NETWORK VIRGILANCE: “CREATING AN EFFECTIVE NIDS”

A PROJECT REPORT

Submitted by

UJJWAL JAIN	(22BCY10016)
SAJAL SETH	(22BCY10013)
SHUBHAM AGARWAL	(22BCY10025)
DEO SAGAR KUMAR	(22BCY10027)
ANKIT KUMAR	(22BCY10224)

in partial fulfilment for the award of degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Specialisation in Cybersecurity and Digital Forensics)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRI KALAN, SEHORE

MADHYA PRADESH - 466114.

OCTOBER 2023

VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **NETWORK VIRGILANCE: "CREATING AN EFFECTIVE NIDS"** is the Bonafide work of **"UJJWAL JAIN (22BCY10016), SAJAL SETH (22BCY10013), SHUBHAM AGARWAL (22BCY10025), DEO SAGAR KUMAR (22BCY10027), ANKIT KUMAR (22BCY10224*)*"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion to this or any other candidate.

PROGRAM CHAIR
Dr. D.Saravanan , Program Chair
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Dr.Soma Saha, Supervisor
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition I Examination is held on 26.10.2023

ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to **Dr. Pushpinder Singh Patheja**, Division Head of the Department, Division of Cyber Security and Digital Forensics for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide **Dr. Soma saha**, for continually guiding and actively participating in my project, and giving valuable suggestions to complete the project work.

Last, but not least, I am deeply indebted to my parents who have been the greatest support.

LIST OF ABBREVIATIONS

1. TCP: Transmission Control Protocol
2. HTTP: Hypertext Transfer Protocol
3. UDP: User Datagram Protocol
4. SSL: Secure Sockets Layer
5. IP: Internet Protocol
6. PCAP: Packet Capture
7. TLS: Transport Layer Security
8. HTTP2: Hypertext Transfer Protocol version 2
9. HTML: Hypertext Markup Language
10. UTF8: Unicode Transformation Format 8
11. URL: Uniform Resource Locator
12. API: Application Programming Interface
13. VM: Virtual Machine
14. JSON: JavaScript Object Notation
15. GUI: Graphical User Interface
16. ASCII: American Standard Code for Information Interchange
17. HTTP: HyperText Transfer Protocol
18. CLI : Command Line Interface

LIST OF FIGURES AND GRAPHS

ABSTRACT

In recent years, network security has faced rapidly evolving threats, making security mechanisms like firewalls, antivirus software, and Intrusion Detection Systems (IDS) crucial for network defense. IDS, in particular, plays a vital role by identifying potentially malicious activities in a network. Ongoing research in Network-Based Intrusion Detection Systems (NIDS) focuses on innovative approaches and techniques to improve their effectiveness, often using benchmark datasets for assessment.

This analysis reviews research trends in NIDS by considering factors like citation counts and annual article counts, shedding light on the impact of research contributions. It also explores the latest NIDS technologies, benefiting both newcomers and experienced researchers. Furthermore, the study highlights commonly used datasets, essential for testing intrusion detection techniques.

Through a comparative analysis of citations and publications, the study quantitatively measures the popularity of various intrusion detection approaches, helping identify favored research directions. Overall, this comprehensive overview simplifies understanding the current state of NIDS research, making it a valuable resource for cybersecurity and network security professionals

TABLE OF CONTENTS

CHAPTER -1
PROJECT DESCRIPTION AND OUTLINE

1.1 INTRODUCTION

Today’s era is of information and communication, and the numbers of host/terminal are continuously increasing in the scenario of computer networking. Vulnerabilities in security systems and unauthorized access to information systems are also growing tremendously. Many techniques, namely firewalls, access control, anti-virus, anti-malware software, application security, behavioural analytic, data loss prevention, distributed denial of service (DDoS) prevention, and network segmentation are commonly used in the computer world to promote internet security mechanisms due to their capabilities of content filtering, blocking data outflow, and alerting and preventing malicious activities. Firewalls and spam filters are generally used with simple rules-based algorithms to allow and denial of the protocols, port, or IP addresses. But the drawback of these firewalls and filters is that sometimes they are unable to control complex attacks of DoS (denial of service) types, and they are also not capable of making the differences between ‘good traffic’ and ‘bad traffic’. An intrusion detection system (IDS) with anti-virus has a significant impact on computer network security mechanisms that provides a more prominent scenario for protecting a computer network from the unauthenticated access. In the perspective of information systems, intrusion refers to any attempt that compromises the integrity, availability, confidentiality, or bypasses the security mechanism in a computer or a network.

1.2 MOTIVATION FOR THE WORK

In today's era of rapid information and communication expansion, the proliferation of hosts and terminals in computer networks has brought an alarming surge in security vulnerabilities and unauthorized access. Traditional security mechanisms, such as firewalls and spam filters, though effective, struggle to combat complex threats and differentiate between 'good' and 'bad' network traffic. This underscores the vital role of Intrusion Detection Systems (IDS) when integrated with anti-virus solutions. IDS, categorized into Host-based IDS (HIDS), Network-based IDS (NIDS), and Hybrid IDS, promptly detect malicious activities and unauthorized access, serving as a critical early warning system. This report analyzes research trends in IDS, dataset popularity, and intrusion detection approaches, using quantitative measures like citations to provide valuable insights for cybersecurity stakeholders..

1. Combat Escalating Cyber Threats: As cyber threats become more frequent and sophisticated, there's a critical need for effective intrusion detection.
2. Protect Sensitive Data: Safeguarding sensitive information in an age of digital communication is paramount.
3. Meet Regulatory Requirements: Compliance with data protection regulations is not optional and requires robust security measures.
4. Ensure Real-time Threat Detection: Timely threat detection minimizes damage from cyberattacks.
5. Reduce Downtime and Financial Loss: NIDS helps minimize operational disruptions and financial losses due to breaches.
6. Embrace Proactive Security: NIDS promotes a proactive approach to security, preventing vulnerabilities from escalating.

7. Address Network Complexity: Modern networks are intricate, demanding advanced NIDS to secure them effectively.
8. Leverage Technological Advancements: Advancements in intrusion detection tech, including machine learning, offer more accurate threat detection.

NIDS'S Past:

The history of Network Intrusion Detection Systems (NIDS) can be traced back to the evolution of computer networks and the increasing need for security in the digital age. Here's a brief history of NIDS:

1. **Early Networks and Security Concerns (1970s-1980s):** The concept of computer networks started in the late 1960s and 1970s with the development of ARPANET, which later became the foundation of the internet. As networks grew, so did concerns about security. Early security measures were primarily focused on user access control and basic encryption.
2. **First Intrusion Detection Systems (IDS):** In the 1980s, the first Intrusion Detection Systems (IDS) emerged. These were primarily host-based systems that monitored the activities on individual computers. The focus was on identifying suspicious activities, but they lacked the ability to monitor network traffic.
3. **Transition to NIDS (1990s):** The term "Network Intrusion Detection System" (NIDS) came into use in the 1990s. The need to monitor and secure networks, rather than just individual hosts, became increasingly evident. NIDS were designed to passively analyze network traffic and detect anomalies and known attack patterns.
4. **Emergence of Commercial NIDS (Late 1990s):** The late 1990s saw the emergence of commercial NIDS products, such as Snort and Cisco's NetRanger. These systems offered a range of features for network monitoring and security.
5. **Signature-based and Anomaly-based Detection (Late 1990s-2000s):** NIDS technology evolved with the introduction of signature-based detection (matching known attack patterns) and anomaly-based detection (identifying deviations from normal network behavior).
6. **Intrusion Prevention Systems (IPS):** The late 1990s and 2000s also saw the development of Intrusion Prevention Systems (IPS), which not only detected intrusions but could actively block or prevent them. IPS technology often merged with NIDS capabilities.
7. **Integration of Machine Learning and AI (2010s):** In the 2010s, NIDS systems increasingly incorporated machine learning and artificial intelligence for more advanced threat detection and to adapt to evolving attack techniques.
8. **Cloud and Virtual Environments (2010s-Present):** With the rise of cloud computing and virtualization, NIDS evolved to secure these environments. Cloud-based NIDS and virtual appliances became prominent.
9. **Threat Intelligence Integration (Present):** Modern NIDS often integrate threat intelligence feeds and collaborate with other security tools, enhancing their capacity to identify and respond to complex threats.
10. **Continued Evolution (Present-Future):** NIDS continue to evolve to counter ever-more sophisticated threats, with a focus on automation, behavior analysis, and the integration of security information and event management (SIEM) systems.

1.3 ABOUT INTRODUCTION TO THE PROJECT

Our project's aim is to perform network analysis on the server, which is packet capturing based analysis. In short, our main goal is to study network attacks. We aim to:

1. **Understand How It Works:**
   Learn how the NIDS monitors and detects network threats.
2. **Identify Threat Level:**
   Assess the severity of potential risks and its ability to detect threats.
3. **Collect Information:**
   Document NIDS configuration, track alerts, and maintain baseline data.
4. **Share Knowledge:**
   Share insights and collaborate to enhance overall digital security.

1.4 PROBLEM STATEMENT

In recent years, Web Server Network Intrusion Detection Systems (NIDS) have become vital components of network security infrastructure, serving as the first line of defense against a diverse range of cyber threats. However, the efficacy of these NIDS solutions is increasingly compromised due to their vulnerability to an ever-evolving spectrum of cyber threats, which include sophisticated attack techniques such as SQL injection and zero-day vulnerabilities.
The fundamental problem at hand is twofold:

1. **Vulnerability to Emerging Threats:** Web Server NIDS, designed to safeguard the integrity of web-based applications and data, are facing a growing challenge posed by emerging attack vectors. As cybercriminals continually advance their tactics, NIDS technology must keep pace. Unfortunately, the ability of many existing NIDS to adapt and recognize evolving threats, such as zero-day vulnerabilities or novel attack methodologies, is often limited.
2. **Ineffectiveness Against Specific Threats:** Of particular concern is the susceptibility of Web Server NIDS to certain high-impact threats, notably SQL injection attacks. These attacks manipulate web application inputs to exploit vulnerabilities in database queries, often with disastrous consequences. Existing NIDS solutions may struggle to accurately identify and mitigate such threats, leaving web servers exposed to data breaches, integrity compromise, and potential financial and reputational damage.

The overarching problem, therefore, is the need to enhance the resilience of Web Server NIDS in the face of dynamic and sophisticated cyber threats. This entails improving the systems' capabilities to proactively detect and respond to emerging threats, as well as their effectiveness in countering known but high-impact attack vectors, such as SQL injection.
Addressing this challenge is essential to maintain the integrity and availability of web-based services and data, protect sensitive information, and ensure the continued trust of users and stakeholders in the digital age. This report seeks to explore potential solutions and strategies to bolster the security posture of Web Server NIDS and mitigate the vulnerabilities that expose them to evolving cyber threats

1.5 OBJECTIVE OF THE WORK

Our project objective is to to detect and report malicious or potentially harmful network activities. NIDS continuously monitors network traffic, identifies unauthorized or suspicious behavior, and promptly reports these findings to the network manager. This reporting function serves as an early warning system, enabling swift responses to security threats. By providing alerts and detailed information about the nature and source of threats, NIDS empowers network managers to take immediate action, fortify network security, and mitigate potential risks. In a rapidly evolving cybersecurity landscape, the NIDS's objective is to ensure proactive threat detection and response for network protection.

1.6 SUMMARY

In the contemporary era dominated by information and communication, computer networks are experiencing an exponential increase in the number of hosts and terminals. Unfortunately, this surge in network participation also brings forth a proliferation of vulnerabilities in security systems, leading to unauthorized access to information systems. To combat these growing threats, various security mechanisms such as firewalls, access control, anti-virus and anti-malware software, application security, behavioral analytics, data loss prevention, and distributed denial of service (DDoS) prevention are widely employed. These mechanisms are known for their abilities in content filtering, blocking data outflows, and alerting against and preventing malicious activities.

Firewalls and spam filters, while effective, often employ simple rules-based algorithms, which render them less capable of countering complex Denial of Service (DoS) attacks. Additionally, they struggle to differentiate between 'good' and 'bad' network traffic. To address these limitations, Intrusion Detection Systems (IDS) integrated with anti-virus solutions emerge as influential components in the realm of network security. These systems play a pivotal role in safeguarding computer networks against unauthorized access and security breaches.

In the context of information systems, an intrusion pertains to any attempt to compromise the integrity, availability, confidentiality, or bypass the security mechanisms of a computer or network. The National Institute of Standards and Technology (NIST) defines intrusion detection as the process of monitoring events within a computer system or network, analyzing these events for signs of intrusions, and taking measures to secure the system from malicious activities. IDS systems play a central role in this process.

There are three primary categories of IDS installation: Host-based IDS (HIDS), Network-based IDS (NIDS), and Hybrid IDS. HIDS are deployed on individual hosts, monitoring attacks at the single computer system level, and analyzing essential operating system files. NIDS, on the other hand, detect malicious activities across interconnected computers, usually deployed on routers or switches. Hybrid IDS systems can be installed on both hosts and the network.

The core objective of NIDS is to identify malicious or suspicious logging information and promptly report such findings to the network manager, thereby offering an early warning system against potential security threats. These systems, although they do not prevent intrusions, significantly contribute to the protection of network resources by detecting and reporting unauthorized access and potentially harmful activities in real-time or before they materialize.

Understanding the research trends in intrusion detection is paramount in enhancing network security. The article adopts citation analysis to assess the popularity and research trends in the field, enabling the identification of the most commonly employed intrusion detection approaches and methodologies. Moreover, it explores the significance of various benchmark datasets for evaluating NIDS models and delves into the performance metrics used in assessing these systems.

In conclusion, the contemporary cybersecurity landscape demands robust measures to counter evolving threats. IDS, integrated with anti-virus solutions, plays a vital role in bolstering network security. Understanding research trends and leveraging advanced techniques and

datasets are essential for staying at the forefront of intrusion detection and fortifying network defenses.
.

CHAPTER-2:
RELATED WORK INVESTIGATION

2.1 INTRODUCTION

In this section, relevant work in the field of malware detection is discussed, focusing on Static and Dynamic analysis techniques obtained through signature-based approach. Purpose behind this chapter is to provide context for our project by reviewing existing methods and solutions.

2.2 EXISTING WORK

Network Intrusion Detection Systems (NIDS) are a crucial component of modern network security. Their operation revolves around the continuous monitoring of network traffic to identify unauthorized, suspicious, or potentially malicious activities. This process begins with the collection and analysis of data packets traveling across the network. NIDS examine a range of packet attributes, including source and destination IP addresses, port numbers, and the content of data packets. This deep packet inspection allows NIDS to scrutinize both the header and payload of network packets.

NIDS employ two primary detection techniques: signature-based and anomaly-based detection. Signature-based detection involves comparing the network traffic against a database of known attack patterns, or "signatures." When a match is found between the observed traffic and a known signature, the NIDS generates an alert. This method is effective for identifying known threats and attacks, such as viruses, malware, and well-documented attack techniques.

In contrast, anomaly-based detection focuses on establishing a baseline of "normal" network behavior. By monitoring network traffic over time, NIDS develop a profile of expected network activity. Deviations from this established baseline are considered potential indicators of suspicious or malicious activity. Anomaly-based NIDS can detect previously unknown or "zero-day" threats, making them valuable for identifying emerging attack techniques.

The alerts generated by NIDS are crucial for incident response. When suspicious activity is detected, NIDS send alerts to a centralized management system or network administrators. These alerts provide essential information about the nature of the potential threat, its source, and the affected network segment, enabling administrators to investigate and respond promptly.

NIDS are designed to operate continuously, 24/7, offering real-time or near real-time monitoring of network traffic. Their ability to detect and respond to threats in real-time is vital for preventing security breaches and minimizing the potential damage from cyberattacks. Moreover, NIDS maintain logs of all network activity, including alerts and identified threats, which are invaluable for forensic analysis, compliance reporting, and security auditing.

In conclusion, Network Intrusion Detection Systems play a crucial role in safeguarding networked systems from a wide range of cyber threats. Their combination of signature-based, anomaly-based, and deep packet inspection techniques provides a multi-faceted approach to identifying and mitigating network intrusions. This capability is essential in the ever-evolving landscape of network security, where the threat landscape continues to advance in complexity and sophistication.

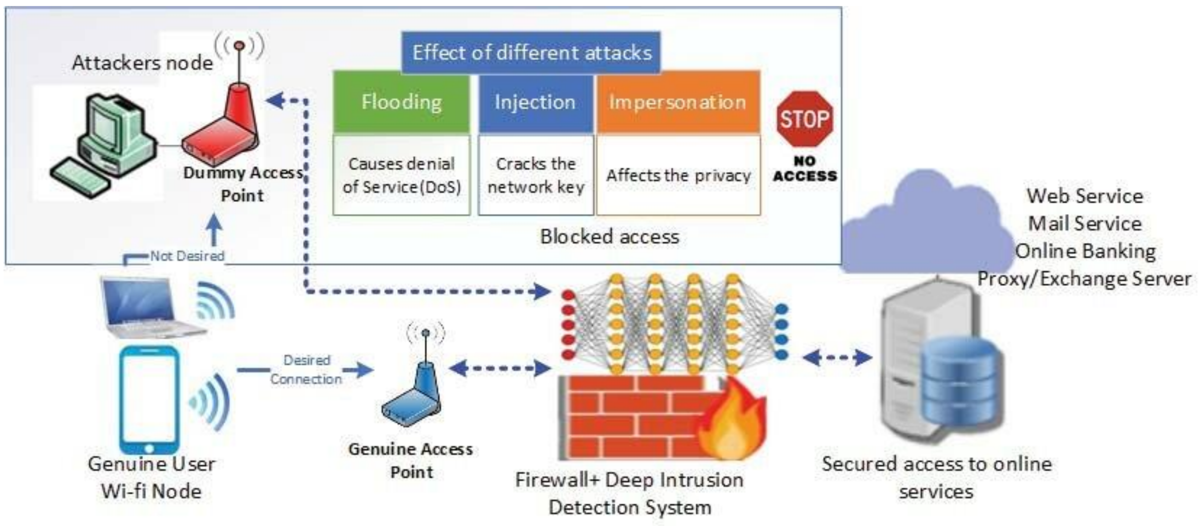


FIGURE 2.1 NIDS MANAGEMENT SYSTEM

2.3 PROPOSED WORK

Our proposed work involves in-depth analysis of networks connected to servers. We will use a combination of packet capturing and logging sessions to analyse the connections along with the generation of report for IPS with Firewall to eradicate unauthorized user or hacker . This analysis will help us develop effective strategies for protection against this and similar network attacks in
future. Furthermore, we will be sharing our findings and insights with the community to contribute to the collective knowledge in the field and help others stay safe online.

CONTRIBUTION OF THE WORK

- **Introduction:**The development and deployment of Network Intrusion Detection Systems (NIDS) are vital in today's cybersecurity landscape to safeguard sensitive data and systems from malicious threats. However, the effectiveness of NIDS tools can be significantly enhanced with custom monitoring and analysis capabilities. In this contribution, we present a Python-based tool that complements standard NIDS by providing advanced features for real-time packet analysis, alerting, and detailed inspection of network traffic.
- **Description of the Tool:** Our custom tool is designed to work in conjunction with existing NIDS systems. It uses various Python libraries, including Scapy, PyShark, and PySimpleGUI, to capture network traffic, process packets, and enable detailed analysis of detected anomalies.

Our custom Python tool serves as a valuable addition to traditional NIDS, offering enhanced capabilities for real-time packet analysis, alerting, and detailed traffic inspection. This tool can significantly improve network security, as it provides users with a deeper understanding of network activity and enables timely response to potential threats.

CHAPTER-3:
REQUIREMENT ARTIFACTS

3.1 INTRODUCTION

This section outlines the essentials for successful intrusion detection, emphasizing the critical components and prerequisites. The effectiveness and efficiency of the evaluation process hinge on the proper availability and installation of hardware, software, and any specialized requirements. This section offers recommendations to guarantee the presence of all essential elements before commencing the analysis procedure.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

3.2.1 Hardware Requirements

The following hardware components are essential for the NIDS system:

- **Host System**:
    - A high-performance computer with adequate CPU, RAM, and storage capacity to host virtual machines.
    - Sufficient disk space for storing VM images, snapshots, and analysis artifacts.
- **Virtualization Platform**:
    - Installation of a compatible virtualization platform such as VMware Workstation, VirtualBox, etc.

3.2.2 Software Requirements

The software requirements include the necessary operating systems, virtual machine images, and analysis tools:

1. Operating System: A suitable operating Windows 10 and Linux
2. Virtualization Software: VM Virtual Box
3. Network Capture Tools: NMAP, PYSHARK, TCAP,
4. Required Python Packages: PySimpleGUI, scapy,
5. IDE (Integrated Development Environment): Visual Studio Code, Command propmt

3.3 Specific Project requirements

We have used various tools:

- PySimple GUI
- SOCKET
- NMAP
- PYSHARK
- PCAP

CHAPTER - 4
DESIGN METHODOLOGY AND ITS NOVELTY

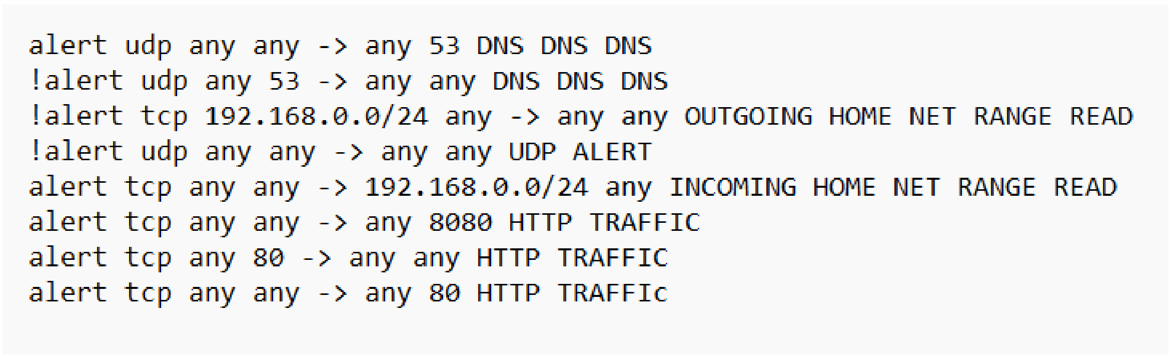
4.1 DESIGN METHODOLOGY

The design method used for this analysis includes methods and generalizations that include Packet Capturing-based analysis methods. It starts with an initial Packet capturing analysis to quickly identify Network Intrusion, followed by TCP AND UDP PACKET analysis to dissect the binary and understand its structure, behaviour, and usability. This is based on SETRULES in a controlled environment to analyze the actual behaviour and interactions of the network intrusion.

1. **Packet capturing analysis**: Packet capturing analysis, often referred to as network packet analysis or packet sniffing, is the process of intercepting and examining data packets that are transmitted over a computer network
2. **TCP Analysis**: TCP analysis often starts with packet capturing, which involves capturing and inspecting TCP packets as they traverse a network.
3. **UDP Packet Analysis:** UDP packets is important for various purposes, including troubleshooting, network monitoring, and security assessment

STEPS:

1) Setting up the system which includes, downloading the windows 10, Oracle VM in your virtual machine, Kali linux(Simulation)
2) Logging confirmation
   • To check authentication and confirm that log data remains secure and has not been tampered with
3) Defining Predefined rules
   • Set of rules which determine either the access is authorized or unauthorized



```
alert udp any any -> any 53 DNS DNS DNS
!alert udp any 53 -> any any DNS DNS DNS
!alert tcp 192.168.0.0/24 any -> any any OUTGOING HOME NET RANGE READ
!alert udp any any -> any any UDP ALERT
alert tcp any any -> 192.168.0.0/24 any INCOMING HOME NET RANGE READ
alert tcp any any -> any 8080 HTTP TRAFFIC
alert tcp any 80 -> any any HTTP TRAFFIC
alert tcp any any -> any 80 HTTP TRAFFIc
```

Figure 4.1 : Predefined Rules

4) Comparing and updating sessions
   - The packet is being captured will be analyzed using set rules with the help of logging confirmation and same will be updated in logging sessions
5) Taking inputs and providing flags

- After updating the session user will input logging id and password the same will be compared with the password manager and if the password is not write it will give an alert as flag to the system and this will helps the system to protect its data.

NOVELTY

The provided code offers a robust framework for real-time network intrusion detection and packet analysis. Notably, it leverages the Scapy library to capture, inspect, and analyze network packets as they traverse the network. This dynamic, real-time packet analysis is instrumental in identifying potential security threats and abnormalities as they occur.

One key feature of the code is its ability to process intrusion detection rules defined in an external "rules.txt" file. These rules can be customized to match specific conditions, such as source and destination IP addresses, ports, and protocols. As network traffic is continuously monitored, any packet that matches these rules triggers an alert. This approach empowers network administrators to respond promptly to potential threats.

The code also offers a user-friendly interface created using PySimpleGUI, making it accessible to a wider range of users. This interface allows users to initiate and halt packet capture, refresh the detection rules, and review alerts. Real-time decoding and analysis of packet payloads enable a deeper inspection of network communications, aiding in the identification of malicious content or potentially harmful objects.

Furthermore, the ability to analyze TCP and HTTP/2 streams enhances the code's functionality for monitoring network activity. Network administrators can use this feature to gain insights into the content exchanged between network nodes, aiding in both threat detection and network management.

In conclusion, the provided code stands as a powerful tool for network security professionals seeking real-time intrusion detection and detailed packet analysis capabilities. Its dynamic rule processing, streamlined interface, and comprehensive payload inspection make it a valuable asset in safeguarding network infrastructure from potential threats.

4.2 FLOW DIAGRAM

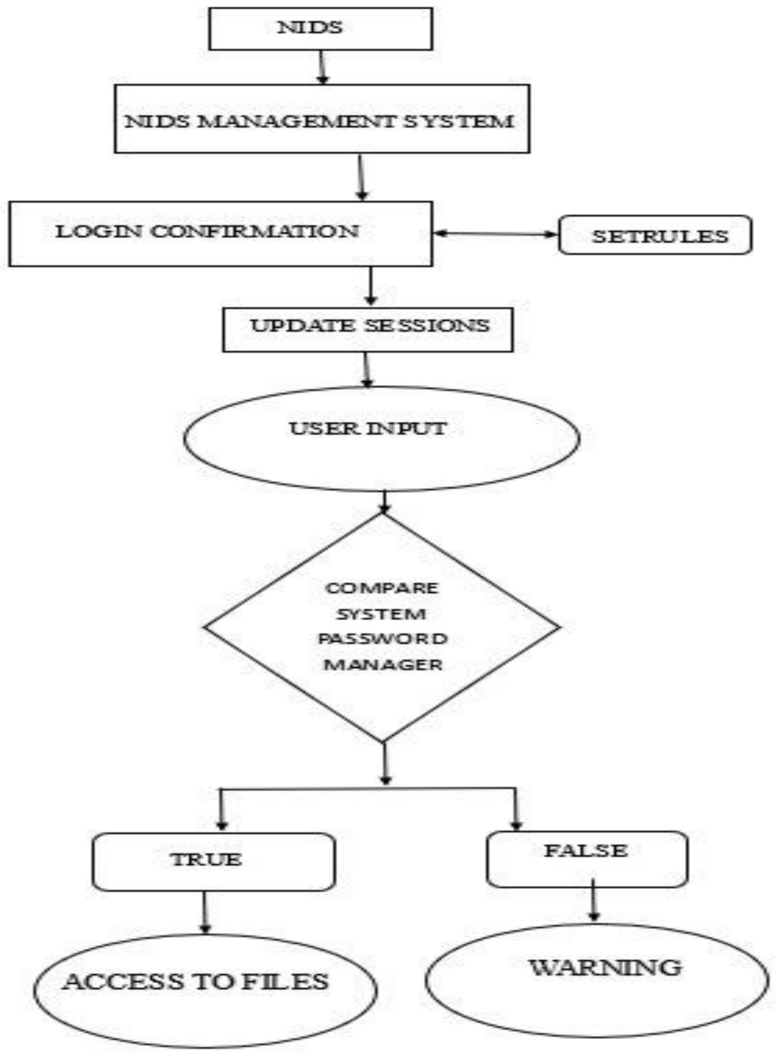


FIGURE 4.2 : Process Flowchart

CHAPTER-5
TECHNICAL IMPLEMENTATION & ANALYSIS

INTRODUCTION

In this technical implementation, we have developed a network traffic analysis tool that allows users to capture and analyze network packets in real-time. The tool offers a user-friendly graphical interface built with PySimpleGUI and utilizes various Python libraries, including PyShark and Scapy, to perform packet capturing and processing.

5.1 WORKING PRINCIPLE

1. It reads network packet capture rules from a file called "rules.txt."
2. It defines alert conditions based on source and destination IP addresses, ports, and protocols, and associates messages with these conditions.
3. It captures network packets from a specified network interface.
4. It processes captured packets and checks if they match any of the predefined alert conditions.
5. If a packet matches an alert condition, it records the packet information and any readable payload.
6. The user interface provides options to start and stop packet capture, display alert packets, and examine details of matched packets

5.2 TOOLS DESCRIPTION

Static analysis

Packet Capture (PCAP)

A Packet Capture (PCAP) is a file format and a set of tools used to capture, store, and analyze network traffic. It's essential for network troubleshooting, security analysis, and network monitoring. PCAP files store captured packets, including content and metadata like source/destination addresses and timestamps. Tools like Wireshark and tcpdump help work with PCAP files. In the code you provided, PCAP is used for capturing and analyzing network packets for various purposes, like intrusion detection. However, remember that PCAP files can contain sensitive information, so handle them securely.

PyShark

PyShark is a Python library that lets you capture, analyze, and work with network packets in real-time or from PCAP files. It provides an easy interface for packet parsing, filtering, and handling, making it a valuable tool for network analysis, monitoring, and troubleshooting. PyShark supports a wide range of network protocols and is useful for various network-related tasks.

Scapy

Scapy is a powerful Python library used for crafting, manipulating, and sending network packets. It allows network engineers, penetration testers, and security researchers to create custom packets, perform network analysis, and automate various network tasks. Scapy is highly versatile and can be used for tasks like network scanning, packet sniffing, crafting exploits, and network protocol development. Its flexibility and extensive protocol support make it a valuable tool for network and security professionals.

PySimpleGUI

PySimpleGUI is a Python library that provides a simple and intuitive way to create graphical user interfaces (GUIs). It allows developers to design and build GUI applications with ease, making it a popular choice for those who want to create desktop applications using Python. PySimpleGUI offers a variety of pre-built elements, such as buttons, text fields, and windows, and it's known for its straightforward and Pythonic syntax, enabling developers to create GUIs quickly without complex code. It's a valuable tool for those who want to develop desktop applications or add a graphical interface to their Python programs efficiently.

Socket

Socket is a fundamental component for networking and communication. It enables programs to establish connections, often identified by IP addresses and port numbers. Socket programming involves creating client-server applications, using different socket types (e.g., TCP and UDP) and libraries (e.g., Python's **socket**). Sockets are vital for various applications, including web servers, email clients, and real-time communication tools.

Different Protocols used-:

UDP

UDP is a simple, connectionless transport layer protocol used in the Internet Protocol Suite. It's fast but lacks built-in mechanisms for ensuring data reliability and order. Applications using UDP must manage these aspects themselves. It's commonly used for real-time applications like VoIP, online gaming, DNS, and more. Datagram Congestion Control Protocol (DCCP) extends UDP, adding congestion control for multimedia applications.

TCP/IP

It is a core communication protocol in the Internet Protocol Suite. It provides reliable, connection-oriented communication, ensuring data integrity, packet order, and flow control. TCP establishes and maintains a connection between sender and receiver. It's used for most internet applications where data reliability is crucial, such as web browsing, email, and file transfers.

5.3 IMPLEMENTATION PROCEDURE

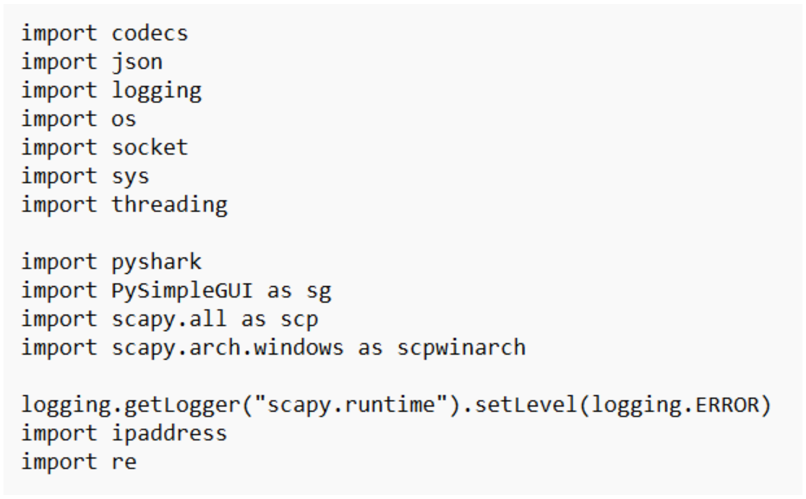
Static analysis

Set Up the Environment:

- Ensure you have the required libraries installed: pyshark, PySimpleGUI, scapy, and ipaddress.

- Import the necessary modules and libraries at the beginning of your script.

  codecs: This module provides functions for working with encodings and Unicode.
  json: Used for working with JSON data.
  logging: Python's built-in logging module for logging messages.
  os: Provides a way of using operating system-dependent functionality.
  socket: Allows access to low-level networking interfaces.
  sys: Provides access to some variables used or maintained by the interpreter and to functions that interact with the interpreter.
  threading: Provides a way to create and manage threads in Python.
  Additional Imports:
  pyshark: A library for packet parsing using the Wireshark library.
  PySimpleGUI: A simple and easy-to-use GUI library for creating desktop applications.
  scapy.all: The Scapy library for packet manipulation.
  scapy.arch.windows: Windows-specific modules for Scapy.
  Logging Configuration:
  The script configures the logging level for the "scapy.runtime" logger to suppress runtime messages. This is done to prevent unnecessary log output.



```
import codecs
import json
import logging
import os
import socket
import sys
import threading

import pyshark
import PySimpleGUI as sg
import scapy.all as scp
import scapy.arch.windows as scpwinarch

logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
import ipaddress
import re
```

Figure 5.1: Imported Modules

Read Intrusion Detection Rules:

- Implement the readrules function to read the rules from the "rules.txt" file.

- Parse the rules and store them in a data structure.

```
alert udp any any -> any 53 DNS DNS DNS
!alert udp any 53 -> any any DNS DNS DNS
!alert tcp 192.168.0.0/24 any -> any any OUTGOING HOME NET RANGE READ
!alert udp any any -> any any UDP ALERT
alert tcp any any -> 192.168.0.0/24 any INCOMING HOME NET RANGE READ
alert tcp any any -> any 8080 HTTP TRAFFIC
alert tcp any 80 -> any any HTTP TRAFFIC
alert tcp any any -> any 80 HTTP TRAFFIc
```

Figure 5.2: Defined Rules.

The provided rules describe network traffic conditions for generating alerts in a security or firewall context. Each rule consists of various components that specify the source and destination of the traffic, as well as the associated message or action. Let's break down the rules in paragraphs:

The first rule focuses on alerting for inbound UDP DNS traffic. It allows any source IP and port to communicate with any destination on port 53. The message "DNS DNS DNS" is associated with this alert, possibly serving as a signature for detecting specific DNS-related patterns.

The second rule is similar to the first but is designed for outbound UDP DNS traffic. It permits any source with port 53 to connect to any destination IP and port. Again, the message "DNS DNS DNS" indicates that the rule is used to capture a particular DNS traffic pattern.

Moving on to the third rule, it targets outgoing UDP traffic from a specific IP range (192.168.0.0/24). The rule permits communication to any destination on any port, and it carries the message "OUTGOING HOME NET RANGE READ," which might signify that it's monitoring outbound traffic from a home network.

The fourth rule is a more generic alert for UDP traffic, not specifying source or destination. It essentially raises an alert for any UDP traffic with the message "UDP ALERT." This rule is broader in scope, capturing any unexpected UDP activity.

The fifth rule is specific to incoming TCP traffic from the home network range (192.168.0.0/24) to any destination. It doesn't specify source port, destination port, or message content but is likely used to monitor incoming traffic from the home network range for security or operational purposes.

The sixth rule narrows down the traffic to alert on outgoing TCP traffic to a specific port, 8080. It allows communication from any source to any destination but triggers the alert with the message "HTTP TRAFFIC." This could be monitoring or alerting on outbound web traffic.

The seventh rule is a specific alert for incoming HTTP traffic, specifically traffic arriving at port 80. It doesn't specify source or destination IP addresses but focuses on the destination port and carries the message "HTTP TRAFFIC."

The eighth rule appears to be a slight variation of the seventh rule, alerting on HTTP traffic arriving at port 80. However, it uses mixed-case text in the message ("HTTP TRAFFIC"), possibly indicating a less strict case-sensitive alert.

Process Rules:

•Implement the process_rules function to parse and organize the rules.

•Extract and store details such as alert protocols, source and destination IPs, source and destination ports, and messages**.**

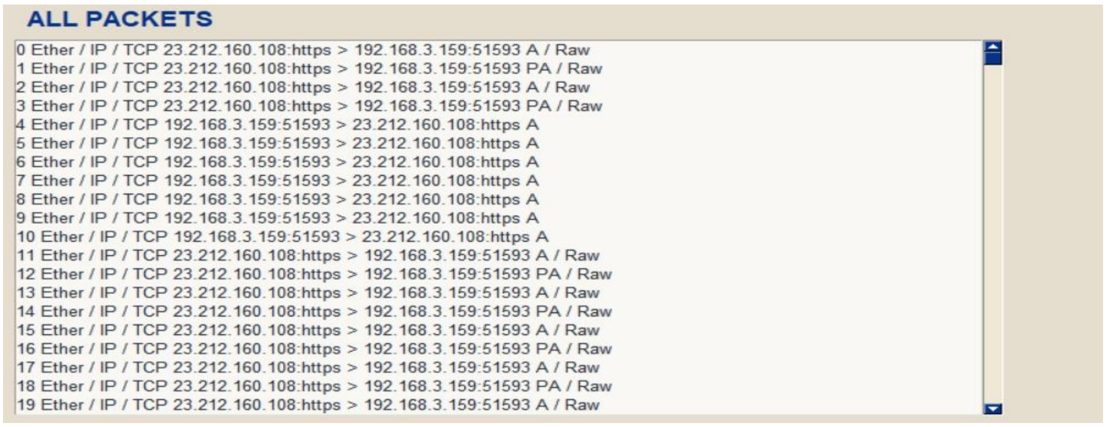
Capture Network Traffic:

Use the scapy library to capture network traffic.

- Set up packet processing logic (pkt_process) to filter and process packets.
  Implement the logic to check if the packet matches any intrusion detection rule (check_rules_warning).

Display Alerts:

- Create a user interface using PySimpleGUI to display alerts and network traffic information.
- Continuously update the UI with incoming packets and detected alerts.
- Allow users to view detailed information about individual packets, TCP streams, and HTTP2 streams.



ALL PACKETS

```
0 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
1 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
2 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
3 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
4 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
5 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
6 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
7 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
8 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
9 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
10 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
11 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
12 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
13 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
14 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
15 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
16 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
17 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
18 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
19 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
```

Figure 5.3: Displaying all packets

Capture and Analyze HTTP2 Streams:

- Implement functions to capture and analyze HTTP2 streams.
- Extract relevant information and headers from the streams.
- Make this information available to users through the UI.

Capture and Analyze TCP Streams:

- Implement functions to capture and analyze TCP streams.
- Extract and display information from TCP streams in the UI.

Save Alerted Packets:

- Allow users to save alerted packets for further analysis.

- Implement the functionality to save packets in a PCAP file.

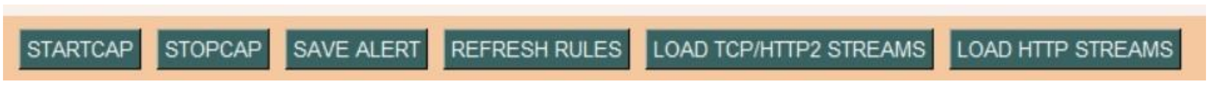


Figure 5.4: All buttons

Manage Rule Updates:

- Implement a mechanism to refresh intrusion detection rules from the "rules.txt" file.

Exit and Cleanup:

- Ensure the application can be terminated gracefully when needed.
- Properly close any open resources, such as PCAP files and network interfaces.

Error Handling:

- Implement error handling to gracefully handle exceptions that may occur during packet processing.

Summmary

1. **Packet Capture and Analysis**: We use Scapy to capture network packets from the specified network interfaces. These packets are then processed for further analysis.

2. **Rule-Based Network Intrusion Detection**: Users can define custom intrusion detection rules in a "rules.txt" file. The rules are parsed to extract criteria such as source IP, destination IP, protocols, and ports. When a packet matches these criteria, it is flagged as suspicious, and a corresponding message is displayed. Users can refresh and modify these rules as needed.

3. **Alerts and Payload Decoding**: The tool displays suspicious packets in real-time, including their packet details and decoded payload. When a packet matches an intrusion detection rule, the alert message is shown alongside the decoded payload, aiding in immediate threat assessment.

4. **TCP and HTTP/2 Stream Analysis**: The tool provides the ability to load and analyze TCP and HTTP/2 streams, making it easier to inspect network communication within specific connections. Users can view the content of these streams, including HTTP headers and payload data.

5. **Saving Captured Traffic**: Users can save captured network traffic as PCAP files for future analysis.

6. **Refresh Rules**: A refresh rules button allows users to update the intrusion detection rules on the fly without restarting the application.

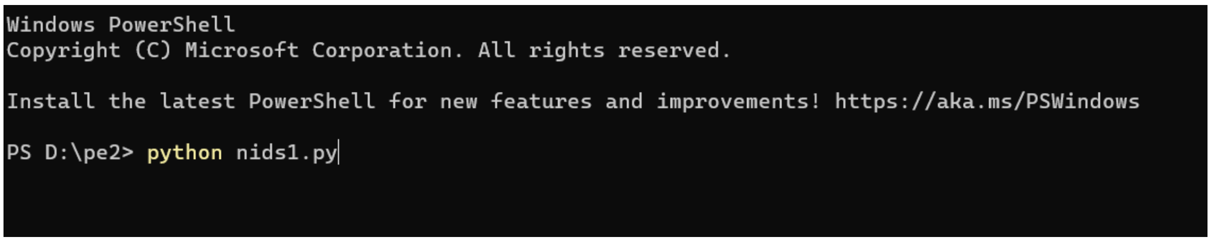
7. **SSL Decryption**: The tool has the capability to decrypt SSL/TLS traffic when provided with the necessary SSL key log file.

8. **HTTP Object Extraction**: The tool can identify and extract HTTP objects (e.g., files, resources) from captured packets, allowing users to review the content of these objects.

This implementation offers a versatile platform for network administrators, security professionals, and researchers to monitor and analyze network traffic, detect suspicious activities, and take appropriate actions. The tool's user-friendly interface and the ability to decode packet payloads enhance the network analysis process, enabling better understanding and faster response to potential threats.

5.3 TESTING

Step1 : NIDS is implemented on windows system.
To start the operation on windows system , we have used python nids1.py



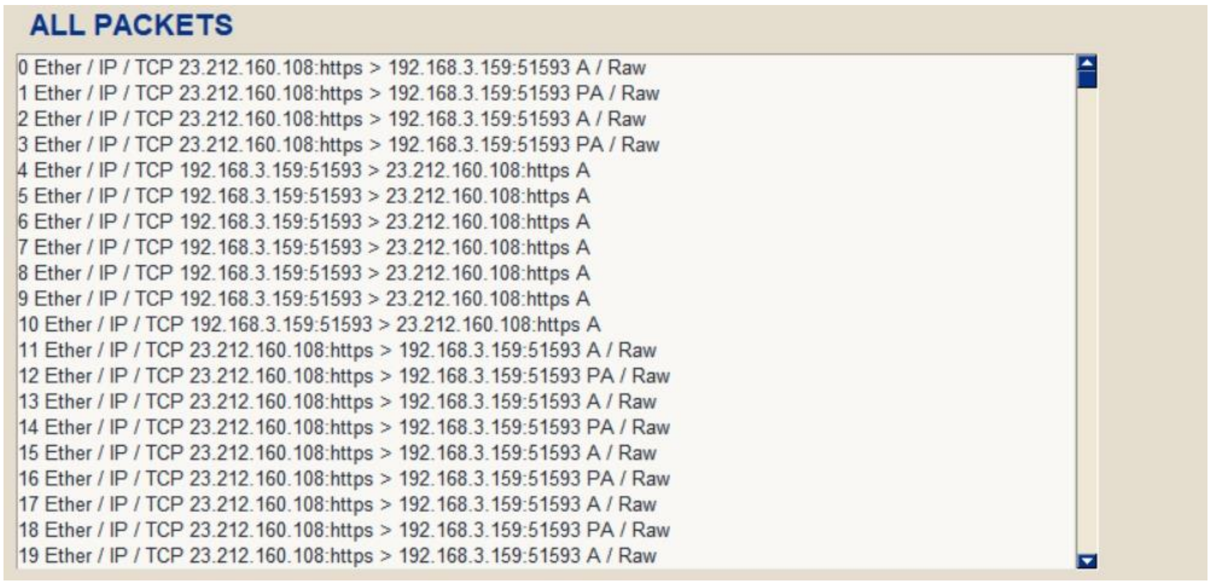
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\pe2> python nids1.py
```

Figure 5.5: Implementation starts

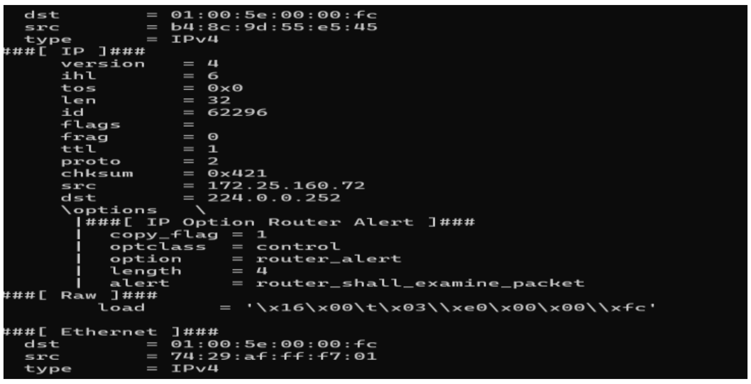
Step2 : Packet capturing using the tools mention above and will get the following results



```
ALL PACKETS
0 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
1 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
2 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
3 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
4 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
5 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
6 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
7 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
8 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
9 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
10 Ether / IP / TCP 192.168.3.159:51593 > 23.212.160.108:https A
11 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
12 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
13 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
14 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
15 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
16 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
17 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
18 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 PA / Raw
19 Ether / IP / TCP 23.212.160.108:https > 192.168.3.159:51593 A / Raw
```

Figure 5.6: Packet capturing and analyzing

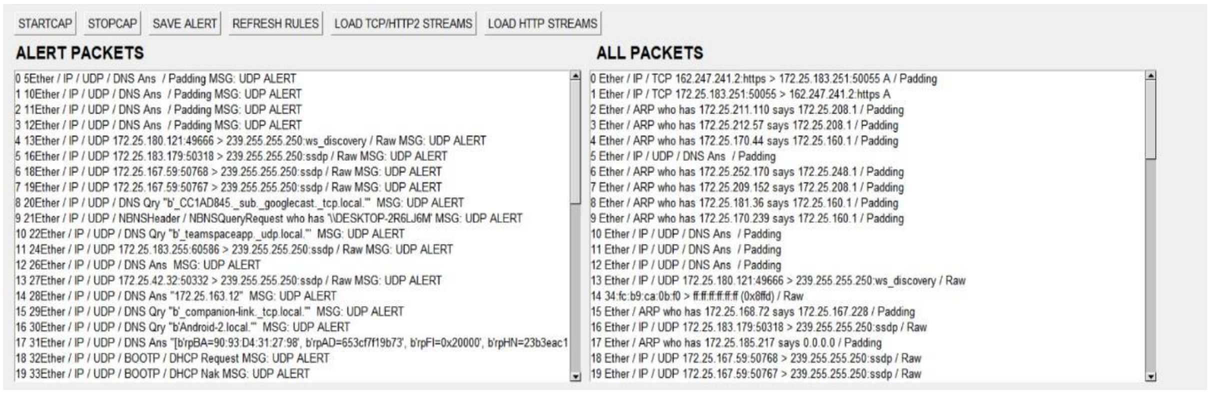
Step3 : After analyzing and capturing the packets we get the desired result in both CLI and GUI



```
  dst        = 01:00:5e:00:00:fc
  src        = b4:8c:9d:55:e5:45
  type       = IPv4
###[ IP ]###
     version   = 4
     ihl       = 6
     tos       = 0x0
     len       = 32
     id        = 62296
     flags     =
     frag      = 0
     ttl       = 1
     proto     = 2
     chksum    = 0x421
     src       = 172.25.160.72
     dst       = 224.0.0.252
     \options   \
      |###[ IP Option Router Alert ]###
      |   copy_flag = 1
      |   optclass  = control
      |   option    = router_alert
      |   length    = 4
      |   alert     = router_shall_examine_packet
###[ Raw ]###
        load      = '\x16\x00\t\x03\\xe0\x00\x00\\xfc'

###[ Ethernet ]###
  dst        = 01:00:5e:00:00:fc
  src        = 74:29:af:ff:f7:01
  type       = IPv4
```

Figure 5.7.1: Result of analyzing(CLI)



ALERT PACKETS

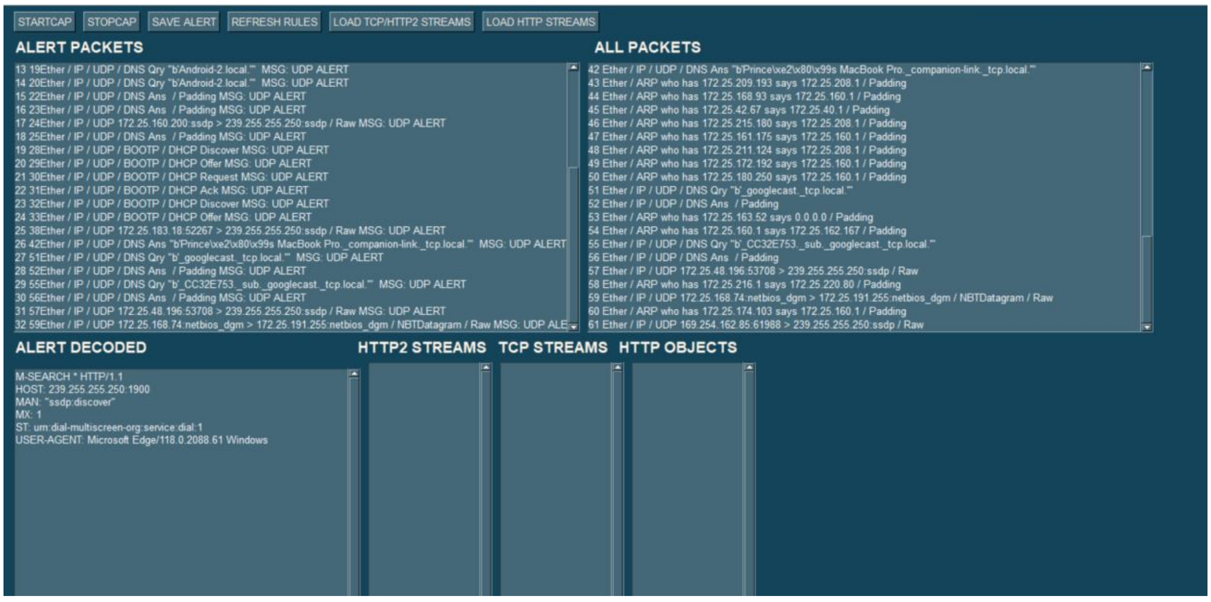
```
0 5Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
1 10Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
2 11Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
3 12Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
4 13Ether / IP / UDP 172.25.180.121:49666 > 239.255.255.250:ws_discovery / Raw MSG: UDP ALERT
5 16Ether / IP / UDP 172.25.183.179:50318 > 239.255.255.250:ssdp / Raw MSG: UDP ALERT
6 18Ether / IP / UDP 172.25.167.59:50768 > 239.255.255.250:ssdp / Raw MSG: UDP ALERT
7 19Ether / IP / UDP 172.25.167.59:50767 > 239.255.255.250:ssdp / Raw MSG: UDP ALERT
8 20Ether / IP / UDP / DNS Qry "b'_CC1AD845._sub._googlecast._tcp.local.'"  MSG: UDP ALERT
9 21Ether / IP / UDP / NBNSHeader / NBNSQueryRequest who has '\\DESKTOP-2R6LJ6M' MSG: UDP ALERT
10 22Ether / IP / UDP / DNS Qry "b'_teamspaceapp._udp.local.'"  MSG: UDP ALERT
11 24Ether / IP / UDP 172.25.183.255:60586 > 239.255.255.250:ssdp / Raw MSG: UDP ALERT
12 26Ether / IP / UDP / DNS Ans  MSG: UDP ALERT
13 27Ether / IP / UDP 172.25.42.32:50332 > 239.255.255.250:ssdp / Raw MSG: UDP ALERT
14 28Ether / IP / UDP / DNS Ans "172.25.163.12"  MSG: UDP ALERT
15 29Ether / IP / UDP / DNS Qry "b'_companion-link._tcp.local.'"  MSG: UDP ALERT
16 30Ether / IP / UDP / DNS Qry "b'Android-2.local.'"  MSG: UDP ALERT
17 31Ether / IP / UDP / DNS Ans "[b'rpBA=90:93:D4:31:27:98', b'rpAD=653cf7f19b73', b'rpFl=0x20000', b'rpHN=23b3eac1
18 32Ether / IP / UDP / BOOTP / DHCP Request MSG: UDP ALERT
19 33Ether / IP / UDP / BOOTP / DHCP Nak MSG: UDP ALERT
```

ALL PACKETS

```
0 Ether / IP / TCP 162.247.241.2:https > 172.25.183.251:50055 A / Padding
1 Ether / IP / TCP 172.25.183.251:50055 > 162.247.241.2:https A
2 Ether / ARP who has 172.25.211.110 says 172.25.208.1 / Padding
3 Ether / ARP who has 172.25.212.57 says 172.25.208.1 / Padding
4 Ether / ARP who has 172.25.170.44 says 172.25.160.1 / Padding
5 Ether / IP / UDP / DNS Ans  / Padding
6 Ether / ARP who has 172.25.252.170 says 172.25.248.1 / Padding
7 Ether / ARP who has 172.25.209.152 says 172.25.208.1 / Padding
8 Ether / ARP who has 172.25.181.36 says 172.25.160.1 / Padding
9 Ether / ARP who has 172.25.170.239 says 172.25.160.1 / Padding
10 Ether / IP / UDP / DNS Ans  / Padding
11 Ether / IP / UDP / DNS Ans  / Padding
12 Ether / IP / UDP / DNS Ans  / Padding
13 Ether / IP / UDP 172.25.180.121:49666 > 239.255.255.250:ws_discovery / Raw
14 34:fc:b9:ca:0b:f0 > ff:ff:ff:ff:ff:ff (0x8ffd) / Raw
15 Ether / ARP who has 172.25.168.72 says 172.25.167.228 / Padding
16 Ether / IP / UDP 172.25.183.179:50318 > 239.255.255.250:ssdp / Raw
17 Ether / ARP who has 172.25.185.217 says 0.0.0.0 / Padding
18 Ether / IP / UDP 172.25.167.59:50768 > 239.255.255.250:ssdp / Raw
19 Ether / IP / UDP 172.25.167.59:50767 > 239.255.255.250:ssdp / Raw
```

Figure 5.7.2: Result of analyzing(GUI)

Step4 : Decoding of alerts and generation of report in GUI



ALERT PACKETS

```
13 19Ether / IP / UDP / DNS Qry "b'Android-2.local.'"  MSG: UDP ALERT
14 20Ether / IP / UDP / DNS Qry "b'Android-2.local.'"  MSG: UDP ALERT
15 22Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
16 23Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
17 24Ether / IP / UDP 172.25.160.200:ssdp > 239.255.255.250:ssdp / Raw MSG: UDP ALERT
18 25Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
19 28Ether / IP / UDP / BOOTP / DHCP Discover MSG: UDP ALERT
20 29Ether / IP / UDP / BOOTP / DHCP Offer MSG: UDP ALERT
21 30Ether / IP / UDP / BOOTP / DHCP Request MSG: UDP ALERT
22 31Ether / IP / UDP / BOOTP / DHCP Ack MSG: UDP ALERT
23 32Ether / IP / UDP / BOOTP / DHCP Discover MSG: UDP ALERT
24 33Ether / IP / UDP / BOOTP / DHCP Offer MSG: UDP ALERT
25 38Ether / IP / UDP 172.25.183.18:52267 > 239.255.255.250:ssdp / Raw MSG: UDP ALERT
26 42Ether / IP / UDP / DNS Ans "b'Prince\xe2\x80\x99s MacBook Pro._companion-link._tcp.local.'"  MSG: UDP ALERT
27 51Ether / IP / UDP / DNS Qry "b'_googlecast._tcp.local.'"  MSG: UDP ALERT
28 52Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
29 55Ether / IP / UDP / DNS Qry "b'_CC32E753._sub._googlecast._tcp.local.'"  MSG: UDP ALERT
30 56Ether / IP / UDP / DNS Ans  / Padding MSG: UDP ALERT
31 57Ether / IP / UDP 172.25.48.196:53708 > 239.255.255.250:ssdp / Raw MSG: UDP ALERT
32 59Ether / IP / UDP 172.25.168.74:netbios_dgm > 172.25.191.255:netbios_dgm / NBTDatagram / Raw MSG: UDP ALE
```

ALL PACKETS

```
42 Ether / IP / UDP / DNS Ans "b'Prince\xe2\x80\x99s MacBook Pro._companion-link._tcp.local.'"
43 Ether / ARP who has 172.25.209.193 says 172.25.208.1 / Padding
44 Ether / ARP who has 172.25.168.93 says 172.25.160.1 / Padding
45 Ether / ARP who has 172.25.42.67 says 172.25.40.1 / Padding
46 Ether / ARP who has 172.25.215.180 says 172.25.208.1 / Padding
47 Ether / ARP who has 172.25.161.175 says 172.25.160.1 / Padding
48 Ether / ARP who has 172.25.211.124 says 172.25.208.1 / Padding
49 Ether / ARP who has 172.25.172.192 says 172.25.160.1 / Padding
50 Ether / ARP who has 172.25.180.250 says 172.25.160.1 / Padding
51 Ether / IP / UDP / DNS Qry "b'_googlecast._tcp.local.'"
52 Ether / IP / UDP / DNS Ans  / Padding
53 Ether / ARP who has 172.25.163.52 says 0.0.0.0 / Padding
54 Ether / ARP who has 172.25.160.1 says 172.25.162.167 / Padding
55 Ether / IP / UDP / DNS Qry "b'_CC32E753._sub._googlecast._tcp.local.'"
56 Ether / IP / UDP / DNS Ans  / Padding
57 Ether / IP / UDP 172.25.48.196:53708 > 239.255.255.250:ssdp / Raw
58 Ether / ARP who has 172.25.216.1 says 172.25.220.80 / Padding
59 Ether / IP / UDP 172.25.168.74:netbios_dgm > 172.25.191.255:netbios_dgm / NBTDatagram / Raw
60 Ether / ARP who has 172.25.174.103 says 172.25.160.1 / Padding
61 Ether / IP / UDP 169.254.162.85:61988 > 239.255.255.250:ssdp / Raw
```

ALERT DECODED

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 1
ST: urn:dial-multiscreen-org:service:dial:1
USER-AGENT: Microsoft Edge/118.0.2088.61 Windows
```

Figure 5.8: Alert decoded and generation of reports

NOTE : The results shown in testing is done in an isolated environment.

CHAPTER-6:
PROJECT OUTCOME AND APPLICABILITY

6.1 PROJECT OUTCOME

In this chapter, we present the outcomes and significance of our NIDS project. The project's findings, impact assessment, threat intelligence, and mitigation strategies are discussed to provide a comprehensive overview of the analysis results.

A Network Intrusion Detection System (NIDS) is a critical component of a network's security infrastructure. Its primary function is to monitor network traffic for suspicious or malicious activity and generate alerts when it detects potential security threats. The project outcomes of implementing a NIDS can be varied, but they generally revolve around improving network security and threat detection. Here are some typical project outcomes of a NIDS implementation:

1. Enhanced Security: One of the primary outcomes is improved network security. NIDS helps in identifying and responding to security incidents promptly, reducing the risk of data breaches and unauthorized access.
2. Real-time Threat Detection: NIDS provides real-time monitoring of network traffic, enabling the detection of malicious activity as it occurs. This rapid detection helps in preventing or mitigating security breaches.
3. Reduced False Positives: An effective NIDS can be tuned to minimize false positive alerts, ensuring that security personnel are not overwhelmed with irrelevant or benign notifications.
4. Incident Response Capability: NIDS alerts serve as triggers for incident response actions. Project outcomes may include well-defined incident response procedures and teams trained to act upon NIDS alerts effectively.
5. Log and Alert Aggregation: Centralized logging and alerting capabilities are often part of NIDS projects. These systems collect data from various network sensors and consolidate it for analysis.
6. Network Visibility: NIDS projects may lead to increased visibility into network traffic and behavior, enabling administrators to understand network patterns and anomalies.
7. Compliance and Reporting: NIDS systems are essential for meeting regulatory compliance requirements. Projects can lead to the development of reporting mechanisms and documentation needed for compliance audits.
8. Threat Intelligence Integration: Some NIDS projects include the integration of threat intelligence feeds to enhance the system's ability to detect emerging threats and known attack patterns.
9. Custom Rule Development: Organizations often create custom detection rules to suit their specific network environment and security needs. A project may involve the development of such custom rules.
10. Network Segmentation: NIDS projects can lead to recommendations or implementations of network segmentation strategies to isolate sensitive data from potential threats and reduce the attack surface.
11. User Training: Implementing a NIDS often involves training network administrators and security staff on how to use the system effectively and respond to alerts.
12. Performance Optimization: NIDS projects may involve performance tuning to ensure that the system can handle the network's traffic volume without introducing latency or affecting network operations

6.2 APPLICABILITY

In this section, we explore how the project outcomes can be applied in various contexts within the field of cybersecurity and threat management.

Practical Use Cases

The analysis findings have practical applications in the following scenarios:

- Incident Response: Our project findings are useful for incident response teams. When a similar malware attack occurs, they can use our insights to understand the threat quickly and take action to stop it before it causes significant damage.
- Monitoring and Compliance: NIDS play a pivotal role in network monitoring and compliance with security standards and regulations. They help organizations track and record network activities, which is essential for compliance reporting and auditing.
- Network Visibility: NIDS provide organizations with enhanced network visibility, allowing them to understand network traffic, user behavior, and potential vulnerabilities. This visibility aids in optimizing network performance and security.
- Forensic Analysis: NIDS maintain logs of network activities, including alerts and identified threats. These logs are valuable for post-incident forensic analysis, helping organizations understand the scope and impact of security incidents.

CHAPTER-7: CONCLUSION

7.1 CONCLUSION

In this final chapter, we reflect on our malware analysis project and draw key conclusions. Summarizing the project's objectives, methodologies and outcomes. Additionally, we discuss the broader context of malware analysis and its significance in the future for evolving cyber threats. In conclusion, this report provides a comprehensive and accessible analysis of various approaches employed in the design of Network Intrusion Detection Systems (NIDS). Drawing from a substantial body of research spanning from 2005 to 2020, we employed citations as a quantitative measure to gauge the popularity and impact of different intrusion detection methods.

The report offers a series of tables that facilitate a quick comparison of diverse NIDS, shedding light on research trends and the overall scope of the field. It includes a thorough review of multiple datasets, encompassing their characteristics, advantages, disadvantages, and citation analysis. Additionally, the report tabulates the various approaches in network intrusion detection, delineating their respective strengths and weaknesses.

Our findings reveal compelling insights into research trends surrounding different IDS techniques, facilitating a comparative analysis based on citations and the number of published articles. Notably, we observed that conference publications tend to garner higher citation counts than journal articles.

This report serves as a valuable resource for professionals, researchers, and practitioners in the realm of cybersecurity, offering a consolidated view of the NIDS landscape, its evolution, and the prominence of various methodologies. It underscores the critical role of quantitative measures in understanding the dynamics of intrusion detection research, contributing to informed decision-making in the ever-evolving field of network security

7.2 KEY FINDINGS AND CONTRIBUTIONS

The provided code appears to be a Python script for network traffic analysis and intrusion detection. It involves several libraries and functionalities to capture, process, and analyze network packets. Here are some key findings and contributions of the code:

1. **Packet Capture and Analysis:** The code leverages the Scapy library to capture network packets, process them, and extract various attributes, such as source IP, destination IP, protocol, and port numbers.
2. **Rule-Based Alerting:** The script allows users to define intrusion detection rules in a "rules.txt" file. These rules can specify conditions based on source IP, destination IP, protocol, and port numbers. When a packet matches a defined rule, an alert is generated, and the matched packets are stored for further analysis.
3. **Alert and Packet Presentation:** The script presents the detected alerts and captured packets in a graphical user interface (GUI) created using PySimpleGUI. It displays alerts based on rule matches and provides details about each packet, such as summary and payload content.
4. **Decoding Payloads:** The code includes functionality to decode payload content from TCP and UDP packets, and it can present this decoded content for further inspection.

5. **Stream Analysis:** The script allows the analysis of TCP and HTTP2 streams. Users can load and view TCP streams or HTTP2 streams in a separate window.
6. **HTTP Object Extraction:** The code extracts HTTP objects from network packets and can display their content.
7. **Save Packet Capture:** Users can save packet captures as PCAP files for later analysis or documentation.
8. **IP Routing Information:** The script identifies available network interfaces and stores information about them, enabling packet capture and analysis on specific interfaces.
9. **Threading for Packet Capture:** It uses a separate thread for packet capture to avoid blocking the main program and ensure continuous capture.
10. **Error Handling:** The script includes error handling to address potential exceptions and issues that may arise during packet processing.

FUTURE SCOPE

The future of Network Intrusion Detection Systems (NIDS) is marked by evolving technology and emerging security challenges. NIDS will advance in threat detection, integrating machine learning and AI for recognizing complex attack patterns. They will specialize in IoT and cloud security, integrate with SIEM systems, and enforce the Zero-Trust model. NIDS will also address machine-to-machine communication security, incorporate user behavior analytics, and consider privacy and ethical concerns. These developments aim to strengthen network security in an ever-changing cyber landscape, ensuring NIDS remain at the forefront of defense against emerging threats.:

1. **Advanced Threat Detection:** NIDS will continue to evolve in their ability to detect more advanced and sophisticated threats. This includes improved machine learning algorithms and artificial intelligence that can recognize complex attack patterns and anomalies.
2. **Behavioral Analysis:** Future NIDS will focus more on behavioral analysis, which goes beyond simple signature-based detection. They will monitor user and network behavior to detect subtle deviations indicative of insider threats or stealthy attacks.
3. **IoT and IIoT Security:** As the Internet of Things (IoT) and Industrial Internet of Things (IIoT) proliferate, NIDS will adapt to secure these devices and networks. NIDS solutions will become more specialized in IoT threat detection.
4. **Cloud Security:** With the increasing adoption of cloud services, NIDS will need to integrate with cloud environments and adapt to the specific challenges of securing data and applications in the cloud.
5. **Integration with SIEM:** NIDS will further integrate with Security Information and Event Management (SIEM) systems for enhanced correlation and analysis of security data. This will improve overall threat detection and response capabilities.
6. **Zero-Trust Security:** The Zero-Trust security model, which assumes that threats can come from within and outside the network, will become more mainstream. NIDS will play a crucial role in enforcing this model by continuously monitoring and verifying network access.
7. **Machine-to-Machine Communication Security:** With the growth of machine-to-machine communication in industries such as healthcare and manufacturing, NIDS will focus on securing these communications to prevent vulnerabilities in automated systems.

7.3 REFERENCES:

[1] R. Heady, G. Luger, A. Maccabe, and M. Servilla, ‘‘The architecture of a network level intrusion detection system,’’ Los Alamos Nat. Lab., New Mexico Univ., Albuquerque, NM, USA, Tech. Rep. LA-SUB-93- 219 ON: DE97002400; TRN: AHC29703%%44, 1990.
[2] https://ieeexplore.ieee.org/.
[3] A. A. Shah, Y. D. Khan, and M. A. Ashraf, ‘‘Attacks analysis of TCP and UDP of UNCW-NB15 dataset,’’ VAWKUM Trans. Comput. Sci., vol. 8, no. 1, pp. 48–54, 2020.,
[4] geeksforgeeks.org
[5] N. Ben-Asher and C. Gonzalez, ‘‘Effects of cyber security knowledge on attack detection,’’ Comput. Hum. Behav., vol. 48, pp. 51–61, Jul. 2015. [49] R. Sommer and V. Paxson, ‘‘Outside the closed world: On using machine learning for network intrusion detection,’’ in Proc. IEEE Symp. Secur. Privacy, May 2010, pp. 305–316
[6] J. Viinikka, H. Debar, L. Mé, and R. Séguier, ‘‘Time series modeling for IDS alert management,’’ in Proc. ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS), 2006, pp. 102–113.
[7] The Top List of Academic Search Engines, Paperpile, Cambridge, MA, USA, Jun. 2021. [Online]. Available: https://paperpile.com/g/academicsearch-engines/
[8] M. Pihelgas, ‘‘A comparative analysis of open-source intrusion detection systems,’’ Tallinn Univ. Technol., Univ. Tartu, Tallinn, Estonia, Tech. Rep., 2012. [Online]. Available: http://mauno.pihelgas.eu/files/Mauno_Pihelgas
A_Comparative_Analysis_of_OpenSource_Intrusion_ Detection_Systems.pdf
[9] https://www.researchgate.net/