Introductory Code Instrumentation

Aka: How to be a Dirty Hooker

Overview

- What??
- Why?
- Where?
- How?
- Case Study: Logging SSL traffic in IE :-)

What??

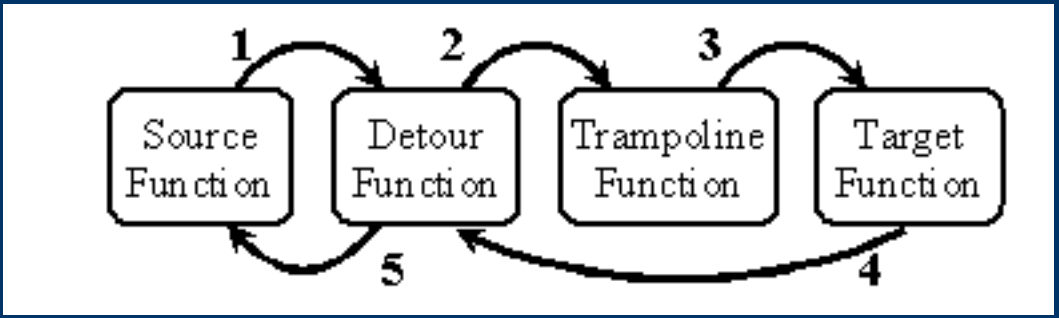
- Instrumentation = Hooking arbitrary function calls
- Idea is to have wrapper functions get called, do some work, and then call the original function
  - DLL functions
  - Random program functions
- Several ways to do this

Why?

- Reverse Engineering
  - Discover function arguments/usage
  - Fuzzing
  - Intercepting encrypted communications
  - Re-implementing select functions
- Malware
  - Intercepting encrypted communications ;-)
  - Covert Channels
  - Remote control

Where

- DLL Function Imports
  - Resolved by ntdll
  - IAT matched to EAT
  - PEView Demo/Explanation
- Direct Binary Patching
  - Not everything uses imports
  - Insert a jmp and create a trampoline buffer with orig instructions



How: Setup

- Need code in the target process
- DLL injection!
  - HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs
  - Detours: CreateProcessWithDll
  - CreateRemoteThread
    - OpenProcess
    - namePtr = VirtualAllocEx
    - WriteProcessMemory(namePtr, “Dllname.dll”)
    - CreateRemoteThread(LoadLibrary, namePtr)

How: The IAT way

- Find target function address with GetProcAddress
- Enumerate all DLLs with Module32First/Next
- Walk IAT and edit entries of each DLL
  - DLL “Handle” is IMAGE_DOS_HEADER
  - Get IMAGE_NT_HEADERS from dosh->e_lfanew
  - Get Import Directory from NT header
  - Translate RVA to VA to get IAT pointer
  - Walk IAT looking for your OrigFunction
  - VirtualProtect and modify

How: The Detour Way

- Need an x86 disassembler/length decoder
- Or just use Microsoft's Detours
  - 2.0 and above mark your target with a 'detoured.dll' :-(

```
DetourTransactionBegin();
DetourUpdateThread(GetCurrentThread());
DetourAttach(&(PVOID&)OrigFcn, NewFcn);
DetourAttach(&(PVOID&)OrigFcn, NewFcn);
DetourTransactionCommit();
```

How: Calling Conventions

- Calling convention must match original!
- WINAPI/stdcall
  - Most common. Used in all windows APIs
  - Called function cleans up stack. Look for ret N's or add esp, N at end.
- cdecl
  - Almost never used in windows
  - Caller cleans up stack, look for add esp, N after cals
- Fastcall
  - Args passed in registers
- C++
  - ecx containsthe *this* pointer

Case Study: Background Investigation

- Want to hijack SSL. Where does SSL live?
- Reverse Engineering isn't all heroics: Use google!
  - SSL lives in the SSPI, mostly implemented in secur32.dll
- Check exports of secur32, check MSDN for interesting ones.
- EncryptMessage/DecryptMessage!

Case Study: Digging in

- Peview iexplore.exe
  - Hrmm no import of secur32.dll... Maybe recursive?
- Check Depends: “Delay Loading”..
  - Means ntdll does LoadLibrary/GetProcAddress
- Options:
  - Hook GetProcAddress and intercept? But what if we don't inject early enough...
  - Detour!

Covering Your Tracks

- IAT modifications are easy to detect
  - Programs can scan own IAT to verify
  - Gets tricky since GetProcAddress uses IAT, but still doable
- Detours much less easy..
  - Compiled code can change with optimizations, etc
- Injected DLLs are detectable
  - Module list
  - Memory contents
  - But, lots of apps install own hook DLLs everywhere
- Detours has 'marker' DLL :-(

Advanced Issues

- Executable compression
  - Does not effect detours, but can make analysis difficult
  - Use PEiD to discover type, google for its unpacker
- PE corruption
  - LordPE, PE Tools