

# Anti-Analysis

Aaron Sedlacek

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
sub_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Why?

- To delay or prevent analysis
  - Requires the analyst to be more skilled
  - By slowing down analysts, authors buy their malware more time to complete their task(s)
  - Subvert detection algorithms and antivirus heuristic engines

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    [ebp+var_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Agenda

Anti-Disassembly  
Anti-Debugging  
Anti-Virtual Machine  
Anti-Antivirus

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Disassemblers

- Linear
  - Process one instruction at a time, linearly, without deviating
- Flow-Oriented
  - Examines each instruction and builds a list of locations to disassemble
    - Looks at things like jumps, calls, etc
  - Method used by most commercial disassemblers (IDA)

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jnz     short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_3140F3
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Anti-Disassembly

- Malware takes advantage of disassembly heuristics
- Exploit the most basic assumptions of the disassembler

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
push    esi
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

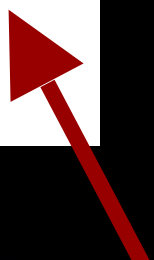
```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Garbage Bytes

```
.text:0040100E      jz      short near ptr loc_401010+1
.text:00401010
.text:00401010 loc_401010:      ; CODE XREF: .text:0040100E↑j
.text:00401010      call     near ptr 8B4C55A0h
.text:00401015      dec     eax
.text:00401016      add     al, 0Fh
.text:00401018      mov     esi, 70FA8311h
.text:0040101D      jnz     short loc_40105E
```



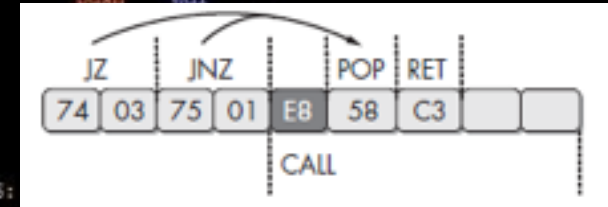
```
.text:0040100E      jz      short loc_401011
.text:0040100E ; -----
.text:00401010      db 0E8h ; F
.text:00401011 ; -----
.text:00401011 loc_401011:      ; CODE XREF: .text:0040100E↑j
.text:00401011      mov     eax, [ebp+0Ch]
.text:00401014      mov     ecx, [eax+4]
.text:00401017      movsx   edx, byte ptr [ecx]
.text:0040101A      cmp     edx, 'p'
.text:0040101D      jnz     short loc_40105E
```



# Opaque predicates

- The most common technique seen in the wild

```
74 03      jz      short near ptr loc_4011C4+1
75 01      jnz     short near ptr loc_4011C4+1
loc_4011C4:      ; CODE XREF: sub_4011C0
                  ; ②sub_4011C0+2j
E8 58 C3 90 90      ❶call     near ptr 90D00521h
```



- Essentially an unconditional jump, but IDA always processes false branches first


# Opaque predicates

- Always takes one branch
  - Doom used this for his death-rays™ in MBE!

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
```

```
xor     eax, eax
jz      short loc_401497
; -----
db 0E9h ; T
; -----

loc_401497:                                ; CODE XREF: .text:00401494↑j
push    offset dword_4014C0
push    large dword ptr fs:0
mov     large fs:0, esp
xor     ecx, ecx
div     ecx
push    offset aForMoreInforma ; "For more information please visit our w"...
call    printf
add     esp, 4
pop     edi
pop     esi
pop     ebx
pop     ebp
retn
```





# Lab 15-01

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax

```

# Backwards Jumps

- Jump into middle of own instruction

```
.text:00401212      mov     byte ptr [eax], 0
.text:00401215 loc_401215:                                     ; CODE XREF: .text:loc_401215
.text:00401215      jmp     short near ptr loc_401215+1
.text:00401217      db      0C0h ; +
.text:00401218      db      48h ; H
.text:00401219      db      0E8h ; F
.text:0040121A      db      0F1h ; ±
.text:0040121B      db      0
.text:0040121C      db      0
.text:0040121D      db      0
.text:0040121E      db      89h ; ë
.text:0040121F      db      85h ; à
.text:00401220      db      58h ; X
.text:00401221      db      0FDh ; º
.text:00401222      db      0FEh ; ¡
.text:00401223      db      0FFh
.text:00401224      db      68h ; h
```

# Backwards Jumps

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi

```

```

.text:00401209 call    ds:stretsr
.text:0040120F add     esp, 8
.text:00401212 mov     byte ptr [eax], 0
.text:00401212 ; -----
.text:00401215 db 0EEh ; d
.text:00401216 ; -----
.text:00401216 inc     eax
.text:00401218 dec     eax
.text:00401219 call    sub_40130F
.text:0040121E mov     [ebp-102A8h], eax
.text:00401224 push    0A000000h
.text:00401229 call    ds:malloc
.text:0040122F add     esp, 4
.text:00401232 mov     [ebp-102ACh], eax
.text:00401238 mov     ecx, [ebp-298h]
.text:0040123E add     ecx, 8
.text:00401241 mov     [ebp-298h], ecx
.text:00401247 push    0
.text:00401249 push    0
.text:0040124B push    0
.text:0040124D push    0
.text:0040124F mov     edx, [ebp-298h]
.text:00401255 push    edx
.text:00401256 mov     eax, [ebp-102A4h]
.text:0040125C push    eax
.text:0040125D call    ds:InternetOpenUrlA

```

```

.text:00401209 call    ds:stretsr
.text:0040120F add     esp, 8
.text:00401212 mov     byte ptr [eax], 0
.text:00401215 nop
.text:00401216 nop
.text:00401217 nop
.text:00401218 nop
.text:00401219 call    sub_40130F
.text:0040121E mov     [ebp-102A8h], eax
.text:00401224 push    0A000000h
.text:00401229 call    ds:malloc
.text:0040122F add     esp, 4

```

```

cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----

loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax

```

# Lab 15-02

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax

```

# Return Pointer Manipulation

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax

```

ESP	
+4	
+8	

```

push    00h
call    sub_31411B

```

```

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49

```

```

call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

```

```

loc_31307D:                                     ; CODE XREF: sub_312FD8

```

```

call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

```

```

loc_31308C:                                     ; CODE XREF: sub_312FD8

```

```

mov     [ebp+var_4], eax

```

(From the PMA Book)

# Return Pointer Manipulation

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax

```

ESP	0x004011C5
+4	
+8	



```

004011C0 sub_4011C0      proc near          ; CODE XREF: _main+19p
004011C0                                     ; sub_401040+88p
004011C0
004011C0 var_4          = byte ptr -4
004011C0
004011C0 call    $+5
004011C5 add     [esp+4+var_4], 5
004011C9 retn
004011C9 sub_4011C0      endp ; sp-analysis failed
004011C9

```

(From the PMA Book)

```

loc_313066:
push    00h
call    sub_31411B

loc_31306D:
; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

loc_31307D:
; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:
; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax

```



# Return Pointer Manipulation

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
mov short loc_313066, eax
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
```

```
004011C0 sub_4011C0      proc near          ; CODE XREF: _main+19p
004011C0                                     ; sub_401040+88p
004011C0
004011C0 var_4          = byte ptr -4
004011C0
004011C0 call    $+5
004011C5 add     [esp+4+var_4], 5
004011C9 retn
004011C9 sub_4011C0      endp ; sp-analysis failed
004011C9
```

ESP	0x004011C5
+4	
+8	

(From the PMA Book)

loc\_313066:

```
push 0Dh
call sub_31411B
```

loc\_31306D:

; CODE XREF: sub\_312FD8  
; sub\_312FD8+49

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

loc\_31307D:

; CODE XREF: sub\_312FD8

```
call sub_3140F3
and eax, 0FFFFh
or eax, 80070000h
```

loc\_31308C:

; CODE XREF: sub\_312FD8

```
mov [ebp+var_4], eax
```

# Return Pointer Manipulation

```

push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax

```

```

004011C0 sub_4011C0      proc near                ; CODE XREF: _main+19p
004011C0                                     ; sub_401040+88p
004011C0
004011C0 var_4          = byte ptr -4
004011C0
004011C0 call    $+5
004011C5 add     [esp+4+var_4], 5
004011C9 retn
004011C9 sub_4011C0      endp ; sp-analysis failed
004011C9

```

ESP	0x004011CA
+4	
+8	



(From the PMA Book)

```

loc_313066:
push    00h
call    sub_31411B

loc_31306D:
; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

loc_31307D:
; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:
; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax

```

# Return Pointer Manipulation

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
```

ESP	0x004011CA
+4	
+8	

```
push    00h
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

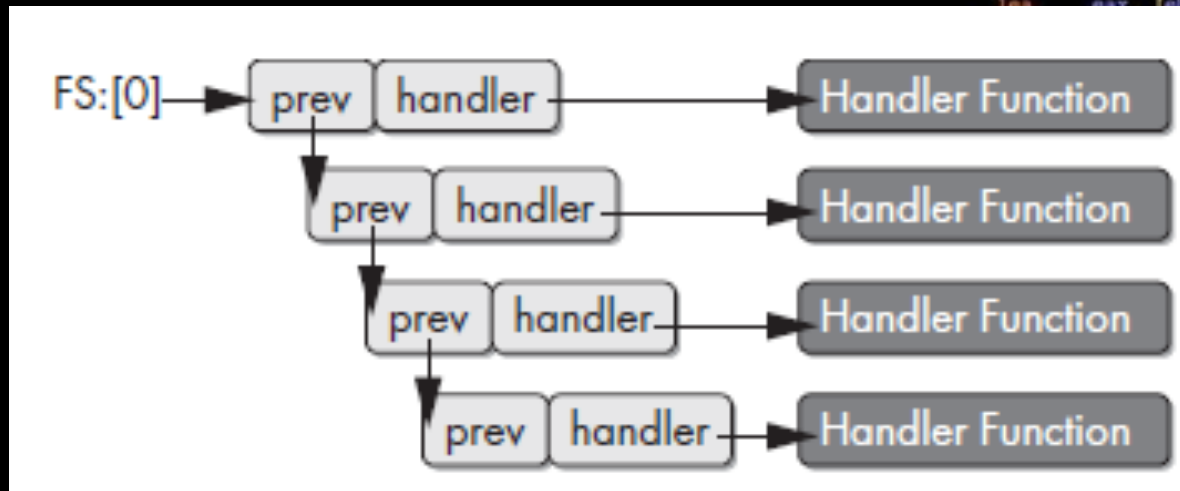
```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

(From the PMA Book)

# Misusing SEH

- SEH - Structured Exception Handling



(From the PMA Book)

# Misusing SEH

- The Disassembler Doesn't understand SEH

```
.text:00401497 loc_401497: ; CODE XREF: .text:00401494fj
.text:00401497      push      offset dword_4014C0
.text:0040149C      push      large dword ptr fs:0
.text:004014A3      mov       large fs:0, esp
.text:004014AA      xor       ecx, ecx
.text:004014AC      div       ecx
.text:004014AE      push      offset aForMoreInforma ; "For more information please visit our w"...
.text:004014B3      call      printf
.text:004014B8      add       esp, 4
.text:004014BB      pop       edi
.text:004014BC      pop       esi
.text:004014BD      pop       ebx
.text:004014BE      pop       ebp
.text:004014BF      retn
.text:004014BF ; -----
.text:004014C0 dword_4014C0 dd 824648Bh, 0A164h, 8B0000h, 0A364008Bh, 0
.text:004014C0 ; DATA XREF: .text:loc_401497fo
.text:004014D4      dd 0EB08C483h, 0E848C0FFh, 0
.text:004014E0 ; -----
```

```
push      edi
call      sub_314623
test      eax, eax
jz        short loc_31306D
cmp       [ebp+arg_0], ebx
jnz       short loc_313066
mov       eax, [ebp+var_70]
cmp       eax, [ebp+var_84]
jb        short loc_313066
sub       eax, [ebp+var_84]
push      esi
push      esi
push      eax
push      edi
mov       [ebp+arg_0], eax
call      sub_31462A
test      eax, eax
jz        short loc_31306D
push      esi
```

```
test      eax, eax
jg        short loc_31307D
call      sub_3140F3
jmp       short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call      sub_3140F3
and       eax, 0FFFFh
or        eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov       [ebp+var_4], eax
```

# Misusing SEH

```
loc_401497:                                ; CODE XREF:
push    offset dword_4014C0
push    large dword ptr fs:0
mov     large fs:0, esp
xor     ecx, ecx
div     ecx
push    offset aForMoreInforma ; "Fo
call    printf
add     esp, 4
pop     edi
pop     esi
pop     ebx
pop     ebp
retn

; -----
dword_4014C0 dd 824648Bh, 0A164h, 8B00000h, 0A3640
; DATA XREF:
dd 0EB08C483h, 0E848C0FFh, 0
; -----
push    ebp
mov     ebp, esp
```

```
loc_4014C0:                                ; CODE
push    offset loc_4014C0
push    large dword ptr fs:0
mov     large fs:0, esp
xor     ecx, ecx
div     ecx
push    offset aForMoreInforma
call    printf
add     esp, 4
pop     edi
pop     esi
pop     ebx
pop     ebp
retn

; -----
loc_4014C0:                                ; DATA
mov     esp, [esp+8]
mov     eax, large fs:0
mov     eax, [eax]
mov     eax, [eax]
mov     large fs:0, eax
add     esp, 8
```

```
loc_31307D:                                ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```



# Stack Frame Analysis

- Code that confuses stack frame analysis
  - IDA Automatically constructs arguments, it tries to map the stack
  - [CTRL+K] allows you to examine the stack frame
  - [ALT+K] lets you adjust the stack pointer

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     eax, [ebp+var_70]
call    sub_31266A
test    eax, eax
short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Agenda

Anti-Disassembly  
**Anti-Debugging**  
Anti-Virtual Machine  
Anti-Antivirus

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi

push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

# Windows API

- Windows API Calls to watch out for:
  - IsDebuggerPresent
    - Checks the **PEB** (Process Environment Block)
  - CheckRemoteDebuggerPresent
    - Actually checks for a process on the local machine
  - NtQueryInformationProcess
    - Can be used to check for a debugger
  - OutputDebugString
    - Sends a string to a debugger for display
    - OllyDBG format string vulnerability

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
loc_313066:
mov     eax, [ebp+arg_0]
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_3140F3
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
push    esi
push    [ebp+arg_4]
push    esi
call    sub_3140F3
test    eax, eax
jz      short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

# Manual Debugging Checks

- Checking the **BeingDebugged** Flag
  - The flag located at **0x02** in the **PEB** structure
    - The **PEB** is located at fs:[**30h**] in x86
- Checking the **ProcessHeap** Flag
  - located at **0x18** in the **PEB** structure
- Checking the **NTGlobal** Flag
  - location **0x68** in the **PEB** structure

```
push     edi
call     sub_314623
test     eax, eax
jz       short loc_31306D
cmp      [ebp+arg_0], ebx
jnz      short loc_313066
mov      eax, [ebp+var_70]
cmp      eax, [ebp+var_84]
jb       short loc_313066
sub      eax, [ebp+var_84]
push     esi
```

```
push     esi
push     eax
push     edi
mov      [ebp+arg_0], eax
call     sub_31486A
test     eax, eax
jnz      short loc_31306D
push     esi
mov      eax, [ebp+arg_0]
push     eax
mov      esi, 1D0h
push     esi
push     [ebp+arg_4]
push     edi
call     sub_314623
test     eax, eax
jz       short loc_31306D
cmp      [ebp+arg_0], esi
jz       short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
```

```
push     0Dh
call     sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
call     sub_3140F3
test     eax, eax
jg       short loc_31307D
call     sub_3140F3
jmp      short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call     sub_3140F3
and      eax, 0FFFFh
or       eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov      [ebp+var_4], eax
```

# Manual Debugging Checks

- Malware can also check for system residue
  - HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug
  - Search the system for specific files or directories
  - Search the current process listing
  - Use **FindWindow** to search for a debugger
    - Similar to what Spotify was doing

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
call sub_31466A
jz short loc_31306D
push eax
lea eax, [ebp+arg_0]
push eax
mov esi, 100h
push esi
push [ebp+arg_4]
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFh
or eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```



# Code Checks

- Malware can search its' own code for software interrupts
  - **INT 3** (aka software breakpoint, **0xCC**)
  - You can beat this with hardware breakpoints
- Malware can also checksum sections of its code as well
  - You can beat this with hardware breakpoints

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    esi
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+55
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```



# Timing Checks

- Malware runs more slowly when being debugged
  - Debugger executes code, too. Has overhead.
  - **rdtsc** Instruction
    - Gets the count of ticks since last system reboot
  - **QueryPerformanceCounter** + **GetTickCount** as well

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+var_70], eax
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
mov     esi, ebx
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], eax
jb      short loc_313066
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# TLS Callbacks

- A section in the PE header that executes before the entry point
  - Frequently used to subvert debuggers, since the .tls section is executed before debugging begins!

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
call    sub_31466A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    esi
push    esi
push    esi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

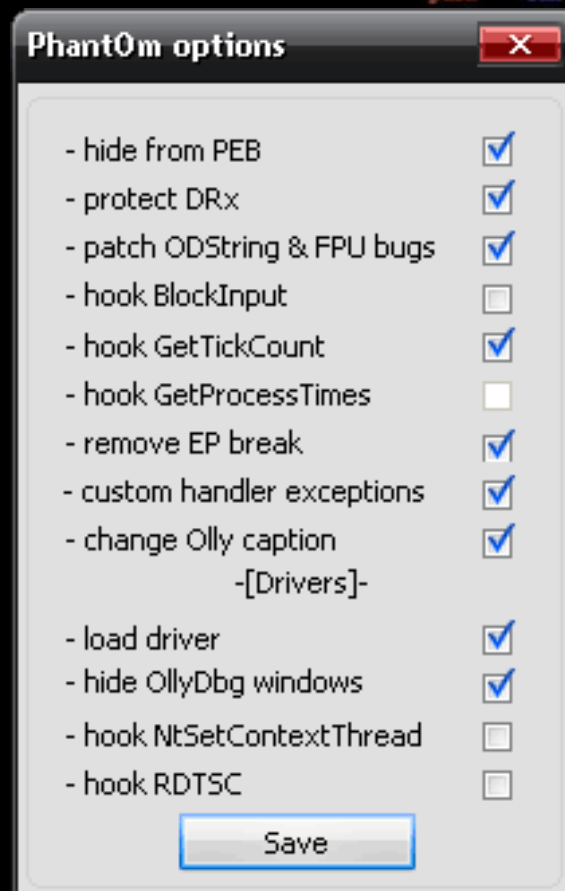
```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Fighting back

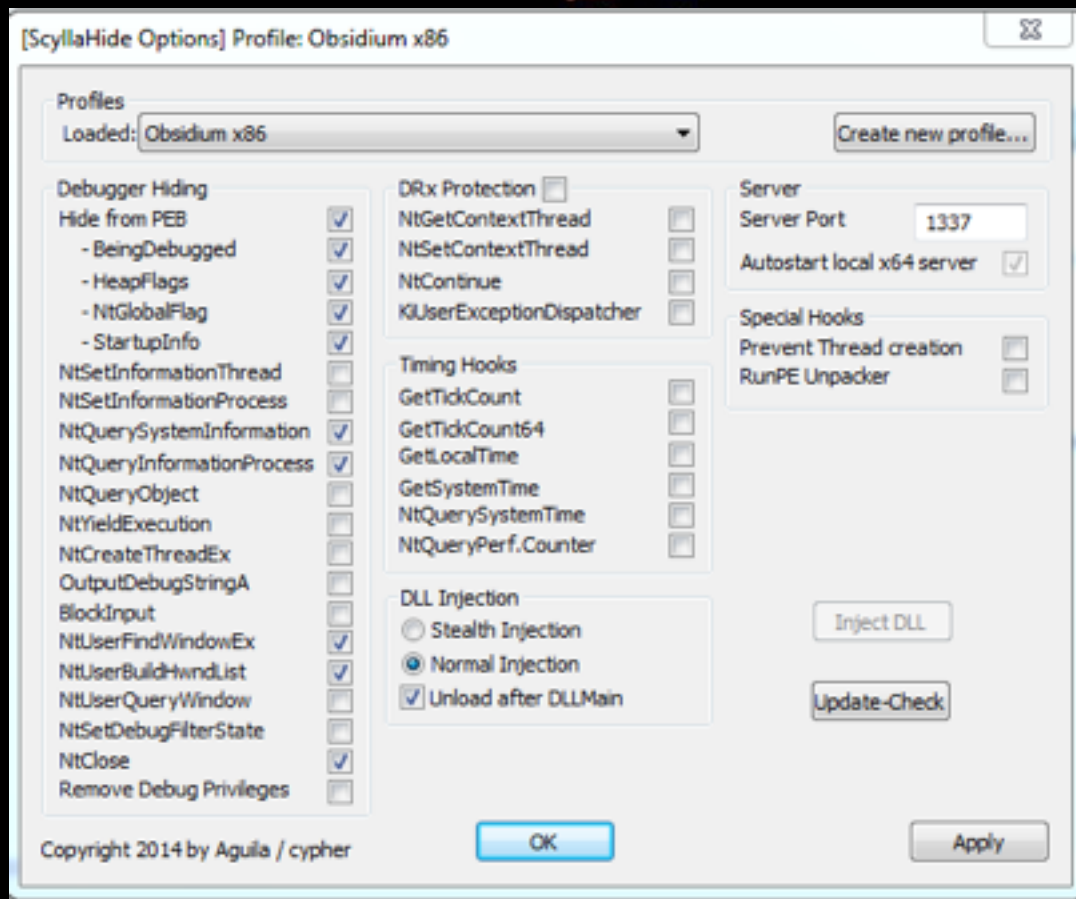
- OllyDbg plugins
  - Phant0m



# Fighting back

- OllyDbg plugins  
– ScyllaHide

Debugger agnostic!



# Lab 16-02

```

push     edi
call     sub_314623
test     eax, eax
jz       short loc_31306D
cmp      [ebp+arg_0], ebx
jnz      short loc_313066
mov      eax, [ebp+var_70]
cmp      eax, [ebp+var_84]
jb       short loc_313066
sub      eax, [ebp+var_84]
push     esi

push     esi
push     eax
push     edi
mov      [ebp+arg_0], eax
call     sub_31486A
test     eax, eax
jz       short loc_31306D
push     esi
lea      eax, [ebp+arg_0]
push     eax
mov      esi, 1D0h
push     esi
push     [ebp+arg_4]
push     edi
call     sub_314623
test     eax, eax
jz       short loc_31306D
cmp      [ebp+arg_0], esi
jz       short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
push     0Dh
call     sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
call     sub_3140F3
test     eax, eax
jg       short loc_31307D
call     sub_3140F3
jmp      short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call     sub_3140F3
and      eax, 0FFFFh
or       eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov      [ebp+var_4], eax

```

# Agenda

Anti-Disassembly  
Anti-Debugging  
**Anti-Virtual Machine**  
Anti-Antivirus

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```



# VM Artifacts

- Just like Debugging, VMs leave artifacts on the system
- You can patch these checks out! #swag

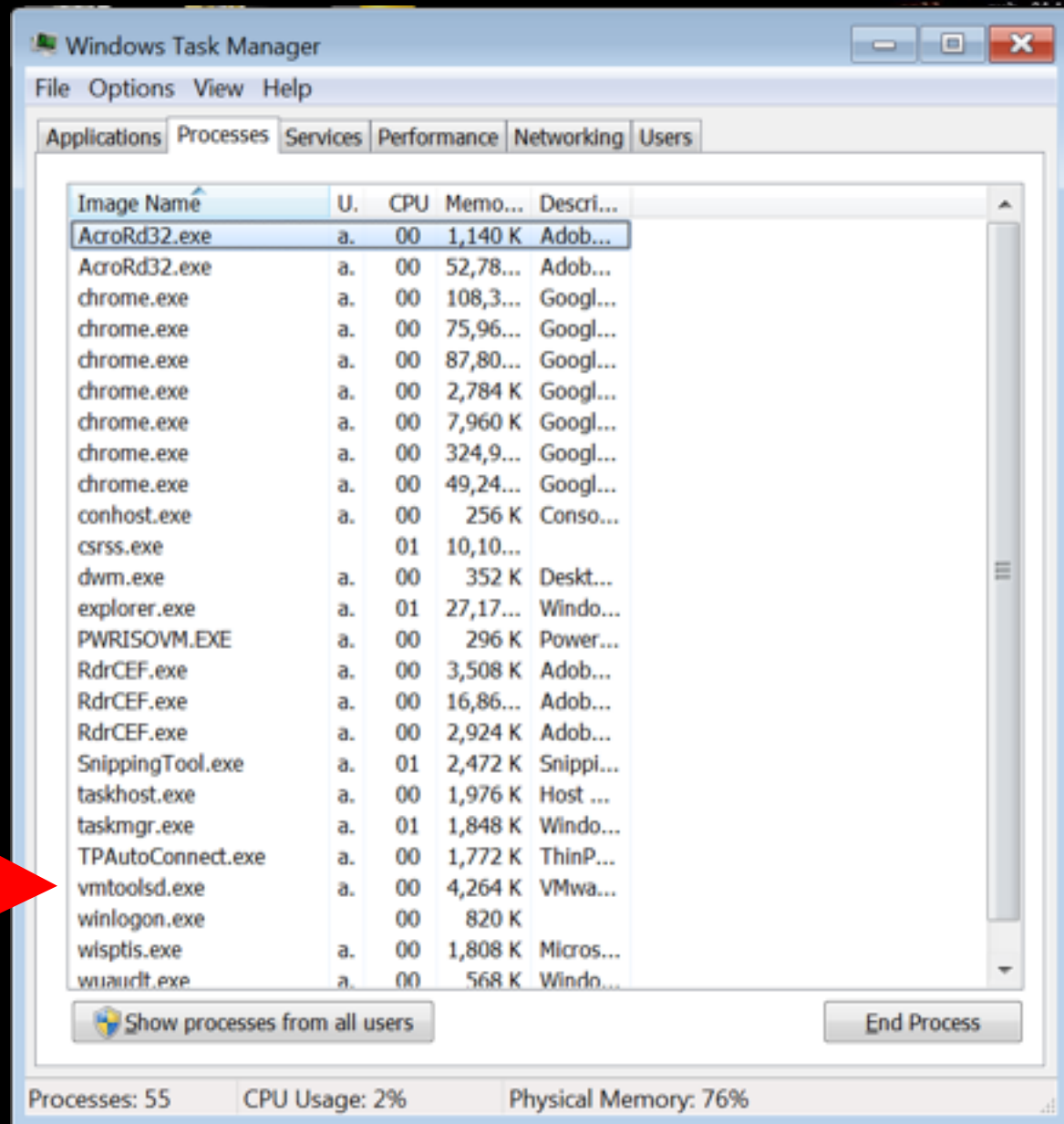
```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, [ebp+var_70]
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```



# Vulnerable Instructions

- Some instructions are executed differently in VM's
  - No Pill Technique - **sgdt**, **sldt**, **sidt**
  - Querying the I/O Comm port - **in**
  - Segment Selector - **str**
  - Other - **cpuid**, **smsw**, see VM specific docs for others
- For now, just patch these out

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push esi
call [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push esi
call sub_314821
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFh
or eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```

# Red Pills

- Instructions with different semantics in-VM/emulator than on real hardware
- 906 instructions in x86 base architecture, lots to get wrong
  - Intel manuals aren't even always correct
- Examples
  - `bsf edi, [esp-0xc]` (produces incorrect status flags in QEMU)
  - `mulps xmm1, [esp-4]` (does not raise fault if source operand is not correctly aligned in Bochs)
  - `push 0x407d` (erroneously alters FPU state in Bochs)

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push esi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
push [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313086: ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
loc_31305D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
loc_31307D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```

# Anti-Virtual Machine

- Most anti-VM can be easily patched out
- ScoopyNG
  - Pretty useful free Anti-VM detection tool
- Side note: Malware can exploit your VM and escape to your host! woohoo...

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    esi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+var_70]
push    esi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```



# Agenda

Anti-Disassembly  
Anti-Debugging  
Anti-Virtual Machine  
Anti-Antivirus

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```



# Antivirus

- Protect computers from malware, both before it can run, and at run time
- Before execution: file hashes/signatures, emulator scans, heuristic code analysis
- Run time: hooks in system libraries, monitoring from the kernel, network/file system monitoring

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
call    [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
push    esi
push    [ebp+arg_4]
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Inline Hooks

- WinAPI is *designed* for hooks
  - Open kernel32 in IDA and you'll see “**mov edi, edi**” in many function prologues
  - Provides a 2-byte **NOP** (but better for cache than two instructions), lets you insert a short jump to a long jump to your hook code

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_31486A
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Anti-Antivirus

- Basically all the same techniques apply
  - Look for AV processes running or installed on the machine
  - Examine libraries for inline hooks
  - Obfuscate code
  - Use packers to change hash signatures

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Active Anti-AV

- Careto (“The Mask”) used some debug code left in Kaspersky’s KLIF.SYS driver to whitelist itself
- Could theoretically exploit AV on scan, AVs are *notoriously* insecure



# AV Emulators

- Everything that applies to VMs and emulators also applies to consumer-level commercial AV emulators
  - Emulation of x86
  - Emulation of WinAPI
  - Run quickly on consumer-level computers
  - Written in-house by tiny teams
  - ...what could go wrong?

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```



# Exploiting AVs

- LOADS of bugs

- [http://mincore.c9x.org/breaking\\_av\\_software.pdf](http://mincore.c9x.org/breaking_av_software.pdf)

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_313066
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F

loc_313066:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                              ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C

; -----
loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```



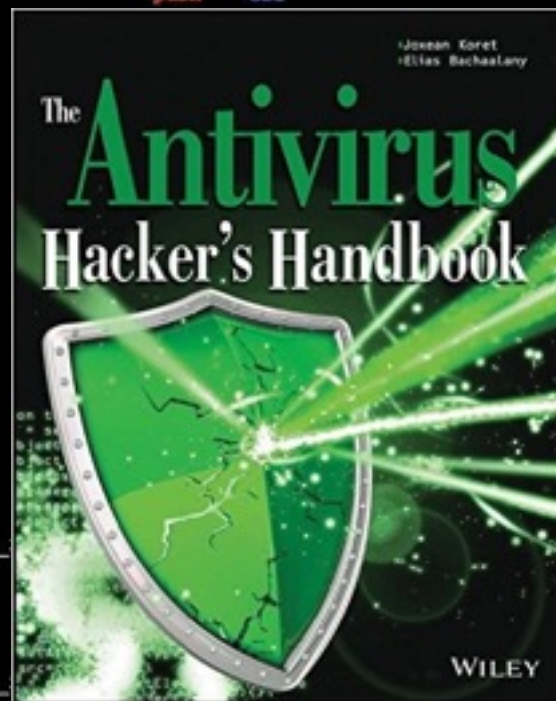
# Real World AV Sandbox Evasion

- Happening in the wild, both with generic behavior (stalling loops, timing checks) and targeted AV-specific checks
- EvilBunny checks if its name is “TESTAPP” (name used for all programs running in BitDefender Emulator)

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
mov     [ebp+arg_0], ebx
mov     eax, loc_31306D
mov     eax, [ebp+var_70]
mov     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     esi, [ebp+arg_0]
call    sub_31486A
test    eax, eax
jz      short loc_31306D
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31308F
; CODE XREF: sub_312FD8
; sub_312FD8+55
push    0Dh
call    sub_31411B
loc_31306D:
; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; -----
loc_31307D:
; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
loc_31308C:
; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

# If (anti-)AV Interests You

“The Antivirus Hacker’s Handbook” by Joxean Koret and Elias Bachaalany



```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnl     short loc_313066
mov     eax, [ebp+var_70]
mov     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
```

```
loc_31306D:  ; CODE XREF: sub_312FD8+55
; sub_312FD8+49
loc_31307D:  ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:  ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:  ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# Questions?

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
```

```
push    0Dh
call    sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call    sub_3140F3
and     eax, 0FFFFh
or      eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov     [ebp+var_4], eax
```

# References

1. Sikorski, Michael, and Andrew Honig. *Practical Malware Analysis the Hands-on Guide to Dissecting Malicious Software*. San Francisco: No Starch, 2012. Print.
2. Paleari, Roberto, Lorenzo Martignoni, Giampaolo Fresi Roglia, and Danilo Bruschi. 2009. "A fistful of red-pills : How to automatically generate procedures to detect CPU emulators." In *WOOT'09 Proceedings of the 3rd USENIX conference on Offensive technologies*
3. Marschalek, Marion. 2014. "EvilBunny: Malware Instrumented By Lua." <http://www.cyphort.com/evilbunny-malware-instrumented-lua/>.