Reverse Engineering PsExec for fun and knowledge

cybergeeks.tech/reverse-engineering-psexec-for-fun-and-knowledge/

Summary

PsExec is a tool developed by Mark Russinovich that can be used to execute applications on remote systems. This post's purpose is to give details about the inner workings of PsExec for research purposes only. This is not an extensive analysis of every argument that PsExec uses, and we only provide details about the general usage of the tool. The idea of Reverse Engineering PsExec was initially proposed in the following tweet: <u>https://twitter.com/DebugPrivilege/status/1512851119688531976</u>.

Disclaimer: Our approach is not intended to break the Sysinternals Software License mentioned at <u>https://docs.microsoft.com/en-us/sysinternals/license-terms</u>. The binary was not decompiled using IDA Pro (disassembled code only).

Analyst: @GeeksCyber

Technical analysis

SHA256: 3B08535B4ADD194F5661E1131C8E81AF373CA322CF669674CF1272095E5CAB95

The blog post is split into two parts. The first part presents a situation where PsExec is running on a remote machine specified by a computer name or an IP address.

First process: psexec.exe \\192.168.164.130 -u test -p test -h cmd.exe

The GetModuleFileNameW API is utilized to retrieve the path of the executable:

Address He	w.				ASCTT	1		00197	CE0 0000	00208			
d Dump 1	Dump 2	🐇 Dump 3	Dump 4	🗑 Dump 5	Watch 1	Locals	2 Struct	00197	CD8 0000 CDC 0019	00000 97CFC			
dword ptr [00427164 <ps< th=""><th>exec.&GetMo xe:\$3D76 #3</th><th>oduleFileNa 8176</th><th>meW>]=<kerne< th=""><th>132.GetModule</th><th>FileNameWa</th><th>•</th><th></th><th></th><th>2: [esp+4] 001 3: [esp+8] 000 4: [esp+C] 042</th><th>97CFC 00208 F2296 L"cm</th><th>d.exe"</th><th></th></kerne<></th></ps<>	exec.&GetMo xe:\$3D76 #3	oduleFileNa 8176	meW>]= <kerne< th=""><th>132.GetModule</th><th>FileNameWa</th><th>•</th><th></th><th></th><th>2: [esp+4] 001 3: [esp+8] 000 4: [esp+C] 042</th><th>97CFC 00208 F2296 L"cm</th><th>d.exe"</th><th></th></kerne<>	132.GetModule	FileNameWa	•			2: [esp+4] 001 3: [esp+8] 000 4: [esp+C] 042	97CFC 00208 F2296 L"cm	d.exe"	
EIP	→• 00403D7 <	6 FF 1	5 64 71 42	00	call dword pt	r ds:[<&Ge	etModuleFileName	w>]	>	Default (stdcall)	000	•	5 🗘 🗌 Unloc
	 00403D6 00403D6 00403D6 00403D7 00403D7 	6 68 0 8 8D 44 F 51 0 6A 0 2 89 4	8 02 00 00 C 24 1C 0 4 24 1C		push 208 lea ecx,dword push ecx push 0 mov dword ptr	ptr ss:[esp+]	esp+1C]		(esp+) (esp+)	x87StatusWord x87SW_B 0 x8 x87SW_C1 0 x8 x87SW_SF 0 x8	0000 75W_C3 0 75W_C0 0 75W_P 0	x875W_C2 x875W_E5 x875W_U	0

Figure 1

The process extracts version information size for PsExec by calling the GetFileVersionInfoSizeW routine:

	 00403D80 00403D81 00403D85 	52 80 44 24 1C 50	<pre>push edx lea eax,dword ptr ss:[esp+1C] push eax</pre>	eax:L'	x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0
EIP	00403D86 <	E8 EF 29 00 00	call <psexec.getfileversioninfosizew></psexec.getfileversioninfosizew>	· · · · · · · · · · · · · · · · · · ·	Default (stdcall)
<pre><psexec .text:0<="" pre=""></psexec></pre>	.GetFileVersionInfo: 0403D86 psexec.exe:	5izew> 53D86 #3186			2: [esp+4] 00197CF8 3: [esp+8] 042F2296 L"cmd.exe" 4: [esp+C] 042F3C40 &L"C:\\Users\\\\Desktop\\
C Dump	0 1 🕴 Dump 2 🐇	Dump 3 🛛 Dump 4 🎯	Dump 5 🥈 Watch 1 🔮 Locals 🦻 Struct	00197CDC 001 00197CE0 001	97CFC L"C:\\Users\\\\Desktop\\psexec.exe" 97CF8

The version information for PsExec is copied to a buffer using GetFileVersionInfoW:



Figure 3

The translation array is retrieved from the version-information resource:

ETP	 00403001 00403002 00403007 00403008 00403004 00403011 00403015 00403015 00403014 	52 66 89 44 24 0C 8D 44 24 08 50 68 E4 A0 42 00 8D 4C 24 14 56 89 4C 24 14 E8 4F 2A 00 00	<pre>push edx mov word ptr ss:[esp+0],ax lea eax,dword ptr ss:[esp+8] push eax push eaxec.42A0E4 lea ecx,dword ptr ss:[esp+14] push esi mov dword ptr ss:[esp+14],ecx call <psexec.verqueryvaluex></psexec.verqueryvaluex></pre>	êdx3&L 42A0€4	X87/W_4 s (Emply) X87/W_5 s (Emply) X87/W_6 3 (Emply) X87/W_7 3 (Emply) X87/W_6 3 (Emply) X87/W_7 3 (Emply) X87/Statusword 0000 X87/SW_C2 0 X87/SW_C1 0 X87/SW_C2 0 X87/SW_C2 0 X87/SW_5 0 X87/SW_7 0 X87/SW_2 0 X87/SW_2 0
	• <			>	Default (stdcall) 🔹 5 🗘 Unlock
<pre><psexec.ver0 .text:004030<="" pre=""></psexec.ver0></pre>	ueryValuew>	\$3D1A #311A			1: [esp] 042F1888 2: [esp+4] 0042A0E4 L"\\VarFileInfo\\Translation 3: [esp+8] 00197CCC 4: [esp+C] 00197CD4 &L"C:\\Users\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
Dump 1	🛢 Dump 2 🐇	Dump 3 🚯 Dump 4	Dump 5 🥈 Watch 1 🌒 Locals 🖉 Struct	00197C88 04 00197C8C 00	2F1818 V42A0E4 L"\\VarFileInfo\\Translation"
Address Hex		10 00 55 75 10 00 81	ASCII	^ 00197CC0 00 00197CC4 00	0197CCC &L"C:\\Users\\\\Desktop\\psexec.exe"

Figure 4

The InternalName string is extracted by calling the VerQueryValueW function (see figure 5). 040904B0 is a combination of 0x409 (English – United States language) and 0x4B0 (UTF_16).

EIP	00403C8F 00403C90 00403C90 00403C97 00403C97 00403C95 00403C9E 00403C9E 00403C9E 00403C9F	50 80 80 DC FD FF FF 51 80 95 E4 FD FF FF 52 56 E8 CA 2A 00 00	push eax lea ecx,dword ptr ss:[ebp-224] push ecx lea edx,dword ptr ss:[ebp-21C] push edx push esi call <psexec.verqueryvaluew></psexec.verqueryvaluew>	edx:L'	x875tatusWord 0000 x875W_B 0 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0 Default (stdcall)
<pre><psexec.v .text:004<="" pre=""></psexec.v></pre>	erQueryValuew> 03C9F psexec.exe:	\$3C9F #309F			1: [esp1 042F1818 2: [esp+4] 00197A94 L"\\StringFileInfo\\04090480 3: [esp+8] 00197A8C 4: [esp+C] 00197A88
Address	👶 Dump 2 🐇	Dump 3 🚯 Dump 4 🎲 Du	mp 5 🎸 Watch 1 🔮 Locals 🤌 Struct	00197A68 042 00197A6C 001 00197A70 001 00197A74 001	1818 97A94 L"\\StringFileInfo\\04090480\\InternalName" 97A8C 97A88

Figure 5

A similar approach is used to extract FileVersion, FileDescription, LegalCopyright, and CompanyName.

There is a function call to GetVersion, and the return value is expected to be < 0x80000000; otherwise, it prints the "PsExec requires Windows NT/2000/XP/2003." message:



Figure 6

The file performs a comparison between the command line arguments and "/accepteula" (or "-accepteula"):

.text:00402320	1
.text:00402320 loc_402320:	
.text:00402320 mov eax, [edi+esi*4]	
<pre>.text:00402323 push offset aAccepteula ; "/accepteula"</pre>	
.text:00402328 push eax ; wchar_t *	
.text:00402329 callwcsicmp	
.text:0040232E add esp, 8	
.text:00402331 test eax, eax	
.text:00402333 jz short loc_402351	
	Figure 7
.text:00402335 mov ecx, [edi+esi*4]	
<pre>.text:00402338 push offset aAccepteula_0 ; "-accepteula"</pre>	
.text:0040233D push ecx ; wchar_t *	
.text:0040233E callwcsicmp	
.text:00402343 add esp, 8	
.text:00402346 test eax, eax	
.text:00402348 jz short loc_402351	

RegCreateKeyW is utilized to create the "Software\Sysinternals\PsExec" registry key (0x8000001 = **HKEY_CURRENT_USER**):



Figure 8

The PsExec executable is looking for a registry value called "EulaAccepted", which determines whether the user has accepted the EULA (License Agreement):

	 00401EF8 00401EF9 00401F00 00401F01 00401F03 00401F03 00401F03 00401F04 00401F08 	50 80 8C 24 51 6A 00 68 80 99 52 C7 44 24	24 02 00 00 42 00 20 04 00 00 0	push eax lea ecx,dword push ecx push 0 push 0 push psexec.42 push edx 0 mov dword ptr	ptr ss:[e 9980 ss:[esp+2	sp+224		[esp+2 429980	A0/1m_+ 3 (cmpty) A0/1m_5 3 (cmpty) x87TW_6 3 (Empty) x87TW_7 3 (Empty) x875X_1000 x875W_8 0 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C3 0 x875W_E5 0 x875W_55 0 x875W_P 0 x875W_E5 0 x875W_55 0 x875W_P 0 x875W_U 0
dword ptr	00401513 (0042704C <psex 01F13 psexec.exe</psex 	FF 15 4C	70 42 00 valueExw>]= <ad< th=""><th>vapi32.RegQueryValu</th><th>ds: [<ℜ</th><th>gqueryvalueEx</th><th><pre>>]</pre></th><th>,</th><th>Default (stdcall) ▼ 5 □ Unlod 1: [esp] 00000220 2 [esp+4] 0429980 L"EulaAccepted" 3: [esp+4] 0000000 4: [esp+4] 00000000 4: [esp+4] 00000000 4: [esp+4] 00000000 1: [esp+4] 00000000 1: [esp+4] 00000000 1: [esp+4] [esp+4]</th></ad<>	vapi32.RegQueryValu	ds: [<ℜ	gqueryvalueEx	<pre>>]</pre>	,	Default (stdcall) ▼ 5 □ Unlod 1: [esp] 00000220 2 [esp+4] 0429980 L"EulaAccepted" 3: [esp+4] 0000000 4: [esp+4] 00000000 4: [esp+4] 00000000 4: [esp+4] 00000000 1: [esp+4] 00000000 1: [esp+4] 00000000 1: [esp+4] [esp+4]
d Dump 1	🛢 Dump 2 📢	Dump 3	Dump 4 🎯 Du	mp 5 🏾 🕉 Watch 1	🙆 Locals	2 Struct	00	0197A88 00 0197A8C 00	000220 4299B0 L"EulaAccepted"
Address 00197AAC 00197ABC	Hex 53 00 6F 00 66 0 5C 00 53 00 79 0	00 74 00 77 0 00 73 00 69 0	0 61 00 72 00 0 6E 00 74 00	ASCII 55 00 5.0.f.t.w.a. 55 00 \.S.y.s.1.n.			^ 00 00	0197A90 00 0197A94 00 0197A98 00 0197A98 00 0197A9C 00	000000 000000 197CC0 197A8

Figure 9

The executable loads the "Riched32.dll" module into the address space of the process:

 00401F39 00401F3E 	68 94 99 42 00 8D 7E 12	<pre>push psexec.429994 lea edi,dword ptr ds:[esi+12]</pre>	429994	x875W_SF 0 x875W_P 0 x875W_U 0
EIP → 00401F41	FF 15 E4 71 42 00	<pre>call dword ptr ds:[<&LoadLibraryw>]</pre>	>	Default (stdcall)
dword ptr [004271E4 <psexec .text:00401E41 psexec.exe:\$</psexec 	.&LoadLibraryW>]= <kernel< th=""><th>1: [esp+8] 0425994 L %1Ched32.d11 2: [esp+8] 0425940 &L"C:\\Users\\ 3: [esp+8] 0000008 4: [esp+C] 00197F2C</th></kernel<>	1: [esp+8] 0425994 L %1Ched32.d11 2: [esp+8] 0425940 &L"C:\\Users\\ 3: [esp+8] 0000008 4: [esp+C] 00197F2C		
Anni An A	A	• Y	00197A90 00/	429994 L"Riched32.dll"

Figure 10

The DialogBoxIndirectParamW routine is utilized to create a modal dialog box based on a dialog box template:

	 004021A9 004021AA 004021AA 004021AZ 004021B2 004021B2 004021B3 004021B7 004021B8 	55 33 C9 68 F0 1C 40 00 51 56 66 89 0C 47 66 01 5E 08 51	<pre>push ebp xor ecx,ecx push psexec.401CF0 push ecx push esi mov word ptr ds:[edi+eax*2],cx add word ptr ds:[esi+8],bx push ecx</pre>		A0'IT=-5 (CHULY) A0'IT=-5 (CHULY) X87TW_5 3 (CHULY) X87TW_7 3 (CHULY) X87StatusWord 0000 X87SW_8 0 X87SW_C3 0 X87SW_C2 0 X87SW_51 0 X87SW_C3 0 X87SW_55 0 X87SW_5F 0 X87SW_P 0 X87SW_U 0 X87SW_5F 0 X87SW_P 0 X87SW_U 0
dword ptr	0040218C < [00427320 <psexec 0218C psexec.exe:\$</psexec 	FF 15 20 73 42 00 &DialogBoxIndirectParam	<pre>call dword ptr ds:[<&DialogBoxIndire >]=<user32.dialogboxindirectparamw></user32.dialogboxindirectparamw></pre>	ctParamw>]	> Default (stdcall) ▼ 5 □ Unlock 1: [esp1 0000000 2: [esp+4] 02605530 3: [esp+6] 0000000 4: [esp+6] 0000000 4: [esp+6] 0000000 4: [esp+6] 000401CF0 5: 6: 0: </th
Address 00197CC0	Dump 2 Solution Dump 2 Solution Doi: 00 00 00 18 18	Dump 3 9 Dump 4 9 Du	mp 5 6 Watch 1 6 Locals 2 Struct	00197A80 00197A84 00197A85 00197A80 00197A90	00000000 02605530 00000000 004010F0 psexec.00401CF0 0042F19C4 L"PsExec"

The text of the window's title bar is changed to "PsExec License Agreement" by calling the SetWindowTextW API:

9 12	● 00401DD 00401DD 00401DD • 00401DD • <	9 52 A 56 B FF 1	5 OC 73 42 0	0	push edx push esi call dword p	tr ds: [<&șe	twindowTextw>]	edx:L'	Def	7 SW_SF 0	x875W_P	1 x875w_U	0 2 5 🗘 🗋 Unlock
dword ptr [.text:00401	DDB psexec.e	exec.&SetWi	indowTextW>]	= <user32.se< th=""><th>tWindowTextW</th><th>></th><th></th><th></th><th>1: 2: 3: 4:</th><th>[esp] 00 [esp+4] [esp+8] [esp+C]</th><th>001973FC L" 00197630 00401CF0 ps</th><th>PsExec Lice</th><th>nse Agreement" FO</th></user32.se<>	tWindowTextW	>			1: 2: 3: 4:	[esp] 00 [esp+4] [esp+8] [esp+C]	001973FC L" 00197630 00401CF0 ps	PsExec Lice	nse Agreement" FO
d Dump 1	Dump 2	🐇 Dump 3	Dump 4	谢 Dump 5	8 Watch 1	Locals	2 Struct	001973D4 000 001973D8 000	82048 1973F	C L"PSE	ec License	Agreement"	
Figure	12												

The binary obtains a handle to a control in the dialog box created above using GetDlgItem:

	 00401DF3 00401DF8 	68 F4 01 00 00 56	push 1F4 push esi		x87SW_SF 0 x87SW_P 1	x875W_U 0
310	→• 00401DF9 <	FF D3	call ebx	ebx:Gt v	Default (stdcall)	▼ 5 🗘 Unlock
ebx= <user3< th=""><th>2.GetDlgItem> (7)</th><th>\$1DF9 #11F9</th><th></th><th></th><th>1: [esp+4] 000001F4 3: [esp+8] 000001F4 4: [esp+6] 00000000</th><th></th></user3<>	2.GetDlgItem> (7)	\$1DF9 #11F9			1: [esp+4] 000001F4 3: [esp+8] 000001F4 4: [esp+6] 00000000	
d Dump 1	🛢 Dump 2 🐇	Dump 3 🛛 🚯 Dump 4 🏻 🎯 Du	mp 5 🧉 Watch 1 🕘 Locals 🦻 Struct	001973C8 008 001973CC 000	320488 0001F4	
	10					

Figure 13

The process sends the **TTM_GETTOOLINFO** (0x435) message to the window in order to get the current tooltip definition:

EIR	<pre>00401DE7 00401DEC 00401DEE 00401DF3 00401DF3 00401DF8 00401DF9 00401DF9 00401DF9 00401E01 00401E01 00401E01 00401E02</pre>	68 00 00 10 00 6A 00 68 35 04 00 00 68 F4 01 00 00 56 FF D3 88 2D 10 73 42 00 56 FF D5	push 100000 push 455 push 1F4 push esi call ebx mov ebp,dword ptr ds:[<&SendMessagew>] push eax call ebp	ebx:Gr ebp:Sr ebp:St	A0/In_t 2 (Empty) A0/In_s 2 (Empty) x87TW_6 3 (Empty) x87TW_7 3 (Empty) x87Tw_6 1 (Empty) x87TW_7 3 (Empty) x87Statusword 4020 x87SW_25 0 x87SW_8 0 x87SW_25 0 x87SW_55 0 x87SW_9 0 x87SW_25 0 x87SW_55 0 x87SW_9 0 x87SW_25 0 x87SW_10 0 x87SW_15 0 x87SW_25 0 x87SW_10 0 Default (stdcall) ▼ 5 0 1000000000000000000000000000000000
ebp= <user32< th=""><th>2.SendMessagew></th><th>(75CFB900) \$1E02 #1202</th><th></th><th></th><th>1: [esp] 008F049E 2: [esp+4] 00000435 3: [esp+8] 0000000 4: [esp+C] 0010000</th></user32<>	2.SendMessagew>	(75CFB900) \$1E02 #1202			1: [esp] 008F049E 2: [esp+4] 00000435 3: [esp+8] 0000000 4: [esp+C] 0010000
Dump 1	🕴 Dump 2 🐇	Dump 3 💧 Dump 4 🎯 (ump 5 🏅 Watch 1 🚯 Locals 🤌 Struct	001973CC 008 001973D0 000 001973D4 000	F049E 000435 000000

Figure 14

The anchor highlight setting for the window's toolbar is set by sending the **TB_SETANCHORHIGHLIGHT** (0x449) message to the window:

	 00401E08 00401E09 00401E08 00401E10 00401E15 00401E15 00401E18 	50 6A 02 68 49 68 F4 56 FF D3 50	04 00 00 01 00 00		push eax push 2 push 449 push 1F4 push esi call ebx push eax				ebx:Ge	xi xi xi xi	s/1W_6 s (Empty) x 87StatusWord 4020 87SW_B 0 x87SW_C3 87SW_C1 0 x87SW_C0 87SW_SF 0 x87SW_P	1 x87SW_C2 0 x87SW_ES 1 x87SW_U	0 0	
ebp= <user32< th=""><th>2. SendMessageW</th><th><pre>FF D5 (75CFB90 (75CFB90 (31E19 #1</pre></th><th>D) 219</th><th></th><th>call ebp</th><th>-</th><th></th><th></th><th>ebp:Sr</th><th>De 1: 2: 3: 4:</th><th>fault (stdcal) [esp] 00BF049E [esp+4] 00000449 [esp+8] 0000002 [esp+C] 001973F0</th><th></th><th>5</th><th>Unlod</th></user32<>	2. SendMessageW	<pre>FF D5 (75CFB90 (75CFB90 (31E19 #1</pre>	D) 219		call ebp	-			ebp:Sr	De 1: 2: 3: 4:	fault (stdcal) [esp] 00BF049E [esp+4] 00000449 [esp+8] 0000002 [esp+C] 001973F0		5	Unlod
d Dump 1	Dump 2	Dump 3	Dump 4	🔮 Dump 5	😽 Watch 1	Locals	Struct	00	01973CC 0	0BF04 00004	9E 49			
Address He	ex				ASCII			^ 00	01973D4 0	00000	02 F0			

Figure 15

The PsExec License Agreement window appears on the screen, and we need to accept the terms in order to continue the execution:

PsExec License Agreement	×	
You can also use the /accepteula command-line switch to accept the EULA.		
SYSINTERNALS SOFTWARE LICENSE TERMS	^	
These license terms are an agreement between Sysinternals (a wholly owned subsidiary of Microsoft Corporation) and you. Please read them They apply to the software you are downloading from Systinternals.com which includes the media on which you received it, if any. The terms a apply to any Sysinternals • updates,	n, Iso	Figure 16
· supplements,		
Internet-based services, and	~	
Print Dec	line	

The PsExec executable destroys the modal dialog box created earlier using EndDialog:

• 00401D90 • 00401D92		6A 01 56		ush 1 ush esi			x875W_SF 0 x875W_P 1 x875W_U 0				
	→• <u>00401D93</u> <	FF 15 14	73 42 00	call dword ptr ds:[<&	EndDialog>]	>	Defa	ault (stdcall)	▼ 5 🗘 Ur	nlock	
dword ptr [00427314 <psex< th=""><th>ec.&EndDialog> :\$1D93 #1193</th><th>>]=≺user32.EndDialo</th><th>og></th><th></th><th></th><th>2: 3: 4:</th><th>[esp+4] 000000 [esp+8] 004010 [esp+C] 00B204</th><th>01 F0 psexec.00401CF0 88</th><th></th></psex<>	ec.&EndDialog> :\$1D93 #1193	>]=≺user32.EndDialo	og>			2: 3: 4:	[esp+4] 000000 [esp+8] 004010 [esp+C] 00B204	01 F0 psexec.00401CF0 88		
d Dump 1	8 Dump 2	Dump 3 🚯 🕻	Dump 4 🎯 Dump 5	😽 Watch 1 🛛 🕘 Locals	Struct	001971C0 00 001971C4 00	082048	38 01			
Liguro	17										

Figure 17

RegSetValueExW is used to set the value of the "EulaAccepted" registry value to 1:

 004021E1 004021E3 004021E4 004021E6 004021E6 004021E7 004021E7 004021E7 004021E7 	6A 04 80 94 24 24 02 00 00 52 6A 04 6A 00 68 80 99 42 00 50	push 4 lea edx,dword ptr ss:[esp+224] push edx push 4 push 0 push psexec.429980 push psexec.429980	[esp 4299	x8/1W_6 5 (EmpLy) x x87StatusWord 4020 x87SW_B 0 x87SW_C3 x87SW_C1 0 x87SW_C3 x87SW_SF 0 x87SW_P	1 x875W_C2 0 0 x875W_E2 0 1 x875W_U 0 2 x875W_U 0
dword ptr [00427050 <psexec. .text:004021F5 psexec.exe:\$2</psexec. 	&RegSetValueExw>]= <advap< th=""><th>132.RegSetValueExw></th><th>Jeexny J</th><th>> Default (stdcall) 1: [esp] 00000220 2: [esp+4] 00429980 L 3: [esp+5] 00000000 4: [esp+C] 00000004</th><th>▼ 5 🛟 🗌 Uniod "EulaAccepted"</th></advap<>	132.RegSetValueExw>	Jeexny J	> Default (stdcall) 1: [esp] 00000220 2: [esp+4] 00429980 L 3: [esp+5] 00000000 4: [esp+C] 00000004	▼ 5 🛟 🗌 Uniod "EulaAccepted"
a Dump 1 🕴 Dump 2 🐇 D	ump 3 🛛 🚷 Dump 4 🛛 🎯 Dur	np 5 🥈 Watch 1 🕘 Locals 🎐 Stru	ct 00197A88	00000220 00429980 L"EulaAccepted"	
Address Hex 00197CC0 01 00 00 00 18 18 2 00197CD0 00 18 2E 04 F6 3E 4	2F 04 C4 19 2F 04 2C 7F 1 40 00 C4 19 2F 04 2C 7F 1	ASCII 9 00/.Ä./ 9 00 D./.ô>@.Ä./	^ 00197A90 00197A94 00197A98 00197A96	00000000 00000004 00197CC0 00000004	

Figure 18

The binary extracts the NetBIOS name of the local computer via a function call to GetComputerNameW:

-	 004063D9 004063DA 004063DF 	52 68 20 C3 43 00 C7 44 24 28 05 01 00 00	push edx push psexec.43C320 mov dword ptr ss:[esp+28],105	x875W_C1 0 x875W_C0 0 x875W_ES 0 x875W_SF 0 x875W_P 1 x875W_U 0
EIP	→ 004063E7 <	FF 15 74 71 42 00	<pre>call dword ptr ds:[<&GetComputerNamew>]</pre>	> Default (stdcall)
dword ptr	[00427174 <psexed 63E7 psexec.exe:</psexed 	.&GetComputerNameW>]= <kern< th=""><th>e132.GetComputerNamew></th><th>1: [esp] 0043C320 psexec.0043C320 2: [esp+4] 0013P734 3: [esp+8] 00409645 <psexec.entrypoint> 4: [esp+C]_00409645 <psexec.entrypoint></psexec.entrypoint></psexec.entrypoint></th></kern<>	e132.GetComputerNamew>	1: [esp] 0043C320 psexec.0043C320 2: [esp+4] 0013P734 3: [esp+8] 00409645 <psexec.entrypoint> 4: [esp+C]_00409645 <psexec.entrypoint></psexec.entrypoint></psexec.entrypoint>
d Dump 1	🛢 Dump 2 🐇	Dump 3 🛛 🛭 Dump 4 🏻 🎯 Dump	5 🍯 Watch 1 🜒 Locals 🦻 Struct	00197F0C -0043C320 psexec.0043C320 00197F10 00197F34

Figure 19

There is a second (redundant) call to GetVersion; however, a different message is printed this time:

```
🔺 🖾
.text:004063FA push
                     offset aPsexecRequires ; "PsExec requires Windows NT or higher.\n"..
.text:004063FF call sub 4075C1
.text:00406404 add eax, 40h ; '@'
.text:00406407 push
                     eax
                                     ; FILE *
                     _fprintf
.text:00406408 call
.text:0040640D add
                     esp, 8
.text:00406410 or
                      eax, ØFFFFFFFh
.text:00406413 jmp
                      loc_406755
```

For example, if an argument is too long, then PsExec displays a message that contains a typo:

🚺 🛃 🖼		
.text:0040650B		
.text:0040650B lo	c_40650B:	
.text:0040650B mov	v ecx, [esp+8028h+var_8014]	
.text:0040650F mov	v edx, [ecx+edi*4]	
.text:00406512 pu	ish edx	
.text:00406513 pu	sh offset aArgumentToLong ; "Argument to long: %s\n\n" Figur	e 21
.text:00406518 ca	11 sub_4075C1	0 2 1
.text:0040651D add	d eax, 40h ; '@'	
.text:00406520 pu	sh eax ; FILE *	
.text:00406521 ca	11 _fwprintf	
.text:00406526 add	d esp, OCh	
.text:00406529 or	eax, 0FFFFFFFh	
.text:0040652C jm	p loc_406755	

The executable retrieves the command-line string for the process by calling the GetCommandLineW routine:

EIP 00406478 FF D6	call est	es1:Gt V	efault (stdcall)	▼ 5 🗘 🗌 Unlocke
esi= <kernel32.getcommandlinew> (76A74BCO)</kernel32.getcommandlinew>	23344	: [esp+4] 00409645 cp : [esp+4] 00409645 : [esp+8] 0019FF80 : [esp+C] 00320000	<pre>sexec.entryPoint></pre>	

Figure 22

GetFullPathNameW is used to extract the full path and file name of PsExec:

	 00406571 00406572 00406577 00406577 	52 68 E0 80 43 00 68 00 20 00 00 56	push edu push pse push 200 push est	xec.4380E0 0		es1:L'	x x	(875W_B 0 x875W_C3 1 x875W_C2 0 (875W_C1 0 x875W_C0 0 x875W_E5 0 (875W_5F 0 x875W_P 1 x875W_U 0
dword ptr	[00427224 <psexed< th=""><th>FF 15 24 72 42 0</th><th>>]=<kernel32.getful< th=""><th>1PathNameW></th><th>setFullPathNamew>]</th><th>></th><th>De 1: 2: 3:</th><th>fault (stdcall)</th></kernel32.getful<></th></psexed<>	FF 15 24 72 42 0	>]= <kernel32.getful< th=""><th>1PathNameW></th><th>setFullPathNamew>]</th><th>></th><th>De 1: 2: 3:</th><th>fault (stdcall)</th></kernel32.getful<>	1PathNameW>	setFullPathNamew>]	>	De 1: 2: 3:	fault (stdcall)
.text:0040	0657D psexec.exe:	657D #597D					4:	: [esp+C]_00197F30
Dump 1	🛢 Dump 2 🐇	Dump 3 🚯 Dump 4	🗑 Dump 5 🦷 🥉 Wat	ch 1 🕘 Locals	2 Struct	00197F04 C0 00197F08 00	0197F	F3A L"C:\\Users\\\\Desktop\\psexec.exe"
Address	Hex		ASCII			^ 00197F0C 00 00197F10 00	04380 0197F	E0 psexec.004380E0 30

Figure 23

The file retrieves the address of "CreateRestrictedToken" and other export functions via a call to GetProcAddress:

	 0040659 0040659 004065A 004065A 004065A 004065A 	6 68 4 8 68 8 0 FF D 2 88 20 8 50	4 BE 42 00 8 74 42 00 3 D E8 71 42	00	push psexec. push psexec. call ebx mov ebp,dword push eax	428E44 427488 d ptr ds:[•	&GetProcAddress>]	428E44 427488 ebx:Lc ebp:Ge	x87StatusWord 4020 x87SW_B 0 x87SW_C3 x87SW_C1 0 x87SW_C0 x87SW_SF 0 x87SW_P	1 x875W_C2 0 x875W_E5 1 x875W_U	0 0 0	
ebp= <kernel< th=""><th>32. GetProcAd</th><th>9 FF D dress> (764</th><th>A750B0)</th><th></th><th>call ebp</th><th>·</th><th></th><th>ebp:Gr v</th><th>Default (stdcal) 1: [esp] 73A40000 adva 2: [esp+4] 00428E44 "C 3: [esp+8] 00409645 <p 4: [esp+C] 00409645 <p< th=""><th>♥ pi32.73A40000 reateRestrict sexec.EntryPo sexec.EntryPo</th><th>5 edTe int</th><th>Unlock</th></p<></p </th></kernel<>	32. GetProcAd	9 FF D dress> (764	A750B0)		call ebp	·		ebp:Gr v	Default (stdcal) 1: [esp] 73A40000 adva 2: [esp+4] 00428E44 "C 3: [esp+8] 00409645 <p 4: [esp+C] 00409645 <p< th=""><th>♥ pi32.73A40000 reateRestrict sexec.EntryPo sexec.EntryPo</th><th>5 edTe int</th><th>Unlock</th></p<></p 	♥ pi32.73A40000 reateRestrict sexec.EntryPo sexec.EntryPo	5 edTe int	Unlock
& Dump 1 Figure	Dump 2 24	🐇 Dump 3	Dump 4	谢 Dump 5	🕉 Watch 1	Locals	Struct	00197F0C 73A 00197F10 004	40000 advap132.73A40000 28E44 "CreateRestricted	Token"		

The function that contains the switch statement, which chooses an execution flow depending on the command line arguments, is shown below (IDA Pro graph):



Firstly, every argument that starts with "-" is compared with "accepteula", "low", "belownormal", "normal", "abovenormal", "high", "realtime", and "background".

Every argument that starts with "-" is converted to uppercase using _toupper. The 0xFFFFBF constant is added to the return value, and then the result is supposed to be between 0x0 and 0x17 (23 in decimal). Just based on this simple calculus, the "-y" and "-z" arguments couldn't be valid:



Figure 26

In the case of invalid parameters, the process prints out the instructions for parameters:

🗾 🚄 🖼						
.text:004056F0						
.text:004056F0						
.text:004056F0						
.text:004056F0	sub 405	6F0 proc near				
.text:004056F0	-					
.text:004056F0	arg 0=	dword ptr 4				
.text:004056F0						
.text:004056F0	push	offset aPsexecExecutes	;	"PsExec e	xecutes	a program on a remote s"
.text:004056F5	call	_printf				
.text:004056FA	push	offset aApplicationsEx	;	"applicat	ions exe	ecute interactively.\n"
.text:004056FF	call	_printf				
.text:00405704	mov	eax, [esp+8+arg_0]				
.text:00405708	push	eax				
.text:00405709	push	offset aUsagePsexecCom	;	"\nUsage:	psexec	[\\\\computer[,computer"
.text:0040570E	call	_printf				
.text:00405713	push	offset aASeparateProce	;	" -a		Separate processors on "
.text:00405718	call	_printf				
.text:0040571D	push	offset aCommasWhere1Is	;			commas where 1 is the l"
.text:00405722	call	_printf				
.text:00405727	push	offset aToRunTheApplic	;			to run the application "
.text:0040572C	call	_printf				
.text:00405731	push	offset aA24 ; "			\"-a 2	,4\"\n"
.text:00405736	call	_printf				
.text:0040573B	push	offset aCCopyTheSpecif	;	" - C		Copy the specified prog"
.text:00405740	call	_printf				
.text:00405745	push	offset aDDonTWaitForPr	;	" -d		Don't wait for process "
.text:0040574A	call	_printf				10 10 10 10 10 10 10 10 10 10 10 10 10 1
.text:0040574F	push	offset aEDoesNotLoadTh	;	" -е		Does not load the speci"
.text:00405754	call	_printf				1000 million (1000 million (10
.text:00405759	push	offset aFCopyTheSpecif	;	" -f		Copy the specified prog"
.text:0040575E	call	_printf				
.text:00405763	push	offset aIRunTheProgram	;	" -i		Run the program so that"
.text:00405768	call	_printf				
.text:0040576D	push	offset aHIfTheTargetSy	;	" -h		If the target system is"
.text:00405772	call	_printf				
.text:00405777	push	offset aRunWithTheAcco	;			run with the account's "
.text:0040577C	call	_printf		1227 0220		
.text:00405781	push	offset aLRunProcessAsL	;	" -1		Run process as limited "
.text:00405786	call	_printf				
.text:0040578B	add	esp, 40h				
.text:0040578E	push	offset aAndAllowsOnlyP	;			and allows only privile"
.text:00405793	call	_printf				
Figure 27						

There is a comparison between the local computer name and the computer name\IP address passed as a parameter:



Figure 28

The NetIsServiceAccount function is used to test whether the user name passed as a parameter exists in the NetIogon store on the local machine:

BIC DODODSFB FF DD Call eax eax with the former of the for	SW_CO 0 x87SW_ES 0 SW_P 1 x87SW_U 0
1: [esp] 0000000	▼ 5 🕏 🗆 Unlod
eax= <logonc11.netisserv1ceaccount> (6CSB0250) 2: [esp+4] 00433 3: [esp+6] 00197 .text:004065FB psexec.exe:\$65FB #59FB 4: [esp+c] 004094</logonc11.netisserv1ceaccount>	u 140 L"test" F24 1645 <psexec.entrypoint></psexec.entrypoint>
Al Dump 1 € Dump 2 € Dump 3 € Dump 4 ♀ Dump 5 ℅ Watch 1 € Locals ≫ Struct O01927F03 00000000 O01937F02 00433140 L"test" O01927F02 00433140 L"test" O01927F03 00000000 O01927F03 O01927F0 O01927F0 O01927 O01927 O01927 O0192 O019 O0192 O0192 O019 O019 O0192 O019 O019 O0192 O019 O01 O019 O01 O01 O01 O01 O01 O01	

The executable creates an unnamed event object by calling the CreateEventW API:

EIP	00404980 00404981 00404981 00404982 00404984 00404986 00404987 00404987 00404987 € <	56 56 01 85 C0 56 94 FF D7	44 24 22		push esi push esi push 1 test eax,eax push esi sete byte ptr call edi	ss: [esp+2	22]		edite	X8 X8 X8 X8 X8 X8 X8 X8 X8 X8 X8 X8 X8 X	7Statusv 7SW_B 0 7SW_C1 0 7SW_SF 0 ault (stdcall	vord 4020 x875W_C3 x875W_C0 x875W_P x875W_P 0000000	1 x875i 0 x875i 1 x875i	LC2 (LES (LU (5 🗘 🗌 Unloc
.text:00404	98C psexec.exe:	\$498C #3	DBC							3:	[esp+8] [esp+C]	00000000			
d Dump 1	🛢 Dump 2 🛭 🗳	Dump 3	Dump 4	谢 Dump 5	🕉 Watch 1	Locals	2 Struct	0	018EBF0 018EBF4	0000000	0				
Address He	x	35 04 0			ASCII	-		^ 0	018EBF8 018EBFC	0000000	0				

Figure 30

A new function is added to the list of handler functions for the current process:

Dump 1	🕴 Dump 2 🐇 I	Dump 3 🚯 Dump 4	🔮 Dump 5 🥈 Watch	1 🚯 Locals 🏅	Struct	0018EBF8 004 0018EBFC 000	03F80 psexec.00403F80 000001		
dword ptr	[004270FC <psexec 49C2 psexec.exe:\$</psexec 	.&SetConsoleCtrlH 49C2 #3DC2	andler>]= <kernel32.se< th=""><th>tConsoleCtrlHandl</th><th>er></th><th>></th><th>2: [esp+4] 0000001 3: [esp+4] 0000001 4: [esp+6] 00432138 L"19</th><th>.00403F80</th><th>·</th></kernel32.se<>	tConsoleCtrlHandl	er>	>	2: [esp+4] 0000001 3: [esp+4] 0000001 4: [esp+6] 00432138 L"19	.00403F80	·
EIP	● 004049A9 ● 004049A8 ● 00404980 ● 00404988 ● 00404982 ■ 004049C2	6A 01 68 80 3F 40 00 C7 84 24 84 00 89 84 24 D4 4A FF 15 FC 70 42	00 00 58 4A mov dword 00 00 call dwor	ec.403F80 ptr ss:[esp+84], ptr ss:[esp+4AD4 i ptr ds:[<&SetCo	4A58],esi nsoleCtrlHandler>]		x87SW_B 0 x87SW_C3 1 x87SW_C1 0 x87SW_C0 0 x87SW_SF 0 x87SW_P 1 Default (stdrall)	x875W_C2 0 x875W_ES 0 x875W_U 0	

Figure 31

An intermediary message that gives details about what action will occur next is displayed (these messages aren't visible during normal execution because they're deleted after the action is complete):

C:\Users\\Desktop\psexec.exe	-	×
PsExec v2.11 - Execute processes remotely Copyright (C) 2001-2014 Mark Russinovich Sysinternals - www.sysinternals.com Connecting to 192.168.164.130		^

Figure 32

The PsExec process makes a connection to the IPC\$ share on the remote machine using the WNetAddConnection2W API. The credentials passed as parameters must be valid on the remote host:

86	00402E65 00402E67 00402E67 00402E68 00402E80 00402E74 00402E74 00402E70 00402E70 00402E70 00402E70 00402E70 00402E78 00402E80 00402E81 00402E81 00402E81 00402E81 00402E81 00402E81 00402E81 00402E81 00402E82 00402E83 00402E84 00402E84 <t< th=""><th>6A 00 56 89 44 24 18 89 44 24 20 89 44 24 20 89 44 24 28 89 44 24 28 89 44 24 28 89 44 24 28 89 44 24 14 89 44 24 14 89 44 24 14 89 44 24 14 80 44 24 18 80 44 24 38 80 40 40 40 40 80 40 40 40 80 40 40 40 80 40 40 40 80 40 8</th><th>push o push esi mov dword ptr ss mov dword ptr ss push edi lea edx, dword pt lea ecx, dword pt so dword ptr ss push edx mov dword ptr ss cs mov dword ptr ss mov dword ptr ss cs dword ptr ss</th><th><pre>cesp+18 , eax cesp+20 , eax cesp+24 , eax cesp+24 , eax cesp+24 , eax cesp+10 , eax cesp+10 , eax cesp+10 , eax cesp+10 , eax cesp+10 , eax cesp+10 , eax cesp+20 , eax cesp+30 , cax cesp+30 , c</pre></th><th>esi:L' [esp+] edi:L' [esp+] [esp+]</th><th>x87r2 000000000000000000000000000000000000</th></t<>	6A 00 56 89 44 24 18 89 44 24 20 89 44 24 20 89 44 24 28 89 44 24 28 89 44 24 28 89 44 24 28 89 44 24 14 89 44 24 14 89 44 24 14 89 44 24 14 80 44 24 18 80 44 24 38 80 40 40 40 40 80 40 40 40 80 40 40 40 80 40 40 40 80 40 8	push o push esi mov dword ptr ss mov dword ptr ss push edi lea edx, dword pt lea ecx, dword pt so dword ptr ss push edx mov dword ptr ss cs mov dword ptr ss mov dword ptr ss cs dword ptr ss	<pre>cesp+18 , eax cesp+20 , eax cesp+24 , eax cesp+24 , eax cesp+24 , eax cesp+10 , eax cesp+10 , eax cesp+10 , eax cesp+10 , eax cesp+10 , eax cesp+10 , eax cesp+20 , eax cesp+30 , cax cesp+30 , c</pre>	esi:L' [esp+] edi:L' [esp+] [esp+]	x87r2 000000000000000000000000000000000000
<pre>sexec.WNet/</pre>	AddConnection2V	(>			,	1: [esp] 0018DC30
						2: [esp+8] 00433140 L"test"
.text:00402E8	B2 psexec.exe:	\$2EB2 #22B2				4: [esp+c] 00000000
Dump 1	🛢 Dump 2 🐇	Dump 3 💧 Dump 4	🍘 Dump 5 🥈 Watch 1 🔮	Locals 🦻 Struct	0018DC14 0 0018DC18 0	0018DC30 0043C100 L"test"
Address Hex			ASCII		▲ 0018DC1C 0	00433140 L"test"
0018DC30 00 0 0018DC40 2C 0 0018DC50 5C 0 0018DC60 38 0 0018DC70 30 0	00 00 00 00 00 0C 18 00 50 DC 00 5C 00 31 00 00 2E 00 31 00 00 5C 00 49 00	00 00 00 00 00 00 01 18 00 00 00 00 00 00 03 39 00 32 00 2E 00 33 36 00 34 00 2E 00 35 50 00 43 00 24 00 00	\$ 00 00 00 ₩ 0 00 00 00 , ŬPÜ 1 00 36 00 \.\.1.9.21.6 0 00 00 00 81.6.41.3 0 00 00 00 0.\.I.P.C.\$		0018DC24 0 0018DC28 0 0018DC20 0 0018DC30 0	0019380C L"PSEXESVC.exe" 10433140 L"test" 10000000 10000000 10000000 10000000
Figure 3	33					

The binary determines the location of a resource called "PSEXESVC" via a function call to FindResourceW:



Figure 34

The resource is loaded in memory, and a pointer to the specified resource in memory is retrieved by calling the following functions: LoadResource, SizeofResource, and LockResource (see figure 35).



Figure 35

The executable creates a file called "PSEXESVC.exe" in the ADMIN\$ share on the remote machine:

	00402D60 68 7C 98 42 00 push psexec.429B7C push ecx 00402D6 S1 push ecx mov ed1,eax				x87SW_C1 0 x87SW_C0 0 x87SW_ES 0 x87SW_SF 0 x87SW_P 0 x87SW_U 0			
EIP	00402D68	E8 A6 4D 00 00	call psexec. 407813	~ ~ ~	Default (stdcall) - 5 - Unlock			
psexec.0	0407B13 402D68 psexec.exe:1	2D68 #2168			1: [csp] 00180E98 L"\\\\192.168.164.130\\ADMINS\ 2: [csp+4] 0042987C L"wb" 3: [csp+8] 0019380C L"PSEXESVC.exe" 4: [csp+C] 0042A994 L"PSEXESVC"			
d Dump	1 🕴 Dump 2 🐇	Dump 3 🛛 🚷 Dump 4 🛛 🧐	Dump 5 🥈 Watch 1 🔮 Locals 🦻 Struct	0018DE5C 001 0018DE60 004	8DE98 L"\\\\192.168.164.130\\ADMIN\$\\PSEXESVC.exe 2987C L"wb"			

Figure 36

The above file is populated using the _fwrite function (see figure below) . The hash of the file is 6A6A9AA6ED43EB3F857392459C7B05A5A0DF89E00A3214D333949A561BCFF368 and we'll describe its purpose in the upcoming paragraphs.

→● 00402D7C 56 00402D7D 53 ● 00402D7E 6A 01	push es1 push ebx push 1		X875W_B 0 X875W_C3 0 X875W_C2 0 X875W_C1 0 X875W_C0 0 X875W_E5 0 X875W_SF 0 X875W_P 0 X875W_U 0
● 00402080 57 ● 00402081 E8 45 4C 00 00	call psexec. 4079CB	fwrite v	Default (stdcall)
psexec.004079CB .text:00402D81 psexec.exe:\$2D81 #2181			1: [esp] 0045E108 psexec.0045E108 2: [esp+4] 00000001 3: [esp+8] 0002E560 4: [esp+C] 00431180 psexec.00431180
🕼 Dump 1 🕴 Dump 2 🐇 Dump 3 🚯 Dump 4 👙 D	Dump 5 🥈 Watch 1 🕚 Locals 🤌 Struct	0018DE54 0045 0018DE58 0000	E108 psexec.0045E108
Address Hex 0045E108 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF	ASCII	0018DE5C 0002 0018DE60 0043 0018DE64 0019	E560 1180 psexec.00431180 1880 I "PSSYESUC ave"

Figure 37

The binary retrieves a handle to the standard output device using GetStdHandle (0xFFFFFF5 = **STD_OUTPUT_HANDLE**):

• 00402CAA 6A F5		push FFFFFFS		
004020AC FF 15	9C 71 42 00	[Call dword ptr ds:[<&GetStdHandle>]	> ``	Default (stdcall) 🔹 5 🔹 🗌 Unlock
dword ptr [0042719C <psexec.&getst .text:00402CAC psexec.exe:\$2CAC #2</psexec.&getst 	dHandle>]= <kerne< td=""><td>32.GetStdHandle></td><td></td><td>1: [esp] FFFFFF5 2: [esp+4] 00403505 psexec.00403505 3: [esp+8] 76ACDC00 <kernel32.createeventw> 4: [esp+C] 00000000</kernel32.createeventw></td></kerne<>	32.GetStdHandle>		1: [esp] FFFFFF5 2: [esp+4] 00403505 psexec.00403505 3: [esp+8] 76ACDC00 <kernel32.createeventw> 4: [esp+C] 00000000</kernel32.createeventw>
1 Duno 1 2 Duno 2 4 Duno 2	A	and Maria A cards & cards	0018E2D4 FFF	FFFFS

Figure 38

GetConsoleScreenBufferInfo is utilized to obtain information about the console screen buffer:

	 00402C8; 00402C8; 00402C8; 00402C8; 	68 20 50 FF 15	21 43 00 90 71 42 0	00	ush psexec. Jush eax	432120	tConsoleScr	eenBufferInfo>]			875W_SF 0	x87SW_P	0 x875W_	U O	
dword ptr [(00427190 <pse 2008 psexec.ex</pse 	e:\$2CB8 #2	nsoleScreen	BufferInfo>]	= <kernel32.< th=""><th>GetConsoles</th><th>ScreenBuffer:</th><th>Info></th><th>></th><th>D 1 2 3 4</th><th>efault (stdcal) : [esp] 00 : [esp+4] : [esp+8] : [esp+C]</th><th>0000098 00432120 p 00403505 p 76ACDC00 <</th><th>sexec.00432 sexec.00403 kernel32.Cr</th><th>• 5 505 eateE</th><th>tentw></th></kernel32.<>	GetConsoles	ScreenBuffer:	Info>	>	D 1 2 3 4	efault (stdcal) : [esp] 00 : [esp+4] : [esp+8] : [esp+C]	0000098 00432120 p 00403505 p 76ACDC00 <	sexec.00432 sexec.00403 kernel32.Cr	• 5 505 eateE	tentw>
d Dump 1	B Dump 2	🐇 Dump 3	🔕 Dump 4	谢 Dump 5	🕉 Watch 1	locals	2 Struct		0018E2D0 0	00000	98 20 psexec	.00432120			

The next action of PsExec is to start the PSEXESVC service on the remote host, as highlighted below:



Figure 40

The binary establishes a connection to the service control manager on the remote machine by calling the OpenSCManagerW routine (0xF003F = **SC_MANAGER_ALL_ACCESS**):

670	 00402B35 00402B3A 00402B3C 	68 3F 6A 00 50	00 OF 00		push F003F push 0 push eax			eax:	L"	x875W_C1 0 x875W_C0 0 x875W_ES 0 x875W_SF 0 x875W_P 0 x875W_U 0
dword ptr ((00427024 <psex)< th=""><th>ec. &OpenSi</th><th>CManagerW>]</th><th><advapi32.0< th=""><th>OpenSCManager</th><th>.M></th><th>Jensemanager wyg</th><th>- 46.</th><th>></th><th>Default (stdcall) 1: [esp] 00432138 L"192.168.164.130" 2: [esp+4] 00000000 2: [esp+4] 00000005</th></advapi32.0<></th></psex)<>	ec. &OpenSi	CManagerW>]	<advapi32.0< th=""><th>OpenSCManager</th><th>.M></th><th>Jensemanager wyg</th><th>- 46.</th><th>></th><th>Default (stdcall) 1: [esp] 00432138 L"192.168.164.130" 2: [esp+4] 00000000 2: [esp+4] 00000005</th></advapi32.0<>	OpenSCManager	.M>	Jensemanager wyg	- 46.	>	Default (stdcall) 1: [esp] 00432138 L"192.168.164.130" 2: [esp+4] 00000000 2: [esp+4] 00000005
.text:00402	2B3D psexec.exe	\$283D #1	F 3D							4: [esp+C] 00430EF0 L"PSEXESVC"
d Dump 1	Dump 2 🔮	Dump 3	Dump 4	🗑 Dump 5	🥉 Watch 1	Locals	2 Struct	0018E298 0018E290	0043	432138 L"192.168.164.130" 000000
Address H	av				ASCTT	1		_ 0018E2A0	0006	DF003F

Figure 41

A new service called "PSEXESVC" is created by the process on the remote host (0xF01FF = SERVICE_ALL_ACCESS, 0x10 = SERVICE_WIN32_OWN_PROCESS, 0x3 = SERVICE_DEMAND_START):

• 0040277 • 0040277 • 0040277 • 0040277 • 0040278 • 0040278 • 0040278 • 0040278 • 0040278 • 0040278 • 0040278 • 0040278 • 0040278 • 0040279 • 0040278 • 0040278	6A 00 6A 00 6A 00 6A 00 51 88 4C 24 1C 6A 00 6A 00 51 88 44 24 30 66 FF 01 0F 00 52 50 51 FF 15 34 70 42 00 ***********************************	<pre>push 0 push 0 push 0 push 0 push 0 push ecx mov ecx,dword ptr ss:[esp+1C] push eax push</pre>	(m>) x87r6 000000000000000000000000000000000000) Unlock
Dump 1 Dump 2	🐇 Dump 3 🛛 🚷 Dump 4 🏻 🎯 D	ump 5 🥈 Watch 1 🜒 Locals 🖉 Struct	0018E24C 024E4E40 0018E250 00430EF0 return to psexec.00430EF0 from ???	
Address Hex 00430EF0 50 00 53 00 45 00430EF0 00 00 00 00 00 00 00430F10 00 00 00 00 00 00 00 00430F20 00 <	00 58 00 45 00 53 00 56 00 00 </th <th>ASCII 42 00 B.S.E.X.E.S.V.C. 00 00</th> <th><pre>0018254 00430EF0 feturn to psexec.00430EF0 from ??? 0018255 000F01F 0018256 0000003 0018264 00000003 0018264 00000000 00182656 00186780 L"%SystemRoot%\PSEXESVC.exe" 018267 0000000 0018277 0000000</pre></th> <th></th>	ASCII 42 00 B.S.E.X.E.S.V.C. 00 00	<pre>0018254 00430EF0 feturn to psexec.00430EF0 from ??? 0018255 000F01F 0018256 0000003 0018264 00000003 0018264 00000000 00182656 00186780 L"%SystemRoot%\PSEXESVC.exe" 018267 0000000 0018277 0000000</pre>	

Figure 42

The number of milliseconds that have elapsed since the system was started is extracted via a function call to GetTickCount:

	call ebp	ebp:Ge v	Default (stdcall)
ebp= <kernel32.gettickcount> (76ACD040)</kernel32.gettickcount>			2: [csp+4] 2F8C48C2 3: [csp+8] FFFFFFE 4: [csp+C] 76295212 sechost.76295212

Figure 43

The executable opens the newly created service using OpenServiceW (0xF01FF = **SERVICE_ALL_ACCESS**):

 00402838 00402830 00402830 00402838 	00402838 68 FF 01 0F 00 push F01FF 00402830 50 push eax 0040283E 51 push eax				x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0		
€IP 0040283F	FF 15 28 70 42 00	call dword ptr ds: [<&op	InServiceW>]	· · · · · · · · · · · · · · · · · · ·	Default (stdcall)		
dword ptr [00427028 <pse .text:0040283F psexec.ex</pse 	xec.&OpenServiceW>]= <advapi e:\$283F #1C3F</advapi 	32.OpenServiceW>			1: [csp+4] 00430EF0 L"PSEXESVC" 3: [csp+8] 000F01FF 4: [csp+C] 0018E7B0 L"%SystemRoot%\\PSEXESVC.exe		
Cump 1 Dump 2	🐇 Dump 3 🛛 🚷 Dump 4 🏾 🎯 D	ump 5 🥈 Watch 1 🔮 Locals	2 Struct	0018E25C 024 0018E260 004	4E4E40 430EF0 return to psexec.00430EF0 from ???		
Address Hex 00430EF0 50 00 53 00 45	00 58 00 45 00 53 00 56 00	ASCII 43 00 P.S.E.X.E.S.V.C.		▲ 0018E264 000 0018E268 001	DF01FF 18E7B0 L"%SystemRoot%\\PSEXESVC.exe"		

The "PSEXESVC" service is started using the StartServiceW routine:

	 00402855 00402857 00402859 	6A 00 6A 00 50	push 0 push 0 push eax		x875W_C1 0 x875W_C0 0 x875W_SF 0 x875W_P 0	x87SW_ES 0 x87SW_U 0
312	0040285A <	FF 15 2C 70 42 00	<pre>call dword ptr ds:[<&StartServiceW>]</pre>	>	Default (stdcall) 1: [esp] 024E1358	▼ 5 🗢 🗌 Unlod
.text:0040	285A psexec.exe:\$	3285A #1C5A	132.StartServicew>		2: [esp+4] 00000000 3: [esp+8] 0000000 4: [esp+C] 024E4E40	
Dump 1	🕴 Dump 2 🐇	Dump 3 🛛 🚷 Dump 4 🏾 🎯 Du	mp 5 🥈 Watch 1 🔮 Locals 🦻 Struct	0018E254 024 0018E258 000	E1358	

Figure 45

The file retrieves the current status of the above service by calling the QueryServiceStatus API:

d Dump 1	🛢 Dump 2 🐇	Dump 3 🛛 🚷 Dump 4 🛛 🧐	Dump 5 🥈 Watch 1 🔮 Locals 🤌 Struct	0018E254 024 0018E258 001	E1358 8E270
esi= <advap< th=""><th>132.QueryServices</th><th>5288C #1C8C</th><th>Lati C31</th><th>></th><th>Default (stdcall)</th></advap<>	132.QueryServices	5288C #1C8C	Lati C 31	>	Default (stdcall)
6.70	 00402885 00402886 00402887 	52 50 BF 01 00 00 00	push edx push eax mov edi,1		x875W_C1 0 x875W_C0 0 x875W_ES 0 x875W_SF 0 x875W_P 0 x875W_U 0

Figure 46

The next step of the execution flow is connecting with the PsExec service on the remote computer:

C:\Users\\Desktop\psexec.exe	-	×
PsExec v2.11 - Execute processes remotely Copyright (C) 2001-2014 Mark Russinovich Sysinternals - www.sysinternals.com		^
Connecting with PsExec service on 192.168.164.130		

Figure 47

The PsExec executables opens the "\pipe\PSEXESVC" pipe from the remote machine (0xC0000000 = **GENERIC_READ** | **GENERIC_WRITE**, 0x3 = **OPEN_EXISTING**):

50	 00404AA0 00404AA2 00404AA4 00404AA6 00404AA6 00404AA6 00404AA6 00404AA6 00404AA6 00404AA6 00404AA6 	6A 00 6A 00 6A 03 6A 00 6B 00 6B 00 00 C0 8D 94 24 EC 4A 00 00 52 EE D6	push 0 push 0 push 3 push 0 push 0 push c0000000 lea edx,dword ptr ss:[esp+4AEC] push edx	edx:L"	X8/18_4 S (Empty) X8/18_5 S (Empty) X87TW_6 3 (Empty) X87TW_7 3 (Empty) X87STW_6 3 (Empty) X87TW_7 3 (Empty) X87SW_8 0 x87SW_20 0 x87SW_20 0 x87SW_20 X87SW_8 0 x87SW_20 0 x87SW_25 0 x87SW_20 X87SW_50 0 x87SW_10 0 x87SW_10 0 x87SW_20
		1. 00 11 10 00	and and an an enterned and		Default (stdcall) 🔻 5 😩 🗌 Unlock
es1= <kerne< th=""><th>04AB7 psexec.exe:\$</th><th>(76ACDDE0) 4AB7 #3EB7</th><th></th><th></th><th>2: [esp+4] C000000 3: [esp+8] 0000000 4: [esp+C] 00000000</th></kerne<>	04AB7 psexec.exe:\$	(76ACDDE0) 4AB7 #3EB7			2: [esp+4] C000000 3: [esp+8] 0000000 4: [esp+C] 00000000
d Dump 1	8 Dump 2 🐇	Dump 3 💧 Dump 4 🎯 Dum	p 5 🥈 Watch 1 🚯 Locals 🤌 Struct	0018EBE4 00193 0018EBE8 C0000	604 L"\\\\192.168.164.130\\pipe\\PSEXESVC"
Address	Hex		ASCII	0018EBEC 00000	000
001936D4 001936E4 001936F4	SC 00 5C 00 31 00 38 00 2E 00 31 00 30 00 5C 00 70 00	39 00 32 00 2E 00 31 00 3 36 00 34 00 2E 00 31 00 3 69 00 70 00 65 00 5C 00 5	00 N.\.1.9.21.6. 00 81.6.41.3. 00 0.\.p.1.p.e.\.P.	0018EBF4 00000 0018EBF8 00000 0018EBF5 00000	0003 10000 10000

Figure 48

The pipe mode is modified by calling the SetNamedPipeHandleState API (0x2 = **PIPE_READMODE_MESSAGE**):



Interestingly, there are some indirect calls (jmp instructions instead of call instructions) that appear in the code. For example, the RtIInitUnicodeString function is used to initialize the "\Device\LanmanRedirector\<Computer name\IP Address>\ipc\$" Unicode string:

EIP 004023EE 004023EE 004023F3 004023F3 004023F3 004023F4 004023F4 004023F4 004023F4 004023F5 004023F5 004023F5 004023F5 FF 15 E8 71 42 00 004023F5 FF 10 FF 1	push psexec.429A86 push psexec.429A86 call dword ptr ds:[<&GetModuleHandlew>] push eax call dword ptr ds:[<&GetProcAddress>] nov dword ptr ds:[<&RtlinitUnicodestring>],eax imp eax	429A88 429ASC eax:Rt eax:Rt eax:Rt >	x87Statusword 0000 x87Sw_B 0 x87Sw_C3 0 x87Sw_C2 0 x87Sw_C1 0 x87Sw_C0 0 x87Sw_E5 0 x87Sw_5F 0 x87Sw_P 0 x87Sw_U 0 Default (stdcal) ▼ 5 ↓ Unlock 1 fespt+1 0018E358
Jump 15 taken eax= <ntdll.rtlinitunicodestring> (77050DA0) .text:00402405 psexec.exe:\$2405 #1805</ntdll.rtlinitunicodestring>			<pre>2: [esp+8] 0018E3C8 L"\\Device\\LanmanRedirector 3: [esp+C] 76ACDC00 <kernel32.createeventw> 4: [esp+10] 0018EC28</kernel32.createeventw></pre>
🕼 Dump 1 👶 Dump 2 🐇 Dump 3 🌒 Dump 4 🎯 Dump 5	🕉 Watch 1 🚯 Locals 🦻 Struct	0018E320 0040 0018E324 0018	262C return to psexec.0040262C from psexec.00402 E358
Address Hex 001863C8 5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 001863D8 4C 00 61 00 6E 00 6D 00 61 00 6E 00 52 00 65 00 001863E8 64 00 69 00 72 00 65 00 63 00 74 00 6F 00 72 00 001863E8 5C 00 31 00 39 00 22 00 2E 00 31 00 36 00 38 00 00186408 2E 00 31 00 36 00 34 00 2E 00 31 00 33 00 30 00 00186408 5C 00 50 00 70 00 63 00 24 00 00 07 F 00 00 00	ASCII %.D.e.v.1.C.e.\. L.a.n.m.a.n.R.e. d.1.r.e.c.t.o.r. (1.9,21.6.8. 1.6.41.3.0. \.1.9.2	0018E328 0018 0018E32C 76AC 0018E330 0018 0018E334 0000 0018E338 0000 0018E336 024C 0018E340 7707	E3C8 ["\\Device\\LanmarRedirector\\192.168.164.1 2000 kernel32.CreateEventW 2000 200 2000 2

Figure 50

The file opens the "\\192.168.164.130\ipc\$" share using NtOpenFile (0x100001 = FILE_READ_DATA | SYNCHRONIZE, 0x1 = FILE_SHARE_READ, 0x90 = FILE_SYNCHRONOUS_IO_ALERT | FILE_CREATE_TREE_CONNECTION):



Figure 51

PsExec obtains connection information by calling the NtFsControlFile function with a specific control code 0x1401a3 = **FSCTL_NETWORK_GET_CONNECTION_INFO**:

00402449 68 AC 9A 42 00 00402453 FF 15 D0 71 42 00 00402453 FF 15 D0 71 42 00 00402455 FF 15 E8 71 42 00 0040245A FF 15 E8 71 42 00 0040245A A3 14 21 43 00 00402465 FF 10 F 10	<pre>push psexec.429AAC push psexec.429ASC call dword ptr ds:[<&GetModuleHandlew>] push eak call dword ptr ds:[<&GetProcAddress>] mov dword ptr ds:[<&MtFsControlFile>],eax]mp eax</pre>	429AAC 429ASC eax:Nt ceax:
Jump 1s taken eax= <rtdll.xtfscontrolfile> (7704EB10) .text:00402465 psexec.exe:\$2465 #1865 Dump 1 Dump 1 Dump 2 Dump 1 Dump 2</rtdll.xtfscontrolfile>	mp 5 🥈 Watch 1 🔕 Locals 🎾 Struct	2: [esp+8] 00000000 3: [esp+2] 00000000 4: [esp+10] 00000000 00165000 00402688 [return to psexec.00402688 from psexec.00402
Address Hex 0018E334 F# 00	ASCII ASCII 0 00 0 00 	0138326 0000000 0138307 0000000 0138310 0000000 0138311 0108338 0138312 0108338 0138312 0140133 0138312 0140133 0138322 0000024 0138328 0138384 0138328 0138384 0138328 0138384
Figure 52		00185328 0000044

There is a second call to NtFsControlFile that sends another control code 0x1401AC = **FSCTL_NETWORK_DELETE_CONNECTION**:

Jump 1s take eax= <ntdll.< th=""><th>0040244 0040244 0040245 0040245 0040245 0040245 0040246 0040246 € 0040246 € 0040246 €</th><th>9 68 AC 68 5C 3 FF 15 9 0 A FF 15 0 A3 14 5 FF 15 0 A3 14 5 FF 15 0 A704EB</th><th>9A 42 00 9A 42 00 D0 71 42 0 E8 71 42 0 21 43 00</th><th>00</th><th>push psexec.4 push psexec.4 call dword ptr push eax call dword ptr mov dword ptr jmp eax</th><th>29AAC 29A5C r ds:[<&Gi r ds:[<&Gi ds:[<&Nti</th><th>etModuleHa etProcAddr FsControlF</th><th>ndlew>] ess>] lle>],eax</th><th>4294 4294 eax: eax: eax:</th><th>AC SC XI NT XI NT XI NT De 11 2: 3: 4:</th><th>87Statusw 87SW_B (87SW_C1 (87SW_SF (87SW_SF (fault (stdcal [esp+4] [esp+2] [esp+10</th><th>vord 0000 x87SW_C3 x87SW_C0 x87SW_P 000002F4 0000000 00000000 00000000 00000000</th><th>0 x875% 0 x875% 0 x875%</th><th>L_C2 0 L_ES 0 L_U 0</th><th>5 🔹 🗌 Unlock</th></ntdll.<>	0040244 0040244 0040245 0040245 0040245 0040245 0040246 0040246 € 0040246 € 0040246 €	9 68 AC 68 5C 3 FF 15 9 0 A FF 15 0 A3 14 5 FF 15 0 A3 14 5 FF 15 0 A704EB	9A 42 00 9A 42 00 D0 71 42 0 E8 71 42 0 21 43 00	00	push psexec.4 push psexec.4 call dword ptr push eax call dword ptr mov dword ptr jmp eax	29AAC 29A5C r ds:[<&Gi r ds:[<&Gi ds:[<&Nti	etModuleHa etProcAddr FsControlF	ndlew>] ess>] lle>],eax	4294 4294 eax: eax: eax:	AC SC XI NT XI NT XI NT De 11 2: 3: 4:	87Statusw 87SW_B (87SW_C1 (87SW_SF (87SW_SF (fault (stdcal [esp+4] [esp+2] [esp+10	vord 0000 x87SW_C3 x87SW_C0 x87SW_P 000002F4 0000000 00000000 00000000 00000000	0 x875% 0 x875% 0 x875%	L_C2 0 L_ES 0 L_U 0	5 🔹 🗌 Unlock
Dump 1	B Dump 2	S Dump 3	Dump 4	🗑 Dump 5	Watch 1	locals	2 Struct		0018E300	0040271	D return	n to psexec	.0040271D	from p	osexec.00402
Address He: 0018E360 00 0018E370 00 0018E380 24 0018E380 21 0018E380 00 0018E380 C2 0018E380 00	x 00 00 00 00 00 04 20 00 00 00 00 00 00 00 00 00 8F 6E 46 F9 00 00 00 00	00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 00 FA 4A 3A 03 00 00 00 50	00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 24 00 00 00 10 00 00 00 10 A3 6D ED 65 00 76 00	ASCII \$ Å. nFüúj:ö. 	\$.£mi e.v.		^	0018E308 0018E30C 0018E310 0018E314 0018E318 0018E312 0018E320 0018E324 0018E328	0000000 0000000 0018E33 0014014 0018E36 0000002 0000000 0000000	00 00 18 18 16 10 10 10 10 10 10 10 10 10 10 10 10 10				

The major and minor version numbers of the OS are retrieved using the GetVersion API:

ETP 00104619 FF 15 68 71 42 00	call dword ptr ds:[<&GetVersion>]	>	Default (stdcall)	▼ 5 ↓ Unlocker
<pre>dword ptr [00427168 <psexec.&getversion>]=<kernel32 #3f19<="" .text:00404819="" pre="" psexec.exe:\$4819=""></kernel32></psexec.&getversion></pre>	.GetVersion>		2: [esp+4] 00432138 L"1 3: [esp+8] 76A750B0 <ke 4: [esp+C] 76A759E0 <ke< td=""><td>92.168.164.130" rnel32.GetProcAddress> rnel32.LoadLibraryW></td></ke<></ke 	92.168.164.130" rnel32.GetProcAddress> rnel32.LoadLibraryW>
— ! — /			1	

Figure 54

The information extracted above is written to the "\pipe\PSEXESVC" pipe by calling TransactNamedPipe:

dword ptr [(● 0040481F 00404821 00404825 00404826 00404826 00404826 00404828 00404833 00404833 00404833 00404839 00404835 00404855 0004055 0004055 0004055 0004055 0004055 0004055 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 000455 0005 000455 000455 000455 0005 0	6A 00 8D 4C 24 1C 51 8B 00 0C 11 43 6A 0C 8D 54 24 7C 52 89 44 24 48 6A 0C 8D 54 24 7C 52 89 44 24 48 6A 0C 51 FF 15 F4 70 42 .4TransactNamedP1	00 mov ecx, d lea ecx, d push ecx mov ecx, d lea edx, d push c lea edx, d push c lea edx, d push ecx mov down d lea edx, d push ecx call dwor call dwor push ecx push ecx	<pre>word ptr ss:[esp+1C] word ptr ds:[43110C] word ptr ss:[esp+7C] i ptr ss:[esp+48], eax word ptr ss:[esp+48] id ptr ds:[<&TransactNam actNamedPipe></pre>	edPipe>]		x87TagWord FFFF x87Tw_0 3 (Empty) x87Tw_1 3 x87Tw_2 3 (Empty) x87Tw_1 3 x87Tw_4 3 (Empty) x87Tw_5 3 x87Tw_6 3 (Empty) x87Tw_7 3 x87Sy_6 0 x87Sw_C 3 0 x87Sw x87Sy_6 0 x87Sw_C 0 x87Sw x87Sy_6 0 x87Sw_C 0 x87Sw x87Sy_6 0 x87Sw_C 0 x87Sw x87Sy_6 0 x87Sw_2 0 x87Sw x87Sw_6 0 x87Sw_2 0 x87Sw_2 0 x87Sw x87Sw_6 0 x87Sw_2 0	(Empty) (Empty) (Empty) (Empty) _C2 0 _ES 0 _U 0
.text:004048	33F psexec.exe:\$	4B3F #3F3F						
d Dump 1	🖲 Dump 2 🐇 I	Dump 3 🛛 🕘 Dump 4	🞯 Dump 5 🥈 Watch	h 1 🕘 Locals 🖉 Struct		0018EBE4 000 0018EBE8 001	002F8 8EC34	
Address He: 0018EC34 C8 0018EC44 00 0018EC54 00	00 00 00 06 02 00 00 00 00 00 00 00 00 00 00 00 00	F0 23 01 00 00 00 00 00 00 00 00 00 00 00 00 00	ASCII 00 00 00 00 02 00 00 00 00 00 00 00 	*	^	0018EBEC 000 0018EBF0 001 0018EBF4 000 0018EBF8 001	0000C 8EC70 0000C 8EC18	

Figure 55

The binary acquires a handle to a key container within a particular CSP (cryptographic service provider) via a call to CryptAcquireContextW (0x18 = **PROV_RSA_AES**):

	 004015E1 004015E3 004015E5 004015E7 004015E9 	6A 00 6A 18 6A 00 6A 00 56	pu pu pu pu	sh 0 sh 18 sh 0 sh 0 sh esi					×87 ×87 ×87	StatusWord 0000 SW_B 0 x87SW_C3 SW_C1 0 x87SW_C0 SW_SF 0 x87SW_P	0 x87SW_C2 0 x87SW_ES 0 x87SW_U	0000	
edi= <advapi< th=""><th>EA psexec.exe:</th><th>EF D7 Contextw> (73A60) \$15EA #9EA</th><th>80)</th><th>l edi</th><th></th><th>_</th><th>_</th><th>edi:Cr</th><th>Defa 1: 2: 3: 4:</th><th>ult (stdcal) [esp] 0018EC14 [esp+8] 0000000 [esp+8] 0000000 [esp+C] 0000018</th><th>•</th><th>5</th><th>🗘 🗌 Unlod</th></advapi<>	EA psexec.exe:	EF D7 Contextw> (73A60) \$15EA #9EA	80)	l edi		_	_	edi:Cr	Defa 1: 2: 3: 4:	ult (stdcal) [esp] 0018EC14 [esp+8] 0000000 [esp+8] 0000000 [esp+C] 0000018	•	5	🗘 🗌 Unlod
Dump 1	🕴 Dump 2 🛛 🐇	Dump 3 🛛 🕚 Dump	4 🎯 Dump 5	Watch 1	l Locals	Struct	00	LSEBEC 00 LSEBC0 00 LSEBC4 00	18EC14				
0018EC70 C8	00 00 00 06 01	B0 1D 01 00 00	00 58 4A 00 00 E		(J.,		00	LSEBCS 00	000018				

Figure 56

CryptCreateHash is utilized to create a hash object (0x8004 = **CALG_SHA1**):

 0040198 0040198 0040198 0040198 0040199 0040199 0040199 0040199 	8 52 9 6A 00 8 6A 00 0 88 48 12 0 88 45 00 3 68 04 80 00 00 8 50	<pre>push edx push 0 mov byte ptr ds:[eax+i2],c1 mov eax,dword ptr ss:[ebp] push 8004 push eax</pre>	eax+12	X8/1W_5 3 (Empty) X8/1W_/ 3 (Empty) X87StatusWord 0000 X87SW_8 0 X87SW_C3 0 X87SW_C2 0 X87SW_C1 0 X87SW_C0 0 X87SW_E5 0 X87SW_5F 0 X87SW_P 0 X87SW_E 0 X87SW_5F 0 X87SW_P 0 X87SW_E 0
EIP 0040199	FF 15 58 70 42 00	[Call dword ptr ds:[<&CryptCreateHasn>]		Default (stdcall) 🔻 5 💠 🗌 Unlock
dword ptr [00427058 <ps .text:00401999 psexec.et</ps 	exec.&CryptCreateHash>]= <adva xe:\$1999 #D99</adva 	p132.CryptCreateHash>		1: [esp] 024E56C0 .4&CPAcquireContext> 2: [esp+4] 00008004 4: [esp+6] 00000000 4: [esp+6] 00000000
Cump 1 B Dump 2	💰 Dump 3 🛛 🚷 Dump 4 🛛 🎯 Dum	np 5 🥈 Watch 1 🚯 Locals 🐉 Struct	0018EBC8 0248 0018EBCC 000	E66C0 08004
Address Hex 0018EBEC 14 EC 18 00 7D	48 40 00 28 EC 18 00 14 EC 1	ASCII 8 00 11.3K@.(11.	0018EBD0 0000 0018EBD4 0000	00000 00000 85855

Figure 57

The executable hashes a buffer that contains 16 bytes (probably generated based on the GetTickCount call) and the "Sysinternals Rocks" string:

004019A8 6A 00 004019AA 56 004019AA 55 004019AB 53 004019AC 51	push 0 push est push ebx push ecx		X875W_B 0 X675W_C3 0 X875W_C2 0 X875W_C1 0 X875W_C0 0 X875W_E5 0 X875W_5F 0 X875W_P 0 X875W_U 0
EIP 004019AD FF 15 5C 70 42 00	call dword ptr ds:[<&CryptHashData>]	> ×	Default (stdcall)
dword ptr [0042705C <psexec.&crypthashdata>]=<advapi32 .text:004019AD psexec.exe:\$19AD #DAD</advapi32 </psexec.&crypthashdata>	CryptHashData>		1: [csp14] 043418A0 3: [csp44] 043418A0 3: [csp45] 00000023 4: [csp45] 00000000
🕼 Dump 1 🕴 Dump 2 🐇 Dump 3 🌒 Dump 4 🎯 Dump 5	i 🍯 Watch 1 🌒 Locals 🤌 Struct 0018E	BD4 0434	E1358 418A0
Address Hex	ASCII	BD8 0000	00023
043418A0 C2 8F 6E 46 F9 FA 4A 3A 03 1F F5 88 1D A3 60 04341880 53 79 73 69 6E 74 65 72 6E 61 6C 73 20 52 6F 043418C0 68 73 00 A8 A8 A8 A8 A8 A8 A8 A8 FE EE FE EE	BD A. nFu01:otm1 Outset 63. Sysinternals Roc 001881 FE KSexexexexet[p1] 001881	BE0 76A0 BE4 0043 BE8 764	DDDD0 kernel32.CreateFileW 32138 L ¹¹ 92.168.164.130 ¹

An AES256 key is derived from the SHA1 hash using CryptDeriveKey (0x6610 = **CALG_AES_256**):

004019C2 52 004019C3 6A 00 004019C5 50 004019C5 50 004019C5 68 10 66 00 00 004019C6 51	push edx push o push eax push 6610 push ecx	x875tatusWord 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_E10 x875w_KC0 0 x875w_E5 0 x875w_SF 0 x875w_P 0 x875w_U 0
<pre>EIC</pre>	CryptDeriveKey>	V Default (stdcal) ▼ 5 □ Unlock 1: [esp1] 024E66C0 <&CPAcquireContext> 2: [esp+4] 00006610 3: [esp+6] 024E358 <&CPCreateHash> 4: [esp+c] 00000000 1: [esp+c] 0000000 1: [esp+c] 00000000 1: 1: [esp+c] 00000000 1: 1: [esp+c] 0000000 1: 1: [esp+c] 0000000 1: 1: [esp+c] 0000000 1: 1: [esp+c] 0000000 1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1: 1:
🕼 Dump 1 🗯 Dump 2 🐇 Dump 3 🔮 Dump 4 🎯 Dump 5	Watch 1 Icoals Struct O018EBCC O018EBC0 O018EBC0	124E66C0 00006610
Address Hex 0018EC24 00 00 00 00 10 00 00 00 10 00 00 78 18 34 0	ASCII 0018EBD4 0 0018EBD4 0 0018EBD4 0 0018EBD4 0 0018EBD4 0	(24£1358) 10000000 1018EC24

Figure 59

The process identifier is obtained via a function call to GetCurrentProcessId:

€10 → 00404C1A FF D3 <	call ebx	ebx:Ge v	Default (stdcall)	▼ 5 C Unlocker
ebx= <kernel32.getcurrentprocessid> (764CDB30)</kernel32.getcurrentprocessid>			2: [esp+4] 00432138 L"1 3: [esp+8] 76A75080 <ke 4: [esp+C] 76A759E0 <ke< td=""><td>92.168.164.130" rnel32.GetProcAddress> rnel32.LoadLibraryw></td></ke<></ke 	92.168.164.130" rnel32.GetProcAddress> rnel32.LoadLibraryw>
. TEXT: 00404CIA pSEXEC. EXE: SACIA #401A			1	1

Figure 60

An event object called "Global\PSEXESVC-<Computer Name\IP address>-<Process ID>" is created:

	 00404C3F 00404C40 00404C42 00404C44 	51 6A 00 6A 00 6A 00	push ecx push 0 push 0 push 0		ecx:L"	x875W_B 0 x875W_C x875W_C1 0 x875W_C x875W_SF 0 x875W_F	3 0 x87SW_C2 0 0 0 x87SW_ES 0 0 x87SW_U 0
	→• 00404C46 <	FF D7	call edi		edi:cn v	Default (stdcall)	▼ 5 😫 🗆 Unlock
edi= <kernel< th=""><th>32.CreateEventW></th><th>(76ACDC00) 4C46 #4046</th><th></th><th></th><th></th><th>2: [esp+4] 0000000 3: [esp+8] 0000000 4: [esp+C] 001936D4</th><th>L"Global\\PSEXESVC-192.168.1</th></kernel<>	32.CreateEventW>	(76ACDC00) 4C46 #4046				2: [esp+4] 0000000 3: [esp+8] 0000000 4: [esp+C] 001936D4	L"Global\\PSEXESVC-192.168.1
Dump 1	🖲 Dump 2 🐇 I	Dump 3 🚯 Dump 4	🗑 Dump 5 🥈 Watch 1 💧	Locals 🖉 Struct	0018EBF0 000 0018EBF4 000	00000	
Address He	EX	C2 00 C1 00 CC 00	ASCII		^ 0018EBF8 000 0018EBFC 001	00000 936D4 L"Global\\PSEXE	ESVC-192.168.164.130-6052"

Figure 61

PsExec displays a message that states the process name passed as a parameter is going to be started on the remote host:



Figure 62

A buffer that contains the following information is encrypted using the AES256 algorithm (figure 63): size of the buffer - 8, process ID in hex, local computer name, and the process that will be spawned.



The encrypted buffer size and the encrypted buffer are written to the "\\192.168.164.130\pipe\PSEXESVC" pipe:

• 00401733 6A 00	push 0	x87TW_6 3 (Empty) x87TW_7 3 (Empty)
00401735 80 54 24 14 00401739 52	push edx	edx:" x87StatusWord 0000
0040173A 6A 04 0040173C 8D 44 24 30	push 4	x875W_B 0 x875W_C3 0 x875W_C2 0
e 00401740 50	push eax	eax:"' x87SW_SF 0 x87SW_P 0 x87SW_U 0
00401741 57	push edi	
		Default (stdcall) 🔻 5 🗘 🗌 Unloc
abu damala britanila (annana)		1: [esp] 00000300
eDx= <kernels2.writerile> (/6ACE250)</kernels2.writerile>		2: [esp+4] 0018EBFC "J" 3: [esp+8] 00000004
.text:00401742 psexec.exe:\$1742 #B42		4: [esp+C] 0018EBE8 "'J"
2 Dump 1 Dump 2 Source Dump 3 Dump 4 Dump 5	Watch 1 0018 2 Struct 0018	EC4 00000300
Address Hex	ASCII	BCC 0000004
0018EBFC 60 4A 00 00 08 00 00 038 21 43 00 80 50 A7 76	0018 0018	EBD0 0018EBE8 " J" EBD4 00000000
Figure 64		
0040174C 6A 00 0040174E 8D 4C 24 14	lea ecx.dword ptr ss:[esp+14]	[esp+1 x87StatusWord 0000
00401752 51	push ecx	x875W_B 0 x875W_C3 0 x875W_C2 0
00401754 56	push esi	x875W_SF 0 x875W_P 0 x875W_U 0
00401755 57	push edi	
	tinh and	Default (stdcail) • 5 🛊 🗌 Unloc
aby-warnallah (76466360)		1: [esp] 00000300
CDA-CREINEISZIWITCEPTTES (/OACE250)		2: [esp+4] 04314C58 3: [esp+8] 00004A60
.text:00401756 psexec.exe:\$1756 #B56		4: [esp+C]_0018EBE8
	0018	2269 00000300
🕼 Dump 1 🕴 Dump 2 🥌 Dump 3 🔮 Dump 4 🤡 Dump 5	🐻 Watch 1 🔮 Locals 🦉 Struct 0018	EBC8 04314C58
Address Hex	ASCII ^ 0018 0018	EBCC 00004A60 EBD0 0018EBE8
04314C58 88 EE 9E 18 75 CF 45 74 83 23 89 A0 30 55 CF 63	0018	EBD4 00000000

Figure 65

PsExec waits until an instance of the "\\192.168.164.130\pipe\PSEXESVC-<Local computer name>-<Process ID>-stdin" pipe is available for connection (see figure below). This pipe and the others that correspond to the standard output\error are created by the PSEXESVC process started on the remote host. The entire execution flow will be explained in the 2nd part of the blog post, when we'll also analyze the execution of that process.



Figure 66

The executable opens the above named pipe using CreateFileW (0x40000000 = **GENERIC_WRITE**, 0x3 = **OPEN_EXISTING**):

ETØ	 0040452D 0040452F 0040453F 00404533 00404535 00404537 00404537 00404538 	6A 00 6A 00 6A 03 6A 00 6A 00 53 57 57		oush 0 oush 0 oush 3 oush 0 oush 0 oush ebx oush edi				edi:L	x875tatusWord 0000 x875tatusWord 0000 x875tatusWord 0x x875tatus x875tatusWord 0x x875tatus x875
ebp= <kernel< th=""><th>32.CreateFile</th><th>N> (76ACDDE0)</th><th></th><th></th><th></th><th></th><th></th><th>3</th><th>Default (stdcal) 5 C Unlock 1: [esp] 001936D4 L"\\\192.168.164.130\\pipe\\P 2: [esp+4] 4000000 3: [esp+8] 00000000</th></kernel<>	32.CreateFile	N> (76ACDDE0)						3	Default (stdcal) 5 C Unlock 1: [esp] 001936D4 L"\\\192.168.164.130\\pipe\\P 2: [esp+4] 4000000 3: [esp+8] 00000000
.text:00404	539 psexec.ex	e:\$4539 #3939							4: [esp+c]_00000000
d Dump 1	Dump 2	🖇 Dump 3 🛛 💧 D	0ump 4 i 🎯 Dump 5	🐻 Watch 1	locals	Struct	00	18EBAC 18EBB0	001936D4 L"\\\192.168.164.130\\pipe\\PSEXESVC-DESKT0
Address He 001936D4 5C 001936E4 38 001936F4 30	x 00 5C 00 31 00 2E 00 31 00 5C 00 70	00 39 00 32 00 00 36 00 34 00 00 69 00 70 00	2E 00 31 00 36 00 2E 00 31 00 33 00 65 00 5C 00 50 00	ASCII N.\.1.9.2 81.6.4 0.\.p.1.p.e.	. 6. . 3. . P.		000 000 000 000 000	18EBB4 18EBB8 18EBBC 18EBC0 18EBC4	00000000 00000000 00000000 00000000 0000

A similar approach is applied to the "\\192.168.164.130\pipe\PSEXESVC-<Local computer name>-<Process ID>-stdout" and "\\192.168.164.130\pipe\PSEXESVC-<Local computer name>-<Process ID>-stderr" pipes, with only one notable different – the requested access is 0x80000000 (**GENERIC_READ**).

The binary retrieves a pseudo handle for the current process via a call to GetCurrentProcess:

EIP 00404F88 FF D6	call est	es1:Ge v	Default (stdcal)	▼ 5 👽 🗌 Unlocke
es1= <kernel32.getcurrentprocess> (76ACDB20) .text:00404FBB psexec.exe:\$4FBB #43BB</kernel32.getcurrentprocess>			2: [esp+4] 10000000 3: [esp+8] 0000000 4: [esp+C] 0000000	

Figure 68

A new thread is created by the process. Please note that the starting address of the thread is different than the actual relevant function, which is sub_404240 in this case (0x4 =

CREATE_SUSPENDED):

312	00408F0D 00408F0E 00408F10 00408F11 00408F16 00408F19 00408F1C 00408F1D 00408F1D	56 6A 04 56 68 35 8E 40 00 FF 75 0C 89 7E 54 53 89 46 58 FF 15 40 72 42 0	push es push 4 push es push push mov dwoi push eb mov dwoi call dw	1 exec.408E35 ord ptr ss:[ebp rd ptr ds:[esi- x rd ptr ds:[esi- ord ptr ds:[<&	94C 54],edi 58],eax reateThread>]		es1+54	x87TW_6 x87Stat x87SW_B x87SW_C x87SW_C x87SW_S	3 (Empty) x usWord 0000 0 x875W_C3 1 0 x875W_C 5 0 x875W_P 1 0 x875W_P 1 0 x875W_P 1 0 x875W_P	0 x875W_C2 (0 x875W_C2 (0 x875W_ES (0 x875W_U	7) 2) 2) 2) 2) 2) 2) 2) 2) 2) 2) 2) 2) 2)
dword ptr [00	0427240 <psexec.< th=""><th>&CreateThread>]=<</th><th>kerne132.CreateThre</th><th>ead></th><th></th><th></th><th></th><th>1: [esp 2: [esp 3: [esp 4: [esp</th><th>00000000 4] 00000000 8] 00408E35 ps C] 043118E0</th><th>exec.00408E35</th><th></th></psexec.<>	&CreateThread>]=<	kerne132.CreateThre	ead>				1: [esp 2: [esp 3: [esp 4: [esp	00000000 4] 00000000 8] 00408E35 ps C] 043118E0	exec.00408E35	
d Dump 1	8 Dump 2 🐇 D	ump 3 🚯 Dump 4	🞯 Dump 5 🥂 Wa	tch 1 🚯 Locals	2 Struct	0018	EBB0 000 EBB4 000	00000			
Address Hex 043118E0 00 0 043118F0 00 0 04311900 00 0	00 00 00 00 00 00 00 00 00 01 00 0 00 00 00 00 00 00 0	00 00 00 00 00 00 00 00 00 00 00 00 00 0	ASCII 00 00 00 00 00 00 00 00 00 00 00 00			0018 0018 0018 0018 0018	EBB8 004 EBBC 043 EBC0 000 EBC4 043 EBC8 76A	08E35 pse 118E0 00004 118E0 CDB20 ker	nel32.GetCurre	ntProcess	
04311910 00 0 04311920 00 0 04311930 00 0	00 00 00 00 00 00 00 00 00 00 00 00 00	0 00 00 00 00 00 00 0 00 00 00 00 00 0 00 68 EC 18 00	00 00 00 00 00 00 00 00 00 08 C9 42 00@B	Ø.hìØÉB.		0018	EBCC 000 EBD0 76A	00000 ker	nel32.GetCurre	ntProcessId	

Figure 69

The thread handle is duplicated using DuplicateHandle (0x10000000 = **GENERIC_ALL**):

 00404F9D 6A 00 00404F9F 6A 00 00404FA1 89 54 24 48 00404FA3 8D 54 24 50 00404FB3 94 24 76 00404FB3 94 24 77 00404FB3 94 42 77 00404FB3 94 42 477 00404FB3 96 44 24 70 00404FB3 90 404FC3 80 44 24 70 00404FC3 80 44 24 000 00404FC4 80 40 42 40 00 00404FC5 80 40 42 000 00404FC5 80 40 42 000 00404FC5 80 44 24 000 00404FC5 80 45 24 000 00404FC5 80 42 20 00 00404FC5 80 54 00 00404FC3 90 404FC3 90 404FC3 90 404FC3 90 404FC3 90 404FC3 90 404FC3 90 404FC4 90 404FC5 90 40 404FC3 90 40 404FC3 90 40 40 90 40 404FC3 90 40 40 40 90 40 404FC3 90 40 40 40 90 40 40 40 90 40 40	push 0 push 0 mov dword pt ss: [esp+48],edx push 10000000 lea ecx,dword pt ss: [esp+50] lea ecx,dword pt ss: [esp+70],ecx mov dword ptr ss: [esp+72],eax call esi push eax lea eax,dword ptr ss: [esp+7C] push eax dd esp,C push eax call esi mov eax push eax pus	esi:Ge esi:Ge edi:Du	X8771 00000000000000000000 572 Empty 0.00000000 X8772 00000000000000000000000 572 Empty 0.00000000 X8774 0000000000000000000 572 Empty 0.00000000 X8776 0000000000000000000 574 Empty 0.00000000 X8776 000000000000000000 575 Empty 0.00000000 X8776 000000000000000000 576 Empty 0.00000000 X8777 000000000000000000 576 Empty 0.00000000 X8777 000000000000000000 576 Empty 0.00000000 X8778 (2000000000000000000000000000000000000
EIP 00404FDC FF D7	call edi	edi:Du 🗸	Default (stdcal)
<pre>di=<kernel32.duplicatehandle> (76ACDB90) .text:00404FDC psexec.exe:\$4FDC #43DC</kernel32.duplicatehandle></pre>		,	1: [esp] FFFFFFF 2: [esp+4] 0000330 3: [esp+6] FFFFFFF 4: [esp+C] 0018EC44
Dump 1 Dump 2 South Dump 3 Dump 4 Dump 5	🖇 Watch 1 🔮 Locals 🤌 Struct	0018EBE4 FFFF	FFFF 0330
Address Hex 0018EC44 00	ASCII A	0018EBEC FFFF 0018EBF0 0018 0018EBF4 1000 0018EBF8 0000 0018EBFC 0000	FFFF EC44 00000 00000
Figure 70			

The CreateThread API is used to create two threads that will eventually execute the sub_4043D0 and sub_404190 functions (0x4 = **CREATE_SUSPENDED**):

	push esi push 4 push sexec.408E35 push dword ptr 55:[ebp+C] mov dword ptr ds:[esi+54],edi push doxd ptr ds:[esi+54],eax coll word ptr ds:[sexecterhread>] eateThread>	X8/1W_4 s (Empty) X8/1W_5 s (Empty) X87TW_6 3 (Empty) X87TW_7 3 (Empty) X87StatusWord 0000 X87SW_C2 0 X87SW_6 0 X87SW_C2 0 X87SW_5 0 X87SW_5 0 X87SW_D 0 X87SW_5 0 X87SW_5 0 X87SW_D 0 X87SW_2 0 Default (stdcal) ▼ 1: [esp1 00000000 1: [esp1 00000000 2: [esp-4] 00000000 1: [esp-4] 04000000 2: [esp-4] 040000000 1: [esp-4] 0430185
.text:00408F20 psexec.exe:\$8F20 #8320	0018EB80 000	000000
Case Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Address Hex 00.00.00 00.00.00 00.00	6 Watch 1 Uccas 2 ^o Struct 0018EB84 001 ASCII	000000 008255 psexec.00408E35 311810 00004 311810
04311840 00 <	0018EG6 76 0018EB0 76 0018EB0 76 0018EB0 76	ACDB20 kernel32.GetCurrentProcess ACDB30 kernel32.DuplicateHandle ACDB30 kernel32.GetCurrentProcessId
Figure 71		
	push esi push 4 push sexec, 408E35 push baced ptr ss:[ebp+C] mov dword ptr ds:[esi+54],edi push ebx mov dword ptr ds:[esi+58],eax cull dword ptr ds:[esi+58],eax	x871m_4 3 (Empty) x871m_5 3 (Empty) x871m_6 3 (Empty) x871m_7 3 (Empty) x875xtatusWord 0000 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C3 0 x875W_C5 0 x875W_C5 0 x875W_5 0 x875W_C0 0 x875W_U 0 0
• •		Default (stdcall)
<pre>dword ptr [00427240 <psexec.&createthread>]=<kernel32.cr .text:00408F20 psexec.exe:\$8F20 #8320</kernel32.cr </psexec.&createthread></pre>	eateThread>	2: [esp+4] 00000000 3: [esp+8] 00408E35 psexec.00408E35 4: [esp+C] 04311D40
Image: Constraint of the state of	Watch 1 Locals Struct ODISEE00 OOI38884 OOI38884 OOI38884 OOI38884 OOI38886 OO OOI38886	000000 00000 408E35 psexec.00408E35 31D040 000004 310040
04311D60 00 <	0018EBC8 76 0018EBC7 76 0018EB00 76	ACDB20 kernel32.GetCurrentProcess ACDB30 kernel32.DuplicateHandle ACDB30 kernel32.GetCurrentProcessId

Figure 72

The PsExec executable changes the title for the console window using SetConsoleTitleW:

• 00405084	50	push eax	eax:L"		
00405085 00405085	FF 15 E8 70 42 00	call dword ptr ds:[<&setConsoleTitlew>]	,	Default (stdcall)	👻 5 🗘 🗌 Unlod
dword ptr [004270E8 <psex .text:00405085 psexec.exe</psex 	<pre>sec.&SetConsoleTitleW>]=<ke st\$5085 #4485</ke </pre>	ernel32.SetConsoleTitlew>		1: [esp] 00193CEC L"\ 2: [esp+4] 0000008 3: [esp+8] 00432138 L 4: [esp+C] 76A750B0 <	<pre>\\\192.168.164.130: cmd.exe "192.168.164.130" kernel32.GetProcAddress></pre>

Figure 73

The binary performs a call to WaitForMultipleObjects in order to suspend the process until the above threads finish and the event object created above is in the signaled state:

	 00405088 0040508D 0040508F 00405095 00405099 0040509A 	6A FF 6A 00 8B 35 14 71 42 00 8D 4C 24 48 51 6A 04	<pre>push FFFFFFFF push 0 mov est,dword ptr ds: [*&WaitForMultip] lea ecx,dword ptr ss: [esp+48] push ecx push 4</pre>	eObjects>] esi:Wa	x875tatusWord 0000 x875W_B 0 x875W_C3 0 x875W_C1 0 x875W_C0 0 x875W_SF 0 x875W_P 0	x87SW_C2 0 x87SW_ES 0 x87SW_U 0
EIP	→ 0040509C	FF D6	call esi	esitWa	Default (stdcall)	▼ 5 🗘 Unlock
esi= <kerne< th=""><th>132.WaitForMulti</th><th>01eObjects> (76ACDD50) 5509C #449C</th><th></th><th></th><th>1: [esp] 00000004 2: [esp+4] 0018EC40 3: [esp+8] 00000000 4: [esp+C] FFFFFFF</th><th></th></kerne<>	132.WaitForMulti	01eObjects> (76ACDD50) 5509C #449C			1: [esp] 00000004 2: [esp+4] 0018EC40 3: [esp+8] 00000000 4: [esp+C] FFFFFFF	
Dump 1	🛢 Dump 2 🐇	Dump 3 💧 Dump 4 🎯 Du	mp 5 🥈 Watch 1 💧 Locals 🦻 Struct	0018EBF0 000 0018EBF4 001	000004 18EC40	
Address H	ex		ASCII	A 0018EBF8 000	000000	
00185640 4	4 02 00 00 24 02	00 00 20 02 00 00 44 02	00 00 0 4 < 0	UUIDEBEC FFF	TTTTTTT	

Figure 74

Thread activity – sub_404190 (handling the standard input)

The thread obtains a handle to the standard input device by calling the GetStdHandle routine (0xFFFFFF6 = **STD_INPUT_HANDLE**):

● 0040419C 6A F6 push FFFFFF6 BIP →● 0040419E FF 15 9C 71 42 00 call dword ptr ds:[<&GetStdHandle>]	v	
<pre>dword ptr [0042719C <psexec.&getstdhandle>]=<kernel32.getstdhandle> .text:0040419E psexec.exe:\$419E #359E</kernel32.getstdhandle></psexec.&getstdhandle></pre>	>	Default (stdcal)
Figure 75	04C5FF2C FFFF	FFF6

PsExec checks whether the event object is in the signaled state via a function call to WaitForSingleObject:



Figure 76

The executable reads a character from the console input buffer using the ReadConsoleW function:

	 004041C0 004041C2 004041C6 004041C7 004041C9 004041C9 004041CE 	6A 00 8D 4C 24 14 51 6A 01 8D 54 24 24 52 53	push 0 lea ecx,dword ptr ss:[esp+1 push ecx push 1 lea edx,dword ptr ss:[esp+2 push edx push ebx	4 4		x875tatusWord 0000 x875W_B 0 x875W_C3 0 x875W_C1 0 x875W_C0 0 x875W_SF 0 x875W_P 0	x87SW_C2 0 x87SW_E5 0 x87SW_U 0
dword ptr [004041CF <	FF 15 50 71 42 00 .:.&ReadConsolew>]= <kernel< th=""><th>call dword ptr ds: [<&ReadCo 32.ReadConsolew></th><th>nsolew>]</th><th>></th><th>Default (stdcal) 1: [esp] 00000094 2: [esp+4] 04C5FF48 3: [esp+8] 0000001 4: [esp+C] 04C5FF40</th><th>▼ 5 🗘 🗆 Unlock</th></kernel<>	call dword ptr ds: [<&ReadCo 32.ReadConsolew>	nsolew>]	>	Default (stdcal) 1: [esp] 00000094 2: [esp+4] 04C5FF48 3: [esp+8] 0000001 4: [esp+C] 04C5FF40	▼ 5 🗘 🗆 Unlock
d Dump 1	🏮 Dump 2 🦂	Dump 3 🛛 🛭 Dump 4 🏻 🎯 Di	ump 5 🥈 Watch 1 🌒 Locals 🤌	Struct 040	SFF1C 00000	0094 FF48	
Address He 04C5FEE8 E0	EX FE C5 04 02 01	00 00 44 02 00 00 00 00	ASCII	^ 04C 04C 04C	5FF24 00000 5FF28 04C58 5FF2C 00000	0001 FF40 0000	

Figure 77

Our objective is to run the "whoami" command in the command prompt. As we can see below, the process encrypts the command byte-by-byte using the AES algorithm:

004 0	01700 52 01701 50 44 24 14 01705 50 01706 56 01707 6A 00 01709 6A 01 01708 6A 01 01708 6A 01 01700 51 01700 55 01701 00 01701 00 00 01701 00 00 01701 00 00 00 00 00 00 00 00 00 00 00 00 00	<pre>push edx lea eax_dword ptr ss:[esp+14] push eax push esi push 0 push 1 push 0 push 0 push ecx mov dword ptr ss:[esp+2C],edi call esp</pre>	ebp:Cr	x87Tw_6 3 (Empty) x87Tw_5 3 (Empty) x87Tw_6 3 (Empty) x87Tw_7 3 (Empty) x87Tw_6 3 (Empty) x87Tw_7 3 (Empty) x87Statusword 0000 x87Sw_C1 0 x87Sw_C2 0 x87Sw_C2 0 x87Sw_C1 0 x87Sw_C 0 x87Sw_C2 0 x87Sw_5 0 x87Sw_P 0 x87Sw_U 0 Default (stdcall)
.text:00401712 psex	ec.exe:\$1712 #B12			2: [csp+8] 00000000 3: [csp+8] 00000001 4: [csp+C] 00000000
🕼 Dump 1 🕴 Dump	2 🐇 Dump 3 🚯 Dump 4	🗑 Dump 5 🥈 Watch 1 🌒 Locals 🎾 Struct	04C5FEEC 0 04C5FEF0 0	2856F48 00000000
Address Hex		ASCII	^ 04C5FEF4 0	0000001
04311F70 77 00 18 0 04311F80 AB AB AB AB A 04311F90 3D 60 75 C	0 3C CC 47 CF <u>35 8E 40 00 35</u> B AB AB AB AB AB 00 00 00 00 00 0 60 36 00 00 <u>58 4C 31 04</u> CC	8E 40.00 W <th< th=""> <th< th=""> <!--</th--><th>04CSFEFC 0 04CSFF00 0 04CSFF04 0</th><th>14311570 14C5FF18 10000010</th></th<></th<>	04CSFEFC 0 04CSFF00 0 04CSFF04 0	14311570 14C5FF18 10000010
E:				

Figure 78

The length of the encrypted data and then the actual data from above are written to the "\\192.168.164.130\pipe\PSEXESVC-<Local computer name>-<Process ID>-stdin" pipe:

EIP	 00401733 00401735 00401739 0040173A 0040173A 00401740 00401741 00401741 	6A 00 8D 54 24 14 52 6A 04 8D 44 24 30 50 57 EF D3	push 0 lea edx,dword ptr ss:[esp+14] push edx push eax lea eax,dword ptr ss:[esp+30] push eax push eax		A0/18_0 5 (EmpLy) A0/18_7 5 (EmpLy) x8758tatusWord 0000 x875W_B 0 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0
		00		>	Default (stdcall) 🔹 5 🗘 🗌 Unlock
ebx= <kernel3< th=""><th>2.WriteFile> (76 42 psexec.exe:\$1</th><th>ACE250) 742 #842</th><th></th><th></th><th>11 [esp] 000002F4 22 [esp+4] 04C5FF2C 32 [esp+8] 0000004 44 [esp+C]_04C5FF18</th></kernel3<>	2.WriteFile> (76 42 psexec.exe:\$1	ACE250) 742 #842			11 [esp] 000002F4 22 [esp+4] 04C5FF2C 32 [esp+8] 0000004 44 [esp+C]_04C5FF18
Dump 1	🛢 Dump 2 🛛 🐇 D	ump 3 💧 Dump 4 🔮	Dump 5 🥈 Watch 1 🕚 Locals 🦻 Struct	04C5FEF4 0000 04C5FEF8 04C5	002F4 FF2C
Address Hex 04C5FEEC 48 04C5FEFC 04 04C5FFC 64 04C5FF1C 0E 04C5FF2C 10	6F 85 02 00 00 0 00 00 00 18 FF 0 02 00 00 70 DD 2 42 40 00 24 EC 3 00 00 00 35 8E 4	0 00 F4 02 00 00 2C F 5 04 00 00 00 00 60 60 F (7 75 94 00 00 00 10 00 8 00 F4 02 00 00 48 F 10 00 35 8E 40 00 78 F	ASCII <u>C 5 04</u> H0, <u>y</u> Å <u>18 00</u> <u>0 00 00</u> <u>ô</u> <u>p</u> Y~ <u>C 5 04</u> H0.5., <u>6</u> .5.0.xyÅ.	A 04CSFEFC 0000 04CSFF00 04CS 04CSFF04 0000 04CSFF08 0010 04CSFF00 0000 04CSFF07 0644 04CSFF10 7644 04CSFF10 7644	00004 FFI8 00000 JEC60 002F4 LDD70 kernel32.WaitForSingleObject
Figure	79				
	 0040174C 0040174E 00401752 00401753 00401754 00401755 	6A 00 8D 4C 24 14 51 52 56 57	push 0 lea ecx,dword ptr ss:[esp+14] push ecx push edx push esi push edi		x875tatusword 0000 x875W_B 0 x875W_C3 0 x875W_C2 0 x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0
EIP	→ 00401756	FF D3	call ebx	ebx:Wr v	Default (stdcall)
ebx= <kernel3< td=""><td>2.WriteFile> (76</td><td>ACE250) 1756 #856</td><td></td><td>,</td><td>1: [esp] 000002F4 2: [esp+4] 04311F70 3: [esp+4] 0400010 4: [esp+C] 04C5FF18</td></kernel3<>	2.WriteFile> (76	ACE250) 1756 #856		,	1: [esp] 000002F4 2: [esp+4] 04311F70 3: [esp+4] 0400010 4: [esp+C] 04C5FF18
A Dump 1	🛔 Dumo 2 🐇 D	umo 3 💧 Dumo 4 🏈	Dump 5 Watch 1 🙆 Locals 🖉 Struct	04C5FEF4 000	002F4
Address Hex 04311F70 C8	A4 C5 2A 44 57 C	03 AC 31 87 57 68 43 7	ASCII F 79 E7 EAADWO-1.WkC.yc	04C5FEF6 043 04C5FEFC 0000 04C5FF00 04C3 04C5FF04 0000	11F70 30010 5F18 30000
Figure	80				

The server end of the above pipe instance is disconnected from the process using DisconnectNamedPipe:

• 0040439B 56	Option 25 56 1 50 71 50 00 1 50 1 50 00 1 50 1 50 00 1 50 1 50 00 1 50 0 1 50 00 1 50 0				
COMOLESSC FF 15 58 71 42 00	[call dword ptr ds:[<&DisconnectNamedPipe>]	>	Default (stdcall)	▼ 5 🗘 🗌 Unlock	
dword ptr [00427158 <psexec.&disconnectnamedpip< td=""><td>>>]=<kernel32.disconnectnamedpipe></kernel32.disconnectnamedpipe></td><td></td><td>1: [esp+4] 00408235 psex 3: [esp+4] 00408235 psex 4: [esp+C] 043118E0</td><td>(ec.00408E35 (ec.00408E35</td></psexec.&disconnectnamedpip<>	>>]= <kernel32.disconnectnamedpipe></kernel32.disconnectnamedpipe>		1: [esp+4] 00408235 psex 3: [esp+4] 00408235 psex 4: [esp+C] 043118E0	(ec.00408E35 (ec.00408E35	
	a ⊻miri a r i (⊕n i)	04A2FF20 0000	00328		

Figure 81

Thread activity – sub_4043D0 (handling the standard error)

The thread reads 4 bytes from the "\\192.168.164.130\pipe\PSEXESVC-<Local computer name>-<Process ID>-stderr" pipe:

 00401622 6A 00 00401624 57 00401625 6A 04 00401625 60 04 00401627 8D 44 24 18 00401628 50 00401628 53 	push o push edi lea eax,dword ptr ss: [esp+10] push eax push ebx	x875tatusword 0000 x875w_B 0 x875w_C3 0 x875w_C2 0 x875w_C1 0 x875w_C0 0 x875w_E5 0 x875w_57 0 x875w_P 0 x875w_U 0
ebp= <kernel32.readfile> (76ACE160) .text:0040162D psexec.exe:\$162D #A2D</kernel32.readfile>	call eop	
🕼 Dump 1 👶 Dump 2 🐇 Dump 3 🕘 Dump 4 🎡 Dump 5	Watch 1 ● Locals O483 O483	FEES 0000032C FEEC 0483FF08
Address Hex 0483FF08 24 EC 18 00 3D 44 40 00 24 EC 18 00 2C 03 00 00	ASCII 0483 \$1D@.\$1	FEF0 00000004 FEF4 0483FF38 FEF8 00000000

Figure 82

The server end of the above pipe instance is disconnected from the process using DisconnectNamedPipe:

EIP → 00404485 00404486	57 FF 15 58 71 42 00	push ed1 call dword ptr ds:[<4DisconnectNamedPipe>		Default (stdcall)
dword ptr [00427158 <psexec.& .text:00404486 psexec.exe:\$44</psexec.& 	DisconnectNamedPipe>]= 86 #3886	<kernel32.disconnectnamedpipe></kernel32.disconnectnamedpipe>		1: [esp] 0000032C 2: [esp+4] 00408E35 psexec.00408E35 3: [esp+8] 00408E35 psexec.00408E35 4: [esp+C] 0485FF78
A	A	5 Y	0483FF24 000	0032C

Figure 83

Thread activity – sub_404240 (handling the standard output)

The thread reads 4 bytes from the "\\192.168.164.130\pipe\PSEXESVC-<Local computer name>-<Process ID>-stdout" pipe:

00402622 6A 00 00402624 57 00402625 6A 04 00402625 6A 04 00402627 8B 44 24 18 00402622 53	push 0 push edi push 4 lea eax,dword ptr ss:[esp+18] push eax push ebx	x875tatusword 0 x875w_B 0 x87 x875w_C1 0 x87 x875w_S7 0 x87 x875w_S7 0 x87	0000 Sw_C3 0 x87Sw_C2 0 Sw_C0 0 x87Sw_E5 0 Sw_P 0 x87Sw_U 0
00401620 FF D5 ebp= <kernel32.readfile> (764CE160) .text:0040162D psexec.exe:\$162D #A2D</kernel32.readfile>	call ebp	ebp:Re v Default (stdcal) 1: [esp] 000003 2: [esp+8] 0000 4: [esp+8] 0000 4: [esp+8] 0000	▼ 5 💭 Unlock FF04 0004 FF30
🕼 Dump 1 👶 Dump 2 🐇 Dump 3 🚷 Dump 4	🍘 Dump 5 🥈 Watch 1 🌒 Locals 🦻 Struct	04A2FEE4 00000328 04A2FEE8 04A2FF04	
Address Hex 04A2FF04 30 FF A2 04 A8 42 40 00 24 EC 18 0	ASCII	04A2FEEC 00000004 04A2FEF0 04A2FF30 04A2FEF4 00000000	

Figure 84

The ReadFile API is utilized to read encrypted data from the above pipe:

0040164A 6A 00 0040164C 57 0040164C 57 0040164D 52 0040164E 88 F0 00401651 563 00401651 563 00401651 FF D5 00501552 FF D5	push 0 push edi push edi mov esi eax push esi push ebx call ebp	ebp:Re 🗸	x87StatusWord 0000 x87SW_B 0 x87SW_C3 0 x87SW_C2 0 x87SW_C1 0 x87SW_C0 0 x87SW_ES 0 x87SW_SF 0 x87SW_P 0 x87SW_U 0 Default (stdcall) ▼ 5 0 Unloc
ebp= <kernel32.readfile> (76ACE160) .text:00401652 psexec.exe:\$1652 #A52</kernel32.readfile>			1: [esp] 00000328 2: [esp+4] 04311F70 3: [esp+8] 00000030 4: [esp+C]_04A2FF30
🕼 Dump 1 🕴 Dump 2 🐇 Dump 3 🌒 Dump 4	🍘 Dump 5 🥈 Watch 1 🕘 Locals 🤌 Struct	04A2FEE0 00000 04A2FEE4 04311	328 F70
Address Hex 01311570 00 F0 40 84 00 F0 40 84 00 F0 49 84 Figure 85	ASCII 00 F0 AD BA 80.0.0.0.0.0.0.0	04A2FEE8 00000 04A2FEEC 04A2F 04A2FEF0 00000	030 F30 000

The buffer is decrypted using the AES algorithm via a call to CryptDecrypt:



MultiByteToWideChar is used to map character strings to UTF-16 (wide character) strings:

	94 24 20 00 01 00 94 24 20 00 01 00 FF 44 24 24 55 0C 30	push 10001 lea edx,dword ptr ss:[esp+1: push edx push edx push ex push ex	0020] 4]],61	eax:"M	x87TW_4 3 (Empty) x87TW_5 3 (Empty) x87TW_6 3 (Empty) x87TW_7 3 (Empty) x87StatusWord 0000 x87SW_6 3 0 x87SW_C2 0 x87SW_8 0 x87SW_C3 0 x87SW_C2 0 x87SW_C2 0 x87SW_5 0 x87SW_C0 0 x87SW_L2 0 x87SW_10 0
	D5	call ebp		ebp:Mu v	Default (stdcall) 🗾 🗸 Unlock
ebp= <kernel32.multibytetowidech .text:004042DE psexec.exe:\$42DE</kernel32.multibytetowidech 	ar> (76A75B40) #36DE				1: [esp] 00000000 2: [esp+4] 0000000 3: [esp+8] 04A2FF38 "Microsoft Windows [Version 4: [esp+6] FFFFFFF
🕼 Dump 1 🕴 Dump 2 🐇 Dump	3 🚯 Dump 4 🞯 Dump !	i 😽 Watch 1 🚯 Locals 🤌	Struct	04A2FF08 0000 04A2FF0C 0000	0000
Address Hex		ASCII	^	04A2FF10 04A2	FF38 "Microsoft Windows [Version 6.1.7600]"
04A3FF3C 00 00 00 00 00 00 00 00 00 00 00 00 00	0 00 00 00 00 00 00 00 00 0 00 00 00 00	00		04A2FF18 04A3 04A2FF1C 0001	FF3C 0001

Figure 88

The process retrieves a handle to the standard output device using GetStdHandle (0xFFFFFF5 = STD_OUTPUT_HANDLE):

• 00404	66E 6A F5	push FFFFFF5			
BLC → 004043	70 FF 15 9C 71 42 00	<pre>call dword ptr ds:[<&GetStdHandle>]</pre>	>	Default (stdcall)	
dword ptr [0042719C <p .text:00404370 psexec.</p 	sexec.&GetStdHandle>]= <kerne exe:\$4370 #3770</kerne 	132.GetStdHandle>		1: [esp] FFFFFF5 2: [esp+4] 042FF38 "Microsoft Windows [Version 3: [esp+8] 0000024 4: [esp+C] 0442FF30	
Acres 1 Acres	4 A	• Ymm A r i @m i	04A2FF0C FEF	FFFFS	

Figure 89

The buffer that was decrypted above is written to the standard output via a call to WriteFile:

00404370 FF 15 9C 71 42 00 [Call dword ptr ds:[<ssjctstdhandle>] 00404376 50 push eax call dword ptr ds:[<ssjctstdhandle>] x875W_SF 0 x875W_P 0 x875W_U 0 ushes call dword ptr ds:[<ssjctstdhandle>]</ssjctstdhandle></ssjctstdhandle></ssjctstdhandle>	1 Unloc
00404350 6 00404350 9 0040435 9 00404 9 0040435 9 0040435 9 0040435 9 00404	Unloc

Figure 90

Figure 91 reveals that we get a shell on the remote machine using the above method:





We continue with the analysis of the main thread.

The PsExec process sets the event object to the signaled state using SetEvent:

004050A4 004050A4 004050A5	52 FF 15 5C 71 42 00	push edx		
• <			>	Default (stdcall)
dword ptr [0042715C <psexed< td=""><td>.&SetEvent>]=<kernel32.< td=""><td>SetEvent></td><td></td><td>1: [esp+4] 00000008 3: [esp+8] 00432138 L"192.168.164.130" 4: [esp+C] 76A75080 <kernel32.getprocaddress></kernel32.getprocaddress></td></kernel32.<></td></psexed<>	.&SetEvent>]= <kernel32.< td=""><td>SetEvent></td><td></td><td>1: [esp+4] 00000008 3: [esp+8] 00432138 L"192.168.164.130" 4: [esp+C] 76A75080 <kernel32.getprocaddress></kernel32.getprocaddress></td></kernel32.<>	SetEvent>		1: [esp+4] 00000008 3: [esp+8] 00432138 L"192.168.164.130" 4: [esp+C] 76A75080 <kernel32.getprocaddress></kernel32.getprocaddress>
		e Vanata Arata Genet	0018EBFC 000	00244

Figure 92

There is a second call to WaitForMultipleObjects that suspends the process until two of the above threads finish:

	 004050AB 004050AD 004050AF 004050B3 004050B4 	6A FF 6A 01 8D 44 24 4C 50 6A 02	push fFFFFFF push 1 lea eax,dword ptr ss:[esp+4C] push eax push 2		x8/5tatusword 0000 x875w_B 0 x875w_C3 0 x875w_C x875w_C1 0 x875w_C0 0 x875w_E x875w_5F 0 x875w_P 0 x875w_U	2 0
EIR	◆ 00405086	FF D6	Call esi	est:wa v	Default (stdcall)	▼ 5 🕏 🗌 Unlock
esi= <kernel< th=""><th>32.WaitForMulti</th><th>pleObjects> (76ACDD50) \$5086 #4486</th><th></th><th></th><th>1: [esp14] 00186C44 3: [esp+8] 0000001 4: [esp+C] FFFFFFF</th><th></th></kernel<>	32.WaitForMulti	pleObjects> (76ACDD50) \$5086 #4486			1: [esp14] 00186C44 3: [esp+8] 0000001 4: [esp+C] FFFFFFF	
Dump 1	🛢 Dump 2 🐇	Dump 3 💧 Dump 4 🧯	Dump 5 🛛 Watch 1 🔮 Locals 🎾 Struct	0018EBF0 000 0018EBF4 001	00002 8EC44	
Address He	x 03 00 00 3C 03	00 00 44 03 00 00 02	ASCII	^ 0018EBF8 000 0018EBFC FFF	00001 FFFFF	

Figure 93

The process reads 4 bytes from the "\\192.168.164.130\pipe\PSEXESVC" pipe:

 00401622 6A 00 00401624 67 00401625 6A 04 00401627 6A 424 00401627 80 44 24 00401628 50 00401626 53 	push 0 push 4 lea eax,dword ptr ss:[esp+18] push eax push ebx	[esp+1	x875tatusWord 0000 x875W_B 0 x875W_C3 0 x875W_C2 0 x875W_C10 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0
30 00401620 FF D5	call ebp	ebp:Re v	Default (stdrall)
		>	1: [esp] 0000200
ebp= <kernel32.readfile> (76ACE160) .text:0040162D psexec.exe:\$162D #A2D</kernel32.readfile>			2: [esp+4] 0018EBE4 3: [esp+8] 0000004 4: [esp+C] 0018EC18
🕼 Dump 1 🕴 Dump 2 🐇 Dump 3 🚯 Dump 4 🎯 Dump 5	🕉 Watch 1 🌒 Locals 🤌 Struct 000	18EBC4 0000 18EBC8 0018	0300 E8E4
Address Hex	ASCII	18EBCC 0000	0004
0018EBE4 24 EC 18 00 F5 50 40 00 24 EC 18 00 00 03 00 00	\$10P@.\$1	18EBD4 0000	0000

Figure 94

In the case of successful command execution, the remote PSEXESVC.exe process sends the result through the above pipe (in this case, the username).

In the case of an error, the binary retrieves the thread's last-error code value using the GetLastError API:

BIP → 0040511E FF D6	call esi	est:Ge v	Default (stdcail)
es1= <kernel32.getlasterror> (76A74F10) .text:0040511E psexec.exe:\$511E #451E</kernel32.getlasterror>			1: [csp+4] 0043218 L"192.168.164.130" 3: [csp+4] 0043218 L"192.168.164.130" 4: [csp+8] 76A75080 <kernel32.getprocaddress> 4: [csp+C] 76A759E0 <kernel32.loadlibraryw></kernel32.loadlibraryw></kernel32.getprocaddress>

Figure 95

The error message is formatted by calling the FormatMessageA routine (0x1300 =

FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS | FORMAT_MESSAGE_ALLOCATE_BUFFER, 0x3B = ERROR_UNEXP_NET_ERR, 0x400 = LANG_USER_DEFAULT):

Contraction Contract					push 0 push 0 lea ecx,dwor push ecx push 400 push ebx push esi push edi	d ptr ss:	esp+	10]	[esp+1)			x87TW_6 3 (Empty) x87TW_7 3 (x87TW_6 3 (Empty) x87TW_7 3 (x875K_1USWOrd 0000 x87SW_8 0 x87SW_C3 0 x87SW, x87SW_5 0 x87SW_C 0 x87SW, x87SW_5 0 x87SW_P 0 x87SW,					pty)			
dword p	tr [0042	<pre>7198 <psexe psexec.exe:</psexe </pre>	52C56 #	atMessag	eA>]=<	kernel32.F	FormatMessag	ieA>	er or ma	umessage	-	_	> C	lefault (std L: [esp] 2: [esp+ 3: [esp+ 4: [esp+	cal) 00001 4] 000 8] 000 C] 000	300 00000 00038 00400			▼ 5	😫 🗌 Unlod
d Dum	p1 8	Dump 2 🐇	Dump 3	🚯 Durr	p4	Dump 5	😽 Watch 1	Locals	2	Struct	4	0018EBCC 0018EBD0	00001	300						
Address 0018EBF 0018EC0 0018EC1	Hex C 3B 00 C 59 C 00 00 C 10 00	00 00 08 00 A7 76 00 00 00 00 00 00 00	00 00 00 00 00 00	38 21 43 18 7D 89 48 6F 85	00 B0 02 00 02 10	50 A7 76 00 00 00 00 00 00	ASCII	. 'P§v			^	0018EBD4 0018EBD8 0018EBDC 0018EBE0 0018EBE4	00000 00000 0018E 00000 00000	038 400 BFC 000 000						

The error message is written to the standard output:

EIP	 00402C60 00402C62 00402C66 00402C67 00402C68 00402C68 00402C66 00402C66 00402C6F 00402C75 	6A 00 8D 54 24 10 50 8B 44 24 20 6A F4 50 6A F4 50 50 7F 15 9C 71 42 00 FF 15 9C 71 42 00	<pre>push 0 lea edx_dword ptr ss:[esp+10] push edx push edx mov eax_dword ptr ss:[esp+20] push eax push FFFFFF4 call dword ptr ds:[<&GetStdHandle>] push eax call dword otr ds:[<</pre>	edx:",	x87TW_45 (Empty) x87TW_53 (Empty) x87TW_45 3 (Empty) x87TW_73 (Empty) x87TW_50 3 (Empty) x87TW_73 (Empty) x87SW_50 x87SW_50 x87SW_52 0 x87SW_52 0 x87SW_50 x87SW_50 0 x87SW_50 0 x87SW_50 x87SW_50 0 x87SW_50 0
	• <	11 12 11 11 11 11 11			Default (stdcall)
dword ptr	[004271D4 <psexe)2C76 psexec.exe:</psexe 	<pre>\$20.00 #20.</pre>	WriteFile>		2: [esp+4] 0284FCF8 "An unexpected network error 3: [esp+8] 0000027 4: [esp+C] 0018EBF4 ",u@"
Dump 1	🖲 Dumo 2 🐇	Dump 3 🚯 Dump 4 🎯 Du	mp 5 🥈 Watch 1 🌒 Locals 🎘 Struct	0016EED4 000	0009C
Address H	Hex		ASCII	^ 0018EBDC 000 0018EBDC 000	8E8F4 ",u®"
Figure	97			0018EBE4 000	00000
1\192.168	8.164.130: cmd.exe				- 🗆 ×
PsExec v Copyrigh Sysinter	/2.11 - Exect nt (C) 2001-2 rnals - www.s	ute processes remot 2014 Mark Russinovi sysinternals.com	ely ich		^
Microsof Copyrigh	ft Windows [nt (c) 2009 M	Version 6.1.7600] Microsoft Corporati	ion. All rights reserved.		
C:\Windo	ows\system32	>whoami			

Figure 98

The WNetCancelConnection2W API is utilized to cancel the existing network connection:

	 00402DD7 00402DD9 00402DD8 00402DDF 	6A 01 6A 00 8D 4C 24 08 51	push 1 push 0 lea ecx,dword ptr st push ecx	:[esp+8]	ecx:L"	x87SW_B 0 x87SW x87SW_C1 0 x87SW x87SW_SF 0 x87SW	_C3 0 x875W_C2 0 _C0 0 x875W_E5 0 _P 0 x875W_U 0
<pre>cpsexec.wNe</pre>	<pre> 004020E0 </pre> <pre> ConcelConnectio </pre>	E8 BF 39 00 00	call <psexec.wnetcar< th=""><th>icelConnection2W></th><th>></th><th>Default (stdcall) 1: [esp] 0018E3A4</th><th>▼ 5 ↓ Unlock L"\\\192.168.164.130\\IPC\$"</th></psexec.wnetcar<>	icelConnection2W>	>	Default (stdcall) 1: [esp] 0018E3A4	▼ 5 ↓ Unlock L"\\\192.168.164.130\\IPC\$"
.text:00402	DE0 psexec.exe:\$	2DE0 #21E0				3: [esp+8] 0000000 4: [esp+C] 005C005	1 C
Dump 1	🛢 Dump 2 🐇 I	Dump 3 💧 Dump 4 🎯 D	ump 5 🕉 Watch 1 🔕 Loca	als 🐉 Struct	0018E39S 001 0018E39C 000 . 0018E3A0 000	L8E3A4 L"\\\192.168 000000 000001	.164.130\\IPC\$"

Figure 99

Second process: psexec.exe -c -f -s win.exe

We'll only highlight the differences between running PsExec on the local machine and the first case.

The process retrieves the content of the %PATH% environment variable by calling the GetEnvironmentVariableW function:

	 004060 004060 004060 004060 004060 	6C 56 6D 88 F8 6F 57 70 68 C8	BC 42 00		oush esi nov edi,eax oush edi oush psexec.	42BCC8			428CC8	x87 x87 x87	SW_B 0 SW_C1 0 SW_SF 0	x87SW_C3 x87SW_C0 x87SW_P	000	x875W_C2 x875W_E5 x875W_U	000	
EIR	> <u>004060</u> <	75 FF DS	er 13.00		all ebp				ebp:Ge v	Defau	lt (stdcall)				- 5	😫 🗌 Unlock
ebp= <kerne< th=""><th>132.GetEnviro</th><th>exe:\$6075 #54</th><th>ew> (76A7-</th><th>4EBO)</th><th></th><th></th><th></th><th></th><th></th><th>2: 2: 4: 4: 5: 0</th><th>esp1 00 esp+4] esp+8] esp+C] esp+10]</th><th>428CC8 L"P) 042A1818 0000034F 00000005 00197F3A 1</th><th>L"C:\</th><th>\\Users\\</th><th></th><th>\Desktop\\</th></kerne<>	132.GetEnviro	exe:\$6075 #54	ew> (76A7-	4EBO)						2: 2: 4: 4: 5: 0	esp1 00 esp+4] esp+8] esp+C] esp+10]	428CC8 L"P) 042A1818 0000034F 00000005 00197F3A 1	L"C:\	\\Users\\		\Desktop\\
Dump 1	Dump 2	🐇 Dump 3	🕘 Dump 4	🔮 Dump 5	👸 Watch 1	Locals	Struct	00	0197C3C 004 0197C40 04	42BCC8 2A1818	L"PATH	to 042A181	ls fr	om 11441	88F	

Figure 100

GetFileAttributesW is used to obtain file system attributes for the specified file:

00404025	56	push esi	esi:L"		300 0 0 00300 3	0 1183.001 B	0
< 00404026	FF 15 54 71 42 00	Can aword per ds: [<@GetFileAttributesw>]	>	Det	ault (stdcall)	•	5 🗘 🗌 Unloci
dword ptr [00427154 <psexec.&getfileattributesw>]=<kernel32.getfileattributesw> .text:00404026 psexec.exe:\$4026 #3426</kernel32.getfileattributesw></psexec.&getfileattributesw>					[esp+4] 0000034F [esp+4] 0000034F [esp+8] 0040608B [esp+C] 00197CF8 [esp+10] 042A1818	psexec.00406088 L"win.exe"	
An	A	_ I Maari I Ar i I Bi⊥ i I	00197C2C 00	01970	8 L"win.exe"		

An intermediary message that gives details about what action will occur next is displayed:

C:\Users\\\Desktop\psexec.exe	-	×
PsExec v2.11 - Execute processes remotely Copyright (C) 2001-2014 Mark Russinovich Sysinternals - www.sysinternals.com		^
Connecting to local system		

Figure 102

The binary initializes the use of the Winsock DLL using the WSAStartup routine:



Figure 103

The gethostname function is utilized to extract the standard host name for the local machine:

	● 00403367 ● 0040336C ● 00403373	68 04 01 00 00 8D 8C 24 D4 03 00 00 51	push 104 lea ecx,dword ptr ss:[esp+304] push ecx		x875W_C1 0 x875W_C0 x875W_5F 0 x875W_P	0 x875W_ES 0 0 x875W_U 0
EIP	00403374	E8 1F 34 00 00	call <pre>sexec.gethostname></pre>	, v	Default (stdcall)	🔻 💈 🗘 Unlod
<pre><psexectext:00< pre=""></psexectext:00<></pre>	gethostname>	\$3374 #2774			1: [esp] 0018E6AC 2: [esp+4] 00000104 3: [esp+8] 76ACDC00 <k 4: [esp+C] 00000000 5: [esp+10] 00432138 L</k 	ernel32.CreateEventW> "DESKTOP-
& Dump	1 🕴 Dump 2 🐇	Dump 3 🛯 🚷 Dump 4 🛛 🎯 Dum	p 5 🥈 Watch 1 🜒 Locals 🤌 Struct	0018E2D4 0018 0018E2D8 0000	8E6AC 00104	

Figure 104

PsExec retrieves host information corresponding to the local host:

• 00403380	52	push edx	edx: "D			0	
	E8 0C 34 00 00	call «psexec.getnostoyname»	,	Default	(stdcall)		▼ 5 ≑ 🗌 Unlock
<pre><psexec.gethostbyname> .text:00403381 psexec.exe:</psexec.gethostbyname></pre>	\$3381 #2781			2: [e 3: [e 4: [e 5: [e	spj 0018E6AC De sp+4] 76ACDC00 sp+8] 00000000 sp+C] 00432138 t sp+10] 76A759E0	kkernel32.Cre kkernel32.cre kkernel32.Lc	eateEventW> " padLibraryW>
An		[M A [@]	0018E2D8 003	LSEGAC	DESKTOP-	**	

Figure 105

The local IP address in hex is converted into an ASCII string in dotted-decimal format:

• 00403381 50	push eax			A
(2) → 00103332 E8 D5 33 00 00	call cpsexec.inet_ntoas		Default (stdcall)	👻 5 🗘 Unlod
<pre><psexec.inet_ntoa> .text:00403382 psexec.exe:\$3382 #2782</psexec.inet_ntoa></pre>			<pre>2: [esp+4] FFFFFFF 3: [esp+4] FFFFFFF 3: [esp+8] 0018E4A4 4: [esp+C] 0000104 5: [esp+10] 76ACDC00</pre>	<kernel32.createeventw></kernel32.createeventw>
An	(A	0018E2CC 80A	4480	

Figure 106

The executable extracts the path of the System directory via a function call to GetSystemDirectoryW:

00403085 0040308A 0040308A 00403091	68 04 01 00 00 8D 94 24 28 02 00 00 52	push 104 lea edx,dword ptr ss:[esp+228] push edx	[esp+2 edx:&"	x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0
CO403092	FF 15 78 71 42 00	<pre>call dword ptr ds:[<&GetSystemDirectoryw>]</pre>	>	Default (stdcall)
dword ptr [00427178 <psexed .text:00403092 psexec.exe:1</psexed 	c.&GetSystemDirectoryW>]= <k \$3092 #2492</k 	ernel32.GetSystemDirectoryW>		1: [csp] 002100 a Destror- 2: [csp+4] 0000104 3: [csp+8] 00430EF0 L"PSEXESVC" 4: [csp+C] 00432138 L"DESKTOP- 5: [csp+10] 00430EF0 L"PSEXESVC"
🕼 Dump 1 🕴 Dump 2 🐇	Dump 3 🛛 🕘 Dump 4 🛛 🎯 Dump	9 5 🥳 Watch 1 🕘 Locals 🖉 Struct	0018DE74 0013 0018DE78 0000	8E0A0 & DESKTOP-
Figure 107				

The same workflow of extracting the PSEXESVC resource as in the first case is repeated. However, this time the parameter is C:\Windows\PSEXESVC.exe, which is created and populated using _wfopen and _fwrite:

210	00402050 68 7C 98 42 00 push psexec.42987C push ecx 00402056 88 F8 mov edi, eax		429B7C	42987C ecx:L"	x875W_C1 0 x875W_SF 0	2		
	< C C C C C C C C C C C C C C C C C C C	an psece.	+0/ 013	ropen >	Default (stdcall)	▼ 5	Unlock
psexec.00	0407B13 02D68 psexec.exe:\$2D68 #2168				1: [esp] 0 2: [esp+4] 3: [esp+8] 4: [esp+C] 5: [esp+10	0042987C L"wb" 001938DC L"PSEXI 0042A994 L"PSEXI 0042A994 L"PSEXI] 00433140 psexed	ESVC.exe" ESVC" c.00433140	ESVC, exe
d Dump 1	🕴 Dump 2 🐇 Dump 3 🚷 D	ump 4 🎯 Dump 5 🥈 Watch 1	🚯 Locals 🛛 🎾 Struct	0018DE5C 00 0018DE60 00	18DE98 L"C:\\ 429B7C L"wb"	WINDOWS//PSEXES	/C.exe"	

Figure 108

The process obtains a handle to the standard output device using GetStdHandle (0xFFFFFF5 = **STD_OUTPUT_HANDLE**):

00402CAA 00402CAA	6A F5	push FFFFFFF5		
C C		carri divi o pri us. ["aderscunario resj	>	Default (stdcall)
dword ptr [0042719C <psexec.&c< td=""><td>GetStdHandle>]=<kernel< td=""><td>32.GetStdHandle></td><td></td><td>1: [csp] FFFFFFS 2: [csp+4] 00403505 psexec.00403505 3: [csp+8] 76ACDC00 <kernel32.createeventw></kernel32.createeventw></td></kernel<></td></psexec.&c<>	GetStdHandle>]= <kernel< td=""><td>32.GetStdHandle></td><td></td><td>1: [csp] FFFFFFS 2: [csp+4] 00403505 psexec.00403505 3: [csp+8] 76ACDC00 <kernel32.createeventw></kernel32.createeventw></td></kernel<>	32.GetStdHandle>		1: [csp] FFFFFFS 2: [csp+4] 00403505 psexec.00403505 3: [csp+8] 76ACDC00 <kernel32.createeventw></kernel32.createeventw>
.text:00402CAC psexec.exe:\$2C/	AC #20AC			5: [esp+10] 00432138 L"DESKTOP-
Acres 40 . 40	a 🗛 . 🖑 a	→ Murris Aris Ømis	0018E2D4 FFF	FFFFS

Figure 109

GetConsoleScreenBufferInfo is used to retrieve information about the console screen buffer:



Figure 110

The next step of the process is to start the PSEXESVC service on the local machine, as highlighted below:



Figure 111

The OpenSCManagerW API is utilized to establish a connection to the service control manager on the local computer (0xF003F = **SC_MANAGER_ALL_ACCESS**):

00402635 00402635 00402636 00402636	68 3F 00 0F 00 6A 00 50	push F003F push 0 push eax	eax:L"	x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0
	FF 15 24 70 42 00	call dword ptr ds:[<&OpenSCManager	>] ····································	Default (stdcall)
dword ptr [00427024 <pse .text:00402B3D psexec.ex</pse 	xec.&OpenSCManagerW>]= <adv e:\$2B3D #1F3D</adv 	2: [esp+4] 0000000 3: [esp+8] 000F003F 4: [esp+C] 00430EF0 L"PSEXESVC" 5: [esp+10] 04403553 psexec.00403553		
Dump 1 Dump 2	🐇 Dump 3 🛛 🚷 Dump 4 🛛 🎯 D	ump 5 🥈 Watch 1 🜒 Locals 🦻 Struct	0018E298 0043 0018E29C 0000 0018E2A0 0000	82138 L"DESKTOP-

Figure 112

A new service called "PSEXESVC" is created on the local host (0xF01FF

- = SERVICE_ALL_ACCESS, 0x10 = SERVICE_WIN32_OWN_PROCESS, 0x3
- = SERVICE_DEMAND_START):

	79 6A 00 78 6A 00 70 6A 00 71 6A 00 81 58 4C 24 1C 88 6A 00 84 58 4C 24 1C 88 6A 03 80 88 44 24 30 91 68 FF 01 0F 00 95 50 95 FF 15 34 70 42 00 EFF 15 34 70 42 00 Sexec.&CreateServicew]= <adva< th=""><th><pre>push 0 push 0 push 0 push 0 push 0 push ex push ex push ex, dword ptr ss:[esp+1C] push 3 push eax mov eax, dword ptr ss:[esp+30] push FOIFF push edx push eax pu</pre></th><th>[esp+1] x87r6 000000000000000000 ST6 Empty 0.00000 x87r7 00000000000000000 ST7 Empty 0.00000 x87rMc 0 3 (Empty) x87rW_1 3 (Empty) x87rWc 0 3 (Empty) x87rW_3 1 (Empty) x87rWc 0 3 (Empty) x87rWc 3 (Empty) x87rWc 3 (Empty) x87rWc 3 (Empty) x87rWc 3 (Empty) x87rWc 3 (Empty) x87sWc 10 x87sWc 20 0 x87sWc 20 x87sWc 10 x87sWc 20 0 x87sWc 50 x87sWc 10 x87sWc 20 0 x87sWc 50 x87sWc 10 x87sWc 20 0 x87sWc 50 x87sWc 10 x87sWc 10 x87sWc 10 x87sWc 10 x87sWc 10 x87sWc 10 x87sWc 10</th></adva<>	<pre>push 0 push 0 push 0 push 0 push 0 push ex push ex push ex, dword ptr ss:[esp+1C] push 3 push eax mov eax, dword ptr ss:[esp+30] push FOIFF push edx push eax pu</pre>	[esp+1] x87r6 000000000000000000 ST6 Empty 0.00000 x87r7 00000000000000000 ST7 Empty 0.00000 x87rMc 0 3 (Empty) x87rW_1 3 (Empty) x87rWc 0 3 (Empty) x87rW_3 1 (Empty) x87rWc 0 3 (Empty) x87rWc 3 (Empty) x87rWc 3 (Empty) x87rWc 3 (Empty) x87rWc 3 (Empty) x87rWc 3 (Empty) x87sWc 10 x87sWc 20 0 x87sWc 20 x87sWc 10 x87sWc 20 0 x87sWc 50 x87sWc 10 x87sWc 20 0 x87sWc 50 x87sWc 10 x87sWc 20 0 x87sWc 50 x87sWc 10 x87sWc 10 x87sWc 10 x87sWc 10 x87sWc 10 x87sWc 10 x87sWc 10
.text:00402799 psexec.	exe: \$2799 #1899	mp 5 🥈 Watch 1 🔮 Locals 🎾 Struct	4: [ssp+c] 000F01FF 5: [csp+10] 0000010 001622CI 0018606 00182250 0018606 return to psyce_00430E50 from 222
Address Hex 00430EF0 50 00	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	ASCIT ASCIT ASCIT ASCIT ASC. AS	

The PsExec process starts the new service by calling the StartServiceW API:

00402855 6A 00 push 0 00402857 6A 00 push 0 00402857 6A 00 push 0 00402859 50 push eax								x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0			
dword ptr [0040285 < (0042702C <ps< th=""><th>Exec.&Start</th><th>5 2C 70 42 0</th><th>advapi32.St</th><th>all dword pt</th><th>tr ds:[<&S</th><th>artServicew>]</th><th></th><th>></th><th>Default (stdcall) 1: [esp] 0278F390 2: [esp+4] 00000000 2: [esp+4] 00000000</th><th>▼ 5 🗘 🗌 Unloc</th></ps<>	Exec.&Start	5 2C 70 42 0	advapi32.St	all dword pt	tr ds:[<&S	artServicew>]		>	Default (stdcall) 1: [esp] 0278F390 2: [esp+4] 00000000 2: [esp+4] 00000000	▼ 5 🗘 🗌 Unloc
.text:0040285A psexec.exe:\$285A #1C5A									4: [esp+C] 027BB608 5: [esp+10] 00432138 L"DESK	FOP-2C3IQHO"	
Dump 1	Dump 2	🐇 Dump 3	Dump 4	谢 Dump 5	Watch 1	Locals	2 Struct	0018E254 0018E258	0278	BF 3 90 00000	

Figure 114

The QueryServiceStatus routine is used to obtain the current status of the above service:

0040288 0040288 0040288 0040288	5 52 5 50 7 BF 01 00 00 00	push edx push eax mov edi,1		x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0				
EIP 0040283	FF D6	call esi	est:Qu v	Default (stdcall)				
es1= <advap132.queryserv .text:0040288C psexec.e</advap132.queryserv 	ceStatus> (73A62F10) <c:\$288c #1c8c<="" th=""><th></th><th></th><th>1: [esp1] 0278F390 2: [esp+4] 0018E270 3: [esp+8] 00430EF0 L"PSEXESVC" 4: [esp+0] 0278B608 5: [esp+10] 00432138 L"DESKTOP-</th></c:\$288c>			1: [esp1] 0278F390 2: [esp+4] 0018E270 3: [esp+8] 00430EF0 L"PSEXESVC" 4: [esp+0] 0278B608 5: [esp+10] 00432138 L"DESKTOP-				
Dump 1 Dump 2	🐇 Dump 3 🛛 🕲 Dump 4 🛛 🧐	Dump 5 🥈 Watch 1 🜒 Locals 🦻 Struct	0018E254 027 0018E258 001	8F390 8E270				

Figure 115

The process prints the next step in the command line prompt:



Figure 116

The binary opens the "\pipe\PSEXESVC" pipe from the local machine via a function call to CreateFileW (0xC0000000 = **GENERIC_READ** | **GENERIC_WRITE**, 0x3

= OPEN_EXISTING):

	push 0 push 3 push 3 push 0 push CO000000 lea edx,dword ptr ss:[esp+4AEC] push edx call est at a statement of the statement o	edx:L" esi:Cr ✓	X8/TM_4 3 LEMPLYJ X8/TM_5 3 LEMPLYJ X87TW_6 3 (Empty) X87TW_7 3 (Empty) X87TW_6 3 (Empty) X87TW_7 3 (Empty) X87SW_5 0 X87SW_C3 0 X87SW_C2 0 X87SW_5 0 X87SW_6 0 X87SW_C0 0 X87SW_5 0 X87SW_5 0 X87SW_5 0 X87SW_P 0 X87SW_U 0 X87SW_5 0 X87SW_5 0 X87SW_P 0 X87SW_U 0 X87SW_5 0
esi= <kernel32.createfilew> (76ACDDE0)</kernel32.createfilew>		>	1: [esp] 001936D4 L"\\\DESKTOP- 2: [esp+4] C0000000 3: [esp+8] 00000000
.text:00404AB7 psexec.exe:\$4AB7 #3EB7			4: [esp+C] 00000000 5: [esp+10] 00000003
🕼 Dump 1 🕴 Dump 2 🐇 Dump 3 🔮 Dump 4 🔮 Dump 5	5 🥈 Watch 1 🜒 Locals 🤌 Struct	0018EBE4 0019 0018EBE8 C000	00000 L"\\\\DESKTOP-
Address Hex	ASCII	0018EBEC 0000	00000
00193604 [SC 00 5C 00] 44 00 45 00 53 00 48 00 54 00 4F 001936F4 50 00 20 00 001936F4 50 00 70 00 69 00 70 00 65 00 5C 00 50	00 N.\.D.E.S.K.T.O. P 00 Np.1.p.e.\.P.	0018E8F4 0000 0018E8F8 0000 0018E8F8 0000 0018E8FC 0000	00003 00000 00000

Figure 117

Due to the fact that "-c" was passed as a parameter, the next step is copying the file specified as a parameter to the local host:

C:\Users' Desktop\psexec.exe	-	×	
PsExec v2.11 - Execute processes remotely Copyright (C) 2001-2014 Mark Russinovich Sysinternals - www.sysinternals.com		î	
Copying win.exe to DESKTOP-2			

Figure 118

The file is copied to the ADMIN\$ share using CopyFileW:

	 004046F8 004046FA 004046FB 	6A 01 53 68 40 C5 43 00	push 1 push ebx push psexec.43C	540	x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0				
esi= <kernel< th=""><th>32.CopyFileW> (7</th><th>FF D6 (64CE480)</th><th>call esi</th><th></th><th>es1:C0 v</th><th>Default (stdcall)</th></kernel<>	32.CopyFileW> (7	FF D6 (64CE480)	call esi		es1:C0 v	Default (stdcall)			
.text:00404	700 psexec.exe:	4700 #3B00				3: [esp+8] 0000001 4: [esp+C] 76ACDC00 <kernel32.createeventw> 5: [esp+10] 76ACDDE0 <kernel32.createfilew></kernel32.createfilew></kernel32.createeventw>			
Dump 1	🕴 Dump 2 🛛 🐇	Dump 3 🚯 Dump 4	Dump 5 Watch 1	Locals 🎾 Struct	0018EBAC 0043 0018EBB0 0019 0018EBB4 0000	3C540 L"win.exe" 93AE4 L"\\\\DESKTOP- L \\ADMIN\$\\win.exe" 00001			

Figure 119

The PsExec process will execute the binary from above, as described in figure 120.



Figure 120

We're going to describe the activity of the PSEXESVC.exe process that was started earlier. The only difference between this case and the first one is that the process is running on the local machine instead of the remote host.

The PSEXESVC.exe process reads data from the "\pipe\PSEXESVC" pipe via a function call to ReadFile:

	 0040164A 0040164C 0040164D 0040164E 00401650 00401651 	6A 00 57 52 8B F0 56 53	pus pus mov pus pus	h o h edi h edx esi,eax h esi h ebx						x87StatusWord 0000 x87SW_B 0 x87SW_C3 0 x87SW_C1 0 x87SW_C0 0 x87SW_SF 0 x87SW_P 0	x87SW_C2 x87SW_E5 x87SW_U	0
ebp= <kernel< th=""><th>→ 00401652 < 32.ReadFile> (76</th><th>ACE160)</th><th>cal</th><th>евр</th><th>-</th><th></th><th>-</th><th>ebp:</th><th>:R# ~ [</th><th>Default (stdcall) 1: [esp] 00000288 2: [esp+4] 001D3C90 3: [esp+8] 00004A60 4: [esp+C] 03138094</th><th>•</th><th>5 🗘 🗌 Unlock</th></kernel<>	→ 00401652 < 32.ReadFile> (76	ACE160)	cal	евр	-		-	ebp:	:R# ~ [Default (stdcall) 1: [esp] 00000288 2: [esp+4] 001D3C90 3: [esp+8] 00004A60 4: [esp+C] 03138094	•	5 🗘 🗌 Unlock
Dump 1	Dump 2	ump 3 Ump 4	Dump 5	Watch 1	[x=] Locals	3 Struct		03138038 00	000028	8		
Address He 001D3C90 CC	x	1D 00 00 00 00 0	AS	CII			- ^	03138040 00 03138044 03 03138048 00	0004A60 3138094 0000000	0 0 4		

Figure 121

The file decrypts the encrypted data using CryptDecrypt (see figure 122). The encryption algorithm is AES256, and the key is derived based on the same approach as in the first case.



The resulting buffer contains the size of the buffer - 8, PsExec process ID in hex, local computer name, and the file that will be executed:

Address	He	x															ASCII
001D3C90	58	4A	00	00	4C	16	00	00	44	00	45	00	53	00	4B	00	XJLD.E.S.K.
001D3CA0	54	00	4F	00	50	00	2D	00									T.O.P
001D3CB0							00	00	00	00	00	00	00	00	00	00	
001D3CC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3CD0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3CE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3CF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	Figure 123
001D3D20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3D90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3DA0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
001D3DB0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
DOEVEC	N/C	~ ~ ~	~ ~		+	2		~ ~ ~	1	-	~ ~	مالہ	a "	11 1.		<u>~\</u> D	CEVECV/C < Local computer

PSEXESVC.exe creates 3 named pipes called "\\.\pipe\PSEXESVC-<Local computer

name>-<PSEXESVC Process ID>-stdin\stdout\stderr" using the

CreateNamedPipeW function (0x80001 = FILE_FLAG_FIRST_PIPE_INSTANCE |

PIPE_ACCESS_INBOUND, 0x6 = PIPE_TYPE_MESSAGE |

PIPE_READMODE_MESSAGE):

	push gai push 10000 push 10000 push 10000 push 5 push 80001 lea ecx.dword ptr ss:[esp+4AD4] call edi	
Ump 1 Ump 2 Ump 3 Ump 4	🕮 Dump 5 👹 Watch 1 🛛 🗱 Struct	0317E058 0317FB30 L"\\\.\\pipe\\PSEXESVC-DESKTOP- 0317B05C 00080001
Address Hex	ASCII	03178060 00000006
0317FB30 SC 00 5C 00 5C 00 70 00 69 00 0317FB40 SC 00 50 00 53 00 45 00 58 00 45 00 58 00 45 00 58 00 45 00 58 00 45 00 58 00 45 00 58 00 45 00 58 00 45 00 58 00 45 00 58 00 45 00 58 00 48 00 10 10 48 00 10 10 32 00 31 00 32 00 33 00 30 30 36 00 31 00 32 00	70 00 65 00 N.\\.p.1.p.e. 53 00 56 00 \.P.S.E.X.E.S.V. 54 00 4F 00 CD.E.S.K.T.O. P 20 00 73 00 03.6.1.25.	03178068 00010000 0317806C 00010000 03178070 FFFFFFF 03178074 031780C0 03178075 00409300 prevenue 00409340

Figure 124

ConnectNamedPipe is used to enable the named pipe server process (PSEXESVC) to wait for a client process (psexec.exe) to connect to the pipes:

	 00405E82 00405E84 00405E85 	6A 00 53 8B F8	push o push ebx mov edi,eax		x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0				
EIP	→• 00405E87 <	FF D6	call esi		es1:Ct v	Default (stdcail)	• 5	🗘 🗌 Unlock	
esi= <kern .text:004</kern 	el32.ConnectNamed 05E87 psexesvc.ex	Pipe> (76A74350) e:\$5E87 #5287				1: [esp] 0000002DC 2: [esp+4] 00000000 3: [esp+8] 004092A0 p 4: [esp+C] 004092A0 p	osexesvc.004092A0 osexesvc.004092A0		
Dump 1	Dump 2	Dump 3 🔛 Dump 4	🕮 Dump 5 🛛 👹 Watch 1 🛛 🕼 🖛 🕅 Locals	2 Struct	03178070 0000 03178074 0000	002DC 00000			
	10-								

Figure 125

The OpenProcessToken function opens the access token associated with the current process (0xB = TOKEN_QUERY | TOKEN_DUPLICATE | TOKEN_ASSIGN_PRIMARY):



The binary creates a new access token that duplicates the above token by calling the DuplicateTokenEx routine (0x2000000 = **MAXIMUM_ALLOWED**, 0x1 = **TokenPrimary**):

EIP dword ptr [4	0040507C 0040507D 0040507D 00405081 00405081 00405082 00405082 € 00425085 € 0042401C <psex< th=""><th>51 6A 0 53 53 68 0 52 FF 1 FF 1</th><th>1 0 00 00 0 5 1C 40 4 DicateTo</th><th>2 2 00 <enex>]=<advap< th=""><th>push ecx push 1 push ebx push ebx push ebx push 2000000 push edx call dword p i32.Duplicat</th><th>tr ds:[<&D eTokenEx></th><th>iplicateToken</th><th>EX>]</th><th></th><th>> De</th><th>875tatusword 0000 875w_B 0 x875w_C3 875w_C1 0 x875w_C 875w_SF 0 x875w_P 875w_SF 0 x875w_P efault (stdcal) (esp+4) 0200000 (esp+4) 00000000</th><th>0 x875W_C2 0 x875W_E5 0 x875W_U</th><th>0 0 2 5 👽 🗆 Unlock</th></advap<></enex></th></psex<>	51 6A 0 53 53 68 0 52 FF 1 FF 1	1 0 00 00 0 5 1C 40 4 DicateTo	2 2 00 <enex>]=<advap< th=""><th>push ecx push 1 push ebx push ebx push ebx push 2000000 push edx call dword p i32.Duplicat</th><th>tr ds:[<&D eTokenEx></th><th>iplicateToken</th><th>EX>]</th><th></th><th>> De</th><th>875tatusword 0000 875w_B 0 x875w_C3 875w_C1 0 x875w_C 875w_SF 0 x875w_P 875w_SF 0 x875w_P efault (stdcal) (esp+4) 0200000 (esp+4) 00000000</th><th>0 x875W_C2 0 x875W_E5 0 x875W_U</th><th>0 0 2 5 👽 🗆 Unlock</th></advap<></enex>	push ecx push 1 push ebx push ebx push ebx push 2000000 push edx call dword p i32.Duplicat	tr ds:[<&D eTokenEx>	iplicateToken	EX>]		> De	875tatusword 0000 875w_B 0 x875w_C3 875w_C1 0 x875w_C 875w_SF 0 x875w_P 875w_SF 0 x875w_P efault (stdcal) (esp+4) 0200000 (esp+4) 00000000	0 x875W_C2 0 x875W_E5 0 x875W_U	0 0 2 5 👽 🗆 Unlock
.text:004050	087 psexesvc.	exe: \$5087	#4487							4	: [esp+C]_00000000		
Ump 1	Ump 2	Dump 3	Ump	4 🗰 Dump 5	🛞 Watch 1	[x=] Locals	Struct		031766C4 031766C8	000002	EC 00		
Address He:	x				ASCII	al an		^	031766CC	000000	00		
031767A8 44 031767B8 00	00 00 00 00 00	00 00 00 00 00	00 00 00 00	00 00 00 00 00 00 00 00 00 00 00	0				031766D4 031766D8	000000000000000000000000000000000000000	01 F0		

Figure 127

The Wow64DisableWow64FsRedirection API is utilized to disable file system redirection for the current thread:



Figure 128

The process forces the system not to display the critical-error-handler messages via a call to SetErrorMode (0x1 = SEM_FAILCRITICALERRORS):

61P 004056F0 004056F2	6A 01 FF 15 AC 41 42 00	push 1 call dword ptr ds: [<&SetErrorMode>]		Default (stdcall)
dword ptr [004241AC <psexes .text:004056F2 psexesvc.exe</psexes 	vc.&SetErrorMode>]= <ker< td=""><td>hel32.SetErrorMode></td><td></td><td>1: [esp] 00000001 2: [esp+4] 000002E8 3: [esp+8] 76A74350 <kernel32.connectnamedpipe> 4: [esp+C] 000002E0</kernel32.connectnamedpipe></td></ker<>	hel32.SetErrorMode>		1: [esp] 00000001 2: [esp+4] 000002E8 3: [esp+8] 76A74350 <kernel32.connectnamedpipe> 4: [esp+C] 000002E0</kernel32.connectnamedpipe>
100 a 100 a 100		- 26	03176608 000	00001

Figure 129

The CreatePipe function is repeatedly used to create three anonymous pipes:



Figure 130

The write handles' properties are modified using SetHandleInformation (0x1 = **HANDLE_FLAG_INHERIT**):

00405752 53 00405753 6A 01 00405755 50			1	push ebx push 1 push eax				x875W_C1 0 x875W_C0 0 x875W_ES 0 x875W_SF 0 x875W_P 0 x875W_U 0				
	→• 0040575 <	G FF D	5		call ebp			ebp: Sł 🗸	Default (s	tdcall) p] 000002F4	▼ 5 €	Unlock
.text:00405	5756 psexesvo	.exe:\$5756	#4856)					2: [es 3: [es 4: [es	p+4] 00000001 p+8] 00000000 p+C] 000002E8	· · ·	
Dump 1	Dump 2	Dump 3	Dump 4	Dump 5	👹 Watch 1	[x=] Locals	Struct	031766D0 000 031766D4 000	002F4 00001			
Address H	ev				ASCTT			031766D8 000	00000			

Figure 131

The PSEXESVC process executes the file passed through the named pipe using the CreateProcessAsUserW API (0x414 = CREATE_UNICODE_ENVIRONMENT | CREATE_NEW_CONSOLE | CREATE_SUSPENDED):

	14 24 24 77 24 D0 00 00 36 28 46 00 00 14 24 24 14 04 00 36 24 48 07 00 00 36 24 48 07 00 00	push edx mov edx,dword ptr ss:[esp+24] lea ecx,dword ptr ss:[esp+00] push ecx push ecx push ecx mov eax,dword ptr ds:[es1+4628] push edx mov edx,dword ptr ss:[esp+24] or eax,414 push ebx push ebx push ebx push ebx push ecx push ecx push ecx push ecx push ebx push ebx push ebx push ebx push ebx push ebx push ebx push ebx push ebx push ebx	ecx:L'	X8/F4 000000000000000000000000000000000000
dword ptr [00424028 <psexesvc.&cr .text:004058F6 psexesvc.exe:\$58F6</psexesvc.&cr 	eateProcessAsUserW>]=<	advapi32.CreateProcessAsUserW>		2: [esp+4] 00000000 3: [esp+8] 03176E04 L"\"win.exe\" " 4: [esp+C] 00000000
Dump 1 Dump 2 Dump 3	Dump 4 Dump 5	🛞 Watch 1 🕅 🖉 Struct 🛛	176680 0000	002F0
Address Hex 02E717F8 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 0	ASCII 03	3176688 0317 317668C 0000 31766C0 0000	'6E04 L'\'win.exe\" " 00000

The execution of the above thread is resumed via a function call to ResumeThread. As in the first case, the process creates three similar threads, and their execution will not be detailed again: sub_404B90, sub_404AD0, and sub_404D10.

PsExec writes the confirmation that the new process was successfully started (including the process ID) and then waits for the process to finish:

INDESKTOP-			×
PsExec v2.11 - Execute proces: Copyright (C) 2001-2014 Mark I Sysinternals - www.sysinterna	es remotely ussinovich s.com		^
win.exe exited on DESKTOP-	with error code 1.		
Figure 133			

The "PSEXESVC" service is stopped by calling the ControlService API (0x1 = **SERVICE_CONTROL_STOP**):

00402A37 52 00402A38 6A 01 00402A3A 56	push edx push 1 push esi	x875W_C1 0 x875W_C0 0 x875W_E5 0 x875W_SF 0 x875W_P 0 x875W_U 0
ETC → 00402A3B FF 15 20 70 42 0	<pre>0 call dword ptr ds:[<&ControlService>]</pre>	Default (stdcall)
<pre>dword ptr [00427020 <psexec.&controlservice>]* .text:00402A38 psexec.exe:\$2A38 #1E38</psexec.&controlservice></pre>	< <advapi32.controlservice></advapi32.controlservice>	1: [esp+4] 00000001 3: [esp+4] 0000001 3: [esp+5] 0018657C 4: [esp+10] 028307C8
Dump 1 Dump 2 Sump 3 Dump 4	🍘 Dump 5 🥈 Watch 1 🚯 Locals 🥬 Struct	0018E560 02833DE0 0018E564 00000001
Address Hex 00430EF0 50 00 53 00 45 00 58 00 45 00 53 00 5	ASCII 56 00 43 00 P.S.E.X.E.S.V.C.	00188568 0018857C 0018856C 00430EF0 return to psexec.00430EF0 from ???

Figure 134

The executable deletes the "PSEXESVC" service via a call to DeleteService:

• 00402ACF	56	push esi				0 110000 0 0
ETP 00402AD0	FF 15 1C 70 42 00	<pre>call dword ptr ds:[<&DeleteService>]</pre>	~	Def	fault (stdcall)	🔻 💈 🗘 Unlock
dword ptr [0042701C <psexed .text:00402AD0 psexec.exe:</psexed 	:.&DeleteService>]= <advap< th=""><th>132.DeleteService></th><th></th><th>1: 2: 3: 4: 5:</th><th>[esp] 02819678 [esp+4] 00430EF0 [esp+8] 028307C8 [esp+C] 004028F8 [esp+10] 028307C0</th><th>L"PSEXESVC" psexec.004028F8</th></advap<>	132.DeleteService>		1: 2: 3: 4: 5:	[esp] 02819678 [esp+4] 00430EF0 [esp+8] 028307C8 [esp+C] 004028F8 [esp+10] 028307C0	L"PSEXESVC" psexec.004028F8
An		• V • • • •	0018E584 028	1967	8	

Figure 135

DeleteFileW is used to delete the PSEXESVC.exe file created earlier:

00402F07 50 00402F03 FF 15 7C 71 42 00 C 00402F03 C 71 42 00	<pre>push eax call dword ptr ds:[<&DeleteFilew>]</pre>	eax:L"	Default (stdcal)
<pre>dword ptr [0042717C <psexec.&deletefilew>]=<kernel3 #23d8<="" .text:00402fd8="" pre="" psexec.exe:\$2fd8=""></kernel3></psexec.&deletefilew></pre>	2.DeleteFilew>		2: [csp+4] 0043218 L'DESKTOP- 3: [csp+8] 76ACDB30 <kernel32.duplicatehandle> 4: [csp+6] 76ACDB30 <kernel32.closehandle> 5: [csp+10] 76A74F10 <kernel32.getlasterror></kernel32.getlasterror></kernel32.closehandle></kernel32.duplicatehandle>
Figure 136	sil ve (∧ (d) ()	0018E5B0 001	8E7DO L"C:\\WINDOWS\\PSEXESVC.exe"

References

MSDN: <u>https://docs.microsoft.com/en-us/windows/win32/api/</u>

PsExec: <u>https://docs.microsoft.com/en-us/sysinternals/downloads/psexec</u>