Blackout: What Really Happened

Jamie Butler and Kris Kendall





Outline

- Code Injection Basics
- User Mode Injection Techniques
- Example Malware Implementations
- Kernel Mode Injection Techniques
- Advanced Code Injection Detection via Raw Memory Analysis



Code Injection Basics

- "Code Injection" refers to techniques used to run code in the context of an existing process
- Motivation:
 - Evasion: Hiding from automated or human detection of malicious code
 - IR personnel hunt for malicious processes
 - Impersonation: Bypassing restrictions enforced on a process level
 - Windows Firewall, etc
 - Pwdump, Sam Juicer





User Mode Injection Techniques

Techniques

- Windows API
- AppInit_DII
- Detours



Injecting code via the Windows API

- Somewhat surprisingly, the Windows API provides everything you need for process injection
- Functions:
 - VirtualAllocEx()
 - WriteProcessMemory()
 - CreateRemoteThread()
 - GetThreadContext() / SetThreadContext()
 - SetWindowsHookEx()



1. OpenProcess





2. VirtualAllocEx





3. WriteProcessMemory





4. CreateRemoteThread





#Inject an infinite loop into a running process

```
import pydbq
k32 = pydbq.kernel32
payload = `\xEB\xFE'
pid = int(args[0])
h = k32.0penProcess(PROCESS ALL ACCESS, \
                      False, pid)
m = k32.VirtualAllocEx(h, None, 1024, \
                         MEM COMMIT, \setminus
                         PAGE EXECUTE READWRITE)
k32.WriteProcessMemory(h, m, payload, \setminus
                         len(payload), None)
k32.CreateRemoteThread(h, None, 1024000,
                         m, None, 0, None)
                                                10
```

Better Payloads

- Breakpoints and Loops are fun, but what about real payloads?
- If we directly inject code it must be "position independent"
- Any addresses that were pre-calculated at compile time would be wrong in the context of a new process



Better Payloads

- Building large position independent payloads is possible, but not trivial
- However, DLL injection is much simpler
- DLLs are designed to be loaded in a variety of processes, addresses are automatically fixed up when the DLL is loaded



DLL Injection

- Use the basic process we just described
- DLLs are loaded using kernel32!LoadLibrary
- kernel32 is at the same address in every process
 → we know its address in the remote process (ignoring ASLR)
- Allocate space for the name of the DLL to be loaded, then create a thread with a start address that points to LoadLibrary



#DLL Injection Excerpt

```
import pydbq
k32 = pydbg.kernel32
pid = int(args[0])
dllname = args[1]
h = k32.0penProcess(PROCESS ALL ACCESS, \
                      False, pid)
m = k32.VirtualAllocEx(h, None, 1024, \
                         MEM COMMIT, \setminus
                         PAGE EXECUTE READWRITE)
k32.WriteProcessMemory(h, m, dllname, \setminus
                         len(dllname), None)
k32.CreateRemoteThread(h, None, 1024,
                         k32.LoadLibrary, m, 0,
None)
                                                14
```

User Mode API Variants

Rather than create a new remote thread, we can hijack an existing thread using GetThreadContext, SetThreadContext

 SetWindowsHookEx can also be used to inject a DLL into a single remote process, or every process running on the current Desktop



SetWindowsHookEx

- SetWindowsHookEx defines a hook procedure within a DLL that will be called in response to specific events
- Example events: WH_KEYBOARD,
 WH_MOUSE, WH_CALLWNDPROC, WH_CBT
- Whenever the hooked event is first fired in a hooked thread, the specified DLL is be loaded



Permissions and Security

- To open a process opened by another user (including SYSTEM), you must hold the SE_DEBUG privilege
- Normally SE_DEBUG is only granted to member of the Administrator group
- However, even if you are running as a normal user, malware can still inject into another process that you own



Injecting code via AppInit_DLLs

The AppInit_DLLs registry value provides another convenient method of DLL injection





Injecting code via Detours

- Detours is a library developed by Microsoft Research in 1999
- The library uses the same techniques already described, wrapped up in slick package





Detours Features

- Function hooking in running processes
- Import table modification
- Attaching a DLL to an existing program file
- Detours comes with great sample programs:
 - Withdll
 - Injdll
 - Setdll
 - Traceapi



Setdll

- Detours can add a new DLL to an existing binary on disk. How?
- Detours creates a section named ".detours" between the export table and debug symbols
- The .detours section contains the original PE header, and a new IAT
- Detours modifies the PE header to point at the new IAT (reversible)



Setdll Demo

PEview - C:\temp\notepad.exe View Go Help File 🙆 🕑 🥝 🕑 💌 💌 🛨 📖 🚥 📼 pFile Data Description Value 🖃 notepad.exe ~ IMAGE DOS HEADER 00010EA8 00014980 Hint/Name RVA 0290 exit MS-DOS Stub Program 00010FAC 000149B8 Hint/Name RVA 00A8 acmdln IMAGE NT HEADERS 00010EB0 000149C2 Hint/Name RVA 006D getmaina IMAGE SECTION HEADER .text 00010FB4 Hint/Name RVA 013B initterm 000149D2 IMAGE SECTION HEADER .data 00010FB8 000149DE Hint/Name RVA 009A setusern IMAGE SECTION HEADER .rsrc 00010EBC 000149F2 Hint/Name RVA 00B6 adjust fdir IMAGE SECTION HEADER .detour 00010EC0 00014A02 Hint/Name RVA 0080 p comr SECTION text 00010EC4 00014A12 Hint/Name RVA 0085 p fmod 00010FC8 00014A20 Hint/Name RVA 0098 set app SECTION .data 00D6 controlfp SECTION .rsrc 00010FCC 00014A32 Hint/Name RVA 0330 wcsncpy SECTION .detour 00010ED0 00014A40 Hint/Name RVA IMPORT Name Table 00010ED4 00000000 End of Imports msvort.dll IMPORT Hints/Names & DLL Names 0001 00010F20 80000001 Ordinal IMPORT Directory Table 00010F24 00000000 End of Imports listener.dll < >

Viewing IMPORT Name Table



Setdll Demo

Untitled - Note	pad	_ 🗆 🛛
File Edit Format	View Help	
File Edit Format	View Help ndows Security Alert To help protect your computer, Windows Firewall has blocked some features of this program. you want to keep blocking this program? Name: Notepad Publisher: Microsoft Corporation Keep Blocking Unblock Ask Me Later dows Firewall has blocked this program from accepting connections from the Internet or twork. If you recognize the program or trust the publisher, you can unblock it When uld l unblock a program?	
Do y Win a ne shou	you want to keep blocking this program? Name: Notepad Publisher: Microsoft Corporation Keep Blocking Unblock Ask Me Later dows Firewall has blocked this program from accepting connections from the Internet or twork. If you recognize the program or trust the publisher, you can unblock it. When uld I unblock a program?	



Avoiding the Disk

- When we perform DLL injection, LoadLibrary expects the DLL to be on the disk (or at least an SMB share)
- The Metasploit project eliminates this requirement using a clever hooking strategy
- By hooking functions that are involved in reading the file from disk, they fool Windows into thinking the DLL is on disk

Meterpreter

- Hook \rightarrow Call LoadLibrary \rightarrow Unhook
- Hooked functions:
 - NtMapViewOfSection
 - NtQueryAttributesFile
 - NtOpenFile
 - NtCreateSection
 - NtOpenSection
- See remote_dispatch.c and libloader.c in MSF 3.0



Meterpreter Demo

Metasploit Framework Web Console v2.4	- Microsoft Internet Explorer	
File Edit View Favorites Tools Help		
🗢 Back 🔹 🔿 🖌 🙆 Search	Favorites CHistory	
Address 🛃 http://127.0.0.1:55555		💌 🖓 Go 🛛 Lini
A CONTRACTOR	CEPT	
~		~ ~ p
EXPLOITS	PAYLOADS	SESSIONS
EXPLOITS 3Com 3CDaem	PAYLOADS Filter Mod	SESSIONS
EXPLOITS SCom 3CDaema AOL Instant Me	PAYLOADS Filter Mod on FTP Server Overflow rssenger goaway Overflow	SESSIONS



Poison Ivy RAT

- Tons of malware uses Code Injection
- We'll quickly dig into the details of one example

🧏 Poison Ivy	
New Server System Folder Windows Folder	<u> </u>
Melt Key logger Persistence Advanced	
Process Mutex:)/VogA.14	
Key Logger Mutex: VLC9032Ca	
🔽 Inject into a custom process 🛛 🕕	
Process: msnmsgr.exe	Icon
	🍓 Build 🔽
/ersion 2.2.1 \Connections \Build {Settings {Stats {About / Port: 0 (Connection(s): C



Poison Ivy Capabilities

🧏 bh_vm [192.168.61.130] - Poison Ivy												
🕕 Information 🛛 🔥	1 1 1 2			Process	s Ma	nader						
📹 Managers 🛛 👘			AL - 222028									
- Files	Image Name	Path	PID	Thre	CPU	Merr 📤						
- 🔁 Search	System Idle Proc		0	1	94	16 Kił						
- A Regedit	System		8	36	1	216 K						
Search	SMSS.EXE	\SystemRoot\System32\smss.exe	160	6	0	344 K						
Processes	CSRSS.EXE		184	10	0	1.66 1						
- Services	WINLOGON.EXE	\??\C:\WINNT\system32\winlogon.exe	180	19	0	1.93 1						
	SERVICES.EXE	C:\WINNT\system32\services.exe	232	40	2	6.30 1						
Windowe	LSASS.EXE	C:\WINNT\system32\Isass.exe	244	20	1	4.731						
	svchost.exe	C:\WINNT\system32\svchost.exe	460	10	0	3.15 1						
Deleu	spoolsv.exe	C:\WINNT\system32\spoolsv.exe	480	14	0	3.89 1						
Antice Deute	msdtc.exe	C:\WINNT\System32\msdtc.exe	508	22	0	4.78 1						
Active Ports	svchost.exe	C:\WINNT\System32\svchost.exe	632	19	0	5.76 N						
Packet Analyzer	LLSSRV.EXE	C:\WINNT\System32\Ilssrv.exe	660	9	0	1.71 1						
Remote Shell	regsvc.exe	C:\WINNT\system32\regsvc.exe	664	2	0	828 K						
Password Audit	mstask.exe	C:\WINNT\system32\MSTask.exe	712	7	0	2.58 1						
🔁 Cached	WinMgmt.exe	C:\WINNT\System32\WBEM\WinMgmt.exe	848	4	0	492 K						
- 🧀 NT/NTLM Hashes	svchost.exe	C:\WINNT\system32\svchost.exe	868	5	0	4.30 1						
	dfssvc.exe	C:\WINNT\system32\Dfssvc.exe	896	2	0	1.30 N						
Surveillance	inetinfo.exe	C:\WINNT\System32\inetsrv\inetinfo.exe	920	27	0	8.70 1						
- 🔄 Key Logger	svchost.exe	C:\WINNT\System32\svchost.exe	1208	11	0	2.80 1						
- 🕪 Audio Capture	explorer.exe	C:\WINNT\Explorer.EXE	708	12	1	1.63 1						
- Screen Capture	VMUSrvc.exe	C:\WINNT\VMADD\VMUSrvc.exe	1184	1	0	1.32 1						
💷 Webcam Capture	wuauclt.exe	C:\WINNT\system32\wuauclt.exe	1372	6	0	2.86 1						
Administration	CMD.EXE	C:\WINNT\system32\cmd.exe	1060	1	0	920 K 🗸						
- Celit ID	<					>						
💜 Share 🛛 💌	Processes: 24 CF	PU Usage: 0 % Mem Usage: 64.95 MiB	Threads: 290	Handles: 45	18							
Downloa	0 B/s	Uploac	0 B/s									



Step 1: Inject to Explorer

- Poison Ivy client immediately injects to Explorer and then exits
- Output from WinApiOverride32 for pi.exe

Id	Dir	Call
52	Out	Process32Next(hSnapshot:0x464C,lppe: 0x12FE88: {dwSize=296,cntUsage=0,th32ProcessID=0x38C,t)
53	In	lstrcmpi(lpString1:0x12FEAC:"dfssvc.exe",lpString2:0x401363:"explorer.exe")
54	Out	Process32Next(hSnapshot:0x464C,lppe: 0x12FE88: {dwSize=296,cntUsage=0,th32ProcessID=0x4B8,t)
55	In	lstrcmpi(lpString1:0x12FEAC:"svchost.exe",lpString2:0x401363:"explorer.exe")
56	Out	Process32Next(hSnapshot:0x464C,lppe: 0x12FE88: {dwSize=296,cntUsage=0,th32ProcessID=0x174,t)
57	In	lstrcmpi(lpString1:0x12FEAC:"explorer.exe",lpString2:0x401363:"explorer.exe")
58	In	CloseHandle(hObject:0x464C)
59	In	OpenProcess(dwDesiredAccess:0x1F0FFF,bInheritHandle:0x0,dwProcessId:0x174)
60	Out	VirtualAllocEx(hProcess:0x464C,lpAddress:0x00000000: Bad Pointer,dwSize:0x1B93,flAllocationType:0x3
61	In	WriteProcessMemory(hProcess:0x464C,lpBaseAddress:0x055F0000: Bad Pointer,lpBuffer: 0x40138E: {55
62	Out	VirtualAllocEx(hProcess:0x464C,lpAddress:0x0000000: Bad Pointer,dwSize:0xB53,flAllocationType:0x30
63	In	WriteProcessMemory(hProcess:0x464C,lpBaseAddress:0x05600000: Bad Pointer,lpBuffer: 0x403500: {00
64	Out	CreateRemoteThread(hProcess:0x464C,lpThreadAttributes:0x00000000: Bad Pointer,dwStackSize:0x0,lp
65	In	CloseHandle(hObject:0x464C)
66	In	TlsFree(dwTlsIndex:0x1)



Step 2: Inject again to msnmsgr.exe

- Explorer.exe injected code then injects again...
- Interestingly, PI does not grab the SE_DEBUG privilege, so we can't inject in many existing processes

Output from WinApiOverride32 for explorer.exe

94	In	lstrcmpi(lpString1:0x477F080:"msiexec.exe",lpString2:0x4670442:"msnmsgr.exe")	Oxfffffff
95	Out	Process32Next(hSnapshot:0x414,lppe: 0x477F05C: {dwSize=296,cntUsage=0,th3	0x00000001
96	In	lstrcmpi(lpString1:0x477F080:"msnmsgr.exe",lpString2:0x4670442:"msnmsgr.exe")	0x00000000
97	In	CloseHandle(hObject:0x414)	0x00000001
98	In	OpenProcess(dwDesiredAccess:0x1F0FFF,bInheritHandle:0x0,dwProcessId:0x5B8)	0x00004b6c
99	Out	VirtualAllocEx(hProcess:0x4B6C,lpAddress:0x00000000: Bad Pointer,dwSize:0xF9C,	0x02870000
100	In	WriteProcessMemory(hProcess:0x4B6C,lpBaseAddress:0x02870000: Bad Pointer,lp	0x00000001
101	Out	VirtualAllocEx(hProcess:0x4B6C,lpAddress:0x00000000: Bad Pointer,dwSize:0xB53,	0x02880000
102	In	WriteProcessMemory(hProcess:0x4B6C,lpBaseAddress:0x02880000: Bad Pointer,lp	0x00000001
103	Out	CreateRemoteThread(hProcess:0x4B6C,lpThreadAttributes:0x00000000: Bad Point	0x00004b70
104	In	CloseHandle(hObject:0x4B6C)	0x00000001



Did it Work?

💱 Process Explorer - Sysinternals: www.sysinternals.com [SHARE\Administrator]											
File Options View	w Process Find I	Handle	Help								
Process		PID	CPU	Description	Company Name						
🖃 🖳 explorer. exe		372		Windows Explorer	Microsoft Corporation						
VMUSrvc.	.exe	1184		Virtual Machine User Services	Microsoft Corporation						
🕢 VMwareTi	ray.exe	1128		VMwareTray	VMware, Inc.						
VMwareU	ser.exe	1124		VMwareUser	VMware, Inc.						
🔰 🔍 procexp.e	xe	360	3.85	Sysinternals Process Explorer	Sysinternals						
🔰 🐴 msnmsgr. e	exe	1436		Messenger	Microsoft Corporation	T					
		•									
Туре 🛆	Name										
Key	HKLM\SYSTEM\Co	ntrolSet()01\Ser	vices\Tcpip\Linkage							
Key	HKLM\SYSTEM\Co	ntrolSet0	001\Ser	vices\Tcpip\Parameters							
Key	HKLM\SYSTEM\Co	ntrolSet(001\Serv	vices\NetBT\Parameters\Interfa	ices						
Key	HKLM\SYSTEM\Co	ntrolSet0	001\Ser	vices\NetBT\Parameters							
Mutant	\BaseNamedObject:	s\RasPb	File								
Mutant	itant \BaseNamedObjects\)!VoqA.14										
Port	\RPC Control\OLE703D9D96C97C454699E2E63899B2										
Section	\BaseNamedObjects\MessengerURL										
Section	\BaseNamedUbject:	\$\H_U		JUUd9_SMem		-					
CPU Usage: 3.85%	Commit Charge:	14.32%	Proces	ises: 27							



Where is the evil?

M Memory map									K Call	sta	ck of th	read 0000	032	8										- 0	×		
Address	Size	Owne	Sec	Contair	s	TVD	el Aci	cess	Initia		Addres	s Is	Stack	Procedure	e /	argu	men	ts			alle	d fr	om		Fran	ne	
025CF000 025E0000 025F0000	00011000 00002000 00001000			stack o	of th	n: Pri Pri Pri	V RW V RW V RW	Gua E	RW RWE RWE		0298FD 0298FD 0298FD	BC 7 DC 7 EØ 9	2C59A28F 2C59A25A 00000064	Includes KERNEL32 Timeou	.Sle t =	HLL.7 PepEx 100.	77F8 (8383	10	KE	RNE	L32. L32.	7C59 7C59	A28D A255	0298 0298	SFDD8 SFDD8	
02600000 02770000 027C0000 027D0000	00001000 00003000 0000C000 00001000					Pri Pri Pri	V RW V RW V RW V RW	E	RWE RW RW RWE		0298FD 0298FD	E4 (E8 (00000000 027D05E5	Alerta Includes	ble KEF	= FA	ALSE	°C59P	25A	02	2700	ISDF			0299	9FFB4	•
027E0000 027F0000	00001000 00001000					Pri	v RW	Ē	RWE		T Thr	ead	5	<i>%</i>	-										1	- 0	×
02800000	00001000					Pri		Ē	RWE		Ident	E	ntry	Data bloo	ck L	.ast	err	or		St	atu	5	Pric	brity	User	• time	
02820000	00001000					Pri	V RW	Ē	RWE		000002	EC 9	0000000	7FFDE000	E	RROF		PEN	DING	i Ao	tiv	e	32	+ 0	6	.0300	
02830000	00001000					Pri		Ę	RWE		000003	28 8 CC 8	00000000	7FFD7000	Ē	RROF	LSU LSU	ICCES	S (0	101 HC	tiv	e e	32	+ 0	6	.0000	
02850000	00001000					Pri	V RW	Ē	RWE		000005		0000000	7FFDB000	E	RROF	CSU	ICCES	S (0	0 Ac	tiv	e	32	+ 0	e e	.0000	!
02860000	00001000					Pri	V RW	E	RWE		000005	80 7	C57B700	7FFDD000	Ē	RROF	LSU	ICCES	S (0	00 Ac	tiv	e	32	+ 0	ē	.0000	í –
02880000	00001000				_	Pri	V RW	Ē	RWE	-	192900.0020	121			120					201022			100.00		32		
02924000	00001000					Pri	v∣R⊎	Gua	‡R₩	1 100				1	1		_			1	_					-	_
C CPU - t	hread 00	0003	28																								×
02700599	8B43 04		1	nov eax,	dwo	rd pt	r ds	:[ebx	+4]			Re	gisters	(FPU)						<		<	<	<	<	<	
0270059C	V74 04			test eax ie short	, еа: : 02	X 70050	4				_	EA	X 00000	064													
027D05A0	8BD8		1	nov ebx,	eax							ED	X 029H60 X 000000	864 000													
02700502	VEB 06 SB9F 00	04000	a li	JMP Show	duo:	27005 ed. pt	n de	·Fedi	+4001			EB	X 02B00	000													
027D05AA	8D87 D0	03000	ŏ	lea eax,	dwo	rd pt	r ds	: [edi	+3001			ES	P 0298FL P 0298FL	DBC													
027D05B0	50	02000	a	oush eax	and .	ntn e	le · Le	4:+25	01			ËS	I 77F88	398 ntdll.	ZwD	elay	Exe	cuti	on								
027D05B7	85DB	02000	°	test eb;	,eb:	X	Dile		.01			ED	I 0298FI	000													
027D05B9	~74 1D	00		je short	: 02	70050	18 F - F					EI	P 77F88	3A3 ntdll.	77F	883A	3										
027D05BF	~75 ØF	02		inz sho	t 0	r as: 27D05	D0	7201,	2			C.	0 ES 00	023 32bit	0(F	FFFF	FFF	Ş									
027D05C1	3BF3			omp esi,	ebx		-					A	0 LS 0	023 32bit	0(F	FFFF	FFF	í									
02700503	69 64		1	jnz shoi bush 64	rt 03	27005	ыu					Z	0 DS 0	023 325it	0(F	FFFF	FFF)									
027D05C7	8845 BC			nov eax,	dwo	rd pt	r ss	:[ebp	-44]			Ť	0 FS 01	038 32011 000 NULL	CEE	0700	OLF	tte?									
027D05CA	FF90 A5	00000 02	0	call dwo	ord p	ptr o	Is: [e	ax+A5 +291	2			D	0														
027D05D4	^74 B2	02		je short	02	70058	18	.201,				0	0 Last	Err ERROR_	SUC	CESS	(0)	0000	000)								
02700506	VEB 0D			jmp show	t Ø	27005	E5					EF	L 00000;	202 (NO,NE	B, NE	, A, N	S,P	O,GE	,G)								
027D05DA	6A 64		- 13	oush 64	est							ST	0 empty	-1.727404	1992	7341	402	840e	+457	0							
027D05DC	8845 BC		_ [i	nov eax,	dwo	rd pt	r ss	: [ebp	-44]			ŝt	2 empty	+UNORM 27	260	77E6	A7A	C 02	75FB	48							
0270050F	8845 BC	00000	0	nov eas	dwo	ptr c rd pt	r ss	ax+H5 :[ebn	-44]			ST	3 empty	+UNORM 12	ES I	77FB	7E6	4 02	75FE	80							
027D05E8	8388 21	01000	0 0	omp dwo	d p	tr ds	:[ea	x+121	1,0			ST	9 empty 5 empty	3.1512302 +UNORM 00	101	9195	(43 1E3	50e-0 0 77	4932 F8C6	7C							
027D05EF	~0F85 CA	FAFFF	F	jnz 0270 oush 800	100BI 10	2						ST	6 empty	+UNORM 00	909	7FFD	SBF	8 00	15E4	68							
027D0EE0	40 00	0000	3	ouch @	~~						_	ST	rempty	-8.969199	1516	2122	132	180e	+460	3	÷.						



Kernel Process Injection





Two Halves of the Process

- User land processes are comprised of two parts
 - Kernel Portion
 - EPROCESS and KPROCESS
 - ETHREAD and KTHREAD
 - Token
 - Handle Table
 - Page Tables
 - Etc.



Two Halves of the Process

User land Portion

- Process Environment Block (PEB)
- Thread Environment Block (TEB)
- Windows subsystem (CSRSS.EXE)
- Etc.



Kernel Process Injection Steps

- Must find suitable target
 - Has a user land portion
 - Has kernel32.dll and/or ntdll.dll loaded in its address space
 - Has an alterable thread (unless hijacking an existing thread)
- Allocate memory in target process
- Write the equivalent of "shellcode" that calls LoadLibrary
- Cause a thread in the parent to execute newly allocated code
 - Hijack an existing thread
 - Create an APC



Allocate memory in parent process

- Change virtual memory context to that of the target
 - KeAttachProcess/KeStackAttachProcess
 - ZwAllocateVirtualMemory
 - (HANDLE) -1 means current process
 - MEM_COMMIT
 - PAGE_EXECUTE_READWRITE



Creating the Shellcode

- "shellcode" that calls LoadLibrary
 - Copy function parameters into address space
 - Pass the address of function parameters to calls
 - Can use the FS register
 - FS contains the address of the TEB
 - TEB has a pointer to the PEB
 - PEB has a pointer to the PEB_LDR_DATA
 - PEB_LDR_DATA contains all the loaded DLLs



Creating the Shellcode

- As an alternative to using the FS register
 - Find the address of ntdll.dll from the driver
 - Parse its exports section
 - Does not work with all DLLs
 - Only address of ntdll.dll returned by ZwQuerySystemInformation



Thread Hijacking

- Cause a thread in the parent to execute newly allocated code - Hijack an existing thread
 - Locate a thread within the parent process
 - Change its Context record
 - Change Context record back when done

Problems:

- Low priority threads
- Blocked threads
- Changing Context back



Thread Context Hijacking

- Hijack and Context records
- Ikd> dt nt!_CONTEXT
- +0x000 ContextFlags : Uint4B
- +0x004 Dr0 : Uint4B
- +0x008 Dr1 : Uint4B
- +0x00c Dr2 : Uint4B
- +0x010 Dr3 : Uint4B
- +0x014 Dr6 : Uint4B
- +0x018 Dr7 : Uint4B
- +0x01c FloatSave : _FLOATING_SAVE_AREA
- +0x08c SegGs : Uint4B
- +0x090 SegFs : Uint4B
- +0x094 SegEs : Uint4B
- +0x098 SegDs : Uint4B
- +0x09c Edi : Uint4B
- +0x0a0 Esi : Uint4B
- +0x0a4 Ebx : Uint4B
- +0x0a8 Edx : Uint4B
- +0x0ac Ecx : Uint4B
- +0x0b0 Eax : Uint4B
- +0x0b4 Ebp : Uint4B
- +0x0b8 Eip : Uint4B
- +0x0bc SegCs : Uint4B
- +0x0c0 EFlags : Uint4B
- +0x0c4 Esp : Uint4B
- +0x0c8 SegSs : Uint4B
- +0x0cc ExtendedRegisters : [512] UChar



Alternative Method: APC

- Cause a thread in the parent to execute newly allocated code - Create an APC
 - Threads can be notified to run an Asynchronous Procedure Call (APC)
 - APC has a pointer to code to execute
 - To be notified, thread should be Alertable



Alertable Threads and APCs – MSDN

Parameter Settings of KeWaitForXxx Routines	Special Ke	rnel-Mode	Normal Ke APC	rnel-Mode	User-Mode	Alerts	
	Wait Aborted?	APC Delivered and Executed?	Wait Aborted?	APC Delivered and Executed?	Wait Aborted?	APC Delivered and Executed?	Wait Aborted?
Alertable = TRUE WaitMode = User	No	If (A) then Yes	No	If (B) then Yes	Yes	Yes, after thread returns to user mode	Yes
<i>Alertable =</i> TRUE <i>WaitMode =</i> Kernel	No	If (A) then Yes	No	If (B) then Yes	No (since <i>WaitMode</i> = Kernel)	No	Yes
<i>Alertable =</i> FALSE <i>WaitMode =</i> User	No	If (A) then Yes	No	If (B) then Yes	No (since <i>Alertable</i> = FALSE)	No (with exceptions, EX. ^C to terminate)	No
<i>Alertable =</i> FALSE <i>WaitMode =</i> Kernel	No	If (A) then Yes	No	If (B) then Yes	No (since Alertable = FALSE and since WaitMode = Kernel)	No	No

- A. IRQL < APC_LEVEL
- B. IRQL < APC_LEVEL, thread not already in an APC, thread not in a critical section



Finding an Alertable Thread

```
PETHREAD FindAlertableThread(PEPROCESS eproc)
   PETHREAD start, walk:
   if (eproc == NULL)
          return NULL:
   start = *(PETHREAD *)((DWORD)eproc + THREADOFFSET);
   start = (PETHREAD)((DWORD)start - THREADFLINK);
   walk = start:
   do
   {
          DbgPrint("Looking at thread 0x%x\n",walk);
          if (*(PUCHAR)((DWORD)walk + ALERTOFFSET) == 0x01)
                    return walk:
          walk = *(PETHREAD *)((DWORD)walk + THREADFLINK);
          walk = (PETHREAD)((DWORD)walk - THREADFLINK);
   }while (walk != start);
```

return NULL;



Kernel Process Injection Demo





Memory Analysis

Motivation

- APIs lie. The operating system can be subverted.
 - Example: Unlink injected DLLs from the PEB_LDR_DATA in the PEB.
 - Example: Hooking the Virtual Memory Manager and diverting address translation.
- APIs are not available to "classic" forensic investigations – offline analysis



Memory Analysis

Requirements

- No use of APIs to gather data.
- Ability to use any analysis solution on both live memory and offline memory image dumps. (Implies the ability to do all memory translation independently.)
- Do not require PDB symbols or any other operating specific information.



Steps to Memory Analysis

- Ability to access physical memory
- Derive the version of the OS important to know how to interpret raw memory
- Find all Processes and/or Threads
- Enumerate File Handles, DLLs, Ports, etc.



Steps to Memory Analysis

- Virtual to Physical Address Translation
 - Determine if the host uses PAE or non-PAE
 - Find the Page Directory Table process specific
 - Translate prototype PTEs
 - Use the paging file



Derive the version of the OS

Find the System Process

- Allows the derivation of:
 - The major operating system version in question
 - The System Page Directory Table Base
 - HandleTableListHead
 - Virtual address of PsInitialSystemProcess
 - PsActiveProcessHead
 - PsProcessType



Patent Pending 1-50

Operating System Version

- Find the System image name
- Walk backwards to identify the Process Block

 The spatial difference between major versions of the OS is enough to begin to tell us about the operating system version



Operating System Version

Drawback: Ghosts

- There can be more than one System Process
 - Open a memory crash dump in Windbg
 - Run a Windows operating system in VMWare
- Solution:
 - Non-paged kernel addresses are global
 - We know the virtual address of PsActiveProcessHead
 - PsActiveProcessHead and other kernel addresses should be valid and present (translatable) in both live or dead memory



Memory Translation

PAE vs non-PAE

- Different ways to interpret the address tables
- The sixth bit in the CR4 CPU register determines if PAE is enabled
- Problem: We do not have access to CPU registers in memory analysis
- Solution?
 - Kernel Processor Control Region -> KPCRB -> KPROCESSOR_STATE -> KSPECIAL_REGISTERS -> CR4



Memory Translation

CR4 Heuristic

 Page Directory Table Base and the Page Directory Table Pointer Base look very different.

CR3 is updated in the KPCR

- This can be used to identify a valid Page Directory Table
- The Page Directory can be used to validate the PsActiveProcessHead



Enumerating Injected DLLs

- Problem:
 - APIs lie.
 - Malware can unlink from the PEB_LDR_DATA lists of DLLs

Solution:

Virtual Address Descriptors (VADs)



Patent Pending 1-55

VADs

- Self balancing binary tree [1]
- Contains:
 - Virtual address range
 - Parent
 - Left Child and Right Child
 - Flags is the memory executable
 - Control Area

1. Russinovich, Mark and Solomon, Dave, Microsoft Windows Internals, Microsoft Press 2005



A Memory Map to a Name

- VAD contains a CONTROL_AREA
- CONTROL_AREA contains a FILE_OBJECT
- A FILE_OBJECT contains a UNICODE_STRING with the filename
- We now have the DLL name



Patent Pending 1-57

Demo





Conclusion



Questions?

Email: jamie.butler AT mandiant.com

