-editor/

[62] M. Ussath, D. Jaeger, F. Cheng, and C. Meinel, "Advanced persistent threats: Behind the scenes," in *2016 Annual Conference on Information Science and Systems (CISS)*. IEEE, 2016, pp. 181–186.

[63] D. LeBlanc and M. Howard, *Writing secure code*. Pearson Education, 2002.

[64] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2015, pp. 45–56.

[65] A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion detection: A survey," in *Managing cyber threats*. Springer, 2005, pp. 19–78.

[66] J. R. Johnson and E. A. Hogan, "A graph analytic metric for mitigating advanced persistent threat," in *2013 IEEE International Conference on Intelligence and Security Informatics*. IEEE, 2013, pp. 129–133.

[67] Z. Saud and M. H. Islam, "Towards proactive detection of advanced persistent threat (apt) attacks using honeypots," in *Proceedings of the 8th International Conference on Security of Information and Networks*, 2015, pp. 154–157.

[68] C. Lei, H.-Q. Zhang, J.-L. Tan, Y.-C. Zhang, and X.-H. Liu, "Moving target defense techniques: A survey," *Security and Communication Networks*, vol. 2018, 2018.

[69] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Machine learning and deep learning approaches for cybersecuriy: A review," *IEEE Access*, 2022.

[70] J. Brownlee, *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch, 2016*. online, 2016. [Online]. Available: https://machinelearningmastery.com/master-machine-learning-algorithms/

[71] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[72] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," *Advances in neural information processing systems*, vol. 12, 1999.

[73] M. Kuhn, K. Johnson *et al.*, *Applied predictive modeling*. Springer, 2013, vol. 26.

[74] J. Brownlee, *Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python*. Machine Learning Mastery, 2020.

[75] M. Kuhn and K. Johnson, *Feature Engineering and Selection: A practical Approach for Predictive Models*. 6000 Broken Sound Parkway NW, Suite 300: Taylor & Francis Group, 2020.

[76] J. Brownlee, *Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning*. Machine Learning Mastery, 2020.

[77] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[78] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[79] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, 2006, pp. 77–86.

[80] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2014.

[81] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, vol. 416, pp. 340–351, 2020.

[82] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[83] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi, "An information-theoretic approach to detecting changes in multi-dimensional data streams," in *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

[84] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[85] C. Yoon, T. Park, S. Lee, H. Kang, S. Shin, and Z. Zhang, "Enabling security functions with sdn: A feasibility study," *Computer Networks*, vol. 85, pp. 19–35, 2015.

[86] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, 2019.

[87] G. A. Ajaeiya, N. Adalian, I. H. Elhajj, A. Kayssi, and A. Chehab, "Flow-based intrusion detection system for sdn," in *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2017, pp. 787–793.

[88] L. Zhang, G. Shou, Y. Hu, and Z. Guo, "Deployment of intrusion prevention system based on software defined networking," in *2013 15th IEEE International Conference on Communication Technology*. IEEE, 2013, pp. 26–31.

[89] A. Abubakar and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in *2017 seventh international conference on emerging security technologies (EST)*. IEEE, 2017, pp. 138–143.

[90] D. Jankowski and M. Amanowicz, "Intrusion detection in software defined networks with self-organized maps," *Journal of Telecommunications and Information Technology*, no. 4, pp. 3–9, 2015.

[91] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in sdn," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, 2013, pp. 487–488.

[92] T. Girdler and V. G. Vassilakis, "Implementing an intrusion detection and prevention system using software-defined networking: Defending against arp spoofing attacks and blacklisted mac addresses," *Computers & Electrical Engineering*, vol. 90, p. 106990, 2021.

[93] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 international conference on wireless networks and mobile communications (WINCOM)*. IEEE, 2016, pp. 258–263.

[94] P. Choobdar, M. Naderan, and M. Naderan, "Detection and multi-class classification of intrusion in software defined networks using stacked auto-encoders and cicids2017 dataset," *Wireless Personal Communications*, pp. 1–35, 2022.

[95] A. O. Alzahrani and M. J. Alenazi, "Ml-idsdn: Machine learning based intrusion detection system for software-defined network," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 1, p. e7438, 2023.

[96] G. F. Scaranti, L. F. Carvalho, S. B. Junior, J. Lloret, and M. L. Proença Jr, "Unsupervised online anomaly detection in software defined network environments," *Expert Systems with Applications*, vol. 191, p. 116225, 2022.

[97] J. Wang and L. Wang, "Sdn-defend: A lightweight online attack detection and mitigation system for ddos attacks in sdn," *Sensors*, vol. 22, no. 21, p. 8287, 2022.

[98] M. S. El Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A flow-based anomaly detection approach with feature selection method against ddos attacks in sdns," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1862–1880, 2022.

[99] K. Jia, C. Liu, Q. Liu, J. Wang, J. Liu, and F. Liu, "A lightweight ddos detection scheme under sdn context," *Cybersecurity*, vol. 5, no. 1, p. 27, 2022.

[100] J. Cao, M. Xu, Q. Li, K. Sun, and Y. Yang, "The attack: Overflowing sdn flow tables at a low rate," *IEEE/ACM Transactions on Networking*, 2022.

[101] D. Tang, X. Wang, Y. Yan, D. Zhang, and H. Zhao, "Adms: An online attack detection and mitigation system for ldos attacks via sdn," *Computer Communications*, vol. 181, pp. 454–471, 2022.

[102] P.-W. Chi, C.-T. Kuo, J.-W. Guo, and C.-L. Lei, "How to detect a compromised sdn switch," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–6.

[103] H. Zhou, C. Wu, C. Yang, P. Wang, Q. Yang, Z. Lu, and Q. Cheng, "Sdn-rdcd: A real-time and reliable method for detecting compromised sdn devices," *IEEE/ACM transactions on networking*, vol. 26, no. 5, pp. 2048–2061, 2018.

[104] A. Khalid, A. Zainal, M. A. Maarof, and F. A. Ghaleb, "Advanced persistent threat detection: A survey," in *2021 3rd International Cyber Resilience Conference (CRC)*. IEEE, 2021, pp. 1–6.

[105] J. M. Ceron, C. B. Margi, and L. Z. Granville, "Mars: An sdn-based malware analysis solution," in *2016 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2016, pp. 525–530.

[106] M. Ammar, M. Rizk, A. Abdel-Hamid, and A. K. Aboul-Seoud, "A framework for security enhancement in sdn-based datacenters," in *2016 8th IFIP international conference on new technologies, Mobility and security (NTMS)*. IEEE, 2016, pp. 1–4.

[107] J. Wu, K. Ota, M. Dong, and C. Li, "A hierarchical security framework for defending against sophisticated attacks on wireless sensor networks in smart cities," *IEEE Access*, vol. 4, pp. 416–424, 2016.

[108] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow world congress*, vol. 48. sn, 2012, pp. 1–16.

[109] Y. Ohsawa, "Modeling the process of chance discovery," in *Chance discovery*. Springer, 2003, pp. 2–15.

[110] D. Bhattacherjee *et al.*, "Stepping stone detection for tracing attack sources in software-defined networks," Master's thesis, Aalto University, 2016. [Online]. Available: http://urn.fi/URN:NBN:fi:aalto-201608263038

[111] T. OConnor, W. Enck, W. M. Petullo, and A. Verma, "Pivotwall: Sdn-based information flow control," in *Proceedings of the Symposium on SDN Research*, 2018, pp. 1–14.

[112] J. Shan-Shan and X. Ya-Bin, "The apt detection method in sdn," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, 2017, pp. 1240–1245.

[113] T. Shimanaka, R. Masuoka, and B. Hay, "Cyber deception architecture: Covert attack reconnaissance using a safe sdn approach," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[114] J. H. H. Jafarian, E. Al-Shaer, and Q. Duan, "Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers," in *Proceedings of the First ACM Workshop on Moving Target Defense*, 2014, pp. 69–78.

[115] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Adversary-aware ip address randomization for proactive agility against sophisticated attackers," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 738–746.

[116] Z. Zhao, F. Liu, and D. Gong, "An sdn-based fingerprint hopping method to prevent fingerprinting attacks," *Security and Communication Networks*, vol. 2017, 2017.

[117] S. Achleitner, T. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Cyber deception: Virtual networks to defend insider reconnaissance," in *Proceedings of the 8th ACM CCS international workshop on managing insider security threats*, 2016, pp. 57–68.

[118] W. Han, Z. Zhao, A. Doupé, and G.-J. Ahn, "Honeymix: Toward sdn-based intelligent honeynet," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2016, pp. 1–6.

[119] G. Zhao, K. Xu, L. Xu, and B. Wu, "Detecting apt malware infections based on malicious dns and traffic analysis," *IEEE access*, vol. 3, pp. 1132–1142, 2015.

[120] X. Wang, K. Zheng, X. Niu, B. Wu, and C. Wu, "Detection of command and control in advanced persistent threat based on independent access," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[121] "Defcon dataset." [Online]. Available: https://www.defcon.org/html/links/dc-ctf.html

[122] "Kdd cup 1999 data." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[123] "Nsl-kdd dataset." [Online]. Available: http://www.unb.ca/cic/datasets/nsl.html

[124] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[125] "ntrusion detection evaluation dataset (cic-ids2017)." [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[126] "Cicflowmeter." [Online]. Available: https://www.unb.ca/cic/research/applications.html#:~:text=CICFlowMeter%20is%20a%20network%20traffic%20flow%20generator%20and%20analyser.

[127] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "Insdn: A novel sdn intrusion dataset," *IEEE Access*, vol. 8, pp. 165 263–165 284, 2020.

[128] S. Myneni, A. Chowdhary, A. Sabur, S. Sengupta, G. Agrawal, D. Huang, and M. Kang, "Dapt 2020-constructing a benchmark dataset for advanced persistent threats," in *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 2020, pp. 138–163.

[129] Y. Ma and H. He, *Imbalanced Learning: Foundations, Algorithms, and Applications*. John Wiley & Sons, 2013.

[130] S. Molin, *Hands-On Data Analysis with Pandas*. Packt Publishing, 2019.

[131] "Apt-sdnmap," 2022, https://github.com/APT-SDNmap.

[132] Mininet, "An instant virtual network on your laptop (or other pc)." [Online]. Available: http://mininet.org/

[133] "Apt-sdn dataset." [Online]. Available: https://github.com/APT-SDNdataset

[134] "Cisco talos observes 'novel increase' in apt activity in q1." [Online]. Available: https://www.techtarget.com/searchsecurity/news/252516380/Cisco-Talos-observes-novel-increase-in-APT-activity-in-Q1

[135] "Cyber security statistics the ultimate list of stats data, & trends for 2022." [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/#:~:text=Advanced%20Persistent%20Threat%20(APT)%20Statistics&text=The%20advanced%20persistent%20threat%20(APT,of%20%249.6%20billion%20in%202026.

[136] S. Lee, J. Kim, S. Woo, C. Yoon, S. Scott-Hayward, V. Yegneswaran, P. Porras, and S. Shin, "A comprehensive security assessment framework for software-defined networks," *Computers & Security*, vol. 91, p. 101720, 2020.

[137] M. Conti, F. De Gaspari, and L. V. Mancini, "A novel stealthy attack to gather sdn configuration-information," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 328–340, 2018.

[138] J. Sonchack, A. Dubey, A. J. Aviv, J. M. Smith, and E. Keller, "Timing-based reconnaissance and defense in software-defined networks," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 89–100.

[139] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense," *Security and Communication Networks*, vol. 2018, 2018.

[140] M. Yu, T. He, P. McDaniel, and Q. K. Burke, "Flow table security in sdn: Adversarial reconnaissance and intelligent attacks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1519–1528.

[141] H. Cui, G. O. Karame, F. Klaedtke, and R. Bifulco, "On the fingerprinting of software-defined networks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 10, pp. 2160–2173, 2016.

[142] "Virusshare.com." [Online]. Available: https://virusshare.com/

[143] "malwaredomains." [Online]. Available: malwaredomains.com

[144] "Snort." [Online]. Available: https://www.snort.org/

[145] "Targeted cyberattacks logbook." [Online]. Available: https://apt.securelist.com/

[146] P. Chen, C. Huygens, L. Desmet, and W. Joosen, "Advanced or not? a comparative study of the use of anti-debugging and anti-vm techniques in generic and targeted malware," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2016, pp. 323–336.

[147] "Virustotal." [Online]. Available: https://www.virustotal.com/

[148] D. X. Cho and H. H. Nam, "A method of monitoring and detecting apt attacks based on unknown domains," *Procedia Computer Science*, vol. 150, pp. 316–323, 2019.

[149] CyberMonitor, "Apt & cybercriminals campaign collection." [Online]. Available: https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections

[150] "Deepend research: List of malware pcaps, samples, and indicators for the library of malware traffic patterns." [Online]. Available: https://contagiodump.blogspot.com/2013/08/deepend-research-list-of-malware-pcaps.html

[151] "Unb dataset." [Online]. Available: https://www.unb.ca/cic/datasets/index.html

[152] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[153] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2013, pp. 4487–4492.

[154] C. V. Neu, C. G. Tatsch, R. C. Lunardi, R. A. Michelin, A. M. Orozco, and A. F. Zorzo, "Lightweight ips for port scan in openflow sdn networks," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–6.

[155] T. A. Tang, L. Mhamdi, D. McLeron, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in sdn-based networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 202–206.

[156] E. M. Rudd, A. Rozsa, M. Günther, and T. E. Boult, "A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1145–1172, 2016.

[157] S. Stafford and J. Li, "Behavior-based worm detectors compared," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2010, pp. 38–57.

[158] T. Halabi, O. A. Wahab, R. Al Mallah, and M. Zulkernine, "Protecting the internet of vehicles against advanced persistent threats: a bayesian stackelberg game," *IEEE Transactions on Reliability*, vol. 70, no. 3, pp. 970–985, 2021.

[159] lastline, "An introduction to advanced malware and how it avoids detection," *lastline*, 2017.

[160] A. Singh and Z. Bu, "Hot knives through butter: Evading file-based sandboxes," *FireEye*, 2013.

[161] "New malware takes 'extended naps' to avoid detection." [Online]. Available: https://securityledger.com/2013/02/new-malware-takes-extended-naps-to-avoid-detection/

[162] S. S. Response, "Have i got newsforyou: Analysis of flamer c&c server," Symantec, Tech. Rep., 2012. [Online]. Available: https://paper.seebug.org/papers/APT/APT_CyberCriminal_Campagin/2012/w32_flamer_newsforyou.pdf

[163] C. Raiu, M. A. Hasbini, S. Belov, and S. Mineev, "From shamoon to stonedrill-wipers attacking saudi organizations and beyond," *Kaspersky Lab, March*, 2017.

[164] R. Falcone, "Shamoon 2: Return of the disttrack wiper," *Paloalto Networks*, 2016.

[165] Z. Ferrer and M. C. Ferrer, "In-depth analysis of hydraq," *The face of cyberwar enemies unfolds. ca isbu-isi white paper*, vol. 37, 2010.

[166] "Forensic methodology report: How to catch nso group's pegasus," Nov 2021. [Online]. Available: https://www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-nso-groups-pegasus/

[167] K. Beaver, *Hacking for dummies*. John Wiley & Sons, 2007.

[168] "Ostinato traffic generator for network engineers." [Online]. Available: https://ostinato.org/

[169] "sklearn." [Online]. Available: https://scikit-learn.org

[170] "Smote." [Online]. Available: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling\.SMOTE.html

[171] "Standardscaler." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.\StandardScaler.html

[172] "Feature importances with a forest of trees." [Online]. Available: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

[173] "Gridsearchcv." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection\.GridSearchCV.html

[174] "opendaylight." [Online]. Available: https://www.opendaylight.org/

[175] "Open network operating system (onos)." [Online]. Available: https://opennetworking.org/onos/

[176] "Ryu." [Online]. Available: https://ryu-sdn.org/

[177] M. Mamun and K. Shi, "Deeptaskapt: Insider apt detection using task-tree based deep learning," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 693–700.

[178] S. K. Dey and M. M. Rahman, "Flow based anomaly detection in software defined networking: A deep learning approach with feature selection method," in *2018 4th international conference on electrical engineering and information & communication technology (iCEEiCT)*. IEEE, 2018, pp. 630–635.

[179] J. Malik, A. Akhunzada, I. Bibi, M. Imran, A. Musaddiq, and S. W. Kim, "Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in sdn," *IEEE Access*, vol. 8, pp. 134 695–134 706, 2020.

[180] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using lstm based autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 2020, pp. 37–45.

[181] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[182] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, 2017.

[183] "Sgdoneclasssvm." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDOneClassSVM.html

[184] X. Yuan, R. Wang, Y. Zhuang, K. Zhu, and J. Hao, "A concept drift based ensemble incremental learning approach for intrusion detection," in *2018 IEEE international conference on internet of things (IThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE, 2018, pp. 350–357.

[185] N. Martindale, M. Ismail, and D. A. Talbert, "Ensemble-based online machine learning algorithms for network intrusion detection systems using streaming data," *Information*, vol. 11, no. 6, p. 315, 2020.

[186] G. Andresini, A. Appice, C. Loglisci, V. Belvedere, D. Redavid, and D. Malerba, "A network intrusion detection system for concept drifting network traffic data," in *International Conference on Discovery Science*. Springer, 2021, pp. 111–121.

[187] P. Horchulhack, E. K. Viegas, and M. A. Lopez, "A stream learning intrusion detection system for concept drifting network traffic," in *2022 6th Cyber Security in Networking Conference (CSNet)*. IEEE, 2022, pp. 1–7.

[188] S. Priya and R. A. Uthra, "Deep learning framework for handling concept drift and class imbalanced complex decision-making on streaming data," *Complex & Intelligent Systems*, pp. 1–17, 2021.

[189] L. Yang and A. Shami, "A lightweight concept drift detection and adaptation framework for iot data streams," *IEEE Internet of Things Magazine*, vol. 4, no. 2, pp. 96–101, 2021.

[190] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in *Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33*. Springer, 2020, pp. 508–520.

[191] R. R. dos Santos, E. K. Viegas, and A. O. Santin, "Improving intrusion detection confidence through a moving target defense strategy," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[192] "Spam e-mail database," 2023, https://cse.usf.edu/ lohall/dm/UCIarff/spambase.arff.

[193] "Combining similarity in time and space for training set formation under concept drift." [Online]. Available: https://sites.google.com/site/zliobaite/resources-1

[194] D. Mulimani, S. G. Totad, P. Patil, and S. V. Seeri, "Adaptive ensemble learning with concept drift detection for intrusion detection," in *Data Engineering and Intelligent Computing*. Springer, 2021, pp. 331–339.