

SECURING PRODUCTIVITY



Crimeware
Anti-Reverse Engineering
Uncovered
APWG 2006

Nicolas Brulez
Virus Researcher

Agenda

- **Introduction**

- **Crimeware and Anti Reverse Engineering Techniques**
 - **Anti Debugging**
 - **Anti Virtual Machine**
 - **Custom Packers / Protectors**

- **Detecting and Removing New Packers**
 - **File Format Analysis**
 - **Unpacking**

- **Questions**

Introduction

- **Anti Reverse Engineering Techniques are becoming very common in CrimeWare.**
- **Phishing Trojans use those techniques as well as custom Packers.**
- **Those trojans are updated quite often with new anti emulation techniques**
- **Bots are using custom packers/protectors as well.**

CrimeWare and Anti Reverse Engineering Techniques

▪ Anti Debugging

Common Techniques:

- IsDebuggerPresent is still the most common and simple.
- CreateFileA on Debugger's Diver: NTICE etc
- SEH : Structured Exception Handling to mess with debuggers

Less Common:

- FindWindow to detect Debuggers like OllyDbg
- CloseHandle on invalid handles to generate an Exception (and then use some timing detection to detect tracing)
- ZwQueryInformationProcess with ProcessDebugPort Parameter
- SetUnhandledExceptionFilter can be used to redirect code flow if a debugger is present
- PEB_LDR_DATA+48h used to check for a debugger. (we should have the imagebase of the process when there is no debugger attached)
- A lot more techniques are found every now and then.

CrimeWare and Anti Reverse Engineering Techniques

■ Anti Virtual Machine

– Example from a Banking Trojan:

```
.text:10003386
.text:10003386 ; ||| S U B R O U T I N E |||
.text:10003386
.text:10003386 ; BOOL __stdcall dllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpvReserved)
.text:10003386 _dllMain@12 proc near ; CODE XREF: dllEntryPoint+6C↓p
.text:10003386 var_128 = dword ptr -128h
.text:10003386
.text:10003386 mov eax, offset loc_1000A390
.text:1000338B call __EH_prolog
.text:1000338B
.text:10003390 sub esp, 118h
.text:10003396 push ebx
.text:10003397 push esi
.text:10003398 push edi
.text:10003399 mov [ebp-10h], esp
.text:1000339C call VMware_Backdoor
.text:1000339C
.text:100033A1 test eax, eax
.text:100033A3 jnz short reverser_detected
.text:100033A3
.text:100033A5 call CheckVirtualPC
.text:100033A5
.text:100033AA test eax, eax
.text:100033AC jnz short reverser_detected
.text:100033AC
.text:100033AE call ds:IsDebuggerPresent
.text:100033B4 test eax, eax
.text:100033B6 jnz short reverser_detected
.text:100033B6
```


CrimeWare and Anti Reverse Engineering Techniques

- Virtual PC detection:

```
text:10007610
text:10007610 checkVirtualPC:                                ; CODE XREF: DllMain(x,x,x)+1F1p
text:10007610                                           ; DllMain(x,x,x)+D91p
text:10007610         push    14h
text:10007612         push    offset stru_1000b190
text:10007617         call   __SEH_prolog
text:10007617
text:1000761C         and     dword ptr [ebp-20h], 0
text:10007620         and     dword ptr [ebp-4], 0
text:10007624         push    ebx
text:10007625         mov     ebx, 0                ; EBX will remain 0 with virtual PC
text:1000762A         mov     eax, 1                ; Virtual PC function number
text:1000762A         ; -----
text:1000762F         db     0Fh, 3Fh, 7, 0Bh     ; VPC Call
text:10007633         ; -----
text:10007633         test   ebx, ebx
text:10007635         setz   byte ptr [ebp-20h] ; set flag if EBX = 0 = VPC running
text:10007639         pop     ebx
text:1000763A         jmp     short loc_10007670
text:10007670
```

CrimeWare and Anti Reverse Engineering Techniques

- Newer version of the Banking Trojan uses Generic Virtual Machine detection

```
.text:10002DB0 ; BOOL __stdcall DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpvReserved)
.text:10002DB0 _DllMain@12 proc near ; CODE XREF: DllEntryPoint+4B1p
.text:10002DB0
.text:10002DB0 hinstDLL = dword ptr 4
.text:10002DB0 fdwReason = dword ptr 8
.text:10002DB0 lpvReserved = dword ptr 0Ch
.text:10002DB0
* .text:10002DB0 mov eax, offset loc_1000A6A4
* .text:10002DB5 call ___EH_prolog
* .text:10002DBA sub esp, 118h
* .text:10002DC0 push ebx
* .text:10002DC1 push esi
* .text:10002DC2 push edi
* .text:10002DC3 mov [ebp-10h], esp
* .text:10002DC6 call Anti_Emulation_SIDT_Based_Check ; Get IDT Base Address and check if its emulated
* .text:10002DC6
* .text:10002DCB test eax, eax
* .text:10002DCD jnz short reverser_detected
* .text:10002DCD
* .text:10002DCF call ds:IsDebuggerPresent ; Check PEB+2 for BeingDebugged Flag.
* .text:10002DD5 test eax, eax
* .text:10002DD7 jnz short reverser_detected
* .text:10002DD7
* .text:10002DD9 mov eax, [ebp+0Ch]
* .text:10002DDC xor edi, edi
* .text:10002DDE sub eax, edi
* .text:10002DE0 jz loc_10002F3D
* .text:10002DE0
* .text:10002DE6 dec eax
* .text:10002DE7 jz short loc_10002E18
* .text:10002DE7
* .text:10002DE9 dec eax
* .text:10002DEA jnz short reverser_detected
```

CrimeWare and Anti Reverse Engineering Techniques

- **Interrupt Descriptor Table Base Address Check. This one will detect about every Virtual Machine you can find:**

```
.text:10007116 ; ||| SUBROUTINE |||
.text:10007116
.text:10007116
.text:10007116 Anti_Emulation_SIDT_Based_Check proc near ; CODE XREF: dllMain(x,x,x)+161p
.text:10007116         call     Get_IDT_base
.text:10007116
.text:10007116         and     eax, 0FF000000h
.text:10007120         xor     ecx, ecx
.text:10007122         cmp     eax, 80000000h ; Real windows Machine always have 0x80 for their MSB
.text:10007127         setnz  cl
.text:1000712A         mov     eax, ecx ; If EAX != 0 we are emulating windows
.text:1000712C         retn
.text:1000712C Anti_Emulation_SIDT_Based_Check endp
.text:1000712C
```

```
.text:10007108
.text:10007108 Get_IDT_base proc near ; CODE XREF: Anti_Emulation_SIDT_Based_Check1p
.text:10007108
.text:10007108 var_8 = qword ptr -8
.text:10007108
.text:10007108         push   ecx
.text:10007109         push   ecx
.text:1000710A         sidt   [esp+8+var_8] ; Get IDT Base Address
.text:1000710F         mov     eax, dword ptr [esp+8+var_8+2] ; EAX = IDT Base
.text:10007113         pop    ecx
.text:10007114         pop    ecx
.text:10007115         retn
.text:10007115
.text:10007115 Get_IDT_base endp
.text:10007115
```

CrimeWare and Anti Reverse Engineering Techniques

- **One year ago a PE file infector « W32.Bayan » has been discovered. It's a Polymorphic Entry Point Obscuring Virus.**
- **It uses a random number of encryption layers. Each layer contains junk code. Some of those junk instructions will crash under Vmware (still does in the latest Vmware)**
- **Yes, it's a virus BUT?**
- **A few months after, some Trojans were using the encryption layers from the virus: No more Vmware for the analysis.**
- **I wrote a custom tracer (as an Ollydbg Script) to trace and bypass every layers, removing the Anti Vmware instructions.**
- **Once the trojan is fully decrypted, we can dump it and analyse the code.**

CrimeWare and Anti Reverse Engineering Techniques

```
IDA View-EIP
.rsrc:0043608A sub_43608A proc near ; CODE XREF: start:loc_43605F↑p
.rsrc:0043608A push ebp ; Most of the code is junk code
.rsrc:0043608B mov ebp, esp
.rsrc:0043608D sbb edx, 20FC6D0h
.rsrc:00436093 cmovl ebx, esi
.rsrc:00436096 fadd st(5), st
EIP .rsrc:00436098 verw ax ; This crashes under VMware: not under real computers
.rsrc:0043609B mov eax, 20FEDF0h
.rsrc:004360A0 neg ebx
.rsrc:004360A2 pop ebp
.rsrc:004360A3 pop edi
.rsrc:004360A4 dec edi
.rsrc:004360A5 not edx
.rsrc:004360A7 verw di ; Anti UM!
.rsrc:004360AA mov bx, 452Bh
.rsrc:004360AE jo short loc_4360BD
.rsrc:004360AE
.rsrc:004360B0 xor edi, ecx
.rsrc:004360B2 mov ebx, ecx
.rsrc:004360B4 cmovbe edi, esp
.rsrc:004360B7 xor edi, ecx
.rsrc:004360B9 xchg ebx, ebx
.rsrc:004360BB cmp al, 0E6h
.rsrc:004360BB
.rsrc:004360BD
.rsrc:004360BD loc_4360BD: ; CODE XREF: sub_43608A+24↑j
.rsrc:004360BD jno short loc_4360DD
.rsrc:004360BD
.rsrc:004360BF lea ebx, ds:215CB78h
.rsrc:004360C5 push ebx
.rsrc:004360C6 pop eax
.rsrc:004360C7 xchg eax, edi
.rsrc:004360C9 xchg edx, eax
.rsrc:004360CB mov ax, 5661h
.rsrc:004360CF stc
.rsrc:004360D0 arpl si, dx
.rsrc:004360D2 bsf eax, edx
.rsrc:004360D5 xchg eax, edi
.rsrc:004360D7 neg edi
.rsrc:004360D9 cmovno esi, eax
.rsrc:004360DC inc ebx
.rsrc:004360DC
.rsrc:004360DD
.rsrc:004360DD loc_4360DD: ; CODE XREF: sub_43608A:loc_4360BD↑j
```


CrimeWare and Anti Reverse Engineering Techniques

- **Example of custom packers:**
 - Section name : .ccg (There is a chinese PE protector from a group called CCG, but they are quite different)
 - 68 Exceptions during Self Unpacking
 - Timing Detection with RDTSC to detect single stepping.
 - Many more Anti Debugging tricks: Detect Software breakpoints inside the Application code with SEH etc.
- **Removing the Packer:**
 - We can (ab)use the exceptions to quickly bypass every detection tricks.
 - On last exception we can start tracing it (we need to take care of the Breakpoint detection tricks: Software BPX detection and Hardware Breakpoints removal)

CrimeWare and Anti Reverse Engineering Techniques

- **Ollyscript to count the number of Exceptions (look inside the View Window)**

```
// Nicolas Brulez SEH counter
```

```
var counter
```

```
eof lbl1
```

```
run
```

```
lbl1:
```

```
cob
```

```
coe
```

```
esto
```

```
add counter, 1
```

```
log counter
```

```
jmp lbl1
```

CrimeWare and Anti Reverse Engineering Techniques

- Exception logs from my script:

```
00321D4C counter = 00000031
Integer division by zero
counter = 00000032
00321DE2 Integer division by zero
counter = 00000033
00321EB7 Access violation when reading [00000000]
counter = 00000034
00321EF2 Access violation when reading [00000000]
counter = 00000035
00321F30 Access violation when reading [00000000]
counter = 00000036
00321F58 Access violation when reading [00000000]
counter = 00000037
00321F80 Access violation when reading [00000000]
counter = 00000038
0032203B Integer division by zero
counter = 00000039
003220EF Access violation when reading [00000000]
counter = 0000003A
003221E1 Integer division by zero
counter = 0000003B
00322239 Access violation when reading [00000000]
counter = 0000003C
00322268 Integer division by zero
counter = 0000003D
0032234F Access violation when reading [00000000]
counter = 0000003E
003223A3 Access violation when reading [00000000]
counter = 0000003F
00322418 Access violation when reading [00000000]
counter = 00000040
00322440 Access violation when reading [00000000]
counter = 00000041
003225DA INT3 command at 003225DA
counter = 00000042
00322682 Integer division by zero
counter = 00000043
counter = 00000044
```

CrimeWare and Anti Reverse Engineering Techniques

- Ollyscript to stop before the last Exception so we can finish the debugging manually.

```
// Nicolas Brulez Stop SEH
```

```
var counter
```

```
mov counter, 43
```

```
eob lbl1
```

```
eof lbl1
```

```
run
```

```
lbl1:
```

```
cob
```

```
coe
```

```
log counter
```

```
cmp counter, 0
```

```
je lbl2
```

```
esto
```

```
sub counter, 1
```

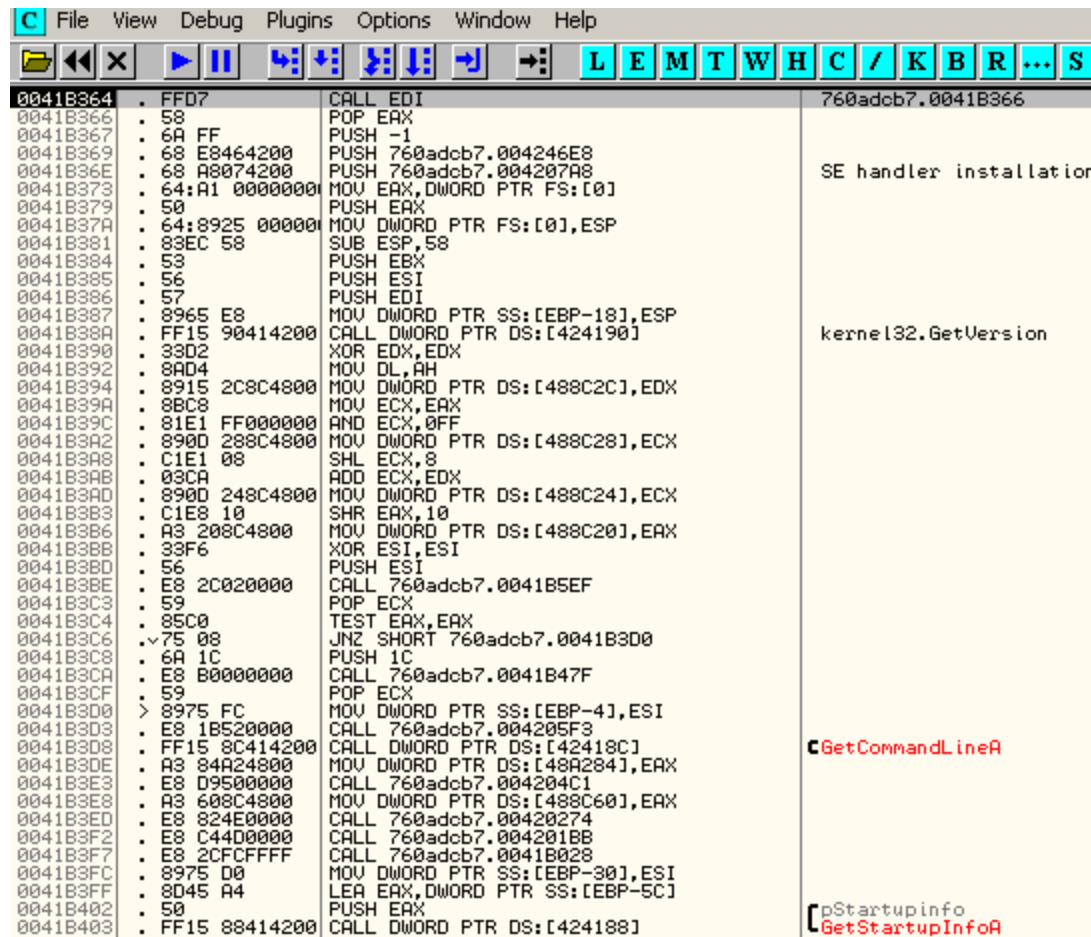
```
jmp lbl1
```

```
lbl2:
```

```
ret
```

CrimeWare and Anti Reverse Engineering Techniques

- A few Anti Break Points later ;-): Fully decrypted sample



The screenshot shows a debugger window with the following assembly code and function names:

Address	Disassembly	Comment
0041B364	FFD7	CALL EDI
0041B366	58	POP EAX
0041B367	6A FF	PUSH -1
0041B369	68 E8464200	PUSH 760adcb7.004246E8
0041B36E	68 A8074200	PUSH 760adcb7.004207A8
0041B373	64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
0041B379	50	PUSH EAX
0041B37A	64:8925 000000	MOV DWORD PTR FS:[0],ESP
0041B381	83EC 58	SUB ESP,58
0041B384	53	PUSH EBX
0041B385	56	PUSH ESI
0041B386	57	PUSH EDI
0041B387	8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
0041B38A	FF15 90414200	CALL DWORD PTR DS:[424190]
0041B390	33D2	XOR EDX,EDX
0041B392	8AD4	MOV DL,AH
0041B394	8915 2C8C4800	MOV DWORD PTR DS:[488C2C],EDX
0041B39A	8BC8	MOV ECX,EAX
0041B39C	81E1 FF000000	AND ECX,0FF
0041B3A2	890D 288C4800	MOV DWORD PTR DS:[488C28],ECX
0041B3A8	C1E1 08	SHL ECX,8
0041B3AB	03CA	ADD ECX,EDX
0041B3AD	890D 248C4800	MOV DWORD PTR DS:[488C24],ECX
0041B3B3	C1E8 10	SHR EAX,10
0041B3B6	A3 208C4800	MOV DWORD PTR DS:[488C20],EAX
0041B3BB	33F6	XOR ESI,ESI
0041B3BD	56	PUSH ESI
0041B3BE	E8 2C020000	CALL 760adcb7.0041B5EF
0041B3C3	59	POP ECX
0041B3C4	85C0	TEST EAX,EAX
0041B3C6	75 08	JNZ SHORT 760adcb7.0041B3D0
0041B3C8	6A 1C	PUSH 1C
0041B3CA	E8 B0000000	CALL 760adcb7.0041B47F
0041B3CF	59	POP ECX
0041B3D0	8975 FC	MOV DWORD PTR SS:[EBP-4],ESI
0041B3D3	E8 1B520000	CALL 760adcb7.004205F3
0041B3D8	FF15 8C414200	CALL DWORD PTR DS:[42418C]
0041B3DE	A3 84A24800	MOV DWORD PTR DS:[48A284],EAX
0041B3E3	E8 D9500000	CALL 760adcb7.004204C1
0041B3E8	A3 608C4800	MOV DWORD PTR DS:[488C60],EAX
0041B3ED	E8 824E0000	CALL 760adcb7.00420274
0041B3F2	E8 C44D0000	CALL 760adcb7.004201BB
0041B3F7	E8 2CFCFFFF	CALL 760adcb7.0041B028
0041B3FC	8975 D0	MOV DWORD PTR SS:[EBP-30],ESI
0041B3FF	8D45 A4	LEA EAX,DWORD PTR SS:[EBP-5C]
0041B402	50	PUSH EAX
0041B403	FF15 88414200	CALL DWORD PTR DS:[424188]

Function names and comments on the right side of the window:

- 760adcb7.0041B366
- SE handler installation
- kernel32.GetVersion
- GetCommandLineA
- GetStartupInfoA

Detecting and Removing New Packers

- **What is Packing anyway ?**
 - **Allows to compress/encrypt applications**
 - **You can't see the code of the application using a disassembler, you need to unpack it first.**
 - **Packers compress applications and add a small loader to the file.**
 - **The loader will uncompress the binary in memory, resolve imports, and call the Original Entry Point (OEP).**
 - **We need to find OEP and dump the process to disk, and rebuild the import table.**

First Steps: Is my file Packed?

- **Is the last section executable ?**
- **Is the first section writeable ?**
- **Is the first section's raw size null ?**
- **Is the Entry Point starting in the last section ?**
- **Check the section names**
- **Check the Import Table : Very few imported functions ?**
- **Check the strings : no strings at all ?**
- **Is the Raw Size way smaller than the Virtual Size?
Compressed!**

First Steps: Is my file Packed?

[PE Editor] - c:\documents and settings\nico\bureau\binaries\day2\unpacking fs

Basic PE Header Information

EntryPoint: 00005000 Subsystem: 0002

[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
	00001000	00003000	00000000	00000000	C00000E0
	00004000	00001000	00000400	000000BA	C00000E0
	00005000	00001000	00000200	00000200	C00000E0

[PE Editor] - c:\documents and settings\nico\bureau\binaries\day2\unpacking p...

Basic PE Header Information

[Section Table]

Name	VOffset	VSize	ROffset	RSize	Flags
.text	00001000	00003000	00000200	00000600	E0000060
.rsrc	00004000	000108CC	00000800	00001200	E0000020

Magic: 010B NumOfRvaAndSizes: 00000010

[Section Flags]

Set Flags:

- Shareable in memory
- Executable as code
- Readable
- Writeable
- Contains extended relocations
- Discardable as needed
- Can't be cached
- Not pageable
- Contains COMDAT data
- Contains comments or other infos
- Won't become part of the image
- Contains executable code
- Contains initialized data
- Contains uninitialized data
- Shouldn't be padded to next boundary

Alignment: default Bytes Current Value: E0000020

Unpacking

- **Unpacking knowledge is very handy for Reverse Engineers.**
- **Most malwares are packed to hide their real code from Disassemblers.**
- **There are a *lot* of different PE packers and PE protectors out there, and many have no public unpackers.**
- **Fortunately, most packers (and “Protectors” :P) are easy to remove.**

First Steps: Unpacking

- *Find the Original Entry Point*
 - Trace slowly until you jump to the real program code.
 - Use Static Disassembly to find the jump to original entry point.
 - Smart use of hardware breakpoints. (Write access is your friend).
 - Breakpoints on API Functions.
 - Use Stack (pushad is your friend)

- *Dump the process to disk*
 - Using tools such as LordPE or Imprec Process dumpers.

First Steps : Unpacking

- ***Reconstruct the Import Table***
 - Trace the packer's code and find where the IAT handling is, so you can grab information about the import table and reconstruct it manually, eventually. (or patch the protector so it will not destroy the imports at all 😊)
 - You can just use “Import Reconstructor” to reconstruct the import table and get ride of the boring work most of the time.
 - Sometimes we need to write plugins for Imprec, but usually it only takes a dozen minutes.

SECURING PRODUCTIVITY



Automatic Anti VM Removal Demo

Questions?

- English is not my native language, please ask slowly ;-)
- Thanks 😊
- If you are interested into Anti Reverse Engineering Techniques : <http://honeynet.org/scans/scan33/>
 - A challenge i made 2 years ago:
 - My complete paper with all the submissions to see how you can attack armored binaries

nbrulez@websense.com

<http://WebsenseSecurityLabs.com>

<http://www.reverse-engineer.org>