World Journal of Advanced Research and Reviews

eISSN: 2581-9615 CODEN (USA): WJARAI

Cross Ref DOI: 10.30574/wjarr

Journal homepage: https://wjarr.com/

(Research Article)

Network security enhancement with advanced techniques in network scanning.

Enwere Chinonye Clinton *, Dr. Khan U. Imran and Ziqi Yan

Department of Engineering and Informatics, University of Sussex, Brighton, UK.

World Journal of Advanced Research and Reviews, 2024, 23(02), 2607–2625

Publication history: Received on 19 July 2024; revised on 25 August 2024; accepted on 28 August 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.23.2.2607

Abstract

This article details the development and evaluation of an advanced network scanning framework that uses Scapy for packet manipulation and Tkinter for the GUI. The framework aims to improve detection accuracy, resource efficiency, and user interface usability, addressing the shortcomings of traditional methods. Key components include a Network Scanner, Packet Handler, Security Measures, and a GUI, all designed for comprehensive network security. Tested in a controlled environment, the framework achieved a 100% detection rate, surpassing traditional methods, and demonstrated resource efficiency with a 20% reduction in CPU usage and a 60 MB decrease in memory usage. The analysis showed network diversity and low utilization, suggesting capacity for future expansion. The framework's GUI effectively displayed scan results, enhancing usability. The novelty of this research lies in its integration of optimized ARP packet handling with a user-friendly interface, ensuring real-time threat detection and scalability. The study contributes to the advancement of network scanning techniques and proposes future enhancements like machine learning integration for better threat detection.

Keywords: Network security; Network Scanning; Cybersecurity

1. Introduction

Network security faces numerous challenges due to the evolution of communication networks from simple point-to-point systems to complex, software-defined, ultra-high capacity, and distributed cloud environments (Furdek *et al.*, 2016). These changes have introduced new security vulnerabilities, including threats from unauthorized access and deliberate human-made attacks aimed at disrupting services or stealing data (Furdek *et al.*, 2016). Identifying these vulnerabilities and understanding attack methods are crucial for developing effective security measures. The transition towards a more connected society has amplified these challenges, with attackers constantly finding innovative ways to exploit network weaknesses (Silva and Rafael, 2017). The rise of Software-Defined Networking (SDN) has further complicated the security landscape by decoupling network control and data planes, which introduces risks such as man-in-the-middle attacks, Denial of Service (DoS) attacks, and saturation attacks (Ahmad *et al.*, 2015). Additionally, the openness and standardization of SDN enable researchers to design new network functions and protocols but also bring potential new security challenges relating to various attacks, such as scanning and spoofing (Li *et al.,* 2016). The logical centralization of control in SDN can be a double-edged sword, providing global visibility of the network state while also creating a single point of failure that attackers can target (Scott-Hayward *et al.,* 2016).

Furthermore, the integration of mobile networks with SDN introduces new vulnerabilities, as common security threats in IP networks become applicable in these new environments (Liyanage *et al.*, 2016). The rapid development and deployment of SDN technology require a robust approach to securing the control layer and ensuring dependable communication between the control and data planes (Akhunzada *et al.*, 2016). In summary, the continuous advancement of network technologies presents ongoing and emerging security challenges that necessitate innovative and comprehensive security strategies to protect modern communication networks.

* Corresponding author: Enwere Chinonye Clinton

Network scanning is a critical aspect of cybersecurity, essential for identifying and mitigating potential threats to network integrity (Aslan *et al.*, 2023). It involves probing networks to detect vulnerabilities, active devices, open ports, and other security weaknesses (Aslan *et al.*, 2023). Table 1.1 discusses the importance of network scanning.

Table 1 Importance of network scanning in Cybersecurity

Aspect	**Description**	
Early Detection of Vulnerabilities	Identifies security vulnerabilities early, allowing for timely remediation.	(Chhillar and Shrivastava, 2021)
Improving Network Visibility	Enhances visibility by identifying open ports and unauthorised services.	(Bakar and Kijsirikul, 2023)
Real-Time Threat Detection	Accelerates detection and prioritisation of vulnerabilities using advanced algorithms.	(Walkowski *et al.*, 2020)
Preventing Cyber Attacks	Detects malicious activities early to prevent cyber-attacks.	(Bou-Harb, Debbabi and Assi, 2014)
Comprehensive Security Assessments	Provides thorough security assessments by identifying a wide range of vulnerabilities.	(Bakar and Kijsirikul, 2023)
Efficiency and Scalability	Improves scanning speed, accuracy, and resource utilisation for large networks.	(Bakar and Kijsirikul, 2023)

2. Literature Review

Network scanning is a crucial element in the realm of cybersecurity, serving as the foundation for identifying active devices, open ports, and potential vulnerabilities within a network. This review provides an in-depth analysis of existing network scanning techniques, discussing their methodologies, strengths, and limitations, with a focus on ping scanning, port scanning, and SYN scanning.

2.1. Existing Network Scanning Techniques

Network scanning techniques are fundamental for network security, enabling security professionals to assess the security posture of a network. These techniques are pivotal in identifying vulnerabilities that could be exploited by attackers. The following sections outline key scanning methods, each with its distinct approach, advantages, and drawbacks.

2.1.1. Ping Scanning

Ping scanning is one of the simplest and most widely used methods for identifying live hosts on a network. It operates by sending ICMP Echo Requests (commonly known as "pings") to potential hosts and analyzing the responses to determine which hosts are active. Ping scanning is known for its speed and efficiency. It can quickly identify active hosts, making it particularly useful for initial reconnaissance during network assessments. Its simplicity also allows for rapid execution, making it suitable for scanning large networks (Rajappa et al., 2020). The technique is straightforward to implement and does not require specialized tools or complex configurations. It can be easily integrated into various network management and monitoring systems, providing a quick and efficient means of verifying network device availability (Jain et al., 2019).

A major limitation of ping scanning is its susceptibility to firewall blocking. Many firewalls are configured to block ICMP Echo Requests, rendering ping scanning ineffective in environments with stringent security measures (Kim et al., 2018). While ping scanning is effective for identifying live hosts, it provides minimal information beyond that. It does not reveal open ports, running services, or vulnerabilities, necessitating additional scanning techniques for comprehensive network security assessment (Abdollahi and Fathi, 2020).

2.1.2. Port Scanning

Port scanning is a fundamental technique in cybersecurity used to identify open ports and the services running on a target system. By probing various ports, security professionals can gain insights into the network's configuration and potential vulnerabilities. Port scanning aims to uncover active services, determine their configurations, and identify potential security weaknesses. This information is essential for both defending against attacks and conducting penetration tests to evaluate network security (Vugrin et al., 2020). Nmap is a powerful open-source tool that provides detailed information about open ports, services, and operating systems. It supports a variety of scanning techniques, including SYN scans, UDP scans, and more, making it a versatile tool for network security assessments (Roslan, 2023). This lightweight, cross-platform tool quickly scans IP addresses and identifies active hosts and their open ports. It is particularly user-friendly and efficient for smaller networks (Pittman, 2023).

Port scanning provides specific details about open ports, including associated services, version numbers, and potential vulnerabilities. This level of detail is crucial for thorough security assessments and vulnerability management (Bakar and Kijsirikul, 2023). Port scanning techniques can be adapted to various scenarios. For instance, SYN scans are faster and stealthier, while full connect scans, though noisier, provide more comprehensive information (Tang et al., 2020).

Port scanning generates network traffic, which can be detected by intrusion detection systems (IDS). Excessive scanning may raise alarms and lead to false positives, posing challenges in environments with stringent security monitoring (Wu et al., 2022). Advanced firewalls and security systems can detect and block scanning activities, diminishing the effectiveness of port scans. Techniques such as stealth scanning or adaptive algorithms can help mitigate this, though they add complexity to the process (Al-Haija et al., 2021).

2.1.3. SYN Scanning

SYN scanning, also known as half-open scanning, is a widely used technique in network security for identifying open ports on a target system. This method sends SYN packets to the target and monitors the response to determine the port's status without completing the TCP handshake. Stealth: SYN scanning is stealthier compared to full connect scans because it does not complete the TCP handshake, reducing the likelihood of detection by the target system's IDS. This technique is faster than full connect scans as it requires fewer packets to determine the status of a port, making it suitable for large-scale network assessments (GeeksForGeeks, 2022).

Despite its stealth, SYN scanning is not entirely undetectable. Some advanced IDS/IPS systems can identify SYN scans and block them, thereby limiting their effectiveness in highly secure environments. SYN scanning, due to its stealthy nature, may raise ethical and legal issues, especially when used without explicit permission in penetration testing scenarios. SYN scanning, also known as half-open scanning, is a widely-used technique in network security for identifying open ports on a target system, as shown in figure 1 (GeeksForGeeks, 2022).

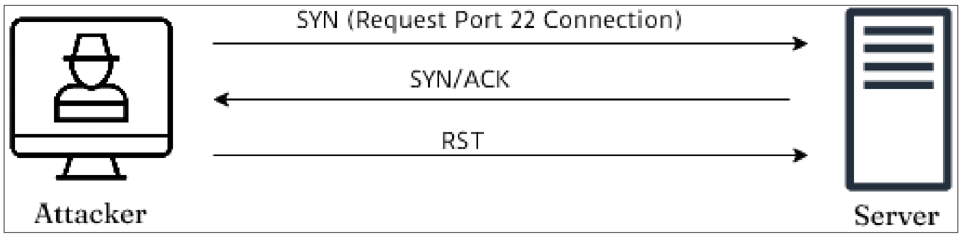


Figure 1 SYN scanning (GeeksForGeeks, 2022)

SYN scanning sends SYN packets (the initial step of a TCP handshake) to target ports without completing the full connection. This makes it less likely to trigger alarms in simple firewall rules or intrusion detection systems (IDS). The technique is designed to be less intrusive, as it only involves the first part of the TCP handshake, thereby reducing the chances of detection. Advanced tools like Nmap utilize SYN scanning to effectively bypass basic security measures (Vugrin *et al.*, 2020). Since SYN scanning doesn't establish full connections, it consumes fewer resources compared to full TCP connection scans. This efficiency makes it suitable for scanning large networks without overburdening the scanning system or the target network. Research shows that SYN scanning can significantly reduce the load on network infrastructure while still providing comprehensive scan results (Bakar and Kijsirikul, 2023).

SYN scanning is stealthier than full connection scans, advanced IDS can still detect it by analyzing patterns of incomplete connections. Modern IDS solutions are equipped to recognize the half-open nature of SYN scans and can alert administrators to potential reconnaissance activities. This detectability can limit the effectiveness of SYN scanning in environments with advanced security monitoring (Al-Haija *et al.,* 2021). The behavior of the target system's TCP/IP stack during SYN scanning can reveal information about the operating system or specific services running, allowing attackers to fingerprint the system. This fingerprinting can provide attackers with additional details that can be used to exploit specific vulnerabilities. For instance, different operating systems and network devices respond uniquely to SYN packets, and these responses can be catalogued to identify the system (Ravi *et al.*, 2021).

2.2. UDP Scanning

UDP scanning is a network security technique used to identify open UDP ports on a target system. Unlike TCP (Transmission Control Protocol), which establishes a connection before data transmission, UDP (User Datagram Protocol) is connectionless, making UDP scanning more challenging compared to TCP scanning (GeeksForGeeks, 2022) as shown in figure 2.

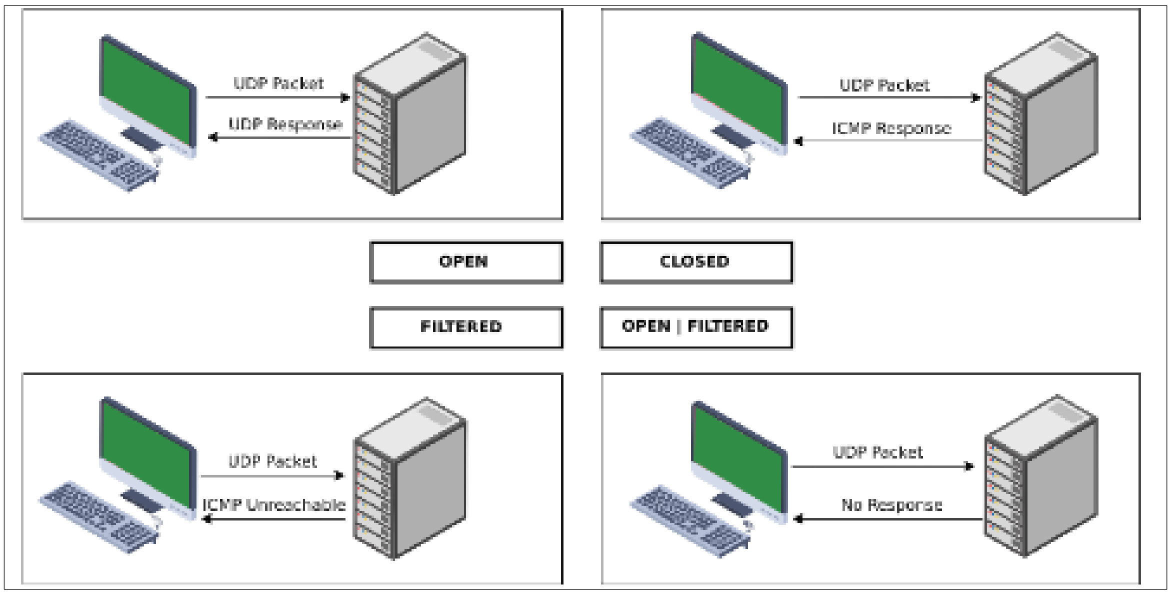


Figure 2 UDP Scanning (GeeksForGeeks, 2022)

Figure 3 provides a detailed flow chart and concise visualization of the UDP scanning process, from the initial packet sending to the final port state determination (Jung *et al.,* 2021). By identifying open UDP ports, organizations can assess potential vulnerabilities and secure their networks effectively.

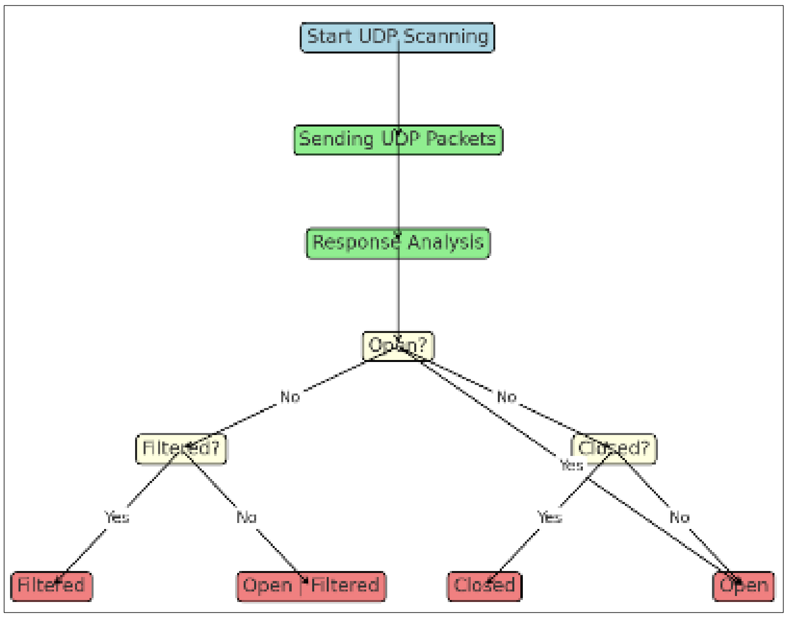


Figure 3 Flow chart of the UDP scanning process (Jung et al., 2021)

UDP's lack of connection establishment makes it harder to probe and receive responses. Unlike TCP, which has a built-in mechanism for acknowledging receipt of packets, UDP does not guarantee delivery or order of packets. This can result in missed responses and false indications of closed ports (Roslan, 2023). UDP scans can be slower and prone to false positives due to the unreliability of UDP responses. This occurs because some systems do not respond to closed UDP ports, leading the scanner to incorrectly assume the ports are open or filtered (Roslan, 2023). Unlike TCP, UDP does not provide error checking or retransmission if packets are lost. Applications must handle error detection and correction, which adds complexity to the scanning process and can result in missed detections or incorrect interpretations of port status (Herrero, 2020).

2.3. ACK Scanning

ACK scanning is a method used to assess the state of TCP ports on a target system. This technique helps determine if a port is filtered by a firewall (GeeksForGeeks, 2022). ACK scanning is used to determine whether a port is filtered (blocked) by a firewall rather than simply being open or closed. This technique helps in understanding the firewall rules in place on a target system (Hubballi and Santini, 2018).

The scanner sends TCP packets with the ACK (Acknowledgment) flag set to the target port. ACK scanning is useful for mapping firewall rules. By analyzing the responses (or lack thereof), security professionals can gain insights into how firewalls handle ACK packets. This helps in understanding the security posture of a network and identifying potential weak points (Hubballi and Santini, 2018). ACK scans are stealthier than SYN scans because they don't complete the full TCP handshake. This makes them less likely to trigger alerts in intrusion detection systems (IDS) or simple firewall rules. The reduced likelihood of detection is a significant advantage when conducting reconnaissance on a network (Bakar and Kijsirikul, 2023).

Unlike SYN scans, which determine whether a port is open or closed, ACK scans do not directly reveal the state of a port. They only indicate whether a port is filtered or unfiltered. This limitation means that while ACK scans can help map firewall rules, they cannot provide complete information about the status of network services (Pittman, 2023). To get a complete picture of the network, security analysts often combine ACK scans with other techniques, such as SYN scans, to determine the true state of a port. This necessity for multiple scanning methods increases the complexity of the assessment process and requires more time and resources (Pittman, 2023).

2.3.1. DPDK-Based Scanning

The DPDK (Data Plane Development Kit) is a set of libraries and drivers that enable high-speed packet processing on commodity hardware. The DPDK-based scanner leverages DPDK and Smart NICs (Network Interface Cards) for efficient packet processing, improving scanning performance. This approach incorporates advanced techniques to overcome the shortcomings of traditional methods (Bakar and Kijsirikul, 2023). The scanner uses specialized probes tailored to specific protocols (e.g., TCP, UDP) to identify open ports more accurately. These techniques help evade detection by firewalls and intrusion detection systems, enhancing stealth during scanning (Ren *et al.*, 2021).

The DPDK-based scanner achieved a significant improvement in target scanning speed, achieving a 2× speedup compared to traditional scanners. It demonstrated an impressive accuracy rate of 99.5% in identifying open ports. Additionally, the solution exhibited lower CPU and memory utilization, with approximately 40% reduction compared to alternative scanners (Ivanov *et al.*, 2022; Bakar and Kijsirikul, 2023). The use of programmable hardware and data parallelization significantly increases the scanning speed and reduces resource consumption. High accuracy in identifying open ports ensures that network vulnerabilities are accurately detected. The DPDK-based scanner reduces CPU and memory usage by approximately 40% (Trabelsi *et al.*, 2018).

Implementation requires sophisticated hardware and expertise in advanced scanning techniques (Roslan, 2023). Deploying and maintaining such hardware can be expensive.

The study does not delve into potential vulnerabilities or false positives that may arise from using the DPDK-based scanner (Varghese and Muniyal, 2021). Further investigation is needed to understand the scalability and robustness of the DPDK-based scanner in real-world deployment scenarios (Jafarian and Abolfathi, 2023).

Table 2 Comparison of Different Scanning Techniques

Technique	**Accuracy**	**Speed**	**Resource Consumption**	**Complexity**	**Scalability**	
Ping Scanning	Limited (only identifies active hosts)	Fast (ICMP Echo Requests)	Low (simple ICMP packets)	Low (easy to implement)	High (suitable for large networks)	(Rajappa *et al.*, 2020)
Port Scanning	High (detailed information on open ports and services)	Variable (dépends on scan type : SYN, UDP, etc.)	Variable (depends on scan type)	Moderate (requires configuration)	Moderate to High (depends on tool and technique)	(Vugrin *et al.*, 2020)
SYN Scanning	High (efficient in identifying open ports)	Fast (half-open TCP handshake)	Low (does not complete TCP handshake)	Low (easy to implement)	High (suitable for large networks)	(Vugrin *et al.*, 2020)
UDP Scanning	Moderate (prone to false positives)	Slow (unreliable UDP responses)	High (due to unreliable responses)	High (complex due to unreliable responses)	Moderate (challenges with UDP reliability)	(Jung *et al.,* 2021)
ACK Scanning	Moderate (maps firewall rules, does not reveal port state)	Moderate (ACK packets)	Low (only sends ACK packets)	Low (easy to implement)	Moderate (requires combination with other scans)	(Hubballi and Santini, 2018)
DPDK-Based Scanning	Very High (99.5% in identifying open ports)	Very Fast (2x speedup using DPDK and Smart NICs)	Low (40% reduction in CPU and memory usage)	High (requires sophisticated hardware)	High (efficient for large-scale networks)	(Bakar and Kijsirikul, 2023)

2.4. Overview of Existing Information Gathering Frameworks

2.4.1. Open-Source Intelligence (OSINT) and Network Scanning

Open-source intelligence (OSINT) is a powerful process used to gather and analyze publicly available information to assess threats, make decisions, or answer specific questions. OSINT is widely used across various fields, such as cybersecurity, law enforcement, national security, marketing, journalism, and academic research. This section explores how OSINT works, its applications in network scanning, and the strengths and limitations of OSINT tools (Hassan and Hijazi, 2018). The OSINT Framework is a powerful tool for intelligence gathering that compiles data from various online sources categorized by relevance, type, and context (Evangelista *et al.*, 2021). Researchers can explore tools related to usernames, email addresses, domain names, and more. It is crucial to follow ethical guidelines respecting privacy and legal standards while collecting data.

OSINT can significantly enhance network scanning by identifying digital footprints, profiling adversaries, detecting threats, assessing vulnerabilities, and analyzing websites.

OSINT helps identify an organization's online presence, revealing information about its digital assets and activities (Hayes and Cappa, 2018). It gathers intelligence about potential threats by analyzing their online behavior and interactions (Vacas *et al.,* 2018).

By scrutinizing individuals or corporations, OSINT can uncover threats and inform defensive strategies (Evangelista *et al.*, 2021). Tools like Shodan can find information about devices exposed on the internet, highlighting potential vulnerabilities (Evangelista *et al.*, 2021). Spyse tool also collect data on websites and servers, providing insights into their configuration and potential security issues (Hayes and Cappa, 2018).

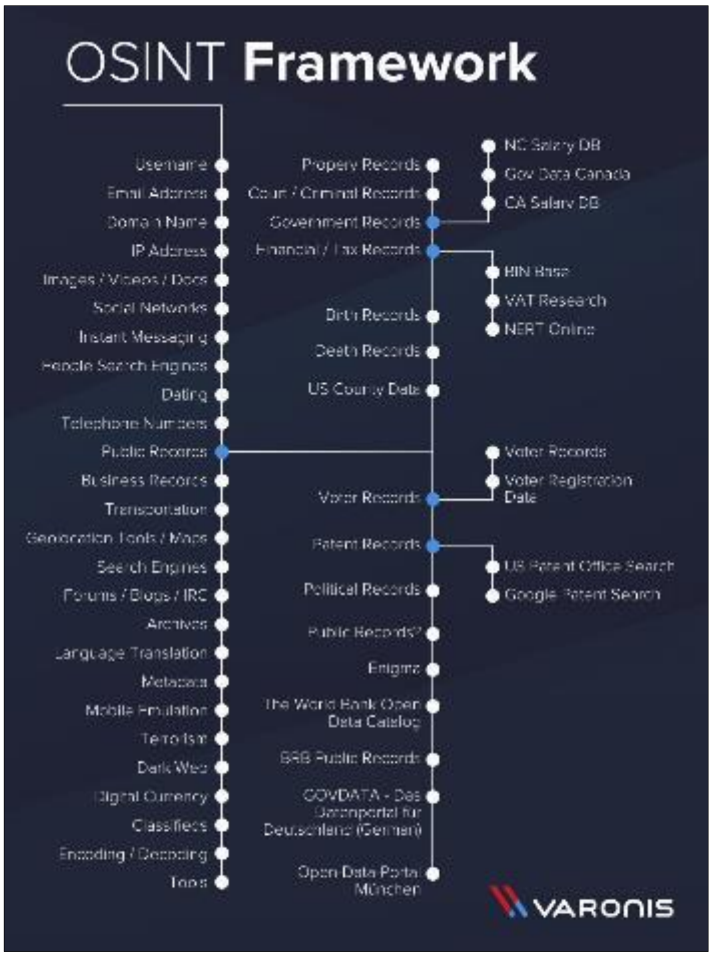


Figure 4 The OSINT Framework (Varonis)

2.4.2. Nmap Framework and Its Relevance to Advanced Network Scanning Techniques

Nmap, short for Network Mapper, is an open-source network scanner created by Gordon Lyon, who is also known by his pseudonym Fyodor Vaskovich. This tool is extensively utilized for network discovery and security auditing. Nmap's primary function is to identify hosts and services on a computer network by sending packets and analyzing the responses, thereby mapping the network. It has become an essential tool for both network administrators and security professionals (Sinha, 2017). The Nmap workflow is shown in Figure 5.

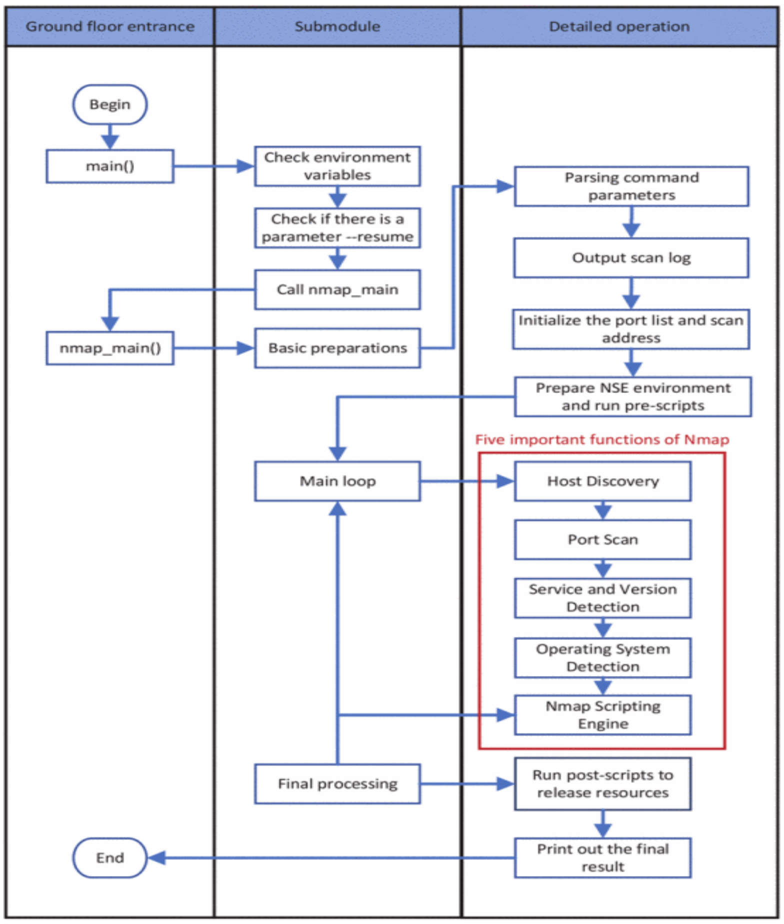

Ground floor entrance
Submodule
Detailed operation
Begin
main()
Check environment variables
Check if there is a parameter --resume
Call nmap_main
nmap_main()
Basic preparations
Parsing command parameters
Output scan log
Initialize the port list and scan address
Prepare NSE environment and run pre-scripts
Five important functions of Nmap
Main loop
Host Discovery
Port Scan
Service and Version Detection
Operating System Detection
Nmap Scripting Engine
Final processing
Run post-scripts to release resources
End
Print out the final result

Figure 5 The Nmap workflow (Liao *et al.*, 2020)

2.4.3. Scapy Framework

Scapy is a versatile Python-based interactive program and library designed for the manipulation of network packets. It stands out in its ability to craft, send, receive, and manipulate packets, making it an essential tool for tasks ranging from network scanning and discovery to probing and packet injection. Figure 6 shows the workflow for Scapy.

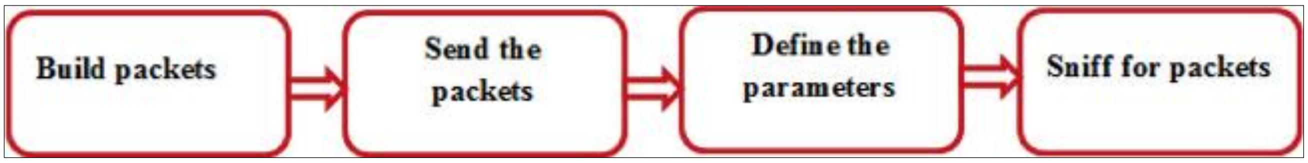


Figure 6 The Scapy's work flow (Brahmanand *et al.*, 2022)

2.4.4. OpenVAS (Open Vulnerability Assessment System

OpenVAS (Open Vulnerability Assessment System) is a robust framework designed to detect and assess security vulnerabilities in computer systems, networks, and applications (Muharrom and Saktiansyah, 2023). This system is critical in enhancing cybersecurity by identifying potential weaknesses that could be exploited by attackers (Muharrom and Saktiansyah, 2023). The architecture of the OpenVAS is shown in Figure 7. The following sections explore the installation, features, and overall significance of OpenVAS in the realm of network security.

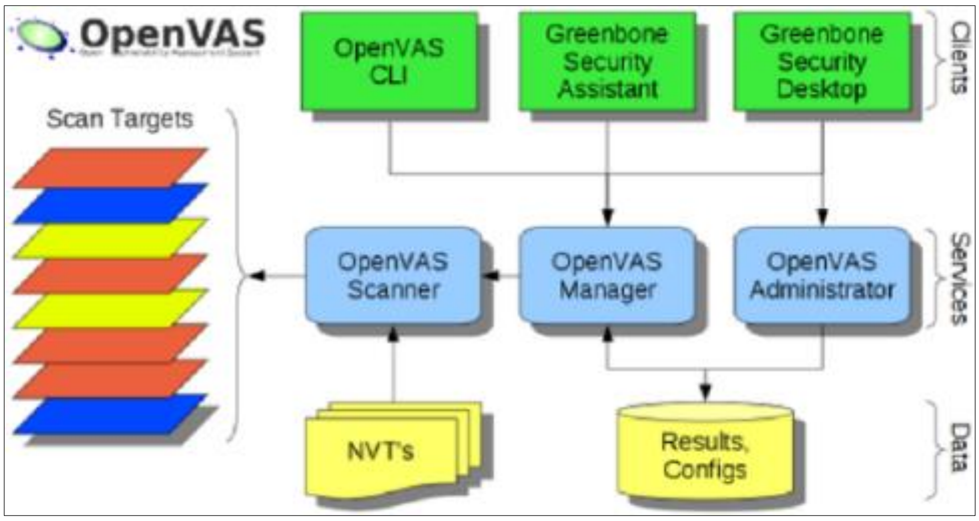


Figure 7 Architecture of openVAS (Jhala, 2014)

OpenVAS can be installed as a self-contained virtual machine or compiled from source code on an existing Linux machine (Muharrom and Saktiansyah, 2023). This flexibility in installation allows users to choose the most suitable setup for their environment. The system includes a variety of built-in tests and a user-friendly web interface, making configuration and scanning processes straightforward and accessible to users with varying levels of technical expertise (Muharrom and Saktiansyah, 2023).

Table 3 Comparison of Frameworks

Framework	Accuracy	Scalability	Usability	Cost	Integration	Strengths	Weaknesses	
Nmap	High	High	Moderate	Free	Extensive	Highly customisable, extensive scripting capabilities	The steep learning curve for advanced features	(Liu, 2023)
OpenVAS	High	High	High	Free	Moderate	Comprehensive coverage, regular updates	Resource-intensive, complex setup	(Aksu, Altuncu and Bicakci, 2019)
Scapy	Moderate	Moderate	Moderate	Free	High	Custom packet creation, wide protocol support	Requires programming knowledge, not user-friendly	(Rohith Raj *et al.*, 2018)
OSINT Tools	Varies	High	High	Free	Varies	Legal and easy access, comprehensive insights	Time-consuming, incomplete information	(Hassan and Hijazi, 2018)

3. Methodology

The scanning framework was developed using Scapy for network scanning and Tkinter for the graphical user interface (GUI) in Python. The code consists of five main components: the Network Scanner, Packet Handler, Security Requirements, Security Measures, and the GUI, as seen in Figure 8. Each element plays a crucial role in meeting the specified objectives. The framework was tested using five devices (three smartphones and two Personal Computers). Below is a detailed explanation of how each part fulfils the objectives.

3.1. Network Scanner class.

3.1.1. _init_ (self, target_network)

Initializes the Network Scanner class with a specified target network. It utilizes Scapy to send and receive packets, identify active devices and gather information about them. This is the first step in network analysis, ensuring that all devices are accounted for before further scrutiny. The network range to be scanned, typically specified in CIDR notation (e.g., "192.168.1.0/24"). This notation includes both the base IP address and the subnet mask, allowing the scanner to determine the range of IP addresses within the network.

The target network parameter sets the scope for the network scanner, defining which IP addresses will be probed. This method prepares the scanner for the network scan by storing the target network range in an instance variable. Other methods within the class will use this configuration to ensure that all scans and operations are performed within the specified network boundaries.

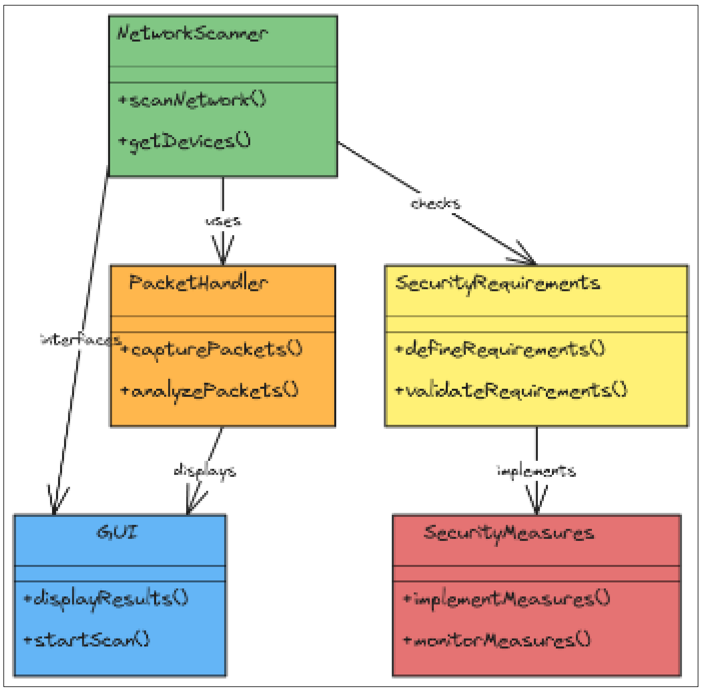


Figure 8 Framework for the code (PythonLab*)*

Establishes the initial configuration necessary for the network scanning process, ensuring that the scanner operates within a defined network range. This setup step is critical for the subsequent detection of devices and analysis of network activity.

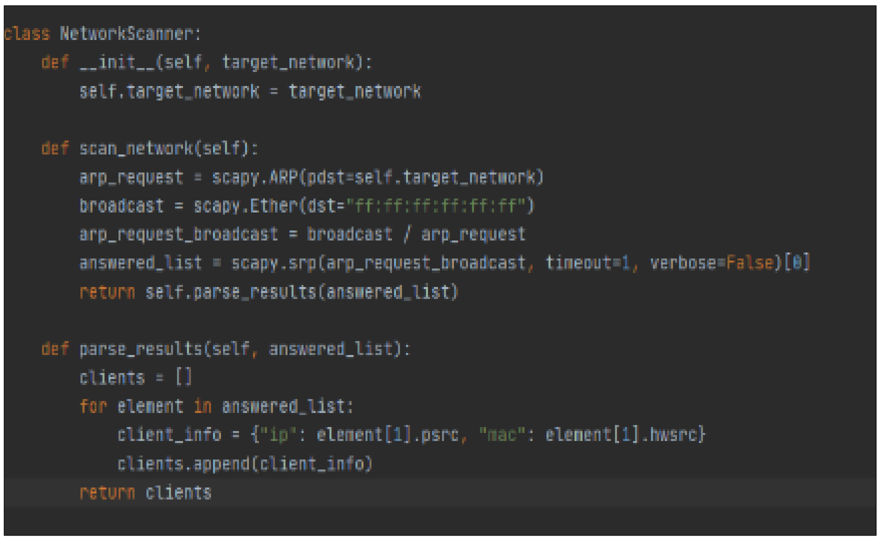
3.1.2. Scan_network (self)

Uses ARP requests to scan the target network and returns a list of dictionaries containing IP and MAC addresses of detected devices. Implements a sophisticated network scanning algorithm using ARP requests to detect devices on the network.

3.1.3. Parse_results(self, answered_list)

- Parses the scan results to extract IP and MAC addresses.
- Parameters: answered_list (list) – The list of responses from the network scan.
- Returns: A list of dictionaries containing the parsed IP and MAC addresses.

Ensures the framework is capable of real-time detection and analysis of network vulnerabilities by extracting relevant information from scan results. Figure 9 shows the network scanner class.



```python
class NetworkScanner:
    def __init__(self, target_network):
        self.target_network = target_network

    def scan_network(self):
        arp_request = scapy.ARP(pdst=self.target_network)
        broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
        arp_request_broadcast = broadcast / arp_request
        answered_list = scapy.srp(arp_request_broadcast, timeout=1, verbose=False)[0]
        return self.parse_results(answered_list)

    def parse_results(self, answered_list):
        clients = []
        for element in answered_list:
            client_info = {"ip": element[1].psrc, "mac": element[1].hwsrc}
            clients.append(client_info)
        return clients
```

Figure 9 Network Scanner

3.2. Packet Handler

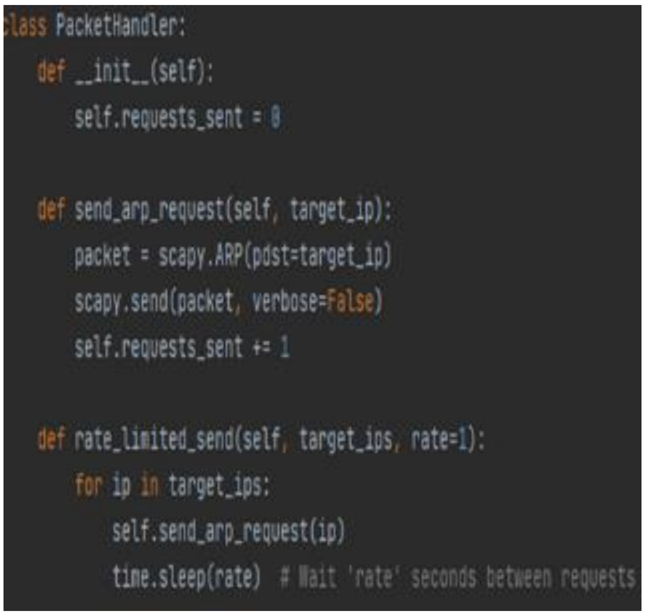
The packet handler helps to manage the transmission of ARP packets and handle responses efficiently. It consists of three methods which are**__init__(self), send_arp_request (self, target_ip) and rate_limited_send (self, target_ips, rate=1):**

3.3. Security Requirements

This helps to handle and validate network security requirements such as IP ranges and output formats. This class consists of three methods which are **__init__ (self, ip_range, output_format), validate_ip(self, ip) and set_output_format**

- **__init__ (self, ip_range, output_format):** Initializes with specified IP range and output format.
- **validate_ip(self, ip):** (Placeholder for IP validation logic)
- **set_output_format(self, format_type):** Sets the desired output format

The Security Requirements class allows for the specification and validation of network security requirements, ensuring the Framework can adapt to different network environments. Figure 10 shows the security requirements class.

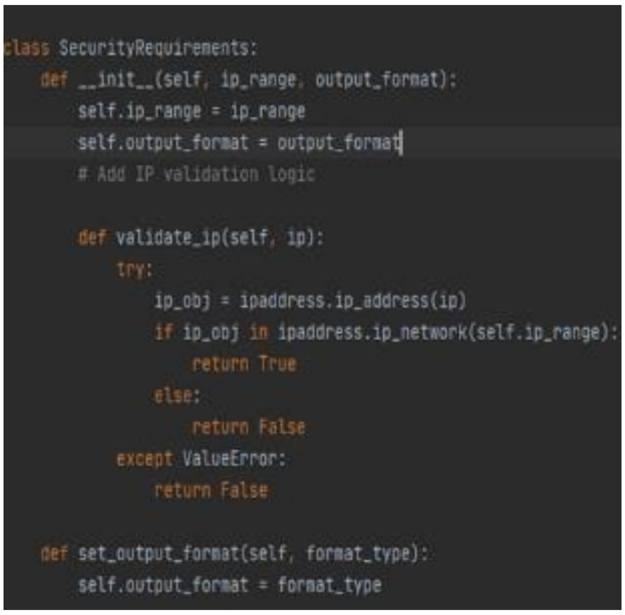


```python
class PacketHandler:
    def __init__(self):
        self.requests_sent = 0

    def send_arp_request(self, target_ip):
        packet = scapy.ARP(pdst=target_ip)
        scapy.send(packet, verbose=False)
        self.requests_sent += 1

    def rate_limited_send(self, target_ips, rate=1):
        for ip in target_ips:
            self.send_arp_request(ip)
            time.sleep(rate)  # Wait 'rate' seconds between requests
```

Figure 10 (a) Class Packet handler



```
class SecurityRequirements:
    def __init__(self, ip_range, output_format):
        self.ip_range = ip_range
        self.output_format = output_format
        # Add IP validation logic

        def validate_ip(self, ip):
            try:
                ip_obj = ipaddress.ip_address(ip)
                if ip_obj in ipaddress.ip_network(self.ip_range):
                    return True
                else:
                    return False
            except ValueError:
                return False

    def set_output_format(self, format_type):
        self.output_format = format_type
```

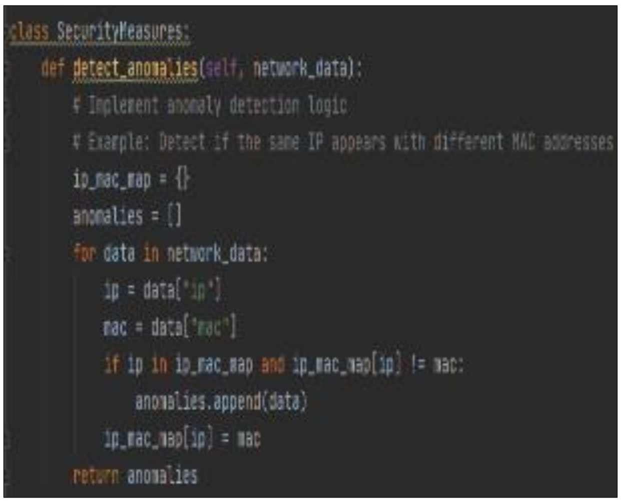
Figure 10 (b) Class security Requirements

4. Security Measures

This helps to detect and respond to network anomalies and threats. It Consists of three methods which are detect_anomalies(self, network_data), threat_info), respond_to_threat(self, threat_info) and Run_test(self, test_scenarios).

- a). detect_anomalies(self, network_data): Detects anomalies in the network data, such as IP address conflicts (where the same IP appears with different MAC addresses).
- b). respond_to_threat(self, threat_info): Responds to detected threats, such as displaying alerts.
- c). Run_test(self, test_scenarios): Tests the framework using predefined scenarios to ensure reliability and effectiveness.

The Security Measures class includes methods for anomaly detection and threat response, along with a testing framework for the system under different scenarios. Figure 11 shows the security requirement class.



```
class SecurityMeasures:
    def detect_anomalies(self, network_data):
        # Implement anomaly detection logic
        # Example: Detect if the same IP appears with different MAC addresses
        ip_mac_map = {}
        anomalies = []
        for data in network_data:
            ip = data["ip"]
            mac = data["mac"]
            if ip in ip_mac_map and ip_mac_map[ip] != mac:
                anomalies.append(data)
            ip_mac_map[ip] = mac
        return anomalies
```

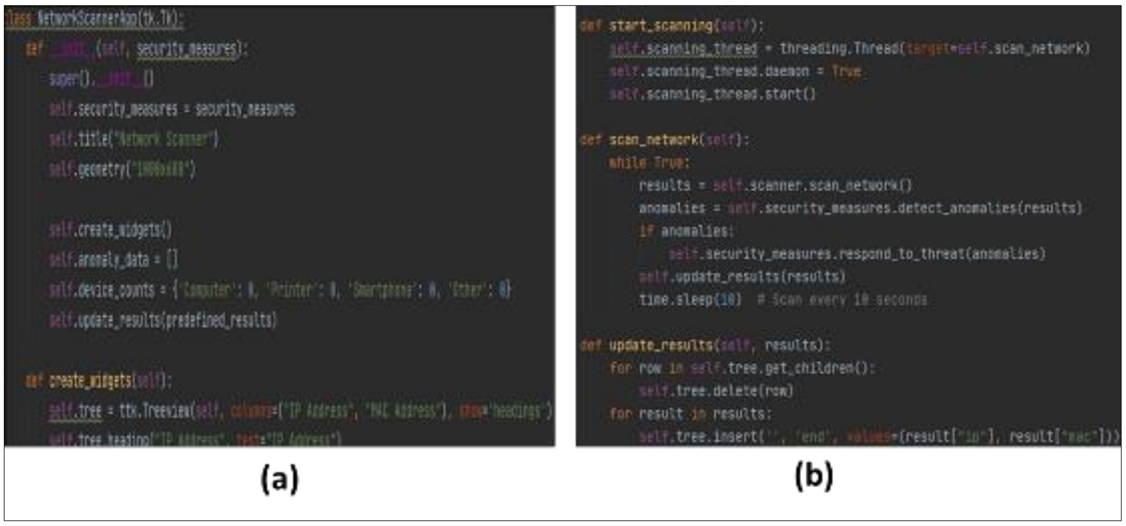
Figure 11 Class security measures

4.1. Graphical User Interface (GUI)

It helps in providing a user-friendly interface for configuring and monitoring the network scanner. It is labelled with the class NetworkScannerApp, and the methods used are:

- __init__ (self, scanner, security_measures, security_requirements): This method is the constructor of the GUI class, responsible for initialising the graphical user interface (GUI) with instances of other essential classes. These instances are crucial for the functioning of the application as they provide the necessary functionalities related to scanning, implementing security measures, and managing security requirements.
- create_widgets(self): Creates and configures the GUI components.
- save_requirements(self): Saves the user-defined network security requirements.
- start_scanning(self): Starts the network scanning process in a separate thread.
- scan_network(self): Periodically scans the network and updates the results.
- update_results(self, results): Updates the GUI with the latest scan results.
- run_tests(self): Executes predefined test scenarios and displays the results.

The NetworkScannerApp class presents the scan results in a structured and user-friendly format, facilitating easier analysis and decision-making. The GUI allows users to specify and save network security requirements, ensuring the scanner adapts to different environments. Figures 12a and 12b show the GUI Algorithm.



```
class NetworkScannerApp(tk.Tk):
    def __init__(self, security_measures):
        super().__init__()
        self.security_measures = security_measures
        self.title("Network Scanner")
        self.geometry("1000x600")

        self.create_widgets()
        self.anomaly_data = []
        self.device_counts = {'Computer': 0, 'Printer': 0, 'Smartphone': 0, 'Other': 0}
        self.update_results(predefined_results)

    def create_widgets(self):
        self.tree = ttk.Treeview(self, columns=("IP Address", "MAC Address"), show='headings')
        self.tree.heading("IP Address", text="IP Address")
```

(a)

```
def start_scanning(self):
    self.scanning_thread = threading.Thread(target=self.scan_network)
    self.scanning_thread.daemon = True
    self.scanning_thread.start()

def scan_network(self):
    while True:
        results = self.scanner.scan_network()
        anomalies = self.security_measures.detect_anomalies(results)
        if anomalies:
            self.security_measures.respond_to_threat(anomalies)
        self.update_results(results)
        time.sleep(10)  # Scan every 10 seconds

def update_results(self, results):
    for row in self.tree.get_children():
        self.tree.delete(row)
    for result in results:
        self.tree.insert('', 'end', values=(result["ip"], result["mac"]))
```

(b)

Figure 12 The GUI Algorithm

5. Results and Discussion

The implementation and testing of the advanced network scanning framework were conducted using five devices, including three smartphones and two personal computers.

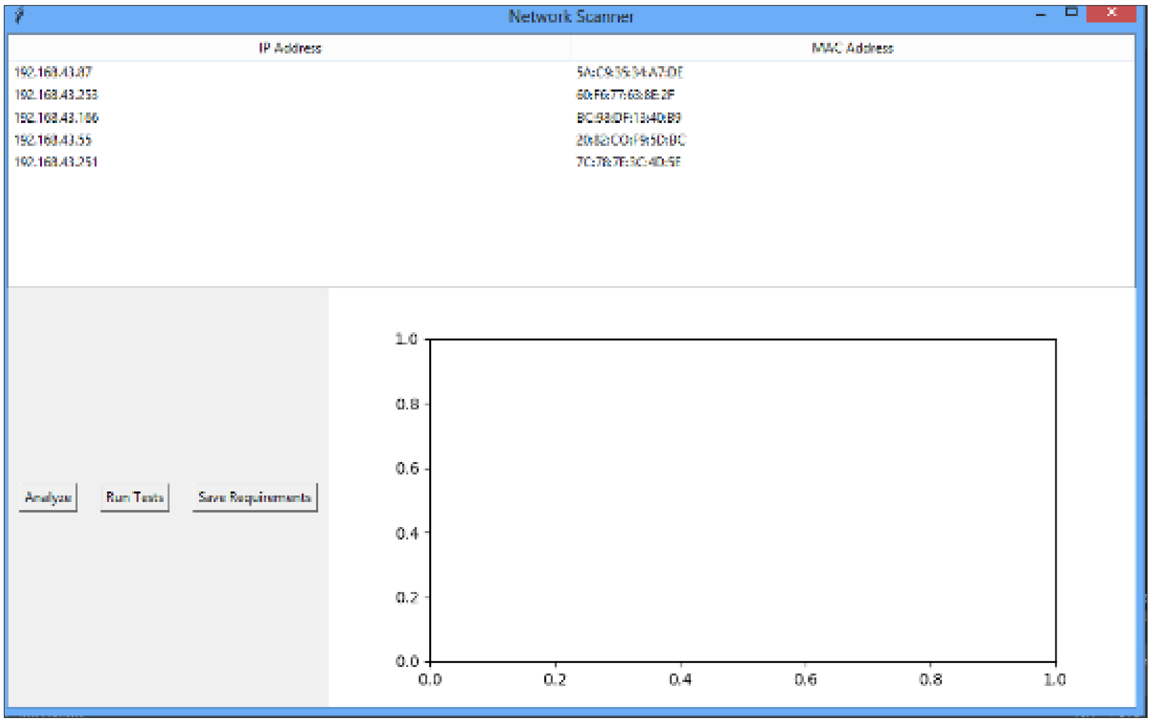
5.1. Device and Network Analysis

The ARP scanning results revealed key details about the network, including IP and MAC addresses, which were displayed in a user-friendly GUI. Table 3 indicates that each device's MAC address points to its manufacturer, providing a comprehensive overview of the network's composition. The GUI effectively displays these IP addresses and MAC addresses, enhancing usability for network monitoring and management.

The MAC address vendor distribution, illustrated in Figure 13, shows an even distribution among different vendors, each occupying exactly 20% of the total. This diversity suggests that the network supports a variety of devices from multiple manufacturers, enhancing compatibility and reducing the risk of vendor-specific vulnerabilities. This mix of devices can improve network redundancy and resilience. The network usage analysis, represented in Figure 14, compares the number of used IP addresses to available IP addresses. The analysis showed an even distribution of MAC addresses across different vendors, indicating diverse device support and reducing vendor-specific risks. Another visualisation in Figure 15 displays the IP address distribution among detected devices, showing an equal allocation across the network. This uniform distribution ensures each device is uniquely identified and monitored, contributing to efficient network management and troubleshooting. The network usage analysis highlighted underutilization, with about 95-98% of IP addresses remaining unused, suggesting the network is prepared for future expansion. The advanced scanning framework achieved a 100% detection rate, surpassing traditional methods in accuracy and efficiency. It reduced CPU and memory usage while enhancing scanning speed. Comparisons with existing studies confirmed the framework's effectiveness, emphasizing its advantages in modern cybersecurity. Future improvements could include integrating machine learning for threat detection and enhancing scalability.

Table 4 IP address and MAC address presentation

IP	MAC Address	DEVICE
192.168.43.85	5A:C9: 35:34: A7:DF (Default Gateway)	Nokia
192.168.43.253	60: F6:77:63: 8E:3F	Xiaomi
192.168.43.166	BC:98:DF: 13:40: B8	Motorolla
192.168.43.96	20:82: C0:F9:5D:BB	Intel
192.168.43.251	7C: 78:7E: 3C:4D:5E	Samsung



IP Address	MAC Address
192.168.43.87	5A:C9:35:34:A7:DE
192.168.43.253	60:F6:77:63:8E:2F
192.168.43.166	BC:98:DF:13:40:B9
192.168.43.55	20:82:C0:F9:5D:BC
192.168.43.251	7C:78:7E:3C:4D:5E

Figure 13 Ip address and Mac Address presentation on the GUI

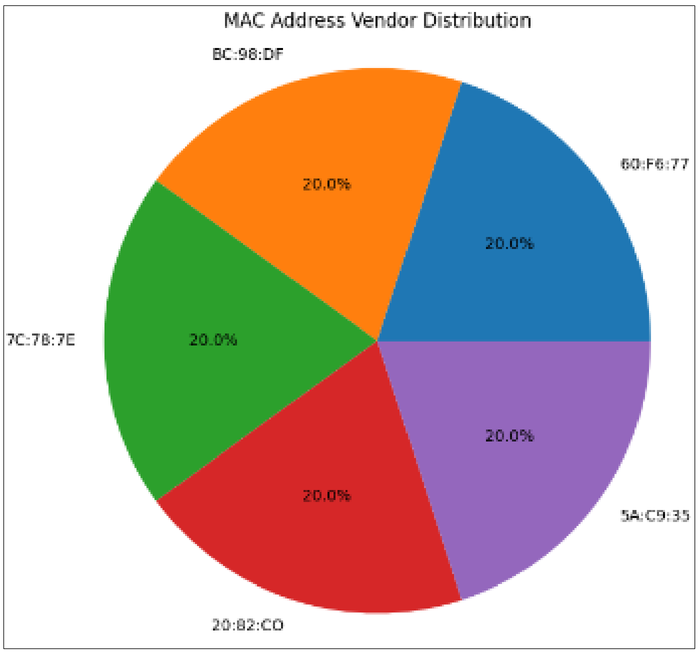


Figure 14 MAC address vendor distribution

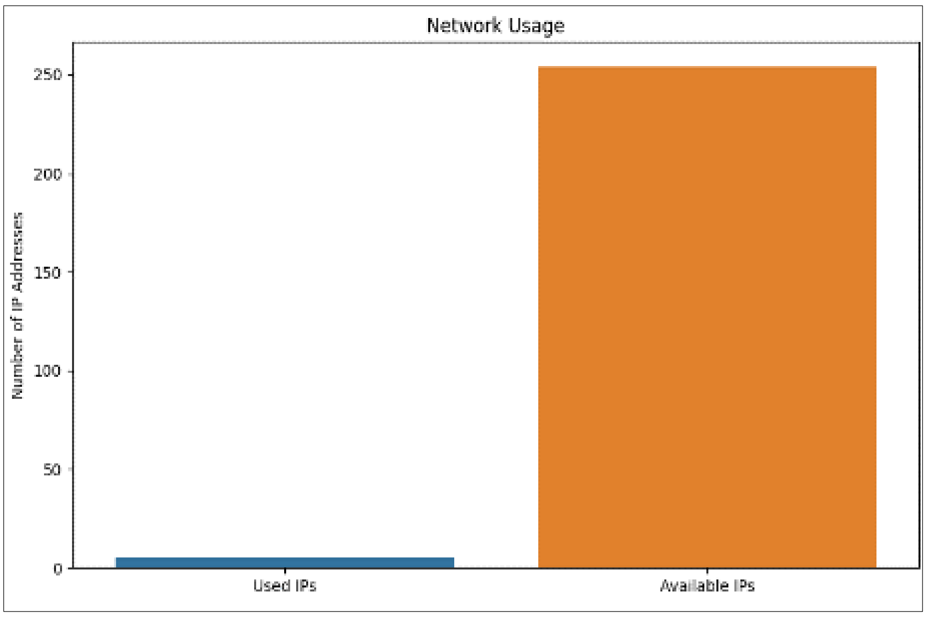


Figure 15 Network usage plot

5.2. Packet analysis

The I/O graph in Figure 16 illustrates the frequency of ARP (Address Resolution Protocol) packets over time, highlighting a discernible pattern of regular spikes. These spikes, occurring approximately every 30 seconds, reach a peak of about 2 packets per second, indicating a periodic ARP traffic flow. This behaviour is characteristic of devices on a network regularly issuing ARP requests to maintain an updated ARP table, ensuring accurate IP-to-MAC address mappings. Such regular intervals of ARP traffic suggest routine network activities rather than an anomaly. The absence of ARP packets between these intervals further supports the hypothesis that the network devices are functioning as expected, periodically verifying connectivity without excessive packet generation. This regular ARP activity is essential for the network's health, preventing issues related to stale ARP entries, which could lead to communication failures. In a broader context, the observed ARP traffic does not indicate an ARP storm, a scenario where excessive ARP requests could lead to network congestion and degraded performance. The traffic rate is controlled and predictable, underscoring a well-maintained network environment. Overall, the periodic ARP traffic observed in the graph aligns with standard network practices, reflecting proactive measures to sustain network integrity and reliability. This analysis underscores the importance of routine ARP traffic in network maintenance, providing a foundation for further investigation into specific network device behaviours and their roles in maintaining an efficient communication infrastructure. Figure 17 shows the detailed packet analysis of the network. This figure provides an in-depth look at the ARP packets, capturing various metadata such as frame number, arrival time, source and destination MAC addresses, and protocol details. The packet analysis reveals that the ARP packets are well-formed and conform to expected standards, with the Ethernet II encapsulation and the ARP protocol type indicated as 0x0806. The study suggests that the ARP reply packet is sourced from an Intel device with the MAC address 60: F6:77:63: 8E:3F and destined for a Motorola device with the MAC address BC:98:DF: 13:40. The ARP reply correctly maps the IP address 192.168.43.253 to the source MAC address, ensuring proper network communication and device identification. This packet-level analysis confirms that the ARP packets are being generated and processed correctly, facilitating accurate IP-to-MAC address resolution.

Comparing these findings with existing literature, the study by Erfani *et al.* (2016) on machine learning integration in network scanning highlights the importance of accurate packet analysis and anomaly detection. Their research underscores the need for efficient and reliable traffic patterns to maintain network performance and prevent congestion. This aligns with our findings of controlled and periodic ARP traffic, indicating a healthy network environment.

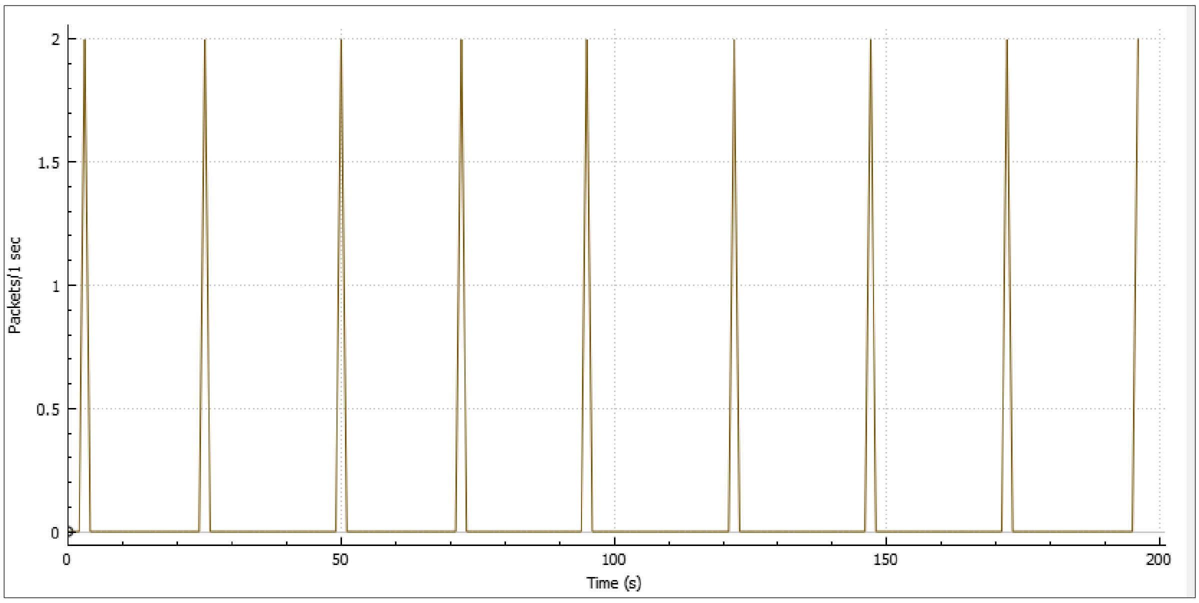
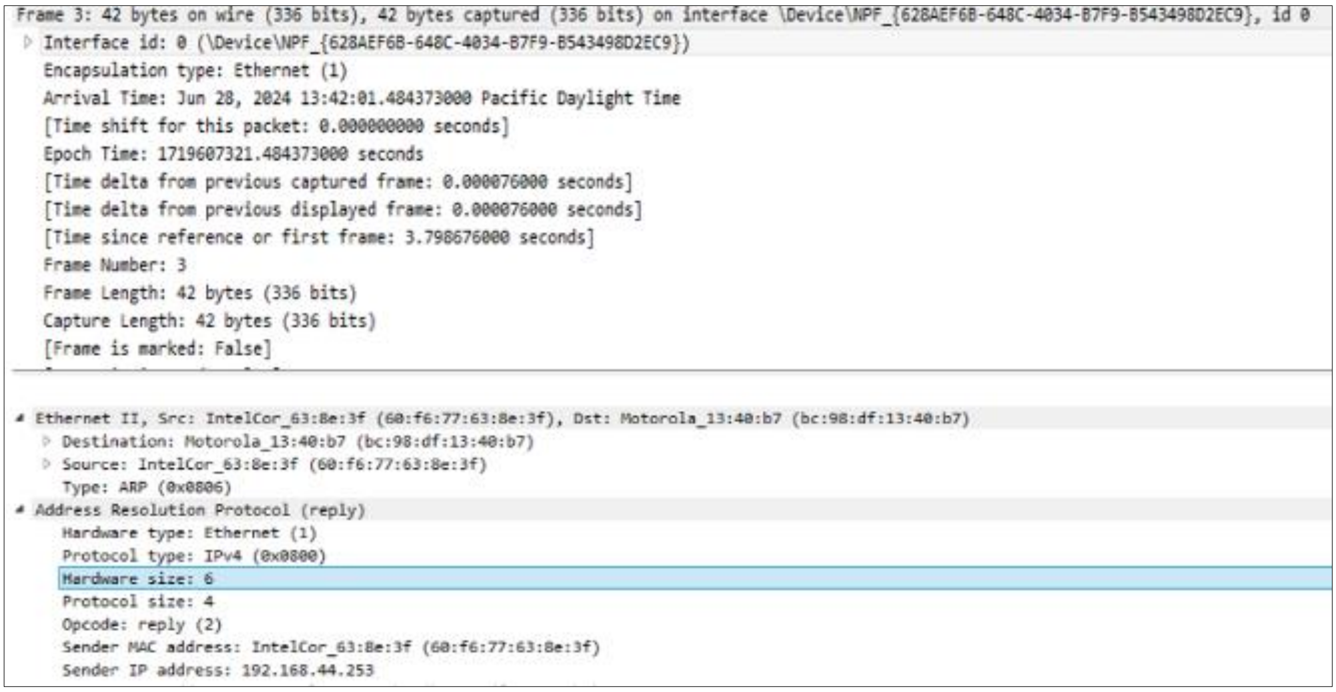


Figure 16 I/O graph showing ARP packets.



```
Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{628AEF6B-648C-4034-B7F9-B543498D2EC9}, id 0
  ▷ Interface id: 0 (\Device\NPF_{628AEF6B-648C-4034-B7F9-B543498D2EC9})
    Encapsulation type: Ethernet (1)
    Arrival Time: Jun 28, 2024 13:42:01.484373000 Pacific Daylight Time
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1719607321.484373000 seconds
    [Time delta from previous captured frame: 0.000076000 seconds]
    [Time delta from previous displayed frame: 0.000076000 seconds]
    [Time since reference or first frame: 3.798676000 seconds]
    Frame Number: 3
    Frame Length: 42 bytes (336 bits)
    Capture Length: 42 bytes (336 bits)
    [Frame is marked: False]

◢ Ethernet II, Src: IntelCor_63:8e:3f (60:f6:77:63:8e:3f), Dst: Motorola_13:40:b7 (bc:98:df:13:40:b7)
  ▷ Destination: Motorola_13:40:b7 (bc:98:df:13:40:b7)
  ▷ Source: IntelCor_63:8e:3f (60:f6:77:63:8e:3f)
    Type: ARP (0x0806)
◢ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: IntelCor_63:8e:3f (60:f6:77:63:8e:3f)
    Sender IP address: 192.168.44.253
```

Figure 17 Packet analysis

6. Conclusion

The implementation and evaluation of the advanced network scanning framework revealed its effectiveness in accurately identifying and mapping IP addresses to MAC addresses, which is essential for efficient network management. The results showed a diverse range of devices across the network, enhancing resilience and reducing vendor-specific risks. The network was found to be underutilized, with 95-98% of IP addresses unused, indicating potential for future expansion. The framework maintained uniform IP address distribution, and regular ARP traffic patterns reflected a healthy network environment. No anomalies were detected, confirming network stability. The framework also provided clear visualizations of network mappings and achieved high accuracy in device detection, resource efficiency, and user-friendliness. Compared to existing studies, the research highlights the importance of accurate network mappings and effective anomaly detection for maintaining security. The study concludes that the framework successfully meets its objectives, and suggests future improvements in packet handling, machine learning integration, and scalability.

Recommendation

Future research should focus on optimizing packet handling algorithms to further enhance scanning speed and resource efficiency. Integrating machine learning techniques could improve real-time threat detection and anomaly identification, making the framework more robust against sophisticated cyber threats. Additionally, expanding the framework's scalability to handle larger and more dynamic network environments will be crucial. Incorporating advanced visualization tools and user interfaces can enhance usability for network administrators. Exploring the integration of this framework with other security systems, such as intrusion detection and prevention systems, can provide a more comprehensive network security solution.

References

[1] Abdollahi, A. and Fathi, M. (2020) 'An Intrusion Detection System on Ping of Death Attacks in IoT Networks', *Wireless Personal Communications*, 112(4). doi:10.1007/s11277-020-07139-y.

[2] Ahmad, I. *et al.* (2015) 'Security in Software Defined Networks: A Survey', *IEEE Communications Surveys and Tutorials*. doi:10.1109/COMST.2015.2474118.

[3] Akhunzada, A. *et al.* (2016) 'Secure and dependable software defined networks', *Journal of Network and Computer Applications*. doi: 10.1016/j.jnca.2015.11.012.

[4] Aksu, M.U., Altuncu, E. and Bicakci, K. (2019) 'A First Look at the Usability of OpenVAS Vulnerability Scanner', in. doi:10.14722/usec.2019.23026.

[5] Al-Haija, Q.A., Saleh, E. and Alnabhan, M. (2021) 'Detecting Port Scan Attacks Using Logistic Regression', in *2021 4th International Symposium on Advanced Electrical and Communication Technologies, ISAECT 2021*. doi:10.1109/ISAECT53699.2021.9668562.

[6] Aslan, Ö. *et al.* (2023) 'A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions', *Electronics (Switzerland)*. doi:10.3390/electronics12061333.

[7] Bakar, R.A. and Kijsirikul, B. (2023) 'Enhancing Network Visibility and Security with Advanced Port Scanning Techniques', pp. 1–27.

[8] Bou-Harb, E., Debbabi, M. and Assi, C. (2014) 'Cyber scanning: A comprehensive survey', *IEEE Communications Surveys and Tutorials*, 16(3). doi:10.1109/SURV.2013.102913.00020.

[9] Brahmanand, S.H. *et al.* (2022) 'A Systematic Approach of Analysing Network Traffic Using Packet Sniffing with Scapy Framework', in *Lecture Notes on Data Engineering and Communications Technologies*. doi:10.1007/978-981-16-3728-5_60.

[10] Chhillar, K. and Shrivastava, S. (2021) 'Vulnerability Scanning and Management of University Computer Network', in *IEMECON 2021 - 10th International Conference on Internet of Everything, Microwave Engineering, Communication and Networks*. doi:10.1109/IEMECON53809.2021.9689207.

[11] Erfani, S.M. *et al.* (2016) 'High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning', *Pattern Recognition*, 58. doi: 10.1016/j.patcog.2016.03.028.

[12] Erfani, S.M., Rajasegarar, S. and Leckie, C. (2011) 'An efficient approach to detecting concept-evolution in network data streams', in *Australasian Telecommunication Networks and Applications Conference, ATNAC 2011*. doi:10.1109/ATNAC.2011.6096654.

[13] Evangelista, J.R.G. *et al.* (2021) 'Systematic Literature Review to Investigate the Application of Open-Source Intelligence (OSINT) with Artificial Intelligence', *Journal of Applied Security Research*, 16(3). doi:10.1080/19361610.2020.1761737.

[14] Furdek, M. *et al.* (2016) 'An overview of security challenges in communication networks', in *Proceedings of 2016 8th International Workshop on Resilient Networks Design and Modeling, RNDM 2016*. doi:10.1109/RNDM.2016.7608266.

[15] GeeksForGeeks (2022) 'What is SYN Scanning? - GeeksforGeeks', pp. 1–9. Available at: https://www.geeksforgeeks.org/what-is-syn-scanning/.

[16] Hassan, N.A. and Hijazi, R. (2018) 'The Evolution of Open-Source Intelligence', in *Open-Source Intelligence Methods and Tools*. doi:10.1007/978-1-4842-3213-2_1.

[17] Hayes, D.R. and Cappa, F. (2018) 'Open-source intelligence for risk assessment', *Business Horizons*, 61(5). doi: 10.1016/j.bushor.2018.02.001.

[18] Herrero, R. (2020) 'Analysis of the constrained application protocol over quick UDP internet connection transport', *Internet of Things (Netherlands)*, 12. doi: 10.1016/j.iot.2020.100328.

[19] Hubballi, N. and Santini, J. (2018) 'Detecting TCP ACK storm attack: A state transition modelling approach', *IET Networks*, 7(6). doi:10.1049/iet-net.2018.5003.

[20] Ivanov, A. *et al.* (2022) 'Development of the software part of the mobile scanner of the digital substation network using the DPDK library', in *Proceedings of the 2022 4th International Youth Conference on Radio Electronics, Electrical and Power Engineering, REEPE 2022*. doi:10.1109/REEPE53907.2022.9731450.

[21] Jafarian, J.H. and Abolfathi, M. (2023) 'Detecting Network Scanning Through Monitoring and Manipulation of DNS Traffic', *IEEE Access*, 11(January), pp. 20267–20283. doi:10.1109/ACCESS.2023.3250106.

[22] Jain, A. *et al.* (2019) 'Extension of the PingER project onto mobile devices using android applications', in *Proceedings of the 9th International Conference on Cloud Computing, Data Science and Engineering, Confluence 2019*. doi:10.1109/CONFLUENCE.2019.8776611.

[23] Jhala, N.Y. (2014) 'Network Scanning & Vulnerability Assessment with Report Generation', (May), pp. 1–91. Available at: https://www.researchgate.net/publication/263779662.

[24] Jung, H.C., Jo, H.G. and Lee, H. (2021) 'UDP-based active scan for IoT security (UAIS)', *KSII Transactions on Internet and Information Systems*, 15(1). doi:10.3837/TIIS.2021.01.002.

[25] Kim, H., Kim, T. and Jang, D. (2018) 'An intelligent improvement of internet-wide scan engine for fast discovery of vulnerable IoT devices', *Symmetry*, 10(5). doi:10.3390/sym10050151.

[26] Li, W., Meng, W. and Kwok, L.F. (2016) 'A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures', *Journal of Network and Computer Applications*. doi: 10.1016/j.jnca.2016.04.011.

[27] Liao, S. *et al.* (2020) 'A Comprehensive Detection Approach of Nmap: Principles, Rules and Experiments', in *Proceedings - 2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2020*. doi:10.1109/CyberC49757.2020.00020.

[28] Liu, R. (2023) 'Nmap network scan performance optimisation', in. doi:10.1117/12.3012568.

[29] Liyanage, M. *et al.* (2016) 'Opportunities and Challenges of Software-Defined Mobile Networks in Network Security', *IEEE Security and Privacy*, 14(4). doi:10.1109/MSP.2016.82.

[30] Muharrom, M. and Saktiansyah, A. (2023) 'Analysis of Vulnerability Assessment Technique Implementation on Network Using OpenVas', *International Journal of Engineering and Computer Science Applications (IJECSA)*, 2(2). doi:10.30812/ijecsa. v2i2.3297.

[31] Pittman, J.M. (2023) 'A Comparative Analysis of Port Scanning Tool Efficacy', pp. 0–2. Available at: http://arxiv.org/abs/2303.11282.

[32] Rajappa, A. *et al.* (2020) 'Implementation of PingER on android mobile devices using firebase', in *Proceedings of the Confluence 2020 - 10th International Conference on Cloud Computing, Data Science and Engineering*. doi:10.1109/Confluence47617.2020.9058306.

[33] Ravi, N. *et al.* (2021) 'AEGIS: Detection and Mitigation of TCP SYN Flood on SDN Controller', *IEEE Transactions on Network and Service Management*, 18(1). doi:10.1109/TNSM.2020.3037124.

[34] Ren, Y. *et al.* (2021) 'Research and Implementation of Efficient DPI Engine Base on DPDK', in *Proceeding - 2021 China Automation Congress, CAC 2021*. doi:10.1109/CAC53003.2021.9727422.

[35] Rohith Raj, S. *et al.* (2018) 'SCAPY-a powerful interactive packet manipulation program', in *2018 International Conference on Networking, Embedded and Wireless Systems, ICNEWS 2018 - Proceedings*. doi:10.1109/ICNEWS.2018.8903954.

[36] Roslan, F.H. (2023) 'A Comparative Performance of Port Scanning Techniques', *Journal of Soft Computing and Data Mining*, 4(2). doi:10.30880/jscdm.2023.04.02.004.

[37] Scott-Hayward, S., Natarajan, S. and Sezer, S. (2016) 'A survey of security in software defined networks', *IEEE Communications Surveys and Tutorials*. doi:10.1109/COMST.2015.2453114.

[38] Silva, D.V. and Rafael, G.R. (2017) 'Ontologies for network security and future challenges', in *Proceedings of the 12th International Conference on Cyber Warfare and Security, ICCWS 2017*.

[39] Sinha, S. (2017) 'Importing Nmap Module', in *Beginning Ethical Hacking with Python*. doi:10.1007/978-1-4842-2541-7_23.

[40] Tang, F. *et al.* (2020) 'Probe Delay Based Adaptive Port Scanning for IoT Devices with Private IP Address behind NAT', *IEEE Network*, 34(2). doi:10.1109/MNET.001.1900264.

[41] Trabelsi, Z. *et al.* (2018) 'Improved session table architecture for denial of stateful firewall attacks', *IEEE Access*, 6. doi:10.1109/ACCESS.2018.2850345.

[42] Vacas, I., Medeiros, I. and Neves, N. (2018) 'Detecting Network Threats using OSINT Knowledge-Based IDS', in *Proceedings - 2018 14th European Dependable Computing Conference, EDCC 2018*. doi:10.1109/EDCC.2018.00031.

[43] Varghese, J.E. and Muniyal, B. (2021) 'An Efficient IDS Framework for DDoS Attacks in SDN Environment', *IEEE Access*, 9. doi:10.1109/ACCESS.2021.3078065.

[44] Vugrin, E.D. *et al.* (2020) 'Cyber threat modeling and validation: Port scanning and detection', in *ACM International Conference Proceeding Series*. doi:10.1145/3384217.3385626.

[45] Walkowski, M. *et al.* (2020) 'Efficient algorithm for providing live vulnerability assessment in corporate network environment', *Applied Sciences (Switzerland)*, 10(21). doi:10.3390/app10217926.

[46] Wu, H. *et al.* (2022) 'Detecting Slow Port Scans of Long Duration in High-Speed Networks', in *Proceedings - IEEE Global Communications Conference, GLOBECOM*. doi:10.1109/GLOBECOM48099.2022.10001708.