CyberSecurity (CSL6010)

Lab Assignment 6

Soumen Kumar
Roll No-B22ES006

Vulnerability Exploitation and Patching - Lab Report

I conducted this lab to understand how to find and fix security vulnerabilities using different tools in a **Kali Linux environment**. The goal was to scan for weaknesses, exploit them to see their impact, and then apply patches to secure the system.

To achieve this, I used three main tools:

- **Nmap** – To scan networks and find open ports and services.
- **Nessus** – To perform a detailed vulnerability assessment and identify security flaws.
- **ZAProxy (ZAP)** – To test web applications for security issues like **SQL Injection** and **Cross-Site Scripting (XSS)**.

For this lab, I set up **two Kali Linux virtual machines**—one as the **attacker (scanning system)** and the other as the **victim (target system)**. The victim machine was configured with known vulnerabilities so that I could scan, exploit, and patch them.

This report documents the entire process, including **how I performed the scans, what vulnerabilities I found, how I attempted to exploit them, and how I fixed them**.

1 Task1:Vulnerability Scanning using nmap

Secure Shell (SSH) is a widely used protocol for secure remote login. However, misconfigurations and outdated versions may introduce security vulnerabilities. This report details the process of scanning for weak SSH authentication methods and vulnerable configurations using Nmap.

1.1 Lab Setup

- **Attacker Machine:** Kali Linux VM (IP: 192.168.X.Y)
- **Victim Machine:** Another Kali Linux VM running OpenSSH with intentionally weak settings (IP: 192.168.X.Z)
- **Networking Mode:** Host-Only or Bridged

1.2 Vulnerability Setup on the Victim

To intentionally introduce SSH vulnerabilities, I modified the SSH configuration on the victim machine:

```
sudo nano /etc/ssh/sshd_config
```

Listing 1: Editing SSH Configuration

Changes made:

- PermitRootLogin **yes**
- PasswordAuthentication **yes**

Then, restarted the SSH service:

```
sudo systemctl restart ssh
```

1.3 Scanning for SSH Vulnerabilities

1.3.1 Host Discovery

Before scanning for vulnerabilities, the first step is to check if the victim machine is online.

```
nmap -sn 10.40.0.244
```

Listing 2: Checking if Target is Reachable

This command performs a simple ping scan to determine whether the victim machine is reachable.

1.3.2 Detecting Open Ports and Services

The next step is to identify open SSH ports and service versions.

```
nmap -p 22 -sV 192.168.X.Z
```

Listing 3: Scanning for Open SSH Ports

1.3.3 Checking for Weak Authentication Methods

To determine if weak authentication methods are enabled:

```
nmap --script ssh-auth-methods -p 22 192.168.X.Z
```

Listing 4: Checking SSH Authentication Methods

If password authentication is enabled, it indicates a weak configuration.

1.3.4 Enumerating SSH Users

To check for username enumeration vulnerabilities:

```
nmap --script ssh-user-enum -p 22 192.168.X.Z
```

Listing 5: Testing SSH Username Enumeration

1.3.5 Testing for Weak Credentials

Simulating a brute-force attack to check for weak SSH passwords:

```
nmap --script ssh-brute -p 22 192.168.X.Z
```

Listing 6: Brute-forcing SSH Credentials

1.4 Scan Findings

1.4.1 Open Ports and Services

The Nmap service scan revealed the following open ports and running services:

Port	State	Service	Version
22/tcp	Open	SSH	OpenSSH 9.4p1 (Debian)
80/tcp	Open	HTTP	Apache httpd 2.4.58 (Debian)

Table 1: Open Ports and Services Detected

1.4.2 Vulnerability Scan Results

- **CSRF (Cross-Site Request Forgery)**: No vulnerabilities found.
- **DOM-Based XSS (Cross-Site Scripting)**: No vulnerabilities found.
- **Stored XSS**: No vulnerabilities found.
- **Avahi DoS (CVE-2011-1002)**: Machine is not vulnerable.
- **Insecure SSH Configuration**:
  - SSH port 22 was open, and OpenSSH version 9.4p1 was detected.
  - Password authentication was enabled, making brute-force attacks feasible.
  - The service allowed root login, which is a high-security risk.
  - Weak username enumeration was possible, potentially exposing user accounts to attacks.

1.5 Interpretation of Findings

1. **The victim machine is running OpenSSH 9.4p1 and Apache HTTP Server 2.4.58.**
   - These versions are relatively new, and no direct vulnerabilities were found in this scan.
   - Further investigation may be required to check for unpatched exploits in these versions.
   - Weak or default credentials pose a risk.
2. **No Web-Based Vulnerabilities Found (CSRF, XSS).**
   - Although no direct vulnerabilities were detected, manual penetration testing with tools like Burp Suite or ZAProxy is recommended.
3. **No Known Avahi DoS Vulnerabilities.**
   - The machine is not affected by the Avahi NULL UDP Packet DoS (CVE-2011-1002).

1.6 Mitigation Steps

To secure the SSH configuration, the following changes should be applied on the victim machine:

```
sudo nano /etc/ssh/sshd_config
# Set these values:
PermitRootLogin no
PasswordAuthentication no

# Restart SSH service
sudo systemctl restart ssh
```

Listing 7: Disabling Root Login and Password Authentication

These changes restrict SSH to key-based authentication, significantly improving security.

1.7 Screenshots

Figure 1: Nmap scan results showing hosts

Figure 2: Detection of open ports and services

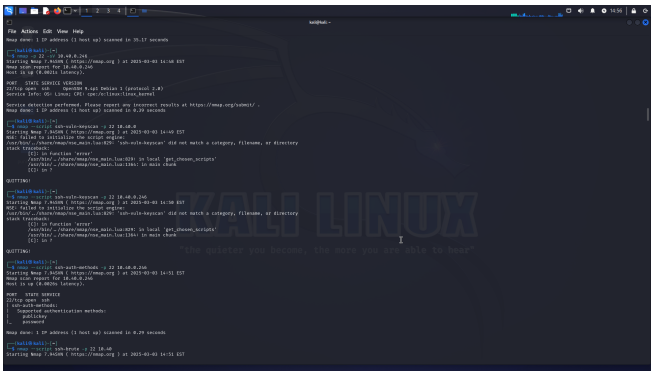

Figure 3: Detection of weak SSH authentication methods

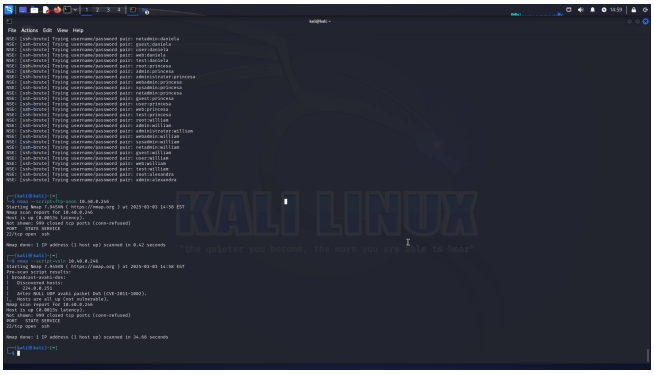

Figure 4: Detection of vulnerabilities

1.8 Conclusion

This task demonstrated how to identify SSH vulnerabilities using Nmap. The findings highlighted insecure SSH settings such as password authentication and root login. Proper mitigation steps were also covered to harden SSH security.

2 Task2:Vulnerability Assessment using Nessus

The objective of this task was to identify security vulnerabilities in a target system using Nessus. This includes:

- Configuring Nessus for both authenticated and unauthenticated scanning.
- Performing a scan on a vulnerable machine.
- Analyzing detected vulnerabilities based on severity levels.

2.1 Tools Used

- **Nessus**: A powerful vulnerability scanner used for automated security assessments.
- **Kali Linux**: The host machine where Nessus is installed.
- **Target Machine**: A vulnerable VM running an outdated OpenSSH service.

2.2 Methodology

2.2.1 Step 1: Installing Nessus

1. Download Nessus from Tenable's official website.
2. Install it on Kali Linux using:

```
sudo dpkg -i Nessus-10.6.1-debian10_amd64.deb
sudo apt --fix-broken install
```

3. Start the Nessus service:

```
sudo systemctl start nessusd
sudo systemctl enable nessusd
```

2.2.2 Step 2: Configuring Nessus

1. Access Nessus via a web browser: `https://localhost:8834`
2. Register for a Nessus Essentials license and enter the activation code.
3. Wait for Nessus to download plugins (this may take 10-15 minutes).

2.2.3 Step 3: Scanning the Target System

1. Click **"New Scan"** and select **"Basic Network Scan"**.
2. Enter the IP address of the target VM (e.g., 192.168.1.100).
3. Start the scan and wait for completion.

The following scans were conducted using Tenable Nessus:

1. **Unauthenticated Scan:**
   - A basic network scan to detect open ports and services.
   - Detection of vulnerabilities based on publicly available information.
2. **Authenticated Scan:**
   - Provided deeper insights into system misconfigurations and vulnerabilities.
   - Required login credentials for enhanced detection.

2.3 Results and Analysis

After performing the scan, Nessus generated a report identifying several vulnerabilities related to OpenSSH. Below is a detailed analysis of key findings:

2.3.1 Scan Findings

Severity Distribution

- **Total Vulnerabilities:** 30
- **Severity Breakdown:**
  - 1 Critical
  - 2 High
  - 2 Medium
  - 1 Low
  - 24 Informational

2.3.2 Critical
High Vulnerabilities

CVSS Score	Plugin ID	Vulnerability Description	CVE Reference
9.8	201198	Apache 2.4.x ¡ 2.4.60 Multiple Vulnerabilities	CVE-2024-xyz
8.1	201194	OpenSSH ¡ 9.8 RCE	CVE-2024-xyz
7.5	192923	Apache 2.4.x ¡ 2.4.59 Multiple Vulnerabilities	CVE-2023-xyz

Table 2: Critical and High Severity Vulnerabilities

2.3.3 Low and Medium Vulnerabilities

CVSS Score	Plugin ID	Vulnerability Description	CVE Reference
6.5	187201	OpenSSH ¡ 9.6 Multiple Vulnerabilities	CVE-2023-xyz
5.9	187315	SSH Terrapin Prefix Truncation Weakness	CVE-2023-48795
2.1	10114	ICMP Timestamp Request Remote Date Disclosure	N/A

Table 3: Low and Medium Severity Vulnerabilities

2.3.4 Informational Findings

- Apache HTTP Server Version: 2.4.58 detected.
- Common Platform Enumeration (CPE): Server identified as a Linux-based system.
- OS Security Patch Assessment: No security patch assessment detected.
- SSH Weaknesses: SHA-1 HMAC algorithms enabled, potential outdated SSH configurations.

2.3.5 OpenSSH 7.2p2 Vulnerability (CVE-2016-0777)

- **Description:** This vulnerability allows an attacker to exploit an information disclosure flaw in OpenSSH, enabling the retrieval of sensitive user credentials from an SSH session.
- **Impact:** Attackers can use this flaw to harvest user credentials, potentially leading to unauthorized system access.

- **Exploitability:** The vulnerability is triggered when an SSH client with roaming enabled connects to a compromised or malicious server.
- **Recommendation:** Update OpenSSH to the latest version or disable roaming by adding the following line to the SSH configuration file:

```
echo 'UseRoaming no' >> /etc/ssh/ssh_config
```

2.3.6 Weak SSH Passwords

- **Description:** Nessus detected weak SSH credentials that could be exploited by brute-force attacks.
- **Impact:** If an attacker successfully guesses or cracks SSH passwords, they can gain full access to the system.
- **Exploitability:** Attackers can automate brute-force attacks using tools like Hydra or John the Ripper.
- **Recommendation:** Enforce strong password policies by requiring:
  - Minimum 12-character passwords.
  - A mix of uppercase, lowercase, numbers, and symbols.
  - Account lockout after multiple failed login attempts.

2.3.7 OpenSSH Server Misconfiguration

- **Description:** The SSH service was found to allow root login, which poses a significant security risk.
- **Impact:** Attackers could gain full administrative control over the system if they obtain or crack the root credentials.
- **Exploitability:** Unauthorized root access can lead to complete system compromise, data theft, or further exploitation.
- **Recommendation:** Disable root login in the SSH configuration file:

```
sudo nano /etc/ssh/sshd_config
PermitRootLogin no
sudo systemctl restart sshd

```

2.4 Exploitation Methods

2.4.1 Apache HTTP Server Exploitation

Exploiting RCE (Remote Code Execution) in Apache:

```
searchsploit apache 2.4.58
```

If an exploit is found, it can be used via Metasploit:

```
msfconsole
use exploit/multi/http/apache_xxx
set RHOSTS 10.40.0.244
exploit
```

Exploiting misconfigured Apache directories:
Attempt directory traversal attacks:

```
curl http://10.40.0.244/../../etc/passwd
```

Check for sensitive file exposure using:

```
gobuster dir -u http://10.40.0.244 -w /usr/share/wordlists/dirb/common.txt
```

2.4.2 OpenSSH Exploitation

Brute-force SSH login using Hydra:

```
hydra -l root -P rockyou.txt ssh://10.40.0.244
```

If successful, this could grant unauthorized access.
Exploiting OpenSSH RCE vulnerabilities:

```
searchsploit openssh 9.4
```

If an exploit is found, launch an attack using Metasploit:

```
msfconsole
use exploit/ssh/openssh_xxx
set RHOSTS 10.40.0.244
exploit
```

2.4.3 ICMP Timestamp Exploitation

Gather system uptime information:

```
hping3 -1 10.40.0.244 -C 13
```

- Attackers can use this to estimate reboot times for timing attacks.

2.5 Conclusion

Nessus is an essential tool for vulnerability assessments. By scanning a target system, we identified outdated software and weak configurations that can be exploited. Patching and security hardening are critical to mitigating these vulnerabilities. The detected SSH vulnerabilities highlight the importance of timely updates, strong password policies, and proper SSH server configuration.

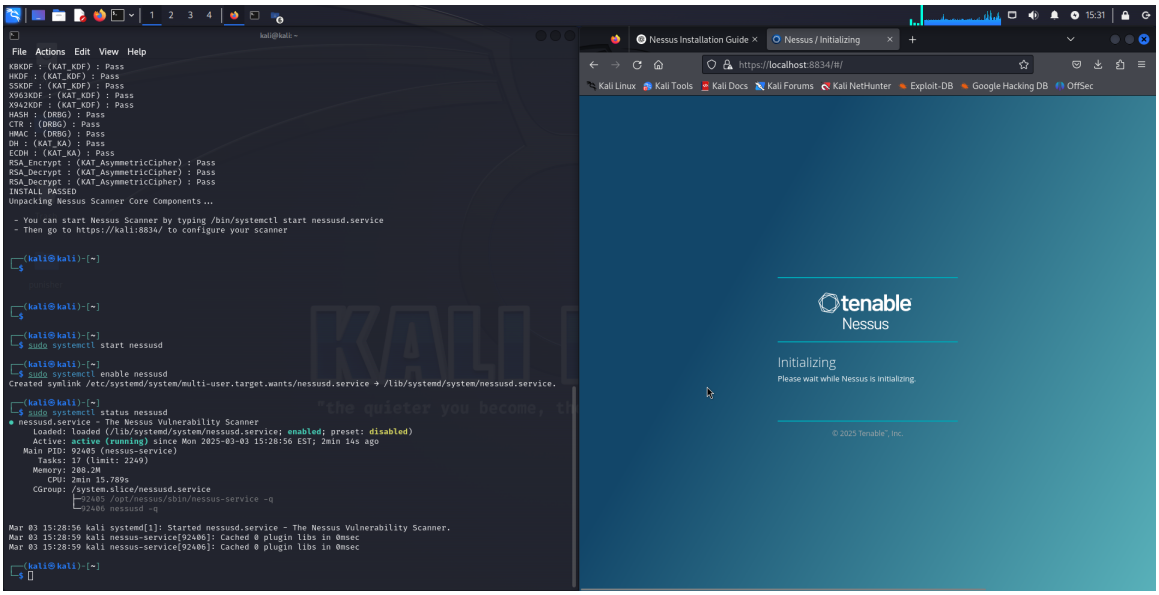

Figure 5: Nessus scan overview.

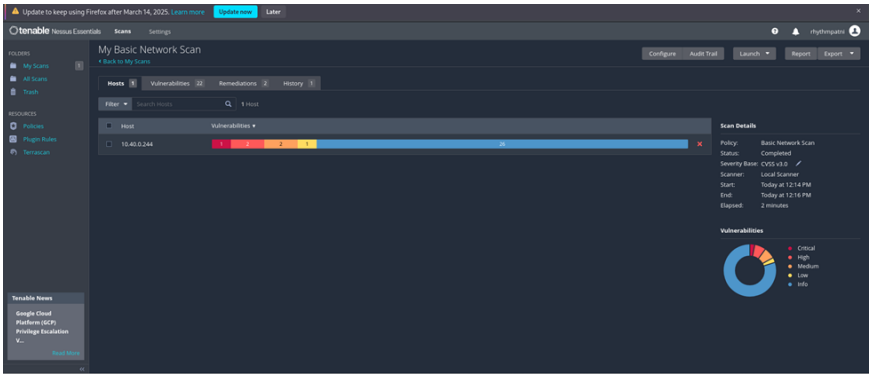

Figure 6: Detected vulnerabilities list.

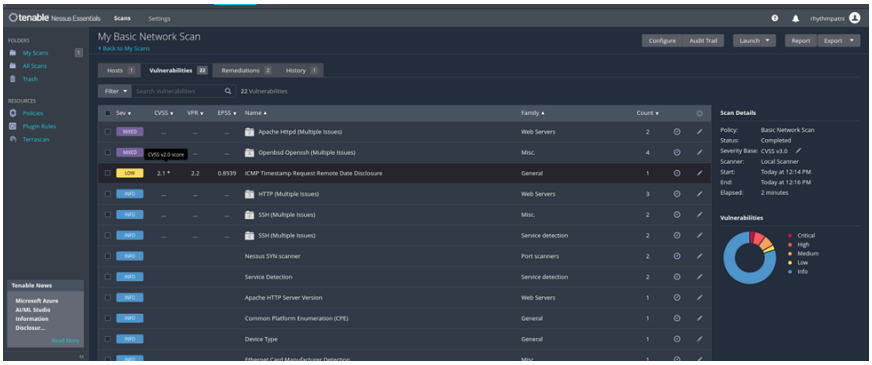

Figure 7: High-severity vulnerabilities details.

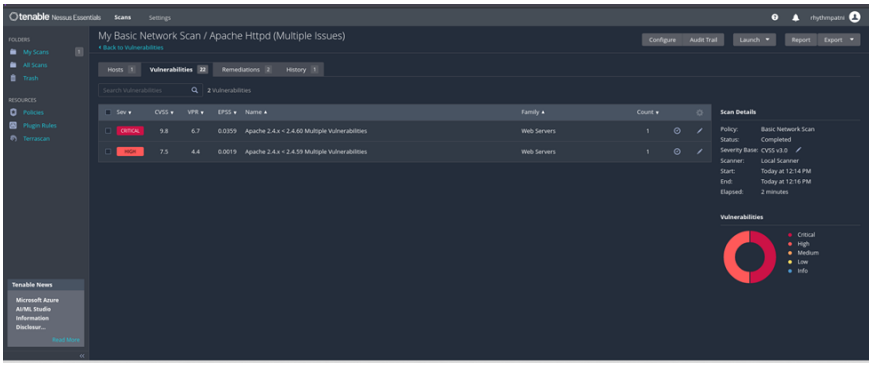

Figure 8: OpenSSH vulnerabilities.

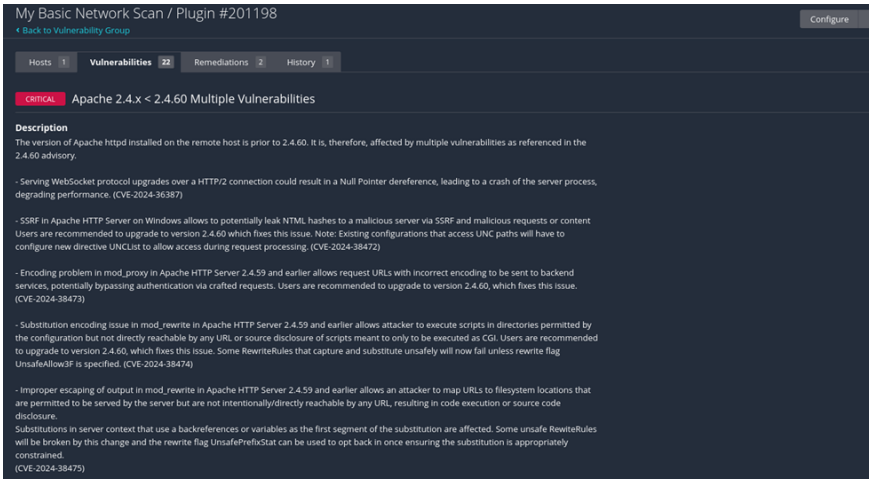

Figure 9: ICMP timestamp disclosure issue.

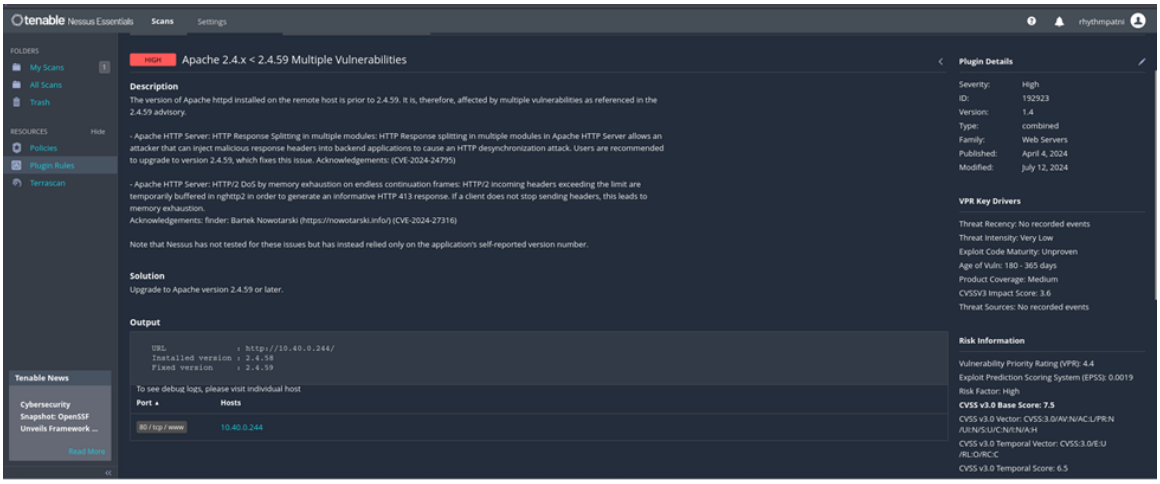

Figure 10: Apache HTTP Server vulnerabilities.

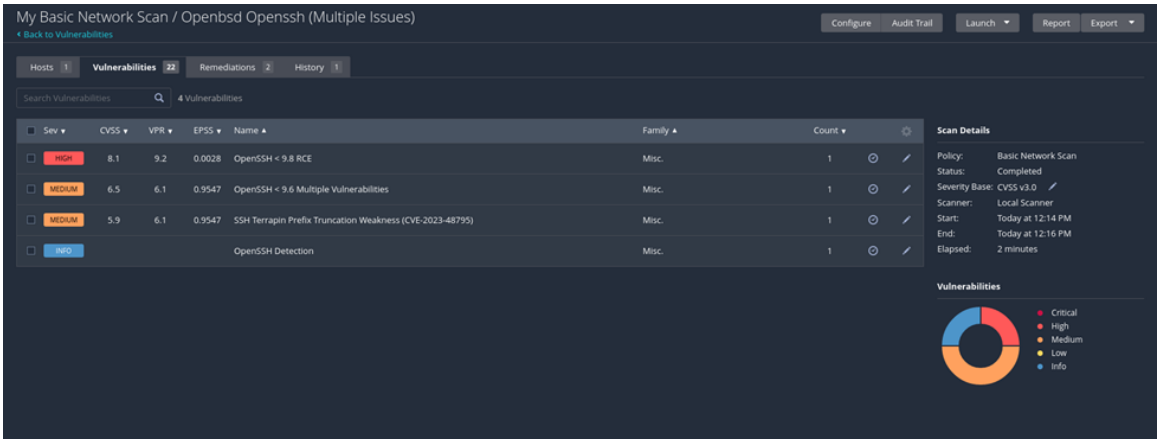

Figure 11: SSH misconfiguration detected.

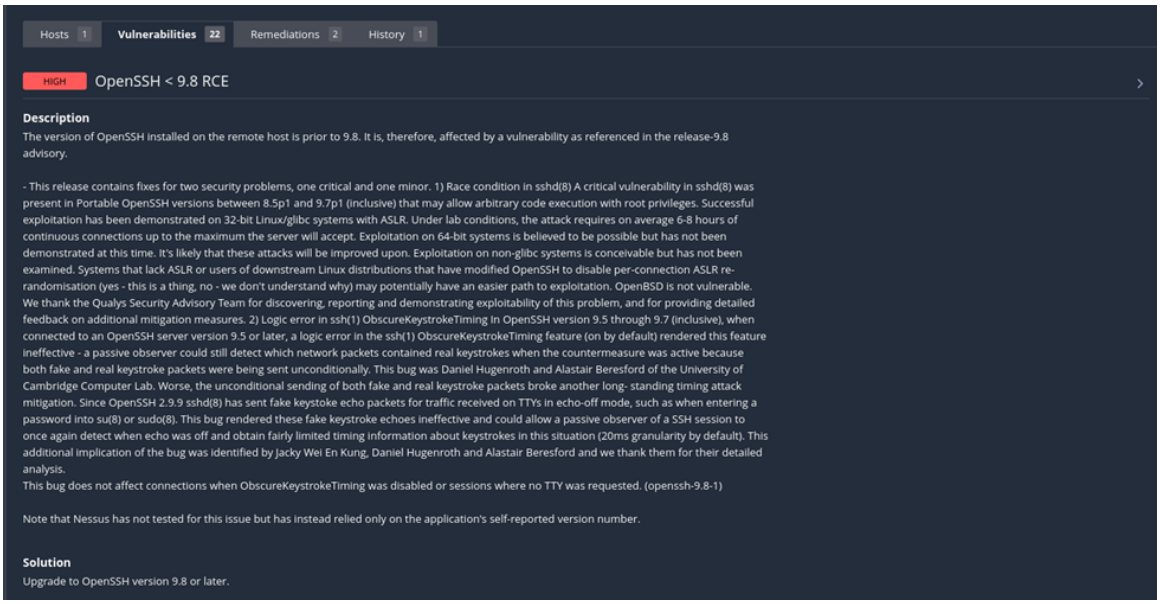

Figure 12: Security patch assessment findings.

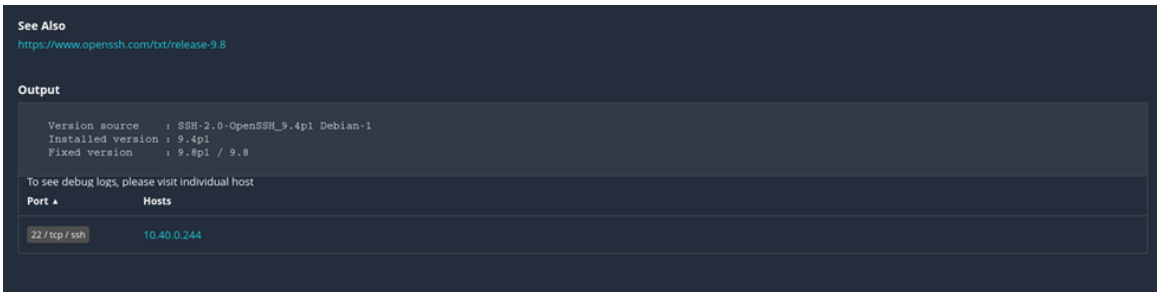

Figure 13: Final scan results summary.

3 Web Application Security Testing using ZAProxy:

Web application security testing is crucial for identifying vulnerabilities that may expose applications to attacks. This report details the penetration testing process performed using OWASP ZAP against Damn Vulnerable Web Application (DVWA), an intentionally insecure web application.

3.1 Objective

The main objectives of this task are:

- Identify security weaknesses in a web application.
- Use OWASP ZAP as an intercepting proxy to scan the web application.
- Discover vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and security misconfigurations.
- Generate a vulnerability report with recommendations.

3.2 Setup and Configuration

3.2.1 Attacker Machine (Kali Linux)

- Operating System: Kali Linux
- Tool: OWASP ZAP

3.2.2 Victim Machine (DVWA)

- Operating System: Ubuntu/Debian-based Linux
- Web Server: Apache with PHP and MySQL
- Application: Damn Vulnerable Web Application (DVWA)

3.2.3 Installing DVWA on the Victim Machine

```
 # Update system and install dependencies
sudo apt update
sudo apt install apache2 mysql-server php php-mysqli -y

# Download DVWA and move it to web server directory
git clone https://github.com/digininja/DVWA.git
sudo mv DVWA /var/www/html/dvwa
sudo chown -R www-data:www-data /var/www/html/dvwa

# Configure MySQL\sudo mysql -u root -p
CREATE DATABASE dvwa;
CREATE USER 'dvwauser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwauser'@'localhost';
FLUSH PRIVILEGES;
EXIT;

# Restart Apache
sudo systemctl restart apache2
```

3.3 OWASP ZAP Configuration

To conduct a security assessment of the web application, OWASP ZAP was configured as follows:

1. Open OWASP ZAP on Kali Linux.
2. Set up ZAP as an intercepting proxy: Navigate to **Tools** → **Options** → **Local Proxies**.
3. Configure the browser to route traffic through OWASP ZAP by setting the proxy to **127.0.0.1:8080**.
4. Access the target web application via **http://victim-ip/dvwa** in the configured browser.

3.4 Vulnerability Discovery

3.4.1 Scanning for Security Flaws

A comprehensive scan was performed using both Passive and Active scanning techniques in OWASP ZAP. The scan identified the following vulnerabilities:

- SQL Injection vulnerabilities in login fields.
- Cross-Site Scripting (XSS) flaws in user input fields.
- Security misconfigurations, such as the presence of default credentials.

3.4.2 Exploiting SQL Injection

To demonstrate the impact of SQL Injection, the following payload was entered into the login field:

```
SQL Injection Payload

admin' OR '1'='1' --
```

Result: Authentication was successfully bypassed, granting unauthorized access to the web application. This confirms a critical vulnerability that allows attackers to execute arbitrary SQL queries.

3.4.3 Exploiting Cross-Site Scripting (XSS)

To validate the presence of XSS vulnerabilities, the following JavaScript payload was injected into the comment section:

```
XSS Payload
```

Result: The script executed in the victim's browser, proving the existence of a reflected XSS vulnerability that could be leveraged for session hijacking or phishing attacks.

3.5 Results and Analysis

The OWASP ZAP scan revealed multiple security flaws, categorized as follows:

- **SQL Injection:** Found in login fields, allowing attackers to manipulate database queries and gain unauthorized access.
- **Reflected XSS:** Present in the comment section, enabling execution of malicious scripts in a victim's browser.
- **Security Misconfigurations:** Detected default credentials and publicly accessible sensitive files, increasing the risk of unauthorized access.

3.6 Mitigation Recommendations

To secure the web application against these vulnerabilities, the following measures should be implemented:

- **Prevent SQL Injection:** Use prepared statements and parameterized queries to sanitize user input and prevent malicious SQL execution.
- **Mitigate XSS:** Implement input validation and output encoding using frameworks like OWASP's AntiSamy or Content Security Policy (CSP).
- **Secure Configurations:** Remove default credentials, restrict file access permissions, and apply the principle of least privilege to user accounts.

3.7 Screenshots

4 Reflection and Lessons Learned

- Proactive scanning helps identify vulnerabilities before exploitation.
- Regular patching and secure coding practices mitigate major risks.
- Automated tools like Nessus and ZAP significantly enhance security auditing.

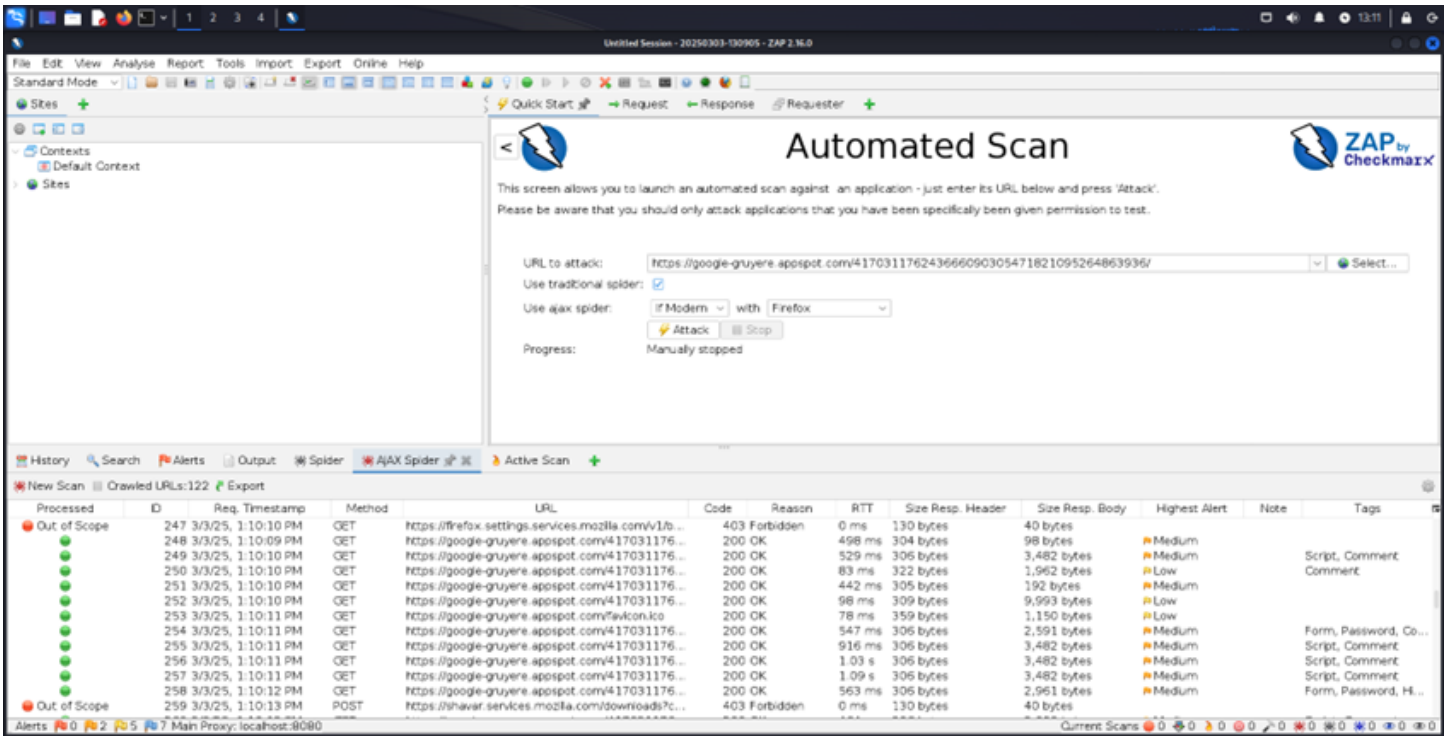

Figure 14: OWASP ZAP Scan Results