Secure Network Monitoring using Software Defined Networking (SDN) with Ryu Controller

[1]Elizabeth A. Amusan, [2]Oluwaseun M. Alade, and [3]John J. Alabi
[1]Department of Cyber Security Science, Ladoke Akintola University of Technology, Ogbomoso, Nigeria.
eaadewusi@lautech.edu.ng, oalade75@lautech.edu.ng, alabijjohn@gmail.com

Received: 22-MAY-2024; Reviewed: 16-JUNE-2024; Accepted: 20-JUNE-2024
https://dx.doi.org/10.4314/fuoyejet.v9i2.12

ORIGINAL RESEARCH

Abstract — Secure network monitoring is critical for detecting and combating threats in real-time especially in today's fast changing cybersecurity terrain. Software defined network (SDN) is an approach to network management that provides visibility, programmability and flexibility. In this work, the implementation of secure network monitoring using the Ryu Controller within a simulated environment created with Mininet on Ubuntu was investigated. Specifically, Mininet was used to set-up a star topology which consists of 8 hosts (h1–h8), including 3 server hosts (h1–h3) and 5 client hosts (h4–h8), along with 1 switch (s1) and a controller. A stateful firewall security measure was implemented leveraging the Ryu Controller's centralized control and programmability. The proposed firewall was evaluated using functional testing such as traffic blocking, malicious packet detection and NAT handling. This work significantly improves resilience against volumetric Denial-of-Service (DoS) attacks through even distribution of traffic across network hosts and disallowing unauthorized communications. Experimental results show the effectiveness of SDN-based monitoring in sustaining network availability and continued service. By adopting SDN technologies, the cybersecurity posture of institutions and organizations can be enhanced by improving their threat detection and incidence response capabilities thereby mitigating the risks associated with modern cyber threats and network vulnerabilities. This approach offers valuable insights for future network monitoring and management and future work may explore the integration of advanced machine learning (CTI Analysis) to further enhance network performance.

Keywords— Cybersecurity, Network Monitoring, Ryu Controller, Software Defined Networking (SDN).

1 Introduction

The ubiquity of interconnected devices and rising digitization of critical infrastructure has made it necessary to ensure robust cybersecurity measures in order to safeguard digital assets. This interconnectedness introduces new vulnerabilities and risks, making them prime targets for cyber threats such as data breaches, ransomware attacks, and network intrusions (Celesova et al, 2019).

The effect of cyber-attacks can be devastating in a smart economy where data is at the core of its operations and decision-making (Deepa et al, 2018; Aladaileh et al, 2020). These attacks can lead to serious financial losses, tarnish reputations, and disrupt important services (Priya and Manjula, 2019; Novaes et al, 2021). Thus, cybersecurity strategies such as software-based networks that not only safeguard data and digital assets but also foster trust can combat these attacks (Alaoui et al, 2021). They constitute the backbone of the smart economy and are therefore not negotiable.

Hence, the aim of this research is to leverage the fundamental principles of SDN, particularly centralized control and programmability, to configure the Ryu controller within the SDN framework. The objective is twofold: first, to distribute network traffic evenly among all server hosts, thereby mitigating any potential Denial-of-Service (DoS) attacks, specifically, we target volumetric DoS attacks, which slows down network (resources) with high volumes of traffic.

Secondly, to implement stateful firewall to dynamically block traffic from unauthorized IP addresses or IP addresses using NAT. Existing SDN solutions often lack these dynamic and robust security measures, which this research aims to provide. By implementing these security measures, the network's resilience against DoS attacks is significantly enhanced, ensuring continuous availability of network security and reliability in a smart economy.

The rest of this paper is organized as follows: section 2 gives a thorough review of the literature on SDN and related works. Section 3 presents the methodology employed in conducting the research. The results obtained and corresponding analysis are discussed in chapter 4. Finally, Section 5 gives the conclusion and highlights recommendations for future research.

2 LITERATURE REVIEW

Secure network monitoring involves the continuous observation and analysis of network traffic and activities to identify possible security threats and vulnerabilities. The primary goal of secure network monitoring is to ensure the integrity, confidentiality, and availability of network resources by detecting and responding to unauthorized access, data breaches, malware infections, and other malicious activities in real-time (Iqbal et al, 2019). Secure network monitoring is an essential component of a comprehensive cybersecurity strategy as it provides organizations with the visibility and insights needed to protect their networks, systems, and data from ever-evolving cyber threats (Etxezarreta et al, 2023).

Traditional network monitoring methods face several challenges that can limit their effectiveness and efficiency,

*Corresponding Author(s): eaadewusi@lautech.edu.ng, oalade75@lautech.edu.ng, alabijjohn@gmail.com

Section B- ELECTRICAL/COMPUTER ENGINEERING & COMPUTING SCIENCES

Can be cited as:
Amusan E.A., Alade O.M., and Alabi J.J., (2024). Secure Network Monitoring using Software Defined Networking (SDN) with Ryu Controller, FUOYE Journal of Engineering and Technology (FUOYE**JET**), **9(2), 235-240.** https://dx.doi.org/10.4314/fuoyejet.v9i2.12

particularly in today's complex and dynamic network environments. These challenges among others include limited visibility, inflexibility, scalability issues, manual configuration. These limitations underscore the need for more advanced, flexible, and scalable approaches like Software-Defined Networking (SDN) and the use of controllers such as the Ryu Controller.

A software-defined network utilizes an SDN controller to manage communications between applications and network devices. This implies that all devices are contained in a central node, and interactions between network devices and network applications are overseen in a secure and transparent manner (Alaoui et al, 2021). It is a networking paradigm that separates the control plane from the data plane. A typical SDN architecture is made up of three (3) layers or planes as illustrated in Figure 1.

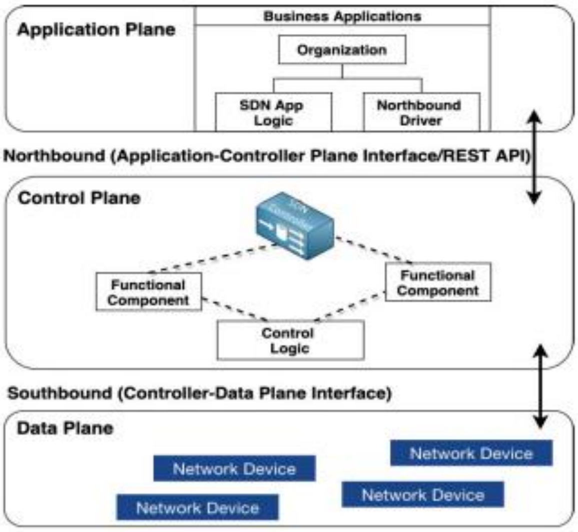


Fig. 1. Overview of SDN architecture (Rahouti et al, 2021)

For effective communication among these layers, SDN uses Northbound interface (NBI) and Southbound interface (SBI) which are application program interfaces (APIs). Highlighted hereunder are some of its key principles.

a. Separation of Control Plane and Data Plane

Network administrators are able to manage the network centrally using software, rather than relying on traditional, hardware-centric networking methods. The principle abstracts the control plane and centralizes it in software-based controllers, while the data plane remains in the networking devices.

b. Network Programmability

SDN facilitates network programmability, allowing administrators to control and manage network performance through software applications rather than manual configuration of each device.

c. Centralized Management

With SDN, network management is centralized within software-based controllers, which function as the brains of the network. Centralized management eases network administration, as administrators can implement changes across the whole network from a single point of control, rather than configuring each device.

The Ryu Controller which is an open-source SDN controller, presents a platform for developing network applications and controlling network resources (Bhardwaj and Panda, 2022). It supports several SDN protocols such as OpenFlow and offers various features for efficient network management. By leveraging SDN and the Ryu Controller, organizations can enhance their network monitoring and incident response capabilities, enhancing their overall cybersecurity posture. This approach allows for initiative-taking threat detection, swift response, and adaptable network management in the face of evolving cyber threats.

2.1 RELATED WORKS

Several researches have been carried out and reported in literature on various aspects of SDN, particularly bothering on security, with the objective to mitigate vulnerabilities, threats, and proffer solutions in this domain. Queiroz et al (2019) employed big data techniques for SDN traffic monitoring. The research used a fine-grained Big Data monitoring approach for collecting and generating traffic statistics through counter values. Urrea and Benitez (2021) appraised different SDN architectures, controllers and solutions and asserted that it can address aspects such as security, network technology independence, and centralized network management. Similarly, Ahmad et al, (2021) did a review on SDN-based network security enhancements where different studies were assessed, presented, and analyzed concerning the enhancement of network security in SDN. Also in 2021, Omran et al used the Ryu controller to monitor network traffic based on different topology scenarios.

Modern economies are confronted with numerous cybersecurity challenges in this age of digital transformation and connected devices. Traditional cybersecurity approaches have inherent limitations in addressing such challenges, as they often rely on perimeter-based security and manual configuration. To effectively mitigate cyber risks in a smart economy, organizations must employ more dynamic security measures that leverage advanced technologies such as Software Defined Networking (SDN).

3 METHODOLOGY

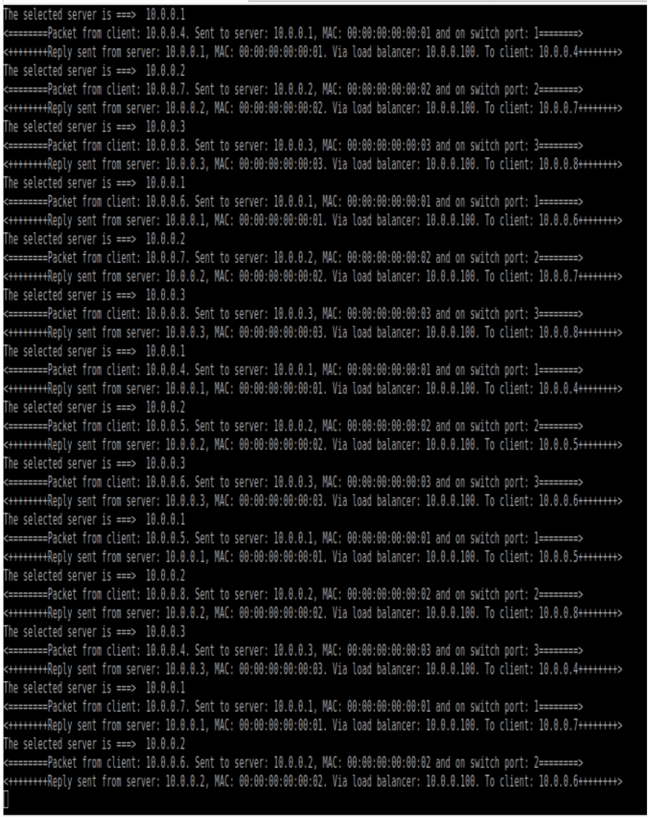
The proposed method for performing the secure network monitoring uses the Mininet software as a simulator on Ubuntu to create the network scenario. Ryu controller log interface and Wireshark were both used in the simulation experiment. The network traffic capture and analysis were conducted using Wireshark (popularly used for network traffic analysis and offers a centralized perspective on traffic distribution), as illustrated in Figure 2. The captured packets were subsequently transferred to DynamiteLab for further analysis. Additionally, the Ryu controller was programmed in Python to enable seamless operation within the simulated network environment.

3.1 TOOLS/MATERIALS USED

Table 1 gives the different tools and platforms used in carrying out the analysis.

Table 1. Tools used

Tool	Specification
Ubuntu	V Box 22.04.3
Ryu framework	Version 4.34
Mininet	V 2.3.0
Wireshark	V 3.2.3
Python	V 3.8.10



```
The selected server is ===>  10.0.0.1
<========Packet from client: 10.0.0.4. Sent to server: 10.0.0.1, MAC: 00:00:00:00:00:01 and on switch port: 1========>
<++++++++Reply sent from server: 10.0.0.1, MAC: 00:00:00:00:00:01. Via load balancer: 10.0.0.100. To client: 10.0.0.4++++++++>
The selected server is ===>  10.0.0.2
<========Packet from client: 10.0.0.7. Sent to server: 10.0.0.2, MAC: 00:00:00:00:00:02 and on switch port: 2========>
<++++++++Reply sent from server: 10.0.0.2, MAC: 00:00:00:00:00:02. Via load balancer: 10.0.0.100. To client: 10.0.0.7++++++++>
The selected server is ===>  10.0.0.3
<========Packet from client: 10.0.0.8. Sent to server: 10.0.0.3, MAC: 00:00:00:00:00:03 and on switch port: 3========>
<++++++++Reply sent from server: 10.0.0.3, MAC: 00:00:00:00:00:03. Via load balancer: 10.0.0.100. To client: 10.0.0.8++++++++>
The selected server is ===>  10.0.0.1
<========Packet from client: 10.0.0.6. Sent to server: 10.0.0.1, MAC: 00:00:00:00:00:01 and on switch port: 1========>
<++++++++Reply sent from server: 10.0.0.1, MAC: 00:00:00:00:00:01. Via load balancer: 10.0.0.100. To client: 10.0.0.6++++++++>
The selected server is ===>  10.0.0.2
<========Packet from client: 10.0.0.7. Sent to server: 10.0.0.2, MAC: 00:00:00:00:00:02 and on switch port: 2========>
<++++++++Reply sent from server: 10.0.0.2, MAC: 00:00:00:00:00:02. Via load balancer: 10.0.0.100. To client: 10.0.0.7++++++++>
The selected server is ===>  10.0.0.3
<========Packet from client: 10.0.0.8. Sent to server: 10.0.0.3, MAC: 00:00:00:00:00:03 and on switch port: 3========>
<++++++++Reply sent from server: 10.0.0.3, MAC: 00:00:00:00:00:03. Via load balancer: 10.0.0.100. To client: 10.0.0.8++++++++>
The selected server is ===>  10.0.0.1
<========Packet from client: 10.0.0.4. Sent to server: 10.0.0.1, MAC: 00:00:00:00:00:01 and on switch port: 1========>
<++++++++Reply sent from server: 10.0.0.1, MAC: 00:00:00:00:00:01. Via load balancer: 10.0.0.100. To client: 10.0.0.4++++++++>
The selected server is ===>  10.0.0.2
<========Packet from client: 10.0.0.5. Sent to server: 10.0.0.2, MAC: 00:00:00:00:00:02 and on switch port: 2========>
<++++++++Reply sent from server: 10.0.0.2, MAC: 00:00:00:00:00:02. Via load balancer: 10.0.0.100. To client: 10.0.0.5++++++++>
The selected server is ===>  10.0.0.3
<========Packet from client: 10.0.0.6. Sent to server: 10.0.0.3, MAC: 00:00:00:00:00:03 and on switch port: 3========>
<++++++++Reply sent from server: 10.0.0.3, MAC: 00:00:00:00:00:03. Via load balancer: 10.0.0.100. To client: 10.0.0.6++++++++>
The selected server is ===>  10.0.0.1
<========Packet from client: 10.0.0.5. Sent to server: 10.0.0.1, MAC: 00:00:00:00:00:01 and on switch port: 1========>
<++++++++Reply sent from server: 10.0.0.1, MAC: 00:00:00:00:00:01. Via load balancer: 10.0.0.100. To client: 10.0.0.5++++++++>
The selected server is ===>  10.0.0.2
<========Packet from client: 10.0.0.8. Sent to server: 10.0.0.2, MAC: 00:00:00:00:00:02 and on switch port: 2========>
<++++++++Reply sent from server: 10.0.0.2, MAC: 00:00:00:00:00:02. Via load balancer: 10.0.0.100. To client: 10.0.0.8++++++++>
The selected server is ===>  10.0.0.3
<========Packet from client: 10.0.0.4. Sent to server: 10.0.0.3, MAC: 00:00:00:00:00:03 and on switch port: 3========>
<++++++++Reply sent from server: 10.0.0.3, MAC: 00:00:00:00:00:03. Via load balancer: 10.0.0.100. To client: 10.0.0.4++++++++>
The selected server is ===>  10.0.0.1
<========Packet from client: 10.0.0.7. Sent to server: 10.0.0.1, MAC: 00:00:00:00:00:01 and on switch port: 1========>
<++++++++Reply sent from server: 10.0.0.1, MAC: 00:00:00:00:00:01. Via load balancer: 10.0.0.100. To client: 10.0.0.7++++++++>
The selected server is ===>  10.0.0.2
<========Packet from client: 10.0.0.6. Sent to server: 10.0.0.2, MAC: 00:00:00:00:00:02 and on switch port: 2========>
<++++++++Reply sent from server: 10.0.0.2, MAC: 00:00:00:00:00:02. Via load balancer: 10.0.0.100. To client: 10.0.0.6++++++++>
```

Fig. 2. Ryu Controller Log Interface

3.2 SETTING UP THE SIMULATED ENVIRONMENT

The study utilizes the Mininet Simulated Environment running on Ubuntu to create a realistic and controllable network environment for experimentation. In SDN-based network simulations, Mininet (Mininet, 2024) is commonly utilized to evaluate their performance. This software is widely regarded as the preferred choice for conducting tests before deploying the final SDN infrastructure. A star topology is configured within the Mininet environment, consisting of 8 hosts (h1–h8), which includes 3 server hosts (h1–h3) and 5 client hosts (h4–h8). Additionally, the network included one switch (s1) to facilitate data forwarding and a controller (Ryu controller) to manage network operations as shown in Figure 3.

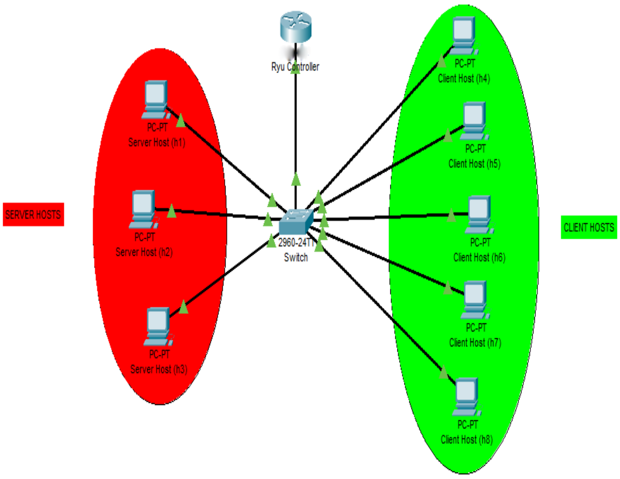


Fig. 3. Topology Model

3.3 IMPLEMENTING SECURE NETWORK MONITORING WITH RYU CONTROLLER

The Ryu Controller, an open-source SDN controller, is deployed within the simulated environment to facilitate secure network monitoring leveraging its centralized control and programmability capability.

The simulation experiment involved analysing the secondary packets of a network under attack and insights gained from which helped to understand the nature of the attack as well as identify the vulnerabilities exploited. Subsequently, the basic principles of Software Defined Networking (SDN), namely centralized control and programmability, are leveraged to develop a software-defined network capable of resisting the denial-of-service attack.

The analysis begins with the use of Wireshark to capture packet traffic, which is then imported into "DynamiteLab" formerly known as PacketTotal, for further examination. This was selected due to its advanced capabilities in detecting and analyzing malicious network packets, providing a more comprehensive assessment of network security compared to Wireshark's primary function of packet capture and display. The methodology for analyzing packet traffic in DynamiteLab follows a systematic approach, consisting of several key steps as follows:

1. Network Graph Analysis: This is needed to see the relationships and connections between various network entities such as devices, servers and endpoints.

2. Timeline Analysis: This involves inspecting network traffic data over a specific period to identify trends, patterns, and events. This helps in understanding the sequence of activities, detecting anomalies in traffic, and correlating events with potential security incidents.

3. Suspicious Traffic Analysis: here, network traffic that exhibits potentially malicious behavior or indicators of compromise are investigated.

4. Host Analysis: this entails investigating the behavior and activity of individual devices within the network. It

https://dx.doi.org/10.4314/fuoyejet.v9i2.12

includes monitoring system logs, process activities, network connections, and other host-related data.

5. Communication Analysis: here, network communication patterns are analyzed.

6. Artifacts Analysis: Artifacts analysis entails examining residual data or traces left behind by network activity, such as log files, temporary files, registry entries, and memory dumps. It assists in reconstructing events, identifying indicators of compromise, and gathering forensic evidence for investigating security incidents.

The process for establishing the SDN network resistant to DoS attacks is outlined as follows:

The SDN network utilizes the Ryu controller as the central controller, which is scripted in Python to integrate two primary security measures which are stateful firewall implementation and traffic handling. A firewall is integrated into the controller to block traffic from sources with IP addresses that utilize NAT or are not within the network. When traffic is directed to the servers, it is first routed to the IP address of the Ryu controller (127.0.0.1). This approach allows the Ryu controller to mask the IP addresses of the three servers (h1 – h3) for efficient traffic control within the software-based network. Subsequently, the Ryu controller evenly routes the traffic across the three servers, effectively implementing the load balancer and enhancing network performance while bolstering resilience against DoS attacks.

4 RESULTS AND DISCUSSION

This section presents the outcome of the traffic analysis and SDN implementation.

4.1 RESULTS OF TRAFFIC ANALYSIS

Network Graph Analysis: Upon conducting network graph analysis, it was observed that the network comprises five hosts with the following IP addresses: 192.168.222.5, 192.168.222.17, 104.236.0.112, 207.246.119.209, and 14.238.2.146 as shown in Figure 4. Additionally, an internal server host with a private IP address of 192.168.22.10 is present. Five client hosts are connected to the server, with three of them directly communicating with the server host by sending traffic, while the remaining two client hosts are solely receiving traffic from the server host.

Further analysis of the network graph revealed that one host is generating heavy traffic, which appears suspicious due to the transmission protocol and signature associated with it. Detailed examination of this suspicious traffic, including communication analysis and transmission protocol, will be conducted in subsequent sections. Figure 5 provides a visual representation of the flagged heavy traffic for reference.

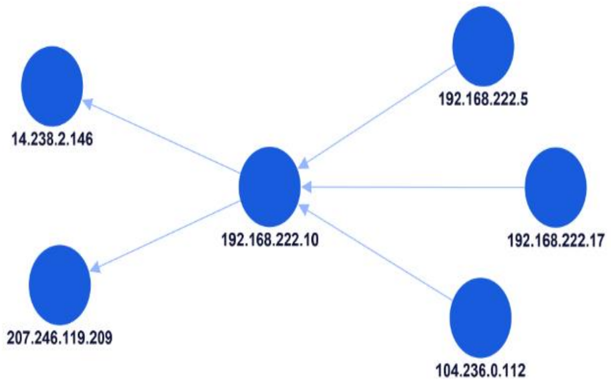


Figure 4: Network graph of the DoS packet

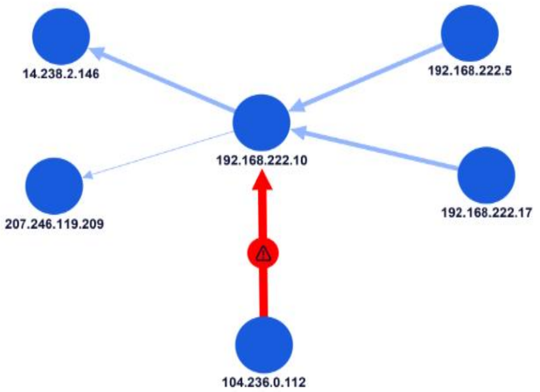
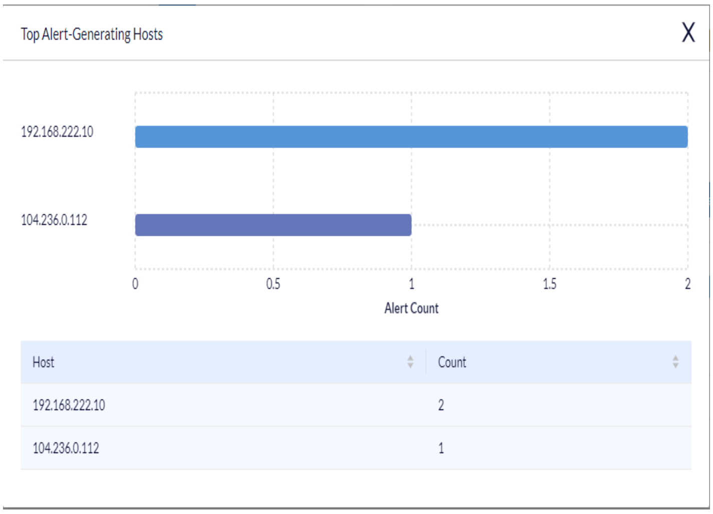


Fig. 5. Network Graph showing flagged traffic.

Suspicious traffic analysis: During the suspicious traffic analysis, it was discovered that there are three instances of suspicious traffic, as indicated by the traffic timeline in Figure 6. Further investigation into these occurrences revealed details about the hosts involved in generating and receiving the suspicious traffic, as well as the criteria used to flag the traffic within the network.


Top Alert-Generating Hosts X

Host	Count
192.168.222.10	2
104.236.0.112	1

Fig. 6. Traffic-generating hosts

Traffic Generating Hosts: Examination of Figure 6

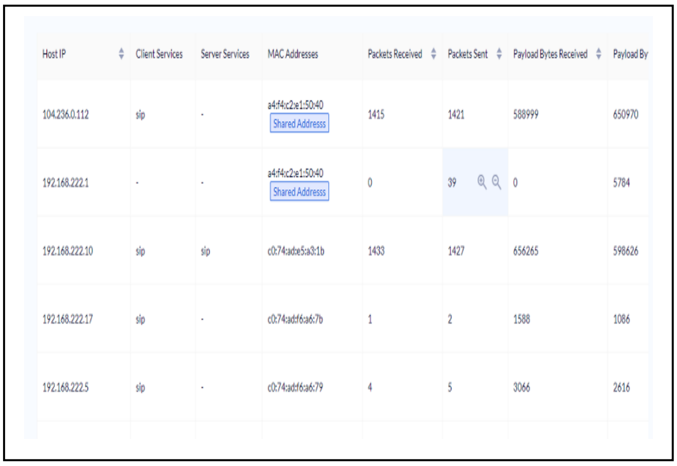
https://dx.doi.org/10.4314/fuoyejet.v9i2.12

revealed that two hosts are responsible for generating the suspicious traffic:

1. The server host with the IP address 192.168.222.10 generated two instances of suspicious traffic.
2. Additionally, one of the client hosts with the IP address 104.236.0.112 generated one instance of the three instances of suspicious traffic observed.

These findings provide insight into the origin of the suspicious traffic and highlight the involvement of specific hosts within the network.

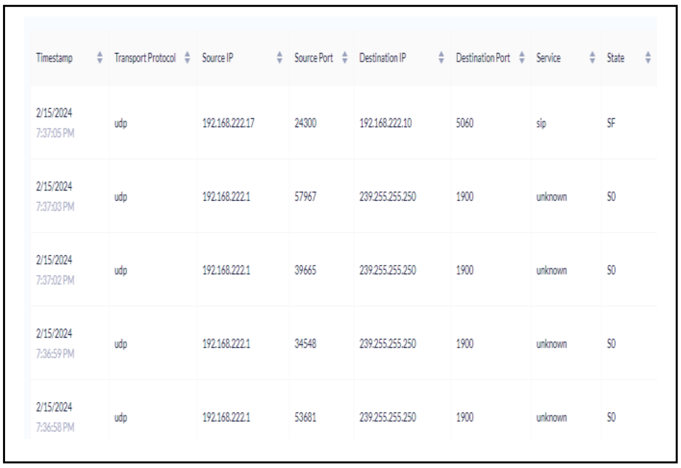
Host analysis result: The host analysis provided comprehensive details about each host engaged in communication within the network, encompassing information about client and server services, host MAC addresses, packet transmission metrics, payload bytes exchanged, connection types, and more. These insights, depicted in Figure 7, serve as valuable resources for forensic investigations or information gathering purposes.



Host IP	Client Services	Server Services	MAC Addresses	Packets Received	Packets Sent	Payload Bytes Received	Payload By
104.236.0.112	sip	-	a4:f4:c2:e1:50:40 Shared Addresss	1415	1421	588999	650970
192.168.222.1	-	-	a4:f4:c2:e1:50:40 Shared Addresss	0	39	0	5784
192.168.222.10	sip	sip	c0:74:ad:e5:a3:1b	1433	1427	656265	598626
192.168.222.17	sip	-	c0:74:ad:f6:a6:7b	1	2	1588	1086
192.168.222.5	sip	-	c0:74:ad:f6:a6:79	4	5	3066	2616

Fig. 7. Host Analysis

Communication analysis: The communication analysis provides in-depth insights into various aspects of network communication, including the transmission protocols utilized, source and destination IP addresses and ports, communication duration, and bytes transmitted and received. This information, illustrated in Figure 8, offers valuable insights for understanding network activities and facilitates informed decision-making in incident response scenarios.

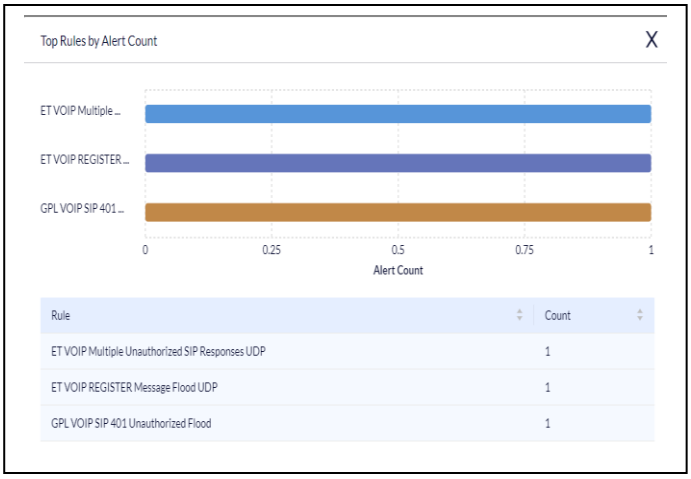


Timestamp	Transport Protocol	Source IP	Source Port	Destination IP	Destination Port	Service	State
2/15/2024 7:37:05 PM	udp	192.168.222.17	24300	192.168.222.10	5060	sip	SF
2/15/2024 7:37:03 PM	udp	192.168.222.1	57967	239.255.255.250	1900	unknown	S0
2/15/2024 7:37:02 PM	udp	192.168.222.1	39665	239.255.255.250	1900	unknown	S0
2/15/2024 7:36:59 PM	udp	192.168.222.1	34548	239.255.255.250	1900	unknown	S0
2/15/2024 7:36:58 PM	udp	192.168.222.1	53681	239.255.255.250	1900	unknown	S0

Fig. 8. Communication Analysis

Firewall rules: further analysis of the rules used to flag the traffic by the IDS/IPS deployed within the network unveiled the signature associated with each suspicious traffic count, as depicted in Figure 9. Each rule is explained, providing insights into the criteria used to identify and classify the traffic as suspicious.

Count 1: This count was flagged by the rule "ET VOIP Multiple Unauthorized Session Initiation Protocol (SIP) Responses UDP" which is a network traffic analysis rule typically used in intrusion detection systems for monitoring VoIP (Voice over Internet Protocol) traffic. It targets instances where multiple unauthorized Session Initiation Protocol (SIP) responses are detected over the User Datagram Protocol (UDP).



Rule	Count
ET VOIP Multiple Unauthorized SIP Responses UDP	1
ET VOIP REGISTER Message Flood UDP	1
GPL VOIP SIP 401 Unauthorized Flood	1

Fig. 9. Rules of Suspicious Traffic

Count 2: The "ET VOIP REGISTER Message Flood UDP" rule is designed to detect and mitigate potential flooding attacks targeting VoIP systems, specifically focusing on REGISTER messages over the User Datagram Protocol (UDP).

Count 3: This count was flagged by the "GPL VOIP SIP 401 Unauthorized Flood" rule and it is part of the General Public License (GPL) set of rules designed to detect and respond to flooding attacks targeting SIP (Session Initiation Protocol) servers. It does this by detecting excessive "401 Unauthorized" responses which indicates improper authentication credentials.

The results obtained demonstrate the effectiveness of our SDN-based firewall and load balancer in mitigating DoS attacks and distributing network traffic. Basically, the implementation of the SDN solution led to a significant reduction in traffic volume on individual servers, indicating effective load distribution. The stateful firewall successfully handles NAT and blocked a high percentage of unauthorized traffic.

This directly address our research objectives by indicating that our SDN-based firewall and load balancer effectively mitigate DoS attacks and distribute network traffic evenly, enhancing overall network security and performance.

https://dx.doi.org/10.4314/fuoyejet.v9i2.12

5 CONCLUSION AND RECOMMENDATIONS

The findings of this study have significant implications for network monitoring and management practices. First, the identification of suspicious traffic patterns and vulnerabilities within the network highlights the importance of proactive monitoring and continuous threat detection measures. By leveraging tools such as Wireshark and DynamiteLab, organizations can gain insights into network activities, detect anomalies, and promptly respond to potential security threats.

Moreover, the implementation of SDN-based secure network monitoring demonstrates promising results in improving threat detection and response capabilities. By centralizing control and enabling programmability, SDN facilitates dynamic and adaptive security measures that can effectively mitigate cyber threats such as DoS attacks. The integration of features such as firewalls and traffic load balancers within the SDN environment enhances network resilience and reduces the impact of malicious activities. However, there are some limitations to our traffic analysis as our analysis primarily focused on detecting and mitigating volumetric DoS attacks, potentially overlooking other types of cyber threats which can be further prevented.

In conclusion, this research highlights the effectiveness of SDN-based secure network monitoring in enhancing cybersecurity posture and mitigating cyber risks. Organizations looking to leverage SDN technologies to bolster their security measures should consider the following recommendations:

1. Invest in SDN Infrastructure: Organizations should invest in robust SDN infrastructure, including controllers, switches, and monitoring tools, to enable centralized control and programmability for enhanced security management.
2. Provide Training and Awareness: Educate network administrators and security personnel on SDN security best practices, emerging threats, and effective incident response strategies. It is obvious that whatever changes will be made to the network or whatever security policies that will be enforced in the network have to be done from the SDN controller which requires that the network administrator acquire scripting knowledge and skill in Python, Java, Go (Goland), Js, and/or Ruby to efficiently manage the network (SDN).

By adhering to these recommendations and leveraging SDN technologies effectively, organizations can significantly improve their ability to detect, respond to, and mitigate cyber threats in today's evolving threat landscape.

REFERENCES

Ahmad A.A., Boukari S., Bello A.M., Madu M. and Gimba S. (2021): A Review on Software Defined Network (SDN) Based Network Security Enhancements. Journal of Software Engineering and Simulation Volume 7, Issue 9 (2021) pp: 01-08.

Aladaileh, M.A.; Anbar, M.; Hasbullah, I.H.; Chong, Y.W.; Sanjalawe, Y.K. (2020): Detection Techniques of Distributed Denial of Service Attacks on Software-Defined Networking Controller—A Review. IEEE Access 2020, 8, 143985–143995.

Alaoui R.M., Claver N.P., Aissata C., Samake M. and Bahnasse A. (2021): Use cases of SDN for network security. Turkish Online Journal of Qualitative Inquiry (TOJQI) Volume 12, Issue 7, July 2021 : 7589 – 7594.

Bhardwaj, S. and Panda, S.N. (2022): Performance Evaluation Using RYU SDN Controller in Software-Defined Networking Environment. Wireless Personal Comm. 122, 701–723 (2022). https://doi.org/10.1007/s11277-021-08920-3

Benzaïd, C.; Boukhalfa, M.; Taleb, T. Robust Self-Protection Against Application-Layer (D)DoS Attacks in SDN Environment (2020): In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 25–28 May 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.

Celesova, B.; Val'ko, J.; Grezo, R.; Helebrandt, P. Enhancing security of SDN focusing on control plane and data plane. In Proceedings of the 2019 7th International Symposium on Digital Forensics and Security (ISDFS), Barcelos, Portugal, 10–12 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.

Deepa, V.; Sudar, K.M.; Deepalakshmi, P. (2018): Detection of DDoS attack on SDN control plane using Hybrid Machine Learning Techniques. In Proceedings of the 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 13–14 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 299–303.

Hameed, S.; Ahmed Khan, H. SDN based collaborative scheme for mitigation of DDoS attacks. Future Internet 2018, 10, 23.

Iqbal M., Iqbal F., Mohsin F., Rizwan M. and Ahmad F. (2019): Security Issues in Software Defined Networking (SDN): Risks, Challenges and Potential Solutions. International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 10, No. 10, pp. 298-303.

Mininet: An Instant Virtual Network on Your Laptop (or Other PC)—Mininet. Available online: http://mininet.org/ (last accessed on 20 February, 2024).

Omran M. A. Alssaheli, Z. Zainal Abidin, N. A. Zakaria, Z. Abal Abas (2021): Implementation of Network Traffic Monitoring using Software Defined Networking Ryu Controller. WSEAS TRANSACTIONS on SYSTEMS and CONTROL DOI: 10.37394/23203.2021.16.23

Queiroz W., Capretz M. and Dantas M. (2019): An approach for SDN traffic monitoring based on big data techniques. Journal of Network and Computer Applications 131 (2019), pp. 28-39.

Novaes, M.P.; Carvalho, L.F.; Lloret, J.; Proença, M.L.(2021): Adversarial Deep Learning approach detection and defense against DDoS attacks in SDN environments. Future Gener. Comput. Syst. 2021, 125, 156–167.

Priya, P.M.; Manjula, K.R. (2019): Cog-SDN: Mitigation Mechanism for Distributed Denial of Service Attacks in Software Defined Networks. In Proceedings of the International Conference on Applications and Techniques in Information Security, Tamil Nadu, India, 22–24 November 2019; Springer: Singapore, 2019; pp. 202–215.

Rahouti M., Xiong K. and Xin Y. (2021): Secure Software-Defined Networking Communication Systems for Smart Cities: Current Status, Challenges, and Trends. IEEE Access. Digital Object Identifier 10.1109/ACCESS.2020.3047996. Vol. 9, 2021, pp. 12083-12113.

Suartana M. and Putra R.E. (2022): Software-Defined Networking (SDN) Traffic Analysis Using Big Data Analytic Approach.

Urrea C. and Benítez D. (2021): Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things: A Review. Sensors 2021, 21(19), 6585; https://doi.org/10.3390/s21196585.

Xabier Etxezarreta, Iñaki Garitano, Mikel Iturbe and Urko Zurutuza (2023): Software-Defined Networking approaches for intrusion response in Industrial Control Systems: A survey. International Journal of Critical Infrastructure Protection 42(2023), 100615, pp. 1-17.

https://dx.doi.org/10.4314/fuoyejet.v9i2.12