Finfisher rootkit analysis

artemonsecurity.blogspot.de/2017/01/finfisher-rootkit-analysis.html

My previous blog post was dedicated to very interesting malware that is called **Wingbird**. This malware has been used by **NEODYMIUM** cyber espionage group and contains rootkit to execute sensitive and important operations for attackers in a system. The first sample used rootkit for injection malicious code into Winlogon with removing ESET driver hooks in kernel SSDT, while second deploys rootkit for bypassing FS sandbox of several security products. Both droppers analyzed in 32-bit environment, while their behaviour in 64-bit Windows versions are interesting too and different from what we have seen in the 32-bit versions.



In 64-bit system, the dropper doesn't resort to the use of kernel mode rootkit (obviously, due to DSE restrictions) for injection malicious code & data into trusted Winlogon process. Instead this, it uses special trick for masking its malicious activity and for performing injection. The dropper uses copy of trusted LSASS process (executable file) and forces it to load malicious dll with standart name that is imported by LSASS.



64-bit GMER anti-rootkit tool demonstrates injection anomalies into Winlogon and Svchost, where malicious code is located.

Rootkit/M	alware >>>	
Туре	Name	Value
Thread	C:\Windows\system32\winlogon.exe [508:884]	00000000cf0000
Thread	C:\Windows\system32\winlogon.exe [508:1172]	000000000ca18ec
Thread	C:\Windows\system32\winlogon.exe [508:2132]	000000002f6ecb0
Thread	C:\Windows\system32\winlogon.exe [508:2060]	000000002f6ff5c
Thread	C:\Windows\system32\winlogon.exe [508:2472]	000000000ca2b30
Thread	C:\Windows\system32\winlogon.exe [508:2544]	000000000ca2b30
Thread	C:\Windows\system32\winlogon.exe [508:2516]	000000000ca2b30
Thread	C:\Windows\system32\winlogon.exe [508:2588]	000000000ca2b30
Thread	C:\Windows\system32\winlogon.exe [508:2480]	000000002f7a588
Thread	C:\Windows\system32\winlogon.exe [508:2856]	000000002f79fd4
Thread	C:\Windows\system32\winlogon.exe [508:864]	000000002f7b0ec
Thread	C:\Windows\system32\sychost.exe [628:792]	000000003a0000

The presence of virtual memory regions into Winlogon with the protection attribute PAGE_EXECUTE_READWRITE is an indicator that the process was compromised.

Dump Region					- • ×	
		Dum	p Refresh			
Address	Size	Access	State	Туре		^
00000000001A1000	000000000000F000	PAGE_NOACCESS	MEM_FREE			
0000000000180000	00000000006E000		MEM_RESERVE	MEM_PRIVATE		
00000000021E000	0000000000002000	PAGE_READWRITE PAGE_GUARD	MEM_COMMIT	MEM_PRIVATE		
000000000220000	0000000000010000	PAGE_READWRITE	MEM_COMMIT	MEM_PRIVATE		
000000000230000	0000000000009000	PAGE EXECUTE READWRITE	MEM COMMIT	MEM PRIVATE	_	
000000000239000	000000000007000	PAGE_NOACCESS	MEM_FREE		_	
0000000000240000	00000000000001000	PAGE EXECUTE READWRITE	MEM_COMMIT	MEM PRIVATE	_	
000000000241000	000000000000F000	PAGE_NOACCESS	MEM_FREE			
000000000250000	0000000000001000	PAGE_READONLY	MEM_COMMIT	MEM_PRIVATE		
000000000251000	000000000000F000	PAGE_NOACCESS	MEM_FREE			
000000000260000	0000000000001000	PAGE_EXECUTE_READWRITE	MEM_COMMIT	MEM_PRIVATE		
000000000261000	000000000000000000000000000000000000000	PAGE_NOACCESS	MEM_FREE		-	
000000000270000	0000000000002000	PAGE_READWRITE	MEM_COMMIT	MEM_PRIVATE		
000000000272000	000000000007E000		MEM_RESERVE	MEM_PRIVATE		
0000000002F0000	0000000000009000	PAGE_EXECUTE_READWRITE	MEM_COMMIT	MEM_PRIVATE		
0000000002F9000	0000000000007000	PAGE_NOACCESS	MEM_FREE			
000000000300000	0000000000001000	PAGE_EXECUTE_READWRITE	MEM_COMMIT	MEM_PRIVATE		
0000000000301000	000000000000000000000000000000000000000	PAGE_NOACCESS	MEM_FREE		•	
000000000310000	0000000000001000	PAGE_EXECUTE_READWRITE	MEM_COMMIT	MEM_PRIVATE	_	
000000000311000	000000000000000000000000000000000000000	PAGE_NOACCESS	MEM_FREE		_	-
	000000000007000	DAGE DEADWOTTE	MEM COMMIT	MEM DOTVATE		

As I already noted in previous blog post, Wingbird malware shares similarities with another malware that is called **Finfisher**. For example, in malicious PE-file that was dumped from Winlogon memory region, we can see reference to name of Finfisher rootkit

(nssoundax.sys).	
	block1E_oE1000	

	b.	lock15_e	51000					↓I	FRO			000	900000	www.hiew	.ru
	-EE¥	рЫ(w	өхөм	Ë\(w	Ë- +w	► +w	∭ + +w	►J'w	∭Б'₩	P[(w	1EE¥	a≝+w	p+)w	Ë5)w	LA'w
	AbE♥	(→	∭f'w	Д+╝	a‼(w	[_] dE♥	PR(w	P= 'w	÷ZE♥	L3'w	п'w	≒ aE♥		@?(w	?(w
	+)w	4^E♥		ËHk	¤∎∙ ok	42 •	QE:W	⊳ъ	• ́р⊳ъ	• ►Вы	• д1ъ	• \$#ъ	• \/ъ	•	P‼>w
			{)GP	🙂 B	<u>↑</u> ‼↑•	Tx	LTX	(AA	КАБА 🕨 🕬	KAЖBA¶ <mark>+</mark> +E	EEEEE	00APAA	('орыл	 700РРИ 	(АИА)
	A 💦	`hhh <mark>••</mark> •x	ppwpp		%	04d-5	602d-	% 0 2 (d %02	2 d : % e	2 : : :	602d		DST	+ ⁻ -
	%	s % 0 2	d % 0 2	2 d U T	C%s	svcho	st.		vcho	st.e	x 🖕 m		n d d x	.sys	🥖 %d
	.%d.%d	.%d	cmd.	ехе		exit	.txt	(94C.d	dat					
											infic	her r	ootki	ЕЙР\$ ЗЦЙЦ\$	SVWHГь
1111	3 HET	HЛёtoHБ∙	∆v † ąk	I • AE x.	HHz LHL\$	хз <mark>нл⊦§_</mark>)	¤ E ^L x,₿HU	lH; w∙uA	∕I∟7ы <mark>●</mark> И∟7 ₁	ız •АЛ НГ		FFFFFLЙL	л\$1∟Й∟\$∴	SVWНГБ Н т ъ	нлёт
-		• • •			valual, c					ue icl			al cool - u	rteur o éu	- 1. ⁻

After Isass service started, it injects code into winlogon and with help of ProcMon boot logging we can identify first actions that come from malicious code.

winlogon.exe winlogon.exe	480 CloseKey	HKLM\SOFTWARE\Microsoft\Windows\Cu HKLM\SOFTWARE\Microsoft\Windows\Cu	🗂 Even	Properties	forExtomalManifort		
winlogon.exe winlogon.exe	480 Arread Create	C:\Windows\System32\userinit.exe	Event	Process Stack			
winlogon.exe	100 Crosto Ele	C:\Windows\System32\ntdll.dll	Evenit	Process brack			
winlogon.exe	480 🛃 Query Standard Information File	C:\Windows\System32\ntdll.dll					
winlogon.exe	480 🛃 ReadFile	C:\Windows\System32\ntdll.dll	Fram	e Module	Location	Address	Path
winlogon.exe			KO	fltmgr.sys	fltmgr.sys + 0x2067	0xfffff880010f3067	C:\Windows\s
winlogon.exe	480 🧟 Load Image	C:\Windows\System32\ws2_32.dll		ntmgr.sys	ritmgr.sys + 0x49aa	Domm880010r59aa	C:\Windows\s
winlogon.exe	480 🧟 Load Image	C:\Windows\System32\nsi.dll	K 2	fitmgr.sys	fitmgr.sys + 0x222a3	0xmm880011132a3	C:\Windows\s
winlogon.exe	480 🧟 Load Image	C:\Windows\System32\shell32.dll	K 3	ntoskml.exe	ntoskml.exe + 0x37e495	0xffff80002b8b495	C:\Windows\s
winlogon.exe	480 🌌 Load Image	C:\Windows\System32\shlwapi.dll	K 4	ntoskml.exe	ntoskml.exe + 0x37ad38	0xfffff80002b87d38	C:\Windows\s
winlogon.exe	480 🛃 Create File	C:\Windows\System32\IPHLPAPI.DLL	K 5	ntoskml.exe	ntoskml.exe + 0x37bf56	0xffff80002b88f56	C:\Windows\s
winlogon.exe	480 🛃 Query Basic Information File	C:\Windows\System32\IPHLPAPI.DLL	K 6	ntoskml.exe	ntoskml.exe + 0x37d85c	0xfffff80002b8a85c	C:\Windows\s
winlogon.exe	480 🛃 Close File	C:\Windows\System32\IPHLPAPI.DLL	K 7	ntoskml.exe	ntoskml.exe + 0x388478	0xfffff80002b95478	C:\Windows\s
winlogon.exe	480 🛃 Create File	C:\Windows\System32\IPHLPAPI.DLL	K	ntoskml eve	ntoskml eve + 0x7f8d3	0xfffff8000288c8d3	C:\Windows\s
winlogon.exe	480 🛃 Create File Mapping	C:\Windows\System32\IPHLPAPI.DLL		ntourd II.co.co	atall all + 0xE19Ca	0.770-100-	C:\Windows\S
winlogon.exe	480 🛃 Create File Mapping	C:\Windows\System32\IPHLPAPI.DLL					C. Windows (5
winlogon.exe	480 🌌 Load Image	C:\Windows\System32\IPHLPAPI.DLL		U KemelBase.	dii KemelBase.dii + UX5516	Ux /refd285516	C:\Windows\S
winlogon.exe	480 🛃 Close File	C:\Windows\System32\IPHLPAPI.DLL	U 1	1 kemel32.dll	kemel32.dll + 0x118ed	0x/6e818ed	C:\Windows\S
winlogon.exe	480 🛃 Create File	C:\Windows\System32\winnsi.dll		2 <unknown></unknown>	0xbb00d7	0xbb00d7	
winlogon.exe	480 🛃 Query Basic Information File	C:\Windows\System32\winnsi.dll					
winlogon.exe	480 🛃 Close File	C:\Windows\System32\winnsi.dll		aiact	ad cada (start a	
winlogon.exe	480 🛃 Create File	C:\Windows\System32\winnsi.dll		ijecu	eu coue s	διαίι αι	uur
winlogon.exe	480 🛃 Create File Mapping	C:\Windows\System32\winnsi.dll					
winlogon.exe	480 🛃 Create File Mapping	C:\Windows\System32\winnsi.dll					
winlogon.exe	480 🏹 Load Image	C:\Windows\System32\winnsi.dll					
winlogon.exe	480 🛃 Close File	C:\Windows\System32\winnsi.dll					
winlogon.exe	480 🌄 Load Image	C:\Windows\System32\setupapi.dll					
winlogon.exe	480 🚑 Load Image	C:\Windows\System32\cfgmgr32.dll					
winlogon.exe	480 🧟 Load Image	C:\Windows\System32\oleaut32.dll					
winlogon.exe	480 🚑 Load Image	C:\Windows\System32\devobj.dll	<				>
winlogon.exe	480 🌉 RegQueryKey	HKLM			Broportion Course	Courses	E avec
winlogon.exe	480 KRegOpenKey	HKLM\SOFTWARE\Microsoft\OLEAUT			Properties Search	source	Save
winlogon.exe	480 🌉 RegQueryKey	HKLM					
winlogon.exe	480 KRegOpenKey	HKLM\SOFTWARE\Microsoft\OLEAUT				Corry All	Close
winlogon.exe	480 🌉 RegQueryKey	HKLM	TI	🔄 🔄 Next Highli	ighted	Copy All	Close
winlogon.exe	480 🌉 RegOpenKey	HKLM\Software\Microsoft\Windows\Curren					
		LUCIE CONTRACTOR AND A LONG		0.000		NUMBER NOT	

After some preliminary actions, malicious code into Winlogon tries to communicate with hard disk on low level, it requests disk geometry info and sends SCSI control code for reading data. In 32-bit version it uses rootkit to perform this operation.

Time	Process Name	PID	Operation	Path	Result	Detail	User
18:25:	winlogon.exe	480	-CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read/Write, Disposition: Open, Opti	NT AUTHORITY\cucrema
18:25:	winlogon.exe	480	DeviceloControl	\Device\Harddisk0\DR0	SUCCESS	Control: IOCTL_DISK_GET_DRIVE_GEOMETRY_EX	NT AUTHORITY\cиcтема
18:25:	winlogon.exe	480	DeviceloControl	\Device\Harddisk0\DR0	SUCCESS	Control: IOCTL_SCSI_PASS_THROUGH_DIRECT	NT AUTHORITY\cиcтема
18:25:	winlogon.exe	480	🛃 Flush Buffers File	\Device\Harddisk0\DR0	INVALID DEVICE REQUEST		NT AUTHORITY\cucrema
18:25:	winlogon.exe	480	🛃 Set Basic Information File	\Device\Harddisk0\DR0	INVALID DEVICE REQUEST	CreationTime: 09.01.2016 18:25:29, LastAccessTime: 09.01	NT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	🛃 Flush Buffers File	\Device\Harddisk0\DR0	INVALID DEVICE REQUEST		NT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	🛃 Close File	\Device\Harddisk0\DR0	SUCCESS		NT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	-CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Generic Read, Disposition: Open, Options: S	NT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	🛃 ReadFile	\Device\Harddisk0\DR0	SUCCESS	Offset: 0, Length: 512, I/O Flags: Non-cached, Priority: Normal	NT AUTHORITY\cuctema
18:25:	winiogon.exe	480	-NUIOSEFIIE	\Device \Harddisku \D Ku	SULLESS		INT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	CreateFile	\Device\Harddisk0\DR0	SUCCESS	Desired Access: Pennic Dear Write, Disposition: Open, Opti	NT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	DeviceloControl	\Device\Harddisk0\DR0	SUCCESS		NT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	DeviceloControl	\Device\Harddisk0\DR0	SUCCESS	Control: IOCTL_SCSI_PASS_THROUGH_DIRECT	NT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	🛃 Flush Buffers File	\Device\Harddisk0\DR0	INVALID DEVICE REQUEST		NT AUTHORITY\cuctema
18:25:	📓 winlogon.exe	480	Set Basic Information File	\Device\Harddisk0\DR0	INVALID DEVICE REQUEST	Creation Time: 09.01.2016 18:25:29, LastAccess Time: 09.01	NT AUTHORITY\cиcтема
18:25:	winlogon.exe	480	🛃 Flush Buffers File	\Device\Harddisk0\DR0	INVALID DEVICE REQUEST		NT AUTHORITY\cucrema
18:25:	winlogon.exe	480		\Device\Harddisk0\DR0	SUCCESS		NT AUTHORITY\cиcтема

It also checks presence of Finfisher files. See details in Symantec blog post here.

18:25:	🔛 winlogon.exe	480 🛃 Close File	C·\ProgramData\SecuritySvc	SUCCESS	
18:25:	winlogon.exe	480 🛃 Create File	C:\ProgramData\SecuritySvc\ico_ty23.ico	NAME NOT FOUND	Desir
18:25:	winlogon.exe	480 🛃 Create File	C:\ProgramData\SecuritySvc\ico_ty23.ico	NAME NOT FOUND	Desir
18:25:	📓 winlogon.exe	480 🛃 Create File	C:\ProgramData\SecuritySvc\ico_ty23.ico	NAME NOT FOUND	Desir
18:25:	📓 winlogon.exe	480 🛃 Create File	C:\ProgramData\SecuritySvc\ico_ty23.ico	NAME NOT FOUND	Desir
18:25:	📓 winlogon.exe	480 🛃 Create File	C:\ProgramData\SecuritySvc\ico_sf46.ico	NAME NOT FOUND	Desir
18:25:	📓 winlogon.exe	480 🛃 Create File	C:\ProgramData\SecuritySvc\ico_sf46.ico	NAME NOT FOUND	Desir
18:25:	winlogon.exe	480 🛃 Create File	C:\ProgramData\SecuritySvc\ico_sf46.ico	NAME NOT FOUND	Desir
18:25:	📓 winlogon.exe	480 🛃 Create File	C:\ProgramData\SecuritySvc\ico_sf46.ico	NAME NOT FOUND	Desir
18:25	🕼 winlogon exe	480 🖶 CreateFile	A REPORT OF A R	au	Desir

In AUTHORITY Curcewa Desired Access: Genetic Write, Read Attributes, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Genetic Write, Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Read Attributes, Delete, Disposition: Open, Options: Non-Direct..., NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Non-Direct..., NT AUTHORITY Curcewa Desired Access: Genetic Write, Read Attributes, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Genetic Write, Read Attributes, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Genetic Write, Read Attributes, Disposition: Open, Options: Non-Direct..., NT AUTHORITY Curcewa Desired Access: Cenetic Write, Read Attributes, Disposition: Open, Options: Non-Directory... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Cenetic Write, Read Attributes, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa Desired Access: Delete, Disposition: Open, Options: Synch... NT AUTHORITY Curcewa

The following indicators show similarities between Wingbird and Finfisher.

Indicator (Wingbird)
Checks presence of ico_ty23.ico
Checks presence of ico_sf46.ico
Contains link to mssounddx.sys
Checks presence of *.dat files

I was able to get 32-bit version of mssounddx.sys rootkit. As you can see on screenshot below, authors masked its file as legitimate Microsoft driver.

mssounddx.sys	Properties	×				
General Security	Details Previous Versions					
Property Description — File description Type File version Product name Product version Copyright Size Date modified Language Original filename	Value Microsoft DirectSound Provider System file 5.3.2600.2180 Microsoft® Windows® Operating System 5.3.2600.2180 ® Microsoft Corporation. All rights reserv 52,8 KB 11.01.2017 17:09 English (United States) mssounddx.sys	-				
Remove Properties and Personal Information						
	OK Cancel App	ly –				

Like Wingbird rootkit, Finfisher rootkit is protected from statical analysis. The code from *DriverEntry* and other functions in mssounddx.sys are representing a loader that decrypts content of BIN resource, where 2nd encrypted driver is located.

📽 [mssounddx.sys] - Cerbero PE Inside	🗃 [mssounddx.sys] - Cerbero PE Insider 1.0.2								
🔁 🖪 🗟 🔍 🖕 🛶 🖢 🔶	🔲 🗔 🗟 🎭 🏣 🐝 🖕 🌩 📓 隆 SHA-1 🔹 F50A7D46E3A8A615CE57C9632D1E3203FF926DBF								
📑 Format 🗗 🗶	🚽 Format 🛛 🗶 🔍 Analysis [Resource Directory]								
📴 Dos Header	Resources Tree	Offset	0 1 2 3 4 5 6 7	8 9 A B C D E F	Ascii				
Rich Signature	✓ 📄 "BIN"	00000000	2A 90 8E 5F 29 90 8E 5F	2D 90 8E 5F D2 6F 8E 5F	*)o				
 It Headers 	101 - [lang:1033]	00000010	6A 6F 8E 5F 6A 6F 8E 5F	2A 6F 8E 5F 2A 6F 8E 5F	jojo*o*o				
📑 File Header	Version info	00000020	2A 6F 8E 5F 2A 6F 8E 5F	2A 6F 8E 5F 2A 6F 8E 5F	*0*0*0*0				
✓		00000030	2A 6F 8E 5F 2A 6F 8E 5F	2A 6F 8E 5F 7A 6D 8E 5F	*o*o*ozm				
Data Directories		00000040	74 72 34 51 74 C6 3D 9C	55 7E 3C DO 98 5F 68 B8	tr4Qt.=.U~ <h.< td=""></h.<>				
Data Directories	Crypted driver	00000050	F1 2C 48 C8 83 43 2F BA	E2 2E OF D9 83 40 61 B6	.,HC/@a.				
Section Headers	crypted arres	00000060	F7 60 03 D3 D7 12 76 BD	F7 7B 18 9D B3 34 4B BD	.`v{4K.				
Import Directory		00000070	DE 5B 2F D8 F0 56 22 D2	D4 56 22 D2 D4 56 22 D2	.[/\"\"\".				
Resource Directory		00000080	D4 56 22 D2 D4 56 22 D2	D4 56 22 D2 D4 56 22 D2	.V"V"V".				
Relocation Directory		00000090	D4 56 22 D2 D4 56 22 D2	D4 56 22 D2 D4 56 22 D2	.V"V"V"V".				
Dahua Disastasi		0A000000	D4 56 22 D2 D4 56 22 D2	D4 56 22 D2 D4 56 22 D2	.V"V"V"V".				
Debug Directory		000000B0	D4 56 22 D2 D4 56 22 D2	D4 56 22 D2 D4 56 22 D2					
IAT Directory		000000000	D4 56 22 D2 D4 56 22 D2	D4 56 22 D2 D4 56 22 D2					
		000000000	D4 56 22 D2 D4 56 22 D2	D4 56 22 D2 D4 56 22 D2	.v"v"v"v".				

Rootkit code does following actions in DriverEntry.

1. It is looking for corresponding BIN resource into .rsrc section.

2. It allocates memory block from kernel pool and copies into it content of BIN resource with size 0xc180 (encrypted driver).

3. Decrypts data in allocated pool block.

4. Prepares PE-file of encrypted driver for work: applies fixups, fills some internal variables (ptrs to import functions).

5. Passes control to DriverEntry of decrypted driver.

.text:F6C71886	mov	esi, edx
.text:F6C71888	xor	esi, 5F1ECA67h
.text:F6C7188E	shr	eax, 2
.text:F6C71891	push	edi
.text:F6C71892	mov	[ecx], esi
.text:F6C71894	lea	edi, [eax-1]
.text:F6C71897	xor	esi, esi
.text:F6C71899	test	edi, edi
.text:F6C7189B	jbe	short loc_F6C718B2
.text:F6C7189D	push	ebx
.text:F6C7189E	-	
.text:F6C7189E loc_F6C7189E:		; CODE XREF: fnDecryptDriver+37įj
.text:F6C7189E	mov	eax, [ecx+esi*4+4]
.text:F6C718A2	MOV	ebx, eax
.text:F6C718A4	xor	ebx, edx
.text:F6C718A6	MOV	[ecx+esi*4+4], ebx
.text:F6C718AA	inc	esi
.text:F6C718AB	cmp	esi, edi
.text:F6C718AD	MOV	edx, eax
.text:F6C718AF	jb	short loc_F6C7189E
.text:F6C718B1	рор	ebx
.text:F6C718B2		
.text:F6C718B2 loc_F6C718B2:		; CODE XREF: fnDecryptDriver+231j
.text:F6C718B2	рор	edi
.text:F6C718B3	рор	esi
.text:F6C718B4	рор	ебр
.text:F6C718B5	retn	8
.text:F6C718B5 fnDecryptDrive	r endp	

Second driver uses following kernel functions.

1 ExFreePoolWithTag 2 _stricmp 3 ZwQuerySystemInformation 4 ExAllocatePoolWithTag 5 wcsicmp 6 ObfDereferenceObject 7 KeWaitForSingleObject 8 ZwClose 9 ZwFreeVirtualMemory 10 KeUnstackDetachProcess 11 PsLookupProcessByProcessId 12 ZwAllocateVirtualMemory 13 ZwOpenProcess 14 KeDelayExecutionThread 15 PsGetVersion 16 MmGetSystemRoutineAddress
17 RtlInitUnicodeString 18 memset 19 memcpy 20 KeServiceDescriptorTable 21 PsCreateSystemThread 22 wcsncpy 23 ZwQueryValueKey 24 ZwOpenKey 25 KeTickCount

Code injection.

.text:00010903push[ebp+var_2C].text:00010906callds:PsLookupProcessByProcessId.text:00010907pusheax, [ebp+var_84].text:00010912pusheax.text:00010913push[ebp+var_20].text:00010916call[ebp+var_34].text:00010917push[ebp+var_34].text:00010918push[ebp+var_30].text:00010917push[ebp+var_30].text:00010922callmemcpy.text:00010927push[ebp+var_40].text:00010928push[ebp+var_40].text:00010928pushesi.text:00010921addesp, 18h.text:00010934leaeax, [ebp+var_84].text:00010938callds:KeUnstackDetachProcess.text:00010938callds:KeUnstackDetachProcess.text:00010938callds:KeUnstackDetachProcess.text:00010934pusheax	.text:00010902	push	eax
.text:00010906callds:PsLookupProcessByProcessId.text:00010900leaeax, [ebp+var_84].text:00010912pusheax.text:00010913push[ebp+var_20].text:00010916call[ebp+pKeStackAttachProcess].text:00010917push[ebp+var_34].text:00010917push[ebp+var_30].text:00010917push[ebp+var_30].text:00010922callmemcpy.text:00010927push[ebp+var_40].text:00010928push[ebp+var_40].text:0010928pushesi.text:0010921callmemcpy.text:0010928pushesi.text:0010928pushesi.text:0010928pushesi.text:0010928pushesi.text:0010931addesp, 18h.text:0010934leaeax, [ebp+var_84].text:0010938callds:KeUnstackDetachProcess.text:0010934pusheax	.text:00010903	push	[ebp+var_2C]
.text:0001090C lea eax, [ebp+var_84] .text:00010912 push eax .text:00010913 push [ebp+var_20] .text:00010916 call [ebp+var_34] .text:00010917 push [ebp+var_34] .text:00010918 push [ebp+var_34] .text:00010917 push [ebp+var_30] .text:00010922 call memcpy .text:00010927 push edi .text:00010928 push [ebp+var_40] .text:00010928 push esi .text:00010931 add esp, 18h .text:00010934 lea eax, [ebp+var_84] .text:00010938 call ds:KeUnstackDetachProcess .text:00010938 call ds:KeUnstackDetachProcess .text:00010934 push eax	.text:00010906	call	ds:PsLookupProcessByProcessId
.text:00010912pusheax.text:00010913push[ebp+var_20].text:00010916call[ebp+pKeStackAttachProcess].text:00010917push[ebp+var_34].text:00010917push[ebp+var_30].text:00010922callmemcpy.text:00010927pushedi.text:00010928push[ebp+var_40].text:00010928pushesi.text:00010928pushesi.text:00010928pushesi.text:00010931addesp, 18h.text:00010934leaeax, [ebp+var_84].text:00010938callds:KeUnstackDetachProcess.text:00010938pusheax.text:00010934leaeax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax.text:00010934pusheax <td>.text:0001090C</td> <td>lea</td> <td>eax, [ebp+var_84]</td>	.text:0001090C	lea	eax, [ebp+var_84]
.text:00010913push[ebp+var_20].text:00010916call[ebp+pKeStackAttachProcess].text:00010919push[ebp+var_34].text:00010910push[ebp+var_30].text:00010917push[ebp+var_30].text:00010922callmemcpy.text:00010927pushedi.text:00010928push[ebp+var_40].text:00010928pushesi.text:00010928pushesi.text:00010928pushesi.text:00010928pushesi.text:00010928pushesi.text:00010928pushesi.text:00010928pushesi.text:00010931addesp, 18h.text:00010934leaeax, [ebp+var_84].text:00010938callds:KeUnstackDetachProcess.text:00010938callds:KeUnstackDetachProcess.text:00010941push2CCh	.text:00010912	push	eax
.text:00010916call[ebp+pKeStackAttachProcess].text:00010919push[ebp+var_34].text:0001091Cpush[ebp+var_30].text:0001091Fpush[ebp+var_30].text:00010922callmemcpy.text:00010927pushedi.text:00010928push[ebp+var_40].text:00010928pushesi.text:00010928callmemcpy.text:00010928pushesi.text:00010928pushesi.text:00010928callmemcpy.text:00010931addesp, 18h.text:00010934leaeax, [ebp+var_84].text:00010938callds:KeUnstackDetachProcess.text:00010938callds:KeUnstackDetachProcess.text:00010941push2CCh	.text:00010913	push	[ebp+var_20]
.text:00010919push[ebp+var_34].text:0001091Cpush[ebp+var_30].text:00010922callmemcpy.text:00010927pushedi.text:00010928push[ebp+var_40].text:00010928pushesi.text:00010920callmemcpy.text:00010928pushesi.text:00010920callmemcpy.text:00010921addesp, 18h.text:00010931addesp, 18h.text:00010934leaeax, [ebp+var_84].text:00010938callds:KeUnstackDetachProcess.text:00010938callds:KeUnstackDetachProcess.text:00010941push2CCh	.text:00010916	call	[ebp+pKeStackAttachProcess]
.text:0001091C push [ebp+var_3C] .text:0001091F push [ebp+var_30] .text:00010922 call memcpy .text:00010927 push edi .text:00010928 push [ebp+var_40] .text:00010928 push esi .text:00010920 call memcpy .text:00010928 push esi .text:00010920 call memcpy .text:00010921 add esp, 18h .text:00010931 add esp, 18h .text:00010934 lea eax, [ebp+var_84] .text:00010938 call ds:KeUnstackDetachProcess .text:00010938 call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:00010919	push	[ebp+var_34]
.text:0001091Fpush[ebp+var_30].text:00010922callmemcpy.text:00010927pushedi.text:00010928push[ebp+var_40].text:00010928pushesi.text:00010920callmemcpy.text:00010931addesp, 18h.text:00010934leaeax, [ebp+var_84].text:00010938callds:KeUnstackDetachProcess.text:00010941push2CCh	.text:0001091C	push	[ebp+var_3C]
.text:00010922 call memcpy .text:00010927 push edi .text:00010928 push [ebp+var_40] .text:00010928 push esi .text:0001092C call memcpy .text:00010931 add esp, 18h .text:00010934 lea eax, [ebp+var_84] .text:0001093A push eax .text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:0001091F	push	[ebp+var_30]
.text:00010927 push edi .text:00010928 push [ebp+var_40] .text:00010928 push esi .text:0001092C call memcpy .text:00010931 add esp, 18h .text:00010934 lea eax, [ebp+var_84] .text:0001093A push eax .text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:00010922	call	memcpy
.text:00010928 push [ebp+var_40] .text:0001092B push esi .text:0001092C call memcpy .text:00010931 add esp, 18h .text:00010934 lea eax, [ebp+var_84] .text:0001093A push eax .text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:00010927	push	edi
.text:0001092B push esi .text:0001092C call memcpy .text:00010931 add esp, 18h .text:00010934 lea eax, [ebp+var_84] .text:0001093A push eax .text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:00010928	push	[ebp+var_40]
.text:0001092C call memcpy .text:00010931 add esp, 18h .text:00010934 lea eax, [ebp+var_84] .text:0001093A push eax .text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:0001092B	push	esi
.text:00010931 add esp, 18h .text:00010934 lea eax, [ebp+var_84] .text:0001093A push eax .text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:0001092C	call	тетсру
.text:00010934 lea eax, [ebp+var_84] .text:0001093A push eax .text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:00010931	add	esp, 18h
.text:0001093A push eax .text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:00010934	lea	eax, [ebp+var_84]
.text:0001093B call ds:KeUnstackDetachProcess .text:00010941 push 2CCh	.text:0001093A	push	eax
.text:00010941 push 2CCh	.text:00010938	call	ds:KeUnstackDetachProcess
	.text:00010941	push	20Ch

Next picture demonstrates logic of 2nd driver execution.



Start of shellcode looks like.

.data:00011080		
.data:00011080 loc_11080:		; DATA XREF: sub_10490+CTo
.data:00011080 E8 E2 0A 00 00	call	sub_11B67
.data:00011085 E8 00 00 00 00	call	\$+5
.data:0001108A		
.data:0001108A loc_1108A:		; DATA XREF: .data:0001108Bjo
.data:0001108A 5D	рор	ebp
.data:0001108B 81 ED 8A 10 01+	sub	ebp, offset loc_1108A
.data:00011091 89 85 65 16 01+	mov	ss:dword_11665[ebp], eax
.data:00011097 89 9D 55 16 01+	mov	ss:dword_11655[ebp], ebx
.data:0001109D 89 B5 5D 16 01+	mov	ss:dword_1165D[ebp], esi
.data:000110A3 8D B5 B5 16 01+	lea	esi, dword_116B5[ebp]
.data:000110A9 8D 8D 71 16 01+	lea	edi, dword_11671[ebp]
.data:000110AF <mark>6A 00</mark>	push	0
.data:000110B1 68 FF 1F 7C C9	push	0C97C1FFFh
.data:000110B6 FF B5 65 16 01+	push	ss:dword_11665[ebp]
.data:000110BC E8 3C 0A 00 00	call	sub_11AFD
		—

Conclusion

As you can see from the analysis, we haven't seen something new in Finfisher rootkit. Like other drivers that are used by attackers, it is intended only for one purpose - for injection malicious code into Winlogon process. Nevertheless, authors use some anti-analysis tricks, including, driver encryption and obfuscation some data that driver keeps in kernel memory.