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

KOTHRIKALAN, SEHORE

MADHYA PRADESH - 466114

OCT 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 D. Saravanan**, Program Chair, Cyber Security and Digital Forensics for much of his valuable support encouragement in carrying out this work.

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

I would like to thank all the technical and teaching staff of the School Computing Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

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 INCLUDING TECHNIQUES

Our project's aim is to perform an analysis on the malware, which is signature based using static and dynamic analysis. In short, our main goal is to study a particular piece of harmful software called malware. We aim to:

1. **Understand How It Works**: Figure out what this malware does and how it can harm our computers.
2. **Identify Its Threat Level**: Determine how dangerous it is and what kind of damage it can cause.
3. **Collect Important Information**: Gather data that can help us and others detect and defend against similar threats in the future.
4. **Share Knowledge**: Share what we learn with others to make the digital world safer for everyone.

Techniques:

The Project is in Python program for monitoring network traffic, detecting suspicious packets based on user-defined rules, and displaying packet details. It uses various libraries like PyShark and Scapy. It also has a graphical user interface built with PySimpleGUI. It can capture and analyze network packets, and it can filter, save, and display relevant information about the packets. Additionally, it can show decoded HTTP2 and TCP streams as well as the payloads from captured packets. The code also reads and processes custom alert rules from a "rules.txt" file.

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 CORE AREA OF THE PROJECT

The core area of this project is network traffic monitoring and analysis. It captures packets, detects predefined patterns, and displays alerts in a user-friendly GUI. Users can inspect packet payloads, analyze network streams, and extract HTTP objects. The tool also logs data, supports SSL/TLS decryption, and allows for rule customization. The primary focus is on providing a versatile network monitoring and analysis tool with an emphasis on user-friendly interaction.

2.3 EXISTING WORK

Methods:

2.3.1 Methods 1: Signature-Based Detection

In the first method, Network Intrusion Detection Systems (NIDS) rely on signature-based detection techniques. They maintain a database of known attack patterns or "signatures." These signatures are compared with the network traffic as it flows through the system. When a match is identified between the observed traffic and a known signature, the NIDS generates an alert. This method is highly effective at identifying well-documented threats, including viruses, malware, and established attack techniques. The detection process is based on predefined patterns of malicious behavior.

2.3.2 Method 2: Anomaly-Based Detection

The second method used by NIDS involves anomaly-based detection. Here, NIDS establish a baseline of normal network behavior by continuously monitoring network traffic over time. They build a profile of expected activity and behavior within the network. When network traffic deviates from this established baseline, it is flagged as potentially suspicious or malicious. Anomaly-based NIDS are capable of detecting previously unknown or "zero-day" threats by identifying unusual patterns or activities in the network. This approach is essential for identifying emerging attack techniques that lack predefined signatures.

2.3.3 Method 3: Deep Packet Inspection

In the third method, NIDS perform deep packet inspection, scrutinizing both the header and payload of network packets. This technique allows NIDS to examine a range of packet attributes, including source and destination IP addresses, port numbers, and the actual content of data packets. By inspecting packets at this granular level, NIDS can uncover hidden threats that might not be apparent through traditional inspection methods. Deep packet inspection enhances the NIDS's ability to detect various forms of malicious activities, making it a valuable part of their overall detection capabilities.

In all methods, NIDS operate continuously, providing real-time or near real-time monitoring of network traffic. They send alerts to centralized management systems or network administrators when suspicious activity is detected. These alerts include crucial information about the potential threat's nature, source, and the affected network segment, enabling swift investigation and response. Additionally, NIDS maintain detailed logs of all network activity, including alerts and identified threats, which are essential for forensic analysis, compliance reporting, and security auditing. Overall, these methods work together to protect networked systems from a wide range of cyber threats, adapting to the evolving landscape of network security

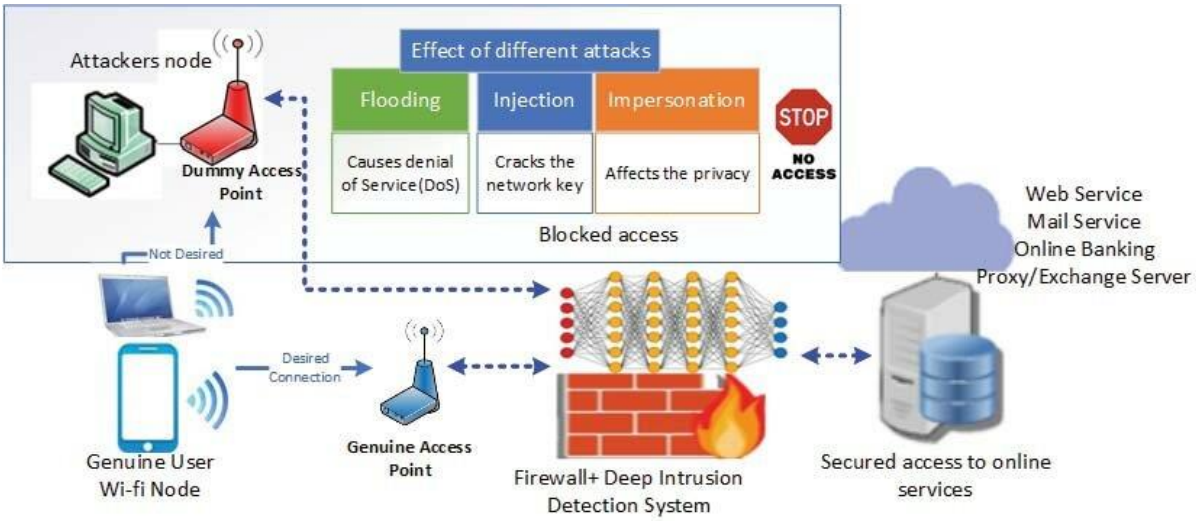

FIGURE 2.1 NIDS MANAGEMENT SYSTEM

2.4 PROS AND CONS OF THE APPROACHES/METHODS

Pros:

1. **Enhanced Security:** NIDS play a crucial role in enhancing network security by continuously monitoring and identifying potential threats and malicious activities. They provide an additional layer of defense against cyberattacks.
2. **Real-Time Detection:** NIDS operate 24/7 and offer real-time or near real-time monitoring of network traffic. This real-time detection allows for immediate responses to potential threats, minimizing the risk of security breaches.
3. **Comprehensive Coverage:** They use a combination of signature-based and anomaly-based detection techniques, along with deep packet inspection, providing comprehensive coverage. This multi-faceted approach increases the likelihood of detecting a wide range of threats.
4. **Zero-Day Threat Detection:** Anomaly-based detection is effective in identifying previously unknown or "zero-day" threats by flagging any deviations from the established baseline of normal network behavior. This is vital in an environment where new threats continually emerge.
5. **Forensic Analysis and Compliance:** NIDS maintain logs of network activity, including alerts and identified threats. These logs are invaluable for forensic analysis, compliance reporting, and security auditing, helping organizations understand and learn from security incidents.

6. **Customizability:** NIDS can often be customized to specific network environments, allowing organizations to tailor their intrusion detection to their unique security needs.

Cons:

1. **False Positives:** One significant drawback of NIDS is the potential for false positives. Anomaly-based detection may flag legitimate network behavior as suspicious if it deviates from the baseline. This can result in unnecessary alerts and increased operational overhead.
2. **Resource Intensive:** Deep packet inspection, in particular, can be resource-intensive and may impact network performance. Organizations need to invest in capable hardware and network resources to avoid performance bottlenecks.
3. **Complex Configuration:** Setting up and configuring NIDS can be complex and requires in-depth knowledge of network security. Misconfigurations can lead to vulnerabilities or a high number of false positives.
4. **Signature Updates:** In signature-based detection, NIDS rely on databases of known attack signatures. Regular updates are necessary to stay effective against new threats. Failing to update signatures can leave the system vulnerable.
5. **Evasion Techniques:** Advanced attackers may use evasion techniques to bypass NIDS, making them less effective against sophisticated and targeted attacks.
6. **Privacy Concerns:** Deep packet inspection raises privacy concerns as it involves examining the content of network traffic. Organizations must consider privacy regulations and user consent when implementing NIDS.
7. **Cost:** Implementing NIDS, especially for large-scale networks, can be costly. It involves expenses for hardware, software, ongoing maintenance, and skilled personnel

2.5 ISSUES/OBSERVATIONS FROM INVESTIGATION

One critical area of concern often revolves around security vulnerabilities. Investigations might reveal the presence of security vulnerabilities within an information system. These vulnerabilities can take the form of software vulnerabilities or misconfigured settings that could potentially be exploited by malicious actors, posing a significant threat to an organization's data and systems.

Another key issue is compliance violations, which might entail the discovery of non-compliance with relevant laws, regulations, or internal policies. In such cases, investigations

may uncover instances of data protection violations or non-adherence to industry standards, potentially exposing an organization to legal and reputational risks.

Data breaches represent a major concern in the digital age. Investigations frequently yield insights into data breaches, including unauthorized access to sensitive data or incidents of data leakage. Such breaches can lead to privacy violations and trigger legal consequences, making them critical issues to address promptly.

Performance problems are commonly observed during investigations, which might include bottlenecks, resource overutilization, or network latency affecting user experience. Identifying and resolving these performance problems are essential for maintaining a seamless and responsive IT environment.

Operational inefficiencies are another area of focus. Investigations may highlight opportunities to optimize and streamline operational processes to enhance efficiency and cost-effectiveness. Operational inefficiencies can result in unnecessary resource expenditures and hinder an organization's competitive edge.

Network anomalies may be detected during investigations, signaling unusual network traffic patterns or suspicious activities. Such network anomalies can be indicative of security breaches or underlying network issues, warranting further investigation and mitigation.

Recognizing gaps in incident response is vital for incident management and recovery. Investigations may unveil areas where an organization's incident response plan falls short, such as delays in response times or inadequate containment measures. Addressing these gaps is critical for minimizing the impact of security incidents.

Observations of user behavior anomalies are also pertinent, as these might indicate compromised accounts or insider threats. Detecting and addressing these anomalies can mitigate the risks associated with malicious insiders and unauthorized access.

In the realm of software and technology, software bugs are often observed. These defects can affect system stability, functionality, or security and require prompt attention and remediation.

Communication breakdowns between teams or departments may be noted, hampering effective collaboration during incidents. Communication breakdowns can lead to misunderstandings, delays in response, and ineffective coordination during critical situations.

Additionally, data integrity issues, third-party risks, inadequate documentation, physical security concerns, resource allocation challenges, regulatory non-compliance, data retention, and deletion issues, as well as cultural or training problems may all feature as issues and observations within the context of an investigation.

Ultimately, addressing these issues and observations often involves developing appropriate recommendations and action plans to mitigate risks, improve processes, and bolster the overall resilience and security of an organization.

2.6 SUMMARY

The project, centered around Network Intrusion Detection Systems (NIDS), plays a pivotal role in enhancing network security. NIDS continually monitor network traffic, utilizing deep packet inspection techniques to scrutinize both packet headers and content. This comprehensive approach allows them to employ two primary detection methods: signature-based and anomaly-based detection.

Signature-based detection, by comparing network traffic to a database of known attack patterns, effectively identifies known threats, such as viruses and malware. Simultaneously, anomaly-based detection establishes a baseline of normal network behavior, detecting deviations that indicate potential threats, including emerging or previously unknown attacks.

However, this approach is not without its drawbacks. Signature-based detection is less effective against zero-day threats, while anomaly-based detection may generate false positives or require complex configuration. Furthermore, the continuous 24/7 operation of NIDS places demands on system resources and necessitates careful tuning to prevent overwhelming network administrators with alerts.

During investigations, numerous issues and observations may emerge. These range from security vulnerabilities and compliance violations to data breaches, performance problems, operational inefficiencies, network anomalies, and gaps in incident response. Additionally, anomalies in user behavior and software bugs can be identified, alongside concerns like communication breakdowns, data integrity, and third-party risks.

In conclusion, NIDS represent a multifaceted approach to network security, with their strengths in identifying known and emerging threats counterbalanced by potential false positives and resource challenges. The issues and observations from investigations underscore the importance of proactively addressing security vulnerabilities, optimizing performance, and enhancing incident response and communication, all vital components in safeguarding networked systems from evolving cyber 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 HARWARE REQUIREMENTS AND SOFTWARE 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.

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

3.3.1 Data Requirements:

- The system shall capture network traffic data.
- The system shall support the capture and storage of network packets in the PCAP (Packet Capture) file format.
- The system shall store network packet data in a structured format for analysis.
- The system shall store alert information, including packet details and decoded payloads.
- The system shall store network packet timestamps, source and destination IP addresses, protocols, ports, and payload data.

3.3.2 Functional Requirements:

- The system shall start capturing network traffic data upon user request (e.g., by clicking the "STARTCAP" button).
- The system shall stop capturing network traffic data upon user request (e.g., by clicking the "STOPCAP" button).
- The system shall display a list of alert packets that match predefined intrusion detection rules.
- The system shall allow users to view detailed information of selected packets, including packet payload.
- The system shall provide the ability to save captured network traffic to PCAP files.
- The system shall allow users to refresh the predefined intrusion detection rules.
- The system shall allow users to load and display TCP and HTTP2 streams.
- The system shall provide the ability to view and analyze HTTP2 streams' details.
- The system shall allow users to load and display HTTP object data within captured network traffic.

3.3.3 Performance and Security Requirements:

- The system should efficiently capture and process network traffic to minimize performance impact on the host machine.

- The system should handle a large volume of network packets without crashing or slowing down.
- The system should ensure the security of captured data and any saved PCAP files.
- The system should prevent unauthorized access to the captured data and provide authentication mechanisms for system access.

3.3.4 Look and Feel Requirements:

- The user interface should be user-friendly, with clear buttons and labels.
- The interface should provide real-time updates for captured network traffic and alerts.
- The system should display network packet data in a readable and well-organized format.
- The user interface should support easy navigation and interaction with network packets and streams.
- The system should provide options to customize the appearance or theme of the user interface.

3.4 SUMMARY

The provided document outlines the requirements and components for a Network Intrusion Detection System (NIDS). It begins by emphasizing the importance of proper hardware and software prerequisites for effective intrusion detection. Hardware requirements include a high-performance host system capable of running virtual machines and sufficient disk space, as well as a virtualization platform like VMware Workstation or VirtualBox. The software requirements cover operating systems (Windows 10 and Linux), virtualization software (VM Virtual Box), network capture tools (NMAP, PYSHARK, TCAP), and various Python packages like PySimpleGUI and scapy. It also mentions specific project requirements, data requirements, functional requirements (including the ability to capture, store, and analyze network packets), performance and security requirements (efficient processing and data security), and look and feel requirements (user-friendly interface with real-time updates). The document provides a comprehensive overview of the NIDS system's prerequisites and functionalities

CHAPTER - 4
DESIGN METHODOLOGY AND ITS NOVELTY

4.1 METHODOLOGY AND GOAL

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

4.2 FUNCTIONAL MODULES DESIGNS AND ANALYSIS

1. Import Statements: The script begins with importing various libraries such as Scapy, PyShark, PySimpleGUI, and others. These libraries are used for packet capture, analysis, and user interface.
2. Rule Parsing: The readrules() function reads intrusion detection rules from a file named "rules.txt." These rules are expected to be in a specific format, starting with "alert" and containing conditions like source IP, destination IP, ports, and a message.
3. Rule Processing: The process_rules() function parses the rules and extracts specific parameters like protocols, source IPs, destination IPs, source ports, destination ports, and messages.
4. Packet Capture: The script uses Scapy to capture network packets. It creates a thread for packet capture, which runs in the background.

5. User Interface: PySimpleGUI is used to create a graphical user interface (GUI) for the script. The GUI allows users to start and stop packet capture, display captured packets, refresh rules, load and analyze TCP/HTTP2 streams, and save alerts.
6. Alert Detection: The script applies the intrusion detection rules to the captured packets. If a packet matches a rule, it's considered an alert, and details about the alert are displayed in the GUI.
7. Stream Analysis: The script can load and analyze TCP and HTTP2 streams. It provides functionality to view the content of specific streams.
8. Save PCAP: Users can save the captured packets to a PCAP file for further analysis.
9. Event Handling: The script uses event-driven programming with PySimpleGUI, allowing users to interact with the GUI, view alerts, and analyze network streams.
10. Loop and Exit: The main loop continuously checks for user events and runs until the user exits the application.

4.3 SOFTWARE ARCHITECTURAL DESIGNS

The provided Python script leverages several libraries, including PySimpleGUI, scapy, and pyshark, to create a graphical user interface for network packet analysis and intrusion detection. Users can initiate packet capture and apply predefined rules to detect suspicious network activity based on criteria such as source and destination IP addresses, protocols, and ports. The code processes and logs captured packets, enabling users to examine detailed information and packet payloads. Additionally, it offers the functionality to inspect and analyze TCP and HTTP2 streams and extract HTTP objects. Users can save the collected packets in a PCAP file. This versatile tool is designed for network administrators and security professionals, providing an intuitive interface for monitoring and responding to potential network threats.

4.4 SUBSYSTEM SERVICES

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.

4.5 USER INTERFACE DESIGNS

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.

4.6 SUMMARY

Outlines a comprehensive design for a Network Intrusion Detection System (NIDS) with a focus on packet capturing-based analysis. The document emphasizes the importance of packet capturing for identifying network intrusions and delves into TCP and UDP packet analysis. It highlights the functional modules and software architecture of the NIDS, incorporating tools such as Scapy, PyShark, and PySimpleGUI for packet capture, rule processing, and user interaction. The system processes predefined intrusion detection rules, captures packets, and offers an intuitive graphical user interface for users to manage and analyze network traffic. It provides real-time alert detection, stream analysis, and the ability to save captured packets for further examination. The NIDS is designed to cater to network administrators and security professionals for monitoring and responding to potential network threats in a user-friendly and efficient manner, making it a valuable tool for network security.

CHAPTER-5
TECHNICAL IMPLEMENTATION & ANALYSIS

5.1 OUTLINE

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.

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 TECHNICAL CODING AND CODING SOLUTIONS AND WORKING

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.

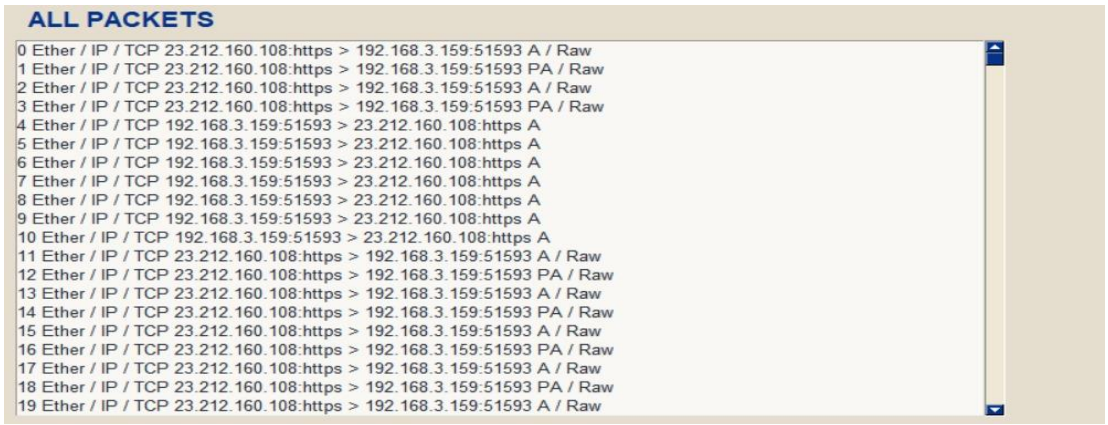

Figure 5.3: Displaying all packets

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:

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

5.4 TEST AND VALIDATION

Step1 : NIDS is implemented on windows system.

To start the operation on windows system , we have used python nids1.py

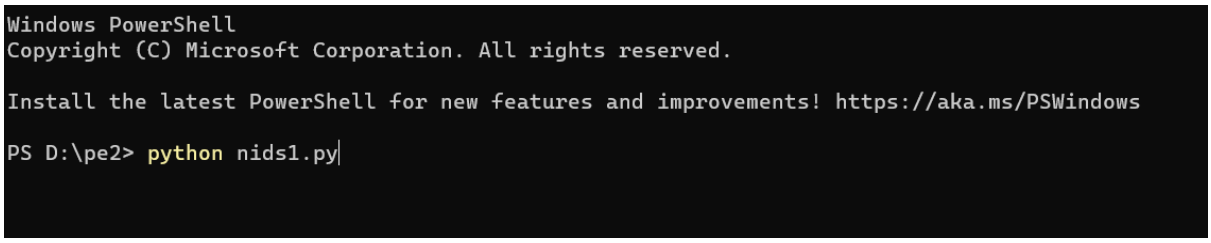

Figure 5.5: Implementation starts

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

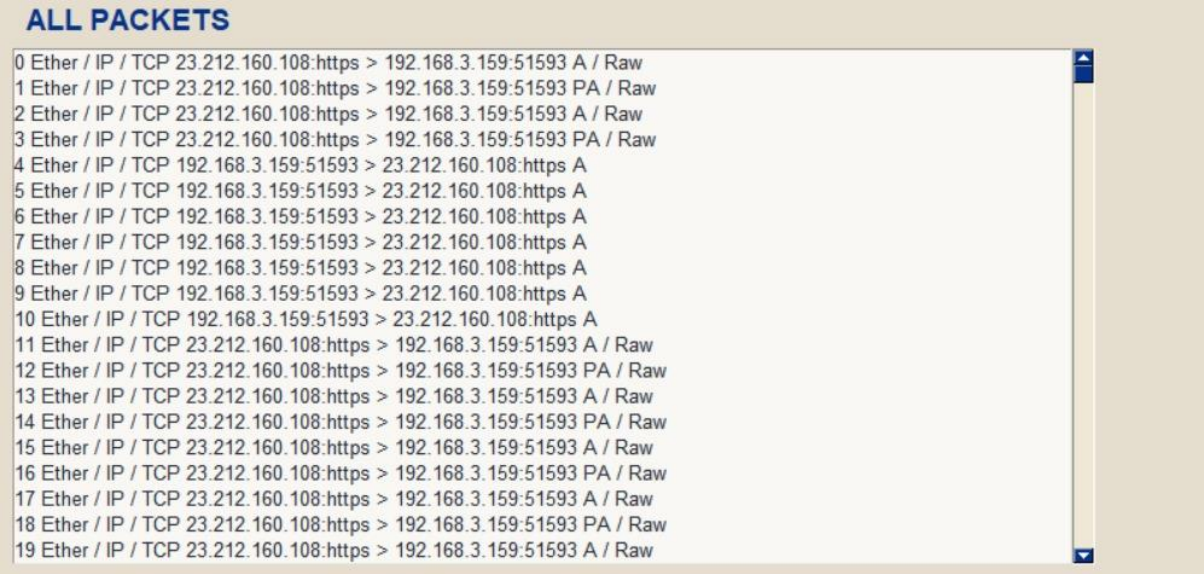

Figure 5.6: Packet capturing and analyzing

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

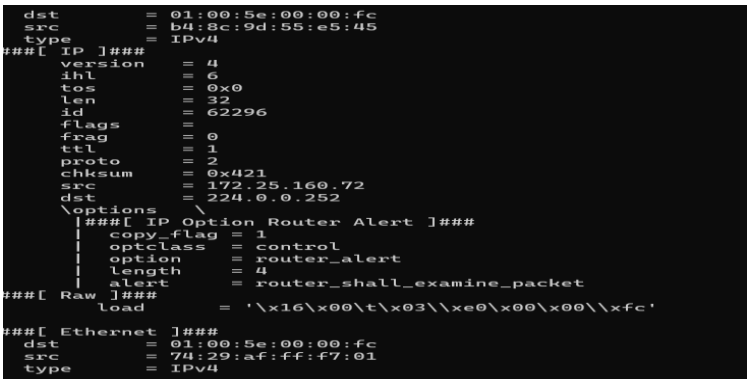

Figure 5.7.1: Result of analyzing(CLI)

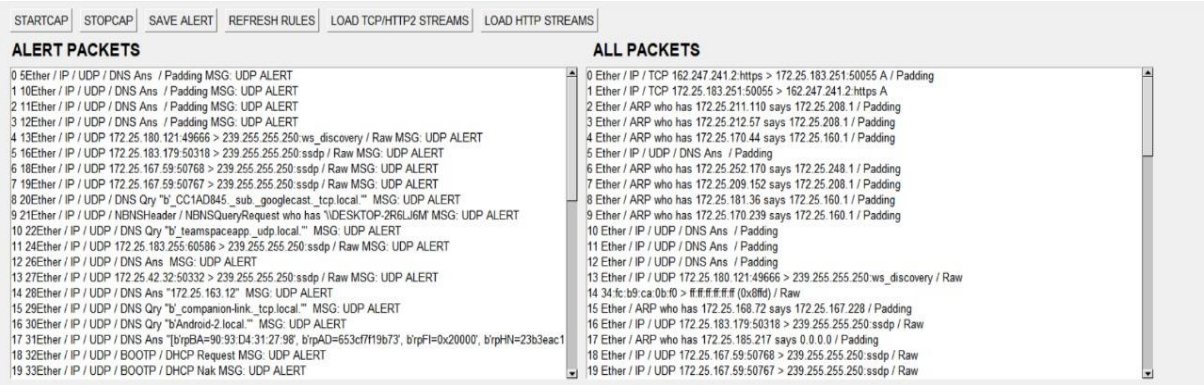

Figure 5.7.2: Result of analyzing(GUI)

Step4 : Decoding of alerts and generation of report in GUI

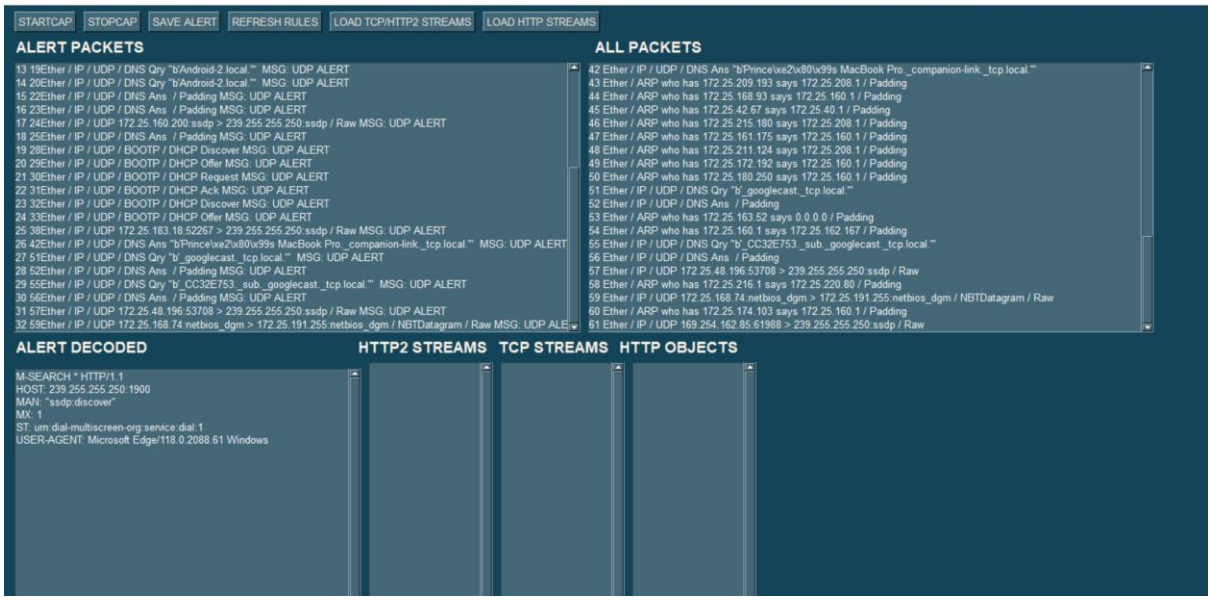

Figure 5.8: Alert decoded and generation of reports

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

5.5 Summary

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.

CHAPTER-6

PROJECT OUTCOME AND APPLICABILITY

6.1 OUTLINE

- The chapter discusses the outcomes and significance of a Network Intrusion Detection System (NIDS) project.
- It highlights the key objectives of a NIDS, which include improved network security, real-time threat detection, reduced false positives, incident response capabilities, and more.
- The chapter also explores the potential outcomes of NIDS implementation, such as enhanced security, threat intelligence integration, compliance and reporting, and user training.
- In the second part of the chapter, it outlines the key implementations of the system, focusing on packet capture and analysis, rule-based alerting, alert presentation, and decoding payloads.
- It describes the project's practical applicability in real-world scenarios, including incident response, monitoring and compliance, network visibility, and forensic analysis.

6.2 KEY IMPLEMENTATIONS OUTLINES OF THE SYSTEM

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.

6.3 SIGNIFICANT 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:

11. 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.

12. 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.
13. 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.
14. 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.
15. 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.
16. Network Visibility: NIDS projects may lead to increased visibility into network traffic and behavior, enabling administrators to understand network patterns and anomalies.
17. 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.
18. 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.
19. 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.
20. 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.
21. User Training: Implementing a NIDS often involves training network administrators and security staff on how to use the system effectively and respond to alerts.
22. 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

6.4 PROJEXT APPLICABILITY ON REAL-WORLD APPLICATIONS

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.

6.5 INFERENCE

The chapter emphasizes the importance of a Network Intrusion Detection System (NIDS) in network security. It outlines the primary objectives of NIDS, which revolve around enhancing security, improving real-time threat detection, and ensuring efficient incident response. The project's outcomes are highlighted, encompassing aspects such as user training, network segmentation, and threat intelligence integration.

The second part of the chapter delves into the key implementations of the system, showcasing its functionality in packet capture and analysis, rule-based alerting, and alert presentation. It provides an in-depth look at the code's capabilities, including the decoding of payloads and the analysis of TCP and HTTP2 streams.

Furthermore, the chapter discusses the practical applicability of the project findings in real-world contexts. It suggests that the outcomes are highly relevant for incident response teams, network monitoring and compliance, enhancing network visibility, and facilitating forensic analysis. These findings can be instrumental in strengthening an organization's cybersecurity posture and threat management capabilities.

CHAPTER-7

CONCLUSION AND RECOMMENDATION

7.1 OUTLINE

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 LIMITATIONS/CONSTRAINTS OF THE SYSTEM

The system described for testing the effectiveness of the NIDS and the proposed enhancements have certain limitations and constraints:

1. Limited OS Compatibility: The system mentions the use of Kali Linux as an attacker's machine and another operating system that fulfills software and hardware

requirements. The compatibility of the NIDS and the testing environment might be limited to specific operating systems, potentially missing out on vulnerabilities in other OS environments.

2. Testing Scope: The proposed testing approach focuses on a specific type of attack scenario involving an isolated system. This scope may not cover the full spectrum of potential threats and vulnerabilities that a network might face in a real-world scenario. It's essential to complement this testing with a more comprehensive and diverse set of testing scenarios.
3. Resource Intensive: Setting up an isolated system with dedicated hardware and software for testing can be resource-intensive. Smaller organizations or those with limited resources might not be able to replicate this environment, which could affect the comprehensiveness of their NIDS testing.
4. Real-World Variability: In practice, network attacks can be highly variable, and the effectiveness of an NIDS may differ based on the specific attack vectors and techniques used by real attackers. Testing under controlled conditions might not fully represent these real-world variations.
5. Ethical Considerations: Conducting attacks, even in a controlled environment, raises ethical and legal concerns. Unauthorized access or disruption of systems can potentially violate laws and regulations. Testing should be conducted within the boundaries of legal and ethical guidelines.
6. Limited Ruleset: The system mentions a limited set of rules for NIDS testing. Effective NIDS should ideally have a comprehensive set of rules to detect various types of threats. Enhancing the ruleset is a valid consideration for better coverage, but it also requires ongoing maintenance to stay up to date with emerging threats.
7. Automation Challenges: Implementing AI for automating NIDS and alert decoding is a complex task. It requires extensive resources, including skilled personnel, data for training AI models, and ongoing monitoring. It may not be feasible for all organizations, especially smaller ones.
8. False Positives/Negatives: NIDS systems often generate false positives (incorrectly identifying normal traffic as threats) and false negatives (failing to detect actual threats). The effectiveness of an NIDS should be measured not only in terms of detections but also in minimizing these false alerts.

9. Cost and Scalability: The proposed system's cost may be a limiting factor for smaller organizations. The scalability of the NIDS to meet the demands of a growing network should also be considered.
10. Maintenance and Updates: Keeping the NIDS ruleset and the system itself up to date is essential. Neglecting this aspect can result in reduced effectiveness over time.

7.3 FUTURE ENHACEMENTS

Future scope:

Testing:

The test result shown above is of random checking of network. To check the effectiveness of the of NIDS it can be tasted under an Isolated system, which will consist of :

i. Two operating system in which one should kali linux and use it as attackers machine.

ii. The second operating system must fulfil the software and hardware requirement of the system.

iii. The attackers should attack the second operating system using the kali linux.

These will result in checking the effectiveness of the NIDS .

Enhancement:

There always scope of enhancement in the field of cybersecurity, and our project is based on that. Following are the enhancement can be made :

1. **Setrules:** The project is working currently on a limited setrules, that can be further increased in order to make it move effective.
2. **Automation:** The project can be automated using AI for maintaining network security and decoding alerts.
3. **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.
4. **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.

5. **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.
6. **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.
7. **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.
8. **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.
9. **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.

7.4 INFERENCE

In conclusion, this project has provided a comprehensive overview of Network Intrusion Detection Systems (NIDS) and their evolution in the field of network security. The project's objectives included analyzing various NIDS methodologies, understanding research trends, and highlighting the critical role of quantitative measures in assessing the effectiveness of intrusion detection methods.

The report has effectively summarized the landscape of NIDS, encompassing research spanning from 2005 to 2020. By employing citations as a quantitative metric, it has measured the popularity and impact of different intrusion detection methods. The project has presented valuable insights into the field, showcasing trends and patterns in intrusion detection research. Notably, the observation that conference publications tend to receive higher citation counts than journal articles underlines the significance of academic conferences in disseminating research findings in this domain.

This report serves as a valuable resource for cybersecurity professionals, researchers, and practitioners. It offers a consolidated view of the NIDS landscape, its historical evolution, and

the prominence of various methodologies. By presenting the strengths and weaknesses of different approaches, it equips professionals with the knowledge needed to make informed decisions in the ever-evolving field of network security.

However, it's important to acknowledge the limitations and constraints associated with the proposed system for testing NIDS effectiveness. These limitations include issues related to operating system compatibility, testing scope, resource requirements, ethical considerations, and the need for ongoing updates and maintenance. Additionally, the enhancements discussed, such as expanding the ruleset and automation using AI, highlight the continuous need for innovation and adaptation in the cybersecurity field.

In the future, NIDS will continue to evolve to address more advanced threats, emphasizing behavioral analysis, IoT and IIoT security, cloud security, integration with SIEM systems, Zero-Trust security, and machine-to-machine communication security. As cyber threats become more sophisticated and diverse, NIDS will play a pivotal role in safeguarding networks and data against a wide range of security challenges. This project provides a foundation for understanding the past and future of NIDS and their essential role in network security.

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/