

Win at Reversing

API Tracing and Sandboxing through
Inline Hooking

Nick Harbour





Agenda

- Reverse Engineering Primer
- Approaches to Dynamic Analysis
- Inline Hooks
- Advantages Over Other Techniques
- Usages

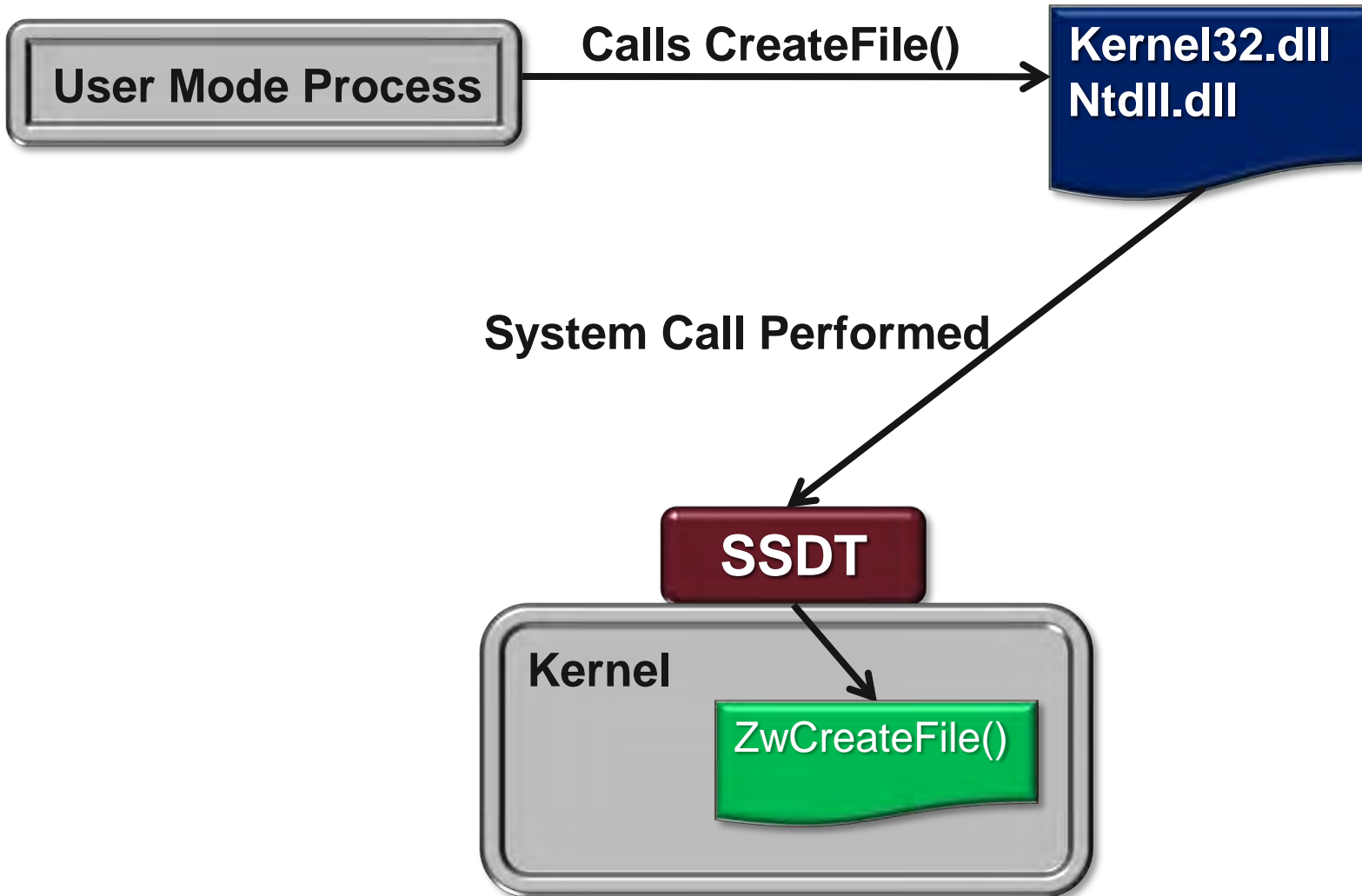
Reverse Engineering Primer

- Reverse Engineering techniques can be divided into two categories: Static and Dynamic Analysis
- Static Analysis
 - Techniques which do not involve running the code
 - Disassembly, file structure analysis, strings, etc.
- Dynamic Analysis
 - Techniques which involve running the code
 - Behavioral analysis

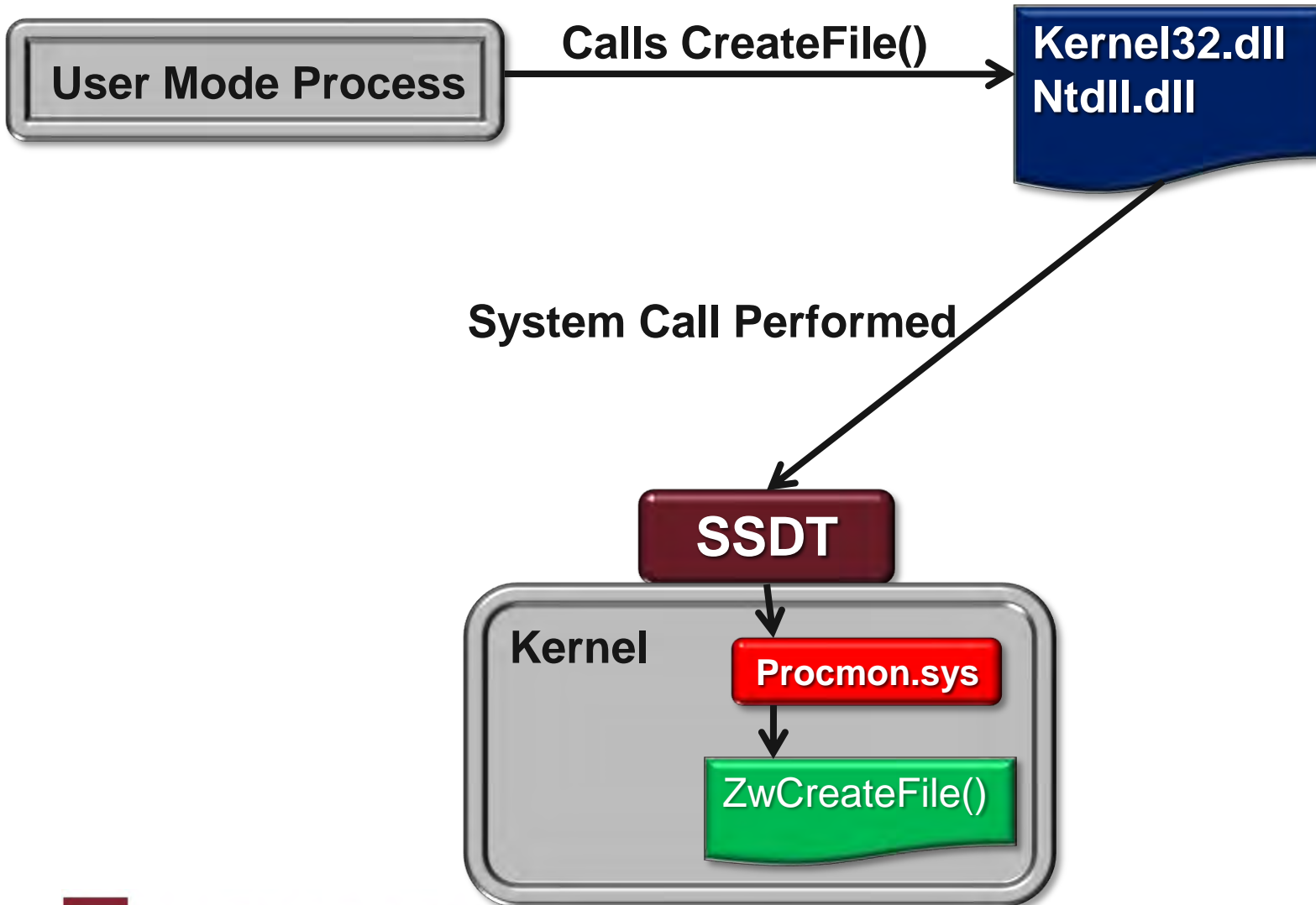
Approaches to Dynamic Analysis

- Network Monitoring
 - Isolated Physical Networks
 - Virtual Networks
- Hardware Emulation
 - Norman Sandbox et al.
- Kernel-Level Monitoring (SSDT hooks)
 - Sysinternals' Process Monitor
- Debuggers

Kernel-Level Monitoring



Kernel-Level Monitoring



Kernel-Level Monitoring

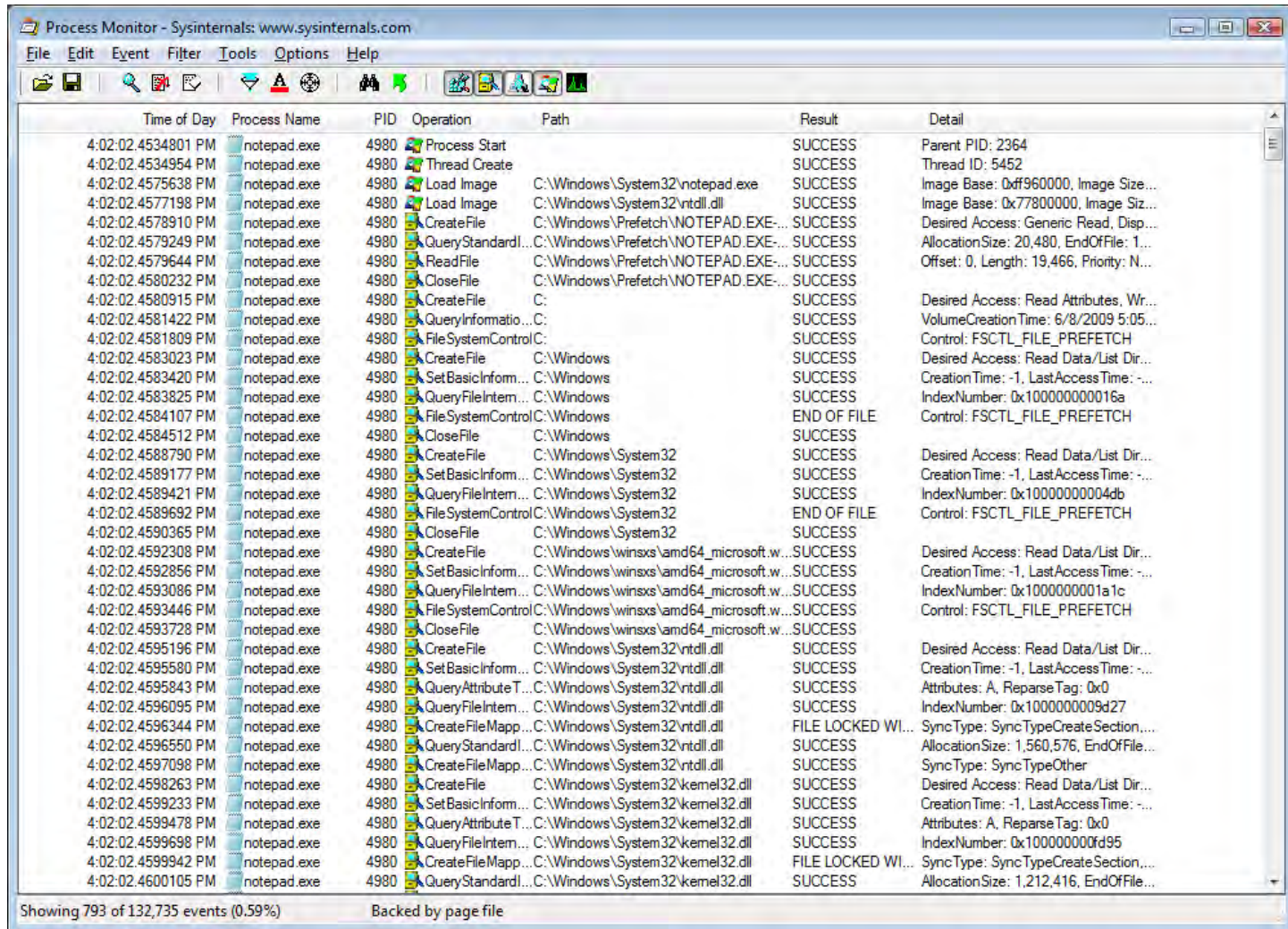
- Advantages

- Captures every system call
- Can't be avoided from userland

- Disadvantages

- Only captures functions implemented as system calls
- Not every important function call in the Win32 API is implemented as a system call
- Tools don't differentiate between process housekeeping and calls from usercode
- Calls to internal DLL's cannot be observed

Process Monitor



Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time of Day	Process Name	PID	Operation	Path	Result	Detail
4:02:02.4534801 PM	notepad.exe	4980	Process Start		SUCCESS	Parent PID: 2364
4:02:02.4534954 PM	notepad.exe	4980	Thread Create		SUCCESS	Thread ID: 5452
4:02:02.4575638 PM	notepad.exe	4980	Load Image	C:\Windows\System32\notepad.exe	SUCCESS	Image Base: 0xff960000, Image Size...
4:02:02.4577198 PM	notepad.exe	4980	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x77800000, Image Size...
4:02:02.4578910 PM	notepad.exe	4980	CreateFile	C:\Windows\Prefetch\NOTEPAD.EXE-...	SUCCESS	Desired Access: Generic Read, Disp...
4:02:02.4579249 PM	notepad.exe	4980	QueryStandardI...	C:\Windows\Prefetch\NOTEPAD.EXE-...	SUCCESS	AllocationSize: 20,480, EndOfFile: 1...
4:02:02.4579644 PM	notepad.exe	4980	ReadFile	C:\Windows\Prefetch\NOTEPAD.EXE-...	SUCCESS	Offset: 0, Length: 19,466, Priority: N...
4:02:02.4580232 PM	notepad.exe	4980	CloseFile	C:\Windows\Prefetch\NOTEPAD.EXE-...	SUCCESS	
4:02:02.4580915 PM	notepad.exe	4980	CreateFile	C:	SUCCESS	Desired Access: Read Attributes, Wr...
4:02:02.4581422 PM	notepad.exe	4980	QueryInformatio...	C:	SUCCESS	VolumeCreationTime: 6/8/2009 5:05...
4:02:02.4581809 PM	notepad.exe	4980	FileSystemControl	C:	SUCCESS	Control: FSCTL_FILE_PREFETCH
4:02:02.4583023 PM	notepad.exe	4980	CreateFile	C:\Windows	SUCCESS	Desired Access: Read Data/List Dir...
4:02:02.4583420 PM	notepad.exe	4980	SetBasicInform...	C:\Windows	SUCCESS	CreationTime: -1, LastAccessTime: ...
4:02:02.4583825 PM	notepad.exe	4980	QueryFileIntem...	C:\Windows	SUCCESS	IndexNumber: 0x100000000016a
4:02:02.4584107 PM	notepad.exe	4980	FileSystemControl	C:\Windows	END OF FILE	Control: FSCTL_FILE_PREFETCH
4:02:02.4584512 PM	notepad.exe	4980	CloseFile	C:\Windows	SUCCESS	
4:02:02.4588790 PM	notepad.exe	4980	CreateFile	C:\Windows\System32	SUCCESS	Desired Access: Read Data/List Dir...
4:02:02.4589177 PM	notepad.exe	4980	SetBasicInform...	C:\Windows\System32	SUCCESS	CreationTime: -1, LastAccessTime: ...
4:02:02.4589421 PM	notepad.exe	4980	QueryFileIntem...	C:\Windows\System32	SUCCESS	IndexNumber: 0x10000000004db
4:02:02.4589692 PM	notepad.exe	4980	FileSystemControl	C:\Windows\System32	END OF FILE	Control: FSCTL_FILE_PREFETCH
4:02:02.4590365 PM	notepad.exe	4980	CloseFile	C:\Windows\System32	SUCCESS	
4:02:02.4592308 PM	notepad.exe	4980	CreateFile	C:\Windows\winsxs\amd64_microsoft.w...	SUCCESS	Desired Access: Read Data/List Dir...
4:02:02.4592856 PM	notepad.exe	4980	SetBasicInform...	C:\Windows\winsxs\amd64_microsoft.w...	SUCCESS	CreationTime: -1, LastAccessTime: ...
4:02:02.4593086 PM	notepad.exe	4980	QueryFileIntem...	C:\Windows\winsxs\amd64_microsoft.w...	SUCCESS	IndexNumber: 0x10000000001a1c
4:02:02.4593446 PM	notepad.exe	4980	FileSystemControl	C:\Windows\winsxs\amd64_microsoft.w...	SUCCESS	Control: FSCTL_FILE_PREFETCH
4:02:02.4593728 PM	notepad.exe	4980	CloseFile	C:\Windows\winsxs\amd64_microsoft.w...	SUCCESS	
4:02:02.4595196 PM	notepad.exe	4980	CreateFile	C:\Windows\System32\ntdll.dll	SUCCESS	Desired Access: Read Data/List Dir...
4:02:02.4595580 PM	notepad.exe	4980	SetBasicInform...	C:\Windows\System32\ntdll.dll	SUCCESS	CreationTime: -1, LastAccessTime: ...
4:02:02.4595843 PM	notepad.exe	4980	QueryAttributeT...	C:\Windows\System32\ntdll.dll	SUCCESS	Attributes: A, Reparse Tag: 0x0
4:02:02.4596095 PM	notepad.exe	4980	QueryFileIntem...	C:\Windows\System32\ntdll.dll	SUCCESS	IndexNumber: 0x10000000009d27
4:02:02.4596344 PM	notepad.exe	4980	CreateFileMapp...	C:\Windows\System32\ntdll.dll	FILE LOCKED WI...	SyncType: SyncTypeCreateSection...
4:02:02.4596550 PM	notepad.exe	4980	QueryStandardI...	C:\Windows\System32\ntdll.dll	SUCCESS	AllocationSize: 1,560,576, EndOfFile...
4:02:02.4597098 PM	notepad.exe	4980	CreateFileMapp...	C:\Windows\System32\ntdll.dll	SUCCESS	SyncType: SyncTypeOther
4:02:02.4598263 PM	notepad.exe	4980	CreateFile	C:\Windows\System32\kernel32.dll	SUCCESS	Desired Access: Read Data/List Dir...
4:02:02.4599233 PM	notepad.exe	4980	SetBasicInform...	C:\Windows\System32\kernel32.dll	SUCCESS	CreationTime: -1, LastAccessTime: ...
4:02:02.4599478 PM	notepad.exe	4980	QueryAttributeT...	C:\Windows\System32\kernel32.dll	SUCCESS	Attributes: A, Reparse Tag: 0x0
4:02:02.4599698 PM	notepad.exe	4980	QueryFileIntem...	C:\Windows\System32\kernel32.dll	SUCCESS	IndexNumber: 0x100000000fd95
4:02:02.4599942 PM	notepad.exe	4980	CreateFileMapp...	C:\Windows\System32\kernel32.dll	FILE LOCKED WI...	SyncType: SyncTypeCreateSection...
4:02:02.4600105 PM	notepad.exe	4980	QueryStandardI...	C:\Windows\System32\kernel32.dll	SUCCESS	AllocationSize: 1,212,416, EndOfFile...

Showing 793 of 132,735 events (0.59%) Backed by page file

Process Monitoring via Debugging

■ Advantages

- Debugger can trap any function call, not just system calls
- Trapped calls are more likely to be highly relevant to the program's operation

■ Disadvantages

- Have to act as a debugger
- Susceptible to countless anti-debugger techniques

Inline Hooks

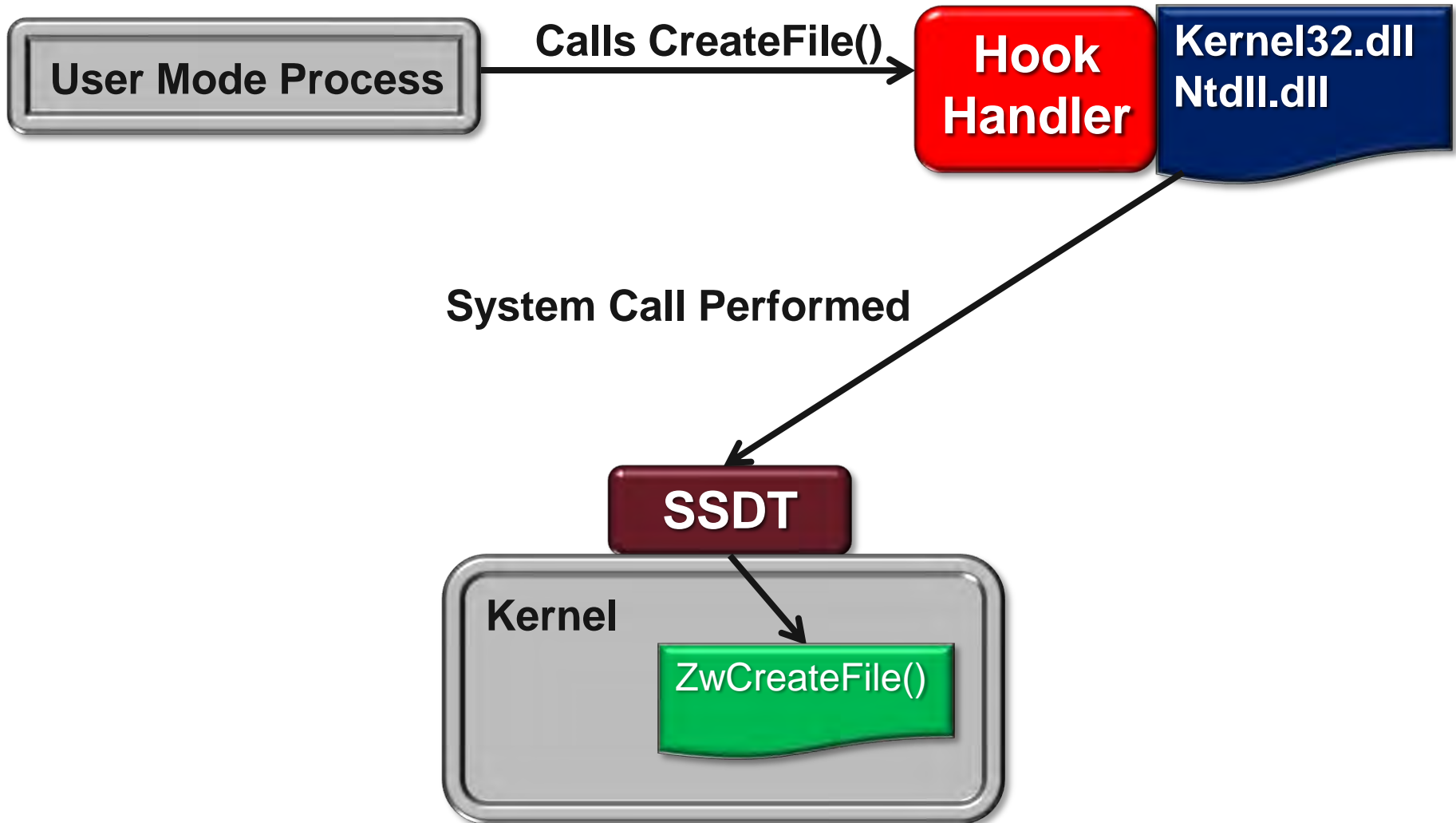
■ Advantages

- Can trap any function call, not just system calls
- Trapped calls are more likely to be highly relevant to the program's operation
- Not operating as a debugger
- No device driver required

■ Disadvantages

- More of a pain in the #@! to implement

Monitoring with Inline Hooks





Implementing Inline Hooks

1. Find a function of interest
2. Disassemble the beginning of the function
3. If possible, overwrite the beginning bytes of the function with a jump or call instruction
4. Implement a handler for the hooked function

Why Disassemble?

- If you attempt to hook every function from a DLL, for example, you might run into a function such as the one below
- Inserting a 5 byte jump or call would write beyond the end of the function. ☹️

somefunction:

```
31 C0    xor eax, eax
C3      retn
```

A Successful Hook Install

`original_function:`

```
55          push  ebp
89 E5      mov   ebp, esp
81 EC 18 00 00 00  sub  esp, 24
31 C9      xor   ecx, ecx
...
```

`hooked_function:`

```
E9 E4 7C FF FF  jmp  <handler>
18 00 00 00      ;unused
31 C9      xor   ecx, ecx
```

What to do with hooked functions.

- Observe and Report
 - Collect data about the current function call by gathering data from stack and report to console
 - Execute any instructions overwritten from the hook
 - Jump back to the next instruction in the hooked function
- Intercept and Emulate
 - Perform a specified action *Instead* of calling the intended function

Roll-your-own Sandbox

- Trap `gethostbyname()` to always return a fixed IP address.
- A pseudo-handle interface to allow fake reads and writes to files and network sockets.
 - Trap `connect()` to connection to a pseudo-socket.
 - `CreateFile()`, `ReadFile()`, `WriteFile()`, `MapViewOfFile()`...

API Thief

- Launches target process in a suspended state
- Injects a DLL into the process.
- The Injected DLL hooks all Win32 API functions before the target process is resumed
- API Call monitoring can be used simply with a process monitor-style console
- Imbedded python can be used to write custom handlers for specific hooked functions
- Obtain API Thief at www.mandiant.com

API Thief Demonstration

- Basic Process Monitoring
- Basic Interception (gethostbyname)
- Pseudo-Handles demonstration
- Automated Unpacking with API Thief

Questions?

nick.harbour@mandiant.com

nickharbour@gmail.com

