Windows Exploitation using Windows API’s

Kevin Phongagsorn

Overview

- Windows API’s
- Stored Procedures
- DLL Injection
- Demo

Calling Windows API’s in C#

1) Add InteropServices namespace to allow DLL imports

```
Using System.Runtime.InteropServices;
```

2) Declare method signature

```
[DllImport(”Kernel32.dll”)]
public static extern int MessageBox(int h, string m, string c, int type);
```

3) Call method

```
MessageBox (0,"API Message Box","API Demo",0);
```

Calling Windows API's in C#

Some API's will require specific structures

```
[StructLayout(LayoutKind.Sequential)]
public struct SYSTEM_INFO {
     public uint dwOemId;
     public uint dwPageSize;
     public uint lpMinimumApplicationAddress;
     public uint lpMaximumApplicationAddress;
     public uint dwActiveProcessorMask;
     public uint dwNumberOfProcessors;
     public uint dwProcessorType;
     public uint dwAllocationGranularity;
     public uint dwProcessorLevel;
     public uint dwProcessorRevision;
}
```

Type Conversion

- Be aware of type conversion
  - Example: IntPtr == HANDLE

C#

```
IntPtr procHandle = OpenProcess(PROCESS_CREATE_THREAD |
PROCESS_QUERY_INFORMATION | PROCESS_VM_OPERATION | PROCESS_VM_WRITE |
PROCESS_VM_READ, false, targetProcess.Id);
```

C

```
HANDLE ProcessHandle;
ProcessHandle = OpenProcess(PROCESS_ALL_ACCESS, FALSE, Pid);
```

Marshal Class

- Collection of methods for allocating unmanaged memory, copying unmanaged memory blocks, and converting managed to unmanaged types, as well as other miscellaneous methods used when interacting with unmanaged code

```
[DllImport("kernel32.dll", SetLastError = true)]
[return: MarshalAs(UnmanagedType.Bool)]

public static extern bool
DeleteFileA([MarshalAs(UnmanagedType.LPStr)]string lpFileName);
```

Useful Documented API’s

- Delete File
- Create a Process
- Find Volumes
- Set File Attribute
- Get all Running Processes
- Get Process Info
- Read/Write Process Memory
- Set/Get/Load/Create/Replace/Delete/Connect WindowsRegistry Key/Value

Useful Undocumented API’s

- `NTPrivilegeCheck`
- `NTShutdownSystem`
- `NtCreateThreadEx`
- `FrostCrashedWindow`
- `IsElevationRequired`
- `DisconnectWindowsDialog`
- `SHGetUserDisplayName`
- `SHSetUserPicturePath`
- `SHUserGetPasswordHint`

Useful Documented Stored Procedures

- Add a user
  - `sp_addlogin`
  - `sp_addsrvrolemember`
- Information about a user
  - `sp_who and sp_who2`
  - `EXEC sp_who2`
- Shell
  - `exec sp_cmdshell`

Useful Undocumented Stored Procedures

- Find version of SQL Server
  - `EXECUTE sp_MSgetversion’‘’`
- Find Access Level
  - This is the example to check what kind of access the current user has in all databases:
    - `EXEC sp_MSdbuseraccess @mode = ‘db’, @qual = ‘%’`
- Drop an Object
  - This is used to drop the object for the given object id, object name, and object owner.
    - `sp_MSdrop_object [object_id] [,object_name] [,object_owner]`

Useful Undocumented Stored Procedures

- Find if some file exists on the server
    - `sp_MSexists_file ‘C:\somedirectory\something\ ‘test.exe’`
- Kill the database
    - `sp_MSkilldb dbname`
- List all fixed drives and free space
    - `exec master..xp_fixeddrives`
- List a directory structure
    - `exec master..xp_dirtree 'C:\Program Files\Microsoft SQL Server\MSSQL\’`

Useful Undocumented Stored Procedures

- Change the owner of an object
    - `EXEC sp_MSchangeobjectowner ‘sales’, ‘jdoe’`
- Check to see if a given file exists
    - `exec master..xp_fileexist ‘C:\somefile.txt'`
- Enumerate Groups
    - `exec master..xp_enumgroups`
- Execute something for all tables in the database
    - `EXEC sp_MSforeachtable @command1="print '?' DBCC DBREINDEX ('?')"`

Dynamic-Link Libraries in Windows

- A Dynamic-link library (DLL) is Microsoft's implementation of the shared library concept in the Microsoft Windows operating systems

- File formats for DLLs are the same as for Windows EXE files:
  - Portable Executable (PE) for 32-bit and 64-bit Windows

- As with EXEs, DLLs can contain code, data, and resources, in any combination

Basic DLL Injection

- DLL Injection is a way of inserting code into a running process
- Usually insert a dynamic-link library (DLL), since DLLs are meant to be loaded as needed at run time
    - However it is possible inject assembly in any other form (eg. executables, handwritten)
- Need to have an appropriate level of privileges on the system to start playing with other program's memory

DLL Injection Steps

1. Attach to the process
2. Allocate memory within the process
3. Copy the DLL or the DLL path into the processes memory; determine memory addresses
4. Instruct the process to Execute your DLL

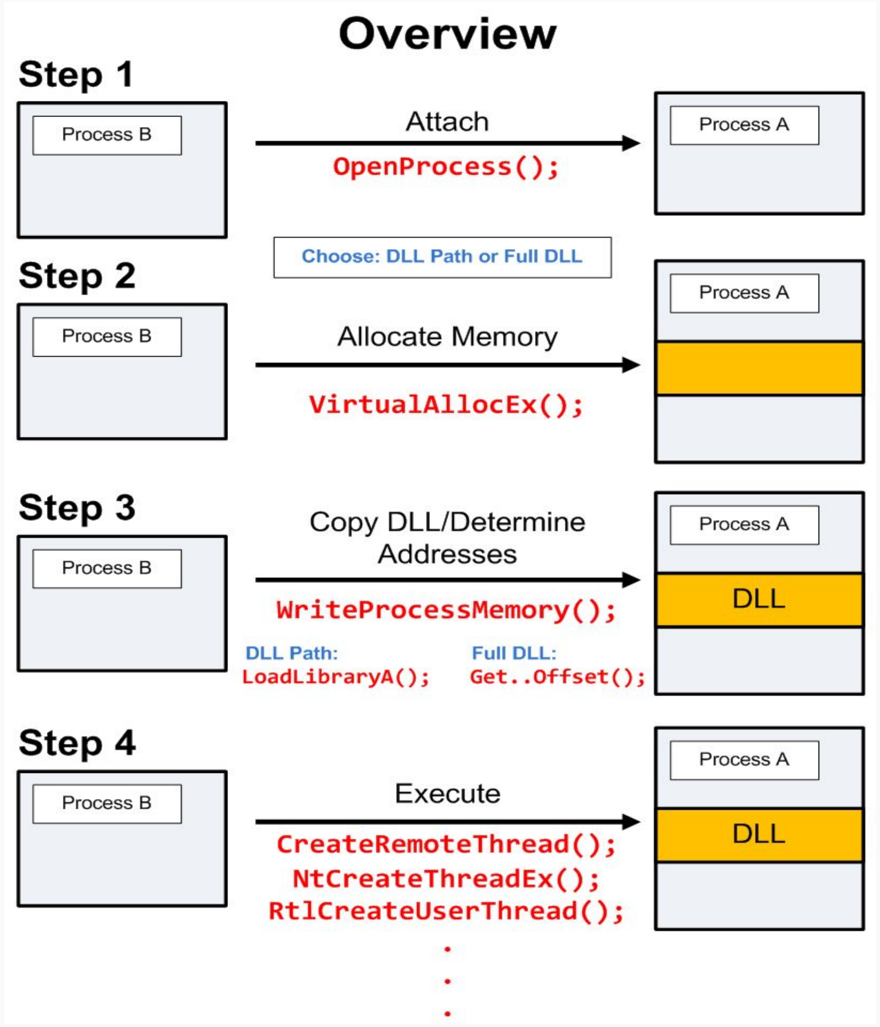
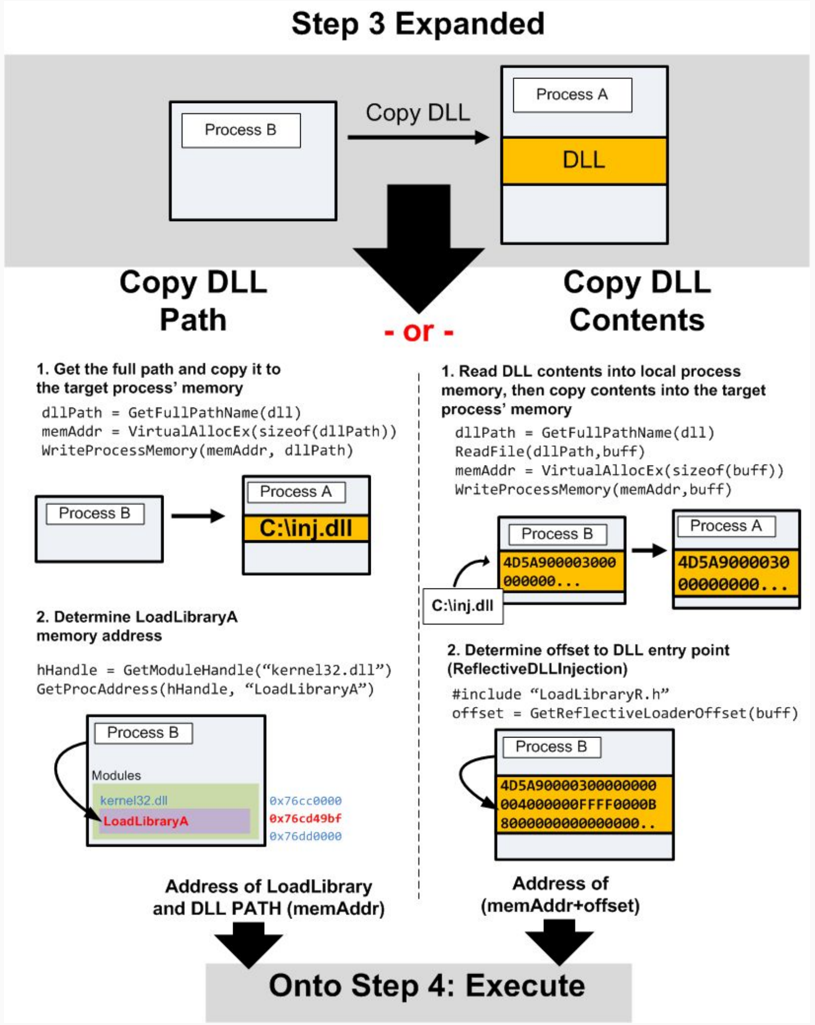


Image source: http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html?m=1

DLL Starting Points

- LoadLibraryA()
  Kernel32.dll function used to load DLLs, executables, and other supporting libraries at run time

- Jumping to DllMain
  Load the entire DLL into memory (Reflective DLL Injection), then determine the offset to the DLL's entry point



1. Get the full path and copy it to the target process' memory

```
dllPath = GetFullPathName(dll)
memAddr = VirtualAllocEx(sizeof(dllPath))
WriteProcessMemory(memAddr, dllPath)
```

2. Determine LoadLibraryA memory address

```
hHandle = GetModuleHandle("kernel32.dll")
GetProcAddress(hHandle, "LoadLibraryA")
```

1. Read DLL contents into local process memory, then copy contents into the target process' memory

```
dllPath = GetFullPathName(dll)
ReadFile(dllPath,buff)
memAddr = VirtualAllocEx(sizeof(buff))
WriteProcessMemory(memAddr,buff)
```

2. Determine offset to DLL entry point (ReflectiveDLLInjection)

```
#include "LoadLibraryR.h"
offset = GetReflectiveLoaderOffset(buff)
```

Image source: http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html?m=1

LoadLibraryA() vs Jumping to DllMain

- LoadLibraryA()
  - Registers the loaded DLL with the program and thus can be easily detected
  - After DLL has already been loaded and executed once, it will not execute again
  - Loads DLL path to process' memory

- Jumping to DllMain
  - Loads entire DLL in process' memory
  - Avoid registering the DLL with the program (stealthy)
  - Repeatedly inject into a process

Examples of DLL Injection

- **Putter Panda** injects specified DLL into a process that would normally be accessing the network, including Outlook Express (msinm.exe), Outlook (outlook.exe), Internet Explorer (iexplore.exe), and Firefox (firefox.exe)
- **Backdoor.Oldrea** injects itself into explorer.exe
- **BlackEnergy** injects its DLL component into svchost.exe
- **Cobalt Strike** can inject a variety of payloads into processes dynamically chosen by the adversary
- **Sykipot** injects itself into running instances of outlook.exe, iexplore.exe, or firefox.exe

Examples of DLL Injection

- **Duqu** will inject itself into different processes to evade detection. (Duqu will inject into different processes depending on which security suite is installed on the infected host)
- **Elise** injects DLL files into iexplore.exe
- **Emissary** injects its DLL file into a newly spawned Internet Explorer process
- **HIDEDRV** injects a DLL for Downdelph into the explorer.exe process
- **JHUHUGIT** performs code injection injecting its own functions to browser processes

Other Process Injection Techniques

- Process Hollowing
- Thread Execution Hijacking
- Hook Injection via SetWindowsHookEx
- Injection and Persistence via Registry Modification
  - AppInit_DLLs
  - AppCertDLLs
- APC Injection and Atom bombing

Demo Time!

References

- https://media.defcon.org/DEF%20CON%2025/DEF%20CON%2025%20workshops/DEFCON-25-Workshop-Chuck-Easttom-Windows-The-Undiscovered-Country.pdf
- https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-trending-process
- http://blog.opensecurityresearch.com/2013/01/windows-dll-injection-basics.html?m=1
- https://attack.mitre.org/wiki/Technique/T1055
- https://msdn.microsoft.com/en-us/library/system.runtime.interopservices.marshal(v=vs.110).aspx
- http://www.codingvision.net/miscellaneous/c-inject-a-dll-into-a-process-w-createremotethread
- https://www.youtube.com/watch?v=C2OtYr0EyOg
- http://winapi.systemoverflow.com/?page_id=138
- https://en.wikipedia.org/wiki/Dynamic-link_library
- https://en.wikipedia.org/wiki/Microsoft_Windows_library_files