NETWORK INTRUSION DETECTION AND ALERT SYSTEM

BY

TO JIN YI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMMUNICATIONS AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

JUN 2024

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Network Intrusion Detection and Alert System

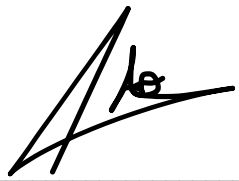
Academic Session: Y5/S3

I TO JIN YI

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.



(Author's signature)

Verified by,

(Supervisor's signature)

Address:
152, JALAN TERAP 4,
TAMAN SRI TERAP,
84000 MUAR, JOHOR

Dr. Gan Ming Lee
Supervisor's name

Date: 12/09/2024

Date: 12/09/2024

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: ___12/09/2024_____

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that _____***To Jin Yi***_____ (ID No: __***1900176***__) has completed this final year project entitled "Network Intrusion Detection and Alert System" under the supervision of Ts Dr Gan Ming Lee from the Department of Computer and Communication Technology, Faculty of Information and Communication Technology.

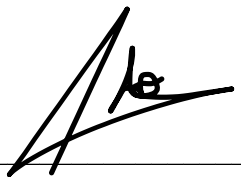
I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

_____
TO JIN YI

DECLARATION OF ORIGINALITY

I declare that this report entitled "**NETWORK INTRUSION DETECTION AND ALERT SYSTEM**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : TO JIN YI

Date : 01/09/2024

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Gan Ming Lee who has given me this bright opportunity to engage in a NIDAS project. It is my first step to establish a career in cybersecurity field. A million thanks to you. Thank you also to Ts Wong Chee Siang and Miss Tan Lyk Yin for providing suggestions and ideas.

To a very special person in my life, Yang Qi, for her patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

Network security has become a critical concern for organizations worldwide as traditional security measures struggle to keep pace with the rapidly evolving landscape of cyber threats. This project aims to develop an intelligent and comprehensive network intrusion detection and alert system (NIDAS) to enhance network security and provide real-time threat mitigation. NIDAS is security technology that enabling security administrators to identify any abnormal or malicious network traffic in real-time.

NIDAS will consist of several key components, including deep network traffic packet inspection, behavior analysis, a prevention rules intrusion detection engine, and an alert prioritization and visualization module. The system will be trained on labeled datasets to identify various types of network attacks and anomalies. By employing a multi-layered approach, NIDAS will be capable of detecting both known and unknown threats, ensuring comprehensive protection against various attack vectors.

The intrusion detection component will utilize a combination of signature-based and anomaly-based detection techniques. Signature-based detection compares network traffic packets with a real-time updated database of known attack patterns, while anomaly-based detection algorithms learn normal behavior patterns and identify deviations. This dual approach will enable the system to effectively detect and respond to both known and zero-day threats.

Upon detecting a potential intrusion, the alert system will generate real-time notifications with relevant details such as the nature of the threat, affected network segments, and recommended mitigation strategies. By integrate Zabbix with IDS capabilities system, the system can reduce false positives and improve the accuracy of threat detection.

This research project aims to create a comprehensive and robust network security solution that provides greater visibility, transparency, and protection against potential threats. By delivering real-time threat detection and actionable insights, the system will significantly enhance an organization's ability to protect its critical assets and maintain secure network infrastructure in the face of ever-changing network threats.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NIDAS	Network Intrusion Detection and Alert System
NIDS	Network Intrusion Detection System
IDS	Intrusion Detection System

Chapter 1

Introduction

In this chapter, we present the background and motivation of our research, our contributions to the field, and the outline of the thesis. This project proposes an integrated network monitoring and intrusion detection system that combines Zabbix and IDS techniques to address the challenges faced by traditional NIDS. By leveraging the information generated by IDS system, the system combines contextual information to prioritize alerts based on risk level and potential impact within the Zabbix configuration. This context-aware approach enables security analysts to focus on the most critical threats, optimizing resource allocation and accelerating incident response processes. The system provides automated mitigation recommendations and a user-friendly interface for visualizing network traffic, alerts, and recommendations. Alert notification mechanisms ensure timely communication of critical security events. By integrating these components into a unified platform, the project aims to enhance the accuracy and efficiency of intrusion detection, reduce false positives, and strengthen an organization's cybersecurity defenses. The successful implementation of this system will contribute to improving the effectiveness of network security monitoring and incident response processes.

1.1 Problem Statement and Motivation

In this rapidly evolving era of information technology, the challenges faced by network security are becoming increasingly formidable. Traditional network security measures are no longer sufficient to address the continuously evolving security threats, such as complex attacks, malware infections, unauthorized access, and even data breaches. Moreover, current intrusion detection systems often suffer from high false positive rates and limited scalability, unable to effectively detect unknown and novel threats. As a result, security teams struggle to accurately prioritize and respond to the vast number of alerts generated by these systems in a timely and effective manner. Therefore, there is a pressing need for a comprehensive, intelligent, and robust network security system that provides enhanced visibility, transparency, and protection against potential threats, while offering real-time threat detection and actionable insights to security personnel.

The motivation behind developing the Network Intrusion Detection and Alert System (NIDAS) lies in addressing the challenges posed by the rapidly evolving landscape of

network attacks. As network threats continue to evolve and become more sophisticated, the methods of network intrusion are becoming increasingly diverse and complex. Such intrusions can have devastating impacts on society, causing significant economic and reputational losses to businesses and nations, and even posing serious threats to national security. Thus, NIDAS aims to provide a powerful and proactive security system that utilizes technologies such as Intrusion Prevention system, deep network packet inspection, and behavioral analysis to accurately detect and classify both known and unknown security threats. The goal is to protect any organization or individual with computers connected to the internet or external networks from network threats, enabling security teams to respond swiftly and effectively to intrusions while minimizing the impact of network attacks. Furthermore, by significantly reducing false positive rates and providing prioritized alerts and actionable measures to security personnel, NIDAS allows them to focus on the most critical incidents in a timely manner. In summary, the motivation behind NIDAS is to develop a system that integrates with existing security infrastructure, promoting a unified and efficient approach to network security management. By providing organizations with a powerful, intelligent, and effective solution for real-time detection and mitigation of network intrusions, NIDAS ultimately aims to strengthen the overall cybersecurity posture of organizations worldwide.

1.2 Objectives

The primary objective of NIDAS is to develop an integrated network monitoring and intrusion detection system that enhances the organization has the ability to detect and respond to cyber threats effectively. Within combining the capabilities of Zabbix and IDS techniques, integrate become a system that can provide visualize network traffic, alerts and mitigation recommendations. By integrating Intrusion Detection system and Zabbix into NIDAS, can achieve comprehensive network security monitoring and event response. IDS focuses on detecting network-level intrusions and threats, while Zabbix provides flexible alerting, correlation analysis, and reporting functionalities. Configure IPS rules and Zabbix monitoring metrics to detect a wide range of network attacks and anomalies, including known and emerging threats. This combination helps you promptly detect and respond to network security incidents, enhancing overall security protection capabilities.

While in the task of alert prioritization and mitigation recommendations, by configuring Zabbix sets the alerts that the threat detected by IDS, incorporating preset rules, threat intelligence and historical data those variety contextual information to implement priority alerts, based on their

risk level and potential impact to the organization so that the security teams enable to focus on the most critical threats. With the detected intrusions and their severity, the system will automated provide the recommendations for mitigation actions, enhance the efficiency of network administrators in incident response processes.

By using Zabbix to design an integrated network traffic visualization, alarm monitoring, and mitigation recommendations, it is more advantageous for network security administrator to monitor the entire network. Zabbix's alarm notification mechanisms, such as email or SMS, can notify security administrator in a timely manner, ensuring that critical security events are promptly communicated to the network security team. This will greatly enhance the system's performance and efficiency in handling large amounts of network traffic and alerts without compromising accuracy or responsiveness. Compared to traditional intrusion detection systems, the goal is to reduce alert handling time by 15%, thereby achieving faster incident response. It is important to note that the project will focus on integrating Zabbix and IDS both component as a system, as well as creating user interface and visualization components to provide a comprehensive network security monitoring and intrusion detection solution. It will not cover the development of the core Zabbix and IDS components themselves, as they are mature open-source tools.

1.3 Project Scope and Direction

The aim of project is to develop NIDAS that can integrated network security monitoring, intrusion detection and automated alert and mitigation recommendations. Although now there are a lot of strong network security software and method, but once the intrusion successfully access into the network it is hard to analyze out the severity of the intrusion immediately. The data will exposed during analyze the intrusion, may cause huge impact to the organization. So in this thesis, NIDAS aims to provide a comprehensive network security monitoring and intrusion detection system that leverages the strengths of open-source tools like Zabbix, while using the techniques of Zabbix to enhance alert prioritization and set mitigation recommendations. By integrating these components, the system will enable real-time monitoring, intrusion detection, and automated response recommendations, ultimately improving an organization's ability to detect and respond to cyber threats effectively.

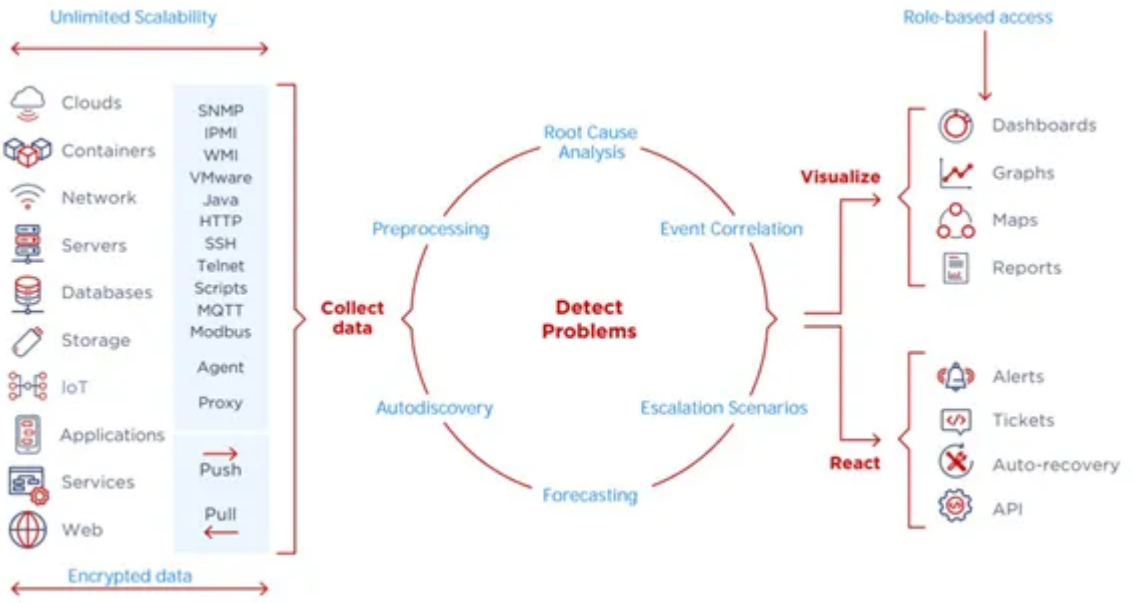


Figure 1.1 Infographic Zabbix from Medium Blog

1.4 Contributions

Network security has always been a critical issue in the rapid development of networks. The proposed integrated network monitoring and intrusion detection system can benefit organizations of various sizes and industries. By breaking through the limitations of traditional intrusion detection systems, this project aims to improve the overall effectiveness of cybersecurity measures in the field of network security, contributing to the establishment of a more secure and resilient digital ecosystem. The integration of Zabbix and OPNsense technologies can bring several key benefits, such as improving the accuracy of threat detection. Under OPNsense suricata detection and prevention rules, the system can compare from historical data and effectively detect known and unknown threats. This approach can significantly reduce false positive alerts, make the system more adaptable to evolving network threats, and decrease the risk of undetected intrusions. Furthermore, alert prioritization and incident response will also benefit overall network security, enabling security teams to focus their efforts on the most critical threats, quickly obtain information on key issues, and swiftly respond to incidents with appropriate measures. By providing automated recommendations for mitigation actions, security incidents can be contained and resolved more rapidly and effectively. The combination of network monitoring and intrusion detection and prevention features improves operational efficiency, reduces the workload of security teams, and enables them to manage security incidents more effectively. Moreover,

through continuous monitoring of network traffic, the network security team within an organization can analyze this data to identify potential security threats, not only effectively reducing risks by staying ahead of emerging threats but also fostering a culture of cybersecurity awareness and preparedness among the network security team.

The importance of this project extends beyond the direct benefits to individual organizations. By enhancing the ability to detect and respond to cyber threats, it can indirectly help protect sensitive data, intellectual property, and critical systems, safeguarding the interests of businesses, governments, and society as a whole. By advancing cybersecurity research and the application of intrusion prevention techniques in intrusion detection, the project has the potential to drive innovation and inspire further developments in the field, ultimately benefiting the broader cybersecurity community. In summary, the proposed integrated network monitoring and intrusion detection system has the potential to make a lasting impact in the field of network security by improving the accuracy of threat detection, enhancing alert prioritization, streamlining incident response, benefiting organizations, fostering cybersecurity awareness, and contributing to a more secure digital environment.

1.5 Report Organization

As the reliance on computer networks continues to grow, so does the need for effective security measures to prevent malicious activities and unauthorized access. Network Intrusion Detection Systems (IDS) have become a crucial component of network security, identifying and responding to intrusion threats caused by security vulnerabilities through real-time monitoring and analysis of network traffic.

Historically, early security measures primarily focused on access control and authentication mechanisms. However, as networks evolved, expanded, and became more complex, it became evident that additional layers of security were necessary. In the 1980s, James P. Anderson introduced the concept of intrusion detection, which aimed to monitor system logs to detect anomalies and suspicious activities [1]. This earliest and initial concept of intrusion detection laid the foundation for the development of future intrusion detection systems. Over the years, intrusion detection systems have evolved and made significant progress alongside the development of computer networks. Early systems heavily relied on rules and predefined signatures of known attacks to identify malicious activities, often leaving them very passive in defending against network threats. New or unknown attacks frequently left systems and organizations vulnerable to significant persecution. To address this issue, anomaly-based

detection techniques were introduced, which involved creating a baseline of normal network behavior and flagging any deviations as potential intrusions. In today's network security, even machine learning and artificial intelligence techniques have been incorporated, enabling them to adapt and learn from past events to enhance their detection capabilities.

The Network Intrusion Detection and Alerting System (NIDAS) is a project designed to enhance network security by providing advanced intrusion detection and monitoring alert capabilities. It builds upon the existing knowledge and techniques in the field of intrusion detection while incorporating novel approaches to improve the accuracy and efficiency of detecting and responding to security threats. NIDAS employs various techniques such as deep packet inspection, real-time network traffic monitoring, behavioral analysis, and machine learning algorithms to identify malicious activities and generate timely alerts for security personnel to take appropriate actions. The four key concepts involved are: Intrusion Detection System (IDS), which is a software or hardware system that monitors network traffic for suspicious activities and alerts administrators when potential security breaches are detected; Deep Packet Inspection (DPI), a technique used to examine the contents of network packets beyond the header information, enabling the analysis of application-level data for security purposes; Behavioral Analysis, the process of studying patterns and behaviors of network entities to identify anomalies and potential security threats, triggering anomaly-based detection and flagging any deviations from the established baseline as potential intrusions; and Signature-based Detection, a detection method that relies on predefined patterns or signatures of known attacks to identify malicious activities, which can significantly reduce false positive alerts. The details of this project are shown in the following chapters. This report is organised into 7 chapters: Chapter 1 Introduction, Chapter 2 Literature Review, Chapter 3 System Methodology, Chapter 4 System Design, Chapter 5 System Implementation, Chapter 6 System Evaluation and Discussion. The first chapter is the introduction of this project which includes problem statement, project background and motivation, project scope, project objectives, project contribution, highlights of project achievements, and report organisation. In Chapter 2, NIDAS backgrounds are reviewed. Chapter 3 and 4 show the system architecture and activity diagram. Chapter 5 will setup the system environment and Chapter 6 will be the penetration testing. Chapter 7 concluded the project and given some recommendation.

Chapter 2

Literature Review

2.1 Traditional Intrusion Detection System

Traditional Intrusion Detection Systems have been a vital component of network security for several decades. Early works, such as the seminal paper by Denning [3], laid the foundation for IDS research. Lunt [4] and Bace [5] further expanded on the concepts and architectures of IDS. IDS typically operate using Signature-Based Detection, it compares network traffic with a database of known attack patterns.

2.1.1 Signature-Based Detection

Signature-Based Detection was first implemented in 1987 from John McAfee an antivirus programme by providing a function called “VirusScan”. It used a database of virus signatures to identify and remove known viruses from infected. This is a common intrusion detection and defense technique used in nowadays computer security to identify known malware by searching for specific patterns or signatures in files or network traffic. According of the signature characteristic that is unique from a specific piece of malware, it can derived by analyzing the code or behavior and identify during in the future instances of the same malware to do the blocking.

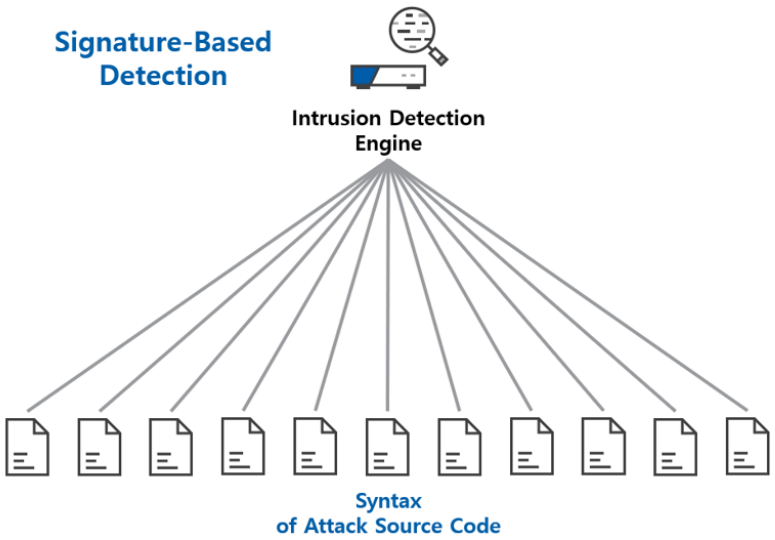


Figure 2.1 Signature-Based Detection

2.1.2 Signature-Based Strengths and Weakness

Strengths:

- High accuracy: Signature-based technology can accurately identify known attack behaviors with almost no false positives
- Efficiency: Signature-based technology is very efficient in detecting and defending against known attacks and can respond quickly and block attacks.
- Low Costing: Low computational overhead

Weakness:

- Unable to detect unknown attacks: Signature-based technology can only detect known attack behaviours, but ineffective against unknown attack, so it is vulnerable to zero-day threats.
- Easily bypassed by attackers: Attackers can evaded the detection of Signature-based technologies by modifying the characteristics or rules of attack behaviors using polymorphic malware or other techniques.
- Maintaining the attack signature database: Signature-based technology needs to establish and maintain a huge attack signature database and updated regularly so that can remain effective, which requires a lot of work and resources.

2.2 Network-Based Intrusion Detection Systems (NIDS)

NIDS ability is to monitor and analyze network traffic for suspicious activities, technique such as anomaly detection [16] and machine learning-based approaches [17], have improved the accuracy and efficiency of detecting network-based attacks. The essential of anomaly detection involves using various approaches such as statistical-based, machine learning-based, and knowledge-based techniques to identify network-based anomalies and intrusions. Effective anomaly detection in NIDS contributes to the overall security of computer networks by detecting and alerting on potential threats. Snort is most widely used open-source NIDS.

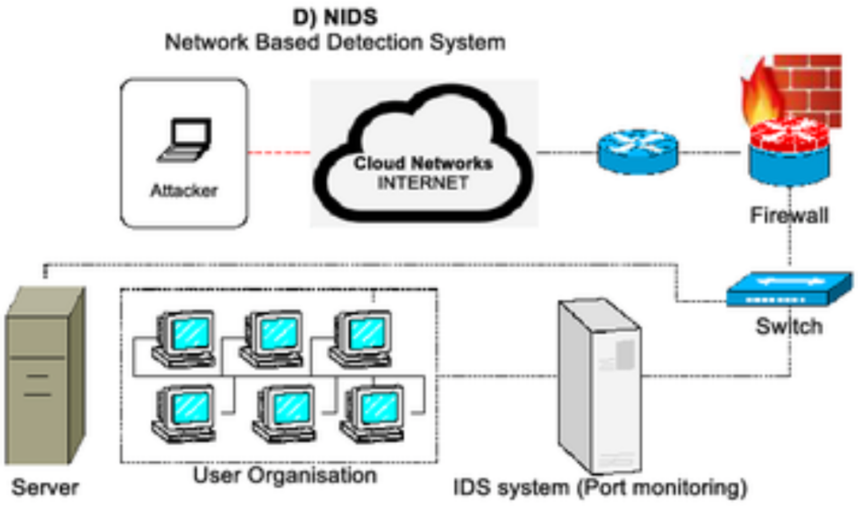


Figure 2.2 Network Intrusion Detection System

2.2.1 Anomaly Detection

Anomaly Detection typically detects anomalous behavior using machine learning algorithms that automatically learn patterns of normal behavior and identify anomalous behavior that can be applied to various data types, such as network traffic, system logs, user behaviour, financial transactions and sensor data. This is a technique used in computer security to identify unusual patterns or behavior that may indicate a security breach. The concept of anomaly detection can be traced back to various fields such as statistics, data mining and machine learning. This method was begin in 1990s by exploring more sophisticated methods for anomaly detection in network security. One of the earliest machine learning-based approaches was proposed by Mukkamala et al. in 2002 [10], which used a neural network to detect network intrusions. Since then, anomaly detection has become a widely used technique and an active area of research in network security, the new techniques and approaches are continually being developed to improve its effectiveness and reliability. Although it produces a high rate of false positives, it is still an important tool for detecting novel threats and identifying suspicious behaviour in various fields.

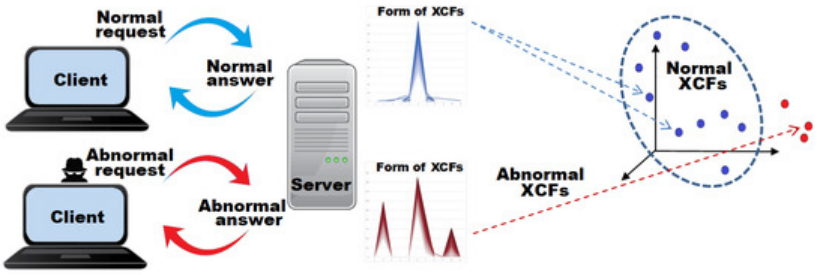


Figure 2.3 Anomaly Detection

2.2.2 Anomaly Detection Strengths and Weakness

Strengths:

- Effective Detection: Can detect unknown or anomalous data points or events, allowing for early detection of potential problems from zero-day threats
- Automation: Detection can be performed by the machine learning, doesn't require signatures or prior knowledge of the attack, reducing the work and manual intervention and improving efficiency and accuracy
- Monitoring: Can be used to detect insider threats or unusual user behaviour in various data types and fields, such as network security, finance, medical care

Weakness:

- High False Alarm Rate: May identify some normal behaviours as abnormal behaviours and resulting in false positives
- Large Amount Dataset for Machine Learning: Requires a large amount of data to train the algorithm in order to accurately identify abnormal behaviour
- High False Negative Rate: May be less effective against complicated attacks that mimic normal behaviour and mistakes some truly abnormal behaviours for normal behaviours, lead to under reporting

2.3 Behaviour Analysis

Behaviour analysis may have similar techniques and goals with Anomaly Detection in cybersecurity to identify potential threats and malicious activity but there are different in their approach and the type of data they analyze. Behaviour analysis more concentrate on users activities that are out of the ordinary or suspicious, based on a baseline of expected behaviour by analyzing historical behaviours and establishing behaviour models to detect abnormal behavious. It provides a more proactive and adaptive approach to network security

monitoring for security management, risk management and other fields, such as detecting employee leaks, illegal command and control activties, or network-based attacks, such as worm propagation and DDOS attacks, etc.

2.3.1 Behaviour Analysis Strengths and Weakness

Strengths:

- Effective Detection: Can analyze multiple indicators and comprehensively analyze the user behaviour and activities to provide more comprehensive analysis results than Anomaly Detection and detect unknown or zero-day threats
- Predict Behaviour: By analyzing historical data and trends can helps to provide insights into the motives and methods of attackers
- Automation: Analysis can be performed by the machine learning to achieve highly automated analysis and identification, greatly improving on efficiency and accuracy

Weakness:

- High False Alarm Rate (Refer 2.2.2)
- Large Amount Dataset for Machine Learning (Refer 2.2.2)
- Resources and Costing: Complex to implement and require significant computing resources and expert skills

2.4 Machine Learning-Based

Machine learning techniques have gained significant attention in the field of network security, particularly in the context of Network Intrusion Detection Systems (NIDS) and Network Intrusion Detection and Alert Systems (NIDAS). Traditional NIDS and NIDAS rely on signature-based and rule-based approaches, which have limitations in detecting novel and evolving threats. Machine learning algorithms have the potential to enhance the detection capabilities of these systems by learning patterns and anomalies from network traffic data [11] ,[17].

2.4.1 Machine Learning-Based Strengths and Weakness

Strengths:

- Improved Detection Accuracy: Machine learning algorithms can learn complex patterns and relationships in network traffic data, enabling them to detect intrusions and anomalies with high accuracy

- Adaptability to Evolving Threats: Machine learning models can adapt to new and evolving threats by continuously learning from new data
- Reduced False Positives: Machine learning techniques can help reduce false positive rates in NIDS and NIDAS by learning to distinguish between normal and malicious network behavior accurately

Weakness:

- Dependency on Training Data Quality: The performance of machine learning-based NIDS and NIDAS heavily relies on the quality and representativeness of the training data. If the training data is biased, incomplete, or not representative of real-world network traffic, the models may fail to generalize well and detect intrusions accurately
- Need for Skilled Professionals: Developing, deploying, and maintaining machine learning-based NIDS and NIDAS require expertise in both machine learning and cybersecurity domains
- Resource Requirements: Training and deploying machine learning models for NIDS and NIDAS can be computationally intensive, requiring significant processing power and memory

2.5 Hybrid Approaches

Hybrid approaches refer to combine elements of signature-based detection, anomaly detection, behaviour analysis, machine learning and other techniques to provide an overall defense against cyber attacks. For example, Signature-based technology is used to detect known attack behaviours, Behaviour Analysis technology is used to detect unknown attack behaviours and zero-day attacks, and Machine Learning technology is used to train algorithms to improve accuracy. With the combining multiple method, hybrid approaches are getting important in recent years because of the cyber threats have become more sophisticated and difficult to detect by using traditional approaches. Within the multiple techniques and methods, hybrid approaches provide a more stronger and a wider range of cyber threats detection and prevention. This technique will keeps increasing it's important and continue to evolve and become more complex in the future of network security.

2.5.1 Hybrid Approaches Strengths and Weakness

Strengths:

- Increase Detection Accuracy and Reduce False Alarm Rate: By combining multiple techniques, accuracy of detection get improved and low false alarm rate so that more availability for the system
- Leverage: Due to the strength of multiple approaches, can upgrade a great level security protection and detection in cybersecurity and provide a more comprehensive network security

Weakness:

- Complexity: Combining multiple techniques require higher technical skill, significant computing resources and that would be more costly than a single approach
- Maintenance and Performance: Because of multiple techniques will require more times and resource to do tuning and optimization during the maintenance for achieving optimal performance

2.6 Cloud-Based NIDAS

Cloud-based NIDAS refers to deploying Network Intrusion Detection and Alerting System (NIDAS) in a cloud computing environment to leverage the scalability and flexibility so that can provide more efficient intrusion detection and cyber threats prevention services. As traditional NIDAS is usually deployed on a local or private network, requiring specialized hardware and software support, while Cloud-based NIDAS can be deployed and managed on the cloud through the virtualization technology provided by cloud service providers, without the need to purchase, install and maintain additional hardware and software equipment so that can be more flexible on organizations to handle large volumes of network traffic and scale their defences up or down as needed. According of the benefits from Cloud-Based NIDAS, there are more organizations move their IT infrastructure to the cloud and leverage their cloud-based machine learning algorithms and artificial intelligence to improve threat detection and prevention.

2.6.1 Cloud-Based Strengths and Weakness

Strengths:

- Scalability and Flexibility: Cloud resources can be dynamically adjusted as needed to adapt to changing network traffic and attacks

- Costless: No need to purchase, install and maintain additional hardware or software equipment, reducing costs and maintenance difficulties
- Reliability and Stability: The cloud computing platform has high availability and fault tolerance can provide real-time monitoring and analysis

Weakness:

- Network Latency: Require high-speed internet connections to ensure real-time performance
- Data Secure and Privacy: The data is stored outside of the organization's control, may concerns the security and privacy of data
- Manage and Control: Require specialized skills and knowledge, which increases the difficulty of cloud management

2.7 Review of Existing Systems

2.7.1 Snort

Snort is an open-source Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) used to monitor network traffic in real-time. It works by analyzing network packets and comparing them against predefined rules to identify and alert on potentially malicious activities, such as port scans, malware, and suspicious behavior. Snort can operate in several modes, including packet logging, network traffic analysis, and intrusion prevention, making it a versatile tool for network security.

Strengths and Weakness

Strengths:

- Signature-based Detection: Snort uses a rule-based system to detect known threats by matching network traffic patterns to a library of attack signatures
- Scalability: Snort is highly configurable and can be integrated into small or large network environments, offering a balance between security and performance
- Customizable Rules: Users can create custom rules to tailor Snort's detection capabilities to specific needs and environments

Weakness:

- Complexity: Require higher technical skill and understanding to pre setup rules
- Unfriendly Interface: All the detected traffic or threats, alerts and log data are output in the form of log files. These log files need to be processed, indexed and visualized with the help of external tools (such as Graylog, Splunk or ELK stack) to provide

administrators with a more intuitive chart and data analysis view. This is not intuitive for monitoring, analysis and management, especially in large-scale network environments, manual log analysis is very time-consuming

- Lack of monitoring functionality: Snort cannot monitor server resources, databases, or application status which the network down or overload it did not know

2.7.2 Graylog

Graylog is an open-source log management and analysis platform used for collecting, storing, and analyzing large amounts of machine data in real time. It helps organizations monitor their IT infrastructure by centralizing logs from various sources (such as servers, network devices, and applications) and enabling users to search, analyze, and visualize this data. is widely used for log aggregation, monitoring, troubleshooting, and security analysis, providing IT teams with insights into their systems' health and performance.

Strengths and Weakness

Strengths:

- Centralized Log Management: Graylog is designed for log collection, storage, and analysis, which means it excels at dealing with large volumes of unstructured data from various sources like servers, network devices, and applications
- Search & Filtering: allows to quickly filter through the alerts, helping isolate specific events or patterns. This helps reduce the time spent on identifying real threats among false positives

Weakness:

- Limited Real-Time Monitoring: Primarily designed to analyze historical data rather than providing real-time system health monitoring
- Complexity in Configuration and Scaling: Limited on handling large-scale monitoring environments. It will be complex to configure and scale in large environments. As the volume of logs grows, it requires robust infrastructure and careful tuning to handle the load efficiently
- Lack of Advanced Alerting: Alerts are typically triggered by specific log patterns or conditions but are not as advanced when it comes to monitoring complex performance metrics over time

2.7.3 Pfsense

PfSense is a firewall and router software distribution based on FreeBSD. It was highly popular for providing enterprise-level network security features at a low cost previously. PfSense is known for its flexibility, performance, and rich feature set, which makes it a comprehensive solution for managing network traffic and enhancing security. It is known for its robustness and reliability, making it a trusted solution for protecting and managing networks of all sizes. However, although Pfsense is closely similar with OPNsense but it needs to be pay now.

Strengths and Weakness

Strengths:

- Comprehensive Feature Set: The feature set is highly comprehensive, allowing administrators to build complex networks without requiring additional tools. It packed with a broad range of firewall, routing, and security features IDS/IPS that are typically found in expensive, proprietary firewall solutions
- Highly Configurable: Offers a high degree of customization, allowing network administrators to tweak and fine-tune firewall rules, NAT, VPN configurations, and routing to meet specific needs. Can control nearly every aspect of network's security and performance with detailed configuration options
- Extensive Plugin/Package System: Allows for the addition of extra features and tools, enhance the flexibility

Weakness:

- Compatibilty of IPv6: PfSense can be more difficult to configure for IPv6 in complex network environments particularly in combination with advanced features like VPNs and with regards to its configuration and functionality in dual-stack environments (IPv4 + IPv6)
- Costly: Previously is an open-source tools but now need to subscribe
- Lack of UI: Harder to navigate and find certain configuration options. It will overwhelming for new users to network management due to its complexity and lack of modernization

2.7.4 Zenarmor

Zenarmor (formerly known as Sensei) is a powerful, next-generation firewall plugin designed for enhancing security on network platforms like OPNsense and pfSense. It provides advanced network protection by offering deep packet inspection (DPI), application awareness, and cloud-based threat intelligence. Zenarmor focuses on delivering enterprise-grade security features without compromising performance, making it suitable for small to large-scale network environments.

Strengths and Weakness

Strengths:

- Deep Packet Inspection (DPI): allowing it to analyze network traffic at a granular level. This capability enables administrators to view, monitor, and control traffic at the application layer, identifying specific apps and protocols being used. This deep insight is useful for detecting and blocking unwanted applications or suspicious activity
- Layer 7 Visibility: Provides detailed insights into user activity, application usage, and potential security risks. Provides administrators Layer 7 visibility, meaning it can classify and monitor traffic at the application layer

Weakness:

- Limited Scale: Limited to environments that require network traffic monitoring. It is not designed to scale across large IT infrastructures that require monitoring of thousands of servers, network devices, and applications
- Historical Data : Lacks long-term trend analysis for system and network performance
- Limited Network Traffic Monitoring: Cannot directly monitor physical device, unable for gathering standard monitoring data or network traffic from network devices such as switches and routers. This makes it limited in managing network infrastructure

Chapter 3

System Methodology

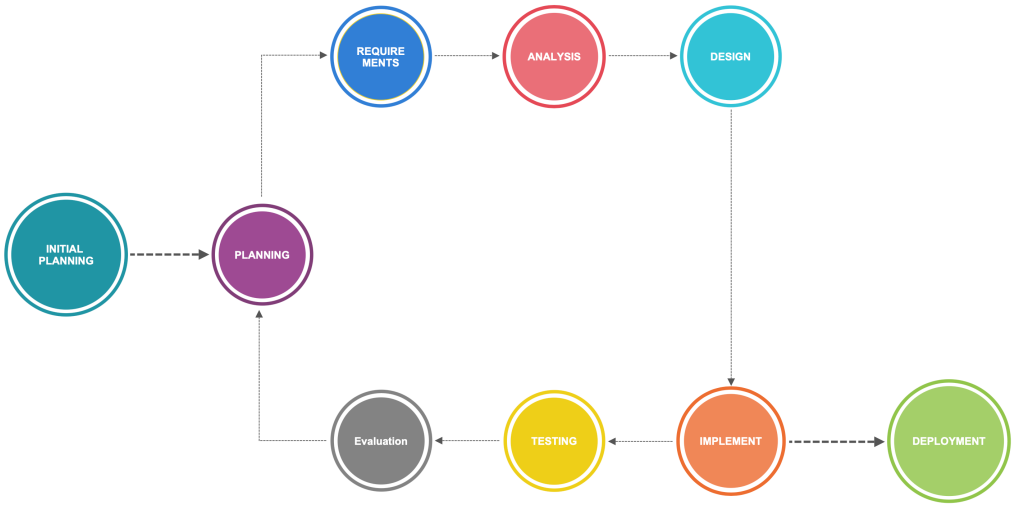


Figure 3.1 Iterative Process Model

Iterative process model represents the project system development lifecycle diagram. In this iterative model lifecycle of project development, each phase has clear tasks and objectives to ensure successful implementation. This article will discuss the various stages of a typical project lifecycle and their importance, covering aspects such as initial planning, detailed planning, requirements gathering, analysis, design, implementation, testing, deployment, and evaluation.

First, the preliminary planning phase is the starting point of the project. During this stage, the broad objectives, scope, and feasibility of the project are determined. The main tasks include preliminary discussions on the project's background, purpose, and expected outcomes, as well as assessing the necessity and potential impact of the project. Although specific technical details may not yet be clear at this stage, the overall framework and feasibility assessment of the project will provide a basis for subsequent in-depth planning. This is followed by the planning phase, which is crucial for developing a detailed project plan. This stage further refines the project schedule, resource allocation, and risk analysis. The project's development will be based on the results of the preliminary planning, establishing a detailed work plan and clarifying task priorities. The success or failure of this phase directly affects the orderly progression of the project.

Entering the requirements gathering phase, the focus shifts to collecting and documenting specific requirements for the project or system. This phase is particularly important as it determines whether the project's final deliverables will meet expectations. During the requirements gathering process, it's essential to ensure that all functional, performance, and user requirements are fully considered and translated into clear documentation. After completing requirements gathering, the analysis phase begins. In this stage, the collected requirements undergo in-depth analysis to ensure that all system functions and performance indicators are fully understood. During this phase, the feasibility and rationality of requirements for development and design need to be carefully examined to avoid significant deviations in subsequent design and implementation processes. Next is the design phase, which is the process of creating the project architecture and system blueprint. Based on the results of the analysis phase, this stage involves designing the system's structure and functional modules. The output of the design phase includes not only the system's technical architecture diagram but may also cover database design, interface design, and system interaction processes. The design work at this stage lays the foundation for the subsequent development.

Entering the implementation phase, the design outputs of the project are transformed into actual code and functionalities. During this stage, code is written and various module functions are implemented according to the technical solutions from the design phase. The implementation phase is typically the core part of the project and also the most time-consuming. Here, it's necessary to ensure that different parts of the system are seamlessly integrated according to design requirements. In the testing phase, the system undergoes comprehensive testing to ensure it meets requirements and functions properly. During the testing process, the system's functionality, performance, compatibility, and other aspects are verified, and potential defects are identified and fixed. The focus of this phase is to guarantee the stability and reliability of the system, avoiding major issues after deployment to the production environment.

Once testing is passed, the project enters the deployment phase. At this stage, the project outcomes are installed and configured in the actual operating environment for end-user use. The deployment process may involve system migration, data import, and infrastructure setup. Ensuring the smooth launch of the system is the core objective of this phase.

Finally, there's the evaluation phase. After the system has been deployed and running for a period, a comprehensive evaluation is conducted. If the system fails to meet business

requirements in certain aspects, adjustments or updates may be necessary. The evaluation work at this stage helps summarize the project's experiences and lessons, providing reference for future project outcomes.

3.1 System Design Diagram

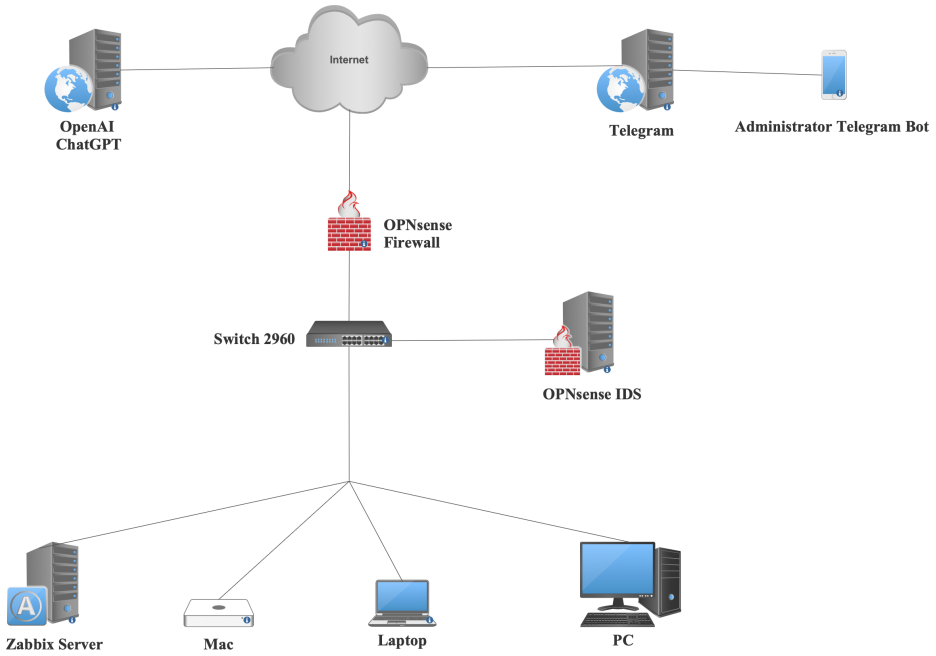


Figure 3.2 Network Diagram

The network topology diagram illustrates a network architecture design that employs multi-layered security protection and monitoring mechanisms, aimed at ensuring secure communication and monitoring between internal devices and external services. This article will analyze the components and functions of this network architecture in detail from the perspectives of network device functionality, communication paths, and security measures.

Firstly, the network entry point is accessed through the Internet, with the network using OPNsense Firewall as the first line of defense for network security. OPNsense is an open-source firewall and routing platform that provides deep packet inspection, traffic filtering, and network security monitoring capabilities. It ensures that traffic from external networks undergoes strict inspection and filtering to prevent potential network attacks.

Connected to the firewall is a Switch 2960, responsible for managing data exchange between devices within the internal network. This switch is connected through multiple interfaces to different types of terminal devices, including Mac, PC, and Laptop workstations. The switch

ensures efficient and secure communication between these terminal devices while allowing different devices to access external networks through the firewall. On the other end of the switch, it is connected to the Zabbix Server and OPNsense IDS (Intrusion Detection System). Zabbix Server is an open-source monitoring tool mainly used for monitoring the performance and health status of network devices and servers.

By configuring OPNsense Intrusion Detection's rule set to detect potential intrusions and malicious activities, deploying OPNsense IDS at the network perimeter or critical nodes, and configuring it to monitor inbound and outbound traffic by using port mirroring method, the detected logs can be sent to Zabbix server for further processing by using zabbix sender. In this network architecture, the Zabbix server can collect real-time performance data from internal devices and generate alert notifications, helping administrators discover potential network issues or device failures. On the Zabbix server, create a item key to receive by using zabbix trapper and process the logs generated by OPNsense IDS. Create corresponding triggers in Zabbix based on the severity and priority of OPNsense IDS alerts, setting appropriate thresholds and conditions to trigger alarms. Configure alarm actions in Zabbix to determine the method of sending alerts and the recipients from the security team based on the trigger conditions and severity.

The OPNsense IDS serves as a second layer of security protection, responsible for monitoring internal network traffic and detecting possible intrusion behaviors. The intrusion detection system analyzes traffic patterns to identify abnormal behaviors and triggers security alerts, further enhancing network security. Utilize Zabbix's graphing and dashboard features to visually correlate OPNsense IDS alerts with other monitoring metrics, such as system performance and network traffic, in a template. Finally, use Zabbix's reporting functionality to generate statistical reports of OPNsense IDS alerts and events for periodic security audits and compliance assessments, providing insights into network security trends and areas for improvement. With this setup, whenever any abnormal network intrusion or threat occurs, the IDS will detect it, generate logs with priority, and promptly send them to Zabbix, triggering alerts to notify the security team members.

This network architecture also integrates OpenAI ChatGPT and Telegram services, including the use of API to input IDS logs into GPT-3.5 to obtain mitigation suggestions, and using Telegram as a communication method to inform network security administrators of current issues and provide the best recommended solutions, implementing automated interaction functions. Thus, communication between the Telegram Bot and administrators can achieve

system status reporting, alert notifications, and the issuance of automated operation and maintenance tasks, to obtain real-time security intrusion detection information and promptly address high-priority critical threads. Through this approach, network administrators can receive real-time system status updates anytime, anywhere, and take action when necessary.

The overall network design architecture demonstrated in this project combines firewall, intrusion detection system, performance monitoring tools, and automated management platforms to form a comprehensive security protection and management system. Each layer in the network has a clear role and function, ensuring secure, stable, and controllable communication between internal devices and external services. Through real-time monitoring and management of network traffic, administrators can quickly identify and address issues, thereby ensuring efficient network operation.

In summary, this network topology diagram shows a comprehensive security architecture that ensures network stability and security by integrating firewalls, intrusion detection systems, performance monitoring tools, and automated management platforms. This design is suitable for various network environments with high security requirements and can effectively address both internal and external security threats.

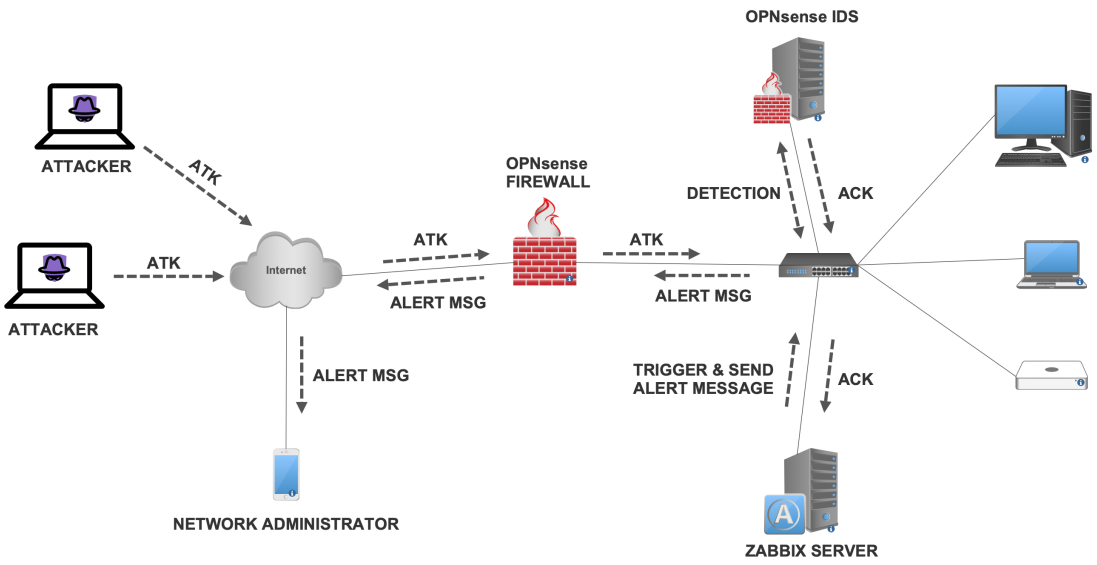


Figure 3.3 Zabbix and IDS Activity Diagram

The activity diagram illustrates the protection mechanisms and automated response processes that occur during a network attack, specifically describing a series of steps including attack detection, alert generation and transmission, administrator notification, and system

confirmation feedback. This article will analyze the workflow of the activity diagram and the function of each component in detail from four aspects: attack detection, alert transmission, administrator response, and system confirmation.

First, according to the activity diagram, it shows the process of an external attacker initiating an attack through the Internet. The attacker exploits network vulnerabilities or weaknesses to send malicious traffic or attempt unauthorized access to system resources. The attack request is transmitted through the Internet to the OPNsense Firewall. At this point, the firewall serves as the first line of defense for the network, responsible for filtering external traffic and blocking potential threats.

When the OPNsense Firewall is breached by attack traffic, the attack will enter the second line of defense and activate the OPNsense IDS (Intrusion Detection System) shown on the right side of the activity diagram, while the firewall continues to process attack traffic, preventing known attacks from further infiltrating the internal network.

The OPNsense IDS system, through in-depth analysis of network traffic, will identify anomalous behavior and further confirm whether it is a malicious attack. If the IDS system detects a threat, it generates an alert log and transmits the information to the Zabbix server via the switch using the Zabbix Sender method.

After receiving the alert log from the OPNsense IDS system, the Zabbix server triggers an automated alert mechanism. This server not only records these logs but can also send alert notifications to network administrators based on preset rules. For example, when the Zabbix server detects multiple attacks or certain types of attacks causing inbound or outbound traffic to become massive, it sends more urgent alert notifications to administrators. At this point, administrators may receive these alerts and suggested solutions on devices such as mobile phones, prompting them to check the network status immediately.

At the center of the activity diagram, the switch's role is to ensure efficient transmission of data and alert information between various network devices. When an attacker attempts to attack terminal devices, steal information, or perform other attacks, after receiving an alert, the OPNsense IDS will actively send logs (ACK) to the Zabbix server through the switch, ensuring the reliability of information transmission. This communication mechanism ensures that network administrators can respond promptly and take measures when necessary, such as isolating attacked devices, closing vulnerabilities, or updating protection rules.

Finally, throughout the entire attack handling process, the network administrator, as the ultimate decision-maker, will determine whether manual intervention or adjustment of the

system's automated response mechanism is needed based on the received alert messages. Administrators can view detailed attack logs, analyze attack patterns, and take countermeasures based on the actual situation by accessing the management interface or directly logging into the firewall and Zabbix server.

In summary, this activity diagram demonstrates a complete automated network security protection process, from attack detection to alert notification and provision of solution suggestions, to administrator response and confirmation. This architecture not only can timely detect and respond to network attacks but also provides administrators with comprehensive monitoring and management tools to ensure the security and controllability of the network system. By integrating firewalls, intrusion detection systems, and automated monitoring platforms, network administrators can effectively prevent various external threats and quickly take countermeasures to ensure stable network operation.

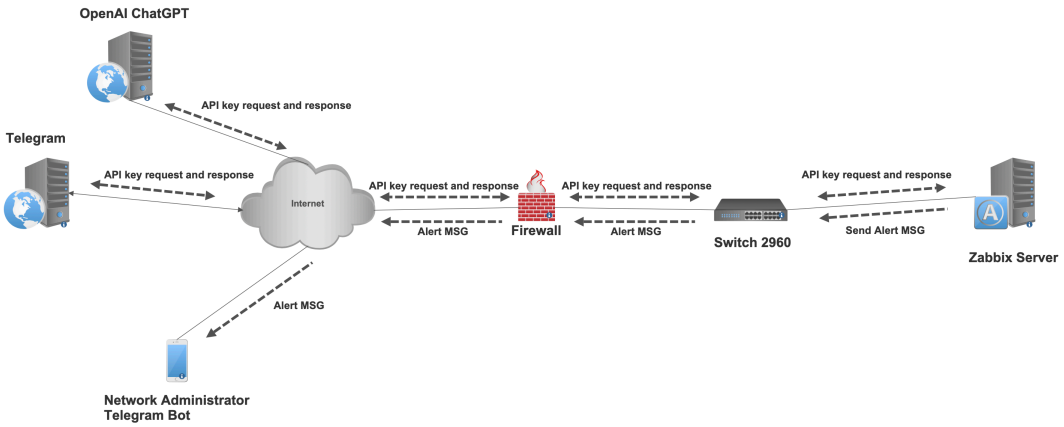


Figure 3.4 Zabbix Alert Activity Diagram

The activity diagram explains the workflow of an automated network monitoring and notification system that interacts between the Zabbix server, ChatGPT, and Telegram using API keys and tokens. The core functionality of this system is to utilize multiple components such as firewalls, switches, Zabbix servers, and Telegram Bots to monitor network conditions in real-time and send alert information to administrators through token requests and responses. First, the Zabbix Server identifies abnormal behavior, such as excessive traffic or device failures, or obtains logs through OPNsense IDS, and generates an alert message. In this diagram, when sending alert information to network security administrators, the Zabbix server uses API key requests and responses to communicate with ChatGPT. After the request message

is sent from the Zabbix server, it is transmitted through the switch to the firewall. The firewall is the first line of defense for the network, which can both filter external malicious traffic and receive and forward request messages through API requests. Before sending the alert information, the Zabbix server sends security threat log information to GPT-3.5 via API and requests a response, providing suggested solutions for the security threat logs. This integration can add more intelligent response capabilities to the system, providing analysis, suggestions, or automated processing solutions for alert events to administrators through natural language processing.Telegram service is used to notify network administrators. The Telegram Bot establishes communication with the Zabbix server through token key requests and responses. When abnormal behavior is detected in the network, alert messages are immediately sent from the Zabbix server to the Telegram server and then sent to the administrator's device through Telegram. Administrators can receive and view alert details in real-time on their mobile devices and take corresponding measures.

The communication mechanism of the entire system is based on API key requests and responses. Each component authenticates and communicates through secure API keys. This method ensures the security and accuracy of information transmission and avoids unauthorized access or tampering. At the same time, alert information is quickly transmitted from the monitoring server to administrators and related systems through these requests and responses, ensuring the system's immediate response capability. Finally, after receiving alerts through the Telegram Bot, network administrators can quickly take action. Therefore, administrators can remotely log into the network system, check the source and details of the alert, and take further measures such as adjusting firewall rules, updating system patches, or adding additional monitoring policies.

This model demonstrates an efficient automated network monitoring and alerting mechanism. Through API key interactions, various components can communicate and collaborate in real-time in a secure environment. The Zabbix server is responsible for generating and sending alert information, while the external platform OpenAI ChatGPT helps administrators quickly receive and analyze alerts. This design not only improves the security and reliability of the network but also provides more intelligent support for network management, significantly enhancing the response speed to network events.

Chapter 4

System Design

4.1 System Block Diagram

4.1.1 SNMP services between Zabbix server and Switch

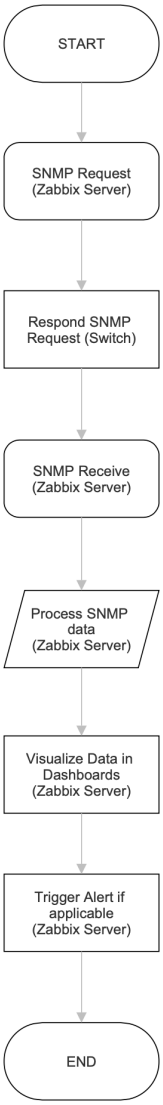


Table 4.1 System Design of Zabbix Monitoring Switch using SNMP

From the table 4.1, Zabbix server will send SNMP request to the switch and waiting the respond. Once the switch received the request, it will reply the SNMP request to the Zabbix server. When Zabbix receive the SNMP from switch, it will start to process the SNMP data and virtualize the data in dashboards. According to the SNMP data, Zabbix server will trigger alert to the administrator if the data priority impact is high.

4.1.2 OPNsense Zabbix Agent and SNMP services

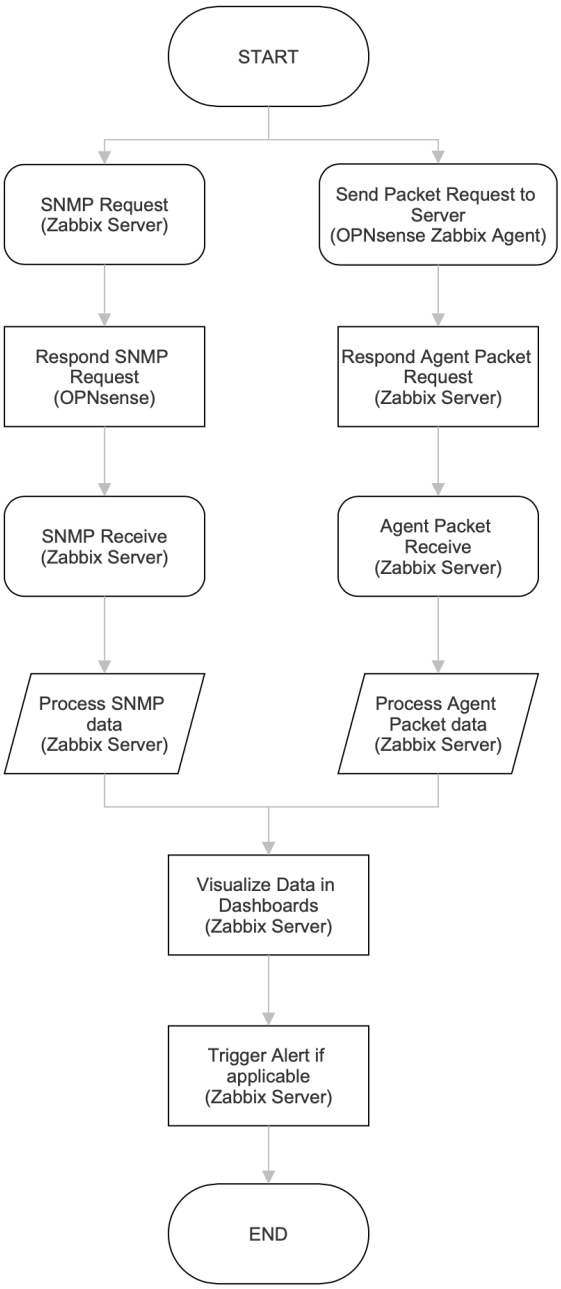


Table 4.2 System Design of Monitoring OPNsense using SNMP and Zabbix Agent

In table 4.2, Zabbix server will send SNMP request to the OPNsense while Zabbix Agent will send packet request to Zabbix server and both waiting the respond. OPNsense will reply SNMP request to the Zabbix server once receive the request while Zabbix server will respond Zabbix Agent packet request. When Zabbix receive the SNMP and Agent packet data from OPNsense, it will start to process the data and virtualize the data in dashboards. According to

the SNMP and Agent packet data, Zabbix server will trigger alert to the administrator if the data priority impact is high.

4.1.3 OPNsense Suricata IDS

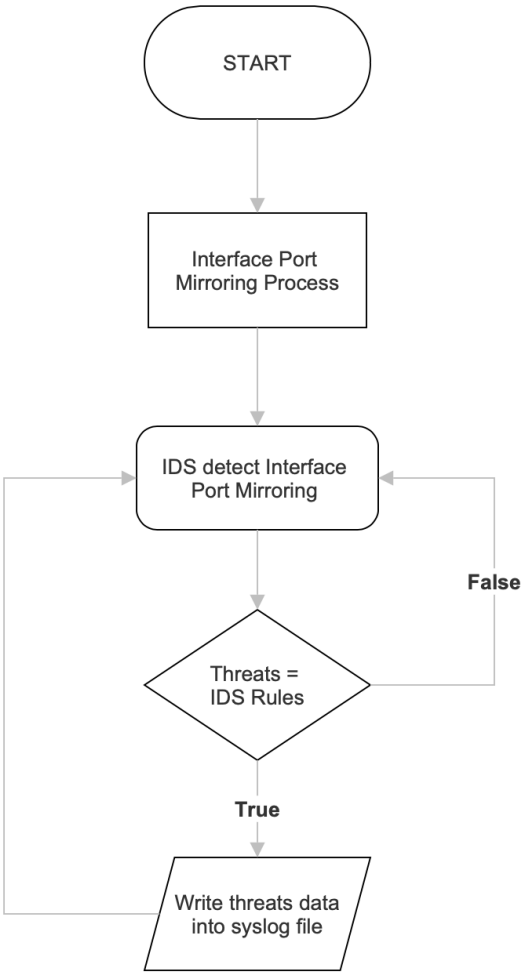


Table 4.3 System Process Design of OPNsense Intrusion Detection

Table 4.3 represents the process of intrusion detection detects threats with a port mirroring network interface. Interface port will be setting up port mirroring in a switch and the IDS monitors the mirrored traffic and detects any activity through the mirrored interface. If the detected traffic matches any predefined threat rules or signatures, the IDS will write the corresponding data into the syslog file for further analysis or alerting. If false the execution will repeat detect the port mirroring interface.

4.1.4 OPNsense Suricata Log Check Process

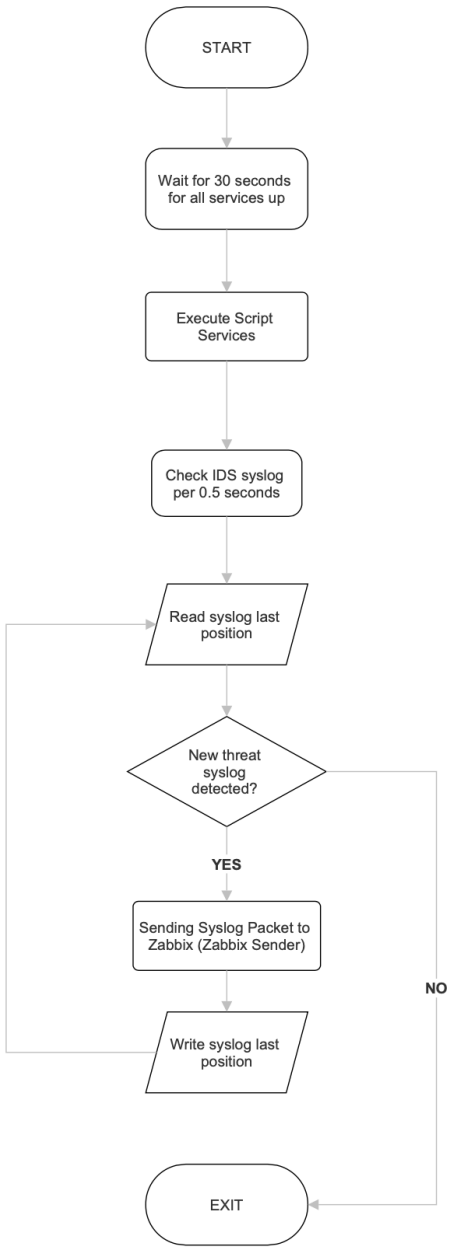


Table 4.4 System Process Design of OPNsense Log Check

The Table 4.4 flowchart describes the process of OPNsense's monitoring system detecting threats in system logs (syslog) and sending this threat information to Zabbix for processing. After OPNsense starts, this bash script waits for 30 seconds to ensure all services are started and the script service is executed. After the delay, a monitoring process begins every 0.5 seconds, where the system checks the Intrusion Detection System (IDS) syslog file every 0.5 seconds to detect any new log updates or changes. The system reads the last processed

position of the syslog file and checks for new threat logs. After detection, it records the last detected position of the syslog to ensure that only new log entries after this position are processed. The system then repeats the cycle, continuing to read the last processed position of the syslog file. If there are no new threats, the system process exits, waits for the next 0.5-second monitoring process, and continues to monitor and detect whether there are new syslog entries. If there are new threats, the newly detected threats will be sent to Zabbix for further processing or alerting through Zabbix Sender. After this, the system writes the last position of the syslog and continues the detection cycle. If there is no new threat information, the process exits. This process avoids repeated processing of threat information and tracks the latest log information in real-time.

4.1.5 Zabbix

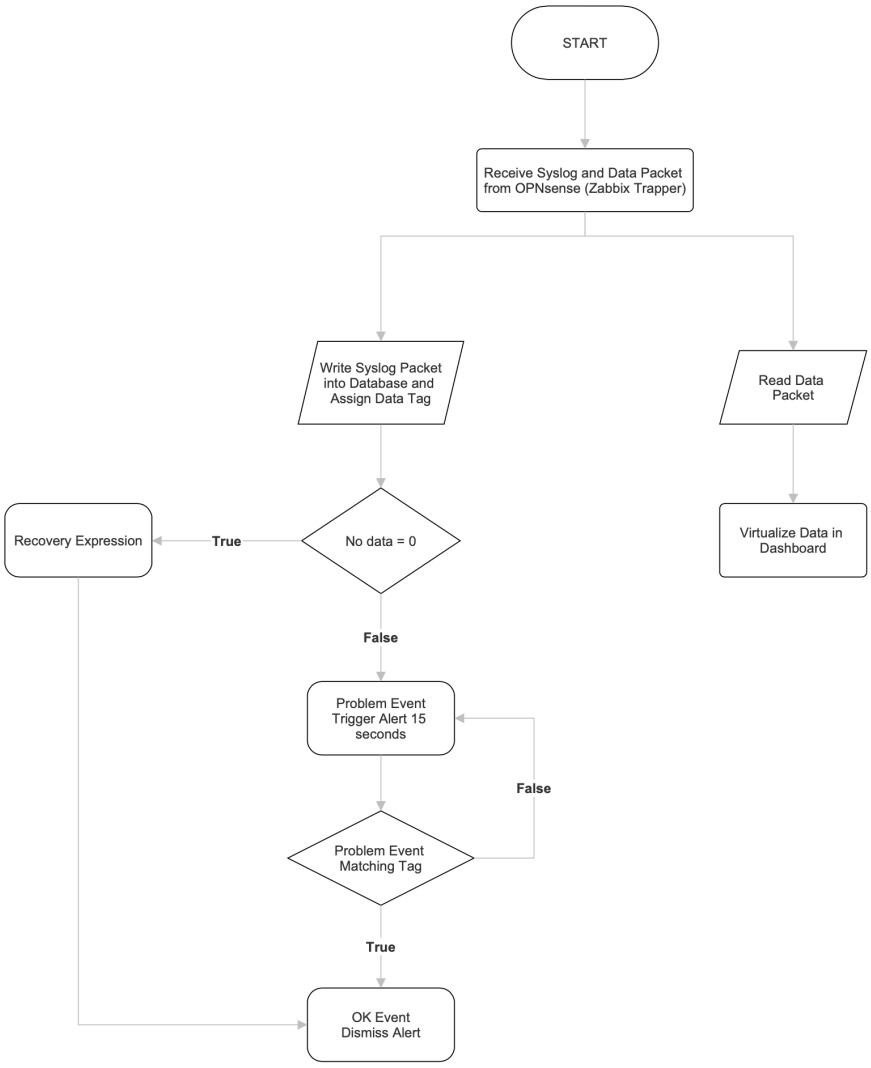


Table 4.5 System Design of Zabbix Monitoring

The Table 4.5 illustrates a Zabbix monitoring system process. After receiving syslog and data packets from OPNsense, the first step is to write the syslog data packet into the database and assign data tags, while simultaneously determining if valid data has been received. If there is no data (True), the process enters the Recovery Expression stage and proceeds to an OK event, canceling the previous event alert. If there is data (False), it triggers a problem event alert for 15 seconds. Typically, the alert signal continues for 15 seconds, after which it checks if the problem event matches the previously assigned data tag.

If there's no match (False), the alert signal continues. If there is a match (True), it enters an OK event to cancel the alert, marking the event status as "OK".

Concurrently, the second process when syslog data packets are received from OPNsense to the Zabbix system is to read the data packets for further data processing and analysis, and to visualize the data on a dashboard for system administrators to view and monitor.

In summary, syslog data packets from OPNsense are received by the Zabbix system, processed, and alerts are triggered based on problem events. Finally, the system checks whether the problem is resolved by matching tags or checking for new data, ultimately canceling the alert. At the same time, the data is visualized on a dashboard for real-time monitoring by administrators.

4.1.6 Zabbix Access MySQL Database

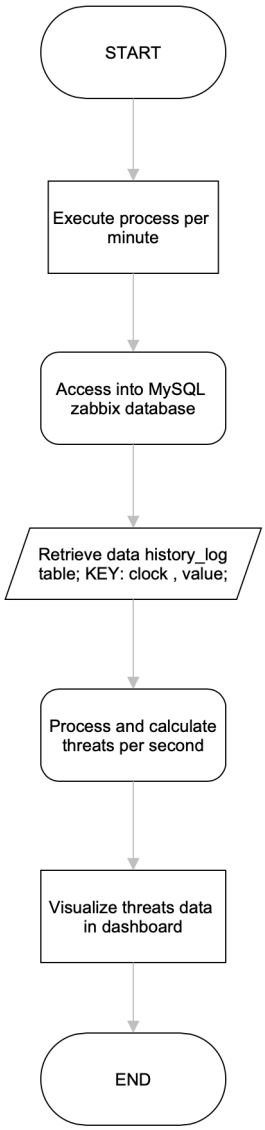


Table 4.6 System Process Design of Zabbix with MySQL

Table 4.6 flowchart illustrates an automated system that runs a process every minute to ensure data and threat calculations are updated in a timely manner. After the process is executing, the system connects to MySQL database associated with the Zabbix monitoring system, which is used to store various monitoring data.

The system retrieves data containing clock (timestamp) and value (log threats data) from the history log table (history_log) in the zabbix database. The clock is used to record the time when events occur, while the value represents specific monitoring data. The system then analyzes and processes the data extracted from the database. Using a specific algorithm, it calculates the threat level per second. Finally, the calculated threat data is visualized and

displayed on a monitoring dashboard for administrators to view and analyze the current security status of the system.

Through this system, historical log data is extracted from the Zabbix database every minute, the number of threats per second is calculated, and the results are displayed on the dashboard. This ensures that real-time data on system threats is available for convenient monitoring and analysis.

4.1.7 Zabbix Trigger Alert with Recommendation

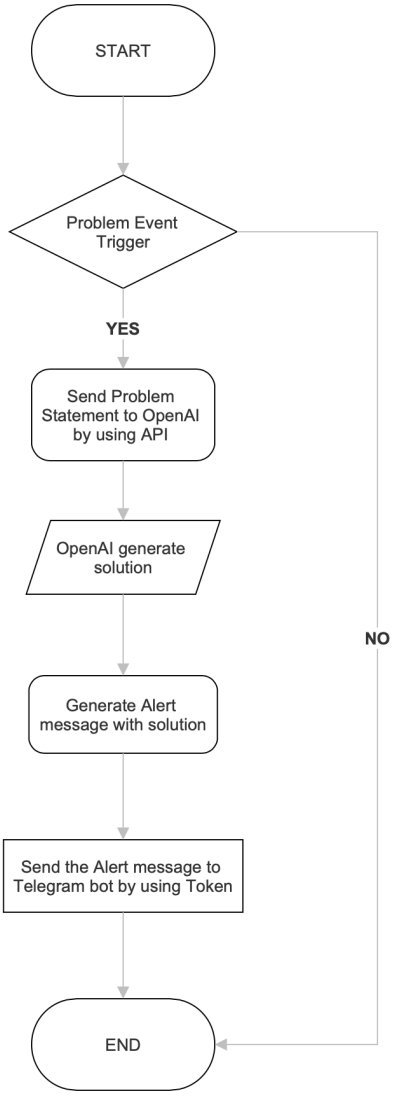


Table 4.7 System Design of Zabbix Trigger Alert

From the Table 4.7, the flowchart illustrates an automated system for handling problem events, integrating OpenAI's capabilities to generate solutions and using a Telegram Bot to

send alerts and solutions to administrators. This design enables the system to automatically generate solutions and quickly notify administrators when problems are detected, thereby improving the efficiency of problem handling.

When a problem event is triggered, there is a decision point where the system determines whether it meets the criteria to initiate this process. If it does not meet the criteria (NO), the process ends. If it does meet the criteria (YES), the system sends detailed information about the problem to OpenAI via API and requests a solution to be generated. After receiving the problem description, OpenAI provides an optimal solution for the issue. The system then combines the generated solution with the alert information to form a complete alert message. Finally, the alert message is sent to the Telegram Bot using a token to notify network administrators or relevant personnel. This allows for real-time understanding and receipt of information about critical issues, thereby improving the efficiency of prioritizing and handling problems.

This approach ensures that when the system detects a problem, it can automatically generate a solution and quickly notify administrators, thus enhancing the overall efficiency of problem management.

4.2 System Components Specification

4.2.1 Hardware Specification

i. Laptop

Table 4.8.1 Specifications of Laptop

Description	Specifications
Model	MacBook Pro 2018
Processor	2.6 GHz 6-Core Intel Core i7
Operating System	macOS Sonama 14.4
Graphic	Radeon Pro Vega 16 4GB Intel UHD Graphics 630 1536MB
Memory	32 GB 2400 MHz DDR4
Storage	Macintosh HD 500GB

ii. Desktop PC

Table 4.8.2 Specifications of Desktop PC

Description	Specifications
Model	Acer
Processor	Intel Core i5-4460 CPU @ 3.20GHz 3.20 GHz
Operating System	Windows 10 Pro
Graphic	Intel HD Graphics 4600
Memory	16,384 MB RAM

iii. Cisco Switch 2960

Table 4.8.3 Specification of Cisco Switch 2960

Description	Specifications
Model	2960-8TC-S
Forwarding Bandwidth	50Gbps
Flash Memory	64MB
Memory DRAM	128MB
Forwarding Rate	2.7 mpps
Fixed Lan Ports	8 Ethernet 10/100 ports and 1 dual-purpose uplink (dual-purpose uplink port has 1 10/100/1000 Ethernet port and 1 SFP-based Gigabit Ethernet port, 1 port active)

4.2.2 Software Components

i. PuTTY: A popular open-source terminal emulator, serial console, and network file transfer application. It supports various network protocols, including SSH, Telnet, rlogin, and raw socket connection. PuTTY is widely used for remote access to servers, network devices, and other computing resources.

ii. SNMP: It is a widely used protocol for monitoring and managing network devices and services, also widely supported by network devices, including routers, switches, servers, printers, and more. Many network management systems and monitoring tools rely on SNMP to collect and analyze data from network devices so that enables network administrators can monitor device performance, traffic statistics, resource utilization, and other important metrics.

iii. OPNsense: An open-source firewall with network intrusion detection system (NIDS) and intrusion prevention system (IPS), is highly configurable and can be deployed in various network environments. It is widely used for real-time traffic analysis and packet logging to detect and prevent network security threats.

Table 4.9.1 Specification of OPNsense

Description	Specifications
Model	Virtual Machine
Processor	2 Processors
Operating System	FreeBSD
Memory	8192MB
Storage	30.00GB
Version	24.1.9_4

iv. Kali Linux: A popular open-source Linux distribution designed for digital forensics and penetration testing. It is widely used by security professionals, ethical hackers, and penetration testers for assessing the security of networks, systems, and applications. It provides a convenient and powerful platform for conducting security testing and identifying vulnerabilities.

Table 4.9.2 Specification of Kali Linux

Description	Specifications
Model	Virtual Machine
Processor	2 Processors
Operating System	Debian
Memory	2048MB
Storage	80.09GB
Version	2024.1

v. Zabbix Server: An open-source monitoring software designed for monitoring the availability and performance of IT infrastructure components, including servers, network devices, services, and applications. It is highly scalable and can monitor thousands of devices and metrics in real-time.

Table 4.9.3 Specification of Zabbix Server

Description	Specifications
Model	Virtual Machine
Processor	4 Processors
Operating System	Red Hat
Memory	4096MB
Storage	10.00GB
Version	6.4

vi. Black-Arch Linux: Black-Arch Linux is a highly specialized Linux distribution focused on penetration testing and security research. It provides a wide range of tools covering various aspects of cybersecurity, making it a valuable resource for offensive security tasks.

Table 4.9.4 Specification of Black-Arch Linux

Description	Specifications
Model	Virtual Machine
Processor	2 Processors
Operating System	Arch Linux
Memory	8192MB
Storage	98.05GB

4.3 System Components Design and Interaction

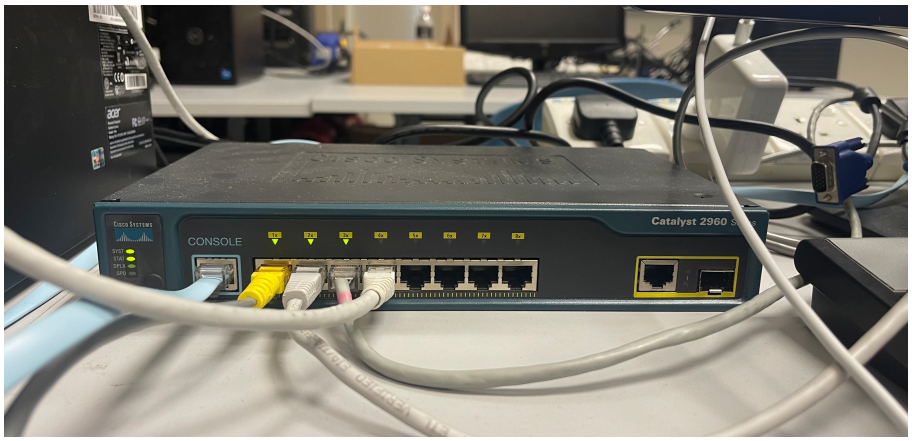


Figure 4.1 Switch Interface

From Figure 4.1, Interface Fa0/1 will connected to the internet, while interface Fa0/2 will set as port mirroring on interface Fa0/1

Figure 4.2 CPU Interface

In Figure 4.2, two network adapter interface need to be install physically. The top interface will connect to the switch interface Fa0/3 to access internet and the down interface will connect to the switch interface Fa0/2 to monitor the port mirroring.

Chapter 5

System Implementation

5.1 Software and Hardware

Before starting implement NIDAS, there are few software and hardware need to be install and setup

5.1.1 Software

1. PuTTY
2. Oracle VM Machine
3. OPNsense Firewall
4. Zabbix Server
5. Arch-Linux
6. Kali Linux
7. Telegram -Bot Father
8. Xampp

5.1.2 Hardware

1. Switch configuration and with SNMP services
2. Switch configuration with Port Mirroring

5.2 Hardware Setup

5.2.1 Setup and Configure Network Devices

1. Set Vlan1 IP address as 192.168.237.177/24, default-gateway as 192.168.237.254
2. Activate SNMP services with Read-Only community string

[snmp-server community 'Any Name' RO]

3. Enable SNMP traps by setting the host to which traps have to be sent

[snmp-server host 'Zabbix server ip addr' version 2c 'community name set on above']

4. Configure interface Fa0/2 as port mirroring of interface Fa0/1

[monitor session 1 source interface Fa0/1]

[monitor session 1 destination Fa0/2]

5. Exit the configuration mode and save the setting

[write memory]

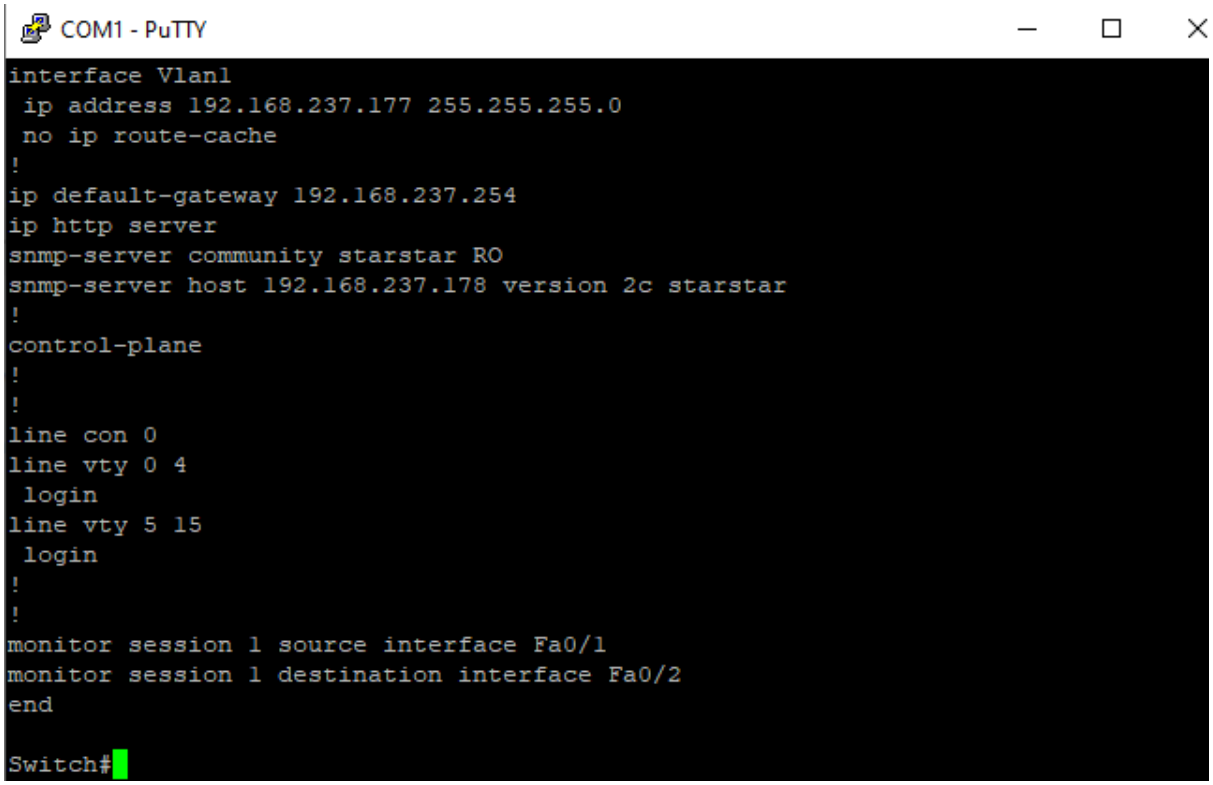


Figure 5.1.1 Switch Configuration

6. Ensure the port is in the same VLAN

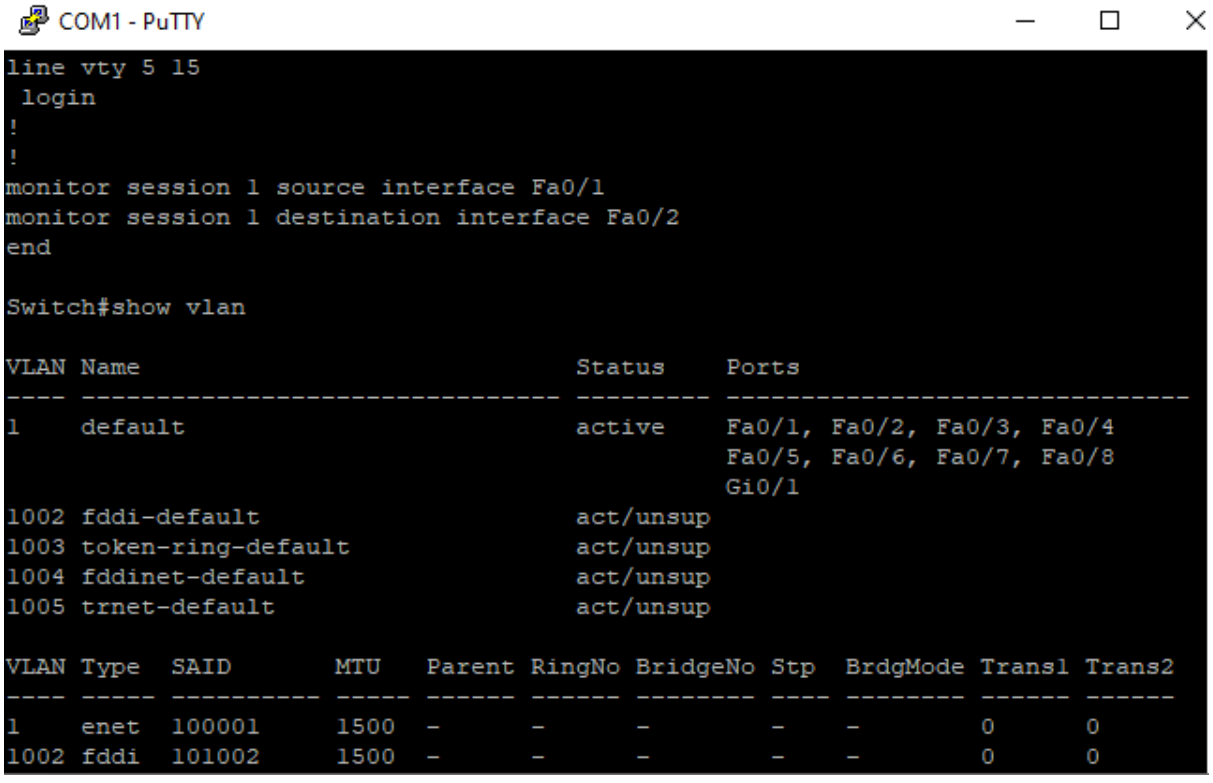


Figure 5.1.2 Switch VLAN

5.3 Software Setup

5.3.1 Install and Setup Zabbix Server

1. Go to the Zabbix website, choose Zabbix Appliance and select .ovf format download the Zabbix server.

2. Open Oracle VM VirtualBox manager, select add and find the Zabbix folder downloaded just now and select the ‘zabbix’ with format VirtualBox Machine.

3. Before turn on the Zabbix server, change the network adapter type to bridged network connection and start the machine.

4. Refer the Zabbix manual appliance, access into the Zabbix server as a root user.

5. Set a static IP address for Zabbix server, input command

[vi /etc/sysconfig/network-scripts/ifcfg-eth0] open file, press ‘insert’ key to change

Replace ‘BOOTPROTO=dhcp’ to ‘BOOTPROTO=none’

Add the following lines into the file below the script:

[IPADDR=<IP address of the appliance>]

[PREFIX=<CIDR prefix>]

[GATEWAY=<gateway IP address>]

[DNS1=<DNS server IP address if no then remove this line>]

Press ‘ESC’ and input ‘:wq’ to write and save the file.

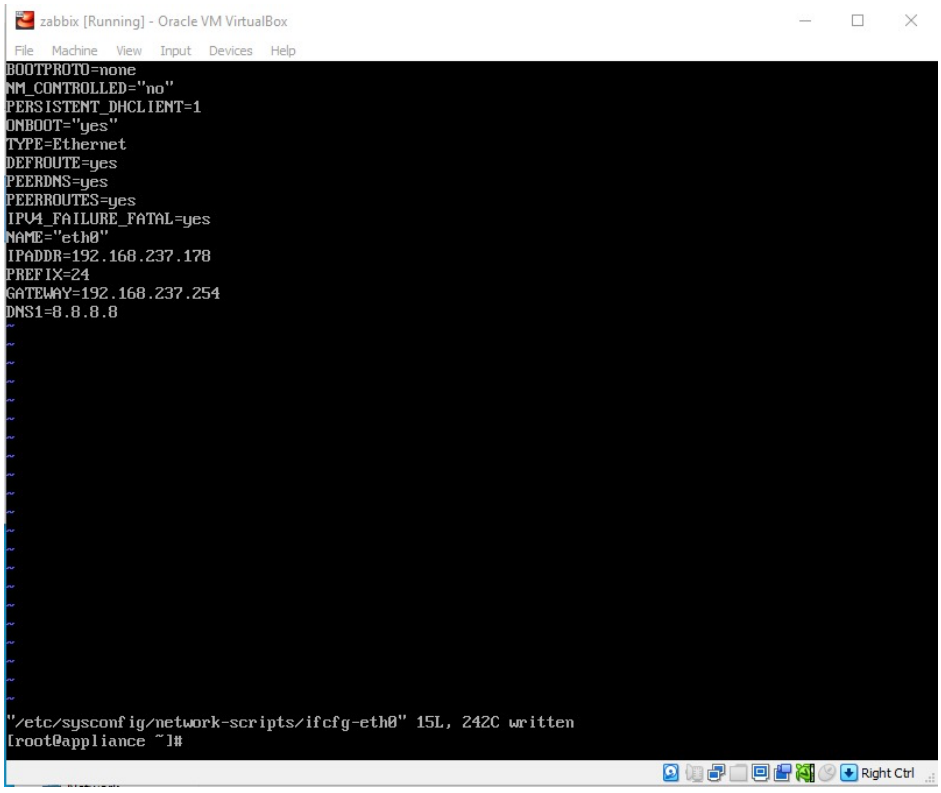


Figure 5.2.1 Zabbix Network Configuration

6. Run [systemctl restart network] to reboot the network.

7. Setup MySQL database user and password for Zabbix server, using the command *[mysql -u root -p]* and root password is blank

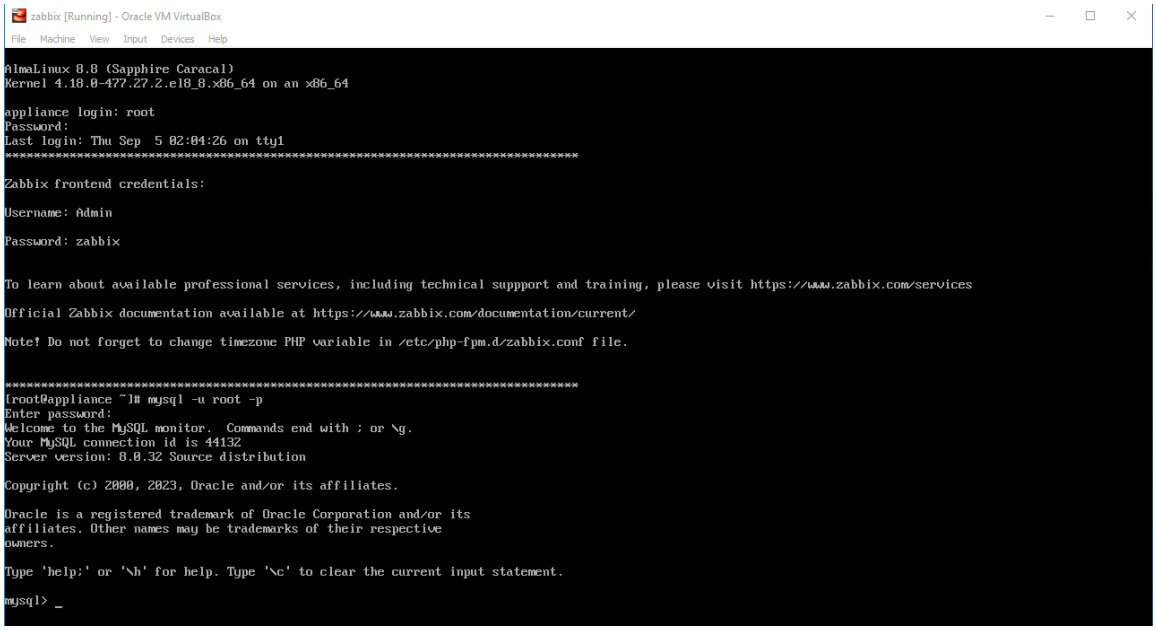


Figure 5.2.2 Zabbix MySQL

8. Create the user and password for Zabbix server to access zabbix databases in MySQL

[create user 'zabbix'@'localhost' identified by '<password>']

[create user 'zabbix1'@'localhost' identified by '<password>']

9. Grant the full privileges access for 'zabbix' and 'zabbix1' in MySQL

[grant all privileges on zabbix. to 'zabbix'@'localhost';]*

[grant all privileges on zabbix. to 'zabbix1'@'127.0.0.1';]*

10. Flush the privileges in MySQL

[flush privileges;]

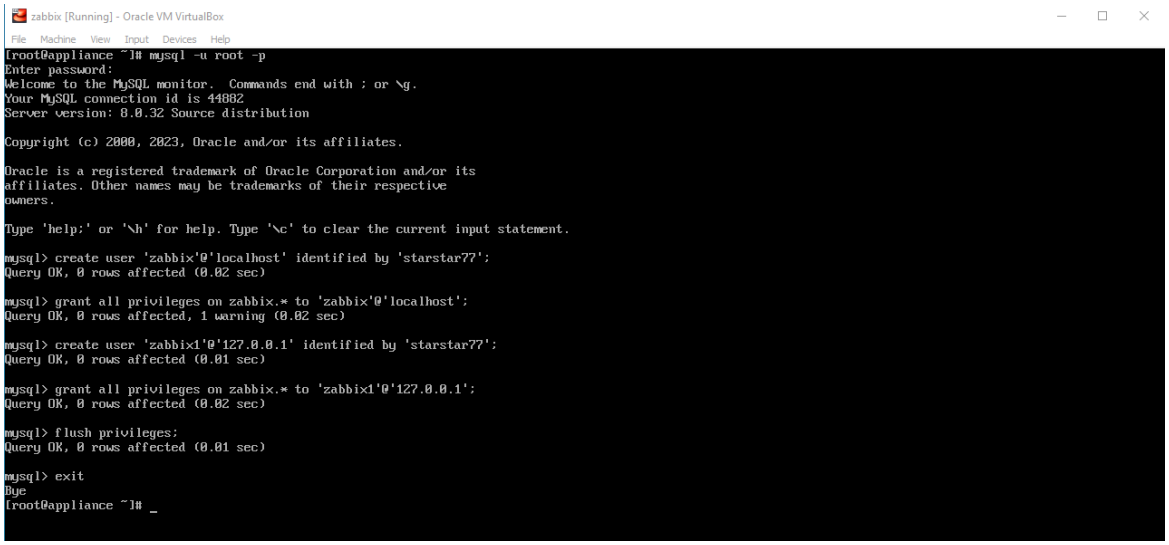


Figure 5.2.3 Zabbix MySQL Configuration

11. Open Zabbix server .conf file

[vi /etc/zabbix/zabbix_server.conf]

```
[root@appliance ~]# vi /etc/zabbix/zabbix_server.conf
```

12. After open the Zabbix server .conf file, input '/ListenIP' to search ListenIP is activate and set as 0.0.0.0

[ListenIP=0.0.0.0]

```
### Option: ListenIP
#       List of comma delimited IP addresses that the trapper should listen on.
#       Trapper will listen on all network interfaces if this parameter is missing.
#
# Mandatory: no
# Default:
ListenIP=0.0.0.0
```

13. Press 'Esc' and input '/StartTrappers' to activate the StartTrappers in Zabbix server .conf file

[StartTrappers=5]

```
### Option: StartTrappers
#       Number of pre-forked instances of trappers.
#       Trappers accept incoming connections from Zabbix sender, active agents and active proxies.
#       At least one trapper process must be running to display server availability and view queue
#       in the frontend.
#
# Mandatory: no
# Range: 0-1000
# Default:
StartTrappers=5
```

14. Press 'Esc', input '/' and search the DBHost, DBName, DBUser and DBPassword locate

15. Ensure DBHost and DBName is correct and activate, set DBUser and DBPassword with the user and password set in MySQL previously

[DBHost=localhost]

[DBName=zabbix]

[DBUser=zabbix]

[DBPassword=<password>]

16. Press ‘Esc’, input ‘:wq’ to write and save the file

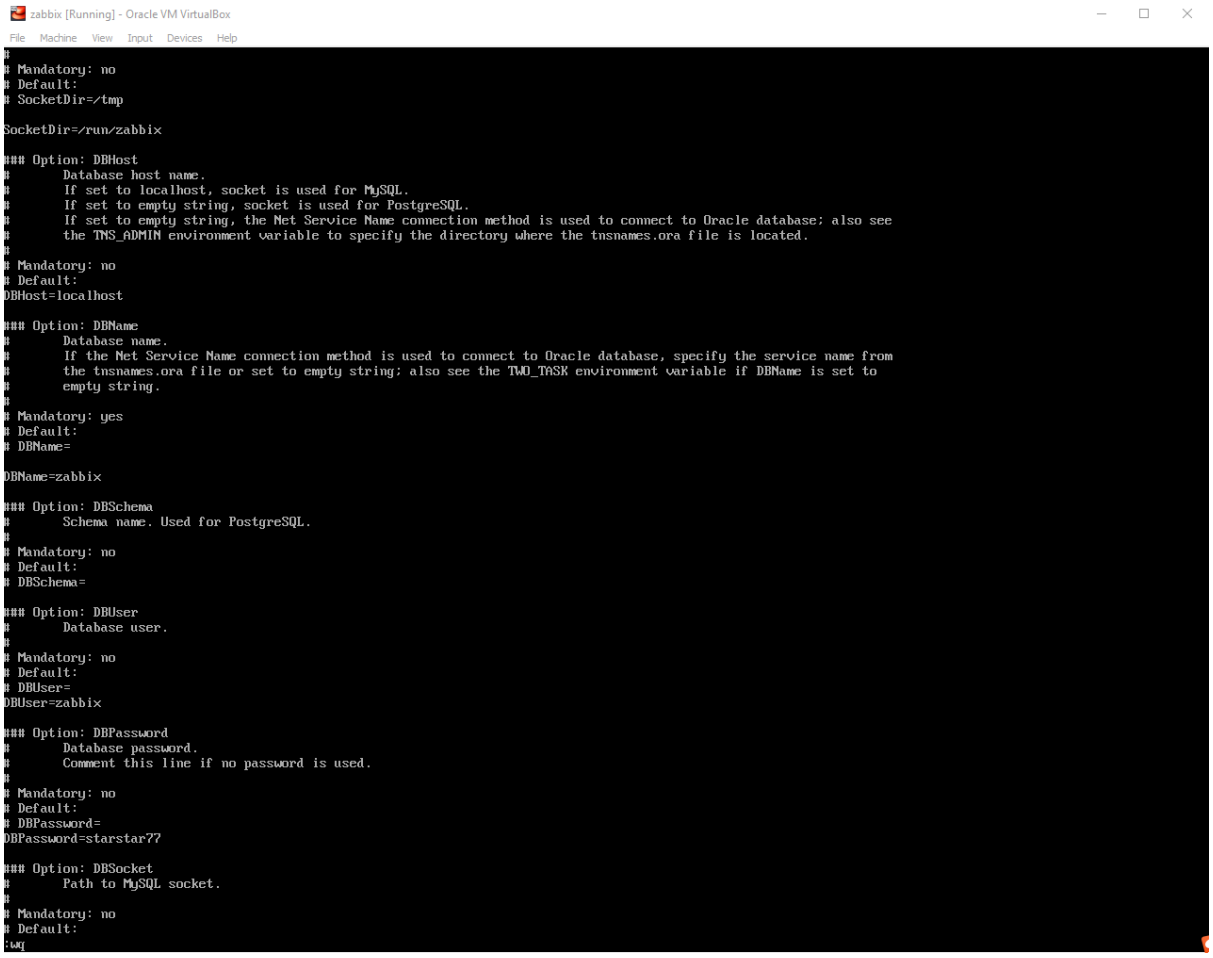
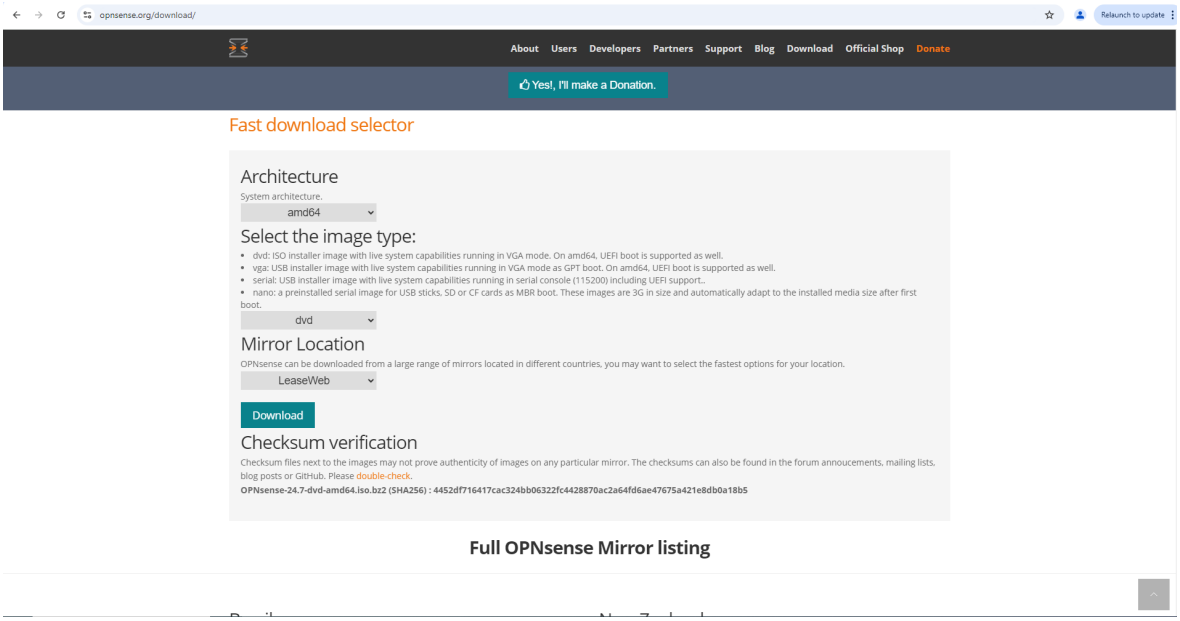


Figure 5.2.4 Zabbix Server Configuration

17. Input ‘reboot’ to reboot the zabbix server with the new configuration

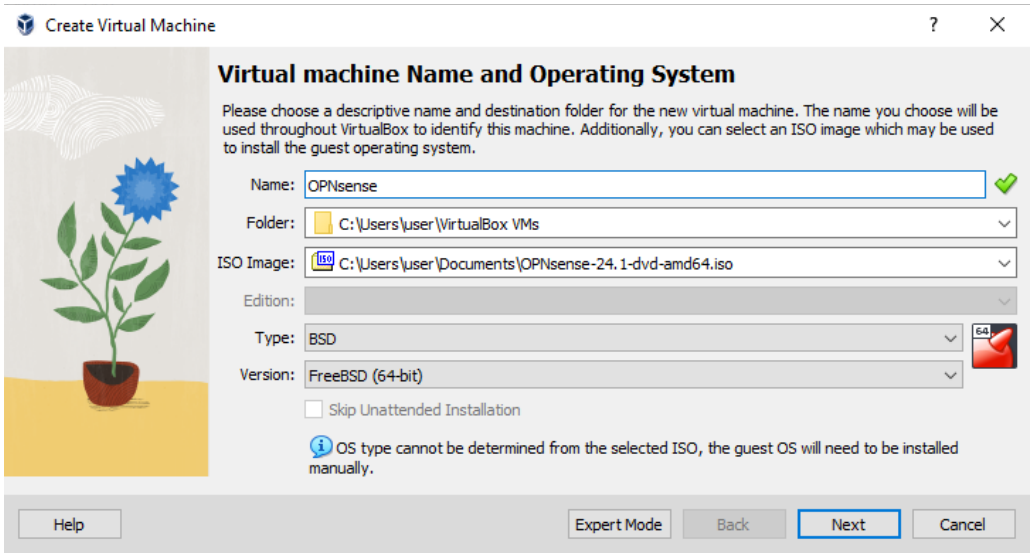
5.3.2 Install and Setup OPNsense

1. Go to OPNsense website download the .iso file, System Architecture select ‘amd64’, image type ‘dvd’ and Mirror Location can be anyway.



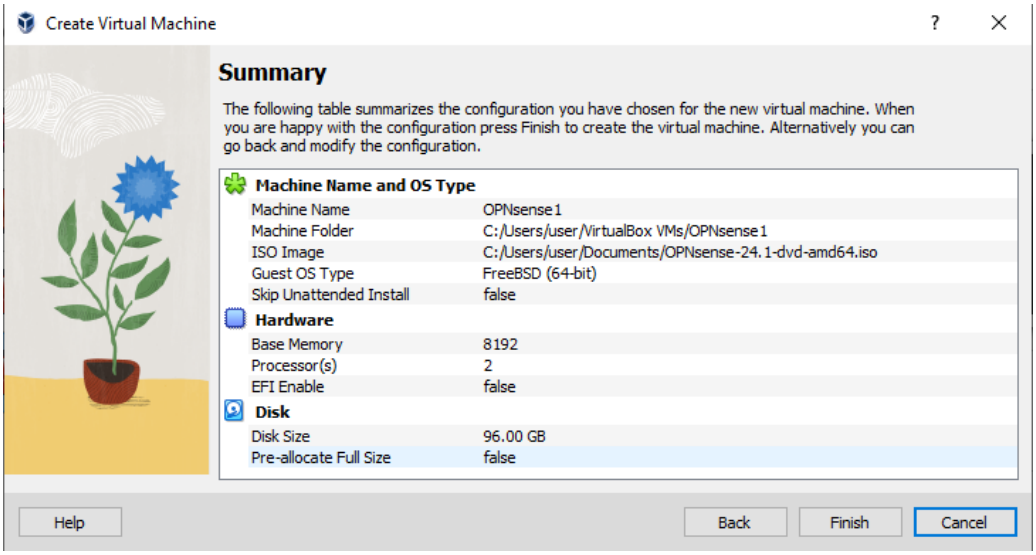
Full OPNsense Mirror listing

2. Open Oracle VM VirtualBox manager, select new and select ISO image downloaded just now, type select ‘BSD’ and change the version to ‘FreeBSD (64-bit)’, click Next.



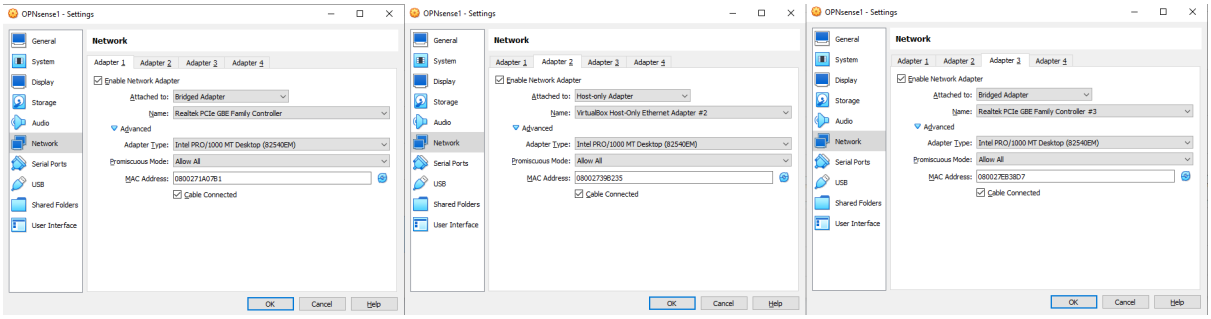
3. Base memory set on ‘8192 MB’ and Processors set on ‘2 CPUs, click Next.

4. Select ‘Create a Virtual Hard Disk Now’ and must set disk size on ‘96.00 GB’.

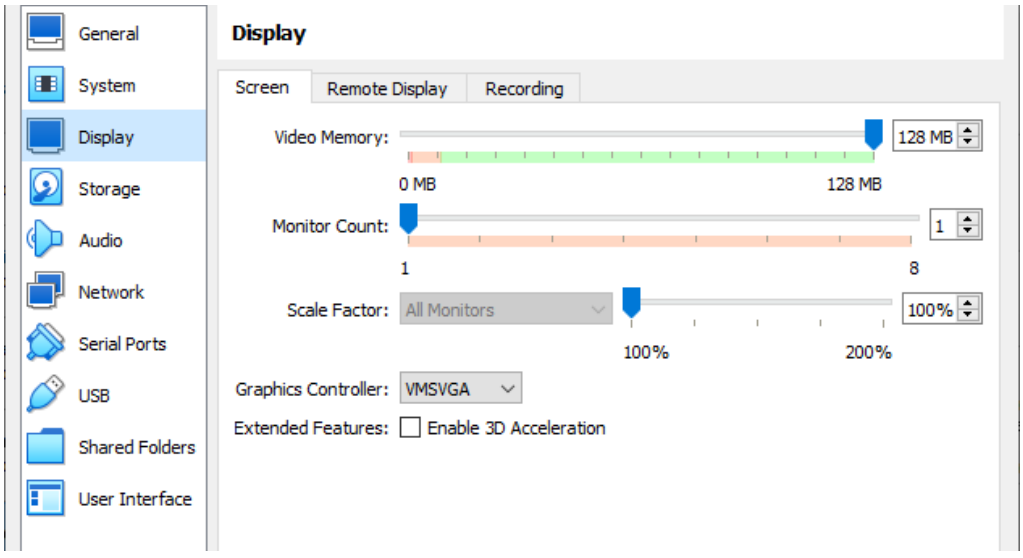


5. Go to OPNsense virtual box setting and add 2 more adapter.

6. Adapter 1 set as Bridged Adapter, Adapter 2 set as Host-only Adapter and Adapter 3 set as Bridged Adapter. All adapter 'Promiscuous Mode' set as 'Allow All'.



7. Go to Display, slide the Video Memory to '128 MB'.



8. Start OPNSense VM and wait for the login prompt to display.

9. Login with the username ‘installer’ and password ‘opnsense’

10. Select ‘Continue with default keymap’

11. Select ‘Install (UFS)’ and click OK

12. Select ‘ada0 <VBOX HARDDISK 10> (96GB)’ as a disk to continue

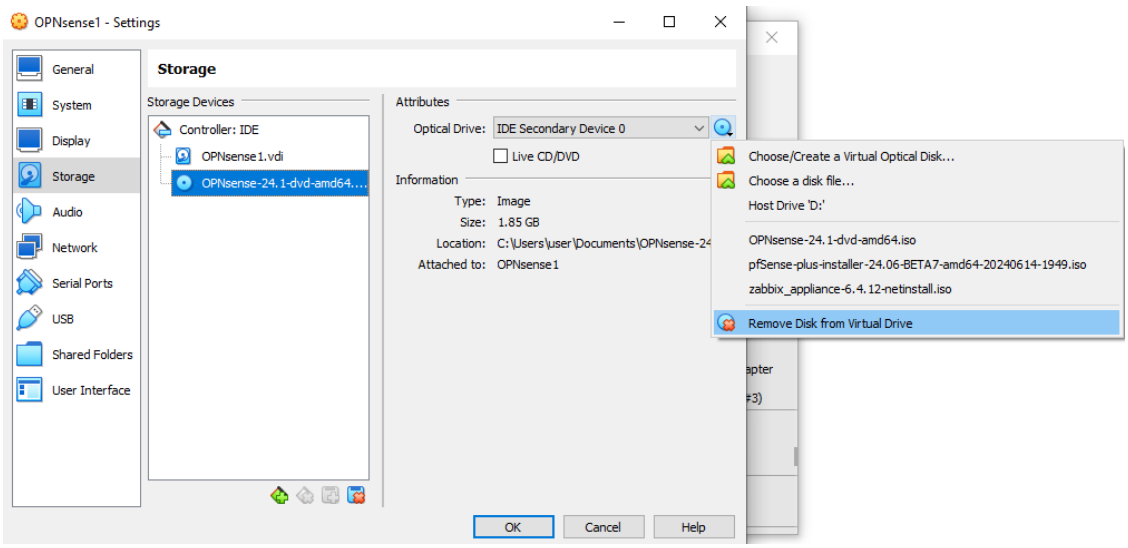
13. UFS Configuration select ‘Yes’ to continue with a recommended swap partition of size 8GB

14. Select Yes to confirm to destroy the current contents of the following disks: ada0

15. Select ‘Root Password’ to change and confirm the new root password (Optional)

16. Select ‘Complete Install’ to exit the installer to reboot the machine

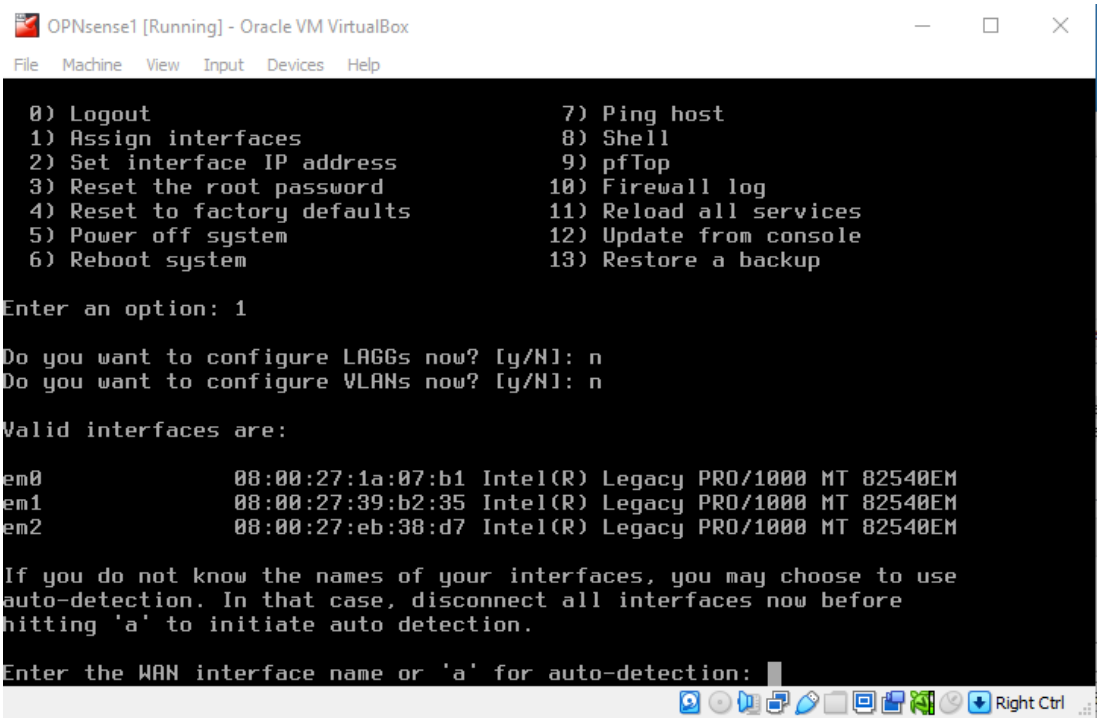
17. Before reboot the OPNsense, go to OPNsense virtual box setting ‘Storage’ and remove the disk from the virtual drive ‘IDE Secondary Device 0: [Optical Drive] OPNsense.iso’



18. Login to the prompt with username root and root password set earlier

19. At the menu, enter 1 as the option to assign interfaces

20. Enter ‘no’ for ‘Do you want to configure LAGGs now?’ and ‘Do you want to configure VLANs now?’



21. Enter ‘em0’ as WAN and ‘em1’ as LAN, leave blank for Optional section and enter ‘y’ to proceed

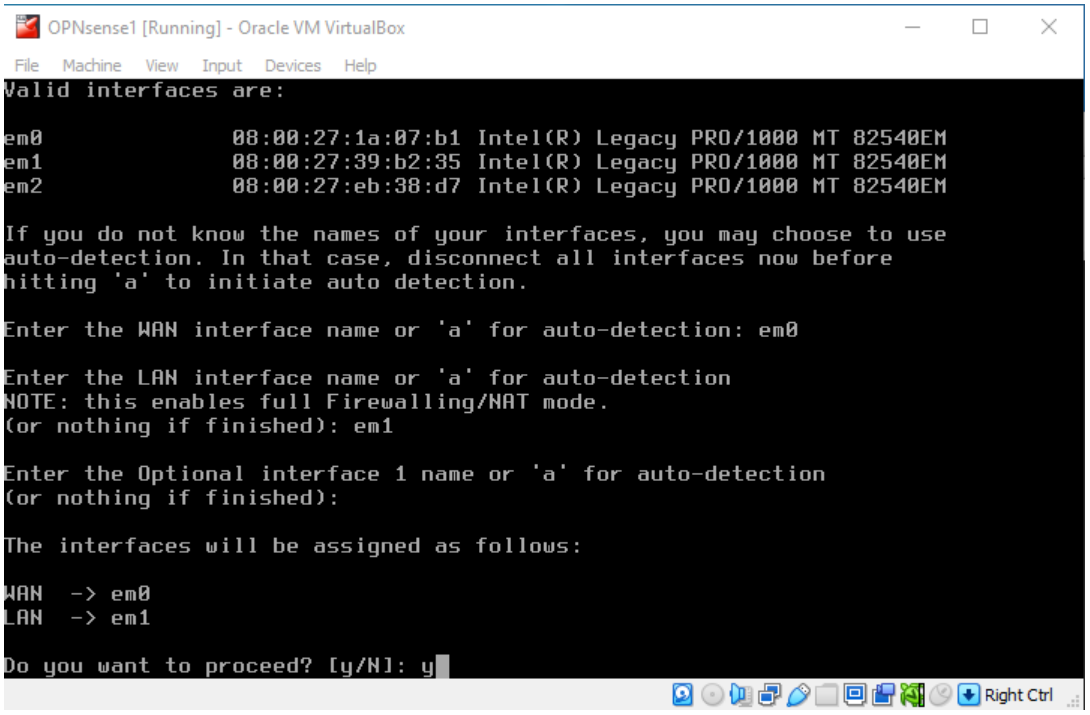
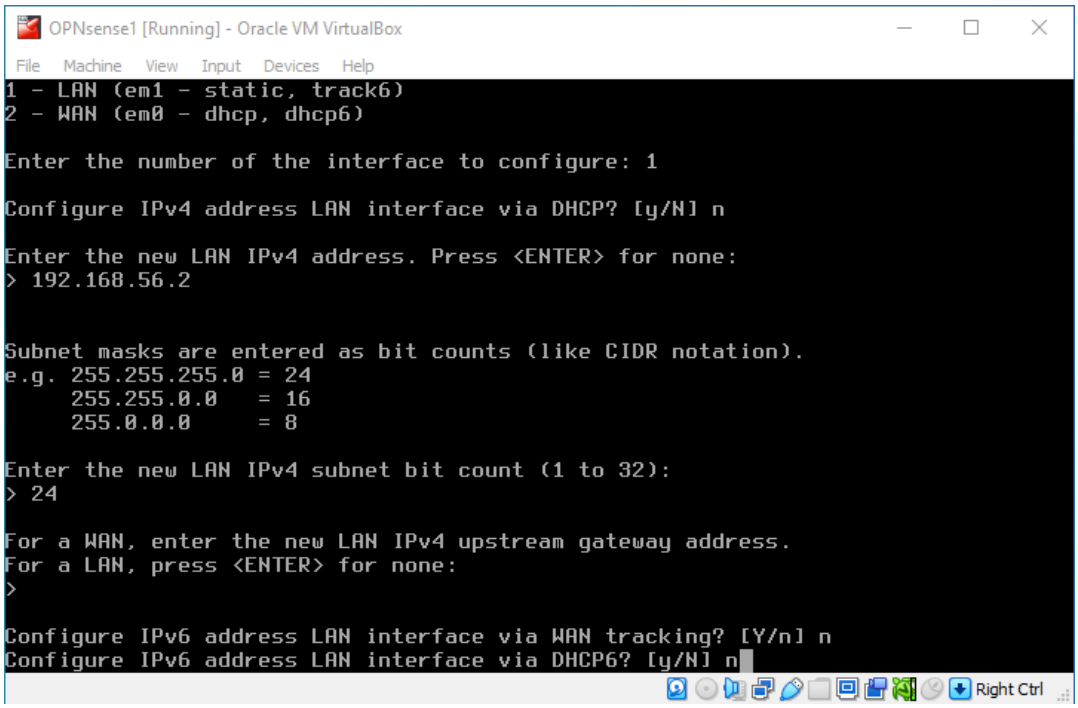


Figure 5.3.1 OPNsense Interface

22. At the menu, enter 2 as the option to set LAN interface IP address
23. Enter 1 to configure LAN interfaces
24. Enter ‘n’ in ‘Configure IPv4 address LAN interface via DHCP?’
25. Enter ‘192.168.56.2’ as new LAN IPv4 address and ‘24’ as new LAN IPv4 subnet
26. Enter ‘n’ in ‘Configure IPv6 address LAN interface via WAN tracking?’
27. Enter ‘n’ in ‘Configure IPv6 LAN interface via DHCP6?’



28. ‘Do you want to enable the DHCP server on LAN?’ is optional choice, if ‘y’ input the start and the end of IPv4 client address range
29. Enter ‘n’ to not to change the web GUI protocol from HTTPS to HTTP
30. Enter ‘y’ to generate a new self-signed web GUI certificate and again ‘y’ to restore web GUI access defaults

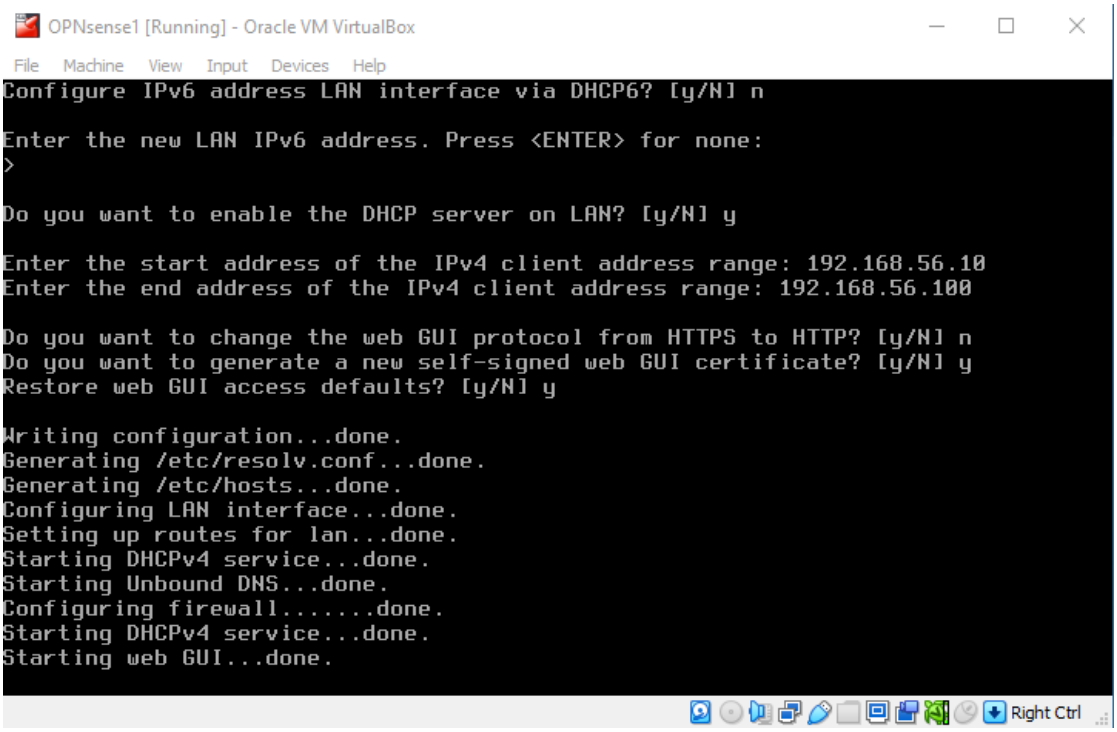
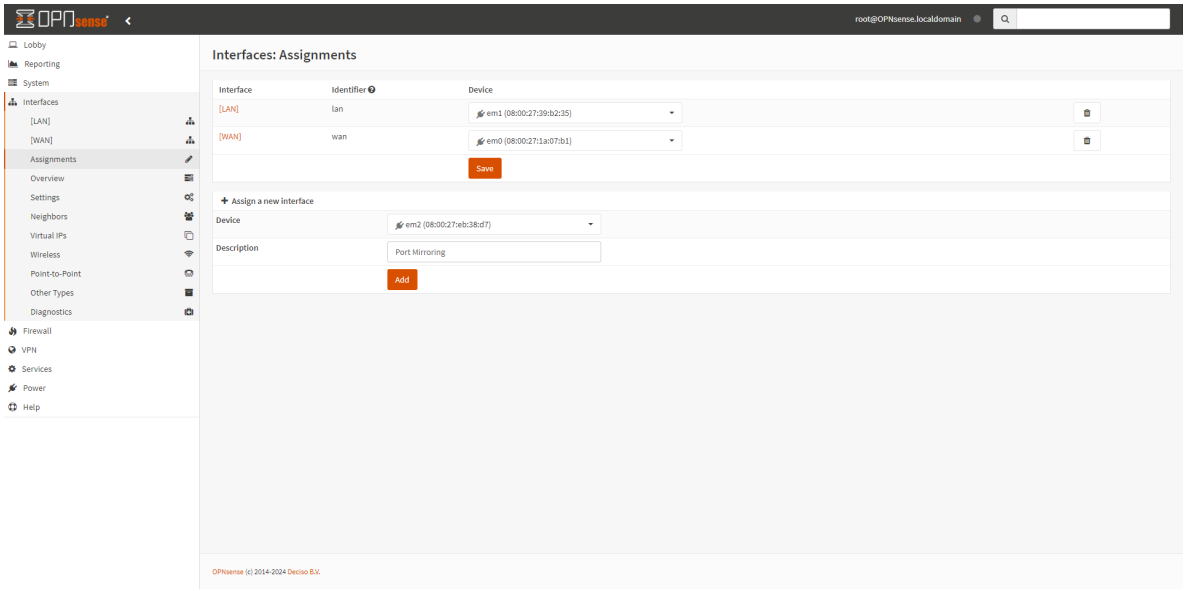


Figure 5.3.2 OPNsense LAN

31. Open the browser, enter the IP address set for LAN and login with user ‘root’
32. Go to the ‘Interfaces; Assignments’ in the menu bar
33. In ‘Assign a new interface’ section add ‘em2’ and name as Port Mirroring



34. Go to the [PortMirroring] section at Interfaces in menu bar, enable the interface and promiscuous mode then click save

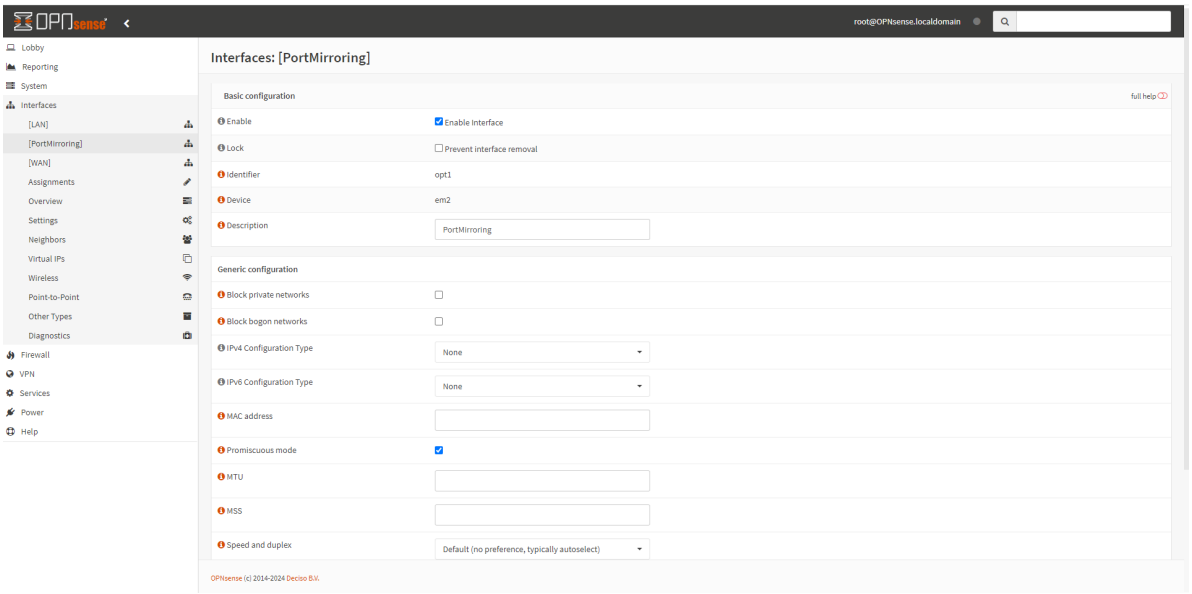


Figure 5.3.3 OPNsense Port Mirroring

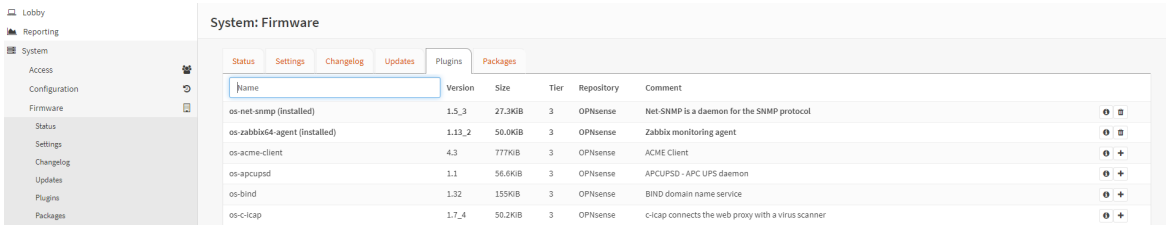
34. Go to ‘Power’ and reboot the system

5.4 System Setting

5.4.1 Install and Setup Plugin in OPNsense

1. Go to the ‘System; Firmware; Plugin’ section

2. Search ‘os-net-snmp’ and ‘os-zabbix64-agent’, install them



3. Go to the ‘Services; Net SNMP’ section

4. Enable SNMP service and the SNMP community ‘starstar’

5. Enable the Layer 3 Visibility and set the ListenIP as OPNsense ‘WanIP address’

6. Click Save button

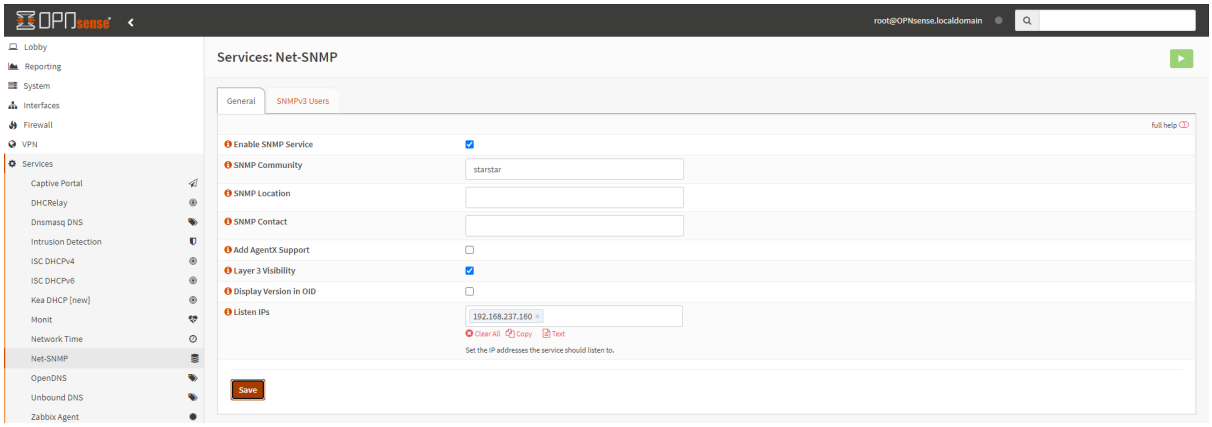


Figure 5.4.1 OPNsense SNMP

6. Go to the ‘Service; Zabbix Agent’ section

7. Enable Zabbix Agent service and setup the configuration

[Hostname: OPNsense]

[Listen Port: 10050]

[Listen IPs: 192.168.237.160]

[Source IP: 192.168.237.160]

[Zabbix Servers: 192.168.237.178]

Enable sudo root permissions

8. Click Apply

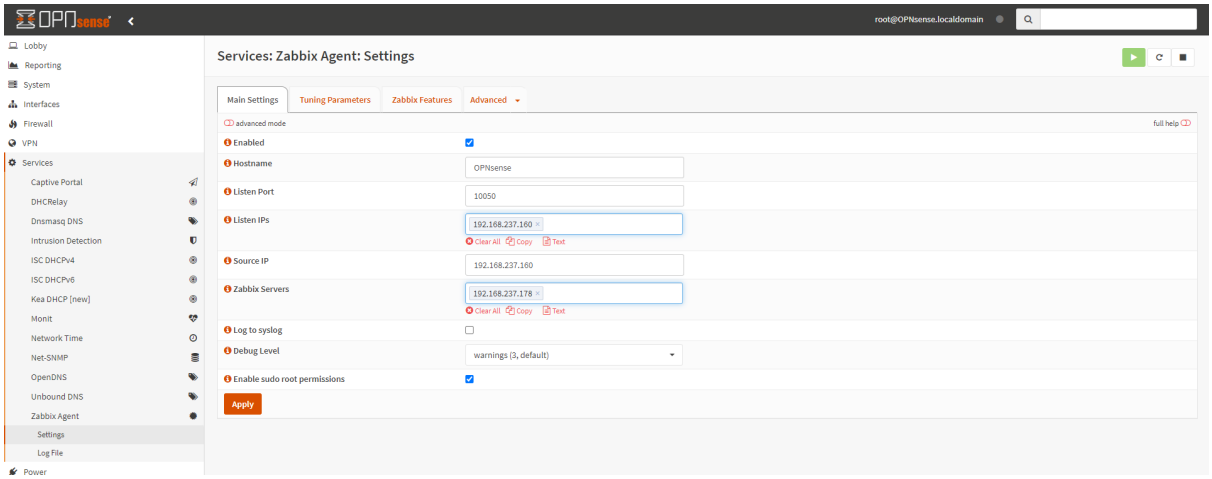


Figure 5.4.2 OPNsense Zabbix Agent

9. Go to ‘Service; Intrusion Detection; Administration’ section

10. In setting section enable Intrusion Detection service and syslog alerts

11. Pattern matcher set as default and Interfaces choose PortMirroring

12. Rotate log set as Daily and saves logs set as 4

13. Click Apply

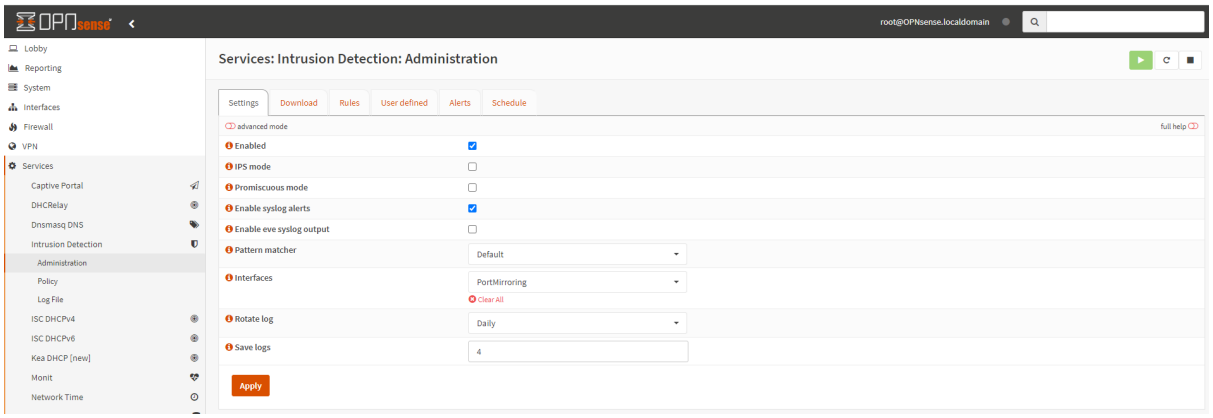


Figure 5.4.3 OPNsense Intrusion Detection

14. In download section, select all rulesets and download

15. After downloaded rulesets, select all rulesets and click ‘Enable selected’

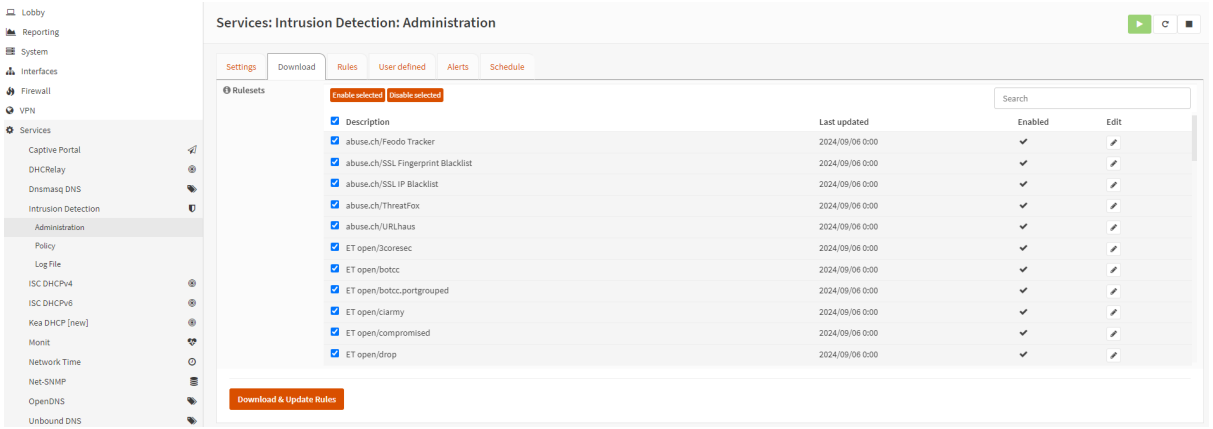
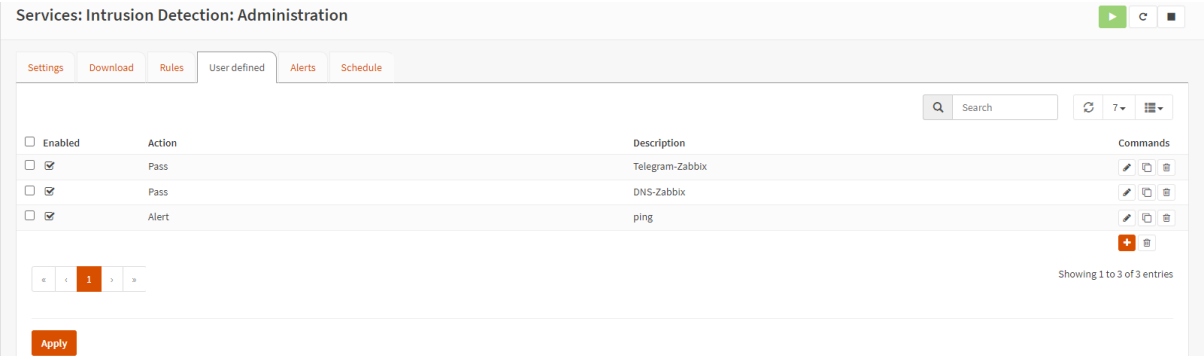


Figure 5.4.4 OPNsense Rulesets

16. Go to User defined section and select add rules action



17. Setup the action rules for Zabbix-DNS to pass through IDS

[Source IP: 192.168.237.178]

[Destination IP: 8.8.8.8]

[Action: Pass]

[Description: Zabbix-DNS]

18. Enable the rule set and click Save

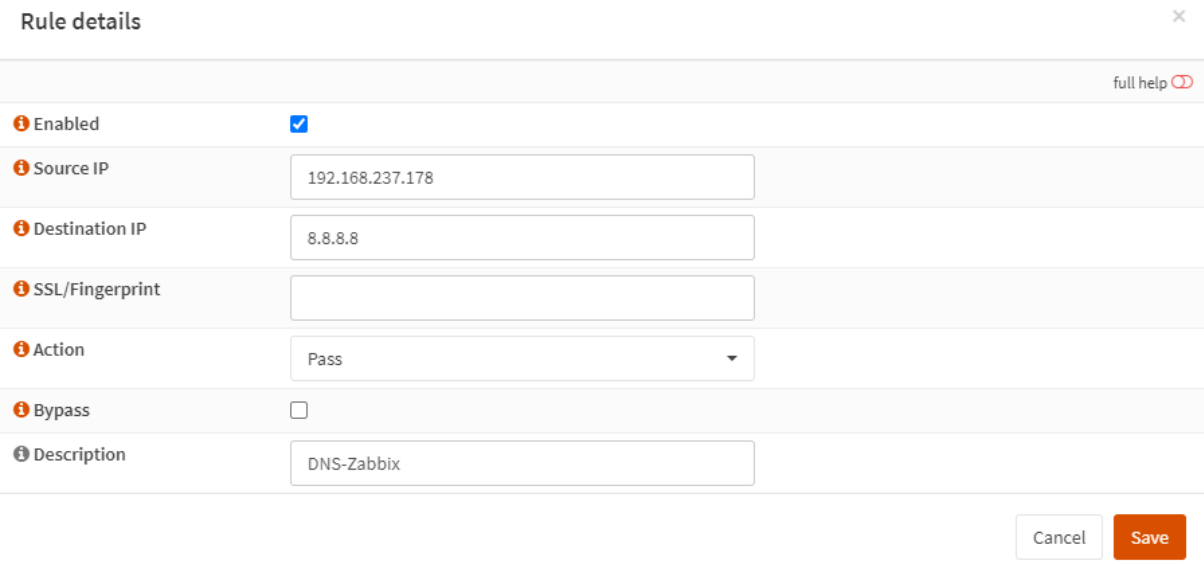


Figure 5.4.5 OPNsense Zabbix-DNS rule set

19. Setup the action rules for Zabbix-Telegram to pass through IDS

[Source IP: 192.168.237.178]

[Destination IP: 149.154.167.220]

[Action: Pass]

[Description: Zabbix-Telegram]

20. Enable the rule set and click Save

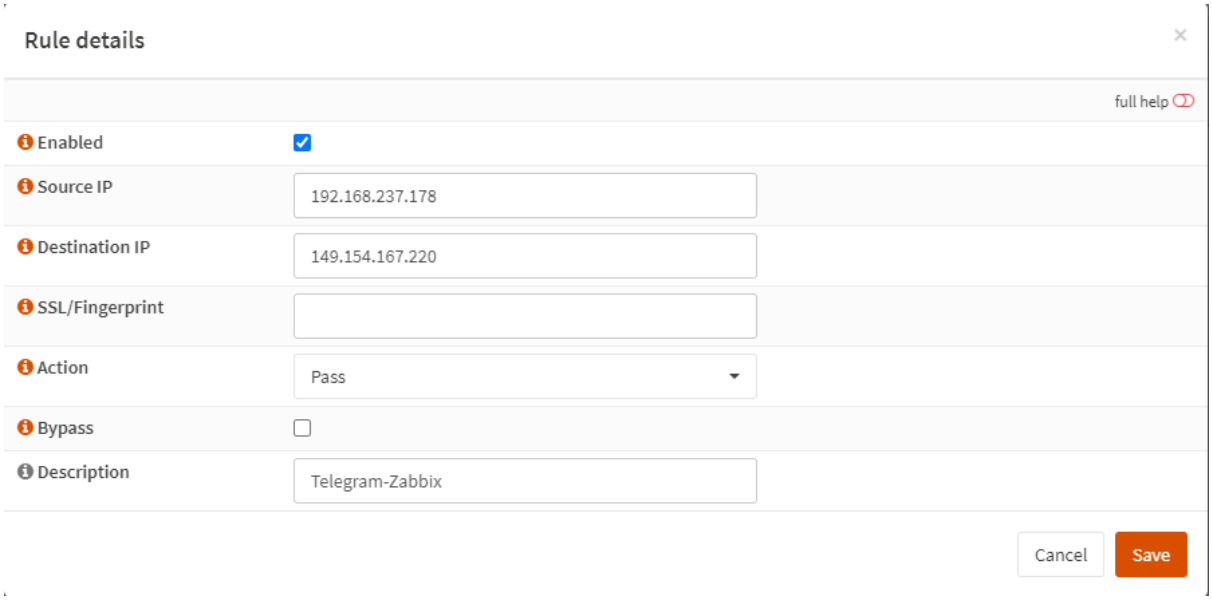


Figure 5.4.6 OPNsense Zabbix-Telegram rule set

5.4.2 Setup OPNsense Firewall TCP and UDP Rule

1. Go to the 'Firewall; Rules; Floating' section

2. Add new rules

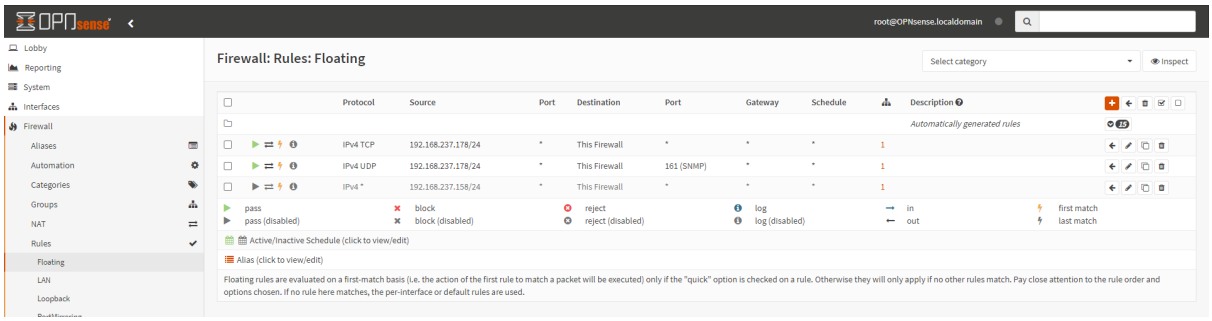


Figure 5.4.7 OPNsense Firewall TCP and UDP rules

3. Preconfig the rules to establish connection between Zabbix server and OPNsense Zabbix Agent

[Action: Pass]

[Quick: Apply the action immediately on match]

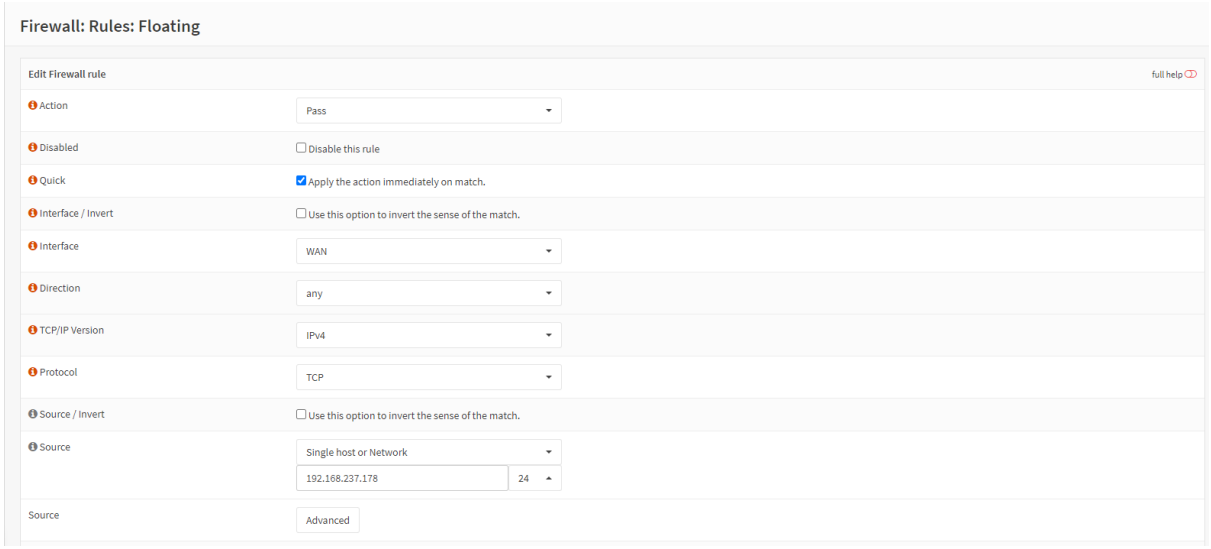
[Interface: WAN]

[Direction: any]

[TCP/IP Version: IPv4]

[Protocol: TCP]

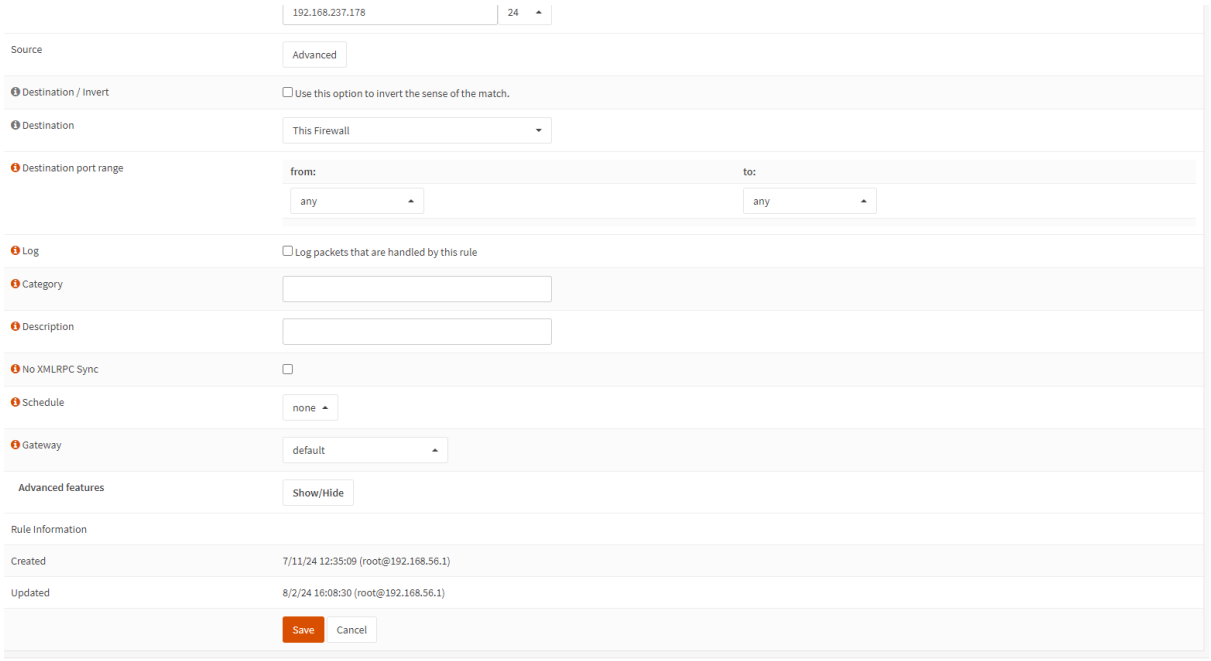
[Source: Single host or Network, 192.168.237.178/24 (Zabbix Server IP address)]



[Destination: This Firewall]

[Destination port range: from 'any' to 'any']

4. Click Save



5. Preconfig the rules to establish connection between Zabbix server and OPNsense SNMP service

[Action: Pass]

[Quick: Apply the action immediately on match]

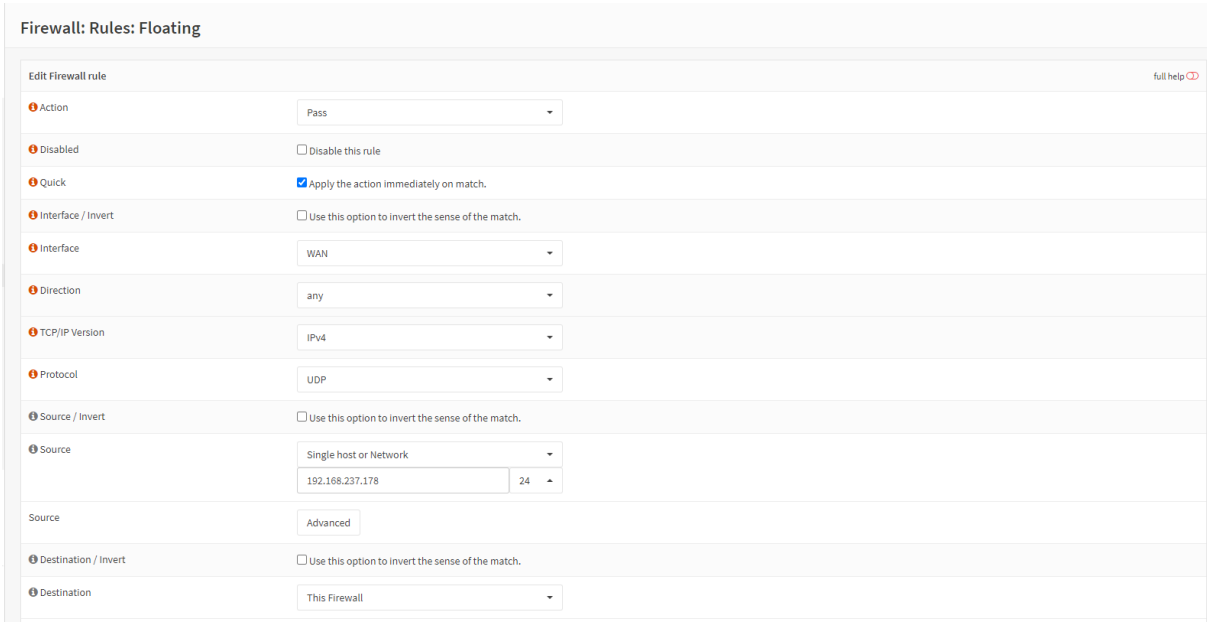
[Interface: WAN]

[Direction: any]

[TCP/IP Version: IPv4]

[Protocol: UDP]

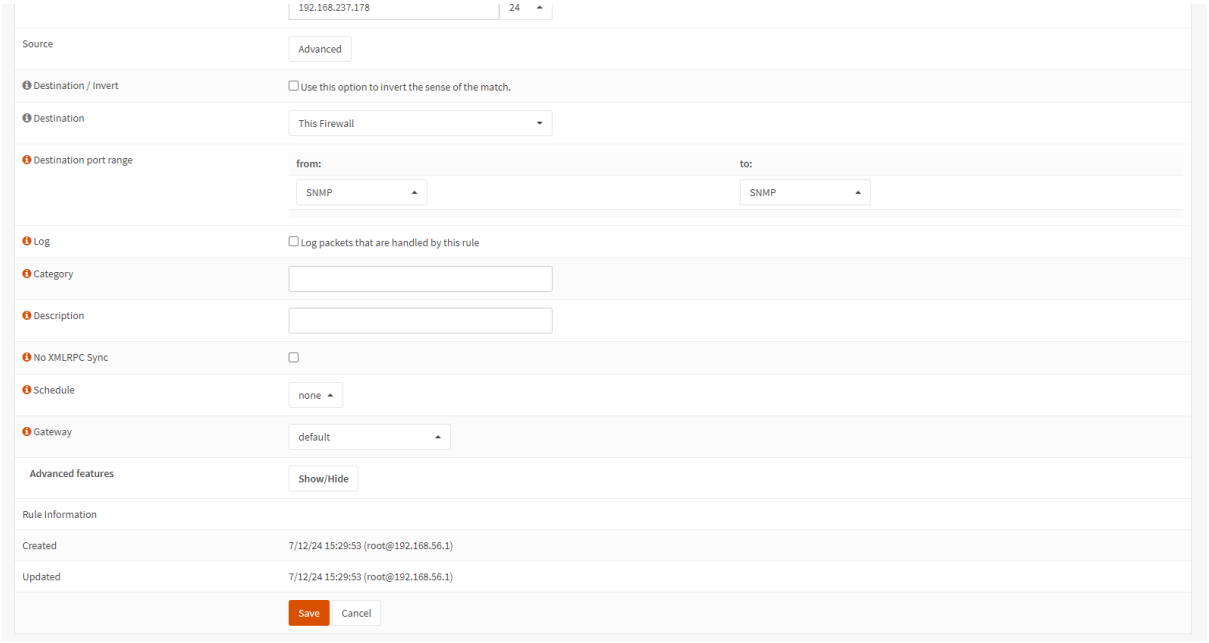
[Source: Single host or Network, 192.168.237.178/24 (Zabbix Server IP address)]



[Destination: This Firewall]

[Destination port range: from 'SNMP' to 'SNMP']

6. Click Save



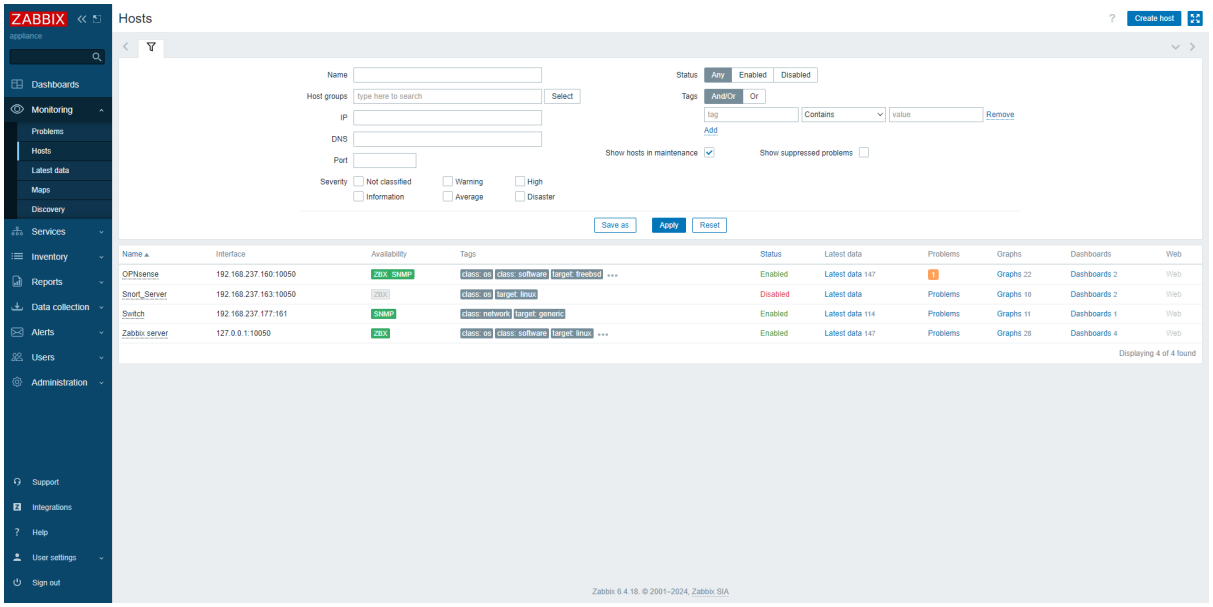
5.5 System Configuration

5.5.1 Monitoring OPNsense by using Zabbix Agent and SNMP

1. Open the browser and input the Zabbix server IP address that has been set in 5.3.1 to open the front end of Zabbix server

2. Login with Username ‘Admin’ and Password ‘zabbix’

3. Once access in the front end, select Monitoring section in the menu and click on host



4. Select Create host, put the host name and select templates find ‘FreeBSD by Zabbix agent’ and ‘OPNsense by SNMP’, Host Groups select ‘Virtual machines’

5. Add Agent as an interfaces, input OPNsense WAN IP address and make sure port is 10050

6. Add SNMP as an interfaces, input OPNsense WAN IP address and make sure port is 161. Select SNMP version ‘SNMPv2’ and input the SNMP community as ‘starstar’ set in 5.5 Install and Setup Plugin in OPNsense

7. Click add and the host is created, check the Availability of OPNsense server and if the status is available now the Zabbix can monitor the OPNsense

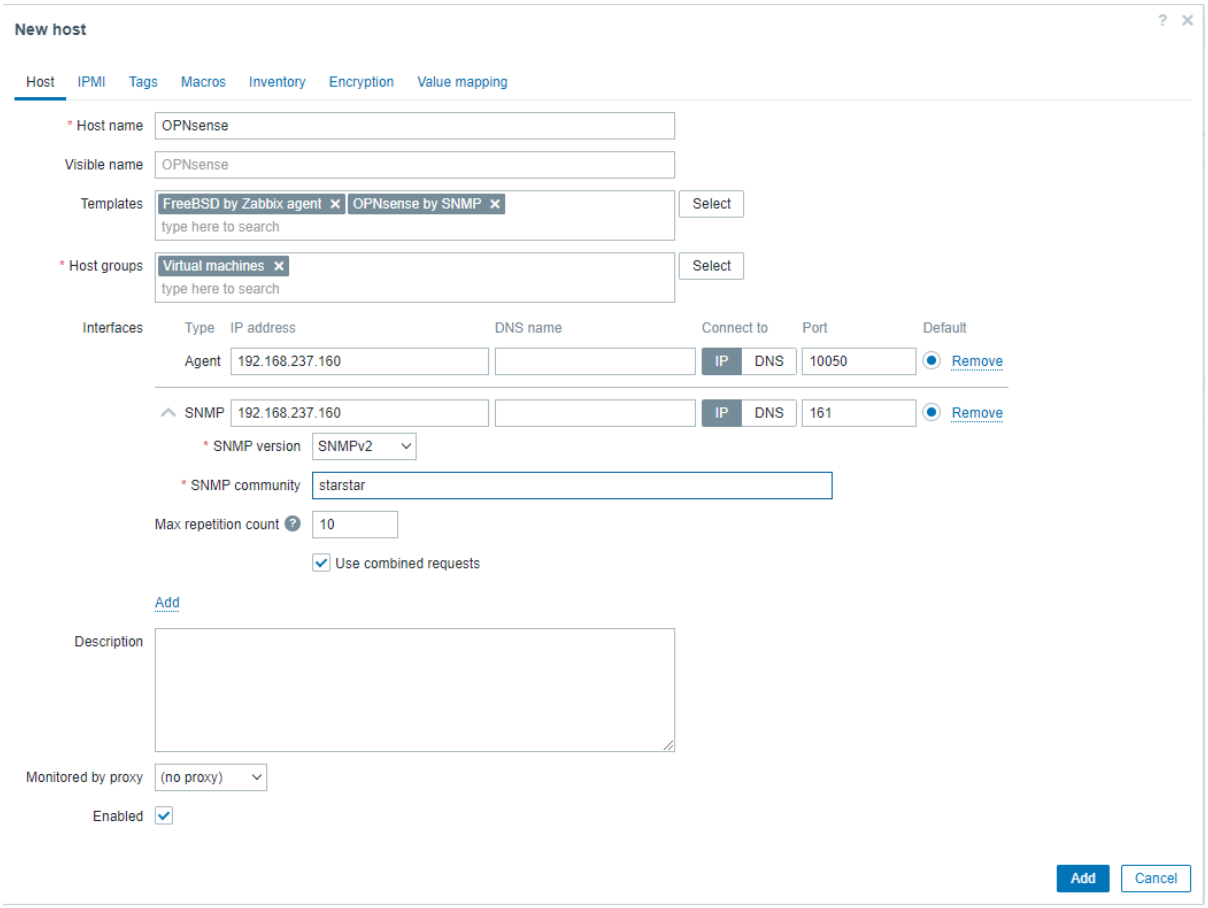


Figure 5.5.1 Create OPNsense Host in Zabbix

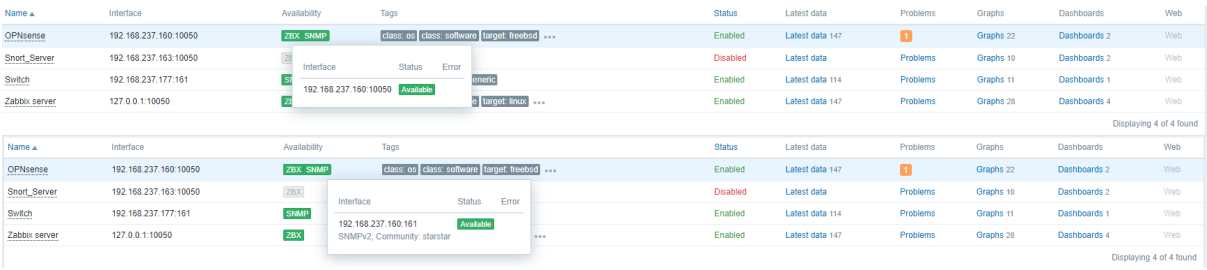


Figure 5.5.2 OPNsense Availablity Status

5.5.2 Monitoring switch by using Zabbix SNMP

1. Select ‘Create Host’ to create host ‘Switch’

2. Put the host name and select templates find ‘Network Generic Device by SNMP’, Host Groups select ‘Template/Network devices’.

3. Add SNMP as an interfaces, input switch IP address and make sure port is 161. Select SNMP version ‘SNMPv2’ and SNMP community input ‘starstar’ that configure in switch.

4. Click add and the host is created, check the Availability of switch and if the status is available now the Zabbix can monitor the switch.

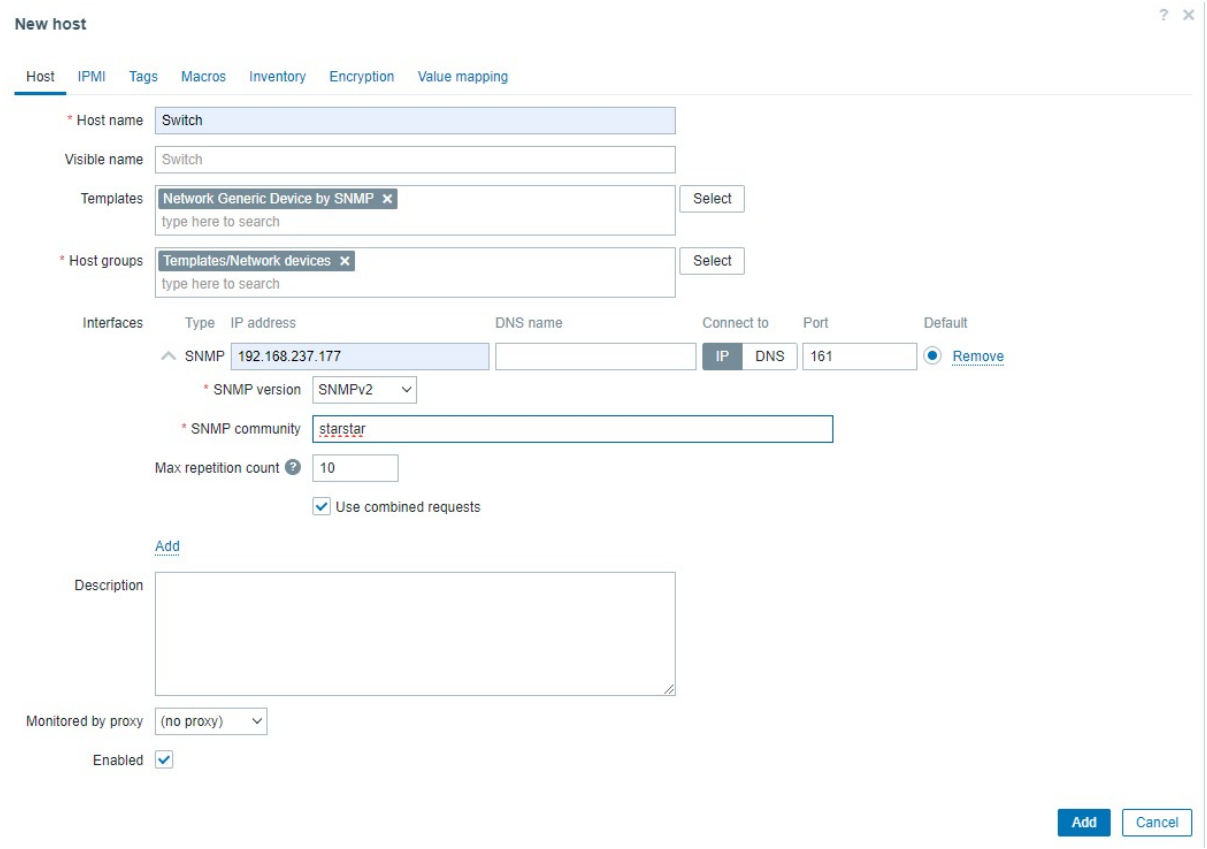


Figure 5.5.3 Create Switch Host in Zabbix

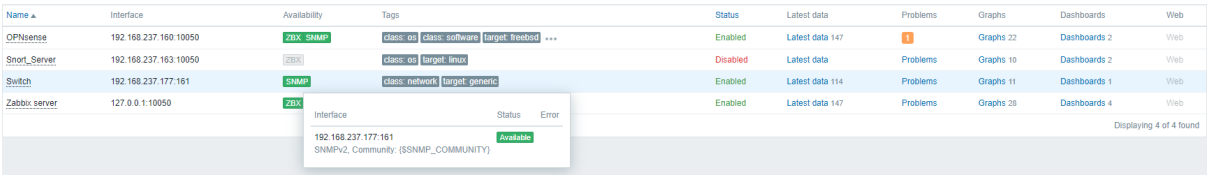


Figure 5.5.4 Switch Availability Status

5.5.3 Monitoring OPNsense IDS by using Zabbix

1. Login as root in OPNsense CLI and menu select 8 enter into Shell mode

2. Create a python script file in /usr/local/bin with using the command

[vim /usr/local/bin/ids.py]

```
|                                             |         @@@@          @@@@
| Website:      https://opnsense.org/         |          @@@\\\    ///@@@
| Handbook:     https://docs.opnsense.org/    |        ))))))))    ((((((((
| Forums:       https://forum.opnsense.org/   |          @@@///    \\\@@@
| Code:         https://github.com/opnsense   |         @@@@          @@@@
| Twitter:      https://twitter.com/opnsense  |          @@@@@@@@@@@@@@@@
-----------------------------------------------

*** OPNsense.localdomain: OPNsense 24.1.9_4 ***

 LAN (em1)       -> v4: 192.168.56.2/24
 PortMirroring (em2) ->
 WAN (em0)       -> v4/DHCP4: 192.168.237.160/24

  0) Logout                              7) Ping host
  1) Assign interfaces                   8) Shell
  2) Set interface IP address            9) pfTop
  3) Reset the root password            10) Firewall log
  4) Reset to factory defaults          11) Reload all services
  5) Power off system                   12) Update from console
  6) Reboot system                      13) Restore a backup

Enter an option: 8

root@OPNsense:~ # vim /usr/local/bin/ids.py
```

3. Write the script to send the ids log file data to Zabbix server, refer from the Appendix A

4. After complete the scripts, press ‘Esc’ and input ‘:wq’ to save the file

```
 1 #!/usr/bin/env python3
 2
 3 import os
 4 #import subprocess
 5 import time
 6 from datetime import datetime
 7
 8 position_file = '/etc/last_read_position.txt'
 9
10 current_date = datetime.now().strftime('%Y%m%d')
11 log_file = f'/var/log/suricata/suricata_{current_date}.log'
12
13 zabbix_server='192.168.237.178'
14 hostname='OPNsense'
15 key ='ids.detect'
16 #zabbix_user = 'zabbix'
17
18 #print(f"Checking log file at: {log_file}")
19
20 if os.path.exists(log_file):
21         last_read_position = 0
22
23         last_read_date = ""
24         if os.path.exists(position_file):
```

5. Create a bash script file in /usr/local/bin with using the command

[vim /usr/local/bin/start_idstimer.sh]

```
root@OPNsense:~ # vim /usr/local/bin/start_idstimer.sh
```

6. Write the script to automate run the python script 24 hours, refer from Appendix B

7. After complete the scripts, press 'Esc' and input ':wq' to save the file

```
#!/bin/sh
# Sleep for 30 seconds to ensure all services are up

sleep 30

PIDFILE="/tmp/ids.pid"

while true; do
        if [ -f $PIDFILE ] && kill -0 $(cat $PIDFILE) 2>/dev/null; then
                sleep 0.5
                continue
        fi


        /usr/local/bin/python3 /usr/local/bin/ids.py &
        echo $! > $PIDFILE


        sleep 0.5
done
~
~
~
~
"/usr/local/bin/start_idstimer.sh" 20L, 293B
```

8. Set the file full execute permission for both scripts in OPNsense by using the command

[chmod +x /usr/local/bin/ids.py]

[chmod +x /usr/local/bin/start_idstimer.sh]

```
root@OPNsense:/ # chmod +x /usr/local/bin/ids.py
root@OPNsense:/ # chmod +x /usr/local/bin/start_idstimer.sh
root@OPNsense:/ # ls -l /usr/local/bin/ids.py
-rwxr-xr-x  1 zabbix  zabbix  1600 Sep  6 23:19 /usr/local/bin/ids.py
root@OPNsense:/ # ls -l /usr/local/bin/start_idstimer.sh
-rwxr-xr-x  1 root  wheel  293 Sep  4 13:18 /usr/local/bin/start_idstimer.sh
root@OPNsense:/ #
```

9. Using crontab service to run the bash script when every time OPNsense is up, open the crontab file by using the command

[vim /etc/crontab]

```
root@OPNsense:/ # vim /etc/crontab
```

10. Add the command in crontab file

[@reboot root /usr/local/bin/start_idstimer.sh]

11. Press 'Esc' and input ':wq' to save the file

```
 1 # /etc/crontab - root's crontab for FreeBSD
 2 #
 3 #
 4 SHELL=/bin/sh
 5 PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
 6 #
 7 #minute hour    mday    month   wday    who     command
 8 #
 9 # Save some entropy so that /dev/random can re-seed on boot.
10 */11    *       *       *       *       operator /usr/libexec/save-entropy
11 #
12 # Rotate log files every hour, if necessary.
13 0       *       *       *       *       root    newsyslog
14 #
15 # Perform daily/weekly/monthly maintenance.
16 1       3       *       *       *       root    periodic daily
17 15      4       *       *       6       root    periodic weekly
18 30      5       1       *       *       root    periodic monthly
19 #
20 # Adjust the time zone if the CMOS clock keeps local time, as opposed to
21 # UTC time.  See adjkerntz(8) for details.
22 1,31    0-5     *       *       *       root    adjkerntz -a
23 @reboot root /usr/local/bin/start_idstimer.sh
24
:wq
```

12. Input 'Reboot' to reboot the OPNsense server

13. Login as root in Zabbix server CLI

14. Create a python scripts file in /usr/lib/zabbix/externalscripts with the command

[vi /usr/lib/zabbix/externalscripts/threats.py]

15. Refer from Appendix C, write the script to calculate the threats data per seconds from the TABLE 'history_log' in MySQL zabbix database based on the key 'clock' and 'value'

16. Ensure the script 'db_config', user, password and the host is the second preset user in MySQL database at 5.3 Install and Setup zabbix server

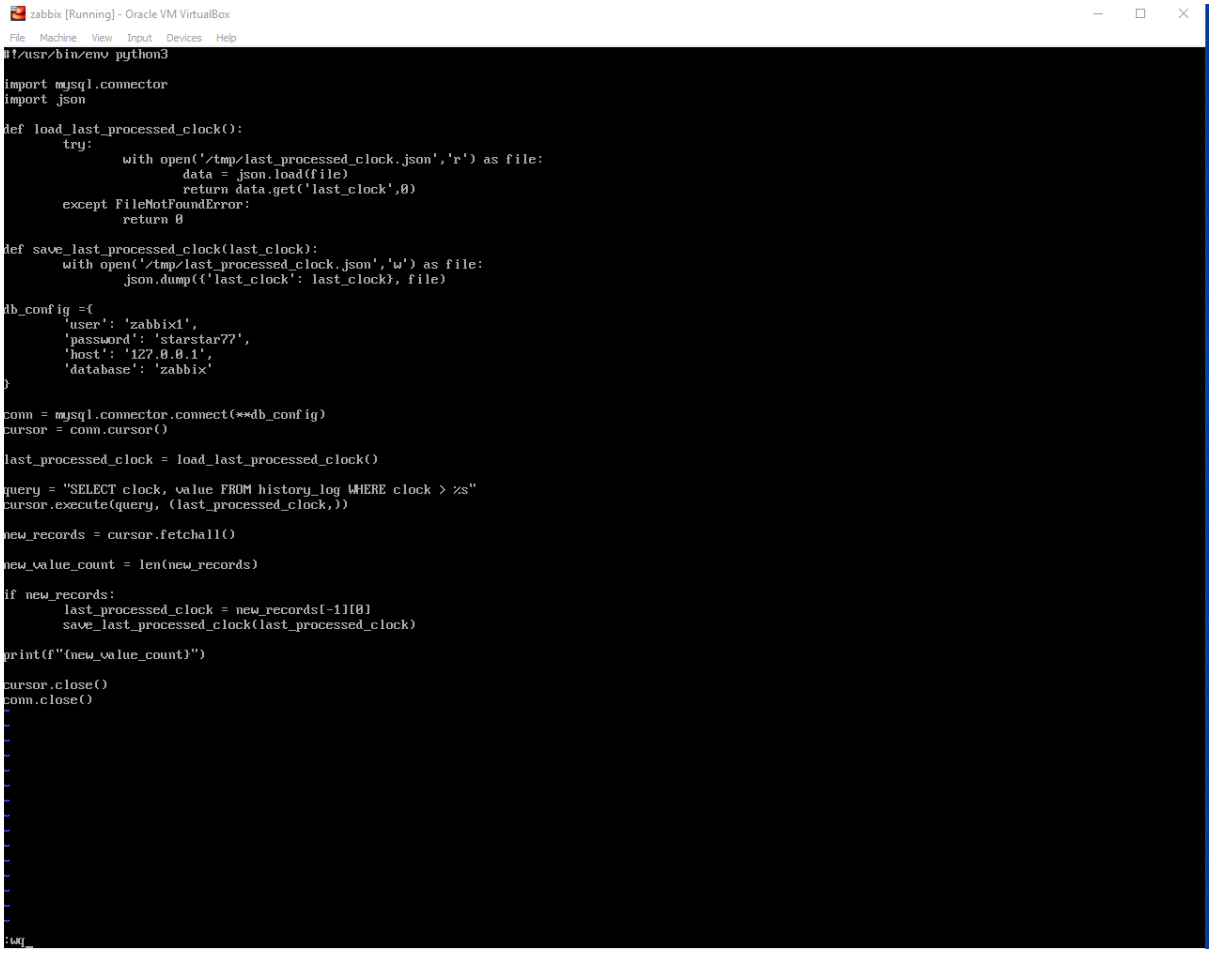
['user' = 'zabbix1']

['password' = 'starstar77']

['host' = '127.0.0.1']

['database' = 'zabbix']

17. Press 'Esc' and input ':wq' write to save the file



18. Set the file full execute permission for python script in Zabbix server by using the command *[sudo chmod +x /usr/lib/zabbix/externalscripts/threats.py]*

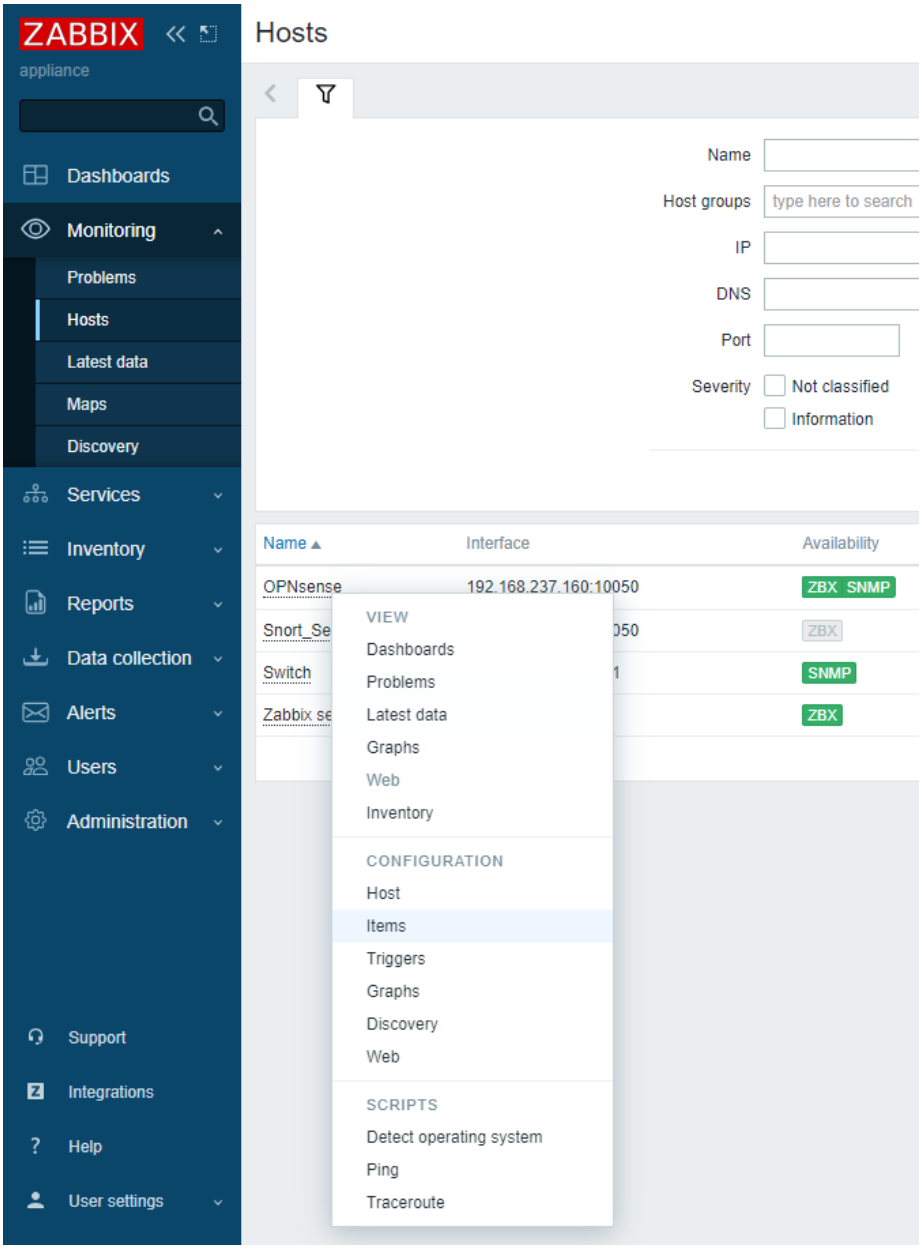
```
[root@appliance ~]# sudo chmod +x /usr/lib/zabbix/externalscripts/threats.py
```

19. Input ‘reboot’ to reboot the zabbix server with the latest configuration

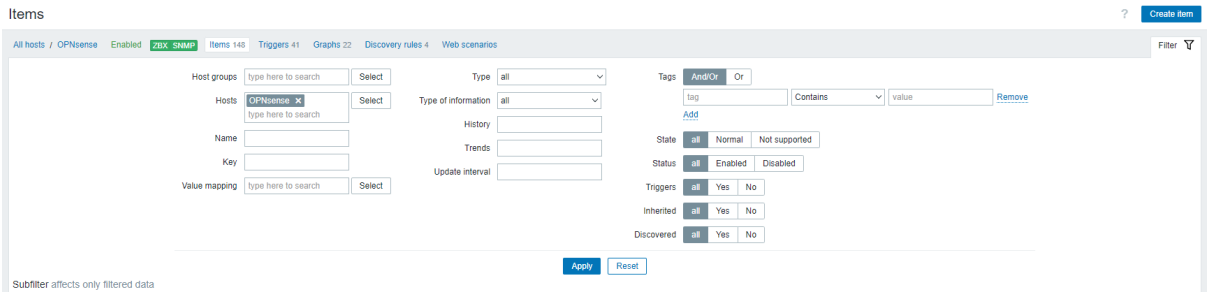
5.5.4 Virtualizing Data by using Zabbix

1. Back to zabbix front-end web, login with Username ‘Admin’ and Password ‘zabbix’

2. Go to ‘Monitoring; Hosts’ section, click on host ‘OPNsense’ and select ‘Items’ in configuration section



3. Create new item in host ‘OPNsense’



4. In item section set with the parameters

[Name: IDS log]

[Type: Zabbix Trapper]

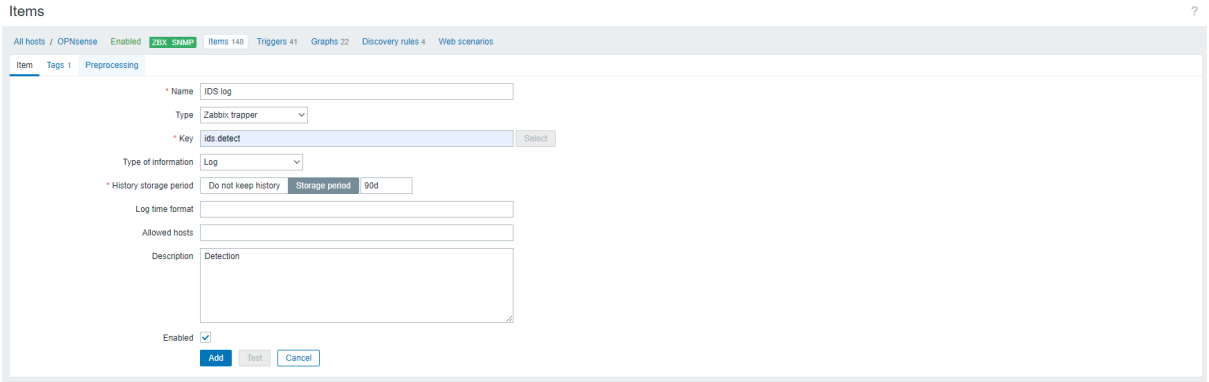
[Key: ids.detect]

[Type of information: Log]

[History storage period: Storage period]

[Description: Detection]

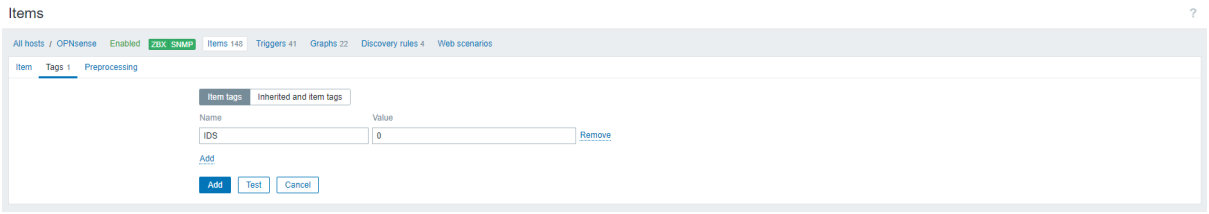
5. Tick the ‘Enabled’



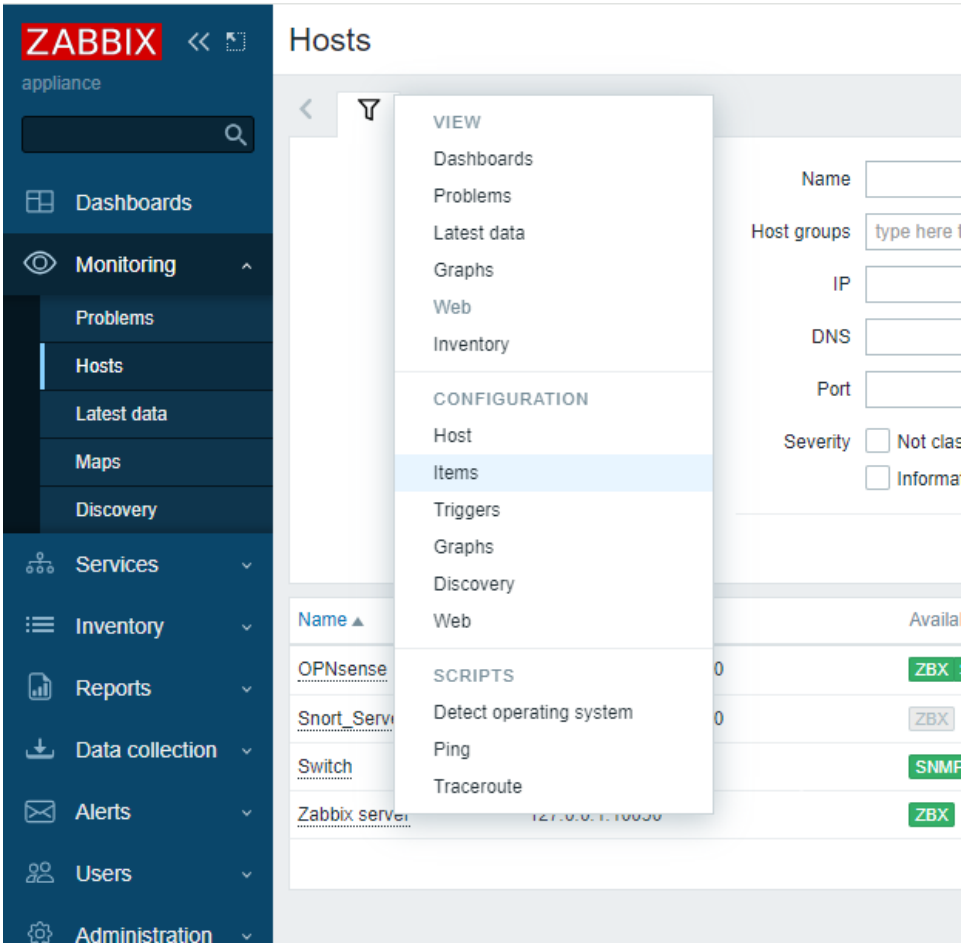
6. In tags set with the parameters

[Name: IDS; Value: 0]

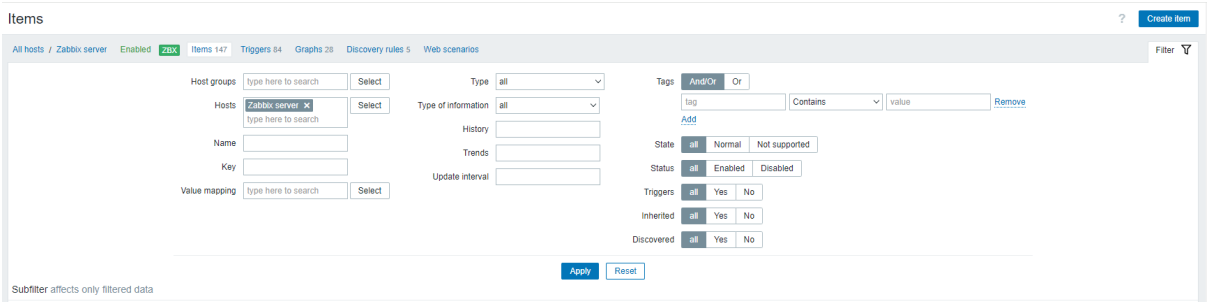
7. Click add to save and create the item



8. Go to ‘Monitoring; Hosts’ section, click on host ‘Zabbix server’ and select ‘Items’ in configuration section



9. Create new item in host ‘Zabbix server’



10. In item section set with the parameters

[Name: Threats]

[Type: External check]

[Key: threats.py]

[Type of information: Numeric (float)]

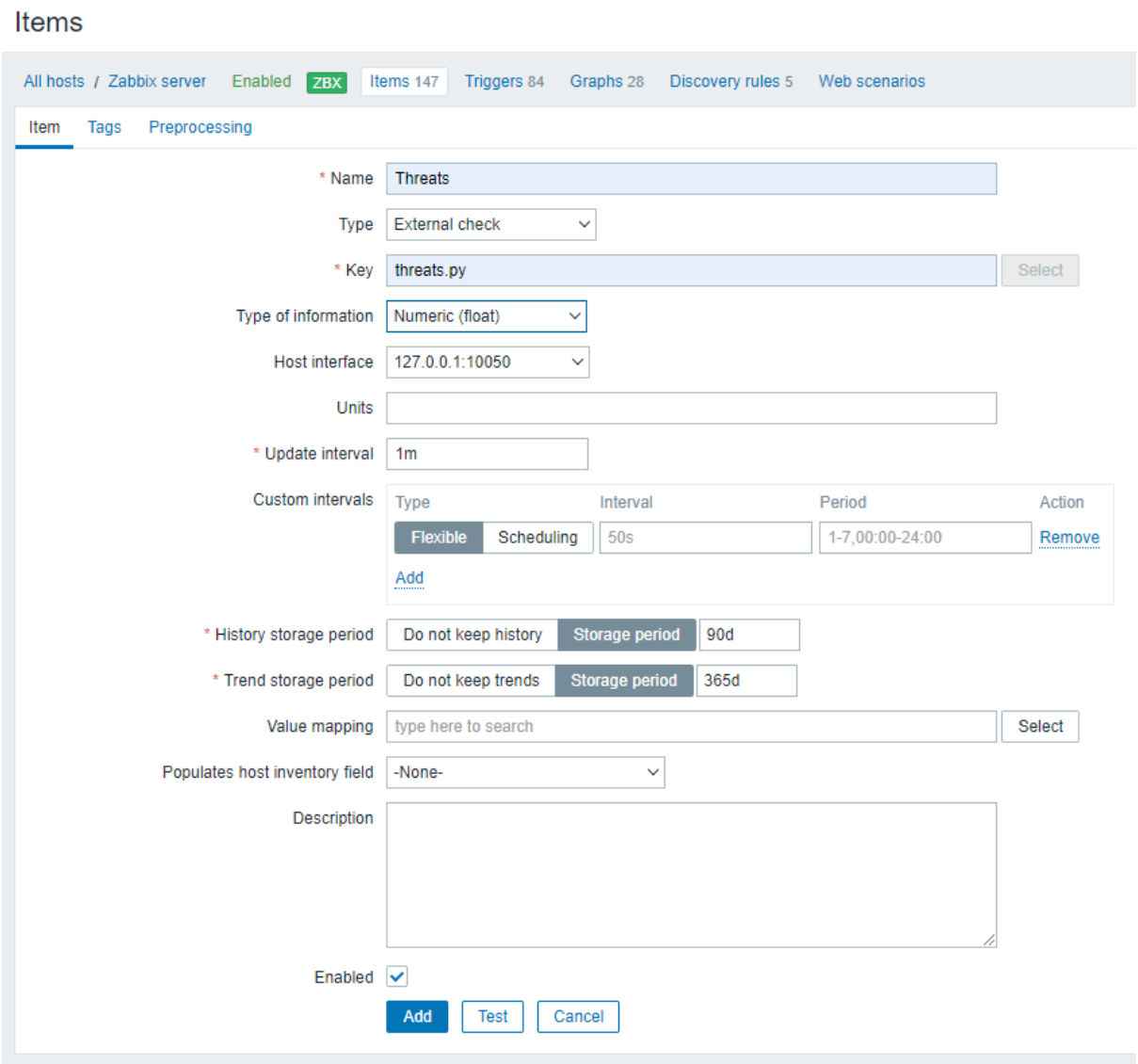
[Host interface: 127.0.0.1: 10050]

[Update interval: 1m]

[History storage period: Storage period]

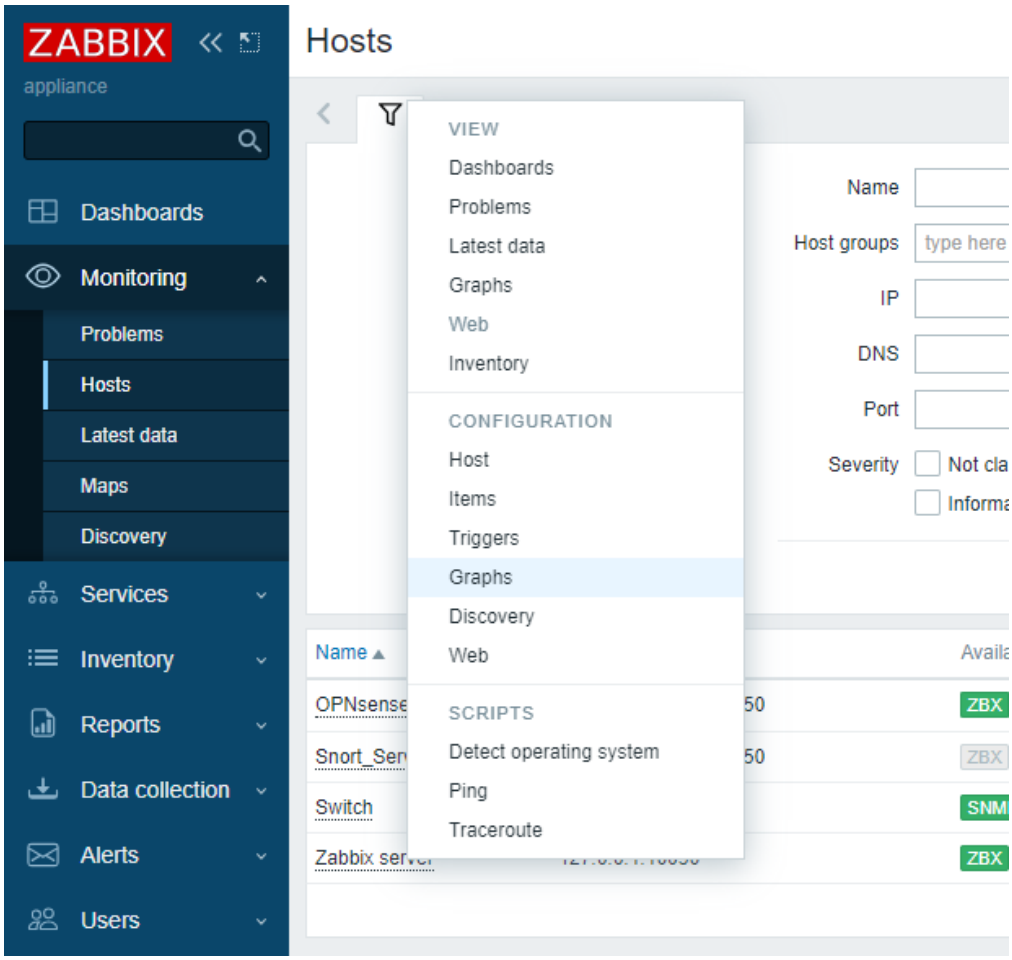
[Trend storage period: Storage period]

11. Tick the 'Enabled' and click Add to save and create the item



12. Go to ‘Monitoring; Hosts’ section, click on host ‘Zabbix server’ and select ‘Graphs’ in configuration section

13. Select Create Graph to create new graph for item ‘threats.py’



14. In Graph section set with the parameters

[Name: Number Threats Detected]

[Width: 900]

[Height: 200]

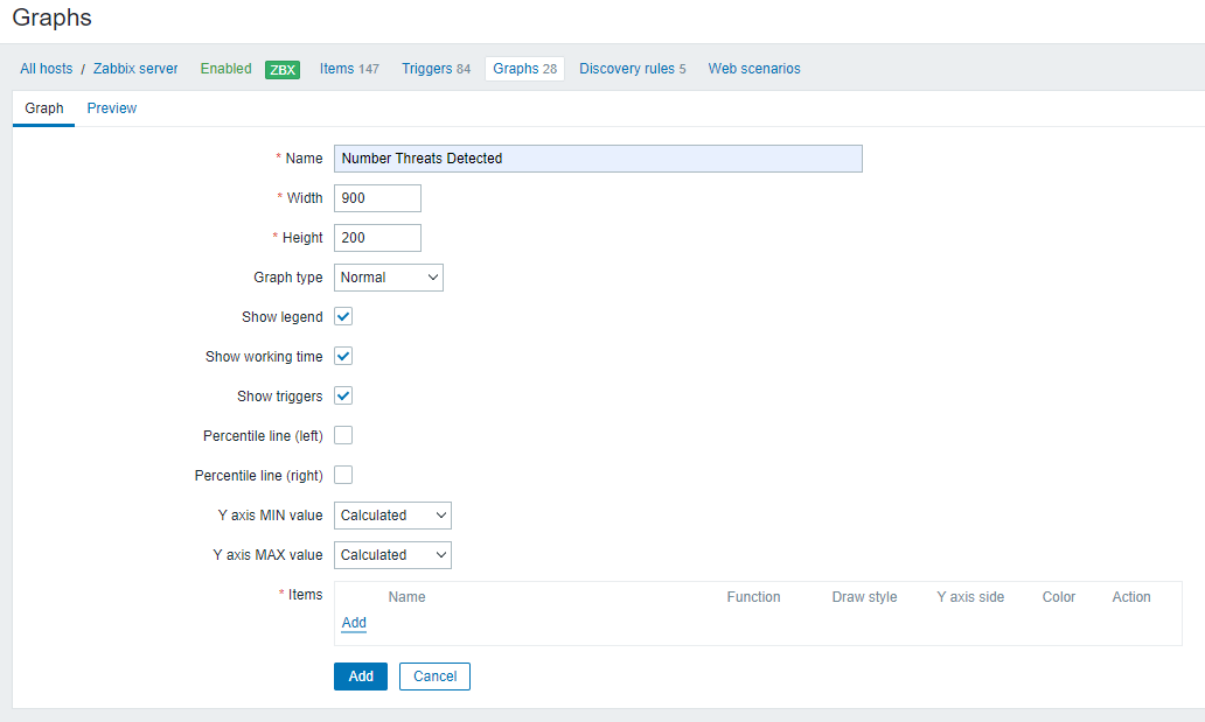
[Graph type: Normal]

[Tick Show legend, Show working time and Show triggers]

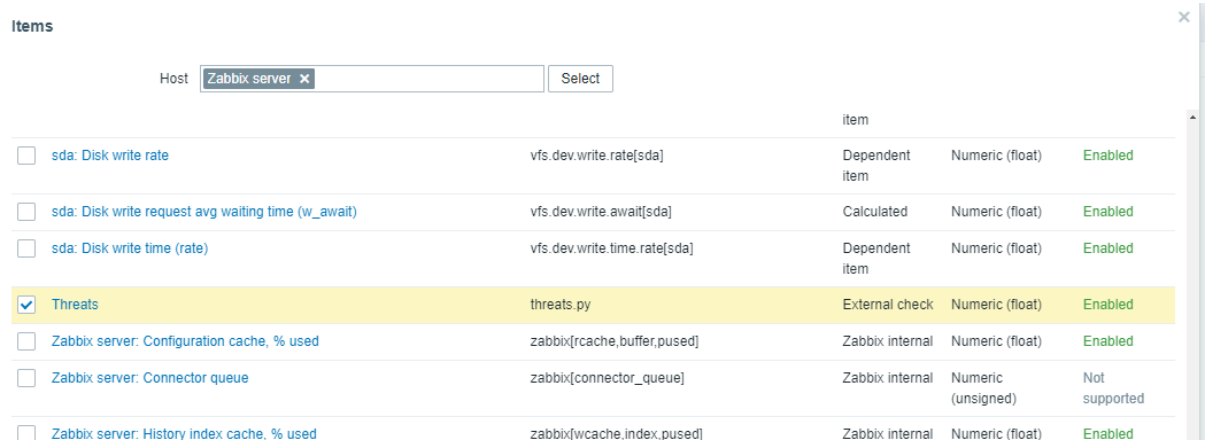
[Y axis MIN value: Calculated]

[Y axis MAX value: Calculated]

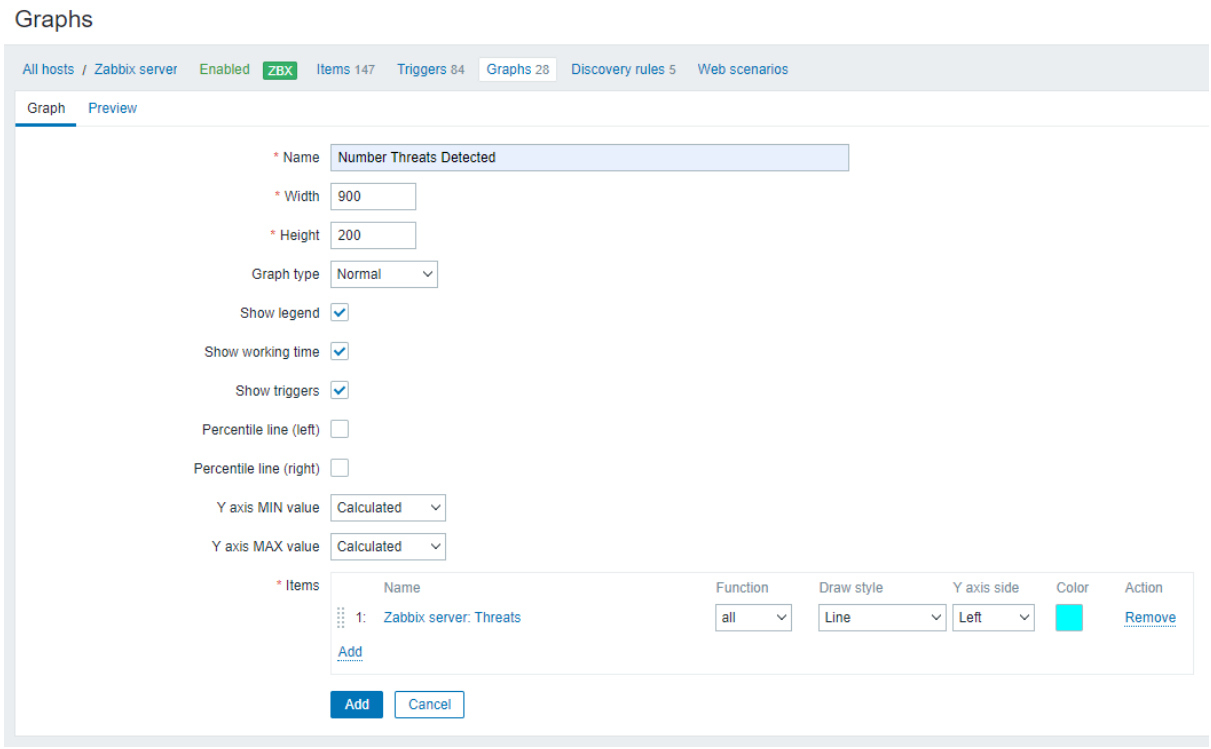
15. Select Add for import items key



16. Select host ‘Zabbix server’ and tick ‘Threats’

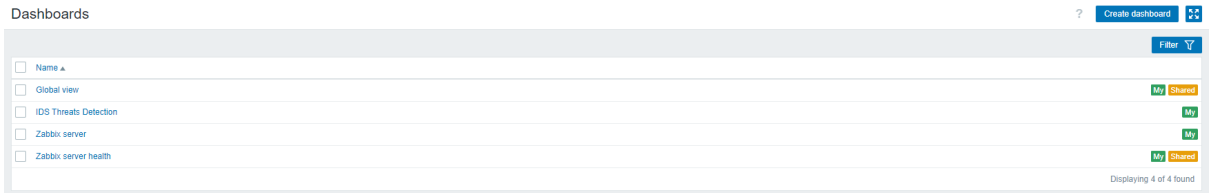


17. Click add to save and create the graph

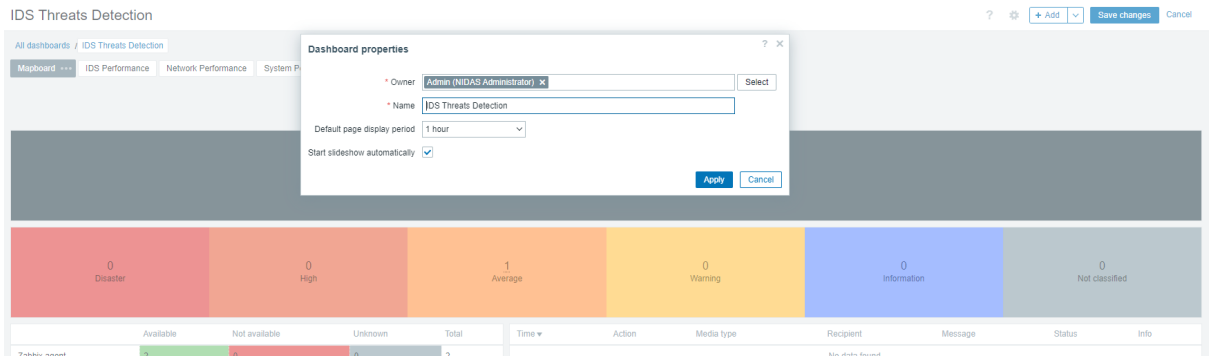


18. Go to Dashboard section in the menu, and click the ‘All dashboards’

19. Click Create dashboard



20. Select the ‘Admin’ as the Owner and name the dashboard then click Apply



21. Click '+Add' in the dashboard and select 'Add widget' to create the graph

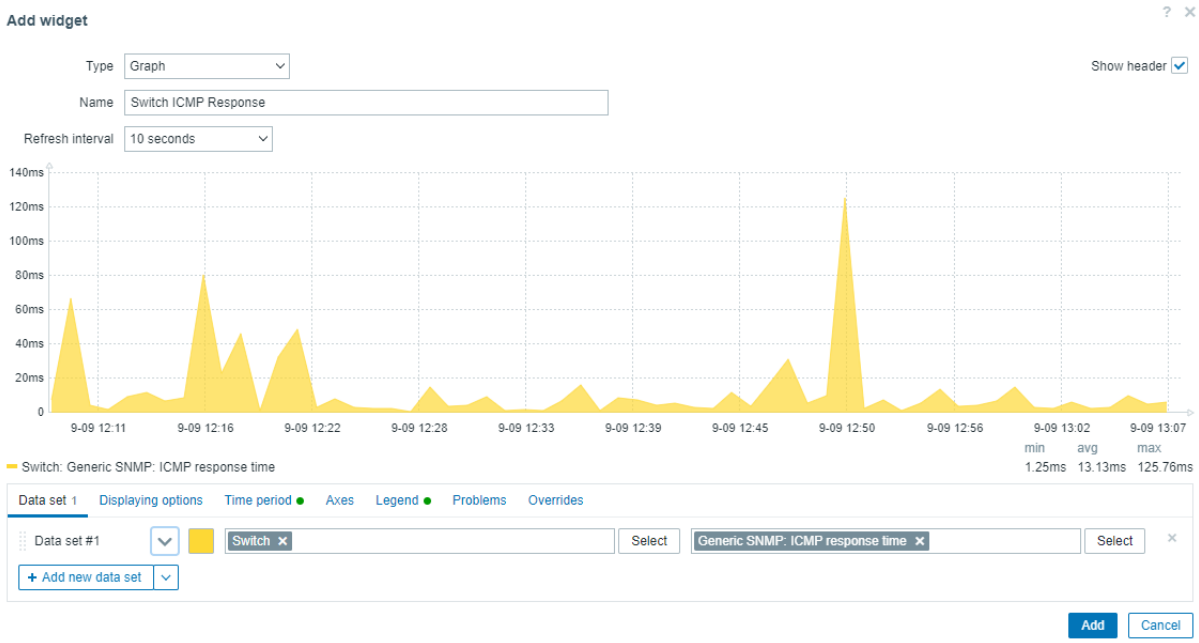
22. Create the widget with the following parameters

[Type: Graph]

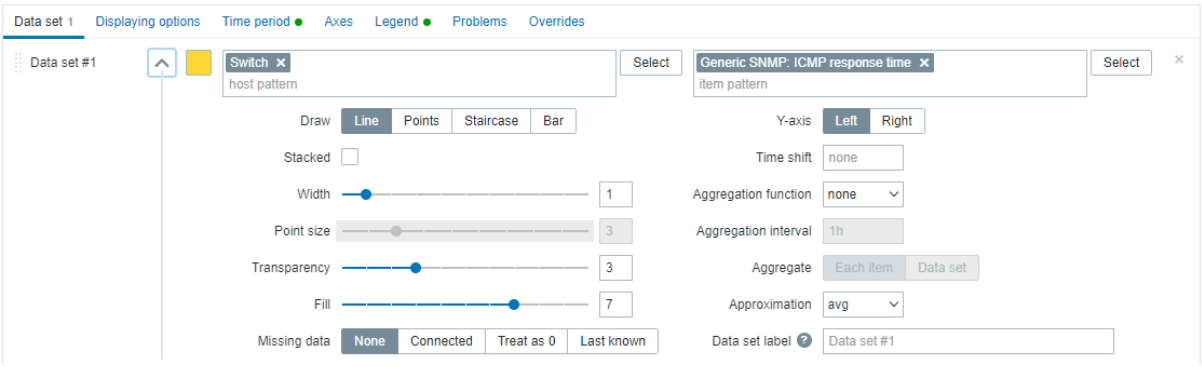
[Name: Switch ICMP Response]

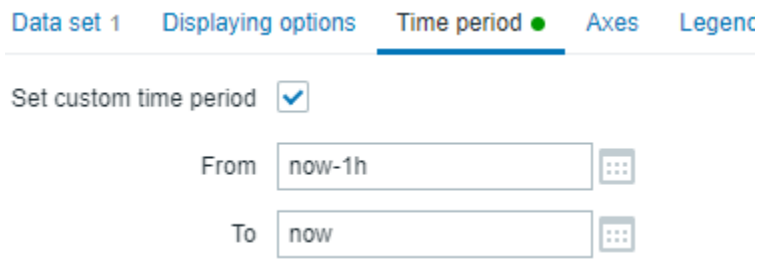
[Refresh interval: 10 seconds]

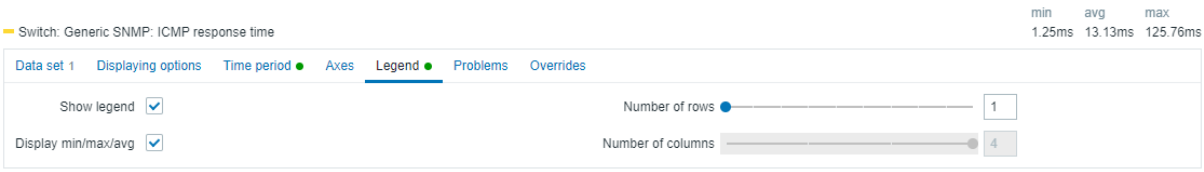
[Data set #1: Select host 'Switch' , Select item 'Generic SNMP: ICMP response time']



23. In Data set #1, design the dashboard with following information

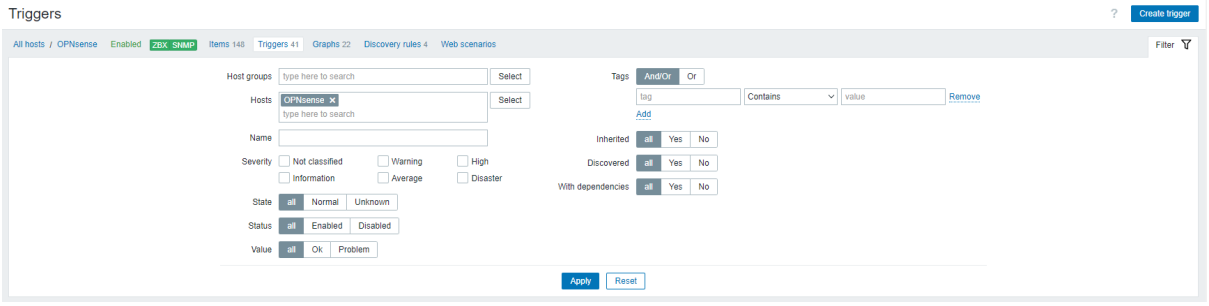






5.5.5 Setup Trigger and Alert by using Zabbix

1. Go to the Trigger section on host 'OPNsense' and select Create trigger



2. In trigger section set with the parameters and Enabled

[Name: IDS Alert Detected]

[Severity: High]

[Problem expression: nodata(/OPNsense/ids.detect,1s)=1]

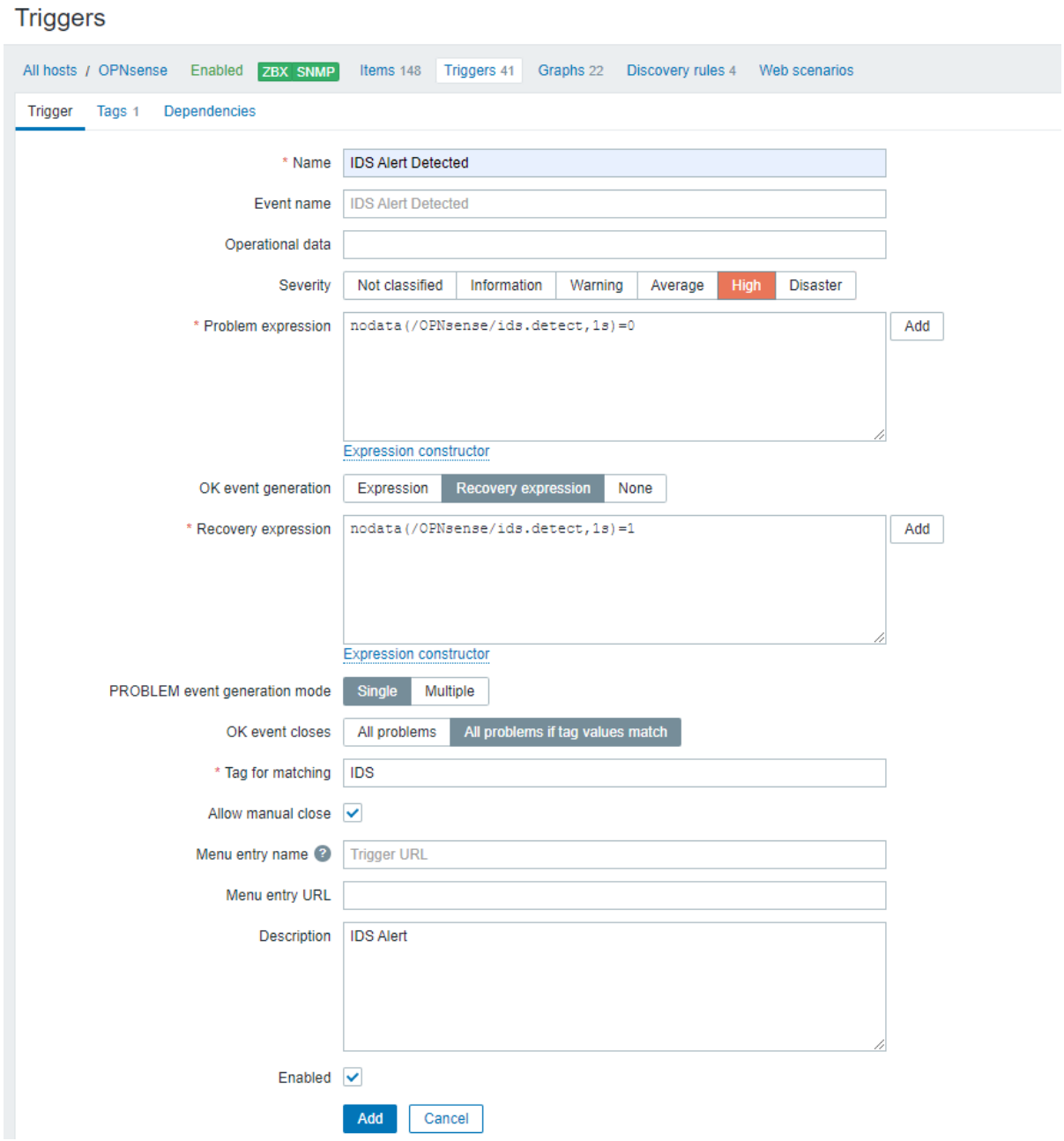
[PROBLEM event generation mode: Single]

[OK event closes: All problems if tag values match]

[Tag for matching: IDS]

[Enable Allow manual close]

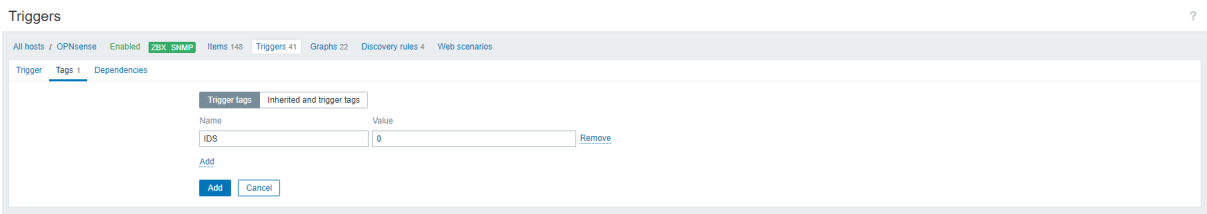
[Description: IDS Alert]



3. In Tags section set with parameters

[Name: IDS; Value:0]

4. Click add to save and create the trigger



5. Create another 2 triggers, ‘Port Mirroring Traffic SNMP’ and ‘Pflog Outgoing Traffic’ with the following parameters

[Name: Port Mirroring Traffic SNMP]

[Severity: Average]

[Problem expression: abs(last(/OPNsense/net.if.in[em2]))>6000000]

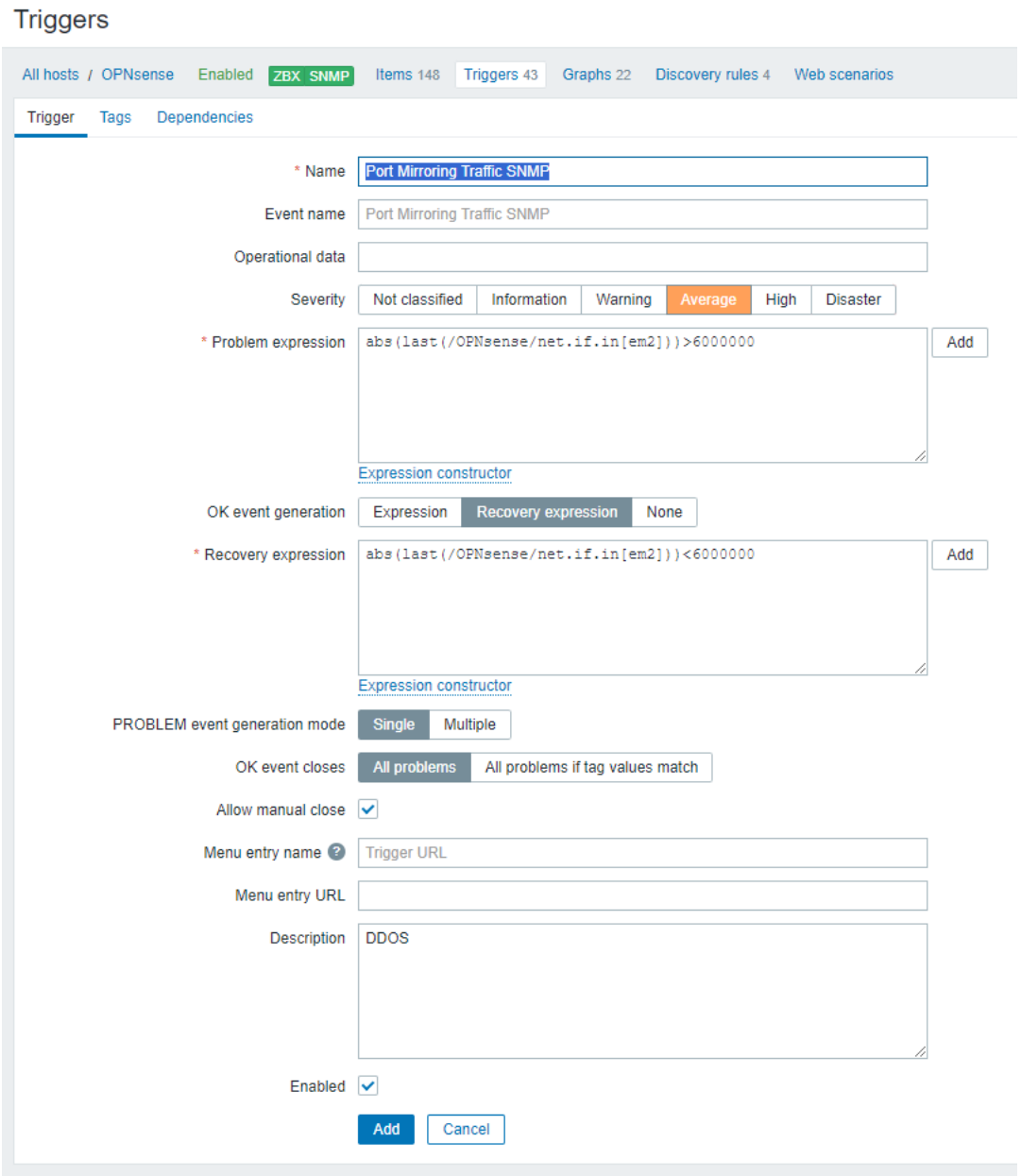
[OK event generation: Recovery expression]

[Recovery expression: abs(last(/OPNsense/net.if.in[em2]))<6000000]

[PROBLEM event generation mode: Single]

[Enable Allow manual close]

[Description: DDOS]



[Name: Pflog Outgoing Traffic]

[Severity: Average]

[Problem expression: abs(last(/OPNsense/net.if.in[pflog0]))>6000000]

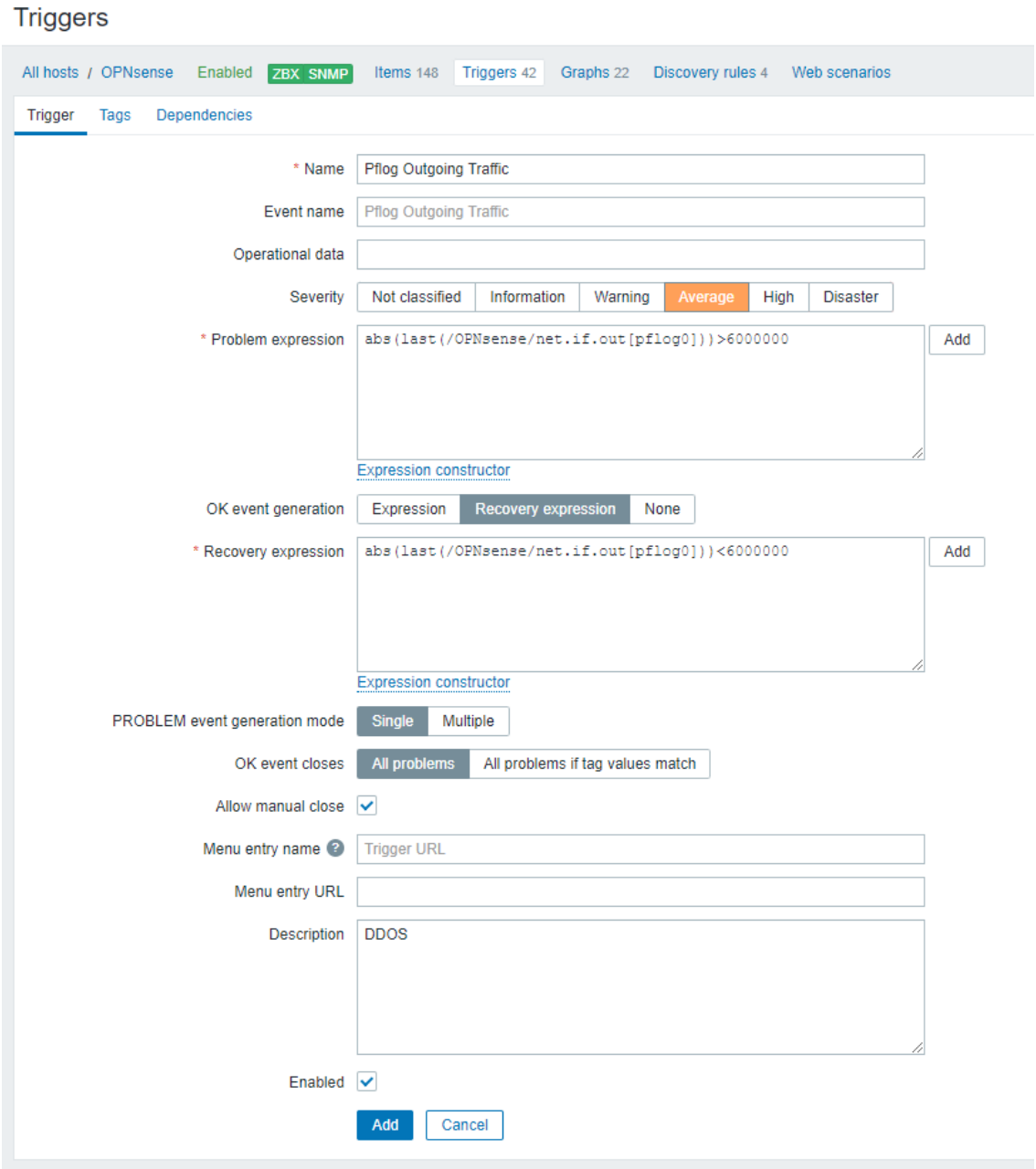
[OK event generation: Recovery expression]

[Recovery expression: abs(last(/OPNsense/net.if.in[pflog0]))<6000000]

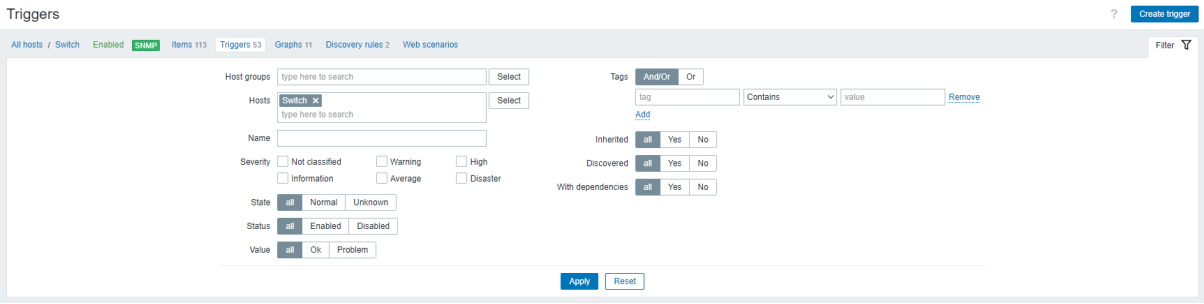
[PROBLEM event generation mode: Single]

[Enable Allow manual close]

[Description: DDOS]



6. Select host ‘Switch’ in Triggers section and create trigger



7. Set with the following parameters and enable the trigger

[Name: ICMP Response Heavy]

[Severity: High]

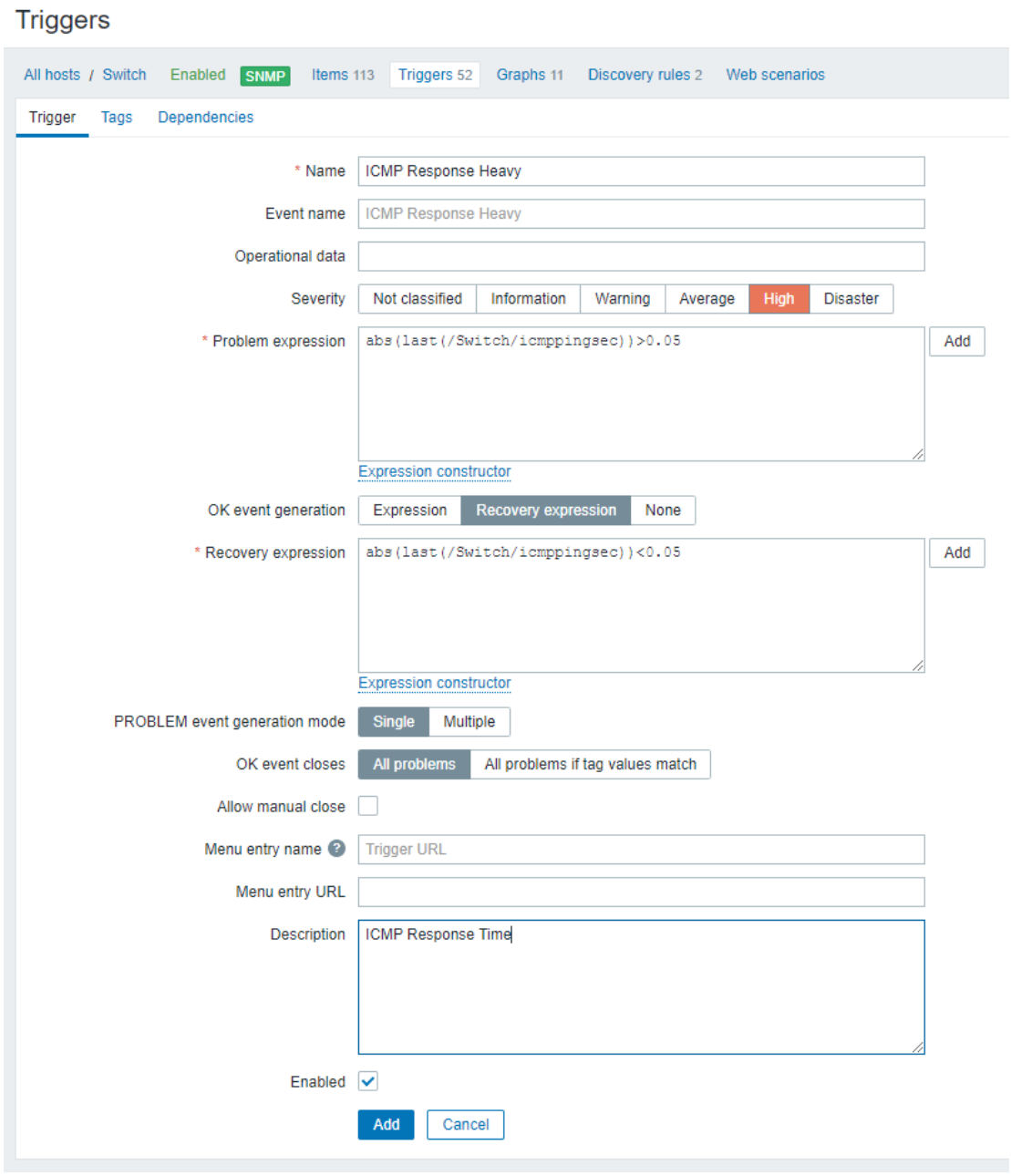
[Problem expression: abs(last(/Switch/icmppingsec))>0.05]

[OK event generation: Recovery expression]

[Recovery expression: abs(last(/Switch/icmppingsec))<0.05]

[PROBLEM event generation mode: Single]

[Description: ICMP Response Time]



5. Go to ‘Alerts; Media types’ section in the menu and select ‘Telegram’

8. Click clone to clone a new Telegram media types

9. Set the parameters and edit Script then click add

[Name: Telegram-GPT]

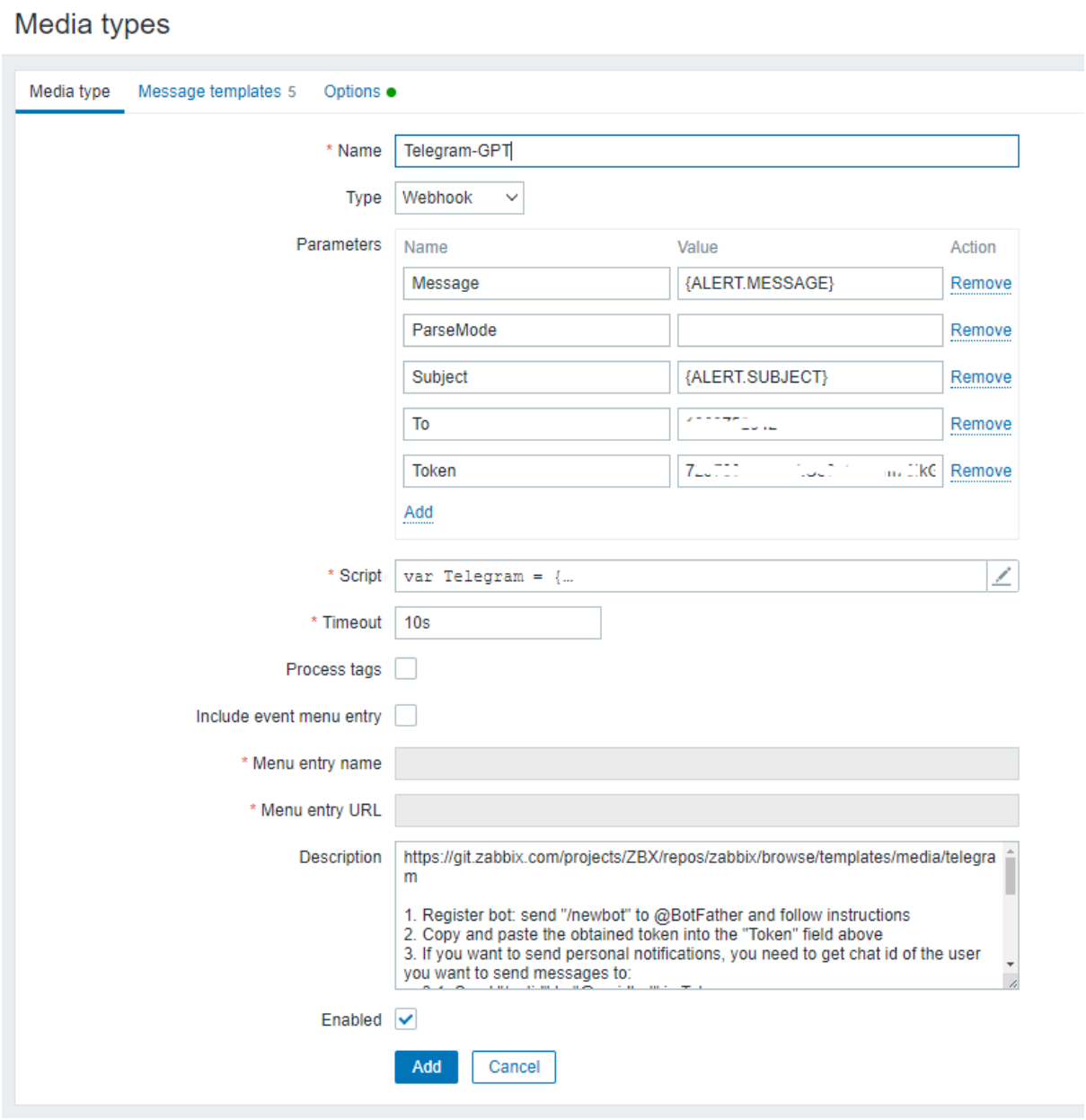
[Message: {ALERT.MESSAGE}]

[ParseMode:]

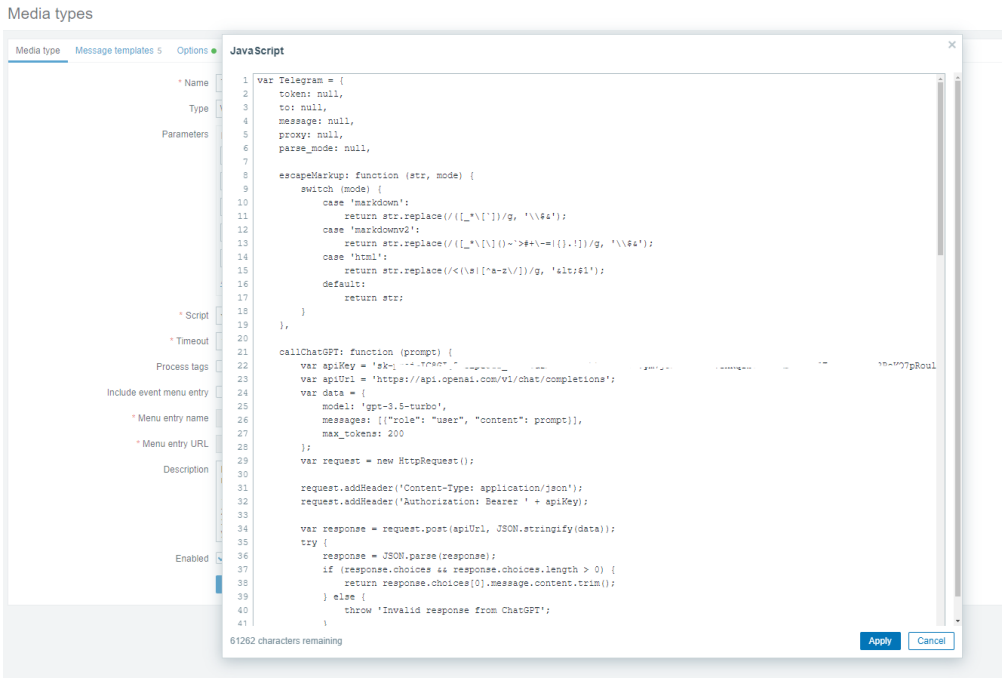
[Subject: {ALERT.SUBJECT}]

[To: <Telegram bot chatID>]

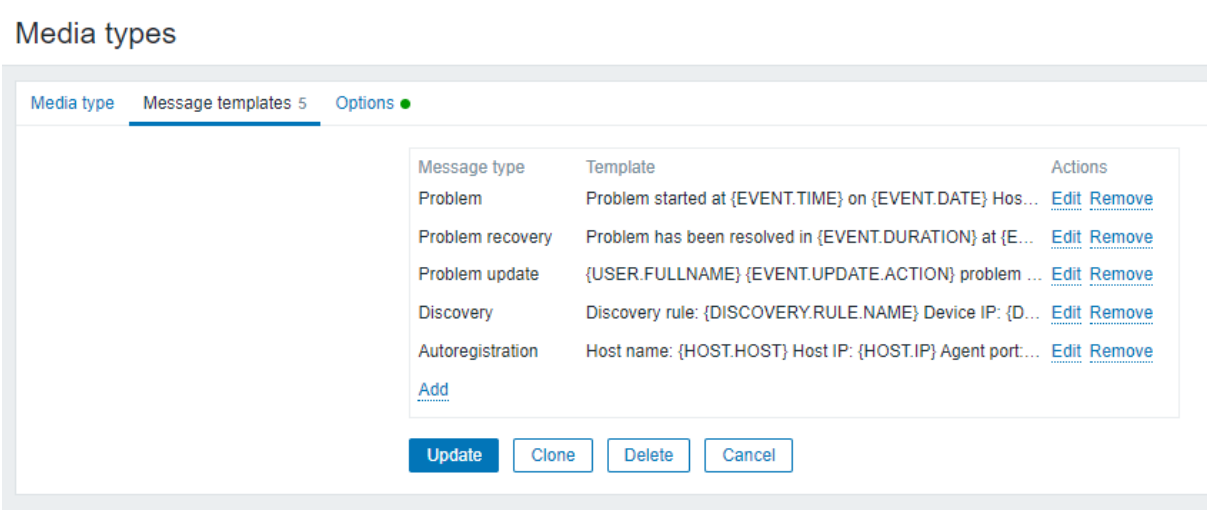
[Token: <Telegram bot token>]



10. Select the ‘Telegram-GPT’ media types just created, rewrite the script access with GPT-3.5 by using API key and click Apply, refer to Appendix D



11. In Message templates section edit ‘Problem’ message type



12. Edit the Message template with the parameters and click Update

[Message type: Problem]

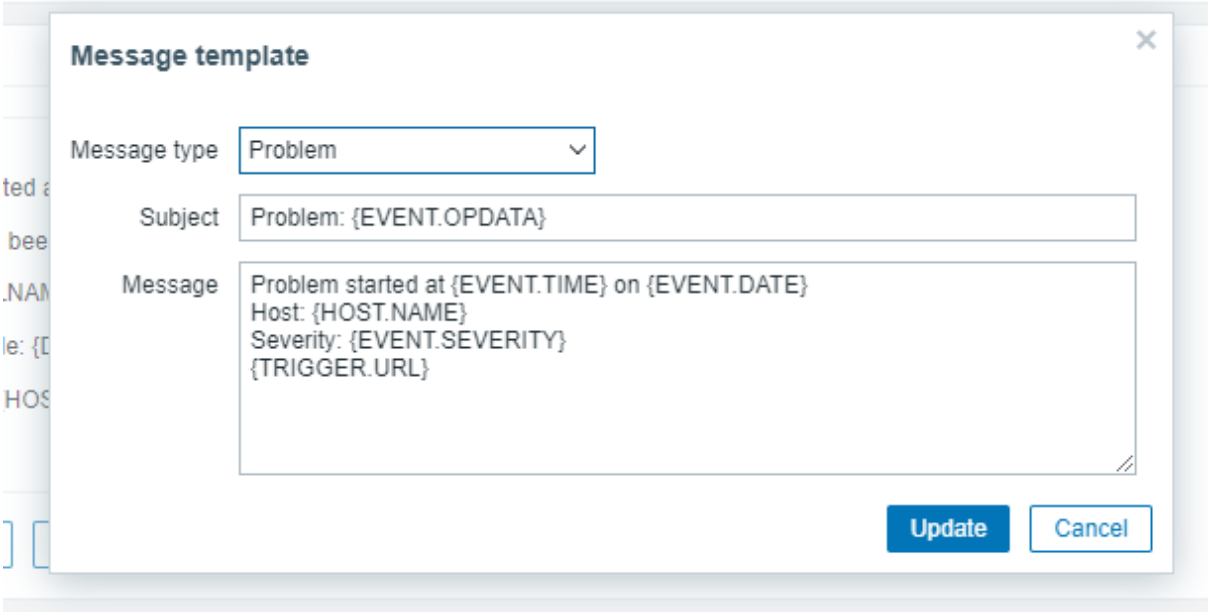
[Subject: Problem: {EVENT.OPDATA}]

[Message: Problem started at {EVENT.TIME} on {EVENT.DATE}

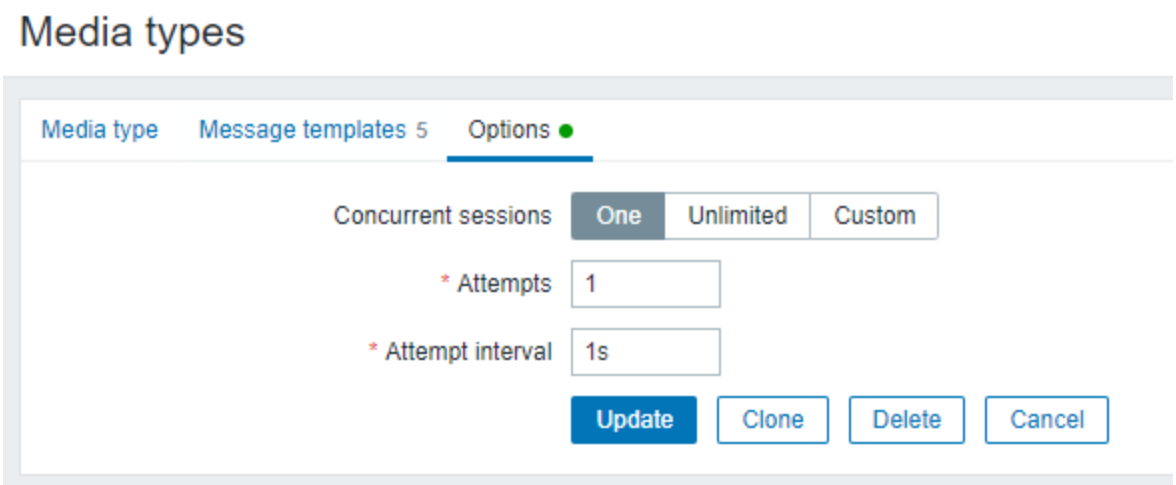
Host: {HOST.NAME}

Severity: {EVENT.SEVERITY}

{TRIGGER.URL}]

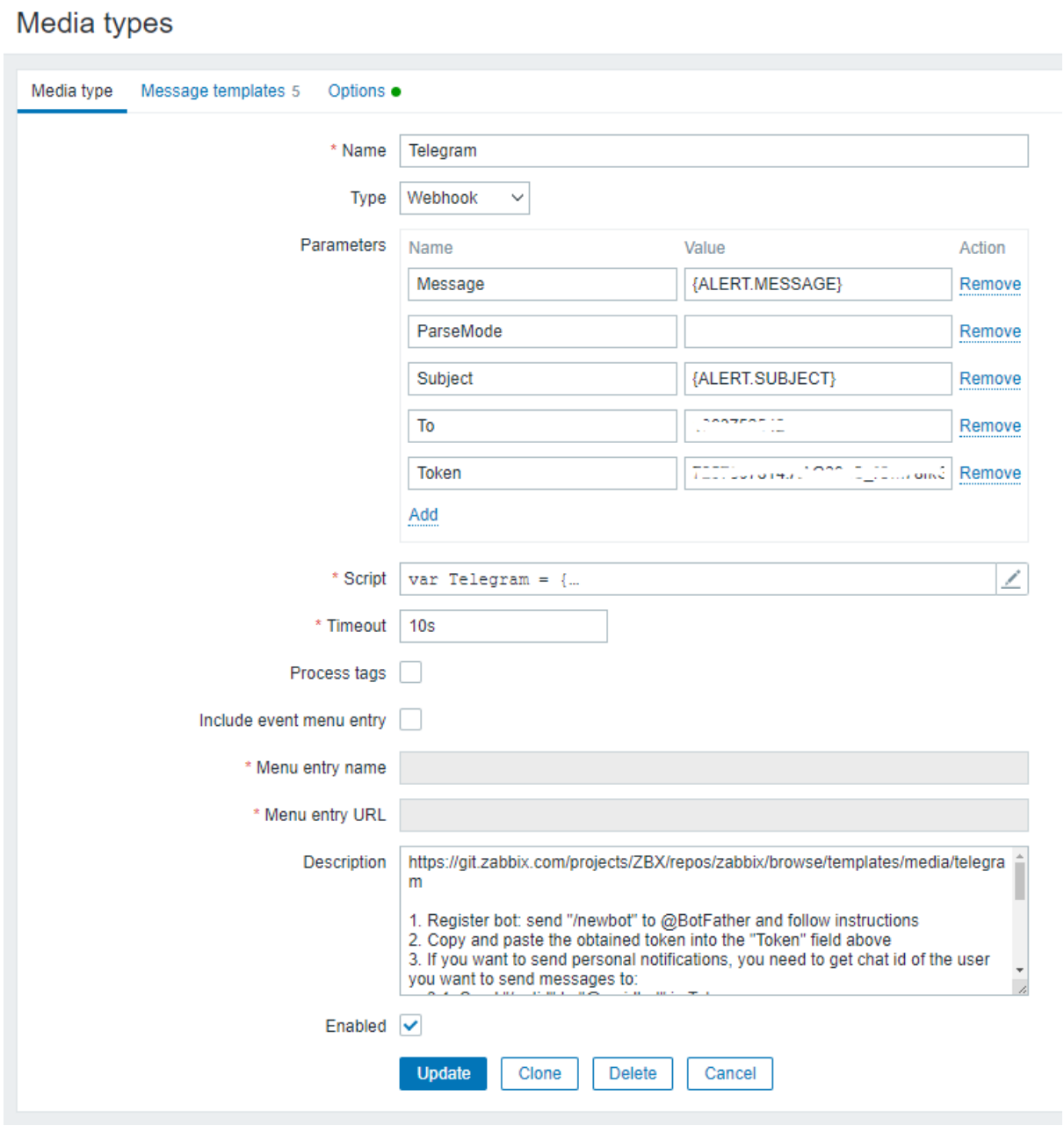


13. Go to Options, set the Attempts as 1 and Attempt interval as 1s, click Update

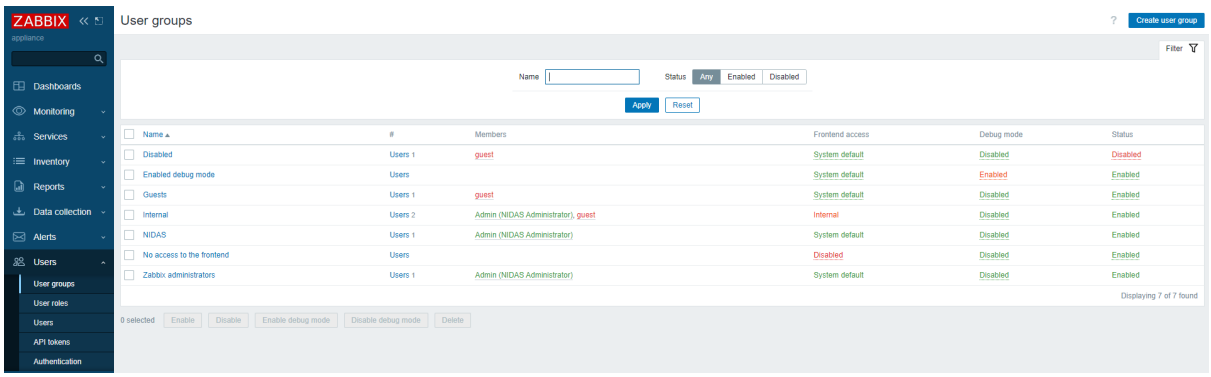


14. Go to the media types, select the default Telegram and enable

15. Set the Bot ChatID and the Token then click Update



16. Go to [Users; User groups] section in the menu and select Create user group

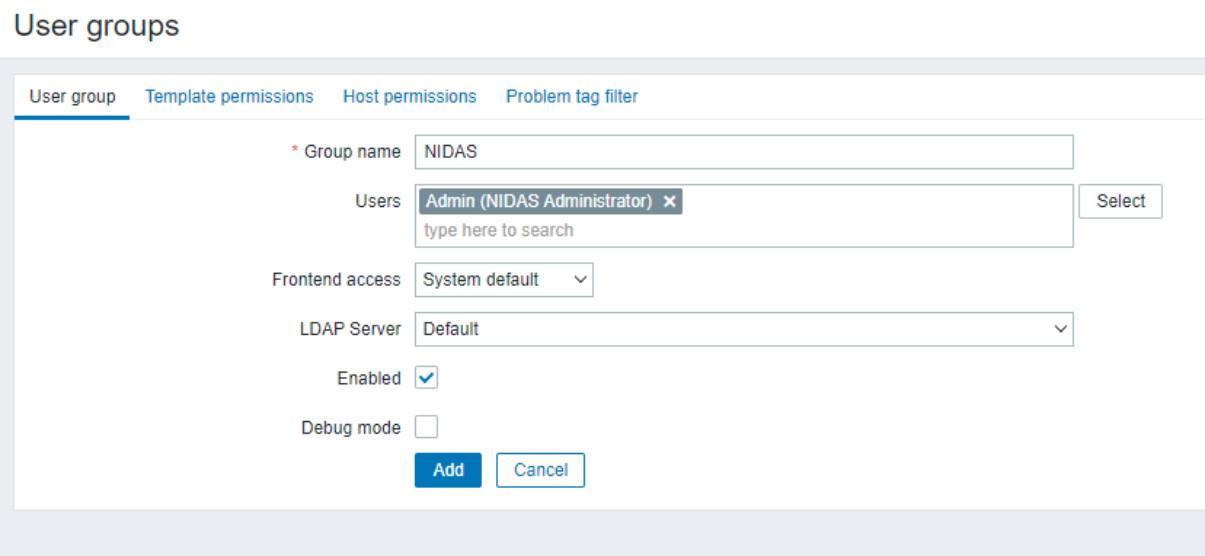


17. In User group section set the parameters and enabled

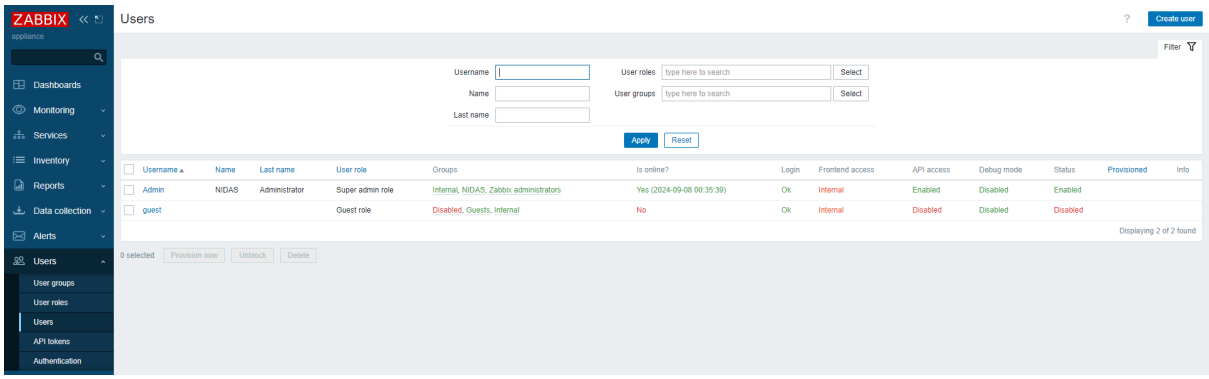
[Group name: NIDAS]

[Users: Admin (NIDAS Administrator)]

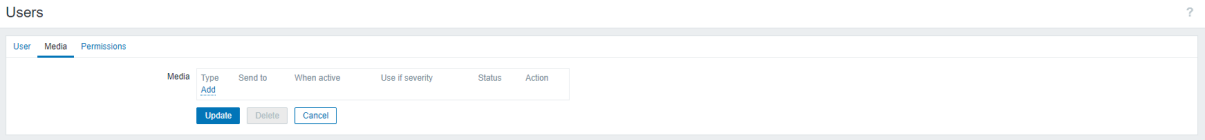
18. Click Add to save and create the new user group



19. Go to the users section, select Admin



20. In Media section, add media



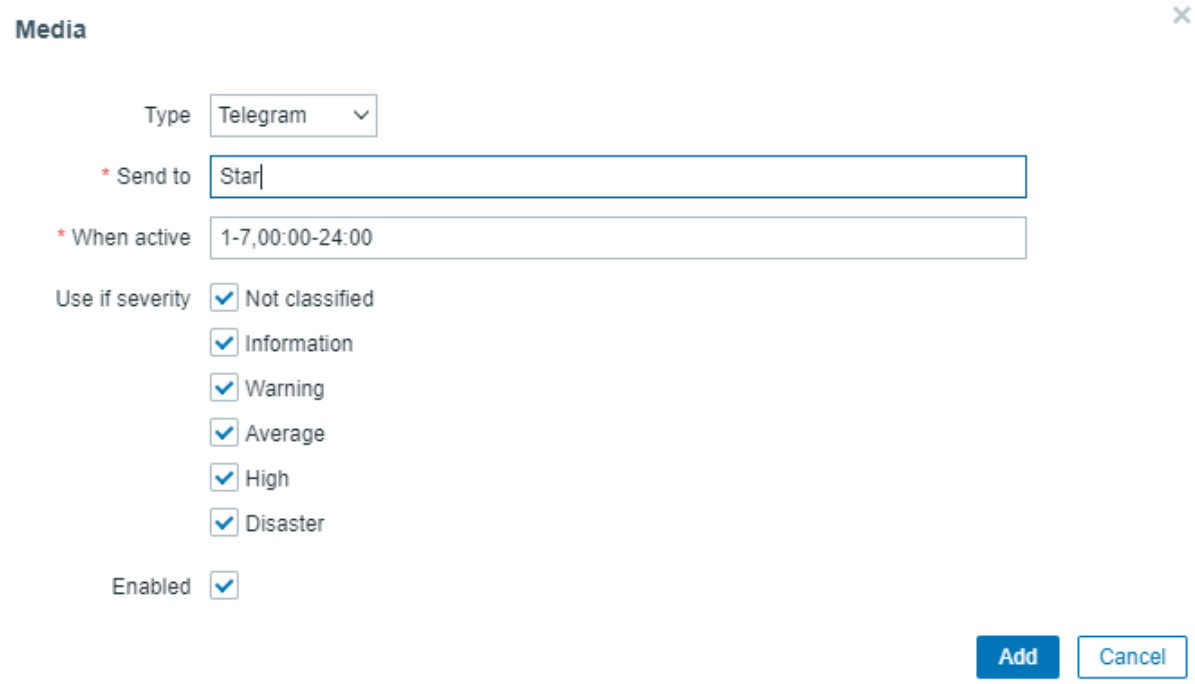
21. Set the new media type as Telegram and parameters with

[Send to: Star]

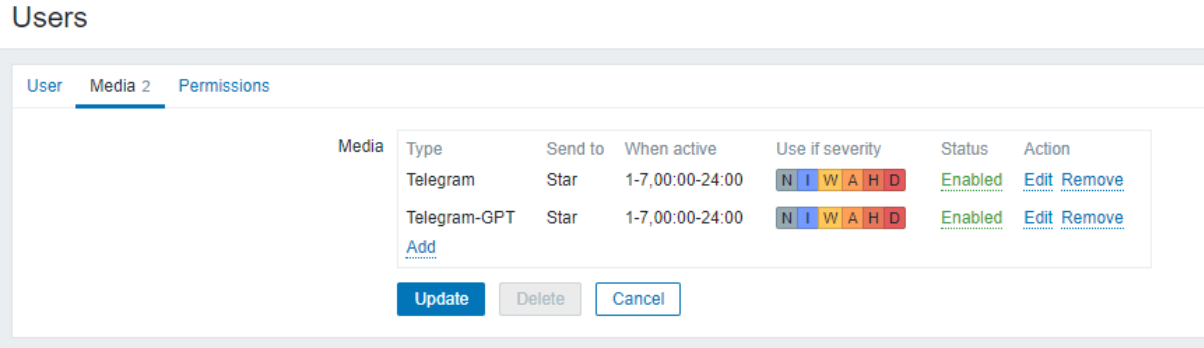
[When active: 1-7,00:00-24:00]

22. Enable all severity in Use if severity

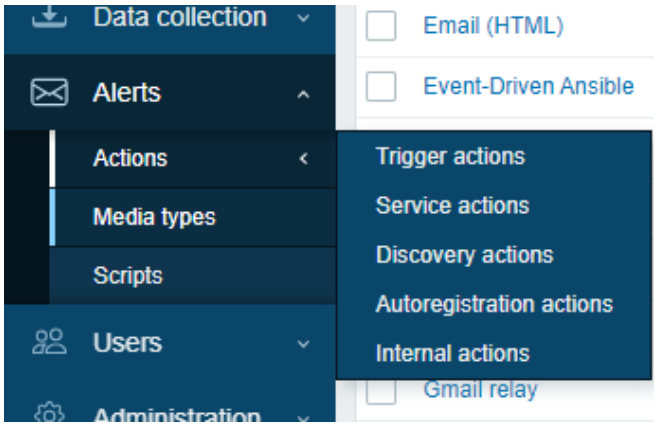
23. Click add to save new media



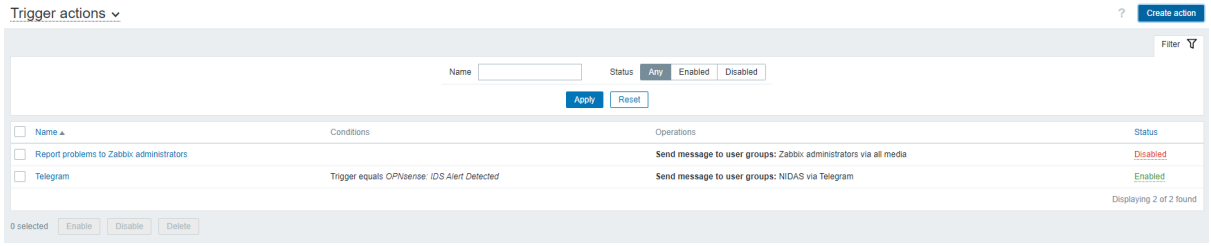
24. Add another media and ensure both of the media Telegram and Telegram-GPT status is Enabled



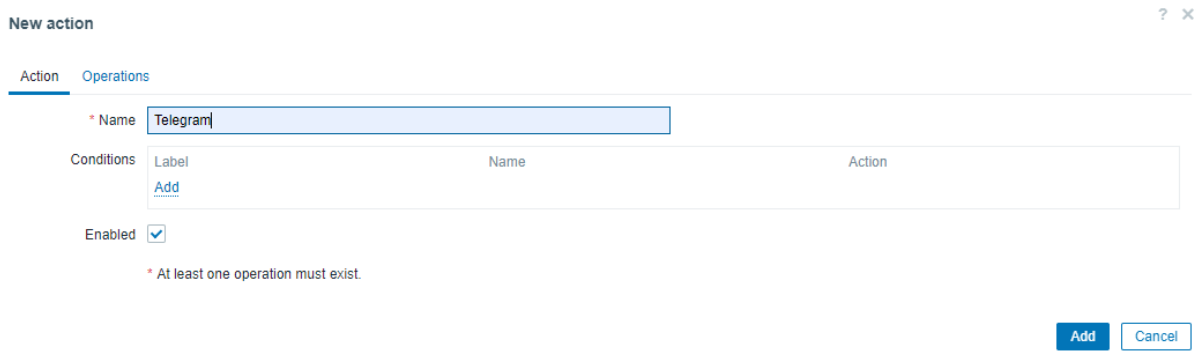
25. Go to Trigger actions in the menu ‘Alerts; Actions’ section



26. Create new action



27. Name the Telegram and add new conditions



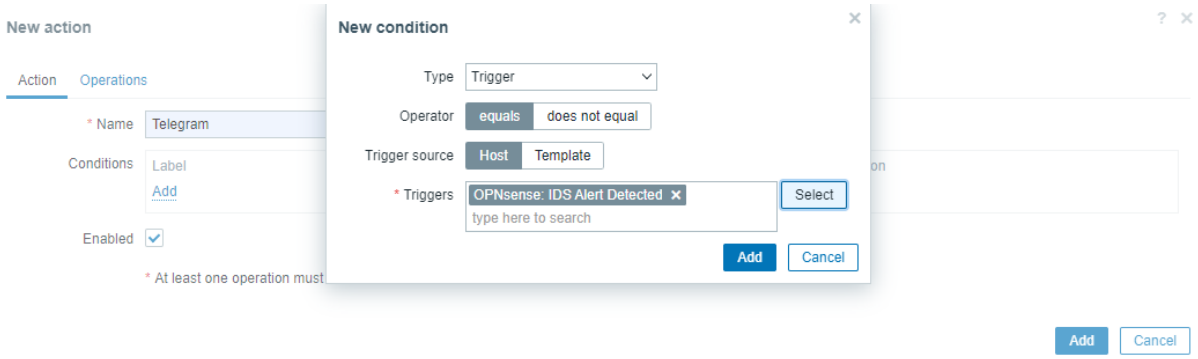
28. Set new condition parameters and click add

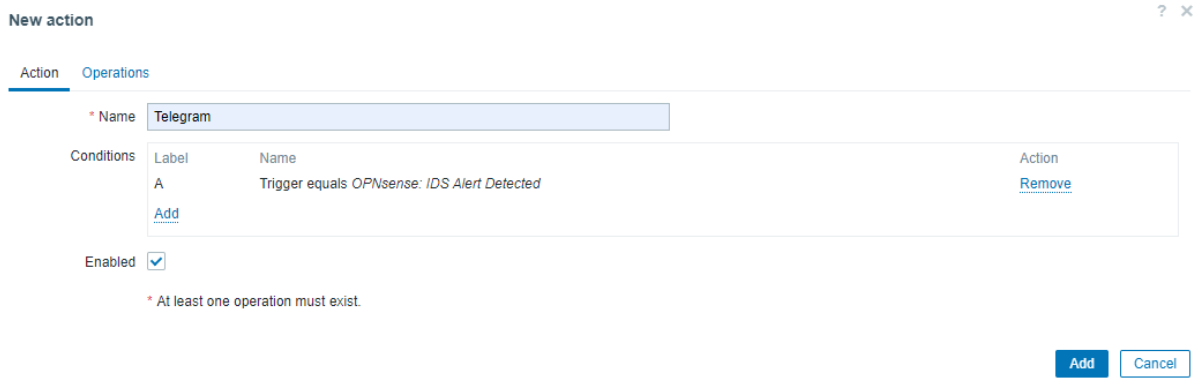
[Type: Trigger]

[Operator: equals]

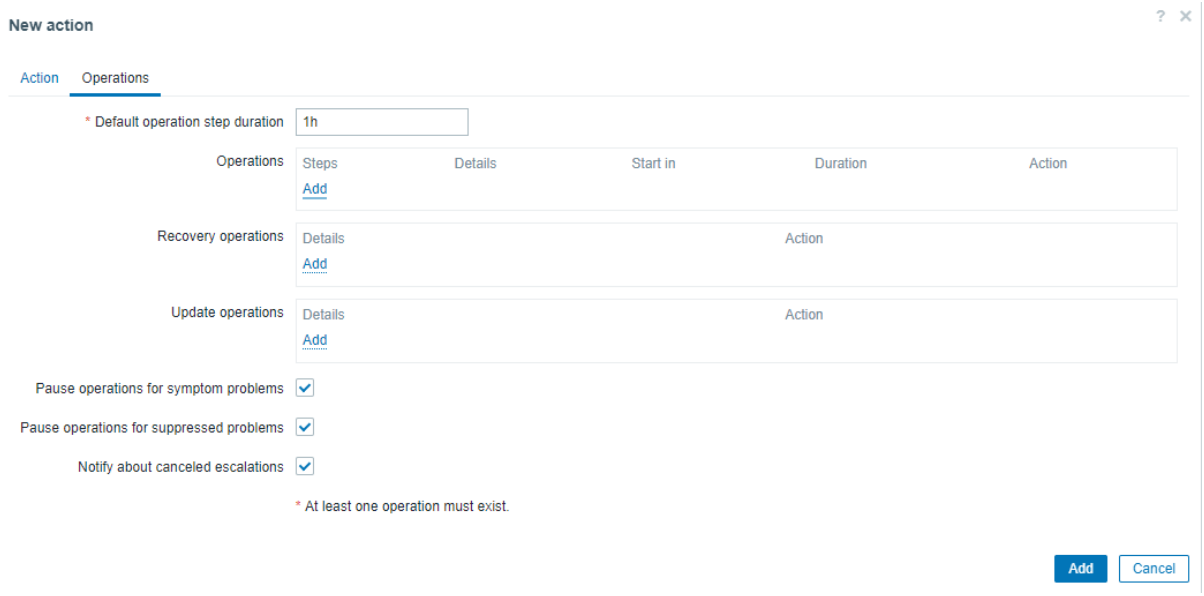
[Trigger source: Host]

[Triggers: OPNsense IDS Alert Detected]





29. Go to Operations site, add new Operations



30. Set Operation details with the conditions and click add

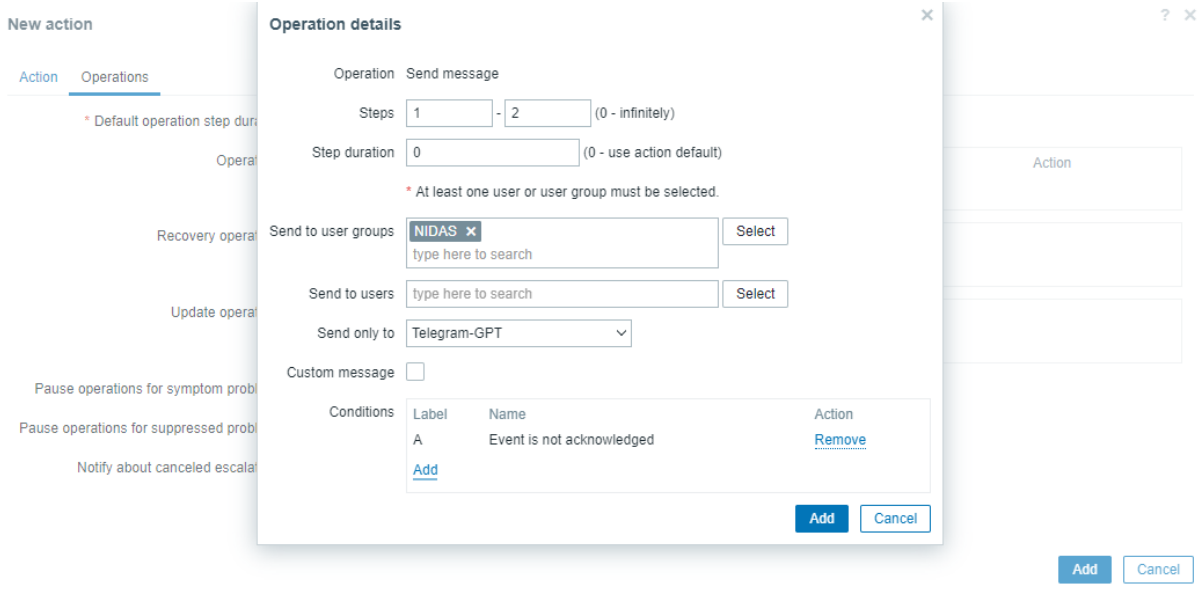
[Steps: 1-2]

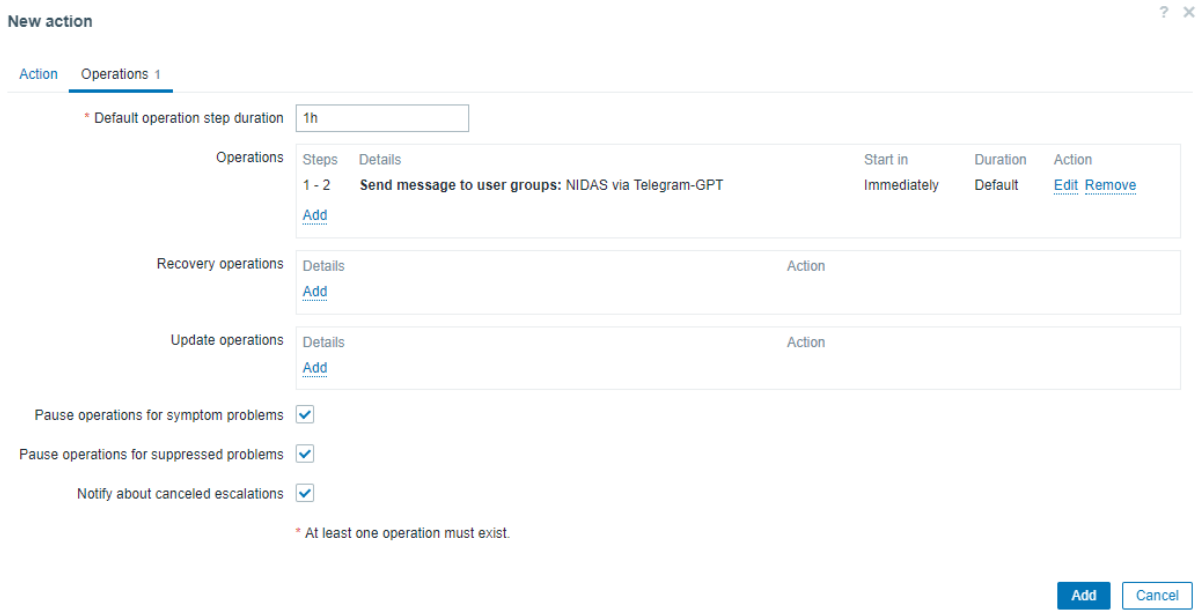
[Step duration: 0]

[Send to user groups: NIDAS]

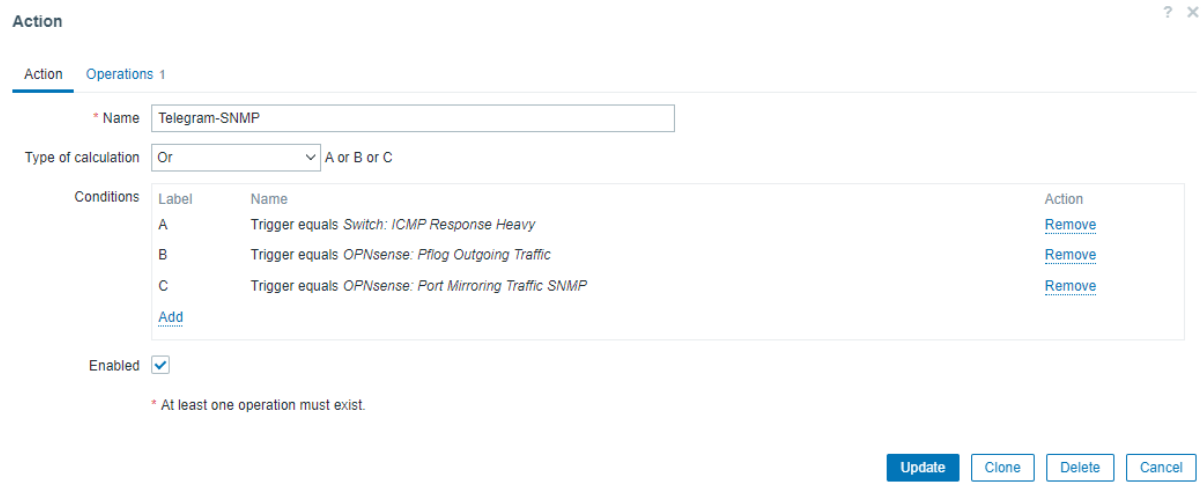
[Send only to: Telegram-GPT]

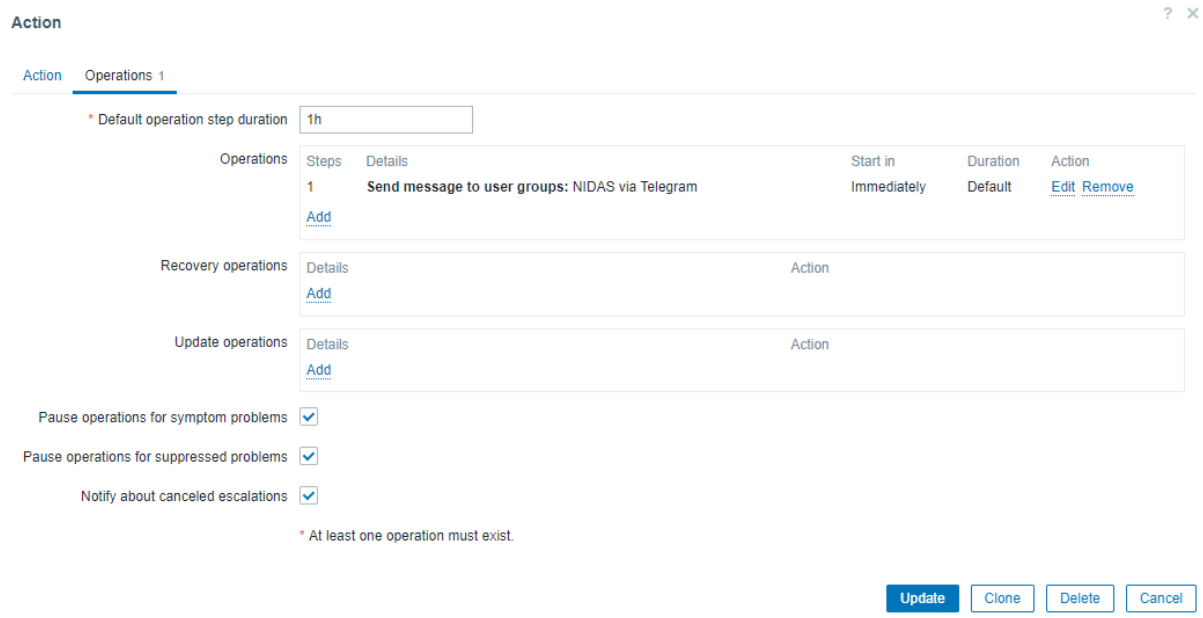
[Conditions: A Event is not acknowledged]

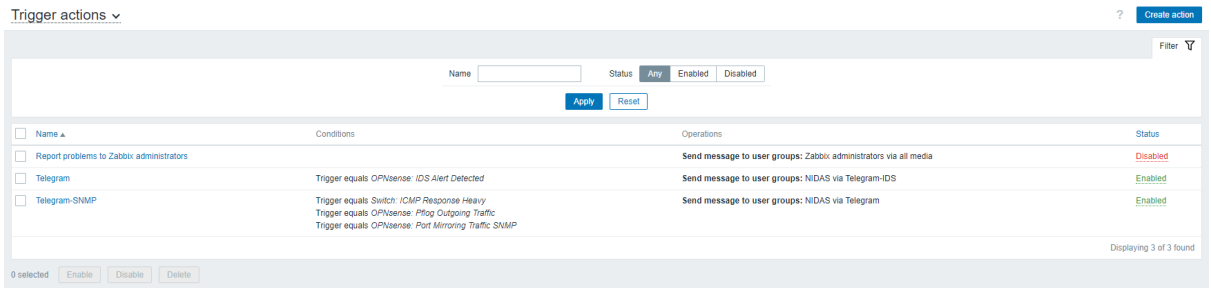




31. Create another new action with the following setting







5.6 System Operation

Figure 5.6.1 Dashboard Mapboard

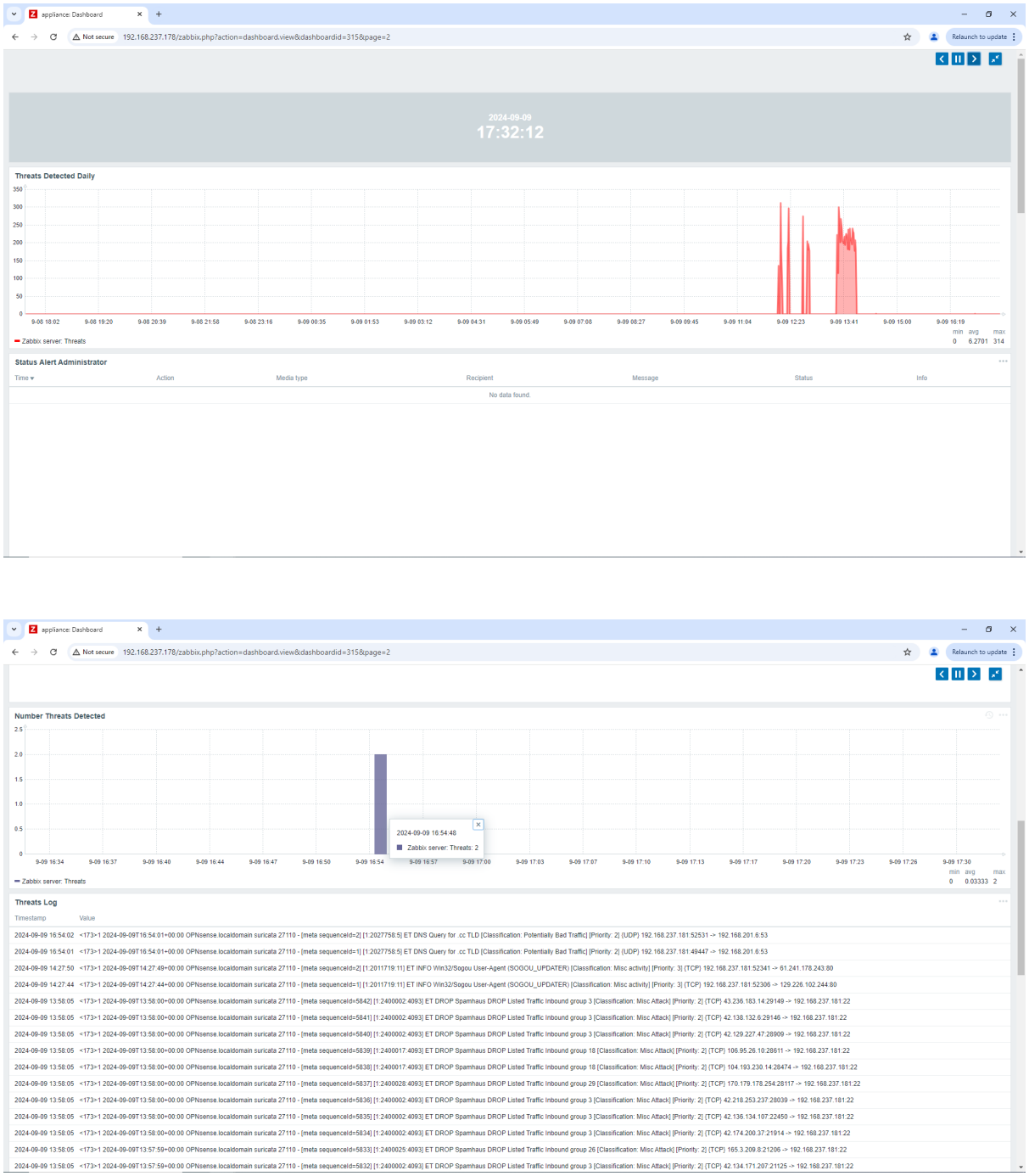


Figure 5.6.2 Dashboard IDS Performance

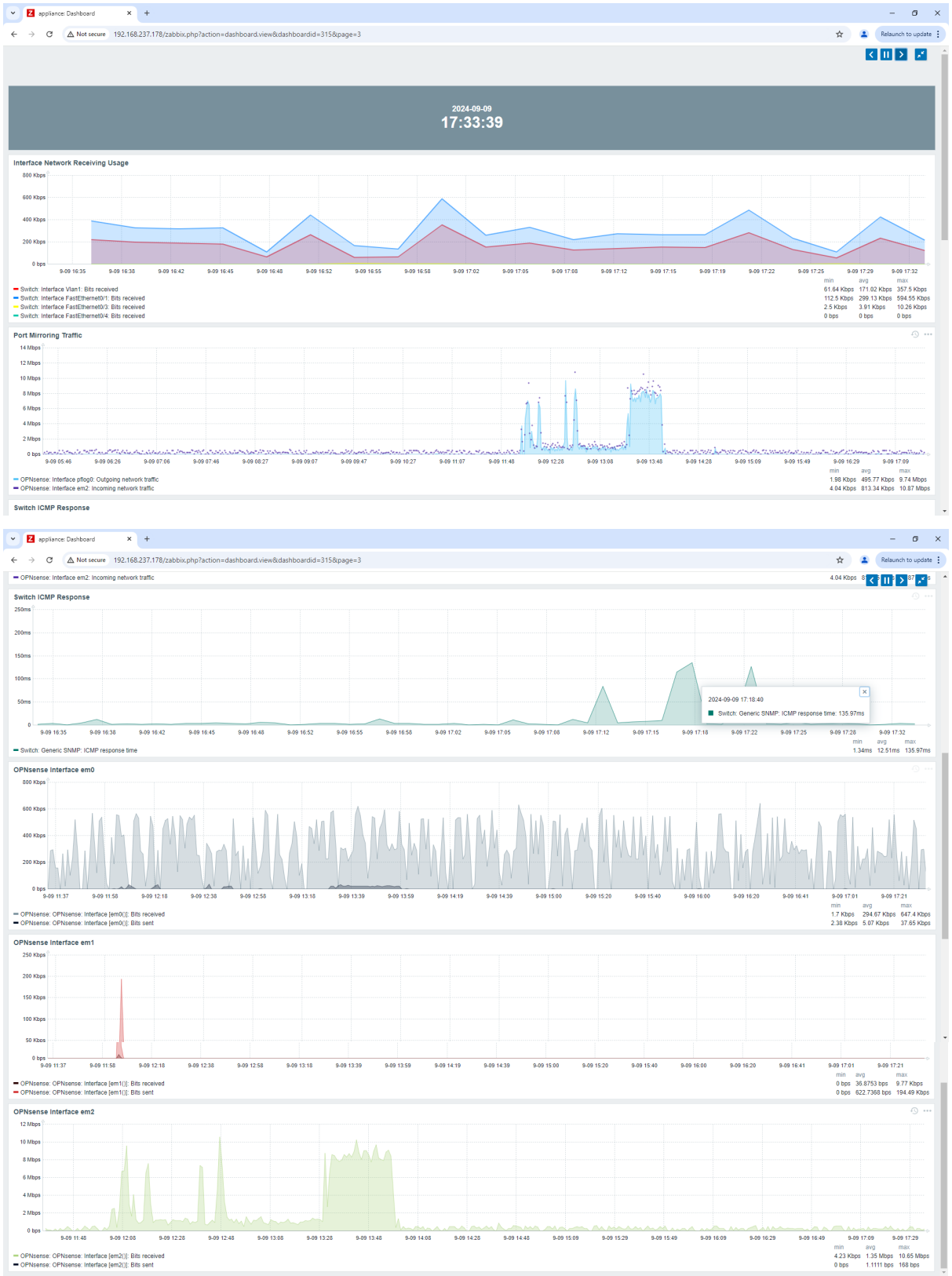


Figure 5.6.3 Dashboard Network Performance

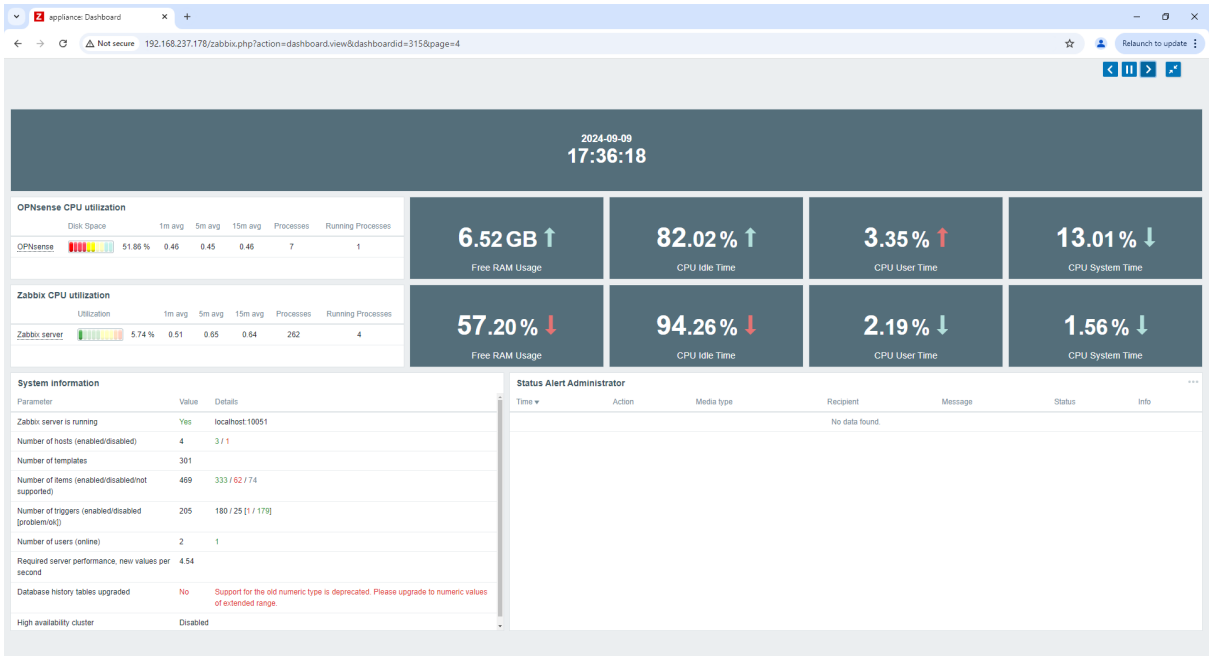


Figure 5.6.4 Dashboard System Performance

5.7 Implementation Issues and Challenges

The project presents several technical challenges that require expertise across various IT domains, making the implementation both time-consuming and demanding. The open-source nature of the tools involved necessitates substantial independent research and configuration, especially when integrating systems like Zabbix and OPNsense.

One key challenge lies in understanding the intergration pathways between Zabbix and OPNsense, particularly how existing sample data is transmitted and what is self-customizable for monitoring. Zabbix utilizes a webhook mechanism to send alert notifications via Telegram, which transmits messages to external servers (such as 8.8.8.8 DNS and Telegram's 149.154.167.220). However, this triggers OPNsense's IDS (Intrusion Detection System), creating a loop of non-stop alerts. To mitigate this, OPNsense's IDS must be configured to allow alerts from Zabbix through these routes.

Additionally, the use of Zabbix agent with an active item key type has led to high CPU performance, which is not conducive to the project's objectives. To resolve this, a shift to using Zabbix sender and trapper is required to reduce the system load and prevent overloading.

Lastly, a crucial goal of the project is to integrate Zabbix with ChatGPT via API to enhance the alerting system by providing recommendations to users. This requires a deeper

understanding of Zabbix's current alerting communication methods and improving them for seamless integration with ChatGPT.

5.8 Concluding Remark

This integrated network security system demonstrates a comprehensive approach to threat detection and response. At its core, the OPNsense Intrusion Detection System (IDS) monitors port mirroring, promptly detecting and relaying threat information to Zabbix. Zabbix then processes this data, providing visualized statistics on its dashboard.

The system's capabilities extend beyond mere threat detection. Zabbix employs SNMP to monitor network traffic across switches and the OPNsense firewall, offering a holistic view of network traffic performance. This data is also visualized on the dashboard, providing network administrators with real-time insights into the network's status. A key strength of this system lies in its automated response mechanism. When Zabbix receives security logs from OPNsense, it triggers an alert process. This process integrates with GPT via API to generate recommended solutions. Both the alert and the AI-generated recommendations are then sent to the network administrator through Telegram, enabling swift, informed responses to potential threats.

Furthermore, the system is designed to detect and alert on abnormal network traffic patterns, providing an additional layer of security awareness.

This multi-faceted approach combines real-time monitoring, AI-assisted threat analysis, and instant communication to create a robust, intelligent, and responsive network monitoring and alerting system.

This setup not only ensures efficient real-time threat detection and network traffic analysis but also empowers network administrators with instant alerts and tailored recommendations. By leveraging various technologies and integrating them seamlessly, this system offers a powerful tool for network administrators to maintain network integrity and respond rapidly to potential security threats.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

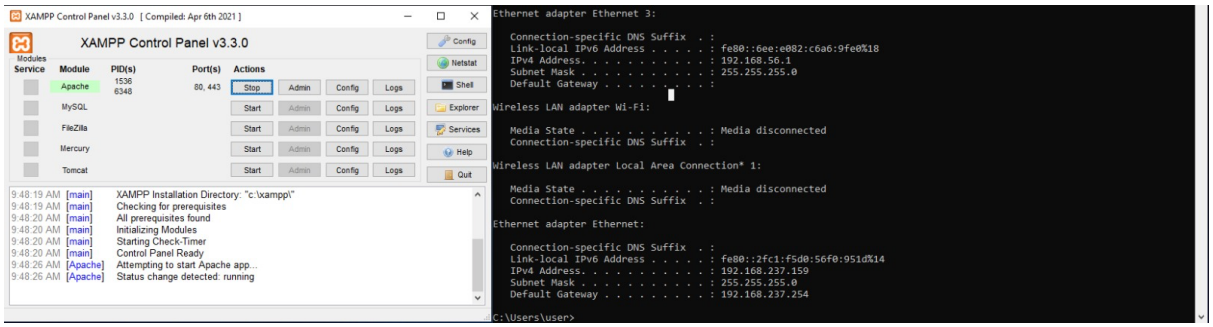


Figure 6.1.1 Xampp Server

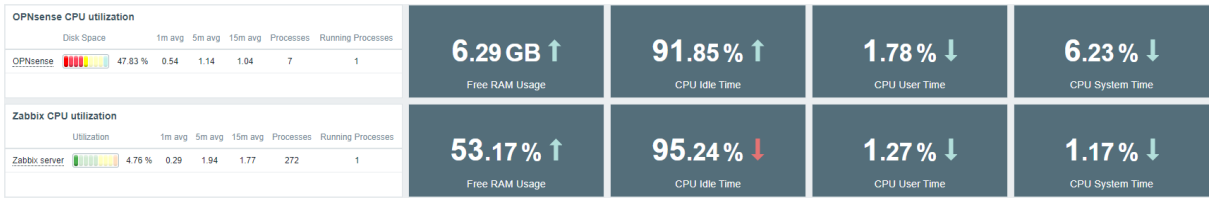


Figure 6.1.2 Zabbix and OPNsense System Performance

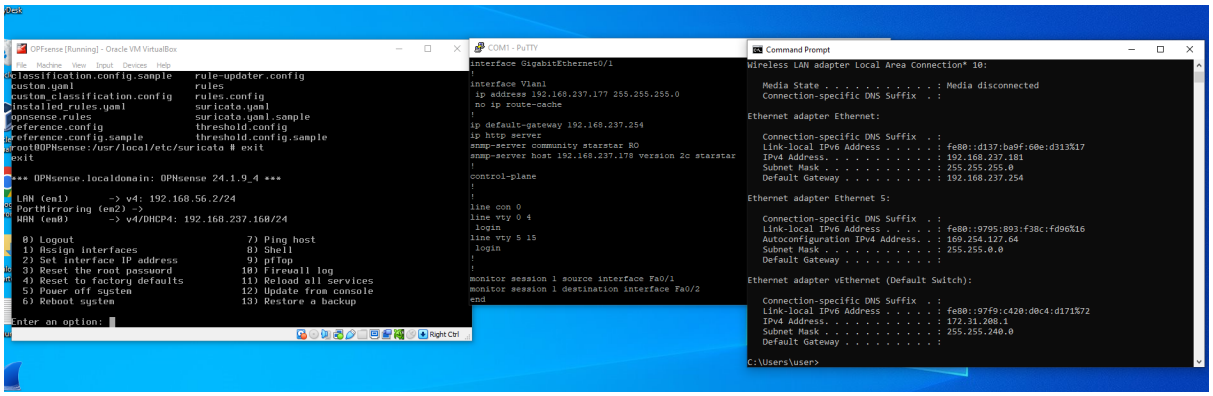


Figure 6.1.3 OPNsense and Physical Device Setup

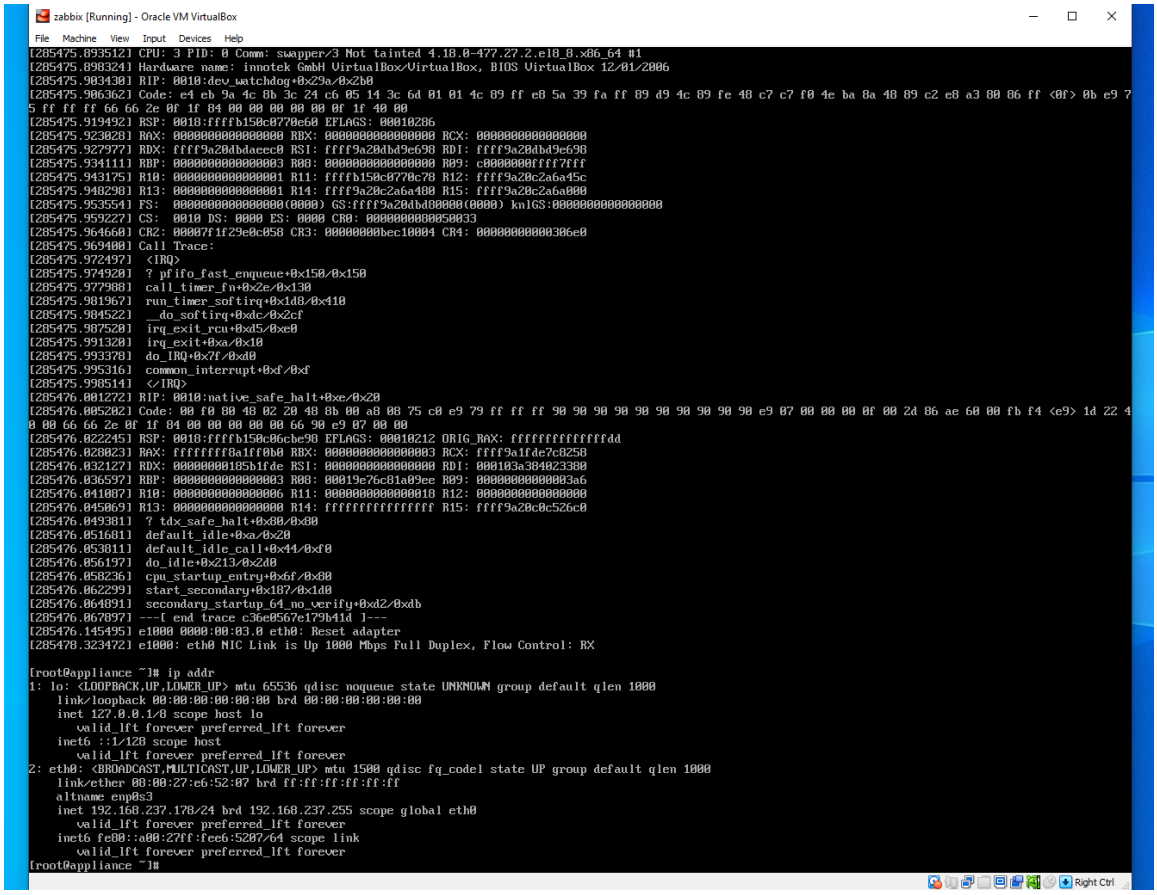


Figure 6.1.4 Zabbix Server Setup

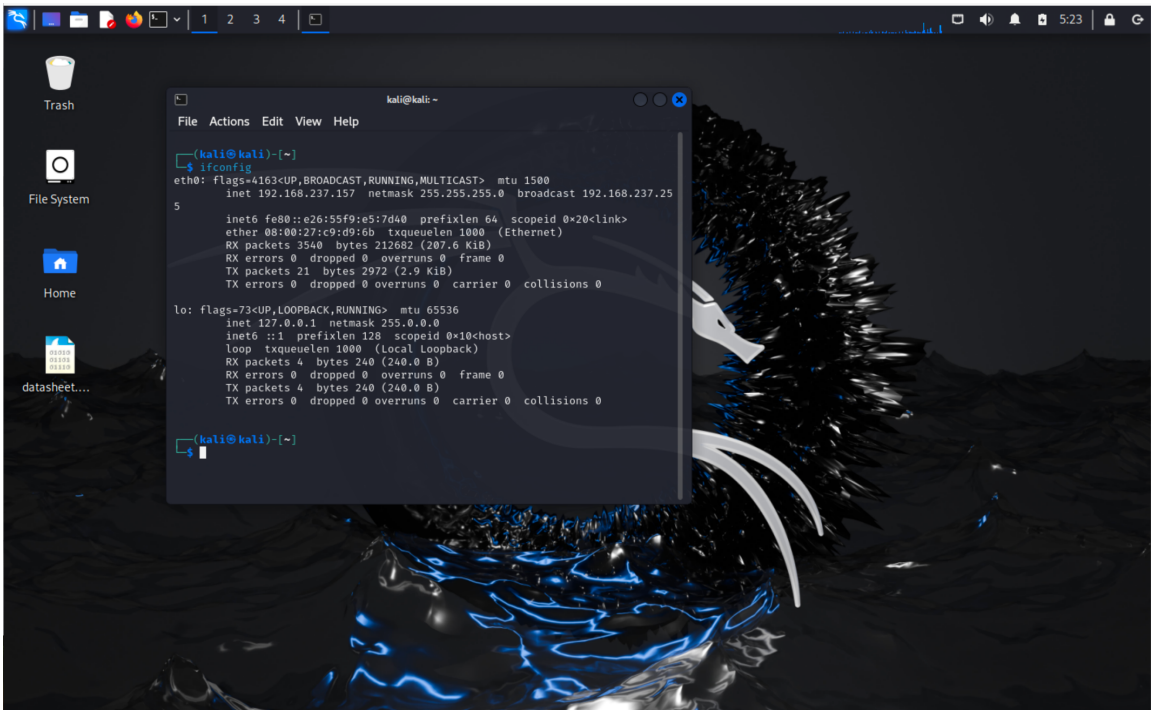


Figure 6.1.5 Kali Linux Setup

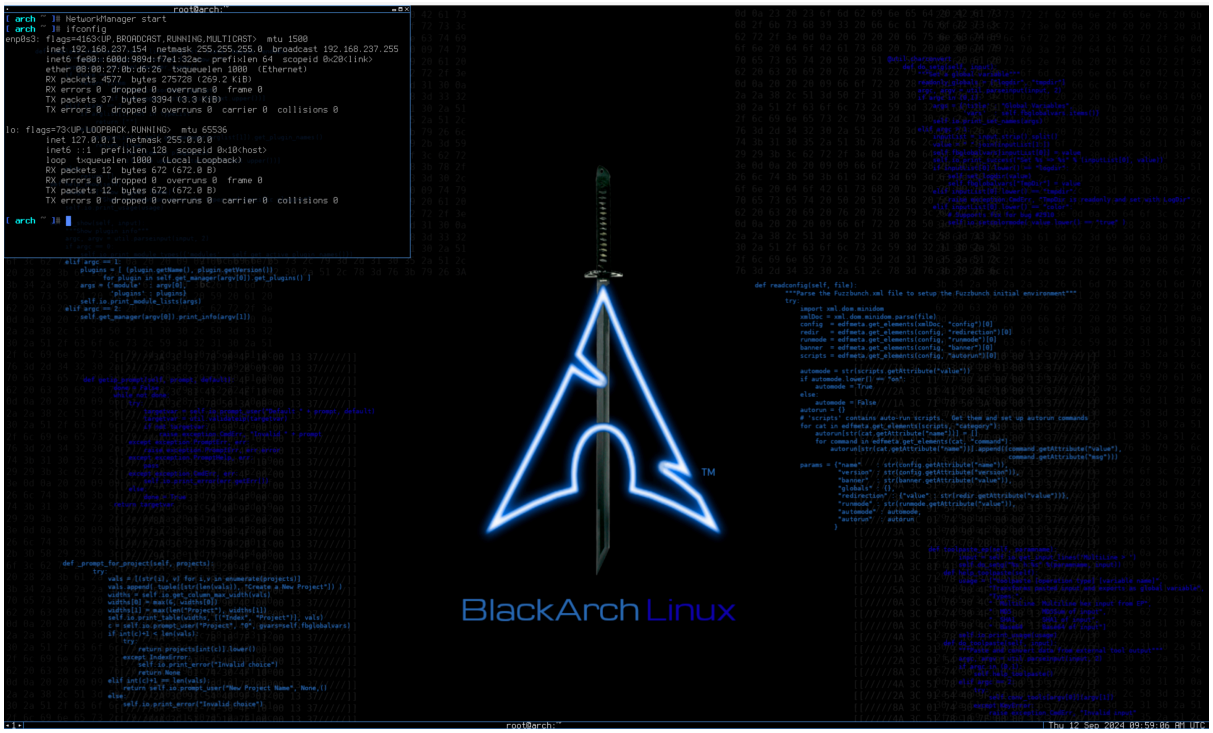


Figure 6.1.6 Arch Linux Setup

6.2 Testing Setup and Result

6.2.1 Web Application Attack

ZAProxy

An open-source security tool designed to intrusion and detect vulnerabilities in web applications, part of the OWASP.

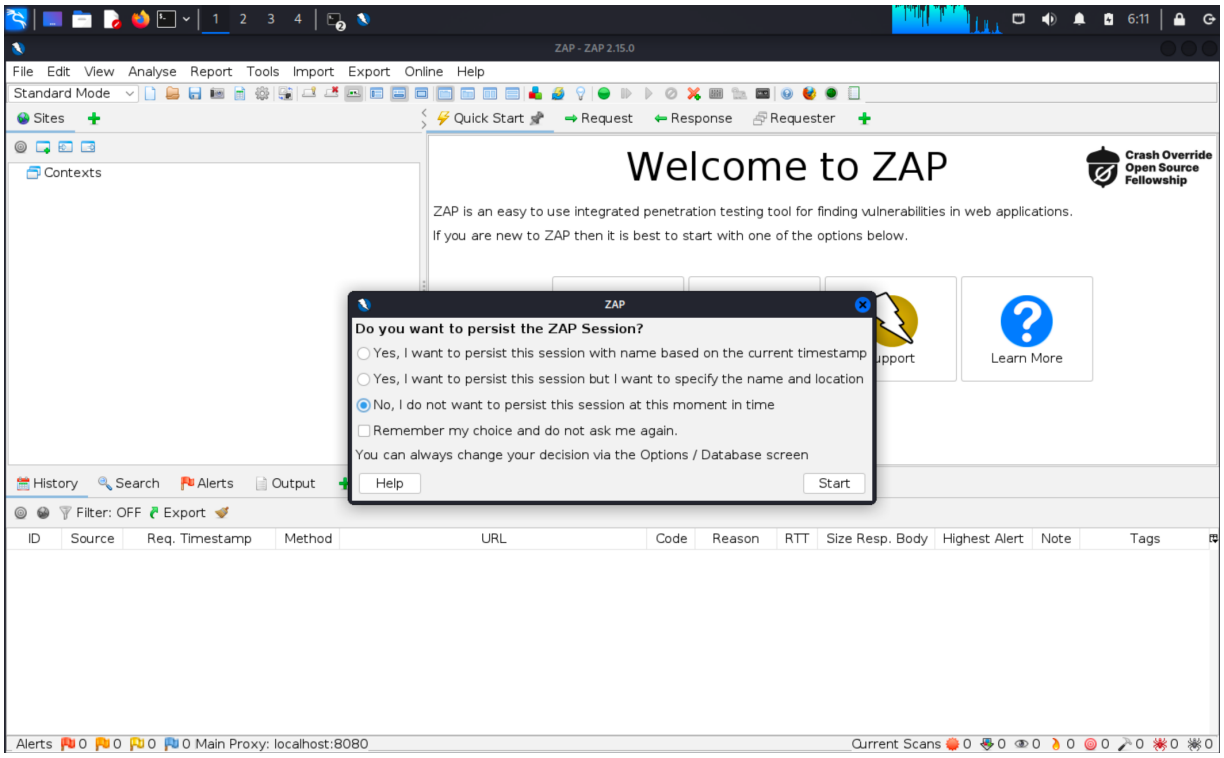
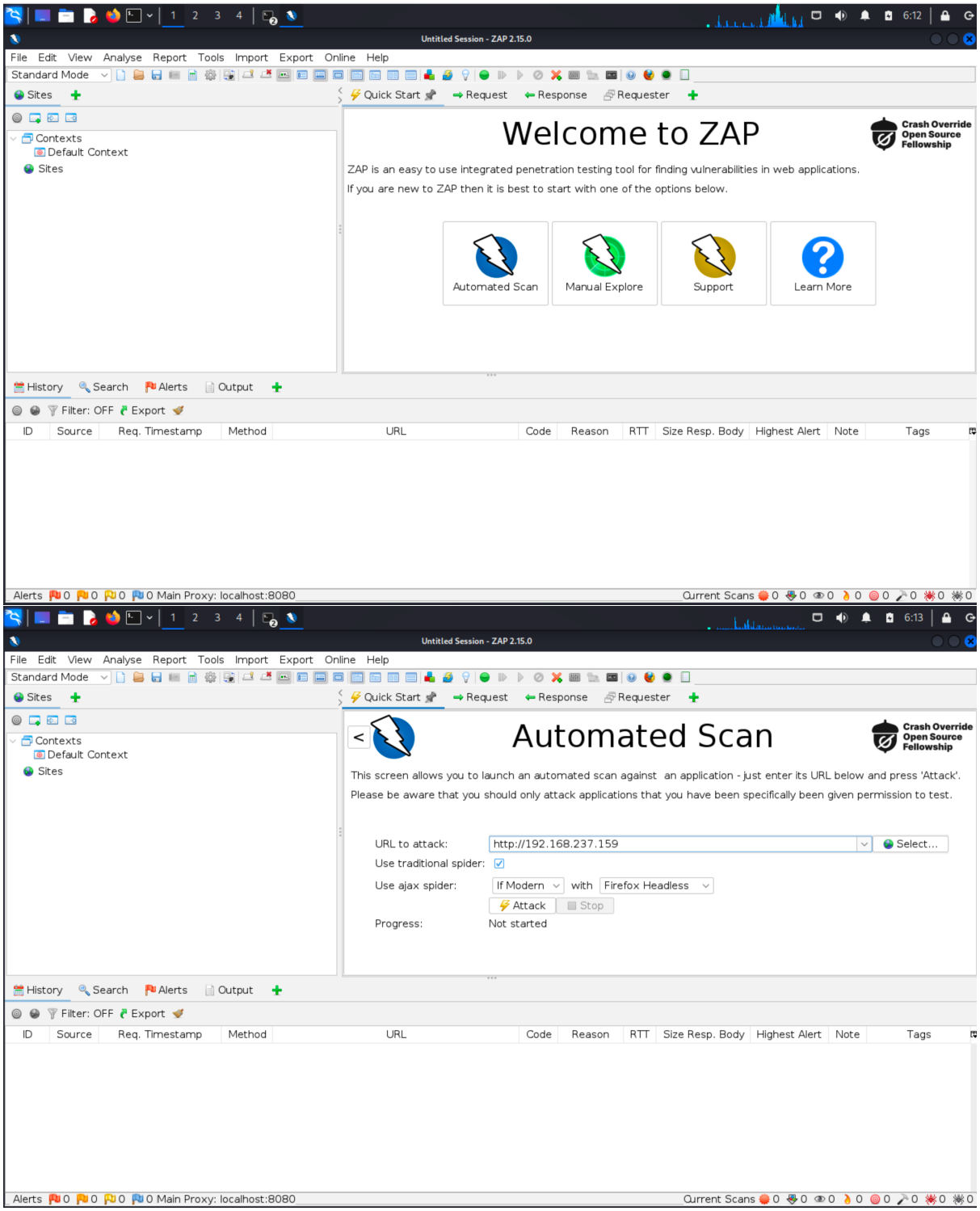


Figure 6.2.1 ZAProxy

Open ZAProxy tools in the kali linux, select with 'No, I do not want to persist the session at this moment in time.



Go with the Automated Scan and input the URL with the ip address ‘http://192.168.237.159’ and direct click on attack to do the vulnerabilities scan.

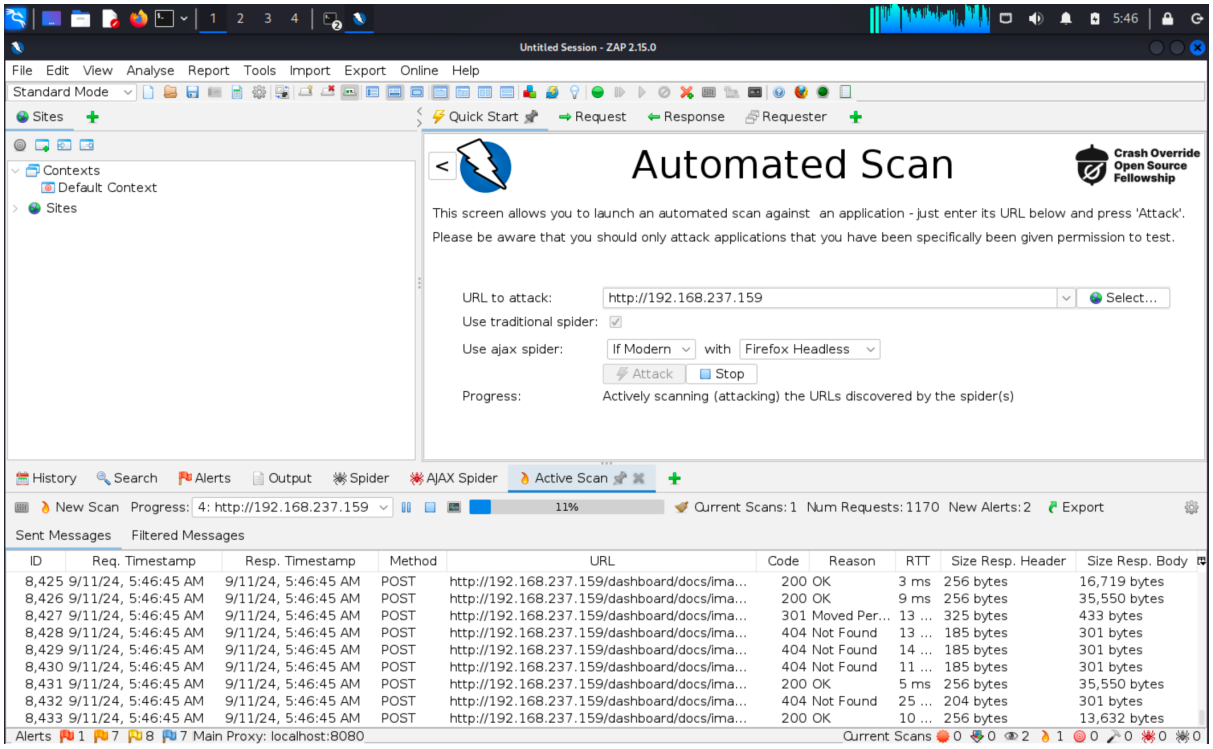


Figure 6.2.2 ZAProxy Intrusion Scan

ZAProxy provide an automation framework support with a series of intrusion scanning, included Spidering, Active Scan and Passive Scan to get the information and resource. Figure 6.2.2, ZAProxy is doing the intrusion scanning.

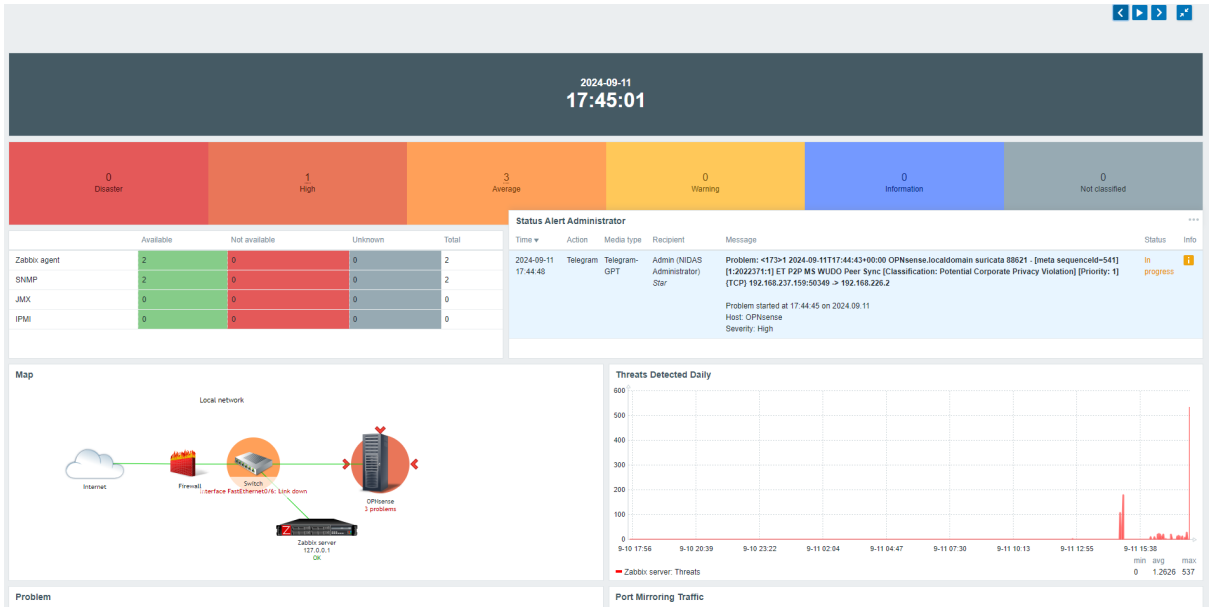


Figure 6.2.3 Zabbix Alert Trigger

While ZAProxy is doing the intrusion scanning, Zabbix receives the intrusion log and triggering the alert to network administrator that detected by OPNsense.

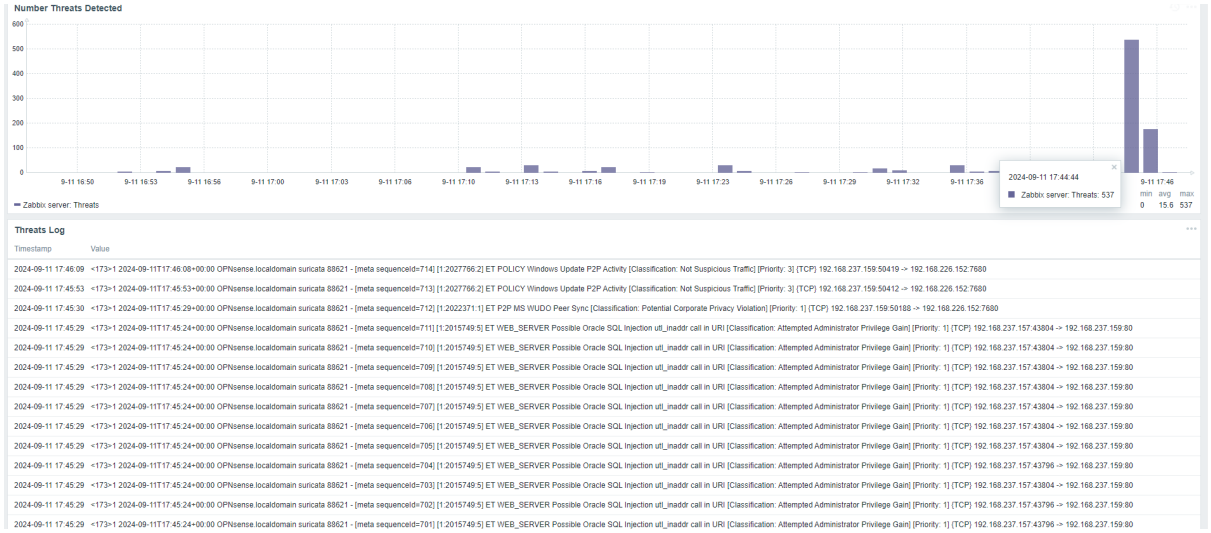


Figure 6.2.4 ZAProxy Number Threats Detected

The Figure 6.2.4 the Number Threats Detected graph shows the 537 maximum number of threats intrusion detected per minute on 2024-09-11 17:44:44.

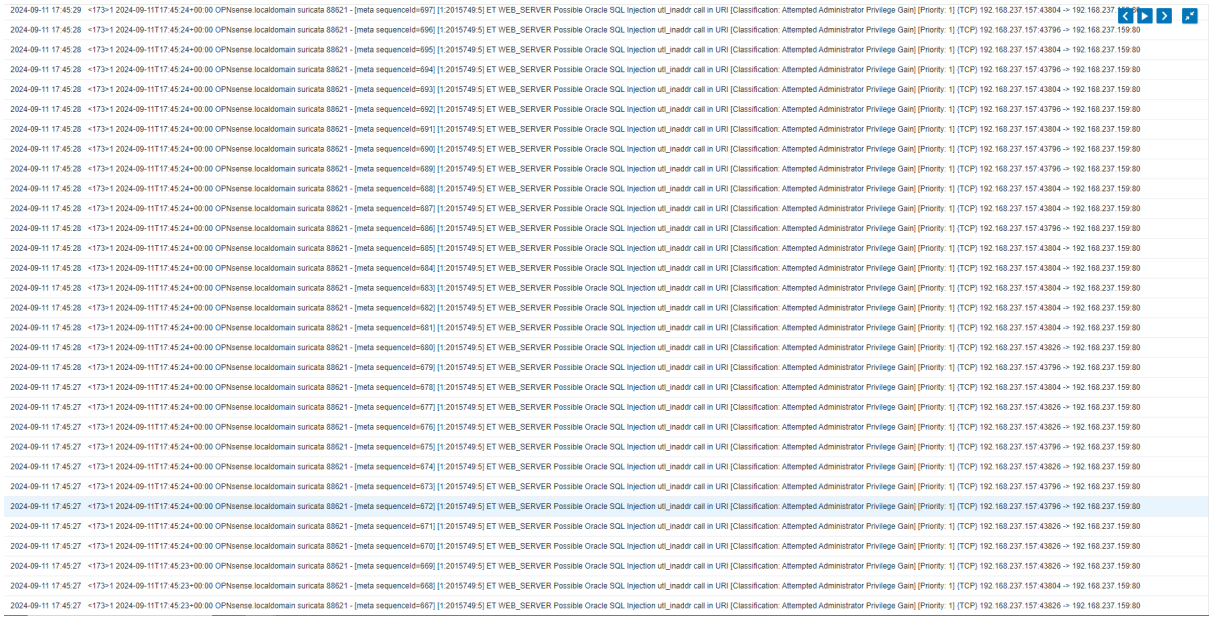


Figure 6.2.5 ZAProxy Information Threats Detected

Threats Log shows in the Figure 6.2.4 and 6.2.5 provide the information of the threads, with time, type of attack, protocols, source IP and destination IP.

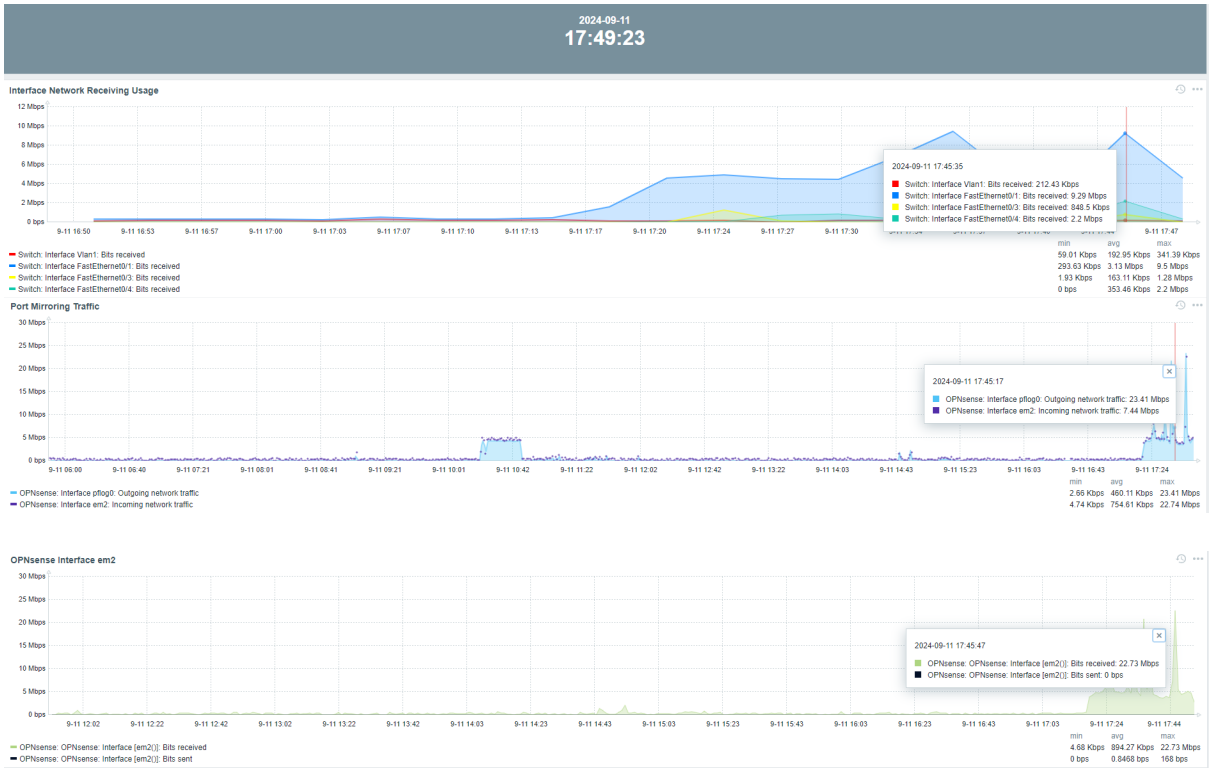
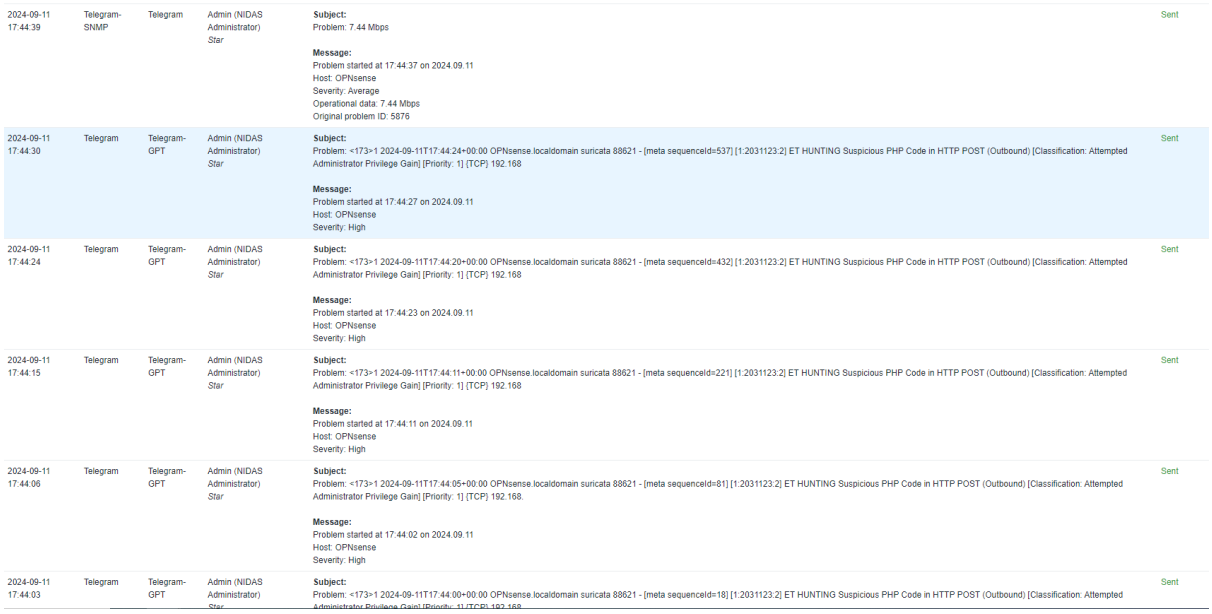


Figure 6.2.6 ZAProxy Intrusion Network Performance

During the ZAProxy intrusion, the switch interface Fa0/1 bits receiving hit to 9.29 Mbps is abnormal compare with normal performance of the interface. Port Mirroring Traffic interface pflog0 Outgoing network traffic also hit with high usage 23.41 Mbps means that the network is having an intrusion, it represent to the intrusion traffic when OPNsense is operating in conjunction with an IDS. The OPNsense interface [em2()] Bits received hits to 22.73 Mbps, means there is a big intrusion packet data.



2024-09-11 17:44:39	Telegram-SNMP	Telegram	Admin (NIDAS Administrator) *Star*	**Subject:** Problem: 7.44 Mbps **Message:** Problem started at 17:44:37 on 2024.09.11 Host: OPNsense Severity: Average Operational data: 7.44 Mbps Original problem ID: 5876	Sent
2024-09-11 17:44:30	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Subject:** Problem: <173>1 2024-09-11T17:44:24+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=537] [1:2031123:2] ET HUNTING Suspicious PHP Code in HTTP POST (Outbound) [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168 **Message:** Problem started at 17:44:27 on 2024.09.11 Host: OPNsense Severity: High	Sent
2024-09-11 17:44:24	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Subject:** Problem: <173>1 2024-09-11T17:44:20+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=432] [1:2031123:2] ET HUNTING Suspicious PHP Code in HTTP POST (Outbound) [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168 **Message:** Problem started at 17:44:23 on 2024.09.11 Host: OPNsense Severity: High	Sent
2024-09-11 17:44:15	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Subject:** Problem: <173>1 2024-09-11T17:44:11+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=221] [1:2031123:2] ET HUNTING Suspicious PHP Code in HTTP POST (Outbound) [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168 **Message:** Problem started at 17:44:11 on 2024.09.11 Host: OPNsense Severity: High	Sent
2024-09-11 17:44:06	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Subject:** Problem: <173>1 2024-09-11T17:44:05+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=81] [1:2031123:2] ET HUNTING Suspicious PHP Code in HTTP POST (Outbound) [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168. **Message:** Problem started at 17:44:02 on 2024.09.11 Host: OPNsense Severity: High	Sent
2024-09-11 17:44:03	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Subject:** Problem: <173>1 2024-09-11T17:44:00+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=18] [1:2031123:2] ET HUNTING Suspicious PHP Code in HTTP POST (Outbound) [Classification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168	Sent

Figure 6.2.7 ZAProxy Intrusion Alert

According to Figures 6.2.5 and 6.2.7, two types of attacks were detected. One shows the possibility of SQL Injection, while the other indicates Suspicious PHP code in HTTP POST. These two detections are core features of Deep Packet Inspection (DPI). When analyzing network traffic, Suricata deeply inspects application layer data in the packets. For instance, the content of the HTTP POST request was judged to contain suspicious PHP code and SQL code 'utl_inaddr' that was flagged as a potential attack. This type of SQL code attempts to embed function calls within the URI of HTTP requests to perform SQL injection attacks. This deep inspection of packet contents, particularly at the application layer, demonstrates the power of DPI in detecting sophisticated attacks that might otherwise go unnoticed. By analyzing the specific content of HTTP requests and identifying suspicious patterns or known attack signatures, the system can provide early warning of potential security threats, allowing for rapid response and mitigation.

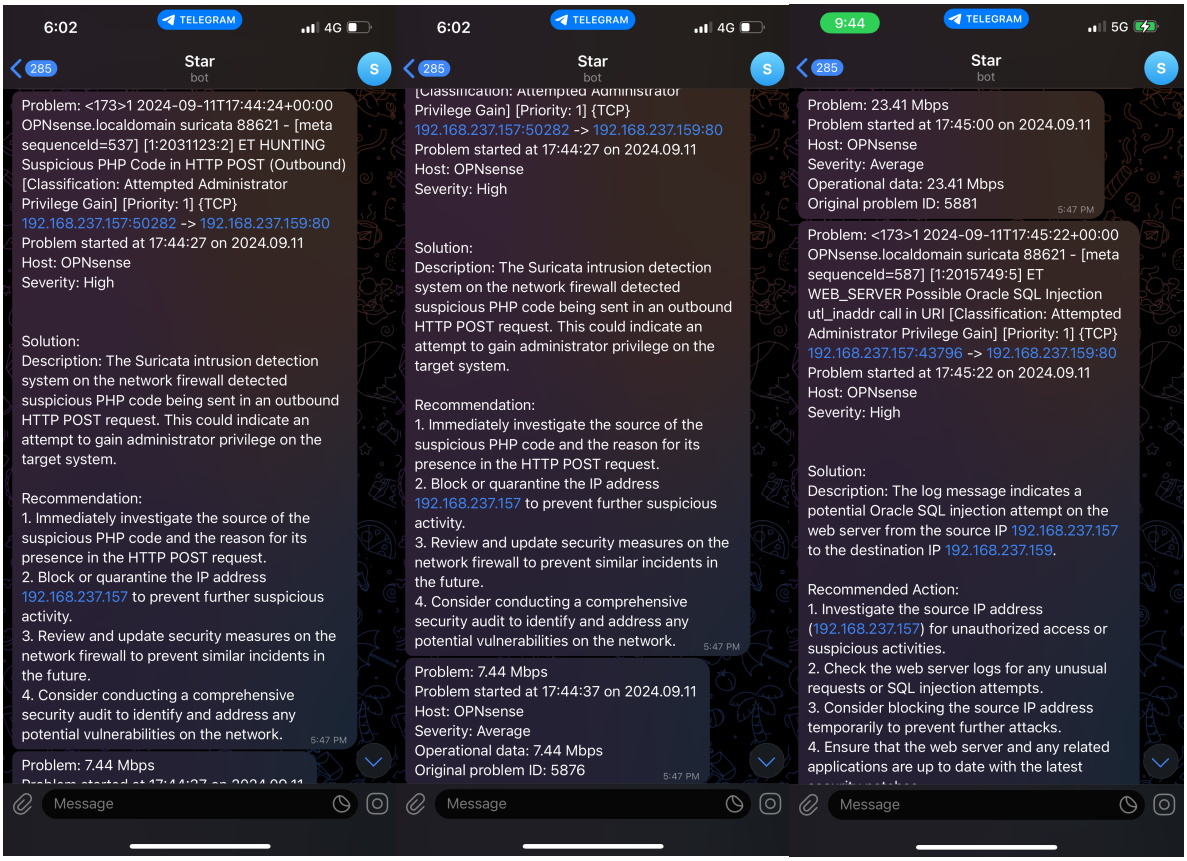


Figure 6.2.8 ZAProxy Intrusion Alert-Telegram

Both of these attacks triggered alert messages, which were then notified to network administrator personnel via Telegram. In Figure 6.2.8, we can see two types of alert messages, one type carries attack information along with recommended solutions, and its severity is marked as high. The other type shows abnormally frequent messages about network traffic status is in high bandwidth, with an average severity level.

6.2.2 Port Scanning

NMAP

Network Mapper is an open source network scanning and security tool, widely used for network discovery and security assessment. It can scan the network and identify connected devices, operating systems, services, open ports and other information.

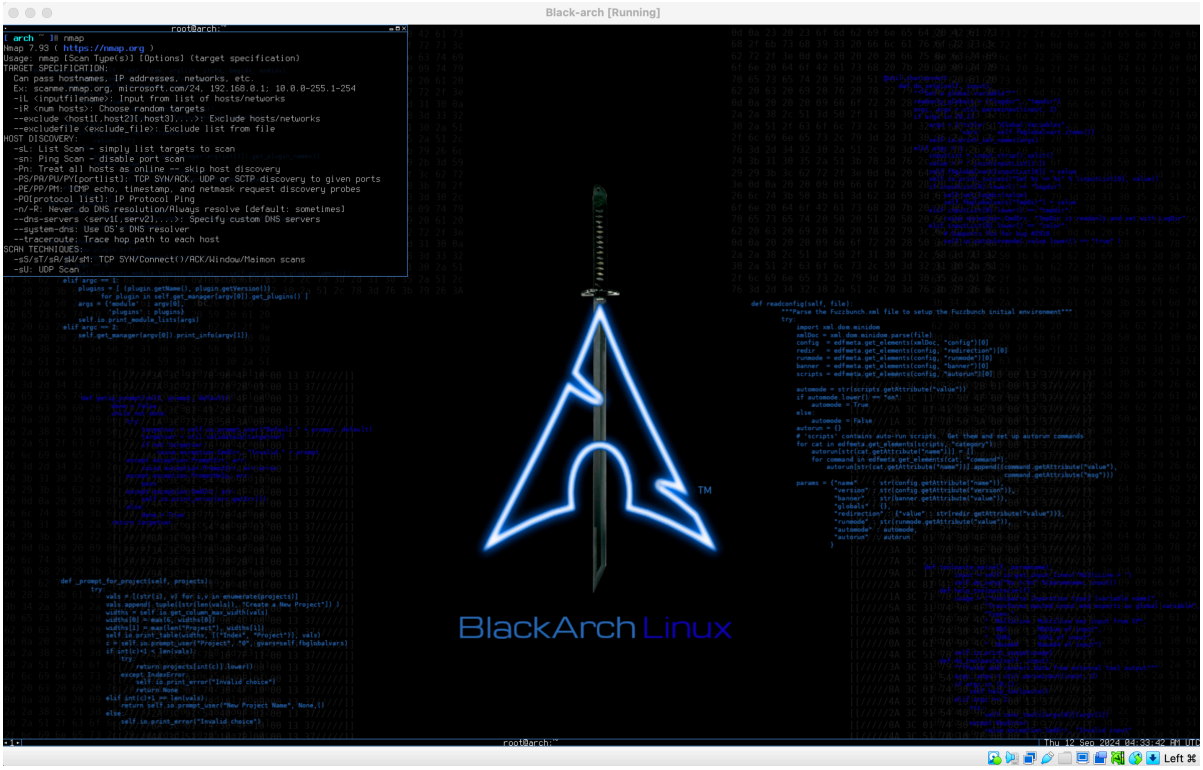


Figure 6.2.9 Nmap

Open Nmap in the Arch Linux, enter the target device ip or network range to scan.

Figure 6.2.10 Nmap TCP-SYN

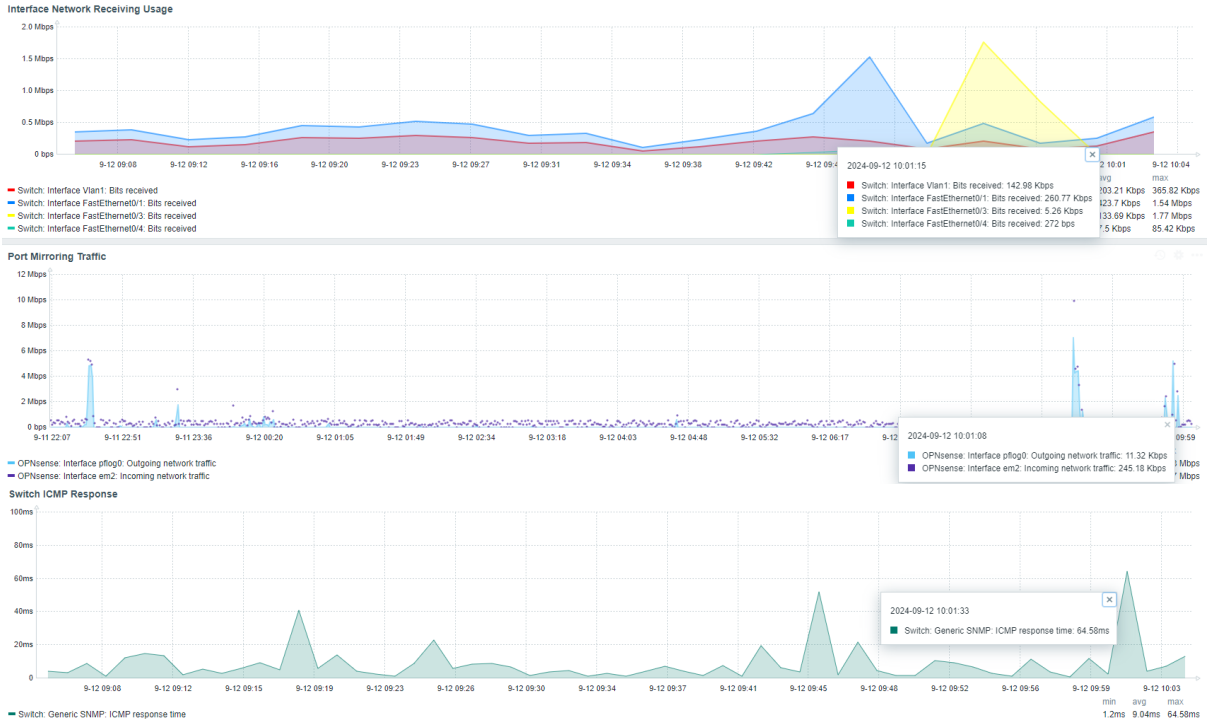
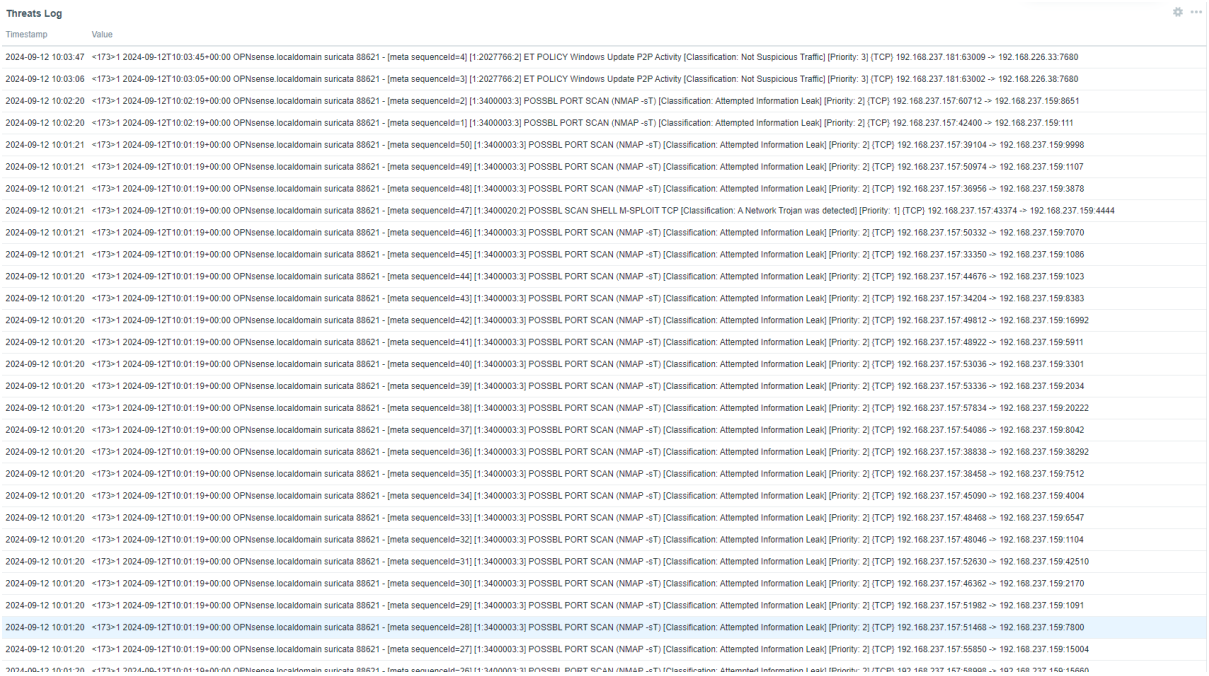


Figure 6.2.11 TCP-SYN Network Performance

Based on the network data shown in Figure 6.2.11, it appears that the Nmap scan did not cause significant stress on the network overall. Specifically the traffic for interface Fa0/1 Bits received and interface pflog outgoing traffic did not show a significant increase. However, during the same time period, the ICMP response time was noticeably affected. The ICMP response value increased to 64.58ms, which is considered an abnormal state. This data suggests that while the Nmap scan didn't cause a substantial increase in overall network traffic, it did have a specific impact on ICMP responses. The increase in ICMP response time to 64.58ms indicates that the system was experiencing some level of strain in processing ICMP packets during the scan.

Figure 6.2.12 TCP-SYN Threats Detected

Figure 6.2.12 shows that Nmap scanning caused 50 threats within one minute.

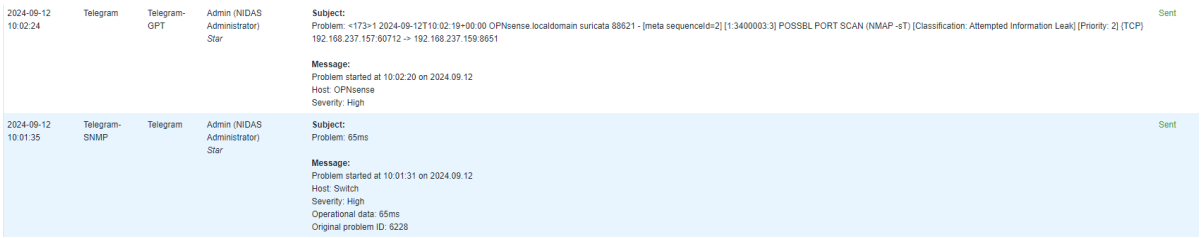


Threats Log

Timestamp	Value
2024-09-12 10:03:47	<173>1 2024-09-12T10:03:45+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=4] [1:2027766:2] ET POLICY Windows Update P2P Activity [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 192.168.237.181:63009 -> 192.168.226.33:7680
2024-09-12 10:03:06	<173>1 2024-09-12T10:03:05+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=3] [1:2027766:2] ET POLICY Windows Update P2P Activity [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 192.168.237.181:63002 -> 192.168.226.38:7680
2024-09-12 10:02:20	<173>1 2024-09-12T10:02:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=2] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:60712 -> 192.168.237.159:8651
2024-09-12 10:02:20	<173>1 2024-09-12T10:02:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=1] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:42400 -> 192.168.237.159:111
2024-09-12 10:01:21	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=50] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:39104 -> 192.168.237.159:9998
2024-09-12 10:01:21	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=49] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:50974 -> 192.168.237.159:1107
2024-09-12 10:01:21	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=48] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:36956 -> 192.168.237.159:3878
2024-09-12 10:01:21	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=47] [1:3400020:2] POSSBL SCAN SHELL M-SPLOIT TCP [Classification: A Network Trojan was detected] [Priority: 1] {TCP} 192.168.237.157:43374 -> 192.168.237.159:4444
2024-09-12 10:01:21	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=46] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:50332 -> 192.168.237.159:7070
2024-09-12 10:01:21	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=45] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:33350 -> 192.168.237.159:1086
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=44] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:44676 -> 192.168.237.159:1023
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=43] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:34204 -> 192.168.237.159:8383
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=42] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:49812 -> 192.168.237.159:16992
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=41] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:48922 -> 192.168.237.159:5911
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=40] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:53036 -> 192.168.237.159:3301
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=39] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:53336 -> 192.168.237.159:2034
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=38] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:57834 -> 192.168.237.159:20222
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=37] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:54086 -> 192.168.237.159:8042
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=36] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:38838 -> 192.168.237.159:38292
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=35] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:38458 -> 192.168.237.159:7512
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=34] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:45090 -> 192.168.237.159:4004
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=33] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:48468 -> 192.168.237.159:6547
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=32] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:48046 -> 192.168.237.159:1104
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=31] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:52630 -> 192.168.237.159:42510
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=30] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:46362 -> 192.168.237.159:2170
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=29] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:51982 -> 192.168.237.159:1091
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=28] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:51468 -> 192.168.237.159:7800
2024-09-12 10:01:20	<173>1 2024-09-12T10:01:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=27] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:55850 -> 192.168.237.159:15004

Figure 6.2.13 TCP-SYN Threats Information

From Figure 6.2.13, it can be seen that most of the 50 Nmap threats triggered detections are related to TCP attempted information leak. The IDS based on DPI analysis have too many similar information leak triggers, subsequently triggered a second detection of 'network Trojan was detected'. The Nmap attack source all came from the IP 192.168.237.157, and these Nmap attack attempted to obtain information from the device with IP 192.168.237.159.



2024-09-12 10:02:24	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Subject:** Problem: <173>1 2024-09-12T10:02:19+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=2] [1:3400003:3] POSSBL PORT SCAN (NMAP -sT) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:60712 -> 192.168.237.159:8651 **Message:** Problem started at 10:02:20 on 2024.09.12 Host: OPNsense Severity: High	Sent
2024-09-12 10:01:35	Telegram-SNMP	Telegram	Admin (NIDAS Administrator) *Star*	**Subject:** Problem: 65ms **Message:** Problem started at 10:01:31 on 2024.09.12 Host: Switch Severity: High Operational data: 65ms Original problem ID: 6228	Sent

Figure 6.2.14 TCP-SYN Intrusion Alert

From Figure 6.2.14, the Nmap detection information and network traffic alerts were triggered, and the alert information was successfully sent to the network administrator.

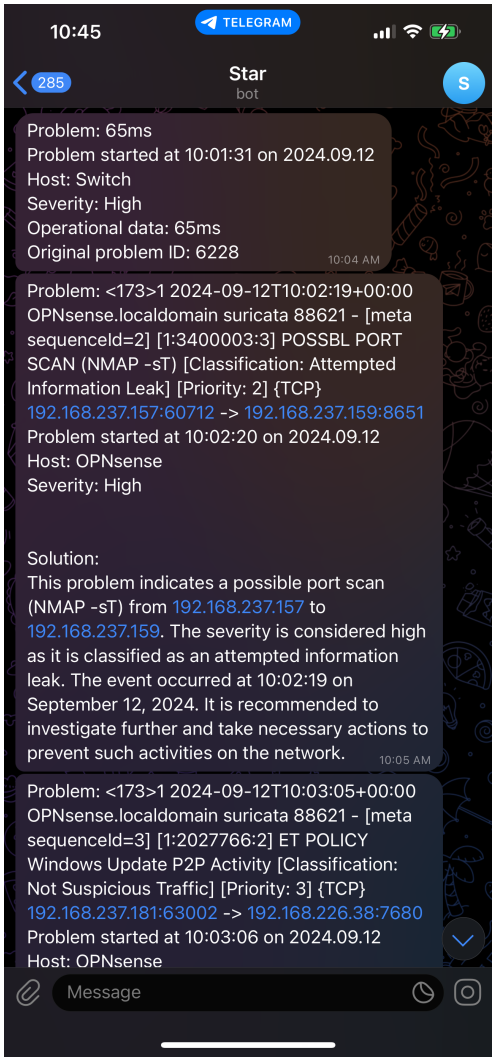


Figure 6.2.15 TCP-SYN Intrusion Alert-Telegram

In Figure 6.2.15, the network administrator receives an intrusion detection alert via Telegram. This alert provides information and severity about the intrusion as well as recommended solutions to help relevant personnel quickly understand the issue. This approach demonstrates an effective use of automated alert systems in cybersecurity. By sending alerts through Telegram, a widely used messaging platform, the system ensures that network administrators can receive critical information in real-time, regardless of their location.

NMAP Xmas

Figure 6.2.16 Nmap Xmas

Nmap Xmas scans use special flags (FIN, PSH, URG) to send non-standard TCP packets that may bypass certain firewall rules and rely on the target system's processing of abnormal TCP packets to determine the port status.

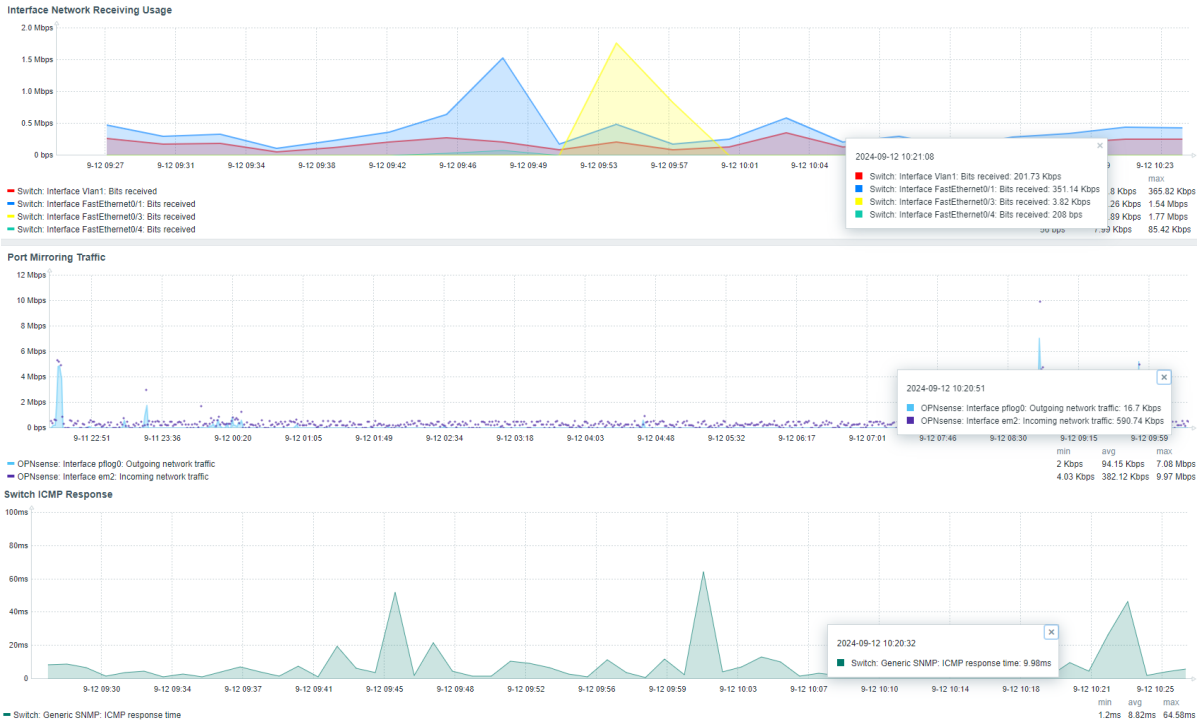


Figure 6.2.17 Xmas Network Performance

Based on the network flow shown in Figure 6.2.17, it appears that the Nmap -sX (Xmas scan) did not cause significant stress on the network overall. Specifically, the traffic for interface Fa0/1 Bits received and interface pflog outgoing traffic did not show a significant increase. However, during the same time period, the ICMP response time was noticeably affected. The ICMP response value increased to 9.98ms, which is considered an abnormal state.

This flow data suggests that while the Nmap Xmas scan didn't cause a substantial increase in overall network traffic, it did have a specific impact on ICMP responses. The increase in ICMP response time to 9.98ms indicates that the system was experiencing some level of strain in processing packets during the scan.

However, due to the unique nature of the Xmas scan, systems processing these abnormal packets may not respond as frequently as they would to a SYN scan. This results in a lower

overall burden on the network, making it easier to overlook and more difficult to detect. The Xmas scan's ability to cause subtle changes in network behavior without significantly increasing overall traffic highlights its stealthy nature. This characteristic makes it a potentially dangerous tool for attackers, as it can probe network defenses without triggering many traditional traffic-based alarms.

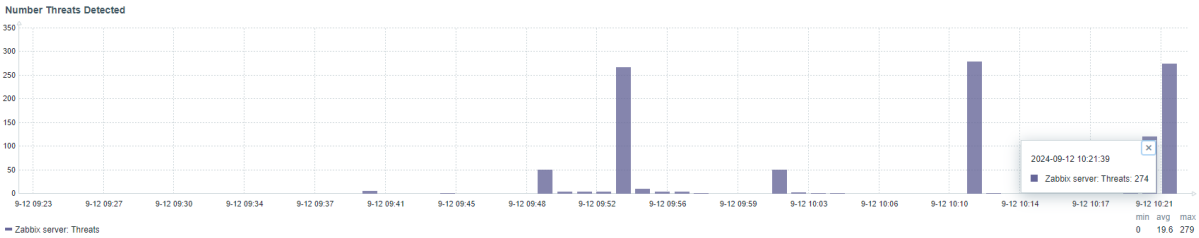


Figure 6.2.18 Xmas Threats Detected

Figure 6.2.18 shows that Nmap -sX scanning caused 274 threats within one minute.

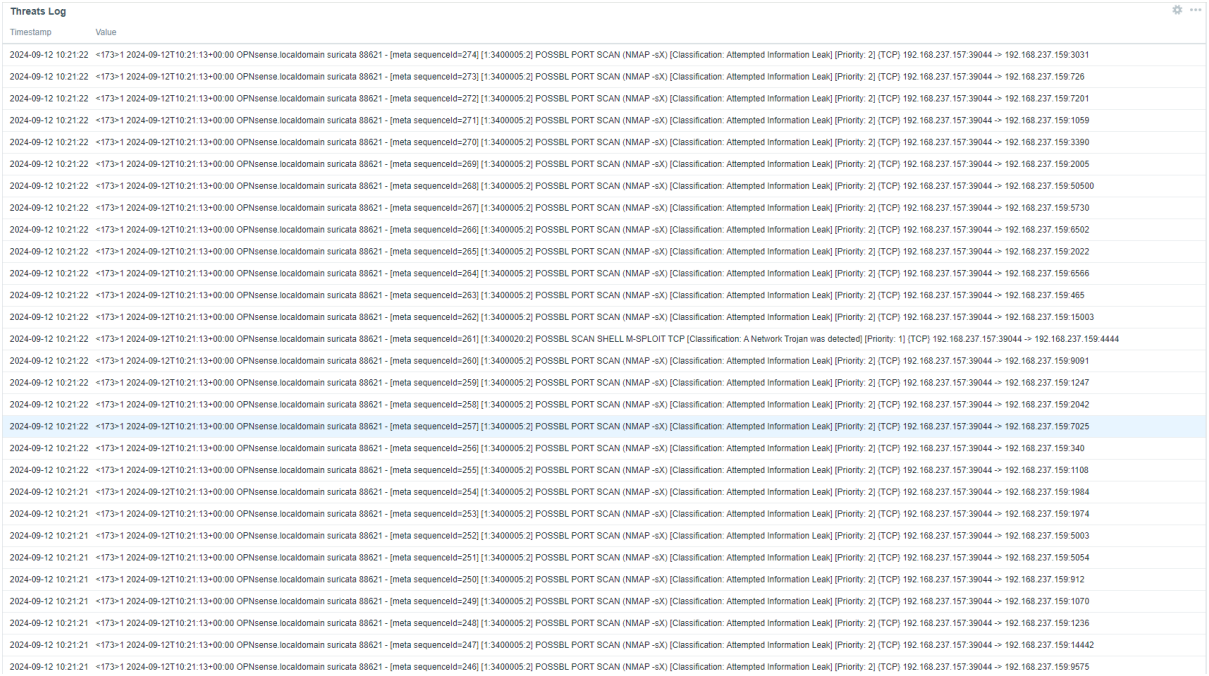
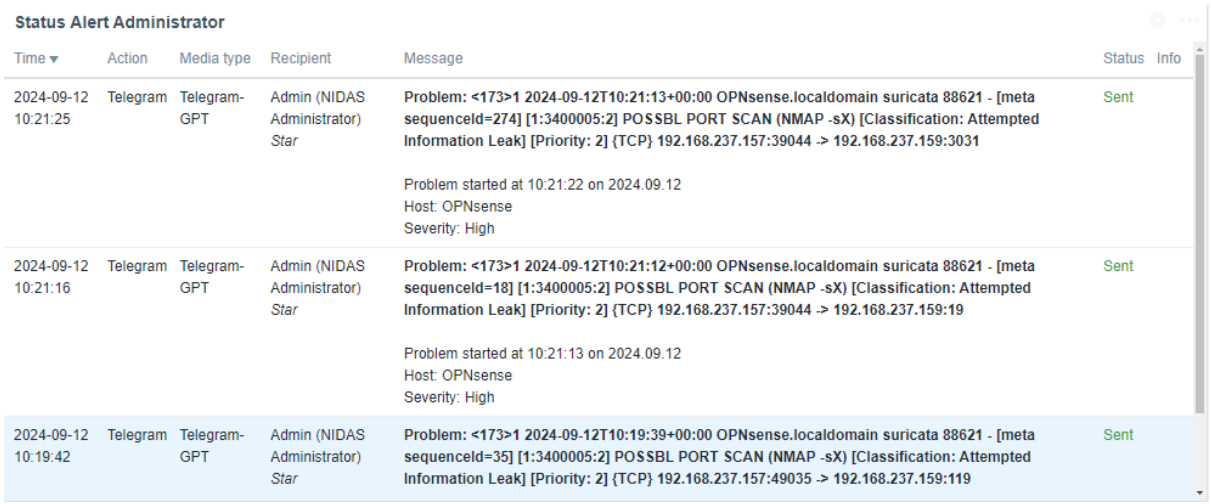


Figure 6.2.19 Xmas Threats Information

From Figure 6.2.19, it can be seen that most of the 274 Nmap -sX threats triggered detections are related to TCP attempted information leak. The IDS based on DPI analysis have too many similar information leak triggers, subsequently triggered a second detection of 'network Trojan was detected'. The Nmap -sX attack source all came from the IP 192.168.237.157, and these Nmap -sX attack attempted to obtain information from the device with IP 192.168.237.159.



Status Alert Administrator

Time ▾	Action	Media type	Recipient	Message	Status	Info
2024-09-12 10:21:25	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Problem: <173>1 2024-09-12T10:21:13+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=274] [1:3400005:2] POSSBL PORT SCAN (NMAP -sX) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:39044 -> 192.168.237.159:3031** Problem started at 10:21:22 on 2024.09.12 Host: OPNsense Severity: High	Sent	
2024-09-12 10:21:16	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Problem: <173>1 2024-09-12T10:21:12+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=18] [1:3400005:2] POSSBL PORT SCAN (NMAP -sX) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:39044 -> 192.168.237.159:19** Problem started at 10:21:13 on 2024.09.12 Host: OPNsense Severity: High	Sent	
2024-09-12 10:19:42	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Problem: <173>1 2024-09-12T10:19:39+00:00 OPNsense.localdomain suricata 88621 - [meta sequenceId=35] [1:3400005:2] POSSBL PORT SCAN (NMAP -sX) [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.237.157:49035 -> 192.168.237.159:119**	Sent	

Figure 6.2.20 Xmas Intrusion Alert

From Figure 6.2.20, the Nmap -sX detection information and network traffic alerts were triggered, and the alert information was successfully sent to the network administrator.

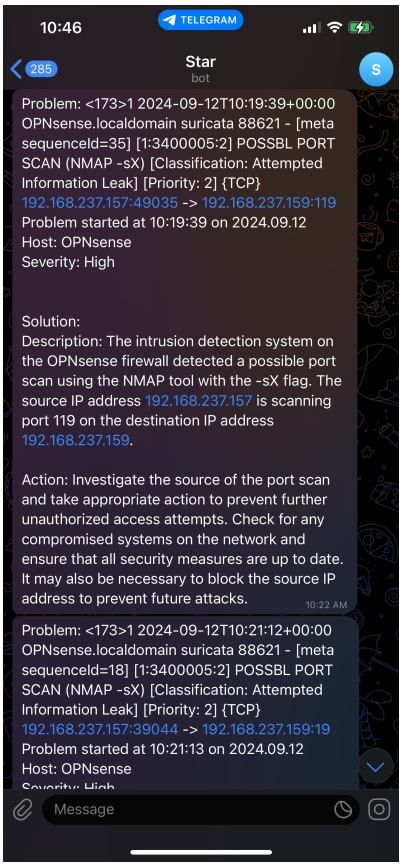


Figure 6.2.21 Xmas Intrusion Alert-Telegram

In figure 6.2.21 shows the Xmas Intrusion Alert message received by the network administrator.

6.2.3 Distributed Denial of Service

ICMP Flood

An ICMP flood attack, also known as a "Ping Flood," involves sending a large number of ICMP Echo requests (pings) to overwhelm the network bandwidth. This can result in slowed or interrupted network communications, and may also exhaust the system's CPU resources.

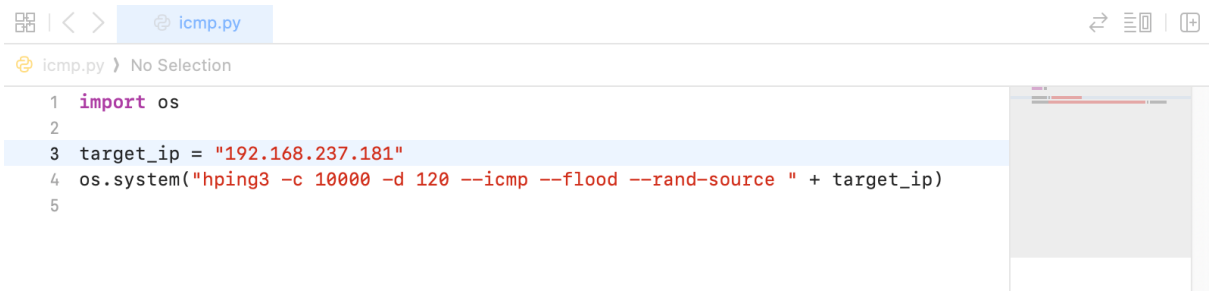


Figure 6.2.22 ICMP Flood Script

Hping3 is a tool, -icmp means ICMP packets, commonly known as "ping" packets, -c 10000 represents sending 10,000 packets, -d 120 specifies a packet size of 120 bytes, --flood means sending packets at the fastest rate, forming a flood attack and --rand-source means using a random source IP address target destination IP address '192.168.237.181'. Execute the script in kali linux.

Figure 6.2.23 ICMP DDoS

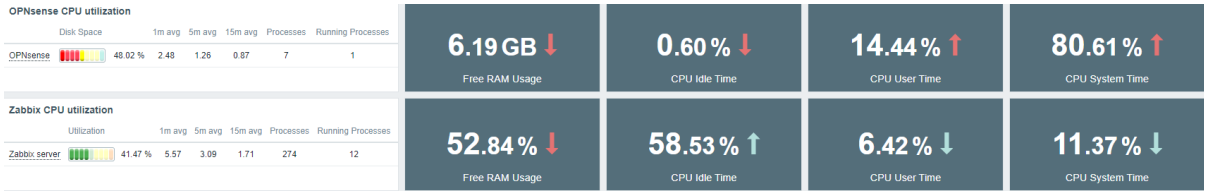


Figure 6.2.24 ICMP DDoS CPU Performance

On Figure 6.2.24 shows after being subjected to an ICMP Flood attack, the CPU's Idle Performance shows a sharp decline. A large portion of the CPU Performance is ineffectively occupied with detecting and processing the ICMP Flood attack.

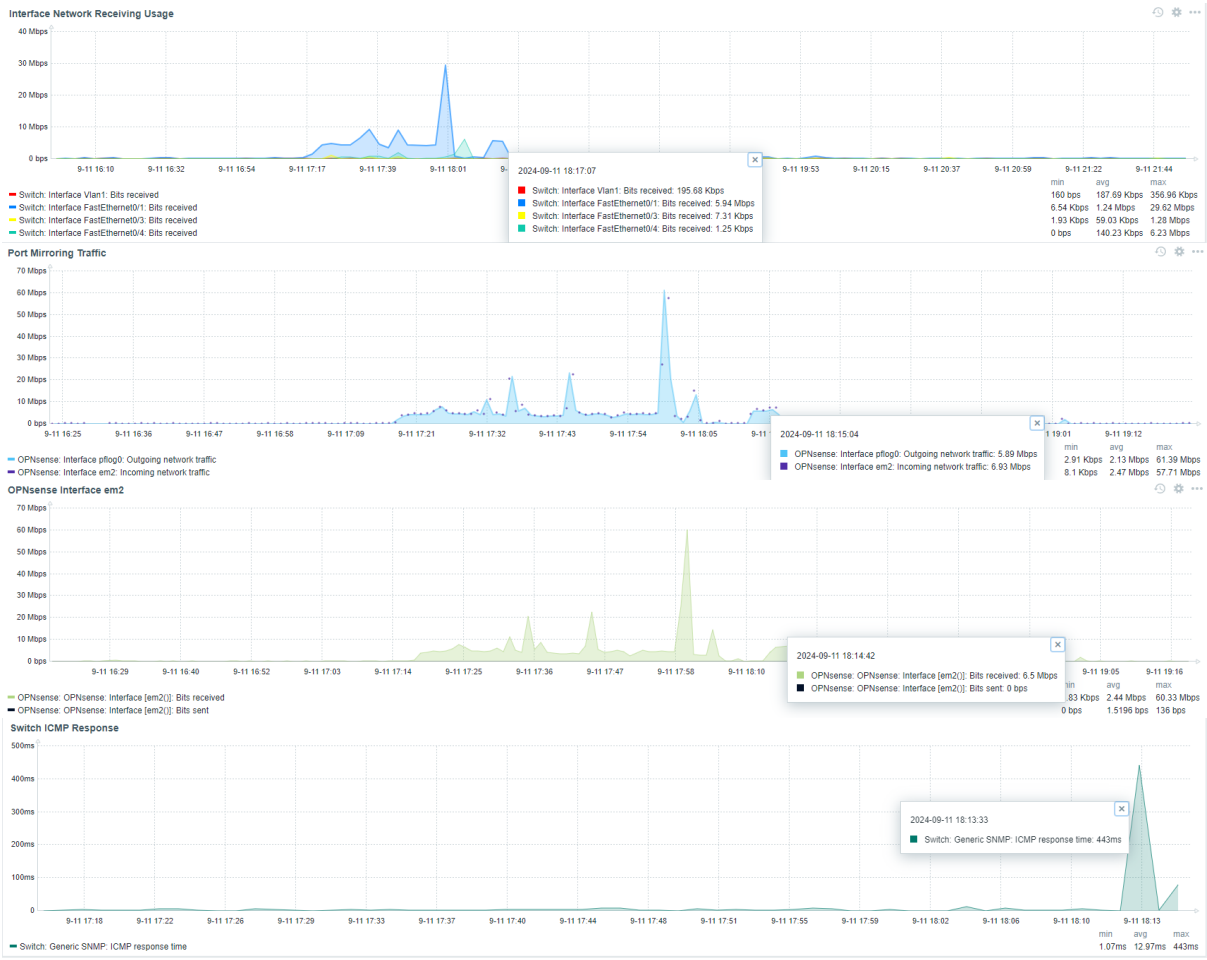


Figure 6.2.25 ICMP DDoS Network Performance

According to Figure 6.2.25, in response to the ICMP Flood attack based on the Interface Fa0/1 bits received, Interface pflog0 Outgoing network traffic, and interface [em2] bandwidth usage show a noticeable increase. However, this increase is not particularly high, generally fluctuating between 5.8Mbps and 7Mbps.In contrast, the ICMP response time shows a direct and significant increase, reaching up to 443ms.

This data illustrates that while the ICMP Flood attack does increase overall network traffic, its most significant impact is on the ICMP response time. The system is forced to process a large number of ICMP Echo requests, which heavily occupies the ICMP response time and consequently slows down network communications.

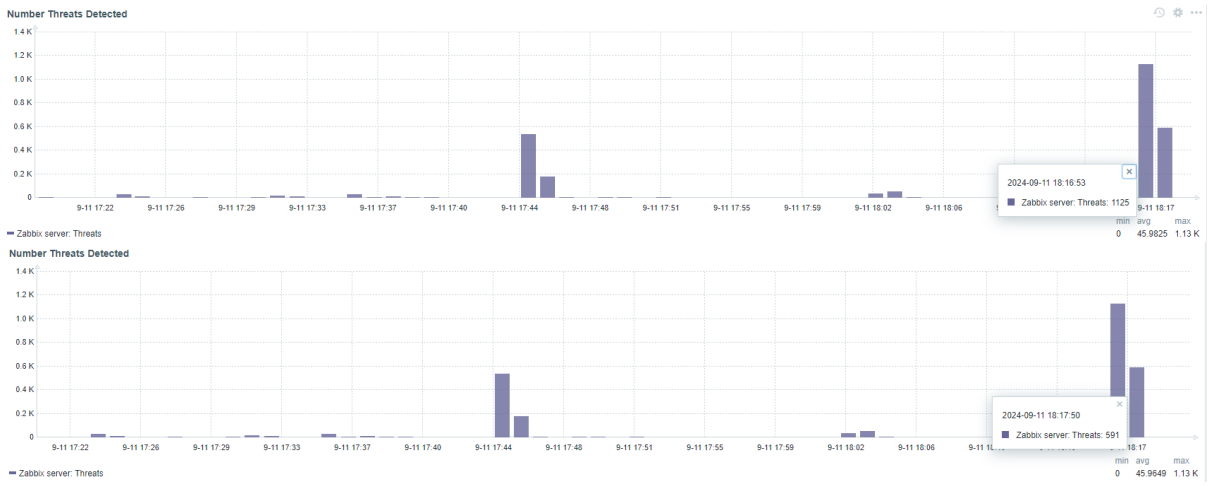


Figure 6.2.26 ICMP DDoS Number Threats Detected

From the Figure 6.2.26, the ICMP DDoS hits 1,716 total of attack in 5 minutes.

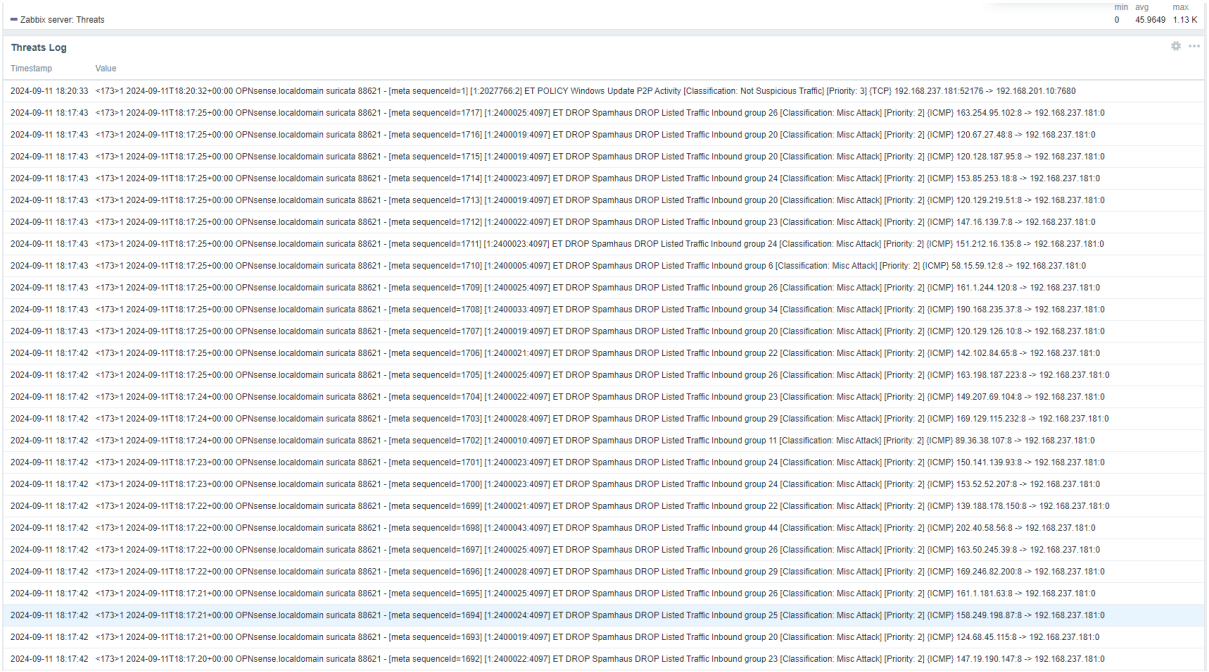


Figure 6.2.27 Information ICMP DDoS Threats

Figure 6.2.27 shows the host 192.168.237.181 is under type of ICMP ‘Misc Attack’ attack from different source IP address and the priority is 2.

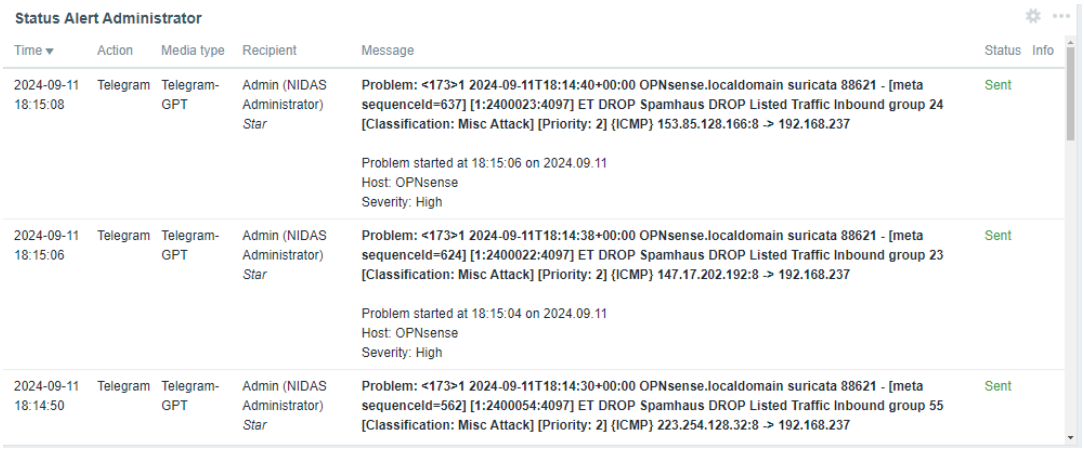


Figure 6.2.28 ICMP DDoS Alert

Figure 6.2.28 shows the status of alert message contain with recommendation has been success sent to the Network Administrator Star.

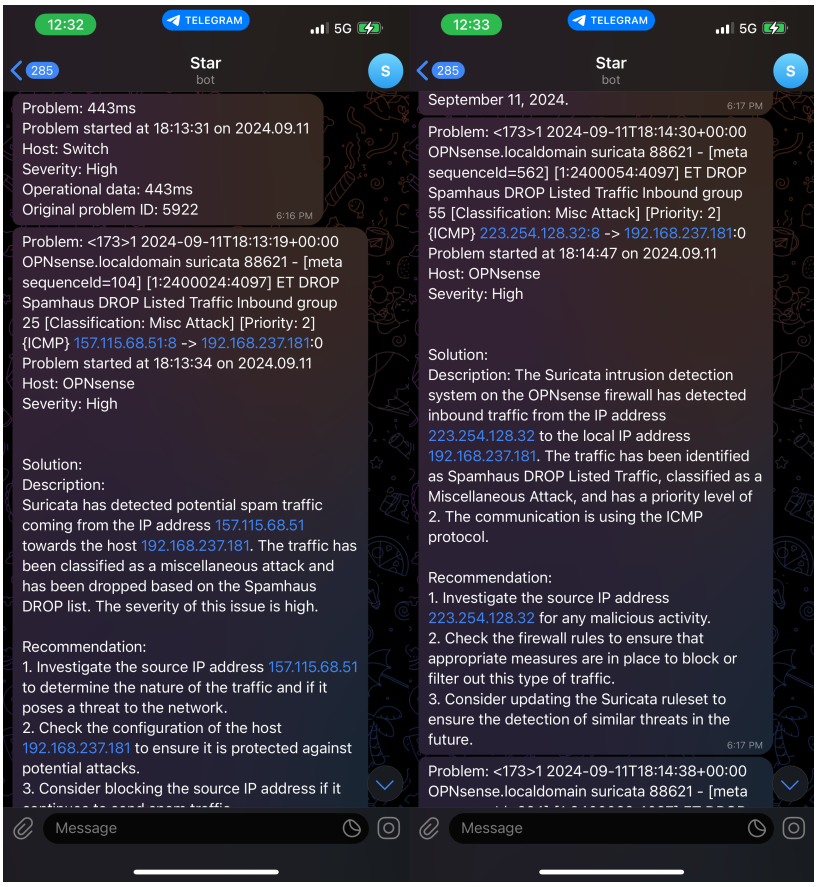


Figure 6.2.29 ICMP DDoS Alert-Telegram

In Figure 6.2.29 shows the alert messages that carries DDoS attack information along with recommended solutions. One of the other type shows abnormally frequent messages about switch ICMP response time status is in high 443ms. Both severity are marked as high.

TCP FIN/ACK Flood

An attack that floods the target server with fake FIN and ACK packets, consuming resources and causing service disruptions. By processing these packets, the server will in order to exhaust its resources, rendering it unable to process legitimate requests and becomes overwhelmed becomes overwhelmed, potentially leading to a denial of service.

finack.py

finack.py ⟩ No Selection

```
1  import os
2
3  target_ip = "192.168.237.177"
4  os.system("hping3 -c 10000 -d 120 -F -A --flood --rand-source " + target_ip)
5
```

Figure 6.2.30 TCP FIN/ACK Flood Script

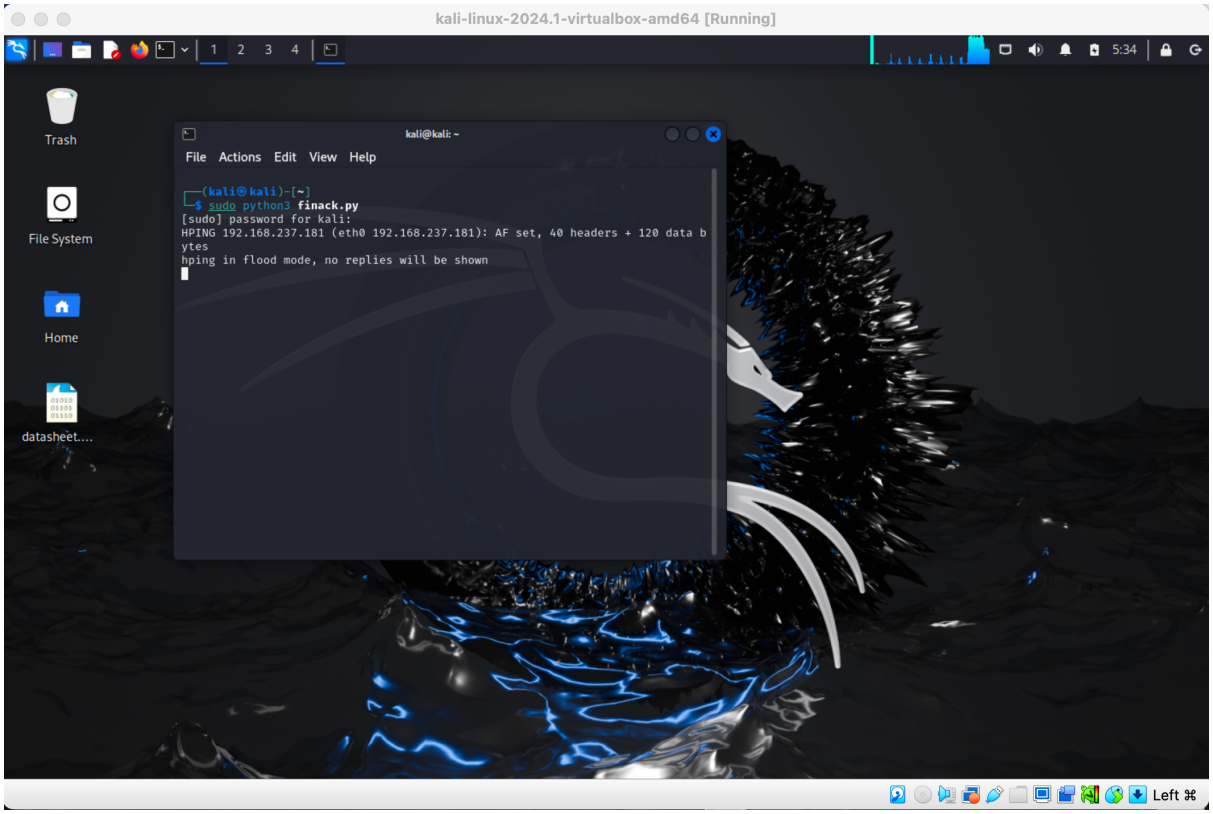


Figure 6.2.31 TCP FIN/ACK DDoS

From the Figure 6.2.30, represent the attack will use of flood mode sending 10,000 TCP packets carried with FIN flag or ACK flag forcing the target server to handle what appears to be many connection termination requests and acknowledgments from random source IP. Execute the script in kali linux.

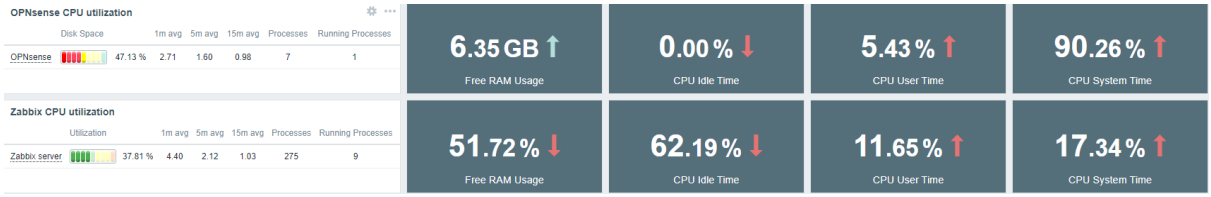


Figure 6.2.32 TCP FIN/ACK DDoS CPU Performance

From Figure 6.2.32, it can be seen that due to the IDS detecting a large number of TCP FIN/ACK DDoS attacks, the CPU idle time of the OPNsense server dropped to zero, with most of the CPU resources being consumed to handle the attack detection. Since the detected attack logs are continuously sent to Zabbix, the CPU usage on Zabbix also shows a significant increase as it processes the attack logs and triggers alert messages.

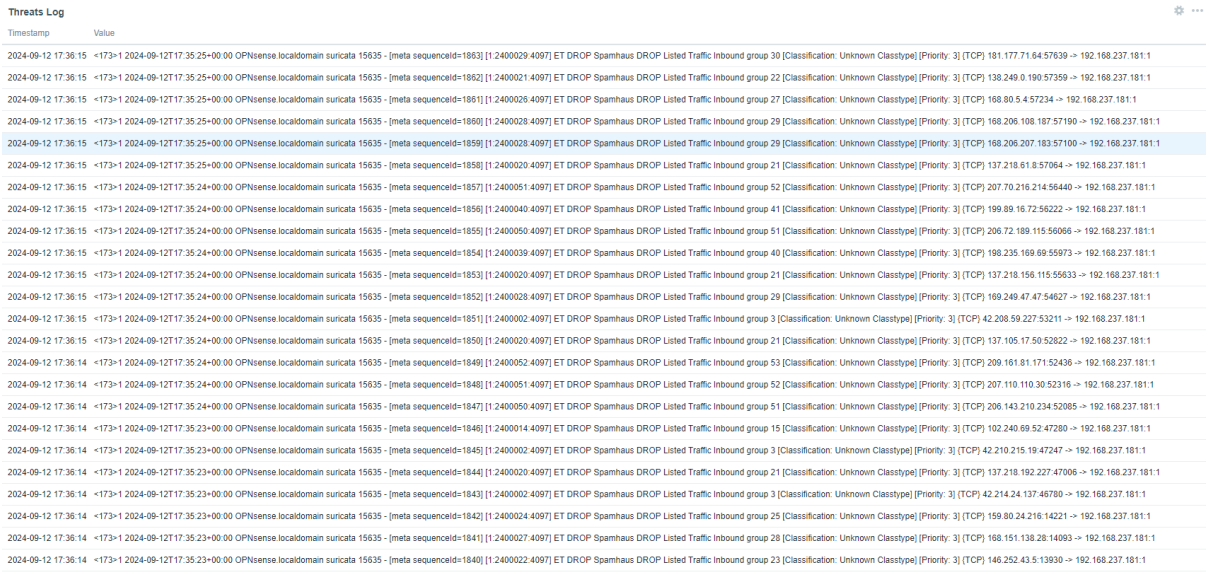
Figure 6.2.33 TCP FIN/ACK DDoS Network Performance

In Figure 6.2.33, when under a TCP FIN/ACK attack, the Interface Fa0/1 bits received, Interface pflog0 outgoing network traffic, and Interface [em2] bandwidth usage are roughly the same as during an ICMP attack, ranging generally between 4.5 Mbps and 8 Mbps. However, the ICMP response time spikes directly to 370.7 ms, and the graph shows that as

each network flow reaches its peak value, it then sharply drops to zero, indicating that data transmission connections have been cut off. Since the Zabbix server is installed on a virtual machine at the endpoint 192.168.237.181, this endpoint entered a crashed state due to the overwhelming number of TCP FIN/ACK attacks, rendering it unable to respond to legitimate network requests in a timely manner, ultimately leading to service interruption.

Figure 6.2.34 TCP FIN/ACK DDoS Number Threats Detected

In Figure 6.2.34, 1,427 TCP attacks were detected by IDS within 3 minutes, this attack causing the endpoint to become overloaded and unable to respond to legitimate network requests in a timely manner.



Threats Log

Timestamp	Value
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:25+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1863] [1:2400029:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 30 [Classification: Unknown Classtype] [Priority: 3] {TCP} 181.177.71.64:57639 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:25+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1862] [1:2400021:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 22 [Classification: Unknown Classtype] [Priority: 3] {TCP} 138.249.0.190:57359 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:25+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1861] [1:2400026:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 27 [Classification: Unknown Classtype] [Priority: 3] {TCP} 168.80.5.4:57234 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:25+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1860] [1:2400028:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 29 [Classification: Unknown Classtype] [Priority: 3] {TCP} 168.206.108.187:57190 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:25+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1859] [1:2400028:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 29 [Classification: Unknown Classtype] [Priority: 3] {TCP} 168.206.207.183:57100 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:25+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1858] [1:2400020:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 21 [Classification: Unknown Classtype] [Priority: 3] {TCP} 137.218.61.8:57064 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1857] [1:2400051:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 52 [Classification: Unknown Classtype] [Priority: 3] {TCP} 207.70.216.214:56440 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1856] [1:2400040:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 41 [Classification: Unknown Classtype] [Priority: 3] {TCP} 199.89.16.72:56222 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1855] [1:2400050:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 51 [Classification: Unknown Classtype] [Priority: 3] {TCP} 206.72.189.115:56066 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1854] [1:2400039:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 40 [Classification: Unknown Classtype] [Priority: 3] {TCP} 198.235.169.69:55973 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1853] [1:2400020:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 21 [Classification: Unknown Classtype] [Priority: 3] {TCP} 137.218.156.115:55633 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1852] [1:2400028:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 29 [Classification: Unknown Classtype] [Priority: 3] {TCP} 169.249.47.47:54627 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1851] [1:2400002:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 3 [Classification: Unknown Classtype] [Priority: 3] {TCP} 42.208.59.227:53211 -> 192.168.237.181:1
2024-09-12 17:36:15	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1850] [1:2400020:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 21 [Classification: Unknown Classtype] [Priority: 3] {TCP} 137.105.17.50:52822 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1849] [1:2400052:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 53 [Classification: Unknown Classtype] [Priority: 3] {TCP} 209.161.81.171:52436 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1848] [1:2400051:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 52 [Classification: Unknown Classtype] [Priority: 3] {TCP} 207.110.110.30:52316 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:24+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1847] [1:2400050:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 51 [Classification: Unknown Classtype] [Priority: 3] {TCP} 206.143.210.234:52085 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:23+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1846] [1:2400014:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 15 [Classification: Unknown Classtype] [Priority: 3] {TCP} 102.240.69.52:47280 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:23+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1845] [1:2400002:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 3 [Classification: Unknown Classtype] [Priority: 3] {TCP} 42.210.215.19:47247 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:23+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1844] [1:2400020:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 21 [Classification: Unknown Classtype] [Priority: 3] {TCP} 137.218.192.227:47006 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:23+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1843] [1:2400002:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 3 [Classification: Unknown Classtype] [Priority: 3] {TCP} 42.214.24.137:46780 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:23+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1842] [1:2400024:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 25 [Classification: Unknown Classtype] [Priority: 3] {TCP} 159.80.24.216:14221 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:23+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1841] [1:2400027:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 28 [Classification: Unknown Classtype] [Priority: 3] {TCP} 168.151.138.28:14093 -> 192.168.237.181:1
2024-09-12 17:36:14	<173>1 2024-09-12T17:35:23+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=1840] [1:2400022:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 23 [Classification: Unknown Classtype] [Priority: 3] {TCP} 146.252.43.5:13930 -> 192.168.237.181:1

Figure 6.2.35 Information TCP FIN/ACK DDoS Threats

Figure 6.2.35 shows the host 192.168.237.181 is under type of TCP 'Unknown Classtype' attack from random source IP address and the priority is 3.

Status Alert Administrator

Time ▾	Action	Media type	Recipient	Message	Status	Info
2024-09-12 17:32:40	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Problem: <173>1 2024-09-12T17:32:09+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=139] [1:2400021:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 22 [Classification: Unknown Classtype] [Priority: 3] {TCP} 138.249.175.102:38495 -> 1** Problem started at 17:32:38 on 2024.09.12 Host: OPNsense Severity: High	Sent	
2024-09-12 17:32:34	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Problem: <173>1 2024-09-12T17:32:07+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=120] [1:2400024:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 25 [Classification: Unknown Classtype] [Priority: 3] {TCP} 158.249.244.42:26510 -> 19** Problem started at 17:32:31 on 2024.09.12 Host: OPNsense Severity: High	Sent	
2024-09-12 17:32:31	Telegram	Telegram-GPT	Admin (NIDAS Administrator) *Star*	**Problem: <173>1 2024-09-12T17:32:06+00:00 OPNsense.localdomain suricata 15635 - [meta sequenceId=105] [1:2400020:4097] ET DROP Spamhaus DROP Listed Traffic Inbound group 21 [Classification: Unknown Classtype] [Priority: 3] {TCP} 137.72.129.78:21121 -> 192**	Sent	

Figure 6.2.36 TCP FIN/ACK DDoS Alert

Figure 6.2.36 shows the status of alert message contain with recommendation has been success sent to the Network Administrator Star.

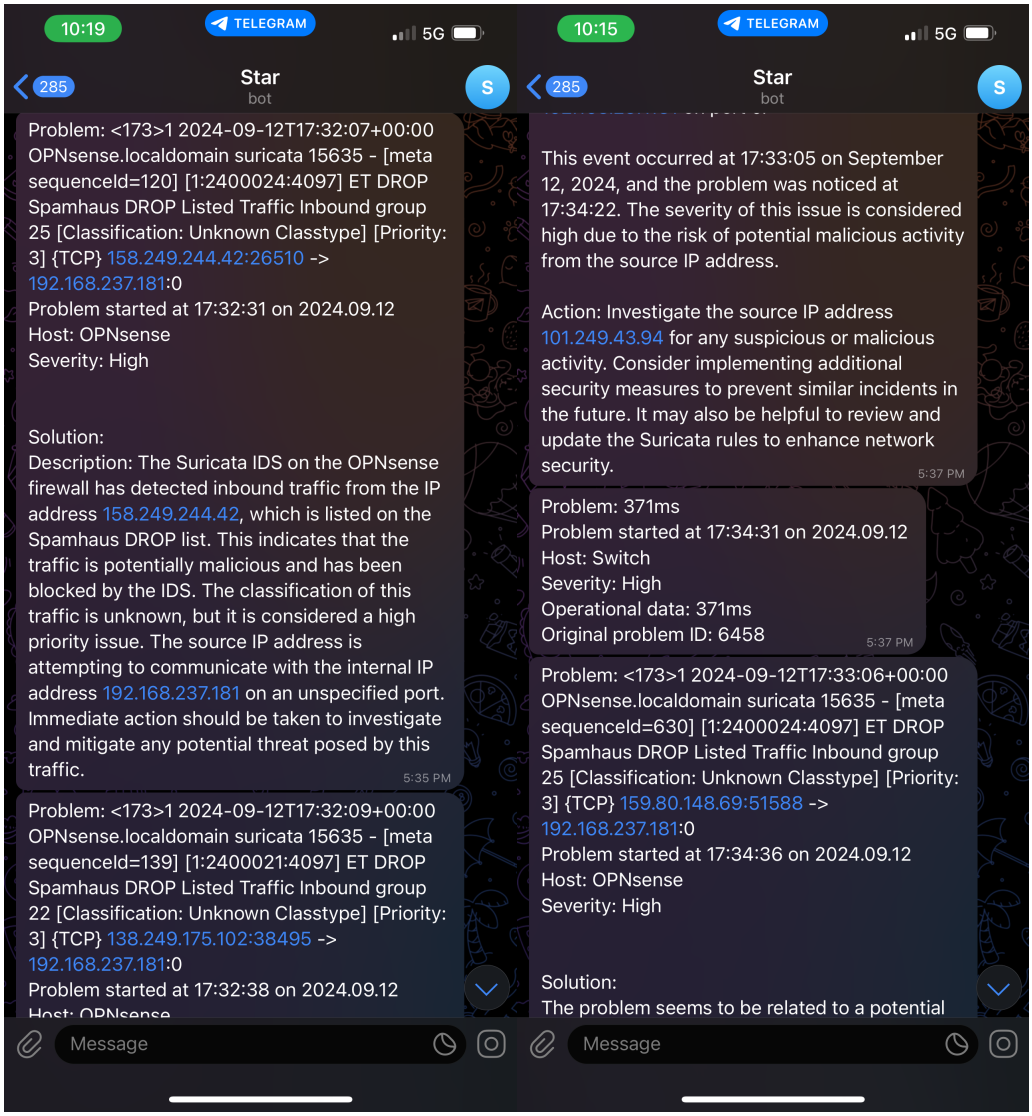


Figure 6.2.37 TCP FIN/ACK DDoS Alert-Telegram

In Figure 6.2.37 represent the alert messages that carries DDoS attack information along with recommended solutions, severity is marked as high. The other type shows abnormally frequent messages about switch ICMP response time status is in high 371ms and also a high severity level.

6.2.4 Brute Force Attack

Hydra

A powerful and widely used brute force password-cracking tool designed to test the security of login credentials on various network protocols and services. It supports multiple protocols and authentication methods, making it highly versatile on penetration testing.

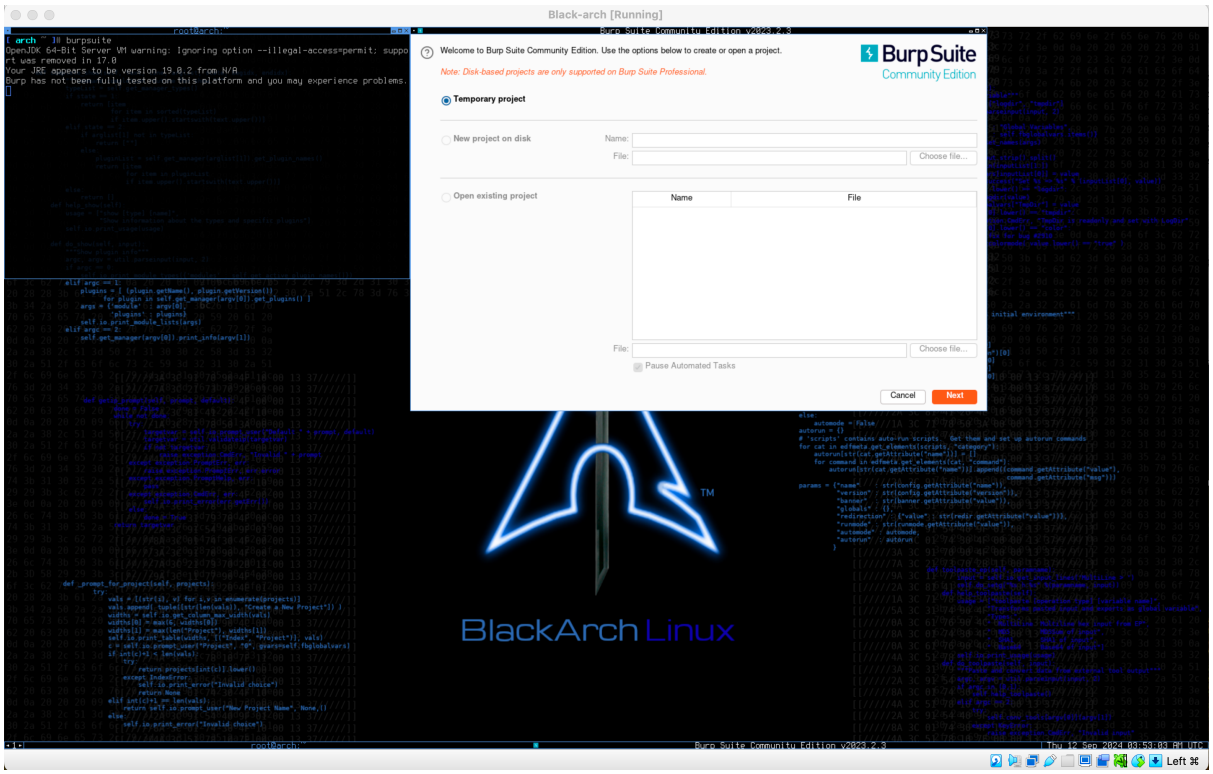
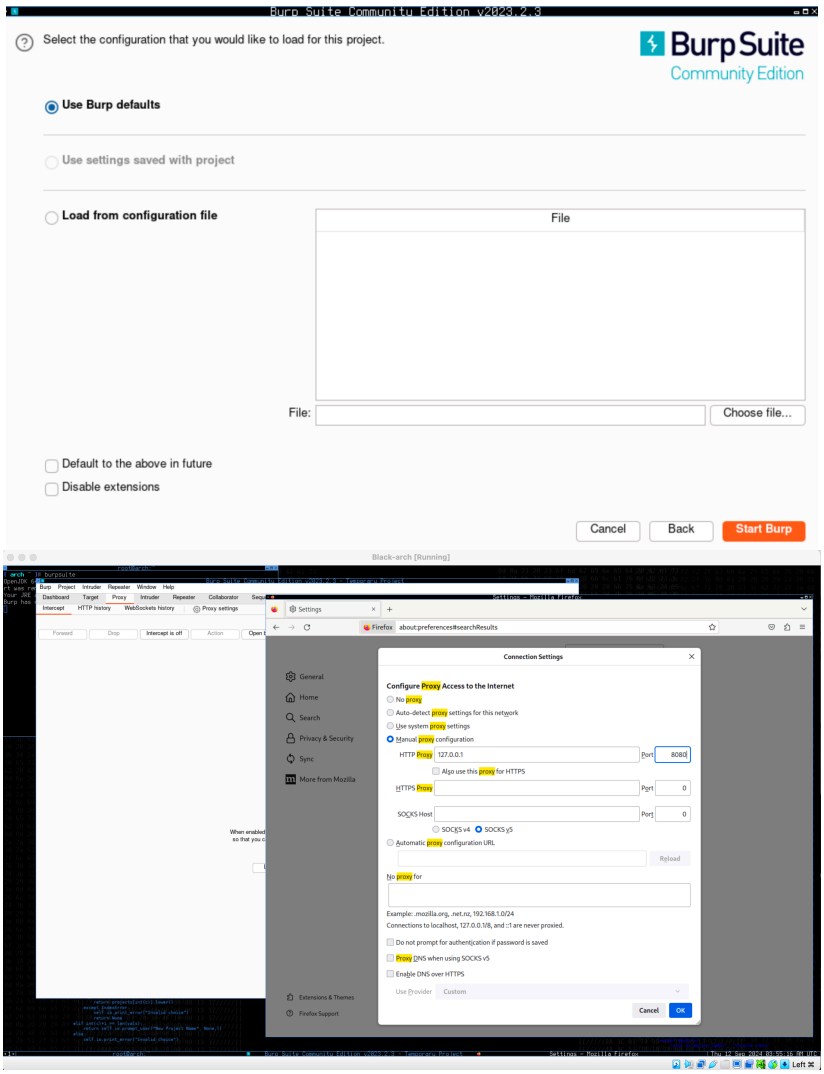
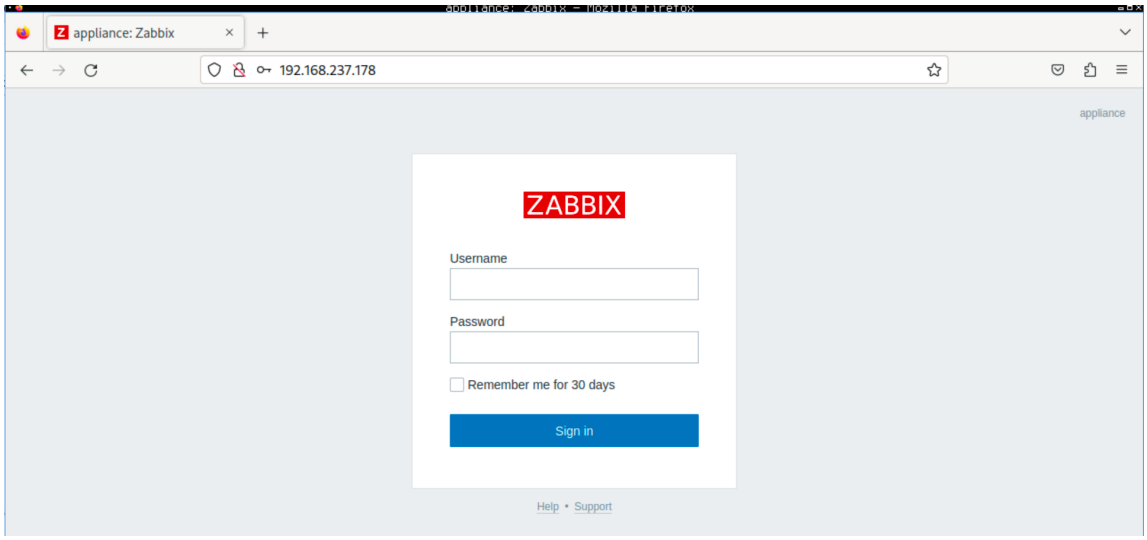


Figure 6.2.38 Burp Suite

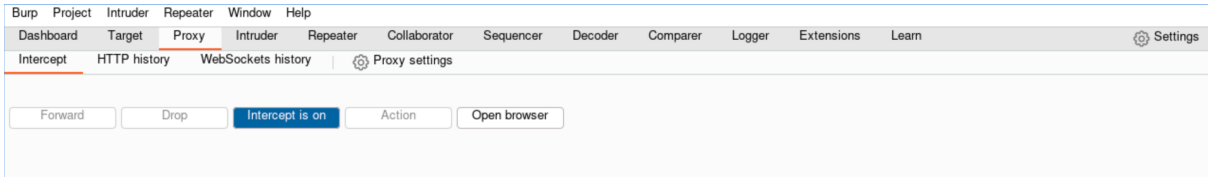
Using Hydra to brute force attack zabbix server password, before that start up Burp Suite in the Arch Linux using proxy to capture the web information. Using ‘temporary project’ to create the task.



Start Burp the Burp Suite, open the firefox browser in arch linux and set the http proxy in local IP address and port 8080 so that the Burp Suite proxy can capture the information.



Before start the proxy, go to the web page of zabbix server and stay at the login page.



Go to proxy, turn on the Intercept and back to zabbix login page. Input any username and password to sign in.

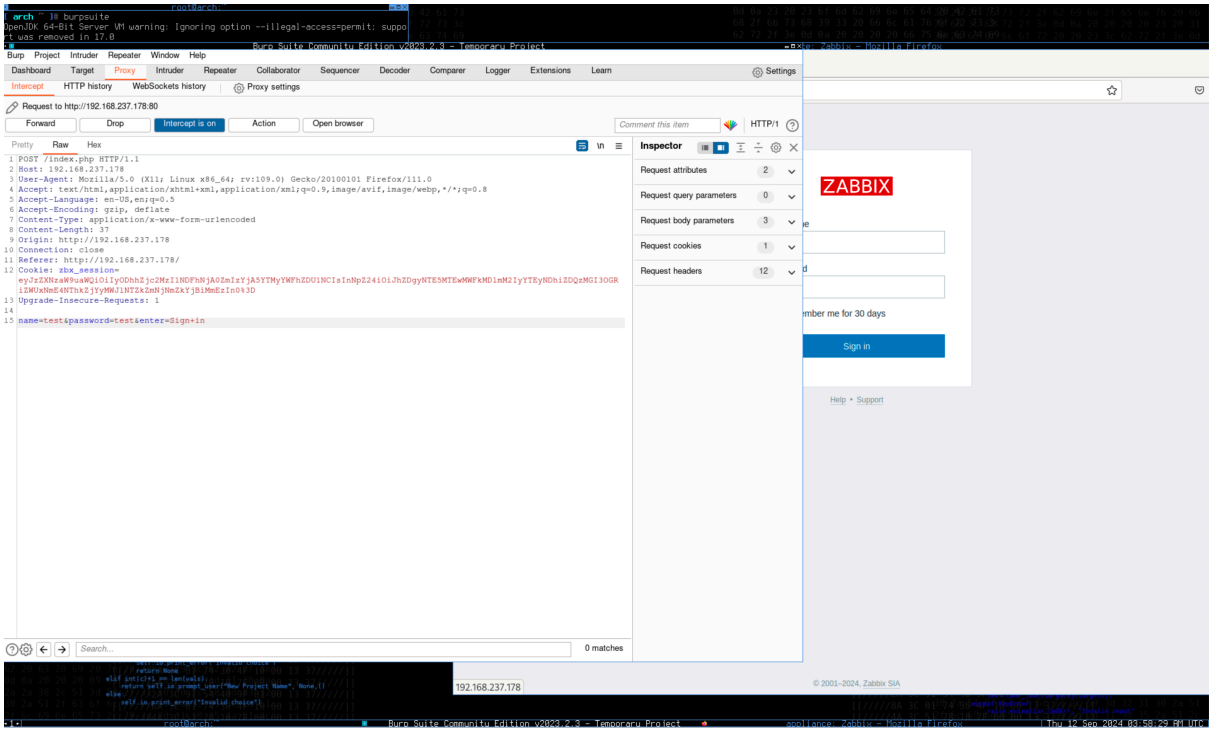


Figure 6.2.39 Burp Suite Data Capture

From the Figure 6.2.39, the key parameter of username and password is ‘name’ and ‘password’. The cookie session is with the long key.

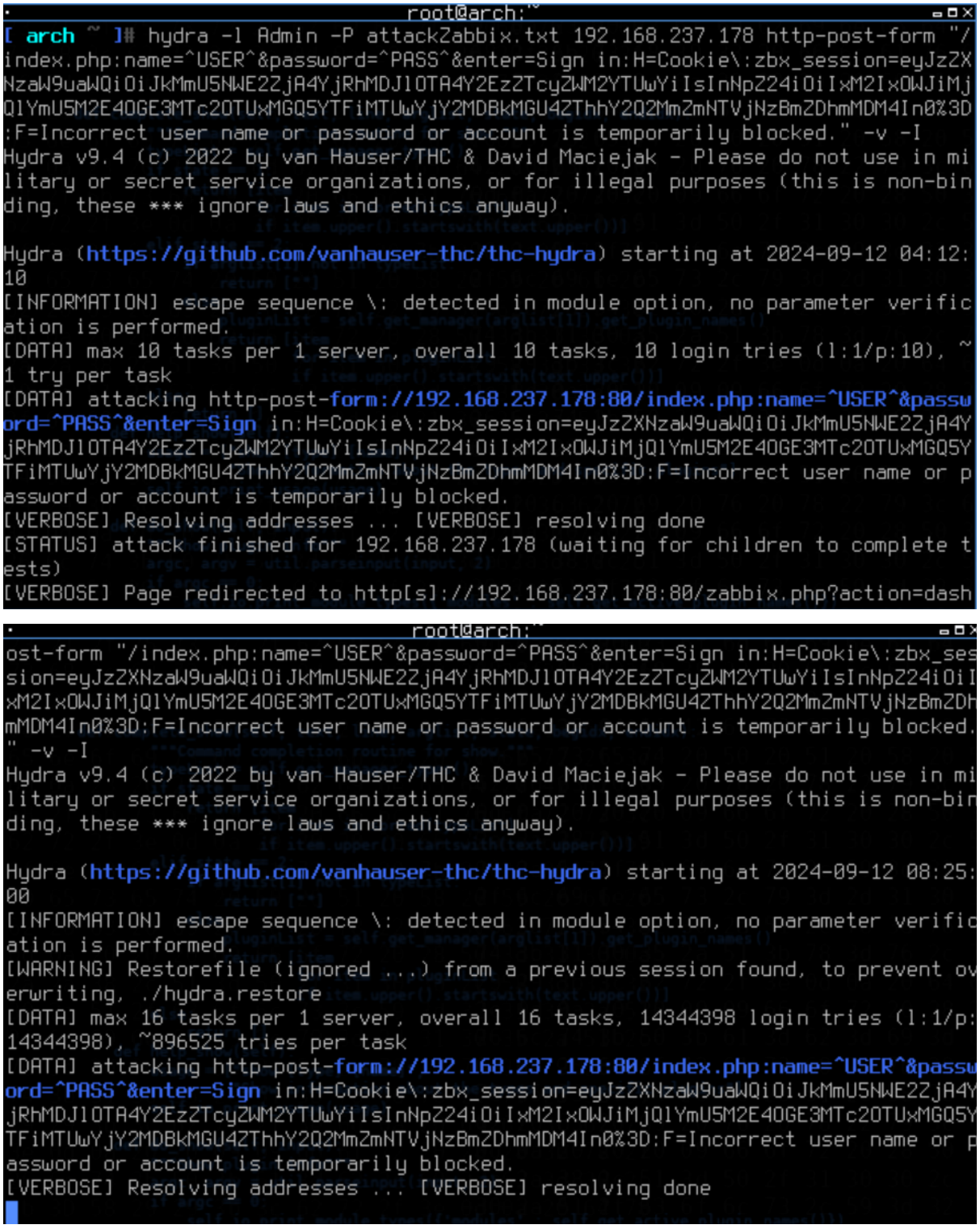


Figure 6.2.40 Hydra

Open the terminal in arch linux, brute force attack the zabbix server with the information capture by Burp Suite and doing 2 version of the hydra brute force attack. One is using 10 words wordlist and second is using rockyou.txt wordlist to do the brute-force on zabbix server.

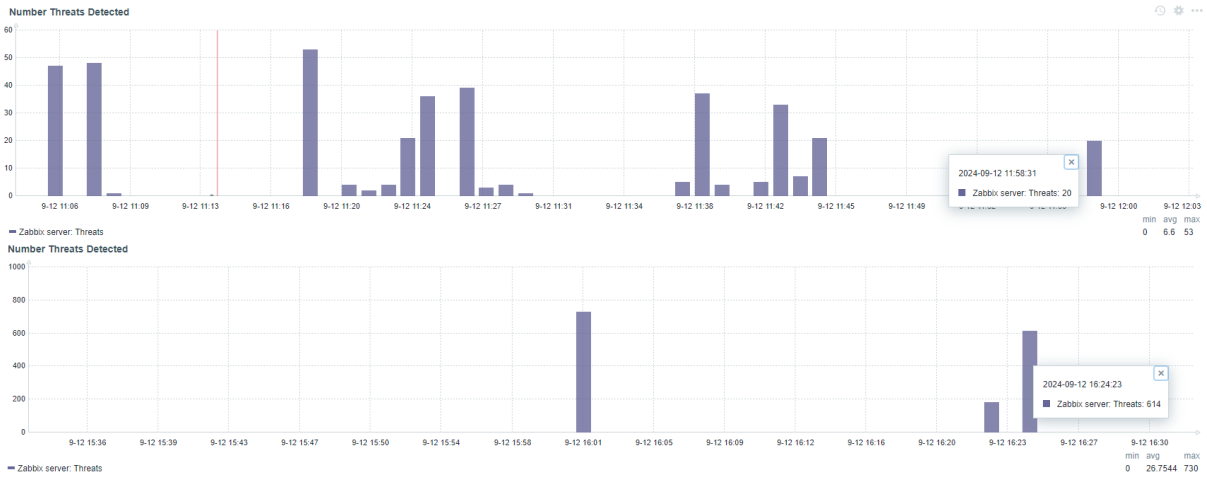


Figure 6.2.41 Hydra Number Threats Detected

From figure 6.2.41, it can be observed that although Hydra only performed 10 brute-force attempts, 20 attacks were detected. The number of detected attacks is twice the actual number of attempts because the IDS not only detects authentication requests but also the process of connection establishment and closure, thereby increasing the number of recorded events. In heavy brute force attack, 614 attacks were detected in a minute.

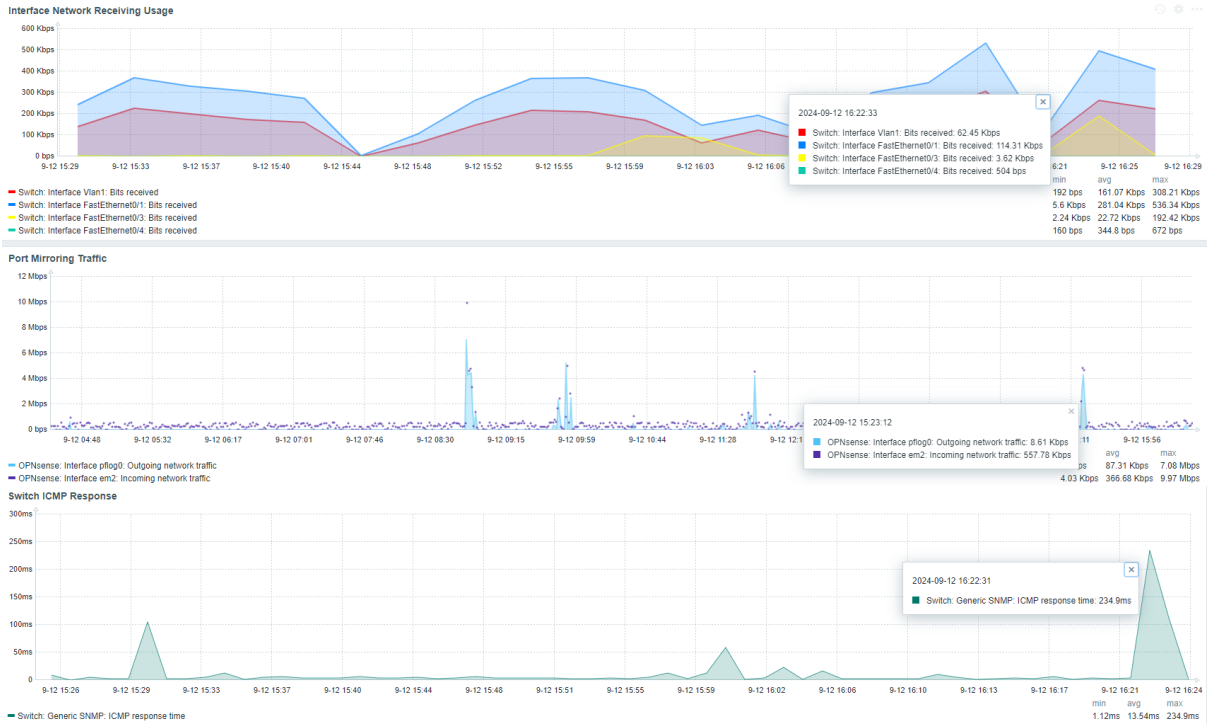


Figure 6.2.42 Hydra Brute Force Network Performance

According to figure 6.2.42, while Hydra conducts a large number of brute-force attacks, the network traffic is not affected in anyway. However, although Hydra does not directly generate ICMP packets, due to the overload caused by a large number of brute-force requests,

it may increase ICMP traffic, leading to high latency or packet loss. As a result, when Zabbix is monitoring ICMP, it will show longer response times.

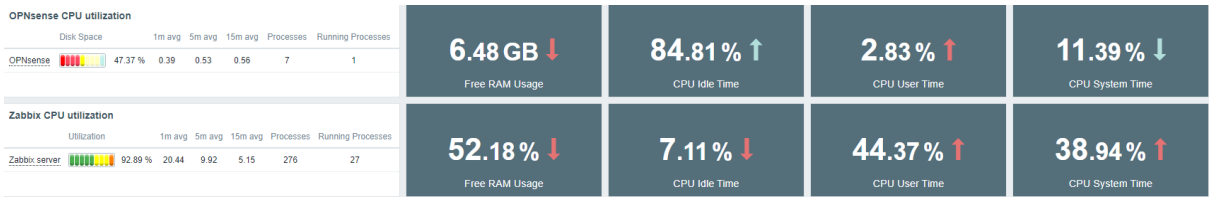


Figure 6.2.43 Hydra Brute Force System Performance

In figure 6.2.43, due to the large number of brute-force attacks by Hydra, the Zabbix server's CPU performance is at its limit, because the CPU is being heavily utilized to process and respond to Hydra's login attempts.

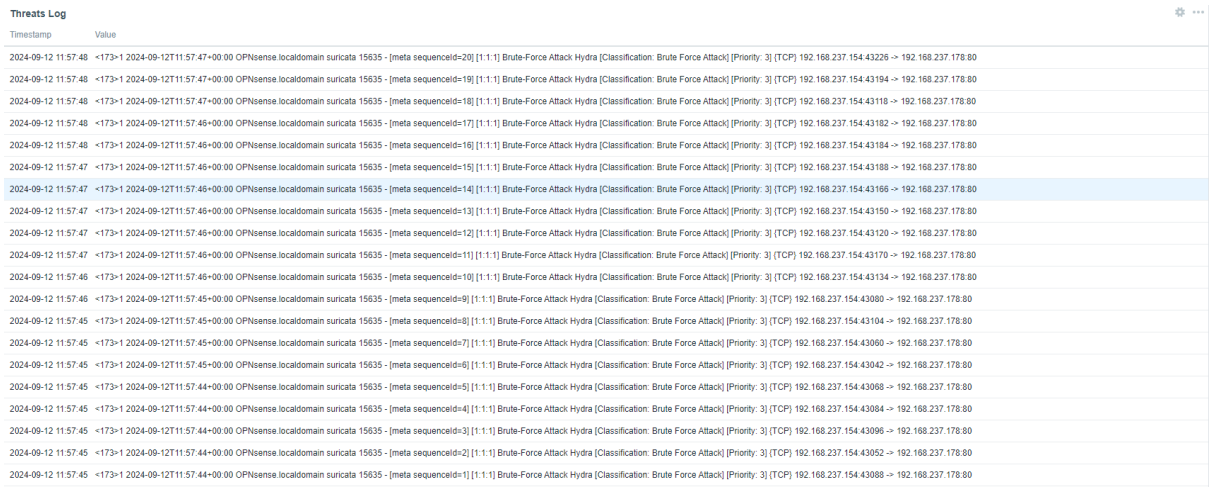


Figure 6.2.44 Information Hydra Brute Force Threats

Figure 6.2.44 shows the host 192.168.237.178 is under type of TCP ‘Brute Force Attack’ attack from source 192.168.237.154 and the priority is 3.

Figure 6.2.45 Hydra Brute Force Alert

Figure 6.2.45 shows the status of alert message contain with recommendation has been success sent to the Network Administrator Star.

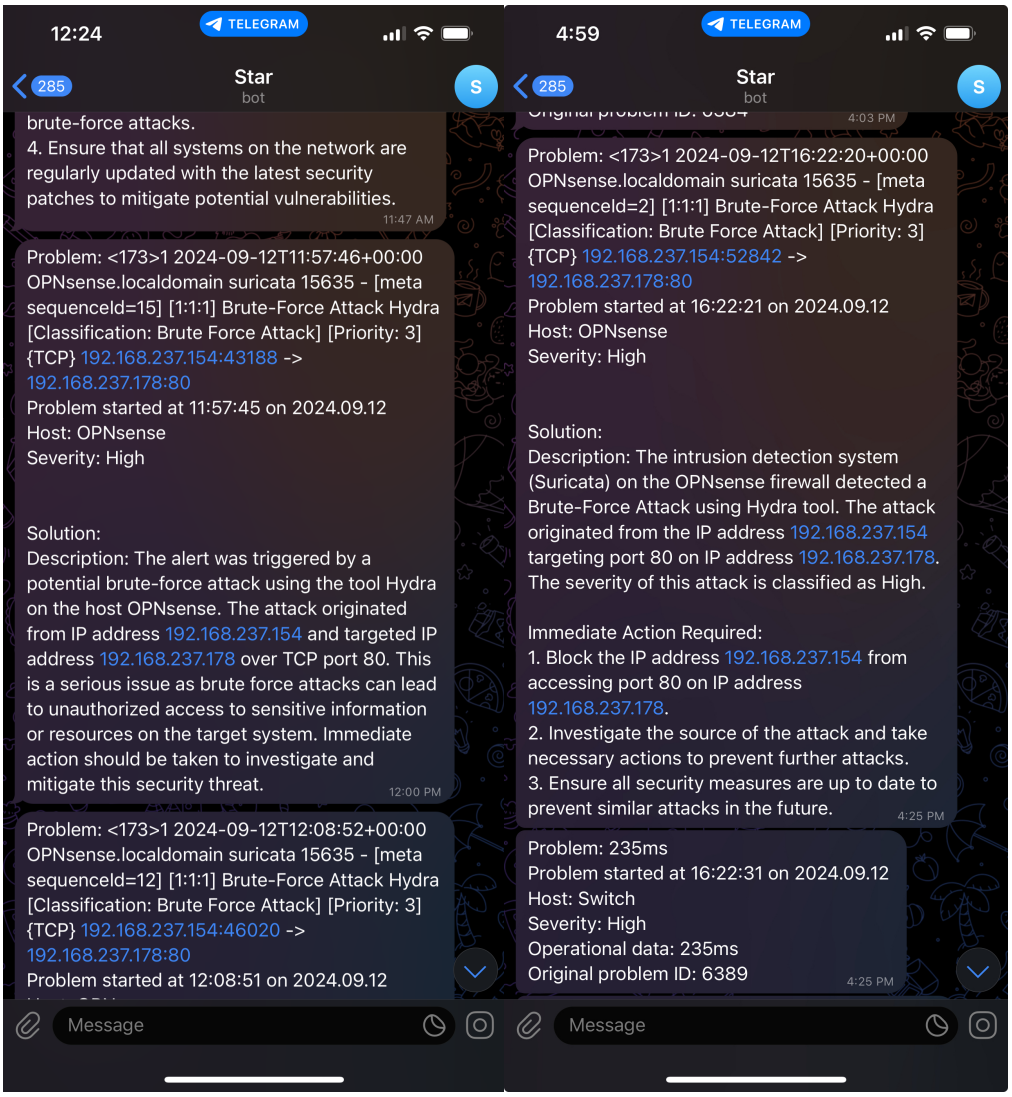


Figure 6.2.46 Hydra Brute Force Alert-Telegram

In Figure 6.2.46 shows the brute force attack alert message that carries with recommended solutions, and its severity is marked as high. The other type shows abnormally frequent messages about switch ICMP response time status is in high 235ms and also a high severity level.

6.3 Project Challenges

From the rusult of this NIDAS project has powerful detection capabilities and advantages, as Zabbix itself can integrate with many external plugins. This brings many security tools integration possibilities to this research to achieve high-security network protection. Although this research can provide many integration functions, it also sets a high threshold for device requirements. The device requirements will vary depending on the integration and deployment

with Zabbix; in other words, the device determines the upper limit of integration and deployment.There is also a hard requirement, since those are open-source tools, rich experience and certain technical skills in IT are necessary. A significant concern is the security of the monitoring system itself. In terms of deployment, because the IDS detection has pre set up 2 rules for Zabbix establish outgoing traffic to the paths 8.8.8.8 and 149.154.167.220, if an attack comes from the DNS server, there is a potential serious risk of information monitored by Zabbix being leaked. The use of public DNS servers (8.8.8.8) and specific IP addresses (149.154.167.220) in the IDS setup could potentially expose the system to attacks originating from compromised DNS servers. This configuration could lead to a serious vulnerability where the very information Zabbix is monitoring could be leaked, compromising the entire security infrastructure. Therefore, figure a good deployment plan is also that needs to be considered on this project so that to achieve high level security protection. This highlights the importance of carefully considering not just the capabilities of such a system, but also its own security measures and potential vulnerabilities in its deployment.

The project's flexibility in integration opens up numerous possibilities for enhancing network security. However, this flexibility comes with increased complexity in terms of hardware requirements and technical expertise needed for implementation. The scalability of the system is also directly tied to the capabilities of the hardware it's deployed on, which means that the potential for integration and deployment is limited by the devices used. While the NIDAS project offers powerful tools for network security, it also requires careful implementation and ongoing management large environment to ensure that it doesn't introduce new vulnerabilities while attempting to protect against existing ones.

6.4 Objective Evaluation

Based on the results, using Zabbix for network monitoring has proven quite successful in environments facing web-application attacks, distributed denial of service (DDoS), port scanning, and brute force attacks. Zabbix itself provides four integration methods, and this project utilized SNMP service and Zabbix agent to achieve effective monitoring of networks and servers. During DDoS attacks, Zabbix's monitoring of the switch's ICMP response time through SNMP successfully captured high ICMP latency issues, promptly alerting network security advisors. Additionally, the Zabbix agent monitoring the server's CPU usage indirectly reflected OPNsense's intensive work in intrusion prevention.

The integration with OPNsense successfully monitored and visualized intrusion detection data, presenting threats and their quantities in a clear, real-time manner, enhancing Zabbix's overall network monitoring capabilities. While Zabbix's network monitoring couldn't directly capture brute force attacks, the integration with IDS allowed for real-time acquisition of intrusion detection information through logs sent to Zabbix, which then informed network administrator. Finally, the optimized alert messages carrying recommended solutions and specific network attack information enabled network administrator to respond efficiently to attacks, significantly reducing the impact of network attacks.

This comprehensive approach demonstrates the power of integrating various security tools and monitoring systems. By combining real-time network monitoring, intrusion detection, and intelligent alerting systems, the project has created a robust security infrastructure capable of detecting, analyzing, and responding to a wide range of network threats quickly and effectively.

6.5 Concluding Remark

The NIDAS project has demonstrated the powerful potential of integrating Zabbix with various security tools to provide a comprehensive and adaptable network security solution. The combination of real-time monitoring, intrusion detection, and intelligent alerting systems significantly enhances the capability to detect, analyze, and respond to various network threats such as DDoS, brute-force attacks, and port scanning. However, while this project offers a high level of security and flexibility, it also introduces complexity in terms of device requirements, technical expertise, and the need for robust deployment plans to avoid creating new vulnerabilities, particularly regarding the security of the monitoring system itself. Overall, the project's success lies in its ability to provide high-level security protections, though it requires careful planning and ongoing management to maximize its potential and maintain network integrity.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

In today's interconnected world, ensuring the security and reliability of computer networks is critical for various industries. This project demonstrates the substantial benefits of integrating Zabbix's monitoring capabilities with OPNsense's intrusion detection system to create a comprehensive and robust Network Intrusion Detection and Alerting System (NIDAS). By addressing the limitations of traditional IDS, such as excessive false positives, lack of contextual insights, and dependence on predefined rules, this integrated solution provides more accurate and efficient network security monitoring.

To overcome the limitations of traditional IDS, which often suffer from excessive false alarms, a lack of contextual information, and a heavy reliance on predefined rules and signatures. The combined capabilities of Zabbix and OPNsense enable deeper visibility into network traffic and system performance, significantly enhancing threat detection. Zabbix's data collection and performance analysis, combined with OPNsense’s traffic inspection based on customizable rules, deliver a more holistic view of potential security incidents. This correlation of data across both systems allows network administrators to respond more effectively by identifying the root causes of issues and prioritizing their mitigation efforts. Zabbix serves as the backbone of this system, providing robust capabilities for collecting and analyzing network metrics, performance data, and system logs. This forms a solid foundation for monitoring the overall health and performance of the network infrastructure. Meanwhile, OPNsense inspects network traffic and detects potential intrusions based on customizable rules and signatures. A key strength of the NIDAS project is its ability to prioritize alerts and offer mitigation recommendations, allowing security teams to focus on the most critical threats. By considering risk levels and leveraging combined network monitoring and intrusion detection data, this system provides an optimized approach to incident response. Moreover, the contextual intelligence it offers helps allocate resources more efficiently, ensuring that high-risk threats are addressed promptly. Mitigation recommendations guide teams in taking appropriate actions to contain and resolve incidents, minimizing damage and ensuring business continuity, thereby reducing the impact of potential harm to a minimum. Although the integration of Zabbix and OPNsense does not incorporate advanced machine learning techniques, it still offers substantial improvements over traditional IDS methods,

especially through the use of Deep Packet Inspection (DPI). Overall, this project successfully demonstrates the potential of integrating network monitoring and intrusion detection tools to strengthen an organization's network security posture. By addressing the limitations of traditional intrusion detection methods, providing intelligent alert prioritization, and offering actionable mitigation strategies, the proposed solution enhances intrusion detection accuracy, reduces false positives, and improves resource allocation for managing high-risk threats. These advancements make NIDAS a powerful tool for protecting network infrastructures in an increasingly interconnected world.

7.2 Recommendation

In today's complex network environment, ensuring system security and efficient operation is crucial. First and foremost, improving deployment security is a pressing matter. Given the potential security risks associated with using public DNS servers (such as 8.8.8.8) and specific IP addresses, it is recommended to use private DNS servers or more secure alternatives to minimize the risk of data exposure. Simultaneously, deploying additional security measures like firewalls or traffic filtering for outgoing connections can further protect against potential attacks. Secondly, planning for scalability to meet future needs is essential. Careful planning of hardware and network resources, along with considering investments in more powerful hardware, can fully support the integration potential of Zabbix and its plugins without overwhelming system performance. This ensures that the system can scale smoothly as demands grow. Thirdly, continuous exploration of automation and optimization avenues is vital. By automating routine tasks and alerts, system performance can be further enhanced. This may include automated load balancing, adaptive threshold setting for alerts, and more sophisticated AI-driven insights through GPT integration. These measures not only improve efficiency but also reduce the human error. Lastly, regular audits and updates should not be overlooked. Implementing regular security audits and software updates ensures that the system remains secure and efficient. This is particularly important for open-source projects like Zabbix and OPNsense, where updates bring both new features and critical security patches. By implementing these strategies, organizations can significantly improve the security, performance, and reliability of their IT infrastructure. This not only addresses current challenges but also lays a solid foundation for future technological developments and business growth. Continuous focus on these aspects will help build a

system environment that is both secure and efficient, providing strong support for an organization's digital transformation.

REFERENCES

[1] J. P. Anderson, "Computer Security Threat Monitoring and Surveillance," James P. Anderson Co., Fort Washington, PA, USA, Tech. Rep., 1980.

[2] Mert, I. S., and T. Çavuşoğlu. "A Novel Approach for Detection of Fake News Based on Behavioral Modeling." Neural Networks, vol. 107, 2018, pp. 192-201. doi: 10.1016/j.neunet.2018.07.006.

[3] D. E. Denning, "An Intrusion-Detection Model," IEEE Transactions on Software Engineering, vol. SE-13, no. 2, pp. 222-232, Feb. 1987, doi: 10.1109/TSE.1987.232894.

[4] T. F. Lunt, "A Survey of Intrusion Detection Techniques," Computers & Security, vol. 12, no. 4, 1993, doi: 10.1016/0167-4048(93)90033-B.

[5] R. G. Bace, Intrusion Detection, Macmillan Technical Publishing, 2000, doi: 10.5555/518389.

[6] A. Halimaa A. and K. Sundarakantham, "Machine Learning Based Intrusion Detection System," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2019, pp. 916-920, doi: 10.1109/ICOEI.2019.8862784.

[7] N. Khamphakdee, N. Benjamas and S. Saiyod, "Improving Intrusion Detection System based on Snort rules for network probe attack detection," 2014 2nd International Conference on Information and Communication Technology (ICoICT), Bandung, Indonesia, 2014, pp. 69-74, doi: 10.1109/ICoICT.2014.6914042.

[8] S. Kumar and R. C. Joshi, "Design and implementation of IDS using Snort, Entropy and alert ranking system," 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies, Thuckalay, India, 2011, pp. 264-268, doi: 10.1109/ICSCCN.2011.6024556.

[9] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," in IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303-336, First Quarter 2014, doi: 10.1109/SURV.2013.052213.00046.

[10] S. Mukkamala, A. Sung, and B. Ribeiro, "Intrusion detection using neural networks and support vector machine," Proceedings of the 2002 IEEE International Joint Conference on Neural Networks, 2002, doi: 10.1109/IJCNN.2002.1007774.

[11] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine Learning and Deep Learning Methods for Cybersecurity," IEEE Access, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950

[12] Debar, H., Curry, D., & Feinstein, B. (2007). The intrusion detection message exchange format (IDMEF). RFC 4765, IETF.

[13] Milenkoski, Aleksandar & Vieira, Marco & Kounev, Samuel & Avritzer, Alberto & Payne, Bryan. (2015). Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices. ACM Computing Surveys. 48. 12:1-. 10.1145/2808691.

[14] Patel, Ahmed & Taghavi, Mona & Bakhtiyari, Kaveh & Jr, Joaquim. (2012). An Intrusion Detection And Prevention System In Cloud Computing: A Systematic Review. Journal of Network and Computer Applications. 36. 10.1016/j.jnca.2012.08.007.

[15] Kumar, Vinod. (2012). Signature Based Intrusion Detection System Using SNORT. International Journal of Computer Applications & Information Technology. 1. 7.

[16] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest trends," Computer Networks, vol. 51, no. 12, pp. 3448-3470, 2007, doi: 10.1016/j.comnet.2007.02.001.

[17] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," in IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153-1176, Secondquarter 2016, doi: 10.1109/COMST.2015.2494502.

[18] Navaz, A. S. Syed & Sangeetha, V. & Prabhadevi, C.. (2013). Entropy based Anomaly Detection System to Prevent DDoS Attacks in Cloud. International Journal of Computer Applications. 62. 42-47. 10.5120/10160-5084.

[19] Veeramreddy, Jyothsna & Prasad, V. & Prasad, Koneti. (2011). A Review of Anomaly based Intrusion Detection Systems. International Journal of Computer Applications. 28. 26-35. 10.5120/3399-4730.

[20] G. Martinez, A. P. Rios, and J. D. Martinez, "Zenarmor: Enhancing Network Security with Deep Packet Inspection and Application-Aware Firewalls," Journal of Network Security, vol. 15, no. 2, pp. 120-135, 2023. doi: 10.1109/JNS.2023.7891234.

[21] S. Jones and M. Peterson, "Comprehensive Comparison of pfSense and OPNsense in Modern Network Environments," International Journal of Network Security, vol. 17, no. 3, pp. 101-115, 2022. doi: 10.1016/j.ijns.2022.03.001.

[22] P. Smith, H. Chang, and A. Gupta, "Evaluating Network Traffic Monitoring Tools: A Case Study of Zabbix and Zenarmor," Proceedings of the IEEE Symposium on Network Monitoring and Analysis, vol. 22, no. 4, pp. 75-89, 2023. doi: 10.1109/NSMA.2023.1112548.

[23] A. Patel, K. Zhao, and M. Lee, "Real-Time Network Performance Monitoring Using Zabbix: A Detailed Review and Implementation Guide," Journal of IT Systems Monitoring, vol. 13, no. 1, pp. 45-57, 2023. doi: 10.1002/itsm.4567.

[24] R. Taylor and D. Wilson, "Intrusion Detection and Prevention: A Review of Zenarmor and its Integration with pfSense and OPNsense," Cybersecurity Journal, vol. 20, no. 3, pp. 201-214, 2022. doi: 10.1049/cyb.2022.0859.

[25] G. Luo, Z.Y. Chen and B. Omar Mohammed "A systematic literature review of intrusion detection systems in cloud-based IoT environment,", 2022, pp. 1-13, doi: 10.1002/cpe.6822

Appendix A

```
#!/usr/bin/env python3

import os
import time
from datetime import datetime

position_file = '/etc/last_read_position.txt'

current_date = datetime.now().strftime('%Y%m%d')
log_file = f'/var/log/suricata/suricata_{current_date}.log'

zabbix_server='192.168.237.178'
hostname='OPNsense'
key='ids.detect'

if os.path.exists(log_file):
    last_read_position = 0
    last_read_date = ""
    if os.path.exists(position_file):
        with open(position_file, 'r') as pos_file:
            position_data = pos_file.read().strip().split(',')
            if len(position_data) == 2 and position_data[0].isdigit():
                last_read_position = int(position_data[0])
                last_read_date = position_data[1]
    if last_read_date != current_date;
        last_read_position = 0
    with open(log_file, 'r') as file:
        file.seek(last_read_position)
        new_lines = file.readlines()
        if new_lines:
            for line in new_lines:
                print(line.strip())
                os.system(f"/usr/local/bin/zabbix_sender -z {zabbix_server} -s {hostname} -k
{key} -o \"{line.strip()}\"")

            with open(position_file, 'w') as pos_file:
                pos_file.write(f"{file.tell()},{current_date}")
        else:
            exit(0)
else:
    print("Log file does not exist")
```

Appendix B

```
#!/bin/sh
# Sleep for 30 seconds to ensure all services are up
sleep 30

PIDFILE="/tmp/ids.pid"

while true; do
    if[ -f $PIDFILE ] && kill -0 $(cat $PIDFILE) 2>/dev/null; then
        sleep 0.5
        continue
    fi

    /usr/local/bin/python3 /usr/local/bin/ids.py &
    echo $! > $PIDFILE

    sleep 0.5
done
```

Appendix C

```
import mysql.connector
import json

def load_last_processed_clock():
    try:
        with open('last_processed_clock.json', 'r') as file:
            data = json.load(file)
            return data.get('last_clock', 0)
    except FileNotFoundError:
        return 0

def save_last_processed_clock(last_clock):
    with open('last_processed_clock.json', 'w') as file:
        json.dump({'last_clock': last_clock}, file)

db_config = {
    'user': 'zabbix1',
    'password': 'starstar77',
    'host': '127.0.0.1',
    'database': 'zabbix'
}

conn = mysql.connector.connect(**db_config)
cursor = conn.cursor()

last_processed_clock = load_last_processed_clock()

query = "SELECT clock, value FROM history_log WHERE clock > %s"
cursor.execute(query, (last_processed_clock,))

new_records = cursor.fetchall()

new_value_count = len(new_records)

if new_records:
    last_processed_clock = new_records[-1][0]
    save_last_processed_clock(last_processed_clock)

print(f"'value' : {new_value_count}")

cursor.close()
conn.close()
```

Appendix D

```
var Telegram = {
    token: null,
    to: null,
    message: null,
    proxy: null,
    parse_mode: null,

    escapeMarkup: function (str, mode) {
        switch (mode) {
            case 'markdown':
                return str.replace(/([_*\[])/g, '\\$&');
            case 'markdownv2':
                return str.replace(/([_*\[\]()~>#+\-=|{}.!])/g, '\\$&');
            case 'html':
                return str.replace(/<(\s|[^a-z\/])/g, '&lt;$1');
            default:
                return str;
        }
    },

    callChatGPT: function (prompt) {
        var apiKey = '<API key>';
        var apiUrl = 'https://api.openai.com/v1/chat/completions';
        var data = {
            model: 'gpt-3.5-turbo',
            messages: [{"role": "user", "content": prompt}],
            max_tokens: 200
        };
        var request = new HttpRequest();

        request.addHeader('Content-Type: application/json');
        request.addHeader('Authorization: Bearer ' + apiKey);

        var response = request.post(apiUrl, JSON.stringify(data));
        try {
            response = JSON.parse(response);
            if (response.choices && response.choices.length > 0) {
                return response.choices[0].message.content.trim();
            } else {
                throw 'Invalid response from ChatGPT';
            }
        } catch (error) {
            throw 'ChatGPT API call failed: ' + error;
        }
    },

    sendMessage: function () {
```

```
        var params = {
            chat_id: Telegram.to,
            text: Telegram.message,
            disable_web_page_preview: true,
            disable_notification: false
        },
        data,
        response,
        request = new HttpRequest(),
        url = 'https://api.telegram.org/bot' + Telegram.token + '/sendMessage';

        if (Telegram.parse_mode !== null) {
            params['parse_mode'] = Telegram.parse_mode;
        }

        if (Telegram.proxy) {
            request.setProxy(Telegram.proxy);
        }

        request.addHeader('Content-Type: application/json');
        data = JSON.stringify(params);

        // Remove replace() function if you want to see the exposed token in the log file.
        Zabbix.log(4, '[Telegram Webhook] URL: ' + url.replace(Telegram.token,
'<TOKEN>'));
        Zabbix.log(4, '[Telegram Webhook] params: ' + data);
        response = request.post(url, data);
        Zabbix.log(4, '[Telegram Webhook] HTTP code: ' + request.getStatus());

        try {
            response = JSON.parse(response);
        }
        catch (error) {
            response = null;
        }

        if (request.getStatus() !== 200 || typeof response.ok !== 'boolean' || response.ok !== true)
{
            if (typeof response.description === 'string') {
                throw response.description;
            }
            else {
                throw 'Unknown error. Check debug log for more information.';
            }
        }
    }
};

try {
```

```
    var params = JSON.parse(value);

    if (typeof params.Token === 'undefined') {
        throw 'Incorrect value is given for parameter "Token": parameter is missing';
    }

    Telegram.token = params.Token;

    if (params.HTTPProxy) {
        Telegram.proxy = params.HTTPProxy;
    }

    params.ParseMode = params.ParseMode.toLowerCase();

    if (['markdown', 'html', 'markdownv2'].indexOf(params.ParseMode) !== -1) {
        Telegram.parse_mode = params.ParseMode;
    }

    Telegram.to = params.To;
    Telegram.message = params.Subject + '\n' + params.Message;

    if (['markdown', 'html', 'markdownv2'].indexOf(params.ParseMode) !== -1) {
        Telegram.message = Telegram.escapeMarkup(Telegram.message, params.ParseMode);
    }

    var chatGPTPrompt = 'Problem: ' + params.Subject + '\nDetails: ' + params.Message;
    var chatGPTSolution = Telegram.callChatGPT(chatGPTPrompt);

    Telegram.message += '\n\nSolution:\n' + chatGPTSolution;

    Telegram.sendMessage();

    return 'OK';
}
catch (error) {
    Zabbix.log(4, '[Telegram Webhook] notification failed: ' + error);
    throw 'Sending failed: ' + error + '.';
}
```

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y5	**Study week no.: 2**
Student Name & ID: To Jin Yi 1900176	
Supervisor: Dr. Gan Ming Lee	
Project Title: Network Intrusion Detection and Alert System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Finish study Zabbix documentation

2. WORK TO BE DONE

Integration Zabbix with IDS

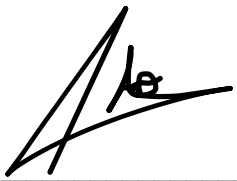
3. PROBLEMS ENCOUNTERED

Snort is not really suitable

4. SELF EVALUATION OF THE PROGRESS

The progress is working fine.

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y5	**Study week no.: 4**
Student Name & ID: To Jin Yi 1900176	
Supervisor: Dr. Gan Ming Lee	
Project Title: Network Intrusion Detection and Alert System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Define the suitable IDS tools for zabbix

2. WORK TO BE DONE

Integration with IDS tools

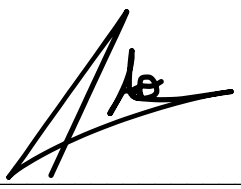
3. PROBLEMS ENCOUNTERED

Pfsense or OPNsense?

4. SELF EVALUATION OF THE PROGRESS

The progress is working fine.

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y5	**Study week no.: 5**
Student Name & ID: To Jin Yi 1900176	
Supervisor: Dr. Gan Ming Lee	
Project Title: Network Intrusion Detection and Alert System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Study documentation of Pfsense and OPNsense

2. WORK TO BE DONE

Integrating Zabbix with OPNsense

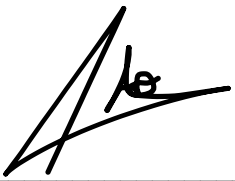
3. PROBLEMS ENCOUNTERED

Some configuration always get erase when I reboot OPNsense

4. SELF EVALUATION OF THE PROGRESS

The progress is working fine.

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y5	**Study week no.: 7**
Student Name & ID: To Jin Yi 1900176	
Supervisor: Dr. Gan Ming Lee	
Project Title: Network Intrusion Detection and Alert System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Success Integrate Zabbix and OPNsense

2. WORK TO BE DONE

Trigger alert and send alert message with recommendation

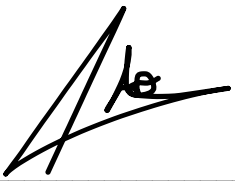
3. PROBLEMS ENCOUNTERED

How GPT can get achieve the problem event, where is the problem event storing

4. SELF EVALUATION OF THE PROGRESS

The progress is working fine.

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y5	**Study week no.: 9**
Student Name & ID: To Jin Yi 1900176	
Supervisor: Dr. Gan Ming Lee	
Project Title: Network Intrusion Detection and Alert System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Success setup the trigger alert and dashboard

2. WORK TO BE DONE

Achieve integrate API with GPT to get recommendation

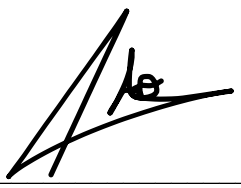
3. PROBLEMS ENCOUNTERED

To find the path of problem event, is in mysql database but need to figure out how to access into it

4. SELF EVALUATION OF THE PROGRESS

The progress is working fine.

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y5	**Study week no.: 10**
Student Name & ID: To Jin Yi 1900176	
Supervisor: Dr. Gan Ming Lee	
Project Title: Network Intrusion Detection and Alert System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Achieve integrate API with GPT to get recommendation

2. WORK TO BE DONE

Implementation and system testing finish

3. PROBLEMS ENCOUNTERED

Define and self evaluate rules

4. SELF EVALUATION OF THE PROGRESS

The progress is working fine.

GML

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y5	**Study week no.: 12**
Student Name & ID: To Jin Yi 1900176	
Supervisor: Dr. Gan Ming Lee	
Project Title: Network Intrusion Detection and Alert System	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Implementation and system testing finish

2. WORK TO BE DONE

Report need to finish

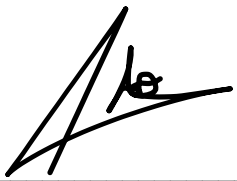
3. PROBLEMS ENCOUNTERED

Try to do the brute borce attack environment

4. SELF EVALUATION OF THE PROGRESS

Too much report, is unexpected

Supervisor's signature



Student's signature

Network Intrusion Detection & Alert System

AUTHORS
TO JIN YI
SUPERVISOR: Dr. GAN MING LEE

AFFILIATIONS
UNIVERSITY TUNKU ABDUL RAHMAN (UTAR)

1 INTRODUCTION

This thesis proposes an integrated network monitoring and intrusion detection system. It combines Zabbix and IDS techniques to address limitations of traditional NIDS. It provides automated mitigation recommendations and a user-friendly interface for visualizing network traffic, alerts, and recommendations.

2 OBJECTIVE

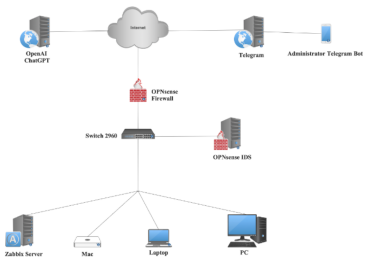
To Setup a network monitoring system using Zabbix

To integrate into the system with network intrusion detection capabilities

To provide alert and mitigation recommendations in the event of an intrusion occurs

3 METHODOLOGY

The seamless integration of OPNsense IDS, Zabbix, and GPT through API creates a robust, intelligent, and responsive network monitoring and alerting system. This setup not only ensures efficient real-time threat detection and network traffic analysis but also empowers network administrators with instant alerts and tailored recommendations. By leveraging advanced tools and automation, the system optimizes network security management, providing a forward-thinking solution for proactive defense and decision-making in dynamic network environments.



4 RESULTS

The integrated platform aims to enhance intrusion detection accuracy, reduce false positives, and strengthen an organization's cybersecurity defenses. The successful implementation of this system will improve the effectiveness of network security monitoring and incident response processes.

5 CONCLUSION

By addressing the limitations of traditional approaches and providing intelligent alert prioritization and mitigation recommendations, the proposed solution contributes to improving the accuracy and efficiency of intrusion detection, reducing false positives, and enabling more effective resource allocation to address high-risk threats.

PLAGIARISM CHECK RESULT

Turnitin Originality Report

Processed on: 13-Sep-2024 04:13 +08
ID: 2451956383
Word Count: 20743
Submitted: 2

FYP2_To Jin Yi_1900176.docx By To Jin Yi

Similarity Index	Similarity by Source	
2%	Internet Sources:	0%
	Publications:	2%
	Student Papers:	0%

1% match (publications)
Zichang Wang. "A new formalized model for motivation", Open Science Framework, 2024

< 1% match (publications)
Inam Ullah Khan, Salma El Hajjami, Mariya Ouaissa, Salwa Belaqziz, Tarandeep Kaur Bhatia. "Cognitive Machine Intelligence - Applications, Challenges, and Related Technologies", CRC Press, 2024

< 1% match (publications)
Zouheir Trabelsi, Kadhim Hayawi, Arwa Al Braiki, Sujith Samuel Mathew. "Network Attacks and Defenses - A Hands-on Approach", Auerbach Publications, 2019

< 1% match (Sabah Alzahrani, Liang Hong. "A Survey of Cloud Computing Detection Techniques against DDoS Attacks", Journal of Information Security, 2018)
Sabah Alzahrani, Liang Hong. "A Survey of Cloud Computing Detection Techniques against DDoS Attacks", Journal of Information Security, 2018

< 1% match (publications)
Masahiko Higashino, Shin Okamoto. "Response Control and Seismic Isolation of Buildings", Taylor & Francis, 2006

< 1% match (Mohamed Ali EL-Omairi, Abdelkader El Garouani. "A review on advancements in lithological mapping utilizing machine learning algorithms and remote sensing data", Heliyon, 2023)
Mohamed Ali EL-Omairi, Abdelkader El Garouani. "A review on advancements in lithological mapping utilizing machine learning algorithms and remote sensing data", Heliyon, 2023

< 1% match (publications)
Arvind Dagur, Dhirendra Kumar Shukla, Nazarov Fayzullo Makhmadiyarovich, Akhatov Akmal Rustamovich, Jabborov Jamol Sindorovich. "Artificial Intelligence and Information Technologies", CRC Press, 2024

< 1% match (publications)
Rajeev Agrawal, Arun Pratap Srivastava, Akihiko Sugiyama. "International Conference on Sustainability in Digital Transformation Era: Driving Innovative & Growth - (Sidte 2023) 16 – 17 September 2023", CRC Press, 2024

< 1% match (publications)
Sasho Kalajdzievski. "Math and Art - Math and Art", CRC Press, 2021

< 1% match (publications)
Vandana Mohindru Sood, Yashwant Singh, Bharat Bhargava, Sushil Kumar Narang. "Intelligent Security Solutions for Cyber-Physical Systems", CRC Press, 2024

< 1% match (publications)
Amit Kumar Tyagi, Niladhuri Sreenath. "Handbook of Research of Internet of Things and Cyber-Physical Systems - An Integrative Approach to an Interconnected Future", CRC Press, 2022

< 1% match (publications)
O. P. Verma, Seema Verma, Thinagaran Perumal. "Advancement of Intelligent Computational Methods and Technologies - Proceedings of I International Conference on Advancement of Intelligent Computational Methods and Technologies (AICMT2023)", CRC Press, 2024

< 1% match (Jessey Bullock. "Wireshark® for Security Professionals", Wiley, 2017)
Jessey Bullock. "Wireshark® for Security Professionals", Wiley, 2017

< 1% match (Abebe Tesfahun, D. Lalitha Bhaskari. "Effective Hybrid Intrusion Detection System: A Layered Approach", International Journal of Computer Network and Information Security, 2015)
Abebe Tesfahun, D. Lalitha Bhaskari. "Effective Hybrid Intrusion Detection System: A Layered Approach", International Journal of Computer Network and Information Security, 2015

< 1% match (publications)
Prashant Pranav, Archana Patel, Sarika Jain. "Machine Learning in Healthcare and Security - Advances, Obstacles, and Solutions", CRC Press, 2024

< 1% match (Chad Russell. "Chapter 5 Frameworks", Springer Science and Business Media LLC, 2016)
Chad Russell. "Chapter 5 Frameworks", Springer Science and Business Media LLC, 2016

< 1% match (publications)
Dhruba Kumar Bhattacharyya. "DDoS Attacks - Evolution, Detection, Prevention, Reaction, and Tolerance", Chapman and Hall/CRC, 2019

< 1% match (publications)

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	TO JIN YI
ID Number(s)	1900176
Programme / Course	CN
Title of Final Year Project	Network Intrusion Detection and Alert System

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 2 % **Similarity by source** Internet Sources: 0 % Publications: 2 % Student Papers: 0 %	
Number of individual sources listed of more than 3% similarity: 0	
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

GMLee

Signature of Supervisor

Name: Gan Ming Lee

Date: 13/9/2023

Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

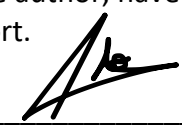
CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	1900176
Student Name	TO JIN YI
Supervisor Name	DR. GAN MING LEE

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.



_____
(Signature of Student)
Date:13/09/2024