

■ positive technologies

Плагин rootkitmon для DRAKVUF. Выявление руткитов с помощью PT Sandbox

Павел Максютин

специалист отдела обнаружения вредоносного ПО экспертного
центра безопасности Positive Technologies (PT ESC)

Алексей Вишняков

руководитель отдела обнаружения вредоносного ПО экспертного
центра безопасности Positive Technologies (PT ESC)



ptsecurity.com

Содержание



- Суть проблемы
- Плагин rootkitmon
- Техники
- Что будет дальше
- Заключение

Содержание



- Суть проблемы
- Плагин rootkitmon
- Техники
- Что будет дальше
- Заключение

Суть проблемы



Загрузка неподписанного драйвера

```
C:\>kdmapper.exe s90ia3c.sys
[<] Loading vulnerable driver, Name: nyYGVhOFwYIMYV
[+] NtLoadDriver Status 0x0
[+] PiDDbLock Ptr 0xffffffff8007c943350
[+] PiDDbCacheTable Ptr 0xffffffff8007c94348c
[+] PiDDbLock Locked
[+] PiDDbCacheTable result -> TimeStamp: 5284eac3
[+] Found Table Entry = 0xFFFFFD50D2C5877E0
[+] PiDDbCacheTable Cleaned
[+] g_KernelHashBucketList Found 0xFFFFF8007D6AC080
[+] g_HashCacheLock Locked
[+] Found In g_KernelHashBucketList: nyYGVhOFwYIMYV
[+] g_KernelHashBucketList Cleaned
[+] MmUnloadedDrivers Cleaned: nyYGVhOFwYIMYV
[+] Image base has been allocated at 0xFFFFFE6058F3D9000
[+] Skipped 0x1000 bytes of PE Header
[<] Calling DriverEntry 0xFFFFFE6058F3D9000
[+] Callback example called
[+] DriverEntry returned 0x0
[<] Unloading vulnerable driver
[+] NtUnloadDriver Status 0x0
[+] Vul driver data destroyed before unlink
[+] success
```

not-wlan / drvmap Public

Code

Issues

Pull requests

Actions

master ▾

3 branches

0 tags

TheCruZ / kdmapper Public

Code

Issues

Pull requests

Actions

master ▾

1 branch

0 tags

Суть проблемы



The agent uses a known technique whereby the VirtualBox driver (VBoxDrv.sys) is leveraged to bypass the Driver Signature Enforcement mechanism in Windows and load Moriya's unsigned driver. DSE is an integrity mechanism mandating that drivers are properly signed with digital signatures in order for them to be loaded, which was introduced for all versions of Windows starting from Vista 64-bit. The technique used to bypass it was seen in use by other threat actors like Turla, Lamberts and Equation.

<https://securelist.com/operation-tunnelsnake-and-moriya-rootkit/101831/>

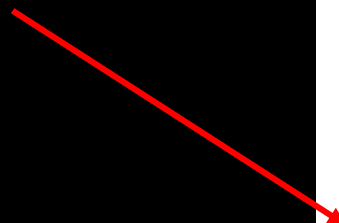
The approach used by the developer of this rootkit allows loading an unsigned driver without modifying the Code Integrity image and dealing with a potential crash. It abuses features of a legitimate and open-source² signed driver named dbk64.sys which is shipped along with Cheat Engine, an application created to bypass video game protections and introduce cheats into them. This driver provides capability to write and execute code in kernel space by design, thus allowing it to run arbitrary code in kernel mode.

<https://securelist.com/ghostemperor-from-proxylogon-to-kernel-mode/104407/>

Суть проблемы



Детект типичной загрузки драйвера



Руткит Rootkit.Win32.Generic.b
Эксплойт Exploit.Win32.KernelLPE.a
Эксплойт Exploit.Win32.VBoxDrv.a
Потенциально опасное поведение
Create.Process.Rundll32.RestrictionsBypass
Create.Service.Driver.Persistence
Create.Service.RPC.Persistence
Read.Device.Name.CheckVM

Содержание



- Суть проблемы
- **Плагин rootkitmon**
- Техники
- Что будет дальше
- Заключение

Плагин Rootkitmon

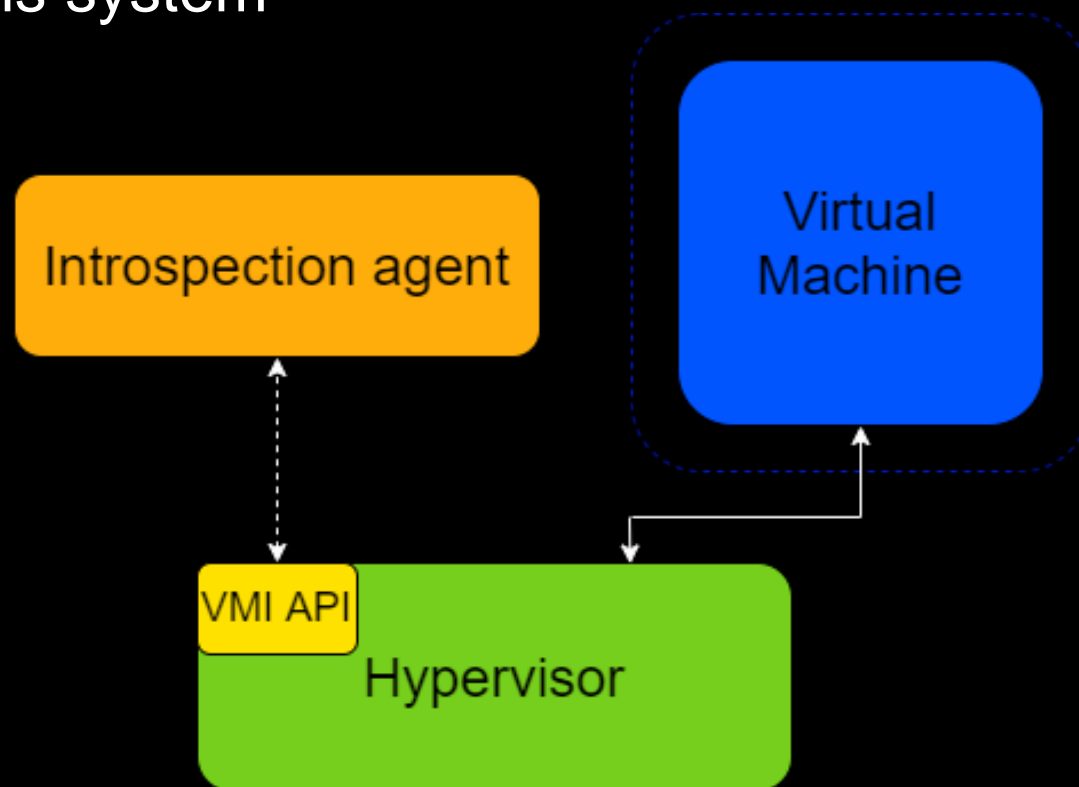


DRAKVUF — Black-box binary analysis system

- Доступ к памяти (r/w/x)
- Прерывания
- MSR-регистры
- Регистры управления

Изменение состояния системы:






- Регистры VCPUs
- Физическая память



Плагин Rootkitmon



Трасса выполнения

 regmon	Update LibVMI & tune formatting (#1167)
 rpcmon	libusermode, rpcmon: fix various errors (#1175)
 socketmon	Update LibVMI & tune formatting (#1167)
 ssdtmon	ssdtmon ssdt shadow update (#1312)
 syscalls	syscalls: no need to count arguments manually (#1304)

[**POOLMON**] TIME:1633628941 VCPU:0 CR3:0x9C3A7000

"\Windows\System32\svchost.exe":**ExAllocatePoolWithTag** SessionID:0 PID:844 PPID:476 VCPU:0

CR3:2621075456 Tag:"Wmip" Type:"PagedPool" Size:232

[**REGMON**] TIME:1633628941 VCPU:0 CR3:0x9C3A7000 "\Windows\System32\svchost.exe":**NtQueryValueKey**

SessionID:0 PID:844 PPID:476 Key:\HKLM\SYSTEM\CONTROLSET001\CONTROL\WM\SECURITY

ValueName:"0"

[**SYSCALL**] TIME:1633628941 VCPU:0 CR3:0x9C3A7000 "\Windows\System32\svchost.exe":**NtQueryValueKey**

SessionID:0 PID:844 PPID:476 Module:"nt" vCPU:0 CR3:0x9C3A7000 Syscall:20 NArgs:6

Плагин Rootkitmon



Что защищает плагин Rootkitmon:

- таблицы системных вызовов
- таблицы прерываний
- таблицы дескрипторов
- MSR LSTAR
- секций кода драйверов
- DRIVER_OBJECT и DEVICE_OBJECT
- различные API для подписки на системные события

Плагин Rootkitmon



Перехват KiDeliverApc для проверки в конце анализа

```
956 bool rootkitmon::stop()
957 {
958     if (!is_stopping() && !done_final_analysis)
959     {
960         m_is_stopping = true;
961         PRINT_DEBUG("[ROOTKITMON] Injecting KiDeliverApc\n");
962         // Hook dummy function so we could make final system analysis
963         auto hook = createSyscallHook("KiDeliverApc", &rootkitmon::final_check_cb);
964         if (!hook)
965         {
966             // Skip final analysis
967             PRINT_DEBUG("[ROOTKITMON] Failed to hook KiDeliverApc\n");
968             done_final_analysis = true;
969             return true;
970         }
971         this->syscall_hooks.push_back(std::move(hook));
972         // Return status `Pending`
973         return false;
974     }
975     // Return status `Pending`
976     return done_final_analysis;
977 }
```

Содержание



- Суть проблемы
- Плагин rootkitmon
- **Техники**
- Что будет дальше
- Заключение

SSDT-перехваты



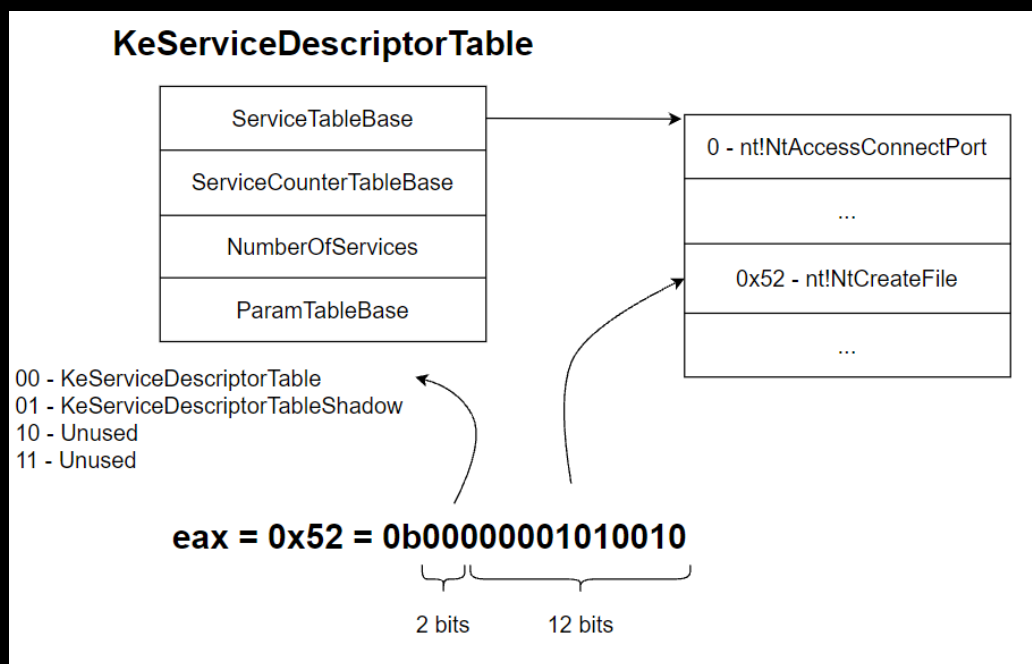
- SSDT – System Service Dispatch Table
- nt!KiServiceTable – таблица обработчиков нативных API
- win32k!W32pServiceTable – таблица обработчиков графической подсистемы win32k

```
0: kd> dd nt!KiServiceTable L10
fffff800`02ade300  040d9a00 02f55c00 fff6ea00 02e87805
fffff800`02ade310  031a4a06 03116a05 02bb9901 02b4f200
fffff800`02ade320  0312cc40 03dd7400 02c84700 02e7d100
fffff800`02ade330  02f68100 02e02301 02dd0601 02d96100
0: kd> dd win32k!W32pServiceTable L10
fffff960`00191f00  fff3a740 fff0b501 000206c0 001021c0
fffff960`00191f10  00096000 00022640 fff9a900 ffde0b03
fffff960`00191f20  ffb7a7c7 00fc5200 ffed2e80 ffe50e00
fffff960`00191f30  000c58c0 000af600 000e8bc0 fffeb300
```

SSDT-перехваты



Подсчет адреса обработчика системного вызова



```
0: kd> u ntdll!NtCreateFile L4
ntdll!ZwCreateFile:
00000000`76fd1860 4c8bd1      mov     r10,rcx
00000000`76fd1863 b852000000    mov     eax,52h
00000000`76fd1868 0f05          syscall
00000000`76fd186a c3            ret
0: kd> dd nt!KiServiceTable + (4 * 0x52) L1
fffff800`02ade448 03071007
0: kd> u nt!KiServiceTable + (0x03071007 >> 4)
nt!NtCreateFile:
fffff800`02de5400 4c8bdc      mov     r11,rsp
fffff800`02de5403 4881ec88000000 sub     rsp,88h
fffff800`02de540a 33c0        xor     eax,eax
fffff800`02de540c 498943f0    mov     qword ptr [r11-10h],rax
```

SSDT-перехваты



KeServiceDescriptorTable и KeServiceDescriptorTableShadow

```
v27 = (_eax >> 7) & 0x20;  
SyscallIndex = _eax & 0xFFF;  
do  
{  
    SDT = &KeServiceDescriptorTable;  
    v30 = &KeServiceDescriptorTableShadow;  
    if ( (*(&v17->0 + 1) & 0x80) != 0 )  
    {  
        if ( (*(&v17->0 + 1) & 0x200000) != 0 )  
            v30 = KeServiceDescriptorTableFilter;  
        SDT = v30;  
    }  
    if ( SyscallIndex < *(&SDT->ntoskrnl.NumberOfService + v27) )  
    {  
        ssdt = *(&SDT->ntoskrnl.ServiceTableBase + v27);  
        offset = ssdt[SyscallIndex];  
        v33 = ssdt + (offset >> 4);  
    }
```

```
0: kd> dps nt!KeServiceDescriptorTable L4  
fffff807`6ea0f8c0 fffff807`6dcd6340 nt!KiServiceTable  
fffff807`6ea0f8c8 00000000`00000000  
fffff807`6ea0f8d0 00000000`000001d7  
fffff807`6ea0f8d8 fffff807`6dcd6aa0 nt!KiArgumentTable  
0: kd> dps nt!KeServiceDescriptorTableShadow  
fffff807`6e909a40 fffff807`6dcd6340 nt!KiServiceTable  
fffff807`6e909a48 00000000`00000000  
fffff807`6e909a50 00000000`000001d7  
fffff807`6e909a58 fffff807`6dcd6aa0 nt!KiArgumentTable  
fffff807`6e909a60 fffff80a`b1b18000 win32k!W32pServiceTable  
fffff807`6e909a68 00000000`00000000  
fffff807`6e909a70 00000000`00000524  
fffff807`6e909a78 fffff80a`b1b199bc win32k!W32pArgumentTable
```

SSDT-перехваты



Получение адреса таблицы SSDT Shadow

```
248     {
249         addr_t w32k_base = 0;
250         vmi_lock_guard vmi(drakvuf);
251         // Locate Win32k.sys base address
252         if (!get_driver_base(vmi, this, "win32k.sys", &w32k_base))
253         {
254             PRINT_DEBUG("[SSDTMON] Failed to find win32k.sys in PsLoadedModuleList\n");
255             throw -1;
256         }
257         // Read ssdt shadow size
258         if (VMI_SUCCESS != vmi_read_32_va(vmi, w32k_base + w32psl_rva, gui_pid, &this->w32pservicelimit))
259         {
260             PRINT_DEBUG("[SSDTMON] Failed to read W32pServiceLimit\n");
261             throw -1;
262         }
263         // NOTE: We use vmi_translate_uv2p instead of vmi_translate_kv2p because Win32k.sys mapping is not present
264         // in system process, only process with GUI dependencies, hence we use explorer.exe as a pid.
265         if (VMI_SUCCESS != vmi_translate_uv2p(vmi, w32k_base + w32pst_rva, gui_pid, &this->w32pservicetable))
266         {
267             PRINT_DEBUG("[SSDTMON] Failed to translate win32k!W32pServiceTable to physical address\n");
268             throw -1;
269         }
270     }
```


SSDT-перехваты



Проверка записи в SSDT и SSDT Shadow

```
event_response_t write_cb(drakvuf_t drakvuf, drakvuf_trap_info_t* info)
{
    ssdtmon* s = (ssdtmon*)info->trap->data;

    if ( info->trap_pa > s->kiservicetable - 8 && info->trap_pa <= s->kiservicetable +
        sizeof(uint32_t) * s->kiservicelimit + sizeof(uint32_t) - 1 )
    {
        int64_t table_index = (info->trap_pa - s->kiservicetable) / sizeof(uint32_t);
        fmt::print(s->format, "ssdtmon", drakvuf, info,
            keyval("TableIndex", fmt::Nval(table_index)),
            keyval("Table", fmt::Qstr("SSDT"))
        );
    }
    else if (info->trap_pa > s->w32pservicetable - 8 && info->trap_pa <= s->w32pservicetable +
        sizeof(uint32_t) * s->w32pservicelimit + sizeof(uint32_t) - 1 )
    {
        int64_t table_index = (info->trap_pa - s->w32pservicetable) / sizeof(uint32_t);
        fmt::print(s->format, "ssdtmon", drakvuf, info,
            keyval("TableIndex", fmt::Nval(table_index)),
            keyval("Table", fmt::Qstr("SSDTShadow"))
        );
    }
    return 0;
}
```

```
TABLER0: FFFFFFFF97FFF078000 public W32pServiceTable
TABLER0: FFFFFFFF97FFF078000 W32pServiceTable dd 4834h
TABLER0: FFFFFFFF97FFF078000
TABLER0: FFFFFFFF97FFF078004 dd 7184h
TABLER0: FFFFFFFF97FFF078008 dd 475Ch
TABLER0: FFFFFFFF97FFF07800C dd 91D0h
TABLER0: FFFFFFFF97FFF078010 dd 6B20h
TABLER0: FFFFFFFF97FFF078014 dd 4738h
TABLER0: FFFFFFFF97FFF078018 dd 6260h
TABLER0: FFFFFFFF97FFF07801C dd 6E78h
```

```
.rdata: 00000001400C8348 align 10h
.rdata: 00000001400C8350 KiServiceTable dd 339F40h
.rdata: 00000001400C8350
.rdata: 00000001400C8354 dd 345430h
.rdata: 00000001400C8358 dd 7067C0h
.rdata: 00000001400C835C dd 8D3F10h
.rdata: 00000001400C8360 dd 5F7590h
.rdata: 00000001400C8364 dd 3FAB10h
.rdata: 00000001400C8368 dd 68BC70h
.rdata: 00000001400C836C dd 674730h
.rdata: 00000001400C8370 dd 68B040h
.rdata: 00000001400C8374 dd 6D68C0h
```

Трасса выполнения



ProcessName = users\\sample.exe, TID = 1934, PID = 832, PPID = 2031,
plugin = ssdtmon, TableIndex = 82, Table = SSDT

ProcessName = users\\sample.exe, TID = 1902, PID = 158, PPID = 217,
plugin = ssdtmon, TableIndex = 172, Table = SSDTShadow

Модификация IDT/GDT



IDT – Interrupt Descriptor Table

```
0: kd> dt nt!_KIDTENTRY64
+0x000 OffsetLow      : Uint2B
+0x002 Selector       : Uint2B
+0x004 IstIndex       : Pos 0, 3 Bits
+0x004 Reserved0      : Pos 3, 5 Bits
+0x004 Type           : Pos 8, 5 Bits
+0x004 Dpl            : Pos 13, 2 Bits
+0x004 Present        : Pos 15, 1 Bit
+0x006 OffsetMiddle   : Uint2B
+0x008 OffsetHigh     : Uint4B
+0x00c Reserved1      : Uint4B
+0x000 Alignment      : Uint8B
```

```
0: kd> r idtr
idtr=fffff8010b461000
0: kd> !idt

Dumping IDT: fffff8010b461000

00: fffff80108a1fc00 nt!KiDivideErrorFault
01: fffff80108a1ff40 nt!KiDebugTrapOrFault Stack = 0xFFFFF8010B49F000
02: fffff80108a20440 nt!KiNmiInterrupt Stack = 0xFFFFF8010B491000
03: fffff80108a20900 nt!KiBreakpointTrap
04: fffff80108a20c40 nt!KiOverflowTrap
05: fffff80108a20f80 nt!KiBoundFault
06: fffff80108a214c0 nt!KiInvalidOpcodeFault
07: fffff80108a219c0 nt!KiNpxNotAvailableFault
08: fffff80108a21cc0 nt!KiDoubleFaultAbort Stack = 0xFFFFF8010B48A000
09: fffff80108a21fc0 nt!KiNpxSegmentOverrunAbort
0a: fffff80108a222c0 nt!KiInvalidTssFault
0b: fffff80108a225c0 nt!KiSegmentNotPresentFault
0c: fffff80108a22980 nt!KiStackFault
0d: fffff80108a22cc0 nt!KiGeneralProtectionFault
0e: fffff80108a23000 nt!KiPageFault
10: fffff80108a23640 nt!KiFloatingErrorFault
```

Модификация IDT/GDT



GDT – Global Descriptor Table

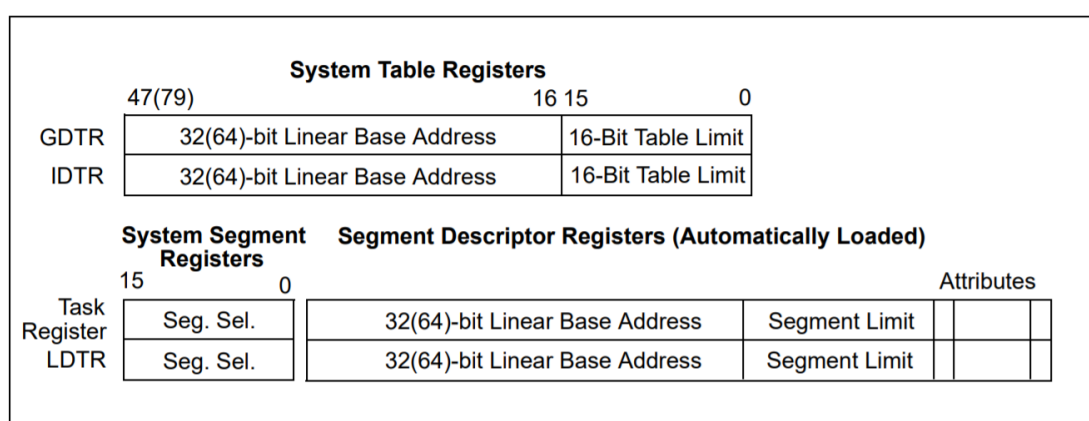
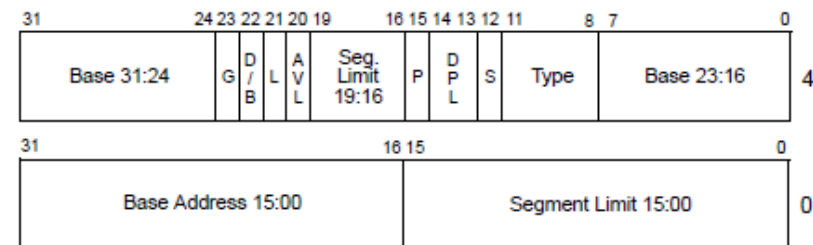


Figure 2-6. Memory Management Registers



- L — 64-bit code segment (IA-32e mode only)
- AVL — Available for use by system software
- BASE — Segment base address
- D/B — Default operation size (0 = 16-bit segment; 1 = 32-bit segment)
- DPL — Descriptor privilege level
- G — Granularity
- LIMIT — Segment Limit
- P — Segment present
- S — Descriptor type (0 = system; 1 = code or data)
- TYPE — Segment type

Figure 3-8. Segment Descriptor

Модификация IDT/GDT



Перечисление таблиц на каждом ядре

```
305 bool rootkitmon::enumerate_cores(vmi_instance_t vmi)
306 {
307     for (size_t vcpu = 0; vcpu < vmi_get_num_vcpus(vmi); vcpu++)
308     {
309         uint64_t idtr_base = 0, gdtr_base = 0;
310         uint64_t idtr_limit = 0, gdtr_limit = 0;
311         uint64_t lstar = 0;
312
313         if (VMI_SUCCESS == vmi_get_vcpureg(vmi, &idtr_base, IDTR_BASE, vcpu) &&
314             VMI_SUCCESS == vmi_get_vcpureg(vmi, &idtr_limit, IDTR_LIMIT, vcpu) &&
315             VMI_SUCCESS == vmi_get_vcpureg(vmi, &gdtr_base, GDTR_BASE, vcpu) &&
316             VMI_SUCCESS == vmi_get_vcpureg(vmi, &gdtr_limit, GDTR_LIMIT, vcpu) &&
317             VMI_SUCCESS == vmi_get_vcpureg(vmi, &lstar, MSR_LSTAR, vcpu))
318         {
319             PRINT_DEBUG("[ROOTKITMON] [VCPU] %zu IDTR 0x%lx:0x%lx\n", vcpu, idtr_base, idtr_limit);
320             PRINT_DEBUG("[ROOTKITMON] [VCPU] %zu GDTR 0x%lx:0x%lx\n", vcpu, gdtr_base, gdtr_limit);
321             PRINT_DEBUG("[ROOTKITMON] [VCPU] %zu LSTAR 0x%lx\n", vcpu, lstar);
322
323             // Calculate IDT checksum
324             auto idt_checksum = calc_checksum(vmi, idtr_base, idtr_limit + 1);
325
326             // Enumerate GDT entries
327             auto gdt = enumerate_gdt(vmi, gdtr_base, gdtr_limit);
```

Модификация IDT/GDT



Сверка значений в конце анализа

```
543     for (const auto& [vcpu, desc_info] : this->descriptors)
544     {
545         const auto& t_desc_info = past_descriptors[vcpu];
546         if (desc_info.idtr_base != t_desc_info.idtr_base)
547         {
548             fmt::print(this->format, "rootkitmon", drakvuf, info,
549                 keyval("Reason", fmt::Qstr("IDTR base modification")));
550             break;
551         }
552         if (desc_info.idt_checksum != t_desc_info.idt_checksum)
553         {
554             fmt::print(this->format, "rootkitmon", drakvuf, info,
555                 keyval("Reason", fmt::Qstr("IDT modification")));
556             break;
557         }
558     }
```

"\Windows\System32\explorer.exe":KiDeliverApc SessionID:0 PID:1968 PPID:476 Reason:"**IDT modification**"

"\Windows\System32\SearchIndexer.exe":KiDeliverApc SessionID:0 PID:1968 PPID:476 Reason:"**GDT modification**"

Перехват MSR LSTAR



Содержит адрес следующей инструкции после системного вызова

```
1: kd> rdmsr 0xc0000082  
msr[c0000082] = fffff804`5881f6c0  
1: kd> u fffff804`5881f6c0 L2  
nt!KiSystemCall64:  
fffff804`5881f6c0 0f01f8          swapgs  
fffff804`5881f6c3 654889242510000000 mov     qword ptr gs:[10h],rsp
```

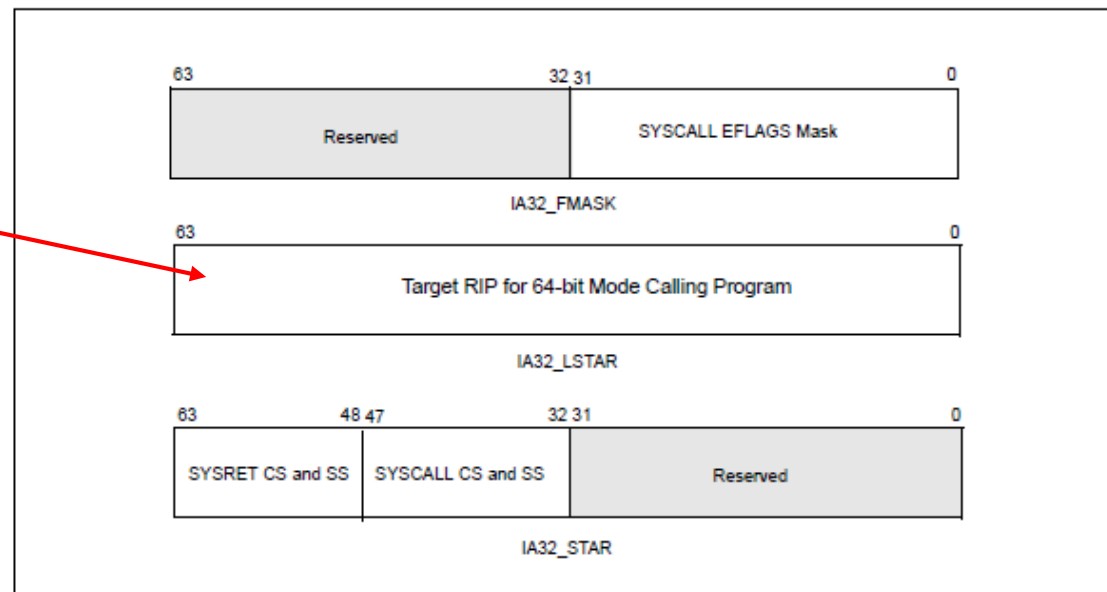


Figure 5-14. MSRs Used by SYSCALL and SYSRET

Перехват MSR LSTAR



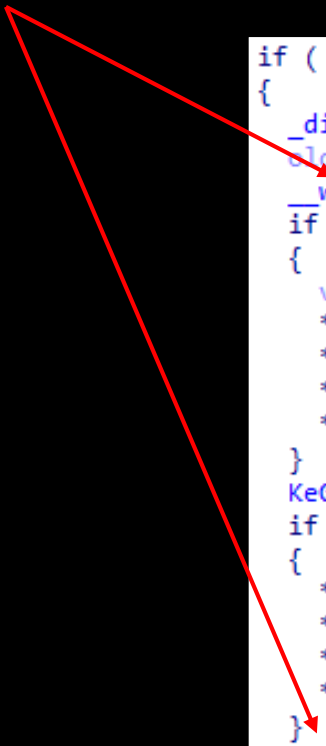
Инициализируется
на ранних этапах
загрузки ОС

```
v15 = KiSystemCall32;  
v16 = KiSystemCall64;  
if ( !v14 )  
{  
    KiEnableKvaShadowing(CurrentPrcb, &Pcr->NtTib.ExceptionList[-763]);  
    v14 = CurrentPrcb->Number;  
}  
if ( KiKvaShadow )  
{  
    v15 = KiSystemCall32Shadow;  
    v16 = KiSystemCall64Shadow;  
}  
if ( !v14 )  
    KiTsxSupportedAtBoot = KiDetectTsx();  
if ( CurrentPrcb->MsrIa32TsxCtrl )  
    __writemsr(0x122u, CurrentPrcb->MsrIa32TsxCtrl);  
if ( !CurrentPrcb->Number )  
{  
    v24 = KiDebugTrapIndex;  
    qword_140C01650 = KiDebugTraps;  
    qword_140C01658 = 64i64;  
    KiDebugTraps[KiDebugTrapIndex] = v16;  
    KiDebugTraps[++v24] = v15;  
    KiDebugTrapIndex = v24 + 1;  
}  
__writemsr(0xC0000081, 0x23001000000000ui64);  
__writemsr(0xC0000083, v15);  
__writemsr(0xC0000082, v16);  
__writemsr(0xC0000084, 0x4700ui64);
```


Перехват MSR LSTAR



Перехват MSR LSTAR PatchGuard'ом

Two red arrows originate from the left side of the slide. One arrow points to the line `__writemsr(0xC0000082, context + 2170);` and the other points to the line `__writemsr(0xC0000082, oldMsrValue);` in the code block.

```
if ( (*(context + 2171) & 1) != 0 )
{
    __disable();                // Disable interrupts
    oldMsrValue = __readmsr(0xC0000082);
    __writemsr(0xC0000082, context + 2170); // Overwrite MSR LSTAR
    if ( (*(context + 613) & 0x20000) == 0 )
    {
        v1499 = KeGetCurrentPrCb();
        **(context + 152) = context - 0x5C5FC0A76E374B18i64;
        **(context + 153) = v1499;
        **(context + 154) = 3221225602i64;
        **(context + 155) = 274i64;
    }
    KeGuardDispatchICall(context + 2172);
    if ( (*(context + 613) & 0x20000) == 0 )
    {
        **(context + 152) = 0xA3A03F5891C8B4E8ui64;
        **(context + 153) = 0i64;
        **(context + 154) = 0i64;
        **(context + 155) = 0i64;
    }
    __writemsr(0xC0000082, oldMsrValue); // Restore original msr value
    __enable();                          // Enable interrupts
}
```

Перехват MSR LSTAR



Проверка целостности регистра в конце анализа

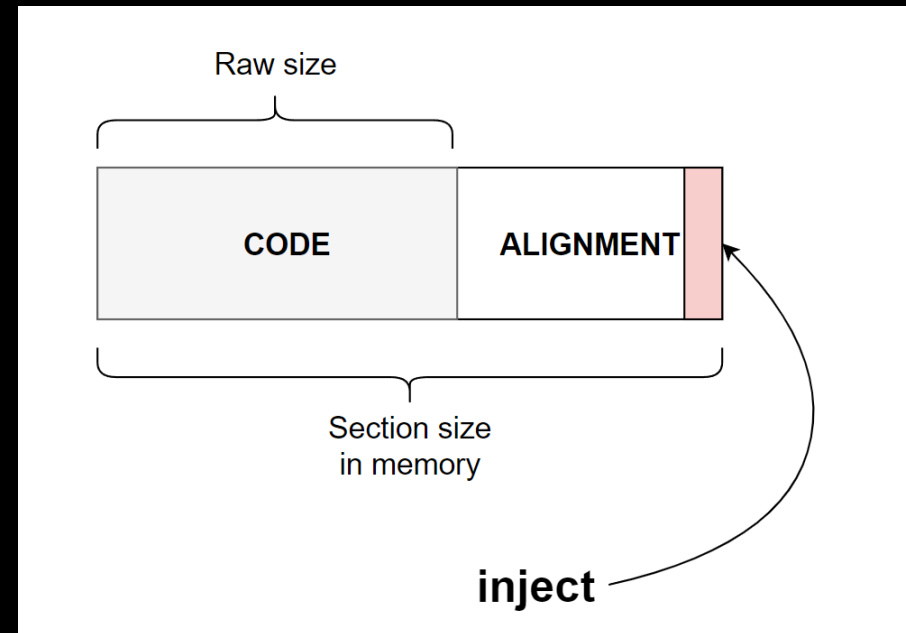
```
591      //=====
592      // Check msr lstar modification
593      for (const auto& [vcpu, lstar] : this->msr_lstar)
594      {
595          if (past_lstar[vcpu] != lstar)
596          {
597              fmt::print(this->format, "rootkitmon", drakvuf, info,
598                          keyval("Reason", fmt::Qstr("LSTAR modification")));
599          }
600      }
601
```

ProcessName = users\\sample.exe, TID = 242, PID = 1107, PPID = 534,
plugin = **rootkitmon**, Reason = **LSTAR modification**

Модификация кода драйвера



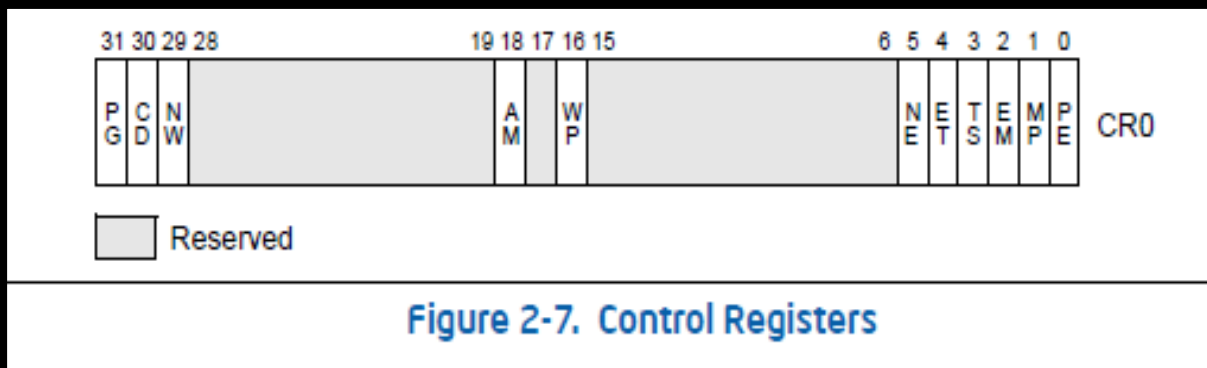
- Перехват отдельных функций кода драйвера
- Запись в пустое пространство секций драйвера



Модификация кода драйвера



Write Protect (16 bit), отключение прав доступа памяти



- Data writes to supervisor-mode addresses.
Access rights depend on the value of CR0.WP:
 - If CR0.WP = 0, data may be written to any supervisor-mode address.
 - If CR0.WP = 1, data may be written to any supervisor-mode address with a translation for which the R/W flag (bit 1) is 1 in every paging-structure entry controlling the translation; data may not be written to any supervisor-mode address with a translation for which the R/W flag is 0 in any paging-structure entry controlling the translation.

Модификация кода драйвера



Rootkit Demodex, patch-секции кода драйвера

```
while ( ADJ(section_header)->Misc.VirtualSize - ADJ(section_header)->SizeOfRawData < 0xE
        || (ADJ(section_header)->Characteristics & 0x28000000) != 0x28000000 )
{
    // IMAGE_SCN_MEM_EXECUTE | IMAGE_SCN_MEM_NOT_PAGED
    //

    ++section_count;
    section_header += 10;
    if ( section_count >= *(nt_headers_offset + pci_sys_image.p_pci_sys_image_buffer + 6) )
        goto end;
}
address_to_patch = c_pci_sys_image_base + ADJ(section_header)->SizeOfRawData + ADJ(section_header)->VirtualAddress;
if ( address_to_patch )
{
    cr0 = __readcr0();
    __writecr0(cr0 & 0xFFFFFFFFFFFFFFFFui64);
    *address_to_patch = 0x68; // push    LODWORD(address)
    *(address_to_patch + 1) = fn_callback_address;
    *(address_to_patch + 5) = 0x42444C7; // mov     dword ptr [rsp+0x04], HIWORD(address)
    *(address_to_patch + 9) = HIWORD(fn_callback_address);
    address_to_patch[13] = 0xC3; // ret
    __writecr0(cr0);
}
```

<https://securelist.com/ghostemperor-from-proxylogon-to-kernel-mode/104407/>

Модификация кода драйвера



```
imageBase = DrvObj->DriverStart;
if ( imageBase )
{
    if ( imageBase->e_magic == 0x5A4D
        && (ntHeader = (imageBase + imageBase->e_lfanew), MmIsAddressValid(ntHeader))
        && (sectionHeader = (&ntHeader->OptionalHeader + ntHeader->FileHeader.SizeOfOptionalHeader),
            MmIsAddressValid_1(sectionHeader))
        && (numOfSections = ntHeader->FileHeader.NumberOfSections, v13 = 0, numOfSections) )
    {
        while ( strcmp(sectionHeader, ".text") || (sectionHeader->SizeOfRawData - sectionHeader->Mi
            {
                ++v13;
                ++sectionHeader;
                if ( v13 >= numOfSections )
                    goto LABEL_16;
            }
        numOfFreeBytes = sectionHeader->SizeOfRawData - sectionHeader->Misc.VirtualSize;
        p_textSectionFreeSpace = imageBase + sectionHeader->VirtualAddress + sectionHeader->Misc.Vir
    }
}
```

Внедрение кода в пустые
области секции .text

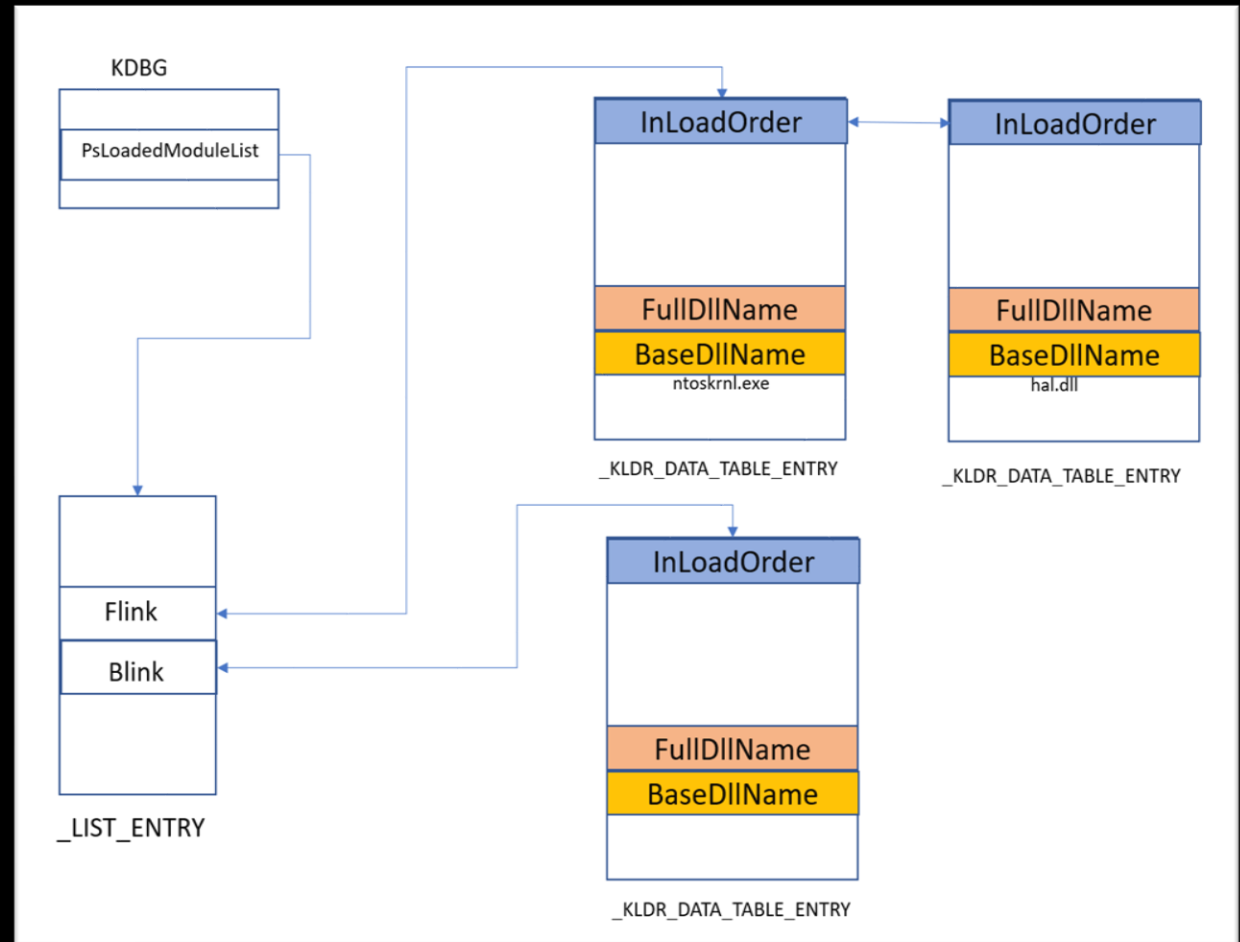
```
// PVOID __fastcall alloc_n_map_md1(*VirtualAddress, Length, AccessMode, LockOperation, **ppMdl)
mappedAddress = alloc_n_map_md1(p_textSectionFreeSpace, 12u, 0, IoReadAccess, &Mdl);
if ( mappedAddress )
{
    *mappedAddress = 0x48; // mov rax, address
    mappedAddress[1] = 0xB8;
    *(mappedAddress + 2) = jmpAddress;
    mappedAddress[10] = 0xFF;
    mappedAddress[11] = 0xE0; // jmp rax
}
if ( Mdl )
    IoFreeMdl(Mdl);
```

Модификация кода драйвера



Перечисление загруженных драйверов

```
0: kd> dt nt!_KLDR_DATA_TABLE_ENTRY
+0x000 InLoadOrderLinks : _LIST_ENTRY
+0x010 ExceptionTable   : Ptr64 Void
+0x018 ExceptionTableSize : Uint4B
+0x020 GpValue          : Ptr64 Void
+0x028 NonPagedDebugInfo : Ptr64 _NON_PAGED_DEBUG_INFO
+0x030 DllBase           : Ptr64 Void
+0x038 EntryPoint        : Ptr64 Void
+0x040 SizeOfImage       : Uint4B
+0x048 FullDllName        : _UNICODE_STRING
+0x058 BaseDllName        : _UNICODE_STRING
+0x068 Flags             : Uint4B
+0x06c LoadCount         : Uint2B
+0x06e u1                : <anonymous-tag>
+0x070 SectionPointer    : Ptr64 Void
+0x078 CheckSum          : Uint4B
+0x07c CoverageSectionSize : Uint4B
+0x080 CoverageSection   : Ptr64 Void
+0x088 LoadedImports      : Ptr64 Void
+0x090 Spare             : Ptr64 Void
+0x098 SizeOfImageNotRounded : Uint4B
+0x09c TimeDateStamp     : Uint4B
```



Модификация кода драйвера



Подсчет контрольных сумм
секций драйверов

```
901
902     drakvuf_enumerate_drivers(drakvuf, driver_visitor, static_cast<void*>(this));
903

351  /**
352   * This callback is executed on every discovered driver.
353   * It is used to calculate checksums of driver sections.
354   * @driver - LDR_DATA_TABLE_ENTRY pointer.
355   */
356  static void driver_visitor(drakvuf_t drakvuf, addr_t driver, void* ctx)
357  {

382      // Checksum every section and save it into `driver_sections_checksums`
383      for (const auto& [virt_addr, virt_size] : get_pe_code_sections(module, imagebase))
384      {
385          auto aligned_size = align_by_page(virt_size);
386
387          checksum_data_t data =
388          {
389              .virtual_address = virt_addr,
390              .virtual_size = aligned_size,
391              .checksum = calc_checksum(vmi, virt_addr, aligned_size)
392          };
393          plugin->driver_sections_checksums[driver].push_back(data);
394
395      }
```


Модификация кода драйвера



Перечисление секций драйвера

```
153  /**
154   * Enumerate PE sections with mem_execute and mem_not_paged flags.
155   * Returns vector of <virtual address, aligned section size>
156   */
157  static std::vector<std::pair<addr_t, size_t>> get_pe_code_sections(void* module, addr_t read_imagebase)
158  {
159      std::vector<std::pair<addr_t, size_t>> out;
160
161      auto dos_h = static_cast<dos_header_t*>(module);
162      for (size_t i = 0; i < dos_h->get_nt_headers()->file_header.num_sections; i++)
163      {
164          auto section = dos_h->get_nt_headers()->get_section(i);
165          // Some sections (PAGE, INIT) might not be present in memory, that's why
166          // we process only non pagable sections.
167          // It is important to check for write access as well since there are sections
168          // with RWX access rights on win7 and below. e.g. RWEXEC in hal.dll and ntoskrnl.
169          if (section->characteristics & mem_execute
170              && section->characteristics & mem_not_paged
171              && !(section->characteristics & mem_write))
172          {
173              auto aligned_size = align_by_page(section->virtual_size);
174              out.emplace_back(section->virtual_address + read_imagebase, aligned_size);
175          }
176      }
177      return out;
178  }
```

Модификация кода драйвера



Проверка контрольной суммы секции

```
if (checksum_data.checksum != p_checksum_data.checksum)
{
    {
        vmi_lock_guard vmi(drakvuf);
        unicode_string_t* drvname = drakvuf_read_unicode_va(vmi, driver + this->offsets[LDR_DATA_TABLE_ENTRY_BASEDLLNAME], 4);
        if (drvname)
        {
            fmt::print(this->format, "rootkitmon", drakvuf, info,
                keyval("Reason", fmt::Qstr("Driver section modification")),
                keyval("Driver", fmt::Qstr((const char*)drvname->contents)));
            vmi_free_unicode_str(drvname);
        }
    }
}
```

"\Windows\System32\explorer.exe":KiDeliverApc SessionID:0 PID:1968 PPID:476
Reason:"Driver section modification" Driver:"ntoskrnl.exe"

Модификация DRIVER_OBJECT



- DRIVER_OBJECT описывает загруженный драйвер в системе
- Создается и передается первым параметром в DriverEntry IO диспетчером Windows

```
1: kd> dt nt!_DRIVER_OBJECT
+0x000 Type           : Int2B
+0x002 Size           : Int2B
+0x008 DeviceObject   : Ptr64 _DEVICE_OBJECT
+0x010 Flags          : Uint4B
+0x018 DriverStart    : Ptr64 Void
+0x020 DriverSize     : Uint4B
+0x028 DriverSection  : Ptr64 Void
+0x030 DriverExtension : Ptr64 _DRIVER_EXTENSION
+0x038 DriverName     : _UNICODE_STRING
+0x048 HardwareDatabase : Ptr64 _UNICODE_STRING
+0x050 FastIoDispatch : Ptr64 _FAST_IO_DISPATCH
+0x058 DriverInit     : Ptr64 long
+0x060 DriverStartIo  : Ptr64 void
+0x068 DriverUnload   : Ptr64 void
+0x070 MajorFunction  : [28] Ptr64 long
```

Модификация DRIVER_OBJECT



- MajorFunction – массив из 28 обработчиков различных IRP запросов:

ReadFile – IRP_MJ_READ

CreateFile – IRP_MJ_CREATE

CloseFile – IRP_MJ_CLOSE

И.Т.Д.

```
Dispatch routines:
[00] IRP_MJ_CREATE                fffff8022fd8efc0    tcpip!NlDispatchCleanup
[01] IRP_MJ_CREATE_NAMED_PIPE    fffff8022db5ab80    nt!IopInvalidDeviceRequest
[02] IRP_MJ_CLOSE                fffff8022fd8efc0    tcpip!NlDispatchCleanup
[03] IRP_MJ_READ                 fffff8022db5ab80    nt!IopInvalidDeviceRequest
[04] IRP_MJ_WRITE                fffff8022db5ab80    nt!IopInvalidDeviceRequest
[05] IRP_MJ_QUERY_INFORMATION    fffff8022db5ab80    nt!IopInvalidDeviceRequest
[06] IRP_MJ_SET_INFORMATION      fffff8022db5ab80    nt!IopInvalidDeviceRequest
[07] IRP_MJ_QUERY_EA            fffff8022db5ab80    nt!IopInvalidDeviceRequest
[08] IRP_MJ_SET_EA              fffff8022db5ab80    nt!IopInvalidDeviceRequest
[09] IRP_MJ_FLUSH_BUFFERS       fffff8022db5ab80    nt!IopInvalidDeviceRequest
[0a] IRP_MJ_QUERY_VOLUME_INFORMATION fffff8022db5ab80    nt!IopInvalidDeviceRequest
[0b] IRP_MJ_SET_VOLUME_INFORMATION fffff8022db5ab80    nt!IopInvalidDeviceRequest
[0c] IRP_MJ_DIRECTORY_CONTROL   fffff8022db5ab80    nt!IopInvalidDeviceRequest
[0d] IRP_MJ_FILE_SYSTEM_CONTROL fffff8022db5ab80    nt!IopInvalidDeviceRequest
```

```
DeviceObject->Flags |= 0x10u;
v7 = GetIrpStackSize(RegistryPath);
DeviceObject->StackSize = v7;
WORD1(v6[1].DriverObject) = RegistryPath->Length;
v6[1].NextDevice = (v6 + 368);
RtlCopyUnicodeString(&v6[1].DriverObject, RegistryPath);
DriverObject->MajorFunction[IRP_MJ_CREATE] = Dispatch;
DriverObject->MajorFunction[IRP_MJ_CLOSE] = Dispatch;
DriverObject->MajorFunction[IRP_MJ_CLEANUP] = Dispatch;
DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = Dispatch;
DriverObject->DriverUnload = Unload;
g_RegistryPath.Buffer = ExAllocatePoolWithTag(512, 2i64 * RegistryPath->Length, 0x00000017);
if ( !g_RegistryPath.Buffer )
    return 0xC0000017;
```

Модификация DRIVER_OBJECT



Подмена major-функций
IRP_MJ_CREATE и
IRP_MJ_READ

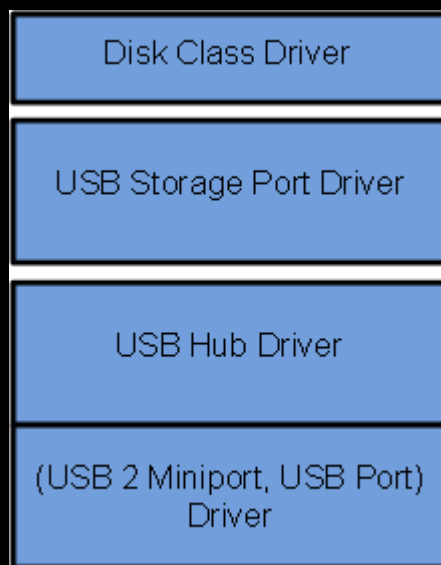
```
p_fsHook = inject_jump_code(L"\\FileSystem\\FltMgr", p_hookFunction, v3);  
if ( !p_fsHook )  
    goto LABEL_15;  
SourceString[13] = ':';  
*SourceString = 'D\\0\\';  
*&SourceString[2] = 's\\0o';  
*&SourceString[4] = 'e\\0D';  
*&SourceString[6] = 'i\\0v';
```

```
if ( v6 < 0 || (v6 = ObReferenceObjectByHandle_1(FileHandle, 1u, 0i64, 0, &Handle,  
{  
    v2 = v6;  
}  
else  
{  
    v7 = IoGetRelatedDeviceObject_64(Handle);  
    if ( v7 )  
    {  
        if ( IRP_MJ_XXX )  
            original_FltMgr_MJ_READ = v7->DriverObject->MajorFunction[IRP_MJ_READ];  
        else  
            original_FltMgr_MJ_CREATE = v7->DriverObject->MajorFunction[IRP_MJ_CREATE];  
        v7->DriverObject->MajorFunction[IRP_MJ_XXX] = p_fsHook;  
    }  
    else
```

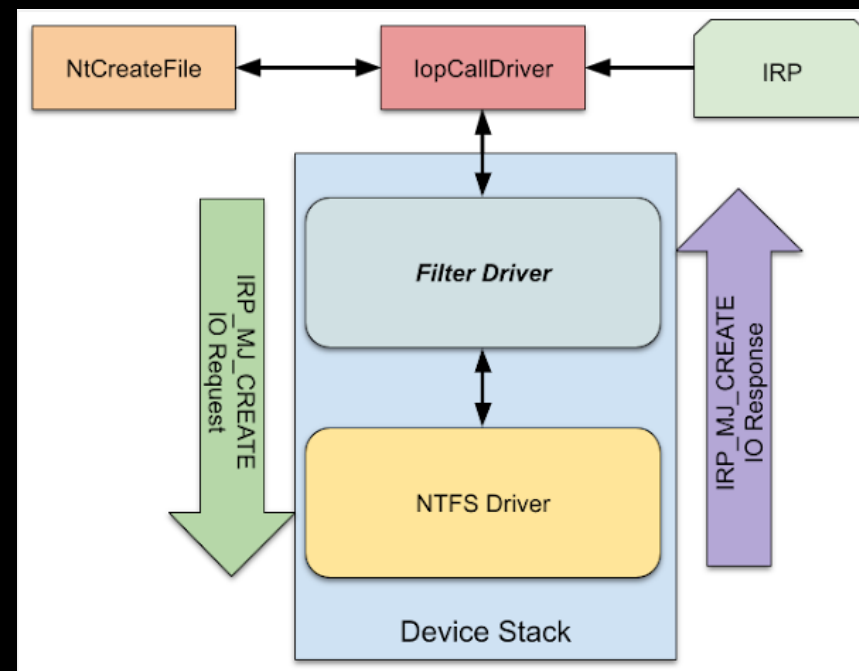
Модификация DRIVER_OBJECT



- Модель IRP-запросов и стека драйверов



<https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/driver-stacks>



<https://googleprojectzero.blogspot.com/2021/01/hunting-for-bugs-in-windows-mini-filter.html>

Модификация DRIVER_OBJECT



- NextDevice – указатель на следующий DEVICE_OBJECT в списке
- AttachedDevice – список подключенных девайсов

```
1: kd> dt nt!_DEVICE_OBJECT
+0x000 Type           : Int2B
+0x002 Size           : Uint2B
+0x004 ReferenceCount  : Int4B
+0x008 DriverObject    : Ptr64 _DRIVER_OBJECT
+0x010 NextDevice      : Ptr64 DEVICE_OBJECT
+0x018 AttachedDevice  : Ptr64 DEVICE_OBJECT
+0x020 CurrentIrp      : Ptr64 _IRP
+0x028 Timer           : Ptr64 _IO_TIMER
+0x030 Flags           : Uint4B
+0x034 Characteristics : Uint4B
+0x038 Vpb             : Ptr64 _VPB
+0x040 DeviceExtension : Ptr64 Void
+0x048 DeviceType      : Uint4B
+0x04c StackSize       : Char
+0x050 Queue           : <anonymous-tag>
+0x098 AlignmentRequirement : Uint4B
+0x0a0 DeviceQueue     : _KDEVICE_QUEUE
+0x0c8 Dpc              : _KDPC
+0x108 ActiveThreadCount : Uint4B
+0x110 SecurityDescriptor : Ptr64 Void
+0x118 DeviceLock       : _KEVENT
+0x130 SectorSize      : Uint2B
+0x132 Spare1          : Uint2B
+0x138 DeviceObjectExtension : Ptr64 _DEVOBJ_EXTENSION
+0x140 Reserved        : Ptr64 Void
```

Модификация DRIVER_OBJECT



OBJECT_HEADER

Специальный заголовок,
который находится перед телом
объекта и которым владеет
диспетчер объектов

- TypeIndex – индекс в массиве nt!ObTypeIndexTable
- Body – тело объекта

```
0: kd> dt _OBJECT_HEADER
nt!_OBJECT_HEADER
+0x000 PointerCount      : Int8B
+0x008 HandleCount      : Int8B
+0x008 NextToFree       : Ptr64 Void
+0x010 Lock              : _EX_PUSH_LOCK
+0x018 TypeIndex        : UChar
+0x019 TraceFlags       : UChar
+0x019 DbgRefTrace       : Pos 0, 1 Bit
+0x019 DbgTracePermanent : Pos 1, 1 Bit
+0x01a InfoMask         : UChar
+0x01b Flags            : UChar
+0x01b NewObject         : Pos 0, 1 Bit
+0x01b KernelObject     : Pos 1, 1 Bit
+0x01b KernelOnlyAccess : Pos 2, 1 Bit
+0x01b ExclusiveObject  : Pos 3, 1 Bit
+0x01b PermanentObject  : Pos 4, 1 Bit
+0x01b DefaultSecurityQuota : Pos 5, 1 Bit
+0x01b SingleHandleEntry : Pos 6, 1 Bit
+0x01b DeletedInline     : Pos 7, 1 Bit
+0x01c Reserved         : Uint4B
+0x020 ObjectCreateInfo : Ptr64 _OBJECT_CREATE_INFORMATION
+0x020 QuotaBlockCharged : Ptr64 Void
+0x028 SecurityDescriptor : Ptr64 Void
+0x030 Body             : _QUAD
```


Модификация DRIVER_OBJECT



Пример получения имени объекта из структуры его типа

```
0: kd> dps nt!ObTypeIndexTable L8
fffff801`09319e80 00000000`00000000
fffff801`09319e88 fffffd001`3858a000
fffff801`09319e90 fffffbb87`6c68c0c0
fffff801`09319e98 fffffbb87`6c68c4e0
fffff801`09319ea0 fffffbb87`6c68c7a0
fffff801`09319ea8 fffffbb87`6c68cbc0
fffff801`09319eb0 fffffbb87`6c68cd20
fffff801`09319eb8 fffffbb87`6c68ce80
```

```
0: kd> dt nt!_OBJECT_TYPE fffffbb87`6c68ce80
+0x000 TypeList : _LIST_ENTRY [ 0xfffffbb87`6c68ce80 -
+0x010 Name : _UNICODE_STRING "Process"
+0x020 DefaultObject : (null)
+0x028 Index : 0x7 ''
+0x02c TotalNumberOfObjects : 0x30
+0x030 TotalNumberOfHandles : 0x29d
+0x034 HighWaterNumberOfObjects : 0x53
+0x038 HighWaterNumberOfHandles : 0x3a6
+0x040 TypeInfo : _OBJECT_TYPE_INITIALIZER
+0x0b8 TypeLock : _EX_PUSH_LOCK
+0x0c0 Key : 0x636f7250
+0x0c8 CallbackList : _LIST_ENTRY [ 0xfffff8f85`69cab080 -
```

Модификация DRIVER_OBJECT



- Директория, содержащая 37 элементов
- Каждый элемент имеет тип `_OBJECT_DIRECTORY_ENTRY`
- Может быть вложенной
- Указатель на корневую директорию содержится в `ObpRootDirectoryObject`

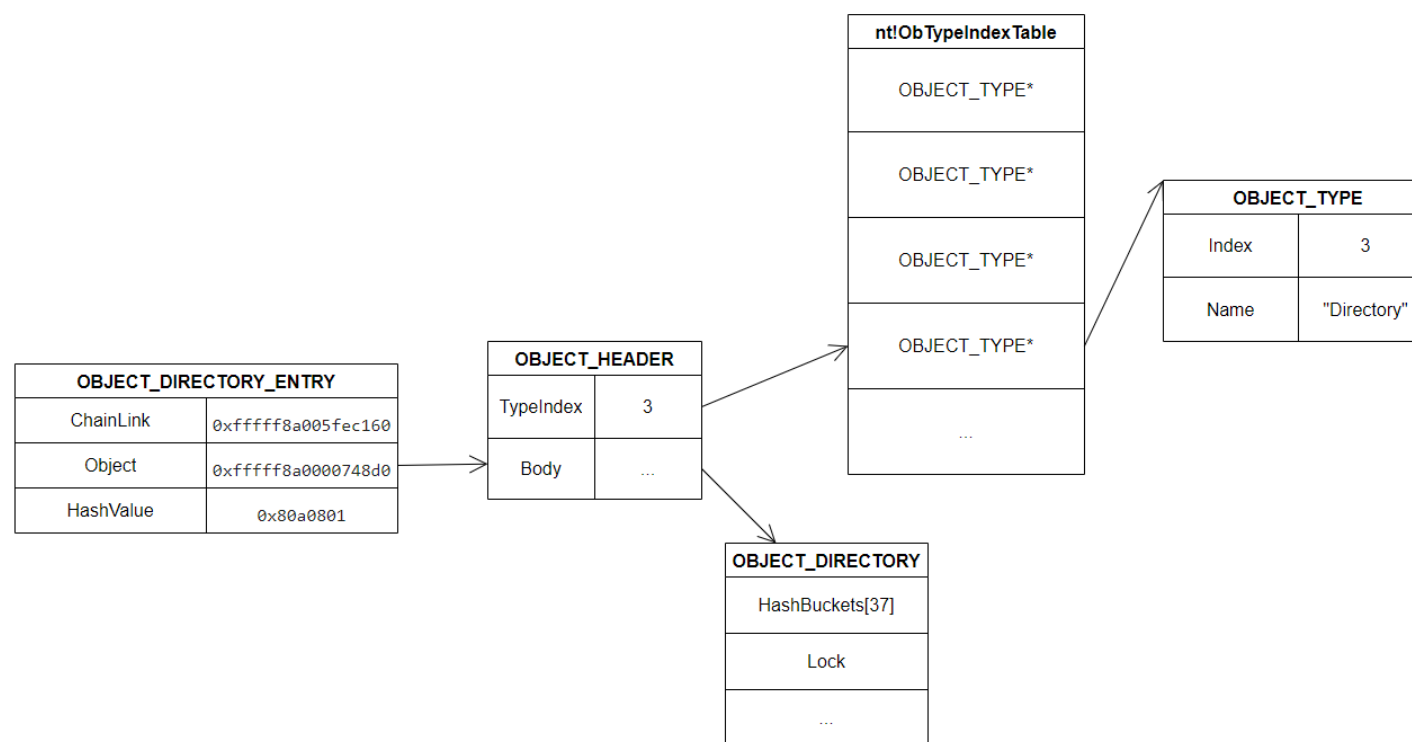
```
0: kd> dt nt!_OBJECT_DIRECTORY
+0x000 HashBuckets      : [37] Ptr64 _OBJECT_DIRECTORY_ENTRY
+0x128 Lock             : _EX_PUSH_LOCK
+0x130 DeviceMap        : Ptr64 _DEVICE_MAP
+0x138 ShadowDirectory  : Ptr64 _OBJECT_DIRECTORY
+0x140 NamespaceEntry   : Ptr64 Void
+0x148 SessionObject    : Ptr64 Void
+0x150 Flags            : Uint4B
+0x154 SessionId        : Uint4B
```

```
0: kd> dt nt!_OBJECT_DIRECTORY_ENTRY
+0x000 ChainLink        : Ptr64 _OBJECT_DIRECTORY_ENTRY
+0x008 Object           : Ptr64 Void
+0x010 HashValue        : Uint4B
```

Модификация DRIVER_OBJECT



Модель вложенной директории диспетчера объектов



Модификация DRIVER_OBJECT



Перечисление объектов драйверов

```
735 std::set<driver_t> rootkitmon::enumerate_directory(vmi_instance_t vmi, addr_t directory)
736 {
737     std::set<driver_t> out;
738
739     for (int i = 0; i < 37; i++)
740     {
741         addr_t hashbucket = 0;
742         if (VMI_SUCCESS != vmi_read_addr_va(vmi, directory + this->guest_ptr_size * i, 4, &hashbucket) || !hashbucket)
743             continue;
744
745         while (true)
746         {
747             addr_t object = 0;
748             if (VMI_SUCCESS != vmi_read_addr_va(vmi, hashbucket + this->offsets[OBJECT_DIRECTORY_ENTRY_OBJECT], 4, &object))
749                 break;
750
751             unicode_string_t* obj_name = get_object_type_name(vmi, object);
752             if (obj_name)
753             {
754                 if (!strcmp((const char*)obj_name->contents, "Driver"))
755                     out.insert(object);
756
757                 if (!strcmp((const char*)obj_name->contents, "Directory"))
758                     for (auto obj : enumerate_directory(vmi, object))
759                         out.insert(obj);
760             }
761         }
762     }
763 }
```

Модификация DRIVER_OBJECT



Получение имени объекта

```
1 OBJECT_TYPE *__fastcall ObGetObjectType(void *object)
2 {
3     return ObTypeIndexTable[ObHeaderCookie ^ *(object - 24) ^ ((object - 48) >> 8)];
4 }
```

```
715 unicode_string_t* rootkitmon::get_object_type_name(vmi_instance_t vmi, addr_t object)
716 {
717     addr_t ob_header = object - this->object_header_size + this->guest_ptr_size;
718     uint8_t type_index;
719
720     // Object header is always present before actual object
721     if (VMI_SUCCESS != vmi_read_8_va(vmi, ob_header + this->offsets[OBJECT_HEADER_TYPEINDEX], 4, &type_index))
722         return nullptr;
723
724     if (this->winver == VMI_OS_WINDOWS_10)
725         type_index = type_index ^ ((ob_header >> 8) & 0xff) ^ this->ob_header_cookie;
726
727     addr_t ob_type;
728     if (VMI_SUCCESS != vmi_read_addr_va(vmi, this->type_idx_table + type_index * this->guest_ptr_size, 4, &ob_type))
729         return nullptr;
730
731     return drakvuf_read_unicode_va(vmi, ob_type + this->offsets[OBJECT_TYPE_NAME], 4);
732 }
```

Модификация DRIVER_OBJECT



Сохранение объекта драйвера и его списка DEVICE_OBJECT

```
for (const auto& drv_object : enumerate_driver_objects(vmi))
{
    auto address = drv_object + offsets[DRIVER_OBJECT_STARTIO];
    // 28 Major functions + DriverUnload + DriverStartIo = 30 pointers
    driver_object_checksums[drv_object] = calc_checksum(vmi, address, this->guest_ptr_size * 30);
    // Enumerate all device_stacks of a particular driver
    driver_stacks[drv_object] = enumerate_driver_stacks(vmi, drv_object);
}
```

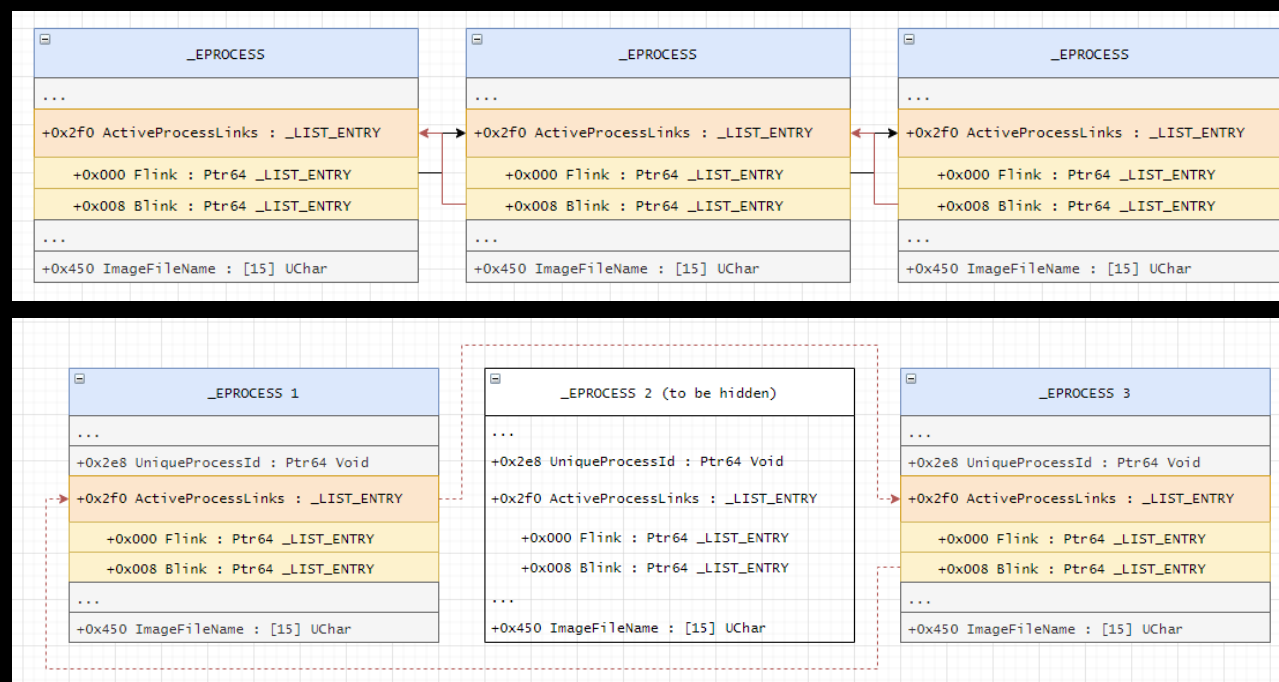
"\Device\HarddiskVolume2\Windows\System32\explorer.exe":KiDeliverApc SessionID:0 PID:1968 PPID:476
Reason:"Driver object modification"

"\Device\HarddiskVolume2\Windows\System32\explorer.exe":KiDeliverApc SessionID:0 PID:1968 PPID:476
Reason:"Driver stack modification"

Соккрытие процесса



EPROCESS – структура, описывающая процесс в системе
PsActiveProcessHead – начало связанного списка структур EPROCESS



<https://www.ired.team/miscellaneous-reversing-forensics/windows-kernel-internals/manipulating-activeprocesslinks-to-unlink-processes-in-userland>

Соккрытие процесса



- Поддерживаем список живых процессов в системе
- На этапе завершения процесса – удаляем
- На этапе создания процесса – добавляем

```
399
400     breakpoint_in_system_process_searcher bp;
401     if (!register_trap(nullptr, delete_process_cb, bp.for_syscall_name("PspProcessDelete"))
402         !register_trap(nullptr, insert_process_cb, bp.for_syscall_name("PspInsertProcess"))
```

```
347
348     PRINT_DEBUG("[DKOMMON] Inserting process %d\n", pid);
349     plugin->live_processes.insert(pid);
350     // In case new process is created with the same pid
351     plugin->dead_processes.erase(pid);
352
```

```
327
328     done:
329         plugin->live_processes.erase(pid);
330         plugin->dead_processes.insert(pid);
331
332     PRINT_DEBUG("[DKOMMON] Terminating process %d\n", (vmi_pid_t)pid);
```


Соккрытие процесса



Проверка структуры EPROCESS при завершении процесса и в конце анализа

```
315     ctx.addr = list_entry_va + plugin->offsets[LIST_ENTRY_FLINK];
316     if (VMI_FAILURE == vmi_read_addr(vmi, &ctx, &flink))
317         goto done;
318
319     ctx.addr = list_entry_va + plugin->offsets[LIST_ENTRY_BLINK];
320     if (VMI_FAILURE == vmi_read_addr(vmi, &ctx, &blink))
321         goto done;
322
323     if (list_entry_va == flink && flink == blink && flink && blink)
324     {
325         print_hidden_process_information(drakvuf, info, plugin, pid);
326     }
```

```
364     auto temp_processes = std::move(plugin->live_processes);
365     plugin->live_processes.clear();
366     drakvuf_enumerate_processes(drakvuf, process_visitor, static_cast<void*>(plugin));
367
368     for (vmi_pid_t pid : temp_processes)
369     {
370         if (std::find(plugin->live_processes.begin(), plugin->live_processes.end(), pid) == plugin->live_processes.end())
371         {
372             print_hidden_process_information(drakvuf, info, plugin, pid);
373         }
374     }
```

"\\Device\\HarddiskVolume2\\Windows\\System32\\explorer.exe":KiDeliverApc
SessionID:0 PID:1968 PPID:476 Reason:"Hidden process" HiddenPid:1624

Регистрация системных callback-функций



- **PspSetCreateProcessNotifyRoutine** – подписка на создание/завершение процесса
- **PspSetCreateThreadNotifyRoutine** – подписка на создание/завершение потока
- **PsSetLoadImageNotifyRoutine** – подписка на загрузку модулей
- **ObRegisterCallbacks** – подписка на получение описателей процесса, потока и рабочего стола

Регистрация системных callback-функций



Внутренний массив callback-функций PspCreateProcessNotifyRoutine

```
int64 __fastcall PspSetCreateProcessNotifyRoutine(_int64 function, unsigned int a2)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

    v2 = a2;
    v3 = a2 & 2;
    if ( (a2 & 1) != 0 )
    {
        CurrentThread = KeGetCurrentThread();
        --CurrentThread->KernelApcDisable;
        v9 = 0i64;
        while ( 1 )
        {
            v10 = ExReferenceCallbackBlock(&PspCreateProcessNotifyRoutine + v9);
            v11 = v10;
            if ( v10 )
            {
                LODWORD(v2) = v2 & 0xFFFFFFFF;
                if ( *(v10 + 8) == function
                    && *(v10 + 16) == v2
                    && ExCompareExchangeCallback(&PspCreateProcessNotifyRoutine + v9, 0i64, v10) )
                {
                    v12 = &PspCreateProcessNotifyRoutineCount;
                    if ( v3 )
                        v12 = &PspCreateProcessNotifyRoutineExCount;
                    _InterlockedDecrement(v12);
                    ExDereferenceCallbackBlock(&PspCreateProcessNotifyRoutine + v9, v11);
                    KiLeaveCriticalRegionUnsafe(CurrentThread);
                    ExWaitForRunDownProtectionRelease(v11);
                    ExFreePoolWithTag(v11, 0);
                    return 0i64;
                }
            }
        }
    }
}
```

Регистрация системных callback-функций



Rootkit Demodex, callback на работу с реестром

```
kd> dd nt!CmpCallbackCount L1
fffff806`0283f76c  00000002
kd> dps nt!CallbackListHead L2
fffff806`0283b2a0  fffffe28a`bfd0c740
fffff806`0283b2a8  fffffe28a`c6055b30
kd> dt nt!_LIST_ENTRY fffff806`0283b2a0
[ 0xfffffe28a`bfd0c740 - 0xfffffe28a`c6055b30 ]
+0x000 Flink          : 0xfffffe28a`bfd0c740 _LIST_ENTRY [ 0xfffffe28a`c6055b30 - 0xfffff806`0283b2a0 ]
+0x008 Blink          : 0xfffffe28a`c6055b30 _LIST_ENTRY [ 0xfffff806`0283b2a0 - 0xfffffe28a`bfd0c740 ]
kd> dps 0xfffffe28a`c6055b30 L8
fffffe28a`c6055b30  fffff806`0283b2a0 nt!CallbackListHead
fffffe28a`c6055b38  fffffe28a`bfd0c740
fffffe28a`c6055b40  00000000`00000001
fffffe28a`c6055b48  01d72c7a`e5e2140e
fffffe28a`c6055b50  00000000`00000000
fffffe28a`c6055b58  fffff806`05220e00 pci!ArbLibraryDeinitialize+0xa4
fffffe28a`c6055b60  00000000`000c000c
fffffe28a`c6055b68  fffffe28a`c58bc250
```

<https://securelist.com/ghostemperor-from-proxylogon-to-kernel-mode/104407/>

Регистрация системных callback-функций



Перехват регистрации callback-функций

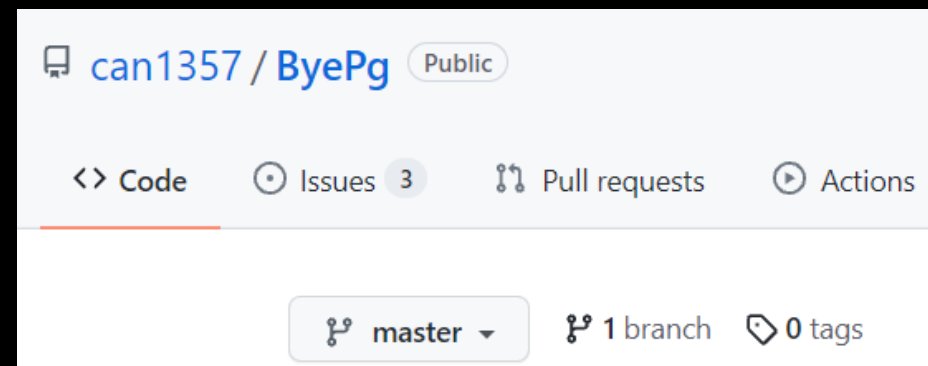
```
854     for (const auto& target : hook_targets)
855     {
856         auto hook = createSyscallHook(target, &rootkitmon::callback_hooks_cb);
857         if (!hook)
858         {
859             PRINT_DEBUG("[ROOTKITMON] Failed to hook %s\n", target);
860         }
861         else
862         {
863             hook->trap_->name = target;
864             this->syscall_hooks.push_back(std::move(hook));
865         }
866     }
```

```
607 event_response_t rootkitmon::callback_hooks_cb(drakvuf_t drakvuf, drakvuf_trap_info_t* info)
608 {
609     fmt::print(this->format, "rootkitmon", drakvuf, info,
610               keyval("Reason", fmt::Qstr(info->trap->name)));
611     return VMI_EVENT_RESPONSE_NONE;
612 }
```

Модификация HalPrivateDispatchTable



Перехват HalPrepareForBugCheck
для восстановления из состояния
ошибки



```
LONG (*HalDpReplaceBegin)(struct _HAL_DP_REPLACE_PARAMETERS* arg1, VOID** arg2);
VOID (*HalDpReplaceTarget)(VOID* arg1); //0x78
LONG (*HalDpReplaceControl)(ULONG arg1, VOID* arg2); //0x7c
VOID (*HalDpReplaceEnd)(VOID* arg1); //0x80
VOID (*HalPrepareForBugcheck)(ULONG arg1); //0x84
UCHAR (*HalQueryWakeTime)(ULONGLONG* arg1, ULONGLONG* arg2); //0x88
VOID (*HalReportIdleStateUsage)(UCHAR arg1, struct _KAFFINITY_EX* arg2); //0x8c
VOID (*HalTscSynchronization)(UCHAR arg1, ULONG* arg2); //0x90
```

Содержание



- Суть проблемы
- Плагин rootkitmon
- Техники
- Что будет дальше
- Заключение

Что будет дальше



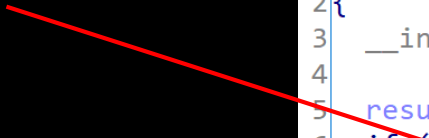
- Защита CI.DLL!g_CiOptions / ntoskrnl!g_CiEnabled
- Защита _OBJECT_TYPE_INITIALIZER
- Защита callback-списка типа объектов
- Защита IAT/EAT
- Защита SERVICE_RECORD
- Защита системных таблиц в драйверах tcpip.sys, http.sys, srv.sys
- Защита списка SAFESSEN

Что будет дальше



Отключение проверки подписи
загружаемых модулей путем
перезаписи g_CiEnabled переменной

```
1 __int64 SeValidateImageData()  
2 {  
3     __int64 result; // rax  
4  
5     result = 0i64;  
6     if ( !g_CiEnabled )  
7         return result;  
8     if ( ValidateImage )  
9         result = ValidateImage();  
10    else  
11        result = 0xC0000428i64;  
12    return result;  
13 }
```



Что будет дальше



Pre и Post operation callback-функции

```
// CALLBACK_ENTRY_ITEM
typedef struct _CALLBACK_ENTRY_ITEM {
    LIST_ENTRY CallbackList; // 0x0
    OB_OPERATION Operations; // 0x10
    DWORD Active; // 0x14
    CALLBACK_ENTRY *CallbackEntry; // 0x18
    PVOID ObjectType; // 0x20
    POB_PRE_OPERATION_CALLBACK PreOperation; // 0x28
    POB_POST_OPERATION_CALLBACK PostOperation; // 0x30
    QWORD unk1; // 0x38
} CALLBACK_ENTRY_ITEM, *PCALLBACK_ENTRY_ITEM; // size: 0x40
```

```
0: kd> dt nt!_OBJECT_TYPE fffffcf85`7688c380
+0x000 TypeList : _LIST_ENTRY [ 0xffffcf85`7688c380
+0x010 Name : _UNICODE_STRING "Process"
+0x020 DefaultObject : (null)
+0x028 Index : 0x7 ''
+0x02c TotalNumberOfObjects : 2
+0x030 TotalNumberOfHandles : 3
+0x034 HighWaterNumberOfObjects : 2
+0x038 HighWaterNumberOfHandles : 5
+0x040 TypeInfo : _OBJECT_TYPE_INITIALIZER
+0x0b8 TypeLock : _EX_PUSH_LOCK
+0x0c0 Key : 0x636f7250
+0x0c8 CallbackList : _LIST_ENTRY [ 0xffff9085`ea272980
```

```
0: kd> dps fffff9085`ea272980
fffff9085`ea272980 fffffcf85`7688c448
fffff9085`ea272988 fffffcf85`7688c448
fffff9085`ea272990 00000001`00000003
fffff9085`ea272998 fffff9085`ea272960
fffff9085`ea2729a0 fffffcf85`7688c380
fffff9085`ea2729a8 fffff804`5a2c3b90 WdFilter+0x43b90
fffff9085`ea2729b0 00000000`00000000
fffff9085`ea2729b8 00000000`00000000
```

Что будет дальше



Функции обработки объектов

```
0: kd> dt nt!_OBJECT_TYPE fffffcf85`7688c380
+0x000 TypeList : _LIST_ENTRY [ 0xffffcf85`7688c380
+0x010 Name : _UNICODE_STRING "Process"
+0x020 DefaultObject : (null)
+0x028 Index : 0x7 ''
+0x02c TotalNumberOfObjects : 2
+0x030 TotalNumberOfHandles : 3
+0x034 HighWaterNumberOfObjects : 2
+0x038 HighWaterNumberOfHandles : 5
+0x040 TypeInfo : OBJECT_TYPE_INITIALIZER
+0x0b8 TypeLock : _EX_PUSH_LOCK
+0x0c0 Key : 0x636f7250
+0x0c8 CallbackList : _LIST_ENTRY [ 0xffff9085`ea272
```

```
0: kd> dt nt!_OBJECT_TYPE_INITIALIZER 0xffffcf857688c3c0
+0x000 Length : 0x78
+0x002 ObjectTypeFlags : 0xca
+0x002 CaseInsensitive : 0y0
+0x002 UnnamedObjectsOnly : 0y1
+0x002 UseDefaultObject : 0y0
+0x002 SecurityRequired : 0y1
+0x002 MaintainHandleCount : 0y0
+0x002 MaintainTypeList : 0y0
+0x002 SupportsObjectCallbacks : 0y1
+0x002 CacheAligned : 0y1
+0x003 UseExtendedParameters : 0y0
+0x003 Reserved : 0y00000000 (0)
+0x004 ObjectTypeCode : 0x20
+0x008 InvalidAttributes : 0xb0
+0x00c GenericMapping : _GENERIC_MAPPING
+0x01c ValidAccessMask : 0x1fffffff
+0x020 RetainAccess : 0x101000
+0x024 PoolType : 200 ( NonPagedPoolNx )
+0x028 DefaultPagedPoolCharge : 0x1000
+0x02c DefaultNonPagedPoolCharge : 0xa98
+0x030 DumpProcedure : (null)
+0x038 OpenProcedure : 0xfffff804`58acc650 long nt!PspProcessOpen+0
+0x040 CloseProcedure : 0xfffff804`58afb6e0 void nt!PspProcessClose+0
+0x048 DeleteProcedure : 0xfffff804`58a31050 void nt!PspProcessDelete+0
+0x050 ParseProcedure : (null)
+0x050 ParseProcedureEx : (null)
+0x058 SecurityProcedure : 0xfffff804`58a7ffe0 long nt!SeDefaultObjectMethod+0
```

Содержание



- Суть проблемы
- Плагин rootkitmon
- Техники
- Что будет дальше
- **Заключение**

Полезные ссылки



PT Sandbox Telegram-чат

<https://t.me/ptsandbox>



PT Sandbox

ptsecurity.com/ru-ru/products/sandbox/



PT ESC Threat Intelligence blog

ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/



PT ESC Incident Response Alert

ptsecurity.com/ru-ru/services/esc/



Вопросы

webinar@ptsecurity.com

Руткиты: что это такое и чем опасны

<https://www.ptsecurity.com/ru-ru/research/webinar/rutkity-cto-eto-takoe-i-chem-opasno/>

Распаковка исполняемых файлов: статический и динамический подход

<https://www.ptsecurity.com/ru-ru/research/webinar/raspakovka-ispolnyaemyh-fajlov-staticeskij-i-dinamicheskij-podhod/>

Павел Максютин

pmaksyutin@ptsecurity.com

Алексей Вишняков

avishnyakov@ptsecurity.com

Twitter: @Vishnyak0v