TheMida : defeating ring0

deroko/ARTeam

Version 1.0 – May 2006

Keywords

Themida, unpacking

1. Abstract

TheMida 1.0.0.5 with offensive ring0 driver is not very used nowadays. But still it doesn't mean that we shouldn't pay attention to it. Even more, when something is not researched and written about it is more fun to play with such protection. I have nothing more to say about it, I will show you my research and all I have done to repair dump.

My target is Advanced Submitter 4.1.4 [3]

Of course, to play with themida 1.0.0.5 we are going to use some tools:

- IDA
- LiveKd
- wARK
- hiew
- tasm32

Forget about debugger it is not useful while playing with themida 1.0.0.5 and ring0 hooks.

S verom u Boga, deroko/ARTeam

2. Know your enemy and dump it

Protection implemented by oreans.sys is weird it hooks some SSDT entries, IDT and some exports of ntoskrnl.exe:

```
SSDT entry 0035 hooked by : 0xF3B7ACBE - NtCreateThread
SSDT entry 003A hooked by : 0xF3B7B1A0 - NtDebugContinue
SSDT entry 00B2 hooked by : 0xF3B7AACA - NtQueryVirtualMemory
SSDT entry 00BA hooked by : 0xF3B7A014 - NtReadVirtualMemory
SSDT entry 0101 hooked by : 0xF3B7A9D0 - NtTerminateProcess
SSDT entry 0115 hooked by : 0xF3B7A000 – NtWriteVirtualMemory
```

Now we will check hooks in exports of ntoskrnl.exe:

```
C:\>scanhook.exe
scanhook       - (c) 2006 deroko/ARTeam
Hooked : KeAttachProcess
Hooked : vsprintf
C:\>
```

Well KeAttachProcess is hooked because themida want to deny any pte changing from ring0 and process dumping from ring0. Well it doesn’t hook KeStackAttachProcess so we can still attach to process from ring0 and dump it, but we are also able to emulate KeAttachProcess very simple.

Now let’s see what we have in IDT:

```
01h   0008:FFFFFFFF     Interrupt   32 bit      03    0
03h   0008:FFFFFFFF     Interrupt   32 bit      03    0
0Bh   0008:EBF0E3E8     Interrupt   32 bit      00    1
0Eh   0008:EBF0E000     Interrupt   32 bit      00    1
```

Oki, int1 and int3 are “hooked” with 0xFFFFFFFF, the reason why themida hooks int 0eh is to catch 0xFFFFFFFF memory access and according to instruction that caused access to 0xFFFFFFFF to transfer execution to KiTrap01 or KiTrap03, in other words, default handlers. Of course if it finds int3h in themida protected process, my best guess is BSOD because I didn’t want to experiment and to cause another BSOD just to learn in hard way to do not mess with themida hooks J

If you try to remove any of these hooks you will get BSOD.

So to understand themida hooks I used one small home made ring0 memory dumper to dump hooks. But before we move to hook dumps I will show you hooks for KeAttachProcess and vsprintf in ntoskrnl.exe dumped with LiveKd:

```
kd> u KeAttachProcess
nt!KeAttachProcess:
804e3173 e9884e5d78       jmp     f8ab8000
804e3178 56               push    esi
804e3179 57               push    edi


kd> u ntoskrnl!vsprintf
nt!vsprintf:
8050716b 33c0             xor     eax,eax
8050716d c3               ret
```

Oki, if we dump memory content of hook stored in KeAttachProcess we will see this code:

```
            push  ebp
            mov   ebp, esp
            pusha
            call  $+5
            pop   edx
            sub   edx, 940FB78h
            push  fs                   ; save FS
            mov   eax, 30h             ; make it point to kpcr
            mov   fs, ax               ;
            mov   eax, large fs:124h   ; grab CurrentThread from kpcr
            mov   eax, [eax+44h]       ; grab EPROCESS from ETHREAD
            mov   [edx+940FC00h],   eax ; save EPROCESS struct
            pop   fs                   ; restore fs
            mov   eax, [ebp+4]         ; now it takes saved EIP from stack
            cmp   eax, [edx+940FBF0h]  ; this part is not interesting for us
            jbe   short loc_F8AB8040   ; because it is junk
            cmp   eax, [edx+940FBF4h]  ;
            jnb   short loc_F8AB8040
            jmp   short loc_F8AB807D
; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

loc_F8AB8040:
            cmp   eax, [edx+940FBF8h]   ;still junk
            jbe   short loc_F8AB8052
            cmp   eax, [edx+940FBFCh]
            jnb   short loc_F8AB8052
            jmp   short loc_F8AB807D
; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ

loc_F8AB8052:
            mov   eax, [ebp+8]          ;oki now we get to good stuff
            cmp   eax, [edx+940FC00h]   ;cmp passed EPROCESS to KeAttachProcess
            jz    short loc_F8AB8076    ;with EPROCESS of protected process
            mov   esi, 0F8AAC000h       ;themida internal struct
            add   esi, 4                ;esi points to EPROCESS field in
                                        ;internal struct


loc_F8AB8065:
            cmp   dword ptr [esi], 47616420h ;is it signature?
            jz    short loc_F8AB807D    ;if so end of loop
            cmp   [esi], eax            ;EPROCESS of protected process
            jz    short loc_F8AB8076    ;compared to saved one
            add   esi, 4                ;go to next entry in struct
            jmp   short loc_F8AB8065    ;loop
; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
; dumped structure comes here so it is easier to follow code
            dd 47616420h                ;signature (' daG'
            dd 81B58DA0h                ;EPROCESS of protected process
            dd 47616420h                ;signature again
            dd 0

; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ


loc_F8AB8076:                            ;return w/o attaching to target process
            popa
            pop   ebp
            xor   eax, eax
            retn  4


loc_F8AB807D:                                ;call some code in oreans.sys
            popa
            pop   ebp
            jmp   short loc_F8AB8095


loc_F8AB8095:
            mov   edi, edi
            push  ebp
            mov   ebp, esp
            jmp   near ptr 804E3178h     ;KeAttachProcess+5
```

To write driver that will be able to use KeAttachProcess we have to write small procedure that will be wrapper for KeAttachProcess, but this is completely unnecessary work because we may attach using KeStackAttachProcess but what the hell, we are doing this for fun and to learn something, at least that's the only reason why I'm playing with themida.

```
KeAttachProcessWraper:
                mov     edi, edi
                push    ebp
                mov     ebp, esp
                iMOV    eax, KeAttachProcess
                add     eax, 5
                jmp     eax     ;execute KeAttachProcess+5
```

Simple isn’t it J

This is very important instruction in above code:

```
            mov   esi, 0F8AAC000h       ;themida internal struct
```

If we take a look in disassembly of hook in IDA we might see that this instruction is located at hook_base + 5Dh:

```
seg000:0000005B                 jz      short loc_76
seg000:0000005D                 mov     esi, 0F8AAC000h
seg000:00000062                 add     esi, 4
```

This structure is very important because it describes state of protected process, it has simple structure:

```
            dd 47616420h    ;signature (' daG'
            dd 81B58DA0h    ;EPROCESS of protected process
            dd 47616420h    ;signature again
            dd 0
            dd 0
```

If we examine address 81B58DA0h in livekd we may notice that it is actually EPROCESS of protected application:

```
kd> dt nt!_EPROCESS 81b58da0
   ...
   +0x168 PageDirectoryPte : _HARDWARE_PTE
   +0x168 Filler           : 0
   +0x170 Session          : 0xf8a55000
   +0x174 ImageFileName    : [16]  "AdvanceSubmitte"      <--- application name
   +0x184 JobLinks         : _LIST_ENTRY [ 0x0 - 0x0 ]
   ...
kd>
```

Voila we have researched internal themida struct.

Next important thing is that themida will remove hooks once protected application is done with its executing. To accomplish this themida uses hook of NtTerminateProcess so we are going to dump that memory region and analyze NtTerminateProcess hook:

```
            push  ebp
            mov   ebp, esp
            pusha
            call  $+5
            pop   edx
            sub   edx, 94552AFh
            push  edx
            push  0
            lea   eax, [edx+945531Fh]
            push  eax
            push  0
            mov   eax, 80559CD8h                ;PsProcessType
            xor   eax, eax                      ;eax set to 0
            push  eax
            push  10h
            push  dword ptr [ebp+8]
            mov   eax, 8055D468h
            call  eax                           ;ObReferenceObjectByHandle
            pop   edx
            cmp   dword ptr [edx+945531Fh], 0
            jz    short loc_F364BA3A
            mov   eax, [edx+945531Fh]           ;EPROCESS of cur process
            mov   ebx, eax
            and   ebx, 7FFFFFFFh
            mov   esi, 0F8AAC000h               ; internal struct

loc_F364BA1D:
            add   esi, 4
            cmp   dword ptr [esi], 47616420h
            jz    short loc_F364BA3A
            cmp   [esi], eax
            jz    short loc_F364BA32
            cmp   [esi], ebx
            jz    short loc_F364BA32
            jmp   short loc_F364BA1D


; ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
```