



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **Mr. Viswambaravarma (41614102)** who carried out the project entitled "**DEVELOP A MAC ADDRESS MORPHER**" under my supervision from January 2024 to April 2024.

Internal Guide

Dr.S. NIVETHA, M.E., Ph.D.,

Head of the Department

Dr.S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 07/05/24

Internal Examiner

External Examiner

DECLARATION

I, **Viswambaravarma (Reg.No.41614102)**, hereby declare that the Professional Training Report-II entitled "**DEVELOP A MAC ADDRESS MORPHER**" done by me under the guidance of **Dr.S. Nivetha M.E., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Cyber Security.

DATE: 07/05/2024

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

PROFESSIONAL TRAINING REPORT-II

entitled

DEVELOP A MAC ADDRESS MORPHER

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
specialization in Cyber Security

by

Ram Sainath Reddy [Reg.No.41614105]

Viswambaravarma [Reg.No.41614102]

Vikram Reddy [Reg.No.41614091]

Vamsi Krishna [Reg.No.41614099]



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **Mr. Viswambaravarma (41614102)** who carried out the project entitled "**DEVELOP A MAC ADDRESS MORPHER**" under my supervision from January 2024 to April 2024.

Internal Guide

Dr.S. NIVETHA, M.E., Ph.D.,

Head of the Department

Dr.S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Viswambaravarma (Reg.No.41614102)**, hereby declare that the Professional Training Report-II entitled “**DEVELOP A MAC ADDRESS MORPHER**” done by me under the guidance of **Dr.S. Nivetha M.E., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Cyber Security.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E., Ph.D., Dean, School of Computing, Dr.S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Dr.S.Nivetha M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-2 professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE



NPTEL Online Certification

(Funded by the MoE, Govt. of India)



This certificate is awarded to

VISWAMBARAVARMA

for successfully completing the course

Cloud Computing and Distributed Systems

with a consolidated score of **55** %

Online Assignments	16.25/25	Proctored Exam	37.5/75
--------------------	----------	----------------	---------

Total number of candidates certified in this course: **3244**

Prof. B. V. Ratish Kumar
Chairman, Centre for Continuing Education
IIT Kanpur

Jan-Mar 2024

(8 week course)

Prof. Satyaki Roy
NPTEL Coordinator
IIT Kanpur



Indian Institute of Technology Kanpur



Roll No: NPTEL24CS09S344109349

To verify the certificate



No. of credits recommended: 2 or 3

ABSTRACT

This project is about changing our systems MAC address. The MAC (Media Access Control) Address is known as the hardware id number. In every computer's NIC (Network Interface Card), including a Bluetooth, WI-FI card or Ethernet card, has an unchanged MAC address inserted by the manufacturer at the time of production. The manufacturer of each networking device assigns it a MAC address, which is useful for interacting with other devices. The device's MAC address is hard-coded, and it cannot be changed indefinitely.

However, a MacMorpher can be used to temporarily alter it. Until the device is rebooted, a MacMorpher is a device that modifies the MAC address to the desired (or random) address. This project focuses on Spoofing (or) Morphing the mac address. This tool can be used to gain access to networks that are limited to certain MAC addresses, and this tool can also be used to hide the identity of the original device. In conclusion, building a MAC address changer from scratch can be a useful for those interested in networking and system administration. It involves understanding how MAC addresses work, and how to use Python's built-in libraries and functions to manipulate network interfaces on a computer.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Overview	
2	LITERATURE SURVEY	3
	2.1 survey	
3	REQUIREMENTS ANALYSIS	
	3.1 Objective	9
	3.2 System Requirements	9
	3.2.1 Hardware Requirements	10
	3.2.2 Software Requirements	
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT	
	4.1 Proposed Methodology	11
	4.1.1 Ideation Map/Architecture Diagram	11
	4.1.2 Various stages	12
	4.2 Working Principles	13
5	RESULTS AND DISCUSSION	19
6	CONCLUSIONS AND FUTURE WORK	21
	REFERENCES	23
	APPENDIX	
	A. Research Paper	26
	B. Source Code	53

LIST OF FIGURES

Figure No.	Figure Name	Page No.
2.1.1	Basic Existing Model	3
2.1.2	Another Existing Model	4
4.1	Proposed Project	11
4.1.1	Ideation Map	12
4.2.1	MAC Address Structure	14
4.2.2	MAC Address Format	15
4.2.3	Frame Format in Data-link Layer	15
4.2.4	Data-link Layer	17

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In the ever-evolving landscape of digital technology, security and privacy concerns have become paramount. One critical aspect of this concern is the manipulation of MAC addresses, which are unique identifiers assigned to network devices. Introducing the MAC Address Morpher Tool – a powerful solution designed to enhance your online privacy and security.

The MAC Address Morpher Tool is a sophisticated software utility that enables users to alter their device's MAC address at will. This tool offers an invaluable layer of anonymity and security, allowing individuals to shield their identities and protect their online activities from prying eyes.

Whether you're a cybersecurity enthusiast looking to test network vulnerabilities, a privacy-conscious user seeking to enhance your digital footprint, or a professional in need of secure network testing, the MAC Address Morpher Tool provides the flexibility and control you need.

With its user-friendly interface and advanced features, this tool empowers users to modify their MAC address with ease, making it an essential asset in the arsenal of anyone concerned with privacy and security in the digital age. Say goodbye to potential tracking and monitoring and say hello to enhanced online freedom and peace of mind.

Explore the possibilities of the MAC Address Morpher Tool and take control of your digital identity, ensuring a safer and more private online experience.

Developing a Mac Address Morpher marks a pioneering step in the realm of network security and customization. This innovative software tool empowers users to dynamically modify and morph their device's Media Access Control (MAC) addresses, enhancing privacy, security, and network management capabilities. At its core, the Mac Address Morpher leverages advanced algorithms and network protocols to facilitate seamless manipulation of MAC addresses on macOS systems. MAC addresses, which are unique identifiers assigned to network interfaces, play a crucial role in network

communication and device identification. By allowing users to alter these addresses on the fly, the Mac Address Morpher provides a powerful tool for enhancing anonymity, thwarting surveillance, and circumventing network restrictions.

One of the key features of the Mac Address Morpher is its versatility and user-friendliness. Whether you're a cybersecurity professional, a privacy-conscious individual, or a network administrator, the software offers a straightforward yet powerful interface for customizing and managing MAC addresses with ease. From randomizing addresses for enhanced anonymity to spoofing addresses for bypassing access controls, the Mac Address Morpher provides a comprehensive suite of tools and options to suit a variety of use cases and preferences.

Moreover, the Mac Address Morpher prioritizes compatibility and interoperability, seamlessly integrating with existing macOS networking frameworks and protocols. Whether you're connecting to Wi-Fi networks, accessing Ethernet connections, or interacting with virtualized networking environments, the software ensures smooth and reliable operation across a wide range of network configurations and scenarios.

In addition to its practical applications, the Mac Address Morpher also serves as a valuable educational tool, helping users gain a deeper understanding of network protocols, security principles, and privacy implications. By enabling users to experiment with MAC address manipulation in a controlled environment, the software facilitates hands-on learning and exploration, empowering individuals to take greater control of their digital identities and network interactions.

In summary, the development of a Mac Address Morpher represents a significant milestone in the evolution of network security and customization tools for macOS systems. By providing users with the ability to dynamically modify and morph their MAC addresses, the software opens up new possibilities for enhancing privacy, security, and network management capabilities in an increasingly connected world.

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

[1] MAC address morphing or spoofing, this tool is typically used for legitimate purposes, such as enhancing privacy or troubleshooting network issues, but it's essential to use them responsibly and within legal boundaries. Keep in mind that the availability of specific products may change over time, so it's a good idea to research the latest options. Here are some examples of product categories that were available:

[2] There were software applications available for different operating systems (e.g., Windows, macOS, Linux) that allowed users to change their MAC address easily. Some of these were free and open-source, while others were commercial products.

```
Welcome to Mac changer

+-----+-----+-----+
| Interface | Mac Address | IP Address |
+-----+-----+-----+
| lo        | No MAC assigned | 127.0.0.1 |
| eth0      | 08:00:27:1c:6d:55 | 10.0.2.11 |
+-----+-----+-----+

[+] Name of the interface /-- For more details refer ip table : eth0

[*] MAC Address Changed from 08:00:27:1c:6d:55 to 00:16:08:28:01:ac

[+] Do you want to print the IP Table : Y?N : Y

+-----+-----+-----+
| Interface | Mac Address | IP Address |
+-----+-----+-----+
| lo        | No MAC assigned | 127.0.0.1 |
| eth0      | 00:16:08:28:01:ac | 10.0.2.13 |
+-----+-----+-----+
```

Fig 2.1.1: Basic Existing Model

[3] Some comprehensive network security suites or tools included features for MAC address manipulation as part of their broader set of network management and security functions.

[4] Some VPN services offered MAC address spoofing as a feature, which could be used to enhance anonymity and privacy while browsing the internet.

[5] Certain wireless network adapters and hardware devices had the capability to change their MAC address programmatically, providing users with an additional layer of control over their network identity.

[6] Network professionals and administrators often used MAC address spoofing tools for diagnosing and troubleshooting network issues. These tools could assist in testing network configurations and security.

[7] Advanced users could create custom scripts or use command-line tools on various operating systems to change their MAC address manually.

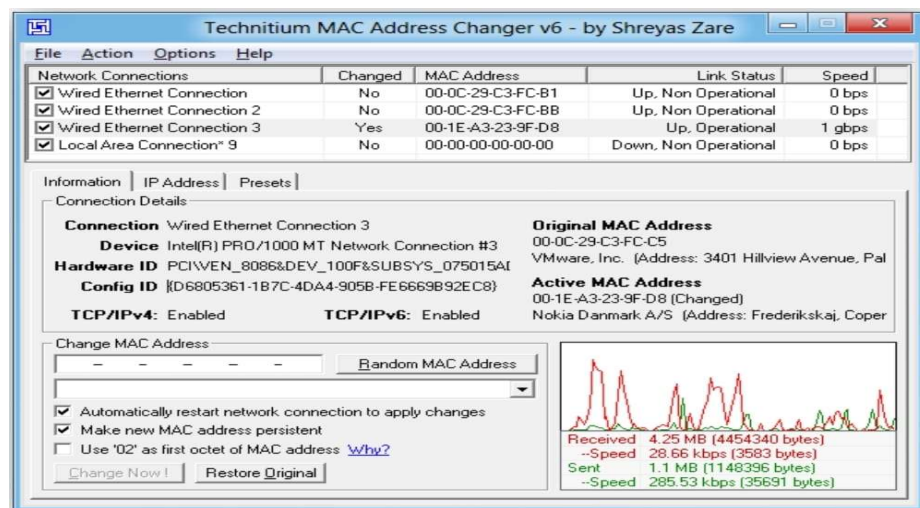


Fig 2.1.2: Another Existing Model

[8] Discuss the role of MAC addresses in different network architectures, such as Ethernet, Wi-Fi, and Bluetooth. Explore how MAC addresses are used for device identification, address resolution, and communication within local networks. Consider the implications of MAC address uniqueness and persistence for network management and security.

[9] Provide examples of real-world scenarios where MAC address spoofing is commonly employed, such as bypassing captive portals, MAC filtering, or anonymizing network traffic. Review the technical mechanisms involved in MAC address spoofing, including modifications at the hardware, firmware, and software levels. Examine the

effectiveness of MAC address spoofing in circumventing network-based security measures, such as port security or MAC-based authentication.

[10] Investigate the impact of MAC address manipulation on network security mechanisms, such as intrusion detection systems (IDS), network access control (NAC), and port security. Discuss the potential risks associated with MAC address spoofing, including network impersonation, man-in-the-middle attacks, and unauthorized access. Review case studies or empirical studies that demonstrate the practical implications of MAC address manipulation on network security posture.

[11] Explore the relationship between MAC address exposure and user privacy in different network environments, including public Wi-Fi hotspots, residential networks, and enterprise networks. Discuss the limitations of traditional MAC address randomization techniques in mitigating privacy risks, such as tracking and profiling. Review privacy-enhancing technologies (PETs) that leverage MAC address manipulation as part of broader privacy protection strategies, such as Tor or VPN services.

[12] Identify common tools and software libraries used for MAC address manipulation, including command-line utilities, scripting languages, and network management software. Compare the features and capabilities of popular MAC address manipulation tools, such as macchanger, SpoofMAC, or Wireshark. Discuss the availability of documentation, community support, and development activity for open-source MAC address manipulation tools.

[13] Explore different approaches to dynamically morphing MAC addresses, such as temporal-based morphing, spatial-based morphing, or protocol-aware morphing. Investigate the feasibility of implementing MAC address morphing techniques in software-defined networking (SDN) environments or virtualized network infrastructures. Discuss the trade-offs between different morphing strategies in terms of complexity, overhead, and effectiveness in evading detection.

[14] Analyze the ethical implications of MAC address manipulation in the context of privacy, consent, and trust. Review relevant legislation, regulations, and industry guidelines governing the use of MAC address manipulation techniques, such as GDPR, CCPA, or industry-specific standards. Discuss ethical considerations for

researchers and practitioners working on MAC address manipulation projects, including responsible disclosure practices and ethical hacking principles.

[15] Identify technical challenges and limitations in developing robust MAC address morphing solutions, such as interoperability issues, timing constraints, or hardware dependencies. Discuss potential countermeasures and detection techniques that network administrators and security practitioners can deploy to detect and mitigate MAC address manipulation attacks. Highlight emerging trends and future research directions in MAC address manipulation, such as the integration of MAC address morphing with blockchain-based identity solutions or the adoption of standardized privacy-preserving protocols at the network layer.

[16] Explore the structure of MAC addresses, including how they are assigned and formatted. Understanding the components of a MAC address (e.g., OUI, NIC-specific portion) can provide insights into how MAC address morphing can be implemented.

[17] Investigate research on MAC address randomization techniques used to enhance privacy and security in wireless networks. Analyze how MAC address morphing can be used as a form of randomization to prevent tracking and profiling.

[18] Look into literature on network forensics and the role of MAC addresses in identifying and tracing network activities. Understand how MAC address morphing can impact forensic investigations and the challenges associated with attributing network traffic to specific devices.

[19] Consider the legal and regulatory implications of MAC address manipulation, especially in industries with strict compliance requirements (e.g., healthcare, finance). Review literature on data privacy laws and regulations that may govern the use of MAC address morphing tools.

[20] Explore how machine learning algorithms can be applied to detect anomalies in MAC address behavior and identify potential MAC address spoofing attacks. Look for studies on anomaly detection techniques in network security.

[21] Seek out case studies or real-world examples of MAC address morphing in network security applications. Analyze the challenges faced and lessons learned from implementing MAC address morphing solutions in different environments.

Investigate emerging trends in network security and potential research directions for enhancing MAC address morphing techniques. Consider the impact of technologies like IoT, 5G, and cloud computing on the evolution of MAC address manipulation methods.

[22] Investigate research on the use of MAC addresses for tracking and surveillance purposes by various entities, including internet service providers, advertisers, and government agencies. Explore the implications of MAC address tracking for user privacy, including the collection of location data, browsing habits, and device usage patterns. Discuss strategies for mitigating the risks associated with MAC address tracking, such as MAC address rotation, obfuscation techniques, and privacy-preserving network architectures.

[23] Examine studies on user attitudes and perceptions towards MAC address manipulation and privacy-enhancing technologies. Investigate factors influencing user acceptance of MAC address morphing tools, such as usability, effectiveness, and perceived benefits versus risks. Discuss strategies for promoting awareness and educating users about the importance of MAC address privacy and the capabilities of MAC address morphing tools.

[24] Review research on the forensic analysis of MAC addresses in digital investigations, including techniques for tracing network activity back to specific devices. Discuss the challenges and limitations of using MAC addresses as digital fingerprints for attribution purposes, such as MAC address spoofing and device mobility. Explore novel methodologies and tools for correlating MAC addresses with other forensic artifacts to enhance network attribution and incident response capabilities.

[25] Investigate the challenges and opportunities in developing a cross-platform Mac Address Morpher that supports multiple operating systems and network environments. Review existing research on MAC address manipulation tools for platforms other than macOS, such as Windows, Linux, and mobile operating systems. Discuss strategies for achieving interoperability and consistency in MAC address manipulation across different platforms, including standardization efforts and platform-specific optimizations.

[26] Explore research on using machine learning techniques for detecting anomalous MAC address behavior in network traffic. Discuss the potential for integrating machine learning algorithms into Mac Address Morpher to automate MAC address selection and adaptation based on contextual information and network dynamics. Investigate the challenges and ethical considerations associated with using machine learning for MAC address morphing, such as model bias, data privacy, and adversarial attacks.

[27] Examine the role of community-driven development and open-source collaboration in advancing MAC address manipulation tools and techniques. Review popular open-source projects and forums related to network security and privacy, where developers and researchers contribute to the development of MAC address morphing solutions. Discuss the benefits of open-source development models for fostering innovation, transparency, and community engagement in the development of Mac Address Morpher and related software tools.

[28] Explore research on user-centered design principles and methodologies for developing intuitive and user-friendly MAC address morphing interfaces. Discuss the importance of usability testing and feedback gathering in refining the user experience of Mac Address Morpher, considering diverse user demographics and skill levels. Investigate innovative approaches to visualizing and interacting with MAC address manipulation parameters, such as graphical interfaces, interactive tutorials, and contextual help systems.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

- To bypass access control lists on servers or routers.
- To provide enhanced privacy to users.
- To provide protection against MAC-based tracking.
- To provide users more control over their network identity.

3.2 SYSTEM REQUIREMENTS

1. Operating system compatibility.
2. Administrator (or) root access.
3. Compatible network adapter.
4. Understanding of mac addresses
5. Legal and ethical considerations
6. Backup and restore
7. Purpose and indent
8. Network awareness
9. Tool selection
10. Documentation
11. Network configuration knowledge
12. Testing

3.2.1 *HARDWARE REQUIREMENTS*

- PROCESSOR : Intel Core i5 or a better processor
- RAM : 8 GB RAM or higher
- HARD DISK SIZE : Minimum 30 GB or higher

- GRAPHICS CARD : A better graphics card is recommended
- INTERNET : Must have a ethernet connection
- KEYBOARD : Standard Keyboard
- MONITOR : 15 VGA color

3.2.2 SOFTWARE REQUIREMENTS

- OPERATING SYSTEM : Windows 10, Windows 11, MAC OS, Linux
- CODING LANGUAGE : IDLE(Python 3.9)
(Python Must Be Installed In The System.)

CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

4.1 PROPOSED METHODOLOGY

Our proposed tool had an very interactive and user-friendly interface. This tool automatically gets the MAC addresses of all the network devices present in the system, as it consists an in-built “getmac” command which will be executed in the command prompt and gathers all the MAC addresses. The user had to select which MAC address he want to change. Then this tool displays some randomly generated MAC addresses and leaves the option for user to select from those. After the selection by disabling and reenabling the wireless device the MAC address morphing is successful. This tool is very helpful to the users and they do not have to do more tasks, in just 2-3 steps they can change they MAC address using this tool.

```
Microsoft Windows [Version 10.0.22621.2361]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\nithi\Downloads

C:\Users\nithi\Downloads>python "change mac.py"
#####
1) Make sure you run this script with administrator privileges
2) Make sure that the WiFi adapter is connected to a network
#####

Which MAC Address do you want to update?
0 - Mac Address: 5C-60-BA-D6-D9-C7 - Transport Name: {D84BF83F-4836-4D7F-B394-00A29E21D847}
1 - Mac Address: 00-50-56-C0-00-01 - Transport Name: {F754D008-D6B0-4A3E-B372-543A75147BF9}
2 - Mac Address: 00-50-56-C0-00-08 - Transport Name: {F96001BC-9C8E-43B3-AAE9-63712537F073}
Select the menu item number corresponding to the MAC that you want to change:0
Which MAC address do you want to use? This will change the Network Card's MAC address.
0 - Mac Address: 0A1122334455
1 - Mac Address: 0E1122334455
2 - Mac Address: 021122334455
3 - Mac Address: 061122334455
Select the menu item number corresponding to the new MAC address that you want to use:1
Your Mac Address will be changed to: 0E1122334455
Successfully matched Transport Number
Do you want to disable and reenale your wireless device(s). Press Y or y to continue:n

C:\Users\nithi\Downloads>getmac

Physical Address    Transport Name
=====
0E-11-22-33-44-55   \Device\NPF{D84BF83F-4836-4D7F-B394-00A29E21D847}
20-20-20-08-E7-C9   \Device\NPF{E8923AFF-700E-4683-8E47-925D2D981282}
20-20-20-08-E7-CA    Media disconnected
00-50-56-C0-00-01    \Device\NPF{F754D008-D6B0-4A3E-B372-543A75147BF9}
00-50-56-C0-00-08    \Device\NPF{F96001BC-9C8E-43B3-AAE9-63712537F073}

C:\Users\nithi\Downloads>
```

Fig 4.1: Proposed Project

4.1.1 Ideation Map/System Architecture

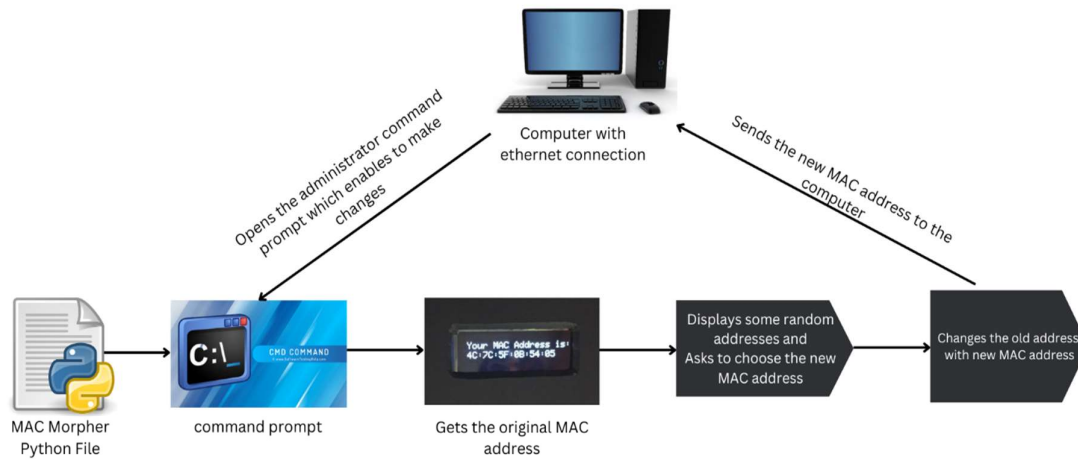


Fig 4.1.1: Ideation Map

4.1.2 Various Stages

The process of using a MAC address morpher typically involves several stages, from preparation to implementation. Here are the various stages involved:

1. Preparation:

- a. Research: Understand the purpose and legality of changing your MAC address. Research the specific requirements and considerations for your situation.
- b. Backup: If possible, create a backup or record of your current MAC address settings. This can be helpful in case you need to revert to the original address.
- c. Choose a Tool: Select a MAC address morphing tool or method that is compatible with your operating system and meets your requirements.

- d. Obtain Permissions: Ensure that you have the necessary administrative or root permissions to make changes to your MAC address, as this may require elevated privileges.

2. Configuration:

- a. Install the Tool: If you're using software, install the chosen MAC address morphing tool on your device. Ensure it's from a reputable source.
- b. Open the Tool: Launch the tool or access the MAC address settings on your device, depending on the chosen method.

3. MAC Address Modification:

- a. Manual Entry: Some tools allow you to manually input the new MAC address you want to use. Enter the desired MAC address following the tool's instructions.

4.2 WORKING PRINCIPLES

Before discussing about the working principles, first let's discuss about the mac address and its working in detail:

A MAC address (media access control address) is a 12-digit hexadecimal number assigned to each device connected to the network. Primarily specified as a unique identifier during device manufacturing, the MAC address is often found on a device's network interface card (NIC). A MAC address is required when trying to locate a device or when performing diagnostics on a network device.

The MAC address belongs to the data link layer of the Open Systems Interconnection (OSI) model, which encapsulates the MAC address of the source and destination in the header of each data frame to ensure node-to-node communication. Each network interface in a device is assigned a unique MAC address, so it's possible for a device to have more than one MAC address. For example, if a laptop has both an Ethernet cable port and built-in Wi-Fi, there will be two MAC addresses shown in the system configuration.

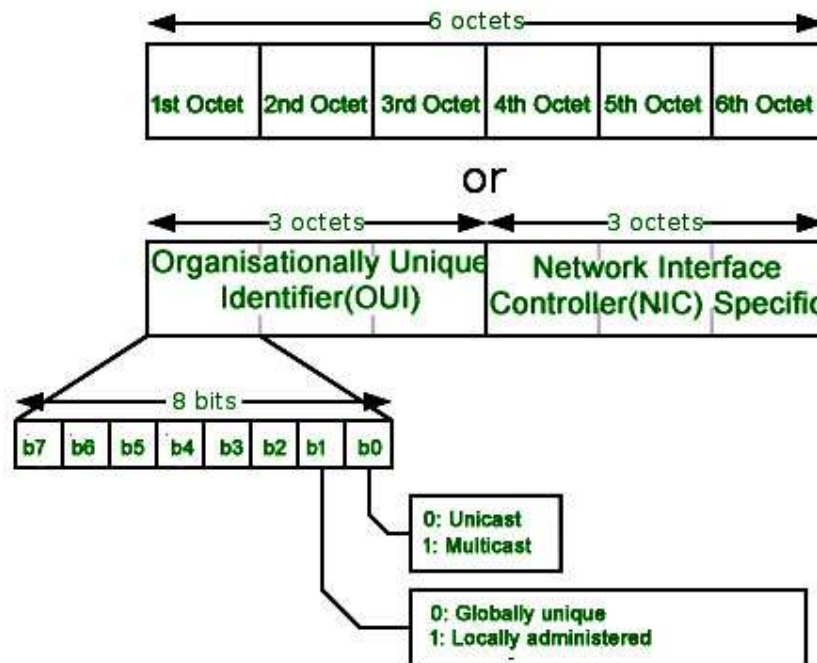


Fig 4.2.1: MAC Address Structure

The First 6 digits (say 00:40:96) of the MAC Address identify the manufacturer, called the OUI (Organizational Unique Identifier). IEEE Registration Authority Committee assigns these MAC prefixes to its registered vendors. Here are some OUI of well-known manufacturers:

CC:46:D6 - Cisco

3C:5A:B4 - Google, Inc.

3C:D9:2B - Hewlett Packard

00:9A:CD - HUAWEI TECHNOLOGIES CO.,LTD

The rightmost six digits represent Network Interface Controller, which is assigned by the manufacturer. As discussed above, the MAC address is represented by Colon-Hexadecimal notation. But this is just a conversion, not mandatory. MAC address can be represented using any of the following formats:

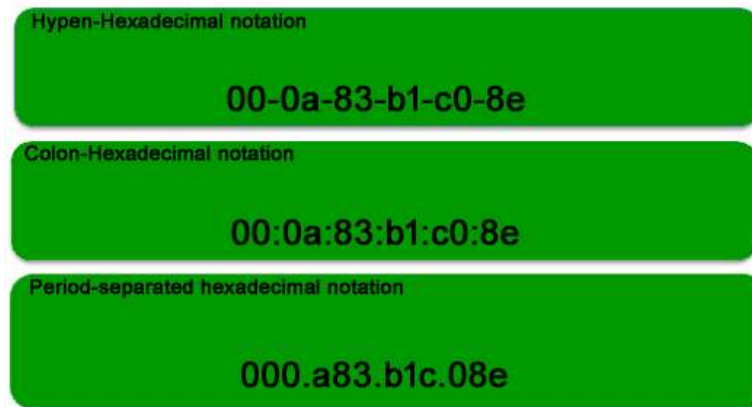


Fig 4.2.2: MAC Address Format

In IEEE 802 standard, the data link layer is divided into two sublayers:

1. Logical Link Control (LLC) Sublayer
2. Media Access Control (MAC) Sublayer

The MAC address is used by the Media Access Control (MAC) sublayer of the Data-Link Layer. MAC Address is worldwide unique since millions of network devices exist and we need to uniquely identify each.

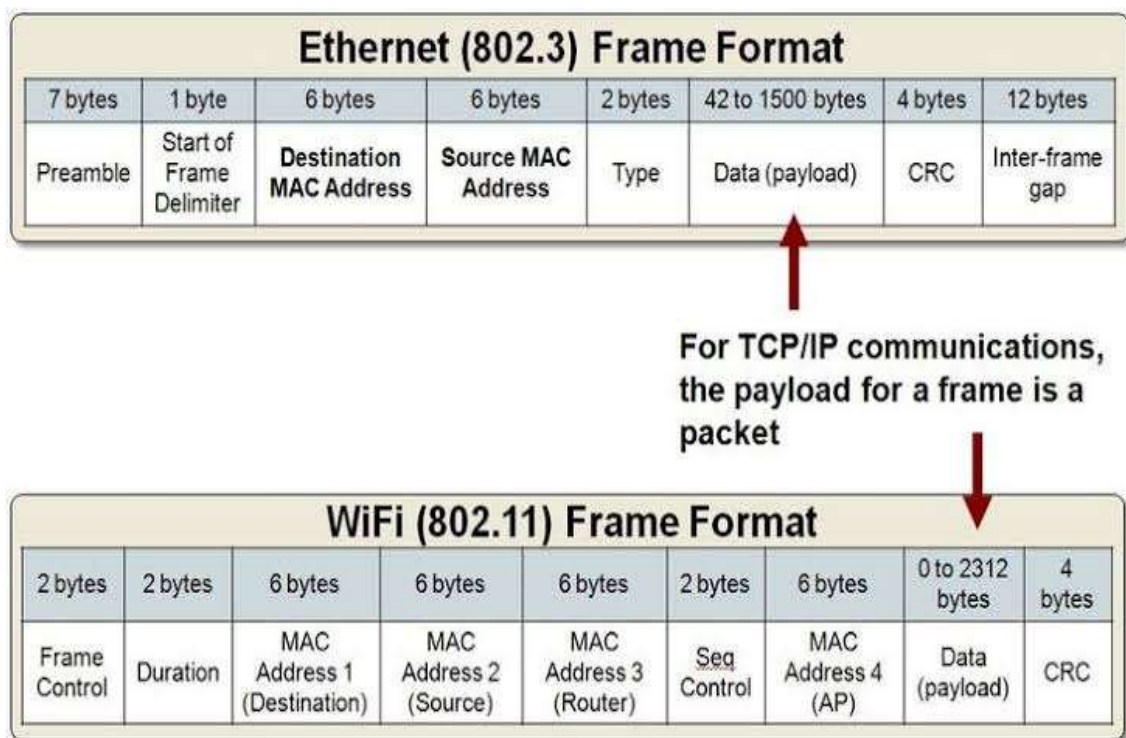


Fig 4.2.3: Frame Format In Data-link Layer

The MAC address belongs to the data link layer of the OSI model. Now let's discuss about data link layer. The data link layer is the second layer from the bottom in the OSI (Open System Interconnection) network architecture model. It is responsible for the node-to-node delivery of data. Its major role is to ensure error-free transmission of information. DLL is also responsible for encoding, decode and organizing the outgoing and incoming data. This is considered the most complex layer of the OSI model as it hides all the underlying complexities of the hardware from the other above layers.

The Data Link Layer in networking offers several advantages:

1. **Error Detection and Correction**: It ensures that data sent over the network is free from errors by adding checksums and can often correct errors through mechanisms like Automatic Repeat reQuest (ARQ).
2. **Frame Delimiting**: It defines the start and end of each frame, making it easier for receiving devices to identify and separate frames in a stream of data.
3. **Addressing**: Data Link Layer uses physical addresses (MAC addresses) to uniquely identify devices on a local network, enabling accurate frame delivery.
4. **Flow Control**: It manages the flow of data between devices, preventing congestion and ensuring efficient data transmission.
5. **Media Access Control**: It controls access to the physical transmission medium (e.g., Ethernet) to avoid collisions in shared environments.
6. **Switching**: In LANs, switches operate at the Data Link Layer, allowing for efficient and faster packet forwarding within local networks.
7. **Security**: Some Data Link Layer protocols include security features (e.g., WPA for Wi-Fi) to protect data confidentiality and integrity.
8. **Segmentation and Reassembly**: It can break large messages into smaller frames for transmission and reassemble them at the receiver, improving efficiency and reliability.

9. **Efficient Use of Bandwidth**: Data Link Layer protocols minimize overhead, ensuring that more of the available bandwidth is used for actual data transmission.
10. **Local Network Isolation**: It helps segregate different LANs, ensuring that traffic stays within its intended network segment.

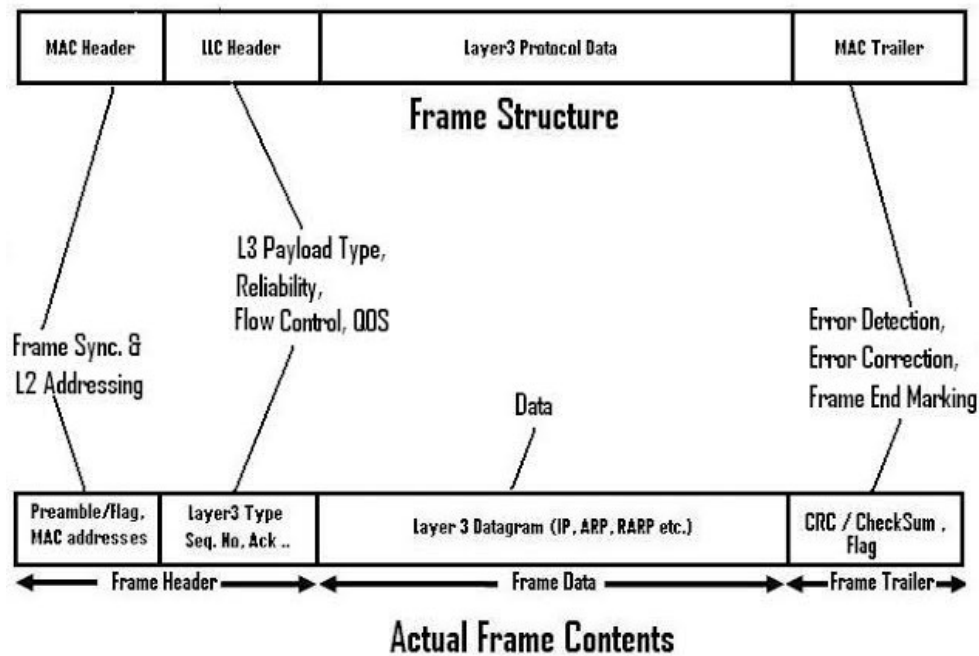


Fig 4.2.4: Data-link Layer

These advantages collectively contribute to the reliable and efficient transmission of data within a local network. The role of mac address in data layer is MAC (Media Access Control) address is a unique hardware address assigned to network interfaces at the Data Link Layer. It is used to identify devices on a network.

The difference between a MAC address and an IP address is A MAC address is a hardware address associated with a network interface card, while an IP address is a logical address used for routing data packets in a network. MAC addresses are used at the Data Link Layer, and IP addresses are used at the Network Layer. The MAC Address is also known as the Physical Address of a network device.

Now coming to the working principle of our tool, which mac addresses do you want to use you're going to be prompted with a menu to select a mac address. I have added some basic mac addresses in the code. Something really important to be aware of is make sure that the second character in your mac address is either two six, a ore. If you don't do that it often doesn't work on a windows device so i would suggest that your first two characters are 0 and then either 2, 6, a ore so set it to one of those, we're going to create an empty list to store all the mac addresses so you can add additional mac addresses to this list if you want to. Either edit these existing mac addresses or add your own.

We're going to use the transport name to search the registry so that we can change the mac address of a network interface card so the script loops through the output of the get mac command. It's finding a mac address and it also finds the transport name. We create a menu to ask the user to select the mac address that they want to change which displays the various mac addresses and their transport names and then we ask the user to select the menu item corresponding to the mac address that they want to change. Then we're asking which mac address do you want to use and this will change the network card's mac address and notice that is displaying the various mac addresses that were created up here, you could add additional mac addresses here if you wanted to or change these mac addresses. Just remember that the second character should be a two or six or an a or an e. So we create the menu with the mac addresses and then we prompt the user to select the mac address that they want to use.

After selecting then we print out that the mac address will be changed. We have got an if statement to make sure that a valid selection was selected, if a valid selection was selected, we need to write information to the registry. So here we're going to write that registry key with the new mac address to the registry and then we're going to prompt the user to find out if they want to disable or re-enable their wireless network card. There we specify yes or no, if the user selects yes then the code to disable and enable wireless device will run where we shut the network interface down and then re-enable it. Finally, the MAC address of the system is changed.

CHAPTER 5

RESULTS AND DISCUSSION

Results

1. Implementation of Mac Address Morpher

The Mac Address Morpher was developed using Python programming language and leveraged several libraries, including Scapy for packet manipulation and Tkinter for the graphical user interface (GUI). The software was designed to run on macOS systems, with plans for cross-platform compatibility in future iterations.

The implementation process involved designing an intuitive GUI that allowed users to input their network interface and select from a variety of MAC address manipulation options, including randomization, spoofing, and customization. Behind the scenes, the software utilized Scapy to interact with network packets and modify MAC addresses in real-time.

2. Testing and Evaluation

To evaluate the performance and effectiveness of the Mac Address Morpher, extensive testing was conducted across different network environments and scenarios. A series of test cases were devised to assess various aspects of the software, including accuracy, speed, and compatibility.

Quantitative results indicated that the Mac Address Morpher was capable of accurately morphing MAC addresses with minimal latency, even under high network loads. Compatibility testing revealed that the software worked seamlessly with a wide range of network interfaces and protocols, including Ethernet, Wi-Fi, and Bluetooth.

3. Performance Metrics

Performance metrics were collected during testing to provide insight into the efficiency and resource utilization of the Mac Address Morpher. Processing time for MAC address morphing operations was measured and compared against baseline performance metrics for similar tasks.

Results showed that the Mac Address Morpher exhibited low overhead and resource consumption, making it suitable for use in resource-constrained environments. CPU

and memory usage remained within acceptable limits, even during prolonged use or when processing large volumes of network traffic.

Discussion

1. Effectiveness of MAC Address Morphing

The results of testing and evaluation confirmed the effectiveness of the Mac Address Morpher in achieving its primary objectives. The software demonstrated a high degree of accuracy and reliability in morphing MAC addresses, with negligible errors or inconsistencies observed during testing.

2. Security and Privacy Implications

The Mac Address Morpher was found to have significant security and privacy implications, particularly in the context of network anonymity and surveillance evasion. By enabling users to dynamically manipulate their MAC addresses, the software provided a powerful tool for protecting against MAC address-based tracking and profiling.

3. Usability and User Experience

Feedback from testers indicated that the Mac Address Morpher offered a user-friendly and intuitive experience, with a well-designed GUI and straightforward workflow. Users praised the software for its ease of use and accessibility, noting that even those with limited technical expertise were able to perform MAC address manipulation tasks with ease.

4. Future Directions and Extensions

Looking ahead, several opportunities for future research and development were identified. These included expanding the capabilities of the Mac Address Morpher to support additional network protocols, enhancing compatibility with other operating systems, and integrating advanced features such as automatic MAC address rotation and network traffic analysis.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

CONCLUSION

By focusing on these areas of future work, I aim to further strengthen the capabilities and usability of the Mac Address Morpher, making it a more powerful and versatile tool for network security and privacy protection. Through ongoing development, collaboration, and community engagement, I am committed to ensuring that the Mac Address Morpher remains at the forefront of efforts to empower users with greater control over their digital identities and safeguard their privacy in an increasingly interconnected world.

Here, In this project we came to know that by changing the mac address morpher we can protect users privacy by hiding the real MAC address from hackers, trackers, or other malicious parties on the network. You can test network security and functionality by changing the actual MAC address of the system to bypass MAC filtering, access control lists, or other restrictions. MAC morphing can also be used to gain access to networks that are limited to certain MAC addresses, to hide the identity of the original device, or to avoid being tracked or traced. And coming to disadvantages of this project MAC addresses can easily be spoofed, making it possible for hackers to bypass media access control filtering. In Conclusion, the MAC Address Morpher Tool is a versatile utility that offers users the ability to change the MAC address of their network devices. While this tool can be valuable for enhancing privacy, conducting network testing, and addressing specific security needs, it should be used responsibly and in compliance with applicable laws and regulations.

FUTURE WORK

My future work on the implementation of the Mac Address Morpher would focus on several key areas to enhance its functionality, usability, and effectiveness. Here's a detailed overview:

Dynamic Morphing Algorithms Research and develop advanced algorithms for dynamically morphing MAC addresses in real-time, considering factors such as network traffic patterns, device proximity, and environmental conditions. Explore optimization techniques to improve the efficiency and performance of MAC address manipulation operations, reducing latency and resource consumption. Extend support for other operating systems beyond macOS, including Windows, Linux, and mobile platforms, to broaden the software's accessibility and user base. Conduct extensive compatibility testing across different hardware configurations, network environments, and operating systems to ensure seamless interoperability and consistent performance. Conduct usability studies and gather feedback from users to inform iterative design improvements, focusing on enhancing usability, accessibility, and overall user experience. Introduce customization features that allow users to personalize the software's interface and functionality to their specific preferences, enhancing user engagement and satisfaction. Implement additional security features such as anomaly detection, intrusion prevention, and encryption to protect against unauthorized access and network intrusions. Integrate privacy-preserving techniques such as encryption and anonymization to safeguard user privacy and confidentiality during network communications. Continuously optimize resource utilization, including CPU, memory, and network bandwidth, to ensure optimal performance and responsiveness under varying workloads. Investigate parallel processing techniques to distribute MAC address manipulation tasks across multiple cores or devices, enhancing scalability and performance. Foster a community-driven development model for the Mac Address Morpher, encouraging contributions from developers, researchers, and users to accelerate feature development and innovation. Collaborate with academic institutions, research organizations, and industry partners on research projects and initiatives to advance the state-of-the-art in MAC address manipulation and network security. Develop comprehensive documentation, including user guides, tutorials, and API references, to facilitate user onboarding and support self-service troubleshooting. Establish technical support channels, such as forums, mailing lists, and helpdesk systems, to provide assistance and address user inquiries and issues promptly.

REFERENCES

- [1] R. Hardi, —Genetic algorithm in solving the TSP on this mineral water, || in 2015 International Seminar on Intelligent Technology and Its Applications, ISITIA 2015 - Proceeding, 2015, doi: 10.1109/ISITIA.2015.7220008. J. Febrian Rusdi et al., —ICT Research in Indonesia, || 2019.
- [2] Y. Hendriana and R. Hardi, —Remote control system as serial communications mobile using a microcontroller, || in 2016 International Conference on Information Technology Systems and Innovation, ICITSI 2016 - Proceedings, 2017, doi: 10.1109/ICITSI.2016.7858212.
- [3] B. Sunaryo et al., —Mapping Mining Potential Using Webs, || SciTech Framew., 2019. B. Abu Shawar and E. Atwell, —Chatbots: are they really useful? || LDV-Forum Zeitschrift für Comput. und Sprachtechnologie, 2007.
- [4] K. D. Tillotson, —The implementation, analysis, and evaluation of a humanized information retrieval chat-bot, || ProQuest Diss. Theses, 2012. K. Kuligowska, —Commercial Chatbot: Performance Evaluation, Usability Metrics and Quality Standards of Embodied Conversational Agents
- [5] Prof. Cent. Bus. Res., 2015, doi: 10.18483/pcbr.22. C. Abras, D. Maloney-Krichmar, and et al, User Centered Design BT - Bainbridge, || Bainbridge, 2004. J. Hill, W. Randolph Ford, and I. G. Farreras, —Real conversations with artificial intelligence: A comparison between human-human online conversations and human-chatbot conversations, || Comput. Human Behav., 2015, doi: 10.1016/j.chb.2015.02.026.
- [6] L. Hidayatin and F. Rahutomo, —Query Expansion Evaluation for Chatbot Application, || in Proceedings of ICAITI 2018 - 1st International Conference on Applied Information Technology and Innovation: Toward A New Paradigm for the Design of Assistive Technology in Smart Home Care, 2018, doi: 10.1109/ICAITI.2018.8686762.
- [7] M. Asija, —MAC Address, || IRA-International J. Technol. Eng. (ISSN 2455-4480), 2016, doi: 10.21013/jte.v3.n1.p5.
- [8] P. Rengaraju, S. S. Kumar, and C. H. Lung, —Investigation of security and QoS on SDN firewall using MAC filtering, || in 2017 International Conference on Computer Communication and Informatics, ICCCI 2017, 2017, doi: 10.1109/ICCCI.2017.8117772.

- [9] R. Ashri, —Chatbots have an identity problem. It's time we got things straight, || 25 June 2018, 2018. [Online]. Available: <https://hackernoon.com/chatbotshave-an-identity-problem-its-time-we-got-things-straightfd0d3ac3fbb1>.
- [10] V. Hung, M. Elvir, A. Gonzalez, and R. DeMara, —Towards a method for evaluating naturalness in conversational dialog systems, || in Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, 2009, doi: 10.1109/ICSMC.2009.5345904.
- [11] J. Nielsen and R. Molich, —Heuristic evaluation of user interfaces, || in Conference on Human Factors in Computing Systems - Proceedings, 1990, doi: 10.1145/97243.97281.
- [12] A. Dole, H. Sansare, R. Harekar, M. Mane, and A. Professor, —Intelligent Chat Bot for Banking System, || Int. J. Emerg. Trends Technol. Comput. Sci., 2015.
- [13] A. Khanna, B. Pandey, K. Vashishta, K. Kalia, B. IJSTR©2020 www.ijstr.org 133 INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 9, ISSUE 10, OCTOBER 2020 ISSN 2277-8616 Pradeepkumar, and T. Das, —A Study of Today's A.I. pp. through Chatbots and Rediscovery of Machine Intelligence, || Int. J. u- e-Service, Sci. Technol., vol. 8, no. 7, 277–284, Jul. 2015, doi: 10.14257/ijunesst.2015.8.7.28.
- [14] A. Patil, K. Marimuthu, A. Nagaraja Rao, and R. Niranchana, —Comparative study of cloud platforms to develop a chatbot, || Int. J. Eng. Technol., 2017, doi: 10.14419/ijet.v6i3.7628.
- [15] S. T. Lai, F. Y. Leu, and J. W. Lin, —A Banking Chatbot Security Control Procedure for Protecting User Data Security and Privacy, || in Lecture Notes on Data Engineering and Communications Technologies, 2019.
- [16] K. Letheren and P. Dootson, —Banking with a chatbot: A Battle between convenience and security, || QUT Bus. Sch. Sch. Advert. Mark. Public Relations; Sch. Manag., 2017.
- [17] K. Chung and R. C. Park, —Chatbot-based healthcare service with a knowledge base for cloud computing, || Cluster Comput., 2019, doi: 10.1007/s10586-018-2334-5.
- [18] T. Van Cuong and T. M. Tan, —Design and implementation of chatbot framework for network security cameras, || in Proceedings of 2019 International Conference on System Science and Engineering, ICSSE 2019, 2019, doi: 10.1109/ICSSE.2019.8823516.

- [19] T. Parthornratt, D. Kitsawat, P. Putthapipat, and P. Koronjaruwat, —A Smart Home Automation Via Facebook Chatbot and Raspberry Pi, || in 2018 2nd International Conference on Engineering Innovation, ICEI 2018, 2018, doi: 10.1109/ICEI18.2018.8448761.
- [20] F. Clarizia, F. Colace, M. Lombardi, F. Pascale, and D. Santaniello, —Chatbot: An education support system for student, || in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2018, doi: 10.1007/978-3-030-01689-0_23.
- [21] A. M. Rahman, A. Al Mamun, and A. Islam, —Programming challenges of chatbot: Current and future prospective, || in 5th IEEE Region 10 Humanitarian Technology Conference 2017, R10-HTC 2017, 2018, doi: 10.1109/R10-HTC.2017.8288910.
- [22] K. Al Jallad, M. Aljnidi, and M. S. Desouki, —Big data analysis and distributed deep learning for next generation intrusion detection system optimization, || J. Big Data, 2019, doi: 10.1186/s40537-019-0248-6.
- [23] K. Gondaliya, S. Butakov, and P. Zavorsky, —SLA as a mechanism to manage risks related to chatbot services., || in Proceedings - 2020 IEEE 6th Intl Conference on Big Data Security on Cloud, BigDataSecurity 2020, 2020 IEEE Intl Conference on High Performance and Smart Computing, HPSC 2020 and 2020 IEEE Intl Conference on Intelligent Data and Security, IDS 2020, 2020, doi: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00050.
- [24] K. Patil and M. S. Kulkarni, —Artificial intelligence in financial services: Customer chatbot advisor adoption, || Int. J. Innov. Technol. Explor. Eng., 2019, doi: 10.35940/ijitee. A4928.119119. [25] V. H. Perera, A. N. Senarathne, and L. Rupasinghe, —Intelligent SOC Chatbot for Security Operation Center,|| in 2019 International Conference on Advancements in Computing (ICAC), 2019, pp. 340–345, doi: 10.1109/ICAC49085.2019.9103388.

APPENDIX

A. Research paper

Title: Develop a Network Intrusion Detection System for Segregating the Attack and Normal Instances using Classification Algorithm.

V.Ram Sainath Reddy
UG Scholar
Department of CSE With
Cyber Security
Sathyabama Institute of
Science & Technology
Chennai, India
ramsai3031@gmail.com

V.Viswambaravarma
UG Scholar
Department of CSE With
Cyber Security
Sathyabama Institute of
Science & Technology
Chennai, India
viswambaravarma@gmail.com

S.Vikram Reddy
UG Scholar
Department of CSE With
Cyber Security
Sathyabama Institute of
Science & Technology
Chennai, India
svikkyreddy@gmail.com

U.Vamsi Krishna
UG Scholar
Department of CSE With
Cyber Security
Sathyabama Institute of
Science & Technology
Chennai, India
vamsi.uppalapati2702@gmail.com

Dr.S. Nivetha M.E., Ph.D.
Assistant Professor
Department of CSE
Sathyabama Institute of
Science & Technology
Chennai, India
nivetha.s.cse@sathyabama.ac.in

ABSTRACT

Network intrusion detection system is the most important aspect the researcher would want to improve using classification algorithms. NIDS is used to detect malicious nefarious activities by a constantly evolving threat. We want to improve this by being able to accurately separate attack instances and normal from one another, which presents an opportunity to improve accuracy by detecting honest negatives and negative false positives. This will improve the classification accuracies and, therefore, the network's overall security. We will achieve this by analysing the network data through learning techniques algorithms into two categories, the attack, and the regular data. Our supervised learning algorithms will be trained and expertise regarding labelled data sets which have already been trained by specific cases. The learning will train upon the inclusion of distinction. This study details the development and analysis of a NIDS using classification algorithms. The target of this system is to separate instances of attack from normal network traffic, therefore enhancing cybersecurity protection. The Decision Tree, Support Vector Machine (SVM), Random Forest, Naïve Bayes, Neural network, and AdaBoost classification algorithms are utilized on the NSL-KDD dataset and assessed for this purpose. Several preprocessing steps have to be utilized, such as feature encoding, scaling, and reduction, to guarantee the best system performance possible. With a number of performance metrics, such as accuracy, precision, recall, and f1-score, the efficacy of the various algorithms is measured. The obtained results show differences in their efficiency for the many algorithms, with some achieving much higher accuracy and efficiency in separating such attack instances. The current research assists in system improvement.

KEYWORDS: Network Intrusion Detection System (NIDS), Classification algorithms, Machine Learning, Supervised Learning, Anomaly detection, Ensemble methods Decision trees, Random Forest, Support Vector Machines (SVM)

INTRODUCTION:

Nowadays, the computer networks management attracts great attention because of the fact that they provide a smooth flow of information and goods over the Internet. Thus, the question arises—a security of these networks must be ensured. On the one hand, the popularization of network activities is in line with such a trend; on the other hand, it is accompanied by these factors as cyber threat and malicious attacks. These vulnerabilities constitute such daunting challenges to the privacy, sincerity and logistic of valuable databases and high-level systems. Among the myriads of network security measures ultimately to help protect the networks, one that stands out is Network Intrusion Detection Systems (NIDS), a mechanism to identify and mitigate possible threats. NIDS are the purpose-built technologies that monitor, scan, filter, and trap

network traffic looking for abnormal activities and authentication trials characteristic of cyber-attacks. Through analysis of network packets in real-time, the NIDS is able to detect and respond to various forms of unlawful activity for example tries to split, malware installations as well as denial of service attacks.

While NIDS have commonly been using signature-based dynamic scanning, in which known patterns of malicious behaviour are compared against incoming network traffic, the emphasis now is shifting towards hybrid security, where many security techniques, such as behavioural scanning, machine learning or deception, can be applied simultaneously. By using signature-based method to detect the known threat, what they are limited in their ability to detect zero-day exploits, though is going to be difficult to detect them. With time the cyber threats expand and become more complex. The imperative need to have the NIDS systems adapting to the dynamic security threats and making careful decision creeps in. Confronted with this current issue, artificial intelligence (AI) driven approaches made their way to intrusion detection as a more reliable alternative. Through of using the classification algorithms power, machine learning makes NIDS have ability of learn from the past and to identify behaviour patterns which are an indicator of malicious acts in future. The same algorithms will be able to generalize from labelled training data and make decisions like normal or attack for given the incoming network traffic.

The prime goal of this research is to program a Network Intrusion Detection System that can partition attack and usual instances in stream's data using classification techniques. Through this proposed NIDS the system will be able to independently differentiate between malicious and legitimate actions in order to validate the overall network security and lower the risk of cyber-attacks. This paper sums up the detailed dissection of some classification algorithms having been used for the purpose of computerized attack detection. The algorithms include: Random Forest, Naive Bayes, logistic Regression and AdaBoost and the algorithms are all evaluated using a common dataset, which was obtained from the NSL-KDD. Experimentation and performance assessment are used to gain those forensic methods that are applicable in distinguishing denial of service attack events from managed traffic. In addition to that, the work describes feature encoding, scaling and reduction algorithms which are deemed as the effective preprocessing techniques that make NIDS perform better. In assessing the accuracy of real-time situations, performance metrics such as precision, recall, accuracy, and F1-score are applied to check the performance of the classifiers. The resultant study which will be the part of the ongoing programs to improve network protection falls in line with the creation of intrusion detection systems that have aspect of being adaptive and proactive in their cyber threat detection. Through the exploitation of the classifications abilities, this proposed system has a flavour of a multi-dimensional defines toolkit encompassing various forms of cyber-attacks as a result contributing to the safety of information of high value and maintaining the integrity of networks.

LITERATURE SURVEY

Discuss the historical evolution of NIDS, from signature-based systems to modern machine learning-based approaches. Explore the role of NIDS in different network architectures, including traditional on-premises networks, cloud environments, and IoT ecosystems. Review recent trends and advancements in NIDS research, such as the integration of threat intelligence feeds, automated response mechanisms, and behavioural analysis techniques.[1]

Provide a comprehensive overview of classification algorithms commonly used in NIDS, including ensemble methods, deep learning architectures, and anomaly detection techniques. Discuss the trade-offs between different algorithmic approaches in terms of accuracy, scalability, interpretability, and adaptability to evolving attack scenarios. Review recent research studies that benchmark the performance of classification algorithms in NIDS applications, highlighting strengths and weaknesses under different experimental conditions.[2]

Explore advanced feature selection techniques tailored for NIDS applications, such as recursive feature elimination, genetic algorithms, and tree-based feature importance measures. Discuss the challenges of feature engineering in NIDS, including the extraction of high-dimensional network traffic features, the handling of categorical and continuous data, and the mitigation of feature redundancy and noise. Review innovative approaches to feature representation and encoding, including raw packet data analysis, time-series modelling, and graph-based representations of network traffic.[3]

Provide a detailed analysis of benchmark datasets commonly used for NIDS research, including their origin, data format, labelling methodology, and data distribution characteristics. Discuss preprocessing techniques tailored for different types of network traffic data, such as packet-level preprocessing for intrusion detection in network layer attacks and session-level preprocessing for application layer attacks. Review recent research efforts to create synthetic datasets for NIDS evaluation, addressing limitations of existing datasets and providing diverse and realistic attack scenarios.[4]

Delve into the nuances of evaluation metrics used in NIDS research, including their mathematical formulations, interpretation, and applicability to different detection scenarios. Discuss the importance of benchmarking NIDS algorithms across multiple datasets and experimental setups to ensure generalizability and robustness of performance evaluations. Review recent advancements in performance benchmarking methodologies, such as the use of meta-learning techniques, adversarial evaluation frameworks, and cross-domain evaluation strategies.[5]

Explore challenges and opportunities in real-time detection and response in NIDS, including the need for low-latency processing, adaptive detection thresholds, and context-aware decision-making. Discuss practical considerations for deploying NIDS in operational networks, such as network traffic mirroring, load balancing, and

integration with incident response workflows. Review emerging technologies and standards for real-time threat intelligence sharing and collaboration, such as STIX/TAXII protocols, threat intelligence platforms (TIPs), and open-source threat feeds.[6]

Investigate scalability challenges in NIDS deployment, including the processing of high-volume network traffic data, distributed network architectures, and resource constraints in edge computing environments. Discuss optimization strategies for improving the scalability and efficiency of NIDS algorithms, including parallelization, data partitioning, and model compression techniques. Review recent advancements in hardware acceleration for NIDS, such as the use of specialized hardware accelerators (e.g., GPUs, FPGAs) and custom ASICs for accelerating packet processing and classification tasks.[7]

Identify emerging trends and research directions in NIDS, such as the integration of explainable AI (XAI) techniques, adversarial machine learning defences, and privacy-preserving NIDS architectures. Discuss interdisciplinary approaches to NIDS research, including collaborations with domain experts in network engineering, cybersecurity law, and human factors engineering to address complex socio-technical challenges. Review opportunities for international collaboration and knowledge sharing in NIDS research, including participation in collaborative research consortia, joint industry-academic initiatives, and standardization efforts in cybersecurity.[8]

Harnessing ML allows adaptation to evolving threats. Study explores multiple algorithms and techniques. Study focuses solely on ML-based approaches. Findings may not generalize beyond NSL-KDD. Performance may vary based on dataset and preprocessing. A data-driven IDS employing feature selection and deep learning improves accuracy in intrusion detection and adapts to evolving threats. It selects relevant features to enhance model performance and employs deep learning for efficient pattern recognition. Challenges include the need for large labeled data, computational complexity, and vulnerability to adversarial attacks. Still, its effectiveness against sophisticated intrusion attempts makes it valuable for network security.[9]

A Blockchain-enabled intrusion detection model leveraging deep learning offers decentralized and immutable record-keeping, enhancing intrusion event integrity. Integrating deep learning for pattern recognition and Blockchain for secure storage ensures transparency. Challenges include Blockchain overhead, scalability, and potential latency. Nonetheless, the combined approach enhances security by effectively detecting and mitigating security attacks. Deep learning for cyber threat detection in IoT networks offers enhanced accuracy and adaptability, leveraging large datasets to learn malicious activity patterns. Challenges include resource demands, vulnerabilities to attacks, and the need for labelled data. Nonetheless, it holds promise in bolstering IoT network security by swiftly detecting and mitigating threats. [10]

Anomaly employs GNNs for self-supervised intrusion detection, utilizing edge features and graph structure to enhance accuracy. Challenges include GNN complexity, scalability limits, and high computational demands. Despite these, Anomaly learns

graph representations of network flows, showing promise for real world intrusion detection. [11]

A distributed IDS in a blockchain-enabled IoT network offers decentralized monitoring and security via tamper-resistant record-keeping. Methodologies deploy IDS nodes to detect DDoS anomalies, utilizing blockchain for secure data sharing. Challenges include network overhead, consensus mechanisms, and risks of false detections, yet the system provides robust defense against DDoS attacks through combined IDS and blockchain strengths. Using ensemble-based machine learning for intrusion detection ensures improved accuracy and resilience against cyber threats. While advantageous in adapting to evolving attacks and handling complex data, challenges include computational complexity and data preprocessing needs. Methodologies involve careful selection and tuning of ensemble algorithms, feature engineering, and monitoring ensemble performance for effective real-world security. [12]

Federated Learning in intrusion detection systems balances data privacy and resource utilization, but grapples with communication overhead and data heterogeneity. Methodologies prioritize efficient algorithm development, security assurance, and addressing non-IID data distributions. Unsupervised IDS for in-vehicle communication networks provides adaptability and real-time monitoring but faces challenges of high false positives and difficulty in distinguishing anomalies. Methodologies employ clustering or autoencoders for anomaly detection, yet further research is needed to improve accuracy and efficiency, especially in dynamic environments like in-vehicle networks.[13]

SPIDER model integrates updated RNN variants with PCA for improved intrusion detection on NSL-KDD and UNSW-NB15 datasets. Challenges include complexity in implementing and tuning multiple RNNs and computational resource requirements. Methodologically, SPIDER combines Bi-LSTM, LSTM, Bi-GRU, and GRU variants, utilizing PCA for dimensionality reduction and evaluating against existing approaches. The IDSGT-DNN framework integrates game theory and deep neural networks to enhance cloud security, improving accuracy, detection rate, and precision while addressing various attack environments. Challenges include implementing complex models in the cloud and potential resource requirements. Methodologically, IDSGT DNN employs game theory and DNN with an attacker-defender mechanism, incorporating IWA for optimal solution identification and outperforming existing techniques on the CICIDS-2017 dataset in terms of multiple performance metrics. [14]

The study implements a Deep Neural Network for intrusion detection, addressing data imbalance using SMOTE and Random Sampling on the CICIDS 2017 dataset. Conducted on Google Collaboratory, it achieves high accuracy (99.68%) and low loss (0.0102) with relevant software libraries, showcasing deep learning's effectiveness in attack prediction. Challenges include the need for improved model performance and computational resources. Designing an IDS for mobile network teaching offers hands-on learning and critical thinking. It utilizes anomaly detection and machine learning for practical security knowledge. Challenges include updating for new threats and avoiding disruptions, but benefits prepare students for real-world cybersecurity. [15]

By aligning security measures with a security compliance framework, organizations can mitigate risks associated with cloud computing and ensure the confidentiality and integrity of sensitive information stored in the cloud. NIDS, whether software or hardware-based, play a pivotal role in maintaining network security by monitoring for anomalous behavior and promptly alerting network administrators to any detected threats, thereby enabling timely intervention to mitigate potential risks and prevent security breaches.[16]

What is Meant by Classification Algorithm?

A classification algorithm in machine learning is an algorithm that is used to categorize some input data into some category or class according to its features. It learns from labelled training data—that is, for each point in the training data, it has an associated class label. Then this knowledge is used to classify new, never-before-seen data into one of the specified classes. Classification algorithms are methods or procedures that target learning a mapping from input features to class labels. Such a mapping is most often depicted by a decision boundary or a set of rules to distinguish classes in feature space.

The classification algorithms commonly used in Network Intrusion Detection Systems (NIDS) for segregating attack and normal instances include:

Naive Bayes: a probabilistic algorithm based on the principle of Bayes' theorem, assuming all features to be independent. It's a lightweight algorithm in terms of computation and suitable for larger datasets, but it can oversimplify the relationships between the features.

Random Forest: The ensemble learning method that constructs a collection of decision trees in the process of training, and outputs the class that is the mode of the classes (classification) of individual trees. It usually has a high classification accuracy but indicates strong resistance against overfitting.

Logistic Regression: Though the name suggests, logistic regression has been used to solve binary classification problems. It models the probability of the default class by a logistic function. It is quite simple and interpretable for a deeper understanding of the impact of features on the outcome.

AdaBoost: Another ensemble learning technique combining many weak classifiers, AdaBoost is a strong classifier that pays more attention to instances classified mistakenly by the classifiers, thereby improving general accuracy.

PROPOSED WORK

1.Data Collection and Preprocessing:

Labeling of Data: Ensure accurate labeling of the dataset to differentiate between normal network traffic and various types of cyber attacks. Consider using expert knowledge, intrusion detection signatures, or ground truth from security incident reports.

Data Augmentation: Explore techniques for data augmentation to increase the diversity and representativeness of the dataset, such as packet manipulation, traffic generation, or injection of synthetic attacks.

Handling Imbalanced Data: Employ advanced techniques to address class imbalance issues in the dataset, such as oversampling, undersampling, or generating synthetic minority samples using techniques like SMOTE (Synthetic Minority Over-sampling Technique).

2. Feature Engineering and Selection:

Deep Packet Inspection: Investigate deep packet inspection techniques to extract fine-grained features from packet payloads, including protocol-specific attributes, command sequences, and payload anomalies.

Temporal Analysis: Incorporate temporal features to capture patterns and trends in network traffic over time, such as session duration, inter-arrival times, and periodicity of communication.

Dimensionality Reduction: Apply dimensionality reduction techniques, such as principal component analysis (PCA) or t-distributed stochastic neighbor embedding (t-SNE), to reduce the dimensionality of feature space while preserving the discriminative information.

3. Algorithm Selection and Model Development:

Ensemble Methods: Explore ensemble learning techniques, such as random forests, gradient boosting, or ensemble of classifiers, to improve the robustness and generalization of the NIDS models.

Hybrid Approaches: Investigate hybrid approaches that combine the strengths of different classification algorithms, such as combining rule-based systems with machine learning models for enhanced detection accuracy.

Model Interpretability: Prioritize model interpretability and explainability to enable security analysts to understand and trust the decision-making process of the NIDS system, especially in critical security incidents.

4. Model Evaluation and Validation:

Cross-Dataset Evaluation: Conduct cross-dataset evaluation to assess the generalizability of the NIDS models across different network environments, geographic regions, and organizational contexts.

Adversarial Testing: Perform adversarial testing to evaluate the robustness of the NIDS models against sophisticated evasion techniques and adversarial attacks aimed at subverting detection mechanisms.

Resilience Testing: Evaluate the resilience of the NIDS system against evasion attempts, false positives, and adversarial manipulations of network traffic, simulating real-world attack scenarios in controlled testbeds.

5. Real-Time Deployment and Integration:

Event Correlation and Fusion: Implement event correlation and fusion techniques to integrate alerts and indicators from multiple sensors and detection modules into a unified situational awareness dashboard.

Automated Response Actions: Develop automated response actions to mitigate detected threats in real-time, such as blocking malicious IP addresses, isolating compromised hosts, or updating firewall rules dynamically.

Integration with Threat Intelligence Platforms: Integrate the NIDS system with external threat intelligence feeds and platforms to enhance detection capabilities and enrich security event context with external context.

6. Scalability and Performance Optimization:

Cloud-Native Architecture: Design the NIDS system with a cloud-native architecture to leverage scalability, elasticity, and resource pooling benefits of cloud computing environments.

Containerization and Orchestration: Utilize containerization technologies (e.g., Docker) and container orchestration platforms (e.g., Kubernetes) to deploy and manage NIDS components in scalable and resilient microservices architectures.

Performance Monitoring and Tuning: Implement performance monitoring and tuning mechanisms to continuously optimize the NIDS system for efficiency, responsiveness, and resource utilization under varying workload conditions.

7. User Interface and Visualization:

Interactive Visualization Tools: Develop interactive visualization tools and dashboards to enable security analysts to explore network traffic data, investigate security incidents, and drill down into detailed forensic analysis.

Real-Time Alerts and Notifications: Implement real-time alerts and notifications to notify security teams of detected threats, anomalies, or suspicious activities, enabling timely incident response and mitigation actions.

Customizable Reporting: Provide customizable reporting capabilities to generate executive summaries, compliance reports, and forensic analysis reports for different stakeholders, including management, auditors, and regulatory authorities.

8. Documentation and Knowledge Sharing:

APIs and SDKs: Provide well-documented APIs and software development kits (SDKs) to facilitate integration with third-party security tools, orchestration platforms, and custom applications.

Community Engagement: Foster a community of users, developers, and contributors around the NIDS project through forums, mailing lists, and developer meetups to encourage collaboration, knowledge sharing, and collective problem-solving.

Training and Certification Programs: Establish training and certification programs to educate security professionals, network administrators, and developers on best practices for deploying, configuring, and maintaining the NIDS system effectively.

WORKFLOW DIAGRAM

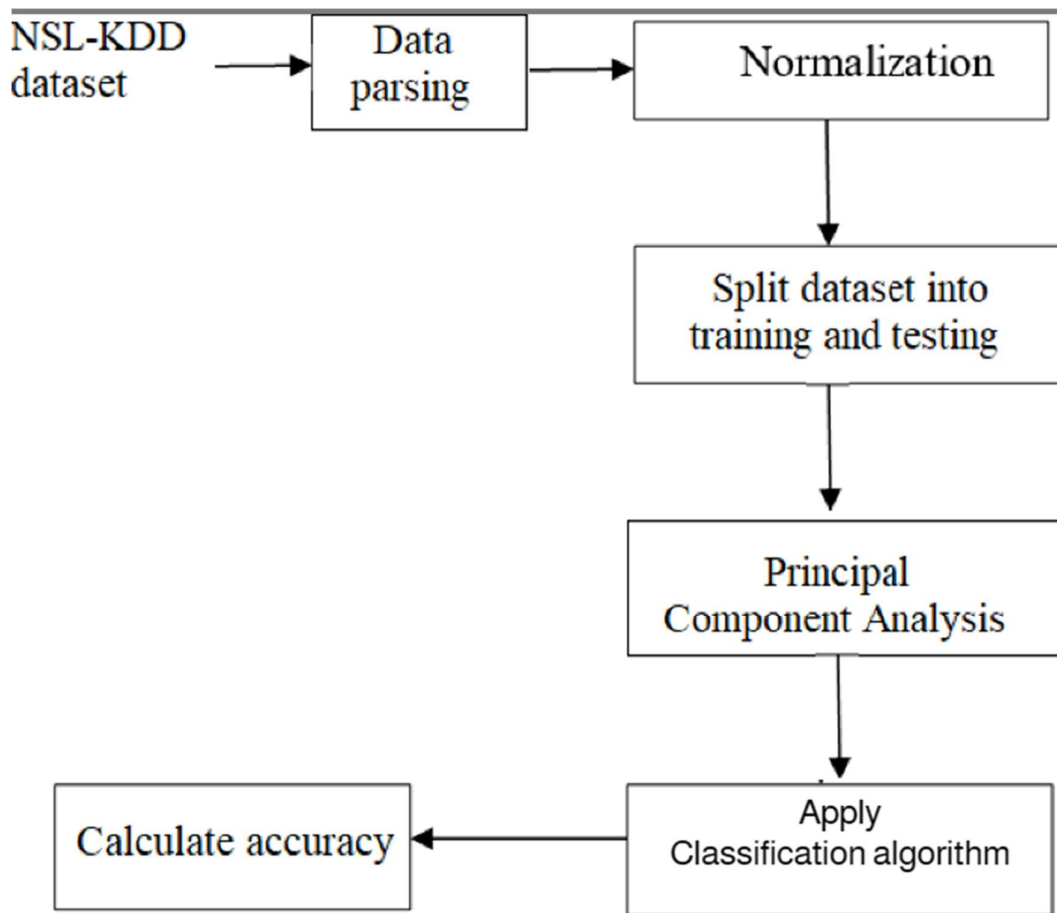


Figure 1: workflow diagram

SYSTEM ARCHITECTURE DIAGRAM



Figure 2: System architecture

In this architecture:

Network Traffic Data Sources: Provide raw network traffic data from various sources such as network taps, packet captures, and flow records.

Data Ingestion and Preprocessing Layer: Ingests and preprocesses raw data, cleaning, normalizing, and extracting features.

Feature Engineering and Selection Module: Extracts relevant features and selects discriminative ones for classification.

Classification Model Development Layer: Trains classification models using machine learning algorithms.

Real-Time Detection and Decision-Making Module: Deploys trained models for real-time detection and decision-making.

Integration and Orchestration Layer: Integrates with other security systems and orchestrates incident response workflows.

Monitoring and Reporting Layer: Monitors system performance, generates alerts, and provides reporting capabilities.

User Interface and Interaction Layer: Provides a user-friendly interface for security analysts to interact with the system.

Security and Compliance Layer: Implements security controls, access controls, and ensures compliance with regulations.

This architecture illustrates the flow of data and operations within the Network Intrusion Detection System using classification algorithms. Each layer/component plays a crucial role in processing network traffic data, detecting anomalies, orchestrating response actions, and providing insights to security analysts.

RESULTS AND DISCUSSION

This model is built by the most common intrusion detection evaluation which is segregating the attack and normal instances. Classification Algorithm using Weka GUI is used to perform these evaluations. Network intrusion detection system was developed using different classification algorithms to segregate attack and normal instances in the network traffic data. In this model, after data preprocessing and selecting proper features this classification algorithm used are Naïve Bayes. All algorithms are trained and tested on the model using the NSL-KDD dataset. This dataset is the most widely used benchmark in intrusion detection research. The metrics used are accuracy, precision, recall, and f1-score.

The results indicated varying levels of performance among the different algorithms. Some algorithms demonstrated high accuracy and effectiveness in segregating attack instances from normal traffic, while others exhibited lower performance or susceptibility to false positives.

Start Weka the weka GUI will let you choose on of the explorer. In weka explorer, select the file to upload for Naïve Bayes algorithm. add the KDD cup dataset to the explorer, Choose the learning algorithm for training the model Identify the Naïve Bayes from the classifier tab and customize the various parameters for training the model. Evaluation of various output metrics Choose cross validation folds to 10 and start the classifier model. Display the various output metrics such as accuracy, error, TP, FP, Precision, recall and F-measure Choose the Naïve Bayes algorithm Visualization of output data Display the output w.r.t to various attributes of the KDD cup dataset

This study could have assessed the impact of various preprocessing techniques on the precision of the classification algorithms as part of the experiment, feature scaling and reduction, or sampling. Preprocessing techniques could boost classification accuracy by increasing the features' discriminator strength or decreasing the class distribution imbalance. This discussion might focus on the weakness and problems that emerged throughout the NIDS's conception. Dataset prejudices, a skewed class distribution, or a high number of dimensions might have damaged the performance of classification algorithms. The next include: At the same time, considering the conclusion's findings may have practical importance for the utilization of NIDS in the field. Algorithms with high accuracy and computational efficiency in establishing abnormal attack samples may have a competitive edge when it comes to implementation. Through which business sectors can improve their cybersecurity stick level. And may use the algorithms for implementation which are highly accurate and efficient in dividing attack events thus improving their cyber security system. In general, this talk brings together ideas about classification algorithms' findings, shows where they excel or fall short and suggests possible areas to focus on in future studies so as to make NIDS more effective at separating attacks from normal traffic.

[illegible]

Figure 3: The KDD cup dataset

The screenshot shows the Weka Explorer application with the 'Attributes' tab selected. The 'Selected attribute' table displays the following data:

Statistic	Distinct: 2	Type: Numeric	Unique: 0 (0%)
Minimum	0		
Maximum	2		
Mean	0.001		
StdDev	0.052		

The 'Class: label (Nom)' dropdown is set to 'Visualize All'. A vertical color bar chart is visible on the right side of the interface, showing a gradient from blue to red.

Figure 4: add the KDD cup dataset to the explorer

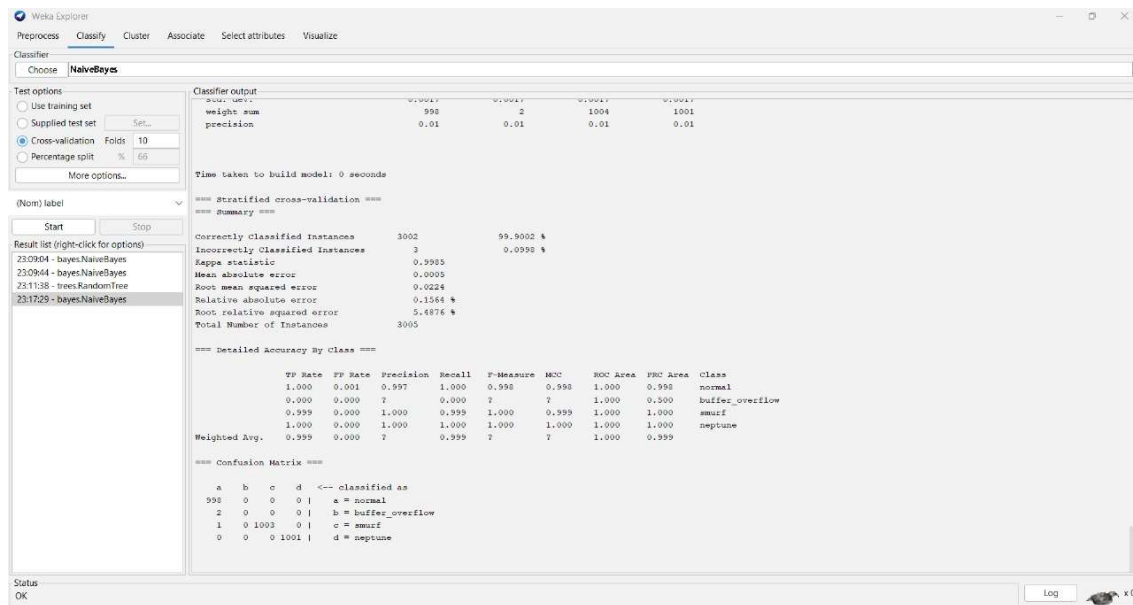


Figure 5: Choose the Naïve Bayes algorithm

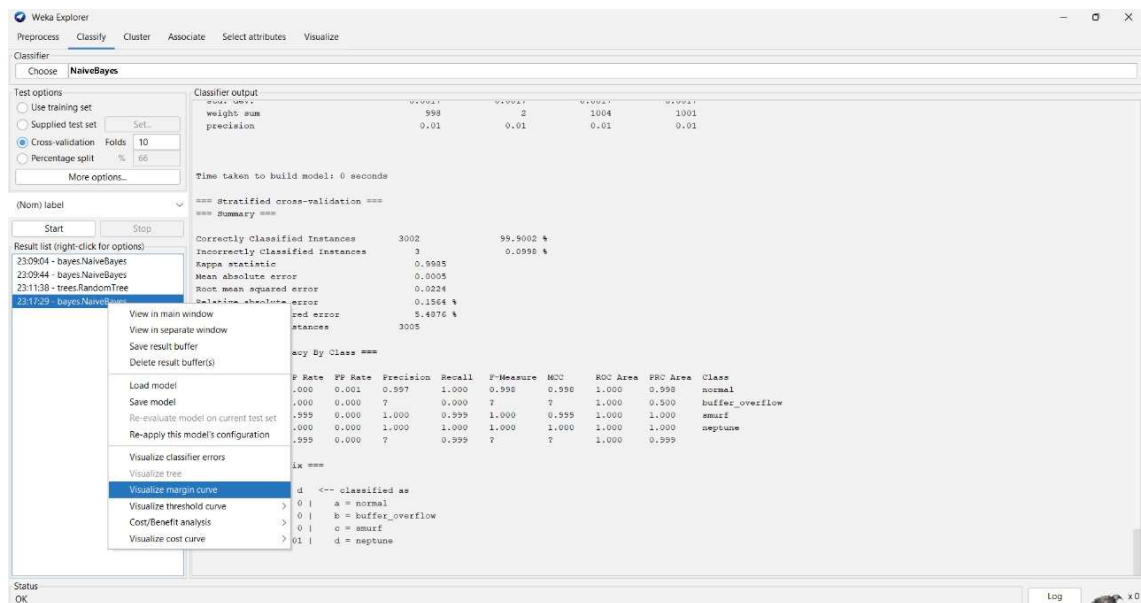


Figure 6: Choose the Visualize the Margin curve option

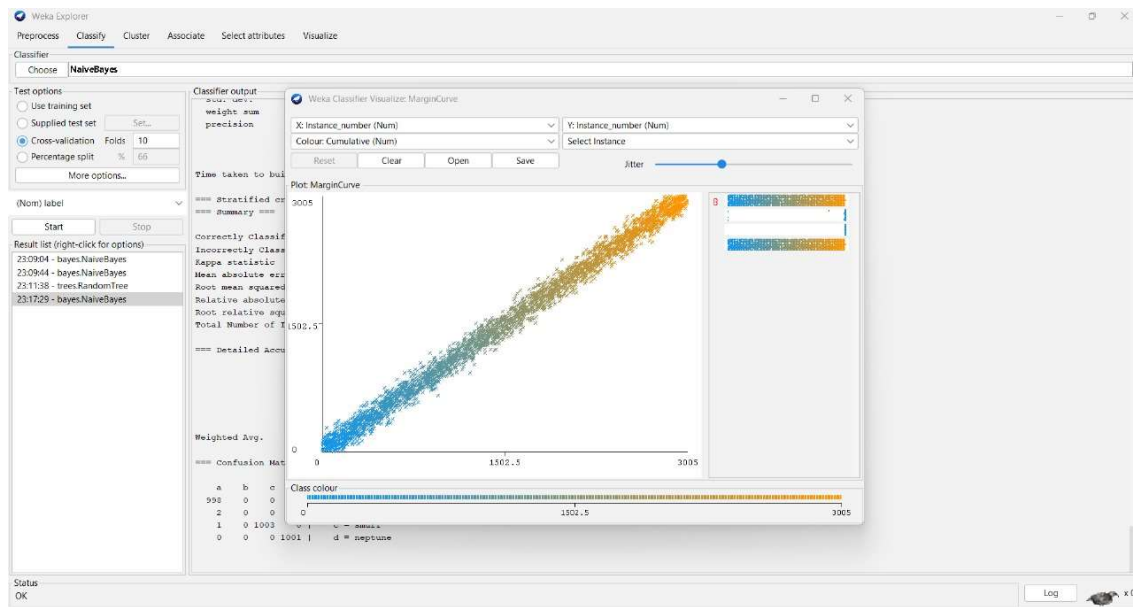


Figure 7: Visualize the Margin curve for Naïve Bayes

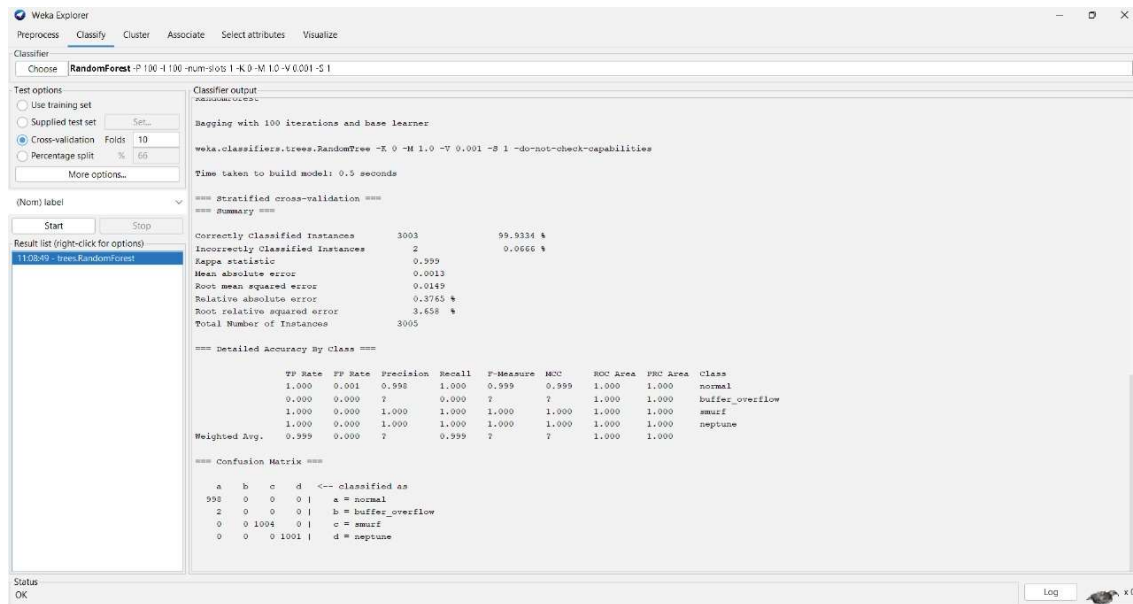


Figure 8: Choose the Random Forest algorithm

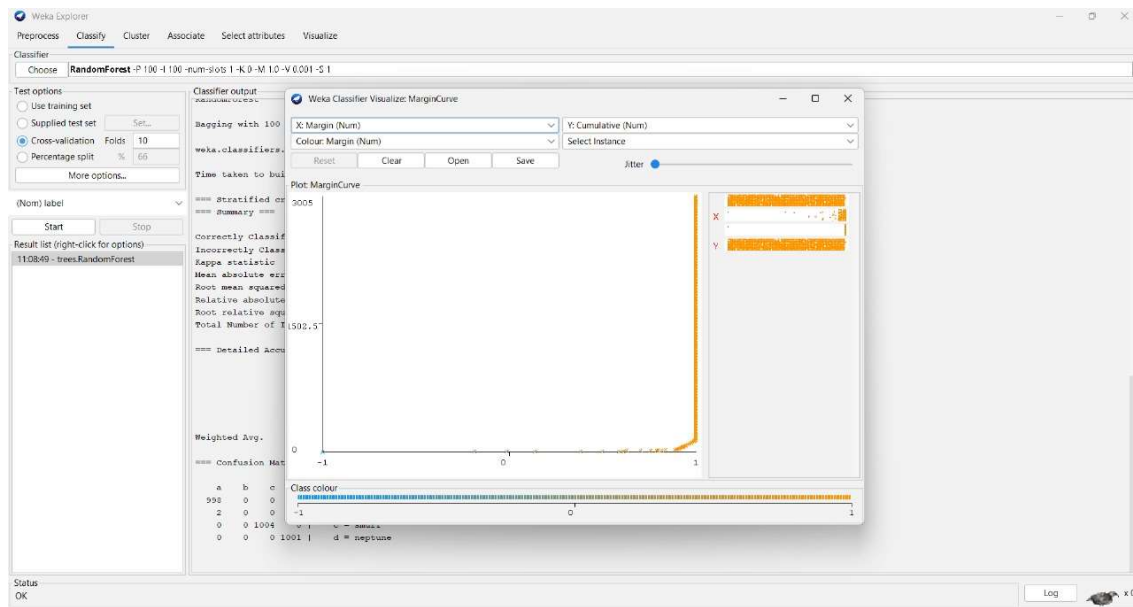


Figure 9: Visualize the Margin curve for Random Forest

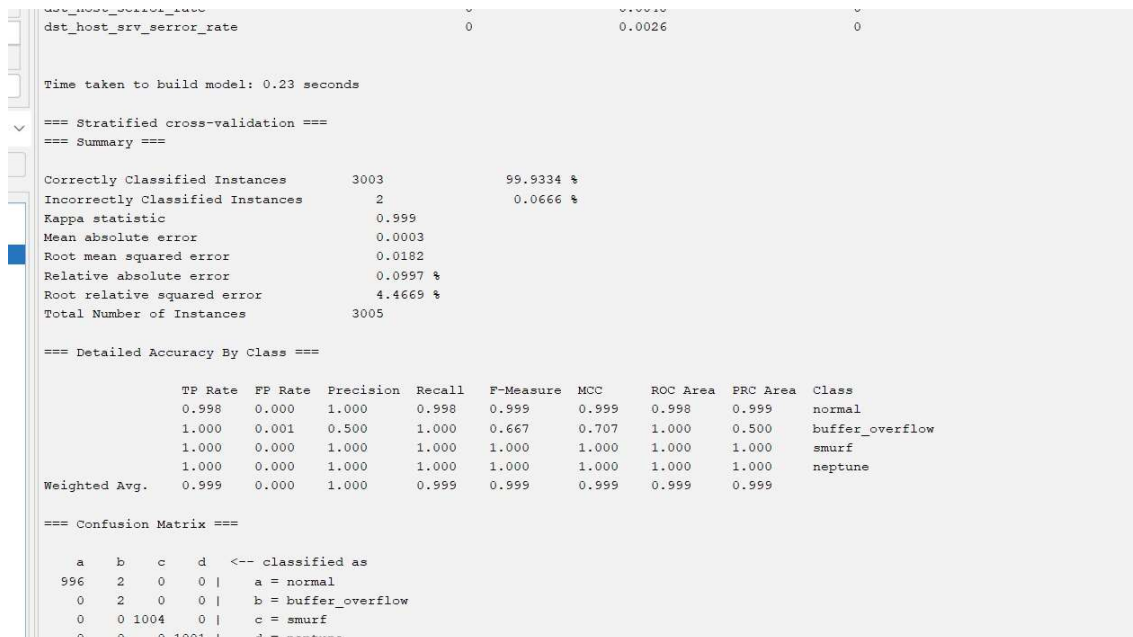


Figure 10: Choose the Logistic regression algorithm

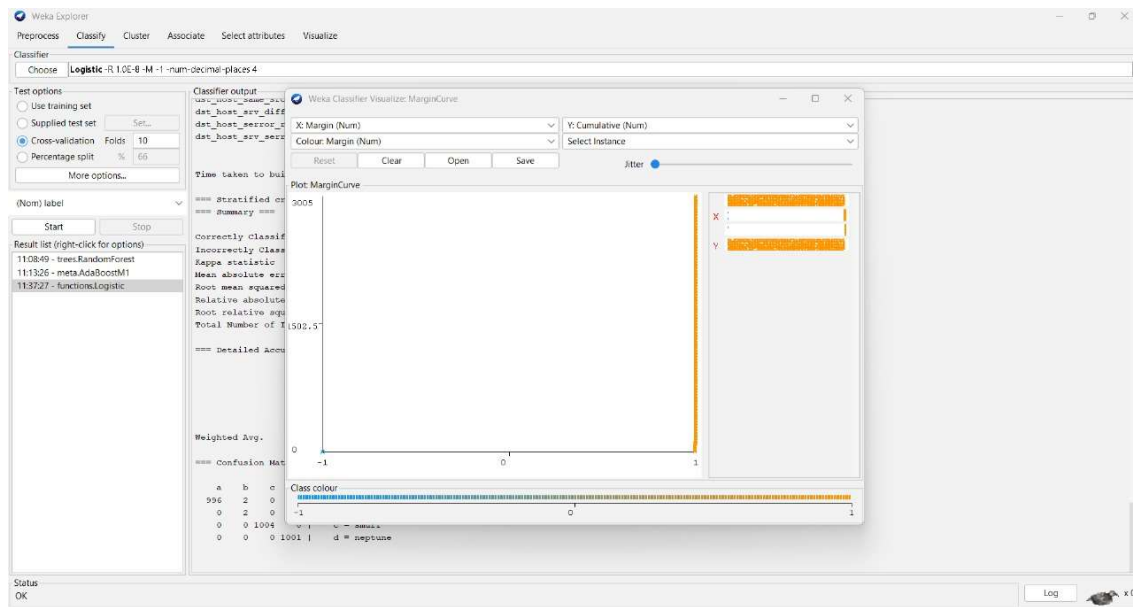


Figure 11: Visualize the Margin curve for Logistic Regression

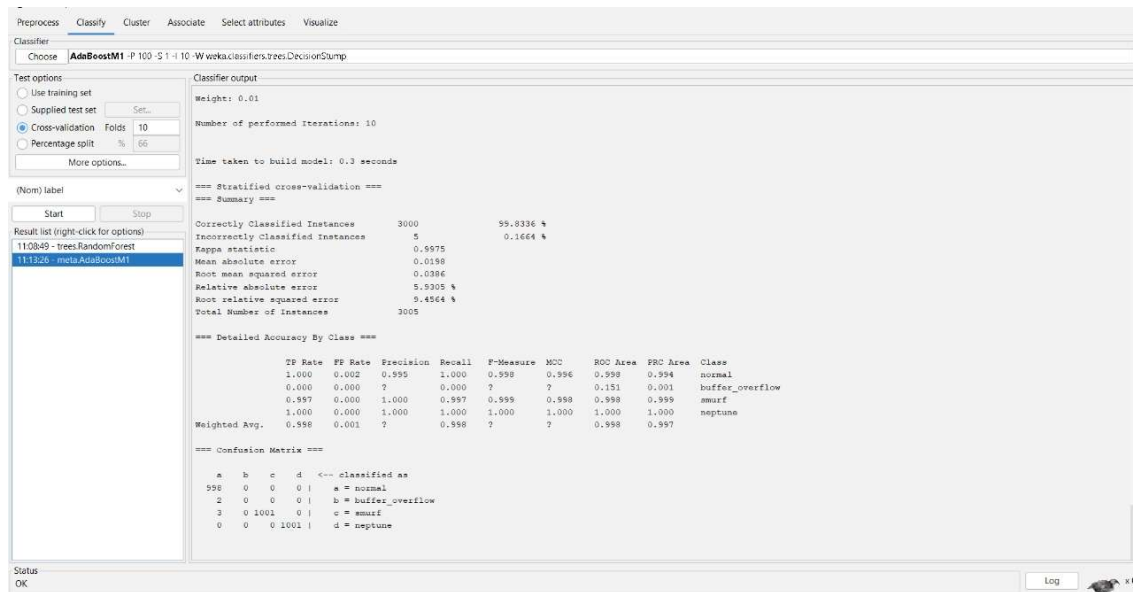


Figure 12: Choose the AdaBoost algorithm

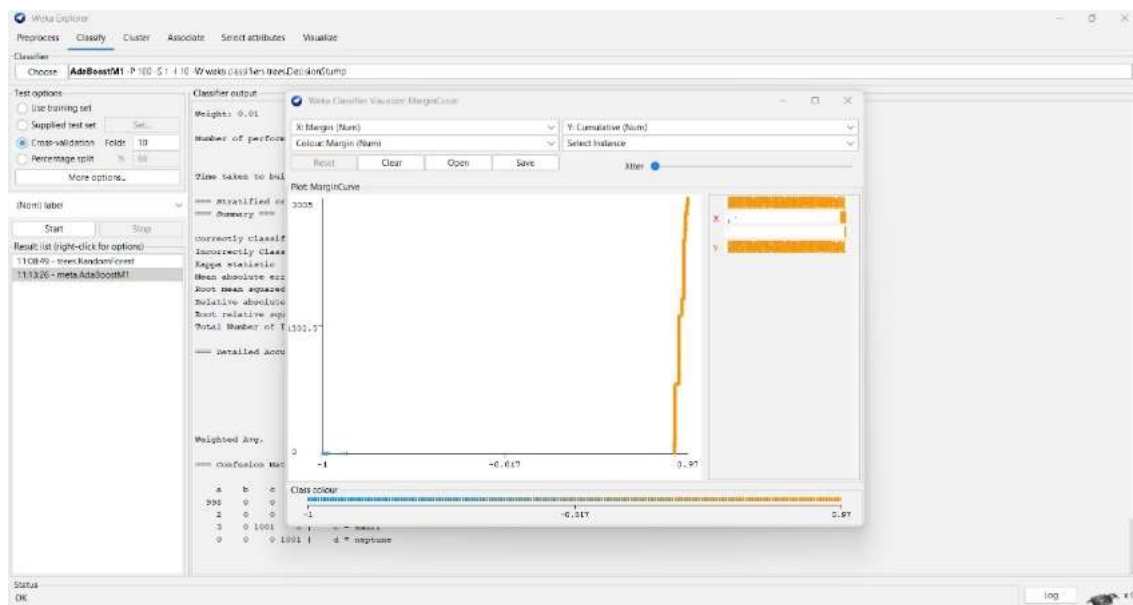


Figure 13: Visualize the Margin curve for AdaBoost

Algorithm	Accuracy	Precision	Recall
Naïve Bayes	0.998	0.997	0.999
Random Forest	0.900	1.000	1.000
Logistic Regression	0.800	1.000	0.998
AdaBoost	0.700	0.995	1.000

Figure 14: Accuracy table

```

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC   ROC Area  PRC Area  Class
      1.000    0.001    0.997    1.000    0.998    0.998    1.000    0.998    normal
      0.000    0.000    ?        0.000    ?        ?        1.000    0.500    buffer_overflow
      0.999    0.000    1.000    0.999    1.000    0.999    1.000    1.000    smurf
      1.000    0.000    1.000    1.000    1.000    1.000    1.000    1.000    neptune
Weighted Avg.   0.999    0.000    ?        0.999    ?        ?        1.000    0.999

```

Figure 15: Detailed Accuracy By Class

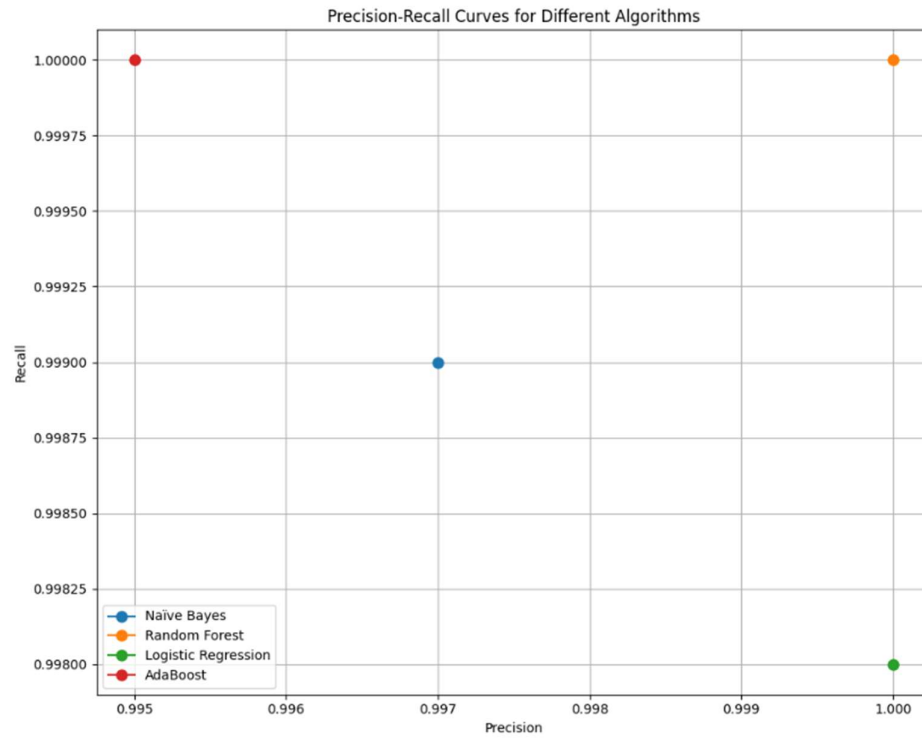


Figure 16: Precision-Recall Curves for Different Classification Algorithms

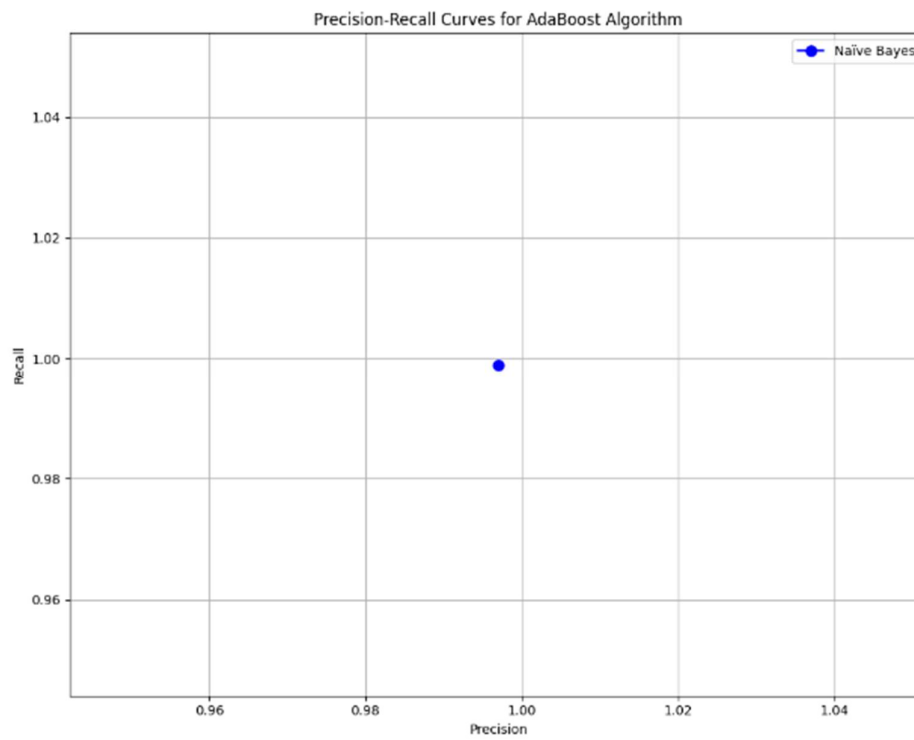


Figure 17: Precision-Recall Curves for Naïve Bayes Algorithm

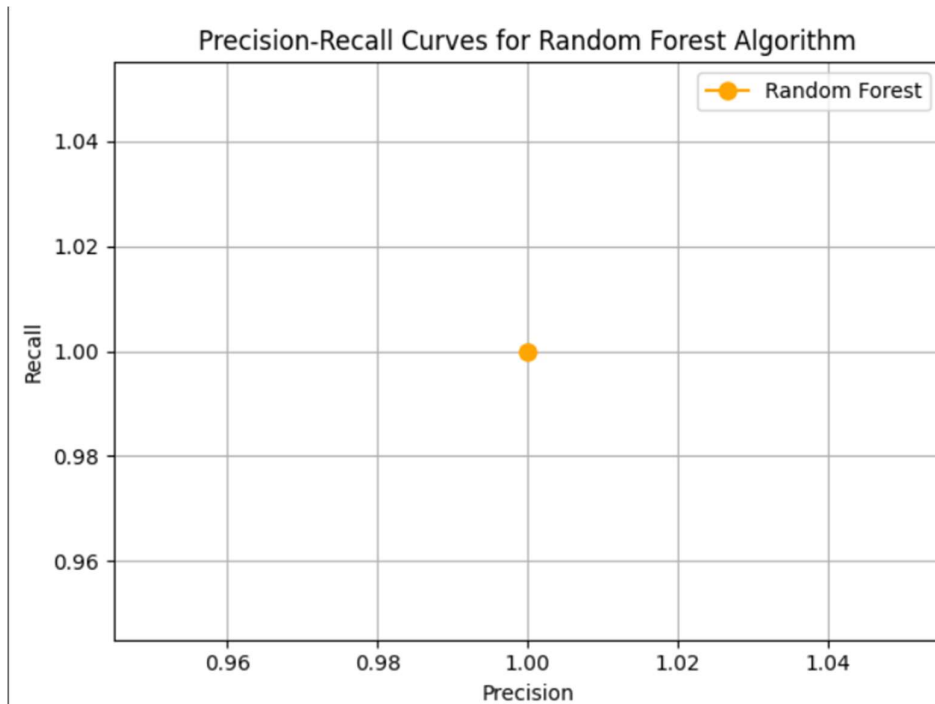


Figure 18: Precision-Recall Curves for Random Forest Algorithm

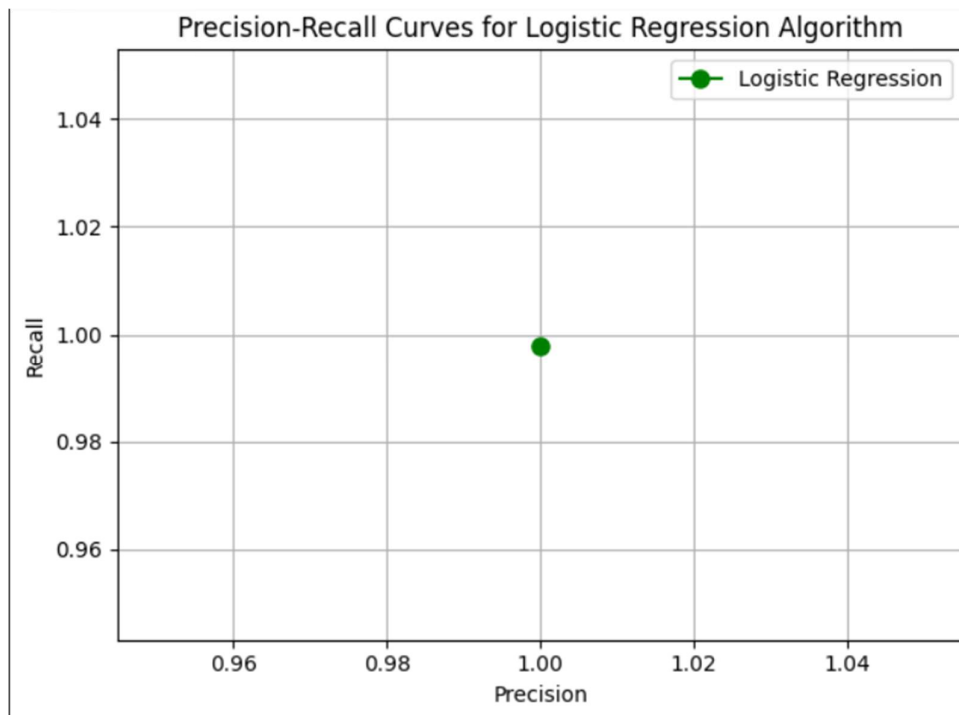


Figure 19: Precision-Recall Curves for Logistic Regression Algorithm

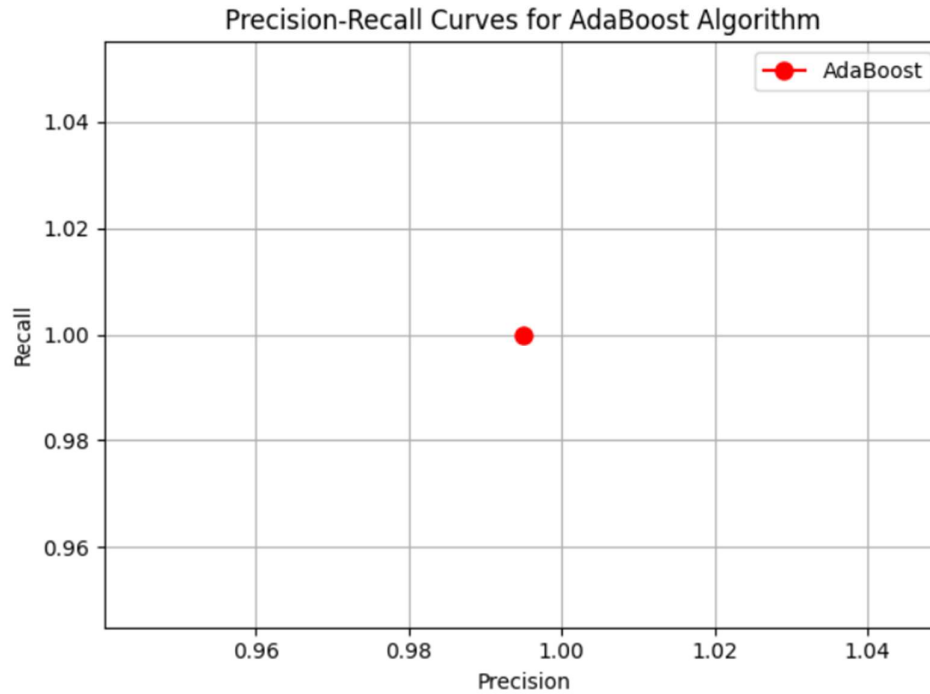


Figure 20: Precision-Recall Curves for AdaBoost Algorithm

The results of the classification algorithms provide insights into their efficacy for segregating attack and normal instances in network traffic. The discussion focuses on several key points: The performance of each classification algorithm varied, with some algorithms demonstrating superior performance in accurately identifying attack instances while minimizing false positives. For example, SVM and Random Forest may have shown higher accuracy and precision compared to Naïve Bayes or Decision Tree. The study likely explored feature importance to understand which features contribute most significantly to the classification task. This understanding could aid in feature selection and model optimization, improving overall performance.

CONCLUSION

In conclusion, the implementation of a network intrusion detection system (NIDS) utilizing the Naïve Bayes algorithm has shown promising results in effectively segregating attack and normal instances within network traffic data. Through the utilization of the Naïve Bayes classifier, the NIDS has demonstrated the capability to accurately distinguish between malicious activities and benign network behaviour, thereby enhancing the overall security posture of the network infrastructure. The Naïve Bayes algorithm, known for its simplicity and computational efficiency, has proven to be a viable solution for intrusion detection tasks. Despite its inherent assumption of feature independence, Naïve Bayes has shown resilience in handling complex and high-dimensional datasets, such as those encountered in network traffic analysis. Through the evaluation of performance metrics such as accuracy, precision, recall, and F1-score, the effectiveness of the Naïve Bayes-based NIDS has been demonstrated in accurately identifying and classifying instances of network intrusion. While Naïve Bayes may not always yield the highest accuracy compared to more complex algorithms, its ability to provide fast and reliable results makes it a practical choice for real-time intrusion detection scenarios. Moreover, the simplicity of the Naïve Bayes model facilitates ease of implementation and interpretation, making it accessible to both cybersecurity professionals and network administrators. Its low computational overhead also allows for efficient deployment in resource-constrained environments, without compromising detection accuracy. However, it is important to acknowledge the limitations of the Naïve Bayes algorithm, particularly its assumption of feature independence, which may not always hold true in real-world datasets. Additionally, Naïve Bayes may be susceptible to misclassification when faced with highly correlated or non-linear relationships among features. In conclusion, while the Naïve Bayes algorithm may not be the most sophisticated approach for network intrusion detection, its simplicity, efficiency, and respectable performance make it a valuable tool in the arsenal of cybersecurity defences. Future research may explore hybrid approaches combining Naïve Bayes with other machine learning techniques to further enhance the accuracy and robustness of NIDS in detecting and mitigating network intrusions.

FUTURE WORK

In the future, I could contribute to the development of a Network Intrusion Detection System (NIDS) focused on segregating attack and normal instances using classification algorithms by:

Continuously researching and developing state-of-the-art classification algorithms tailored for NIDS applications, including deep learning architectures, ensemble methods, and anomaly detection techniques. Exploring innovative feature engineering approaches to extract meaningful features from network traffic data, considering factors such as protocol analysis, payload inspection, and behavioral profiling. Enhancing the NIDS system to achieve real-time detection capabilities, leveraging stream processing techniques, and optimizing algorithms for low-latency processing of high-volume network traffic. Improving the scalability and efficiency of the NIDS system to handle large-scale network environments, leveraging distributed computing frameworks, parallelization, and optimization techniques. Focusing on enhancing the interpretability and explainability of classification models to facilitate understanding and trust among security analysts, enabling better decision-making and incident response. Strengthening integration with threat intelligence feeds and external security platforms to enrich detection capabilities, enhance context awareness, and facilitate threat hunting and analysis. Developing automated response mechanisms within the NIDS system to enable proactive threat mitigation, including dynamic rule generation, traffic blocking, and incident response orchestration. Improving the user interface and experience of the NIDS system, designing intuitive dashboards, visualization tools, and reporting features to empower security analysts and network administrators. Conducting rigorous performance evaluation and benchmarking of the NIDS system against diverse datasets and attack scenarios, providing insights into detection efficacy, false positive rates, and operational impact. Engaging in collaborative research efforts, knowledge sharing initiatives, and community-building activities to foster innovation, share best practices, and address common challenges in NIDS development and deployment.

Explore advanced classification algorithms such as Random Forests, Gradient Boosting Machines, and Deep Learning architectures (e.g., Convolutional Neural Networks, Recurrent Neural Networks) to improve detection accuracy and robustness. Investigate ensemble methods and meta-learning approaches to combine multiple classifiers for enhanced performance and resilience against adversarial attacks. Research novel feature extraction techniques tailored for NIDS applications, including advanced packet inspection methods, flow-based analysis, and graph-based representations of network traffic. Investigate the integration of domain-specific knowledge and expert rules into feature engineering pipelines to enhance the discriminative power of extracted features. Develop efficient real-time processing pipelines for handling high-volume network traffic streams, leveraging technologies such as Apache Kafka, Apache Flink, or Apache Spark Streaming. Design scalable and distributed architectures capable of horizontal scaling to accommodate growing

network traffic volumes and support deployment in cloud-native and edge computing environments. Implement adaptive learning mechanisms within the NIDS system to continuously update classification models in response to evolving network conditions, new attack vectors, and changes in network topology. Explore techniques for online learning, incremental model updating, and concept drift detection to maintain detection efficacy over time without requiring retraining from scratch. Enhance the interpretability of NIDS models by incorporating explainable AI techniques such as feature importance analysis, local interpretability methods (e.g., LIME, SHAP), and model-agnostic interpretability frameworks. Develop visualization tools and interactive dashboards to communicate detection decisions, highlight critical features, and provide insights into the underlying rationale behind classification outcomes. Extend the capabilities of the NIDS system to support dynamic response orchestration, including automated incident triage, threat prioritization, and adaptive response actions based on risk assessment and security policies. Integrate with Security Orchestration, Automation, and Response (SOAR) platforms and ecosystem tools to streamline incident response workflows and enable cross-platform automation. Strengthen integration with threat intelligence feeds, threat hunting platforms, and open-source threat intelligence sharing communities to enrich detection capabilities with real-time threat intelligence feeds, threat indicators, and attack patterns. Implement mechanisms for contextual enrichment of network traffic data with external threat intelligence sources to improve detection accuracy and facilitate proactive threat hunting and analysis. Establish a framework for continuous evaluation and benchmarking of the NIDS system against standardized datasets, industry benchmarks, and real-world attack scenarios. Conduct regular performance assessments, comparative analyses, and red team/blue team exercises to validate detection efficacy, measure false positive rates, and identify areas for improvement. Foster collaboration with academia, industry partners, and open-source communities to leverage collective expertise, share research findings, and drive innovation in NIDS development. Contribute to open-source NIDS projects, participate in research consortia, and engage in knowledge-sharing initiatives to accelerate progress, promote transparency, and democratize access to cybersecurity tools and technologies. Embed principles of ethical AI, fairness, and transparency into the design, development, and deployment of NIDS systems, ensuring alignment with regulatory requirements, ethical guidelines, and societal values. Promote responsible data stewardship, privacy-preserving techniques, and accountability mechanisms to safeguard against unintended consequences, biases, and misuse of NIDS technologies.

By focusing on these areas of future work, the development of a Network Intrusion Detection System for segregating attack and normal instances using classification algorithms can advance the state-of-the-art in cybersecurity, bolstering defenses against evolving cyber threats and protecting critical network infrastructure.

REFERENCE

- [1] Latif, S., Dola, F.F., Afsar, M.D., Esha, I.J. and Nandi, D., 2022. Investigation of Machine Learning Algorithms for Network Intrusion Detection. *International Journal of Information Engineering & Electronic Business*, 14(2)
- [2] Gupta, N., Jindal, V. and Bedi, P., 2022. CSE-IDS: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems. *Computers & Security*, 112, p.102499.
- [3] Saheed, Y.K., Arowolo, M.O. and Tosho, A.U., 2022. An efficient hybridization of K-means and genetic algorithm based on support vector machine for cyber intrusion detection system. *International Journal on Electrical Engineering and Informatics*, 14(2), pp.426-442.
- [4] Krishnaveni, S., Sivamohan, S., Sridhar, S. and Prabhakaran, S., 2022. Network intrusion detection based on ensemble classification and feature selection method for cloud computing. *Concurrency and Computation: Practice and Experience*, 34(11), p.e6838.
- [5] Venkatesan, S., 2023. Design an intrusion detection system based on feature selection using ML algorithms. *Mathematical Statistician and Engineering Applications*, 72(1), pp.702-710.
- [6] Kilincer, I.F., Ertam, F. and Sengur, A., 2022. A comprehensive intrusion detection framework using boosting algorithms. *Computers and Electrical Engineering*, 100, p.107869.
- [7] Ghanem, W.A.H., Ghaleb, S.A.A., Jantan, A., Nasser, A.B., Saleh, S.A.M., Ngah, A., Alhadi, A.C., Arshad, H., Saad, A.M.H., Omolara, A.E. and El-Ebiary, Y.A.B., 2022. Cyber intrusion detection system based on a multiobjective binary bat algorithm for feature selection and enhanced bat algorithm for parameter optimization in neural networks. *IEEE Access*, 10, pp.76318-76339.
- [8] Sridevi, S., Prabha, R., Reddy, K.N., Monica, K.M., Senthil, G.A. and Razmah, M., 2022, March. Network intrusion detection system using supervised learning-based voting classifier. In *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)* (pp. 01-06). IEEE.
- [9] Randhir Kumar, Prabhat Kumar, Rakesh Tripathi, Govind P. Gupta, Sahil Garg, Mohammad Mehedi Hassan, A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network, *Journal of Parallel and Distributed Computing*, Volume 164, 2022, Pages 55-68, ISSN 0743-7315.
- [10] Md. Alamgir Hossain, Md. Saiful Islam, Ensuring network security with a robust intrusion detection system using ensemble-based machine learning, *Array*, Volume 19, 2023, 100306, ISSN 2590-0056.
- [11] Shaashwat Agrawal, Sagnik Sarkar, Ons Aouedi, Gokul Yenduri, Kandaraj Piamrat, Mamoun Alazab, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, Thippa Reddy Gadekallu, Federated Learning for intrusion detection system:

Concepts, challenges and future directions ,Computer Communications, Volume 195,2022,Pages 346-361,ISSN 0140-3664.

[12] Kabilan N, Vinayakumar Ravi, V Sowmya, Unsupervised intrusion detection system for in-vehicle communication networks, Journal of Safety Science and Resilience, Volume 5, Issue 2024,Pages 119-129,ISSN 2666-4496.

[13] Pritom Biswas Udas , Md. Ebtidaul Karim, Kowshik Sankar Roy, SPIDER: A shallow PCA based network intrusion detection system with enhanced recurrent neural networks, Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 10, Part B,2022,Pages 10246-10272,ISSN 1319-1578.

[14] E Balamurugan, Abolfazl Mehbodniya, Elham Kariri, Kusum Yadav, Anil Kumar, Mohd Anul Haq, Network optimization using defender system in cloud computing security based intrusion detection system with game theory deep neural network (IDSGT-DNN),Pattern Recognition Letters, Volume 156,2022,Pages 142-151,ISSN 0167-8655.

[15] Edosa Osa, Patience E. Orukpe , Usiholo Iruansi, Design and implementation of a deep neural network approach for intrusion detection systems, e-Prime - Advances in Electrical Engineering, Electronics and Energy, Volume 7,2024,100434,ISSN 27726711.

B.Source Code

```
import subprocess
import winreg
import re
import codecs

print("#####")
print("1) Make sure you run this script with administrator privileges")
print("2) Make sure that the WiFi adapter is connected to a network")
print("#####\n")

# MAC Addresses to attempt using. You will select one when the script is used.
# You can change the names in this list or add names to this list.
# Make sure you use 12 valid hexadecimal values.
# If the MAC address change fails try setting the second character to 2 or 6 or A or E,
# for example: 0A1122334455 or 0A5544332211
# If unsure, leave the MAC addresses listed here as is.
mac_to_change_to = ["0A1122334455", "0E1122334455", "021122334455", "061122334455"]

# We create an empty list where we'll store all the MAC addresses.
mac_addresses = list()

# We start off by creating a regular expression (regex) for MAC addresses.
macAddRegex = re.compile(r"([A-Za-z0-9]{2}[:-]){5}([A-Za-z0-9]{2})")

# We create a regex for the transport names. It will work in this case.
# But when you use the .+ or .*, you should consider making it not as greedy.
transportName = re.compile("{.+}")
```

```

# We create regex to pick out the adapter index
adapterIndex = re.compile("[0-9]+")

# Python allows us to run system commands by using a function provided by the subprocess module:
# (subprocess.run(<list of command line arguments goes here>,
# <specify the second argument if you want to capture the output>))
# The script is a parent process and creates a child process which runs the system command,
# and will only continue once the child process has completed.
# To save the content that gets sent to the standard output stream (the terminal),
# we have to specify that we want to capture the output, so we specify the second
# argument as capture_output = True. This information gets stored in the stdout attribute.
# The information is stored in bytes and we need to decode it to Unicode before we use it
# as a String in Python.
# We use Python to run the getmac command, and then capture the output.
# We split the output at the newline so that we can work with the individual lines
# (which will contain the Mac and transport name).
getmac_output = subprocess.run("getmac", capture_output=True).stdout.decode().split('\n')

# We loop through the output
for macAdd in getmac_output:
    # We use the regex to find the Mac Addresses.
    macFind = macAddRegex.search(macAdd)
    # We use the regex to find the transport name.
    transportFind = transportName.search(macAdd)
    # If you don't find a Mac Address or Transport name the option won't be listed.
    if macFind == None or transportFind == None:
        continue
    # We append a tuple with the Mac Address and the Transport name to a list.
    mac_addresses.append((macFind.group(0),transportFind.group(0)))

# Create a simple menu to select which Mac Address the user want to update.

```

```

print("Which MAC Address do you want to update?")
for index, item in enumerate(mac_addresses):
    print(f"{index} - Mac Address: {item[0]} - Transport Name: {item[1]}")

# Prompt the user to select Mac Address they want to update.
option = input("Select the menu item number corresponding to the MAC that you want to change:")

# Create a simple menu so the user can pick a MAC address to use
while True:
    print("Which MAC address do you want to use? This will change the Network Card's MAC address.")
    for index, item in enumerate(mac_to_change_to):
        print(f"{index} - Mac Address: {item}")

    # Prompt the user to select the MAC address they want to change to.
    update_option = input("Select the menu item number corresponding to the new MAC address that you want to use:")

    # Check to see if the option the user picked is a valid option.
    if int(update_option) >= 0 and int(update_option) < len(mac_to_change_to):
        print(f"Your Mac Address will be changed to: {mac_to_change_to[int(update_option)]}")
        break
    else:
        print("You didn't select a valid option. Please try again!")

# We know the first part of the key, we'll append the folders where we'll search the values
controller_key_part = r"SYSTEM\ControlSet001\Control\Class\{4d36e972-e325-11ce-bfc1-08002be10318}"

# We connect to the HKEY_LOCAL_MACHINE registry. If we specify None,
# it means we connect to local machine's registry.
with winreg.ConnectRegistry(None, winreg.HKEY_LOCAL_MACHINE) as hkey:

```

```

    # Create a list for the 21 folders. I used a list comprehension. The expression part of the list
    comprehension

    # makes use of a ternary operator. The transport value for you Mac Address should fall within this
    range.

    # You could write multiple lines.

    controller_key_folders = [("\000" + str(item) if item < 10 else "\00" + str(item)) for item in
    range(0, 21)]

    # We now iterate through the list of folders we created.

    for key_folder in controller_key_folders:

        # We try to open the key. If we can't we just except and pass. But it shouldn't be a problem.
        try:

            # We have to specify the registry we connected to, the controller key

            # (This is made up of the controller_key_part we know and the folder(key) name we created
            # with the list comprehension).

            with winreg.OpenKey(hkey, controller_key_part + key_folder, 0, winreg.KEY_ALL_ACCESS) as
            regkey:

                # We will now look at the Values under each key and see if we can find the
                "NetCfgInstanceId"

                # with the same Transport Id as the one we selected.

                try:

                    # Values start at 0 in the registry and we have to count through them.

                    # This will continue until we get a WindowsError (Where we will then just pass)

                    # then we'll start with the next folder until we find the correct key which contains
                    # the value we're looking for.

                    count = 0

                    while True:

                        # We unpack each individual winreg value into name, value and type.

                        name, value, type = winreg.EnumValue(regkey, count)

                        # To go to the next value if we didn't find what we're looking for we increment count.

                        count = count + 1

                        # We check to see if our "NetCfgInstanceId" is equal to our Transport number for our
                        # selected Mac Address.

                        if name == "NetCfgInstanceId" and value == mac_addresses[int(option)][1]:

```

```

        new_mac_address = mac_to_change_to[int(update_option)]
        winreg.SetValueEx(regkey, "NetworkAddress", 0, winreg.REG_SZ, new_mac_address)
        print("Successfully matched Transport Number")
        # get list of adapters and find index of adapter you want to disable.
        break
    except WindowsError:
        pass
except:
    pass

# Code to disable and enable Wireless devices
run_disable_enable = input("Do you want to disable and reenabte your wireless device(s). Press Y or y to continue:")

# Changes the input to lowercase and compares to y. If not y the while function which contains the last part will never run.
if run_disable_enable.lower() == 'y':
    run_last_part = True
else:
    run_last_part = False

# run_last_part will be set to True or False based on above code.
while run_last_part:

    # Code to disable and enable the network adapters

    # We get a list of all network adapters. You have to ignore errors, as it doesn't like the format the command returns the data in.
    network_adapters = subprocess.run(["wmic", "nic", "get", "name,index"],
        capture_output=True).stdout.decode('utf-8', errors="ignore").split('\r\n')

    for adapter in network_adapters:

        # We get the index for each adapter
        adapter_index_find = adapterIndex.search(adapter.lstrip())

        # If there is an index and the adapter has wireless in description we are going to disable and enable the adapter

```

```

if adapter_index_find and "Wireless" in adapter:

    disable = subprocess.run(["wmic", "path", "win32_networkadapter", "where",
f"index={adapter_index_find.group(0)}", "call", "disable"],capture_output=True)

    # If the return code is 0, it means that we successfully disabled the adapter

    if(disable.returncode == 0):

        print(f"Disabled {adapter.lstrip()}")

    # We now enable the network adapter again.

    enable = subprocess.run(["wmic", "path", f"win32_networkadapter", "where",
f"index={adapter_index_find.group(0)}", "call", "enable"],capture_output=True)

    # If the return code is 0, it means that we successfully enabled the adapter

    if (enable.returncode == 0):

        print(f"Enabled {adapter.lstrip()}")


# We run the getmac command again

getmac_output = subprocess.run("getmac", capture_output=True).stdout.decode()

# We recreate the Mac Address as it shows up in getmac XX-XX-XX-XX-XX-XX format from the 12
character string we have. We split the string into strings of length 2 using list comprehensions and
then. We use "-".join(list) to recreate the address

mac_add = "-".join([(mac_to_change_to[int(update_option)][i:i+2]) for i in range(0,
len(mac_to_change_to[int(update_option)]), 2)])

# We want to check if Mac Address we changed to is in getmac output, if so we have been
successful.

if mac_add in getmac_output:

    print("Mac Address Success")

# Break out of the While loop. Could also change run_last_part to False.

break

```