**Information Security**

**Assignment 1**

**I210844**

**Abdullah Chaudhary**



**Submitted To:**

**Fast NUCES**

# 

# 

[**Task 1 2**](#_tyjcwt)

[Introduction: 2](#_3dy6vkm)

[Tools Used: 2](#_1t3h5sf)

[Steps Taken: 3](#_4d34og8)

[1. IP Address Identification: 3](#_2s8eyo1)

[2. Connectivity Check: 3](#_17dp8vu)

[3. Vulnerability Scan with Nmap: 3](#_3rdcrjn)

[4. Launching Metasploit: 3](#_26in1rg)

[5. Searching for the Vulnerability: 3](#_lnxbz9)

[6. Selecting the Exploit: 4](#_35nkun2)

[7. Setting the Target IP: 4](#_1ksv4uv)

[8. Executing the Exploit: 4](#_44sinio)

[9. Post-Exploitation: 4](#_2jxsxqh)

[10. Shell Access and User Creation: 4](#_z337ya)

[11. Persistence: 4](#_3j2qqm3)

[Countermeasures and Mitigation: 5](#_1y810tw)

[Conclusion: 5](#_4i7ojhp)

[Task 2 5](#_2xcytpi)

[Steps Taken: 5](#_1ci93xb)

[1. File Type Determination: 5](#_3whwml4)

[2. VirusTotal Analysis: 6](#_2bn6wsx)

[3. Finding the Roll Number: 6](#_qsh70q)

[4. Malicious File Details: 6](#_3as4poj)

[5. API Analysis: 6](#_1pxezwc)

[**Task 3 6**](#_49x2ik5)

[Task 1: Reconnaissance and Vulnerability Identification 7](#_2p2csry)

[Tools Used: 7](#_147n2zr)

[Task 2: CVEs and Potential Exploits in the Tech Stack 7](#_3o7alnk)

[Identified Vulnerabilities: 7](#_23ckvvd)

[Task 3: Results of Nmap Scan 8](#_ihv636)

[Nmap Scanning Details: 8](#_32hioqz)

[Exploits and Mitigation: 8](#_1hmsyys)

[Conclusion: 9](#_41mghml)

# 

# Task 1

## Introduction:

One vulnerability assessment of the organization's critical systems led me to encounter the Windows 7 Operating System vulnerability known as Eternal Blue. This is a remote code execution flaw of MS17-010 in Microsoft's SMBv1 protocol. The application of this vulnerability allows hackers to take control of a machine from a distance, from which they can proceed with further exploitation and persistence. This paper details the steps followed during a simulated assessment using Kali Linux and Metasploit, which includes mitigation recommendations for the risk.

## Tools Used:

* **Kali Linux VM**: Penetration testing operating system
* **Windows 7 OS**: Target system vulnerable to Eternal Blue
* **Nmap**: For scanning and identifying the SMB vulnerability
* **Metasploit Framework**: For exploiting the vulnerability

## Steps Taken:

### **1. IP Address Identification:**

I began by identifying the IP address of the vulnerable Windows 7 machine. This was critical for targeting the system with Nmap and Metasploit.

* Command: ipconfig (on Windows 7) to retrieve its IP address.

### **2. Connectivity Check:**

After switching to Kali Linux, I initiated a ping test to ensure network connectivity between the Kali and the Windows 7 system.

* Command: ping <Windows\_7\_IP>
* Output: A successful response confirmed that the target system was reachable.

### 3. Vulnerability Scan with Nmap:

I proceeded by scanning the target system to identify any vulnerabilities and open ports using the Nmap tool. Nmap is efficient in providing information about services running on a system.

* Command: nmap -sv <Windows\_7\_IP>
* Output: The scan revealed that the SMBv1 service was active and vulnerable to MS17-010.

### **4. Launching Metasploit:**

Next, I launched Metasploit (msfconsole) on Kali Linux, which is one of the most powerful tools for exploiting vulnerabilities.

* Command: msfconsole

### **5. Searching for the Vulnerability:**

I searched for the Eternal Blue vulnerability using the built-in Metasploit database.

* Command: search ms17

This resulted in listing multiple exploits for the Eternal Blue vulnerability, with **exploit/windows/smb/ms17\_010\_eternalblue** being the specific one I selected.

### **6. Selecting the Exploit:**

After identifying the correct exploit, I chose the first option:

* Command: use 0

### **7. Setting the Target IP:**

I configured the target machine’s IP address using the **RHOST** parameter to ensure the exploit targeted the correct system.

* Command: set RHOST <Windows\_7\_IP>

### **8. Executing the Exp**loit:

With the settings configured, I launched the exploit. This step initiated the remote execution process, leveraging the vulnerability.

* Command: exploit

Upon successful exploitation, a meterpreter session was opened, granting control of the target system.

### **9. Post-Exploitation:**

With access to the Windows 7 machine, I began gathering information by listing directories and checking the current working directory.

* Commands: pwd and ls

### **10. Shell Access and User Creation:**

To establish further control, I dropped into a command shell on the Windows machine and created a new user for persistence.

* Commands: shell, followed by net user newadmin password /add

### **11. Persistence:**

To ensure long-term access, I employed techniques such as adding the user to administrative groups and setting up services that allow me to maintain access even after a reboot.

* Command: net localgroup administrators newadmin /add

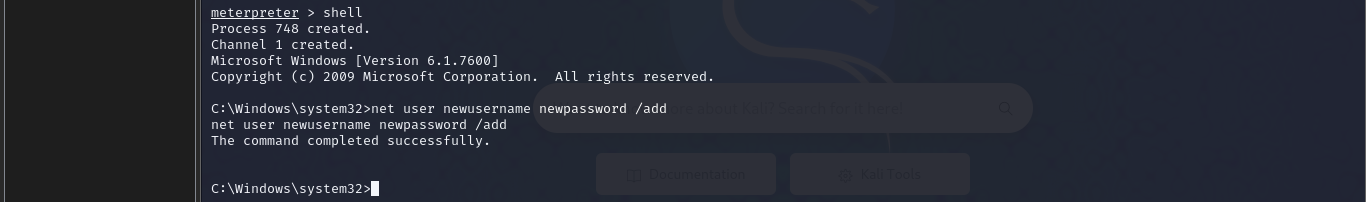
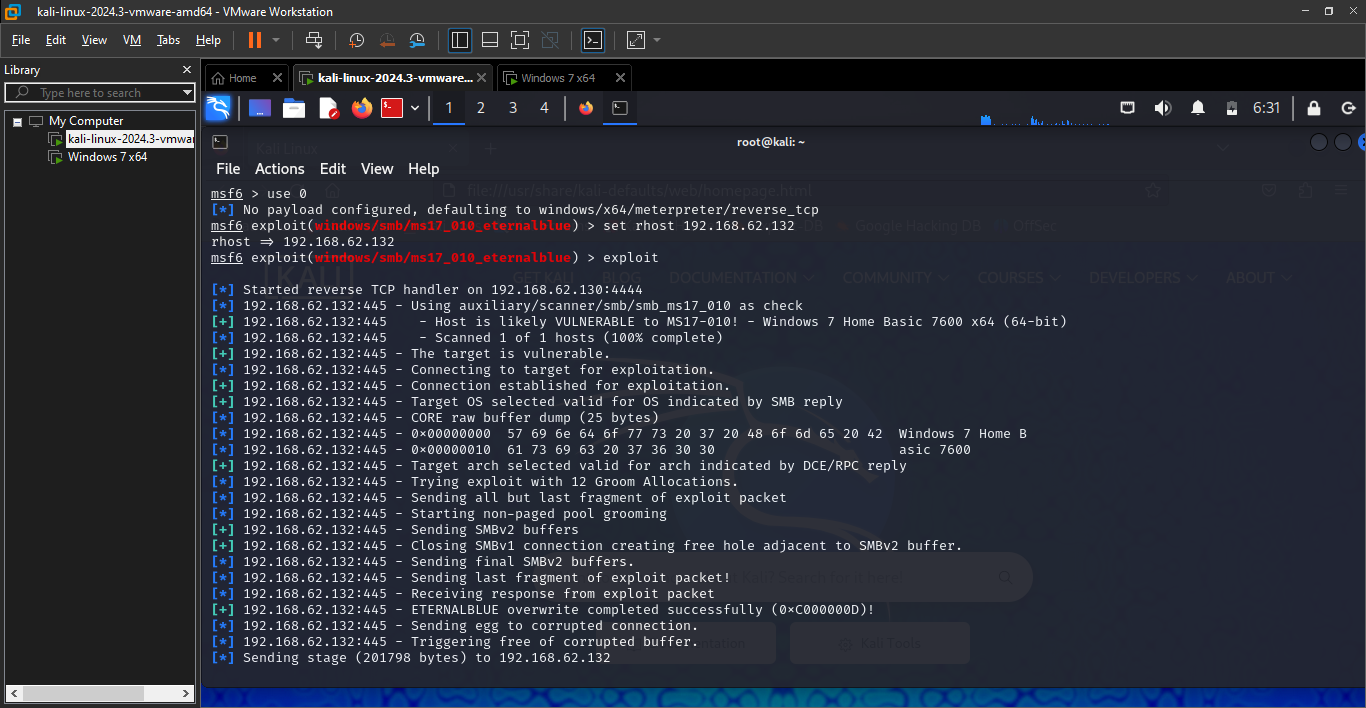
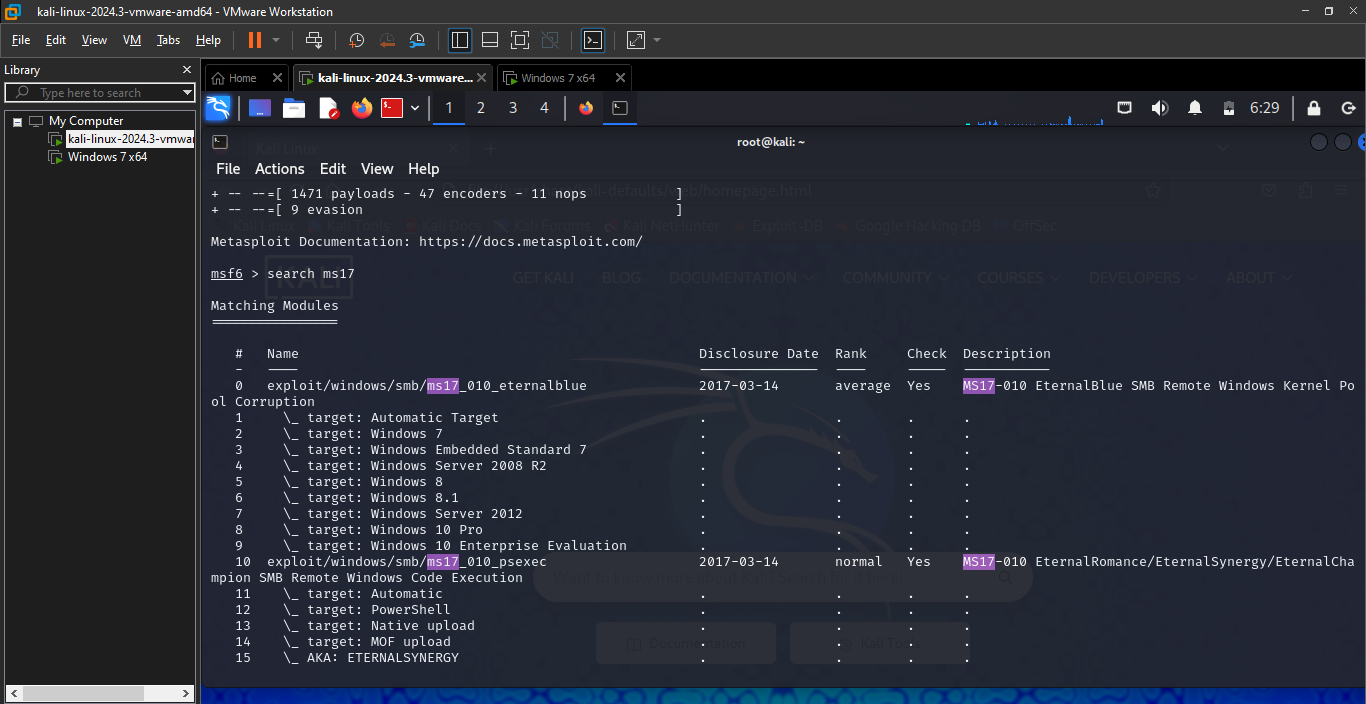
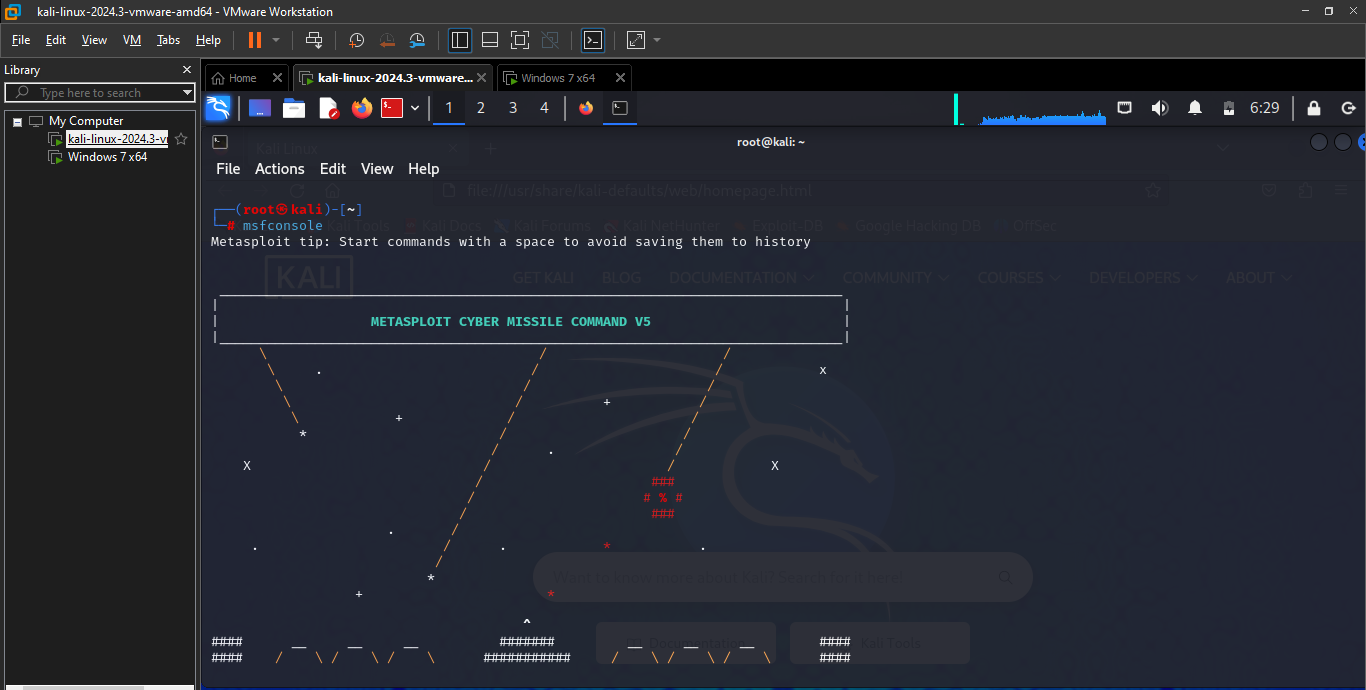
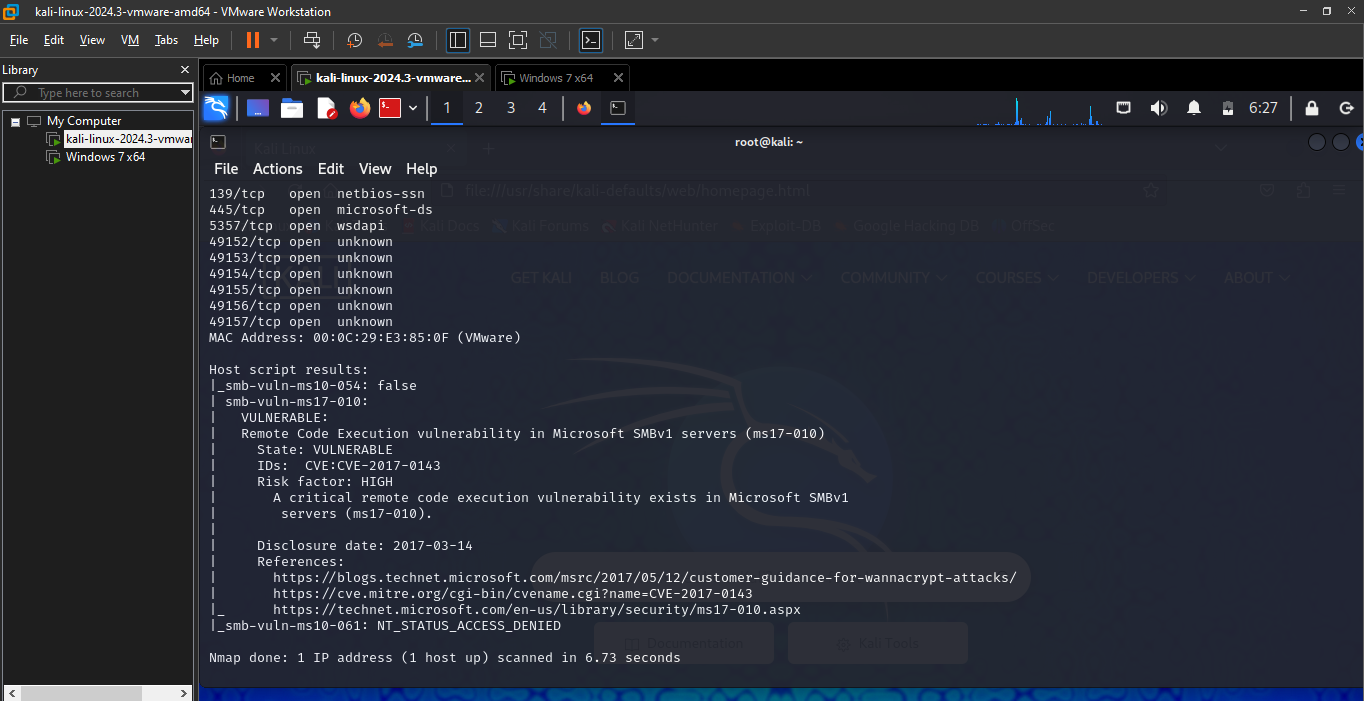
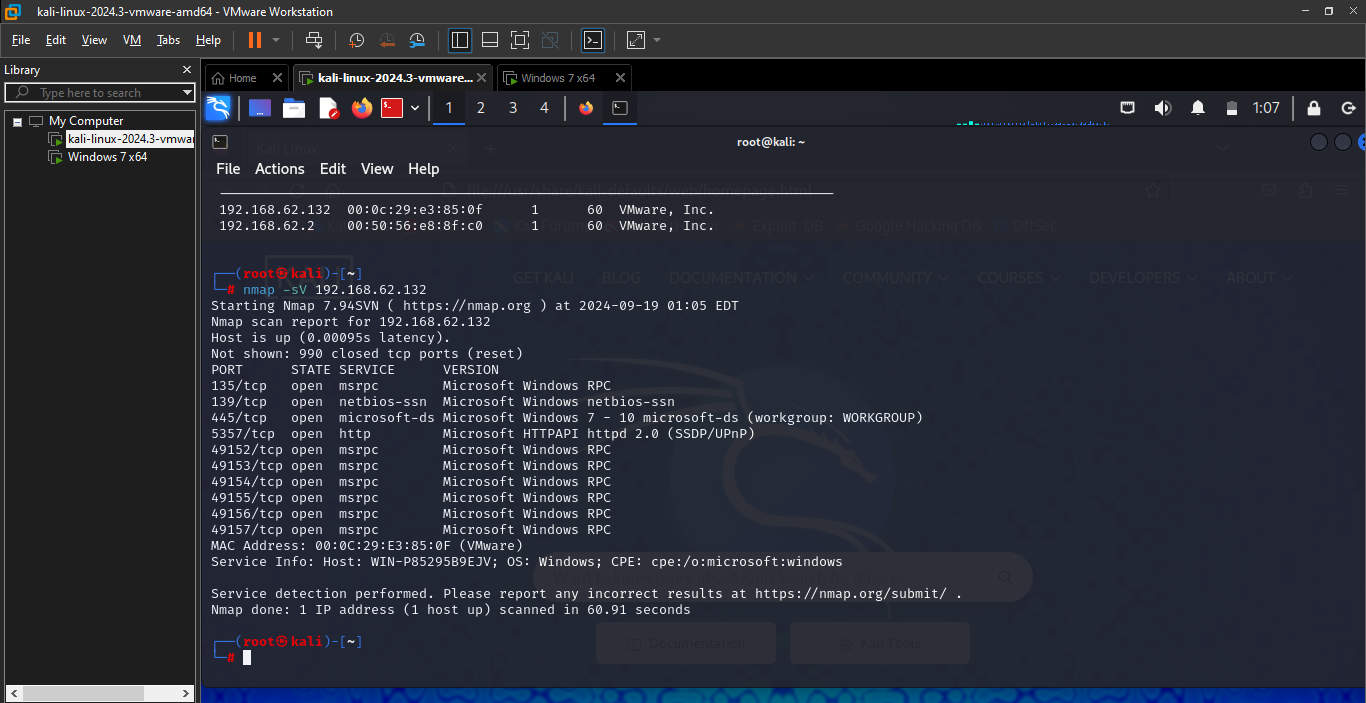
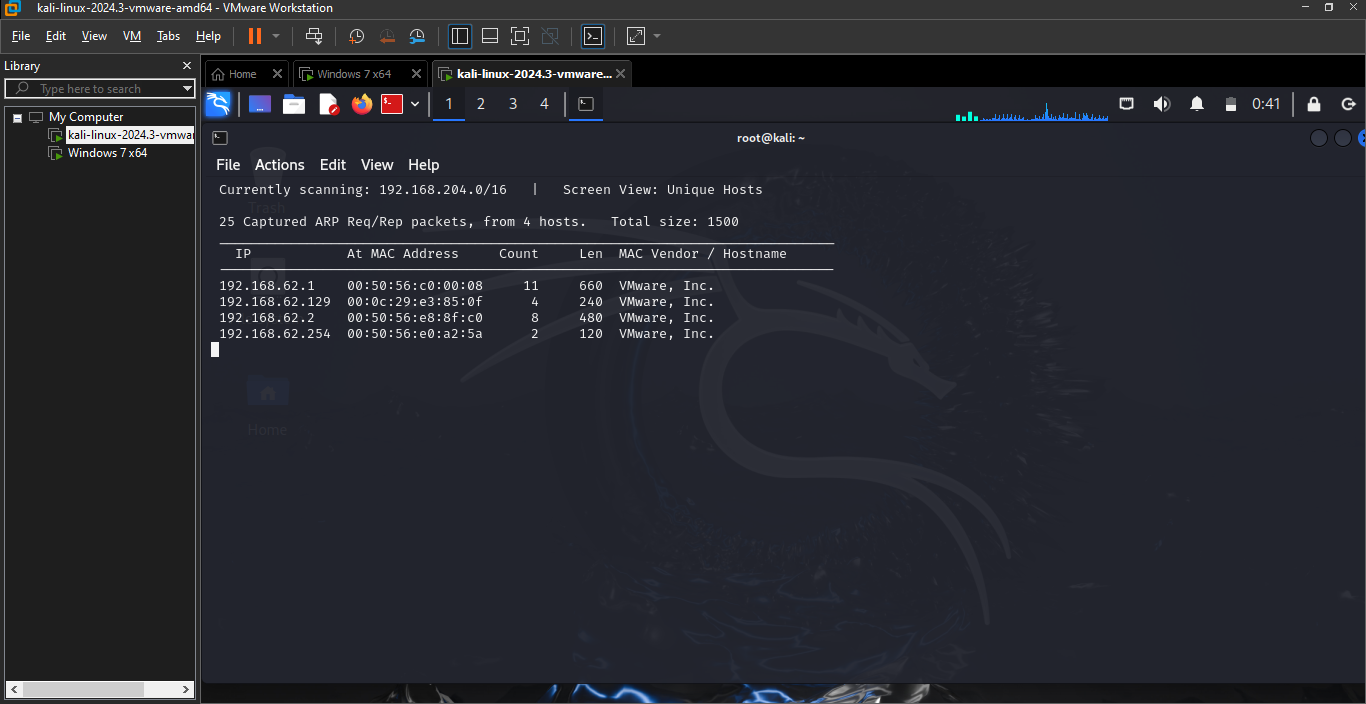
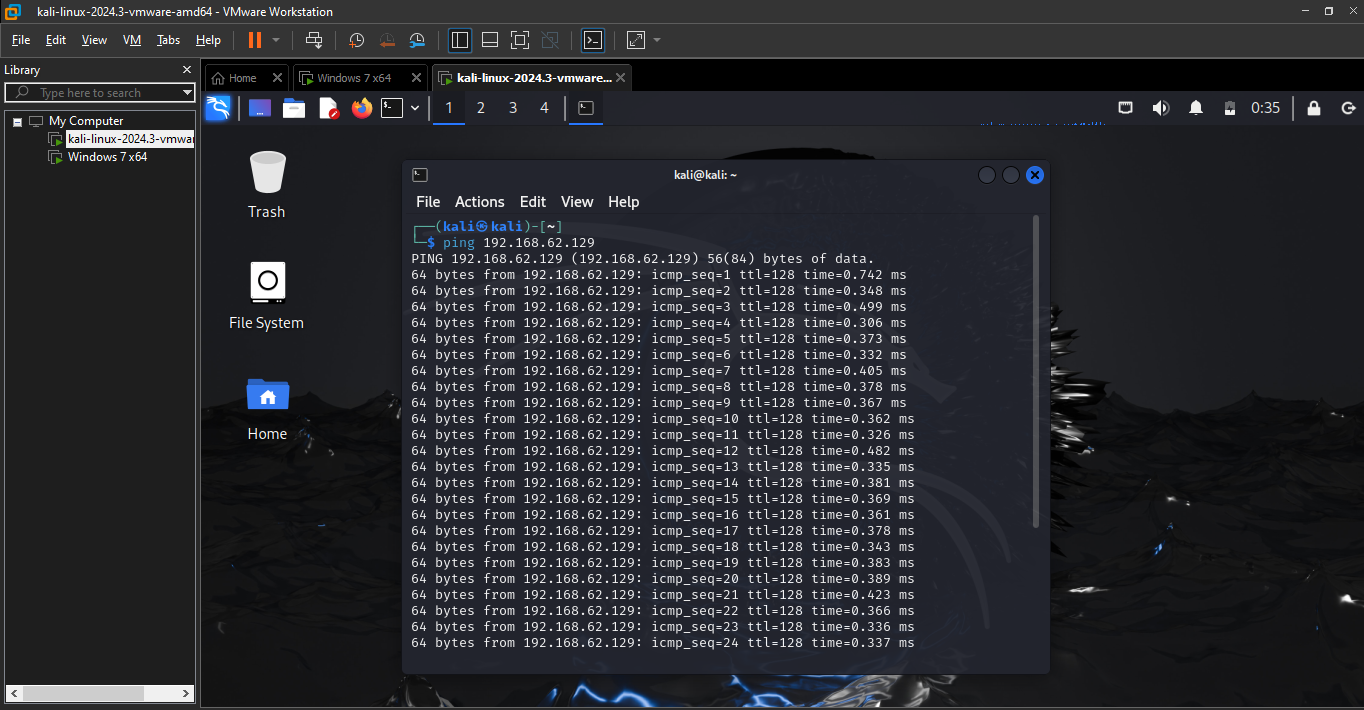
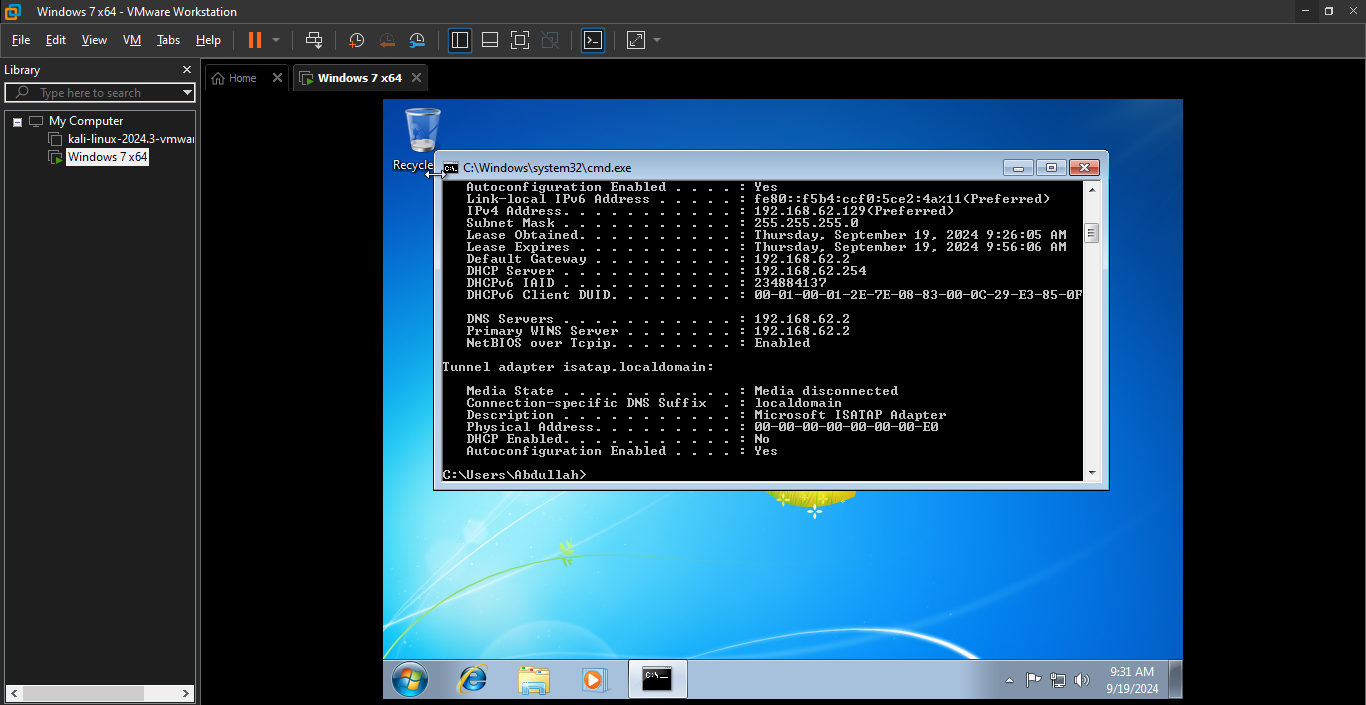
### **Countermeasures and Mitigation:**

The exploitation of Eternal Blue poses a significant risk to systems using outdated SMB protocols. To mitigate this threat, the following actions are recommended:

* **Apply MS17-010 Patch**: Ensure all Windows systems are updated with the MS17-010 security patch, which addresses the vulnerability in SMBv1.
* **Disable SMBv1**: Since SMBv1 is outdated and prone to security issues, disabling it can prevent further attacks. This can be done via PowerShell commands or through group policies.
* **Firewall Configuration**: Block access to SMB-related ports (such as 445) at the network perimeter to prevent external threats from reaching vulnerable systems.
* **Regular Vulnerability Scanning**: Continuously scan systems for known vulnerabilities to stay ahead of potential exploits.
* **User Privileges and Network Segmentation**: Ensure that only necessary users have administrative rights, and segment critical systems from less secure areas of the network.

### **Conclusion:**

The Eternal Blue vulnerability, MS17-010, is a critical risk if it goes unpatched. In the assessment, I was able to exploit this vulnerability and gain persistence on the target machine by using Kali Linux and Metasploit. Organizations can cut down the chance of such exploits in the future by using proper patches to remediate vulnerability, disabling SMBv1, and having network best practices in place.



## **Task 2**

The main tasks of this assignment were to analyze some sample files given, determine their type, whether they are malicious or benign, and extract information from the malicious ones. A virtual machine was used in the analysis so as not to risk the primary system.

## Steps Taken:

### **1. File Type Determination:**

The first was to write down what each file is. That's essential for understanding the nature of the file and how it works. Using the command-line tool, I used a file command on each of the samples to try and recognize whether it is a text document, an image, or some executable.

* Command: file <sample\_filename>

./Data mining concepts and techniques.docx: PDF document, version 1.6 (zip deflate encoded)

./memory.mp3: PDF document, version 1.7, 12 page(s)

./python-tutorial.mp4: PE32 executable (GUI) Intel 80386, for MS Windows, 6 sections

./hd: assembler source, ASCII text

./file\_types.txt: ASCII text

./fragment.xml: PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections

./Ch # 14 Winding Up.pdf: PDF document, version 1.7

./notes.txt: PE32+ executable (GUI) x86-64, for MS Windows, 7 sections

./exit: assembler source, ASCII text

./ltrace\_output.txt: empty

./tutorial.md: PE32 executable (GUI) Intel 80386, for MS Windows, 6 sections

./19. Near-Memory & In-Memory Detection of Fileless Malware-2020.pdf: PDF document, version 1.7, 12 page(s)

./extracted\_strings.txt: assembler source, ASCII text, with very long lines (1264)

./d': assembler source, ASCII text, with very long lines (1264)

./c++ primer.pdf: PE32 executable (GUI) Intel 80386, for MS Windows, 5 sections

./c: ASCII text

./rkhunter.tar.gz: empty

./capture.pcap: pcap capture file, microsecond ts (little-endian) - version 2.4 (Ethernet, capture length 262144)

./adioqhfeioj: assembler source, ASCII text

./adl': assembler source, ASCII text

./assignment1.docx: PE32+ executable (GUI) x86-64, for MS Windows, 7 sections

./vlc-3.0.20-win32.exe: PE32 executable (GUI) Intel 80386 (stripped to external PDB), for MS Windows, Nullsoft Installer self-extracting archive, 7 sections

./hello.py: PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows, 3 sections

./calculations.pdf: PE32 executable (GUI) Intel 80386, for MS Windows, 5 sections

./;kx: assembler source, ASCII text

./Data mining concepts.docx: PDF document, version 1.6 (zip deflate encoded)

### **2. VirusTotal Analysis:**

Besides, to find out whether those files were malicious or benign, I uploaded the sample files to VirusTotal. VirusTotal is a free online service that submits files for scanning by numerous antivirus engines. Comprehensive analysis was done, and details on the detection ratio, malicious indicators, and related metadata were included.

* Outcome: VirusTotal successfully flagged some files as malicious while others were marked benign. The platform also provided detailed reports on any URLs the malicious files contacted and any dropped files.

### **3. Finding the Roll Number:**

One of the tasks was to find my roll number hidden within one of the files. To accomplish this, I used the grep command, which is a powerful tool for searching text within files.

* Command: grep <roll\_number> <sample\_filename>

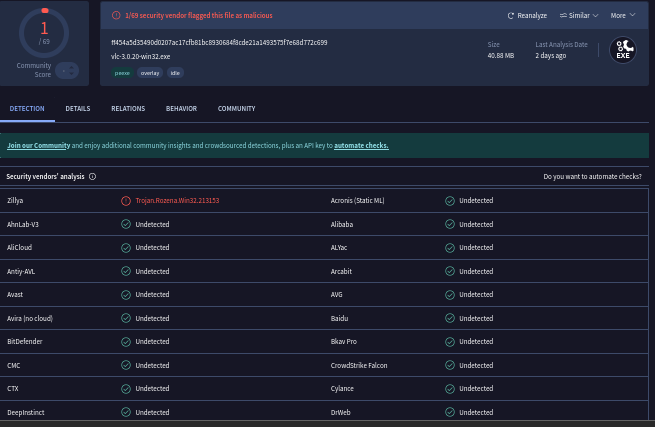
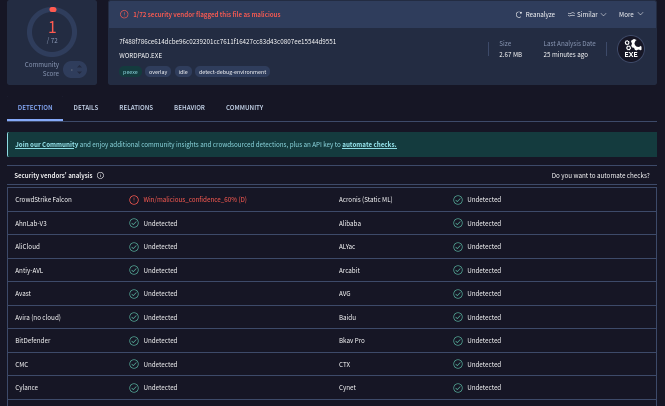
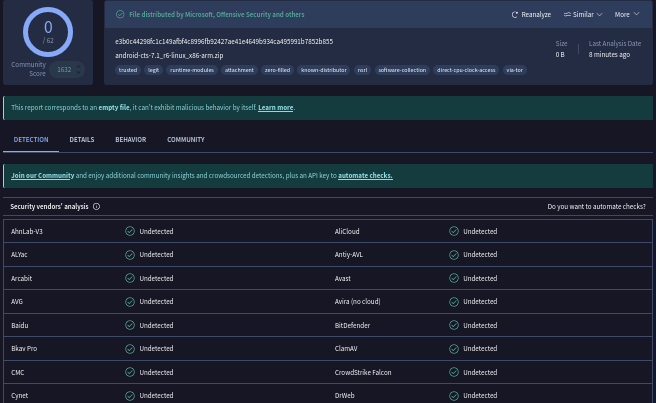
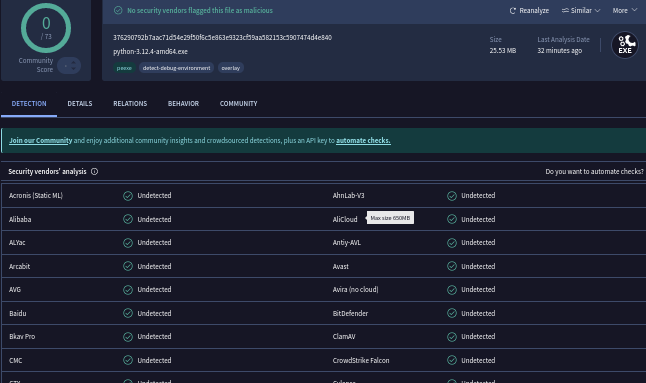
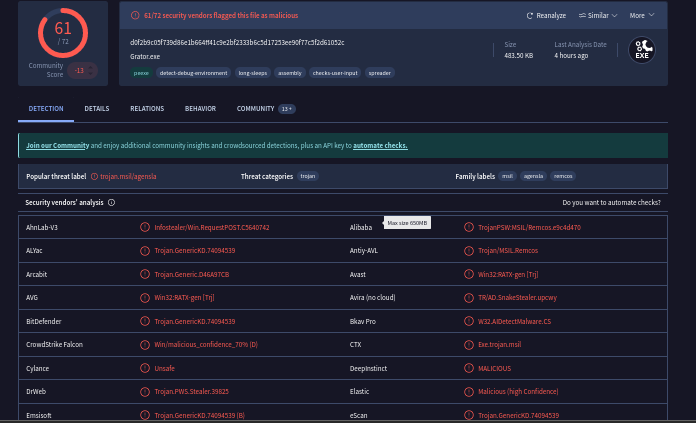
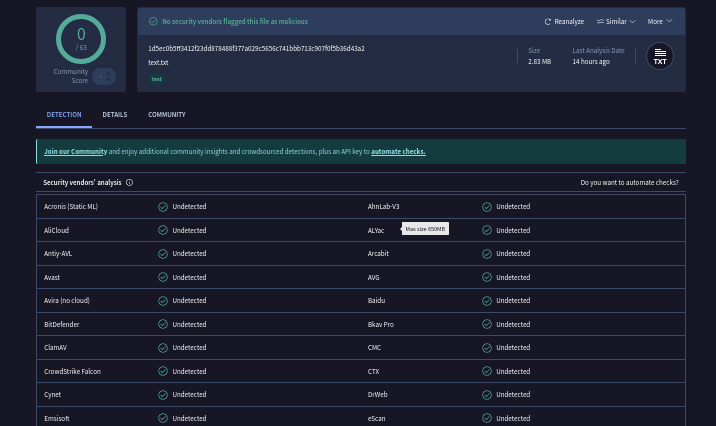
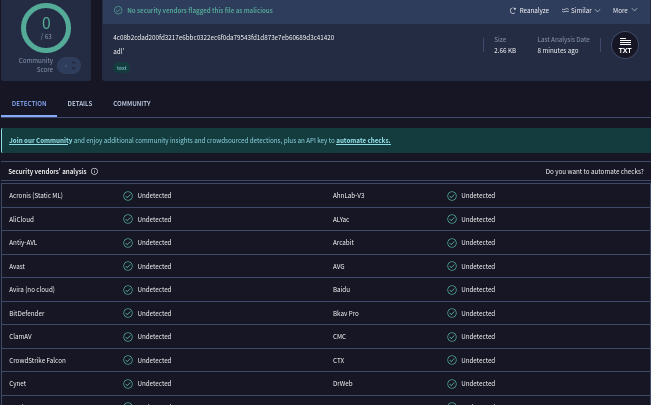
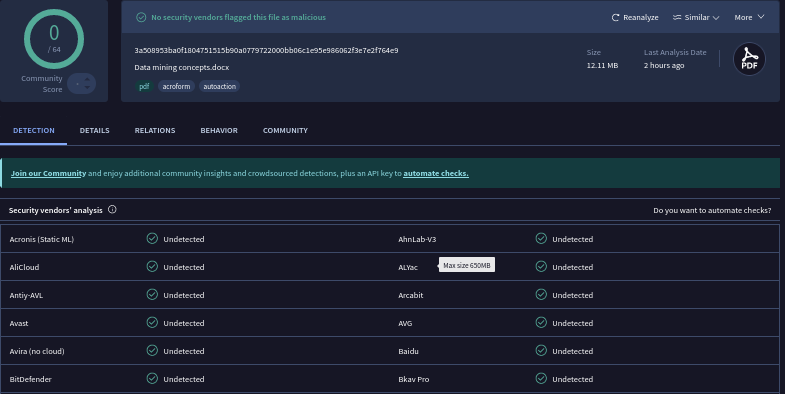
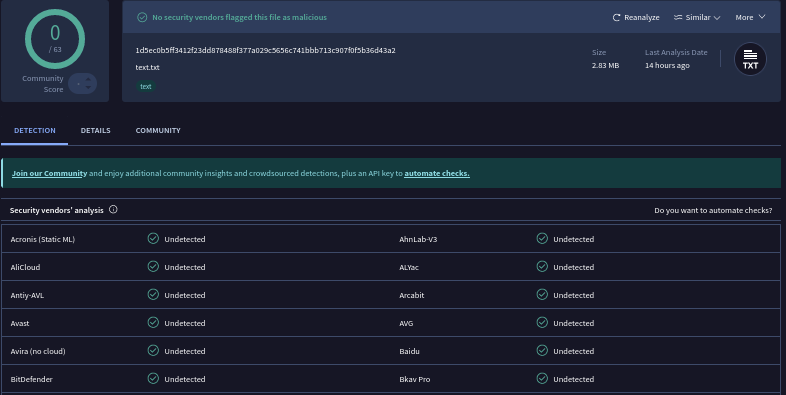
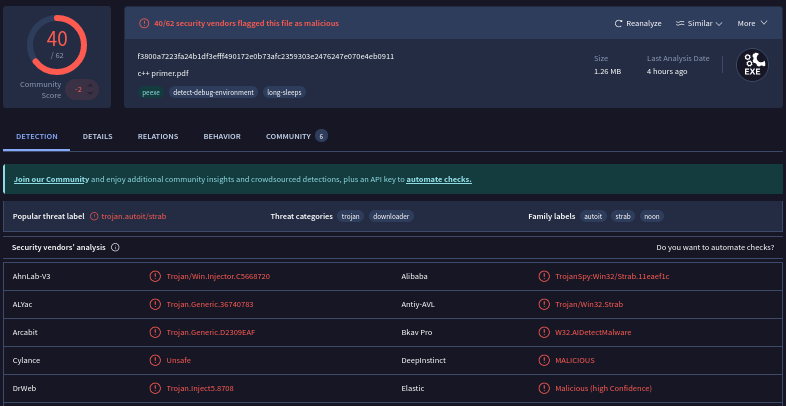
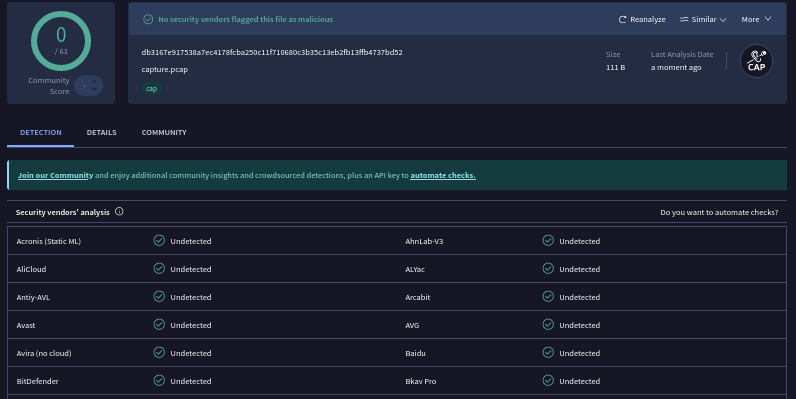
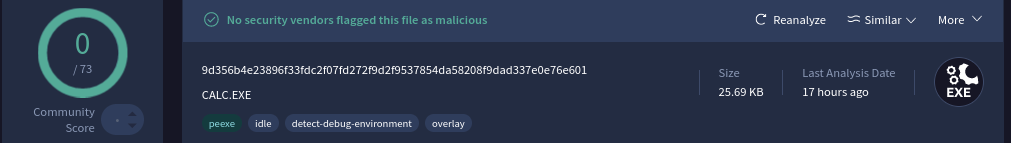
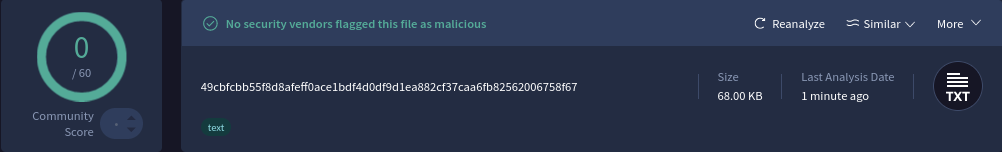
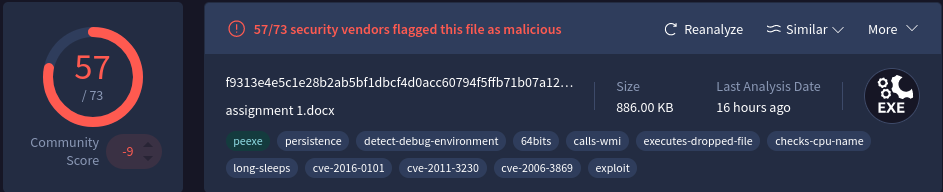
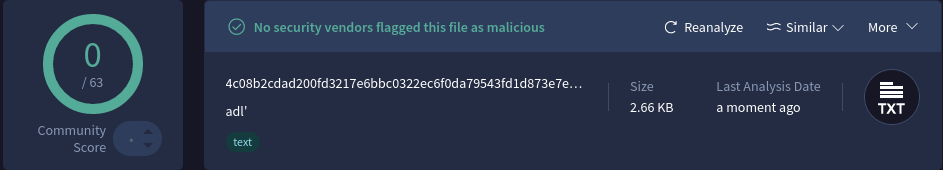
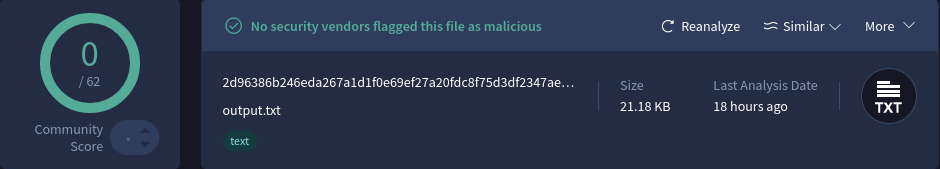
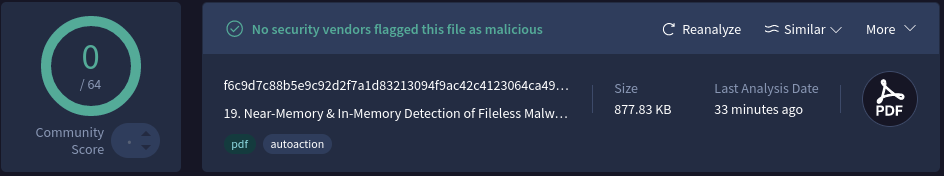
After running the command on multiple files, I was able to locate my roll number hidden within one of the text files. This confirmed that the specified file was mine.

### **4. Malicious File Details:**

The identified files, targeted maliciously and flagged, were dispersed under various malware categories, including Trojan or Worm. In this regard, VirusTotal also indicated if the identified malicious files contacted any suspicious URLs and whether they dropped any additional files onto the system, furthering their knowledge on how it works and what damages it may cause.

### **5. API Analysis:**

Though one of the assignments was to identify the APIs used by executable files, my assigned file was not executable. This made it impossible for me to get information about API usage. In the normal case of an executable file, API calls will also explain how a file can interface with the system and how malicious activities are implemented, such as manipulating files, accessing networks, or creating registry changes.



# Task 3

This task aims to scan vulnerabilities on the website **http://www.cms.comsats.edu.pk:8083** using reconnaissance tools. The results of this exercise provide an assessment of critical security flaws in the technology stack, possible exploits, and mitigation strategies to ensure security for the website. The tools applied were **Whois, dig, Nmap, WhatWeb, and zaproxy,** which altogether provided insight into the infrastructure of the website and possible attack vectors.

## **Task 1: Reconnaissance and Vulnerability Identification**

### Tools Used:

1. **Whois**: Gathered domain-related information, including IP address and hosting details.
2. **dig**: Used to query DNS records to gain insight into domain infrastructure.
3. **Nmap**: Employed for scanning services, version detection, and vulnerability assessment.
4. **WhatWeb**: Detected underlying web technologies and frameworks.
5. **Zaproxy**: Assisted in identifying the websites built.

## **Task 2: CVEs and Potential Exploits in the Tech Stack**

### Identified Vulnerabilities:

1. **ASP.NET 4.0**:
   * **Potential Exploit**: Cross-Site Scripting (XSS)
   * **Description**: Malicious scripts can be injected into the website, causing sensitive information to be exposed or the site to be defaced.
   * **Mitigation**: Use robust input validation and encoding techniques for user inputs to prevent untrusted scripts from executing.
2. **TypeForm**:
   * **Potential Exploit**: Form Injection or Data Exfiltration
   * **Description**: Unsanitized input in forms can lead to the injection of malicious code or unauthorized access to user data.
   * **Mitigation**: Validate form inputs, encrypt data during transmission, and regularly audit forms for vulnerabilities.
3. **AJAX Libraries**:
   * **Potential Exploit**: Improper Input Handling (XSS or CSRF)
   * **Description**: Lack of input sanitization in AJAX requests can lead to cross-site scripting or cross-site request forgery attacks.
   * **Mitigation**: Implement Content Security Policies (CSPs) and anti-CSRF tokens to ensure secure handling of input data.
4. **Twitter Bootstrap**:
   * **Potential Exploit**: Outdated Libraries Leading to XSS
   * **Description**: Outdated versions of Bootstrap may contain vulnerabilities that allow attackers to exploit user input forms.
   * **Mitigation**: Regularly update Bootstrap libraries and sanitize all inputs in forms and modals to prevent script injections.
5. **JavaScript Frameworks**:
   * **Potential Exploit**: DOM-based XSS
   * **Description**: Manipulation of untrusted inputs within JavaScript can result in DOM-based XSS, leading to a compromised user experience.
   * **Mitigation**: Ensure dynamic content within JavaScript is properly validated and sanitized to prevent malicious script execution.

## **Task 3: Results of Nmap Scan**

### Nmap Scanning Details:

The **Nmap** scan of the website was conducted using the following command:

bash

Copy code

nmap -sv -a https://cms.comsats.edu.pk:8083

This scan provided detailed information about the services running on the site, their versions, and associated vulnerabilities. The results are summarized below:

* **Open Ports**:
  + Port 8083: HTTP service identified
  + Port 80: Standard HTTP service
  + Port 443: HTTPS
* **Service Versions**:
  + The site runs **ASP.NET 4.0**, **TypeForm** for form handling, and various **AJAX libraries**.
* **Vulnerabilities Detected**: The Nmap vulnerability scan, using the -sv vuln option, identified potential risks associated with **Cross-Site Scripting** (XSS), **CSRF**, and **insecure form handling** in the web application's technology stack.

## **Exploits and Mitigation:**

1. **Cross-Site Scripting (XSS) via ASP.NET**:
   * **Exploit**: Attackers inject scripts through form inputs that are then executed in users' browsers.
   * **Mitigation**: Sanitize all inputs and implement secure coding practices to neutralize potentially dangerous characters.
2. **Form Injection via TypeForm**:
   * **Exploit**: Malicious code can be inserted into unvalidated form fields, potentially exfiltrating sensitive data.
   * **Mitigation**: Implement strict input validation, encrypt data in transit, and enforce CAPTCHA to prevent automated attacks.
3. **XSS and CSRF via AJAX Libraries**:
   * **Exploit**: Unvalidated data passed to AJAX requests may be exploited for XSS or CSRF attacks.
   * **Mitigation**: Enforce CSPs and use anti-CSRF tokens to secure AJAX requests.
4. **Outdated Bootstrap Libraries**:
   * **Exploit**: Known vulnerabilities in old Bootstrap versions can be exploited for XSS attacks.
   * **Mitigation**: Update to the latest version of Bootstrap and sanitize user inputs before rendering in the DOM.
5. **DOM-based XSS via JavaScript**:
   * **Exploit**: Manipulation of untrusted inputs in JavaScript leads to the execution of malicious scripts.
   * **Mitigation**: Use proper input sanitization and encoding in all JavaScript handling dynamic content.

## **Conclusion:**

The reconnaissance of the CMS website did expose a few potential vulnerabilities in its technology stack with improper input validation and outdated libraries used. Though tools like Whois, dig, Nmap, WhatWeb, and Zaproxy do give some valuable results, more important is placing the necessary mitigation strategy in the right place that would definitely keep the website safe and away from these potential exploits: updating regularly, sanitizing inputs, and following secure coding practices.

