Real-Time Threat Detection: Suricata Log Analysis and Visualization for Network

MSc Research Project

Master of Science in Cyber Security

Nikhil Nikhil

Student ID: 23161442

School of Computing

National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Nikhil Nikhil		
Student ID:	23161442		
Programme:	Master of Science in Cyber Security	**Year:**	2023-24
Module:	Practicum Part 2		
Supervisor:	Michael Pantridge		
Submission Due Date:	12th August		
Project Title:	Research Project part 2		
Word Count:	8525	**Page Count**	24

Signature:	Nikhil Nikhil
Date:	11-08-24

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Real-Time Threat Detection: Suricata Log Analysis and Visualization for Network

Nikhil Nikhil
23161442

Abstract

Systems that help in detecting different kinds of intrusions in networks are called Intrusion detection Systems (IDSs). There are many popular IDSs that are used both by companies and individuals in the detection of intrusions. Suricata is one such tool. This tool helps in the detection of intrusions and networks. However, studies need to be conducted based on the data collected in the logs generated by the Suricata. So in the study proposed here a system that uses the Suricata to detect intrusions in a network is proposed. In the study the Suricata is set up on a computer connected to a network and cyber-attacks are simulated to assess its performance of the Suricata in detecting intrusions. Along with detecting intrusions the system proposed in the study sends an email as a notification when the Suricata detects an intrusion. The log generated by Suricata when an attack is detected is uploaded to Google Drive. The data associated with the logs generated by Suricata is analysed and visual representations of the data in the logs are generated. The results of the study show that the Suricata is effective in detecting intrusions and that insights about the kind of attacks can be gained when the data associated with logs of the Suricata is analysed. The email sending and analysis of the data associated with the logs is done using methods in Python.

1 Introduction

1.1 Background

Identifying intrusions or cyber-attacks in networks is a crucial aspect of cyber security. Cyber-attacks are happening frequently on a global scale. Cybercriminals carry out attacks against both individuals and organizations to obtain financial gains. In 2023, there was a significant rise in cyber-attacks, with over 343 million people falling victim to these incidents (St. John, 2024). Additionally, it was discovered that data breaches increased by 72% between 2021 and 2023(St. John, 2024). So, for making sure that the data being transferred over a network is safe a layer of security needs to be added. This layer of security can be anything from encrypting the data to adding an Intrusion Detection System (IDS) that helps in detecting the intrusions or cyber-attacks that enter a network (Chu, 2019; Khraisat et al., 2019). In the study performed here, the focus is on the IDSs that detect intrusions in networks.

1.2 Problem Definition

An intrusion in a network can only be removed when it is detected properly. If an intrusion in a network is not detected properly in time, then functionalities and the operation of the network may be compromised due to the intrusions causing a negative impact on the working of the network (Hu et al., 2023). So, the intrusions in the networks must be effectively detected. IDS is a tool that detects intrusions in a network. IDSs work by detecting intrusions and notifying the people using the network in which the intrusion was detected. The IDS is usually placed in front of the Firewall in the network and the network first encounters the IDS before the systems in the network (Figure (1)). An IDS machine gathers data from multiple sources and examines information from various areas within a computer or network to identify potential security breaches, including both external attacks (intrusions) and internal attacks (misuse). IDS utilizes vulnerability assessment (also referred to as scanning) to assess the security of a computer system or network (Jain and Anubha, 2021). The IDSs can help in tracing the changes made in a network, it differentiates between abnormal and normal activities in the network and audits the activity of the system (Jain and Anubha, 2021). IDSs are built using different techniques and one major method that has been used to build IDSs is machine learning (Othman et al., 2018). The intrusion detection can also be carried out using tools that are open source. There are different kinds of IDSs like signature-based IDSs and anomaly-based IDSs (Einy, Oz and Navaei, 2021).). IDSs based on techniques like machine learning are anomaly-based IDSs while IDSs based on open-source tools are signature-based IDSs. The intrusion detection models based on machine learning are not able to capture the packets of data in a network and these models are normally built based on the data previously collected from networks (Abdallah, Eleisah and Otoom, 2022). The signature-based IDSs like Snort, Suricata etc., do not need to be built based on the data associated with networks. These kinds of IDSs can be used readily to detect intrusions in real-time (Díaz-Verdejo et al., 2022). In the study performed here, intrusion detection using an open-source tool is proposed. There are several open-source tools that detect intrusions from networks like Snort, Suricata and Bro (Bada, Nabare and Quansah, 2020). The different open-source IDS tools are:

- **Snort:** Snort is a lightweight intrusion detection tool that logs and analyzes packets passing through the network (Gupta et al., 2017). It is an IDS that employs a set of rules to identify and describe malicious network activities. It uses these rules to detect mismatched packets and generates alerts for users. Snort compares incoming packets to user-defined rules and generates alerts if any matches are found (Gupta et al., 2017).
- **Suricata:** Suricata is a fast, open-source, free, mature, and strong network threat detection system (IPS (Suricata) — NethServer 7 Final, 2024). Suricata has the Multiple Threading functionality (Ouiazzane, Addou and Barramou, 2022). Suricata is a rule-based Intrusion Detection and Prevention engine that operates according to user-defined rules (Saive and Saive, 2022).
- **Bro:** Bro is a Network Intrusion Detection System (IDS) based on Unix. Bro observes network activity and identifies intrusion attempts by analyzing both the content and characteristics of the traffic (Paxson et al., 2006). Bro identifies intrusions by examining network traffic in relation to rules that outline potentially problematic events.

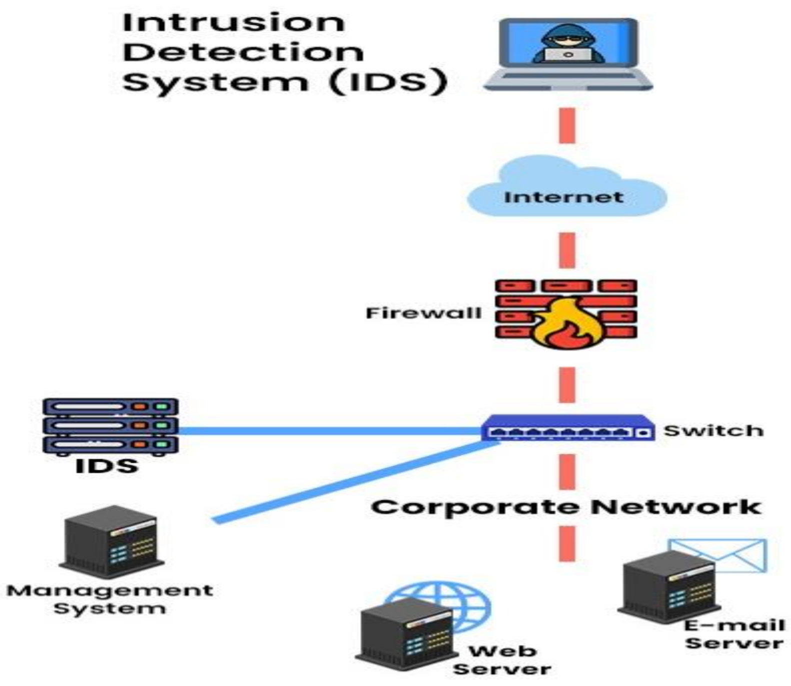


Figure (1): IDS placed in a network (PyNet Labs, 2023)

The research question of the study is "What insights can be gained by analyzing the data stored in the logs by Suricata?".

The aim of the study is to detect network intrusions using Suricata and analyze the logs generated by Suricate when it analyses a network. The objectives of the study are:

- Install the Suricata tool on the PC to use it as an IDS.
- Configure the Suricata so it can be run.
- Define the rules of the Suricata to get a system capable of detecting attacks.
- The logs of Suricata needs to be stored in a drive to obtain logs of attacks detected by the Suricata.
- Simulate attacks for the Suricata to detect.
- Analyze the logs of Suricata and generate visual representations of the data in the logs of Suricata.
- Implement a mail sending functionality when the Suricata detects an attack in the network.

In the study that is carried out here the focus is on the IDS tool Suricata and in the study the Suricata is used to analyze networks and detect intrusions. The novelty of the study is that existing studies that use the Suricata for cyber-attack detection do not analyze the logs of data that is captured and stored by the Suricata from the network. So, in the study performed here the data in the logs of Suricata will be analyzed.

Section 1 of this report contains the background information associated with the detection of intrusions in networks using open-source IDS tools. Section 2 of this report contains the details of the literature associated with the detection of intrusion using open-source IDS tools and the existing studies like the one being performed here. Section 3 of this report contains details about the different methods used in the study. Section 4 contains the details about the implementation of the IDS and the analysis of the logs generated by the Suricata. Section 5 contains a discussion of the results of the study. Section 6 contains details like the methods used in the study, the results and findings from the study and the enhancements that can be made to the study in the future.

2 Literature review

The existing studies that propose a system for detecting intrusions were studied and this section is a review of the different literature studied. The literature studied included IDS based on machine learning and open-source intrusion detection tools.

2.1 IDS based on machine learning

An Experimental Cyber Attack Detection Framework (ECAD) is proposed in the study by (Mironeanu et al., 2021). Analytical tools and machine learning techniques are used in the study to detect cyber-attacks in real-time. The data used in the study was obtained from system and network traffic logs. The low-level information in the data associated with networks was extracted using multiple reconnaissance agents. The model proposed in the study has a module that detects connections that are potentially suspicious. This study shows that the logs associated with networks can be used for understanding and studying more about cyber-attacks. However, in this study for the real-time detection of cyber-attacks, the model proposed here uses offline data that was previously recorded and due to this, the cyber-attack detection model proposed here may not be effective. The data obtained from networks is used to build a cyber-attack detection system in the study by (Delplace et al., 2019). Machine learning algorithms are used to detect malicious traffic in the study. The machine learning algorithms used in the study are Logistic Regression (LR), Support Vector Machine (SVM), Gradient Boosting (GB), Random Forest (RF) and Extreme Learning Machine (ELM). The data used in the study is the NetFlow dataset and it contains the data associated with network traffic. The best network features from the dataset were extracted from the data using a feature selection method and 22 features were obtained. The RF model showed the best performance as it achieved an accuracy of 95%. However, the dataset used in the study contained only Botnets, so the model proposed here detects different Botnets only. Both studies use machine learning algorithms for the detection of cyber-attacks. The study by Mironeanu et al. (2021) used an analytics tool which was not done in the study by Delplace et al. (2019). These studies show that the network features and logs can be used to analyze the data associated with the network.
The machine learning models can detect intrusions in networks, but these are anomaly-based systems and do not scan or analyze the packets of data transferred over the network. The network data is not captured and stored by the IDSs based on machine learning. So, in the next section, the signature based IDSs that analyze the packets of data by capturing and storing the data are studied.

2.2 Signature based anomaly detection tools

An open-source IDS named Snort was used for the detection of intrusions in the study by (Rafa et al., 2022). The data used in the study contained patterns associated with network traffic. The Snort analyzed the data packets sent from the source to a destination in a Wide Area Network (WAN). The results of the study showed that the signature-based method, like the Snort, is effective in detecting cyber-attacks in networks. This study shows that a signature-based tool can be used to effectively detect cyber-attacks in a network. However, this study was done focusing only on a cloud

environment and no other kind of networks were considered. The Poisson distribution model was used along with Snort to detect DDoS attacks in networks in the study by (Chen and Lai, 2023). The data used in the study were samples of DDoS attacks which were simulated using four virtual machines on the CDX 3.0 platform. The network was analyzed using tools like PRTG and Wireshark. The study showed that different kinds of attacks had different impacts on a network. The study found that ICMP flood attacks affected the memory utilization of the system affected by it while the TCP SYN flood attack affected the performance time and CPU utilization. However, the model proposed in the study has an issue and that is the overreaction of the model to a sudden change in network traffic. Another limitation of the study is that only detected DDoS attacks, and no other kinds of attacks were considered in this study. Both studies considered here used the Snort for intrusion detection in a network. The study by Chen and Lai (2023) used additional methods like the Poisson distribution model, to detect of intrusions in a network. Both studies show that a signature-based IDS tool is effective in detecting intrusions in network traffic.
Wireshark was used for detecting different kinds of cyber-attacks in LANs in the study by (Iqbal and Naaz, 2019). In the study, the rules of the Wireshark tool were configured to detect network intrusions. The remote packet capture or Switched Port Analyzer (SPAN) was used for the analysis of data flow in the study. The results showed that the Wireshark tool is effective in detecting attacks that are commonly seen like DHCP spoofing, DOS attacks, MAC flooding and ARP poisoning. Mitigation techniques for each of the attacks are also defined in the study. The study showed that the rules of an IDS tool can be configured for detecting different kinds of attacks. However, this study only focused on the use of Wireshark in a LAN and no other networks were considered in the study. The packets in the network were analysed using Wireshark to detect different attacks in the study by (Dodiya and Singh, 2022). In the study, the behaviour of a network was studied by analysing the packets of data captured by the Wireshark tool. The malicious activity was detected using Indicators of Compromise (IOCs). The study showed that the Wireshark tool was able to collect data associated with parameters in IOC like MAC addresses of hosts, hostnames, domain names, host IP addresses and hashes. The study showed the effectiveness of Wireshark and its ability to analyse the traffic transferred in a network at a microscopic level. This study showed that the important parameters that can be used to find malicious network traffic in a network can be obtained when an IDS tool is used. However, the rules of the Wireshark were not configured in this study to detect intrusions. Both of these studies showed the effectiveness of Wireshark while Iqbal and Naaz(2019) configured the Wireshark for detecting cyber-attacks in the study by Dodiya and Singh(2022) only used the Wireshark for analysis of the packets of data sent over networks.

Bro was used for generating metadata to detect intrusions and analyze the SMB protocol in the study by (Cyrus, 2016). Data for the research was produced in a controlled testing environment designed to mimic a corporate network. This setup included virtual machines running Security Onion, Windows 7 and Windows Server 2012. Traffic was monitored through port mirroring. Tailored Bro scripts were developed to suit the network environment, utilizing string pattern or event-based indicators to detect potentially suspicious SMB activity. The findings of the study illustrated Bro's effectiveness in promptly detecting malicious activity in real-time, along with its capability to retrospectively analyze packet captures. The study shows that writing the rules of a rules-based IDS can help in detecting intrusions in the network. However, false positives may be generated by the Bro due to the detection of legitimate administrative tasks as intrusions.

The studies discussed here show that the signature-based IDSs are highly effective and the data analysed and captured in the network by these IDSs are stored as logs. As the signature-based IDSs were found to be effective in detecting intrusions, the signature-based IDS- Suricata is discussed in the next section.

2.3 Intrusion detection using Suricata

An IDS was implemented to analyse the network and detect intrusions in the network of Small Medium Enterprise (SMEs) in the study by (Veerasingam et al., 2023). The IDS proposed in the study was implemented in a Raspberry Pi 2B platform in the network of the SME using Suricata. A web application was developed in this study and it notified people working in the SME of an intrusion in their network detected using the Suricata. The notifications were generated based on the data collected from the logs of Suricata. The results of the study showed that the Suricata is effective in detecting intrusions in networks based on the rules that are configured in the Suricata. However, this study only focused on the detection of intrusion in the SME network. The alert detection performance of Suricata is analysed in the study by (Raharjo and Salman, 2023). This study had a two-phase methodology and it included a designing phase and testing phase. The design phase of the study included the designing of the testing environment, defining performance parameters and designing the generation of the dataset used in the study. In the testing phase, the designs were implemented. The dataset that was used in the study was created using Pcap datasets that contain IoC parameters. Five different datasets were created in this study to represent the five types of IoCs. The local IoC rules were extracted from the Pcap datasets in this study and these rules were tested in different situations. The results of the study showed that the accuracy of the Suricata in detecting intrusions decreased when the number of rules increased. This study showed how the rules in the Suricata can be configured to get an effective intrusion detection performance from the Suricata. However, in this study, the testing was carried out on a medium-range device, so the performance of the Suricata on a high-range device cannot be studied. The studies by Veerasingam et al., (2023) and Raharjo and Salman(2023) both used the Suricata however in the study by Raharjo and Salman(2023) the intrusion detection performance of Suricata was tested while the study by Veerasingam et al., (2023) used the Suricata to detect the intrusions in a SME network. Both studies showed that the Suricata is effective in detecting intrusions and generating alerts if the rules of the Suricata are configured properly.

The Suricata is effective in detecting intrusions and generating alerts when intrusions are detected. The Suricata works based on the rules defined in it. The Suricata also stored the packets of data captured by it in the network.

2.4 Summary

Study	Methods	Results	Limitations
Mironeanu et al., (2021)	The study proposed the Experimental Cyber Attack Detection Framework (ECAD), utilizing analytical tools and machine learning techniques to detect	The proposed model successfully identified potentially suspicious connections, demonstrating that network logs can be instrumental in	The study's reliance on previously recorded offline data for real-time detection limits the model's effectiveness in live cyber-attack scenarios.

	cyber-attacks in real-time. They analyzed system and network traffic logs, extracting low-level network information using multiple reconnaissance agents.	understanding cyber-attacks.	
Delplace et al., (2019)	The study developed a cyber-attack detection system using machine learning algorithms, including Logistic Regression, SVM, Gradient Boosting, Random Forest, and Extreme Learning Machine. They utilized the NetFlow dataset, applying feature selection to identify 22 key features for training the models.	The Random Forest model achieved the highest accuracy of 95% in detecting malicious traffic.	The dataset used contained only Botnets, limiting the model's detection capabilities to various Botnet attacks exclusively.
Rafa et al., (2022)	The study utilized the open-source IDS Snort for intrusion detection, focusing on patterns in network traffic data. The study specifically analysed data packets exchanged across a Wide Area Network (WAN) within a cloud environment.	The study demonstrated the effectiveness of Snort's signature-based approach in detecting cyber-attacks within the cloud network environment.	The research was limited to examining cyber-attacks exclusively in cloud environments, without consideration for other network types or environments.
Chen and Lai, (2023)	The study employed the Poisson distribution model in conjunction with Snort for detecting DDoS attacks. They simulated DDoS	The study identified varied impacts of different DDoS attacks on network performance metrics. For instance, ICMP flood attacks	The model exhibited overreaction to sudden changes in network traffic, potentially leading to false positives. Additionally, the

	attacks using virtual machines on the CDX 3.0 platform and analyzed network data using PRTG and Wireshark.	affected system memory utilization, while TCP SYN flood attacks impacted CPU performance and response times.	study focused exclusively on DDoS attacks, omitting analysis of other types of cyber-attacks.
Iqbal and Naaz, (2019)	The study utilized Wireshark for detecting various cyber-attacks within Local Area Networks (LANs). They configured Wireshark rules to identify network intrusions and used remote packet capture or Switched Port Analyzer (SPAN) for data flow analysis.	The study demonstrated Wireshark's effectiveness in detecting common attacks such as DHCP spoofing, Denial of Service (DOS), MAC flooding, and ARP poisoning. Mitigation techniques for each attack type were also outlined.	The research was confined to the use of Wireshark within LAN environments, with no exploration of its effectiveness in other network types.
Dodiya and Singh, (2022)	The study analysed network packets using Wireshark to detect malicious activity. They focused on capturing and studying network behaviour through packet analysis, leveraging Indicators of Compromise (IOCs) to identify suspicious patterns.	The study illustrated Wireshark's capability to gather data related to IOCs such as MAC addresses, hostnames, domain names, host IP addresses, and hashes. It highlighted Wireshark's effectiveness in providing detailed insights into network traffic at a granular level.	The study did not configure Wireshark rules specifically for intrusion detection purposes, potentially limiting its ability to proactively detect intrusions based on predefined criteria.
Cyrus, (2016)	The study employed Bro for generating metadata and analysing the Server Message Block (SMB) protocol to detect intrusions. The study utilized a	The research demonstrated Bro's effectiveness in real-time detection of malicious activities and its ability to retrospectively analyse packet captures. The study	One limitation identified was the potential for false positives generated by Bro, particularly due to its detection of legitimate administrative tasks as intrusions. This could lead to

	controlled testing environment mimicking a corporate network, with virtual machines running Security Onion, Windows 7, and Windows Server 2012. Traffic monitoring was conducted through port mirroring, and customized Bro scripts were developed to detect suspicious SMB activity using string pattern or event-based indicators.	highlighted Bro's capability to promptly detect intrusions while providing insights into network traffic through metadata generation.	operational challenges in distinguishing between malicious activities and normal network operations.
Veerasingam et al., (2023)	The study implemented an Intrusion Detection System (IDS) using Suricata to analyse and detect intrusions within the network of Small Medium Enterprises (SMEs). The study utilized a Raspberry Pi 2B platform to host Suricata, and a web application was developed to notify SME personnel of detected intrusions based on Suricata's log data.	The study demonstrated Suricata's effectiveness in detecting intrusions based on configured rules within SME networks. Notifications were generated promptly upon detection, leveraging Suricata's log data to inform SME personnel of potential security breaches.	One limitation of the study was its focus solely on detecting intrusions within SME networks. The applicability of Suricata's performance in larger enterprise environments or under different network conditions was not explored.
Raharjo and Salman, (2023)	The study analysed the alert detection performance of Suricata using a two-phase methodology. The study included a design phase where the testing	The research found that as the number of rules increased, the accuracy of Suricata in detecting intrusions decreased. Five different datasets representing	A limitation of the study was the use of a medium-range device for testing, which restricted the exploration of Suricata's performance on high-end devices.

	environment was established, performance parameters were defined, and datasets were generated using Pcap datasets containing Indicators of Compromise (IoC) parameters. In the testing phase, the effectiveness of local IoC rules extracted from the Pcap datasets was evaluated under various conditions.	various types of IoCs were used to test Suricata's performance, highlighting the impact of rule complexity on detection accuracy.	This limitation may affect the generalizability of the findings to different hardware configurations or larger-scale network environments.

Table (1):Summary of literature review findings

The literature studied here showed that the data captured and stored in the logs by the Suricata is not analysed or studied, so in the study that is done here the data in the logs of the Suricata is analysed for determining if any insight associated with the intrusions in the network can be gained from the logs. The study that is performed here proposes the detection of intrusions using Suricata.

3 Methodology

This research project introduces a real-time intrusion detection system utilizing the Suricata tool in a controlled VirtualBox environment, complemented by cloud-based storage integration. It aims to detect and respond to various network attacks simulated between the Ubuntu (victim) and Kali Linux (attacker) machines. By monitoring network traffic and applying Suricata's detection rules, the system identifies threats and securely logs them in Google Drive. The project emphasizes proactively enhancing network security measures through effective intrusion detection and reliable cloud-based log storage, enhancing accessibility and response capabilities against cyber threats.

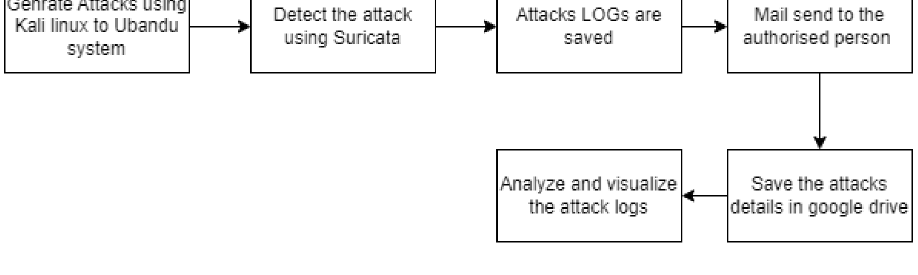

Genrate Attacks using Kali linux to Ubandu system
Detect the attack using Suricata
Attacks LOGs are saved
Mail send to the authorised person
Save the attacks details in google drive
Analyze and visualize the attack logs

Figure (2): Workflow of network security system from attack generation to log analysis

Figure (2) shows the block diagram of the research system, illustrating its workflow from attack generation using Kali Linux to detection with Suricata, saving logs, sending email alerts to administrators, storing attack details on Google Drive, and ultimately analysing and visualizing attack logs. This diagram provides a clear overview of how the system operates in detecting and responding to network security threats.

3.1 Setting up the virtual machines

A network simulation was necessary to enable Suricata to analyse network activity, generate logs, and detect intrusions effectively. In this study, the simulation involved setting up two virtual machines on a single PC using VirtualBox, one running Ubuntu as the victim machine and the other running Kali Linux as the attacker. Ubuntu, known for its user-friendly interface and extensive community support, is widely adopted in both personal and professional settings due to its robust security architecture and frequent updates, ensuring resilience against vulnerabilities. Kali Linux, based on Debian, specializes in digital forensics and penetration testing, equipped with a comprehensive suite of tools for network analysis and vulnerability assessment. Despite its strengths, challenges such as resource consumption (RAM and processing power) and time-consuming setup for running two operating systems concurrently on a single machine were encountered during the implementation. Next, the Ubuntu system was prepared to ensure it was updated and ready for new software installation. This step is crucial in ensuring that all software dependencies required by Suricata are up to date, thereby preventing potential conflicts or issues during the installation process. The first step in this process was to update the package lists and upgrade existing packages. This was done using the commands ‘sudo apt update’ and ‘sudo apt upgrade -y’. Executing ‘sudo apt update’ ensures that the system retrieves information about their dependencies and the latest versions of packages, while ‘sudo apt upgrade -y’ upgrades all existing packages to their latest versions without user intervention.

3.2 Suricata Installation

Suricata is a free and open-source signature-based intrusion detection system, that offers a quick and resilient engine for detecting network threats (Bada, Nabare and Quansah, 2020). In the study by Kumar et al. (2023), Suricata was compared with other intrusion detection tools like Snort, Bro and OSSEC. The results of the study by Kumar et al. (2023), found that Suricata had better performance in the detection of intrusions as it showed low false negative and false positive rates, high rate of detection and accuracy. So the Suricata has a better performance in detecting intrusions in networks, than Bro and OSSEC. Following the system preparation phase, the installation of Suricata commenced to bolster the system's security capabilities. This involved adding the Open Information Security Foundation (OISF) PPA repository, renowned for hosting stable Suricata releases, and updating the package list to incorporate this repository. The installation process proceeded with careful consideration of system dependencies and potential conflicts, aiming to ensure seamless integration.

3.3 Network Interface Configuration Setup

Configuring the network interface for Suricata involves a series of steps aimed at ensuring it can effectively monitor and analyse network traffic. It begins with identifying the correct network interface that Suricata will monitor. This involves using commands like 'ip a' to list available interfaces and select the appropriate one, such as 'eth0' or 'ens33'. Once the interface is identified, the next step is to configure Suricata's settings to optimize its performance. This includes editing the Suricata configuration file located at '/etc/suricata/suricata.yaml'. In this configuration file, parameters under the 'af-packet' section are adjusted to match the specifics of the network interface. These adjustments are crucial as they dictate how Suricata interacts with network packets, for example, setting the number of threads to handle packet processing efficiently. Additionally, settings related to packet defragmentation and cluster IDs are configured to ensure Suricata can handle various types of network traffic effectively. One of the challenges encountered during this process is ensuring that the configuration accurately reflects the network setup. Mistakes in configuring the network interface or mismatched settings can lead to Suricata either missing important network events or generating false alerts, which can impact the overall effectiveness of the intrusion detection system. The ultimate goal of configuring the network interface for Suricata is to align its intrusion detection capabilities with the specific characteristics of the network environment. By fine-tuning parameters like packet handling and monitoring settings, Suricata can effectively monitor for suspicious activities such as port scans, denial-of-service attacks, or unauthorized access attempts. This proactive approach not only enhances the security posture of the network but also ensures that potential threats are identified and mitigated in a timely manner. Configuring the network interface for Suricata is a critical aspect of setting up an effective intrusion detection system. It involves careful selection and configuration of the network interface, adjustment of performance parameters, and consideration of resource constraints. This process ensures that Suricata operates optimally within the network environment, providing robust monitoring and detection capabilities against evolving cyber threats.

3.4 Setting up rules

Configuring Suricata involves crucial steps to ensure effective network monitoring and threat detection. Starting with updating its rule set, this process ensures Suricata remains equipped to identify evolving threats. However, managing Suricata's service can be challenging due to resource constraints, such as high RAM usage and processing demands, especially when enabling automatic initiation on system boot. Verification, includes checking Suricata's operational status and monitoring real-time logs to confirm its functionality in capturing and analysing network traffic, which can be time-consuming. Testing involves generating specific network activities to trigger custom rules, validating Suricata's responsiveness in alerting to potential security breaches. Adding Custom rules to Suricata's configuration enhances its ability to detect ICMP Echo Requests, TCP port scans, and SSH authentication attempts, tailored to specific network threats for proactive security monitoring and response.

3.5 Email Sending and Uploading to Google Drive

In response to detecting malicious attacks, an automated workflow initiates the prompt dispatch of email notifications and uploads of log files to Google Drive, facilitated by Python libraries like os, smtplib, email.mime, and watchdog. This system relies on a meticulously crafted Python script that monitors the Suricata log directory for new files, signalling potential security breaches. Upon detection, the script triggers email alerts, containing detailed breach information and actionable guidance, ensuring swift communication with relevant stakeholders. The use of smtplib ensures secure email transmission, authenticating sender credentials for reliable delivery. Meanwhile, the watchdog library's observer function maintains continuous folder surveillance, enabling real-time incident detection to swiftly inform security administrators. This proactive approach empowers timely response to security threats, bolstering network defences and pre-emptively mitigating risks. The email notification mechanism serves a crucial role in promptly notifying designated recipients about detected security breaches, enabling swift response actions to mitigate potential risks. However, a challenge encountered during implementation involved managing dependencies and compatibility issues among Python libraries, which required careful configuration to ensure seamless functionality across different environments. Additionally, setting up and configuring Google Drive API credentials for file uploads posed initial complexities, necessitating a clear understanding of developer options and proper credential management. These challenges underscored the importance of meticulous setup and testing to ensure the reliable and effective operation of the automated security protocol.

4 Design Specification

4.1 Suricata

Suricata is an open-source Intrusion Detection and Prevention System (IDPS) renowned for its robust capabilities in monitoring and protecting computer networks against cyber threats. Developed by the Open Information Security Foundation (OISF), Suricata operates by analyzing network traffic in real-time, detecting suspicious patterns, and alerting administrators to potential security breaches. Snort is similar to Suricata and it generates an alert when intrusions are detected in a network. In the study by Sharma et al. (2021), it was found that the Snort failed to generate alerts for two kinds of cyber-attacks that were defined in its rules but the Suricata successfully generated alerts for all the cyber-attacks or intrusions that were defined in its rules. It was specified in the study by Sharma et al. (2021) that Suricata was more computationally expensive than Snort but since the main aim of the study is to generate alerts properly according to the rules defined for Suricata was a better choice for the study that was carried out here. The study used computers that were high end and had high computational power so these computer systems could be used to handle the installation and working of Suricata effectively. Suricata was also found to be more versatile than Snort in the study by (Sharma et al. 2021).

5 Implementation

The Real-Time Intrusion Detection System using Suricata is designed to strengthen network security by leveraging advanced monitoring and cloud-based log storage capabilities. Implemented within a controlled VirtualBox environment, the project focuses on detecting diverse network attacks originating from an attacker machine (Kali Linux) directed towards a victim machine (Ubuntu). The implementation begins with setting up VirtualBox to create a secure testing environment conducive to simulating network scenarios. This environment hosts Ubuntu as the victim machine and Kali Linux as the attacker, each configured with specific network settings to facilitate controlled attack simulations. Suricata, a powerful Intrusion Detection and Prevention System, is then deployed on the Ubuntu victim machine. Installation involves downloading and configuring Suricata to optimize its performance in detecting various types of threats. This includes setting up essential configuration files and directories, crucial for defining rules that govern the detection of specific attack patterns, once configured, Suricata actively monitors network traffic in real-time, analyzing packets to identify suspicious activities based on predefined rules. Concurrently, attacks are generated from Kali Linux, simulating scenarios like ICMP Echo Requests, TCP port scans, and SSH login attempts. Suricata's ability to detect these attacks is critical in demonstrating effective proactive security measures. To enhance accessibility and reliability, the study integrates cloud-based storage via Google Drive for storing Suricata's logs securely.

This setup ensures that log files, containing crucial information about detected threats, are readily accessible and protected against unauthorized access or tampering. The implementation includes automated email notifications to alert administrators in real-time upon detecting potential security breaches. This proactive alert system utilizes Python scripts to monitor Suricata's log directory continuously. Upon detecting new logs indicative of an attack, the system triggers email alerts to designated recipients, providing actionable insights to mitigate risks promptly. The implementation includes automated email notifications to alert administrators in real-time upon detecting potential security breaches. This proactive alert system utilizes Python scripts to monitor Suricata's log directory continuously. Upon detecting new logs indicative of an attack, the system triggers email alerts to designated recipients, providing actionable insights to mitigate risks promptly. In addition to real-time alerting, the project focuses on analyzing incidents afterward using Python scripts. These scripts facilitate detailed analysis of attack patterns and trends, offering insights that aid in refining Suricata's detection capabilities and strengthening overall network defenses. The Real-Time Intrusion Detection System integrates VirtualBox, Suricata, cloud-based storage, and automated alerting mechanisms to create a comprehensive defense strategy against cyber threats. By implementing these steps from environment setup to attack simulation, detection, storage, and analysis the project aims to showcase effective network security practices, bolstered by cloud technology's scalability and reliability in safeguarding critical network infrastructures.

6 Results and Evaluation

6.1 Results

Results of EDA

The results of the EDA are shown in the figures presented here.

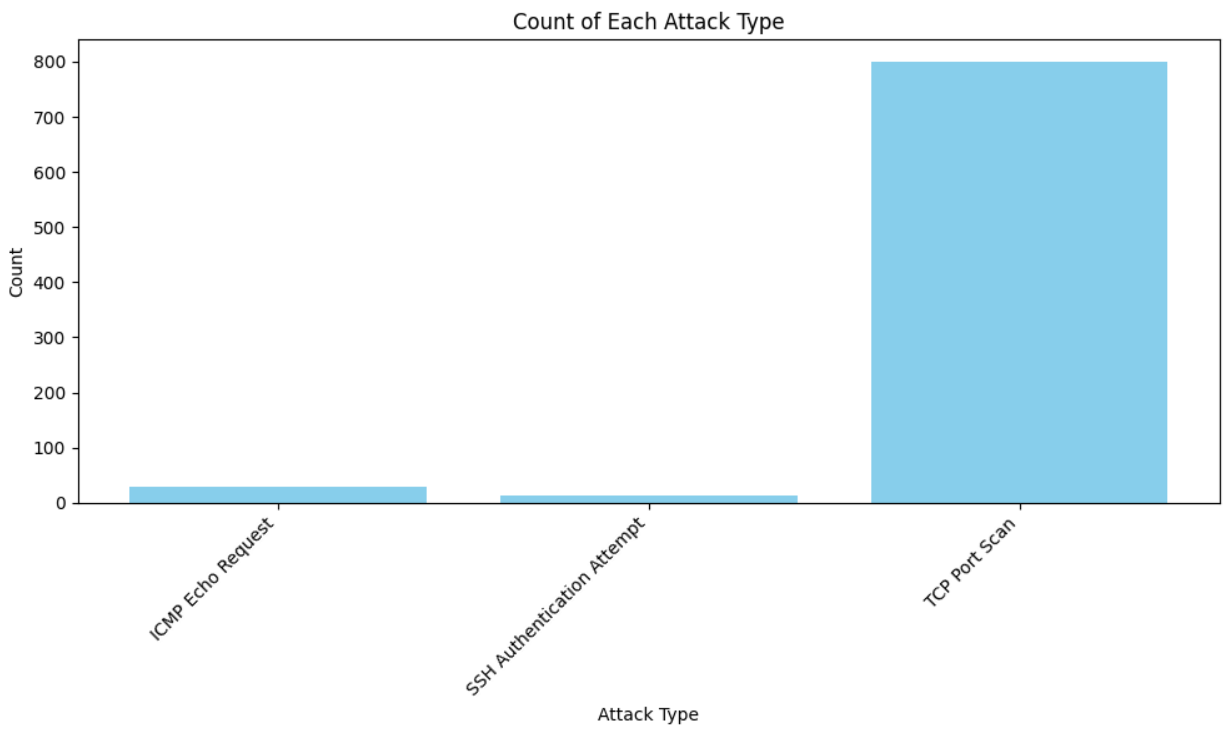


Figure (3): Count of each attack type

A bar plot to visualize the frequency of various attack types detected in network logs was generated, as shown in the figure(3). The process begins by preparing two lists: one for the attack types and another for their corresponding counts. A bar chart is created, with the attack types along the x-axis and their counts on the y-axis. Labels for the x-axis and y-axis are added, indicating 'Attack Type' and 'Count' respectively. This visualization helps in understanding the distribution and frequency of different attack types, making it easier to identify prevalent threats and prioritize security measures accordingly.

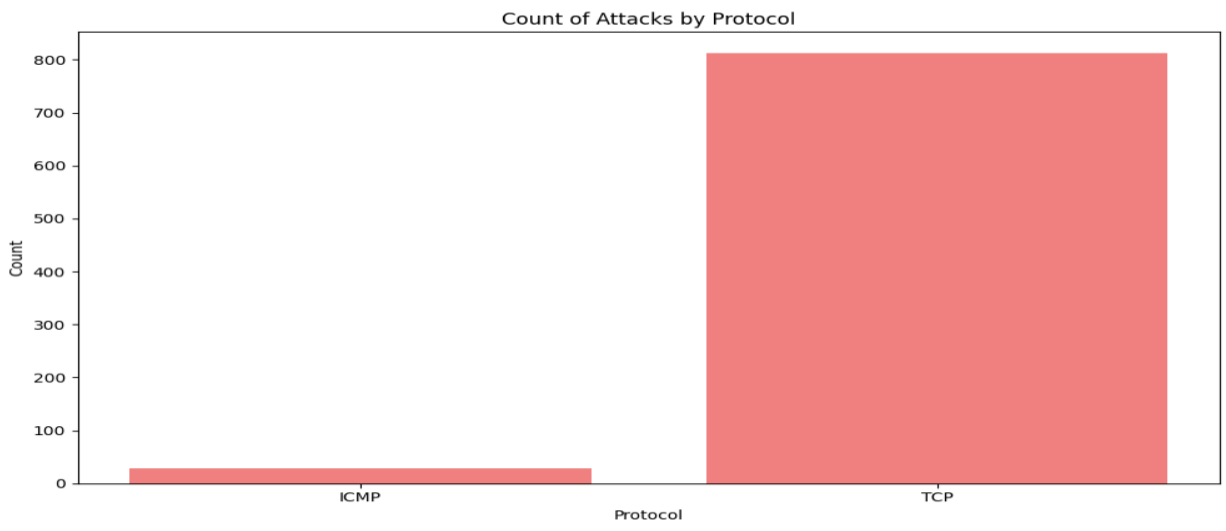
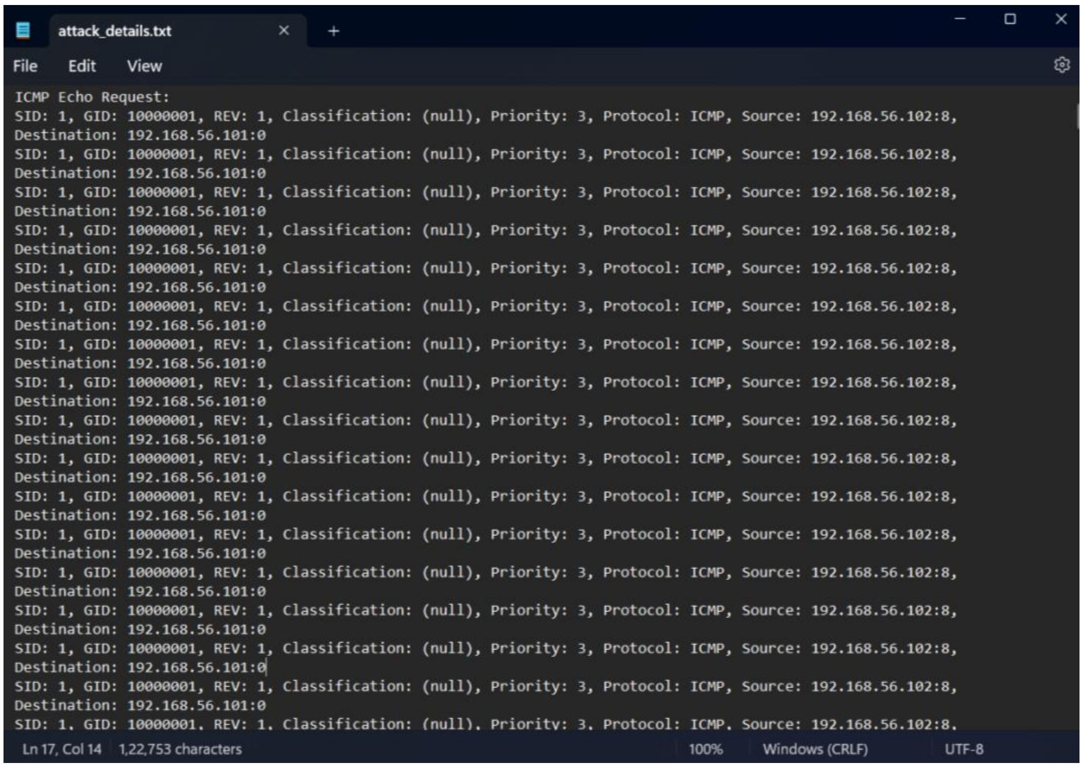


Figure (4): Count of attacks by protocol

A bar plot is shown in figure(4) to visualize the number of attacks categorized by protocol was generated. The process starts by initializing a data structure to count occurrences of each protocol. It then iterates through the provided log data, incrementing the count for the corresponding protocol each time it is encountered. After processing the data, two lists are prepared: one containing the different protocols and another containing their respective counts. A bar chart is then created, with protocols along the x-axis and their counts on the y-axis. The bars are coloured light coral to make the plot visually appealing. Labels for the x-axis and y-axis are added, indicating 'Protocol' and 'Count' respectively. This visualization helps in understanding the distribution of attacks across different protocols, making it easier to identify which protocols are most frequently targeted and thus prioritize security measures accordingly.



```
ICMP Echo Request:
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
Destination: 192.168.56.101:0
SID: 1, GID: 10000001, REV: 1, Classification: (null), Priority: 3, Protocol: ICMP, Source: 192.168.56.102:8,
```

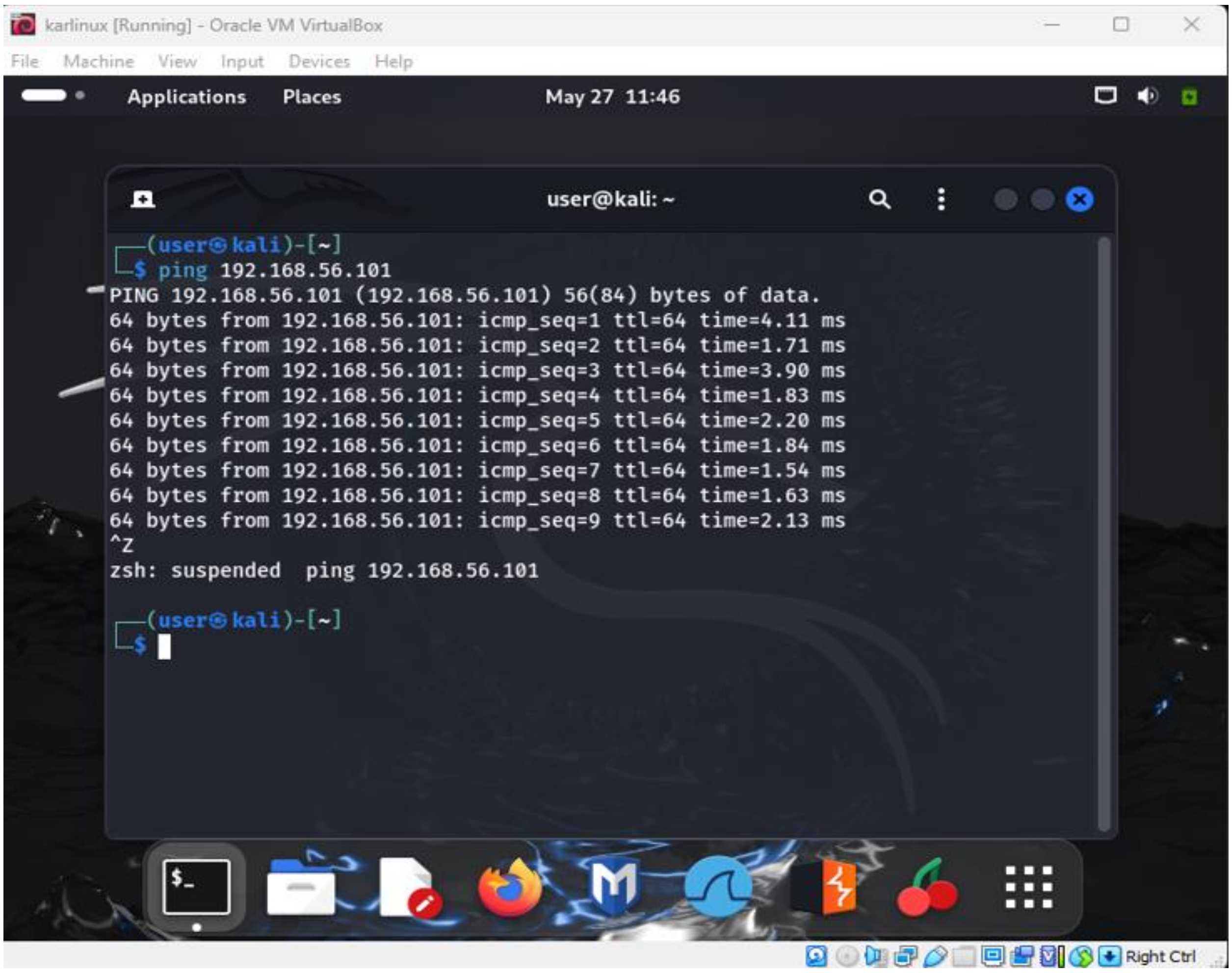
Figure (5): Attack details

The log in the figure(5) shows the entries depicting a series of ICMP Echo Request events, each identified by a unique SID (Signature ID) of 1, belonging to the Group ID (GID) 10000001, and having a revision (REV) of 1. These requests originate from the source IP address 192.168.56.102 and are directed towards the destination IP address 192.168.56.101. The classification field is marked as "null", indicating an unspecified classification for these requests. With a priority level of 3, these ICMP Echo Requests denote a relatively moderate significance level in terms of network traffic. The protocol used is ICMP (Internet Control Message Protocol), and it is utilized for control and diagnostic purposes in network communications. These repeated requests suggest a potential pattern of communication or testing activity between the source and destination IP addresses.

Attack Detection

A network connectivity test from a Kali Linux system through the utilization of the ping command was performed. The command prompt initiates a ping command targeting the IP address 192.168.56.101, a network utility used to ascertain the reachability of a host on a network. Subsequently, the output confirms pings that are successful to the specified host, round trip times are displayed for each ping sent. The ping command was then suspended for unspecified reasons.

The text "user@kali: -" presents user-related information, including the username "user" logged into the host named "kali", a machine running the Kali Linux distribution renowned for security and penetration testing. Following this, a command that is partially typed "ping 192.168.56.101" shows an attempt to make use of the "ping" network utility tool to evaluate the reachability of a device using the IP address that is specified. Subsequent lines display output from the ping command, affirming successful communication with the target device. However, the last line, "zsh: suspended ping 192.168.56.101", suggests an interruption or suspension of the ping command execution. This comprehensive analysis highlights the user's engagement with network testing activities on a Kali Linux system through the terminal interface.

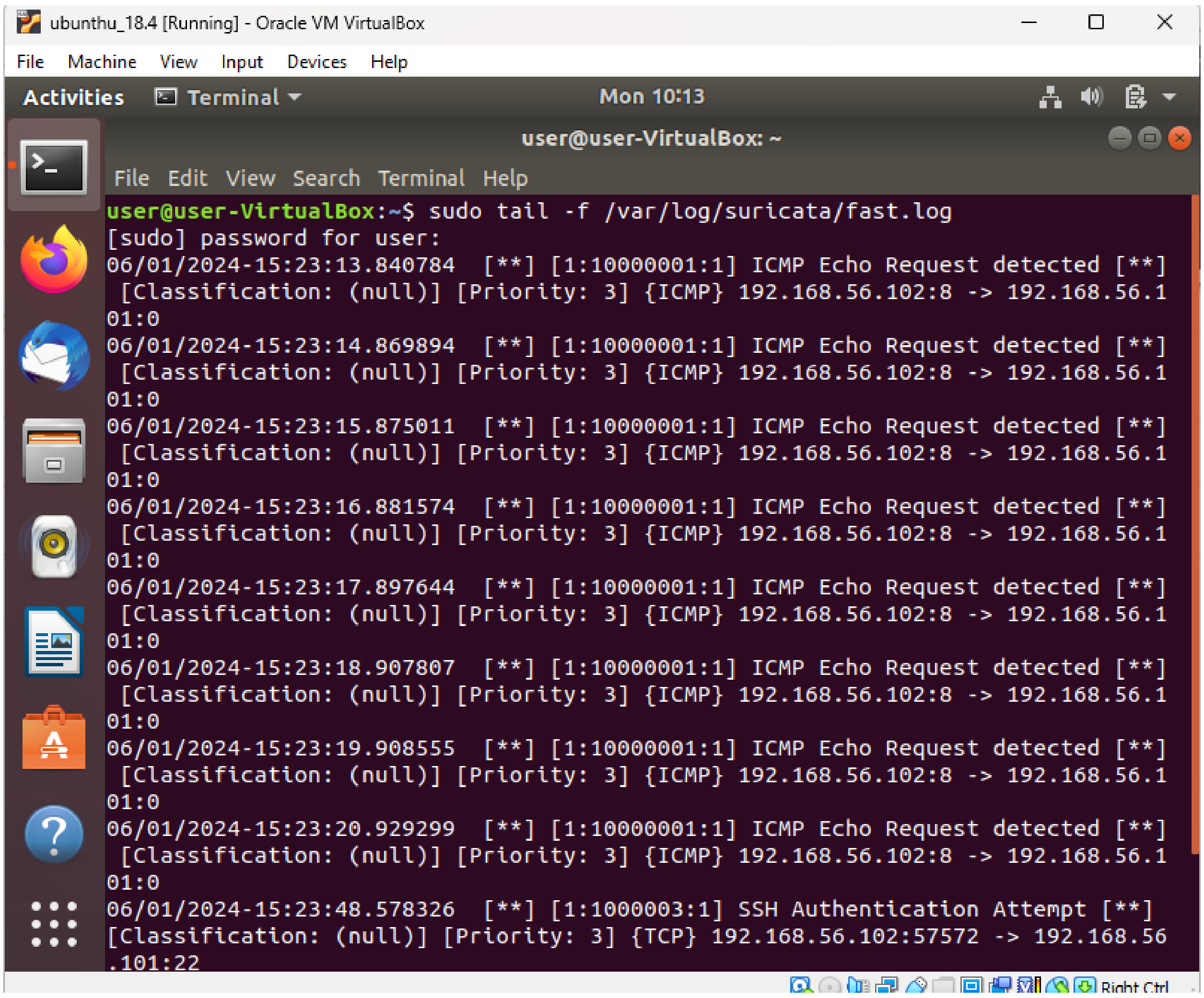

karlinux [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places
May 27 11:46
user@kali: ~
Right Ctrl

```
┌──(user㉿kali)-[~]
└─$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=4.11 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=1.71 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=3.90 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=64 time=1.83 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=64 time=2.20 ms
64 bytes from 192.168.56.101: icmp_seq=6 ttl=64 time=1.84 ms
64 bytes from 192.168.56.101: icmp_seq=7 ttl=64 time=1.54 ms
64 bytes from 192.168.56.101: icmp_seq=8 ttl=64 time=1.63 ms
64 bytes from 192.168.56.101: icmp_seq=9 ttl=64 time=2.13 ms
^Z
zsh: suspended  ping 192.168.56.101

┌──(user㉿kali)-[~]
└─$ 
```

Figure (6): attack generation

The text "ping 192.168.56.101" suggests the user's initiation of the "ping" command, a network utility tool utilized to ascertain the reachability of a host on a network, with the specified IP address serving as the target. Displayed lines exhibit output from the ping command, confirming its success and receipt of responses from the device with the IP address 192.168.56.101. Additionally, the round trip time for each ping is likely provided. The final line "zsh: suspended ping 192.168.56.101" indicates the suspension of the ping command for unspecified reasons, with "zsh" denoting the utilized shell as shown in the figure(6).


ubunthu_18.4 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Mon 10:13
user@user-VirtualBox: ~
File Edit View Search Terminal Help
Right Ctrl

```
user@user-VirtualBox:~$ sudo tail -f /var/log/suricata/fast.log
[sudo] password for user:
06/01/2024-15:23:13.840784  [**] [1:10000001:1] ICMP Echo Request detected [**]
 [Classification: (null)] [Priority: 3] {ICMP} 192.168.56.102:8 -> 192.168.56.1
01:0
06/01/2024-15:23:14.869894  [**] [1:10000001:1] ICMP Echo Request detected [**]
 [Classification: (null)] [Priority: 3] {ICMP} 192.168.56.102:8 -> 192.168.56.1
01:0
06/01/2024-15:23:15.875011  [**] [1:10000001:1] ICMP Echo Request detected [**]
 [Classification: (null)] [Priority: 3] {ICMP} 192.168.56.102:8 -> 192.168.56.1
01:0
06/01/2024-15:23:16.881574  [**] [1:10000001:1] ICMP Echo Request detected [**]
 [Classification: (null)] [Priority: 3] {ICMP} 192.168.56.102:8 -> 192.168.56.1
01:0
06/01/2024-15:23:17.897644  [**] [1:10000001:1] ICMP Echo Request detected [**]
 [Classification: (null)] [Priority: 3] {ICMP} 192.168.56.102:8 -> 192.168.56.1
01:0
06/01/2024-15:23:18.907807  [**] [1:10000001:1] ICMP Echo Request detected [**]
 [Classification: (null)] [Priority: 3] {ICMP} 192.168.56.102:8 -> 192.168.56.1
01:0
06/01/2024-15:23:19.908555  [**] [1:10000001:1] ICMP Echo Request detected [**]
 [Classification: (null)] [Priority: 3] {ICMP} 192.168.56.102:8 -> 192.168.56.1
01:0
06/01/2024-15:23:20.929299  [**] [1:10000001:1] ICMP Echo Request detected [**]
 [Classification: (null)] [Priority: 3] {ICMP} 192.168.56.102:8 -> 192.168.56.1
01:0
06/01/2024-15:23:48.578326  [**] [1:1000003:1] SSH Authentication Attempt [**]
[Classification: (null)] [Priority: 3] {TCP} 192.168.56.102:57572 -> 192.168.56
.101:22
```

Figure (7): attack detection

A command, "sudo tail -f /var/log/suricata/fast.log", is used, where "sudo" grants superuser privileges and "tail -f /var/log/suricata/fast.log" continuously monitors a security-related log file, possibly associated with the Suricata security tool. This comprehensive analysis suggests a user interaction scenario within a virtualized Ubuntu 18.4 environment, characterized by active monitoring of a security log file via the terminal interface as shown in the figure(7).

6.2 Evaluation

The study was successfully completed and from the results of the study it was seen that the attacks were successfully detected by the Suricata based on the rules defined in the Suricata. Logs generated by the Suricata were saved in the Google Drive and analysed using Python. The system also sent notifications to users when an attack was detected by the Suricata, this mail sending was implemented using Python. The main objectives of this study were achieved.

The investigated study leveraged the powerful Suricata tool for the purpose of cyber-attack detection within network environments. The implemented Suricata-based system demonstrated remarkable efficacy in promptly identifying potential security threats and generating pertinent alerts. Leveraging predefined rules within Suricata, the system adeptly detected a spectrum of cyber-attacks based on these established parameters. Notably, the system's ability to detect attacks was contingent upon the existence of appropriate rules, underscoring the pivotal role of defined rules in Suricata's detection capabilities. In instances where specific attack types were not encompassed by defined rules, the system's ability to detect these threats would be compromised, emphasizing the critical nature of a comprehensive rule set in fortifying network security. It's important to note that while the system excelled in the detection of cyber-attacks, the study did not delve into the active prevention of these identified threats. By elaborating on the feasibility of integrating preventive measures following the system's successful detection of attacks within a network, the study could further enhance the depth and impact of its findings. Introducing the capability to prevent detected attacks would augment the system's operational value, fostering a more robust and proactive security posture. This proactive functionality would empower security stakeholders to effectively mitigate potential threats upon detection, thereby minimizing the impact and risk imposed by cyber-attacks on network infrastructure.

7 Conclusion and future enhancements

The study conducted focused on developing and evaluating a cyber-attack detection system leveraging Suricata, which proved successful in achieving its primary objectives. To commence, the robust system integrated Suricata, thereby enabling the detection of cyber-attacks, and subsequently, custom rules were meticulously defined to further fortify the system's defensive capabilities. This cultivated a highly effective cyber-attack detection mechanism, showcasing the system's adeptness in identifying and responding to potential security threats in real-time. Moreover, the study aptly incorporated the implementation of email-sending and log-saving functionalities, seamlessly accomplished through the integration of Python, highlighting the system's multifaceted operational proficiency. The successful transmission of email notifications upon attack detection, as well as the seamless archiving of logs in Google Drive, underscored the system's comprehensive approach to security threat management. Furthermore, the study comprehensively tested the system's resilience by simulating cyber-attacks using various commands, affirming Suricata's efficacy in detecting and responding to diverse forms of cyber threats. Notably, the study culminated in the production of detailed visual representations based on the logged data, shedding light on critical insights such as the type and distribution of attacks, attack frequencies by protocol and IP addresses, and comprehensive attack details. Through this comprehensive visualization, the study provided a deep understanding of the attack landscape, further fortifying the system's analytical capabilities and facilitating proactive security measures. Consequently, the study outlined an intricate and robust cyber-attack detection system facilitated by Suricata, underlining its efficacy and multifaceted functionality in fortifying network security. The research question proposed in the study was answered:

- **What insights can be gained by analyzing the data stored in the logs by Suricata?**

  This study showed that the data associated with the logs generated by Suricata can be analyzed to gain different insights about the type of attacks, protocols etc., taking place in a network detected by the Suricata.

The email sending implemented in the system, the storing of the logs generated by the Suricata, and analysis of the data associated with the logs generated by Suricata. This study helps in providing insights about how the data in the logs or the data collected by Suricata can be analyzed and how the analysis of the data helps in improving the security of a network. The study also showed ways in which functionalities can be integrated with IDSs. The integration of Python with IDSs can help in improving the functionalities of IDSs and results in the addition of several functionalities to an IDS like it has been done with the Suricata in this study.

In the future, a machine learning method can be integrated with the Suricata to make the system built here more robust as the machine learning models will help the Suricata in detecting cyber-attacks more effectively. Methods for preventing the cyber-attacks detected by the Suricata can be integrated into the system and the detected attacks can be prevented. More rules can be defined in the Suricata so that more kinds of cyber-attacks can be detected.

References

Abdallah, E.E., Eleisah, W. and Otoom, A.F. (2022). Intrusion Detection Systems using Supervised Machine Learning Techniques: A survey. Procedia Computer Science, 201, pp.205–212. doi:https://doi.org/10.1016/j.procs.2022.03.029.

Bada, G.K., Nabare, W.K. and Quansah, D.K.K. (2020) 'Comparative analysis of the performance of network intrusion Detection systems: SNORT, Suricata and BRO Intrusion Detection Systems in perspective,' International Journal of Computer Applications, 176(40), pp. 39–44. https://doi.org/10.5120/ijca2020920513.

Chen, C.-L. and Lai, J.L. (2023) 'An experimental detection of distributed denial of service attack in CDX 3 platform based on SNorT,' Sensors, 23(13), p. 6139. https://doi.org/10.3390/s23136139.

Chu, W. (2019) 'Application of data encryption technology in computer network Security,' Journal of Physics. Conference Series, 1237(2), p. 022049. https://doi.org/10.1088/1742-6596/1237/2/022049.

Chu, W. (2019) 'Application of data encryption technology in computer network Security,' Journal of Physics. Conference Series, 1237(2), p. 022049. https://doi.org/10.1088/1742-6596/1237/2/022049.

Cyrus, R. (2016) Detecting Malicious SMB Activity Using Bro. PhD. Dissertation. The SANS Institute. https://www.giac.org/paper/gcia/10091/detecting-malicious-smb-activity-bro/140938.

Delplace, A., Hermoso, S. and Anandita, K. (2020) 'Cyber Attack Detection thanks to Machine Learning Algorithms,' arXiv (Cornell University) [Preprint]. https://doi.org/10.48550/arxiv.2001.06309.

Díaz-Verdejo, J., Muñoz-Calle, J., Estepa Alonso, A., Estepa Alonso, R. and Madinabeitia, G. (2022). On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks. Applied Sciences, 12(2), p.852. doi:https://doi.org/10.3390/app12020852.

Dodiya, B. and Singh, U.K. (2022) 'Malicious Traffic analysis using Wireshark by collection of Indicators of Compromise,' International Journal of Computer Applications, 183(53), pp. 1–6. https://doi.org/10.5120/ijca2022921876.

Einy, S., Oz, C. and Navaei, Y.D. (2021). The Anomaly- and Signature-Based IDS for Network Security Using Hybrid Inference Systems. Mathematical Problems in Engineering, 2021, pp.1–10. doi:https://doi.org/10.1155/2021/6639714.

Gupta, R. et al. (2017) 'Intrusion detection system using SNORT,' International Research Journal of Engineering and Technology (IRJET), 4(4), pp. 2100–2104. https://www.irjet.net/archives/V4/i4/IRJET-V4I4439.pdf.

Hoover, C. (2022) 'Comparative Study of Snort 3 and Suricata Intrusion Detection Systems,' Computer Science and Computer Engineering Undergraduate Honors Theses. https://scholarworks.uark.edu/csceuht/105

Hu, X. et al. (2023) 'Toward early and accurate network intrusion detection using graph embedding,' IEEE Transactions on Information Forensics and Security, 18, pp. 5817–5831. https://doi.org/10.1109/tifs.2023.3318960.

IPS (Suricata) — NethServer 7 Final (2024). https://docs.nethserver.org/en/v7/suricata.html.

Iqbal, H. and Naaz, S. (2019) 'Wireshark as a tool for detection of various LAN attacks,' International Journal of Computer Sciences and Engineering, 7(5), pp. 833–837. https://doi.org/10.26438/ijcse/v7i5.833837.

Jain, G. and Anubha, N. (2021) 'Application of SNORT and Wireshark in network traffic analysis,' IOP Conference Series. Materials Science and Engineering, 1119(1), p. 012007. https://doi.org/10.1088/1757-899x/1119/1/012007.

Khraisat, A. et al. (2019) 'Survey of intrusion detection systems: techniques, datasets and challenges,' Cybersecurity, 2(1). https://doi.org/10.1186/s42400-019-0038-7.

Kumar, A. et al. (2023) 'A COMPARATIVE ANALYSIS OF DIFFERENT INTRUSION DETECTION SYSTEMS,' International Research Journal of Modernization in Engineering Technology and Science, 05–05(4), pp. 3029–3030. https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2023/36289/final/fin_irjmets1681923481.pdf.

Mironeanu, C. et al. (2021) 'Experimental Cyber Attack Detection Framework,' Electronics, 10(14), p. 1682. https://doi.org/10.3390/electronics10141682.

Othman, S.M. et al. (2018) 'Intrusion detection model using machine learning algorithm on Big Data environment,' Journal of Big Data, 5(1). https://doi.org/10.1186/s40537-018-0145-4.

Ouiazzane, S., Addou, M. and Barramou, F. (2022) 'A Suricata and machine learning based hybrid network intrusion detection system,' in Lecture notes in networks and systems, pp. 474–485. https://doi.org/10.1007/978-3-030-91738-8_43.

Paxson, V. et al. (2006) Bro Intrusion Detection System. https://www.osti.gov/biblio/1245188.

PyNet Labs (2023) Difference between IDS and IPS - PyNet Labs. https://www.pynetlabs.com/difference-between-ids-and-ips/.

Rafa, F. et al. (2022) Detecting Intrusion in Cloud using Snort: An Application towards Cyber-Security. https://doi.org/10.1145/3542954.3542984.

Raharjo, D.H.K. and Salman, N.M. (2023) 'ANALYZING SURICATA ALERT DETECTION PERFORMANCE ISSUES BASED ON ACTIVE INDICATOR OF COMPROMISE RULES,' Jurnal Teknik Informatika, 4(3), pp. 601–610. https://doi.org/10.52436/1.jutif.2023.4.3.1013.

Saive, R. and Saive, R. (2022) Suricata - Intrusion Detection and Prevention Security Tool. https://www.tecmint.com/suricata-intrusion-detection-prevention-linux/.

Sharma, N.V. et al. (2021) 'Performance study of SnOrT and Suricata for Intrusion Detection System,' IOP Conference Series. Materials Science and Engineering, 1099(1), p. 012009. https://doi.org/10.1088/1757-899x/1099/1/012009.

St. John, M. (2024). Cybersecurity Stats: Facts And Figures You Should Know. [online] Forbes Advisor. Available at: https://www.forbes.com/advisor/education/it-and-tech/cybersecurity-statistics/#:~:text=As%20the%20globe%20becomes%20more,%25%2C%20surpassing%20the%20previous%20record. [Accessed 17 Apr. 2024].

Veerasingam, P. et al. (2023) 'INTRUSION DETECTION AND PREVENTION SYSTEM IN SME'S LOCAL NETWORK BY USING SURICATA,' Malaysian Journal of Computing and Applied Mathematics, 6(1), pp. 21–30. https://doi.org/10.37231/myjcam.2023.6.1.88.