An Intrusion Detection System for Software Defined Network based on machine learning algorithms based on time-sampled DDoS attack dataset

Abstract

Table of Contents

[Chapter 1 – Introduction 7](#_Toc92569198)

[1.1 Background study 7](#_Toc92569199)

[1.2 Aim 7](#_Toc92569200)

[1.3 Objectives 7](#_Toc92569201)

[1.4 Research Qns 7](#_Toc92569202)

[1.5 Methodology 7](#_Toc92569203)

[1.6 Limitations 7](#_Toc92569204)

[Chapter 2 – Literature Review and Analysis – SDN - IDS 8](#_Toc92569205)

[2.1 Enterprise Networks 8](#_Toc92569206)

[2.2 Firewalls 8](#_Toc92569207)

[2.3 Software-Defined Networking 9](#_Toc92569208)

[2.3.1 Components of SDN Architecture 10](#_Toc92569209)

[2.3.2 SDN Controllers 11](#_Toc92569210)

[2.3.3 OpenFlow 14](#_Toc92569211)

[2.3.4 Switches, virtual machines and virtual switches 14](#_Toc92569212)

[2.4 Vulnerabilities in SDN 15](#_Toc92569213)

[2.5 Threat Detection 20](#_Toc92569214)

[2.6 Chapter conclusion and research gap 23](#_Toc92569215)

[Chapter 3 Methodology and data 24](#_Toc92569216)

[3.1 Business understanding 25](#_Toc92569217)

[3.2 Data Acquisition 28](#_Toc92569218)

[3.2.1 Data Source 28](#_Toc92569219)

[3.2.2 Data pre-processing 30](#_Toc92569220)

[3.3 Modelling 31](#_Toc92569221)

[3.3.1 Feature selection and extraction 31](#_Toc92569222)

[3.3.2 Model selection and training 31](#_Toc92569223)

[3.3.3 Model evaluation 31](#_Toc92569224)

[3.4 Deployment 32](#_Toc92569225)

[3.5 Chapter Summary 32](#_Toc92569226)

[Chapter 4 Artefact and discussion 33](#_Toc92569227)

[Chapter 5 Conclusion and future direction 34](#_Toc92569228)

[References 35](#_Toc92569229)

[Annexure 40](#_Toc92569230)

[Code segment – for ISCX 2012 40](#_Toc92569231)

[Code segment – for ISCX 2017 40](#_Toc92569232)

Table of Figures

[Figure 5. The TDSP lifecycle. Source - (marktab et al., 2021) 27](#_Toc92582411)

[Figure 3. 1. Enterprise Network Scenario running on SDN 29](#_Toc92582412)

[Figure 3. 2. An IDS system for SDN enterprise network 30](#_Toc92582413)

[Figure 3. 3. Testbed architecture for ISCX 2012. Source (Shiravi et al., 2012) 32](#_Toc92582414)

[Figure 3. 4. ISCX 2012 dataset population. Source (Shiravi et al., 2012) 32](#_Toc92582415)

[Table 3. 1. ICI\_IDS 2017 attack-response activity list. Source- (Sharafaldin, Habibi Lashkari and Ghorbani, 2018). 33](#_Toc92582416)

[Figure 3. 5. Testbed architecture for CIC-IDS 2017. Source - (Sharafaldin, Habibi Lashkari and Ghorbani, 2018). 33](#_Toc92582417)

[Table 3. 2. TCP features extracted from PCAP file of ISCX2012 33](#_Toc92582418)

List of Acronyms

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Chapter 1 – Introduction

## 1.1 Background study

## 1.2 Aim

The aim is to simulate an SDN with firewall properties capable of thwarting DDoSNet attacks using machine learning or deep learning algorithms.

## 1.3 Objectives

The objectives include,

1. Learn and understand the elements in an SDN using a literature survey
2. Understand the threats in the environment and the latest advancements in the area
3. The use of ML/DL techniques in the process
4. Finding and understanding data sets in the process
5. Theoretical assessment on the deployability and limitations of the developed system.

## 1.4 Research Qns

The following are questions in the research,

1. How is SDN significant in network enhancement and increasing security?
2. What does AI mean in this field?
3. Is it possible to assess the gap between a deployable and experimental system?

## 1.5 Methodology

The research shall use Microsoft's team data science process in data process, algorithm design and deployability assessment. The detailed discussion is in the methodology chapter.

## 1.6 Limitations

The availability of a conclusive dataset for the experiment is a limitation. The collection of threat-related network data for the same is impossible due to the cross-sectional study timeline. It shall use only a theoretical assessment in the deployability of the model.

# Chapter 2 – Literature Review and Analysis – SDN - IDS

## 2.1 Enterprise Networks

Shin defines a computer network as a collection of wired and wireless links through which computers and other hardware devices exchange data and information. An Enterprise network comprises physical and virtual networks and protocols that connect all the users and systems in the network to a server, applications in the data centre, or to the cloud, to access data in the network. **(Shin, 2021)**

Peterson and Davie presented a network architecture as the configuration of all available hardware and software components arranged together to form a complete network system. The authors also described the most crucial characteristic of a computer network as its generality. Today's computer networks can perform many different functions, carry multiple types of data, and support a wide and increasing range of applications. They are not limited to particular functions like facilitating phone calls, directing television signals, or other operations performed on traditional single-use networks. **(Peterson and Davie, 2020)**

## 2.2 Firewalls

He presents a firewall as a security defence tool used between intranet and extranet in computer network security applications. The intranet is considered a secure network, while the extranet is considered relatively less safe. The firewall has both software and hardware and is used to analyse the information flow through it, filter analysed the information flow, and restrict the flow of information if deemed unsafe. The functions of firewalls include denying access to the intranet and authorising secure information flow into the intranet. Physically, firewalls are a combination of hardware devices like routers, hosts etc., and software that is used to protect confidential network data, resources and users from threats and attacks on the network. Firewalls have predefined configurations and rules to function. These rules and configurations monitor all data flow through it, allowing only authorised data flows and facilitating administrator monitoring and tracking. They are also responsible for recording the relevant connection sources, the server's communication display, and attempts to break into the network. **(He, 2021)**

## 2.3 Software-Defined Networking

According to Kirkpatrick, a Software-Defined Network (SDN) is a three-tier stacked architecture with an SDN controller sitting in the middle managing and directing data traffic between two Application Programming Interfaces, namely the Northbound APIs and the Southbound APIs **(Kirkpatrick, 2013)**. All computer networks have three planes of functionality: the data, control, and management planes. The control plane maps network traffic and decides for the data plane to forward the traffic accordingly. Whereas the management tools mainly involve software services for monitoring and configuring control functions. In all traditional vertically integrated networks, there is a coupling in the data and the control planes, in contrast to SDN decoupling. The decoupling feature is the most powerful feature of the Software-Defined network. The separation of the control plane (the network's control logic) from the resident routers and switches forwarding the traffic (the data plane) makes the switching nodes become simple forwarding devices controlled logically by a centralised controller. However, a logically centralised controller model should not be confused with a physically centralised controller model. Significant advantages of achieving this decoupling of the data and control planes include the desired flexibility, ease of development and deployment of new networking features, and network evolution and innovations scope. **(Kreutz et al., 2015) (Kirkpatrick, 2013) (Casado and McKeown, 2005)**.

According to Kreutz et al., there are three fundamental abstractions in the SDN concept, namely i) the forwarding abstraction, ii) the distribution abstraction, and iii) the specification abstraction. The function of the forwarding abstraction is to allow any desirable forwarding behaviour desired by the network application or the control program while hiding details of the hardware involved. On the other hand, the distribution abstraction functions as a shield for the various SDN applications from the vagaries associated with the distributed state by replacing the distributed control problem with a logically centralised controller. The feature requires a common distribution layer, should lie in an SDN's network operating system, install control commands on the forwarding devices, collect status information about the network devices and links in the forwarding layer, and offer a global network view to the network application. The last abstraction, the specification abstraction, is similar in function to the forwarding abstraction in that it should also allow any desirable network behaviour. However, it should not be responsible for implementing that behaviour itself. Network virtualisation solutions and network programming languages achieve this functionality in an SDN. These approaches relate the abstract configurations expressed by the network applications into the physical configuration abstraction of the global network view presented by the SDN controller. **(Kreutz et al., 2015) (Casado and McKeown, 2005)** **(Kobayashi et al., 2014)**.

There are four significant features of an SDN network architecture,

1. The separation in data and control planes – with in-built controls removed, the routers, switches and other network devices will revert to be simple packet forwarding elements.
2. Flow-based – instead of destination-based, the data forwarding decisions will be flow-based. In Software-Defined Networking, flow is the sequence of packets between the source and a destination.
3. Logical centralisation - the control logic shifts to a logically centralised SDN controller or a Network operating System (NOS), an external entity.
4. Programmed network – software applications running on the Network Operating System interact with the data plane devices and can be used to program the network, thus defining the characteristics of any Software-Defined Network. **(Kreutz et al. (Kirkpatrick, 2013) (Casado and McKeown, 2005)**

### 2.3.1 Components of SDN Architecture

The following are the different elements present in a typical SDN,

* Forwarding Devices (FD): Forwarding devices are hardware or software-based data plane devices. They are used to take action on incoming data packets. They have well-defined instructions (flow rules) defined by the Southbound interfaces. The SDN controllers implement these Southbound protocols.
* Data Plane (DP): The interconnected forwarding devices (via wired or wireless media) together forms a network infrastructure referred to as the Data Plane.
* Southbound Interface (SI): The Southbound Application Programing Interfaces (Southbound APIs) defined the instructions for the forwarding devices part of the Southbound Interface. The southbound interface also defines the communication protocols between the control plane and the underlying forwarding devices.
* Control Plane (CP): The control plane is the central hub for all applications and controllers. Control plane elements abstract the programming of the forwarding devices in the network.
* Northbound Interface (NI): the Northbound Interface abstracts the low-level instruction sets used by Southbound interfaces in programming the forwarding devices. Northbound API refers to the familiar interfaces for developing applications used by the application developers.
* Management Plane (MP): The functions offered by the Northbound Interface to implement network control and operation logic leveraged by a set of applications makes up the management plane. Such applications include routing, firewalls, monitoring, load balancing etc. **(Kreutz et al., 2015) (Casado and McKeown, 2005)** **(Kobayashi et al., 2014)**.

Diagram

Description automatically generated

Figure 1. A three-level logical abstraction structure of SDN. Source - (Kreutz et al., 2015)

### 2.3.2 SDN Controllers

An SDN controller sits in the middle of the SDN architecture and manages the data traffic. According to Nadeau and Gray, an SDN controller is a collection of physical or software systems that work up a network state management and distribution. These network systems might involve a database. The information controlled by the SDN Controller includes data on the network state, learned topology, configuration data, and control session data. In-memory database strategies may apply to manage this information, and the controller may have multiple purpose-driven data management processes. An SDN controller also provides a high-level data model to capture the relations between the services provided by the SDN controller, managed resources and policies. **(Casado and McKeown, 2005)** **(Kobayashi et al., 2014)**

The SDN controller services are exposed to the applications using a Representational State Transfer (REST) API. Together, the controller and its API formulate a development environment to generate the API code. Additionally, in some cases, it may even allow expansion of core capabilities and subsequent publishing of APIs for new modules like those supporting the dynamic addition on the controller capabilities. The SDN controllers are also responsible for providing a secure TCP control session between the controller and the network elements for stateful provisioning in the network elements, such as topology and service discovery mechanisms, path computation systems and other network-centric or resource-centric information services. In a distributed control paradigm, the network information is constructed as part of the host build process and run as a host or SDN controller service. SDN controller thus becomes a critical management interface for software switches and routers on hosts in a data centre, also responsible for the associated states of their ephemeral network entities like analytics and event notification. **(Nadeau and Gray, 2013) (Casado and McKeown, 2005)** **(Kobayashi et al., 2014)** **(Kreutz et al., 2015)**.

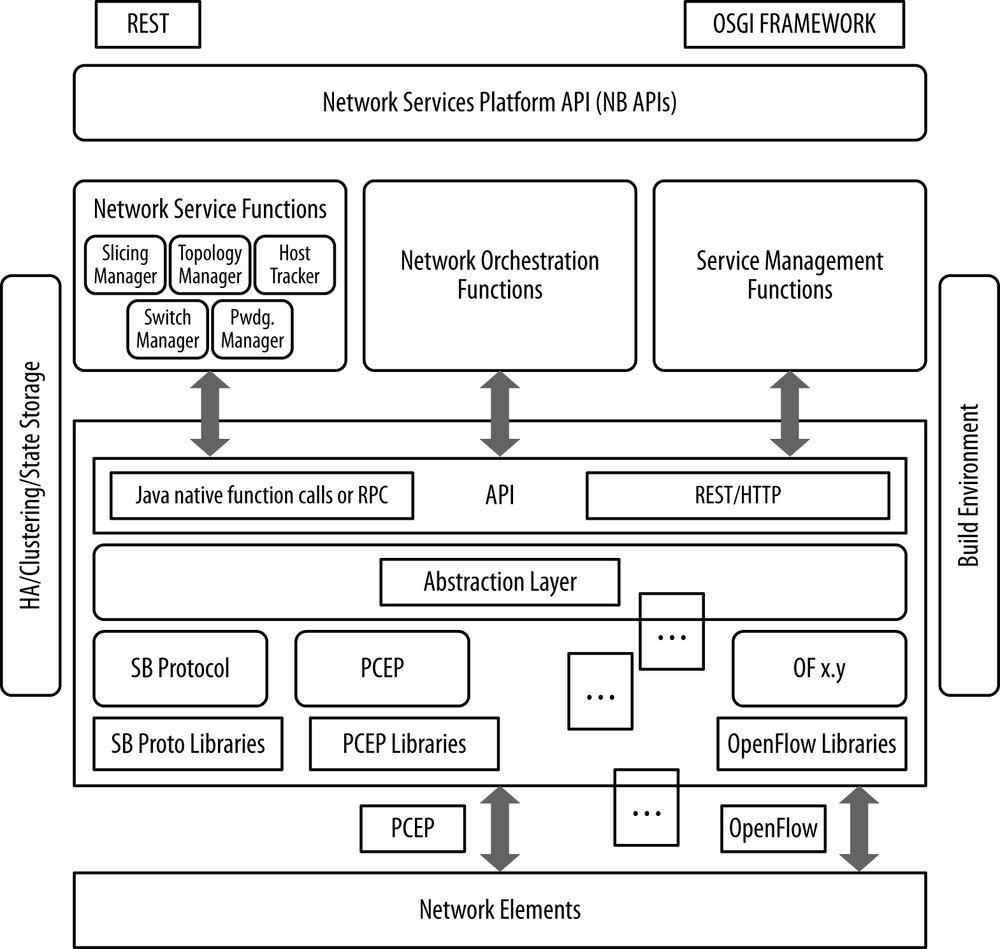


Figure 2. An ideal controller framework. Source -(Nadeau and Gray, 2013)

Ryu is a popular open-source Software-Defined Network controller and a component-based framework implemented entirely in Python, supporting components developed in other languages. Features of a Ryu controller includes an OpenFlow wire protocol support, a series of reusable libraries, application management, event management, and infrastructure services. Additionally, applications like Snort (a layer-2 switch), Virtual Router Redundancy Protocol (VRRP), GRE tunnel abstractions and topology and statistics services are available in Ryu controllers. Ryu also supports a REST interface for its OpenFlow operations and has an Open stack Quantum plug-in to support GRE overlay and VLAN. **(Nadeau and Gray, 2013)** **(Nippon Telegraph and Telephone Corporation, 2011)**.

### 2.3.3 OpenFlow

Southbound Interfaces or APIs connect links between control and forwarding elements, and OpenFlow is the widely deployed open southbound standard for SDN. OpenFlow is a communication protocol and not an API and allows the user to communicate between the SDN controllers and the forward plane of network devices, which direct traffic through the switches and routers from various vendors and manage them. OpenFlow remotely controls the access points, network switches and routers and provides an open interface for these networking nodes. There are two divisions in OpenFlow protocols: a Wire protocol and a Configuration and management protocol. The wire protocol establishes a control session, define the fundamental structure of switches, and message structures for exchanging flow mods and collecting statistics. However, a Configuration and management protocol allocates physical switch ports to a particular controller and ensures high availability and behaviours upon controller connection failure. OpenFlow protocols cannot directly provide network slicing, i.e., dividing an element into separately controlled port groups or a network into separate administrative domains. Additional tools like Flow Visor and specific vendor implementations can achieve this function. OpenFlow rules can also define various firewall applications. **(Nadeau and Gray, 2013)** **(Casado and McKeown, 2005)** **(Das and Mckeown, 2012)** **(Kobayashi et al., 2014)**.

### 2.3.4 Switches, virtual machines and virtual switches

All internetworking between LANs uses LAN switches. Layer 2 switches or Bridges filter and forward data packets and learn solely based on the destination locations and source addresses. As long as the network supports unique addressing, they can be interconnected using bridges. The bridge does not have any privileges on its attached Local area networks (LANs) and should not violate any access control rules. Therefore, a bridge becomes a networking device with many ports running an application to transparently forward data among those ports. A layer-3 switch or a routing switch is a networking switch used for forwarding data packets from one network to another network or device. A layer-3 switch can do all the traditional functions of a Layer-2 switch and perform additional tasks like static and dynamic routing. Multi-layer Switch (MLS) refers to networking devices that can fulfil all Layer-2 and Layer 3 functionalities and some Layer-4 features. **(Seifert and Edwards, 2008)** **(Casado and McKeown, 2005)** **(Das and Mckeown, 2012)** **(Kobayashi et al., 2014)**.

Virtual Machines (VMs) are logical machines comparable to a real host machine with an operating system. A VM system comprises multiple virtual machines, running operating systems in a single physical host machine. The Virtualisation layer is the software for hosting and managing all the VMs in a Virtual Machine system. Virtualisation, Resource scheduling, Migration, Security and Performance Evaluation are the critical technologies of VM Systems. Network virtualisation is the abstracting of traditional hardware to software delivered network resources. Migration refers to transferring one resource in a network to another location or network, i.e., making the VM, the operating system or memory transferable. Resource Scheduling refers to scheduling system resources for different tasks using different resource scheduling algorithms. As VMs are more prone to attacks, the security expectations are higher in Virtual machine systems. Various network parameters and methods can facilitate the Performance Evaluation. Although current performing monitoring methods can monitor and predict the behaviour and performance of a single VM, it becomes challenging in cases of multiple VMs in a virtual machine system. **(Li, Li and Jiang, 2010)**

Studies say virtual machines (VMs) running on the same host system need interconnection and physical network connections. For switching between VMs, software switches, called virtual switches, thus perform VM bridging, providing additional features apart from traditional switching features. The lowest common denominator in a networking environment is the Virtual switches or routers. In a Virtual LAN, the Virtual switches facilitate the forwarding of a small number of data packets compared to physical switches and routers **(Nadeau and Gray, 2013) (Emmerich et al., 2014) (Li, Li and Jiang, 2010)** **(Casado and McKeown, 2005)** **(Kobayashi et al., 2014)**.

## 2.4 Vulnerabilities in SDN

According to Zerkane et al., a Security vulnerability is a weakness in the network, if exploited, will transform the same into a flawed system. A breach in the network system security will expose additional weaknesses and violate its security characteristics, and identifying and eliminating vulnerabilities in any system becomes crucial to Data and operations. **(Zerkane et al., 2016) (Kreutz et al., 2015)**.

Any possible security vulnerabilities in a Software-Defined Network occur under one of the four layers of an SDN architecture – 1) the Application layer, 2) the Control layer, 3) the Data-Plane layer, and 4) the Management layer. Considering the attacks on these four layers, the attacks on the assets located in the Data-Plane and Application layers primarily come from the network and result from user interactions or external processes. But, the attacks on the Control layer does not require user interactions to succeed and may come from adjacent neighbours or a limited vector. All issues in these four layers fall into seven categories, namely

1. Unauthorised Access
2. Data Leakage
3. Data Modification
4. Malicious/Compromised Application,
5. Denial of Service
6. Configuration Issues, and
7. System-level Issues

The pathway by which the attacker exploits vulnerabilities in any form is called the attack vector, classifiable based on the attacker location. If it is an external location, then the attack is Remote, and if it is from a neighbouring network, which shares the same infrastructure with the enterprise, it classifies as Adjacent. A third scenario is when the attacker exploits the vulnerabilities within the SDN network; it becomes Local. Additionally, if the attacker is physically accessing the material, the attack vector will be considered physical. Finally, complexity refers to the conditions and efforts required by the attacker to exploit vulnerabilities in a system, which is in a low state if the attacker can identify the network state in a few steps just by going through SDN specifications and standards; the reverse is valid for high complexity. **(Zerkane et al., 2016) (Kreutz et al., 2015)**.

Milenkoski et al. identified six major challenges to securing the SDN environments utilising Network Functions Virtualization (NFV), which are 1) hypervisor dependencies, 2) elastic network boundaries, 3) dynamic workloads, 4) service insertion, 5) stateful versus stateless inspection, and 6) scalability of available resources. Furthermore, NFV networks contain a level of abstraction not appearing in traditional networks, which leads to additional complexities and challenges for security controls. In addition, centralised control often employed in the virtual layer, coupled with an SDN model, facilitate data centre consolidation. **(****Milenkoski et al., 2016)**.

Slavov, Migault and Pourzandi presented four threats to SDN based network operations - 1) provision of false feedback, 2) modification of valid on path requests, 3) forwarding wrong and unwanted traffic, and 4) gaining unauthorised access to any components. **(Slavov, Migault and Pourzandi, 2015)**. Scott-Hayward, Natarajan and Sezer described seven threats to SDN architecture - unauthorised access, data leak, data modification, malicious/compromised application, denial of service, configuration issues and system-level SDN security. **(Scott-Hayward, Natarajan and Sezer, 2016)**

Kreutz, Ramos and Verissimo, highlight the main vulnerability-inducing factors in an SDN as the network programmability and centralised network intelligence, which leads to mainly two threat vectors. First, an attacker can use network elements to launch a DoS attack against OpenFlow switches and controller resources by 'Faked or Forged' traffic flows. An authentication mechanism can prevent this, but if the attacker can assume control over the application server, they can inject authorised fake or forged traffic using authenticated Ports and MAC addresses. Attacks on Vulnerabilities on switches can also have drastic consequences. These attacks can clone or deviate network traffic, inject traffic, forge requests to overload the controller and neighbouring switches and drop or slow down data packets. The open nature of the network, dynamic design, software homogeneity, and centralised control plane pose vulnerabilities to SDNs. Advanced persistent threats, like Worms targeting specific network infrastructures, can automatically impair thousands of devices by modifying their control programs and configurations in case of opportunities against SDN networks. **(Kreutz, Ramos and Verissimo, 2013)**

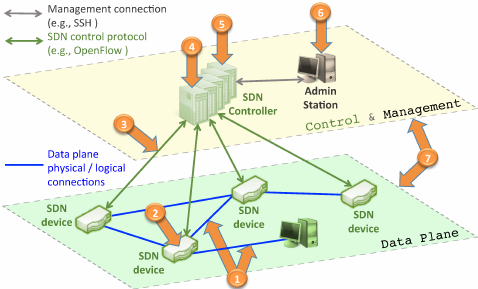


Figure 3: Threat Vectors in SDN, Source: (Kreutz, Ramos and Verissimo, 2013)

Chartuni and Márquez proposed that different types of attacks create varying anomalies in an SDN, with the significant attack vector being Denial of Service (DoS) attacks. DDoS attacks can cause exhaustion of bandwidth, memory, database, and server operations (input and output) and simultaneously hit across a computer network. **(Chartuni and Márquez, 2021)**

Mani and Nene classified DDoS attacks into the following. 1) volume-based attacks – overwhelming resources with colossal traffic, thus denying access to users, 2) protocol attacks – by exploiting protocol weaknesses, and 3) application layer attacks or Layer 7 (OSI Layer 7) attacks – to take down web servers. **(Mani and Nene, 2021)**.

Lawal and Nuray found that SDN Distributed Denial of Service (DDoS) attacks are launched against targeted victims (hosts) on the network from various compromised devices. The switch sends a request after receiving it to the controller to include missing packets in the flow table. If granted, the network can be brought down or rendered out of communication by attackers flooding or saturating SDN resources. **(Lawal and Nuray, 2018)**

Hsieh et al. state that a link-flooding attack is a DDoS attack that reduces the connectivity between end devices in a network by congesting the target link with a large number of low-speed flows. DDoS attacks are difficult to detect using intrusion detection systems or firewalls as they do not attack the target devices directly. Link flooding attacks can be conducted by the attacker using multiple bots at low costs with low-rate flows. It is difficult to identify low flow-rate flows as malicious because they use normal network traffic flows to avoid detection. A large quantity of seemingly normal traffic can congest traffic links in the network in a Link flooding attack. **(Hsieh et al., 2021)**

Foschini et al. proposed that every level and component of the SDN architecture is prone to attacks. The security issues present in each of the SDN planes are a) lack of authorisation, fraudulent flow rule insertion and lack of access control in the application plane, b) DoS attacks, unauthorised controller access and scalability along with availability issues, present in the control plane, and c) fraudulent flow rules, flooding attacks, controller compromising, TCP level attacks and Man-in-the-Middle (MIM) attacks associated with the data plane. The author also identifies four attack vectors associated with SDNs. The trust between applications and controllers, the presence of physical devices, lack of adequate authentication, authorisation, accounting mechanisms, and the vulnerabilities in the SDN Controllers, in the form of weak points, are the attack vectors associated with SDN. **(Foschini et al., 2021)**

Wang et al. propose that incorrectly classified attack data will affect the intrusion detection systems. Traditional classifiers frequently cause misjudgments, leading to degradation or deterioration of the Intrusion Detection System in an SDN. Intrusion detection models need to detect and give feedback to network traffic while improving detection efficiency and reducing latency. **(Wang et al., 2021)**

Foschini et al. propose another vulnerability in SDN networks, which can affect specific points in the SDN and can cause degradation of the entire network performance. The attackers use malicious traffic to trigger table miss processes in the switch with unknown or random protocol headers. When the switch cannot match such packets, it encapsulates them and forwards them to the SDN controller. As the switch has to forward each unknown packet in large malicious traffics, they can be overwhelming for the SDN controllers, causing bandwidth exhaustion between the elements. The continuous table-miss processes can consume all the switch capacity and the controller, running them out of resources while processing the fake traffic. In general, DoS and DDoS attacks transform the SDN controller into a single failure point, thus making the network visibility a vulnerability, thus requiring invisibility to all applications not requiring controller functionalities in an SDN. **(Foschini et al., 2021)**

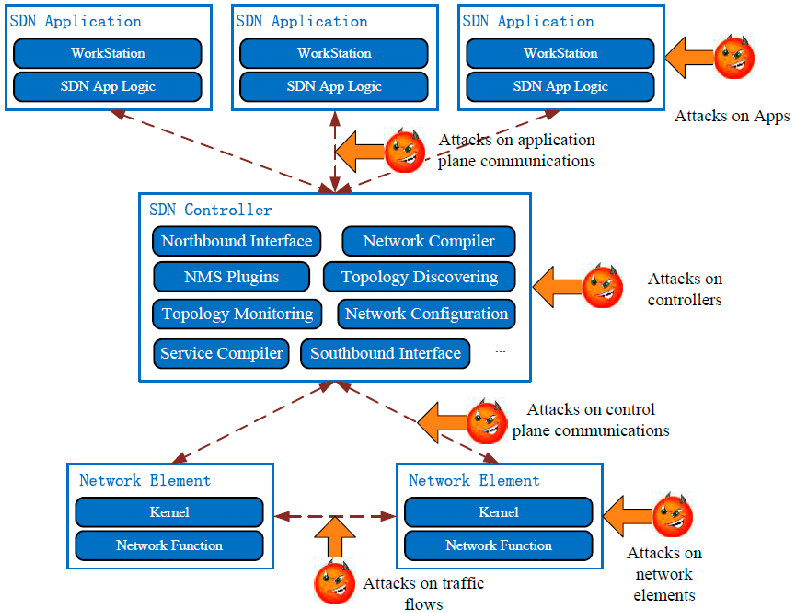


Figure 4: A comprehensive security attack vectors map of SDN, Source: (Luo et al., 2015)

In their study of SDN Mobile Networks (SDN-MNs), Luo et al. found that SDN-MNs can directly program the whole network and create dynamic flow policies, virtual property of the SDN. Consequently, it induces dynamic properties in the network. The addition of Network Function Virtualization (NFV) brings programmability into application networks and introduces new vulnerabilities in SDNs. Due to its inherent centralised design, without TLS, OpenFlow is vulnerable to Man-in-the-Middle (MITM) Attacks and Denial of Service (DoS) attacks. **(Luo et al., 2015)**

Kreutz, Ramos and Verissimo, proposed that traditional networks have natural defences compared to SDNs. The closed or proprietary nature of the network devices, their near-static design, a wide range of software, and decentralised control planes etc., acts as natural barriers against threats. The exploitation of a specific part or set of devices on the network can harm only that particular part of the network, unlike in an SDN, where exploiting one set of devices can potentially bring down the whole network. Using a common standard (e.g., OpenFlow) can also increase the risk and possibility of introducing common faults in the control plane software and implementing protocols. **(Kreutz, Ramos and Verissimo, 2013)**

Tang et al. suggest an Intrusion Detection System as a crucial part of the SDN architecture. The two classifications of Intrusion Detection Systems are Misuse detection and Anomaly detection, and several intrusion detection algorithms exist that can secure SDN networks based on OpenFlow. Support Vector Machines (SVM) and Self-organising Maps (SOM) best suit this purpose. **(Tang et al., 2020)**

## 2.5 Intrusion Detection Systems (IDS)

Slavov, Migault and Pourzandi propose encryption to prevent data leaks and a mutual authentication method to protect against man-in-middle attacks in SDN. Communication within the control plane should mutually authenticate, and protocols like TLS and IPsec employed for replay attack protection, confidentiality, and integrity protection. Encryption methods utilised along with Mutual Authentication is found to be more secure when compared to either technique employed alone. **(****Slavov, Migault and Pourzandi, 2015)**

Palanivel and Selvadurai suggested two different Microsoft threat modelling processes: STRIDE and DREAD. STRIDE is an acronym of six threats: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege, and apply to risk analysis in a system. On the other hand, DREAD stands for Damage, Reproducibility, Exploitability, Affected-Users and Discoverability. It is used for rating threats and prioritising, quantifying, and comparing risks associated with each type of identified threat. **(****Palanivel and Selvadurai, 2014)**

Milenkoski et al. proposed a framework providing functions like Intrusion Detection and prevention, malware detection, DoS protection, access management, etc., for SDN networks using Network Functions Virtualisation called NFV Security Framework. Network Function Chaining is the principle in this framework that directs the inbound traffic, first, to the firewall and then other elements. The NFV directs the traffic in the network. **(Milenkoski et al., 2016)**

Zhou et al. propose two methods to defend against interference attacks utilising flow table overflows. 1) Routing Aggregation: combining multiple entries in the flow table, such as a global routing schedule using packing optimisation algorithms for scenarios like load balancing etc., and 2) Multilevel Flow Table Architecture: implementing a larger flow table to store more flow entries, achieved through implementing multilevel flow-tables. The authors also suggest four improvements for security issues associated with current OpenFlow Switch and Flow table designs. These include 1) employing a new OpenFlow Switch architecture with a larger flow capacity, 2) a new flow table maintaining mechanism which can transfer flow entry while deleting the workload from the controller to the switch, 3) matching groups of flows using routing aggregation and 4) by developing interference detecting applications. **(Zhou et al., 2018)**

Ferrazani Mattos and Duarte proposed AuthFlow, an authentication and access control mechanism using host credentials to prevent unauthorised access. AuthFlow consists of an OpenFlow controller, an authenticator and a RADIUS server. The Extensible Authentication Protocol (EAP) messages between the requesting host and the RADIUS server are intercepted by an authenticator, allowing or denying traffic based on authentication response. AuthFlow performs authentication using Layer 2 protocols and maps the host's identity into the flow it creates on the network. **(Ferrazani Mattos and Duarte, 2016)**

Wen et al. propose PermOF, a permission system with permissions and an isolation mechanism to enforce the licenses at the API entries. A set of 18 permissions enforced at the controller's API entry and a customised isolation mechanism ensures resource isolation and access control. **(Wen et al., 2013)**

Porras et al. propose FortNOX, a security enforcement kernel, to defend against malicious controller applications. The FortNOX engine is used for role-based authentication to determine OpenFlow applications security authorisation. Three Flow rule producers, OF Operator, OP Security and OF application, are defined in this method for role-based authentication. However, this system does not provide some functions like priority enforcement. **(****Porras et al., 2012)**

Chandrasekaran and Benson proposed a method called LegoSDN to avoid the crash of SDN controllers caused by SDN applications failures. LegoSDN consists of an isolation layer between SDN applications, a transaction system covering the entire network to support atomic updates, efficient rollbacks, and a fault tolerance layer to detect and overcome crash triggering events.  **(****Chandrasekaran and Benson, 2014)**

Yao, Bi and Xiao introduce Virtual Source address Validation Edge (VAVE) for protection against DoS attacks via IP spoofing. The traffic analysis capabilities of SDN-Based networks are employed together with dynamically updating rules to prevent IP spoofing. Upon spoofing detection, the NOX based OpenFlow controller installs a firewall rule to stop the traffic from the harmful source address at the switch. A similar adaptor also limits the number of individual flow rules, further limiting the flow-table flood-attack capacity on the switch. **(****Yao, Bi and Xiao, 2011)**

Al-Shaer and Al-Haj, proposed FlowChecker using binary decision diagrams for detecting network errors. Misconfigurations arise when users write conflicting rules for single flow tables, and FlowChecker identifies any such intra-switch configuration mistakes. **(Al-Shaer and Al-Haj, 2010)**

Khurshid et al. propose VeriFlow for real-time policy checking of invariants by intercepting flow rules before they reach the network. It is a layer between the SDN controller and network device and checks for invariant violations dynamically during the insertion of each forwarding rule. It can also achieve very low latency, allowing the network to remain unaffected during each rigorous check within hundreds of microseconds per rule insertion. **(****Khurshid et al., 2012)**

Shin et al. propose FRESCO, an OpenFlow security application development framework incorporating FortNOX for the secure implementation of Software Defined Networks. FRESCO allows rapid design and development of security-specific modules which can be integrated as OpenFlow applications and provides a library of reusable modules for detection and mitigation of network threats. **(Shin et al., 2013)**



## 2.6 Threat Detection – relevance of AI

Alzahrani and Alzahrani presented that DDoS attacks threaten network security across any industry regardless of their size due to the continuous increase in complexity, volume, and frequency. DDoS attacks can be divided into two parts, Reflection based DDoS attacks and exploitation based DDoS attacks. In Reflection based attacks, the attacker's identity is hidden, and cyberspace devices are used to transmit attack traffic like HTTP calls to the target. Exploitation based DDoS attacks are TCP based, much similar to UDP-based attacks. **(****Alzahrani and Alzahrani, 2021)**

Wang et al. describe the three aspects of Intrusion detection technologies. They are 1) Intrusion detection based on data mining, using data mining technologies for intrusion detection, 2) Intrusion detection based on Machine learning, such as a cloud intrusion detection system combined with stacked compression encoder and support vector machine or an integrated intrusion detection algorithm using a decision tree, random forests etc., with adaptive ability, and 3) Intrusion Detection using Neural Networks, such as using teaching-learning metaheuristic optimisation algorithms to optimise neural network parameters for intrusion detection. **(Wang et al., 2021)**

Awan et al. define intrusion as the bypassing of the security mechanisms in a network or violating its security policies, i.e., Confidentiality, Integrity and Availability (CIA) of the network. Intrusion detection consists of monitoring intrusions and analysing events to seek malicious packets or the intrusion source in a network. It detects intrusion as an event different from legitimate or authorised events **(Awan et al., 2021)**.

According to Tang et al., Intrusion detection systems classify as Anomaly detection systems and Misuse detection systems. Anomaly detection models create a normal behaviour baseline and detect deviations from this model. The method can detect Zero-day attacks. Meanwhile, the Misuse detection model creates a signature database based on the patterns from intrusion behaviour and detects intrusion by monitoring and matching the user behaviour pattern. The method has a low false alarm rate but cannot prove ineffective against new or unknown attacks. Support Vector Machines (SVM) can effectively detect DDoS attacks in SDN networks. Self-Organised Map (SOM), a neural network-based technique, is commonly used to classify intrusion. **(Tang et al., 2020)**

Zhang et al. proposed a framework to detect intrusion in a shorter time, rather accurately using the Random Forest algorithm in Big data Apache Spark. The RF algorithm implemented in Big data Apache Spark used to detect intrusion in high-speed networks also evaluated efficiency and accuracy compared with existing systems. The proposed model achieved higher accuracy and shorter time for intrusion detection for high-speed network data. **(Zhang et al., 2018)**

Wang et al. propose a framework for intrusion detection by blocking abnormal traffic. A characteristic database of the attack type continuously enhances and improves the system defence. Machine Learning or Deep Learning techniques best solve the anomalies in networks that are binary or multi-classification problems. Data Mining, Machine learning, and Neural Networks have extensively produced positive results in network anomaly detection. However, one disadvantage of using data Mining and traditional Machine Learning techniques is that they rely too heavily on feature extraction and data selection, which is insufficient for good performance. **(Wang et al., 2021)**

Priya et al. propose a Machine Learning model using three different machine learning models, K-Nearest Neighbors, Random Forests, Naïve Bayes Classifier, to counter ML-based DDoS attacks. The study found that using machine learning principles in DDoS detection is highly effective and accurate. **(Priya et al., 2020)**

Zekri et al. proposed a real-time anomaly detection and mitigation technique against Distributed Denial of Service (DDoS) attacks on clouds. The model presented a DDoS protection design for cloud applications and other security challenges using machine learning (ML) techniques like Naive Bayes, Decision Trees (DT) and K-Means (KM). **(Zekri et al., 2017)**

Akbari et al. proposed a framework to design complex Reinforced Learning agents for network threat detection and mitigation. Threat detection is vital in every network for the safe and secure operations and transmissions of information. However, Attack vectors build for stealth-like Advance Persistence Threats (APTs) change their behaviour to adapt to the environment and deceive analysts. APTs appear to be dormant under scrutiny but will slowly and steadily take small steps over a long time. The author proposes a framework to detect and mitigate an APT attack and formulates threat mitigation as a Reinforced Learning problem. The proposed solution rapidly designs RL-based applications for network security and the complete formulation of network threat attacks similar to an RL problem with reward functions and action sets. The automation of networks and cyber security has been impacted significantly by advances in Machine Learning. **(Akbari et al., 2020)**

Mani and Nene proposed a method for utilising Policy-Based Flow Management (PBFM) capability with the sFlow-RT application of SDN networks to detect and mitigate DDoS attacks on Full Mesh Networks. The attack type is detected using defined key parameters from packet samples. Once a DDoS attack starts, with the threshold limit, the attack type is identified using sFlow-RT. The router or switch to which the attacker connects can be pinpointed and flow entries made in the respective element, based on the identified attack type, to block the threat traffic flow. A script with a predefined threshold and the parameters required to detect the attack type of the DDoS detect the attack. The DDoS attack is mitigated by making a flow entry in the identified switch or router using the PBFM capabilities of SDN. **(Mani and Nene, 2021)**

Aryal, Abbas and Collings proposed a model to detect DDoS attacks in 5G Networks using SDN by calculating a dynamic threshold using five machine learning classifiers with SDN embedded within the system with different sets of rules. The classifiers used in this model are RepTree, Naive Bayes, BayesNet, Random Tree and J48 algorithms. A correlation measure is calculated between the different features extracted and the dynamic threshold and is used to identify attacks. The data segment identified as having 100% sensitivity classifies as an attack. One advantage of this model is the regular examination and scheduled screening, improving the controller capacity to handle workloads. **(Aryal, Abbas and Collings, 2021)**

## 2.6 Chapter conclusion and research gap

It is evident from the references presented in the above sections, even though SDN approaches have revolutionised networking standards, the same has brought severe threats into the network. Intrusion detection systems prevent the frequency and extent of damage of such attacks. Most of these state-of-the-art machines combine software, hardware and AI learning architectures.

# Chapter 3 Methodology and data

The project's process is Microsoft's TDSP framework – Team DataScience Process. It imbibes AGILE essence in its understanding and application of process pathways giving intervention and feedback mechanisms into the modelling of a project. It is also tailor-made for predictive modelling and deployment in intelligent applications, unlike generic productised frameworks like SCRUM, near waterfall models like CRISP-DM or SEMMA, or other highly projectised frameworks. Data science projects tend to lose the deployability check mechanism if launched as experimental models alone; processes like TDSP ensure the inclusion of that vital feature. Each stage in the lifecycle inherently possesses three entities – goals, methods and artefacts.

The goals include the key variables that define project success, the target model and its metrics, and the identification of sources (of data) relevant to business understanding. The methods (how-to-do) relate, centrally, to the business problem knowledge and its solutional significance to stakeholders and finding relevant data to reach the solutions. The relevance changes at each lifecycle stage, depending on the specificity. Finally, artefacts are essentially deliverables or outcomes of that stage.

Diagram, schematic

Description automatically generated

Figure 5. The TDSP lifecycle. Source - (marktab et al., 2021)

The FIVE stages in TDSP, the business understanding, data acquisition and understanding, modelling, deployment and customer acceptance, have the three process entities mentioned above.

## 3.1 Business understanding

Understanding how an Intrusion Detection System and an SDN network can co-function shall draw the parameters of business requirements and corresponding research perimeters. Detailing on the system mentioned above, the components and entities in the coexistent are in figures (Figure 3.1, 3.2). The prominent reason for using SDN in any network is the elasticity and centralised control it gives the network, with flexible and dynamic policy creation. SDN switches installed with OpenFlow (or similar virtualisation software protocols) dissociate the packet traffic into data and management planes. Depending on the topology, physical ports can be connected in a ring or mesh within one or more switches. Such connectivity ensures traffic engineering based on various operational parameters, such as the QoS.

An enterprise network consists of collocated system users connected via LAN switches to the core network and remote users, accessing the various services and access packages. For example, the service inside an enterprise vary from a web server, intranet server, corporate mail server, FTP services, DNS and DHCP services to the network, other cloud and SaaS services, etc. Each of these critical service systems is quintessential to the operational quality of the business. Therefore, any breach in the network would make these entities vulnerable and expose invaluable data to the attacker. These attacks would come from any user points within or outside the network. For example, an attack such as the DDoS creates multiple bots or ghost machines that flood the traffic with TCP, UDP packets making service quality drops and finally shutting down services. As indicated in the figure, a DDoS threat vector can place itself anywhere in a network. The vulnerabilities detailed in the previous chapter suggests the breakpoints in an enterprise network. (Figure 3.1).

The context of an IDS system for SDN contains a machine learning module that has trained on one or more datasets built from actual data taken from a testbed and external sources. The testbed should consist of a packet sniffer such as Wireshark that generates the experiment results in a PCAF file. Then, an algorithm should create the respective features and populate the dataset. Any number of external learning data also can add the dataset via the feature creation module.

Core Switch L3 / SDN Controller

SDN OpenFlow / L2 LAN Switch

SDN OpenFlow / L2 LAN Switch

SDN OpenFlow / L2 LAN Switch

Enterprise Web Server

Enterprise DHCP Server

Enterprise Intranet Server

Enterprise DNS Server

Enterprise Mail Server

Enerprise Gateway / Router / Firewall router / VPN Router

Internet Cloud

VPN User OpenFlow / L2 LAN Switch

L2 LAN Switch 1

L2 LAN Switch N

Insitu User 1 OpenFlow / L2 LAN Switch

Insitu User N OpenFlow / L2 LAN Switch

BOT

BOT

BOT

BOT

BOT 1

BOT

**Enterprise Sevices**

**Enterprise VPN Cloud**

**Enterprise Intranet**

Other Attacks / Attack Vector Insertion

DDoS

DDoS

DDoS

DDoS

DDoS

DDoS

Figure 3. 1. Enterprise Network Scenario running on SDN

SDN controller and OpenFlow Switches

Virtual Machine Based Interface Packet Sniffer (eg WireShark)

All external Network elements

**Mgmt Plane**

**Data Plane**

All Enterpirse Entities

Internet CLOUD

PCAF File recorder

Machine Learning multi-model

Dataset Generation

Feature Creation

Feature extraction

**External Data Pipelines**

DDoS Threat Vectors

Threat Responder

**Isolation instruction**

Deployed model

**Data formatting**

**Firewall Rules**

Attack Simulator (BoNeSi)

Figure 3. 2. An IDS system for SDN enterprise network

A single or multimodel system can fit the training and testing data for readiness, which leads to an application-ready module when deployed (details available in the deployment section). (Figure 3.2). An attack simulator such as BoNeSi can test the model for threat perception. The deployed model detects DDoS intrusion and spurs a software-defined firewall threat responder entity. The module instructs the SDN controller to follow shut-off functions that include blocking IP traffic, shutting down ports, and isolating devices. The generated firewall rules can further train the learning model for evolving against future attacks.

The probability of including the entire context schema in the project scope is an obtuse problem-solution fit. The study shall explore the vital part of selecting an external dataset, feature extraction, model selection and training, and explore deployment methodologies. Further, it shall compare another dataset to benchmark the findings from the first dataset.

## 3.2 Data Acquisition

### 3.2.1 Data Source

After considering many options, there are two sources – ISCX IDS 2012 and CIC-IDS 2017. Both are from the same vendor, the University of New Brunswick, Canada. The attributed reasons for selecting the dataset are a) the dataset is not anonymised, b) real network elements used to derive the same, c) coherence in features and no post-capture insertion, d) threat-based interaction capture with full network trace, for data points, e) multiple intrusion scenario attributed to making the dataset, and f) both are evolvable. (Shiravi et al., 2012) (Sharafaldin, Habibi Lashkari and Ghorbani, 2018).

The ISCX 2012 is in PCAP derived format that needs to be converted to XML and later CSV for convenient use in modelling. Interestingly, the testbed, set over seven days, generates corresponding threat-response data points, indicating different conditions. Each day includes traffic, as in the self-explanatory figure below (Shiravi et al., 2012).

Diagram, schematic

Description automatically generated

Figure 3. 3. Testbed architecture for ISCX 2012. Source (Shiravi et al., 2012)

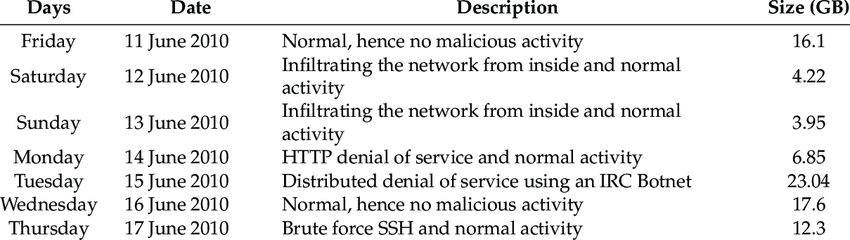


Figure 3. 4. ISCX 2012 dataset population. Source (Shiravi et al., 2012)

The CIC-IDS 2017 dataset follows a similar testbed, generation process and formats (PCAP) but has CSV formatted data converted and labelled using their then in-house development tool CICFlowMeter. In addition to the 2012 dataset, the new one has the most prominent attack, such as heart-bleed and bot. The detailed list of attacks in the CIC-IDS dataset is as follows and consists of activities over five day period, compared to ISCX 2012. (Sharafaldin, Habibi Lashkari and Ghorbani, 2018).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Activity #1 | Activity #2 | Activity #3 | Activity #1 | Activity #1 | Activity #1 | Activity #1 | Activity #1 |
| Monday | Non-malicious | Nil | Nil | Nil | Nil | Nil | Nil | Nil |
| Tuesday | Brute Force | FTP-Patator | SSH-Patator | Nil | Nil | Nil | Nil | Nil |
| Wednesday | DoS / DDoS | DoS slowloris | DoS Slowhttptest | DoS Hulk | DoS GoldenEye | Heartbleed Port 444 | Nil | Nil |
| Thursday | Web Attack – Brute Force | Web Attack – XSS | Web Attack – Sql Injection | Infiltration – Dropbox download | Meta exploit Win Vista | Infiltration – Cool disk – MAC | Infiltration – Dropbox download | Win Vista |
| Friday | Botnet ARES | Port Scan: Firewall Rule on | Port Scan: Firewall Rule off | DDoS LOIT | Nil | Nil | Nil | Nil |

Table 3. 1. ICI\_IDS 2017 attack-response activity list. Source- (Sharafaldin, Habibi Lashkari and Ghorbani, 2018).

Diagram

Description automatically generated

Figure 3. 5. Testbed architecture for CIC-IDS 2017. Source - (Sharafaldin, Habibi Lashkari and Ghorbani, 2018).

### 3.2.2 Data pre-processing

The ISCX 2012 dataset consists of PCAP trace files converted to CSV using XML extractor to 193 features classified under TCP, UDP, and ICMP calls. These are data from frame and packet header information. The features can be extended to 550 using Onut's feature classification schema into derivatives. **(Onut and Ghorbani, 2007)**. For avoiding complexities in classification, this research uses thirty features from the PCAP, pre-extracted based on TCP flow only, corresponding to DDoS attacks on June 14th and June 15th in the dataset (Table 3.2). **(Shiravi et al., 2012)**. The rest of the statistical, UDP and ICMP features are filtered off.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| frame.encap\_type | frame.len | frame.protocols | ip.hdr\_len | ip.len | ip.flags.rb | ip.flags.df |
| p.flags.mf | ip.frag\_offset | ip.ttl | ip.proto | ip.src | ip.dst | tcp.srcport |
| tcp.dstport | tcp.len | tcp.ack | tcp.flags.res | tcp.flags.ns | tcp.flags.cwr | tcp.flags.ecn |
| tcp.flags.urg | tcp.flags.ack | tcp.flags.push | tcp.flags.reset | tcp.flags.syn | tcp.flags.fin | tcp.window\_size |
| tcp.time\_delta | class |  |  |  |  |  |

Table 3. 2. TCP features extracted from PCAP file of ISCX2012

Increased feature dependence tend to deteriorate the model performance and further cause problems in deployment as use. However, the dataset inherently lacks in the Traffic Sockets Layer / Transport Layer Security (SSL/TLS) traffic, making a sizeable modern real-time internet traffic.

However, CIC-IDS 2017 includes an exhaustive compilation of intrusion data, with three different operating systems with a reduced domain set (85 features only) available in raw PCAP and extracted CSV formats. It also takes in the McAfee report on the most prevalent attacks for testbed simulation.

Feature drops in ISCX 2012 –

Feature drops in CIC-IDS 2017 –

## 3.3 Modelling

### 3.3.1 Feature selection and extraction

Feature extraction in ISCX 2012 –

Feature extraction in CIC-IDS 2017 –

### 3.3.2 Model selection and training

Models selected for the training are LSTM for ISCX 2012 and Naïve Bayes, random forest classifier with k-fold cross-validation and Deep Neural Networks, for CIC-IDS 2017.

Long-short-term memory (LSTM) is a unique recurrent neural network that uses previous information for the current evaluation neuron net. The model addresses the vanishing gradient problem in deep learning that stops learning due to minor weight updates in consequent training epochs. As the IDS falls in a time series problem, the research assessed LSTM would ensure better model performance.

Random forest algorithm is exceptional in classification problems, using multiple decision trees to generate a forest-like model. The grouping collates the performances of individual uncorrelated tree functions to yield better results from the model. Random forest is different from regular CART models as constituent tree follows randomly selected features and observations to build the forest and average the results. The algorithm's disadvantage is onverfitting caused by training on high variance (error from fluctuations in dataset during training). It happens when the irrelevant data (noise) overfits the model during training, making it inappropriate for actual deployment. However, the problem can be overcome using k-fold cross-validation, which segregates the training dataset into many groups; each successive fold may validate the model trained on the preceding fold.

### 3.3.3 Model evaluation

The following chapter discusses the results of applied models and performance evaluation.

## 3.4 Deployment

## 3.5 Chapter Summary

# Chapter 4 Artefact and discussion

# Chapter 5 Conclusion and future direction

# References

Akbari, I., Tahoun, E., Salahuddin, M.A., Limam, N. and Boutaba, R. (2020). ATMoS: Autonomous Threat Mitigation in SDN Using Reinforcement Learning. *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. [online] Available at: https://www.researchgate.net/publication/338779314\_ATMoS\_Autonomous\_Threat\_Mitigation\_in\_SDN\_using\_Reinforcement\_Learning [Accessed Nov. 2021].

Al-Shaer, E. and Al-Haj, S. (2010). FlowChecker. *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration - SafeConfig ’10*. [online] Available at: <https://doi.org/10.1145/1866898.1866905> [Accessed Nov. 2021].

Alzahrani, R.J. and Alzahrani, A. (2021). Security Analysis of DDoS Attacks Using Machine Learning Algorithms in Networks Traffic. *Electronics*, [online] 10(23), p.2919. Available at: https://www.mdpi.com/2079-9292/10/23/2919/htm [Accessed Nov. 2021].

Aryal, B., Abbas, R. and Collings, I.B. (2021). SDN Enabled DDoS Attack Detection and Mitigation for 5G Networks. *Journal of Communications*, [online] pp.267–275. Available at: https://researchers.mq.edu.au/en/publications/sdn-enabled-ddos-attack-detection-and-mitigation-for-5g-networks.

Awan, M.J., Farooq, U., Babar, H.M.A., Yasin, A., Nobanee, H., Hussain, M., Hakeem, O. and Zain, A.M. (2021). Real-Time DDoS Attack Detection System Using Big Data Approach. *Sustainability*, [online] 13(19), p.10743. Available at: https://www.mdpi.com/2071-1050/13/19/10743/htm [Accessed Nov. 2021].

Casado, M. and McKeown, N. (2005). The virtual network system. *ACM SIGCSE Bulletin*, [online] 37(1), pp.76–80. Available at: http://yuba.stanford.edu/~casado/vns\_sigcse.pdf [Accessed Nov. 2021].

Chandrasekaran, B. and Benson, T. (2014). Tolerating SDN Application Failures with LegoSDN. *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. [online] Available at: <https://dl.acm.org/doi/10.1145/2670518.2673880> [Accessed Nov. 2021].

Chartuni, A. and Márquez, J. (2021). Multi-Classifier of DDoS Attacks in Computer Networks Built on Neural Networks. *Applied Sciences*, [online] 11(22), p.10609. Available at: https://www.mdpi.com/2076-3417/11/22/10609/htm#B5-applsci-11-10609 [Accessed Nov. 2021].

Das, S. and Mckeown, N. (2012). Why OpenFlow/SDN Can Succeed Where GMPLS Failed. *Stanford database*, [online] 2012(2012). Available at: http://yuba.stanford.edu/~nickm/papers/ECEOC-2012-Tu.1.D.1.pdf [Accessed Nov. 2021].

Emmerich, P., Raumer, D., Wohlfart, F. and Carle, G. (2014). Performance Characteristics of Virtual Switching. *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. [online] Available at: https://ieeexplore.ieee.org/document/6968979 [Accessed Nov. 2021].

Ferrazani Mattos, D.M. and Duarte, O.C.M.B. (2016). AuthFlow: Authentication and Access Control Mechanism for Software Defined Networking. *Annals of Telecommunications*, [online] 71(11-12), pp.607–615. Available at: <https://www.researchgate.net/publication/299458634_AuthFlow_authentication_and_access_control_mechanism_for_software_defined_networking> [Accessed Nov. 2021].

Foschini, L., Mignardi, V., Montanari, R. and Scotece, D. (2021). An SDN-Enabled Architecture for IT/OT Converged Networks: a Proposal and Qualitative Analysis under DDoS Attacks. *Future Internet*, [online] 13(10), p.258. Available at: https://doi.org/10.3390/fi13100258 [Accessed 27 Oct. 2021].

He, X. (2021). Research on Computer Network Security Based on Firewall Technology. *Journal of Physics: Conference Series*, [online] 1744(4), p.042037. Available at: https://iopscience.iop.org/article/10.1088/1742-6596/1744/4/042037.

Hsieh, C.-H., Wang, W.-K., Wang, C.-X., Tsai, S.-C. and Lin, Y.-B. (2021). Efficient Detection of Link-Flooding Attacks with Deep Learning. *Sustainability*, [online] 13(22), p.12514. Available at: https://www.mdpi.com/2071-1050/13/22/12514/htm [Accessed 6 Dec. 2021].

Khurshid, A., Zhou, W., Caesar, M. and Godfrey, P.B. (2012). Veriflow. *ACM SIGCOMM Computer Communication Review*, [online] 42(4), pp.467–472. Available at: <https://doi.org/10.1145/2377677.2377766> [Accessed Nov. 2021].

Kirkpatrick, K. (2013). Software-defined Networking. *Communications of the ACM*, [online] 56(9), p.16. Available at: https://dl.acm.org/doi/10.1145/2500468.2500473 [Accessed Nov. 2021].

Kobayashi, M., Seetharaman, S., Parulkar, G., Appenzeller, G., Little, J., van Reijendam, J., Weissmann, P. and McKeown, N. (2014). Maturing of OpenFlow and Software-defined Networking through deployments. *Computer Networks*, [online] 61(August), pp.151–160. Available at: http://yuba.stanford.edu/~nickm/papers/openflow\_deployment\_journal\_paper\_aug2012.pdf [Accessed Nov. 2021].

Kreutz, D., Ramos, F.M.V. and Verissimo, P. (2013). Towards Secure and Dependable software-defined Networks. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking - HotSDN '13*. [online] Available at: https://www.researchgate.net/publication/255993819\_Towards\_secure\_and\_dependable\_software-defined\_networks.

Kreutz, D., Ramos, F.M.V., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S. and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, [online] 103(1), pp.14–76. Available at: https://ieeexplore.ieee.org/document/6994333 [Accessed Nov. 2021].

Lawal, B.H. and Nuray, A.T. (2018). Real-time Detection and Mitigation of Distributed Denial of Service (DDoS) Attacks in Software Defined Networking (SDN). *2018 26th Signal Processing and Communications Applications Conference (SIU)*. [online] Available at: https://www.researchgate.net/publication/326284088\_Real-time\_detection\_and\_mitigation\_of\_distributed\_denial\_of\_service\_DDoS\_attacks\_in\_software\_defined\_networking\_SDN.

Li, Y., Li, W. and Jiang, C. (2010). A Survey of Virtual Machine System: Current Technology and Future Trends. *2010 Third International Symposium on Electronic Commerce and Security*. [online] Available at: https://kevincurran.org/com320/papers/VmsPaper.pdf [Accessed Nov. 2021].

Luo, S., Dong, M., Ota, K., Wu, J. and Li, J. (2015). A Security Assessment Mechanism for Software-Defined Networking-Based Mobile Networks. *Sensors*, [online] 15(12), pp.31843–31858. Available at: https://www.researchgate.net/publication/291009784\_A\_Security\_Assessment\_Mechanism\_for\_Software-Defined\_Networking-Based\_Mobile\_Networks.

Mani, S. and Nene, MJ (2021). Distributed Denial of Service Attacks Detection and Mitigation in Software Defined Mesh Networks. *Recent Trends in Intensive Computing*. [online] Available at: https://www.researchgate.net/publication/356751099\_Distributed\_Denial\_of\_Service\_Attacks\_Detection\_and\_Mitigation\_in\_Software\_Defined\_Mesh\_Networks.

marktab, Alexbikgit, v-kents and microsoft (2021). What is the Team Data Science Process? - Azure Architecture Center. *docs.microsoft.com*, [online] 2021(11). Available at: https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview [Accessed Nov. 2021].

Milenkoski, A., Jaeger, B., Raina, K., Harris, M., Chaudhry, S., Chasiri, S., David, V. and Liu, W. (2016). *Network Function Virtualization*. [online] Available at: <https://downloads.cloudsecurityalliance.org/assets/research/virtualization/Security_Position_Paper-Network_Function_Virtualization.pdf>.

Nadeau, T.D. and Gray, K. (2013). *SDN : software defined networks*. 1st ed. O'Reilly Media Inc.

Nippon Telegraph and Telephone Corporation (2011). Welcome to RYU the Network Operating System(NOS) — Ryu 4.34 documentation. *ryu.readthedocs.io*, [online] 2011(537f35f4). Available at: https://ryu.readthedocs.io/en/latest/index.html [Accessed Nov. 2021].

Palanivel, M. and Selvadurai, K. (2014). Risk-driven Security Testing Using Risk Analysis with Threat Modeling Approach. *SpringerPlus*, [online] 3(1). Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4320241/> [Accessed Nov. 2021].

Peterson, L.L. and Davie, B.S. (2020). *COMPUTER NETWORKS : a Systems approach.* 6th ed. SL: Morgan Kaufmann.

Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M. and Gu, G. (2012). A Security Enforcement Kernel for OpenFlow Networks. *Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, 2012*, [online] pp.121–126. Available at: <http://conferences.sigcomm.org/sigcomm/2012/paper/hotsdn/p121.pdf>.

Priya, S.Shanmuga., Sivaram, M., Yuvaraj, D. and Jayanthiladevi, A. (2020). Machine Learning Based DDOS Detection. *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/9167642.

Scott-Hayward, S., Natarajan, S. and Sezer, S. (2016). A Survey of Security in Software Defined Networks. *IEEE Communications Surveys & Tutorials*, 18(1), pp.623–654.

Seifert, R. and Edwards, J. (2008). *The all-new Switch Book : the Complete Guide to LAN Switching Technology*. Indianapolis: Wiley Pub.

Shin, B. (2021). *A Practical Introduction to Enterprise Network and Security Management*. Boca Raton: Auerbach.

Shin, S., Porras, P.A., Yegneswaran, V., Fong, M.W., Gu, G. and Tyson, M. (2013). *FRESCO: Modular Composable Security Services for Software-Defined Networks*. [online] Semantic Scholar. Available at: <https://www.semanticscholar.org/paper/FRESCO%3A-Modular-Composable-Security-Services-for-Shin-Porras/ca089bc0bf0214da61098d06b5ad16ecf28f37a3> [Accessed Nov. 2021].

Slavov, K., Migault, D. and Pourzandi, M. (2015). *Identifying and Addressing the Vulnerabilities and Security Issues of SDN - Ericsson Technology Review*. [online] Ericsson.com. Available at: <https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/identifying-and-addressing-the-vulnerabilities-and-security-issues-of-sdn> [Accessed 14 Nov. 2020].

Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M. and El Moussa, F. (2020). DeepIDS: Deep Learning Approach for Intrusion Detection in Software Defined Networking. *Electronics*, [online] 9(9), p.1533. Available at: https://www.researchgate.net/publication/344450711\_DeepIDS\_Deep\_Learning\_Approach\_for\_Intrusion\_Detection\_in\_Software\_Defined\_Networking.

Wang, H. and Li, W. (2021). DDosTC: a Transformer-Based Network Attack Detection Hybrid Mechanism in SDN. *Sensors*, [online] 21(15), p.5047. Available at: https://www.mdpi.com/1424-8220/21/15/5047/htm [Accessed Dec. 2021].

Wang, Z., Zeng, Y., Liu, Y. and Li, D. (2021). Deep Belief Network Integrating Improved Kernel-Based Extreme Learning Machine for Network Intrusion Detection. *IEEE Access*, [online] 9, pp.16062–16091. Available at: https://ieeexplore.ieee.org/abstract/document/9319853 [Accessed Nov. 2021].

Wen, X., Chen, Y., Hu, C., Shi, C. and Wang, Y. (2013). Towards a Secure Controller Platform for Openflow Applications. *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking - HotSDN ’13*. [online] Available at: <https://www.scholars.northwestern.edu/en/publications/towards-a-secure-controller-platform-for-openflow-applications>.

Yao, G., Bi, J. and Xiao, P. (2011). Source address validation solution with OpenFlow/NOX architecture. *2011 19th IEEE International Conference on Network Protocols*. [online] Available at: <https://www.researchgate.net/publication/220978090_Source_address_validation_solution_with_OpenFlowNOX_architecture> [Accessed Nov. 2021].

Zekri, M., Kafhali, S.E., Aboutabit, N. and Saadi, Y. (2017). *DDoS Attack Detection Using Machine Learning Techniques in Cloud Computing Environments*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/8284731 [Accessed Nov 2021].

Zerkane, S., Espes, D., Le Parc, P. and Cuppens, F. (2016). Vulnerability Analysis of Software Defined Networking. *Foundations and Practice of Security*, [online] pp.97–116. Available at: <https://www.researchgate.net/publication/310463569_Vulnerability_Analysis_of_Software_Defined_Networking> [Accessed Nov. 2021].

Zhang, H., Dai, S., Li, Y. and Zhang, W. (2018). Real-time Distributed-Random-Forest-Based Network Intrusion Detection System Using Apache Spark. *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/8711068 [Accessed Nov. 2021].

Zhou, Y., Chen, K., Zhang, J., Leng, J. and Tang, Y. (2018). Exploiting the Vulnerability of Flow Table Overflow in Software-Defined Network: Attack Model, Evaluation, and Defense. *Security and Communication Networks*, [online] 2018, p.e4760632. Available at: <https://www.hindawi.com/journals/scn/2018/4760632/> [Accessed Nov. 2021].

# Annexure

## Code segment – for ISCX 2012

## Code segment – for ISCX 2017