A ROBUST MACHINE LEARNING BASED IDS DESIGN AGAINST ADVERSARIAL ATTACKS IN SDN

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

ALPER SARIKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

JANUARY 2024

A ROBUST MACHINE LEARNING BASED IDS DESIGN AGAINST ADVERSARIAL ATTACKS IN SDN

submitted by **ALPER SARIKAYA** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Information Systems Department, Middle East Technical University** by,

Date: 17 January 2024

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: ALPER SARIKAYA

ABSTRACT

A ROBUST MACHINE LEARNING BASED IDS DESIGN AGAINST ADVERSARIAL ATTACKS IN SDN

SARIKAYA, ALPER

Ph.D., Department of Information Systems

Supervisor: Prof. Dr. Banu Günel Kılıç

Co-Supervisor: Assoc. Prof. Dr. Mehmet Demirci

January 2024, 94 pages

Machine learning-based intrusion detection systems (IDS) are essential security functions in conventional and software-defined networks alike. Their success and the security of the networks they protect depend on the accuracy of their classification results. Adversarial attacks against machine learning, which seriously threaten any IDS, are still not countered effectively. In this thesis, firstly, a method is designed that employs generative adversarial networks to produce adversarial attack data. Then, RAIDS, a robust IDS model, is proposed which is resilient against adversarial attacks. In RAIDS, an autoencoder's reconstruction error is used as a prediction value for a classifier. Also, to prevent the attacker from guessing about the feature set, multiple feature sets are created and used to train baseline machine learning classifiers. A LightGBM classifier is then trained with the results produced by two autoencoders and an ensemble of baseline machine learning classifiers. The results show that the proposed robust model can increase overall accuracy by at least 13.2% and F1-score by more than 110% against adversarial attacks without the need for adversarial training.

Keywords: Intrusion detection, adversarial attack, autoencoder, InSDN, CICIDS 2017, adversarial robustness

ÖZ

YAZILIM TANIMLI AĞDA YANILTICI SALDIRILARA KARŞI MAKİNE ÖĞRENİMİ TABANLI DİRENÇLİ BİR SALDIRI TESPİT SİSTEMİ TASARIMI

SARIKAYA, ALPER

Doktora, Bilişim Sistemleri Bölümü

Tez Yöneticisi: Prof. Dr. Banu Günel Kılıç

Ortak Tez Yöneticisi: Doç. Dr. Mehmet Demirci

Ocak 2024, 94 sayfa

Makine öğrenimi tabanlı saldırı tespit sistemleri (STS), hem geleneksel hem de yazılım tanımlı ağlarda temel güvenlik işlevleridir. Başarımları ve korudukları ağların güvenliği, sınıflandırma sonuçlarının doğruluğuna bağlıdır. Herhangi bir STS'yi ciddi şekilde tehdit eden, makine öğrenimine yönelik yanıltıcı saldırılara karşı hâlâ etkili bir karşılık verilmemiştir. Bu tezde öncelikle, yanıltıcı saldırı verilerini üretmek için çekişmeli üretici ağları kullanan bir yöntem tasarlanmıştır. Daha sonra, yanıltıcı saldırılara karşı dayanıklı, sağlam bir IDS modeli olan RAIDS modeli önerilmiştir. RAIDS'de, bir otomatik kodlayıcının yeniden yapılandırma hatası, bir sınıflandırıcı için tahmin değeri olarak kullanılır. Ayrıca saldırganın özellik seti hakkında tahminde bulunmasını önlemek için birden fazla özellik seti oluşturulur ve temel makine öğrenimi sınıflandırıcılarını eğitmek için kullanılır. Daha sonra bir LightGBM sınıflandırıcısı, iki otomatik kodlayıcı ve bir temel makine öğrenimi sınıflandırıcıları topluluğu tarafından üretilen sonuçlarla eğitilir. Sonuçlar, önerilen modelin, yanıltıcı eğitimine ihtiyaç duymadan, yanıltıcı saldırılara karşı genel doğruluğu en az %13,2 oranında ve F-1 metriğini %110'dan fazla artırabildiğini göstermektedir.

Anahtar Kelimeler: Saldırı tespiti, yanıltıcı saldırı, otomatik kodlayıcı, InSDN, CICIDS 2017, dayanıklılık

"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards—and even then I have my doubts."

Eugene H. Spafford

ACKNOWLEDGMENTS

First of all, since 2016, my advisor, Prof. Dr. Banu Günel Kılıç has always enlightened me and given support to complete this hard way. Without her, it is impossible to complete the Philosophy of Doctorate. I am grateful to her.

I would like to express my gratitude and my respect to my thesis advisor Assoc. Prof. Dr. Mehmet Demirci for her priceless guidance, encouragement and continuous support to make this research possible.

I also thank my thesis jury members, Prof. Dr. Nazife Baykal, Prof. Dr. Ahmet Burak Can, Prof. Dr. Altan Koçyiğit and Prof. Dr. Sevil Şen for their suggestions and reviewing my work.

Special thanks to The Scientific and Technological Research Council of Turkey (TÜBİTAK) for 2211-A scholarship.

Finally, I would like to express my deepest gratitude to my wife, Özlem and my daughter İdil, for their love, encouragement and support in all my life. This thesis would not have been written without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BIM	Basic Iterative Method
CW	Carlini Wagner
DDoS	Distributed Denial of Service
DoS	Denial of Service
DT	Decision Tree
EFB	Exclusive Feature Bundling
EMD	Earth Mover's Distance
ENISA	The European Union Agency for Cybersecurity
FLA	Feature Level Attack
FSGM	Fast Gradient Sign Method
GAN	Generative Adversarial Network
GBDT	Gradient Boosting Decision Tree
GFLA	Generative Feature Level Attack
GOSS	Gradient-based One-side Sampling
HTTP	Hyper-Text Transfer Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IPS	Intrusion Prevention System
JSMA	Jacobian Saliency Map Attack
k-NN	k Nearest Neighbour
LR	Logistic Regression
ML	Machine Learning

NFV	Network Function Virtualization
NIDS	Network Intrusion Detection System
NN	Neural Network
PGD	Projected Gradient Descent
RL	Reinforcement Learning
QoE	Quality of Experience
QoS	Quality of Service
R2L	Remote to Local
RDP	Remote Desktop Protocol
RE	Reconstruction Error
RF	Random Forest
RNN	Recurrent Neural Network
SDN	Software Defined Network
SIEM	Security Information and Event Management
SPOF	Single Point of Failure
SVM	Support Vector Machine
TLS	Transport Layer Security
UR2	User to Root
VPN	Virtual Private Network
WAF	Web Application Firewall

CHAPTER 1

INTRODUCTION

1.1 Motivation

Networks are evolving at a fast pace due to disruptive technologies such as the Internet of Things (IoT) and cloud/edge computing [1]. Malicious network activities are also evolving and becoming ever more dangerous. Cyberattacks constantly target networks with the aim of preventing reliable/secure communications, making services inaccessible, stealing user data and other sensitive information.

However, malicious network activities remain harmful and everyday new one comes to the scene. The cyberattacks constantly target these networks and the attacks aim to prevent reliable/secure communications, break apart networks and take benefits from victims. The top 5 cyber threats remain the same (only order change) and have been expected to be active for years (See Table 1). Between 2017 and 2022, the concept and complexity of these threats increased dramatically [2].

Malicious activities on a conventional network pose a significant threat to the security and functionality of the network. These activities can range from the introduction of malicious code that causes damage to the network and related information resources [3], to various forms of attacks such as Distributed Denial-of-Service (DDoS) attacks, modification attacks, and double spending [4] and lastly ransomware. Malicious nodes within the network can engage in activities such as spoofing as legitimate users, modifying data packets, interrupting the network, and dropping or delaying packets routed through them. Furthermore, the growth of the internet and networked systems has exposed software to an increased amount of security threats [5].

On the other hand, new communication technologies are developing and changing the structure of the conventional network. On the other hand, scalability of network, monitoring and management of diverse hardware standard stand as a hard question for every network administrator. Software-defined network (SDN) and network function virtualization (NFV) are two prominent technological developments changing the conventional network structure to deal with scalability of network, resource management and vendor specific configuration issues.

Software-Defined Networking is an emerging paradigm that separates the network's control logic from the underlying routers and switches, enabling centralization of network control and the ability to program the network [6]. This functions allows network administrators to manage network services through the abstraction of higher-level functionality, enabling quick responses to changing business requirements [7].

Table 1: Cyber Threats and Status Change according to ENISA Threat Landscape Report 2022

Threats in 2020	**Threats in 2022**	**Status Change**
1. Malware	1. Malware	$\leftrightarrow$
2. Web based attacks	2. Web based attacks	$\leftrightarrow$
3. Web application attacks	3. Phising	$\uparrow$
4. Phising	4. Web application attacks	$\downarrow$
5. Spam	5. Spam	$\leftrightarrow$
6. Denial of service	6. Denial of service	$\leftrightarrow$
7. Ransomware	7. Ransomware	$\leftrightarrow$
8. Botnets	8. Data breach	$\uparrow$
9. Insider threat	9. Insider threat	$\leftrightarrow$
10. Physical manipulation	10. Botnet	$\downarrow$

SDN has been recognized for its potential to simplify network management, avoid configuration errors, and automate infrastructure sharing in wired networks [8]. By decoupling the control plane from the data plane, SDN enables the dynamism and flexibility of the network compared with traditional static network architecture [9]. Furthermore, SDN has been identified as a promising control platform that facilitates network programmability and bandwidth flexibility [10].

The significance of SDN also extends to the Internet of Things and the power grid. SDN has been proposed as a vital solution for the challenges in traditional IoT architecture, integrating network, storage, and security into a unified control model [11]. Furthermore, SDN has been recognized as a promising framework for robust software-defined optical networking in the power grid, facilitating network programmability and bandwidth flexibility [10].

Software-defined networking (SDN) has been applied in many domains in recent years to improve flexibility, ease of monitoring and management, and cost-effectiveness. SDN has also been analyzed extensively in terms of what it means for cybersecurity [12]. The security implications of SDN are vast, with new threats and defense approaches emerging incessantly in this relatively young field.

Each vendor-specific device has different characteristics, and the configuration of diverse devices creates a headache for network administrators. To eliminate these challenges, SDN provides a centralized view and programmability to network with different components [13]. Besides, SDN offers tremendous benefits, such as resource provisioning and cost-efficiency.

SDN separates control and data planes so that it provides flexible network management. By providing network programmability, such as OpenFlow protocols, SDN makes it easy to implement new network security algorithms in the control plane. An SDN controller is the backbone of the network. It provides different advantages to network administrators, such as controlling all physical devices at one point. Thus, it is easier to develop new security solutions than conventional networks [14].

One of the pioneer solutions for security challenges in the network is an intrusion detection/prevention system (IDS/IPS). IDS/IPS solutions are located outside of the layered security structure of the network (Fig. 1). IPS is more active than IDS since it both gives an alert and takes an action. Along with IDS/IPS, firewall, web application firewall, data loss prevention system, load balancer and Security

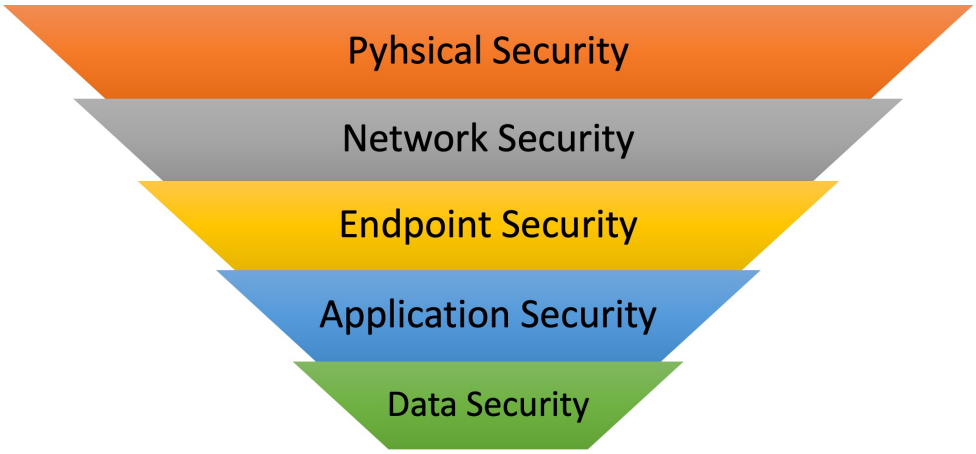

Figure 1: The Layered Structure of Network Security

Information and Event Management (SIEM) work together to defend the network from malicious network activity.

A fundamental task of intrusion detection is solving the traffic classification problem, i.e., distinguishing numerous types of attacks from normal traffic and each other. Proactive and self-adaptive IDS solutions should be considered against malicious network activities.

Machine learning is widely used for intrusion detection in network security. Machine learning-based intrusion detection systems have gained significant attention in recent years due to their ability to effectively detect and classify various cyber-attacks. These systems leverage machine learning algorithms to analyze network behaviors and identify anomalies indicative of potential intrusions [15]. The use of machine learning in IDS allows for the development of fully data-oriented models that reflect the behavioral patterns of cyber-attacks, enhancing the system's ability to adapt to evolving threats [16]. Additionally, machine learning algorithms enable the training of IDS to learn attack patterns and subsequently detect previously unseen attacks, thereby improving the system's detection capabilities [17].

Many machine learning-based IDS models achieve high accuracy with a low false alarm. Due to novel algorithms and promising architecture, machine learning-based IDS is one of the best application areas in cybersecurity. To detect malicious network activity precisely, IDS should be able to handle high volumes of network traffic [18].

Furthermore, the deployment of machine learning based IDS in SDN is getting more promising results. The reason is that SDN provides easy monitoring of network-wide traffic and supports elasticity to controlling of devices. With the help of the availability of network flow data, the programmability of the controller, centralization of control and diverse machine learning algorithms, it is more suitable for designing a more accurate IDS solution. There is a difference between conventional network and SDN about security application placement (Fig. 2). In the conventional network, each network security device runs separately and works distinct security mechanism. However, the controller can handle the security of the network collectively.

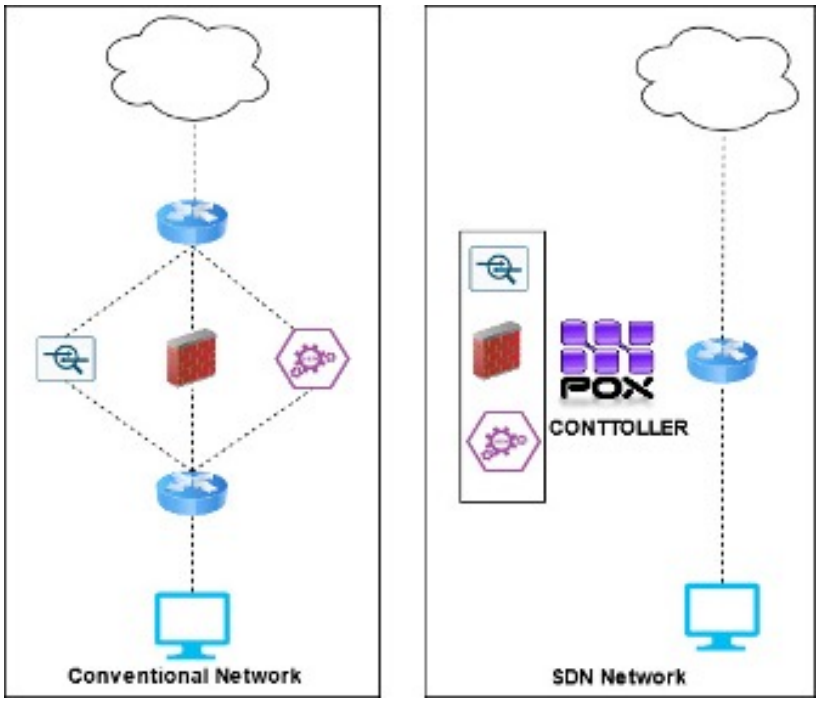

Figure 2: The Security Device Placement Conventional Network vs. SDN

The anomaly-based IDS is a type of machine learning-based IDS model. It first creates normal network traffic profiles. When suspicious traffic is detected which deviates from normal traffic, IDS creates an attack alert. This solution is attractive due to detecting zero-day attacks. Therefore, anomaly-based IDS is much more suitable for attack detection. There are numerous machine learning algorithms in the literature, including decision trees (DT), naive Bayes, support vector machines (SVM), neural networks, random forest, k-nearest neighbor (k-NN), and logistic regression (LR) [19].

Besides, deep learning architectures are becoming more prominent components of machine learning studies in various fields. Since deep learning can discover from raw data, this can yield a better representation of the IDS model. Also, recurrent neural network (RNN) is one of the most active research areas in deep learning [20] which works in a manner of bidirectional flow.

One of the common solutions for security challenges in both traditional and software-defined networks is an intrusion detection/prevention system. IDS/IPS solutions are located outside of the layered security structure of the network. IPS is more active than IDS since it both gives an alert and takes an action to neutralize the attack. Along with IDS/IPS, firewalls, web application firewalls, data loss prevention systems, load balancers, and security information and event management tools work together to defend networks from malicious activity.

Machine learning-based network intrusion detection systems (ML-NIDS) are used widely against network attacks. There are many examples of IDS solutions in the literature with high accuracy and low false-positive rate [19, 21]. The IDS types generally fall under two categories: anomaly-based and signature-based IDS (or hybrid). In an anomaly-based IDS, network traffic that is different from the normal network traffic pattern is labeled as malicious. In signature-based IDS, the known attack signature is compared with network traffic. However, it is essential to address challenges such as adversarial

Figure 3: The pipeline of ML-based IDS

attacks that aim to mislead machine learning-based IDS by maliciously manipulating the training data, highlighting the importance of ensuring the security of these systems [22].

Despite the respectable achievements of machine learning classifiers, adversarial attacks are a big threat against machine learning models [23, 24, 25]. In adversarial attacks, the original data is manipulated with imperceptible noise and these changes cannot be detected by classifiers precisely [26]. Especially in image detection tasks, models are easily tricked by adversarial attacks. As a result, machine learning classifiers can produce poor detection results against adversarial examples, as demonstrated in domains such as image classification, malware detection, and network attack detection.

One research shows that the accuracy of a network IDS may decrease from 100% to 0% as a result of adversarial attacks [27]. Another study argues that classifiers used for DDoS detection are affected negatively by adversarial attacks [28]. Therefore, any useful IDS should also be robust against adversarial attacks. The adversarial robustness of a classifier was discussed in the important work by Madry et al. [29]. In this context, robustness depends on the implausibility or difficulty of an attacker finding strong adversarial examples.

Researchers [24] have proposed various countermeasures against adversarial attacks. These measures include adversarial training, ensemble methods, robust optimization, data sanitization, and feature selection. Despite these countermeasures, adversarial attacks remain an important threat against the success of machine learning and all ML applications should consider the robustness of their classifiers.

1.2 Aim of the Study

Software-defined network helps in the control and management of networks without complex configurations. Also, programmability and data-control plane distinction by controller provide better management in a different vendor-specific network. NIDS is one of the active countermeasures for network security and is a fundamental requirement for the detection of attacks in SDN.

In [12], researchers review several papers about communication networks. In the SDN section, researchers argue that machine learning based IDS are a highly important topic and the security of the controller is an important aspect of the management of the network due to Single point of failure (SPOF). A more robust IDS solution should be implemented in SDN to protect the network from cyber-attacks.

In the literature, there exist lots of proposals against adversarial machine learning attacks in image classification/computer vision. However, there is a gap in the cybersecurity domain against adversarial attacks [30]. Besides, only a few papers propose intrusion detection models which are robust against

adversarial attacks. To address the above problems, a novel approach is proposed to defend against adversarial attacks in the IDS domain.

1.3 Scope of the Study

This research wants to propose anomaly-based IDS which is robust against adversarial attacks in an SDN environment and a conventional network. It only detects the network attack and does not prevent it, since it is not an IPS. The proposed model is anomaly-based and it does not use signatures for attack detection. It uses extracted features from network flow based on layer 2 to layer 4 packet properties. Some of the features are statistical and it is not available until the completion of network communication. Also, the deep packet inspection is not considered in this research. Both conventional networks and software-defined network are evaluated for testing.

1.4 Research Questions

The security of machine and deep learning models is getting more important for researchers due to adversarial attacks. The adversarial attack can result in decreasing the overall accuracy of the model and increasing the false alarm rate. For these reasons, every machine and deep learning model should check against adversarial attacks. There are countermeasures against adversarial attacks such as adversarial training, ensemble learning and feature optimization. The most preferable option is adversarial training.

Adversarial training is the easiest countermeasure against adversarial attacks because using statistical and probabilistic models, adversarial instances can be generated easily. These generated instances are used for the model training against adversarial attacks. According to the literature, adversarial training is applied in the image detection domain excessively. The image data consists of the same pixel scale (0 to 255) and this makes it easy for adversarial training. However, this may not be possible in the network domain due to the network attack surface, the extracted network flow characteristic, the scale/type of network flow data and network traffic characteristics. For this reason, using adversarial training may not provide robustness against adversarial attacks in the network domain. Also, the generated data does not become applicable to the network environment. For these reasons, this thesis aims to find an applicable countermeasure against adversarial attacks to create a robust IDS solution.

1.5 Contributions of the Study

The primary objective of this thesis is to design a robust machine learning-based IDS (called RAIDS) in the network domain against adversarial attacks. The main contributions of this thesis are summarized as follows:

1. To propose a new approach against adversarial attacks rather than adversarial training in the network domain,

2. Using autoencoders' reconstruction error values as inputs to another machine learning classifier for detecting adversarial examples,

3. Using network packet crafting tools, to test the feasibility of adversarial attacks in a real-time SDN environment and to evaluate the RAIDS model,

4. Demonstrating the effectiveness of GAN-based adversarial attacks on ML-based IDS/IPS systems.

1.6 Organization of the Thesis

The thesis is organized as follows: Section 2 focuses on background of machine learning based IDS, software defined networks and adversarial attacks. Section 3 reviews the literature. Section 4 provides an initial evaluation of the network dataset and a description of the generation of adversarial examples. Section 5 explains the details of the RAIDS model. Section 6 describes the methodology followed by the experiments, and Section 7 presents the experiment results and discusses. Finally, Section 8 concludes the research.

CHAPTER 2

BACKGROUND: SOFTWARE DEFINED NETWORK, NETWORK SECURITY AND ADVERSARIAL ATTACKS

In this chapter, network security, software defined network and adversarial attacks are discussed along with literature.

2.1 SDN Architecture

Software defined network (Fig.4) is decoupling of the control plane from the data plane. The network logic is controlled by the controller and this provides isolation of the network logic and intelligence from data devices. Another benefit of decoupling is minimizing the complex network functionality because there are several different types of network devices and each of them requires different configuration jobs. The programmability of SDN provides simplicity of control.

There are three different parts of the SDN architecture. The application, control, and data plane are the sub-layers of the SDN architecture. The application layer is responsible for policies, schemas, and upper layer functionality. Using the northbound interface, it communicates with the control plane. The control plane is the logic of the SDN and controls the management of the network, flow rules, application plane's policies' implementations, and security of the network. The data plane consists of the number of different forwarding devices. Each of these devices takes forwarding decisions and all decisions are derived from the control plane. Data plane communicates with the control plane via southbound interfaces. One of the southbound protocols is OpenFlow.

SDN requires a controller to manage the network. There is one controller in many topologies, but distributed or hierarchical controller placement is also suitable for a large network. According to [12] , nearly half of the studies (41%) have used the POX controller to implement their proposals. Besides, NOX, Ryu, Beacon, Open Daylight and Floodlight are the other most preferred controllers (Fig. 5). POX is a Python-based controller and suitable for elastic design.

OpenFlow is the southbound protocol that provides control of data plane devices. Policies that are taken by the application layer and control plane are applied by OpenFlow to the data plane. With the help of Transport Layer Security (TLS) protocol, communications between forwarding devices and control plane can be secured. The most used version of OpenFlow protocol is 1.3 and many vendor-specific switches are compatible with OpenFlow 1.3. (See Fig.6)

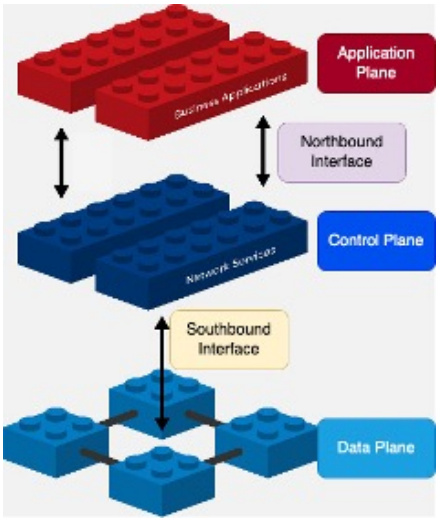

Figure 4: The Layered Structure of SDN

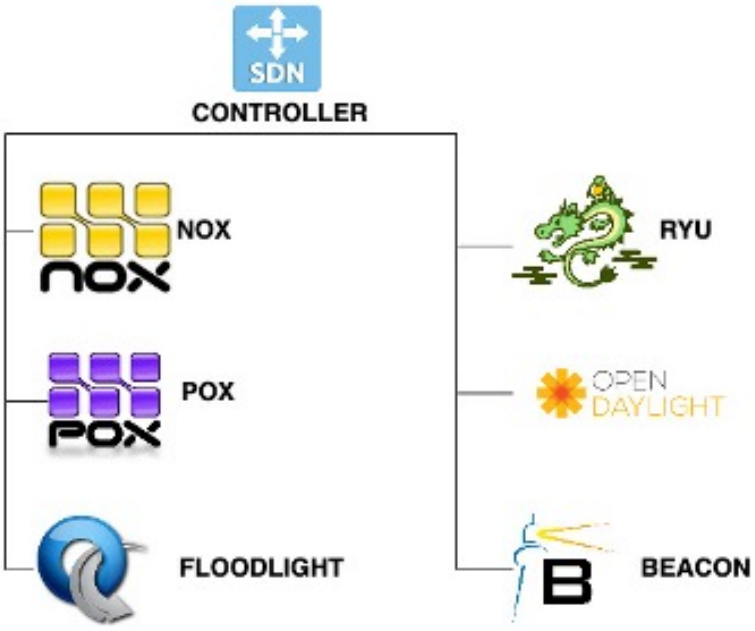

Figure 5: The SDN Controller

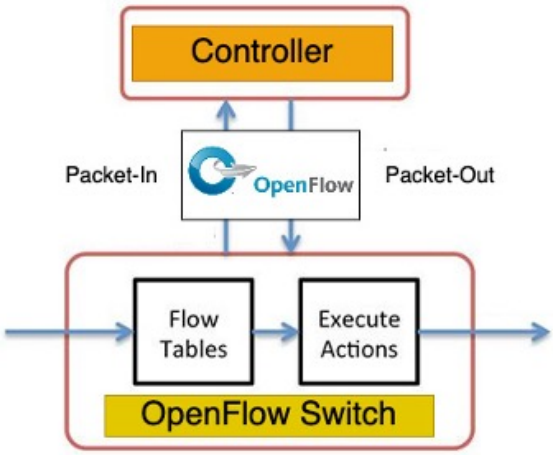

Figure 6: The OpenFlow Switch Decisions

Each switch has flow tables and flow rules are kept in these tables. Whenever a packet arrives at a switch, the switch firstly checks whether there is a matching entry in the flow tables. If there is no matching rule, the switch takes the default action according to the OpenFlow version. In OpenFlow 1.3, if there is no matching flow rule, a packet is dropped or discarded. A simple flow chart of OpenFlow [31] is presented in Fig. 7:

2.2 Network Security and Security Applications

Today, the information age presents many practicalities to us. People can easily check their bank accounts, look at social media, send e-mail to their employer, take a family photo and many other individual processes by using a computer, mobile phone or other smart devices. Although many commercial organizations and state agencies use the internet for their daily business, too, they have other special services, such as Virtual Private Network (VPN), cloud services etc. Even military operations rely on the Internet, which shows us that the Internet is indispensable for our everyday lives.

Basically, the Internet consists of a distributed local network. We keep, work, produce and distribute our data through this network. For all of these reasons, protecting our network is critical. Cyber security is all the measures that protect our resource, network and data. It simply provides the hardware and software for this purpose.

However, hackers, hacktivists and even state sponsored hackers use the Internet for making money, deactivate devices, changing political behavior, destroying our resources, eavesdropping, changing information and many other purposes. These make cyber security an indispensable part of our network.

Some security software and hardware ([32]) which are prevalent are;

- Antivirus / End-point security software,
- Data Loss Prevention Software,
- Software Based Firewall (Snort, Bro etc.),

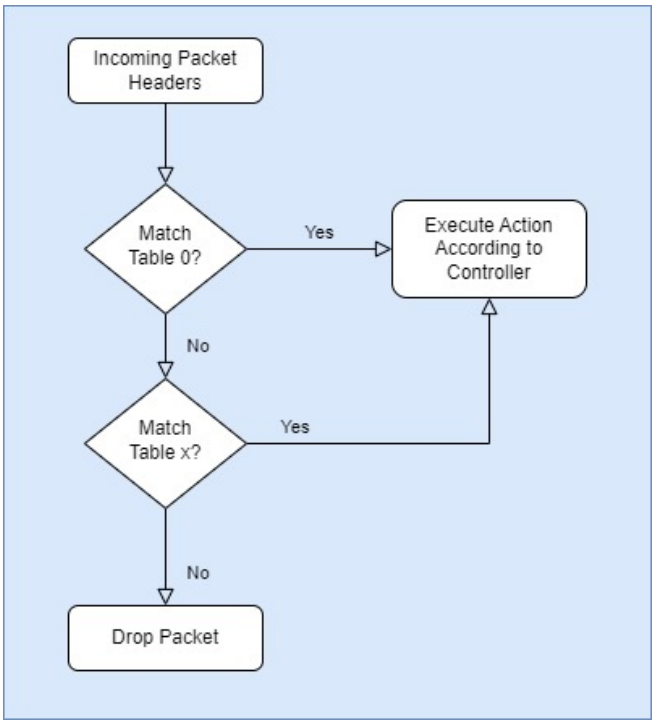

Figure 7: The OpenFlow Flow Chart

- Hardware Based Firewall,
- Web Application Firewall (WAF),
- Load Balancer,
- Intrusion Detection / Prevention System (IDS/IPS),
- Security Information and Event Management (SIEM).

Among these, the most critical measure is the Intrusion Detection/Prevention System. IDS/IPS monitors all the network traffic and alerts when it detects an attack. Besides, IPS behaves actively to suppress the attack. The systems are suitable for network security and provide very detailed network traffic logs. They mostly use attack signatures to detect an attack and generate logs. However, periodical updates for attack signatures are critical for these systems.

There are many examples of software-based IDS/IPS. Surricata, Snort and Bro are freeware versions of IDS/IPS systems. These applications use attack signatures and rely on daily/hourly signature update, which is critical for success. These applications produce logs which are related to signature/warning and errors. Often, the logs which are produced are hard to analyze and trace.

Network security engineers actively use IDS/IPS, firewalls and other security software/hardware in many networks. However, despite the active/passive measures, logs which are produced by this security software/hardware are plenty and hard to evaluate manually. For example, port scan is an alert for many IDS/IPS. They produce thousands of logs in every hour. On the other hand, a malicious

Table 2: The tactics of attackers

Tactic/Procedure	Definition	ID
Reconnaissance	The attacker is trying to gather information related future attacks.	TA0043
Resource Development	The attacker is trying to establish resources they can use to support attacks.	TA0042
Initial Access	The attacker is trying to get into your network.	TA0001
Execution	The attacker is trying to run malicious code.	TA0002
Persistence	The attacker is trying to maintain their stay out.	TA0003
Privilege Escalation	The attacker is trying to gain admin permissions.	TA0004
Defense Evasion	The attacker is trying to evade detection.	TA0005
Credential Access	The attacker is trying to steal usernames and passwords.	TA0006
Discovery	The attacker is trying to explore domain/environment.	TA0007
Lateral Movement	The attacker is trying to pass other devices.	TA0008
Collection	The attacker is trying to gather important data.	TA0009
Command and Control	The attacker is trying to communicate with C&C.	TA0011
Exfiltration	The attacker is trying to steal data.	TA0010
Impact	The attacker is trying to interrupt, or destroy the victim.	TA0040

request to port 139 with RDP is much more critical than port scan and this might happen once a week. Moreover, many IDS/IPS provide alerts or warnings but all decisions are taken by a security expert, which is not ideal for network security.

2.3 The Attack Framework

While evaluating the attackers' tactics, techniques, and procedures, the security administrators need a framework for assessing the impact of cyber attacks. Each attack has different and unique attack phases and life cycles. For these reasons, the threat model is a guide for the evaluation of cyber attacks.

Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) [33] is a guideline for classifying and analyzing cyber attacks. It addresses adversary behaviors, the life cycle of attacks, applicability to real environments, and taxonomy. The framework consists of 14 different tactics: reconnaissance, resource development, initial access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement, collection, command and control, exfiltration, and impact (See Table 2 and Fig.8).

2.4 Attack Surfaces / Vector in the SDN

Despite the number of benefits of the SDN, there are several different security threats against SDN, some of which are unique to the SDN. In many situations, there is one controller in the network, and this causes Single-Point-of-Failure (SPOF). In many attack situations, controllers and the communication channel between the planes are targeted. Also, apart from the controller, OpenFlow has

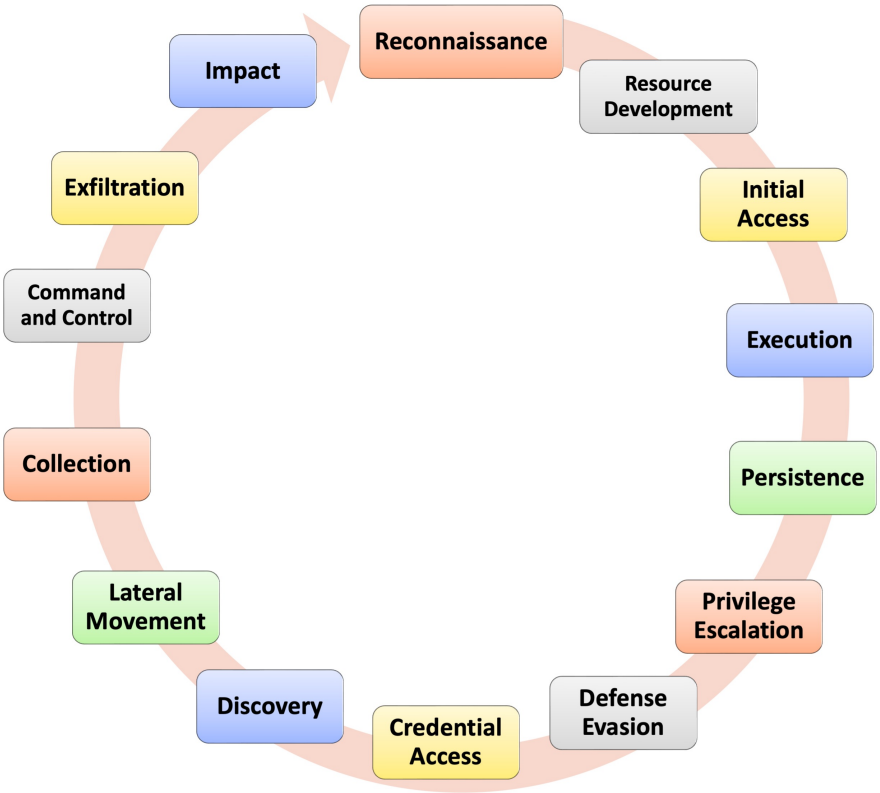

Figure 8: The MITRE ATT&CK Framework

some security issues. In [34], the researchers insist there are four different attack vectors in the SDN network:

- **Attacks on the data plane:**

  End-user devices are one of the easiest targets for attackers. The attacker's main priority is to gain unauthorized access to the victim's host and then using this host, the controller is targeted with malicious traffic. Flow entry manipulation, controller resource consumption, DDoS attacks are examples of data plane attacks.

- **Attacks on the control plane communication:**

  The controller is the critical point in the SDN topology. The controller communicates with the application plane by the northbound interface and with the data plane by the southbound interface. Each forwarding device has a different communication channel. However, there are limited number of physical links between devices. In this attack vector, the attacker's main aim is to disrupt the channel between the planes and break apart the switch from the network by flooding the attack. Also, man in the middle attack, probing and privilege escalation are other types of attacks [35].

- **Attacks on the SDN Controller:**

  In this attack vector, the controller is directly targeted by attackers. Attackers want to access the controller to bring it bring down. Another way is for attackers to take control of the controller and manage the network with their intention.

- **Attacks on the application plane:**

  Application plane software is targeted by the attackers. This helps attackers to manipulate security policies, firewall rules and other security precautions.

2.5 Attack Types in the SDN

Attack surfaces are large in conventional networks and SDN also brings unique threats to the network. Also, each plane has different types of attacks. In [13], researchers group attack types and provide a detailed view of the attack surfaces (See Fig.9). Brief information about the attack types is provided below.

- **Application Plane:**

  Application termination by abusing fixed privileges and authority: Network control application can be compromised by attackers, which enables attackers to execute malicious commands [36, 37]. Service neutralization: Malicious application which run in the application plane can alter control packets, disrupt control packet forwarding, and sniff sensitive network information to manipulate the control packet [36, 37]. Attacks to vulnerable northbound APIs: Northbound channel is vulnerable to misconfigurations, which enable attackers to change the network configuration [38].

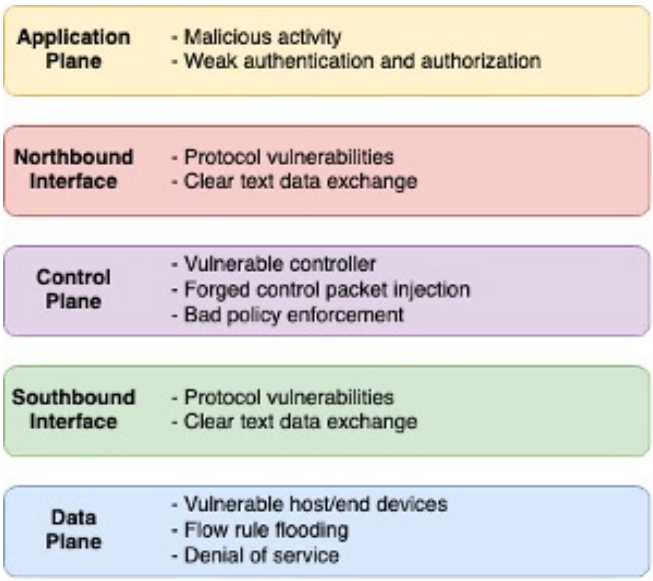

Figure 9: The Attack Surfaces in SDN

- **Control Plane:**

  Network operating system misuse: Malicious software and rogue forwarding switch can exploit the controller via its operating system [39]. Force switch disconnection: Spanning Tree poisoning and switch identity hijacking can be used by attackers to disconnect legitimate switch [40]. Controller's switch table flooding: TLS is an important security mechanism and without it (authentication and verification), malicious packet-in messages can overload switch flow table [39]. Controller poisoning: Attackers can use malicious application and network protocols to poison the controller. Then, the attackers can easily change the network rules and the management of the network [38].

- **Control Channel:**

  Eavesdropping: An attacker can sniff the control channel which is unencrypted. In this way, the attacker can listen and maintain critical information about the network [37]. Man in the Middle: Using ARP poisoning, an intruder can divert all network communication to that point, and he can control every communication [37].

- **Data Plane:**

  Denial of service (DoS/DDoS): Attacker can access vulnerable host and create a great number of packets to overload the controller. This makes the controller to disconnect from the network. Flow-rule flooding: Firstly, switch's flow table is filled completely by the attacker. Then, whenever the arriving packet misses the flow-rule in the switch, the switch creates a request for installation of new rules. This affects the network performance negatively [37]. Side-channel attacks: It is similiar to reconnaissance attack which aims to get information about in the network, configuration, and other important information about the controller [37].

2.6 Black-Box and White Box Adversarial Attacks

The attacker's knowledge about the target system dictates the type of adversarial attacks. The knowledge about the target system can change the adversary behavior. If an attacker does not know anything about the target system, this is called as a "Black-box attack" (Also known as a blind attack") [41]. The adversary can make an assumption about the target system. Similar machine learning classifiers are tested with perturbated features. On the other hand, the attacker can know the target system. This is called as a "White-box attack". It is easier and more powerful than a "Black-box attack" due to broad information about the system.

Training Phase Capabilities:

In this phase, the attacker attempts to corrupt the dataset. The training data is targeted by the attacker. This is the most straightforward attack, but accessing data or model is not an easy task. According to [39], there are three attack strategies during the training phase:

- **Data injection**

  The attacker does not access the training data or model but has the ability to augment new data to the training set. The adversarial example can be added the original data to modify the learning model.

- **Data Modification** The attacker does not have access to the model, but the training data. The attacker can change the training data and trick the model to allow adversarial training.

- **Logic Corruption** The attacker has the ability to modify the model. All model assumptions or the objection function can be changed by the attacker.

Testing Phase Capabilities:

At the testing time, the attacker wants to produce incorrect results by the model rather than tamper with the model training or the training data. However, the attacker should know enough information about the model to trick the output.

2.6.1 Attack Algorithms

2.6.1.1 Fast Gradient Sign Method

Fast gradient sign method was proposed to search for adversarial examples in an easy way [14]. It is the simplest and the most widely used untargeted adversarial attack method. The one step gradient update is performed along the direction of the sign of the gradient. The perturbation can be expressed as:

$$\eta = \epsilon \cdot sign\left(\bigtriangledown_x J\left(w, x, y\right)\right) \tag{1}$$

where ϵ is the magnitude of the perturbation, x is the input, w is the weight and y is the label. The generated adversarial example $x^{'}$ is calculated as $x^{'}$ = x + n.

2.6.1.2 Jacobian-Based Saliency Map Attack Method

Jacobian matrix is used for the creation of adversarial examples via L_0 distances [14]. For given x, JSMA computes the Jacobian matrix:

$$J_f(x) = \frac{\partial_f(x)}{\partial_x} = \left[\frac{\partial_{f_j}(x)}{\partial_{x_i}}\right] ixj \tag{2}$$

This identifies most significant features of the x, which largely changes the output.

2.6.1.3 Carlini Wagner Attack Method

It is an iterative gradient method that uses Adam optimizer and tanh function to produce the adversarial attack. To defeat defensive distillation, Carline and Wagner [13] proposed a new objective function g:

$$\min_n \|n\|_p + c \cdot g(x+n)\, s.t. x + n\epsilon [0,1]^n \tag{3}$$

where g(x') 0 if and only if f(x') = l', and l' is the label of the targeted adversarial examples. They suggested that the distance and the penalty term can be better optimized in this approach.

2.7 Defensive Measures Against Adversarial Attack in SDN

The defenses should be considered to mitigate the effect of adversarial attacks. The success of adversarial attacks can directly relate to countermeasures. According to [24], security-by-design countermeasures can be effective against adversarial attacks. The defense techniques could be divided into two types: proactive and reactive [42]. While the proactive approach is to obstruct the model during training from adversarial examples, the reactive approach is to detect adversarial attacks during testing. The countermeasures against adversarial attacks:

- **Adversarial training** The method is proposed by Ian J. Goodfellow et al. to enhance the model robustness against adversarial attacks [43]. During the training phases, the model should be trained with adversarial attack samples. So, the model's loss function becomes familiar to the adversarial samples. The generative adversarial method (GAN) is one example of automatic creation of adversarial examples. However, due to the abundant space of adversarial examples, some adversarial examples are not detected by the model. Besides, this technique may decrease the overall accuracy of the model.
- **Ensemble methods** Multiple classifiers and defense techniques can be used to increase resiliency of the proposed model for attack detection. By using multiple methods, the attacker's face up multiple defense layer to evade the model. Each classifier has different feature importance, and, in this way, the attacker cannot manipulate the whole IDS solution easily. However, the selection of the weak defense algorithm does not provide enough protection against adversarial attacks [44].

- **Feature selection** Some of the network flow features can be vulnerable to easy manipulation. Removing these features from the training set becomes a defense technique against adversarial attacks. Besides, using a high number of features also causes overfitting. Removing these features decreases the model's complexity. Furthermore, the selection of features that can be manipulated by the attacker are extracted from the training set and in this way, the attacker cannot change feature parameters. However, applying this technique can decrease the model's overall accuracy and increase the false alarm rate.

- **Robust optimization** The decision boundaries of the machine learning algorithms can be adjusted to reduce the effect of adversarial attacks. This technique improves the optimization function such as using regularization terms, upper and lower bands bounds of the cost function or adding extra layers to the model to improve adversarial resiliency [45].

- **Data sanitization** This techniques can be used against the poising attacks. The basis of this defense is augmenting the training dataset with examples which overlaps goal-specific criteria. The poisoned samples can be identified by the this technique [46]. However, according to [47], these approaches are not suitable for the IDS solutions.

2.8 Datasets and SDN Compatibility

A dataset is a fundamental part of training/testing of machine learning models. There is a number of public IDS datasets and each of them has different characteristics. KDD'99, NSL-KDD and ISCX 2012 are used in previous research. According to [48], the importance of datasets dictate research and can produce biased results. A brief review of each dataset is provided below.

KDD'99 is the first public dataset and it is widely used in IDS evaluation. DARPA network traffic was used for creation of this dataset. The dataset contains 41 features and these features are grouped as basic, traffic and content based features. Attack classes are Denial of Service, Remote to Local (R2L), User to Root (U2R) and probe. KDD 99 dataset has some drawbacks: (1) time to live value in attack data packet is 126 or 153, but these values do not occur in the training records of attack data, (2) probability distributions of the training and test sets are different from each other, (3) the dataset does not include low footprint attack and (4) numbers of redundant records are too high.

NSL-KDD dataset is an improved version of the KDD 99 dataset, where the duplications were removed and the unbalanced dataset problem was eliminated. However, attack size in the testing set is higher than the training set. Both KDD'99 and NSL-KDD are obsolete datasets.

ISCX 2012 is a relatively more up-to-date dataset than NSL-KDD and KDD'99. The alpha-profile is used for attack creation and beta-profile is used for normal traffic generation. Denial of service and brute force attack are two attack types in this dataset. The data set consists of 20 features. The main drawbacks of this dataset are that there is only HTTP related traffic, the attack ratio is low and there are unsuitable features extracted from the OpenFlow protocols.

UNSW-NB15 is a highly up-to-date dataset and the attack classes presented are enough for attack classification. There are 43 features for the classification task and the number of instances are consistent in both the training and the testing sets. However, in previous research, the attack classification is not well-designed and some attack classes are hard to detect.

Table 3: Dataset Comparison

Dataset	Year	Number of Attributes	Number of Attack Classes	Network Type
KDD'99 [49]	1998	41	4	Conventional
NSL-KDD [50]	2009	41	4	Conventional
ISCX 2012 [51]	2012	-	4	Conventional
UNSW-NB15 [52]	2015	43	9	Conventional
CSE-CIC-IDS 2018 [53]	2018	83	9	AWS
InSDN [34]	2020	83	7 (6 for only SDN)	SDN

CSE-CIC-IDS2018 was presented by the Canadian Institute for Cybersecurity and Communications Security Establishment. A cloud platform was used for the dataset retrieval. The normal network traffic was created by B-profiles and the attack traffic was created by M-Profiles. The number of features in the dataset is high, which is 83 features and various for the classification task. Redundant records and missing class labels are main drawbacks of the dataset.

InSDN is one of the first examples of comprehensive SDN datasets in the intrusion detection domain. CICFlowMeter were used for feature creation and in total 83 features were created for attack classification. There are SDN related features in the dataset (48 features) and six attack classes are represented in the dataset, which are DoS, DDoS, botnet, web attack, probe and U2R. Researchers want to solve inherent problems in previous research and SDN elements are used in traffic/attack generation.

According to the comparison in Table 3, InSDN dataset is a preferable dataset for SDN based research. This dataset consists of 48 features and there are flow-based and subflow-based attributes which are directly related to the SDN domain.

2.9 Performance Metrics

Accuracy, Precision, Recall and F1-Score are used for measuring the performance of models. For multiclass classification, overall accuracy, class detection rate and class FP rate are used. True Positive (TP), i.e., positive instances that are classified as a positive; True Negative (TN), i.e., negative instances that are classified as a negative; False Positive (FP), i.e., negative instances that are classified as a positive; False Negative (FN), i.e., positive instances that are classified as a negative, are our basis terms.

Accuracy is the percentage of correctly classified instances and is measured as

$$\frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4}$$

Precision is the percentage of positive instances that are correctly labeled and is measured as

$$\frac{(TP)}{(TP + FP)} \tag{5}$$

Recall (Detection Rate) is the percentage of actually positive instances and is measured as

$$\frac{(TP)}{(TP+FN)} \tag{6}$$

F1-Score is the weighted average of the precision and recall:

$$2 \times (Precision \times Recall)/(Precision + Recall) \tag{7}$$

CHAPTER 3

THE RELATED WORK

In this chapter, the previous research about ML based IDS, adversarial attacks and countermeasures are discussed with literature.

3.1 Machine Learning Techniques with Intrusion Detection System

Machine learning is a subset of artificial intelligence that a program learns from experience with the help of data without being explicitly programmed. The main idea behind machine learning is making a human-level task [19]. Machine learning helps us in many fields, such as spam detection, face recognition, drug discovery, driverless car, cyber-attack detection, speech recognition, etc.

Machine learning can be easily applied to SDN and this can support many different tasks which can be controlled by machine learning algorithms. Advances in the computing technology, data-driven management of the network in SDN and retrieving real-time network information or statistics are the main benefits of SDN.

In the SDN context, machine learning can overtake different types of tasks, such as traffic classification, routing optimization, QoS/QoE prediction, resource management and security [14] (Fig. 10). Logically centralized control, traffic monitoring, rule manipulation and a complete view of the current network situation can provide valuable data to machine learning algorithms.

Intrusion detection system is one of the key solutions in the area of network security. There is a layered structure in many network topologies and each layer has different security purposes, such as firewall, load balancing, honeypots, etc. In many situations, the IDS/IPS solution takes an active role in the security of the network.

There are three main types of IDS proposals in literature: signature-based, anomaly-based and hybrid. In signature-based IDS solution, attack signatures are kept in the system and when there is a matching signature, an alert rises. It is effective when detecting the known type of attacks. Regular update of signature databases is required in this solution. Also, the false positive alarm rate is less than the anomaly-based IDS solution. However, it is not suitable against zero-day attacks.

Anomaly-based IDS analyzes the network traffic. It is simple to monitor normal network behavior. When it detects suspicious behavior, which is different from the normal network traffic, it gives an

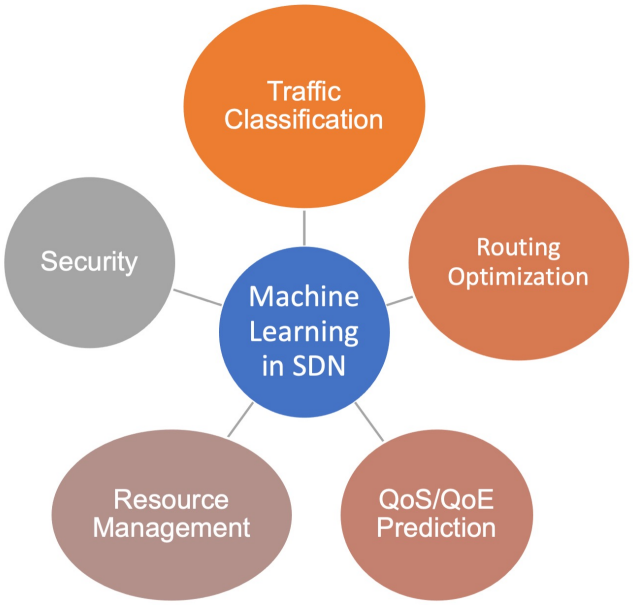

Figure 10: Machine Learning in SDN

alert. It is effective against zero-day attacks, but it can also create false alarms. It also requires clean normal network traffic for baseline.

Hybrid techniques take advantage of both IDS solutions. They aim to decrease the false alarm rate and increase the detection of novel attacks. In the literature, hybrid IDS provides better detection rate/accuracy. However, training takes more time than the other two methods due to its complex structure.

Another division of IDSs is based on network level or host level. A network-based IDS model monitors all network traffic and it usually uses agents to gather the network information. A host-based IDS solution only runs in a specific host/device.

There are four different types of machine learning algorithms: supervised, unsupervised, semi-supervised and reinforcement learning. In supervised learning, the model uses labeled data to learn from experience. Every instance in the data set has features and a class label. The model uses features and label to build up a classifier. After training, the model predicts new instances. Classification and regression are the main tasks in supervised learning. Decision tree, Naïve Bayes, support vector machine, multilayer perceptron are examples of supervised learning. The advantages and disadvantages of widely used classifiers are presented in Table 4:

IDS can help the security of SDN and provides great advantages to the security of the controller. There is a number of IDS solutions on SDN. Machine learning based IDSs are proposed in previous research [19]. They achieve a high detection rate with a false positive rate.

In the literature, supervised machine learning based algorithms are used widely in IDS proposals. Decision tree, random forest, naïve Bayes, support vector machines are examples of supervised algorithms. Also, deep learning-based IDS models are used relatively. However, used datasets in research are obsolete and none of them is an example of SDN flow traffic. Also, researchers generally deal with

Table 4: The Advantages and Disadvantages of Classifier

	ADVANTAGES	DISADVANTAGES
Decision Tree	- Handling continuous and discrete data - Simple data classification - Easy implementation	- Pruning may be required - Overfitting
Random Forest	- Robust against overfitting - Work on large dataset	- Handling imbalanced data set - Training speed
k-NN	- Low parameter - Also unsupervised approach	- Intention memorize instances
SVM	- Working on linear and non-linear data - Handling high dimensional data	- Training time - Vulnerable to noise in the data
Bayes	- Simple implementation	- Not compatible with continuous data - Data should be "Normal Distribution"
Deep Learning	- Effective in many different tasks	- Requiring high dimensional data
RL	- Data is not required. - Effective on human based task (Trial and learn)	- Large Action – State space

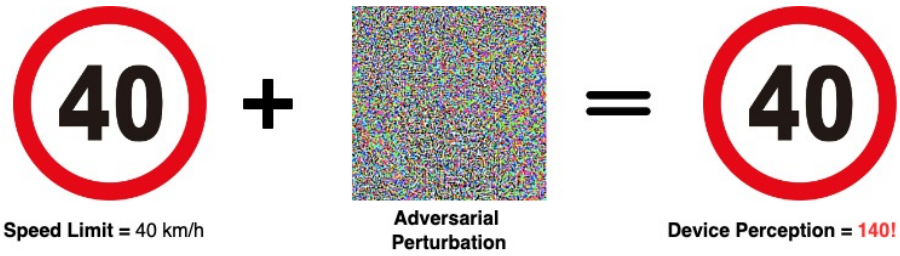

Figure 11: The Noise Effect on Traffic Sign

one specific class, such as DoS or malware detection. The algorithms of some IDS proposals with basic details are presented in Table 3.

3.2 Adversarial Attacks

Despite impressive achievements made by machine learning algorithms (especially in deep learning), they are easily tricked by modified input data. In many cases, modified data is nearly similar to the original data, but slight perturbation on these data heavily deteriorates machine learning algorithms' accuracy. Adversarial attacks target machine learning models severely.(See 11)

Adversarial attacks in the network security domain have not yet been explored sufficiently [62]. In the literature, there are some papers on adversarial attacks in cyber-attacks against conventional networks, but in the SDN domain, only a few are available [63]. Also, there are several IDS proposals for SDN, yet these lack a robust IDS design against adversarial attacks.

Table 5: Some of the IDS Proposals in SDN

Learning Method	**Dataset**	**Number of Features**	**Attack Detection/ Classification**	**Accuracy**
RF [15]	KDD 99, NSL-KDD	41 features	Attack Classification	85.42% (NSL-KDD), 97.85% (KDD 99)
DT, RF [54]	KDD 99	10 features	Attack Detection	82,48 % (DT), 98.75% (RF)
DT, BayesNet, Naïve Bayes [55]	Longtail	-	Attack Detection	86.19% (DT), 91.68% (Bayes Net), 87.78% (Naïve Bayes)
k-NN, k-medoids [56]	Real-time	-	Attack Detection	90% (k-NN)
SVM [57]	Real-time	-	Attack Detection	88.7%
KPCA+SVM [58]	Real-time	27 features	Attack Detection	98,55% 98,71% 98.90%
SVM [59]	KDD 99	23 features	Attack Classification	97.55%
Neural Network [60]	NSL-KDD	6 features	Attack Detection	75.75%
Recurrent NN [60]	NSL-KDD	6 features	Attack Detection	89%
DNN [61]	ISCX	20 features	Attack Detection	97.56%

One of the earliest papers about the adversarial attacks is [64]. They present a taxonomy of adversarial attacks against the IDS. According to the researchers, three key elements for the evaluation of adversarial attacks are measurement, classification and response in the network domain.

Studying the taxonomy of adversarial attacks is important to recognize the differences among various types of attacks. In [64], researchers identify six types of adversarial attacks: evasion, over-stimulation, poisoning, denial of service, response hijacking, and reverse engineering. In evasion attacks, attackers aim to evade detection by changing network feature characteristics. Overstimulation attack is different in that it generates many false alerts to distract operator response. In poisoning attacks, attacker-crafted data are inserted into the training data to mislead the model and reduce detection accuracy. DoS attack overloads IDS sensors so that IDS drops some packets or allows the passage of malicious traffic. Response hijacking attacks generate incorrect alert descriptions and mislead the IDS. Reverse engineering is a technique for gathering information about the IDS.

In [24], researchers classify adversarial attacks according to two properties: influence, which indicates whether the attack is performed at the training or testing phase, and violation, which indicates whether the attack affects the availability or the integrity of the system (See Table 6). They also assert that the attacker's knowledge about the IDS can play a significant role in attack planning. In a black-box attack, attackers do not know anything about the IDS and the training data, while in a white-box attack, they have complete information both about the IDS and the data. Also, [41] describes white-box and black-box attack approaches.

The white-box attacks need access to training data and models, so it is not suitable for realistic ML models. For this reason, the researchers focus on black-box adversarial attacks. In [65], the researchers propose APOLLON. It uses a diverse set of ML classifiers to detect attacks and a reinforcement learning algorithm, Multi-armed Bandits with Thompson sampling for ML classifier selection. The main focus of this research is to prevent learning of IDS behavior and craft adversarial examples. They use the newest dataset but real-time environment testing and feasibility of generated attack are missing.

Feature-level attacks (FLA) are a common way to generate adversarial examples, especially in image classification tasks. Fast Gradient Sign Method (FSGM) [66], Jacobian-Based Saliency Map Attack (JSMA) Method [66], Basic Iterative Model(BIM) [67], Projected Gradient Descent(PGD) [67] and Carlini Wagner(CW) [66] attack methods are examples of FLA. In [68], the researchers use PGD attacks to generate more realistic adversarial attack data. However, these types of attacks are not practical for network data. To generate more practical adversarial examples, Generative Feature-level Attacks (GFLA) are considered [63].

According to [28], evasion attacks can decrease the performance of random forest, logistic regression, and support vector machine classifiers by 50% with only one feature perturbation. The authors use SYN Flood DDoS attacks for adversarial examples and test different types of machine learning classifiers. They only provide adversarial attack results. k-NN displays some robustness against adversarial attacks, but other classifiers suffer heavily from adversarial attacks.

In [69], the researchers consider attacking shallow ML classifiers (naive bayes, decision tree, support vector machine, logistic regression) using a self-generated dataset. The attacks are generated randomly and they use packet delay time, packet loss, packet damage and payload fragmentation etc. for testing the ML classifier's F1-score. They show that shallow ML classifiers are susceptible to adversarial attacks. In [70], the researchers investigate the effects of adversarial attacks on deep neural networks.

They use F1-score as a metric and show that adversarial examples can mislead the model. However, the dataset (NSL-KDD) is outdated and obsolete. In [71], the researchers propose Attack-GAN (Generative Adversarial Network) to generate adversarial network traffic based on SeqGAN. They show that generated adversarial samples easily deceive many IDSs.

3.3 The Countermeasures Against Adversarial Attacks

The general adversarial defense techniques are adversarial learning (training), model robustness design and adversarial perturbation structure destruction [72]. Adversarial learning is a proactive measures that the model is trained with adversarial examples. The main issue for this countermeasures is the quality of adversarial examples. The model robustness design uses the filtering structures for building the model to enhance robustness against adversarial attacks. The adversarial perturbation structure destruction can uses filtering algorithms, noise structure destruction algorithms and noise coverage algorithms.

Mitigating evasion attacks is a challenging task due to the variety of ML models and easy data generation. ML models heavily depend on non-robust features. Some defense strategies against evasion attacks can be adversarial training and preprocessing such as feature squeezing, feature removal, defensive distillation, etc. [73].

To enhance the success of the IDS model, the researchers develop adversarial detector using transfer learning of DNNs [74]. They uses parallel intrusion detection system design, to augment detection capabilities of the multiple classifiers. The model uses ensemble learning and produces better detection results.

In [28], adversarial training is used as a countermeasure against adversarial attacks. Generative Adversarial Network is used for DDoS sampling. Using these samples, the NIDS model is trained and tested with real-time network traffic. The authors argue that a state-of-the-art NIDS with high accuracy can suffer from adversarial attacks, and using adversarial training, the NIDS model can achieve a better detection rate. However, adversarial training is a complicated task and the paper was limited to DoS attacks.

In [75], the researchers propose practical traffic-space evasion attacks for ML-based NIDS. The attack is successful to achieve a rate of more than 97% in half of the cases. When they compare the performance of defense schemes, they observe that adversarial training has a very limited and unstable defensive approach against evasion attacks.

In [76], ensemble adversarial training is proposed. It helps to improve robustness against attacks. The effectiveness of this technique relies on different ML algorithms. However, a common problem in adversarial training is that if the model is trained with only some types of attacks, an attacker can use other attack types.

Finding the most vulnerable features and removing them from training data can also be a good choice against evasion attacks. In [77], the researchers remove the most vulnerable features from the dataset. This defense method decreases model complexity and reduces attack surfaces. However, removal of features decreases model accuracy.

Table 6: Influence and Violation factor in SDN

	Training Phase	**Testing Phase**
Influence	- Manipulation of training set - Removing specific samples - Known as “poisoning attack”	- Subverting IDS behavior using specific samples during running time
	Availability	**Integrity**
Violation	- To cause lots of false alarms - To disrupt capability of IDS	- To increase false negative rate for classified malicious samples as benign - Known as "evasion attack"

According to [78], adversarial attacks should be considered during the development of a machine learning classifier. They asserted that adversarial examples appear to be in contradiction to achieve high generalization performance. Some of the adversarial examples are hard to distinguish from the regular examples.

CHAPTER 4

THE NETWORK DATASET EVALUATION AND ADVERSARIAL SAMPLES GENERATION

In this thesis, the attacker's main target is conducting an "evasion attack", which means the attacker tries to evade the IDS system by adjusting network traffic features. In this attack scenario, the amount of information about IDS is very limited. The attacker does not know the IDS model algorithms/parameters and cannot influence the training data. Therefore, in this attack model, the attacker can only perturb feature characteristics.

4.1 Feature Selection Process

LightGBM [79] is a type of boosting framework proposed by Microsoft. Gradient boosting decision tree (GBDT) trains a decision tree model and then uses the first model's error to build more successful models sequentially. Each model improves the previous model's accuracy. LightGBM, which is a type of GBDT, combines gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB). The main point in GOSS is that each data instance has different gradients and plays different roles in calculating the information gain. GOSS selects informative samples. There is no native weight for data instances in GBDT and it is obvious that some instances are more important than others.

This algorithm proposes that the exclusive features are bundled safely to a single feature. LightGBM has a "feature_importance" function, which provides the importance of a feature, based on how many times the feature is used in LightGBM tree creation. This function is utilized as an evaluation method in our work.

4.2 ML Classifiers and Metrics

For classification task, Random Forest (RF) and k-Nearest Neighbor(k-NN) are selected. Random Forest is an ensemble method and can handle overfitting problem. It is also robust against adversarial attack. For these reasons, one of the classifiers should be Random Forest. Another classifier is k-NN. It is a lazy algorithm and non-parametric model. It provides higher accuracy in the previous IDS research.

Overall accuracy, recall, precision, and F1-Score are used for evaluation metrics. These are common in literature and can help us for comparison of robust model.

Table 7: Instances Size in InSDN Attack Dataset

	Instances Sizes
Brute-Force Attack	295
DDoS	73529
DoS	1145
Probe	61757
U2R	17

4.3 InSDN Dataset

InSDN dataset was created on the SDN based topology and it consisted of SDN based attacks. OVS server was used, and six different attack types were represented: botnet, brute force attack, DoS, DDoS, web attack and probe attack. Three different network flow data files were created under different circumstances: benign network dataset, SDN dataset and attack dataset. For each of them, there are totally 83 features. Attack types and instance sizes in cyber attack dataset are presented in Table 7:

4.3.1 Feature Importance for InSDN Dataset for Multiclass Classification

For feature selection purpose, LightGBM classifier is trained with 50% of 136,743 instances in the attack dataset. The other half of the dataset is used for validation purpose. The LightGBM classifier parameters are selected according to their baseline capabilities. The objective function of the classifier is “Multiclass classification”, evaluation metric is “multi_logloss”, “learning_rate” is 0.03 and “tree_learner” is “voting.”

According to the feature importance result (See Fig. 12), “Flow_IAT_MinMin time between two packets sent in the flow”,"Flow_Duration", “Bwd_IAT_Min Min of time between two packets sent in the backward direction” “Bwd_Pkts/s Number of backward packets per second” and “Bwd_IAT_Tot Sum of time between two packets sent in the backward direction” are top 5 features. For IDS development, top 16 features are selected for classifier’s training.

The top selected features generally come from packet-based and flow-based attributes rather than flag-based or network-identifiers attributes. During the research, different train-validation-test ratio is set, and selected features generally come from packet-based and flow-based attributes. The results derived from InSDN dataset and different network topology show that packet-based and flow-based attributes are important for attack classification.

4.3.2 Classification Result of InSDN Dataset: Attack Classification

Random Forest and k-NN classifier are trained with attack dataset. 5-Fold cross validation is applied, multiclass classification (attack classification) is conducted. The result of overall accuracy is 99.71% with std. 0.01 in Random Forest and 99.78% with std. 0.01 in 5-NN. The other evaluation metrics are presented in Table 8 and 9.

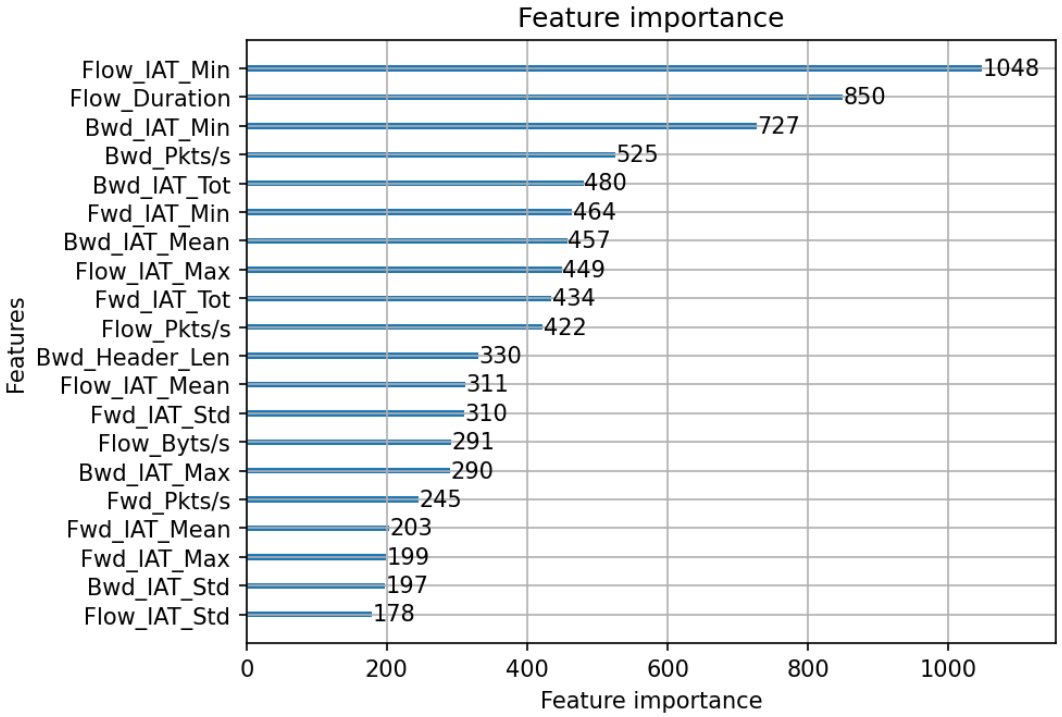

Figure 12: InSDN Dataset, The Number of Times Feature is used in training

Table 8: Evaluation metrics of InSDN Dataset, Classifier: Random Forest, Multiclass Classification, Overall Accuracy 99.71% with std. 0.01

	Precision	**Recall**	**F1-Score**
Brute-Force Attack	0.36	0.38	0.37
DDoS	1.00	1.00	1.00
DoS	0.99	0.99	0.99
Probe	1.00	1.00	1.00
U2R	0.83	1.00	0.91

Table 9: Evaluation metrics of InSDN Dataset, Classifier: k-NN (5-NN), Multiclass Classification, Overall Accuracy 99.78% with std. 0.01

	Precision	**Recall**	**F1-Score**
Brute-Force Attack	0.68	0.19	0.30
DDoS	1.00	1.00	1.00
DoS	0.98	1.00	0.99
Probe	1.00	1.00	1.00
U2R	0.00	0.00	0.00

The two different classifiers provide above 90% overall accuracy. Random forest classifier is better than k-NN in the context of overall accuracy. The above 95% overall accuracy results can help us to measure the impact of the adversarial attack. The only deficit is that Brute-Force Attack and U2R classes have insufficient instances in InSDN dataset, and these two classes can be affected from adversarial attack more than other classes due under-fitting.

Furthermore, if two classifiers cannot achieve respectable results at two different purposes (attack detection or attack classification), the impact of adversarial attack could be biased. The unexpected result could be titled as "the capacity problem".

4.4 SDN-IoT Dataset

SDN-IoT dataset was created on the SDN topology using IoT devices ([80]). ONOS controller was selected for orchestration of network and OpenFlowSwitch was used for data plane. The five different attack type were implemented with different configurations: DoS, DDoS, port scanning, OS fingerprinting and fuzzing. Two different datasets were presented and only difference of them is number of IoT devices used in the topology: 5 vs. 10, respectively. The flow entries were used for feature creation and as a result, the dataset contains 33 features. For IDS evaluation, the researchers present equal instance in two datasets. (For each category 3500 instances, totally 210,000)

4.4.1 Feature Importance for SDN-IoT Dataset for Multiclass Classification

LightGBM classifier is trained with 50% of 210,000 instances in the 5-IOT-SDN dataset. The other half of the dataset is used for validation purpose. The LightGBM classifier parameters are selected according to their baseline capabilities, too. The objective function of the classifier is "Multiclass classification", evaluation metric is "multi_logloss", "learning_rate" is 0.03 and "tree_learner" is "voting."

According to feature importance result (See 13, "N_IN_Conn_P_DstIP Number of inbound connections per destination IP", "Max Maximum duration of aggregated records", "TnP_Per_Dport Total number of packets per destination port", "Dur Record total duration" and "Sum Total duration of aggregated records" are top 5 features. For IDS development, top 14 features are selected for classifier's training.

4.4.2 Classification Result of SDN-IOT Dataset: Attack Classification

Random Forest and k-NN classifier are trained with 5-IoT-SDN dataset. 5-Fold cross validation is applied, multiclass classification is conducted. The result of overall accuracy is 98.24% with std. 0.02 in Random Forest and 92.37% with std. 0.03 in 5-NN. The other evaluation metrics are presented at the table 10 and 11.

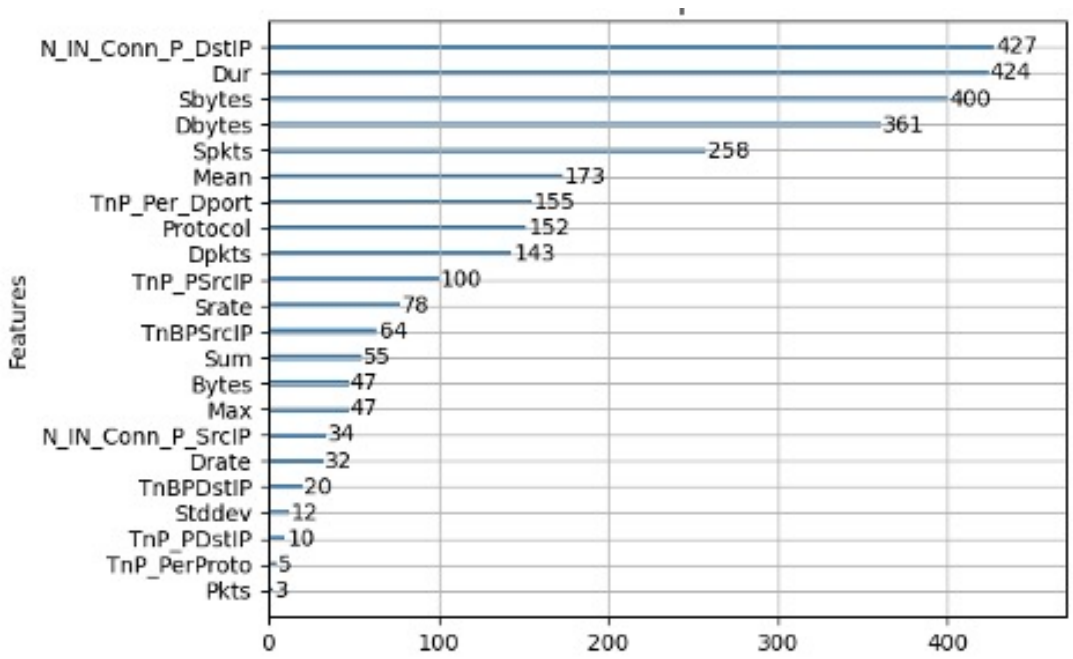

Figure 13: SDN-IoT Dataset, The Number of Times Feature is used in training

Table 10: Evaluation metrics of SDN-IoT Dataset, Classifier: Random Forest, Multiclass Classification, Overall Accuracy 98.24% with std. 0.02

	Precision	**Recall**	**F1-Score**
Normal	0.97	0.93	0.95
DoS	0.98	0.99	0.99
DDoS	0.98	0.99	0.99
Port Scanning	0.98	0.99	0.99
OS and Service Detection	0.99	0.99	0.99
Fuzzing	0.99	0.99	0.99

Table 11: Evaluation metrics of SDN-IoT Dataset, Classifier: k-NN (5-NN), Multiclass Classification, Overall Accuracy 92.37% with std. 0.03

	Precision	**Recall**	**F1-Score**
Normal	0.88	0.65	0.75
DoS	0.92	0.98	0.95
DDoS	0.93	0.98	0.95
Port Scanning	0.93	0.99	0.96
OS and Service Detection	0.96	0.96	0.96
Fuzzing	0.91	0.99	0.95

Table 12: Instance sizes in the CICIDS 2017 dataset

Traffic Type	Number of Instances
Benign	2,358,036
DoS Hulk	231,073
Port Scan	158,930
DDoS	41,835
DoS GoldenEye	10,293
FTP Patator	7938
SSH Patator	5897
DoS Slow Loris	5796
DoS HTTP Test	5499
Botnet	1966

4.5 CICIDS 2017 Dataset

The CICIDS 2017 dataset consists of benign traffic and up-to-date network attacks. CICFlowMeter was used for feature creation and 83 features were created. They used a wide range of network attacks such as web based, brute force, DoS, DDoS, infiltration, heart-bleed, bot and scan attack types. There are 2,830,743 instances in total, distributed as shown in Table 12.

4.5.1 Feature Importance for CICIDS 2017 Dataset for Binary Classification

For feature selection purpose, LightGBM classifier is trained with 30% of 59,446 instances in the CICIDS 2017 dataset. The other half of the dataset is used for validation purpose. The LightGBM classifier parameters are selected according to their baseline capabilities. The objective function of the classifier is "Multiclass classification", evaluation metric is "multi_logloss", "learning_rate" is 0.03 and "tree_learner" is "voting."

According to feature importance result (See Fig.33), "init_win_bytes_forward", Destination_port, "init_win_bytes_backward", "Flow_IAT_Min Min time between two packets sent in the flow" and "Fwd_IAT_min Min of time between two packets sent in the forward direction" are top 5 features. For IDS development, top 26 features are selected for classifier's training.

The top selected features generally come from packet-based and flow-based attributes rather than flag-based or network-identifiers attributes. The results derived from two different dataset and different network topology show that packet-based and flow-based attributes are important for attack classification.

4.5.2 Classification Result of CICIDS 2017 Dataset: Attack Detection

Random Forest and k-NN classifier are trained with attack dataset. 5-Fold cross validation is applied, binary classification (attack detection) is conducted. The result of overall accuracy is 99.85% with

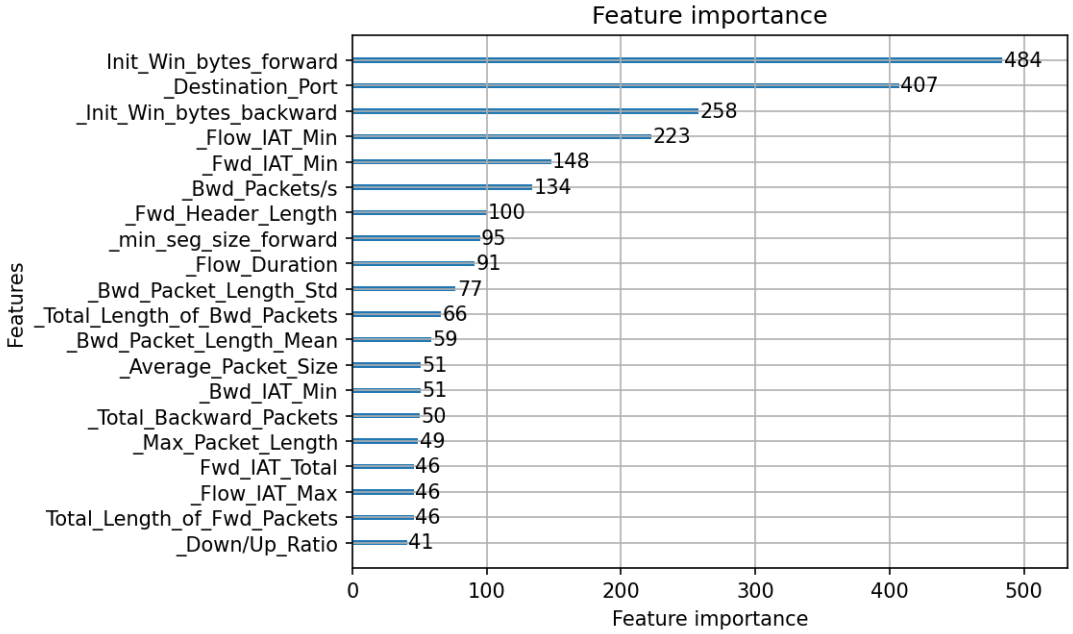

Figure 14: The feature importance plot of CICIDS 2017

Table 13: Evaluation metrics of CICIDS Dataset, Classifier: Random Forest, Binary Classification, Overall Accuracy 99.85% with std. 0.001

	Precision	**Recall**	**F1-Score**
Benign	1.00	1.00	1.00
Malicous	1.00	1.00	1.00

std. 0.001 in Random Forest and 99.19% with std. 0.001 in 5-NN. The other evaluation metrics are presented at the below table (See Table 13,14):

The two different classifiers provide above 90% overall accuracy. Random forest classifier is better than k-NN in the context of overall accuracy. The above 95% overall accuracy results can help us to measure the impact of adversarial attack. The only deficit is that Brute-Force Attack and U2R classes have insufficient instances in InSDN dataset, and these two classes can be affected from adversarial attack more than other classes due to under-fitting.

Table 14: Evaluation metrics of CICIDS Dataset, Classifier: k-NN (5-NN), Binary Classification, Overall Accuracy 99.19% with std. 0.002

	Precision	**Recall**	**F1-Score**
Benign	1.00	0.99	1.00
Malicous	0.97	0.98	0.98

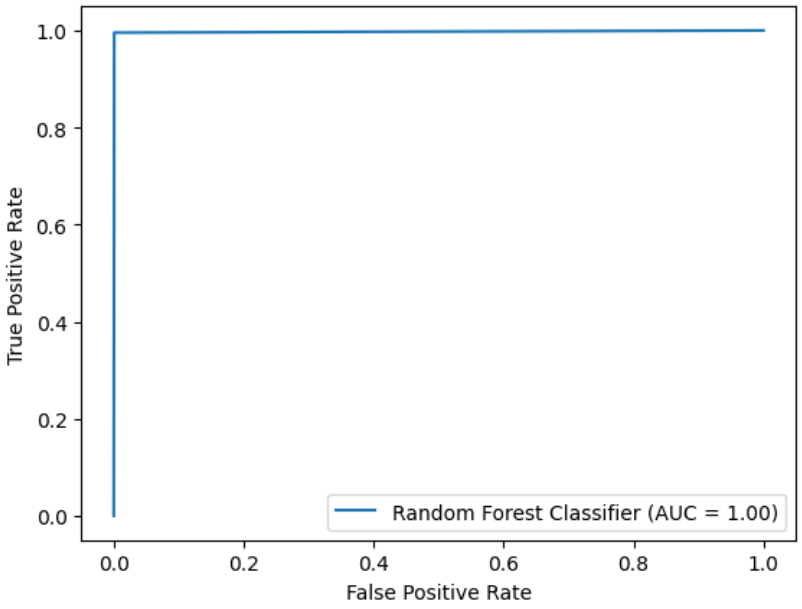

Figure 15: The ROC Curve of CICIDS 2017

Furthermore, if two classifiers cannot achieve respectable results at two different purposes (attack detection or attack classification), the impact of adversarial attack could be biased. The unexpected result could be titled as "the capacity problem".

4.6 The Attacker's Knowledge and Capability

4.6.1 Knowledge of the Adversary

The knowledge about the target system can change the adversary behavior. If an attacker does not know anything about the target system, this is called as a "Black-box attack" (Also known as a blind attack"). The adversary can make an assumption about the target system. Similar machine learning classifiers are tested with perturbated features. On the other hand, the attacker can know the target system. This is called as a "White-box attack". It is easier and more powerful than a "Black-box attack" due to broad information about the system.

4.6.2 Capability of the Adversary

The attacker can use a different type of attack against the target system. The adversary can change the parameters of features to increase the impact of the attack. The payload size, packet rate, packet count and flow statistics were used in previous research. Also, different types of network features, such as the number of flow packets per second or protocol flag could be used in adversarial examples.

4.7 Generative Feature-Level Attacks

Generative Feature-level Attacks [63] use generative algorithms such as Generative Adversarial Networks to understand the network traffic features and aim to create realistic network data.

Generative Adversarial Network is a deep learning model with the capability of learning the distribution of the input data and generation of adversarial examples. Typically, GAN has two different deep learning modules: a discriminator and a generator. Whereas the discriminator works as a classifier of fake/real data, the generator always wants to fool the discriminator by capturing the training data distribution to generate fake examples.

Traditional GAN is trained with binary cross-entropy loss. Due to the training scheme, the loss function is an average of the cost for the discriminator misclassified real and fake observations. When the cost function results in a higher value, the discriminator performs worse. So, the generator wants to maximize the cost function while the discriminator wants to minimize it. When the discriminator gets better during the training, the feedback for the generator loses gradient. This is called the "Vanishing gradient problem".

Another problem for traditional GAN is mode collapse. Sometimes, the generator wants to produce the same fake output due to understanding the distribution of input data which is diverted from the discriminator. So, the feedback for the discriminator is not viable and the learning process is futile.

To overcome these two problems, different types of GAN are proposed in the literature. For synthetic data generation which can be used for adversarial tasks, Wasserstein GAN and CTGAN are two well-known GAN types.

4.7.1 Wasserstein GAN

Unlike the traditional GAN's BCE loss function, Wasserstein GAN uses Earth Mover's Distance (EMD) as a cost function. It is a distance metric between two distributions and measures how different two distributions are by estimating the amount of effort it takes for the generated distribution to equal the real distribution. Instead of decreasing the loss function at "zero", the cost function continues to grow regardless of how far apart the distribution is.

Simply, W-loss works via approximating the EMD between the real and generated distribution. The equation of the loss function is:

$$E(c(x)) - E(c(g(z))) \tag{8}$$

where g is generator, c is critic (Discriminator), c(x) is the cost function and g(z) is the value of the generator's output. The function calculates the differences between the expected values of the predictions of the discriminator. WGAN introduces a critic instead of traditional GAN's discriminator. It predicts the Wasserstein distance to maximize the objective function. The key changes compared to the original GAN algorithm are:

- After every update on the critic function, it limits the weights to a fixed range l-c, cl,
- As an optimizer for model training, RMSProp is used,

- Discriminator helps estimation of Wasserstein metric between data distribution.

4.7.2 CTGAN

Another type of GAN used for synthetic data generation is CTGAN [81]. According to [81], tabular data has unique properties for data generation. There are mixed data types, non-Gaussian distribution, multimodal distribution, learning from one-hot encoded vectors, and highly imbalanced categorical columns. To deal with these problems, CTGAN is proposed. CTGAN uses mode-specific normalization to overcome non-Gaussian distribution and a conditional generator and training-by-sampling to deal with the imbalanced discrete columns.

In CTGAN, for each continuous column C_i, variational Gaussian mixture model estimates the number of modes mi and fit a Gaussian mixture. Each Gaussian mixture has the weight and standard deviation of a mode. Later, these metrics are used for data generation with the help of mode probability density.

Traditional GAN is trained with a standard multivariate normal distribution. When a model faces imbalance in categorical columns, sampling does not provide enough data for minor categories. So, to overcome "class imbalance", CTGAN uses conditional generators and training-by-sampling.

The main differences between WGAN and CTGAN are, (i) cost function, (ii) model optimizer, (iii) handling imbalance in categories and (iv) activation function of the GAN. Based on these differences, generated data by WGAN and CTGAN may be different from each other.

4.8 Feature-Level Attacks (FLA)

This type of attack directly manipulates the network data features. The degree of perturbation is limited within l_p norm. White-box and black-box attacks comprise these types of attacks.

Fast gradient sign method is proposed to search for adversarial examples easily. It is the simplest and the most widely used untargeted adversarial attack method. One-step gradient update is performed along the direction of the sign of the gradient. Jacobian-based saliency map attack uses a Jacobian matrix for the creation of adversarial examples via l_0 distances. Carline and Wagner Method is an iterative gradient method that uses the Adam optimizer and Tanh function to produce the adversarial attack. Projected Gradient Descent(PGD) can generate more precise perturbation using uniform random distribution to find suitable adversarial examples.

4.9 Attack Generation Structure: GAN based Adversarial Samples

The generator and discriminator work in a competitive manner in the GAN structure. The generator's main task is to fool the discriminator by generating realistic adversarial instances. It takes a fixed-length random input with noise to check the discriminator's learning process. Simply, the attack generation structure is presented in Fig. 16.

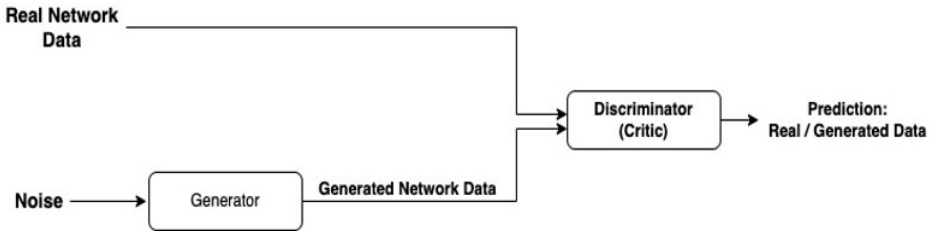

Figure 16: The structure of WGAN

4.9.1 The Generator

To train the GAN, the generator creates a random input with noise, then the discriminator predicts whether it is a real or generated sample. The underlying data distribution dictates the generator's learning pace. Due to the characteristics of the network data, some features can be categorical and these features should be handled differently. One of the suitable solutions for these features is to encode categorical features then generate based on the generator output of the binarized feature. Also, each feature must comply with domain/protocol constraints and appropriate ranges of the traffic features such as TCP packet could not start with the "FIN" flag and packet length could not exceed 1526 bytes.

4.9.2 The Discriminator

The discriminator of the GAN is responsible for the detection of the counterfeited instances. Owing to the loss function, the discriminator evaluates its weights and passes it to the generator. In the beginning, it is an easy task for the discriminator. This process repeats until the discriminator cannot detect real or generated data.

After the training process, it mimics the capabilities of the IDS. These capabilities can be used to detect adversarial examples by separating from the GAN's structure.

CHAPTER 5

ROBUST AUTOENCODER BASED IDS MODEL

Rather than relying on the detection capacity of only one machine learning classifier, multiple machine learning classifiers can be used as an ensemble when providing robustness to an IDS. Countermeasures against adversarial attacks should be provided not only to focus on adversarial attack detection, but also on overall accuracy.

According to attack principles and behaviors, each attack has its functional features. While creating adversarial attack examples, we should keep in mind that adversarial generated data should keep attack functional features to avoid invalidating attack traffic. These data do not convey benign data characteristics.

Autoencoders have been chosen for many different ML tasks such as denoising data or anomaly detection [82]. An autoencoder trained on a data distribution can reconstruct new data with a small reconstruction error. If attackers produce evasion data, these data should have higher reconstruction errors because evasion data should be different from real network data. This point helps the detection of adversarial attacks.

This section presents the RAIDS structure. Here, the details of the IDS structure and descriptions on the parts of the model are presented.

5.1 Autoencoders

A neural network can learn the relationship between the input data and their given label. This is called supervised learning. On the other hand, if we do not have labeled data, we can also use a special neural network for understanding the core definition of the data.

Sparse, denoising, and variational autoencoders are types of the autoencoder, each with its own unique characteristics and functionalities. Sparse autoencoders (SAEs) are useful for their ability to detect sparse representations of the input data, which means only a small number of units in the hidden layer are activated at a time. For this reason, sparse autoencoders are effective for feature selection and dimensionality reduction [83]. Besides, sparse autoencoder has ability to learn important structures of deep networks due to their high capacity for learning, which necessitates architecture regularization [84].

Denoising autoencoders (DAEs) are provided to learn robust representations of data by training the network to recreate the original input from a distorted version of the original data. This ability helps the DAE to filter out noise and extract essential features from the input data. It is useful for image detection tasks [85].

Variational autoencoders (VAEs) are a type of autoencoder that understand the probability distribution of the input data. VAEs are capable of extricated learning, which allows them to separate and manipulate specific features of the input data, making them useful for tasks such as image generation and manipulation [86].

In RAIDS structure, the autoencoder task is to learn key latent space of the input data and to grasp informative representation of the data. For these reasons, sparse autoencoder is chosen for RAIDS development.

An autoencoder is an algorithm that gives us outputs like input data. It learns an "informative" representation of the data and reconstructs the output. The main components of the autoencoder are the encoder and decoder unit (Fig. 17). Together, they form a neural network. An encoder takes the input data and at each step, it compresses the data and creates a latent space. A decoder reconstructs the data using the latent space.

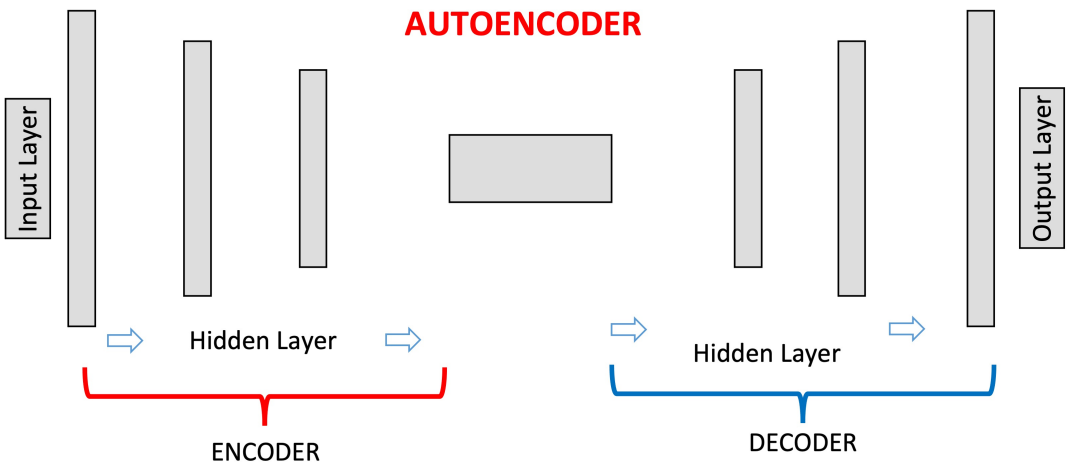

Figure 17: The structure of an autoencoder

Each autoencoder wants to regenerate input data as similarly as possible. An important term to measure the success of autoencoders is the reconstruction error (RE). It is a metric that gives us an indication of how well our autoencoder can reconstruct the input. The RE can be calculated by MSE as follows;

$$RE = MSE = \frac{1}{M}\sum_{i=1}^{M} |x_i - \overline{x}_i|^2 \tag{9}$$

where x_i is the input data and $\overline{x}_i$ is the reconstructed output. In previous research, the reconstruction error was used by researchers to detect an anomaly. The main idea is that since the autoencoder has learned benign data, it cannot reconstruct unlearned data with low reconstruction error. Therefore, if an autoencoder takes the anomaly data, it can only reconstruct this data with high reconstruction error.

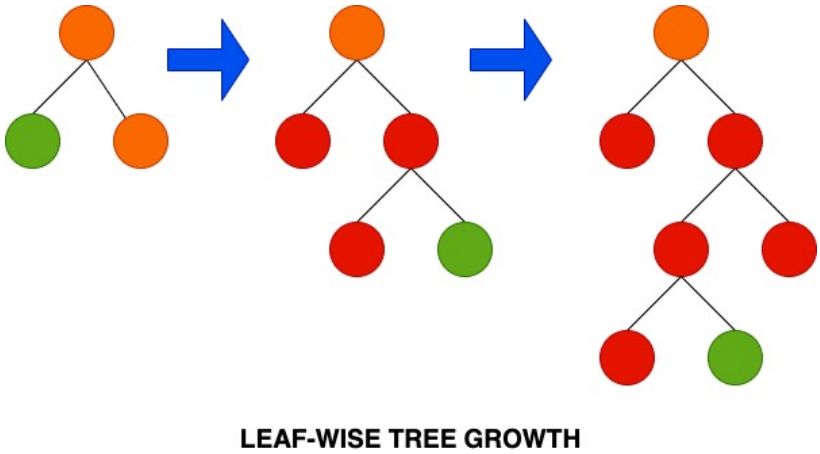

Figure 18: The LightGBM's Tree Growth

If an autoencoder can be trained with only legitimate network data, any adversarial example will be reconstructed with high reconstruction error. Using two different autoencoders, if one is trained with real benign data and the other is trained with real attack data, together they can be used to detect both malicious traffic and adversarial attack. The key point is if an autoencoder can be trained with only legitimate network data, any adversarial example will be reconstructed with high reconstruction error.

5.2 Machine Learning Classifiers

Previous research suggests that using ensemble models could be a good idea as a countermeasure against adversarial attacks [73]. LightGBM is a gradient-boosting tree algorithm and has the capability of creating a bagging tree [79] (See Fig. 18). It uses gradient-based one-side sampling and exclusive feature bundling to decrease training time and to find the optimal split point.

LightGBM bins continuous features to construct the tree model. Model success comes from continuous features' histogram. Also, LightGBM has advantages to handle imbalanced datasets [87]. Furthermore, the detection ability of LightGBM has been widely accepted in several data mining and machine learning contests such as Kaggle. For these reasons, LightGBM is a suitable ML algorithm for attack detection tasks without need for long training time.

At the evaluation step of the dataset features, some features are more susceptible to perturbation. Features such as port number, protocol number, TCP SYN packet, TCP FIN packet, and acknowledgment number are individual packet features and they are not suitable for perturbation due to network traffic protocol, as each TCP connection begins with the SYN flag and finishes with the FIN flag. However, statistical and time-related features such as total forwarded packets, forwarded packet length mean, and backward packets per second are easily manipulated due changeable behavior of network flows.

If we have m features and n features from this group with a high possibility of being manipulated by the attacker, we can develop a group of machine learning classifiers such that these n features are not used together (i.e., they are split up) in the training process of any of the classifiers. Given that n features selected from the dataset feature set and a random number of machine learning classifiers are

trained with n features, the attacker will face difficulty in manipulating all machine learning classifiers due to randomness and the fact that each machine learning classifier uses a different feature set.

5.3 Design of RAIDS v2.0

The proposed IDS structure depends on the reconstruction error of the autoencoder and randomness of the machine learning classifier. The attacker does not know the IDS model structure and feature set in the evasion attack scenario. Ensemble learning plus randomness of the feature set can also increase countermeasures against adversarial attacks.

The value of the reconstruction error and the classifier's prediction are used as another LightGBM classifier's training data. Through this stage, the LightGBM classifier will become accustomed to the effect of adversarial attacks and be able to distinguish an adversarial attack. The flow chart of the proposed model is presented in Fig. 19).

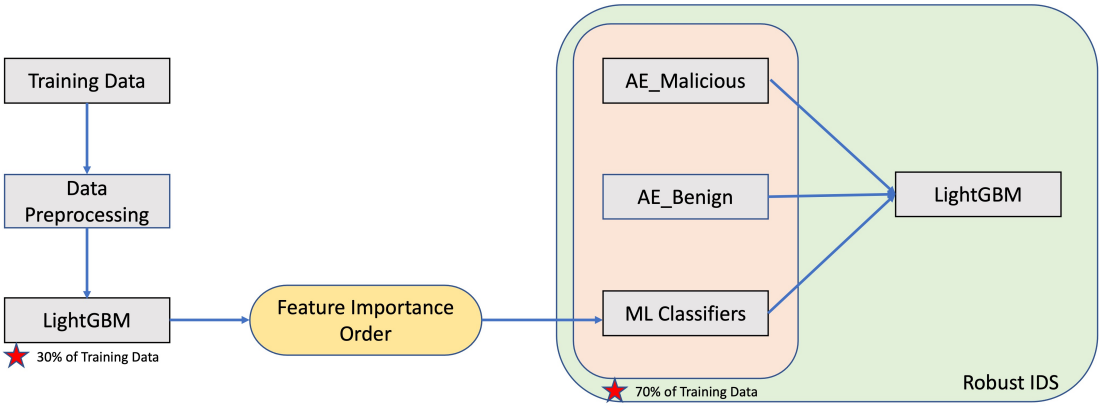

Figure 19: Flowchart of the RAIDS v2.0 model

Part I: Feature Importance Order and ML Classifier Designment

To minimize the perturbation of the feature sets by an attacker and provide ensemble learning for better detection, multiple machine learning classifiers are trained with legitimate network traffic. Some statistical features such as header size, average inflow bytes, average size of the packet per connections etc. are more susceptible to perturbation due to the attacker's manipulation on these features easily.

LightGBM has a built-in "feature_importance" function, which provides the importance of a feature, based on how many times the feature is used in LightGBM tree creation during training of LightGBM. For this research, "split" type feature importance (number of times the feature is used in a gradient boosting model) is used for ranking features. After the training of LightGBM, the feature importance order is decided.

After the feature ranking is decided by LightGBM, ML classifiers are ordered randomly. (E.g., First ML classifier is SVM, second ML classifier is k-NN, third ML classifier is decision tree) Then, the features are distributed to the classifiers according to importance order (First feature goes to first clas-

sifier, second feature goes to second classifier, etc.) So, the RAIDS v2.0 model can create a number of machine learning classifiers such as decision tree, k-NN, support vector machine random order with different feature set. This will provide a random feature set distribution for each machine learning classifier.

Later, each machine learning classifier is trained with only legitimate network data and its task is to decide whether an instance is benign or attack. They provide binary classification results, and these results are used as training data for the LightGBM classifier.

Part II: Measuring RE and Distinguishing Adversarial Examples

If an autoencoder is trained with real network data, any unknown examples should produce a high reconstruction error. This can be used for the detection of unknown classes. For this reason, two autoencoders are trained: one is trained with real benign data and other is trained with real attack data. They give reconstruction error for every instances.

Part III: Training the RAIDS v2.0 model

After training the two autoencoder and machine learning classifiers, the last stage LightGBM is trained with the results of these (Fig. 20). The two autoencoders provide a reconstruction error value (numeric) and machine learning classifiers provide a classification result as benign/malicious. These results are the input data for the last stage LightGBM to classify traffic as normal traffic or attack traffic (real malicious traffic plus adversarial attack).

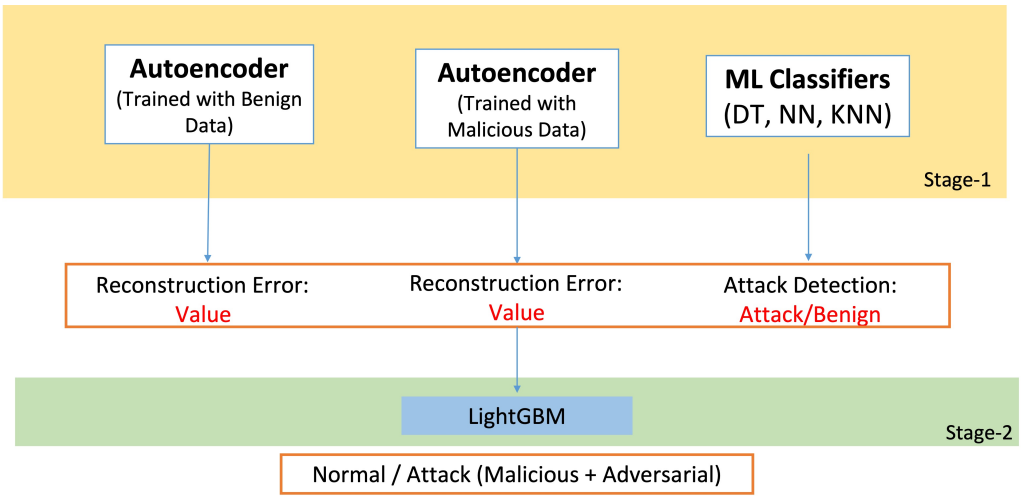

Figure 20: Training phases of the proposed RAIDS v2.0 model

5.4 Training Steps of the RAIDS v2.0

The step-by-step description of the training phase of RAIDS v2.0 is detailed:

1. Feature importance was determined by LightGBM's "feature_importances" function using 30% of the training set.

2. The features were distributed to n different sets according to feature importance order. Baseline machine learning classifiers such as Decision Tree, k-NN and SVM were constructed using the divided feature set.

3. Each ML classifier was trained with a completely different feature set. Besides, one of the autoencoders (the benign autoencoder) was trained with only benign data and the other was trained with only real attack data.

4. After the training phases of classifiers and autoencoders, all ML classifiers and autoencoders were tested with the training dataset, and classifiers' prediction results and autoencoder reconstruction error values were tabulated to feed the Stage-2 LightGBM in Fig. 20.

5. These tabulated data were then used for training the Stage-2 LightGBM. Hence, RAIDS v2.0 training was completed.

5.5 Pre-Model: Design of RAIDS v1.0

In the first RAIDS design, RAIDS v1.0, the GAN's discriminator is the key element of the RAIDS v1.0. The value of the reconstruction error and the LightGBM classifier's prediction are used as another LightGBM classifier's training data. However, the autoencoder is only responsible for measuring the reconstruction error of the input whether original network instance or generated adversarial instance. Besides, during training of the GAN, the discriminator is trained for predicting original or adversarial instances. In the second stage, LightGBM is trained with the autoencoder's reconstruction error, discriminator's prediction result, and LightGBM's attack detection. The stage of the proposed model is presented in Fig. 21).

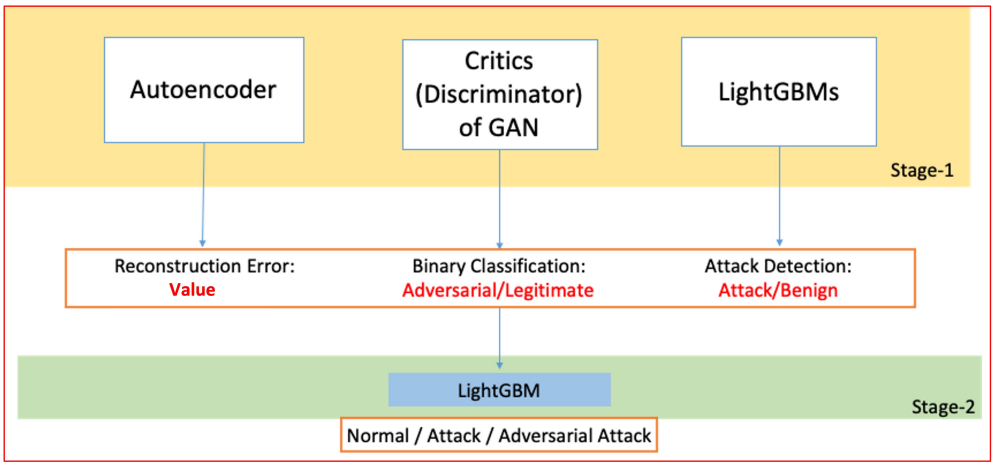

Figure 21: Training phases of the RAIDS v1.0 model

5.6 Training Steps of the RAIDS v1.0

The step-by-step description of the training phase of RAIDS v1.0 is detailed:

1. Feature importance was determined by LightGBM's "feature_importances" function using 30% of the training set.
2. LigthGBM classifier was trained with top 30 features according to feature importance order. Besides, an autoencoder was trained with only original network flow data and the discriminators are taken from the trained WGAN (Using crafting adversarial examples).
3. After the training phases of LightGBM classifier and autoencoder, they tested with the training dataset, and LightGBM's prediction results, autoencoder reconstruction error and discriminator's prediction value were tabulated to feed the Stage-2 LightGBM in Fig. 21.
4. These tabulated data were then used for training the Stage-2 LightGBM. Hence, RAIDS v1.0 training was completed.

5.7 The Motivation of Change in RAIDS v1.0 to v2.0

Firstly, in RAIDS v1.0, the critic (discriminator) is extracted from WGAN after the training of the GAN. However, this part of the GAN may be dependent on the dataset. Each dataset has different feature flow parameters and different scales. Secondly, measuring the reconstruction error of benign and malicious network flow data can provide better detection results against adversarial instances. Thirdly, dividing all features of the dataset by n number of different ML classifiers can make it difficult for attackers to manipulate features. Each ML classifier is trained different feature set. This provides robustness against adversarial attacks.

The key differences between v1.0 and v2.0 are (i) v1.0 calculates the reconstruction error of the only original network flow data, however, v2.0 calculates benign and malicious network flow data separately, (ii) v2.0 uses different types of ML classifiers in an ensemble approach, but v1.0 uses only LightGBM, (iii) v1.0 depends on discriminator of WGAN.

CHAPTER 6

DESIGN OF EXPERIMENTS

6.1 Datasets

For our evaluation, two different datasets are used: InSDN [34] and CICIDS 2017 [53]. Both datasets are up-to-date and consist of modern attack scenarios. Besides, the InSDN dataset was created on an SDN environment and includes an SDN-specific attack pattern.

6.1.1 InSDN

InSDN is one of the first examples of comprehensive SDN datasets in the intrusion detection domain. CICFlowMeter was used for feature creation and in total 83 features were created for attack classification. InSDN was created on an SDN based topology and it consists of SDN based attacks. Open vSwitch was used, and six different attack types were represented: botnet, brute force attack, DoS, DDoS, web attack and probe attack. Three different files were created under different circumstances: benign network dataset, SDN dataset and cyber attack dataset. For each of them, there are 48 SDN related features. The features used in our research is presented in Table 16. Benign network dataset consists of only benign network data with 68,424 instances. For our research, benign network dataset and SDN dataset are chosen. Attack types and instance sizes are presented in Table 15.

Table 15: Instance sizes in the InSDN datasets

File Name	Traffic Type	Number of Instances
Benign network dataset	Benign	68,424
SDN dataset	DoS	52,471
	DDoS	48,413
	Probe	36,372
	Brute Force Attack	1110
	Web Attack	192
	Botnet	164

Table 16: InSDN Features

Features	Features
Flow Duration	Bwd IAT Min
Tot Fwd Pkts	Bwd PSH Flags
Tot Bwd Pkts	Bwd URG Flags
TotLen Fwd Pkts	Fwd Header Len
TotLen Bwd Pkts	Bwd Header Len
Fwd Pkt Len Max	Fwd Pkts/s
Fwd Pkt Len Min	Bwd Pkts/s
Fwd Pkt Len Mean	Pkt Len Min
Fwd Pkt Len Std	Pkt Len Max
Bwd Pkt Len Max	Pkt Len Mean
Bwd Pkt Len Min	Pkt Len Std
Bwd Pkt Len Mean	Pkt Len Var
Bwd Pkt Len Std	Pkt Size Avg
Flow Byts/s	Fwd Seg Size Avg
Flow Pkts/s	Bwd Seg Size Avg
Flow IAT Mean	Subflow Fwd Pkts
Flow IAT Std	Subflow Fwd Byts
Flow IAT Max	Subflow Bwd Pkts
Flow IAT Min	Subflow Bwd Byts
Fwd IAT Tot	Fwd Act Data Pkts
Fwd IAT Mean	Active Mean
Fwd IAT Std	Active Std
Fwd IAT Max	Active Max
Fwd IAT Min	Active Min
Bwd IAT Tot	Idle Mean
Bwd IAT Mean	Idle Std
Bwd IAT Std	Idle Max
Bwd IAT Max	Idle Min

Table 17: Instance sizes in the CICIDS 2017 dataset

Traffic Type	Number of Instances
Benign	2,358,036
DoS Hulk	231,073
Port Scan	158,930
DDoS	41,835
DoS GoldenEye	10,293
FTP Patator	7938
SSH Patator	5897
DoS Slow Loris	5796
DoS HTTP Test	5499
Botnet	1966

6.1.2 CICIDS 2017

The CICIDS 2017 dataset consists of benign traffic and up-to-date network attacks. CICFlowMeter was used for feature creation and 83 features were created. The features used in our research is presented in Table 18. The top priority of the researchers was generating a realistic dataset with current CVE and attack scenarios. They used a wide range of network attacks such as Web based, brute force, DoS, DDoS, infiltration, heart-bleed, bot and scan attack types. All traffic flows are converted and there are 2,830,743 instances in total, distributed as shown in Table 17.

6.2 Adjustment of Datasets

The scope of this study is limited to attack detection. To this end, all labels in the datasets were changed to either normal or attack. After that, all individual characteristics of network traffic features such as source IP, destination IP, source and destination port numbers, time etc. were deleted.

For the InSDN dataset, normal dataset and SDN dataset are divided with a 70/30% ratio for training/testing purposes(See Fig. 22 and 23). The training set of InSDN consists of 54,624 normal and 124,922 attack instances. The testing set of InSDN consists of 13,800 normal and 13,800 attack instances.

For the CICIDS dataset, 10% of all instances were selected randomly keeping the normal/attack ratio of the original dataset file. The training set consists of 198,152 instances and testing set contains 84,922 instances. Training and testing workflows are depicted in Fig.24. The training set was used for two purposes: to determine feature importance via LightGBM and autoencoders, and ML classifier training. 30% of the training set was used by LightGBM, and then, these features were divided into n parts and used for ML classifier build-up. 70% of the training set was used to train the benign/attack autoencoder and ML classifier. The testing set was used for testing and WGAN/CTGAN training. When WGAN/CTGAN was trained with data, the generator part of the GAN was used for generating specific size adversarial examples. The original testing part of the dataset and GAN generated examples are combined into a new set used to test RAIDS.

Table 18: CICIDS 2017 Features

Features	Features
Flow Duration	RST Flag Count
Total Fwd Packets	PSH Flag Count
Total Backward Packets	ACK Flag Count
Total Length of Fwd Packets	URG Flag Count
Total Length of Bwd Packets	CWE Flag Count
Fwd Packet Length Max	ECE Flag Count
Fwd Packet Length Min	Down/Up Ratio
Fwd Packet Length Mean	Average Packet Size
Fwd Packet Length Std	Avg Fwd Segment Size
Bwd Packet Length Max	Avg Bwd Segment Size
Bwd Packet Length Min	Fwd Header Length
Bwd Packet Length Mean	Fwd Avg Bytes/Bulk
Bwd Packet Length Std	Fwd Avg Packets/Bulk
Flow IAT Max	Fwd Avg Bulk Rate
Flow IAT Min	Bwd Avg Bytes/Bulk
Fwd IAT Total	Bwd Avg Packets/Bulk
Fwd IAT Max	Bwd Avg Bulk Rate
Fwd IAT Min	Subflow Fwd Packets
Bwd IAT Total	Subflow Fwd Bytes
Bwd IAT Max	Subflow Bwd Packets
Bwd IAT Min	Subflow Bwd Bytes
Fwd PSH Flags	Init_Win_bytes_forward
Bwd PSH Flags	Init_Win_bytes_backward
Fwd URG Flags	act_data_pkt_fwd
Bwd URG Flags	min_seg_size_forward
Fwd Header Length	Active Mean
Bwd Header Length	Active Std
Bwd Packets/s	Active Max
Min Packet Length	Active Min
Max Packet Length	Idle Mean
Packet Length Mean	Idle Std
Packet Length Std	Idle Max
FIN Flag Count	Idle Min
SYN Flag Count	

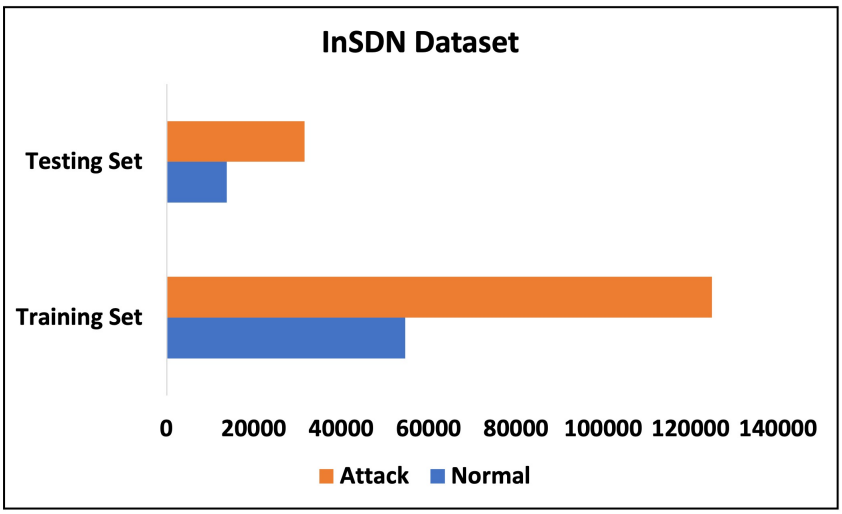

Figure 22: The train/test ratios of the InSDN datasets

6.3 The Generation of Adversarial Examples

The perturbations in our approach are applied to the statistical network traffic features available in the datasets. All features (See Table 16 and 18) are used for adversarial example generation and the model testing. As a practical example, by increasing duration time, header length, or reducing the packet length, we want to fool ML-NIDS. These types of attacks, called feature-space attacks, require higher-level abstraction of an attack's workflow. Both our datasets consist of time-related and statistical features extracted from raw network traffic that are eligible for perturbation. So, all features are targeted by GAN and PGD attacks.

To test the proposed RAIDS, we needed to generate adversarial examples. WGAN and CTGAN are chosen for generative-level feature attack type. Both GANs are trained with only attack data which are obtained from testing data.

The parameters of WGAN are presented in Table 19. Except for the last activation function (Sigmoid), all activation functions are ReLu. RMSprop optimizer is used to train the WGAN. During the training phases, discriminator and generator cost function are checked by loss value to control overfitting. After the training phases, the generator part of WGAN was used in adversarial examples. It created 13,800 adversarial examples for InSDN and 25,000 for CICIDS. For CTGAN, SDV python package was used [88], which automatically selects parameters, to generate adversarial examples of the same size.

In addition to the above, the projected gradient descent (PGD) attack to generate adversarial examples is used by feature-level attack type for comparison . The parameters of PGD are epsilon=0.2, clip_max=0.2 and order of maximum distortion=2.

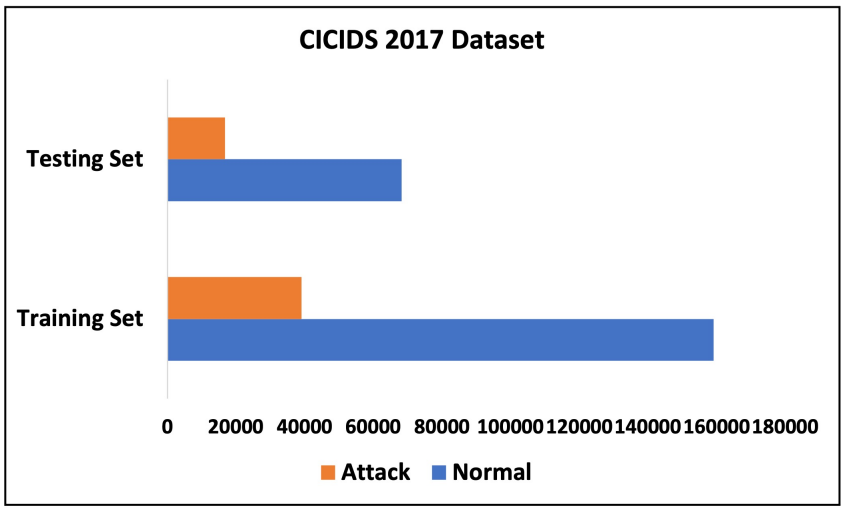

Figure 23: The train/test ratios of the CICIDS 2017 datasets

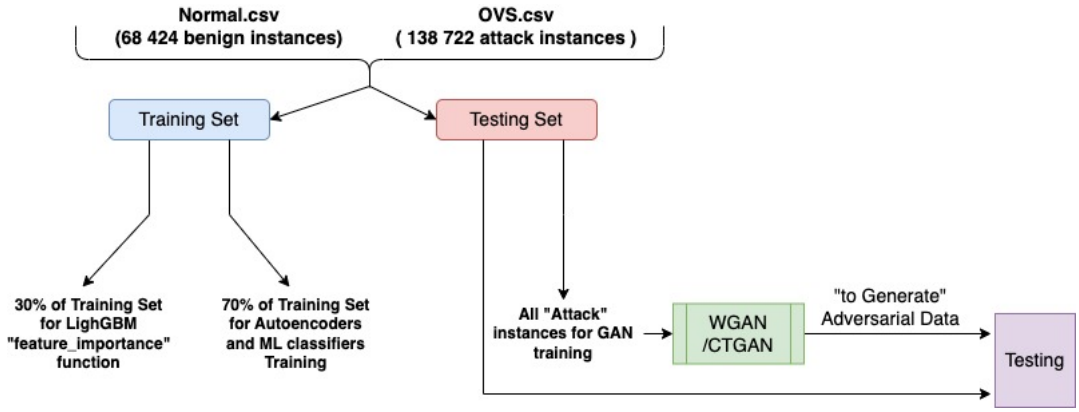

Figure 24: The Usage of the InSDN Dataset

6.4 The SDN Test Environment

To test the proposed RAIDS model, a virtual network simulation environment is created. GNS3 is a virtualization platform for network labs. It supports SDN-based network devices and uses Network Address Translation so you can also add a virtual PC which is running on VMWare or VirtualBox.

Faucet [89] is an open-source controller for the SDN platform. It supports vendor-independent network functionality, OpenFlow v1.3, transport layer security (TLS), etc. You can configure it via the YAML configuration file. For the switching function, OpenvSwitch GNS3 appliance is used. It supports OpenFlow protocol and is compatible with the Faucet controller.

To mimic the attacker and the victim, 2 virtual PCs are connected to the SDN platform. The RAIDS is located between the attacker and the victim. All traffic is sent to the RAIDS via the OpenvSwitch port mirroring feature.

Table 19: The Parameters of WGAN

	WGAN
Discriminator Structure	68->100->30->1
Generator Structure	30->150->100->68
Learning Rate	0.0001
Activation Function	ReLu
Optimizer	RMSprop

Table 20: The environment components

Components	**Definition**
Network Virtualization	GNS3 v.2.2
Faucet SDN Controller	Faucet v1.10.8
OpenVSwitch	OpenvSwitch GNS3 Appliance v3.0.4
Attacker	Kali 2022.3
Victim	Ubuntu Server 22.04
The RAIDS	Running on Ubuntu 18.04/Pytorch v 1.11.0
CICFlowMeter	cicflowmeter 0.1.6 (Python)
Scapy	scapy v2.5.0

All network traffic propagates to the IDS and CICFlowMeter converts network data to the features. The CICFlowMeter is a network traffic flow generator that extracts features from the network data. It creates 83 features from the network flow data. Using these features, the RAIDS runs on Jupyter Notebook, and it analyzes the network traffic. The SDN environment is presented in Fig. 25 and the parameters of components are listed in Table 20.

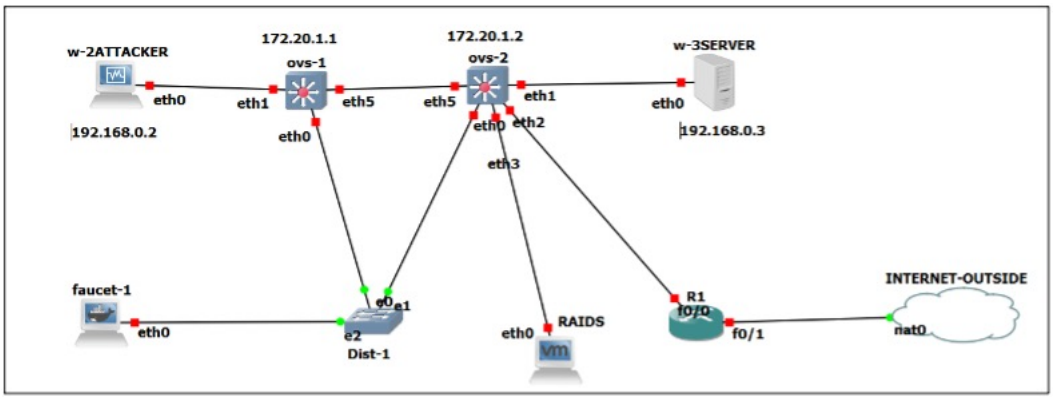

Figure 25: The test environment

To manipulate/alter the network packet, “scapy” is used. It is a Python-based packet manipulation library. Using scapy, network traffic features such as IP packet lengths, flags, and packet fragmentation can be manipulated. It can conduct scanning, probing, unit tests and attacks. Furthermore, it performs specific attacks such as VLAN Hopping, ARP cache poisoning, etc.

From the point of the network attack, one example may be a DoS attack. SYN Flooding attack is a type of DoS attack when an attacker sends an SYN request to the server to consume server resources, so it makes the server unresponsive. Also, specific features of IP packets such as ID and TTL value can be used to conceal the attack. Overlapping fragments occurs when the attacker exploits the datagram fragmentation mechanism. Basically, IP datagrams are broken down into smaller chunks during transmission. MTU dictates the size of each datagram. Each datagram has an offset so this has to be used in a destination. If an attacker alters/creates a random offset number, this can create a headache in the destination.

Furthermore, the attacker wants to learn which port is open at the destination side. To gather information about the victim, the attacker conducts a network discovery attack (probe). The attacker can send a SYN packet, a bad checksum packet to the destination to initiate ICMP. Besides, the attacker wants to control and establish a TCP session to eavesdropping the user data. This can be done with a hijacking attack.

CHAPTER 7

EXPERIMENTAL RESULTS

This section provides results and discussions on the overall performance of the RAIDS model. Firstly, the overall accuracy of the baseline approach was evaluated, then the same test settings were applied to the RAIDS model. All results were evaluated using performance metrics such as overall accuracy and F1-score. In the evaluation of our RAIDS model, the model was trained without using any adversarial examples, and then in a separate experiment, the model was trained it with adversarial examples to see the effect of adversarial training.

7.1 Results of baseline machine learning algorithms

Neural networks, k-NN, SVM and LightGBM are selected as our baseline machine learning algorithms. Before testing, all classifiers were validated and grid search hyperparameter tuning was applied. The tested parameters are presented in Table 21 for each ML classifiers. The selected parameters are stated with '*'. The other parameters of the ML classifiers are default. To build the machine learning models, Python's Scikit-learn library was selected and tests were run on Jupyter Notebook. To check the WGAN learning process, the loss value of the function is tracked. After 3000 epochs, the generator and discriminator are stable.(See 26, 27, 28 and 29)

Furthermore, Kolmogorov-Smirnov [90] test is conducted for measure significance of feature's differences across dataset. This uses the two-sample Kolmogorov–Smirnov test to compare the distributions of continuous columns using the empirical Cumulative distribution function (CDF). D statistic, which indicates the maximum distance between the expected CDF and the observed CDF values. The test results are presented in Table 22. According to the two-sample K-S test, the generated data is not different from original network data (we do not have enough data to reject null hypothesize) by p-value of 0.05. Besides, data generated by CTGAN is more different from real data in both datasets according to the D statistic. It can be concluded that CTGAN based generated adversarial examples may become hard to detect.

Firstly, the classifiers were tested with real testing data without any adversarial data, then classifiers were tested with only adversarial examples. Before testing with any adversarial data, neural network, k-NN and LightGBM attained at least 98% overall accuracy and F1-score on the InSDN dataset. All classifiers were capable of performing successful normal/attack classification. When the classifiers were tested with WGAN based adversarial examples, neural network and LightGBM suffered heavily against adversarial examples. Also, k-NN's accuracy decreased to 53% from 98%. Furthermore, the classifiers also performed badly on CTGAN-based adversarial examples. For the CICIDS dataset, the

Table 21: The parameters of ML models for grid search hyperparameter tuning

ML Models	Parameters
Neural Network (sklearn)	activation = logistic, tanh, relu* solver = lbfgs, sgd*, adam learning rate = 0.001*, 0.003, 0.01 (for sgd and adam) momentum = 0.5, 0.9* (for sgd) output layer = cross_entropy
Support Vector Machine (sklearn)	kernel = linear, poly*, rbf, sigmoid gamma = scaled, auto* c = 0.5, 1.0*
k-NN (sklearn)	n = 3, 5*, 11 weights = uniform, distance* algorithm = auto*, ball_tree, kd_tree metric = minkowski*, manhattan
LightGBM (lightgbm)	boosting = gdpt*, rf, dart learning_rate = 0.01*, 0.05, 0.1 min_data_in_leaf = 20*, 50, 100 drop_rate = 0.1, 0.2*, 0.5 data_sample_strategy = bagging, goss*

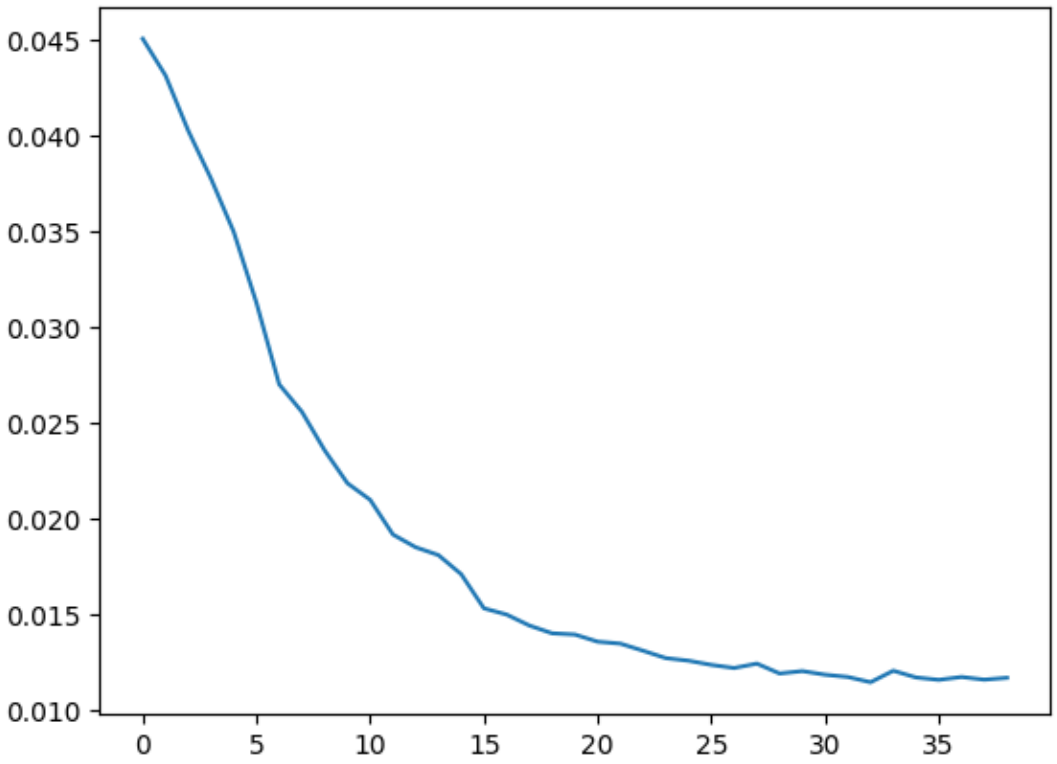

Figure 26: Generator's Loss in InSDN Dataset During Training(x 100 Epochs)

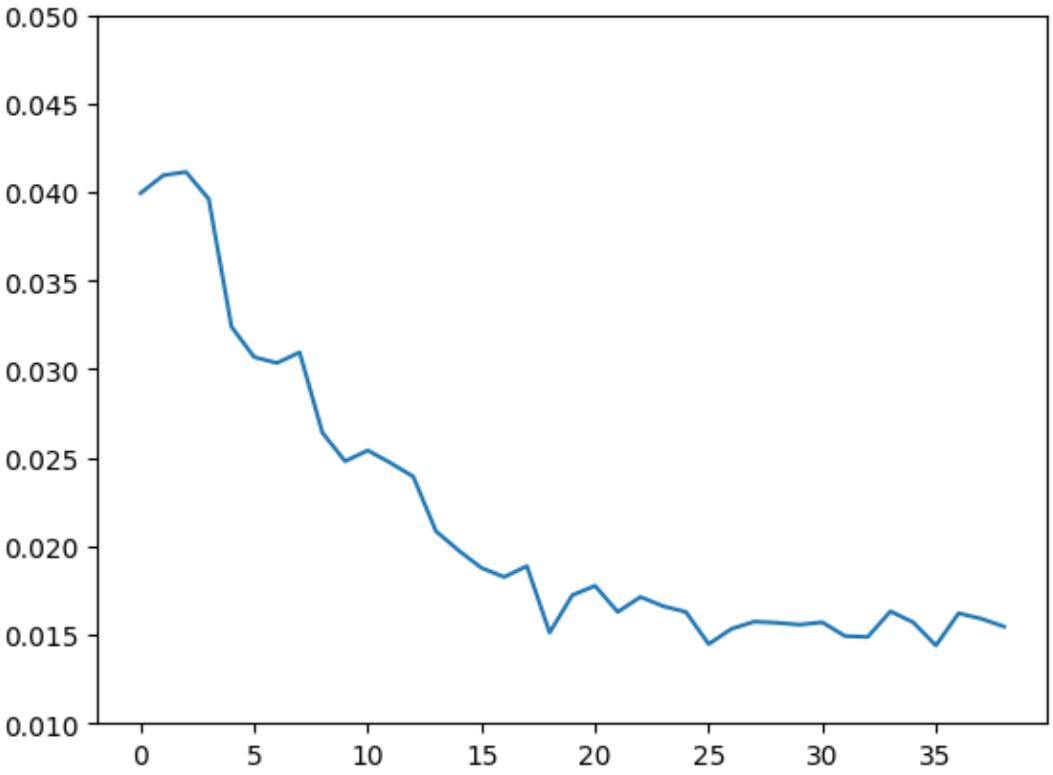

Figure 27: Critic's Loss in InSDN Dataset During Training (x 100 Epochs)

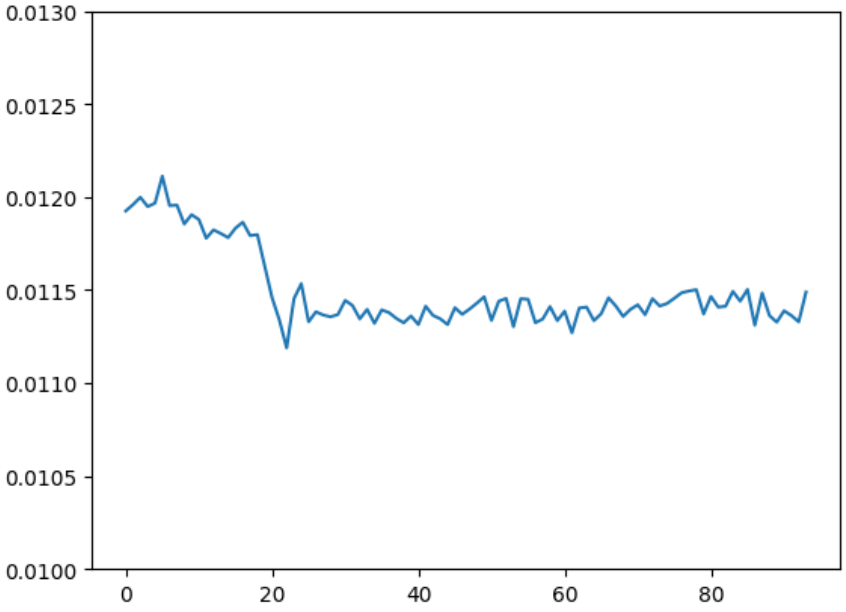

Figure 28: Generator's Loss in CICIDS 2017 Dataset During Training (x 100 Epochs)

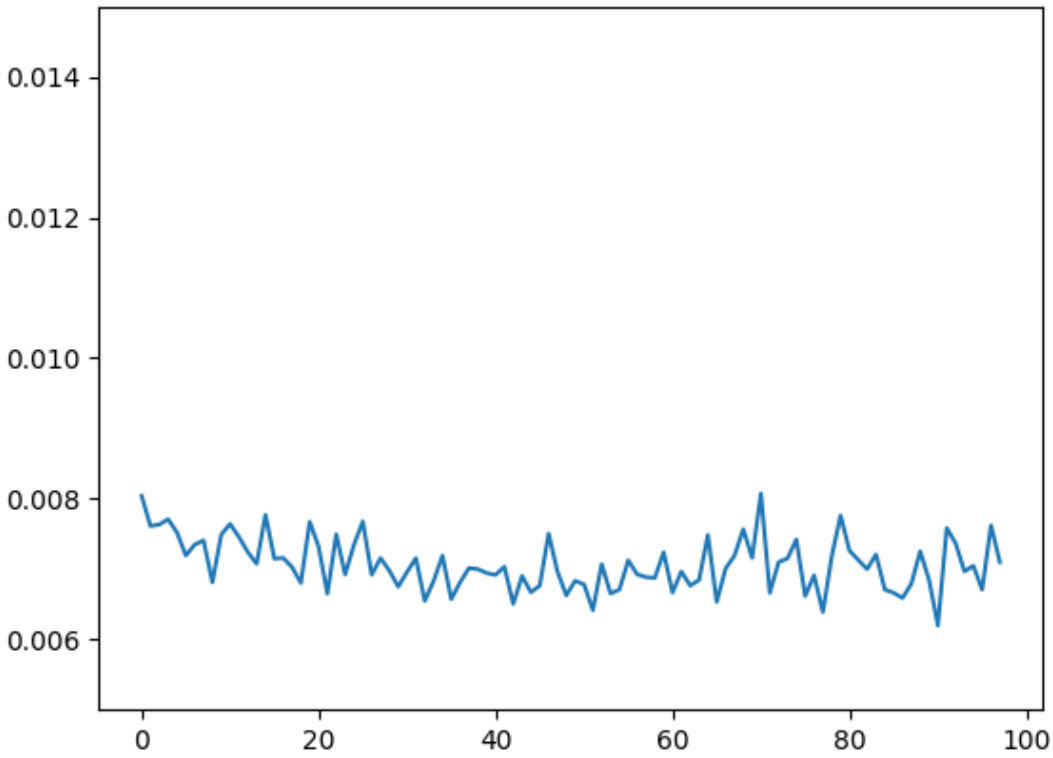

Figure 29: Critic's Loss in CICIDS 2017 Dataset During Training (x 100 Epochs)

Table 22: Kolmogorov-Smirnov Test Result

Data	K-S D Statistic	P-value
Real InSDN Attack Data vs. Generated Data by WGAN	0.143289	0.262
Real InSDN Attack Data vs. Generated Data by CTGAN	0.442373	0.478
Real IoT-SDN Attack Data vs. Generated Data by WGAN	0.087309	0.215
Real IoT-SDN Attack Data vs. Generated Data by CTGAN	0.696785	0.389

Table 23: Baseline machine learning classifier results for the InSDN dataset

	Accuracy				F1-Score			
	TC I	TC II	TC III	TC IV	TC I	TC II	TC III	TC IV
NN	0.98	0.11	0.68	0.12	0.98	0.08	0.41	0.09
k-NN	0.98	0.53	0.47	0.22	0.98	0.35	0.32	0.18
LightGBM	0.99	0	0.41	0.13	0.99	0	0.29	0.10

Test Case I: Only Real test dataset
Test Case II: Only WGAN Generated Attack Data (13,800 instances)
Test Case III: Only CTGAN Generated Attack Data (13,800 instances)
Test Case IV: Only PGD Generated Attack Data (13,800 instances)

Table 24: Baseline machine learning classifier results for the CICIDS 2017 dataset

	Overall Accuracy				F1-Score			
	TC I	TC II	TC III	TC IV	TC I	TC II	TC III	TC IV
NN	0.98	0	0.12	0.22	0.98	0	0.11	0.13
k-NN	0.98	0.23	0.54	0.19	0.98	0.19	0.35	0.10
LightGBM	0.99	0	0.45	0.08	1.0	0	0.31	0.08

Test Case I: Only Real test dataset
Test Case II: Only WGAN Generated Attack Data (25,000 instances)
Test Case III: Only CTGAN Generated Attack Data (25,000 instances)
Test Case IV: Only PGD Generated Attack Data (25,000 instances)

classifiers were vulnerable to adversarial examples. Their overall accuracy decreased to 0% at the lowest and 54% at the highest. PGD attacks succeeded in decreasing the accuracy of ML classifiers. Interestingly, LightGBM also suffered from the presence of adversarial examples although it is a state-of-the-art classifier which typically performs well in intrusion detection. The results are presented in Table 23 and 24. Thus, we can conclude that GAN based adversarial examples successfully tricked machine learning classifiers.

7.2 RAIDS model results

To test the RAIDS model, two different settings are defined. The first setting is the RAIDS model without adversarial training, and the second setting is the RAIDS model with adversarial training. Also, to measure the success of the autoencoder, using only the autoencoder for RAIDS is tested. The use of the abbreviation "RAIDS" means RAIDS v2.0.

Table 25: The RAIDS v2.0 vs. RAIDS v1.0 model using the InSDN dataset

	The RAIDS v2.0 Model		The RAIDS v1.0 Model	
	Overall Accuracy	F1-Score	Overall Accuracy	F1-Score
Only Real Data	0.98	0.98	0.89	0.86
WGAN-based Adversarial Examples	0.60	0.74	0.61	0.63
CTGAN-based Adversarial Examples	0.82	0.90	0.65	0.62
PGD Attack Adversarial Examples	0.39	0.40	0.35	0.31

Table 26: The RAIDS v2.0 vs. RAIDS v1.0 model using the CICIDS 2017 dataset

	The RAIDS v2.0 Model		The RAIDS v1.0 Model	
	Overall Accuracy	F1-Score	Overall Accuracy	F1-Score
Only Real Data	0.99	1.00	0.90	0.91
WGAN-based Adversarial Examples	0.99	1.00	0.82	0.77
CTGAN-based Adversarial Examples	0.69	0.81	0.59	0.60
PGD Attack Adversarial Examples	0.49	0.48	0.29	0.26

7.2.1 RAIDS v1.0 Model Without Adversarial Training

The RAIDS v1.0 model was tested with only real testing data. Then, the model was tested against WGAN, CTGAN based adversarial examples and PGD attack with the same instance size. When compared to the RAIDS v2.0 model, the RAIDS v2.0 provided better overall accuracy and F1-score on InSDN for both GAN based adversarial examples (See Table 25). When the RAIDS v2.0 was tested on the CICIDS 2017 dataset, the results showed a similar pattern: RAIDS v2.0 provided better overall accuracy and F1-score for both GAN-based adversarial examples and PGD attack (See Table 26).

7.2.2 RAIDS v2.0 Model Without Adversarial Training

Firstly, the RAIDS v2.0 model was tested with only real testing data. Then, the model was tested against WGAN, CTGAN based adversarial examples and PGD attack with the same instance size. RAIDS v2.0's autoencoders validation results are presented Fig. 30 and 31. The feature importance plots are presented in Fig. 32 and 33. When compared to the baseline ML classifiers, RAIDS v2.0 provided better overall accuracy and F1-score on InSDN for both GAN based adversarial examples. Besides, its real data performance was as good as the best classifiers, as shown in Table 27, and better than the other classifiers. When the model was tested on the CICIDS 2017 dataset, the results showed a similar pattern: RAIDS v2.0 provided better overall accuracy and F1-score for both GAN-based

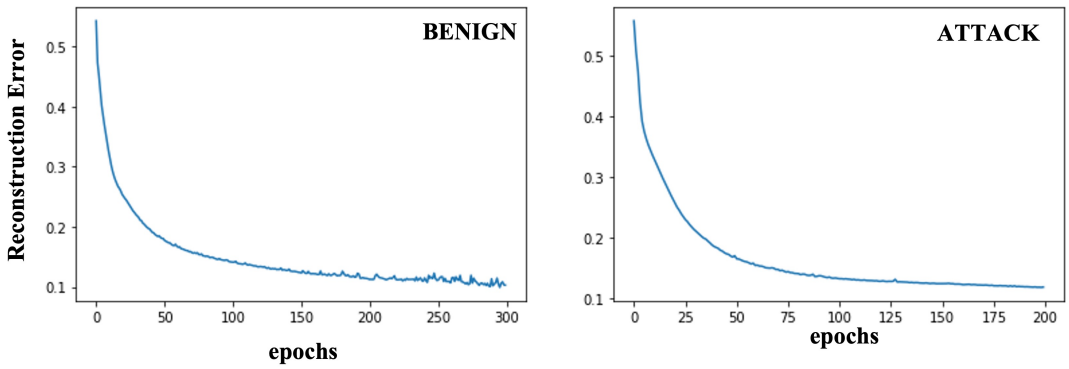

Figure 30: The Validation Result of Autoencoders-InSDN

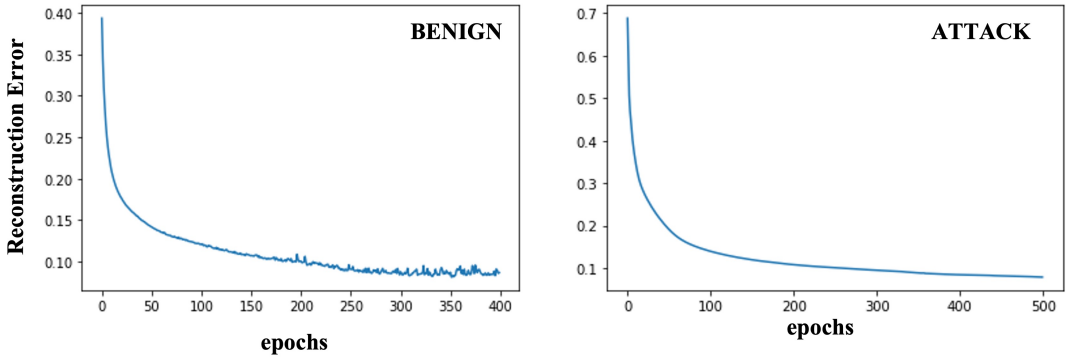

Figure 31: The Validation Result of Autoencoders-CICIDS 2017

adversarial examples and PGD attack (See Table 28), with a more pronounced improvement over the best ML classifier.

7.2.3 RAIDS v2.0 Model Only Using Two Autoencoders Without Adversarial Training

To measure the success of the autoencoders, ensemble ML classifier part of the RAIDS v2.0 is removed. The structure of the this model is presented in Fig. 34. Firstly, the model was tested with only real testing data. Then, the model was tested against WGAN, CTGAN based adversarial examples and PGD attack with the same instance size. When compared to the original RAIDS v2.0, RAIDS v2.0 provided slightly better overall accuracy and F1-score on InSDN for both GAN based adversarial examples. When the model was tested on the CICIDS 2017 dataset, the results was similar. (See Table 29 and 30)

7.2.4 RAIDS v2.0 Model With Adversarial Training

In order to check whether adversarial training improves the accuracy and F1-score of the model, the RAIDS v2.0 model is also tested with adversarial data. The model was trained with one GAN's

Table 27: RAIDS v2.0 vs. Baseline ML classifiers using the InSDN dataset

	The RAIDS v2.0 Model		Baseline ML Classifier		
	Overall Accuracy	F1-Score	Best ML Classifier	Overall Accuarcy	F1-Score
Only Real Data	0.98	0.98	LightGBM	0.99	0.99
WGAN-based Adversarial Examples	0.60	0.74	k-NN	0.53	0.35
CTGAN-based Adversarial Examples	0.82	0.90	k-NN	0.68	0.41
PGD Attack Adversarial Examples	0.39	0.40	k-NN	0.22	0.18

Table 28: RAIDS v2.0 vs. Baseline ML classifiers using the CICIDS 2017 dataset

	The RAIDS v2.0 Model		Baseline ML Classifier		
	Overall Accuracy	F1-Score	Best ML Classifier	Overall Accuracy	F1-Score
Only Real Data	0.99	1.00	LightGBM	0.99	1.00
WGAN-based Adversarial Examples	0.99	1.00	k-NN	0.23	0.19
CTGAN-based Adversarial Examples	0.69	0.81	k-NN	0.54	0.35
PGD Attack Adversarial Examples	0.49	0.48	NN	0.22	0.13

Table 29: The RAIDS v2.0 vs. RAIDS v2.0 with only Autoencoders using the InSDN dataset

	The RAIDS v2.0 Model		The RAIDS v2.0 with only AE	
	Overall Accuracy	F1-Score	Overall Accuracy	F1-Score
Only Real Data	0.98	0.98	0.97	0.97
WGAN-based Adversarial Examples	0.60	0.74	0.58	0.60
CTGAN-based Adversarial Examples	0.82	0.90	0.83	0.90
PGD Attack Adversarial Examples	0.39	0.40	0.39	0.38

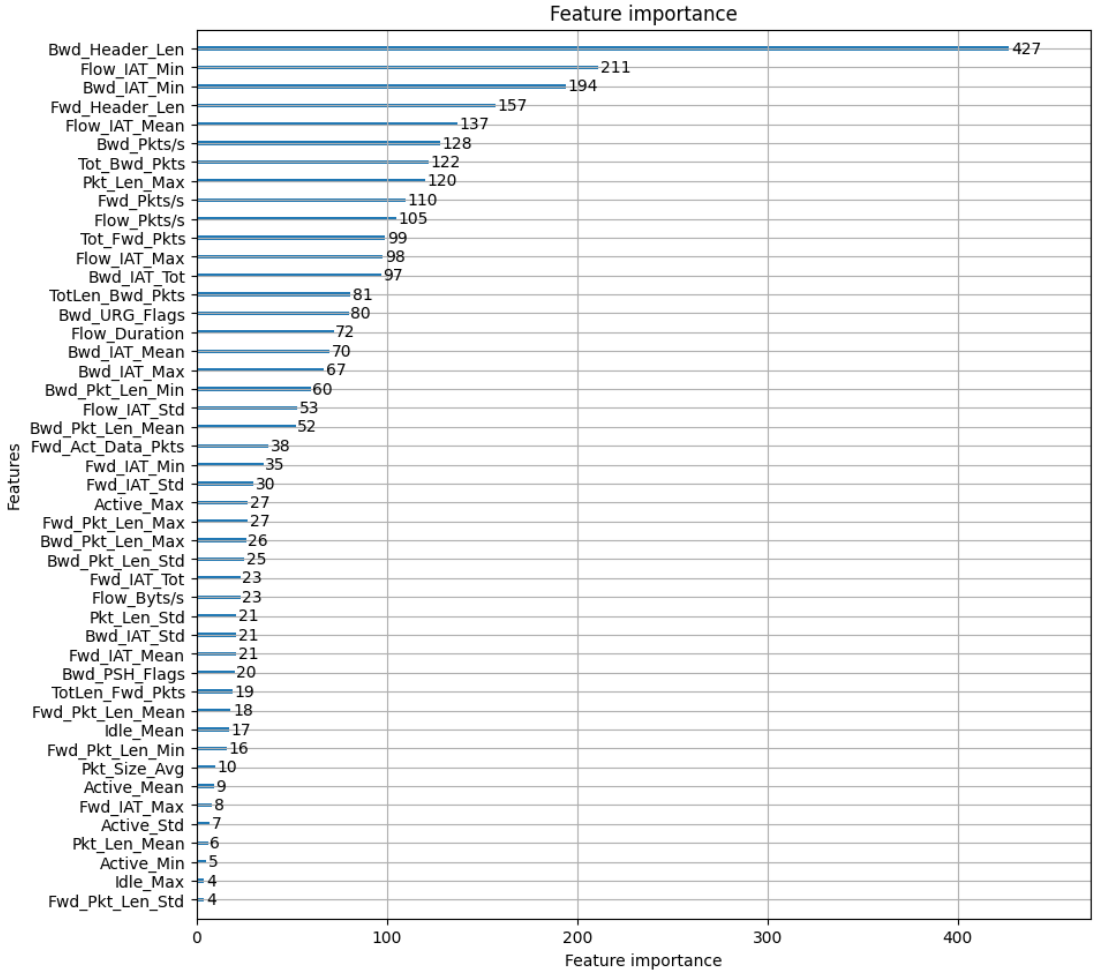

Figure 32: The feature importance plot of InSDN

adversarial examples (e.g. WGAN-based adversarial attack data) and then tested with the other GAN's adversarial attack examples (e.g. CTGAN-based adversarial attack data). When we compared the results for the InSDN dataset, CTGAN-based adversarial training increased the overall accuracy and F1 score more than WGAN-based training. On the other hand, adversarial training did not make any significant impact for the RAIDS v2.0 model on the CICIDS dataset, as can be seen in Tables 31 and 32.

7.2.5 Cross-Dataset Experiments

To evaluate the RAIDS v2.0 model's effectiveness with the help of reconstruction error based adversarial attack detection, the RAIDS v2.0 model is trained with one dataset (e.g. InSDN) and tested with adversarial attacks which are produced by the other dataset (e.g. CICIDS 2017). Both datasets have some similar feature sets. Using these features, the performance of the RAIDS v2.0 model are measured. The same instance size is generated from WGAN and CTGAN and used for testing. During this test, the adversarial training is not implemented.

Table 30: The RAIDS v2.0 vs. RAIDS v2.0 with only Autoencoders using the CICIDS 2017 dataset

	The RAIDS v2.0 Model		The RAIDS v2.0 with only AE	
	Overall Accuracy	F1-Score	Overall Accuracy	F1-Score
Only Real Data	0.99	1.00	0.99	0.99
WGAN-based Adversarial Examples	0.99	1.00	0.98	0.98
CTGAN-based Adversarial Examples	0.69	0.81	0.70	0.80
PGD Attack Adversarial Examples	0.49	0.48	0.48	0.48

Table 31: RAIDS v2.0 results using the InSDN dataset

		RAIDS v2.0 Model with Adversarial Training		**RAIDS v2.0 Model without Adversarial Training**	
Adversarial Training Data	Testing Data	Overall Accuracy	F1-Score	Overall Accuracy	F1-Score
WGAN-base Adversarial Samples	Only Real Test Data	0.98	0.98	0.98	0.98
	CTGAN-based Adv. Examples	0.60	0.75	0.59	0.74
CTGAN-base Adversarial Samples	Only Real Test Data	0.97	0.97	0.98	0.98
	WGAN-based Adv. Examples	1.00	1.00	0.82	0.9

Table 32: RAIDS v2.0 results using the CICIDS 2017 dataset

		RAIDS v2.0 Model with Adversarial Training		**RAIDS v2.0 Model without Adversarial Training**	
Adversarial Training Data	Testing Data	Overall Accuracy	F1-Score	Overall Accuracy	F1-Score
WGAN-base Adversarial Samples	Only Real Test Data	0.99	1.00	0.99	1.00
	CTGAN-based Adv. Examples	0.68	0.81	0.69	0.81
CTGAN-base Adversarial Samples	Only Real Test Data	0.99	1.00	0.99	1.00
	WGAN-based Adv. Examples	1.00	1.00	0.99	1.00

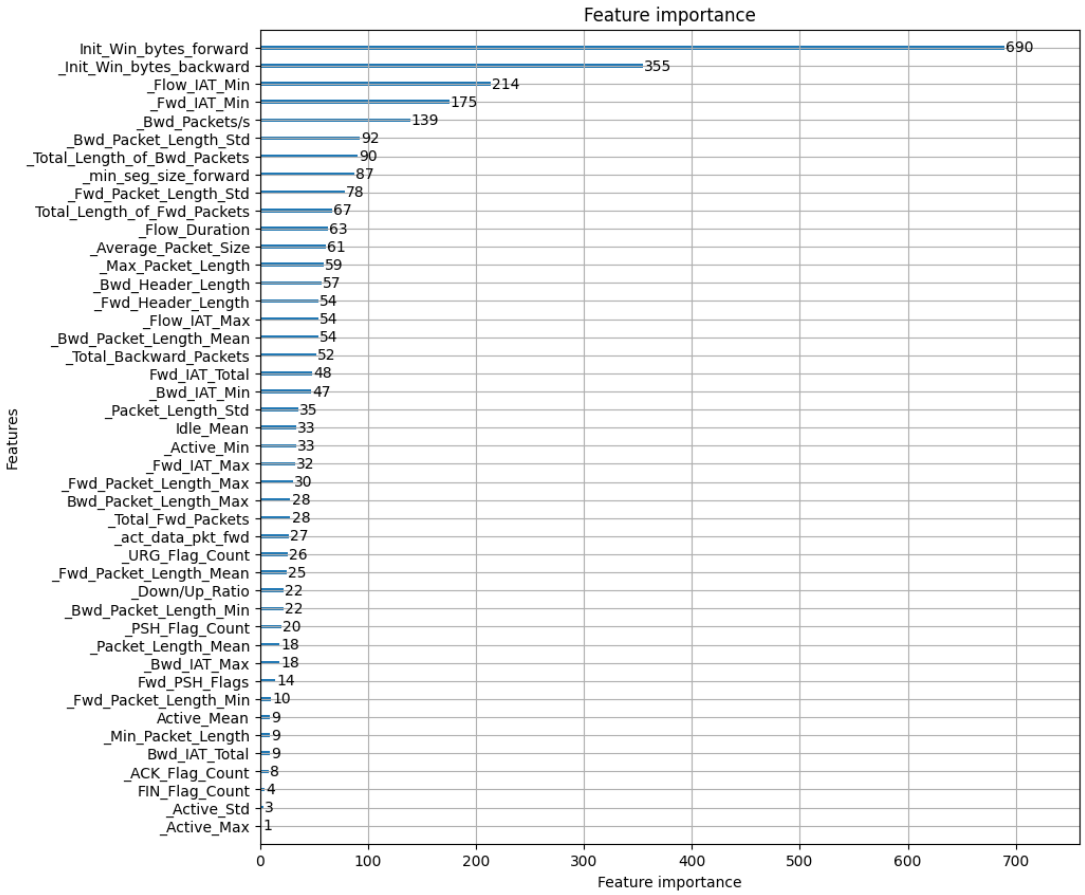

Figure 33: The feature importance plot of CICIDS 2017

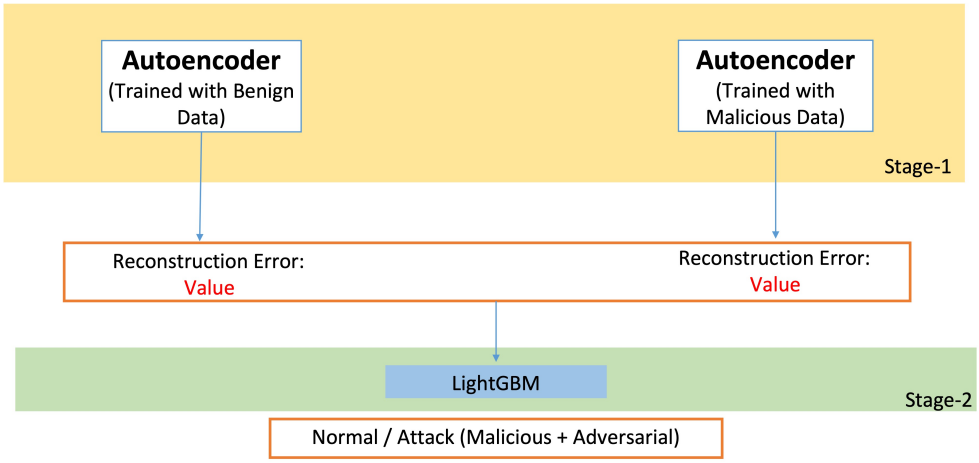

Figure 34: The Structure of the RAIDS v2.0 only Autoencoders model

Table 33 presents the performance metrics of the RAIDS v2.0 model trained with InSDN and tested against adversarial examples generated from the CICIDS 2017 dataset. Table 34 presents the perfor-

Table 33: The result of RAIDS v2.0 model trained with InSDN dataset

RAIDS v2.0 model		**k-NN**		**LightGBM**	
Acc.	*F1-Score*	*Acc.*	*F1-Score*	*Acc.*	*F1-Score*
0.68	0.56	0.44	0.40	0.32	0.28

Table 34: The result of RAIDS v2.0 model trained with CICIDS 2017 dataset

RAIDS v2.0 model		**k-NN**		**LightGBM**	
Acc.	*F1-Score*	*Acc.*	*F1-Score*	*Acc.*	*F1-Score*
0.78	0.75	0.51	0.48	0.28	0.27

mance metrics of the RAIDS v2.0 model trained with CICIDS 2017 and tested against adversarial examples generated from the InSDN dataset. Without adversarial training, RAIDS v2.0 model produces better overall accuracy and F1-score than the baseline machine learning model using different dataset characteristics.

7.3 Testing the Feasibility of the Adversarial Attack

To test the feasibility of an adversarial attack via feature perturbation and the performance of RAIDS v2.0 in a realistic scenario, Cross-Site Scripting (XSS) attack was chosen. An attacker can use XSS attacks to inject malicious code into a web application. Both datasets contain XSS attacks in their PCAP captures.

By using scapy, the original packets are manipulated without modifying the payload(See Fig. 35). By doing so, features such as flow duration, flags, and forwarded packet per seconds(Related with transport and network layer) were altered. CICFlowMeter tool extracted the features and same features sets tested in the research were used. The manipulation settings of the network traffic features and evaluation results of the RAIDS v2.0 model are presented in Table 35.

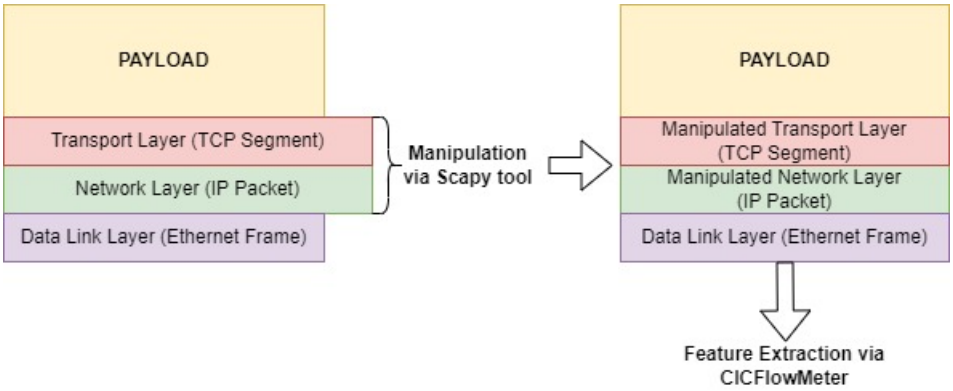

Figure 35: The flowchart of XSS packet generation

Table 35: The performance of RAIDS v2.0 model (trained with InSDN) in an adversarial attack conducted via traffic feature perturbation

Packet Manipulation Settings	**RAIDS v2.0 Detection Rate**
Unaltered XSS Packet	96.81%
Flow Duration [0.02 to 0.04] URG Flags = 0 PSH Flags = 1 Fwd Packet Length Mean = 80 Fwd Pkts/s = 1	66.23%
Flow Duration [0.04 to 0.06] URG Flags = 0 PSH Flags = 1 Fwd Packet Length Mean = 80 Fwd Pkts/s = 2	71.78%
Flow Duration [0.04 to 0.06] URG Flags = 1 PSH Flags = 1 Fwd Packet Length Mean = 100 Fwd Pkts/s = 6	88.45%
Flow Duration [0.1 to 0.11] URG Flags = 0 PSH Flags = 1 Fwd Packet Length Mean = 200 Fwd Pkts/s = 10	94.56%

SYN, FIN and ACK flags are specific flags used for the three-way handshake for establishing a TCP connection and the two-way handshake for finishing the connection. For this reason, other flags such as URG(indicates that the packet should be processed immediately by the TCP stack) and PSH (indicates that incoming data should be passed on directly to the application layer) are more viable options for manipulation. If a ML based IDS checks these flags to detect attack, it will be meaningful to manipulate these features. Also while using scapy, we can optimize the flow duration of a connection, and fragment a packet to adjust the forwarded packet per second and the length of the packet in each flow. According to the results presented in Table 35 and 36, flow duration and fwd packet length are the best options for the attacker to evade the IDS model during an XSS attack.

7.4 Comparison with Other Models

Table 37 presents a comparison of the RAIDS v2.0 model vs. other state-of-the-art AML models and their performance. When we compared the RAIDS v2.0 model against them, the RAIDS v2.0 has achieved fairly considerable accuracy improvements. Also, some of the previous research used obsolete datasets to measure their defense mechanism's success. Our proposed approach for the detection of adversarial examples is more successful than the adversarial training approach in terms of both accuracy and F1-score improvements.

Table 36: The performance of RAIDS v2.0 model (trained with InSDN) in an adversarial attack conducted via traffic feature perturbation

Packet Manipulation Settings	**RAIDS v2.0 Detection Rate**
Unaltered Link Flooding Attack	82.11%
Flow Duration [0.01 - 0.015] Fwd Packet Length Mean = 60 Fwd Pckts/s = 10	79.14%
Flow Duration [0.015 - 0.020] Fwd Packet Length Mean = 200 Fwd Pckts/s = 100	67.15%
Flow Duration [0.005 - 0.01] Fwd Packet Length Mean = 60 Fwd Pckts/s = 1000	56.14%

Table 37: RAIDS v2.0 model vs. Other state-of-the-art models

Model	Defense Type	**Attack Type**	Dataset	**Metrics (M) & Improvement (I)**
RAIDS v2.0	Reconstruction Error	GAN, PGD	InSDN	82%Accuracy (M), 90%f1-Score (M) Average 37% (Accuracy) (I) Average 107% (F1-Score) (I)
RAIDS v2.0	Reconstruction Error	GAN, PGD	CICIDS 2017	99%Accuracy (M), 1.00%f1-Score (M) Average 59% (Accuracy) (I) Average 129% (F1-Score) (I)
Reconstruction from Partial Observation (RePO) [91]	Adversarial Training	Hashemi	CICIDS 2017	49% (Detection) (I)
Deep Neural Network [92]	Adversarial Training	CW	CICIDS 2017	15.90% (Accuracy) (I) 22.32 % (F1-Score) (I)
AIDTF [93]	Adversarial Training	FGM, FSGM, JSMA, PGD	NSL-KDD	From 11.90% to 100% (Accuracy using Logistic Regression) (I)
TAD [94]	Transfer learning-based adversarial detector	FSGM, DeepFool, PGD	NSL-KDD CICIDS 2017	74.50 % Adversarial training(Accuracy)-(M) 83.67 % TAD(Accuracy-(M)

Table 38: Comparison of the computation times of models

Model	**Training time (s)**
RAIDS v2.0 (CICIDS 2017)	138
RAIDS v2.0 (InSDN)	217
Al-Qatf et al.[95]	673
Wang et al.[96]	1075
Naveed et al [97]	138

Table 38 shows the training time of the RAIDS v2.0 model compared with other deep learning models. We can see that RAIDS v2.0 does not need longer training time compared to alternative models from the literature. In fact, there are mitigating factors reducing the complexity and running time of RAIDS v2.0: In stage-1, both autoencoders and ML classifiers are trained simultaneously (in parallel). Also, in stage-2, only LightGBM is trained.

7.5 Discussion

The evaluation showed that baseline machine learning classifiers used in intrusion detection are susceptible to adversarial examples. The attackers usually do not know IDS model parameters and structure. For this reason, they try to manipulate network data characteristics such as statistical features. The approach to perturbing network data is different from adversarial attacks in the image domain. The reason is that image data use the same scales and all data are continuous. On the other hand, network data consists of different types of features and these have different scales.

Feature-level attacks are a common way to generate adversarial examples in image detection, but they are unpractical for the network data. Data produced through FLAs are constrained within a norm. However, network data features may need to differ from this constrained norm. For instance, when a network traffic snapshot has a high number of bytes, number of packets sent should also be high. For this reason, feature-level attacks are not practical for network data.

To generate more realistic and applicable adversarial examples, two different GANs, namely CTGAN and WGAN were used. I observed that the generator part of these can create adversarial examples effectively. The overall accuracy of state-of-the-art ML classifiers such as LightGBM decreases to as low as 0% from 100% when faced with GAN-based adversarial examples. We can conclude that GAN-based adversarial examples are one of the easiest paths for generating effective adversarial examples. Also, PGD attack decreases the accuracy and F1-Score of classifiers. Furthermore, K-NN proved to be the more robust ML classifier against adversarial attacks in our test environment.

The histogram plot of some features from the generated adversarial examples are presented in Figure 36. The generated adversarial examples have similar statistical properties compared to real attack data but they can easily fool an IDS. An attacker can easily use network data crafting tools such as scapy to conduct an adversarial attack with the help of generative feature-level attacks, especially using statistical features.

According to our experiments, autoencoder reconstruction error is the best advantage against adversarial examples. Because normal data and attack data have special latent features and when an au-

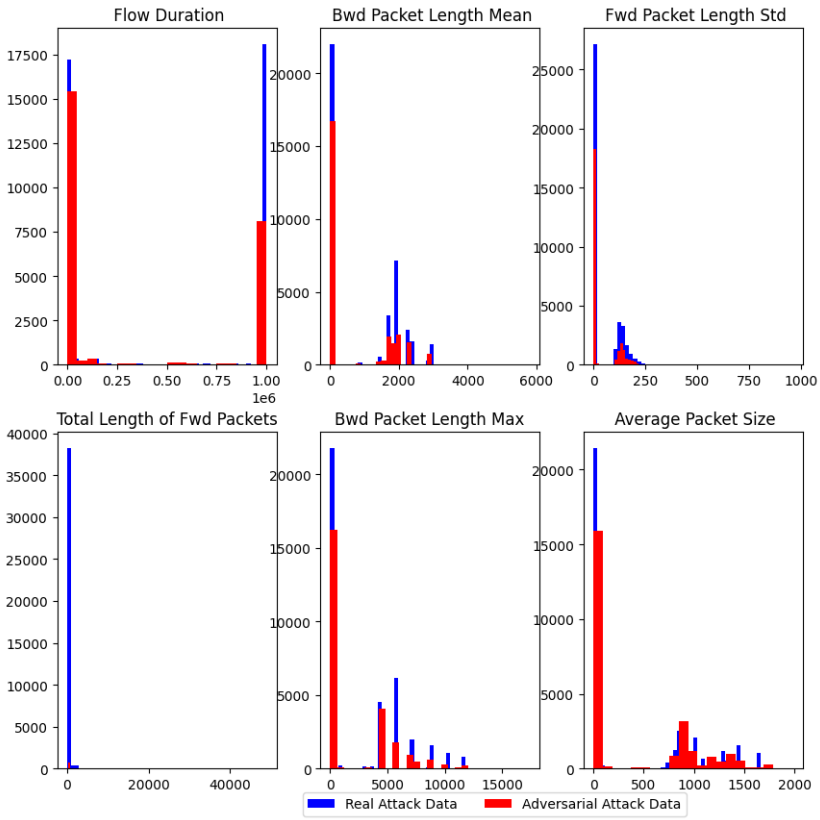

Figure 36: The histogram plot of real attack data vs. CTGAN generated adversarial attack using CICIDS 2017

toencoder learns these latent features, they can easily detect different instances. According to the confusion matrices in Tables 39 and 40, while working on the adversarial test case, the RAIDS v2.0 model keeps the precision metric the same as the "Only Real Data" case. In the worst-case scenario, RAIDS v2.0 provided 13.2% better overall accuracy compared to the best ML classifier's overall accuracy. RAIDS v2.0 achieved a bigger improvement in F1-scores over baseline ML classifiers. Another important point is that when RAIDS v2.0 was tested on the InSDN dataset, performance improvement was lower than that on the CICIDS dataset. The reason for this may be that the InSDN dataset features cannot provide better latent features for autoencoders. While the overall accuracy of RAIDS v2.0 against CTGAN-based adversarial examples was better than WGAN-based adversarial examples using InSDN, the situation is the opposite on CICIDS 2017. When the test is conducted for feasibility of the adversarial attack, the statistical features of the network flow are easily manipulated by the attacker to conduct an adversarial attack. Finally, contrary to findings in the literature pointing to adversarial training as the best countermeasure against adversarial examples [98], our experiments showed that adversarial training did not matter, possibly due to dataset characteristics.

Table 39: RAIDS v2.0 model confusion matrix for InSDN

		Predicted	
		Benign	Attack
Actual	Benign	13312	488
	Attack	92	13708

(a)

	Predicted	
	Benign	Attack
Benign	13312	488
Attack	2690	24910

(b)

(a) Real Testing Data (Overall Accuracy = 97.90%)
(b) Real Testing Data with WGAN based Adversarial Samples- 13,800 instances (Overall Accuracy = 92.32%)

Table 40: RAIDS v2.0 model confusion matrix for CICIDS 2017

		Predicted	
		Benign	Attack
Actual	Benign	67940	253
	Attack	288	16442

(a)

	Predicted	
	Benign	Attack
Benign	67940	253
Attack	8094	33636

(b)

(a) Real Testing Data (Overall Accuracy = 99.36%)
(b) Real Testing Data with CTGAN based Adversarial Samples- 25,000 instances (Overall Accuracy = 92.40%)

CHAPTER 8

CONCLUSION

8.1 Conclusion

In this thesis, a novel robust IDS model called RAIDS, which uses the reconstruction error of two autoencoders, along with the classification results of an ensemble of ML classifiers, to train a LightGBM classifier tasked with categorizing traffic data as attack or benign is introduced.

The evolving surface of cyber attacks necessitates a comprehensive approach to network security. The intrusion detection and prevention systems (IDS/IPS) play an important role in identifying possible threats, trying to prevent them, and alerting them to security administrators. For this reason, IDS is an important part of network security and it can be used heavily in the network security domain.

To detect zero-day attacks and prevent any malicious attempt targeting the network, anomaly-based IDS is an appropriate choice for security administrators. It has the ability to detect novel attacks. Machine learning algorithms, both supervised and unsupervised, are used in developing IDS by researchers. If it is used with other security devices/software such as firewalls in a defense-in-depth manner, the protection of the network can be assured.

Despite achievements made by machine learning algorithms (especially in deep learning), the models are easily tricked by modified input data. Adversarial attacks target ML-based IDS model. In many cases, modified data is nearly similar to the original data, but slight perturbation on these data heavily deteriorates machine learning algorithms' accuracy. Only one feature manipulation by an attacker can lead to some misclassification. For this reason, adversarial attacks target machine learning models severely.

Using gradient-based, optimization-based, and search-based attack methods, any attacker easily create adversarial examples. Besides, generative models, especially generative adversarial networks, can create adversarial examples, too. Whereas the creation of the adversarial examples is easy, testing the IDS model with these examples is not an easy task.

Adversarial training is the preferable approach against adversarial attacks, especially in the image detection task, however, this can not be the suitable approach for the network domain due to the volume, scale, and velocity of the network flow data. Simply, the network data has a different scale and the attack surface is much larger than the image detection domain. The adversarial attacks are generally dependent on the input data types. Because images have the same scale (0 to 255) implementing

feature-level attacks and generative feature-level attack is meaningful. However, the network data is different than this characteristic.

Using an autoencoder's reconstruction error as a defensive approach against adversarial attacks is the novelty of this thesis. If a generative model can understand data distribution and specification of the network data, any deviation from this basis can produce a higher reconstruction error value. This is the key for this research.

Firstly, by using generative adversarial networks, more realistic adversarial examples were created and tested against baseline machine learning classifiers. WGAN and CTGAN were chosen for adversarial attack generation algorithms. These are called generative feature-level adversarial attacks. Projected gradient descent was chosen as a feature-level attack algorithm. Attack results show that machine learning classifiers are indeed susceptible to adversarial attacks. This result is similar to the literature. Besides, CTGAN can generate more examples which are difficult to detect.

Secondly, in order to achieve adversarial robustness, we tried a different approach compared to the literature. Our approach is based on autoencoders: an autoencoder can measure the difference between input and output, and two trained autoencoders can measure normal vs. others, and attack vs. others. Using these measurements, the RAIDS model was developed and tested against adversarial attacks using captured data and a software-defined network environment. Results are promising, even more so thanks to the model not needing adversarial training, which is a hard task due to the attack domain. The cross-dataset experiment also shows that reconstruction error is a more feasible and alternative approach against adversarial attacks. The classification results of an ensemble of ML classifiers also provide valuable feedback for the second layer LightGBM classifier.

Implementing adversarial attacks in a real-time network environment is not an easy task for attackers due to the protocol's header structure and well-defined network protocols. However, it is not an impossible task. Any machine/deep learning researcher should be aware of the adversarial attacks against his/her model. The security of the model should always be considered. To test RAIDS against adversarial attacks, an SDN environment is created and the results show that the autoencoder's reconstruction error provides valuable feedback for detecting adversarial examples. One important point is that projected gradient descent (PGD) based adversarial examples are not feasible due to the compatibility of the generated data with the TCP/IP protocol. PGD is a more powerful adversarial attack type but it is not usable in a real-time network environment.

8.2 Future Work

New network technologies bring new threats to the network environment. Software defined networks are becoming popular, and new applications and protocols are emerging in information technology. Software defined network provides cost-effectiveness, resource management, and easy management of the devices, however, it has specific attack surfaces for attackers. The complexity of the SDN protocol, the configuration schema of the controller, and the volume of the ingress data make network protection difficult. The new IDS proposal and dataset should consider these phenomena.

Adversarial attacks will affect ML classification tasks and target deep learning models excessively. New attack types will appear in the near future. One important point for the network domain is the applicability of adversarial attacks. The projected gradient descent attack is decreasing the IDS model

accuracy heavily but the applicability of this attack in real-time network environment is questionable. Because the network protocol has strict rules and any violation of the rules invalidates the adversarial attack. However, the creation of the adversarial examples requires high computational and time costs. Future investigation will focus on reducing these costs and the transferability of attacks across different datasets and models.

The security of the machine learning model and robustness of the adversarial attacks should be researched in the future. A quantitative framework can measure the security of the ML model and explain the weakness of the machine/deep learning model. Any standardization for the security of the machine learning model can address adversarial attacks.

REFERENCES

[1] S. Mishra and A. K. Tyagi, "The role of machine learning techniques in internet of things-based cloud applications," *Artificial intelligence-based internet of things systems*, pp. 105–135, 2022.

[2] ENISA, "Enisa threat landscape 2022." `https://www.enisa.europa.eu/publications/enisa-threat-landscape2022`, 2023. [Online; accessed 23-July-2023].

[3] L. Xu and K. Tang, "A malware analysis method based on behavioral knowledge graph," 2023.

[4] D. Chulerttiyawong and A. Jamalipour, "A blockchain assisted vehicular pseudonym issuance and management system for conditional privacy enhancement," *Ieee Access*, vol. 9, pp. 127305–127319, 2021.

[5] M. Frydman, G. Ruiz, E. Heymann, C. Eduardo, and B. Miller, "Automating risk analysis of software design models," *The Scientific World Journal*, vol. 2014, pp. 1–12, 2014.

[6] D. Kreutz, F. Ramos, P. Verıssimo, C. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the Ieee*, vol. 103, pp. 14–76, 2015.

[7] I. Grgurević, Z. Kavran, and A. Pušeljić, "Simulation analysis of characteristics and application of software-defined networks," 2015.

[8] G. Segura, S. Skaperas, A. Chorti, L. Mamatas, and C. Margi, "Denial of service attacks detection in software-defined wireless sensor networks," 2020.

[9] W. Han and J. Xue, "Review about software defined networking," 2017.

[10] H. Rastegarfar and D. Kilper, "Robust software-defined optical networking for the power grid," 2016.

[11] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, "Sdiot: a software defined based internet of things framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, pp. 453–461, 2015.

[12] O. Yurekten and M. Demirci, "Sdn-based cyber defense: A survey," *Future Generation Computer Systems*, vol. 115, pp. 126–149, 2021.

[13] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in sdn: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102595, 2020.

[14] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2018.

[15] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[16] H. Alqahtani, I. Sarker, A. Kalim, S. Hossain, S. Ikhlaq, and S. Hossain, "Cyber intrusion detection using machine learning classification techniques," pp. 121–131, 2020.

[17] B. Aslahi-Shahri, R. Rahmani, M. Chizari, A. Maralani, M. Eslami, M. Golkar, and A. Ebrahimi, "A hybrid method consisting of ga and svm for intrusion detection system," *Neural Computing and Applications*, vol. 27, pp. 1669–1676, 2015.

[18] S. Park, S. Seo, C. Jeong, and J. Kim, "Online eigenvector transformation reflecting concept drift for improving network intrusion detection," *Expert Systems*, vol. 37, no. 5, p. e12477, 2020. e12477 EXSY-Nov-18-453.R3.

[19] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[20] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[21] I. Ahmad, S. Shahabuddin, H. Malik, E. Harjula, T. Leppänen, L. Loven, A. Anttonen, A. H. Sodhro, M. M. Alam, M. Juntti, *et al.*, "Machine learning meets communication networks: Current trends and future challenges," *IEEE Access*, vol. 8, pp. 223418–223460, 2020.

[22] M. Barreno, B. Nelson, R. Sears, A. Joseph, and J. Tygar, "Can machine learning be secure?," 2006.

[23] K. Sethi, E. Sai Rupesh, R. Kumar, P. Bera, and Y. Venu Madhav, "A context-aware robust intrusion detection system: a reinforcement learning-based approach," *International Journal of Information Security*, vol. 19, pp. 657–678, 2020.

[24] G. Apruzzese, M. Colajanni, L. Ferretti, and M. Marchetti, "Addressing adversarial attacks against security systems based on machine learning," in *2019 11th international conference on cyber conflict (CyCon)*, vol. 900, pp. 1–18, IEEE, 2019.

[25] M. Pawlicki, M. Choraś, and R. Kozik, "Defending network intrusion detection systems against adversarial evasion attacks," *Future Generation Computer Systems*, vol. 110, pp. 148–154, 2020.

[26] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When nas meets robustness: In search of robust architectures against adversarial attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 631–640, 2020.

[27] J. Aiken and S. Scott-Hayward, "Investigating adversarial attacks against network intrusion detection systems in sdns," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 1–7, IEEE, 2019.

[28] M. Abdelaty, S. Scott-Hayward, R. Doriguzzi-Corin, and D. Siracusa, "Gadot: Gan-based adversarial training for robust ddos attack detection," in *2021 IEEE Conference on Communications and Network Security (CNS)*, pp. 119–127, IEEE, 2021.

[29] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[30] H. Jiang, J. Lin, and H. Kang, "Fgmd: A robust detector against adversarial attacks in the iot network," *Future Generation Computer Systems*, vol. 132, pp. 194–210, 2022.

[31] OpenFlow, "Openflow switch specification."

[32] C. Zhong, T. Lin, P. Liu, J. Yen, and K. Chen, "A cyber security data triage operation retrieval system," *Computers Security*, vol. 76, pp. 12–31, 2018.

[33] MITRE, "Mitre attck." `https://attack.mitre.org/`, 2023. [Online; accessed 23-July-2023].

[34] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "Insdn: A novel sdn intrusion dataset," *IEEE access*, vol. 8, pp. 165263–165284, 2020.

[35] M. Liyanage, A. Braeken, A. D. Jurcut, M. Ylianttila, and A. Gurtov, "Secure communication channel architecture for software defined mobile networks," *Computer Networks*, vol. 114, pp. 32–50, 2017.

[36] J. Wang, Y. Tan, and J. Liu, "Topology poisoning attacks and countermeasures in sdn-enabled vehicular networks," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2020.

[37] C. Yoon, S. Lee, H. Kang, T. Park, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Flow wars: Systemizing the attack surface and defenses in software-defined networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3514–3530, 2017.

[38] T.-H. Nguyen and M. Yoo, "Analysis of link discovery service attacks in sdn controller," in *2017 International Conference on Information Networking (ICOIN)*, pp. 259–261, IEEE, 2017.

[39] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 151–152, 2013.

[40] A. Sangodoyin, B. Mohammed, M. Sibusiso, I. Awan, and J. P. Disso, "A framework for distributed denial of service attack detection and reactive countermeasure in software defined network," in *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 80–87, IEEE, 2019.

[41] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.

[42] I. Debicha, B. Cochez, T. Kenaza, T. Debatty, J.-M. Dricot, and W. Mees, "Review on the feasibility of adversarial evasion attacks and defenses for network intrusion detection systems," *arXiv preprint arXiv:2303.07003*, 2023.

[43] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[44] W. He, J. Wei, X. Chen, N. Carlini, and D. Song, "Adversarial example defense: Ensembles of weak defenses are not strong," in *11th USENIX workshop on offensive technologies (WOOT 17)*, 2017.

[45] S. H. Silva and P. Najafirad, "Opportunities and challenges in deep learning adversarial robustness: A survey," *arXiv preprint arXiv:2007.00753*, 2020.

[46] S. Venkatesan, H. Sikka, R. Izmailov, R. Chadha, A. Oprea, and M. J. De Lucia, "Poisoning attacks and data sanitization mitigations for machine learning models in network intrusion detection systems," in *MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM)*, pp. 874–879, IEEE, 2021.

[47] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli, "Bagging classifiers for fighting poisoning attacks in adversarial classification tasks," in *Multiple Classifier Systems: 10th International Workshop, MCS 2011, Naples, Italy, June 15-17, 2011. Proceedings 10*, pp. 350–359, Springer, 2011.

[48] A. Sarıkaya and B. G. Kılıç, "A class-specific intrusion detection model: hierarchical multi-class ids model," *SN Computer Science*, vol. 1, pp. 1–11, 2020.

[49] V. Bolon-Canedo, N. Sanchez-Marono, and A. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5947–5957, 2011.

[50] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in nsl-kdd cup 99 dataset employing svms," in *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*, pp. 1–6, IEEE, 2014.

[51] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[52] 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)*, pp. 1–6, IEEE, 2015.

[53] 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.

[54] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," in *2017 26th international conference on computer communication and networks (ICCCN)*, pp. 1–9, IEEE, 2017.

[55] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in sdn using machine learning approach," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 167–172, IEEE, 2016.

[56] N. Meti, D. Narayan, and V. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," in *2017 international conference on advances in computing, communications and informatics (ICACCI)*, pp. 1366–1371, IEEE, 2017.

[57] A. S. da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, "Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 27–35, IEEE, 2016.

[58] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, and D. Burgos, "An evolutionary svm model for ddos attack detection in software defined networks," *IEEE access*, vol. 8, pp. 132502–132513, 2020.

[59] P. Wang, K.-M. Chao, H.-C. Lin, W.-H. Lin, and C.-C. Lo, "An efficient flow control approach for sdn-based network threat detection and migration using support vector machine," in *2016 IEEE 13th international conference on e-business engineering (ICEBE)*, pp. 56–63, IEEE, 2016.

[60] 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)*, pp. 258–263, IEEE, 2016.

[61] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of ddos attack–based on deep learning in openflow-based sdn," *International Journal of Communication Systems*, vol. 31, no. 5, p. e3497, 2018.

[62] H. Jmila and M. I. Khedher, "Adversarial machine learning for network intrusion detection: A comparative study," *Computer Networks*, vol. 214, p. 109073, 2022.

[63] K. He, D. D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2023.

[64] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Information Sciences*, vol. 239, pp. 201–225, 2013.

[65] A. Paya, S. Arroni, V. García-Díaz, and A. Gómez, "Apollon: A robust defense system against adversarial machine learning attacks in intrusion detection systems," *Computers & Security*, vol. 136, p. 103546, 2024.

[66] J. Clements, Y. Yang, A. A. Sharma, H. Hu, and Y. Lao, "Rallying adversarial techniques against deep learning for network security," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 01–08, IEEE, 2021.

[67] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in iot networks," in *2019 IEEE global communications conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[68] Y. Jiang, G. Yin, Y. Yuan, and Q. Da, "Project gradient descent adversarial attack against multisource remote sensing image scene classification," *Security and Communication Networks*, vol. 2021, pp. 1–13, 2021.

[69] I. Homoliak, M. Teknos, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanacek, "Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach," *arXiv preprint arXiv:1805.02684*, 2018.

[70] K. Yang, J. Liu, C. Zhang, and Y. Fang, "Adversarial examples against the deep learning based network intrusion detection systems," in *MILCOM 2018-2018 ieee military communications conference (MILCOM)*, pp. 559–564, IEEE, 2018.

[71] Q. Cheng, S. Zhou, Y. Shen, D. Kong, and C. Wu, "Packet-level adversarial network traffic crafting using sequence generative adversarial networks," *arXiv preprint arXiv:2103.04794*, 2021.

[72] R. Kaur, B. Bansal, S. Majhi, S. Jain, C. Huang, and C. Yuen, "A survey on reconfigurable intelligent surface for physical layer security of next-generation wireless communications," *IEEE Open Journal of Vehicular Technology*, 2024.

[73] A. Alotaibi and M. A. Rassam, "Adversarial machine learning attacks against intrusion detection systems: A survey on strategies and defense," *Future Internet*, vol. 15, no. 2, p. 62, 2023.

[74] I. Debicha, R. Bauwens, T. Debatty, J.-M. Dricot, T. Kenaza, and W. Mees, "Tad: Transfer learning-based multi-adversarial detection of evasion attacks against network intrusion detection systems," *Future Generation Computer Systems*, vol. 138, pp. 185–197, 2023.

[75] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin, "Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 8, pp. 2632–2647, 2021.

[76] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017.

[77] G. Apruzzese, M. Colajanni, and M. Marchetti, "Evaluating the effectiveness of adversarial attacks against botnet detectors," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pp. 1–8, IEEE, 2019.

[78] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[79] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[80] A. Kaan Sarica and P. Angin, "A novel sdn dataset for intrusion detection in iot networks," in *2020 16th International Conference on Network and Service Management (CNSM)*, pp. 1–5, 2020.

[81] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[82] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018.

[83] Y. Liu, J. Kang, C. Guo, Y. Bai, and S. Meng, "Fault diagnosis algorithm of gearbox based on napso-vmd self-adaptive noise reduction and dual-sensor feature fusion," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–22, 2022.

[84] S. Amini, M. Soltanian, M. Sadeghi, and S. Ghaemmaghami, "Non-smooth regularization: improvement to learning framework through extrapolation," *Ieee Transactions on Signal Processing*, vol. 70, pp. 1213–1223, 2022.

[85] E. Solovyeva and A. Abdullah, "Dual autoencoder network with separable convolutional layers for denoising and deblurring images," *Journal of Imaging*, vol. 8, p. 250, 2022.

[86] S. Hasan and C. Linte, "Learning deep representations of cardiac structures for 4d cine mri image segmentation through semi-supervised learning," *Applied Sciences*, vol. 12, p. 12163, 2022.

[87] J. Liu, Y. Gao, and F. Hu, "A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm," *Computers Security*, vol. 106, p. 102289, 2021.

[88] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 399–410, Oct 2016.

[89] FAUCET, "Faucet sdn controller." `https://faucet.nz/`. Accessed: 22.06.2023.

[90] V. W. Berger and Y. Zhou, "Kolmogorov–smirnov test: Overview," *Wiley statsref: Statistics reference online*, 2014.

[91] M. J. Hashemi and E. Keller, "Enhancing robustness against adversarial examples in network intrusion detection systems," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 37–43, IEEE, 2020.

[92] K. Roshan, A. Zafar, and S. B. Ul Haque, "A novel deep learning based model to defend network intrusion detection system against adversarial attacks," in *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 386–391, 2023.

[93] W. D. Xiong, K. L. Luo, and R. Li, "Aidtf: Adversarial training framework for network intrusion detection," *Computers Security*, vol. 128, p. 103141, 2023.

[94] I. Debicha, R. Bauwens, T. Debatty, J.-M. Dricot, T. Kenaza, and W. Mees, "Tad: Transfer learning-based multi-adversarial detection of evasion attacks against network intrusion detection systems," *Future Generation Computer Systems*, vol. 138, pp. 185–197, 2023.

[95] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[96] Z. Wang, Y. Liu, D. He, and S. Chan, "Intrusion detection methods based on integrated deep learning model," *Computers Security*, vol. 103, p. 102177, 2021.

[97] M. Naveed, F. Arif, S. M. Usman, A. Anwar, M. Hadjouni, H. Elmannai, S. Hussain, S. S. Ullah, and F. Umar, "A deep learning-based framework for feature extraction and classification of intrusion detection in networks," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[98] T. Bai, J. Luo, J. Zhao, B. Wen, and Q. Wang, "Recent advances in adversarial training for adversarial robustness," *arXiv preprint arXiv:2102.01356*, 2021.

APPENDIX A

INSDN SOME FEATURES' HISTOGRAM PLOT

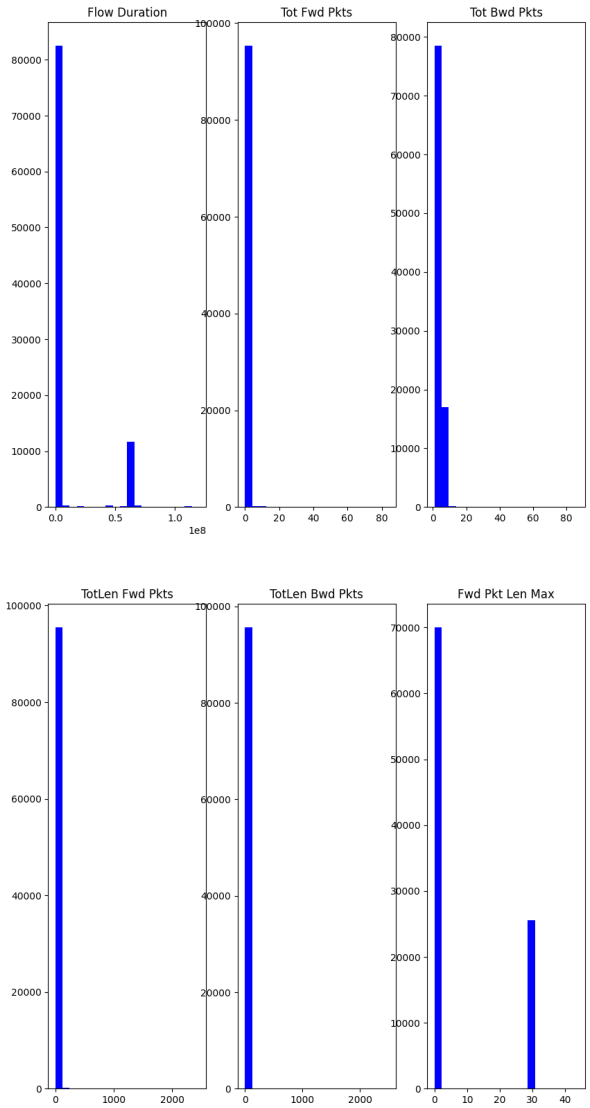

Figure 37: InSDN 2017 Some Features' Histogram

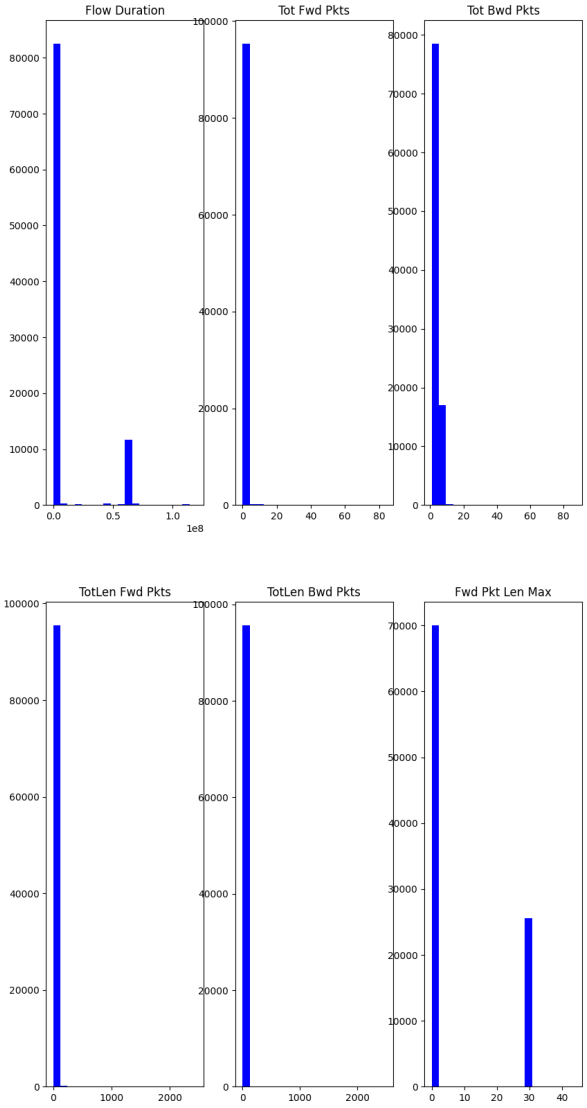

Figure 38: InSDN 2017 Some Features' Histogram

APPENDIX B

CICIDS 2017 SOME FEATURES' HISTOGRAM

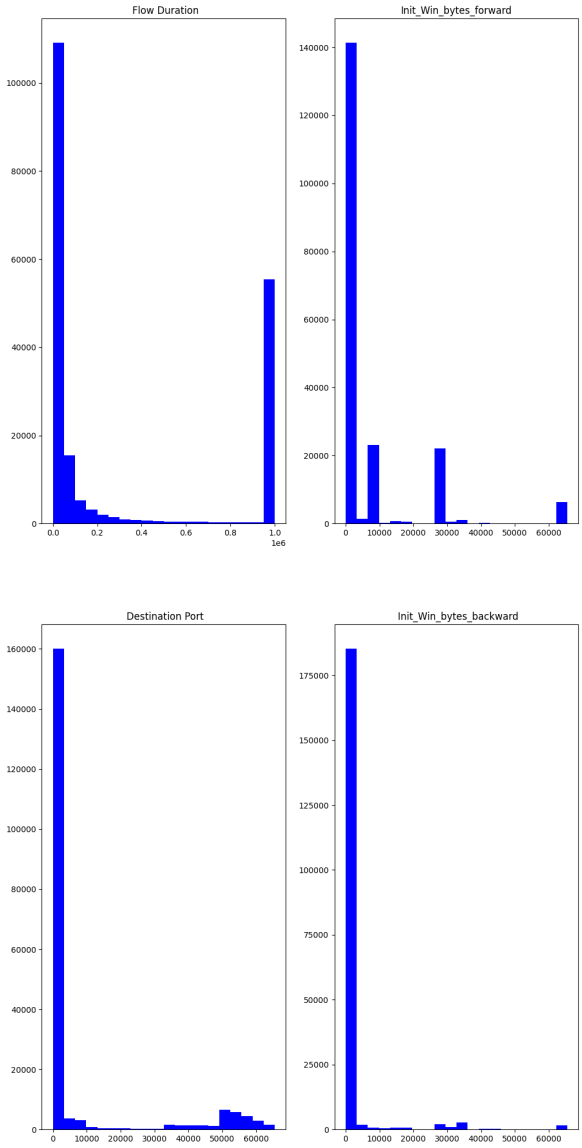

Figure 39: CICIDS 2017 Some Features' Histogram

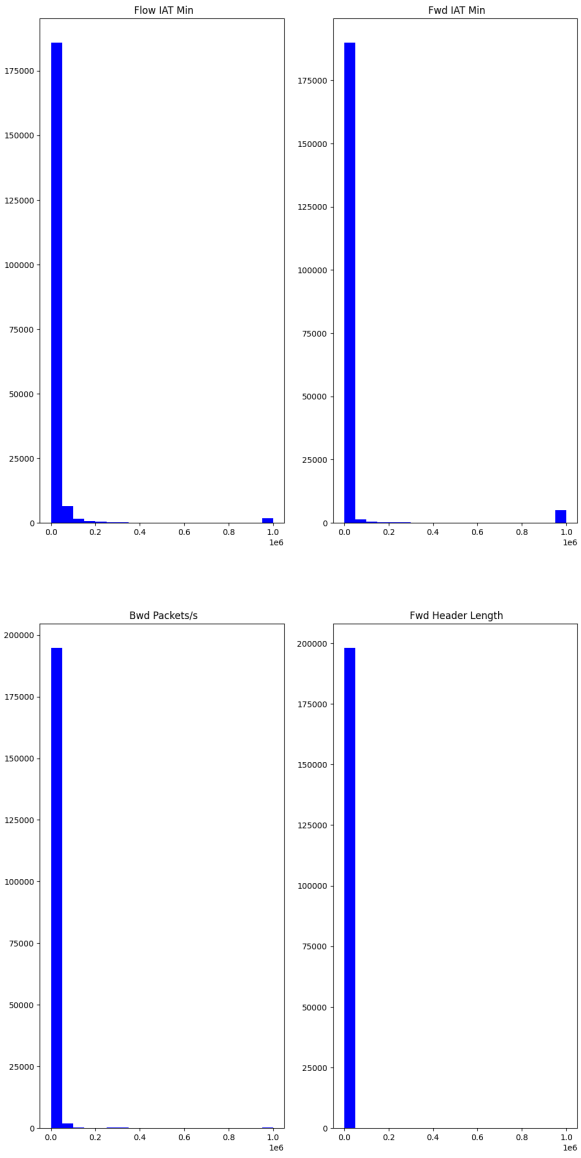

Figure 40: CICIDS 2017 Some Features' Histogram

APPENDIX C

CURRICULUM VITAE

Personal

Name: Alper Sarıkaya

Education

PhD. Degree
Middle East Technical University,
Graduate School of Informatics,
Information Systems 2018-2014

MSc. Degree
Middle East Technical University,
Graduate School of Informatics,
Information Systems 2016-2018

Undergraduate
Turkish Military Academy,
System Engineering,
2004-2008

Work Experience

Platoon and Company Commander
Turkish Land Forces, 2009-2016

Acting Chief of The Cyber Operation Centre
Turkish Armed Forces,2016-2018

Network and Security Instructor
Communication and Information School of Land Forces,2018-2020

IT Plan Officer
NATO Rapid Deployable Corps.-Türkiye,2020-2023

Functional Area Service Staff Officer
NATO Rapid Deployable Corps.-Türkiye,2023-Ongoing

Publications

1. Sarıkaya, A. (2018). Anomaly-based cyber intrusion detection system with ensemble classifier [M.S. - Master of Science]. Middle East Technical University.

2. Sarıkaya, A., & Kılıç, B. G. (2020). A class-specific intrusion detection model: hierarchical multi-class ids model. SN Computer Science, 1, 1-11.

3. Sarıkaya, A., Günel Kılıç, B., & Demirci, M. (2022). GRU-GBM: A combined intrusion detection model using LightGBM and gated recurrent unit. Expert Systems, 39(9), e13067.

4. Sarıkaya, A., Kılıç, B. G., & Demirci, M. (2023). RAIDS: Robust autoencoder-based intrusion detection system model against adversarial attacks. Computers & Security, 135, 103483.