Following Malware Execution in IDA or Ghidra

Following Malware Execution

- The Ghidra decompilation view and control flow graphs are very useful for analyzing the malware's possible execution paths
  - Function calls, loops, if statements, etc.
- But execution can change in ways other than jumps and calls
- Often need to find out how the malware is executing different areas of code

DLLs

DLL review

- Dynamic Link Library
- Exports functions for other executables to use
- Advantage: can be shared among running processes, saving memory

How Malware Uses DLLs

- By storing malicious code
  - May export functions to other malware files
  - May be loaded into another process
- By using Windows DLLs
  - To interact with the operating system via Windows API functions
- By using third-party DLLs
  - To interact with other non-Windows programs
  - To use a library that may not be on the victim's machine

Analyzing DLLs

- DLLs have many points from which code can be executed
  - Each exported function
  - DllMain
- DllMain is called whenever a process loads or unloads the DLL
  - Normally used for managing any resources specific to a process, but malware sometimes uses it for other purposes

Processes

Process Review

- Process – program in execution

- Used to keep programs from interfering with each other
  - Each process has a separate address space (whereas threads share address space)
  - OS manages how processes access shared resources (CPU,RAM, filesystem, hardware, etc)

Creating a Process

- The CreateProcess function is typically used to create a process

- Has many parameters, gives caller a large amount of control over how the process is created
  - How many parameters? See here

Running an Embedded Executable

- Malware contains an executable as a resource
- Uses FindResource, LoadResource, CreateFile, etc to write resource to disk
- Uses CreateProcess to run the resource

Creating a Remote Shell

- Remote shell – allows an attacker to run commands on the victim's computer remotely

- Can create a remote shell by opening a socket and using a single call to CreateProcess!

Creating a Remote Shell

- Need to pass specific arguments to CreateProcess
  - The lpStartupInfo parameter points to a STARTUPINFO struct
  - This struct contains handles to stdin, stdout, and stderr
  - Point stdin, stdout, and stderr to the socket
  - Call CreateProcess

- All input from the malware actor over the socket is run on the command line

Creating a Remote Shell – Sample Code

```
004010DA  mov     eax, dword ptr [esp+58h+SocketHandle]
004010DE  lea     edx, [esp+58h+StartupInfo]
004010E2  push    ecx             ; lpProcessInformation
004010E3  push    edx             ; lpStartupInfo
004010E4 ❶mov     [esp+60h+StartupInfo.hStdError], eax
004010E8 ❷mov     [esp+60h+StartupInfo.hStdOutput], eax
004010EC ❸mov     [esp+60h+StartupInfo.hStdInput], eax
004010F0 ❹mov     eax, dword_403098
004010F5  push    0               ; lpCurrentDirectory
004010F7  push    0               ; lpEnvironment
004010F9  push    0               ; dwCreationFlags
004010FB  mov     dword ptr [esp+6Ch+CommandLine], eax
004010FF  push    1               ; bInheritHandles
00401101  push    0               ; lpThreadAttributes
00401103  lea     eax, [esp+74h+CommandLine]
00401107  push    0               ; lpProcessAttributes
00401109 ❺push    eax             ; lpCommandLine
0040110A  push    0               ; lpApplicationName
0040110C  mov     [esp+80h+StartupInfo.dwFlags], 101h
00401114 ❻call    ds:CreateProcessA
```

Process Injection

- Malware can inject its own code into a different process
- Typically performed using the VirtualAlloc, WriteProcessMemory, and CreateRemoteThread API calls
- Will cover this and other covert launching techniques later

Threads

Thread Review

- Thread – sequence of instructions belonging to a process that is executed by the CPU

- Each process contains one or more threads
  - All threads share the process's memory space
  - Each thread has its own values for registers and the stack
    - Storing and restoring these is the substance of a "context switch"

Creating a Thread

- Done using the CreateThread function
  - Takes lpStartAddress, a pointer to a function
  - Also takes lpParameter, a single parameter to the function
  - The thread executes the function until it returns

Covertly Loading a Malicious Library

- Can use CreateThread to covertly load a malicious library into a process
- Need to set certain parameters to CreateThread
  - Pass the address of the LoadLibrary Windows API function as the lpStartAddress parameter
  - Pass the name of the desired library as lpParameter
- Even more stealthy if “LoadLibrary” and the name of the library are obfuscated – which is easy to do

Services

Services Review

- Service – a task that runs in the background without an associated process or thread or user

- Managed by the Windows Service Manager

Why Malware Uses Services

- Can be set to automatically run when the computer boots
  - Gives persistence
- Often run with SYSTEM privileges
  - But need admin to specify this

Creating / Starting a Service

- OpenSCManager – Returns a handle to the service control manager, which is needed for all other service-related API calls
- CreateService – Adds a new service to the service control manager
  - Can specify that the service automatically runs at boot
- StartService – Starts a service manually

Types of Services

- WIN32_SHARE_PROCESS – Stores code for a service in a DLL, run by svchost.exe
- WIN32_OWN_PROCESS – Stores code in an EXE, runs as an independent process
- KERNEL_DRIVER – Used for loading code into the kernel

Exceptions

Exceptions Review

- Exception – allows a program to handle events outside its normal execution path

- Can be triggered by:
  - Errors (such as a divide by 0)
  - Hardware (such as invalid memory access)
  - Manual call to RaiseException

Structured Exception Handling

- Structured Exception Handling (SEH) – Windows mechanism for handling exceptions
  - List of functions for handling exceptions
  - Each function can handle the exception or pass it to the next handler
  - If an exception makes it to the end of the list without being handled, it is considered an unhandled exception and crashes the process

How Malware Uses Exceptions

- The SEH is a type of flow control that can't be followed by disassemblers and can fool debuggers

- Malware can add its own custom exception handler to the SEH and then use trigger an exception to transfer execution to the handler