Smart IDS

A Graduation Project Report Submitted
to
Faculty of Engineering, Helwan University
in partial fulfillment of the requirements of the degree
of
Bachelor of Science in Computer Engineering

Presented by

Mohamed Mahmoud Abdelmoneam **Mohamed Mahmoud Ahmed**

Bahaa-Eldeen Gamal Mahmoud

Supervised by
Dr. Ahmed Badawy

July 20, 2023

Abstract

Cyber attacks pose a significant threat to individuals, organizations, and governments worldwide. There are various types of attacks, including malware, phishing, denial-of-service (DoS) attacks, and social engineering. Malware attacks can be delivered through email, websites, or network vulnerabilities, infecting systems, and stealing data or causing damage.

Intrusion detection systems (IDS) are critical components of network security, designed to identify and prevent unauthorized access, misuse, and modification of data. They work by monitoring network traffic, identifying patterns that indicate an attack, and generating alerts or taking action to block the threat. In recent years, machine learning techniques, such as deep neural networks, have been applied to IDS to improve their accuracy and ability to detect previously unknown attacks. However, IDS also face challenges, such as the need for continuous updating to remain effective against new threats, and the potential for false alarms. Despite these challenges, IDS remains an essential tool for ensuring network security and protecting against cyber attacks.

Deep neural networks (DNNs) offer a promising solution for improving IDS accuracy by leveraging their ability to learn complex patterns and features from large amounts of data. In this paper, we review recent developments in IDS using DNNs, including the different types of architectures and training techniques. We also discuss the challenges of using DNNs for IDS, such as the need for large amounts of labeled data and the potential for adversarial attacks. Overall, the use of DNNs for IDS has the potential to greatly improve network security, but further research is needed to address the challenges and limitations of this approach.

Acknowledgment

Our gratitude to those who helped us cannot be put into words. First, we would like to express our appreciation and thankfulness to our supervisor,

Dr. Ahmed Badawy, for accepting our approach and his utmost care and support for us during our work. His guidance and mentorship have been critical to our success.

We would also like to thank our family, friends, and colleagues, who have always been there to support us and push us forward. We are defined by the people who surround us, and those people made us who we are today.

Contents

List of Figures

List of Tables

List of Abbreviations

ACM	Association for Computing Machinery
AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Access points
API	Application Programming Interface
ARP	Address Resolution Protocol
CAIDA	Cooperative Association for Internet Data Analysis
CIA	Confidentiality, Integrity andAvailability
CTF	Capture The Flag
DARPA	Defense Advanced Research Projects Agency
DDOS	Distributed Denial-of-Service
DEFCON	Defense Readiness Condition
DNN	Deep Neural Networks
DNS	Domain Name System
DOS	Denial of Service
FTP	File Transfer Protocol
GB	Giga Byte
GPU	Graphics Processing Uni
GT	Ground Truth
HIDS	Host-based Intrusion Detection System
HIPAA	Health Insurance Portability and Accountability Act
HTTP	Hyper Text Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Message Control Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
ISCX	Installation Support Center of Expertise
JSON	JavaScript Object Notation

KDD	Knowledge Discovery And Data
KU	Kyoto University
LAN	Local Area Network
LBNL	Lawrence Berkeley National Laboratory
LLDOS	Lincoln Laboratory DOS
MAC	Media Access Control
MADAM ID	Mining Audit data for automated models for Intrusion Detection
MD5	Message Digest Method 5
MIT	Massachusetts Institute of Technology
NFS	Network File System
NIDS	Network-based intrusion detection systems
NIS	network and information systems
NP	No Problem
NSL	Network Security Laboratory
OS	Operating System
PCI-DSS	Payment Card Industry Data Security Standard.
POP	Post Office Protocol
R2L	Remote to Local
REST	Representational State Transfer
RMSP	Root Mean Squared Propagation
SFTP	Secure File Transfer Protocol
SGD	Stochastic gradient descent
SIGKDD	Special Interest Group Knowledge Discovery and Data Mining
SMTP	Simple Mail Transfer Protocol
SPAN	Switched Port Analyzer
SQL	Structured query language
SRD	Structured requirements Definition
SSH	Secure Shell
SUID	Set-user Identification
SYN	Synchronized Sequence Number
TAP	Test Access Point
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TUIDS	temporally unique identifiers

U2R	User to Root
UDP	User Datagram Protocol
UNB	University of New Brunswick
UNIBS	University of Brescia
UNIX	UNiplexed Information Computing System
US	United States
VM	Virtual Machine
VSC	Visual Studio Code
WAN	Wide Area Network

Table 1: List of Abbreviations

Contacts

Team Members

Name	Email	Phone number
Mohamed Mahmoud Abdalmoneam	Mohamed.Elmoneam.98@h-eng.helwan.edu.eg	
Mohamed Mahmoud Ahmed	Mohamed.Hussien.73@h-eng.helwan.edu.eg	
Bahaa-Eldeen Gamal Mahmoud	Bahaa.Abdelmohsen.52@h-eng.helwan.edu.eg	

Table 2: Team members information

Supervisor

Name	Email	Phone number
Dr. Ahmed Badawy		

Table 3: Supervisor information

This page is intentionally left blank.

1. Introduction

The continuous development of computer systems has led companies, organizations, and individuals to increasingly rely on computer networks to perform their functions and provide their services in modern ways. However, at the same time, these networks have become vulnerable to hacking by attackers seeking to achieve illegitimate gains by exploiting some security vulnerabilities, leading to increased interest in issues of protection and security of these systems.

Intrusion detection is one of the most important and wide-ranging topics in the field of network and organizational security and protection of individuals. There are many methods used in this field, as well as many intrusion detection systems available. Despite the prominent role it plays, it still suffers from some drawbacks, so there is a need to continue researching intrusion detection systems to develop an optimal structure that achieves a high level of protection.

1.1 Motivation and Justification

Intrusion Detection Systems (IDS) are an essential component of modern security infrastructure for networks and organizations. IDS are designed to detect and respond to security threats, such as hacking attempts, malware infections, and unauthorized access.

The motivation for using IDS is to protect critical assets and data from various types of attacks, including insider threats, external attacks, and advanced persistent threats. IDS can help organizations identify security breaches early on, allowing them to take appropriate measures to mitigate damage and prevent further attacks. Furthermore, IDS can provide organizations with valuable insights into their security posture, by analyzing network traffic and identifying patterns and anomalies that may indicate potential security threats. This information can then be used to improve security policies and procedures, and to implement additional security measures as needed.

The justification for IDS lies in the fact that threats to network security are constantly evolving, and traditional security measures such as firewalls and antivirus software may not be enough to protect against them. IDS can provide an additional layer of defense, allowing organizations to identify and respond to threats in real-time, and to continuously monitor and improve their security posture. Overall, IDS are a critical tool in the fight against cyber threats, and their use is essential for organizations that want to protect their assets and data from malicious actors.

1.2 Project Objectives and Problem Definition

Project objectives for an IDS implementation:

- Detection of Security Threats: The primary objective of an IDS is to detect security threats on the network, such as unauthorized access, malware, and other attacks.
- Real-time Alerting: IDS should provide real-time alerts to security personnel when a threat is detected, allowing for immediate action to be taken to mitigate the threat.
- Incident Response: IDS should provide information to support incident response activities, including forensic analysis of security incidents.
- Compliance: IDS should help organizations meet regulatory compliance requirements, such as PCI-DSS, HIPAA, and others.
- Reporting and Analytics: IDS should provide comprehensive reporting and analytics capabilities to help security personnel identify trends and patterns in network traffic, and to help identify areas where additional security measures may be needed.

The problem definition for an IDS implementation:

- Network Complexity: As networks become more complex, it becomes increasingly difficult to monitor network traffic and detect security threats. IDS must be able to cope with the scale and complexity of modern networks.
- False Positives and False Negatives: IDS can generate false positives (alerts for non-security events) and false negatives (failures to detect actual security events). These issues can reduce the effectiveness of IDS and increase the workload for security personnel.

- Cost: IDS can be expensive to implement, maintain, and operate. Organizations must balance the cost of IDS against the potential benefits of improved security.
- Integration: IDS must be integrated with other security systems, such as firewalls, antivirus software, and intrusion prevention systems, to provide comprehensive protection against security threats.
- Skilled Personnel: Effective IDS implementation requires skilled personnel with experience in network security, intrusion detection, and incident response. Organizations may struggle to find and retain personnel with these skills.

1.3 Project Outcomes

- System Requirements: The project scope should define the requirements for the IDS system, including hardware, software, and network infrastructure.
- Network Coverage: The scope should define the network coverage of the IDS system, including the types of traffic that will be monitored and the network segments that will be covered.
- Integration: The scope should define the integration of the IDS system with other security systems, such as firewalls, antivirus software, and intrusion prevention systems.
- Alerting and Reporting: The scope should define the alerting and reporting capabilities of the IDS system, including the types of alerts that will be generated and the reporting formats that will be used.
- Policies and Procedures: The scope should define the policies and procedures that will be developed and implemented to support the operation of the IDS system, including incident response and escalation procedures.
- Training and Documentation: The scope should define the training and documentation requirements for the IDS system, including training for security personnel and documentation of system configuration and operation.
- Testing and Validation: The scope should define the testing and validation requirements for the IDS system, including the types of tests that will be conducted and the validation criteria that will be used.

- Maintenance and Support: The scope should define the maintenance and support requirements for the IDS system, including system updates, upgrades, and troubleshooting.

1.4 Document Organization

- Introduction: This section should provide an overview of the document and the IDS implementation project. It should also include information on the purpose and scope of the document.
- System Requirements: This section should outline the hardware, software, and network infrastructure requirements for the IDS system.
- Network Coverage: This section should define the network coverage of the IDS system, including the types of traffic that will be monitored and the network segments that will be covered.
- Integration: This section should define the integration of the IDS system with other security systems, such as firewalls, antivirus software, and intrusion prevention systems.
- Alerting and Reporting: This section should define the alerting and reporting capabilities of the IDS system, including the types of alerts that will be generated and the reporting formats that will be used.
- Policies and Procedures: This section should define the policies and procedures that will be developed and implemented to support the operation of the IDS system, including incident response and escalation procedures.
- Training and Documentation: This section should define the training and documentation requirements for the IDS system, including training for security personnel and documentation of system configuration and operation.
- Testing and Validation: This section should define the testing and validation requirements for the IDS system, including the types of tests that will be conducted and the validation criteria that will be used.
- Maintenance and Support: This section should define the maintenance and support requirements for the IDS system, including system updates, upgrades, and troubleshooting.
- Conclusion: This section should summarize the key points of the document and provide any additional information that may be relevant to the IDS implementation project.

2. Network Security and Intrusion Detection System

2.1 Computer security

The concept of computing has changed the meaning of the internet as we know it. The possibilities and opportunities available are unlimited, and with them comes an increased risk of security breaches. Computer security focuses primarily on protecting a specific source or data and valuable information within a single computer device. Security is defined as the reaction taken against security threats resulting from harmful actions by some individuals. Data value can be violated through three third-party methods that violate privacy, safety, and information availability. Generally, computer protection is referred to as the CIA acronym representing the following three concepts:

- **Confidentiality**: Preventing unauthorized individuals from disclosing or accessing information. This means that information can only be accessed by authorized individuals.
- **Integrity**: Maintaining the integrity of information by preventing unauthorized modification.
- **Availability**: Refers to the computer's ability to work and provide the expected resources and services to legitimate individuals upon request.

2.2 Network security

Certainly, here is an expanded version of your text on network security:

Network security is the process of protecting computer networks from unauthorized access, theft, damage, and misuse of data. It involves the implementation of various security measures to safeguard the confidentiality, integrity, and availability of network resources. Organizations use network security to ensure that their sensitive data and operations remain protected from internal and external threats.

To implement effective network security, organizations need to identify all potential security threats and implement a set of security measures to prevent them.

These measures can include access control, firewalls, intrusion detection systems, antivirus software, and encryption. Access control is an essential component of network security that involves limiting network access to authorized users only. Firewalls are used to monitor incoming and outgoing network traffic and prevent unauthorized access to the network. Intrusion detection systems are used to detect and alert security personnel of any unauthorized access attempts or security breaches. Antivirus software is used to detect and prevent malware and virus attacks, while encryption is used to protect sensitive data by converting it into a code that can only be deciphered with a decryption key.

Despite the implementation of these security measures, network security threats continue to evolve, and organizations must remain vigilant to protect their networks. One of the most significant challenges facing organizations in network security is the lack of cybersecurity expertise and resources. Organizations need to invest in training their employees on how to identify and prevent security threats, as well as hiring cybersecurity experts to monitor and manage the network security system.

Another challenge facing organizations in network security is the increasing sophistication of cyber threats. Cybercriminals are constantly developing new and more sophisticated methods of attacking networks, making it difficult for organizations to keep up. To stay ahead of these threats, organizations need to adopt a proactive approach to network security, regularly updating their security measures and staying informed of the latest security threats.

In conclusion, network security is a critical aspect of modern-day business operations. It is essential for organizations to implement effective security measures to protect their networks from internal and external threats. By identifying potential security threats, implementing security measures, and staying informed of the latest security threats, organizations can ensure the safety and integrity of their networks and safeguard their sensitive data and operations.

2.3 Firewalls

A firewall is a software or device that is installed to protect a network. It is the first line of defense for the network and works as a filter. Firewalls are not capable of detecting attacks, but they block all traffic except for packets that match certain rules, such as packets directed to a specific port or coming from a specific IP address. These rules are manually configured by network administrators based on the security policy of the organization. This means that the effectiveness of the firewall depends on the skill of the network administrator.

The firewall can be considered a simple barrier that protects access to a specific location. Therefore, it is essential to place these barriers in strategic locations to ensure their full effectiveness. The effectiveness of a firewall can also depend on its type and configuration. There are different types of firewalls, such as packet filtering firewalls, stateful inspection firewalls, and application-level gateways.

Packet filtering firewalls are the most basic type of firewall and work by inspecting packets and allowing or blocking them based on certain criteria, such as source and destination IP addresses and port numbers. Stateful inspection firewalls, on the other hand, keep track of the state of network connections and use this information to make decisions about whether to allow or block traffic.

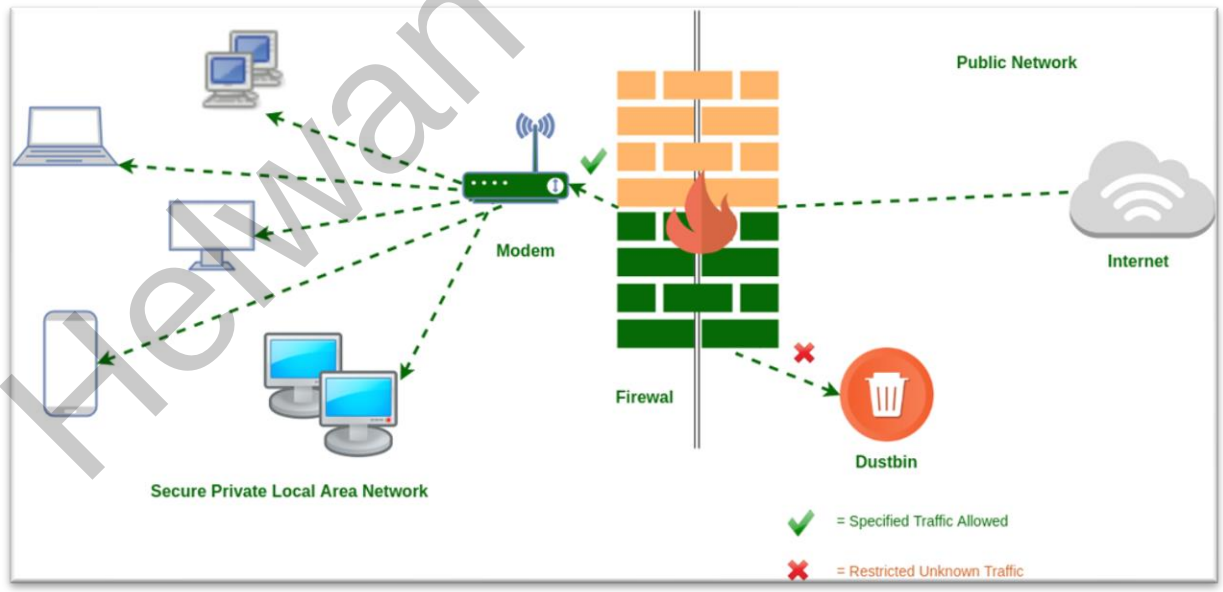


Figure 1: Packet filtering firewalls

Application-level gateways, also known as proxy servers, act as intermediaries between clients and servers and can inspect application-level data to determine whether to allow or block traffic.

In conclusion, firewalls are an essential component of network security and play a crucial role in protecting networks from unauthorized access and security threats. By filtering traffic and blocking potentially harmful packets, firewalls create a barrier between the network and potential attackers. However, the effectiveness of a firewall depends on its configuration and the skill of the network administrator, and it is important to place firewalls in strategic locations to ensure their full effectiveness.

The security policy determines which network devices will operate at certain points within the network. Communications to the Internet pass through the firewall, which blocks all traffic except for that which is permitted by the security policy. For example, blocking traffic on a specific port is simply a matter of notifying the firewall to close that port.

However, firewalls are not sufficient to protect systems for several reasons. First, unauthorized activities on the Internet are not only carried out by external attackers but also by internal sources, such as fraudulent employees or individuals misusing their privileges for personal gain or revenge. These internal activities cannot always be prevented by a firewall that sometimes stops traffic from entering or leaving the network. On the other hand, firewalls work by allowing only pre-defined legitimate traffic to pass through, without inspecting the contents of that legitimate traffic. There is another reason that firewalls cannot prevent all types of attacks. Standard network security solutions with firewalls were not designed to deal with network and application layer attacks, such as Denial of Service (DoS) attacks, worms, viruses, and Trojan horses. These reasons, along with the increasing prevalence of online threats, have led to the use of Intrusion Detection Systems (IDS).

In conclusion, while firewalls provide an essential layer of network security, they are not sufficient to protect against all types of security threats. Internal threats and application layer attacks require additional security measures, such as Intrusion Detection Systems. By implementing IDS, organizations can detect and respond to potential security incidents, reducing the risk of a security breach and protecting their networks and sensitive data.

2.4 Intrusion Detection System (IDS)

Intrusion Detection Systems (IDS) are used to monitor and analyze events occurring in a computer system or network in order to detect signs of intrusion. Intrusion has been defined as attempts to compromise the confidentiality, integrity, and availability of information, resources, and services of a computer or network system, or interventions that go beyond the security mechanisms of the system and that can be carried out by attackers who have gained access to the system via the Internet, authorized users who attempt to obtain unauthorized privileges, or users who abuse their granted privileges to harm the system or network.

An IDS is the hardware and software that controls this monitoring and analysis process. The main aim of an IDS is to detect intrusion while it is happening, not after it has occurred, and then to alert the responsible person of the problem by sending an email or launching some type of warning, and to be able to take any action to minimize the harm that may be caused to the system due to the intrusion.

The second goal is to gather data from the system, record all important events, and determine the source of the attack. This data is used for legal purposes as evidence or proof against the attacker.

2.4.1 Main Advantage and Features of IDS

The following items summarize the characteristics that describe an effective intrusion detection system:

1. The IDS must be able to operate continuously without human monitoring. The system must be reliable enough to be allowed to work in the background of the system, with a focus on the ability to examine its internal operations from the outside.

2. The IDS must be able to recover in case of system failure, whether accidental or due to malicious activity. In the event of a failure, the IDS should be able to recover to its previous state and continue its operations unaffected.

3. The IDS must be resistant to tampering:

a. There should be difficulty for attackers to disable or modify the IDS.

b. The IDS must be able to monitor itself and detect whether there has been any modification to the intrusion by the attacker.

4. The IDS should impose minimal costs on the systems where it is deployed to avoid interference with the normal operation.

5. The IDS must be adjustable to enforce the security policies of the system being monitored accurately.

6. The IDS should be easy to implement. This can be achieved by making it executable on different operating systems and architectures, with simple loading mechanisms and ease of use and understanding by the operator.

7. The IDS must be able to adapt to changes in the system and user behavior over time. For example, when a new application is installed, users alternate between activities or new resources become available, these changes can cause changes in the system that are detected by the IDS.

8. The IDS must be able to detect attacks in a way that:

a. The IDS should not identify any legitimate activity as an attack (false positives).

b. The IDS should not fail to detect a real attack as an attack (false negatives). It should be difficult for the attacker to hide their actions to avoid detection.

c. The IDS should be able to report attacks as quickly as possible from the time they occur.

d. The IDS should be comprehensive enough to detect different types of attacks.

2.4.2 positioning IDS in network

An intrusion detection system should be strategically placed in locations where it can see network traffic to analyze it and achieve maximum benefit. Most companies and organizations are typically supported by network-based intrusion detection systems (NIDS) in addition to firewalls. It is important to consider the location of the intrusion detection system in relation to the firewall.

The intrusion detection system can be placed between the firewall and the internal network to detect intrusion that may pass through the firewall, or between servers

and user groups to detect internal breaches, or before the firewall as shown in the figure.

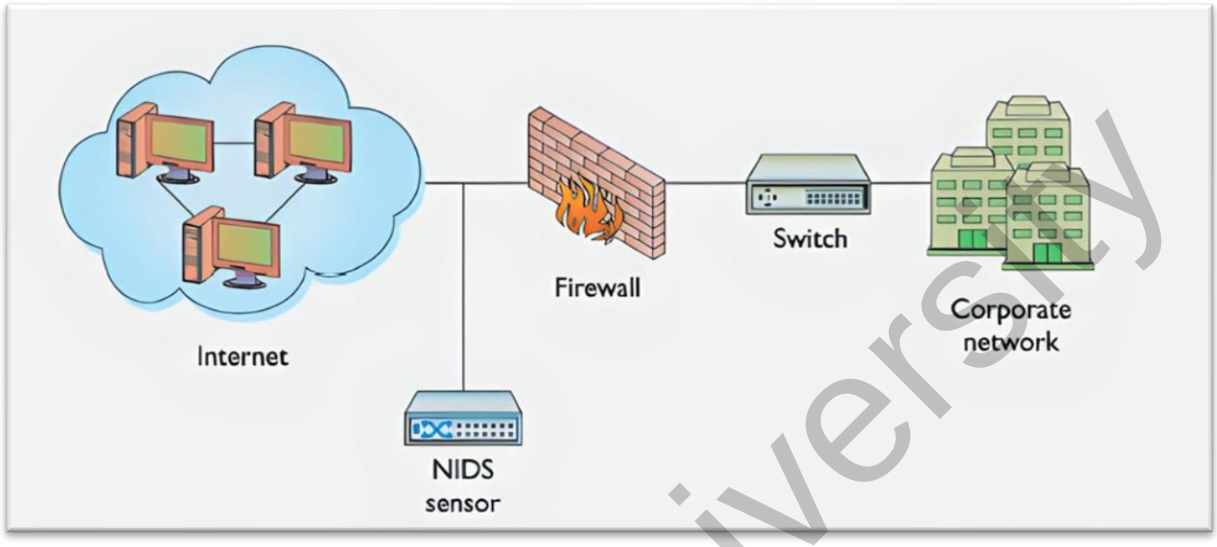


Figure 2: IDS located in front of the firewall

The figure shows an intrusion detection system placed before the firewall. In this case, the NIDS will monitor the traffic coming from the Internet, including attacks against the firewall. This includes traffic that is blocked by the firewall and not allowed into the network. In this type of deployment, the NIDS will generate a large number of alerts, including alerts on traffic that may be blocked by the firewall.

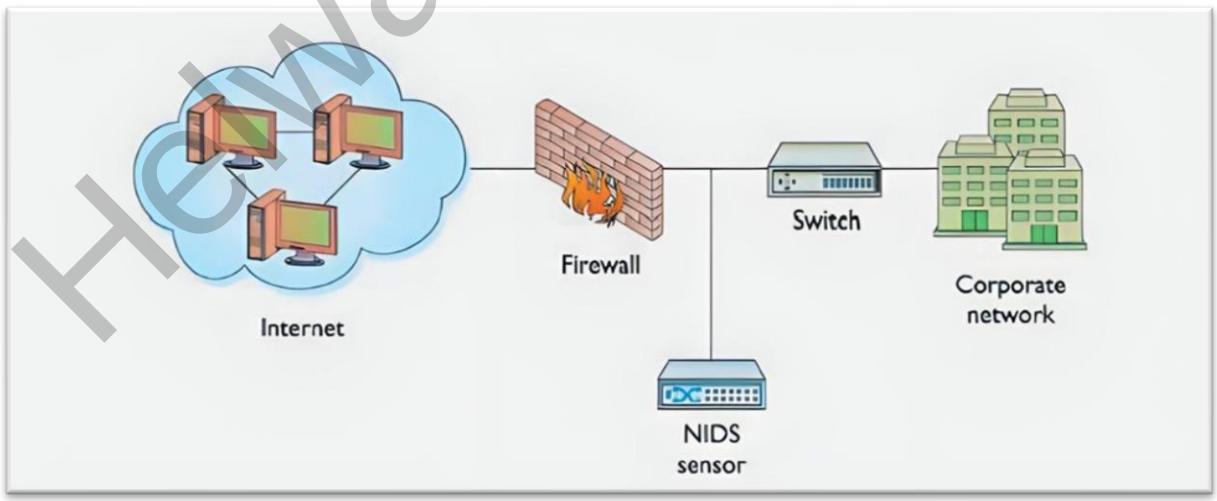


Figure 3: IDS located behind the firewall

The figure shows an intrusion detection system placed after the firewall. In this case, the NIDS will monitor and analyze the traffic passing through the firewall into the network. While this deployment does not allow the NIDS to see attacks against the firewall, it generates fewer alerts.

2.4.3 Classifying of IDS

There are several methods for classifying intrusion detection systems based on different analysis and monitoring techniques. Each method has its advantages and disadvantages, and the choice of method depends on the system's needs and the intended purpose.

2.4.3.1 Classifying Intrusion Detection Systems by Information Source

The most common method for classifying intrusion detection systems is based on the location of the information source they monitor. The primary information sources are network packets captured from the network backbone or local network segments, operating systems, and important files. Intrusion detection systems can be classified into network-based systems, also known as network intrusion detection systems (NIDS), host-based systems, and hybrid systems that combine both types.

- **Network based Intrusion Detection System (NIDS)**

NIDS is the most common form of intrusion detection system. These systems detect attacks by capturing and analyzing network packets through sniffing at network segments or switches. In this case, the system is placed on a network segment rather than a single device within the network, or it is placed to monitor a gateway on the switch. This allows the system to monitor all packets moving between a group of computers connected to the network, by matching one or more packets with a database of attack signatures, or by analyzing traffic anomalies.

NIDS can be placed outside firewalls to alert the responsible person to incoming packets that may bypass the firewall.

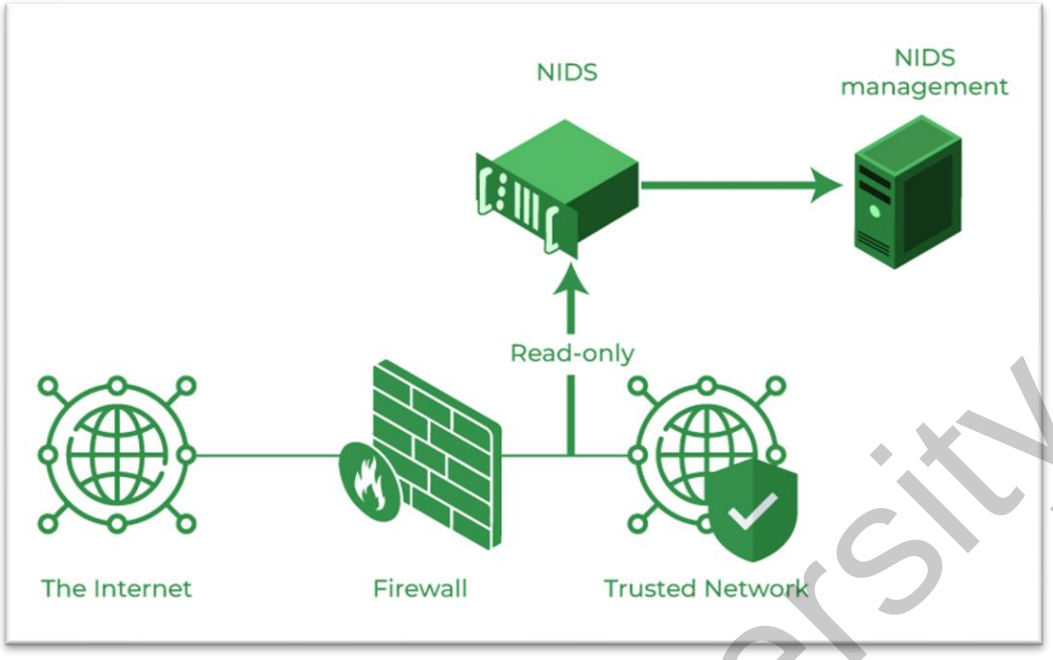


Figure 4: Network using three NIDS units

The figure shows a network using three NIDS units placed on strategic network segments to monitor network traffic for all devices on the segment. This topology represents a standard network configuration, where public servers in subnets are protected by NIDSs. When a public server in the subnet is at risk, it can become a platform for launching additional security breaches. Therefore, it is essential to ensure accurate monitoring to prevent further damage.

Advantages of Network-based Intrusion Detection Systems (NIDS):

- Relatively inexpensive, as NIDS can be applied to each network segment, reducing management burden as there is no need to install the detection software at the host level.
- NIDS are well-protected due to the difficulty of identifying their presence or location on the network.
- A small number of these systems placed in suitable locations can monitor a large network.
- Easy to implement as they do not affect standard systems or infrastructure. These systems are independent operating systems that monitor all attacks within the network sector regardless of the type of program the host is running.
- Detect failed attacks, NIDS implemented outside the firewall can detect attacks that may be rejected by the firewall but can be useful for forensic analysis.

Disadvantages of Network-based Intrusion Detection Systems (NIDS):

- NIDS may fail to identify an attack when the network size becomes too large.
- NIDS cannot analyze encrypted packets, making some traffic invisible to the process, reducing the effectiveness of NIDS.
- Attacks that include fragmented or corrupted packets cannot be easily detected.

❖ **Host-based Intrusion Detection System (HIDS)**

is a system that operates on information collected from a single computer system. HIDS employs monitoring sensors, also known as clients, on each host to be monitored. Generally, the most common forms of information sources used by HIDS are audit logs, system logs, and critical system files. The client checks these sources for unauthorized changes or suspicious activity patterns. This allows HIDS to analyze activities reliably and accurately identify users and processes involved in a specific attack on the operating system.

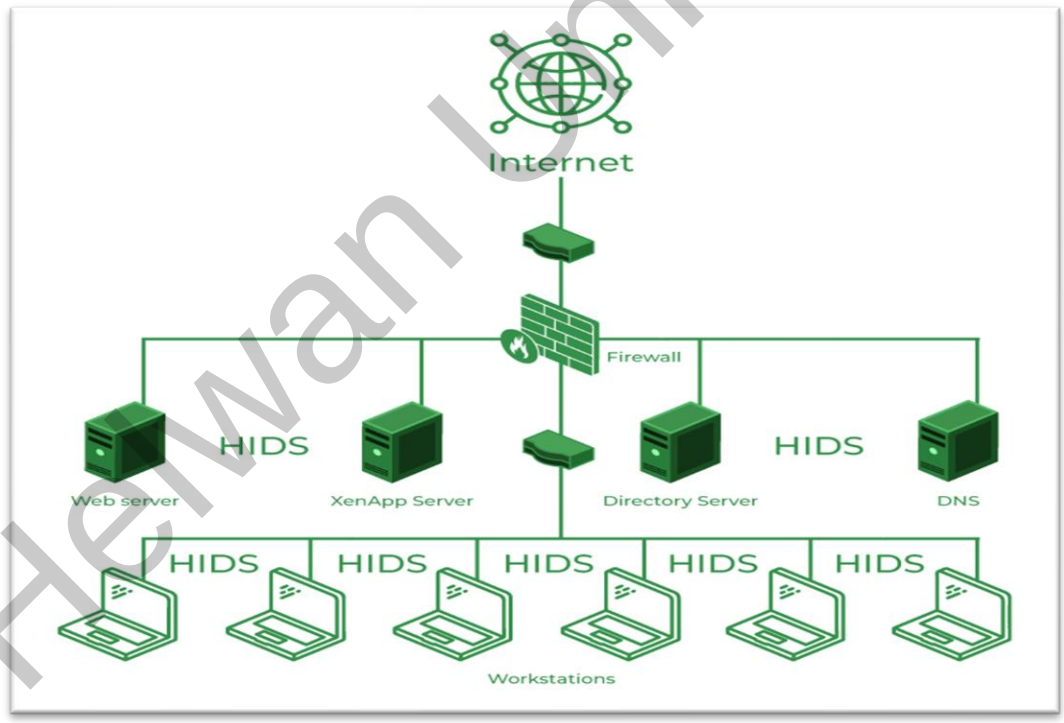


Figure 5: Network that uses HIDS

The diagram shows a network that uses HIDS on specific servers and host computers. The HIDS rule set is assigned to the mail server to protect it from security vulnerabilities within the mail server. During installation, individual host devices

can be configured with a rule set. New rules can be periodically loaded to address new weaknesses.

- **Advantages of Host-based Intrusion Detection System (HIDS)**
  - Verification of successful or failed attacks: Since HIDS uses system logs that contain events that have already occurred, it can accurately determine whether an attack has occurred or not with fewer false positives than network-based IDS.
  - System activity monitoring: HIDS sensors monitor user and file access activities, file modifications, changes to file permissions, attempts to install new executable files, as well as all login and logout activities, user activities during network connections, file system changes, and activities typically performed by administrators. Operating systems record any event when user accounts are added, deleted, or modified.
  - Detection of attacks that network-based IDS may miss: HIDS can detect attacks that network-based IDS fail to detect. For example, when an unauthorized user makes changes to system files from the system console, this type of attack is not usually noticed by network sensors.
  - No additional hardware required: HIDS sensors are deployed on host systems, so no additional physical equipment is required.

- **Disadvantages of Host-based Intrusion Detection System (HIDS)**
  - Additional management efforts: Additional management efforts are required to install and manage HIDS.
  - Vulnerability or reduced performance: Both direct attacks and attacks against the host operating system can expose HIDS to risk or reduce its performance.
  - Large log file size: Audit and review log files for the operating system occupy a large amount of disk space and require an increase in disk capacity, which can reduce system performance.
  - Installation on every device on the network: HIDS must be installed on every device on the network.
  - Can be disabled by a denial-of-service attack: HIDS can be disabled by a denial-of-service attack.

2.4.3.2 Classifying Intrusion Detection Systems by detection method

These methods are the foundation of intrusion detection technology and are responsible for detecting malicious activities within a system. Detection methods analyze monitored information and trigger alerts when detecting malicious traffic. Based on these methods, intrusion detection systems can be classified into anomaly-based detection systems and signature-based detection systems.

❖ **Anomaly-based intrusion detection systems**

rely on defining what is considered normal or allowed behavior within the system and then alerting on any action or event that falls outside of these boundaries. The system assumes that malicious events differ from normal behavior, and therefore searches for deviations to detect an attack. These systems create profiles of historical data collected during a period of normal operation. Then, event data is collected to determine when the monitored activity deviates from normal behavior and triggers an alert.

Special tools are used to perform statistical studies of the normal behavior of users or the system as a whole, and statistical information is usually about system load, memory usage, network traffic, certain system call sequences, or event timestamps. In all cases, normal behavior is determined based on usage patterns, which may be permanently modified based on changing user needs. Techniques such as neural networks can be applied to automatically adjust the behavior patterns.

➢ **Advantages of Anomaly-based Intrusion Detection Systems**

- The information generated by anomaly detectors can be used to determine attack signatures for misuse detectors.
- Effective in detecting new and unknown attacks.
- Does not require continuous maintenance.

➢ **Disadvantages of Anomaly-based Intrusion Detection Systems**

- Require very large training sets to distinguish between legitimate and illegitimate traffic.
- Generate a large number of false alarms.

- Cannot identify a specific attack and cannot determine if the attack was successful or not.

❖ **Signature-based Intrusion Detection Systems**

also known as Misuse Detection, generate alerts based on specific attack signatures. This type of signature-based attack includes specific traffic patterns or activity based on known malicious activity. The basic assumption is the model that is written to describe the bad behavior, after which the system compares the information sequence to this model to determine what is normal and what is malicious. These systems are accurate and generate fewer false alarms, but they can only detect intrusions if there is a specific pre-defined model for them.

➢ **Advantages of Signature-based Intrusion Detection Systems**

- Very effective in detecting attacks without generating a large number of false alarms.
- Difficult to bypass.

➢ **Disadvantages of Signature-based Intrusion Detection Systems:**

- Can only detect attacks that are pre-defined.
- Require continuous updates of signatures for new attacks.
- Ineffective at high speeds, as it must match all packets with signatures to detect an attack, which requires a significant amount of computing resources.

❖ **Hybrid-based Intrusion Detection**

combines both network-based and host-based intrusion detection systems, taking advantage of the strengths and benefits of both solutions. The next generation of intrusion detection systems needs to combine both technologies to improve the network's resistance to attacks and misuse, as well as enhance security policies and provide greater flexibility in application and deployment options.

The hybrid intrusion detection system is a combination of both network-based and host-based intrusion detection systems, providing a mix of the strengths of both

approaches. The way it operates varies from one product to another, making it difficult to define and identify hybrid intrusion detection systems accurately.

2.4.4 Connecting IDS

There are many ways to connect a network intrusion detection device to capture and monitor network traffic. Below are three scenarios, each with servers, a workstation, and an IDS all connected with a network tool linked to an uplink port. In information technology, the term "uplink" refers to the wireless connection established from the ground to a satellite orbiting the earth. The same term is used in computer networks to refer to a wired or wireless connection from a local area network (LAN) to a wide area network (WAN). The servers and workstation are considered internal, and the uplink leads to an external network. The red lines represent the connections used to monitor traffic, while the dashed black line represents the connection that can be used to remotely manage the IDS by a system within the internal network.

The diagram shows IDS connected either to a hub distributor or to a switch exchange capable of configuring a Port Switched Analyzer (SPAN) port.

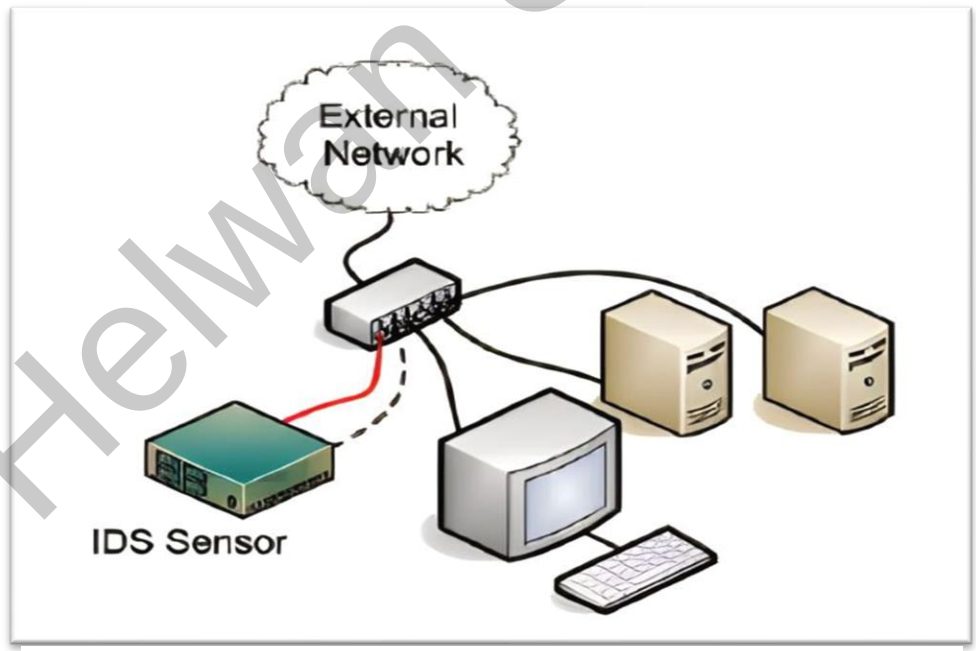


Figure 6: Connect the IDS using a Hub or Switch

A network hub is a physical device that, when data frames reach it, broadcasts them to all ports. Only the destination system processes the data, while other devices ignore it. In such an environment, an IDS can be connected to one of the hub's ports, allowing it to obtain all the traffic passing through the network.

In a network that uses switches, packets are only passed from the source device to the destination device, as indicated in the IP address, unlike the case with a network connected through a hub, where packets are broadcast to every device on the network. In this environment, a technology called Mirroring Port or SPAN can be used, where the reflecting port receives a copy of the packets from all ports. The IDS device is connected to the reflecting port or SPAN port so it can process all packets regardless of their destination.

As for the second scenario, it involves using a Point Access Test Network (TAP), as shown in the diagram (2-7). A TAP is an external monitoring device that copies the traffic passing between two network nodes. Network TAPs are not usually found in typical computer networks but can be purchased when needed for a quick monitoring solution, for example, to monitor a problem or to temporarily apply an IDS. They are needed when unmanaged switches or hubs are used in the network.

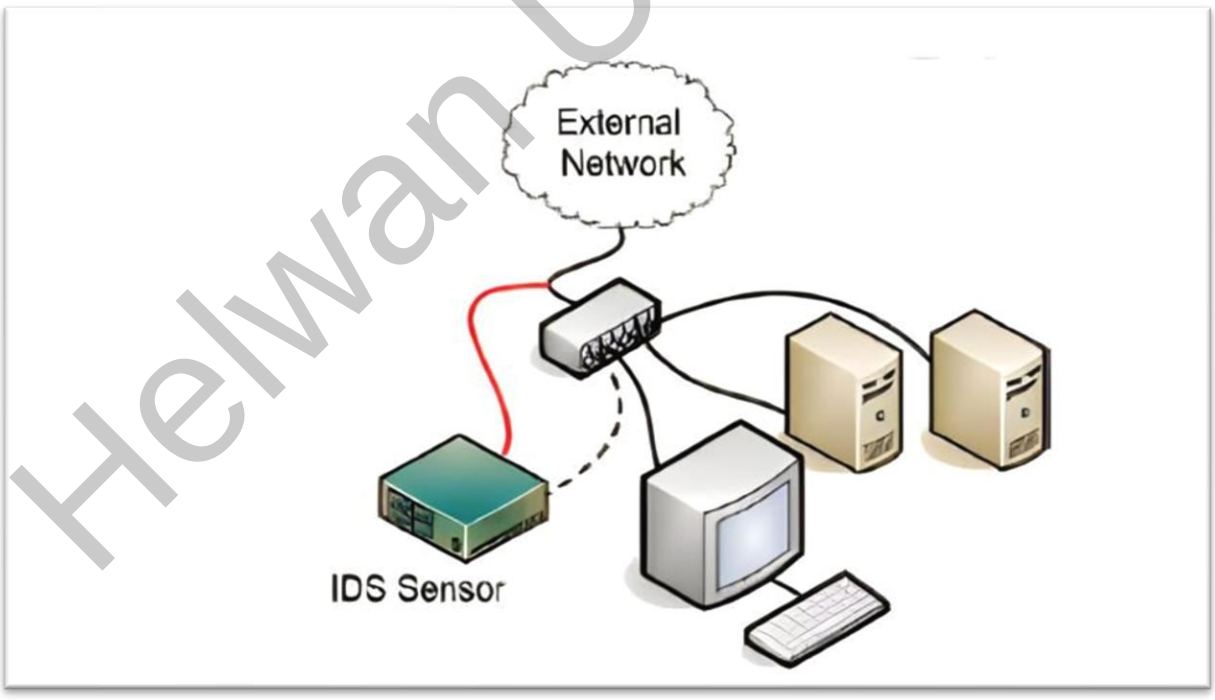


Figure 7: Connect IDS using Network TAP

Network access points (APs) are devices that have three interfaces: input, output, and test port. The IDS is connected to the test port, where it can see all the traffic. These devices do not cause any delay or affect the data traffic, and they do not have an IP or device address.

The last scenario shows the IDS connected in an "inline" mode, as shown in the diagram. This case involves two connections, one connected to the uplink port of the switch and the other connected to the external network.

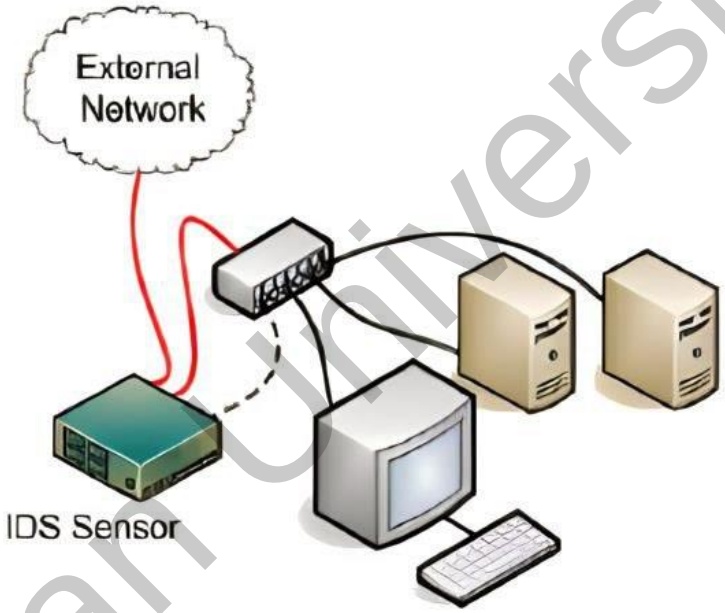


Figure 8: The IDS connected in an "inline" mode

In most cases, this is not the best method to use because a failure of the IDS system will prevent internal network systems from communicating with external systems. The benefit of this method is to ensure that all packets are seen by the IDS, which does not happen when using a SPAN port, especially when the switch is busy processing a large volume of traffic.

This chapter discussed the security threats faced by networks and organizations, as well as the most important methods and technologies used to protect and prevent them, including intrusion detection systems explained in their various types and a discussion of the features and drawbacks of each.

3. Literature Survey

This section will provide an overview of the techniques and working methods that we found while researching and preparing to work on this project. Many researchers and scientists have developed many algorithms to deal with intrusion detection systems. The section also explains what are the standard stages that we follow to reach a satisfactory final result in this subject.

With all this progress in research to build IDS in terms of finding software solutions for them was very weak. Therefore, we will mention in this section generally about the methods, datasets, and the phases of completing a project related to dealing with IDS programmatically.

3.1 Introduction

Creating a dataset suitable for testing IDS is expensive and requires a lot of precision and time, as it is necessary to adjust the real working environment to explore all possibilities of attack, it is a process of examination, transmission, Data modeling and deciding how to organize, classify, link and display this data.

3.2 The importance of datasets

It is difficult to evaluate, compare, and apply a system capable of detecting new attacks in the field of intrusion detection, especially when using anomaly-based intrusion detection. It is essential that these systems be tested and evaluated before being applied in any real environment, using real addressed traffic that includes a comprehensive set of attacks. This is a significant challenge given the scarcity of such data sets. Therefore, detection methods and systems are evaluated using several publicly available data sets that may lack comprehensiveness, completeness, or may be outdated. There are many reasons that justify the importance of the datasets used, here are some of them:

1. Researchers must repeat experiments on the data set and obtain similar results when using the same method. It is very important to test the proposed method and its compatibility with the evolving nature of attacks and network scenarios.

2. Research is still ongoing in the field of intrusion detection in search of new methods and techniques to effectively detect anomalies within the network. Data sets are an important and essential factor in validating and testing the performance of each new approach.

3. Use datasets to show improvements in the performance of old methods in a measurable way, i.e. by comparison between them. For example, over the past years the DARPA dataset has been used to evaluate the performance of systems anomaly detection, this can be leveraged to compare one method against the others.

3.3 Requirements

The process of generating a suitable data set takes a long time, due to the importance of the data it contains in verifying the data intrusion detection and evaluation systems. The generation of the dataset must fulfill the following requirements:

1. Focus on securing properly and accurately labeled datasets for each connection record within the traffic.
2. The data set generated should be as unbiased as possible in terms of size for each of the traffic states natural and offensive traffic. However, most of the existing reference datasets have a percentage of traffic normal and offensive are not the same, since normal traffic is more common than anomalous traffic. Most of the existing following hypotheses:
    - Anomalous traffic is statistically different from normal traffic.
    - Most cases of network traffic are normal
3. As attacks increase in size, variety, and sophistication, sneak threats become more sophisticated, including choosing target applications and services. It is necessary to consider different attack scenarios while generating the dataset, with the aim of including a variety of attacks.

3.4 Datasets

A set of good quality data makes it possible to determine the ability of a method or system to detect anomalous behavior, especially when applying systems in real operating environments. There are several data sets as shown in Figure. In order to test and evaluate systems and methods for detecting retinal anomalies, they fall into three categories as follows:

3.4.1 Synthetic Datasets

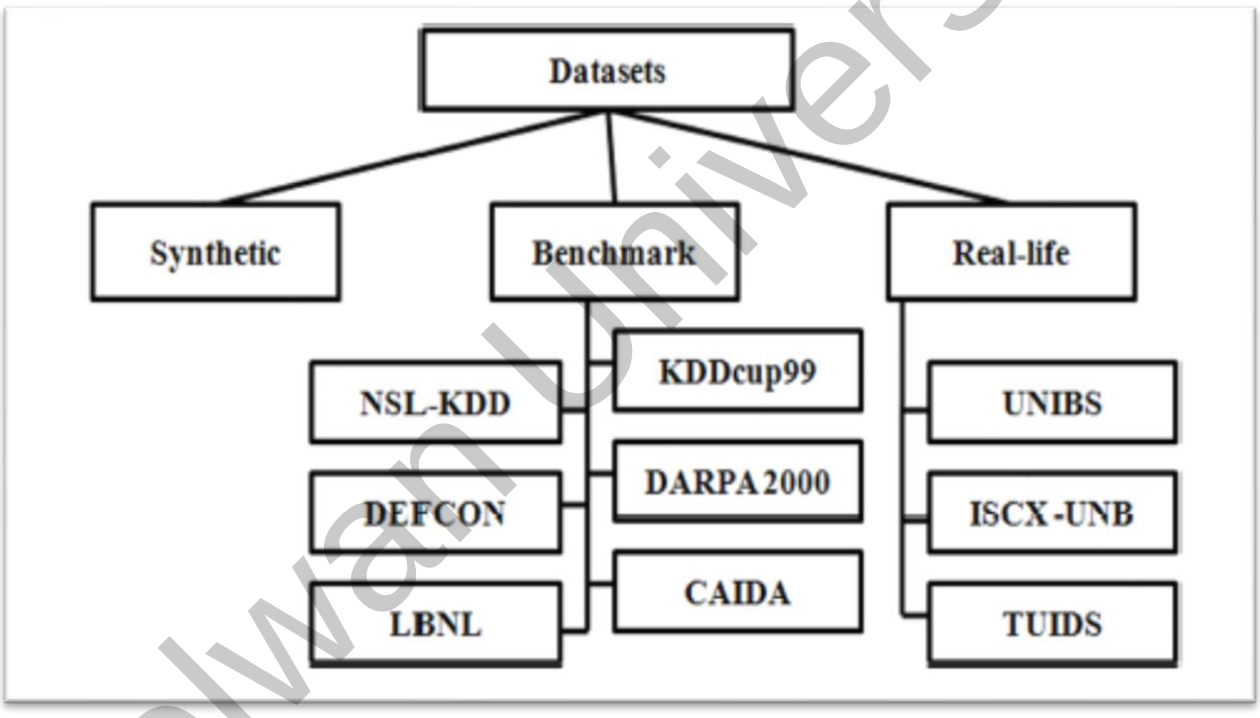


Figure 9: Types of datasets used in evaluating intrusion detection methods

Synthetic datasets are generated to meet specific needs or conditions. These datasets are useful when designing any system model for theoretical analysis so that design improvement can be worked on. can use synthetic data set to test and create many different types of test scenarios. This enables designers it builds real-world behavior profiles for ordinary users and attackers based on the dataset to test the system proposed.

3.4.2 Benchmark Datasets

The following are six standard combinations available that were generated using simulated environments in large networks simulate different attack scenarios while generating these datasets.

3.4.2.1 KDDcup99 dataset

Since 1999, it is still a data set KDDcup99 the most used group for evaluation methods and systems for detecting retinal anomalies. The dataset was prepared by Stolfo and others .Based on the data captured in the DARPA98 Intrusion Detection System Evaluation Program, a dataset is formed KDD training from approximately 4,900,000 single connection beams, each containing 41 attributes labeled as either normal or type. Specific attack. The test dataset contains about 300,000 samples with 24 training attack types, and 14 attack types. additional within the test data set only. Tracking attacks are of four main types: Denial of Service, Remote to Local, User to Root, and monitoring or verification probing.

The training set consists of seven weeks of data labeled as a result of a simulated natural traffic incorporating known attacks, which are available to all IDS developers. The same applies to the test group some attacks not included in the training set.

3.4.2.2 NSL-KDD dataset

It is based on the KDD data set, where it addressed some issues that may affect the evaluation process of detection methods anomalies, made up of records selected from the full KDD data set. The NSL-KDD dataset consists of two parts as in KDD, one data set for training and another for testing.

3.4.2.3 DARPA2000 dataset

The goal of the DARPA Assessment Project was to detect complex attacks involving multiple steps. It was completed simulation of two attack scenarios in the DARPA 2000 evaluation competition, under the names (LLDOS 1.0) and (LLDOS

2.0). These two scenarios across multiple networks to achieve diversity. Sessions are grouped into four attack phases: validation, penetration system by exploiting vulnerabilities, downloading DoS software for the compromised system, launching a DOS attack against a target. LLDOS 2.0 differs from LLDOS 1.0 in that the attacks are more stealthy and therefore more difficult to detect. As this dataset includes multi-stage attack scenarios, which are often used to evaluate coherence methods alert Correlation.

3.4.2.4 DEFCON dataset

The DEFCON data set is another set used to evaluate IDSs. Network traffic captured during a competition called Catch the Flag (CTF) in which the competing teams are divided into two groups: attackers and defenders. The traffic generated through a CTF is very different from CTF traffic on the real network because it contains only stealth traffic without any normal traffic. Given this limitation, the DEFCON dataset is only useful for evaluating alarm correlation techniques.

3.4.2.5 CAIDA dataset

The CAIDA dataset is very specific to specific attacks and events. Most of the traffic data is time-consuming be anonymous. The dataset contains the CAIDA DDOS attack 2007 full hours of anonymous traffic made up of DDoS attacks on Aug 4, 2007, which you try to consume a large amount of network resources when connecting to Internet services. Traffic only includes attacks on the victim and the responses by the victim in the form of an interval of 5 minutes between them. The designers removed the traffic as non-offensive as possible when creating the CAIDA DDOS 200 dataset.

3.4.2.6 LBNL dataset

This dataset suffers from masking and separating any traffic content information extracted from it to remove any information that could identify IPs. LBNL traffic is described in the following:

- Normal LBNL traffic: The dataset can be obtained from the Bakerley Lawrence national Laboratory at United States of America. Traffic within this group consists of inbound, outbound, and outbound data internally at the edge routers of LBNL. The main applications in internal and external traffic are by backup, network file and windows. Other services such as names, email, and web services were used internal hosts.
- Offensive LBNL Traffic: This dataset identifies offensive traffic through scan isolation within the total traffic. Scanning is determined by indicating hosts that have not been successfully verified for most of the 20 hosts, 16 of them were verified in ascending or descending order of IP. Malicious traffic is often made up of failed incoming TCP SYN requests.

3.4.3 Real-life dataset

Below are three real datasets that were generated by collecting network traffic for several consecutive days during a week or a month. Details include normal and offensive traffic in close proportions.

3.4.3.1 UNIBS dataset

The packet paths of this dataset were collected on a barefoot router of the Berisha university campus network in Italy, over three consecutive working days. The dataset includes traffic captured or aggregated and stored using 20 a workstation, each running the GT (Ground Truth) client. Those responsible for creating the dataset gather traffic is generated using tcpdump, which is a packet parsing software that allows the user to view packets TCP/IP and other packets that are transmitted or received through the network to which the computer is connected. Captured and stored traffic on a disk dedicated to the workstation connected with a router.

3.4.3.2 ISCX-UNB dataset

This dataset is based on the concept of profiles that include detailed descriptions of intrusions and samples short for applications, protocols, or network components. Real packet paths were analyzed to generate profiles for customers responsible for

generating real traffic for. FTP, POP3, IMAP, SSH, SMTP, HTTP, and search in different scenarios of attacks with multiple stages with the aim of generating diversified traffic.

3.4.3.3 TUIDS dataset

The TUIDS dataset was prepared in the Network Security Laboratory of the University of Tezpur, India, based on several attack scenarios. At first the traffic was captured using gulp and nfdump, then it was processed and addressed traffic as attack or normal. The attributes were extracted and organized as basic attributes, content attributes, time, window, and irrelevant attributes from the pre-processed data, then correlate these attributes and generate clusters final data.

3.4.3.4 KU dataset

The Kyoto University dataset is a collection of traffic data collected obtained from a group of honeypots. This data consists of 24 statistical features, 14 of which were extracted. important ones. Dataset developers have extracted 10 additional attributes that can be used to investigate network events within the network university in a more efficient way. The 14 significant traits that were extracted and relied upon are similar to those in dataset KddCup99. Only these 14 attributes were used during training and testing.

3.5 KDD Cup99 dataset

This research task requires a diverse set of data on the ground, and the effectiveness of the experiments and the accuracy of their results depend on the quality of data attributes for a specific domain. For example: image processing, website analysis internet, medical applications, remote sensing, etc. All of the above have a standard data set for analysis. Likewise, most computer network intrusion detection systems rely on the standard KDD Cup99 dataset for classification analysis in network traffic and testing proposed new methods and systems for intrusion detection. This research relied on data for this normative group. The KDD data set is characterized by:

- Availability Since there are many anonymised datasets, this is a potential security risk to the organization. Hence, it is not accessible to researchers such as DARPA, CAIDA, and others.
- Detailed traffic attributes that are lacking in other datasets such as LBNL. It is worth noting that the study of traffic characteristics is one of the pillars of this research.
- Used by many researchers in the field of intrusion detection compared to other datasets provides the opportunity to follow up and compare research results.
- In its inclusiveness, a variety of offensive scenarios were taken into consideration during the creation of the KDD commensurate with the requirements of the research, for example, this group includes a variety of denial-of-service attacks that are considered the most dangerous attacks suffered by wireless networks.

3.5.1 KDD CUP Data

ACM collaborated with the Special Interest Group on knowledge disclosure and data mining SIGKDD or KDD which is the most popular professional organization for data miners. in response to this, an annual competition for data mining and knowledge disclosure in various fields was organized, which was called KDD.

the year	Area of interest
KDD-CUP 1997	Direct marketing to improve the lift curve
KDD-CUP 1998	Direct marketing to improve profits
KDD-CUP 1999	Network intrusion detection
KDD-CUP 2000	Click analysis on online retail websites
KDD-CUP 2001	Molecular bioactivity and spatial prediction of a protein
KDD-CUP 2002	Biomedical documentation and genetic role classification
KDD-CUP 2003	Network mining and usage history analysis

KDD-CUP 2004	Particle physics, plus protein isomerism prediction
KDD-CUP 2005	Internet user search query classification
KDD-CUP 2006	Detection of pulmonary embolisms from image data
KDD-CUP 2007	consumer recommendations
KDD-CUP 2008	breast cancer
KDD-CUP 2009	Get fast result in big database
KDD-CUP 2010	Evaluation of student performance
KDD-CUP 2011	Anticipate music ratings and identify favorite songs

Table 4: The areas that KDD Cup focused on

3.5.2 Description of the KDD-CUP-99 dataset

Sponsored by the US Defense Advanced Research Project Agency (DARPA) and a research laboratory the Air Force, MIT's Lincoln Laboratory, generated standard network traffic data to evaluate networked intrusion detection systems. These efforts were conducted between 1998 and 1999. The aim was to review and evaluate research activities in the field of intrusion detection.

The KDDCUP99 data is the data set consisting of a pre-prepared version of the assessment data DARPA in 1998, which was used in the third international competition for knowledge detection and data mining, and was held in conjunction with KDD99, the Fifth International Conference on Cognitive Detection and Data Mining. The task of the competition was to build a detector network infiltration, a predictive model that can distinguish between "bad" communication records called intrusions or attacks, and "good" which are natural contact records. This dataset contains a standard set of data that includes a variety of simulated intrusions within a military network environment.

Data mining was used as a "preparatory" stage to extract the characteristics of infiltration traits from audit data raw TCP/IP. The training data was about 4GB of

compressed tcpdump binary data obtained from MIT's network traffic within the first seven weeks. Pretreatment was done using the feature architecture MADAM ID (Data Audit and Mining Inspection for Automated Intrusion Detection Models) is a set of data mining algorithms developed specifically for handling network intrusions and datasets check records, resulting in about five million contact records.

A connection is defined as a series of TCP packets that begin and end at a certain length of time during which data flows from and to a source IP address to a target IP address under some well defined protocol. Also, each connection is marked as either normal or Attack with the attack type it belongs to. Each connection record is about 100 bytes long.

Ten percent of the test data collected over a two-week period were pre-processed to obtain a preparatory study what is less than half a million contact records. It is worth noting that the test data is not from the same probability distribution as the training data, as it includes attack types not found in the training data. The total amount of test data will not be included in this dataset.

In 1999, KDD recognized and approved the DARPA data to be a traditional benchmark for a data set. It is used for intrusion detection systems (IDS) and is named.

3.5.3 Contact record features

The KDDCUP99 data includes the primitive attributes extracted from the connection register called attributes the basic features of a single TCP connection, such as: duration, protocol type, number of bytes transmitted, and the flag indicates the normal state or error state of the connection. These attributes provide information for network traffic analysis purposes public. In addition to the “same host” attributes, which only scan connections per second. The former two that have the same destination host as the current connection, and compute statistics related to the behavior of the protocol, service, etc. and similar “same service” features that only scan connections in the two seconds preceding them same service as an existing connection. The "same host" and "same service" attributes are collectively called time-dependent attributes. From contact records.

Some verification attacks scan hosts (or ports) using intervals greater than 2 seconds, for example in a minute. Therefore, connection records were stored by the

destination host, and attributes were generated using window for 100 connections to the same host instead of the time window. This resulted in a set of traits called dependent traits. Host-based Features.

On the other hand, unlike DOS attacks and Probe verification, there don't seem to be any serial patterns (repeated) in logs of remote attacks to R2L resolvers and user attacks to the U2R root. This is due to attacks DoS and verification involves many connections to the same host(s) in a short period of time, but U2R attacks the R2L is contained in the data portions of the packets, and usually includes a single connection. So, domain knowledge was used to add attributes that look for suspicious behavior in pieces of data, such as the number of failed login attempts. This is called content attributes.

In general, there are 42 attributes (including attack type) in each connection record since most of them have continuous values.

NO	Feature Name	Description
1	Duration	The number of seconds to connect
2	Protocol_type	Protocol type, eg UDP,TCP,....
3	Service	network service at the destination, eg telnet,http,…
4	Flag	connection status
5	Src_bytes	The number of bytes of data from the source to the destination.
6	Dst_bytes	The number of bytes of data from the destination to the source
7	Land	1: Connect to/from the same host/port. 0: otherwise
8	Wrong_fragment	The number of "wrong" hashes
9	Urgent	Number of emergency (urgent) packages
10	Hot	Number of entries to system directories, creating and executing programs
11	Num_failed_logins	The number of failed login attempts
12	Loggrd_in	1: Logged in successfully. 0: otherwise

13	Num_compromised	Number of "Compromised" Conditions
14	Root_shell	1: The root shell is obtained. 0: otherwise
15	Su_attempted	1: The command 'su root' was attempted. 0: otherwise
16	Num_root	The number of "root" accesses.
17	Num_file_creations	The number of file creations
18	Num_shells	The number of "shell" claims.
19	Num_access_files	The number of writes, deletes, and creations to access control files
20	Num_outbound_cmds	The number of commands issued in the FTP session
21	Is_hot_login	1: Login belongs to the ('hot') list. 0: otherwise
22	Is_guest_login	1: Login is ‘guest’. 0: otherwise
23	Count	The number of connections to the same host as the current connection in the last 2 seconds
24	Srv_count	The number of connections to the same service where the current connection is in the last two seconds.
25	Serror_rate	Percentage of connections that have a 'SYN' error to the same host
26	Srv_serror_rate	The percentage of connections that have a 'SYN' error to the same service
27	Rerror_rate	Percentage of connections that have a 'REJ' error to the same host
28	Sev_rerror_rate	The percentage of connections that have a 'REJ' error to the same service
29	Same_srv_rate	The percentage of connections to the same service and the same host
30	Did_srv_rate	The percentage of connections to different services and the same host

31	Srv_diff_host_rate	The percentage of connections to the same service and different hosts
32	Dst_host_count	The number of connections to the same host to the destination host where the connection is current in the last two seconds
33	Dst_host_srv_count	The number of connections from the same service to the destination host where the connection is current in the last two seconds
34	Dst_host_same_srv_rate	The percentage of connections from the same service to the destination host
35	Dst_host_diff_srv_rate	The percentage of connections from different services to the destination host
36	Dst_host_sane_src_port_rate	The percentage of connections from the services port to the destination host
37	Dst_host_srv_diff_host_rate	The percentage of connections from different hosts of the same service to destination host
38	Dst_host_serror_rate	Percentage of connections that have a 'SYN' error from the same host to the destination host
39	Dst_host_srv_serror_rate	Percentage of connections that have a 'SYN' error from the same service to destination host
40	Dst_host_rerror_rate	Percentage of connections that have a 'REJ' error from the same host to destination host
41	Dst_host_srv_rerror_rate	Percentage of connections that have a 'REJ' error from the same service to destination host

Table 5: Description of the Feature of the KDD99 dataset

The type of attribute values varies between numeric, binary, and literal values, as shown in the following table:

Type	Feature
Categorical	Protocol_type(2), Service(3), Flag(4)
Binary	Land(7), logged_in(12), root_shell(14), su_attempted(15), is_host_login(21), is_guest_login(22)

digital	Duration(1), src_bytes(5), dst_bytes(6), wrong_fragment(8), urgent(9), hot(10), num_failed_logins(11), num_compromised(13), num_root(16), num_file_creations(17), num_shells(18), num_access_files(19), num_outbound_cmds(20), count(23) srv_count(24), serror_rate(25), srv_serror_rate(26), rerror_rate(27), srv_rerror_rate(28), same_srv_rate(29) diff_srv_rate(30), srv_diff_host_rate(31), dst_host_count(32), dst_host_srv_count(33), dst_host_same_srv_rate(34), dst_host_diff_srv_rate(35), dst_host_same_src_port_rate(36), dst_host_srv_diff_host_rate(37), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41)

Table 6: The type of KDD99 dataset feature values.

According to Table above, KDD99 has three characters with Categorical values, namely, Protocol_type(2), Service(3), Flag(4), For the “Protocol_type” feature, there are three different values representing the protocols. Considered in KDDCup99 are ICMP, UDP, TCP. while the "Flag" feature has 11 different values. Whereas the "66" Service feature has a value.

The "flag" feature indicates the handshaking process used by the protocol to ensure that the connection between the two is reliable users after verifying communication between them. Next table shows the description of the different “Flag” attribute values.

Flag	Description
S0	Connection attempt seen, no response
REJ	The connection attempt was rejected
S1	Connection achieved, but not terminated
SF	Normal creation and termination
OTH	No SYN seen, only halfway traffic ("partial connection" not closed anymore)
S2	Connection achieved and attempt to close was seen by originator (but no response from responder)
RSTO	Connection achieved; originator aborted (RST Sent).
S3	Connection has been achieved and the connection attempt has been seen by the respondent (but no response has been received before originator)
RSTR	Investigator, respondent abort
RSTOS0	The originator sent a SYN followed by RST, the SYN ACK was not seen by the respondent
SH	Creator send SYN followed by 'FIN' (finish 'flag', SYN ACK has not been seen before

	Respondent (therefore the connection was half open)

Table 7: Description of the values of the flag feature

For the service attribute, it has 66 values denoting the network service at the destination, and each service is assigned to one port used in TCP to name the ends of logical connections that carry long running conversations

3.5.4 KDDCup99 dataset attacks

In its broadest definition, a computer attack is any malicious activity directed at a computer system or the services it provides. Examples of computer attacks are viruses, use of the system by any unauthorized person, denial of service by exploitation of a software bug or abuse of a feature, verification of a system to collect information, or a physical attack on devices the computer. A subset of computer attacks is included in DARPA's Intrusion Detection Evaluation Program including attacks that allow the hacker to work on the system and enjoy more privileges than the security policy allows to the system, attacks that prevent another person from accessing some of the services provided by the system or attempts to verify into the system with the aim of finding potential vulnerabilities.

23 attack types are included in the training data and 15 new attacks are added to the test data, to test IDS performance on known and unknown attacks. Every kind of attack falls into one of the four main categories which are. Probe, R2L, U2R, DOS.

Class	attacks in the training dataset	attacks in the test dataset
DOS	Back, land, Neptune, pod, smurf, teardrop	apache2, mailbomb, processtable, udpstorm
Probe	mscan, saint	ipsweep, portsweep, satan, nmap
R2L	ftp_write, guess_passwd, imap, multihop, warezmaster, warezclient, spy, phf	httptunnel, worm, named, sendmail, xlock, xsnoop, snmpgetattak, snmpguess
U2R	buffer_overflow, loadmodule, rootkit, perl	ps, xterm, sqlattack

Table 8: Attacks found in both the training and test data sets

As shown in the table above, there are four invasive classes within KDD99 for each of the classes the four are a variety of attacks, and other different types have been

added to the test data set to verify the effectiveness of the proposed intrusion detection system.

3.5.4.1 Denial of Service (DOS) Attacks

A denial-of-service attack is an attack in which an attacker makes computing resources or memory so busy or full is unable to respond to legitimate requests, or as a result denies users access to the Device. There is a variety of extensive DoS attacks. Some DoS attacks (such as Mailbomb, Neptune, Smurf) abuse a completely legitimate feature, other DoS attacks (Teardrop, Ping of Death) generate corrupted packets. Obfuscation of the TCP/IP stack of a device trying to rebuild the packet. Other attacks such as (Apache2, Back, Syslogd) Errors in a certain hidden network daemon (a program running in background without being under the control of the user.) Each type of DoS attack is explained in detail below DoS included in KDD.

3.5.4.2 Remote to Local (R2L)

These attacks occur when an attacker with the ability to send packets to a device over the network (But does not have an account on the device) by exploiting some vulnerabilities to gain local access as a user on that device. There are many ways an attacker can gain unauthorized access to a local account on a device. Some attacks exploit buffer overflows in web server software (imap, named, sendmail) Xsnoop, Guest, Ftp-Write, and Directory attacks attempt to exploit weak or incomplete security policies. correct for the system. In Xlock attack, in order for the attack to succeed, the attacker must trick users into handing over passwords their password to the screensaver is actually a Trojan.

3.5.4.3 User to Root Attacks (U2R)

It is a class of attack in which an attacker initiates access to a normal user account on the system (maybe earned by password sniffing, dictionary attack, or social engineering) and able to exploit certain vulnerabilities to gain root access to the system.

There are several different types of U2R attacks. The most common type is the buffer attack, overflow attack, and the load module that exploits programs that make assumptions about the environment in which they are running. Exploit attacks other U2R programs that don't care how you manage temporary files. Finally, some vulnerabilities are present because an exploitable conflict exists in

the actions of a single program, or of two or more programs that are executing concurrently. Bugs like these are present in every major version of UNIX available today.

3.5.4.4 Probes

In recent years, a number of programs have been distributed that can scan a network of computers with the aim of collecting data information or finding known vulnerabilities. These network checks are very useful for any attacker planning a future attack. An attacker with a map of the devices and services available on the network could use this information to search for vulnerabilities. some of these scanning tools (satan, saint, mscan) enable any inexperienced attacker to quickly check hundreds or even thousands of devices on the network searching for known vulnerabilities. Below are the details of each Probe attack it was used in KDD.

4. System Design and Architecture

In this chapter, there is a full description of each module design and architecture in our project. First, an overview and any needed assumptions we used will be explained. Then the overall system architecture and block diagram. Then for each module, there is a functional description, modular decomposition, design constraints, and any other needed descriptions. In addition to the decisions, we took about the modules' functionalities.

During the development and implementation of our project, we made sure the project code is modular and clear enough to be understood, in case of any future need for the code itself.

The overall system could be broken down to mainly **11** components:

- **Attacks Detection:** By using AI, System tries to detect many common attacks via analyzing each packet.
- **Arp Poisoning Detection:** monitoring the Arp table within seconds checking if it has been poisoned.
- **Privilege Escalation Detection:** monitoring and detecting some of privileges escalations attacks that could give a malicious user a high privilege.
- **Passwd File Monitor**: monitoring the passwd file to check any changes and alert you with the changes that happens in the file compared to the old one which could be a malicious change.
- **Tmp Directory Monitor:** Monitoring the Tmp directory to check any malicious files.
- **History File Checker:** monitoring the history file for a given user and check for commands that could be malicious commands that leak passwords etc.
- **Port Blocker:** using iptables to block ports such as unused ports or ports that can be used in malicious traffic.

- **Services Monitor:** monitoring network services so you can check listening services, connected services and more.

- **Devices on Network Detection:** scanning all devices on the network to check if any malicious devices.
- **Malicious Files Detection:** scanning any malicious files or malware in the system.
- **Unencrypted protocols Detection:** detecting any unencrypted protocols.

4.1 Overview and Assumptions

Smart IDS project consists of 11 main modules that need to be completely understood before starting to implement such a thing. Each of these modules is described in detail in the following section, and how they all connect to each other, and how they represent the system architecture.

Some assumptions are also considered in delivering this project are:

4.1.1 Accuracy

Since the security sector has difficult problems to deal with such as determining attacks and the source of them that is different from environment to other, and from age to age. We will use hybrid technology to introduce good accuracy in recognizing attacks and malicious activities in whatever style happens.

4.1.2 Speed

The system will try to recognize attacks that have a lot of styles and features in a reasonable time.

4.1.3 Friendly Interface

The system is developed Linux web application with a friendly interface that gives the user the accessibility to have an account and ability to start IDS. There are test cases for each feature and module of the system with the ability to trend feature to specific directories or files.

Alerting signs are set up to warn the administrator whether if attack or malicious activity happens.

4.2 System Architecture

The system's main modules can be partitioned into Attack Detection, Vulnerabilities Scan, System Monitor, Fast Response and Malware Analysis.

- **Attack Detection** represents in features: common attacks detection using AI, Arp poisoning detection and Privilege Escalation Detection.
- **Vulnerabilities Scan** represents in feature: Unencrypted protocols Detection.
- **System Monitor** represents in feature: Passwd File Monitor, Tmp Directory Monitor, History File Checker, Services Monitor and Devices on Network Detection.
- **Fast Response** represents in features: Port Blocker.
- **Malware Analysis** represents in features: Malicious Files Detection.

4.3 Block Diagram

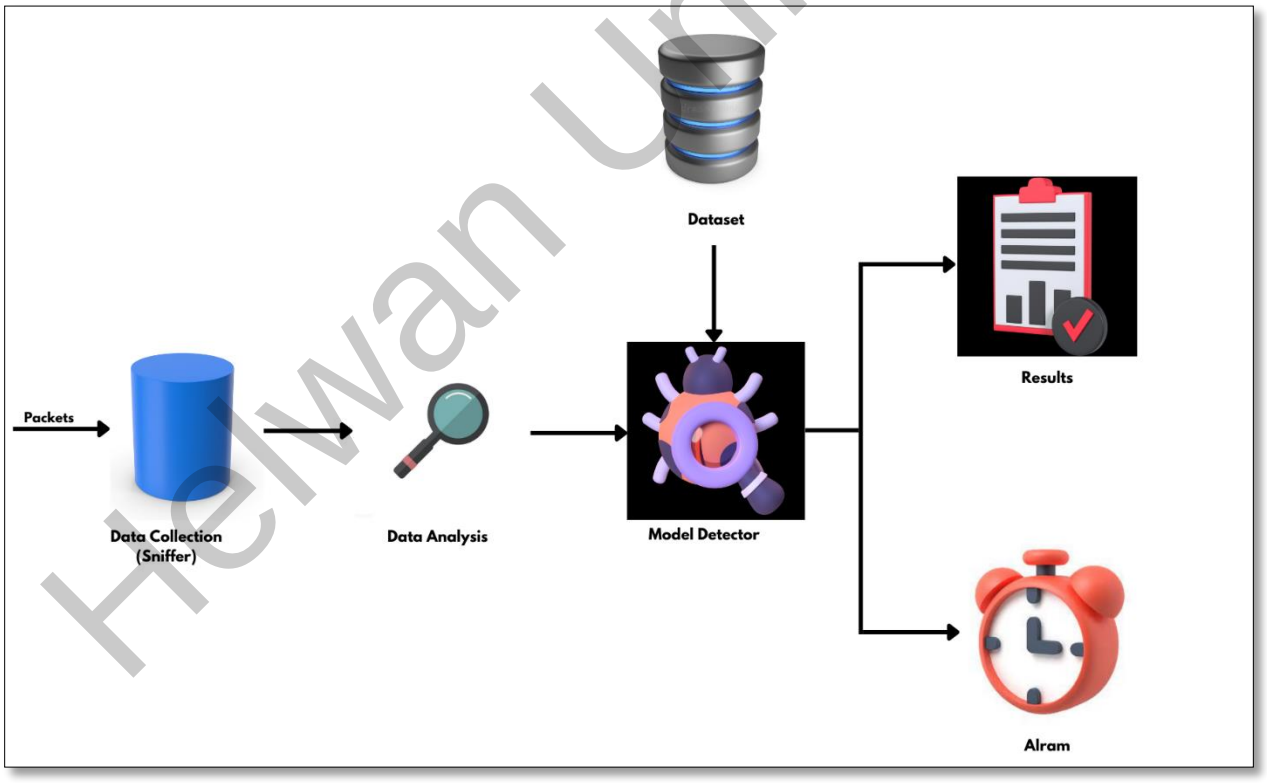


Figure 10: Block Diagram

4.4 Sniffer

A sniffer is a tool or software program used to capture and analyze network traffic. It is also known as a network analyzer or packet analyzer. A sniffer works by intercepting and analyzing network traffic as it flows through a network interface or a specific network segment. It captures the packets of data that are sent between devices on a network, and then decodes and analyzes the contents of those packets to provide insights into network activity and behavior.

Sniffers can be used for a variety of purposes, including network troubleshooting, security auditing, and performance analysis. They can also be used maliciously to capture sensitive data such as passwords, credit card numbers, and other confidential information. There are many different types of sniffers, ranging from basic tools that capture and display network packets to more advanced tools that can perform deep packet inspection, protocol analysis, and even intrusion detection. Sniffers can be either hardware or software-based, and they can be run on a variety of platforms, including Windows, Linux, and macOS.

4.4.1 Capturing Packets

When a sniffer captures packets, it intercepts and analyzes the traffic flowing through a network interface or segment. The packets can contain various types of data, such as HTTP requests, email messages, instant messages, and file transfers. The captured packets are usually displayed in a list format, with each packet represented by a row in the list. The list can show various details about each packet, such as the source and destination IP addresses, the protocol used, the packet size, and the timestamp.

In addition to the list view, sniffers also provide a detailed packet analysis view that shows the contents of each packet in a hierarchical format. This view can be used to inspect the individual fields and values of each packet and can be particularly useful for troubleshooting network issues or analyzing network

behavior. Sniffers can be used for a variety of purposes, including network troubleshooting, security auditing, and performance analysis. They can also be used maliciously to capture sensitive data such as passwords, credit card numbers, and other confidential information. Therefore, it is important to use sniffers responsibly and with proper authorization.

4.4.2 Features Extraction

KDDCUP99 is a well-known dataset in the field of intrusion detection, and it contains network traffic data captured from a simulated environment. The data has been preprocessed and labeled with different types of attacks and normal traffic. To extract features from the network traffic data in the KDDCUP99 dataset using a sniffer, we use the following steps:

1. Capture the network traffic data using a sniffer tool such as Wireshark or tcpdump.
2. Filter the captured data to extract only the relevant packets for the analysis. For example, you may filter by protocol type, source/destination IP address, or port number.
3. Extract the features of interest from the filtered packets. Some common features used in intrusion detection include packet size, duration, source/destination IP addresses and port numbers, protocol type, and number of packets sent/received.

Here are the names of 41 features and sample of values of them that were captured by the sniffer:

```
[['timestamp', 'src_ip', 'src_port', 'dst_ip', 'dst_port', 'index', 'idx', 'duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'la
nd', 'wrong_fragment', 'urgent', 'hot', 'num_failed_logins', 'logged_in', 'lnum_compromised', 'logged_in', 'num_compromised', 'root_shell', 'su_attempte
d', 'num_root', 'num_file_creations', 'num_access_files', 'num_outbound_cmds', 'is_hot_login', 'is_guest_login', 'count', 'srv_count', 'same_srv_rate',
'diff_srv_rate', 'long_same_src_port_rate', 'srv_long_diff_host_rate', 'serror_rate', 'srv_serror_rate', 'long_rerror_rate', 'srv_long_rerror_rate'], ['
1682963255.657608239', '192.168.140.132', 40439, '192.168.140.2', 53, 192168140002, 1, 0, 'udp', 'Unassigned', 'SF', 75, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1.0, 0, 1.0, 0, 0.0, 0.0, 0.0, 0.0]]
```

Figure 11: Names of 41 features and sample of values of them

4.5 Features Preparation

Preprocessing and cleaning the extracted features are an essential step in preparing the data for analysis. Here are used techniques for preprocessing and cleaning the features:

Removing duplicates: If there are any duplicate packets in the dataset, it is important to remove them to avoid biasing the analysis.

Handling missing data: If any packets are missing some of the features of interest, you may choose to either remove those packets from the dataset or fill in the missing values using techniques such as mean imputation or interpolation.

Normalizing or scaling the data: To ensure that the features are on a consistent scale, you may choose to normalize or scale the data. Normalization involves scaling the features so that they have a mean of zero and a standard deviation of one, while scaling involves rescaling the features to a specified range.

Removing outliers: Outliers can skew the analysis, so it may be helpful to remove any packets that are significantly different from the rest of the data.

Encoding categorical features: If any of the features are categorical, they will need to be encoded in a numerical format so that they can be used in machine learning algorithms. Common encoding techniques include one-hot encoding and label encoding.

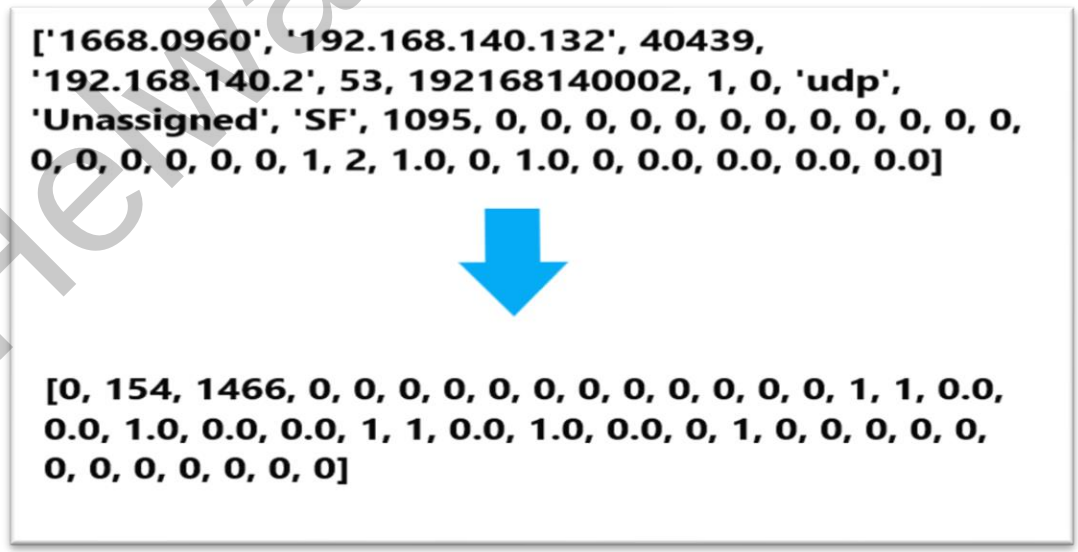


Figure 12: Encoding categorical features

4.6 Model Development

Neural network technology was used to build a network intrusion detection system classifier based on detection anomalies, based on the standard KDD99 dataset. Google Colab software was used to conduct experiments and evaluate the performance of the classifier after each experiment and compare with the aim of reaching the optimal structure of the network classifier system neuron. The implemented system can be represented by the block diagram shown in the next Figure.

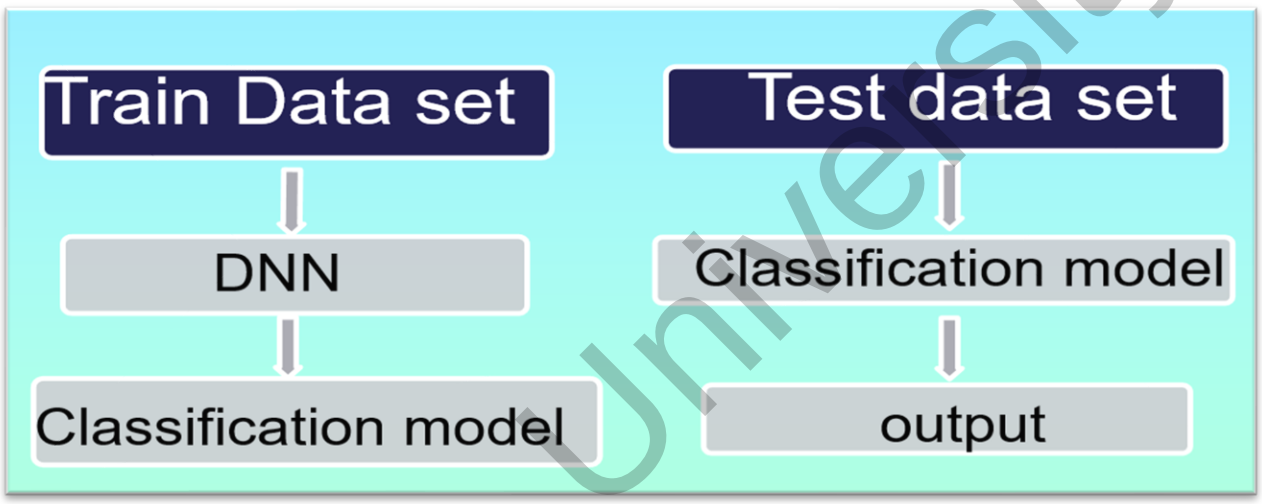


Figure 13: System diagram

The figure on the left shows the stages of designing the system classifier using the processed training dataset, which will be executed it is then evaluated as shown in the figure on the right using the test dataset that will be classified as offensive log or natural. This chapter includes an explanation of the tools and methods that were relied upon to complete the research and their location within the research.

4.6.1 Training dataset

The “10% KDD” dataset is used to train different intrusion detection systems. It is a copy abbreviated from the full KDD dataset, comprising 494,021 contact records. This group contains 22 attack types. examples of offensive communication logs include more than normal communication logs, noting the types of attacks it is not equally represented as shown in the table and the denial of service attack accounts for the majority of connection logs in the dataset.

Class	No. of samples	Type
Normal	97,277	Normal
DOS	280,790	Smurf
DOS	107,201	Neptune
DOS	2,203	Bac
DOS	979	Teardrop
DOS	264	Pod
DOS	21	Land
Probe	1,589	Satan
Probe	1,247	Ipsweep
Probe	1040	Portsweep
Probe	231	Nmap
R2L	1,020	Warezclient
R2L	53	guess_passwd
R2L	20	Warezmaster
R2L	12	Imap
R2L	8	Ftp_write
R2L	7	Multihop
R2L	4	Phf
R2L	2	Spy
U2R	30	Buffer_overflow
U2R	10	Rootkit
U2R	9	Loadmodule
U2R	3	Perl

Table 9: The number of samples in detail in the training dataset

4.6.2 Test dataset

The dataset “Corrected KDD” is used for testing purposes. This group provides distributed data statistically different compared to the training dataset, whether it is “10% Partial KDD” or “Full KDD”, the “311029 Corrected KDD” dataset includes a contact record labeled Normal or one of the four types of attacks. Contact record addresses are used to verify the classification predictions obtained them during testing.

Class	The whole KDD		10%KDD		Corrected KDD	
	samples	percentage	samples	percentage	samples	percentage
Normal	972781	19.8590%	97278	19.6911%	60593	19.4815%
DOS	3883366	79.2778%	391458	79.2391%	229853	73.9008%
Prope	41102	0.8391%	4107	0.8313%	4166	1.3394%
R2L	1126	0.0230%	1126	0.2279%	16347	5.2558%
U2R	52	0.0011%	52	0.0105%	70	0.0225%
total attacks	3925646	80.1409%	396743	80.3089%	250436	80.5185%
total records	4898427	100%	494021	100%	311029	100%

Table 10: The number of samples in both the full and train dataset and test

The previous table shows a comparison between the complete data set “KDD99” and the partial “10%KDD” and the data set test data “Corrected KDD” used in the practical testing phase to verify the performance of the proposed system, this is in terms of the number of natural and intrusive communication records contained within each of the previous datasets.

4.6.3 Neural network classifier

There are many reasons for using neural networks as an analytical engine in an intrusion detection system:

- Its generalization ability makes it suitable for attacking well-known targets. Unknown attacks. The neural network is able to handle uncertain, inaccurate, and valid data partially, therefore, they are able to recognize patterns not presented during the learning phase, which makes them attractive for application in intrusion detection.
- Characteristics of artificial intelligence represented by its adaptive ability and self-learning. Neural networks are an adaptive system it changes its structure according to external or internal information flowing through the network during the learning phase. Implicitly reveal non-relationships complex linearity between independent and dependent variables, can deal with data confusion, and provides many learning Algorithms.
- It is a powerful tool for multiclass classification.

- One of the branches of elastic computing used in solving NP-complete problems, such as searching for Infiltration and its detection.
- The speed of neural networks. Protecting computing resources requires rapid identification of attacks, and therefore speed remediation is an essential step to respond to intrusions before any damage is done to the system.
- Effective intrusion detection is a difficult goal to achieve by system administrators and researchers in information security, given the complexity of attacking computer systems, the variety of potential vulnerabilities, and the skill of attackers creates a difficult problem selection.
- On the other hand, neural networks have become widely used and have proven successful with regard to complex problems such as pattern recognition, where the neural network classifier is a universal classifier with the appropriate choice of its structure can solve any problem, even very complex classification tasks.

Feed-forward neural networks are considered one of the most important modern methods that have high efficiency in giving satisfactory and good results in recognizing patterns, in which the direction of the signals entering the network is always forward the outgoing signal from any neuron depends only on the incoming signals, and neural networks need feeding the front indicates that there are two pairs of vectors, the input vector, and the desired output vector. An important part of building a neural network is the use of an accurate and powerful algorithm in learning, where the backpropagation algorithm is one of the most important algorithms used in training neural networks.

Training the network by backpropagation involves three phases: feed-forward input training models, calculation and associated error back propagation, weight adjustment. Input and output are available in training, where they are processed inputs and compare the resulting output with the desired output, and on the basis of that calculate the errors and propagate them back through the system, with the aim of the system adjusting the weights that control the network. This process is repeated and with it the weights are adjusted continuously. The data set that enables training is called the training set. While training a network, the TFs are processed set the data several times, until the weights are adjusted appropriately

after training, includes network application only feed-forward stage calculations. Even if the training is slow, the trained network can produce its output quickly big.

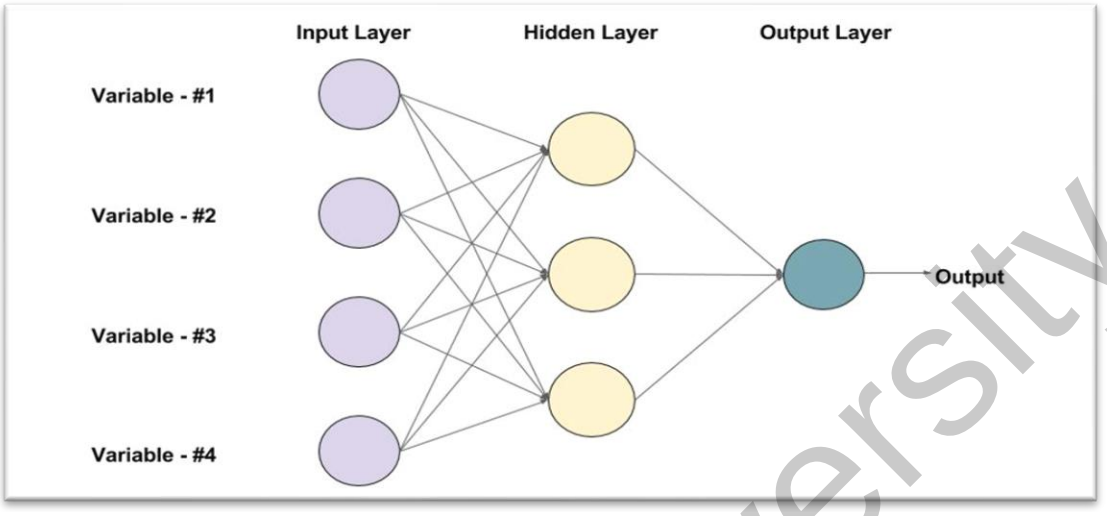


Figure 14: Feed-forward Neural Network with one hidden layer (with 3 neurons)

A single-layer network (with only an input layer and an output layer) is of limited use, in contrast, multilayer network layers (with one or more hidden layers) are more common. There can be more than one hidden layer useful for some applications, but a single hidden layer will suffice.

4.6.3.1 Proposed architecture of a neural network

The construction of a neural network depends on the number of layers, the number of neurons in each layer, the activation function, and the algorithm Training as it determines the final value of the weights. In this research, a multi-layered feed-forward neural network was chosen (Multilayer Feed Forward Neural Network) to construct the classifier on which the proposed system is based, and the dependency on the Back Propagation Algorithm to apply the feedback to adjust the weights during training.

As for the hidden layers and the number of neurons within them, they were chosen based on doing several experiments and depending on findings from studies using neural networks as classifiers. Thus, it represents the following structure of the network the proposed neuron structure with the necessary parameters for the training process:

- One hidden layer.

- The number of neurons in the first two layers is 50 neurons.
- The number of neurons in the output layer is 5 neurons.
- The activation function in the first two layers is relu function.
- The activation function in the output layer is softmax function:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                2100

 dense_1 (Dense)             (None, 50)                2550

 dense_2 (Dense)             (None, 5)                 255

=================================================================
Total params: 4,905
Trainable params: 4,905
Non-trainable params: 0
_________________________________________________________________
```

Figure 15: Structure of the network

An artificial neuron or neural node is a mathematical model. In most cases, it computes the weighted average of its input and then applies a bias to it. Post that, it passes this resultant term through an activation function. This activation function is a nonlinear function such as the sigmoid function that accepts a linear input and gives a nonlinear output. The following figure shows a typical artificial neuron:

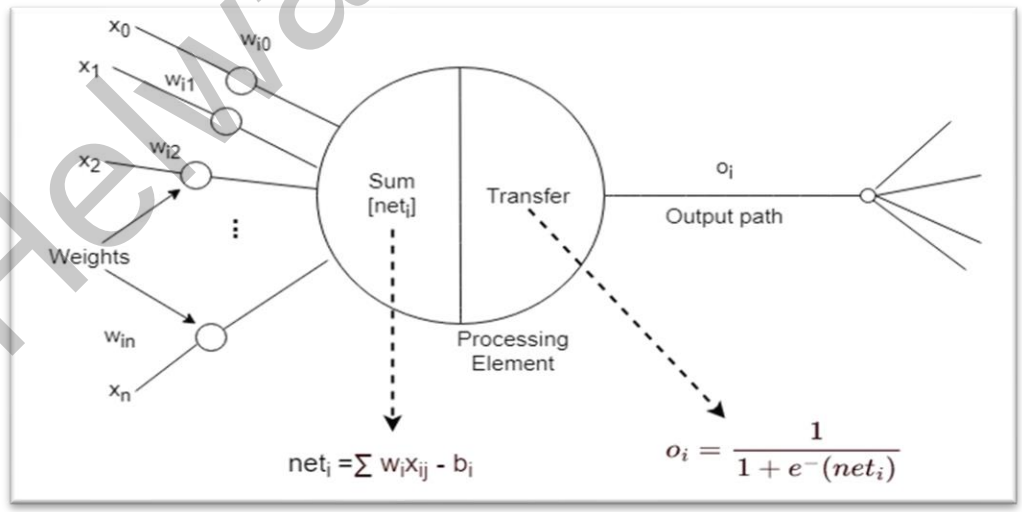


Figure 16: Artificial neuron

A typical neural network consists of layers of neurons called neural nodes. These layers are of the following three types:

1. input layer (single)
2. hidden layer (one or more than one)
3. output layer (single)

Each neural node is connected to another and is characterized by its weight and threshold. It gets an input on which it does some transformation and posts that, it sends an output. If the output of any individual node is above the specified threshold value, that node gets activated. Then, it sends data to the next layer of the network. Otherwise, it remains dormant and thus doesn't transmit any data to the next layer of the network.

The following figure marks all three types of layers:

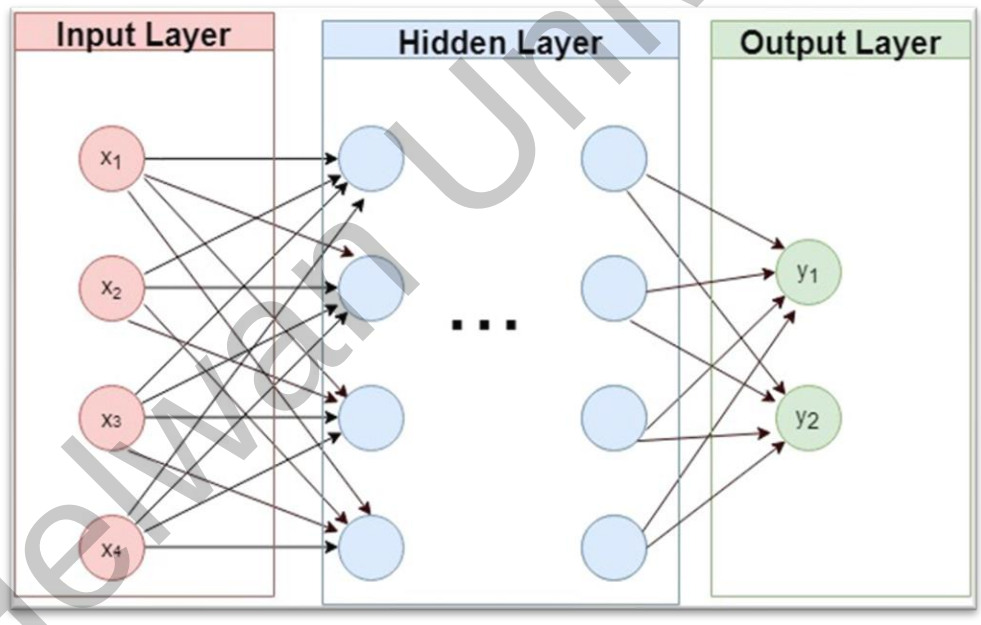


Figure 17: Three types of neural network layers

Neural networks consist of an input layer that receives and processes input data, a hidden layer that receives input from the input layer or previously hidden layers, and an output layer that produces the final result. The output layer's result can be a binary classification or multi-class classification and can be used as a predicted value or input for another neural network.

4.6.4 Action scenario and practical results

In this section, we will explain the practical steps that have been taken and explain the results in detail.

4.6.4.1 changing attack labels to their respective attack class

Both the training dataset and the test dataset consist of different types of attack, in this step we have converted the different attack types into only four classes, so that each type of attack is within its own class.

For the training data set, this function was used to convert all attack types to their four classes, which are: (Dos, Prope, R2L, U2R)

```
# changing attack labels to their respective attack class
def change_train_label(df):
  df.target.replace(['back.','land.','neptune.','pod.','smurf.','teardrop.'],'Dos',inplace=True)
  df.target.replace(['guess_passwd.','imap.','ftp_write.','multihop.','phf.','spy.','warezclient.','warezmaster.'],'R2L',inplace=True)
  df.target.replace(['ipsweep.','nmap.','portsweep.','satan.'],'Probe',inplace=True)
  df.target.replace(['buffer_overflow.','loadmodule.','perl.','rootkit.'],'U2R',inplace=True)
```

Figure 18: Function was used to convert all attack types to their four classes

After applying the above function, the data label was converted to contain the four types of attacks that we mentioned, instead of the different types of attacks, as follows:

```
smurf.              280790
neptune.            107201
normal.              97277
back.                 2203
satan.                1589
ipsweep.              1247
portsweep.            1040
warezclient.          1020
teardrop.              979
pod.                   264
nmap.                  231
guess_passwd.           53
buffer_overflow.        30
land.                   21
warezmaster.            20
imap.                   12
rootkit.                10
loadmodule.              9
ftp_write.               8
multihop.                7
phf.                     4
perl.                    3
spy.                     2
Name: target, dtype: int64
```

```
Dos        391458
normal.     97277
Probe        4107
R2L          1126
U2R            52
Name: target, dtype: int64
```

Figure 19: After applying the convert function

Figure 20: Before applying the convert function

The same function was done and changed the attack types depending on the types in the test dataset.

```
# changing attack labels to their respective attack class in test data
def change_test_label(df):
  df.target.replace(['back.','land.','neptune.','pod.','smurf.','teardrop.','apache2.', 'mailbomb.', 'processtable.','udpstorm.'],'Dos',inplace=True)
  df.target.replace(['guess_passwd.','imap.','ftp_write.','multihop.','phf.','spy.','warezclient.','warezmaster.','httptunnel.', 'worm.', 'named.', 'sendmail.','xlock.', 'xsnoop.', 'snmpgetattack.','snmpguess.'],'R2L',inplace=True)
  df.target.replace(['ipsweep.','nmap.','portsweep.','satan.','mscan.', 'saint.'],'Probe',inplace=True)
  df.target.replace(['buffer_overflow.','loadmodule.','perl.','rootkit.','ps.', 'xterm.', 'sqlattack.'],'U2R',inplace=True)
```

Figure 21: Function was used to convert all attack types to their four classes in the test dataset

```
smurf.              164091
normal.              60592
neptune.             58001
snmpgetattack.        7741
mailbomb.             5000
guess_passwd.         4367
snmpguess.            2406
satan.                1633
warezmaster.          1602
back.                 1098
mscan.                1053
apache2.               794
processtable.          759
saint.                 736
portsweep.             354
ipsweep.               306
httptunnel.            158
pod.                    87
nmap.                   84
buffer_overflow.        22
multihop.               18
named.                  17
sendmail.               17
ps.                     16
rootkit.                13
xterm.                  13
teardrop.               12
xlock.                   9
land.                    9
xsnoop.                  4
ftp_write.               3
loadmodule.              2
perl.                    2
udpstorm.                2
worm.                    2
phf.                     2
sqlattack.               2
imap.                    1
Name: target, dtype: int64
```

Figure 22: Before applying the convert function

```
Dos       229853
normal.    60592
R2L        16347
Probe       4166
U2R           70
Name: target, dtype: int64
```

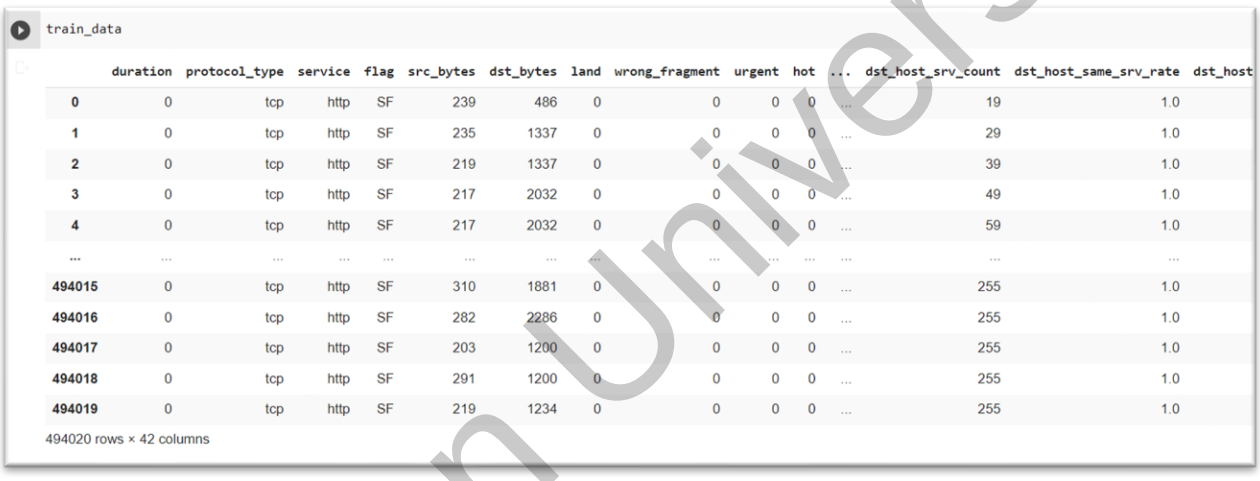
Figure 23: After applying the convert function

4.6.4.2 Data Pre-Processing

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

4.6.4.2.1 Drop Duplicate Rows

In this step all the duplicate rows were deleted, resulting in a significant drop in the number of records, in the training data set it decreased from 494020 to 145584, and in the test data set it decreased from 311028 to 77287.

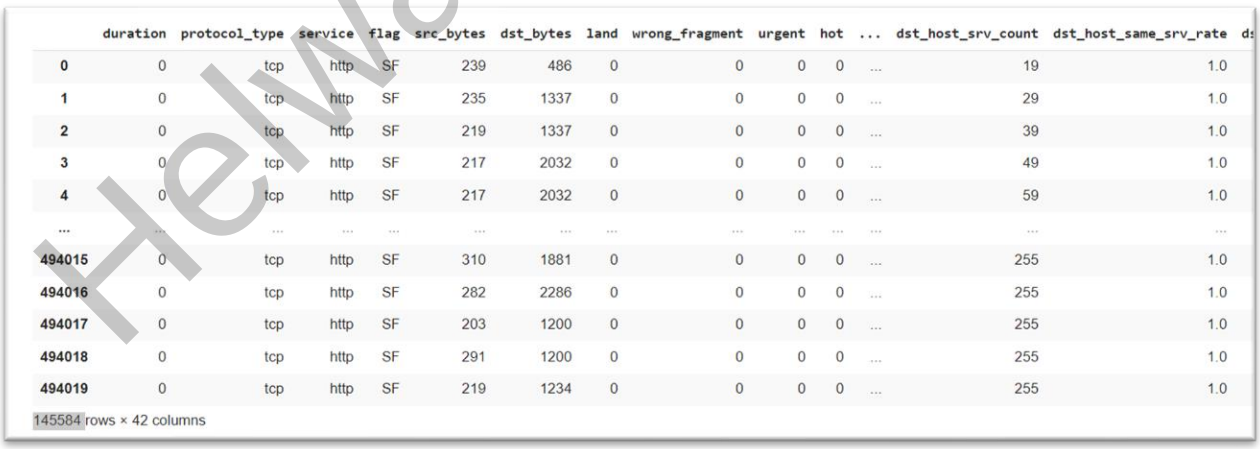


train_data

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host
0	0	tcp	http	SF	239	486	0	0	0	0	...	19	1.0	
1	0	tcp	http	SF	235	1337	0	0	0	0	...	29	1.0	
2	0	tcp	http	SF	219	1337	0	0	0	0	...	39	1.0	
3	0	tcp	http	SF	217	2032	0	0	0	0	...	49	1.0	
4	0	tcp	http	SF	217	2032	0	0	0	0	...	59	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
494015	0	tcp	http	SF	310	1881	0	0	0	0	...	255	1.0	
494016	0	tcp	http	SF	282	2286	0	0	0	0	...	255	1.0	
494017	0	tcp	http	SF	203	1200	0	0	0	0	...	255	1.0	
494018	0	tcp	http	SF	291	1200	0	0	0	0	...	255	1.0	
494019	0	tcp	http	SF	219	1234	0	0	0	0	...	255	1.0	

494020 rows × 42 columns

Figure 24: Training dataset before deleting duplicate rows



	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	ds
0	0	tcp	http	SF	239	486	0	0	0	0	...	19	1.0	
1	0	tcp	http	SF	235	1337	0	0	0	0	...	29	1.0	
2	0	tcp	http	SF	219	1337	0	0	0	0	...	39	1.0	
3	0	tcp	http	SF	217	2032	0	0	0	0	...	49	1.0	
4	0	tcp	http	SF	217	2032	0	0	0	0	...	59	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
494015	0	tcp	http	SF	310	1881	0	0	0	0	...	255	1.0	
494016	0	tcp	http	SF	282	2286	0	0	0	0	...	255	1.0	
494017	0	tcp	http	SF	203	1200	0	0	0	0	...	255	1.0	
494018	0	tcp	http	SF	291	1200	0	0	0	0	...	255	1.0	
494019	0	tcp	http	SF	219	1234	0	0	0	0	...	255	1.0	

145584 rows × 42 columns

Figure 25: Training dataset after deleting duplicate rows

```
[ ] test_data
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	ds
0	0	udp	private	SF	105	146	0	0	0	0	...	254	1.0	
1	0	udp	private	SF	105	146	0	0	0	0	...	254	1.0	
2	0	udp	private	SF	105	146	0	0	0	0	...	254	1.0	
3	0	udp	private	SF	105	146	0	0	0	0	...	254	1.0	
4	0	udp	private	SF	105	146	0	0	0	0	...	255	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
311023	0	udp	private	SF	105	147	0	0	0	0	...	255	1.0	
311024	0	udp	private	SF	105	147	0	0	0	0	...	255	1.0	
311025	0	udp	private	SF	105	147	0	0	0	0	...	255	1.0	
311026	0	udp	private	SF	105	147	0	0	0	0	...	255	1.0	
311027	0	udp	private	SF	105	147	0	0	0	0	...	255	1.0	

311028 rows × 42 columns

Figure 26: Test dataset before deleting duplicate rows

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_s
0	0	udp	private	SF	105	146	0	0	0	0	...	254	1.0	
2	0	udp	private	SF	105	146	0	0	0	0	...	254	1.0	
3	0	udp	private	SF	105	146	0	0	0	0	...	254	1.0	
4	0	udp	private	SF	105	146	0	0	0	0	...	255	1.0	
5	0	udp	domain_u	SF	29	0	0	0	0	0	...	3	0.3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
310329	0	icmp	ecr_i	SF	30	0	0	0	0	0	...	55	1.0	
310353	0	icmp	ecr_i	SF	30	0	0	0	0	0	...	56	1.0	
310403	0	udp	private	SF	105	105	0	0	0	0	...	254	1.0	
310666	0	icmp	ecr_i	SF	30	0	0	0	0	0	...	67	1.0	
310930	0	icmp	ecr_i	SF	30	0	0	0	0	0	...	76	1.0	

77287 rows × 42 columns

Figure 27: Test dataset after deleting duplicate rows

Removing duplicate values from your data set plays an important role in the cleansing process. Duplicate data takes up unnecessary storage space and slows down calculations at a minimum. At worst, duplicate data can skew analysis results and threaten the integrity of the data set.

4.6.4.2.2 Data Correlation

In this scenario, the values of the variances of the traits were calculated and these values were studied in search of the traits with zero variances. When the value of the variance is zero, this indicates that the value of the trait has not changed intended for all communication records, whether natural or intrusive, and therefore not contributing to the classification process, so it is possible to do without it.

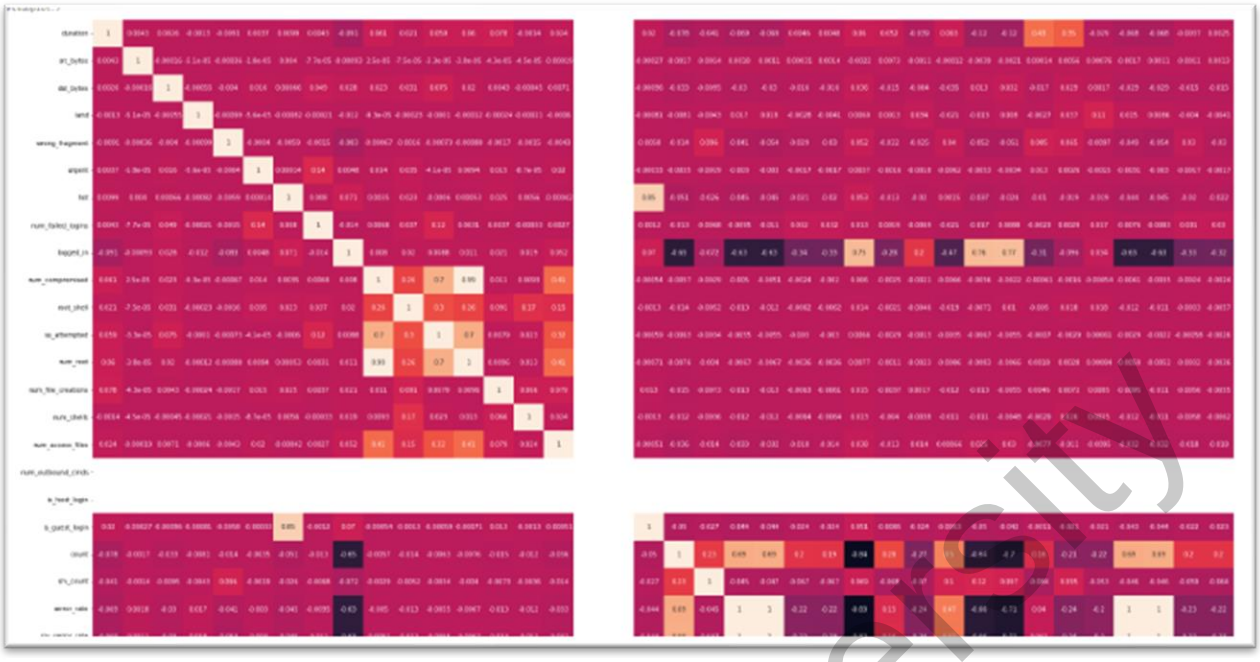


Figure 26: Variance values for each feature of the data set

After studying the previous table, it is possible to notice the zero values for each of the two attributes (num_outband_cmds. Therefore, in this experiment, they were omitted from the training data set and the test data set (is_hot_logins), assuming that they do not contribute to the process intrusion detection. Based on the above, the data set contains 39 features as input to the neural network.

Then, through the previous table, the values of the Correlation Coefficient were calculated between each pair of the 39 traits resulting from previous experience. This is in search of features that are strongly associated, that is, with close correlation coefficient values of one, 0.9 or more. With the aim of dispensing with one of the features of each strongly correlated pair, given that the effect of one of them represents other effect, thus reducing the number of attributes involved in the classification currency without negatively affecting the accuracy of the system. Show the following table is the pairs of traits with strong correlation.

Associated feature pairs				correlation coefficient
13	num_compromised	16	num_root	0.9941
25	serror_rate	26	sev_serror_rate	0.9977
25	serror_rate	38	dst_host_serror_rate	0.9981
25	serror_rate	39	dst_host_srv_serror_rate	0.997
26	sev_serror_rate	38	dst_host_serror_rate	0.997
26	sev_serror_rate	39	dst_host_srv_serror_rate	999،0
27	rerror_rate	28	sev_rerror_rate	0.9943
27	rerror_rate	40	dst_host_rerror_rate	0.9852
27	rerror_rate	41	dst_host_srv_rerror_rate	0.9834
28	sev_rerror_rate	40	dst_host_rerror_rate	0.98
28	sev_rerror_rate	41	dst_host_srv_rerror_rate	0.9847
33	dst_host_srv_count	34	dst_host_same_srv_rate	0.9491
38	dst_host_serror_rate	39	dst_host_srv_serror_rate	0.9974
40	dst_host_rerror_rate	41	dst_host_srv_rerror_rate	0.9828

Table 11: Strongly Correlated Trait Pairs

The above table shows only pairs of traits with a strong correlation, that is, those whose correlation value was greater than 0.9 where the value of correlation generally ranges between zero and one, as mentioned earlier, the closer it is to one, he greater the correlation more powerful. On the basis of this, a single feature of each correlated pair was preserved because the effect of one represents the effect of the other. On for example, the first line of the table represents the correlation between the two attributes (13) and (16). Each of these attributes has been omitted. pair separately and compare the results when each of them is absent to note that the results are similar, i.e., the presence of one of the two characteristics compensates For the presence of the other characteristic, therefore, one of the two characteristics was chosen. In the same way for the second line of the table we note that the correlation between the two traits (25) and (26) is strong, and each of these two traits is strongly correlated with the same set of traits as is it is shown in the table. Therefore, it is sufficient to retain the feature (25) or feature (26) and dispense with the rest of the features associated with it. num_compromised(13), sev_serror_rate(26), rerror_rate(27), on this basis the attributes have been deleted dst_host_same_srv_rate(34), dst_host_serror_rate(38), dst_host_srv_serror_rate(39), num_root(16), and keep attributes dst_host_rerror_rate(40), dst_host_srv_rerror_rate(41) characteristic as 31. Then

serror_rate(25), sev_rerror_rate(28), dst_host_srv_count(33) is used. As an input for a neural network classifier and test.

The features Loggrd_in and service were deleted because they did not affect the accuracy of the system, so the number of features eventually became 29.

4.6.4.2.3 Divide Data into features & labels

When splitting the data, X is conventionally the features and y is the label. We should start with separating features for our model from the target variable. Notice that in our case all columns except ‘target’ are features that we want to use for the model. Our target variable is ‘target’. We can use the following code to do target separation.

```
[ ]  # creating a dataframe with multi-class labels (Dos,Probe,R2L,U2R,normal)
     x_train = train_data.drop('target',axis=1)
     y_train = pd.DataFrame(train_data.target)
     x_test =  test_data.drop('target',axis=1)
     y_test = pd.DataFrame(test_data.target)
     pred_f=x_train.columns
     x_train.head()
```

Figure 27: Divide Data into Feat & labels.

4.6.4.2.4 Categorical Features Transformation

The 29 features within KDDCup99 are converted to a unified digital representation. As long as there are attributes with non-symbolic values numeric available in KDDCup99, these values are converted to a numeric one, since the neural network only accepts income with insignificant values. Each of the attributes, flag, protocol_type they are labeled character features, and each has values different, the 'protocol_type' feature has three different values, and the 'flag' feature has it 11 different values.

This is why, we need encoding methods to convert non-numerical data to meaningful numerical data. For this we look at Pandas get_dummies method.

get_dummies is one of the easiest way to implement one hot encoding method and it has very useful parameters, of which we will mention the most important ones.

Pandas uses the object data type to indicate categorical variables/columns because there are categorical (non-numerical) columns, and we need to transform them. For this, we will implement get_dummies.

get_dummies it creates a one-hot encoded matrix for every target column we specify.

```
x_test = pd.get_dummies(x_test)
x_test
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	root_shell	su_attempted	...	flag_REJ	flag_RSTO	flag_RSTOS0	flag_RSTR
0	0	105	146	0	0	0	0	0	0	0	...	0	0	0	0
2	0	105	146	0	0	0	0	0	0	0	...	0	0	0	0
3	0	105	146	0	0	0	0	0	0	0	...	0	0	0	0
4	0	105	146	0	0	0	0	0	0	0	...	0	0	0	0
5	0	29	0	0	0	0	0	0	0	0	...	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
310329	0	30	0	0	0	0	0	0	0	0	...	0	0	0	0
310353	0	30	0	0	0	0	0	0	0	0	...	0	0	0	0
310403	0	105	105	0	0	0	0	0	0	0	...	0	0	0	0
310666	0	30	0	0	0	0	0	0	0	0	...	0	0	0	0
310930	0	30	0	0	0	0	0	0	0	0	...	0	0	0	0

77287 rows × 41 columns

Figure 28: Apply get_dummies function on test dataset

4.6.4.2.5 Numerical Feature Scaling

Feature Scaling is a method to transform the numeric features in a dataset to a standard range so that the performance of the machine learning algorithm improves. It can be achieved by normalizing or standardizing the data values. This scaling is generally preformed in the data pre-processing step when working with machine learning algorithm.

Example, if we have weight of a person in a dataset with values in the range 15kg to 100kg, then feature scaling transforms all the values to the range 0 to 1 where 0 represents lowest weight and 1 represents highest weight instead of representing the weights in kgs.

In feature scaling, we scale the data to comparable ranges to get proper model and improve the learning of the model. The scale which we choose is not important but each of the feature in the dataset should be on the same scale. Example, if one feature is chosen to be in range 0 to 1 then all the remaining features in the same dataset should also be in range 0 to 1.

Why Feature Scaling?

To avoid wrong predictions, the range of all features are scaled so that each feature contributes proportionately, and model performance improves drastically.

Another reason for feature scaling is that if the values of a dataset are small then the model learns fast compared the unscaled data. Example, in gradient decent, to minimize the cost function, if the range of values is small then the algorithm converges much faster.

Normalization:

This is also known as Min-Max scaling. It scales the data to the range between 0 and 1. This scaling is performed based on the below formula.

$$xnew = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Where x is the current value to be scaled, min(X) is the minimum value in the list of values and max(X) is the maximum value in the list of values.

Numerical Feature Scaling

```
[ ]  scaler_x_train = MinMaxScaler().fit(x_train)
     scaler_x_test = MinMaxScaler().fit(x_test)

[ ]  x_train = scaler_x_train.transform(x_train)
     x_test = scaler_x_train.transform(x_test)
     x_train=pd.DataFrame(x_train)
     x_test=pd.DataFrame(x_test)
     x_train
     x_test
```

	0	1	2	3	4	5	6	7	8	9	...	31	32	33	34	35	36	37	38	39	40
0	0.0	1.514331e-07	0.000028	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1	0.0	1.514331e-07	0.000028	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	0.0	1.514331e-07	0.000028	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	1.514331e-07	0.000028	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
4	0.0	4.182437e-08	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
77282	0.0	4.326659e-08	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
77283	0.0	4.326659e-08	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
77284	0.0	1.514331e-07	0.000020	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
77285	0.0	4.326659e-08	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
77286	0.0	4.326659e-08	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

77287 rows × 41 columns

Figure 29: Apply MinMaxScaler function on train dataset and test dataset

4.6.4.3 Building neural network Model

The model consists of:

- One hidden layer.
- The number of neurons in the first two layer is 50 neurons.
- The number of neurons in the output layer is 5 neurons.
- The activation function in the first two layers is relu function.
- The activation function in the output layer is softmax function.

```
mlp2 = Sequential() # initializing model
# input layer and first layer with 50 neurons
mlp2.add(Dense(units=50, input_dim=x_train.shape[1], activation='relu'))
mlp2.add(Dense(units=50, activation='relu'))
# output layer with softmax activation
mlp2.add(Dense(units=5,activation='softmax'))
```

Figure 30: Building neural network Model

- **Sequential**

The core idea of Sequential API is simply arranging the Keras layers in a sequential order and so, it is called Sequential API. Most of the ANN also has layers in sequential order and the data flows from one layer to another layer in the given order until the data finally reaches the output layer.

- **Add layers**

To add a layer, simply create a layer using Keras layer API and then pass the layer through add() function as specified below.

```
from keras.models import Sequential

model = Sequential()
input_layer = Dense(32, input_shape=(8,)) model.add(input_layer)
hidden_layer = Dense(64, activation='relu'); model.add(hidden_layer)
output_layer = Dense(8)
model.add(output_layer)
```

Figure 31: created one input layer, one hidden layer and one output layer

Here, we have created one input layer, one hidden layer and one output layer.

4.6.4.4 Model Compilation

The compilation is the final step in creating a model. Once the compilation is done, we can move on to training phase.

```
# defining loss function, optimizer, metrics and then compiling model
mlp2.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 32: Model Compilation

Let us learn few concepts required to better understand the compilation process.

- **Loss**

In machine learning, Loss function is used to find error or deviation in the learning process. Keras requires loss function during model compilation process.
Keras provides quite a few losses function in the losses module, and they are as follows:

- mean_squared_error
- mean_absolute_error
- mean_absolute_percentage_error
- mean_squared_logarithmic_error
- categorical_crossentropy
- sparse_categorical_crossentropy

- **Optimizer**

In machine learning, Optimization is an important process which optimize the input weights by comparing the prediction and the loss function. Keras provides quite a few optimizers as a module, optimizers and they are as follows:

- SGD − Stochastic gradient descent optimizer.
- RMSprop − RMSProp optimizer.
- Adagrad − Adagrad optimizer.
- Adadelta − Adadelta optimizer.
- Adam − Adam optimizer.Adamax − Adamax optimizer from Adam.

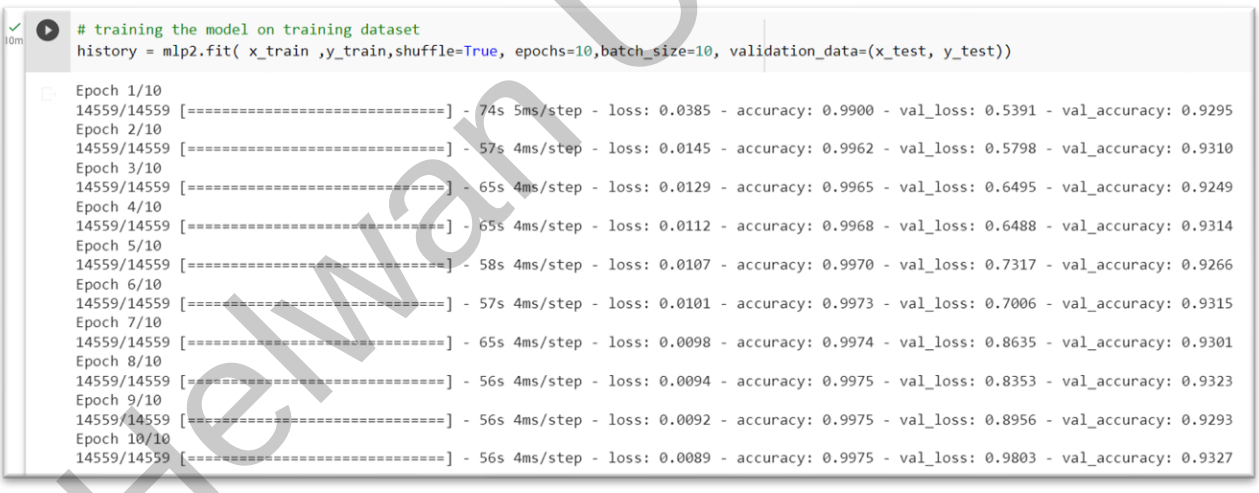
- **Metrics**

In machine learning, Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in training process. Keras provides quite a few metrics as a module, metrics and they are as follows:

- accuracy
- binary_accuracy
- categorical_accuracy
- sparse_categorical_accuracy
- top_k_categorical_accuracy
- sparse_top_k_categorical_accuracy
- cosine_proximity
- clone_metric

4.6.4.5 Model Training

Models are trained by NumPy arrays using fit (). The main purpose of this fit function is used to evaluate your model on training. This can be also used for graphing model performance.



```
# training the model on training dataset
history = mlp2.fit( x_train ,y_train,shuffle=True, epochs=10,batch_size=10, validation_data=(x_test, y_test))

Epoch 1/10
14559/14559 [==============================] - 74s 5ms/step - loss: 0.0385 - accuracy: 0.9900 - val_loss: 0.5391 - val_accuracy: 0.9295
Epoch 2/10
14559/14559 [==============================] - 57s 4ms/step - loss: 0.0145 - accuracy: 0.9962 - val_loss: 0.5798 - val_accuracy: 0.9310
Epoch 3/10
14559/14559 [==============================] - 65s 4ms/step - loss: 0.0129 - accuracy: 0.9965 - val_loss: 0.6495 - val_accuracy: 0.9249
Epoch 4/10
14559/14559 [==============================] - 65s 4ms/step - loss: 0.0112 - accuracy: 0.9968 - val_loss: 0.6488 - val_accuracy: 0.9314
Epoch 5/10
14559/14559 [==============================] - 58s 4ms/step - loss: 0.0107 - accuracy: 0.9970 - val_loss: 0.7317 - val_accuracy: 0.9266
Epoch 6/10
14559/14559 [==============================] - 57s 4ms/step - loss: 0.0101 - accuracy: 0.9973 - val_loss: 0.7006 - val_accuracy: 0.9315
Epoch 7/10
14559/14559 [==============================] - 65s 4ms/step - loss: 0.0098 - accuracy: 0.9974 - val_loss: 0.8635 - val_accuracy: 0.9301
Epoch 8/10
14559/14559 [==============================] - 56s 4ms/step - loss: 0.0094 - accuracy: 0.9975 - val_loss: 0.8353 - val_accuracy: 0.9323
Epoch 9/10
14559/14559 [==============================] - 56s 4ms/step - loss: 0.0092 - accuracy: 0.9975 - val_loss: 0.8956 - val_accuracy: 0.9293
Epoch 10/10
14559/14559 [==============================] - 56s 4ms/step - loss: 0.0089 - accuracy: 0.9975 - val_loss: 0.9803 - val_accuracy: 0.9327
```

Figure 33: Model training

Here,

(x_train, y _train): It is a tuple to evaluate your data.

(x_test, y _test): It is a tuple to validate your model.

Epochs: no of times the model is needed to be evaluated during training.

batch_size: training instances.

4.6.4.6 practical results

4.6.4.6.1 Model Evaluation

Evaluation is a process during development of the model to check whether the model is best fit for the given problem and corresponding data. Keras model provides a function, evaluate which does the evaluation of the model. It has three main arguments:

- Test data
- Test data label
- Verbose - true or false

```
score = mlp2.evaluate(x_test, y_test, verbose = 0)
print('Test accuracy:', score[1])

Test accuracy: 0.9326536059379578
```

Figure 34: Model Evaluation

The test accuracy is 93.27%.

4.6.4.6.2 Model Prediction

Prediction is the final step and our expected outcome of the model generation. Keras provides a method, predict to get the prediction of the trained model. The signature of the predict method is as follows:

```
pred = mlp2.predict(x_test)
pred = np.argmax(pred, axis = 1)[:5]
label = np.argmax(y_test,axis = 1)[:5]

print(pred)
print(label)

2416/2416 [==============================] - 4s 2ms/step
[0 0 0 0 0]
   target
0       0
2       0
3       0
4       0
5       0
```

Figure 35: Model Prediction

Here,

- Line 1 call the predict function using test data.
- Line 2 gets the first five prediction.
- Line 3 gets the first five labels of the test data.
- Line 5 - 6 prints the prediction and actual label.

The output of both array is identical, and it indicate that our model predicts correctly the first five images.
0 expresses normal attack.

4.6.4.6.3 Confusion matrix

A confusion matrix, also known as error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one; in unsupervised learning it is usually called a matching matrix. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa – both variants are found in the literature. The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e. commonly mislabeling one as another).

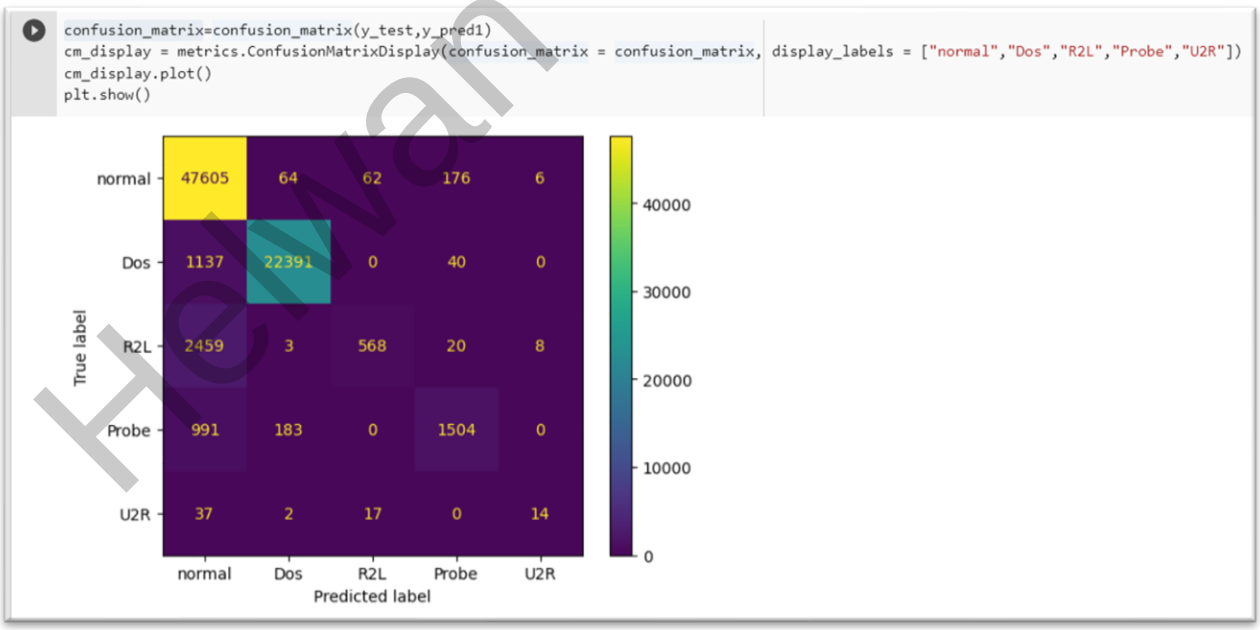


Figure 36: Confusion matrix of model

4.7 Extra System Features Design

We built another 10 functions in the system to make it full package system to deal with all types of malicious activities or attacks. Here are the design concept behind them:

4.7.1 Arp poisoning detector

ARP stands for Address Resolution Protocol. It is a protocol used in computer networking to map an IP address to a MAC address. Every device connected to a network has a unique MAC address, which is a hardware address assigned by the manufacturer. However, IP addresses are assigned dynamically, and they may change over time.

When a device wants to communicate with another device on the network, it needs to know the MAC address of the destination device. It can use the ARP protocol to map the IP address of the destination device to its MAC address. The ARP protocol works by broadcasting a message to all devices on the network, asking for the MAC address associated with a particular IP address. The device with the corresponding IP address then responds with its MAC address, and the requesting device can use this information to send packets to the destination device.

A man-in-the-middle situation is simulated to simulate the attack. The attacker launches a ARP Poison attack which targets the network (or specific hosts) by sending poisoned/false ARP packets, which allows the attacker to intervene, intercept and/or modify the network traffic. Anything and everything that passes through the target's (victim's) device can be intercepted by the attacker once he is Man-in-the-Middle.

The basic idea behind the function is to passively scan or monitor the network (keep sniffing the incoming ARP packets). After capturing the ARP packet, it is analyzed to obtain the following two components:

• The source MAC address of the sniffed ARP packet (could be spoofed) .

• The real physical address of the sender (can be found by starting an ARP request for the source IP address).

These two components are then compared. If found to differ, then the system is definitely facing an ARP attack.

The detection algorithm contains the following functions:

• Obtain MAC Address: This function is for obtaining the MAC address of the Gateway (or any other device) given the IP address. This function basically makes an ARP request to the server to obtain the real MAC address of the given IP address.

• Process each sniffed ARP Packet: This function is applied to each packet sniffed. It compares the response MAC address (the address sent by the packet itself) and the Real MAC address obtained using the get mac function for the given IP. If they are different, a warning message is printed and the packet is discarded.

A script written using Scapy library is used to obtain the response MAC address from the sniffed ARP Packet received by the target machine.

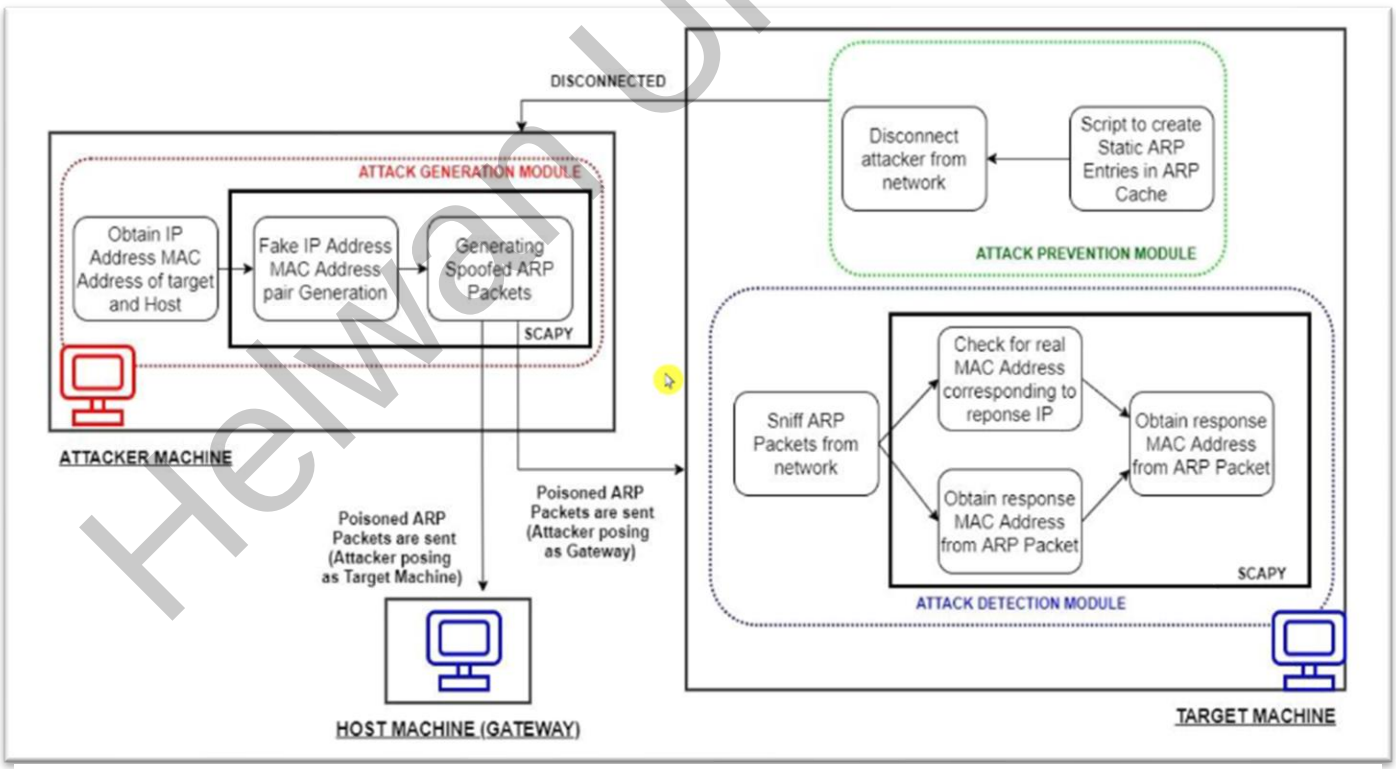


Figure 37: Proposed architecture for ARP

Here's a breakdown of each of the methods:

__init__(self, interface): This is the constructor method of the arp_scanner class. It takes one argument, which is the name of the network interface to use for sniffing.

mac(self, ipadd): This method takes an IP address as an argument and returns the corresponding MAC address. It does this by sending an ARP request to the IP address and receiving the ARP response containing the MAC address.

start(self, timeout=10): This method starts the ARP poisoning detection process. It prints a message indicating that the detection process has started, and then starts sniffing network traffic on the specified interface. If an ARP poisoning attack is detected, the **process_sniffed_packet()** method is called. The sniffing process runs for a specified timeout period, after which it stops and prints a message indicating that the detection process has stopped.

process_sniffed_packet(self, packet): This method is called for each packet sniffed during the start() method. It checks whether the packet contains an ARP protocol and whether the ARP operation is a "reply" (opcode=2). If so, it checks the original MAC address for the IP address in the packet, and compares it to the MAC address in the ARP response. If the two MAC addresses are different, it means that an ARP poisoning attack is occurring, and a message is printed indicating that the attack has been detected. The **set_response()** method is also called to save the message in a file.

set_response(self, res): This method saves the given response message in a file called arp-response.cache on the local file system.

get_response(self): This method reads the response message from the arp-response.cache file and returns it. If the file does not exist or is empty, it returns a default message indicating that the ARP table is safe.

run(self): This method calls the **start()** method to begin the ARP poisoning detection process and then calls the **get_response()** method to retrieve the response message saved during the detection process. It returns the retrieved response message.

4.7.2 Privilege Escalation Detection

Privilege escalation is a security vulnerability that occurs when an attacker gains access to higher-level permissions or administrative privileges on a system. This allows the attacker to perform actions that they would not normally be able to do, such as accessing sensitive data, modifying system settings, or executing malicious code. Privilege escalation attacks can take many forms, and attackers may use a variety of techniques to gain elevated privileges. Here are common methods of privilege escalation:

- Exploiting software vulnerabilities: Attackers may exploit vulnerabilities in software to gain elevated privileges on a system. For example, they may use a buffer overflow attack to execute arbitrary code with elevated privileges.
- Stealing credentials: Attackers may steal login credentials for a privileged account to gain access to sensitive resources. For example, they may use a phishing attack to trick a user into revealing their password.
- Misconfigured permissions: Misconfigured permissions on a system can also lead to privilege escalation. For example, if a user is given write access to a sensitive file or directory, they may be able to modify the file to gain elevated privileges.
- Backdoor access: Attackers may create a backdoor on a system to maintain access even after they have been detected and removed. This backdoor can allow them to regain elevated privileges later.

The used detection Algorithm searches for **SUID files** and looks for writable **/etc/passwd** and **/etc/shadow** files, which could allow an attacker to escalate their privileges. It also checks if the **/etc/sudoers** file is readable or writable, which could enable an attacker to run commands with superuser privileges. The algorithm takes a single command-line argument, request_input, which is not used in the script. It then uses various shell commands to search for **SUID** files, and to check the permissions of **/etc/passwd**, **/etc/shadow**, and **/etc/sudoers.**

The output of the script is printed to the console. It first prints a list of SUID files found on the system. It then checks if /etc/passwd is writable by non-root

users and prints a message indicating whether it is safe or not. Similarly, it checks if **/etc/shadow** is readable or writable by non-root users and prints a message indicating its safety. Finally, it checks the permissions of **/etc/sudoers** and prints a message indicating whether it is safe or not.

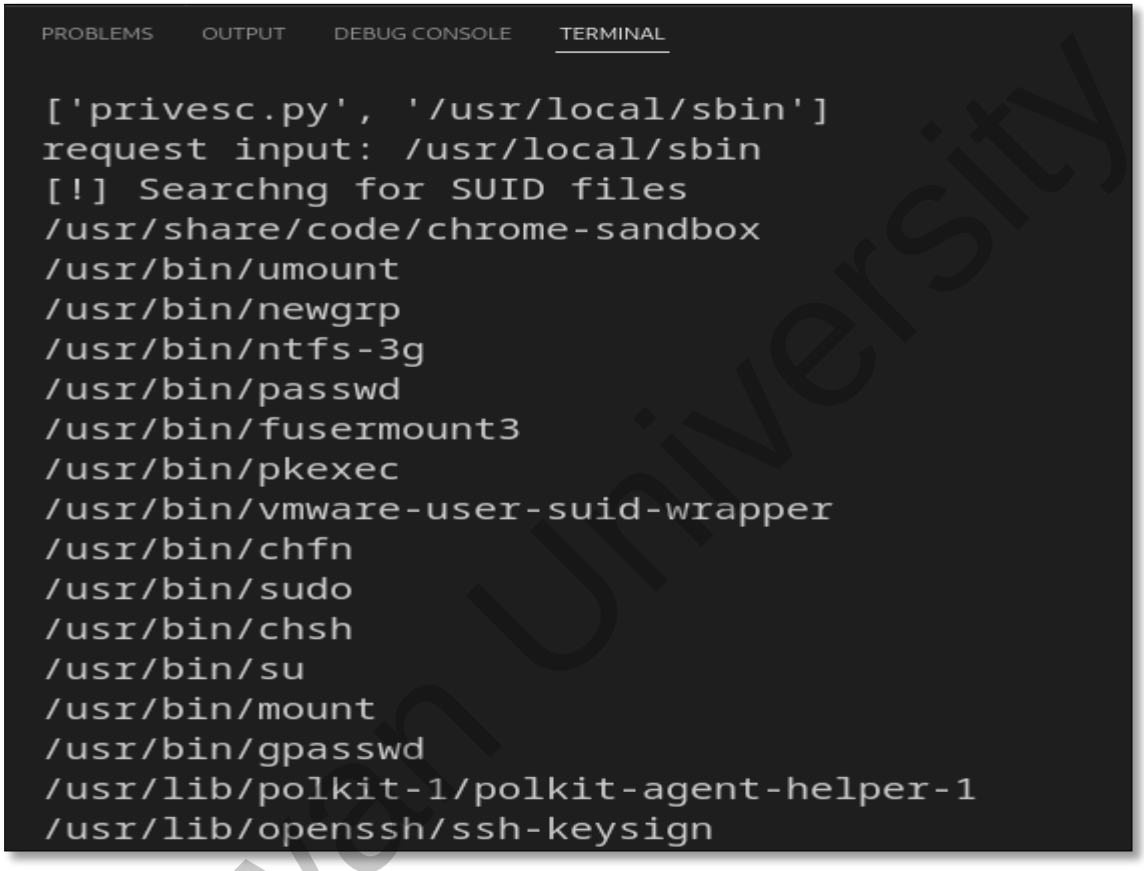


Figure 39: List of SUID files found on the system

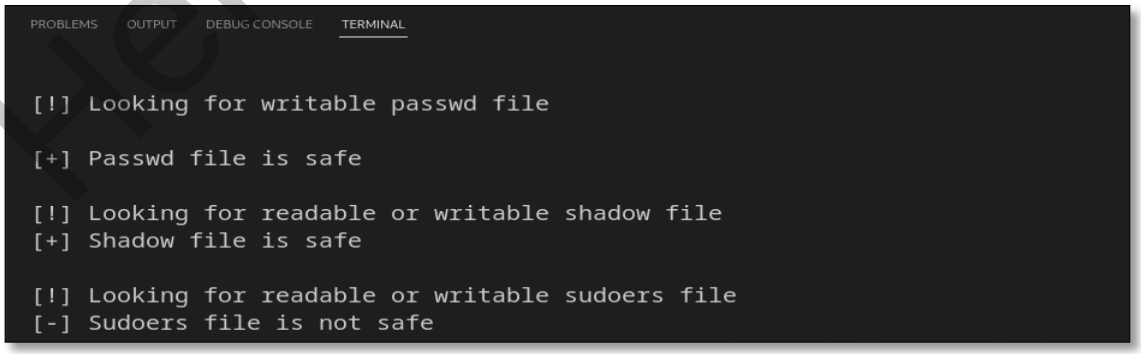


Figure 38: message indicating whether permissions is safe or not

4.7.3 Passwd File Monitor

The passwd file is a system file on Unix-like operating systems that stores user account information, such as the username, encrypted password, user ID, and group ID. It is usually located in the /etc directory and is readable by all users. The file is used to authenticate users when they log in and is essential for the proper functioning of the operating system. The passwd file can only be modified by the root user or a user with root privileges. It is important to keep the passwd file secure and writable only by authorized users to prevent unauthorized access or modification.

Figure 40: The passwd file

This function scans the content of the passwd file and finds the differences between the cached passwd file and the current one.

It returns a dictionary that contains the following keys:

changed: a Boolean indicating whether there are differences between the cached passwd file and the current one.

before: a list containing the cached passwd file.

after: a list containing the current passwd file.

diff: a list of different lines between the two files.

This function uses the get_cached_passwd() and get_passwd() functions to get the cached and current passwd files, respectively. It also uses the difflib library to find the differences between the two files.

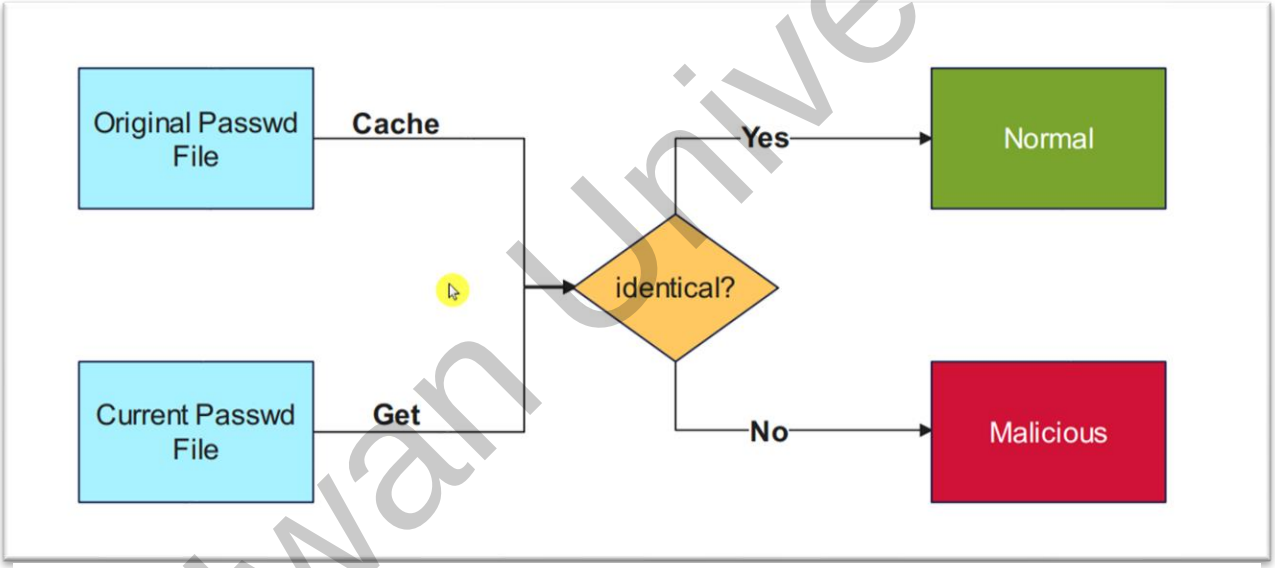


Figure 41: Scanning cached and current passwd files

By scanning this file, we can check whether if any malicious activity happened in the background. These malicious activities may be attacker made a user for himself to use in the system or any change in that file will be a sign for any malicious activity in case that the admin didn't do that change. In the case of the passwd file, changes to this file could potentially allow unauthorized users to gain access to the system or escalate their privileges. By monitoring this file, administrators can quickly detect any changes and take appropriate action to mitigate the risk.

Here is a sample of the output:

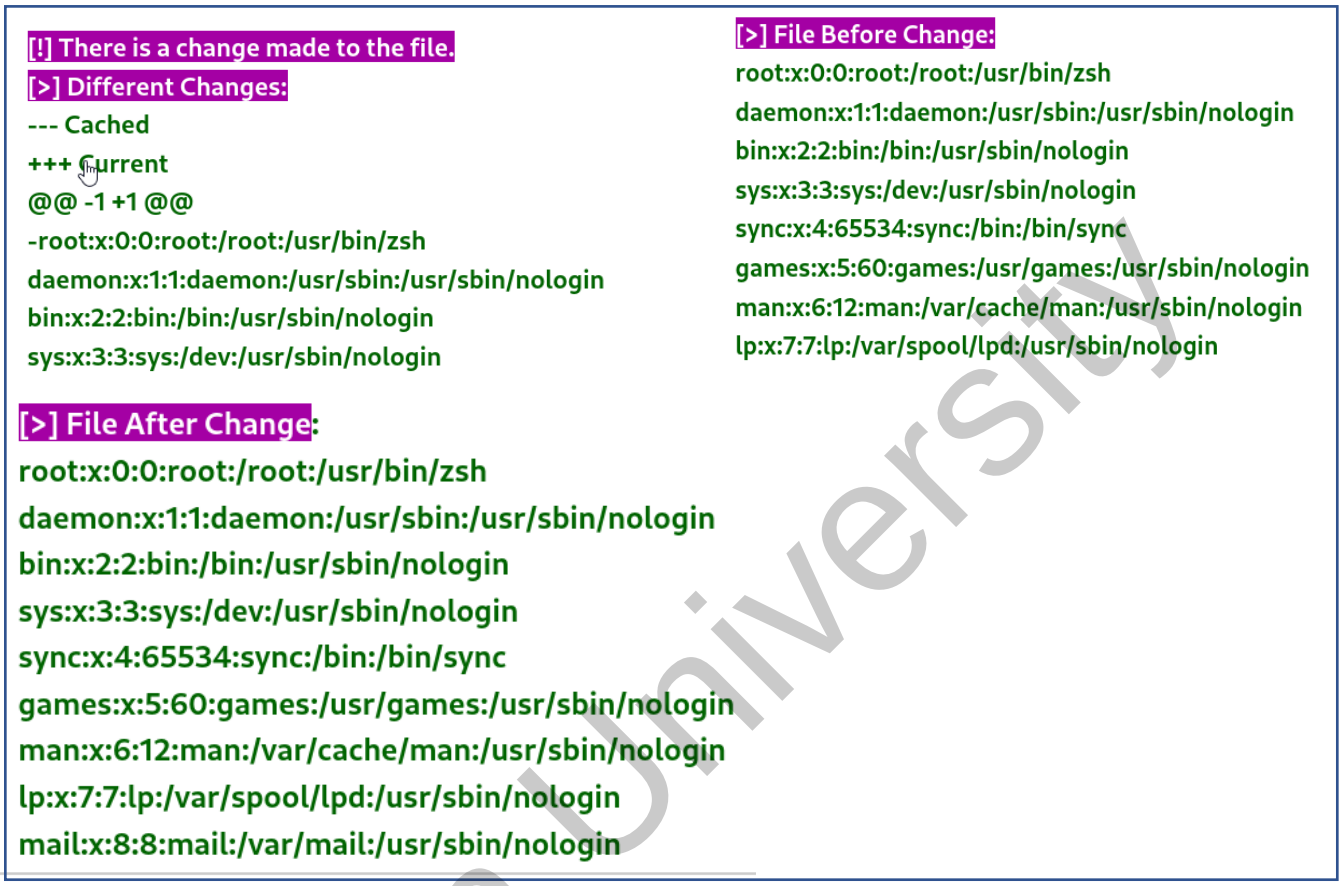


Figure 42: Sample of the output of Passwd File

4.7.4 Tmp Directory Monitor

The tmp directory is a directory on a Linux system where temporary files are stored. This directory is usually located at /tmp and is often used by programs and applications to store files that are only needed temporarily, such as logs, caches, and other data. As the name suggests, the files stored in the tmp directory are not meant to persist across reboots and are usually deleted automatically by the system. However, some programs may not clean up after themselves and leave their temporary files in the tmp directory, which can lead to clutter and potential security issues.

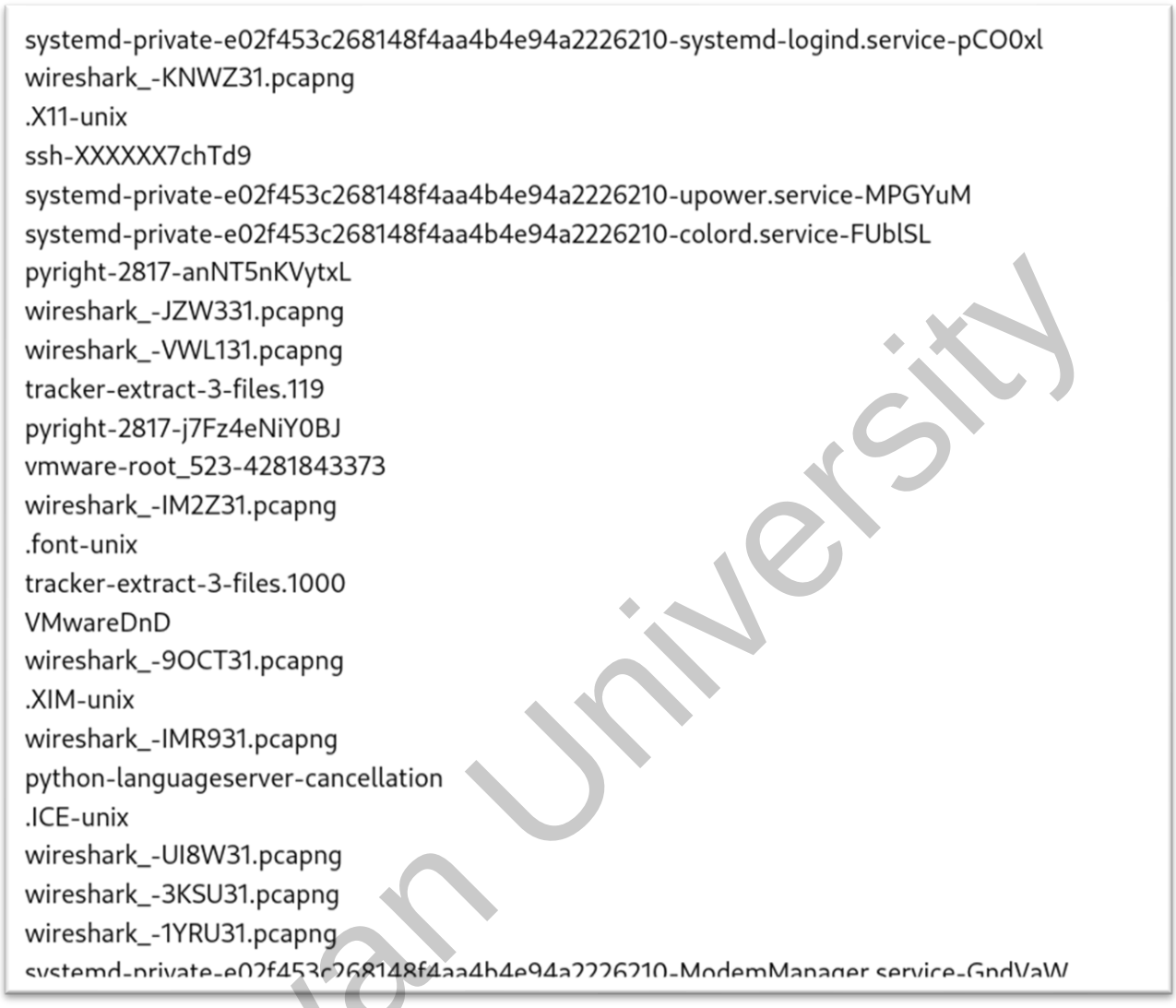


Figure 43: Tmp Directory Monitor

A temporary directory monitor is a software tool that monitors the contents of a specified directory and alerts the user or performs some action when changes occur within that directory. This type of tool is useful in situations where temporary files are being created or modified in a directory, and it's important to keep track of those changes. For example, some software applications may use a temporary directory to store temporary files during processing, and if those files are not properly cleaned up or deleted, it can cause problems for the system or other applications.

A temporary directory monitor can help keep track of these changes and ensure that the temporary files are properly managed. The tool may be configured to monitor the directory for specific types of changes, such as file creations, modifications, deletions, or renames. When a change is detected, the tool may alert the user or perform some other action, such as automatically deleting old or unused temporary files.

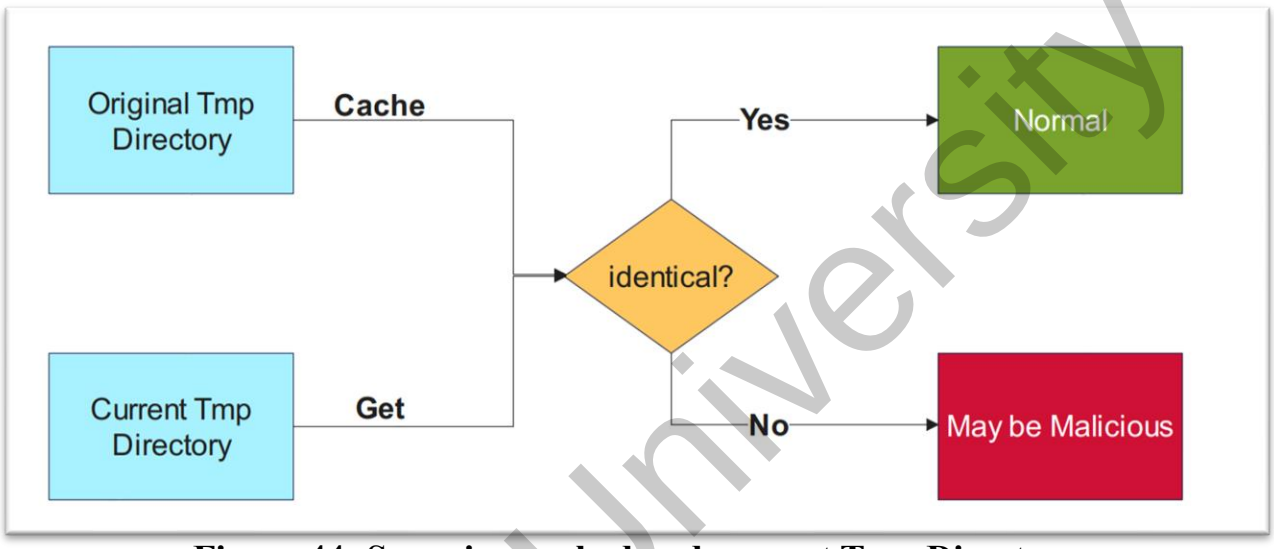


Figure 44: Scanning cached and current Tmp Directory

Here is a sample of the output:

```
[!] tmp Directory has Modified.
[>] Different Changes:
test1.txt

[>] Directory Before Change:
systemd-private-e02f453c268148f4aa4b4e94a2226210-systemd-logind.service-pCO0xl
wireshark_-KNWZ31.pcapng
.X11-unix
ssh-XXXXXX7chTd9
[>] Directory After Change:
systemd-private-e02f453c268148f4aa4b4e94a2226210-systemd-logind.service-pCO0xl
wireshark_-KNWZ31.pcapng
.X11-unix
ssh-XXXXXX7chTd9
systemd-private-e02f453c268148f4aa4b4e94a2226210-upower.service-MPGYuM
systemd-private-e02f453c268148f4aa4b4e94a2226210-colord.service-FUblSL
pyright-2817-anNT5nKVytxL
wireshark_-JZW331.pcapng
wireshark_-VWL131.pcapng
```

Figure 45: Sample of the output of Tmp Directory

4.7.5 History file monitor

In Linux, the shell maintains a history file that keeps track of the commands executed by the user in a terminal session. This file is typically stored in the user's home directory and is named .bash_history or .zsh_history, depending on the user's default shell. To monitor changes to the history file in real-time, we used the cat command to list all history of the user.

```
sudo apt update\

sudo apt install software-properties-common apt-transport-https curl\

curl -sSL https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -\

sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"\

sudo apt update\
```

Figure 46: History file monitor

4.7.6 Port Blocker

A port blocker is a software tool that allows you to block network traffic to or from specific network ports on a computer or network device. This can be useful for improving security by preventing unauthorized access to sensitive services or applications that are running on the computer or device. When using a port blocker, it's important to carefully consider which ports to block and how to configure the blocker to avoid unintended consequences. For example, blocking port 80 (HTTP) could prevent users from accessing websites, while blocking port 22 (SSH) could prevent remote access to the computer. Additionally, it's important to periodically review and update the port blocking rules to ensure they remain relevant and effective in the face of changing threats.

This function takes three arguments:

port: the port number to block or unblock.

interface: the network interface on which to apply the rule.

action: the action to take, which can be either 'block' or 'unblock'.

If the action argument is 'block', the function will execute an iptables command to block incoming traffic on the specified port and network interface. If the action argument is 'unblock', the function will execute an iptables command to remove the blocking rule and allow incoming traffic on the specified port and network interface.

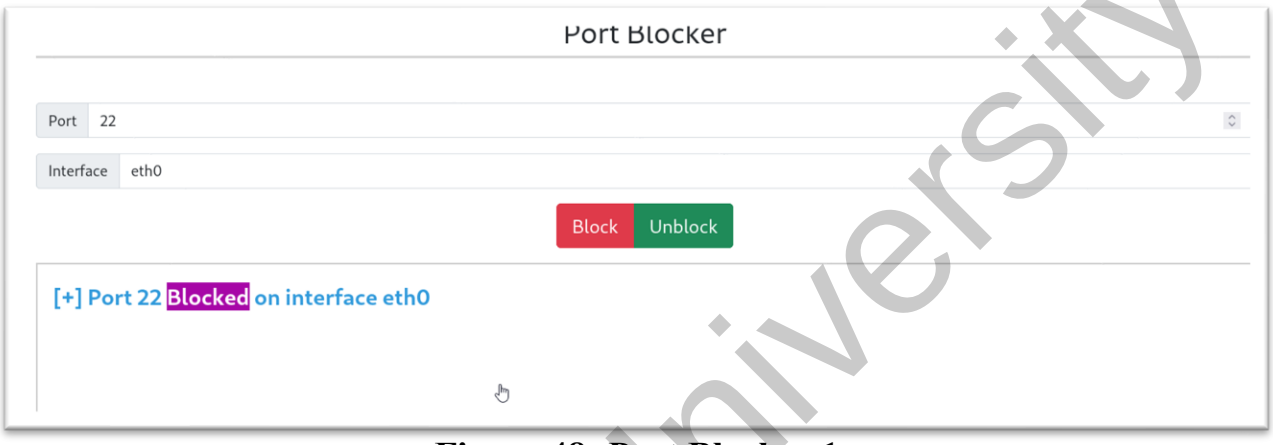


Figure 48: Port Blocker 1

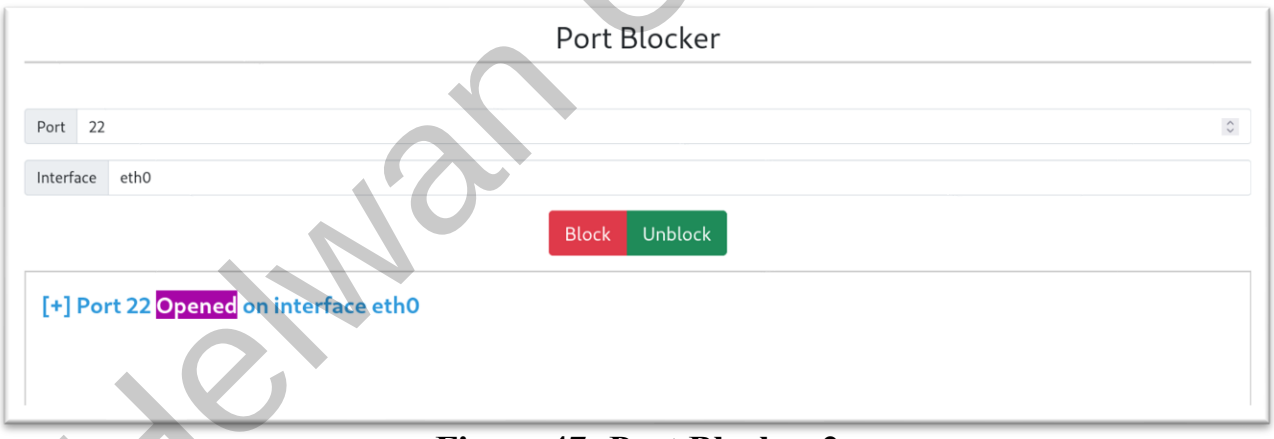


Figure 47: Port Blocker 2

4.7.7 Services Scanner

services scanner is a tool that allows you to scan a Linux system and discover what services are running on it. This can be useful for system administrators who need to monitor the services running on their network or for security professionals who are conducting a vulnerability assessment. The scan_services() function you provided is a Python function that uses the os module to execute the lsof command and list the running services on a Linux system. This function takes no arguments and returns a string that contains the output of the lsof command.

Here is the sample of the output:

```
COMMAND    PID USER   FD   TYPE  DEVICE SIZE/OFF NODE NAME
NetworkMa  550 root   26u  IPv4   18142      0t0  UDP 192.168.140.132:68->192.168.140.254:67
code      2654  mo   24u  IPv4 1409044      0t0  TCP 192.168.140.132:48406->51.144.164.215:443 (CLOSE_WAIT)
code      2726  mo   44u  IPv4   27266      0t0  TCP 127.0.0.1:40129 (LISTEN)
python3  12306 root   9u  IPv4 1395026      0t0  TCP 127.0.0.1:30001 (LISTEN)
python3  12306 root  11u  IPv4 1395026      0t0  TCP 127.0.0.1:30001 (LISTEN)
python3  12384 root   9u  IPv4 1395026      0t0  TCP 127.0.0.1:30001 (LISTEN)
python3  12384 root  11u  IPv4 1411297      0t0  TCP 127.0.0.1:30001->127.0.0.1:35150 (ESTABLISHED)
python3  12384 root  12u  IPv4 1395026      0t0  TCP 127.0.0.1:30001 (LISTEN)
```

Figure 49: Sample of the output of services scanner

4.7.8 Devices Monitor

This function scans all devices on the network to check if there is any malicious or unknown devices on the network. The function uses the os module to execute the arp command and list the devices and their IP addresses that are currently connected to the same network as the machine running the arp command.

```
Address          HWtype  HWaddress          Flags Mask      Iface
192.168.140.1    ether   00:50:56:c0:00:08  C               eth0
192.168.140.136  ether   00:0c:29:fa:32:90  C               eth0
_gateway         ether   00:50:56:ea:70:12  C               eth0
192.168.140.254  ether   00:50:56:f6:08:90  C               eth0
```

Figure 50: Devices Monitor

4.7.9 Malicious files detector

Malicious files are files that contain malicious code or software that is designed to harm a computer system or steal sensitive information. Malicious files can come in various forms, including viruses, trojans, worms, ransomware, and spyware.

Here are some common types of malicious files:

- Viruses - These are programs that replicate themselves and infect other files on the system. They are often spread through email attachments or downloads from the internet.
- Trojans - These are programs that are disguised as legitimate software but contain malicious code. They often trick users into installing them by pretending to be useful or desirable software.
- Worms - These are self-replicating programs that spread through networks and can infect multiple systems.
- Ransomware - These are programs that encrypt files on the system and demand payment in exchange for the decryption key.
- pyware - These are programs that are designed to monitor a user's activity on their computer and collect sensitive information such as passwords, credit card numbers, and browsing history.

The function is designed to scan a specific directory, in this case the /home/mo/Desktop directory, for potentially malicious files. It does this by first defining a list of known malicious file hashes (malicious_hashes). Then, it iterates over each file in the directory using the os.listdir() function and checks whether each file is a regular file using the os.path.isfile() function. For each regular file found in the directory, the function reads the contents of the file and calculates its MD5 hash using the hashlib.md5() function. It then checks whether the calculated hash is present in the malicious_hashes list. If a match is found, the file is added to the malicious_files list.

Finally, if any malicious files were found, the function returns a string indicating the number of malicious files found and their paths. Otherwise, it returns a string indicating that no malicious files were found in the directory.

Here is a sample for the output:

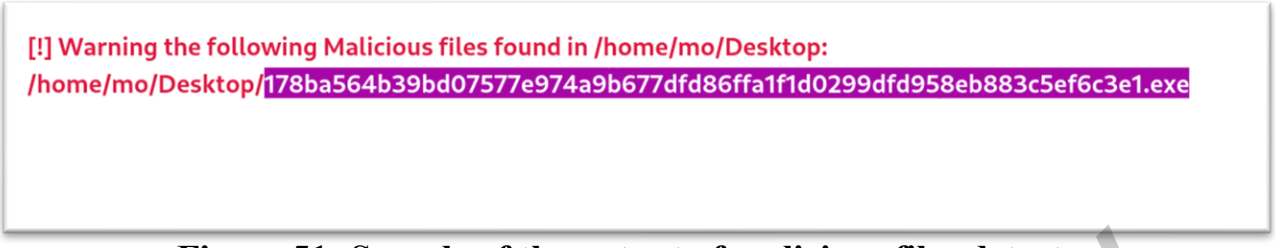


Figure 51: Sample of the output of malicious files detector

4.7.10 Unencrypted Protocols detector

Unencrypted protocols are communication protocols that do not provide any encryption for the data being transmitted between two points. This means that anyone with access to the network traffic can potentially intercept and read the data being transmitted. Some examples of unencrypted protocols include:

- HTTP (Hypertext Transfer Protocol) - used for transmitting web pages and other web content over the internet.
- FTP (File Transfer Protocol) - used for transferring files between computers on a network.
- Telnet - used for remote login to a computer over a network.
- SMTP (Simple Mail Transfer Protocol) - used for sending email messages over the internet.
- DNS (Domain Name System) - used for translating human-readable domain names into IP addresses.

In contrast, encrypted protocols use various encryption algorithms to protect the data being transmitted over a network. Examples of encrypted protocols include HTTPS (secure HTTP), SFTP (secure FTP), SSH (secure shell), and encrypted email protocols such as SMTPS and IMAPS. The use of encrypted protocols is important for protecting sensitive data, such as login credentials, financial information, and personal information, from unauthorized access.

An unencrypted protocols detector is a tool or software that can identify network traffic that is being transmitted without encryption. It analyzes the headers

and payloads of network packets to determine if they are using unencrypted protocols such as HTTP, FTP, Telnet, or SMTP. One common way to detect unencrypted protocols is through the use of network sniffers or packet sniffers. These tools capture and analyze network traffic in real-time, allowing them to identify any unencrypted protocols that are being used. Another approach is to use intrusion detection systems (IDS) or intrusion prevention systems (IPS). These systems can monitor network traffic and alert network administrators if they detect any unencrypted protocols being used. In addition, some firewalls have the ability to inspect network traffic and block any unencrypted protocols from being transmitted. This can help to prevent sensitive information from being transmitted in plain text over the network.

Overall, detecting unencrypted protocols is an important aspect of network security. By identifying and blocking unencrypted protocols, organizations can better protect their sensitive information from being intercepted and compromised by malicious actors.

This function lists unencrypted protocols by executing a shell command that uses the netstat utility. The command searches for network connections that are not using HTTPS and are using HTTP, Telnet, FTP, or TFTP (port 69).

The function is named list_unencrypted_protocols() and takes no arguments.

The function returns a string that contains the result of the shell command executed using os.popen().

The shell command uses netstat -a to list all network connections and pipes the output to grep to filter out any connections that are using HTTPS.

The filtered output is then piped to grep again to search for connections using HTTP, Telnet, FTP, or TFTP.

The final result is a string that lists the unencrypted protocols found by the shell command.

Here is a sample of the ouput:

Figure 52: sample of the output unencrypted protocols detector

5. System Development

This chapter will include the software specifications and the system's design with detailed information about our system including functional requirements, non-functional requirements, user requirements, system architecture, use case, data design following the logic in the SRD document.

5.1 Functional Requirements

Correctly defining the requirements that our work is asked to satisfy, is one of the most important steps during our project development. During this chapter, we will analyze requirements by describing those that are functional and those that are non-functional. Then in a second section, we will define the structure and the dynamic behavior of the system, by introducing the interaction between its different actors and the entire system.

Functional requirements are directly related to system services. It helps you capture files of the system's intended behavior. This behavior can be expressed as functions, services, tasks, or any system required to perform. In this subsection, we list the functions that are required in our system.

5.1.1 Business

• The system should be 85% to 95% accurate.

5.1.2 Authentication

- The signing up of all users would be done by the admin panel. To sign up for the system, the basic information of the user would be required and will be added to the system.

5.2.3 Dashboard

- In the dashboard, there is information about the system functions and features and about the system developer.
- There is also icon “Manage Users” where can find, create and delete users.
- There is also side-navigation bar with each feature in the system so that user can navigate between them easily.

5.2.3 Attack Detection

- In this page, there are 2 buttons one for start IDS and another for restart IDS.
- On clicking start IDS, the system will show in Sniffer section each captured and prepared packet.
- At the same time, in the Detection section there will be a message if each packet normal or it represents an attack.
- If the any packet represents an attack, there will be recommendations steps with links to deal with this attack based on the type of attack.
- There are also 2 LEDs, one green for normal and on red for attack happening.

5.2.4 Arp Poisoning Detector

- In this page, there are input field with label “Network Interface” to determine the interface where the function does its job.
- On clicking scan, the system will show message for loading scan, then the system will show message if Arp table is safe or poisoned.

5.2.5 Privilege Escalation Detector

- In this page, there is an input field with label “Enter your ‘PATH’” to determine the path where the function does its job.
- On clicking scan, the system will show message for loading scan, then the system will show information about “writable passwd file”, “readable or writable shadow file”, “readable or writable sudoers file” all is safe or not safe.

5.2.6 Passwd File Monitor

- On navigating to this page, there is an upper section to show the current passwd file.
- On clicking scan, the system will show message whether any change was made to this file and if any change was made, it will show the change, the file before change and file after the change.

5.2.7 Tmp Directory Monitor

- On navigating to this page, there is an upper section to show the current Tmp Directory.
- On clicking scan, the system will show message whether any change was made to this file and if any change was made, it will show the change, the file before change and file after the change.

5.2.8 History File Checker

- In this page, there is an input field with label “User Path” to determine the path where the function does its job.
- On clicking scan, the system will show all commands and tools the user type.

5.2.9 Port Blocker

- In this page, there are 2 input fields with label “Port” and “Interface” to determine the port required to block or unblock.
- On clicking block, the system will show message that port was blocked.
- On clicking unblock, the system will show message that port was opened.

5.2.10 Services Scanner

- In this page, there are button to scan all services in the system.
- On clicking scan, the system will show all services on the system with information like user and process ID.

5.2.11 Devices on the Network Scanner

- In this page, there are button to scan all Devices on the network.
- On clicking scan, the system will show all devices on the network with information like hardware connection type and the MAC address.

5.2.12 Malicious Files Detector

- In this page, there is an input field with label “File Path” to determine the path where the function runs.
- On clicking scan, the system will show all malicious files found in that path.

5.2.13 Unencrypted Protocols Detector

- On clicking scan, the system will show unencrypted protocols found in that network.

5.2 Nonfunctional Requirements

Following are the non-functional requirements of our system:

1. **Security**: The user's data that would be required during sign up should be confidential, safe, and secure.
2. **Performance**: The system should less than a minute to process the image and displays the results.
3. **Compatible**: The system should be compatible with all browsers and Linux distributions.
4. **Usability**: The interface of the system should be simple, and flow of app should be understandable by anyone that non-technical person.
5. **Scalability**: The system should be scalable so that it can later be used in real world or at the industry level.
6. **Maintainability**: The system should be easily maintainable so that it can become more advance and efficient when use in industry.
7. **Documentation**: The solution should be well-documented in order to provide the best use for the customer.

5.3 Use Cases

The use case diagram captures the behavior of a software system. This diagram describes the various actions, which can be made by an outside user. It splits the feature of the system into coherent units, the use cases, having a sense for the actors. This section presents detailed use cases of our system. The **figure** shows the general use case diagram that includes the common functionalities for all components.

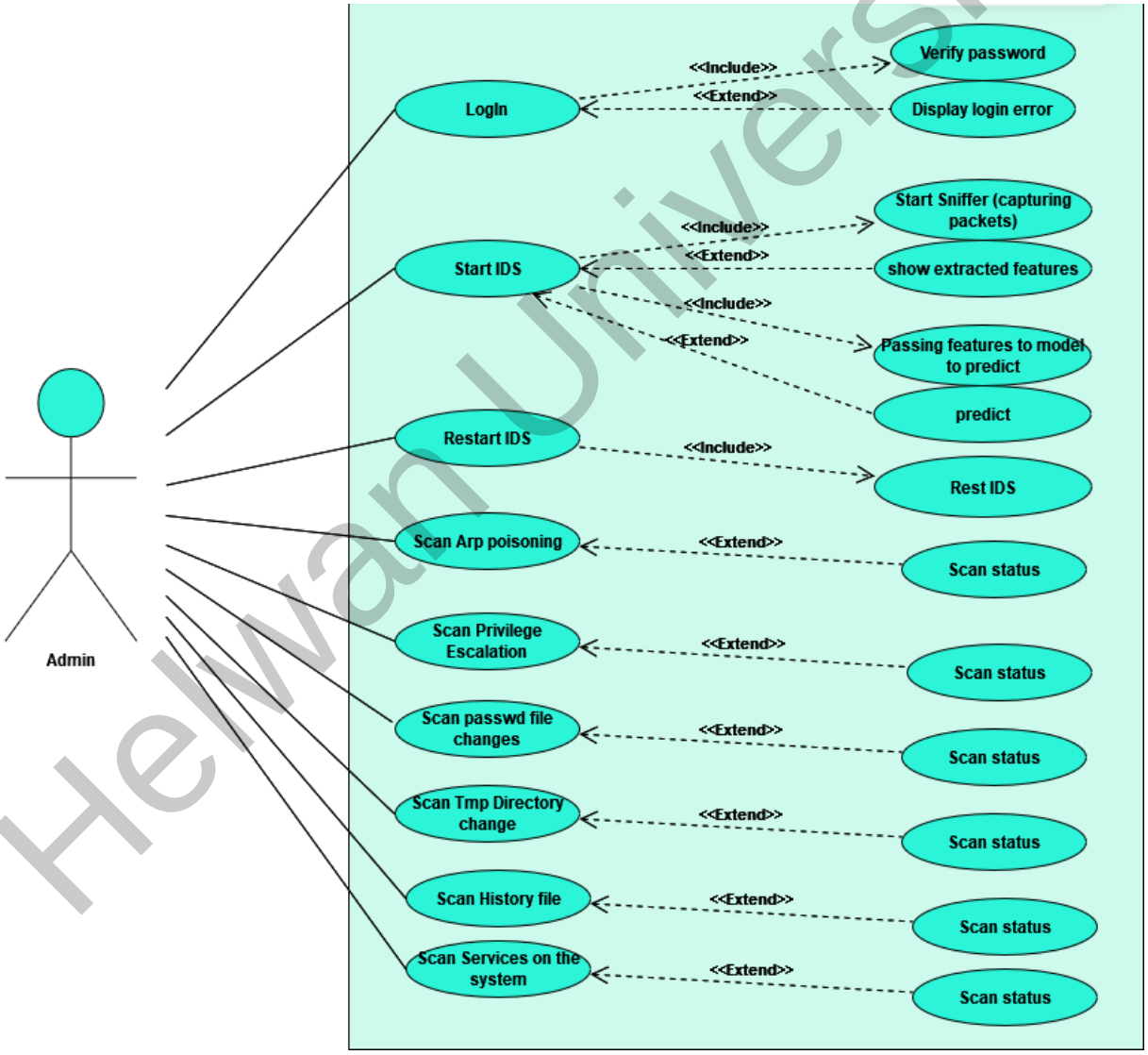


Figure 53: Use case diagram

5.4 Data Design

This section presents the different diagrams for clarifying our system data design. We will represent in detail the UML class diagram to present our database structure and relationships between entities. UML class diagram and gives a brief description of each class in our system. Attributes and methods of each class and relationships among classes are clearly presented. Our project contains the following classes shown in figure .

- **Arp:** This class represents a main function to detect arp spoofing.
- **PacketSniffer:** This class represents a sniffer that capture data from the traffic.
- **User:** This class represents a generic user in our system. The class encapsulates information like username and password for a particular instance of a user.

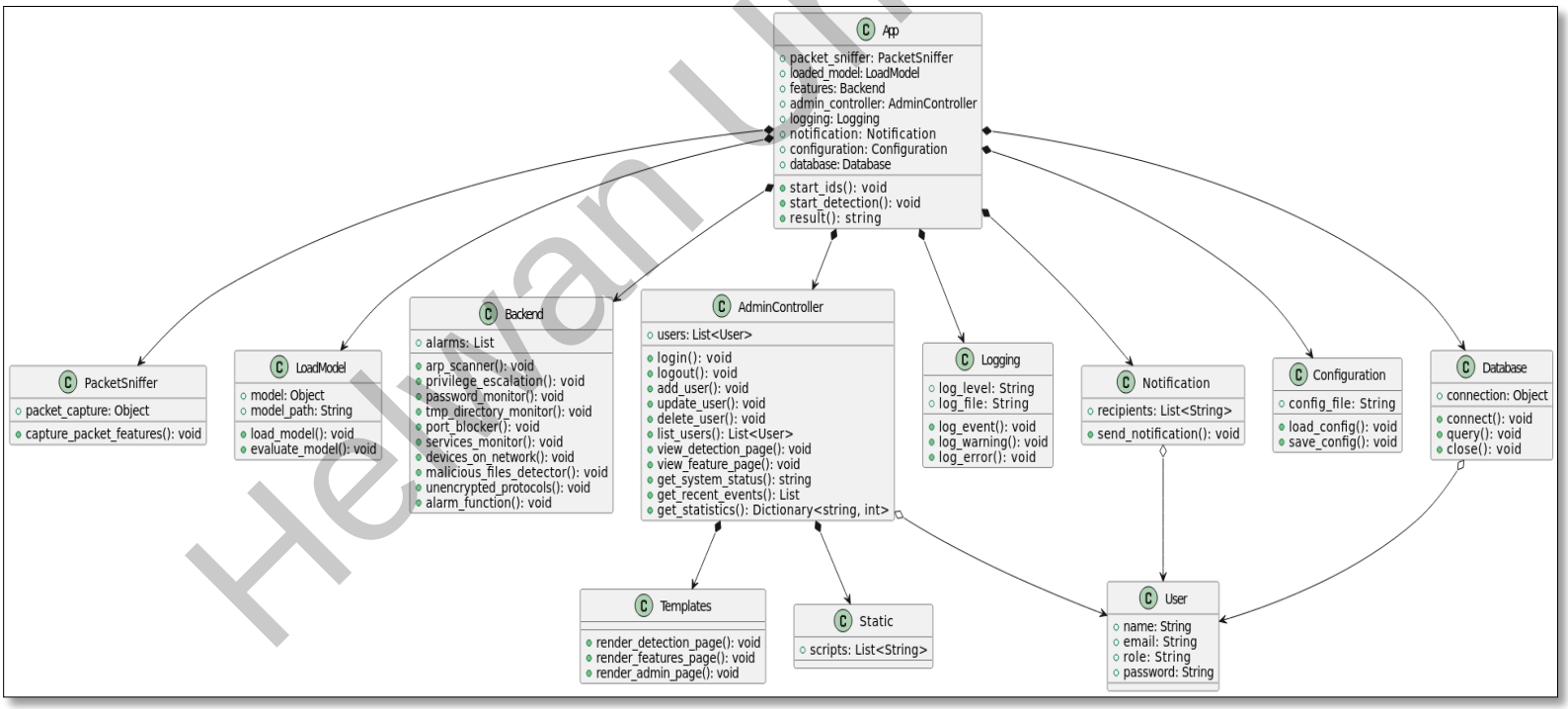


Figure 54: Class diagram

5.5 Architecture Design

The software architecture is structured around main 4 layers. Every layer presents a different psychical level of application structure. Figure 4.4 shows the architecture diagram of our system.

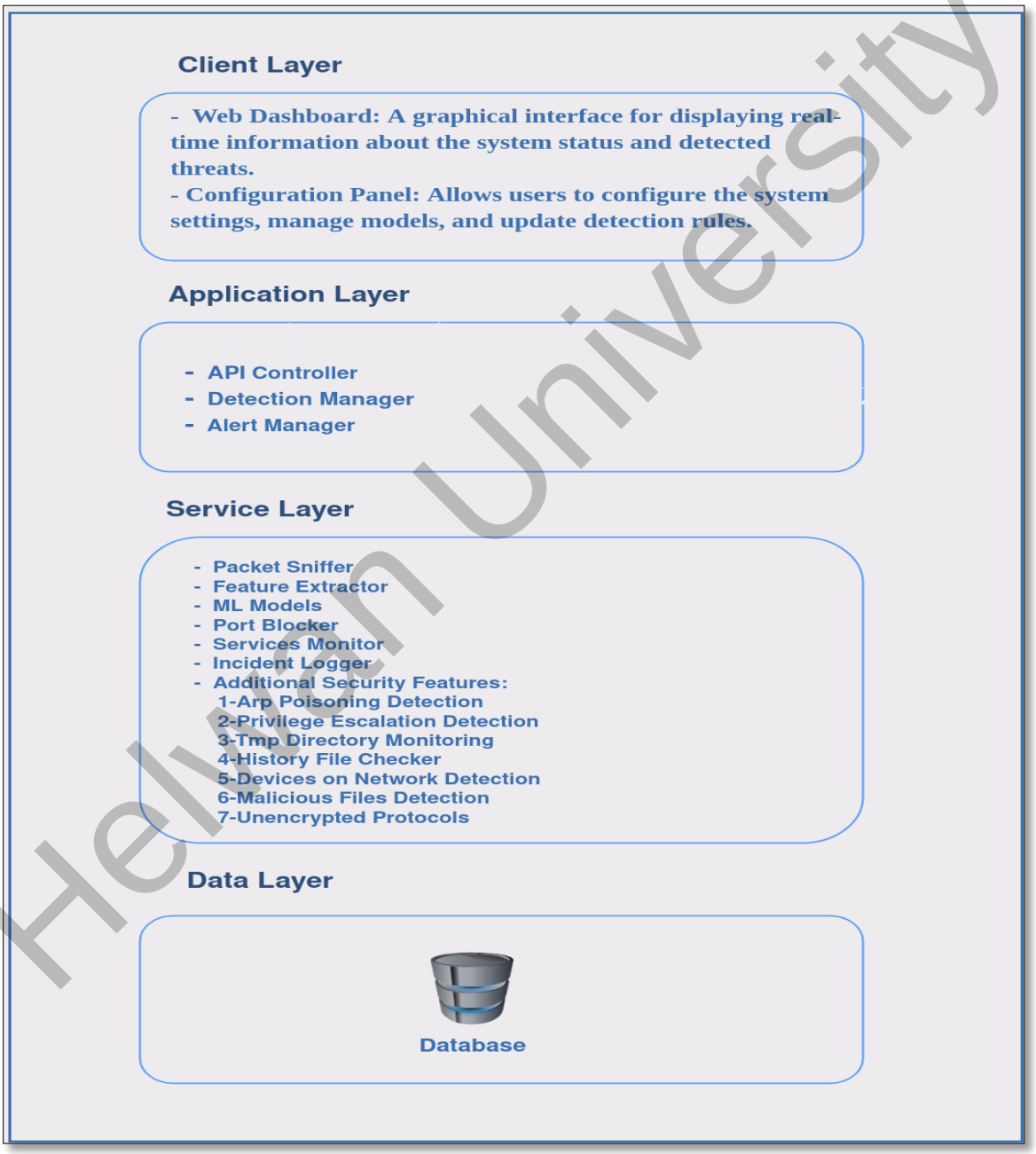


Figure 54: Architecture diagram

6. System Testing and Verification

This chapter introduces you to the testing setup and how we tested each module in isolation and the end-to-end integration that we planned and followed to prove to a high certainty the correctness of the system.

Testing of the project is done to check whether the actual results match what is expected, ensure that each module is doing what it is intended to do, and to ensure that the system as a whole is working correctly. Our project depends on real world manuscript document images, so, most of our testing was done through testing the modules on different images, and checking whether the output is satisfactory compared to the results of similar projects. In the end, we test the workflow of the project to check whether the results are clear enough to interpret or not.

6.1 Testing Setup

6.1.1 power on virtual machines

Because our project targets enterprises' security and issues, it's based on Linux. So, for testing we set up 2 virtual machines: one as an attacker to perform type of attack and another is victim that has IDS installed in it. We chose Kali Purple as the machine to test. Kali Purple is an operating system designed and developed by Offensive Security for defending and responding to cyber threats.

Note: any Linux distribution can be used.

- To download and install VMware you can follow these steps in this link:

  https://kb.vmware.com/s/article/2057907

- To create virtual machine in VMware you can follow these steps in this link:

  https://kb.vmware.com/s/article/1018415

- Then you can choose any Linux distribution and install it.

6.1.2 Setup Web Server

Since the web server is built using Flask to accelerate the development of the web application and the API (Application Programming Interface), we will install the flask framework and other libraries that depend on it in this section. Also, we will set up a machine learning environment for detecting attack and recognize of its type.

6.1.2.1 Installing Required Libraries

Run the following:

$ python --version

If it's 3.x, you are good to go with installing the other libraries. The libraries are grouped into a text file to install all of them once by one command. Here is a sample of file requirements.txt file content:

Flask==2.2.2

Flask-Cors==3.0.10

Flask-JWT-Extended==4.4.4

Flask-SocketIO==5.3.3

Flask-SQLAlchemy==3.0.3

SQLAlchemy==1.4.46

sqlparse==0.4.2

keras==2.12.0

numpy==1.23.5

sklearn==0.0.post4

pandas==1.5.3

scapy==2.5.0

scikit-learn==1.2.2

scipy==1.10.0

colored==1.4.4

vt==3.1.3.7

Once you copied and created this text file, run the following:

$ pip install -r requirements.txt

6.1.1.2 Start Script

After installing all required libraries, run the following script to start flask app to start the web application and the API.

$ sudo /bin/python3 /path/to/IDS/app/app.py

6.2 Testing Plan and Strategy

Our main goal in evaluating each module in IDS was to make sure it met all functional and non-functional requirements that were specified earlier. Testing is almost as challenging as development.

6.2.1 Attack Detection Testing

We follow a separate module testing to test IDS. For each module, we take the output of the previous module and test the output resulting from the module. We perform one sample Dos attack shown in figure 5.1 to forward test each module on it separately.

6.2.1.1 Sniffer Testing

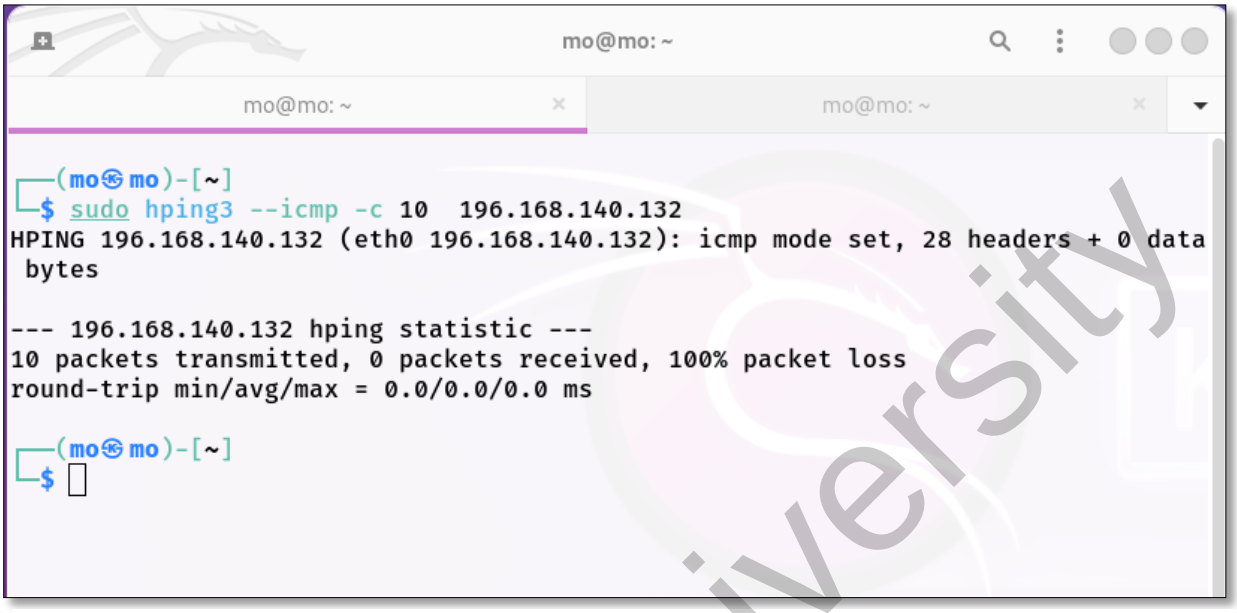


Figure 56: Packet sniffing

We built this module using pyshark. In this module, the system capture packet and extract specific features then it makes preparation on it. Here samples of data in each part:

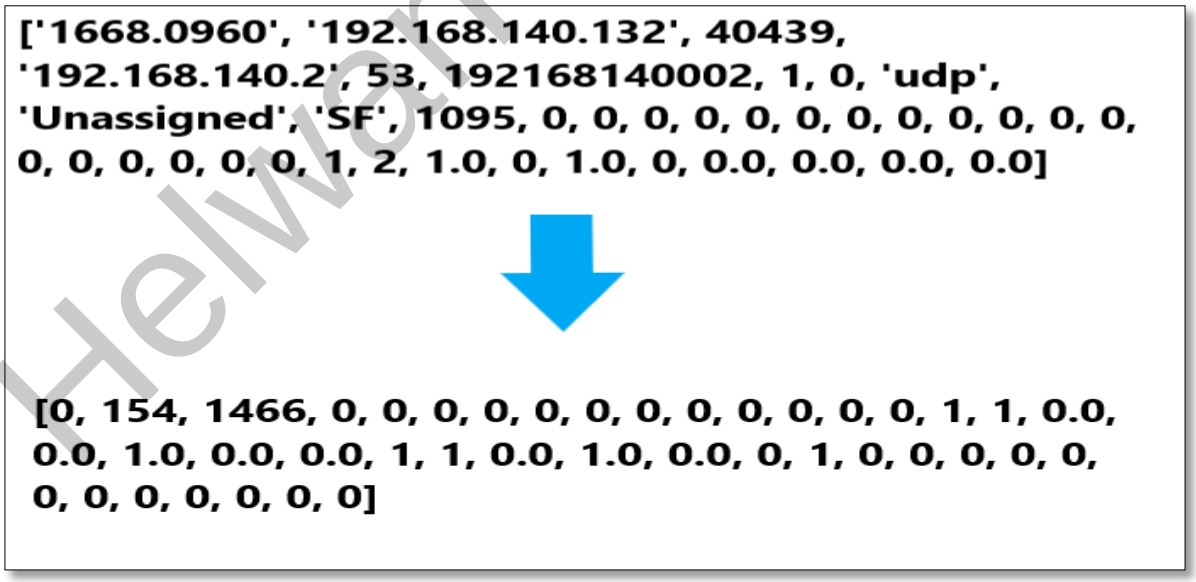


Figure 57: Convert categorical variables into numerical

6.2.1.2 Model Detection Testing

In this module the system takes the previous suitable list then passes it to the machine learning model to predict if it's normal or malicious then classify if it **Dos**, **R2L**, **U2R** or **Probes** attack. Here is sample of the output:

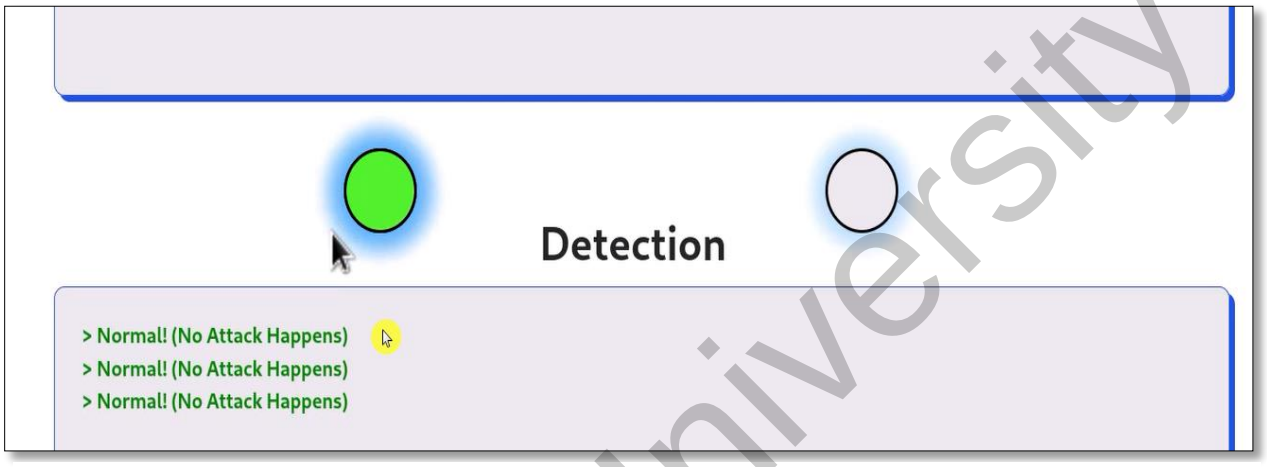


Figure 58: Output of model detection 1

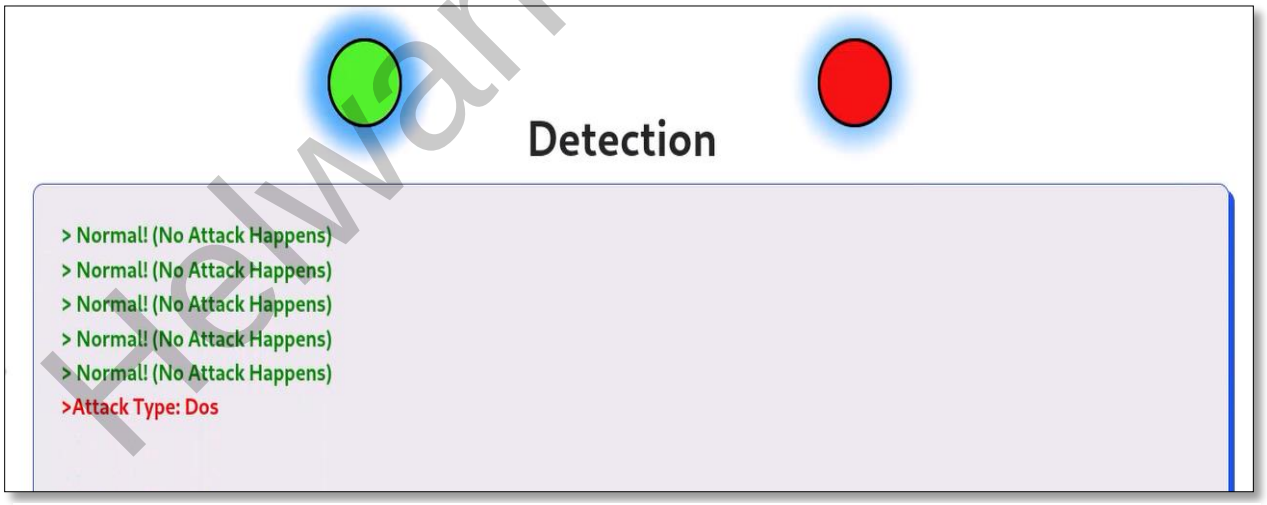


Figure 59: Output of model detection 2

6.2.2 Arp Poisoning Detector Testing

In this module, we perform Arp spoofing attack as a test then watch if can detect

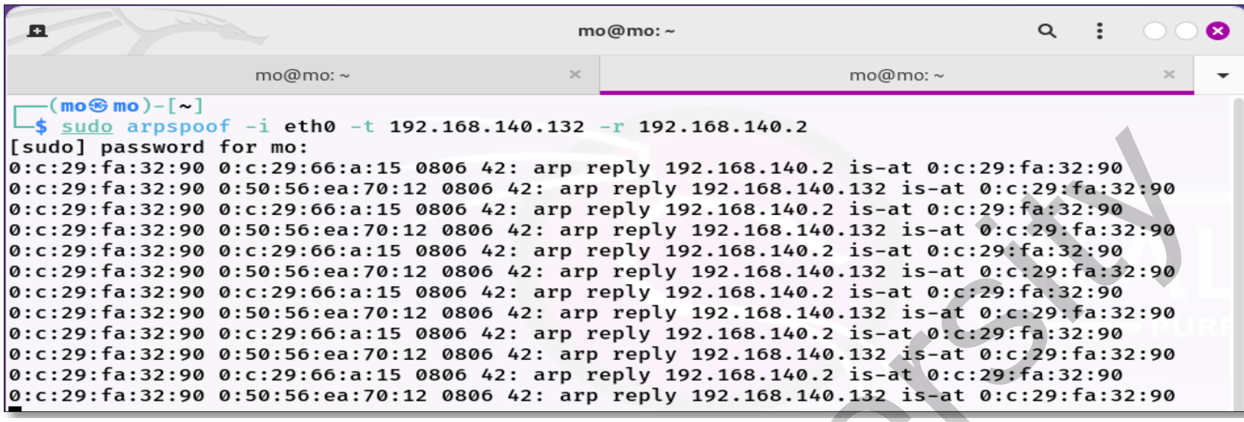


Figure 60: Arp Poisoning Detector

Figure 61: Output of Arp Poisoning Detector 1

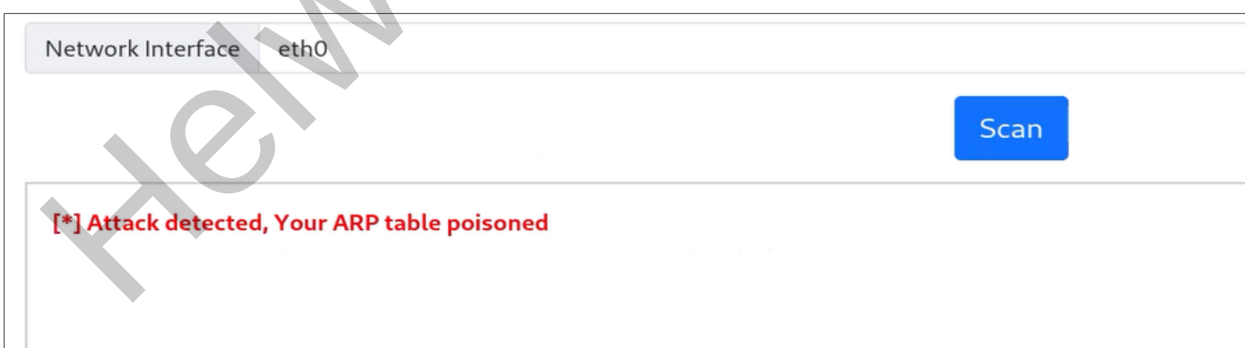


Figure 62: Output of Arp Poisoning Detector 2

6.2.3 Privilege Escalation Detection Testing

In this module, we test if any Privilege Escalation happened:

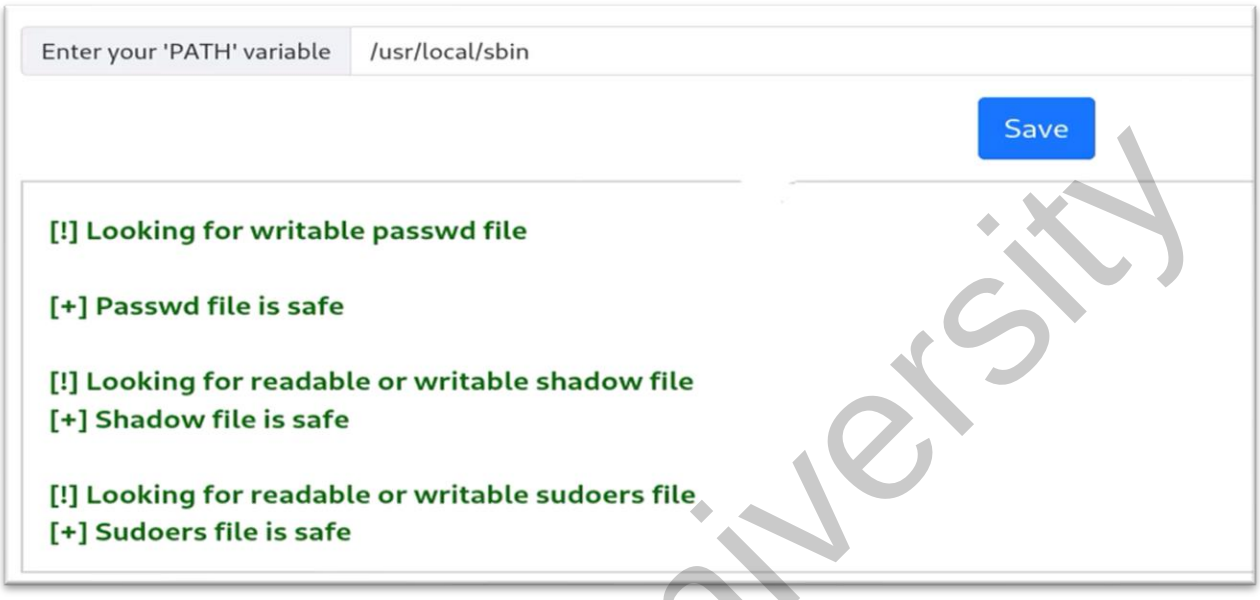


Figure 63: Output of Privilege Escalation Detection 1

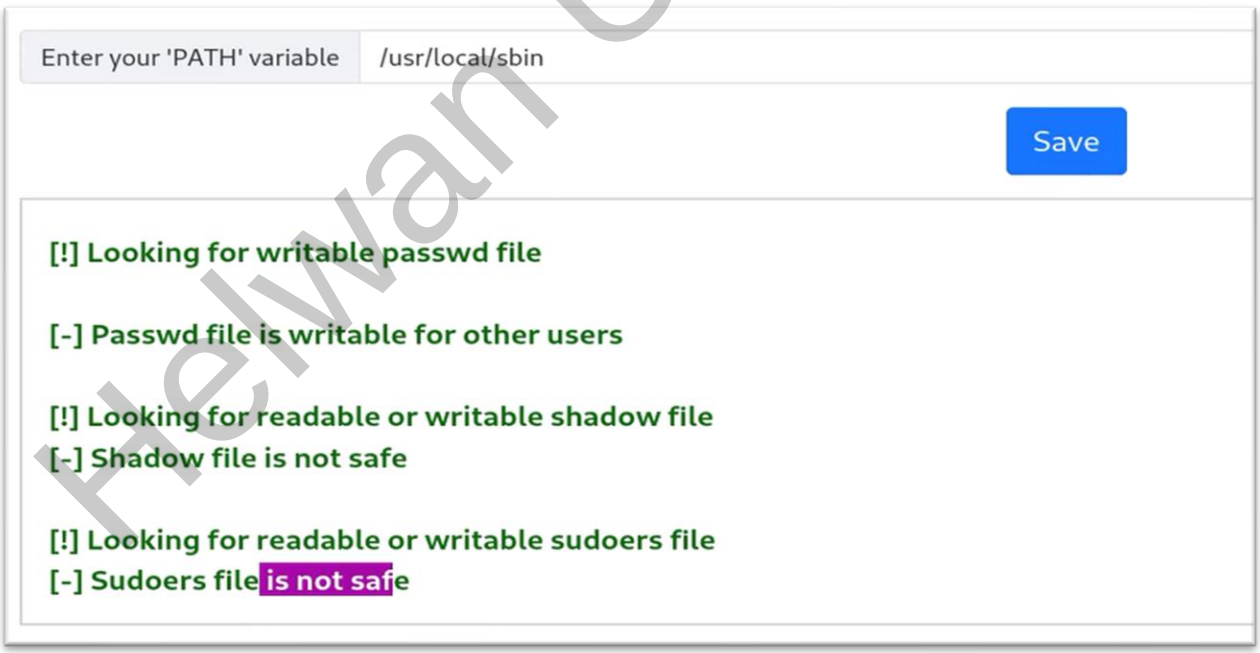


Figure 64: Output of Privilege Escalation Detection 2

6.2.4 Passwd File Monitor Testing

In this module, we test if any change happened to Passwd File:

Figure 65: Output of Passwd File Monitor

6.2.5 Tmp Directory Monitor Testing

In this module we test if any change happened in Tmp Directory:

```
[!] tmp Directory has Modified.
[>] Different Changes:
test1.txt

[>] Directory Before Change:
systemd-private-e02f453c268148f4aa4b4e94a2226210-systemd-logind.service-pCO0xl
wireshark_-KNWZ31.pcapng
.X11-unix
ssh-XXXXXX7chTd9
[>] Directory After Change:
systemd-private-e02f453c268148f4aa4b4e94a2226210-systemd-logind.service-pCO0xl
wireshark_-KNWZ31.pcapng
.X11-unix
ssh-XXXXXX7chTd9
systemd-private-e02f453c268148f4aa4b4e94a2226210-upower.service-MPGYuM
systemd-private-e02f453c268148f4aa4b4e94a2226210-colord.service-FUblSL
pyright-2817-anNT5nKVytxL
wireshark_-JZW331.pcapng
wireshark_-VWL131.pcapng
```

Figure 66: output of Tmp Directory Monitor

6.2.6 History File Monitor Testing

In this module, we test history of commands:

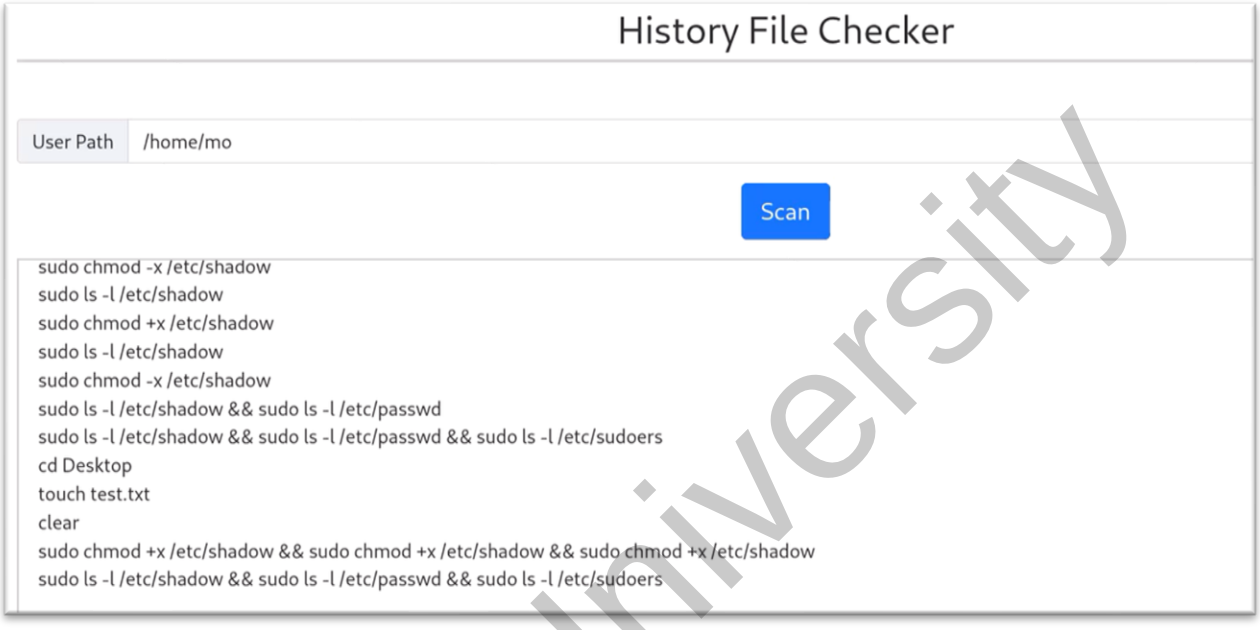


Figure 67: Output of History File Monitor

6.2.7 Port Blocker Testing

In this module, we test port blocking and unblocking:

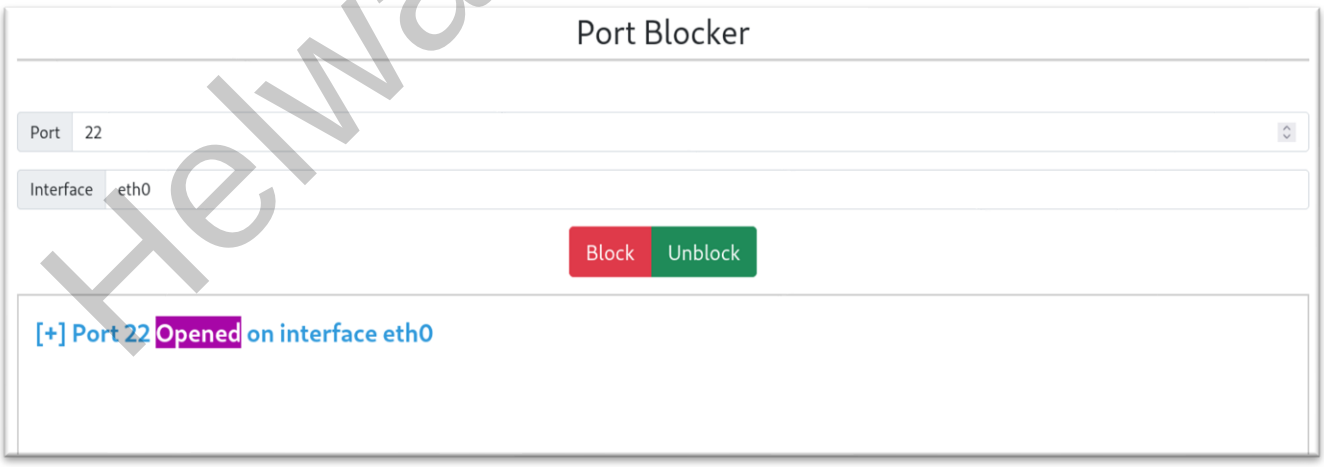


Figure 68: Output of Port Blocker 1

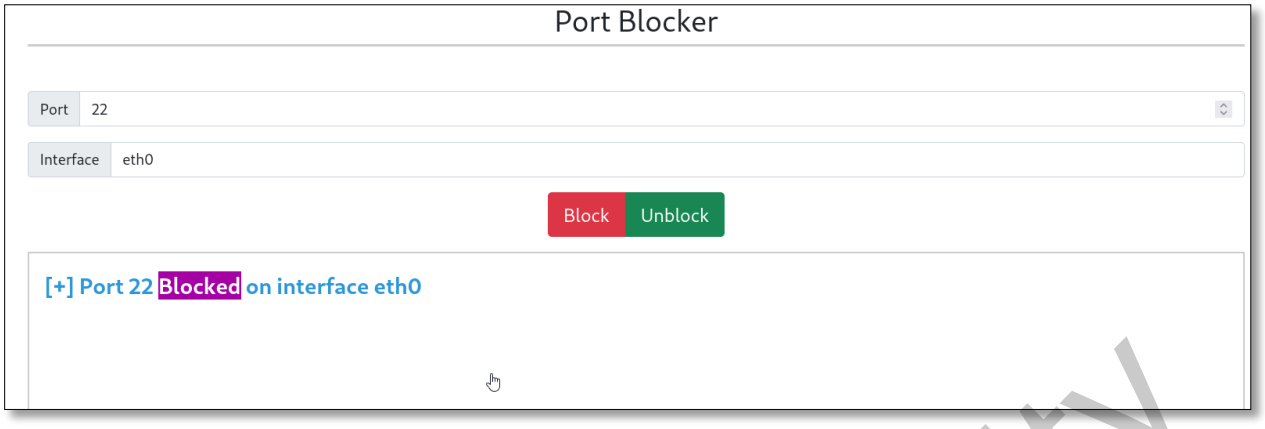


Figure 69: Output of Port Blocker 2

6.2.8 Services Scanner Testing

In this module, we test scanning services on the system:

Figure 70: Output of Services Scanner

6.2.9 Devices Monitor Testing

In this module we test devices monitoring on the network:

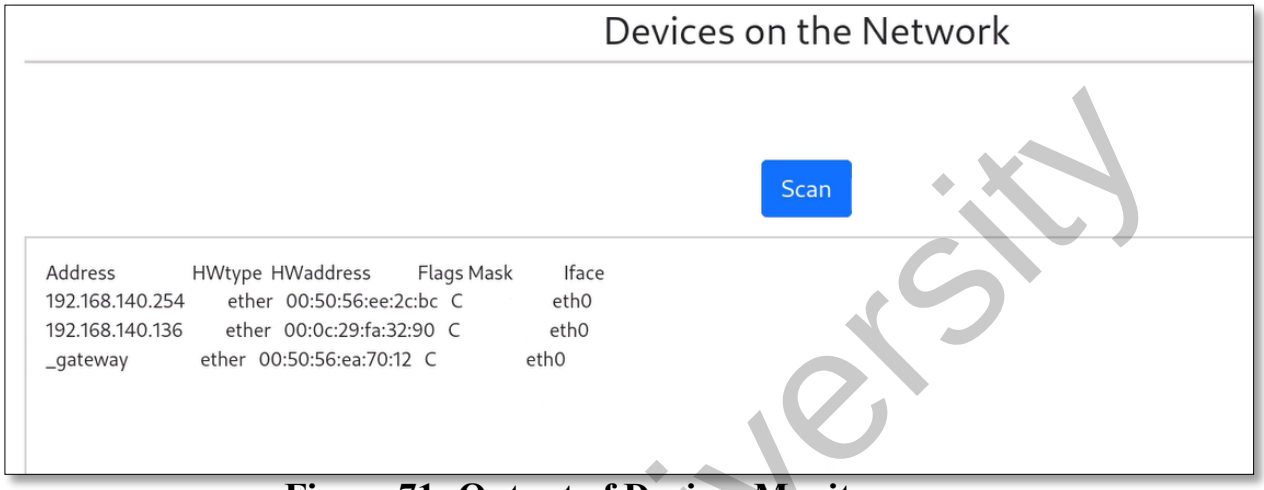


Figure 71: Output of Devices Monitor

6.2.10 Malicious Files Detector Testing

In this module, we test detecting malicious files on the system:

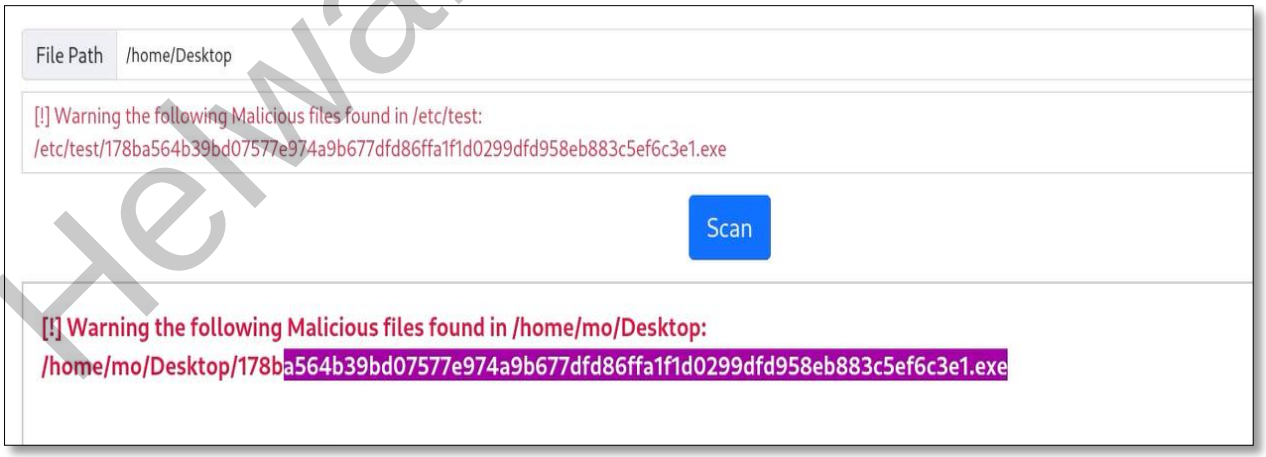


Figure 72: Output of Malicious Files

6.2.11 Unencrypted Protocols Detector Testing

In this module, we test detecting Unencrypted Protocols in the network:

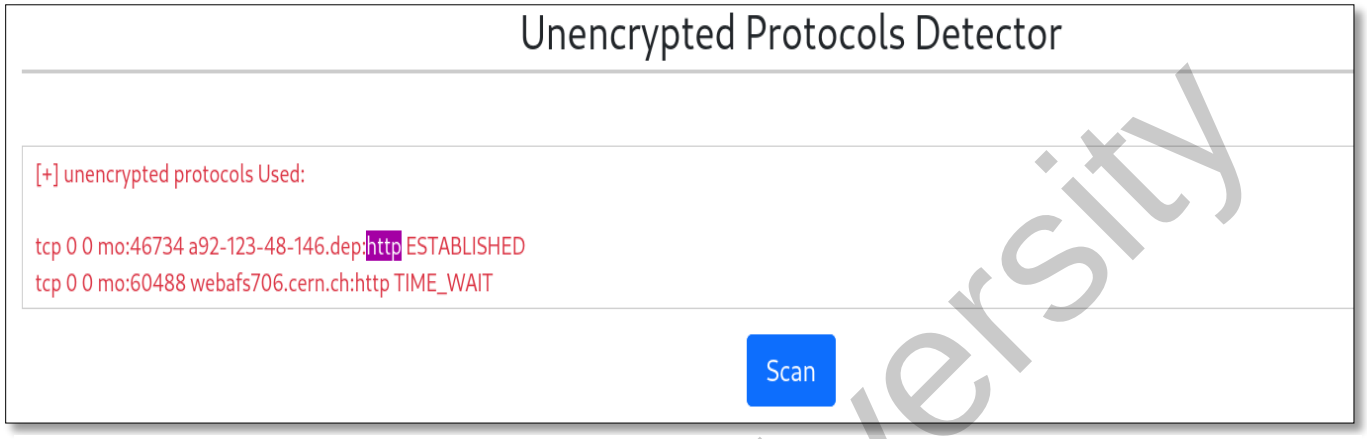


Figure 73: Output of Unencrypted Protocols Detector

7. Conclusions and Future Work

We have introduced IDS project as a software solution detecting cyber attacks and malicious activities by using deep learning techniques. In this chapter, we let you catch a glimpse of how our journey looked like. We describe the challenges that we faced, and the lessons that we learned, and summarize what we achieved. Finally, we discuss what the future may look like.

7.1 Faced Challenges

7.1.1 Data Preparation

Since we have KDDcup99 dataset that contains specific features used to detect attacks, we face a deep challenge to build sniffer capture packet and extract that features form this packet. And, because we wanted to build our project in python, we had to make sniffer in python, which was hard to do and little resources, but we reached to pyshark, a Python wrapper for the Wireshark network protocol analyzer allows to capture, dissect and analyze network packets in real-time or from a saved capture file, and we build that sniffer in big effort.

7.1.2 Resources

IDS using Neural Network didn’t apply before. So, it was difficult to find any previous experiments. Also, the practical resources were rare and not up to date, so we only depended on the mentioned scientific papers in the references section to understand each module and how it works.

7.1.3 Testing the project

Since our project targets the cyber security field, testing functions of project required performing real attacks and bringing malicious files which is very dangerous for systems and hard to find testing samples like that.

7.1.4 Model Training

The IDS model used is a large complicated one that needs high-resolution data of large amounts, and these constraints make it harder to train on the available resources we have. It needs a fast GPU of large memory size, preferably 16GB, which we didn't have. So, we used google Colab cloud service.

7.2 Gained Experience

Working on developing ASAR has been an enriching experience that we learned from in different ways:

- Researching and reading dozens of papers to achieve working results helped us to be exposed to multiple methods in every sub module, and to see the evolution in the techniques through time.
- Working with different environments and integrating them together to build an intelligent system. Learning the process of end-to-end deep learning project linked with web application.
- We learned several tools and technologies, as described in the appendix A section, and learned to work on a remote server to train models on.
- Teamwork, effective collaboration, and asynchronous online communications.

7.3 Conclusions

Throughout this document, we demonstrated the idea of our project in order to and detecting cyber-attacks and malicious activities, and the steps applied when capturing packets, extraction the features, preparing them then passing them to model to predict Also, we showed the results for each module and developed web applications to consume and integrate them together.

7.4 Future Work

The model has some drawbacks and little percent of non-accuracy, so we need to enhance it and work and test new algorithms to get to high level in accuracy and we have shortage of records in datasets in the categories: Remote to Local (R2L) and User to Root Attacks (U2R) so we need to train the model on more records from this category.

There are also other features like Arp spoof detector, Privilege Escalation detectors, Malicious files detector and Unencrypted protocols detector, these features we can make use of AI to build accurate models to detect instead of normal detection.

Our application is a main web linux application, so as a future work, we can build version for windows or build version desktop.

Finally, most important future work feature is to make our IDS is hybrid by convert it into IPS too so that it can respond and react to threats automatically, but this feature needs more work to achieve the balance between the speed of traffic and make active responds to incidents.

Bibliography

[1] ABDULLAH, M.; ALSANEE, E.; ALSEHEYMI, N. *Energy Efficient Cluster-Based Intrusion Detection System for Wireless Sensor Networks*. International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 5, No. 9, 2014, 10-15.

[2] ALLEN, J.; CHRISTIE, A.; FITHEN, W.; MCHUGH, J.; PICKEL, J. *State of the practice of intrusion detection technologies*. Carnegie Mellon University and Software Engineering Institute, 2000.

[3] AXELSSON, S. *Aspects of the modelling and performance of intrusion detection.* Department of Computer Engineering, Chalmers University of Technology, 2000.

[4] BEKKAR, M.; DJEMAA, H. K.; ALITOUCHE, T. A. *Evaluation Measures for Models Assessment over Imbalanced Data Sets*. Information Engineering and Applications, Vol.3 No.10, 2013, 27-39.

[5] BHUYAN, M. H.; DHRUBA, K. B.; JUGAL, K. K. *Network anomaly detection: methods, systems and tools*. IEEE communications surveys & tutorials.2014, Vol. 16, No. 1, 303-336.

[6] BHUYAN, M. H.; DHRUBA, K. B.; JUGAL, K. K. *Towards Generating Real-life Datasets for Network Intrusion Detection.* IJ Network Security, 2015, Vol. 17, No.6, 683-701.

[7] CASWELL, B.; BEALE, J. *Snort 2.1 intrusion detection*. 2nd edition, Syngress, 2004, 700.

[8] CHUNG, Y. Y.; WAHID, N. *A hybrid network intrusion detection system using simplified swarm optimization (SSO)*. Applied Soft Computing, Vol. *12*, No. 9, 2012, 3014-3022.

[9] DAOUDI, M.; BOUKRA, A.; AHMED-NACER, M. *Security audit trail analysis with biogeography based optimization metaheuristic*. Informatics Engineering and Information Science. 2011, 218-227.

[10] DEVIKRISHNA, K. S.; RAMAKRISHNA, B. B. *An artificial neural network based intrusion detection system and classification of attacks*. International Journal of Engineering Research and Application (IJERA). 2013, Vol. 3, No. 4, 1959-1964.

[11] DIAZ-GOMEZ, P. A.; HOUGEN, F. *Improved Off-Line Intrusion Detection Using a Genetic Algorithm*. In International Conference on Enterprise Information Systems (ICEIS). 2005, Vol.2, 66-73.

[12] DINAKARA, K. *Anomaly based Network Intrusion Detection System*.Indian Institute of Technology. 2008.

[13] EESA, A. S.; ORMAN, Z.; BRIFCANI, A. M. A. *A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems*. Expert Systems with Applications, Vol. *42*, No. 5, 2015, 2670-2679.

[14] FAROOK, S. M.; REDDY, K. N. *Implementation of Intrusion Detection Systems for High Performance Computing Environment Applications.* International Journal of Scientific Engineering and Technology Research (IJSETR). Vol. 04, No. 41, 2015, 8958-8963.

[15] GHORBANI, A. A.; Lu, W.; TAVALLAEE, M. *Network intrusion detection and*

prevention: concepts and techniques. Springer Science & Business Media. Vol. 47 , 2009.76
[16] GUPTA, M.; SHRIVASTAVA, S. K. *Intrusion Detection System based on SVM and Bee Colony*. International Journal of Computer Applications, Vol.111, No.10, February 2015, 27-32.
[17] HOQUE, M. S.; MUKIT, M.; BIKAS, M.; NASER, A. *An implementation of intrusion detection system using genetic algorithm*. International Journal of Network Security & Its Applications (IJNSA), Vol.4, No.2, March 2012.
[18] JOSHI, P. *Soft computing and classification approach to Anomaly Based intrusion detection system: A Survey.* International Journal of Emerging Trends and Technology in Computer Science (IJETTCS). 2017, Vol. 6, No. 5, 27-34.
[19] KENDALL, K. *A database of computer attacks for the evaluation of intrusion detection systems.* Department of Defense Advanced Research Projects Agency, 1999. 1-124.
[20] LAKSHMI, T. V.; BABU, V. K. *Detection of User to Root Attacks using Machine Learning Techniques.* International Journal of Advanced Engineering and Global Technology (IJAEGT), 2015, Vol.3, No.3, 418-424.
[21] LEVIN, I. *KDD-99 classifier learning contest: LLSoft's results overview.* SIGKDD explorations. Vol.1, No. 2, 2000, 67-75.
[22] LIN, W. C.; KE, S. W.; TSAI, C. F. *CANN: An intrusion detection system based on combining cluster centers and nearest neighbors*. Knowledge-based systems, Vol. *78*, 2015, 13-21.
[23] LIPPMAN, R. P.; FRIED, D. J.; GRAF, I.; HAINES, J. W.; KENDALL, K. R.; MCCLUNG, D.; ZISSMAN, M. A. *Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation.* In DARPA Information Survivability Conference and Exposition, 2000 , Vol. 2, 12-26.
[24] MEWADA, H. K.; PATEL, S. *Advances in Intrusion Detection Algorithms for Secure Ebusiness Using Artificial Intelligence.* Research Journal of Information Technology. 2017, Vol. 9, No. 1, 1-6.
[25] MORADI, M.; ZULKERNINE, M. *A neural network based system for intrusion detection and classification of attacks*. In Proceedings of the IEEE International Conference on Advances in Intelligent Systems-Theory and Applications. 2004, 15-18.
[26] PAPPAS, N. *Network IDS & IPS Deployment Strategies*. SANS Institute, 2008.
[27] PLANQUART , J. P. *Application of neural networks to intrusion detection.* SANS Institute. 2001.
[28] PRADHAN, M.; PRADHAN, S. K.; SAHU, S. K. *A survey on detection methods in intrusion detection system*. International Journal of Computer Engineering, Vol. 3, No. 2, 2012, 81-90.
[29] RAJ, A.A. *A Study on Data Mining Based Intrusion Detection System*. International Journal of Innovative Research in Advanced Engineering (IJIRAE). 2014, Vol. 1, No. 1.
[30] RHODES, B. C.; MAHAFFEY, J. A.; CANNADY, J.D. *Multiple self-organizing maps for intrusion detection*. In Proceedings of the 23rd national information systems security conference. 2000, 16-19. 77

[31] SABHNANI, M.; SERPEN, G. *KDD Feature Set Complaint Heuristic Rules for R2L Attack Detection.* In: Security and Management. 2003, 310-316.
[32] SHAKYA, S.; KAPHLE, B. R. *Intrusion Detection System Using Back Propagation Algorithm and Compare its Performance with Self Organizing Map*. Journal of Advanced College of Engineering and Management. 2016, Vol. 1, 127-138.
[33] SHYU, M.; et al. *Handling Nominal Features in Anomaly Intrusion Detection Problems*. In Proc. of the 15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications, April 03 - 04 2005, 55-62.
[34] SIDDIQUI, M. A. *High performance data mining techniques for intrusion detection.* Engineering and Computer Science. 2004.
[35] STOLFO, S. J.; FAN, W.; LEE, W.; PRODROMIDIS, A.; CHAN, P. K. *Cost-based modeling for fraud and intrusion detection: Results from the JAM project*. In: DARPA Information Survivability Conference and Exposition, 2000, 130-144.
[36] TOOSI, A. N.; KAHANI, M. *A novel soft computing model using adaptive neuro-fuzzy inference system for intrusion detection*. In Networking, Sensing and Control, 2007 IEEE International Conference on (pp. 834-839).IEEE.
[37] UGTAKHBAYAR, N.; USUKHBAYAR, B.; NYAMJAY, J. *An approach to detect TCP/IP based attack*. International Journal of Computer Science and Network Security (IJCSNS), Vol. 16, No. 4, 2016, 37-40.
[38] WHITE, G.; CONKLIN, W. A.; WILLIAMS, D.; DAVIS, R.; COTHREN, C. *Principles of Computer Security, CompTIA Security+ and Beyond*. 3rd edition, McGraw-Hill, 2012, 684.
[39] YANG, G. *Introduction to TCP/IP Network Attacks.* Secure Systems Lab, 1997.
[40] YOUSIF, H. M. *Wireless Intrusion Detection Systems*. Diss. College of Science Wireless Intrusion Detection Systems, Baghdad University, 2011.
[41] ACM, https://www.acm.org/ . 12/5/2015
[42] CAIDA, http://www.caida.org/home/ . 11/2/2016
[43] DARPA 2000, https://www.ll.mit.edu/ideval/data/2000data.html . 1/12/2015
[44] DEFCON communication, https://www.defcon.org/ . 11/2/2016
[45] ISCX-UNB, http://www.unb.ca/cic/research/datasets/ids.html . 11/2/2016
[46] KDD Cup 1999 Task , http://kdd.ics.uci.edu/databases/kddcup99/task.html . 1/5/2015
[47] KDD Cup 1999 Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html . 1/5/2015
[48] LBNL/ICSI Enterprise Tracing Project, http://www.icir.org/enterprise-tracing/ .
[49] Linkoln Laboratory MIT, DARPA INTRUSION DETECTION EVALUATION, https://www.ll.mit.edu/ideval/docs/attackDB.html#ffb . 11/2/2016
[50] NSL-KDD dataset: http://www.unb.ca/cic/research/datasets/nsl.html . 11/2/2016
[51] SIGKDD, http://www.kdd.org/ 1/5/2015

Appendices

Appendix A

Development Platforms and Tools

A.1 Hardware Tools

Personal computers running windows or Linux-based operating systems that can hold virtual machines.

A.2 Software Tools

A.2.1 Programming Languages

A.2.1.1 Python

Python is a high-level programming language that is interpreted and object-oriented. It's ideal for Rapid Application Development, as well as using as a scripting or glue language to link together existing components.

Usage: To build model and the web application

A.2.1.2 JavaScript

JavaScript is a scripting or programming language used to create and control dynamic website content, and it allows you to implement complex features on web pages.

Usage: Used in web application for streaming and responsive output.

A.2.2 Libraries and Frameworks

A.2.2.1 TensorFlow

It is a symbolic math library, and is also used for machine and deep learning applications such as neural networks.

Usage: For building our model-based DNN.

A.2.2.2 Keras

Keras it is a software library that provides a Python interface for artificial neural networks.

Usage: For testing and training our model based DNN.

A.2.2.3 NumPy

Numpy it is a library for Python programming language, adding support for large, multidimensional arrays and matrices.

Usage: For preparing the data for training process.

A.2.2.4 Pandas

Pandas is a popular open-source Python library for data manipulation and analysis. It provides data structures for efficiently storing and processing large datasets, as well as tools for cleaning, transforming, and visualizing data.

Usage: For preparing the data for training process.

A.2.2.5 Flask

Flask is a web framework, it's a Python module that lets you develop web applications easily. It has a small and easy-to-extend core.

Usage: For building web applications and API.

A.2.2.6 Flask-Cors

Flask-Cors is a Python package that provides Cross-Origin Resource Sharing (CORS) support for Flask web applications. CORS is a mechanism that allows a web page to request resources from a different domain than the one from which the page was served. Without CORS, web pages can only make requests to resources on the same domain.

Usage: to request resources from a different domain

A.2.2.7 Flask-JWT-Extended

Flask-JWT-Extended is a Python package that provides JSON Web Token (JWT) support for Flask web applications. JWT is a widely used standard for representing claims securely between two parties, such as a web application and a client. JWT can be used for authentication, authorization, and exchanging data between different systems.

Usage: for representing claims securely between the web application and client.

A.2.2.8 Flask-SocketIO

Flask-SocketIO is a Python package that provides WebSocket support for Flask web applications. Web Sockets allow bidirectional communication between the web browser and the server, enabling real-time updates and event-driven applications. Flask-SocketIO allows you to build such applications with Flask and Socket.IO, a popular WebSocket library for JavaScript.

Usage: showing real-time updates in functions of system.

A.2.2.9 SQLite

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.

Usage: For building databases.

A.2.2.10 SQL Alchemy

SQL Alchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

Usage: For storing and manipulating the database.

A.2.2.11 Swagger

Swagger is a set of open-source tools for writing REST-based APIs. It simplifies the process of writing APIs by notches, specifying the standards and providing the tools required to write beautiful, safe, performant, and scalable APIs.

Usage: For documenting and handling APIs.

A.2.2.12 Scapy

Scapy is an open-source Python library for packet manipulation and network analysis. It allows users to create, send, and capture network packets, as well as analyze network traffic.

Usage: to build sniffers to capture packets.

A.2.2.13 Colored

Colored is a Python library that provides colored text output in the terminal. It allows users to add color and formatting to text output, making it easier to read and understand. The library supports a wide range of text styling options, including foreground and background colors, bold, underline, and italic styles.

Usage: to make some statements colored in console.

A.2.3 Tools and Platforms

- Jupyter Notebook: it is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media.
- VSCode: is a free and open-source code editor developed by Microsoft. It provides a wide range of features and extensions for programming and development.
- Google Colab: it is a cloud service for accelerating and managing the machine learning project lifecycle.

Appendix B

User Guide

B.1 Web Application

Figure 72 shows the first page of web application which is login page, the user should login to our app to make use of the full app functionalities. There is an account for admin who's only able to register new users to system as layers for security.

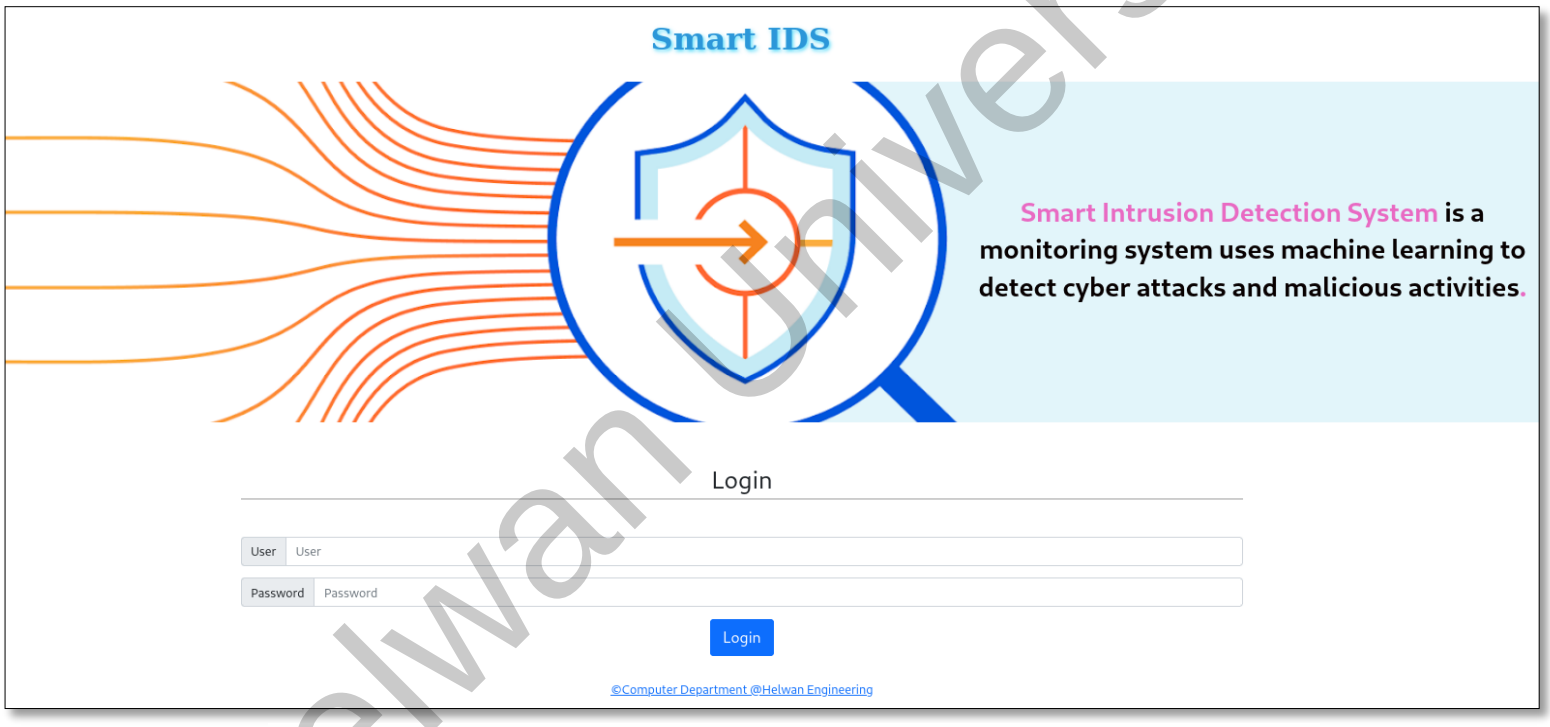


Figure 74: Login Page

After logging into our app, he will redirect to Dashboard page which contains basic information about features, team members and supervisor then on the left there is navigation bar to all features as shown in figure 73.

Figure 75: Dashboard

By navigating to Attack Detection it redirects to page as shown in figure 74 and by clicking start button the IDS starts and begin sniffing and showing results as figure

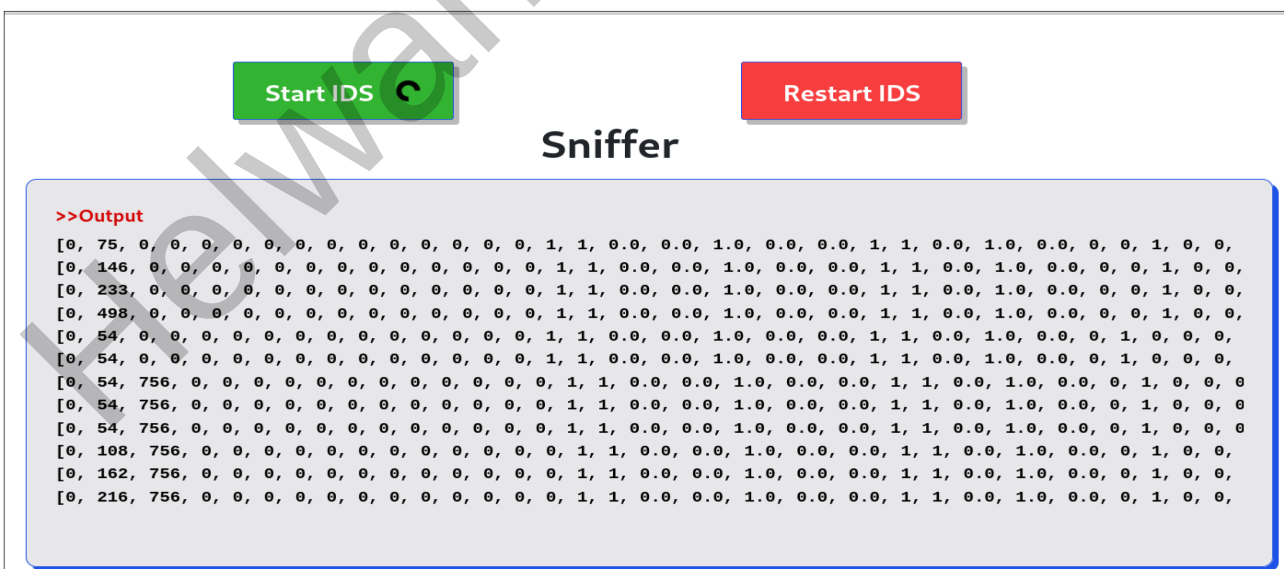


Figure 76: Sniffer Section

and the in detection section, there are the result for each packet as shown in figure 75:

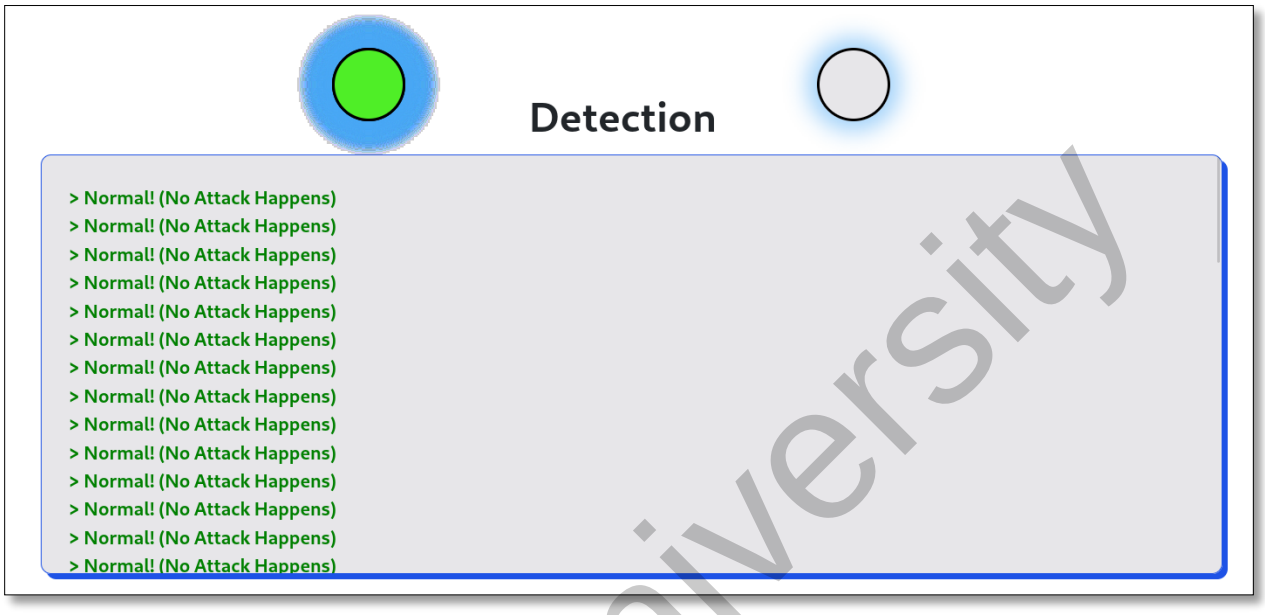


Figure 77: Detection Section

If there is any attack in section result the output will like as shown in figure 76:

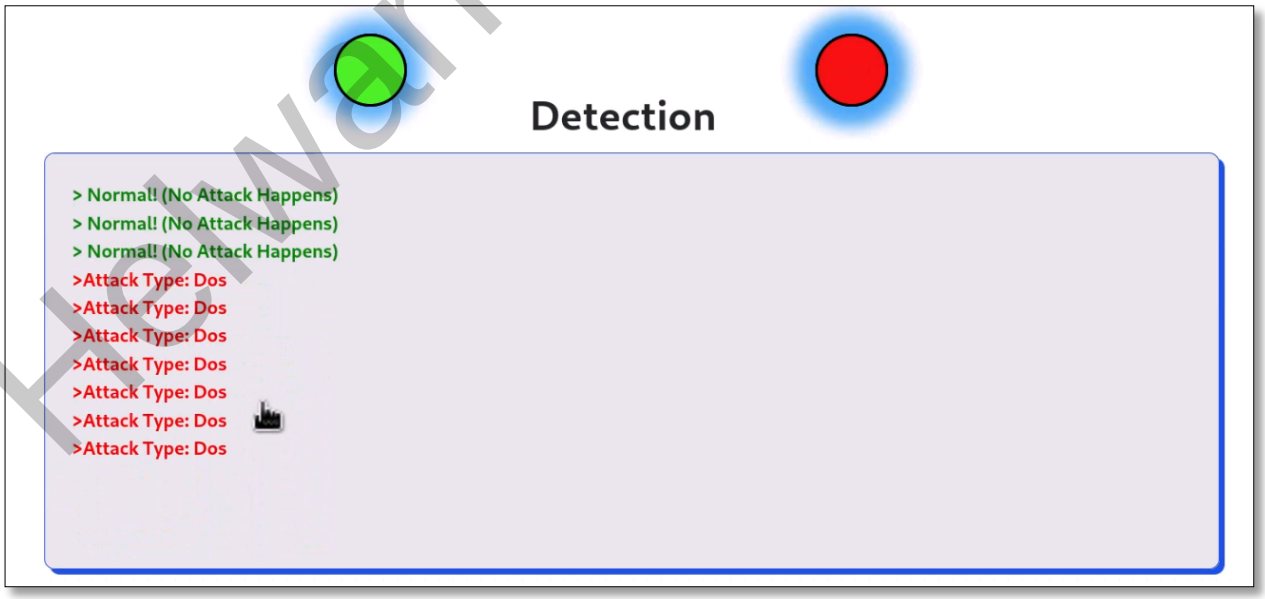


Figure 78: Detection section result in case of Dos attack.

Appendix C
KDDcup99 Attacks

C.1 Dos Attacks

- **Apache2**

  This is a DDoS attack against an Apache web server, where the client sends a request with several HTTP headers. If the server receives many of these requests, it will slow down and may eventually crash. Each request submitted as part of this attack contains multiple HTTP headers. Although the number and value of these headers may vary by the attacker, the version of this attack used in the DARPA evaluation program sent HTTP GET requests with a repeated header 10,000 times in each request. The actual content of the header is not important for the attack, as the attack relies on the multiplicity of headers in each request. A typical HTTP request includes about twenty headers or less, so the use of around 10,000 headers by this attack is a clear anomaly.

- **Back**

  In this type of DDoS attack against an Apache web server, the attacker submits requests with website addresses that contain numerous forward slashes. When the server attempts to process these requests, it will become slower and unable to process other requests. An intrusion detection system that searches for Back attacks needs to identify document requests that contain a certain number of forward slashes within the address, and based on this, consider these requests as an attack. Certainly, a request that includes 100 forward slashes within its website address will be considered an illegal request in most systems. This threshold can vary in order to find the optimal balance between intrusion detection and false alarms.

- **Land**

  A Land attack is an effective denial-of-service attack in older TCP/IP applications. A Land attack occurs when the attacker sends a spoofed SYN packet with the same source and destination address. This causes the system to enter an infinite loop as it receives a large number of SYN/ACK messages. The Land attack is distinguishable because it is not possible for packets to have the same source and destination addresses within any properly functioning network.

- **Mailbomb**

  In a Mailbomb attack, the attacker sends numerous messages to a server, causing its email queue to overflow and potentially causing the entire system to fail. An intrusion detection system searching for Mailbomb attacks can look for thousands of emails coming from or going to a particular user within a short period of time. This definition is somewhat subjective as each site can define the amount of email that can be sent by a single user before it is considered part of a Mailbomb attack.

- **SYN Flood (Neptune)**

  It is a denial-of-service attack that exploits vulnerabilities in the TCP/IP protocol suite. When a connection is established between a server's "tcpd" and a device, a record is added to the data structure that stores information on all pending connections. This data structure has a limited size, which can be overflowed by intentionally creating a large number of partially open connections. The victim server's data structure will eventually be filled with these partially open connections and will not be able to accept any new incoming connections until it is cleared. There is usually a time-out associated with the pending connections, so they will eventually expire, and the victim server will resume normal operation. However, the attacker system can continue to send spoofed packets requesting new connections faster than the victim system can clear the pending connections. In some cases, the victim server can

become overwhelmed, causing it to crash or become unresponsive. The Neptune attack can be distinguished from normal traffic by looking for a high number of synchronized SYN packets directed at a specific device from a host that cannot be reached. A host-based intrusion detection system can monitor the size of the tcpd data structure and alert the user when it approaches the maximum limit.

- **Ping of Death (PoD)**

It is a denial-of-service attack that affects many old operating systems. This attack exploits a flaw in the Internet Control Message Protocol (ICMP), which is part of the TCP/IP protocol suite and operates at the network layer, using the Information packets (IP datagram) belonging to the IP protocol to deliver messages. The ping command involves sending an IP packet to a specific network address or hostname to check for a response from that address or host, but sending a large number of ping packets can cause some systems to overflow and stop functioning. This attack can take another form that involves the attacker sending an illegal ping packet that contains more bytes than allowed and exceeds the size specified in the TCP/IP protocol suite descriptions, which requires data fragmentation. However, storing this data causes overflow in the caches or disabling the kernel, thus halting the processing operation. This type of attack was designed by some Windows systems to generate ICMP messages of non-standard sizes, as the maximum size of ICMP messages is 65,507 bytes, and each ECHO ICMP packet exceeds this size, which is then fragmented by the sender, and the receiver attempts to reassemble it when the overflow occurs. An attempt to Ping of Death attack can be detected by observing the size of all ICMP packets and identifying those longer than 64,000 bytes.

- **Process Table**

A DDoS (Distributed Denial of Service) attack specifically designed for this evaluation program. This attack can be launched against a massive number of network services on a variety of different UNIX

systems. The attack is launched against network services that use the "fork" command, which creates a new process, meaning against services that dedicate a new process for each incoming TCP/IP connection. While the standard UNIX operating system determines the number of processes a user can launch, there are no restrictions on the number of processes the root user can create, except for those imposed by the operating system itself.

An example of a service vulnerable to this attack is the Finger service, which provides information about users on a remote system, whether detailed information about users or information about a single user. On most computers, Finger is started by inetd, a primary server on most Unix systems that provides internet services. The authors of inetd have included several checks in the source code that must be bypassed to launch a successful process attack. In perfect execution (where some determinants vary depending on the UNIX version used), if inetd receives more than 40 connections to a specific service in one minute, that service will be disabled for ten minutes. The purpose of these checks is not to protect the server against process table attacks, but to protect the server against any buggy code that can create a large number of consecutive connections quickly and without interruption. To launch a successful process table attack against a computer running inetd and Finger, the following sequence is followed: (1) open a connection to the target's finger port, (2) wait 4 seconds, and (3) repeat steps 1 and 2.

- **Smurf**

In a Smurf attack, attackers use directed ICMP echo request packets to broadcast IP addresses of distant locations to create a denial-of-service attack. There are three participants in this attack: the attacker, the intermediary, and the victim (the intermediary may also be the victim). The attacker sends ICMP echo request packets to the broadcast address (xxx.xxx.xxx.255) of a number of subnetworks with a spoofed source address to appear as the intended victim. All devices listening on these subnetworks will respond by sending ICMP echo reply packets to the

victim. The Smurf attack is effective because the attacker can use broadcast addresses to amplify what might otherwise be harmless ping floods. In the best case (from the attacker's point of view), the attacker can flood the victim with packets that are 255 times larger than they would be without this amplification. **Figure 2-3** illustrates the effect of amplification. The attacking device (located on the left side of the figure) sends a single spoofed packet to the network's broadcast address, and every device on that network responds by sending a packet to the victim. Because there can be 255 devices within an Ethernet segment, the attacker can use this amplification to generate ping flood packets that are up to 255 times larger than they would otherwise be. This figure is a simplification of the Smurf attack. The attacker sends a flood of ICMP echo request packets to the broadcast address of many subnetworks, resulting in a massive, successive flood of echo replies that inundate the victim. A intrusion detection system can detect a Smurf attack by recognizing a large number of echo replies being sent to a particular victim from different locations, but no outgoing echo requests from the victim device.

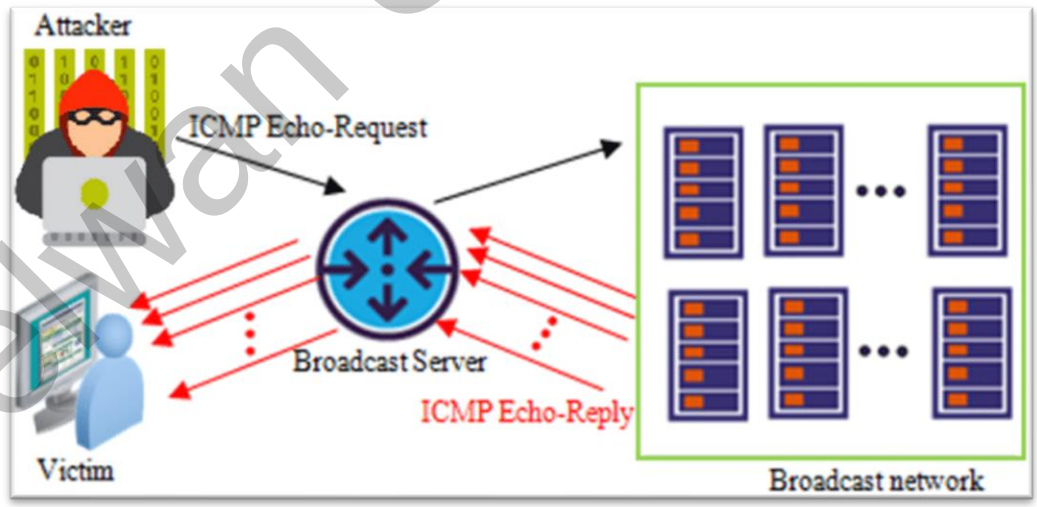


Figure 79: The Smurf attack

- **Teardrop**

  A Distributed Denial of Service (DDoS) attack exploits a vulnerability in the old stack implementation of TCP/IP, where some applications may not handle IP fragmentation properly with overlapping IP fragments. This attack involves sending fragmented packets to the target device. Since the device receiving such packets cannot reassemble them due to a programming error in the fragmentation reassembly of TCP/IP, the packets overlap with each other, causing the target device to crash. This typically occurs in old operating systems such as Windows 95, Windows 3.1x, Windows NT, and earlier versions of Linux kernel 2.1.63. One of the fields in the IP header is the "Fragment Offset" field, which indicates the starting position, or offset, of the data contained in a fragmented packet with respect to the data in the original packet. If the sum of the offset and size of one fragmented packet differs from that of the next fragmented packet, the packets will overlap. When this happens, the server vulnerable to Teardrop attacks will not be able to reassemble the packets, leading to a DDoS condition [52].

- **Udpstorm**

  A Distributed Denial of Service (DDoS) attack causes network congestion and slowdown. When a connection is established between two services, UDP for example, both services can produce a large number of packets that can lead to blocking the service on the device/devices that provide these services. Anyone with network access can launch the attack, no login required.

  For example, by connecting the host's "chargen" service with the "echo" service on the same device or on another device, all affected devices can be taken out of service due to the enormous number of packets produced. Figure (3-3) illustrates this attack. The figure shows how an attacker can create an endless flood of packets between the "echo" ports of two victims by sending a single spoofed packet. Initially, the attacker forges a single packet to appear as if it is coming from the "echo" port on the first

victim's device and sends it to the second victim. The "echo" service responds to any request it receives by echoing back the request data to the device and port that sent the "echo" request. Therefore, when the first victim receives this spoofed packet, it sends a response back to the "echo" port of the second victim. The second victim responds in a similar way, and the traffic loop continues until it is stopped by external intervention.

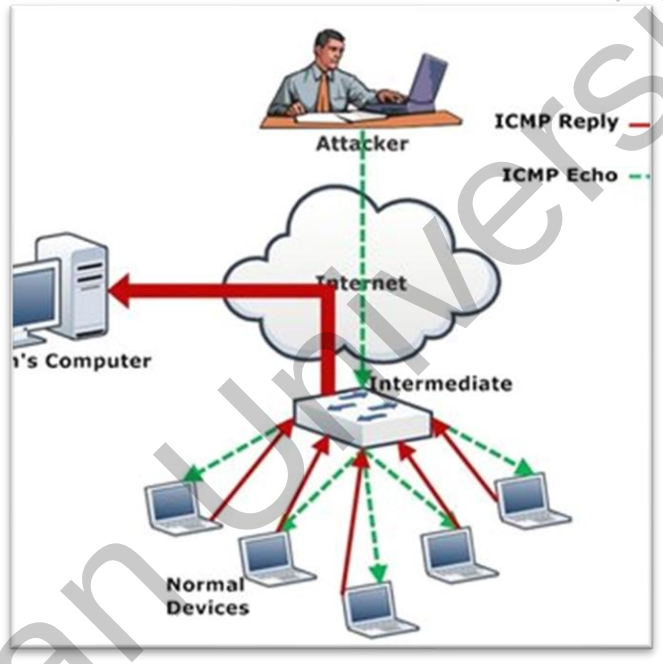


Figure 80: The Udpstorm attack

This attack can be detected in two ways. First, the single packet that initiates the attack can be identified because it is a packet coming from outside the network that has been forged to appear as if it is coming from a device within the network. Second, since the network traffic loop has started, an intrusion detection system that can see network traffic from within the network can detect that traffic is being sent from the "chargen" or "echo" port of one device to the "chargen" or "echo" port of another device.

C.2 R2L Attacks

- **Ftp-write**

It is an R2L attack that exploits misconfigured anonymous FTP. As a result, the attacker can add files, such as a "rhosts" file, and ultimately gain local access to the system. The "rhosts" file contains host-user mappings, not just hosts in general. If a host-user association is present in the list, the designated user is allowed to log in remotely from the specified host without needing a password. Unfortunately, this file poses a security problem, as any user can create an "rhosts" file that allows access to anyone without the system administrator's knowledge. An intrusion detection system can monitor this attack by monitoring all anonymous FTP sessions and ensuring that no files are created in the root directory.

- **Imap**

This attack exploits a buffer overflow vulnerability in the Imap cache of Red hat Linux 4.2, which allows remote attackers to execute arbitrary commands with root privileges. The Imap server must run with root privileges in order to access the mail folders and perform some file manipulation on behalf of the logged-in user. After logging in, these privileges are ignored. However, there is a programming error related to buffer overflow in the login authentication code, which can be exploited to gain root access to the server. By carefully crafting a text message to a vulnerable Imap server, remote users can cause a buffer overflow and execute arbitrary commands with root privileges. The Imap attack can be detected by monitoring network traffic for large Imap authentication strings, using an intrusion detection system.

- **Sendmail**

To begin with, Sendmail is the most popular application for the SMTP protocol used for email transmission over the internet. When a Sendmail

server receives an email, it attempts to deliver it to the intended recipient immediately. If the recipient is not found, the message is stored in a temporary cache for later delivery. As for the attack, it exploits a buffer overflow vulnerability in version 8.8.3 of Sendmail, allowing a remote attacker to execute commands with root-level privileges by carefully crafting an email message and sending it to a vulnerable version of Sendmail. This enables intruders to make Sendmail execute arbitrary commands with root-level privileges.

- **Xlock**

X Window System, referred to as X, is a system for managing graphical user interfaces on a single computer or a network of computers. The X protocol is a standard protocol used by X Window System, which defines a set of rules for transmitting information between computers about graphical user interfaces. The xlock attack is an attack in which a remote attacker gains local access by deceiving a legitimate user into revealing their password. The attacker can display a modified version of the xlock program on the user's screen, hoping to convince the user to stay in that control unit and enter their password. If the user on the attacked machine actually types their password into a Trojan horse version of xlock, it will be sent to the attacker. Two factors contribute to making this attack difficult to detect by intrusion detection systems. First, this attack does not exploit any programming errors within the system, but relies on the assumptions of the person currently using the system. Second, this attack is embedded within the X protocol, and an intrusion detection system that tries to detect the xlock attack must be familiar with and understand all X traffic. A less than ideal way to deal with these difficulties is to be suspicious of any X traffic originating from an unknown device and directed at a monitored device.

- **Xsnoop**

Xserver is a program that runs on local devices (i.e., computers used directly by the user) and handles all access to graphics cards, displays,

and input devices (keyboard and mouse) on these computers. It also communicates with Xclients, which are application programs displayed on it, but may run on the local device or remote devices (i.e., another computer on the network). In a Xsnoop attack, the attacker monitors keystrokes from an unprotected Xserver to attempt to obtain information that can be used to gain local access to the victim's system. The attacker can monitor keystrokes of a user who left an X display screen open. A keystroke log will be extremely useful to the attacker because it may contain sensitive information or information that can be used to gain access to the system, such as the username and password of the person being monitored. An intrusion detection system can detect the attack by analyzing X protocol information in packets directed to remote Xclients and observing keystroke events that are transmitted to a remote device. Because Xsnoop attacks result from poor security policies (such as leaving an Xserver unprotected) rather than a programming error, the existence of these packets alone is not evidence of an attack. It is up to the intrusion detection system to determine whether the system's security policies allow unauthorized X communications from anywhere.

- **Warezmaster**

This attack exploits a vulnerability associated with a File Transfer Protocol (FTP) server that typically disallows users or guests from writing to the server, therefore preventing them from uploading files. Most public FTP servers have guest accounts for downloading data, and anyone can log in using these guest accounts. The attack occurs when the FTP server mistakenly grants permissions to users on the system, allowing any user to log in and download files. During the attack, the attacker logs in to the server using a guest account, creates a hidden directory, and uploads copies of illegal software, or "warez," to the server. A simple and clear way to prevent this attack is to correctly set user permissions on the server [31].

- **Warezclient**

An attack called warezclient can be launched by any legitimate user during an FTP connection, following a successful attack by warezmaster. During the attack, warezclient downloads previously released illegal "warez" software to users. The only detectable feature of this attack is downloading files from hidden directories or directories that are not typically accessible by guest users on the FTP server. This requires monitoring all legitimate directories and verifying whether the files being downloaded during FTP sessions belong to legitimate directories or not [31].

C.3 U2R Attacks

- **Buffer overflow**

This attack occurs when a program copies a large amount of data to a cache without verifying if the data will fit or not. For example, if the program expects the user to enter their first name, the programmer must specify the number of characters that the cache for the first name requires. If the program allocates 20 characters for the cache for the first name, and the user's first name consists of 35 characters, the last 15 characters will overflow from the cache for the name. When this overflow occurs, the extra characters will be placed on the stack, and therefore written over the next set of information that was supposed to be executed. An attacker can manipulate the data stored in the stack to cause arbitrary commands to be executed by the operating system [20].

- **Loadmodule**

This is a U2R attack against SunOS 4.1 systems that use the xnews window system. The loadmodule program is used by the xnews window system server to load kernel modules into the currently running system. Due to a programming error during loadmodule program initialization, unauthorized users can gain root access to the local system.

- **Rootkit**

  This is a malicious program designed to hide the existence of specific processes or programs from normal detection methods. This program can provide continuous access to computer privileges [20].

- **PS**

  This attack exploits race conditions within Solaris 2.5 and allows the attacker to execute arbitrary code with root privileges. Exploiting the race condition only allows the attacker to gain root access if the user has access to the temporary files. Access to temporary files can be obtained if permissions to the /tmp and /var/tmp directories are incorrectly specified. Thus, any user who has logged into the system can gain unauthorized root privileges by exploiting this race condition.

- **Xterm**

  The Xterm exploit leverages a buffer overflow vulnerability in the Xaw library, distributed with the Redhat operating system (Linux 5.0, as well as other non-simulated operating systems), allowing an attacker to execute arbitrary instructions with root privileges. Exploiting these cache overflow vulnerabilities in Xterm or any program that uses the Xaw library allows an unprivileged user to gain root access to the system.

C.4 Probe Attacks

- **Ipsweep**

  This is a reconnaissance scan to determine which hosts are listening on the network. This information is useful for an attacker trying to launch an attack and looking for vulnerable devices. A intrusion detection system that searches for simple ipsweep used in the simulation should search for several packets, Pings, directed to each potential device on the network and coming from the same source.

- **Portsweep**

  Port scanning is the process of scanning a system's ports to determine which ones are open and what services are running on them. Most packets that leave a device leave through a specific port and are directed to another port on another system. This type of attack involves scanning ports to determine which services are supported over a period of time by a specific host, in order to gather information to find specific vulnerabilities.

- **Mscan**

  It is a verification tool that uses DNS zone transfer, which is one of the mechanisms used by the DNS system to store, replicate, and exchange DNS databases between servers, in order to determine the location of devices and explore them for vulnerabilities. The detection of attacks depends on the vulnerabilities that have been scanned and the number of devices that have been scanned. Generally, an intrusion detection system can detect an Mscan attack by searching for connections from a single external device to specific ports on one or more devices within a short period of time.

- **Nmap**

  Nmap is a free and open-source general-purpose tool primarily used for scanning addresses, IPs, port scanning, firewall and operating systems using IP packets directed towards the victim's computer. Nmap also allows the user to specify which ports to scan, how long to wait between each port, and whether to scan ports sequentially or randomly. The distributed nature of the sources used in the attack and the slow pace of the attack make it difficult to detect the scan performed using Nmap.

- **Saint**

  It is a comprehensive network tool for a security administrator. In its simplest form, it collects as much information as possible about remote hosts and networks by scanning network services such as ftp, NIS, NFS, finger, rexd, tftp, statd, and others. The collected information includes various details about network services as well as potential security vulnerabilities. These vulnerabilities may include misconfigured or poorly tuned network services, known software errors within the system or network, and incorrect security policy decisions. Although Saint is not intended to be used as an attack tool, it provides valuable security information to attackers. A Saint network scan will leave a signature that varies depending on the level of the scan applied to the network. The Saint program performs each scan in a specific way. Intrusion detection systems need to recognize a distinctive set of network traffic patterns that the scan forms.

- **Satan**

  "Satan is the predecessor to the network scanning tool Saint discussed in the previous section. While Satan and Saint are similar in purpose and design, they differ in the vulnerabilities each tool checks for. The network scan performed by Satan can be identified by a consistent pattern of network traffic created by the program. Verification of the vulnerabilities mentioned above is always done in the same order."