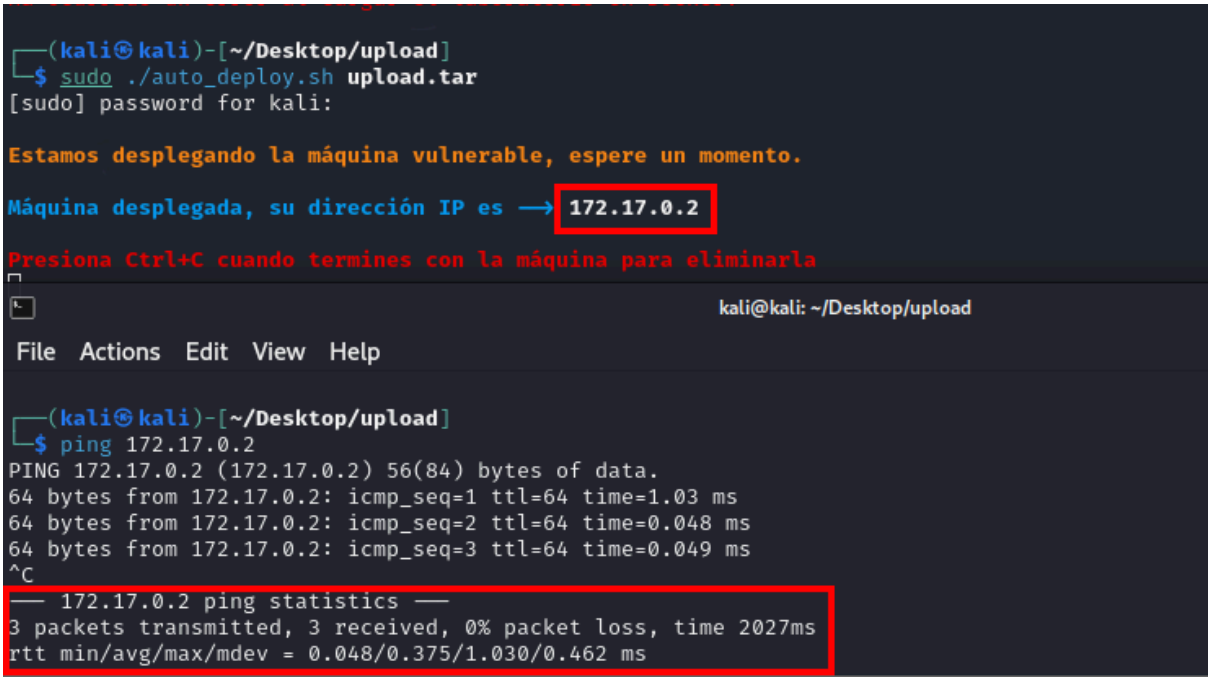
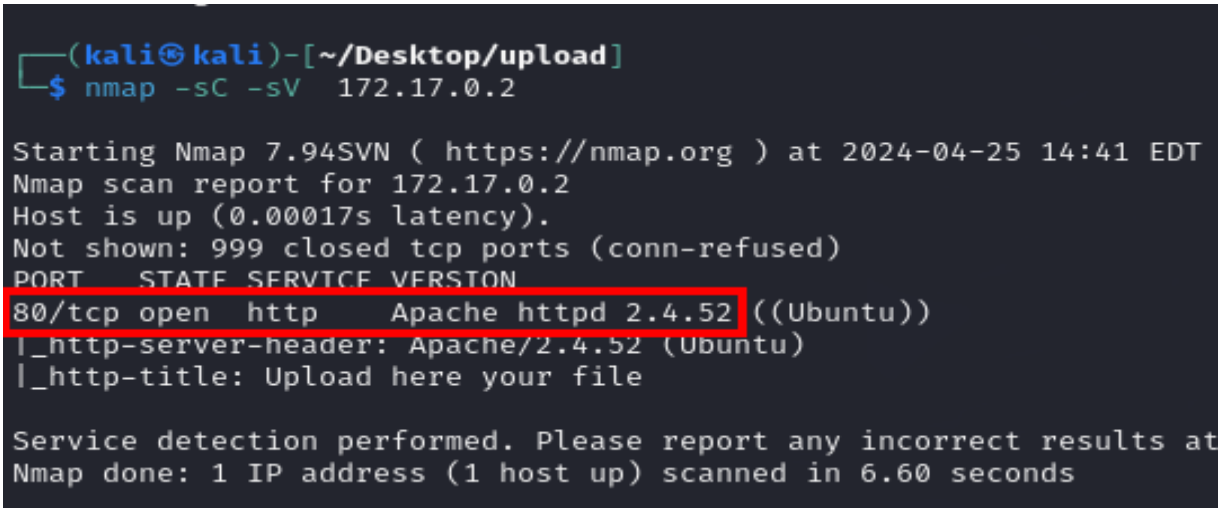
Upload

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



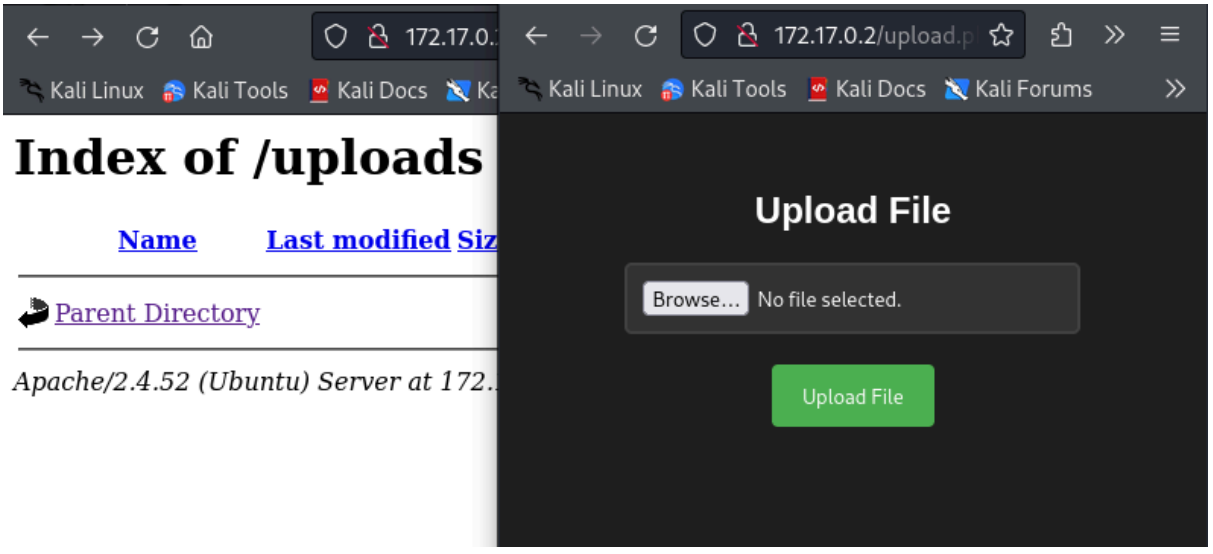
Siguiente paso será la enumeración de servicios y puertos, para ello haremos uso de nmap.



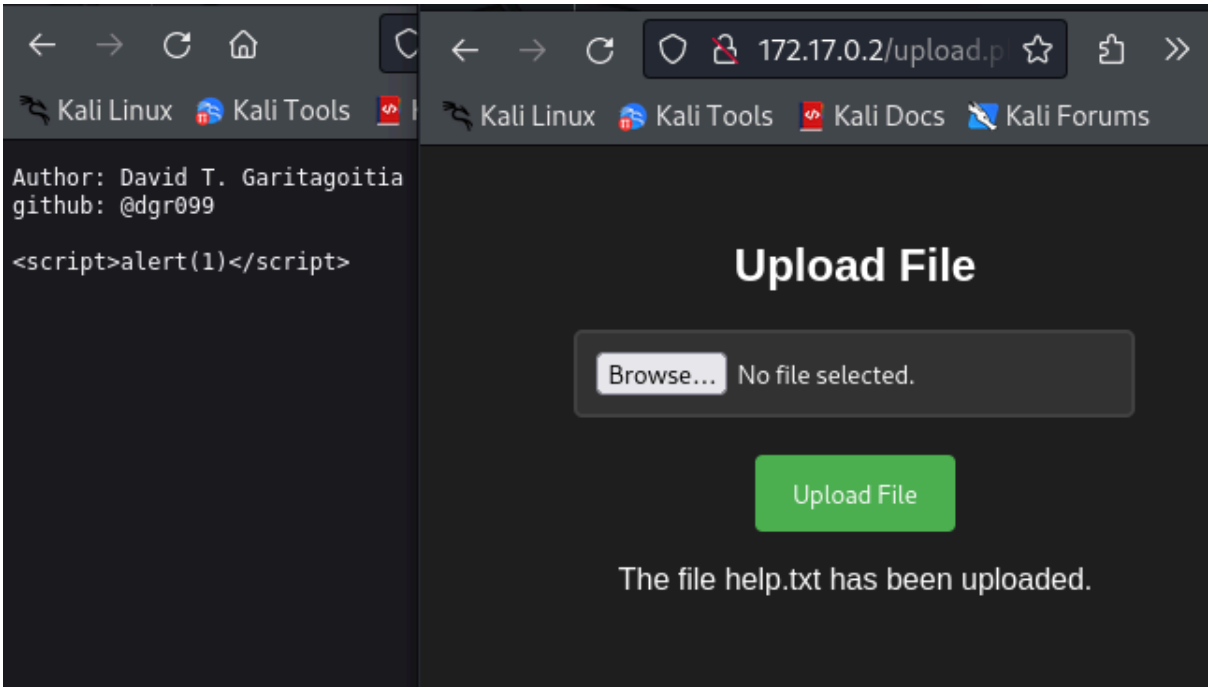
El único puerto que vemos abierto es el 80 que esta corriendo un servidor Apache http. Vamos a aplicar fuzzing sobre esta mediante gobuster.

```
┌──(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
===============================================================
```

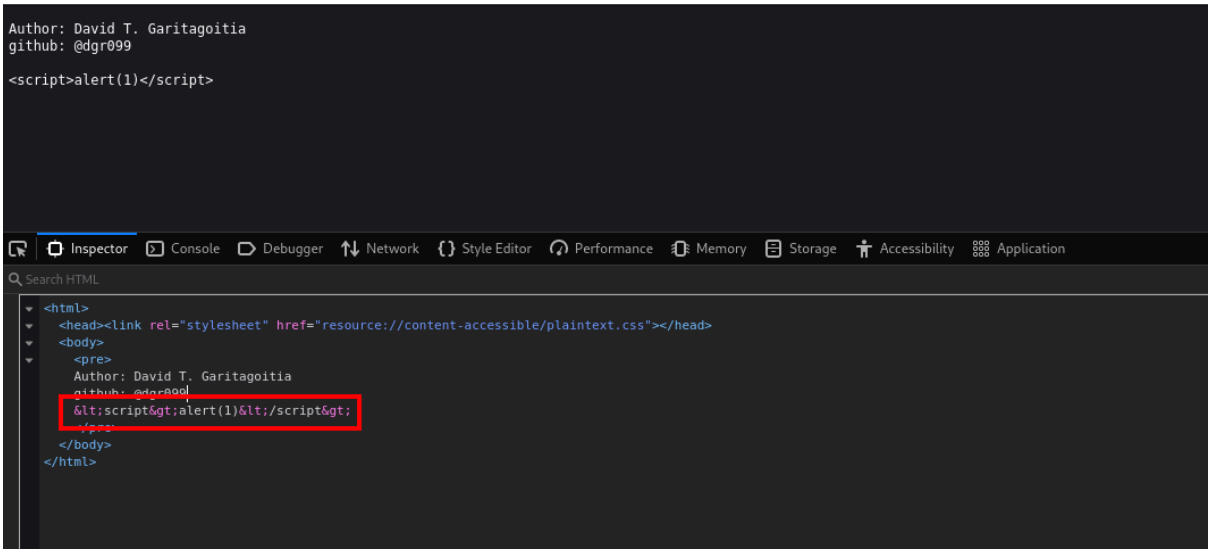
De los resultados llama la atención esos dos directorios señalizados, /uploads y /upload.php, por los nombres ya podemos intuir lo que nos podemos encontrar.



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 carácteres por si fuera posible un XSS.



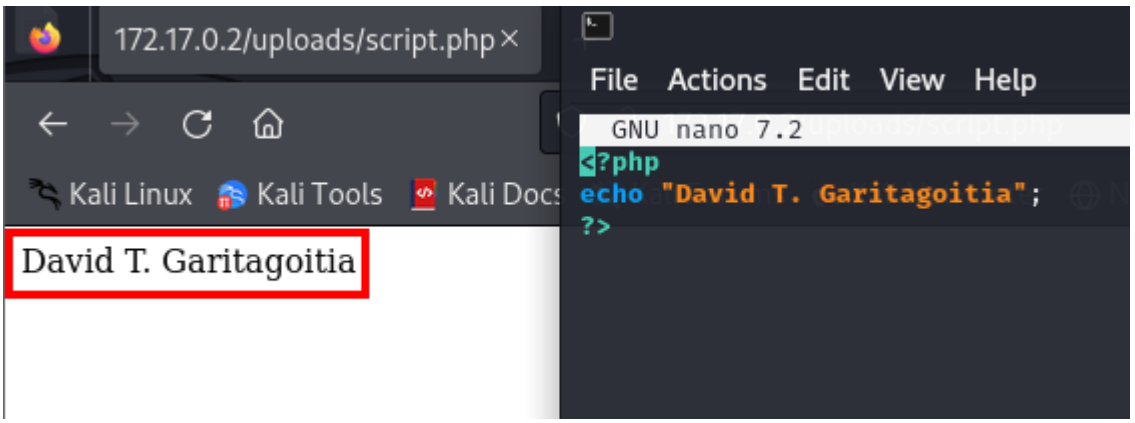
Parece que está limpiando los carácteres así que un xss no parece opción.



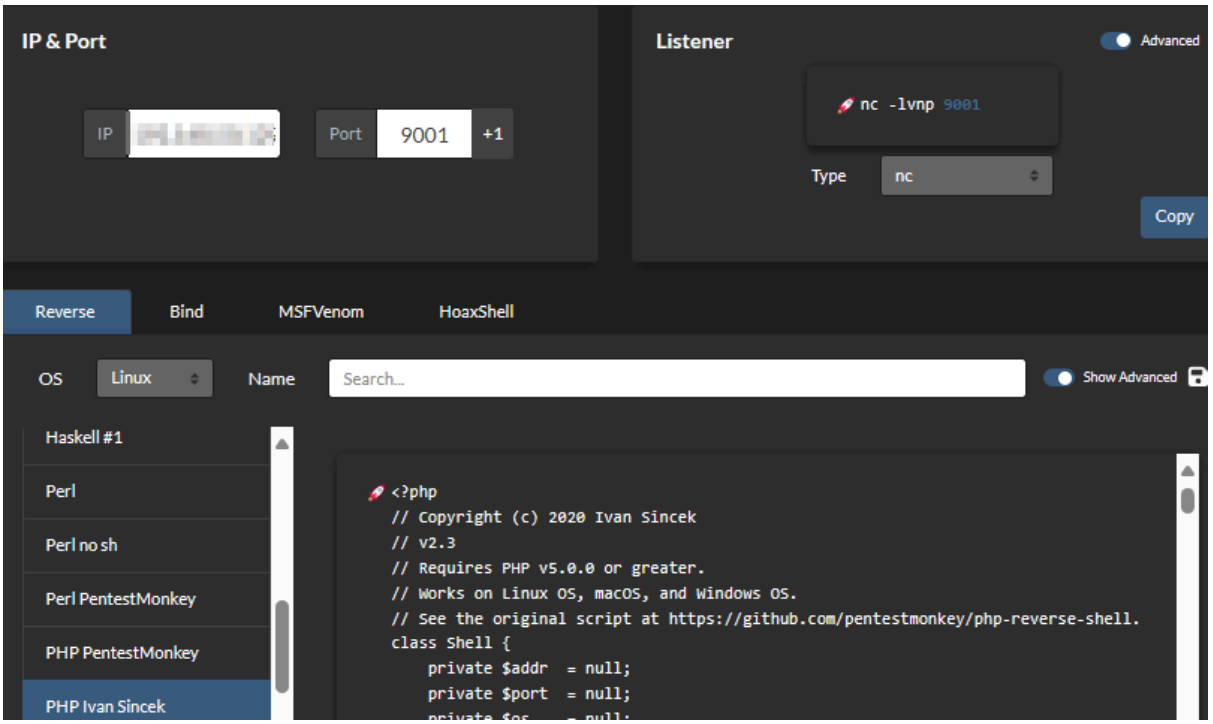
Vamos entonces con la primera opción cuando se nos permite subir archivos que es intentar lanzar código malicioso php.

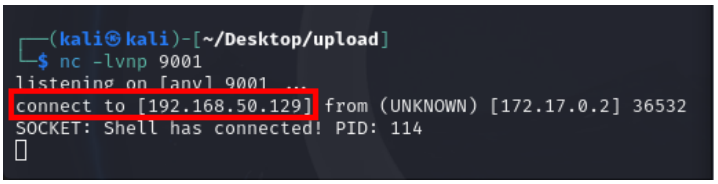
PHP (Hypertext Preprocessor) es un lenguaje de scripting del lado del servidor ampliamente utilizado para el desarrollo web. Se integra con HTML para generar contenido dinámico y páginas web interactivas.

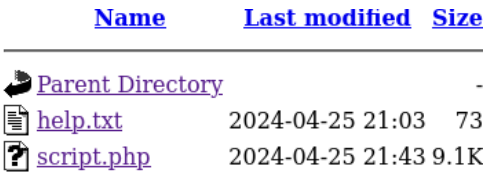
Vamos a comprobar si efectivamente somos capaces de ejecutar código php.



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







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.