A Thorny Piece of Malware (And Me)

A Talk about Excpetion Handlers, VFTables, Multi-Threading and other Nastiness

Marion Marschalek @pinkflawd
DEFCON21

[user@host:~$ whoami]

[user@host:~$ whoami]

Outline

- **Malware Functionality**
  Fancy Fun Facts
- **Anti-Analysis**
  Exceptions for Fun & Profit
  Auto-Junk & Obfuscation
- **Analysts' Headaches**
  C++ or: Function Calls to Nirvana
  Thread me to Hell

My Favorite Piece
of Malware

what is it?

An asian
multi-threaded
non-polymorphic
file-infecting
spy-bot.

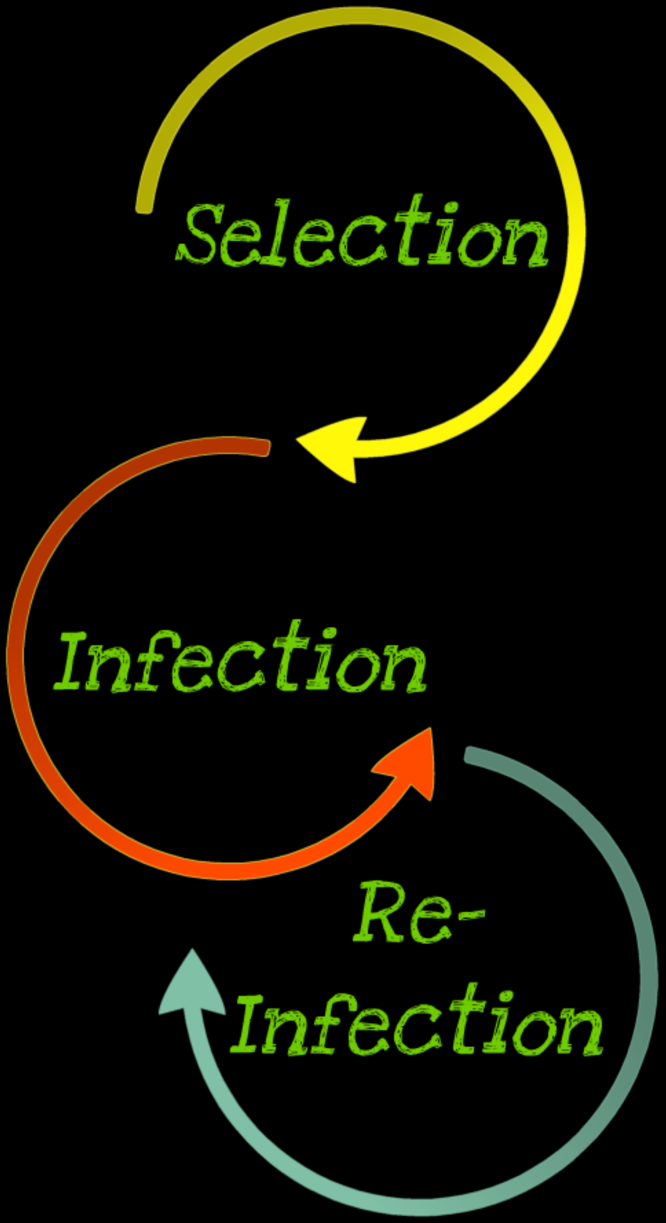
Fancy Fun Facts

Picky Old-School File Infector

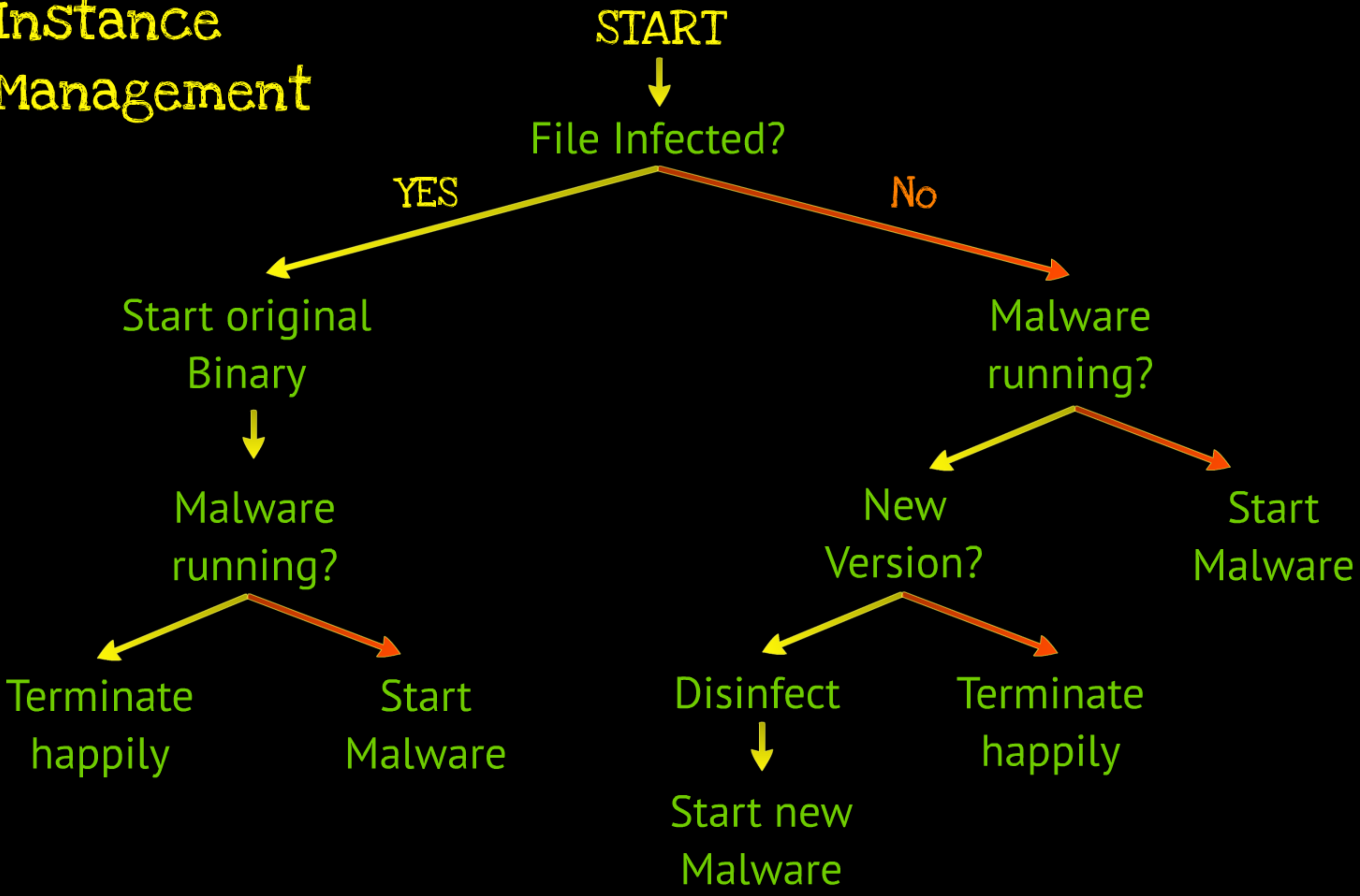
Filter Function

when Qihoo360 or Rising AV running
stop!
when process name contains
- netthief
- visual studio
- world of warcraft, …

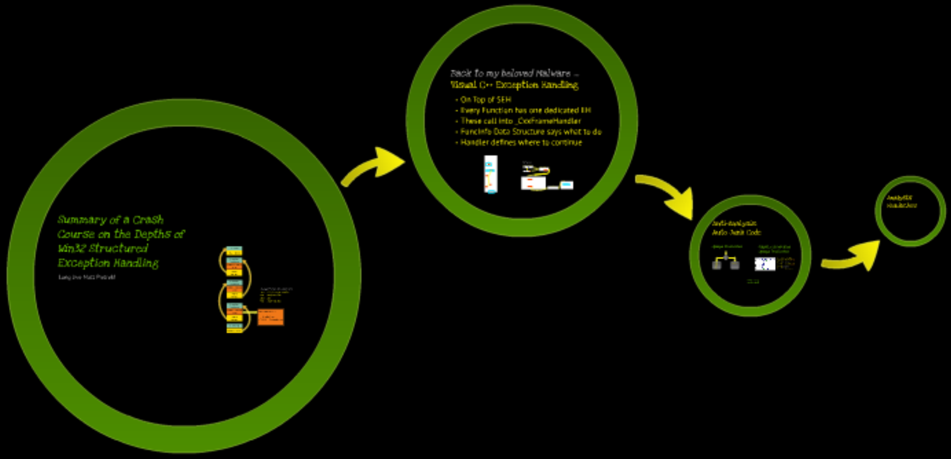
exclude!
Now.. What does that mean?



Startup & Instance Management



Anti-Analysis:
Exceptions for Fun & Profit


Summary of a Crash Course on the Depths of Win32 Structured Exception Handling

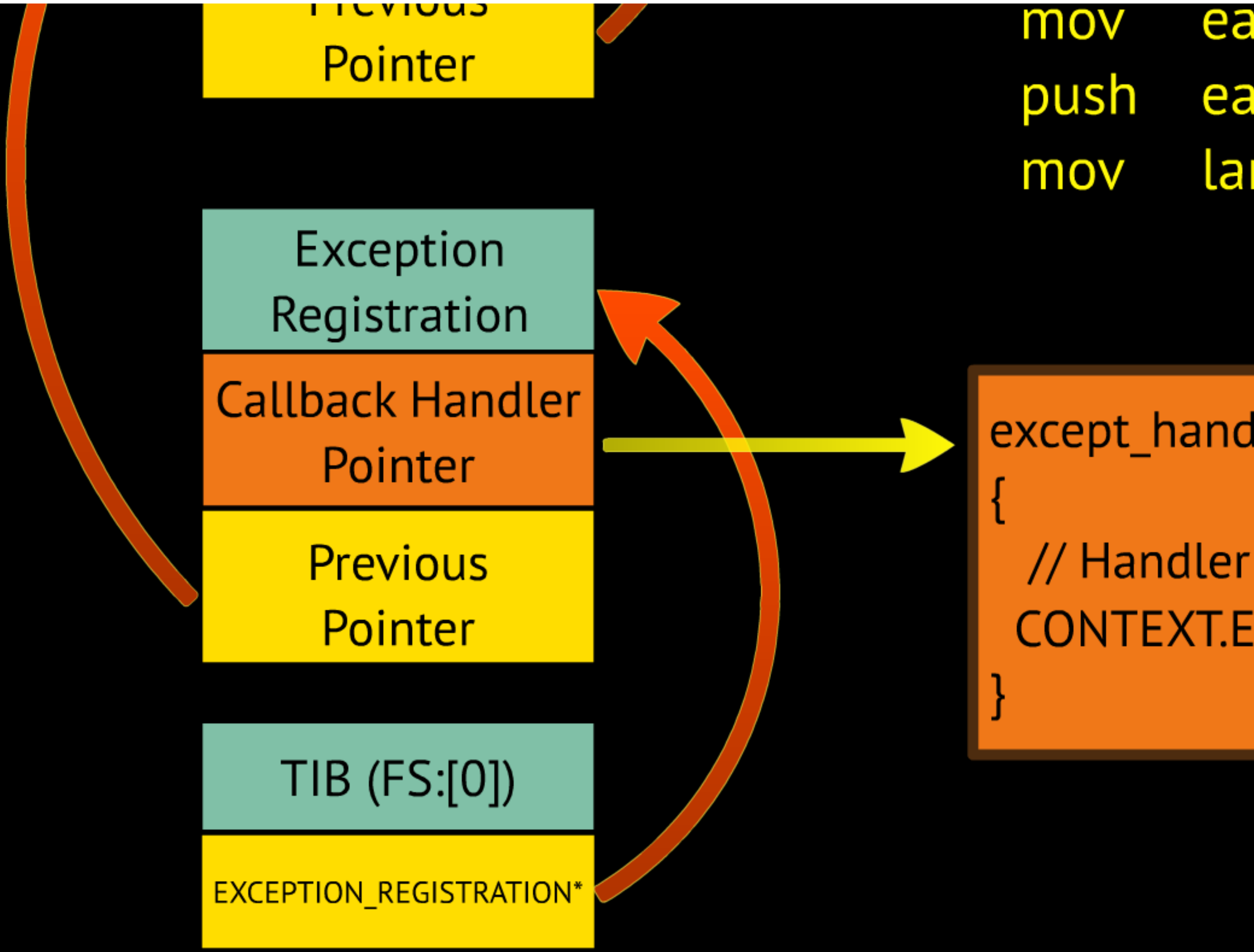
Back to my beloved Malware ...
Visual C++ Exception Handling

- On Top of SEH
- Every Function has one dedicated EH
- These call into _CxxFrameHandler
- FuncInfo Data Structure says what to do
- Handler defines where to continue

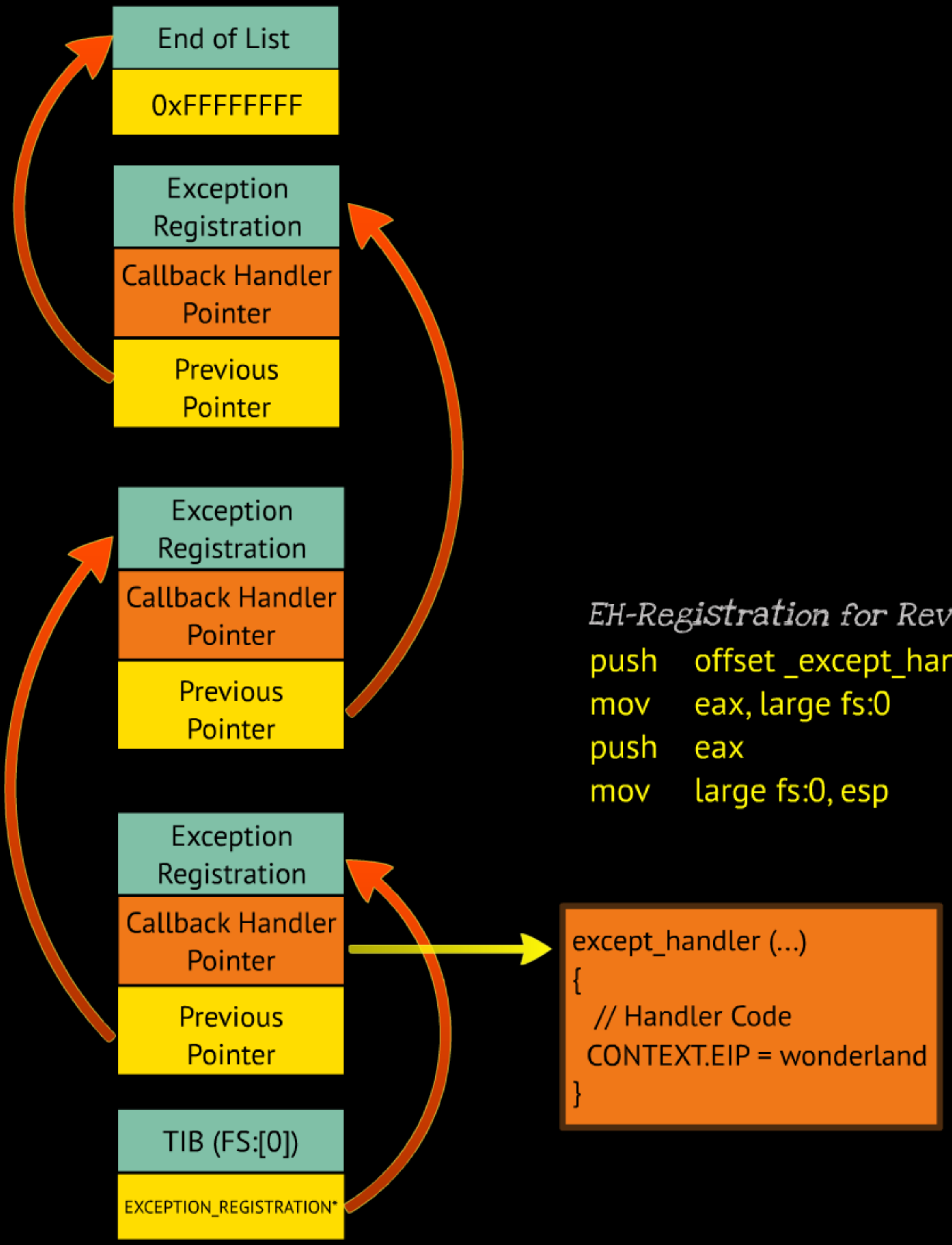
Anti-Analysis:
Auto Junk Code

Summary of a Crash Course on the Depths of Win32 Structured Exception Handling

Long live Matt Pietrek!



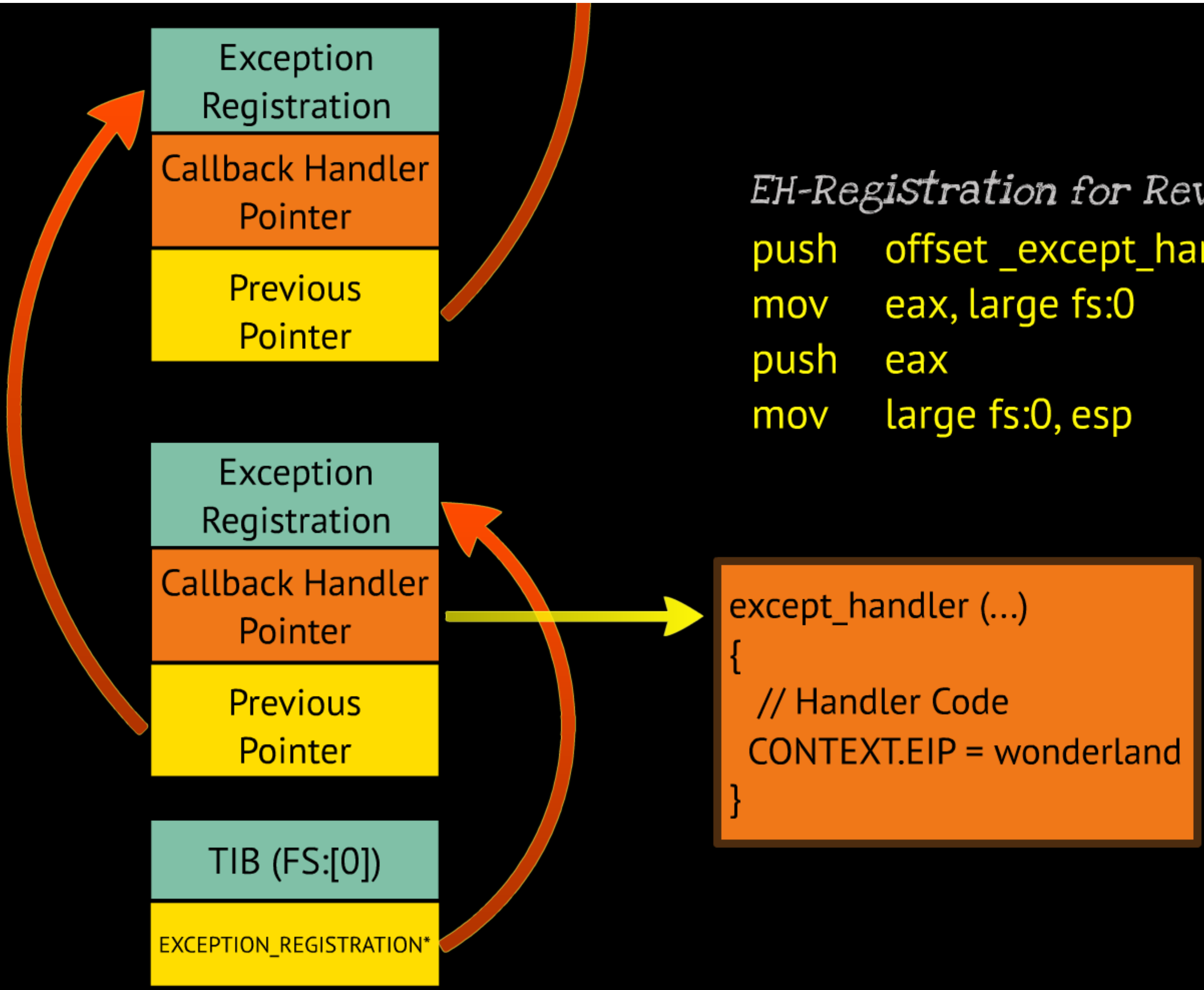
```
mov     ea
push    ea
mov     la
```



```
except_handler (...)
{
  // Handler Code
 CONTEXT.EIP = wonderland
}
```

EH-Registration for Reversers:

```
push    offset _except_handler
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
```



EH-Registration for Reversers:

```
push    offset _except_handler
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
```

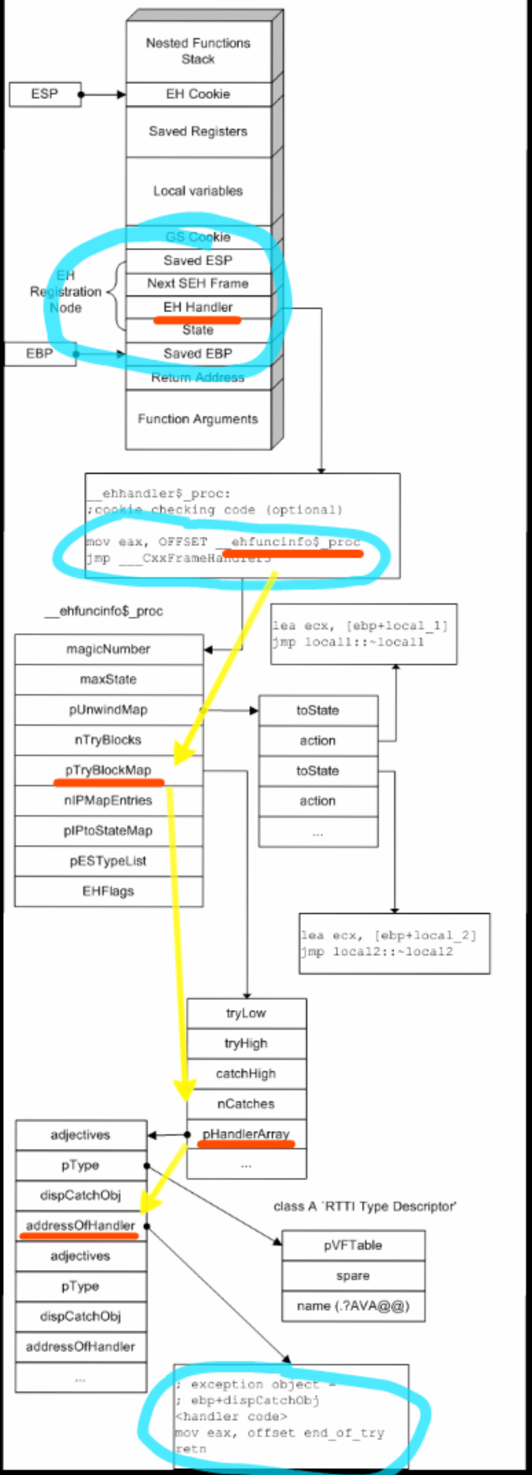
```
except_handler (...)
{
   // Handler Code
  CONTEXT.EIP = wonderland
}
```

Back to my beloved Malware ...

Visual C++ Exception Handling

- On Top of SEH
- Every Function has one dedicated EH
- These call into _CxxFrameHandler
- FuncInfo Data Structure says what to do
- Handler defines where to continue

C++ EH Stack Layout



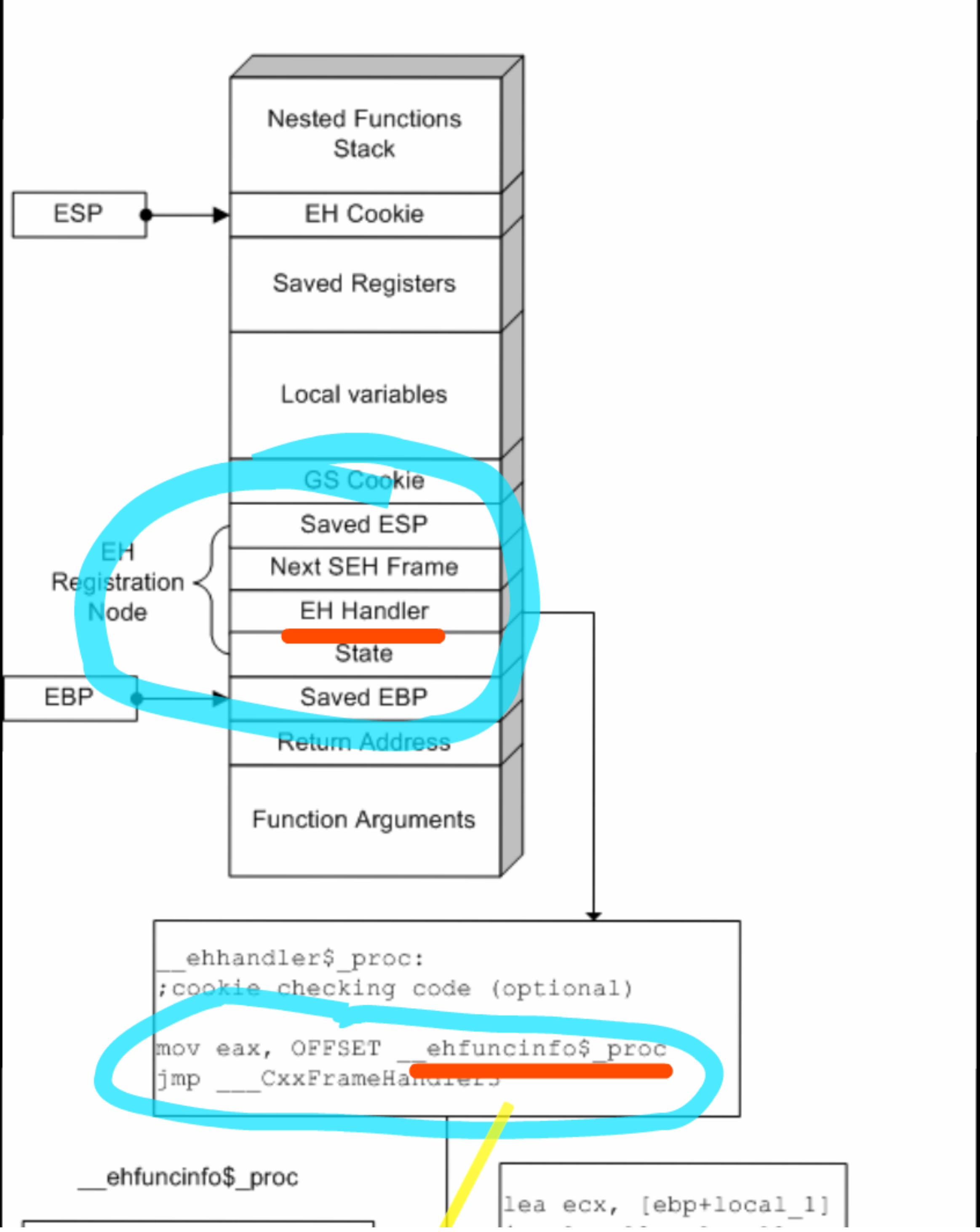
```
__ehhandler$_proc:
;cookie checking code (optional)

mov eax, OFFSET __ehfuncinfo$_proc
jmp ___CxxFrameHandler3
```

```
lea ecx, [ebp+local_1]
jmp local1::~local1
```

```
lea ecx, [ebp+local_2]
jmp local2::~local2
```

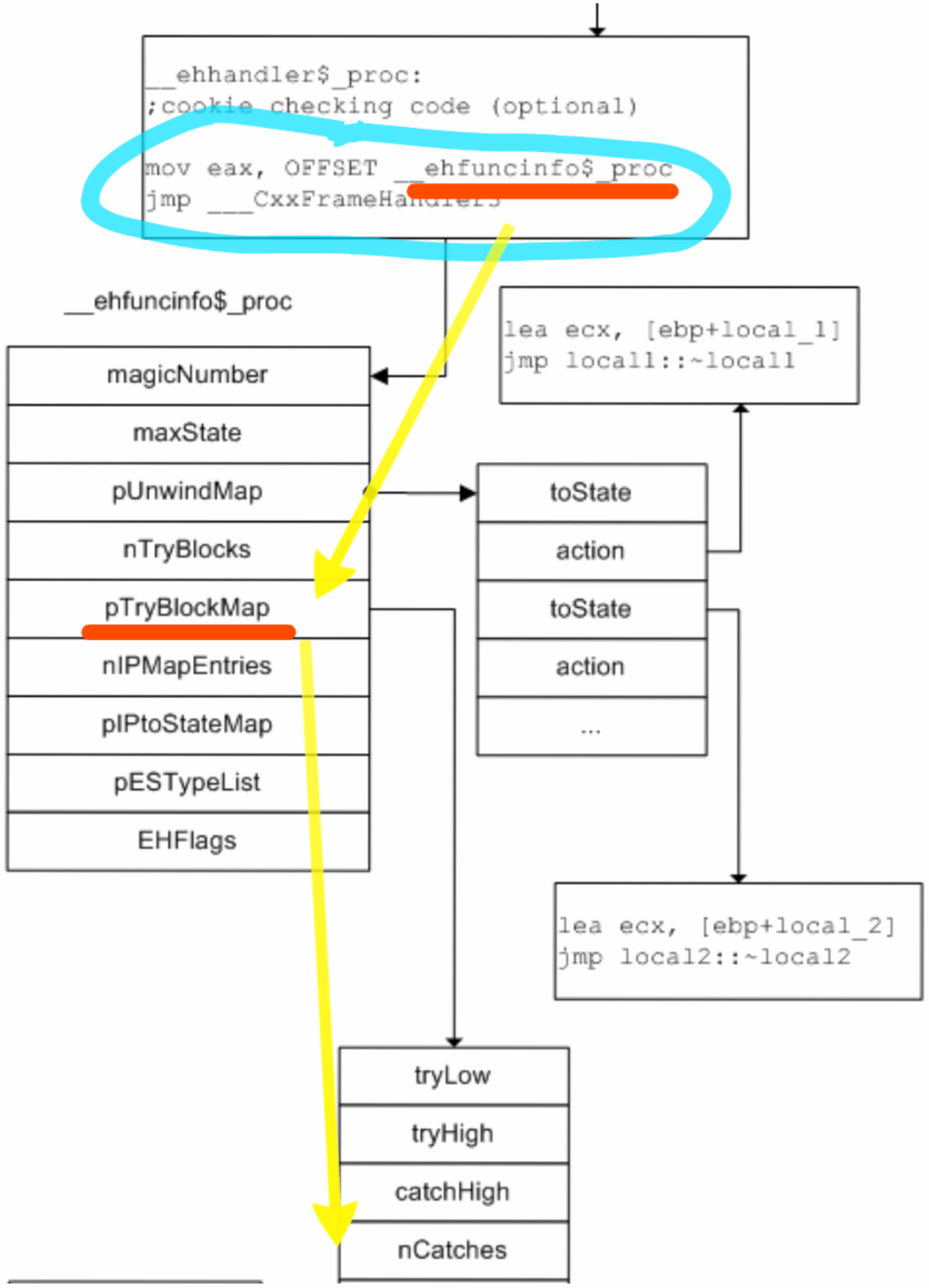
```
; exception object =
; ebp+dispCatchObj
<handler code>
mov eax, offset end_of_try
retn
```

C++ EH Stack Layout



```
__ehhandler$_proc:
;cookie checking code (optional)

mov eax, OFFSET __ehfuncinfo$_proc
jmp ___CxxFrameHandler3
```

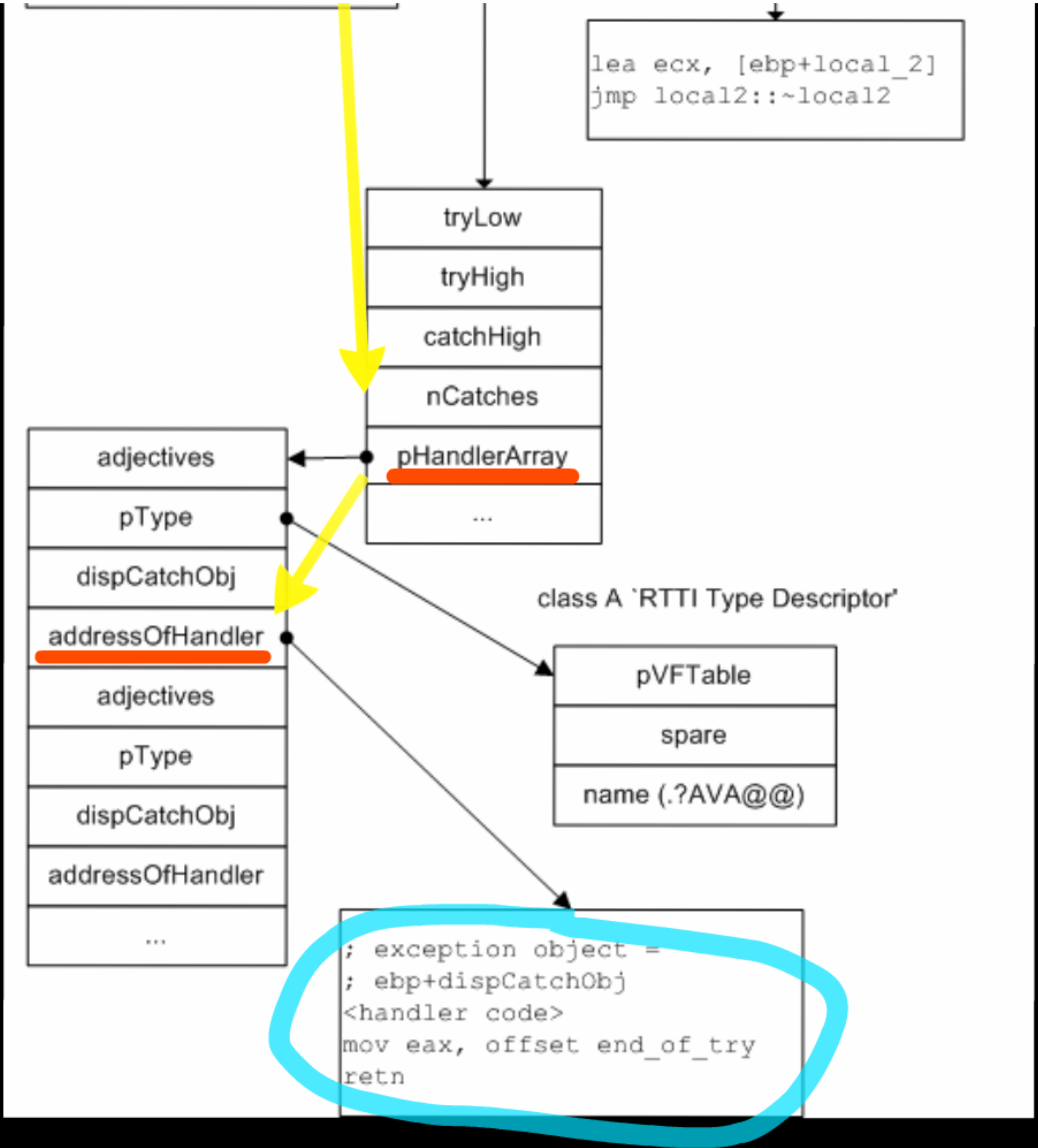
```
lea ecx, [ebp+local_1]
```



```
__ehhandler$_proc:
;cookie checking code (optional)
mov eax, OFFSET __ehfuncinfo$_proc
jmp ___CxxFrameHandler3
```

```
lea ecx, [ebp+local_1]
jmp local1::~local1
```

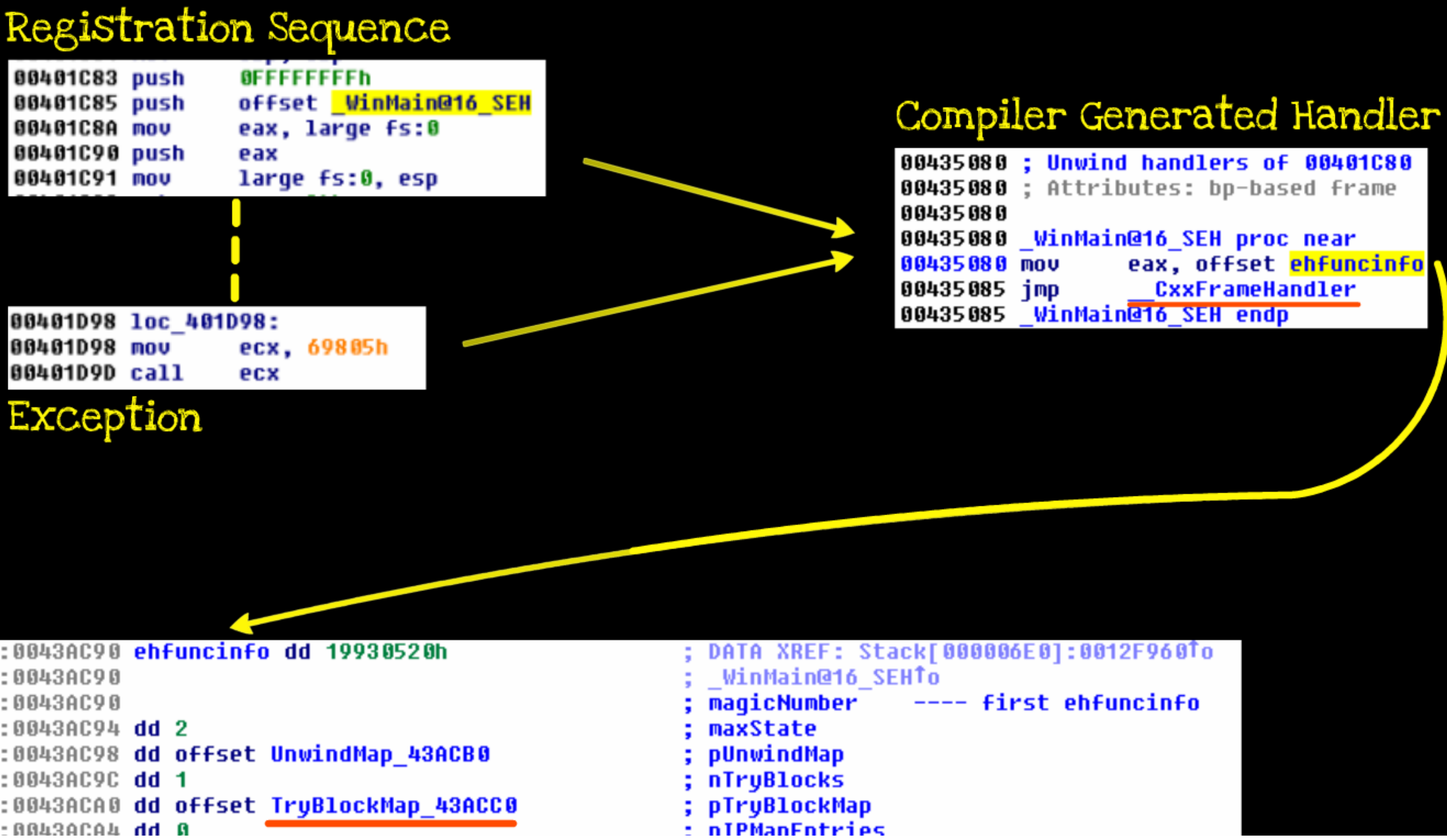
```
lea ecx, [ebp+local_2]
jmp local2::~local2
```



```
lea ecx, [ebp+local_2]
jmp local2::~local2
```

```
; exception object =
; ebp+dispCatchObj
<handler code>
mov eax, offset end_of_try
retn
```

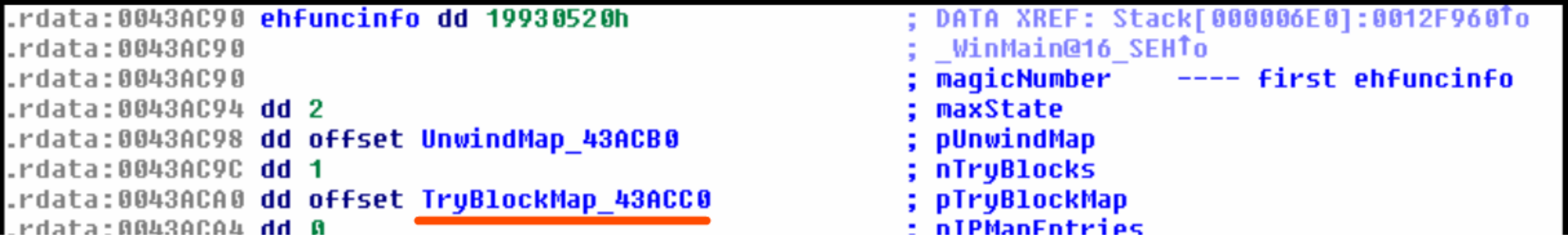
In Practice



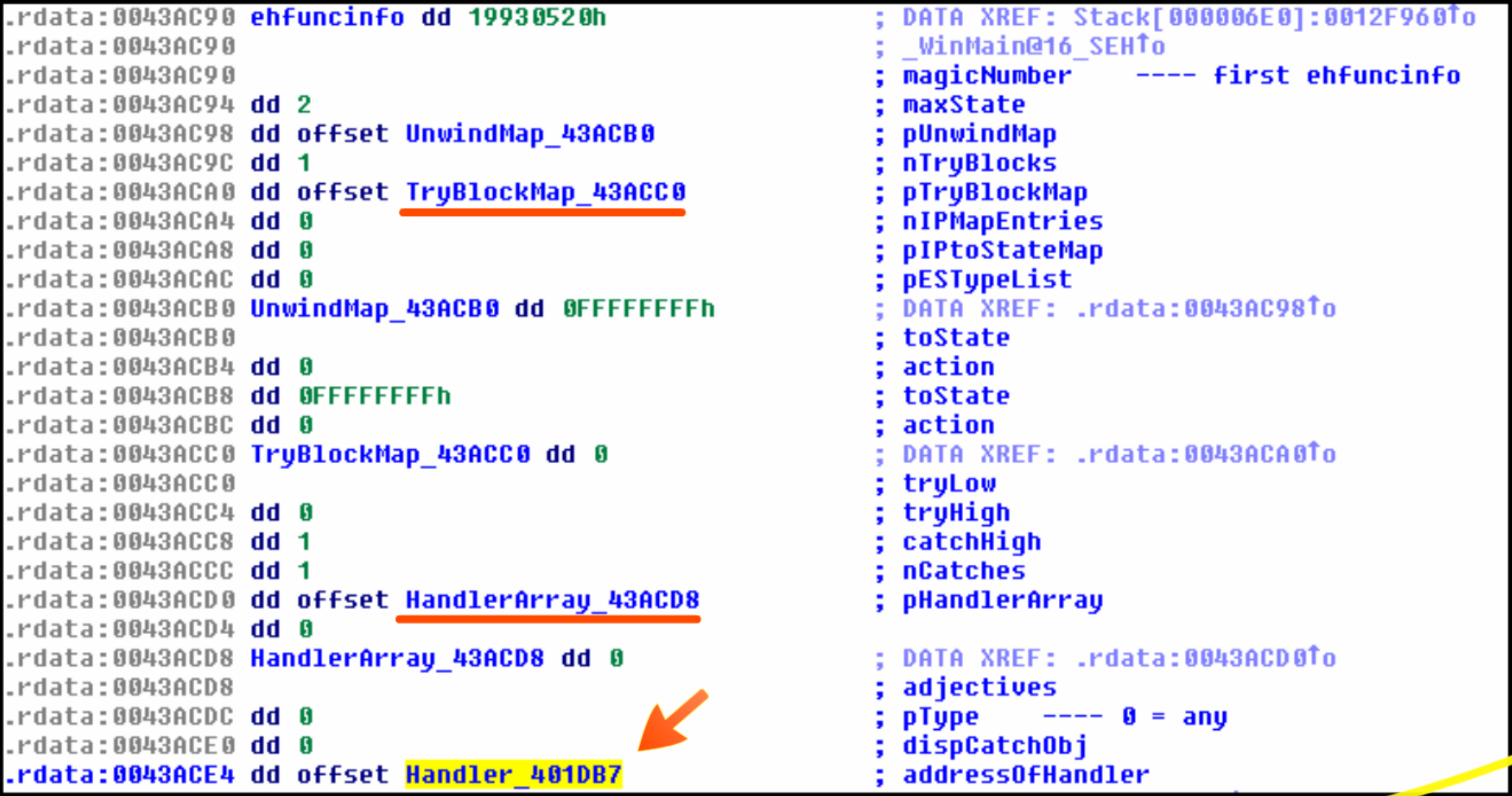
```
00401C83 push    0FFFFFFFFh
00401C85 push    offset _WinMain@16_SEH
00401C8A mov     eax, large fs:0
00401C90 push    eax
00401C91 mov     large fs:0, esp
```

```
00401D98 loc_401D98:
00401D98 mov     ecx, 69805h
00401D9D call    ecx
```

```
00435080 ; Unwind handlers of 00401C80
00435080 ; Attributes: bp-based frame
00435080
00435080 _WinMain@16_SEH proc near
00435080 mov     eax, offset ehfuncinfo
00435085 jmp     __CxxFrameHandler
00435085 _WinMain@16_SEH endp
```



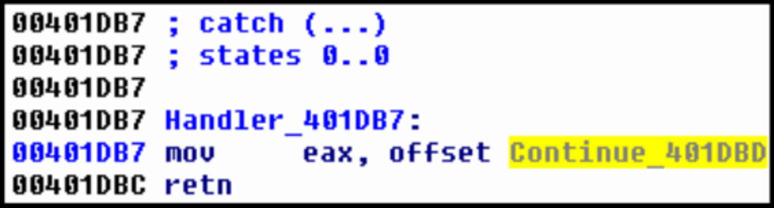
```
.rdata:0043AC90 ehfuncinfo dd 19930520h          ; DATA XREF: Stack[000006E0]:0012F960↑o
.rdata:0043AC90                                  ; _WinMain@16_SEH↑o
.rdata:0043AC90                                  ; magicNumber    ---- first ehfuncinfo
.rdata:0043AC94 dd 2                             ; maxState
.rdata:0043AC98 dd offset UnwindMap_43ACB0       ; pUnwindMap
.rdata:0043AC9C dd 1                             ; nTryBlocks
.rdata:0043ACA0 dd offset TryBlockMap_43ACC0     ; pTryBlockMap
.rdata:0043ACA4 dd 0                             ; nIPMapEntries
```



```
.rdata:0043AC90 ehfuncinfo dd 19930520h                   ; DATA XREF: Stack[000006E0]:0012F960↑o
.rdata:0043AC90                                           ; _WinMain@16_SEH↑o
.rdata:0043AC90                                           ; magicNumber    ---- first ehfuncinfo
.rdata:0043AC94 dd 2                                      ; maxState
.rdata:0043AC98 dd offset UnwindMap_43ACB0                ; pUnwindMap
.rdata:0043AC9C dd 1                                      ; nTryBlocks
.rdata:0043ACA0 dd offset TryBlockMap_43ACC0              ; pTryBlockMap
.rdata:0043ACA4 dd 0                                      ; nIPMapEntries
.rdata:0043ACA8 dd 0                                      ; pIPtoStateMap
.rdata:0043ACAC dd 0                                      ; pESTypeList
.rdata:0043ACB0 UnwindMap_43ACB0 dd 0FFFFFFFFh            ; DATA XREF: .rdata:0043AC98↑o
.rdata:0043ACB0                                           ; toState
.rdata:0043ACB4 dd 0                                      ; action
.rdata:0043ACB8 dd 0FFFFFFFFh                             ; toState
.rdata:0043ACBC dd 0                                      ; action
.rdata:0043ACC0 TryBlockMap_43ACC0 dd 0                   ; DATA XREF: .rdata:0043ACA0↑o
.rdata:0043ACC0                                           ; tryLow
.rdata:0043ACC4 dd 0                                      ; tryHigh
.rdata:0043ACC8 dd 1                                      ; catchHigh
.rdata:0043ACCC dd 1                                      ; nCatches
.rdata:0043ACD0 dd offset HandlerArray_43ACD8             ; pHandlerArray
.rdata:0043ACD4 dd 0
.rdata:0043ACD8 HandlerArray_43ACD8 dd 0                  ; DATA XREF: .rdata:0043ACD0↑o
.rdata:0043ACD8                                           ; adjectives
.rdata:0043ACDC dd 0                                      ; pType    ---- 0 = any
.rdata:0043ACE0 dd 0                                      ; dispCatchObj
.rdata:0043ACE4 dd offset Handler_401DB7                  ; addressOfHandler
```

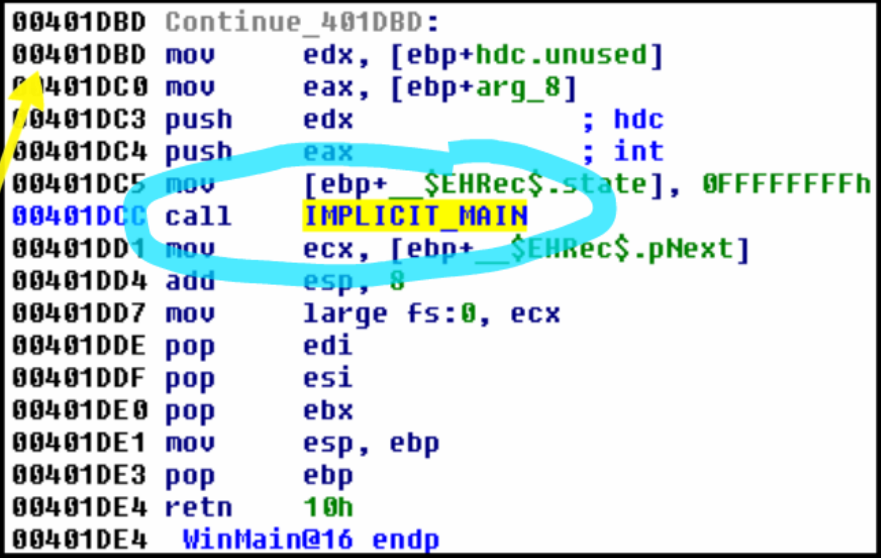
```
960fo

info
```

User Generated Handler



```
00401DB7 ; catch (...)
00401DB7 ; states 0..0
00401DB7
00401DB7 Handler_401DB7:
00401DB7 mov     eax, offset Continue_401DBD
00401DBC retn
```

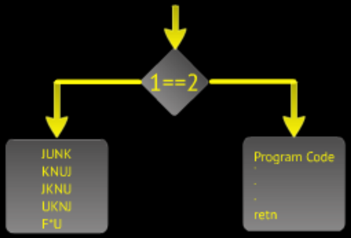
New Entry Point



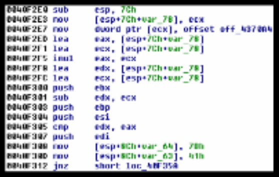
```
00401DBD Continue_401DBD:
00401DBD mov     edx, [ebp+hdc.unused]
00401DC0 mov     eax, [ebp+arg_8]
00401DC3 push    edx                 ; hdc
00401DC4 push    eax                 ; int
00401DC5 mov     [ebp+__$EHRec$.state], 0FFFFFFFFh
00401DCC call    IMPLICIT_MAIN
00401DD1 mov     ecx, [ebp+__$EHRec$.pNext]
00401DD4 add     esp, 8
00401DD7 mov     large fs:0, ecx
00401DDE pop     edi
00401DDF pop     esi
00401DE0 pop     ebx
00401DE1 mov     esp, ebp
00401DE3 pop     ebp
00401DE4 retn    10h
00401DE4  WinMain@16 endp
```

Anti-Analysis: Auto Junk Code

Opaque Predicates

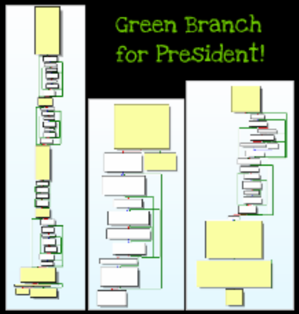


Slightly Obfuscated Opaque Predicates



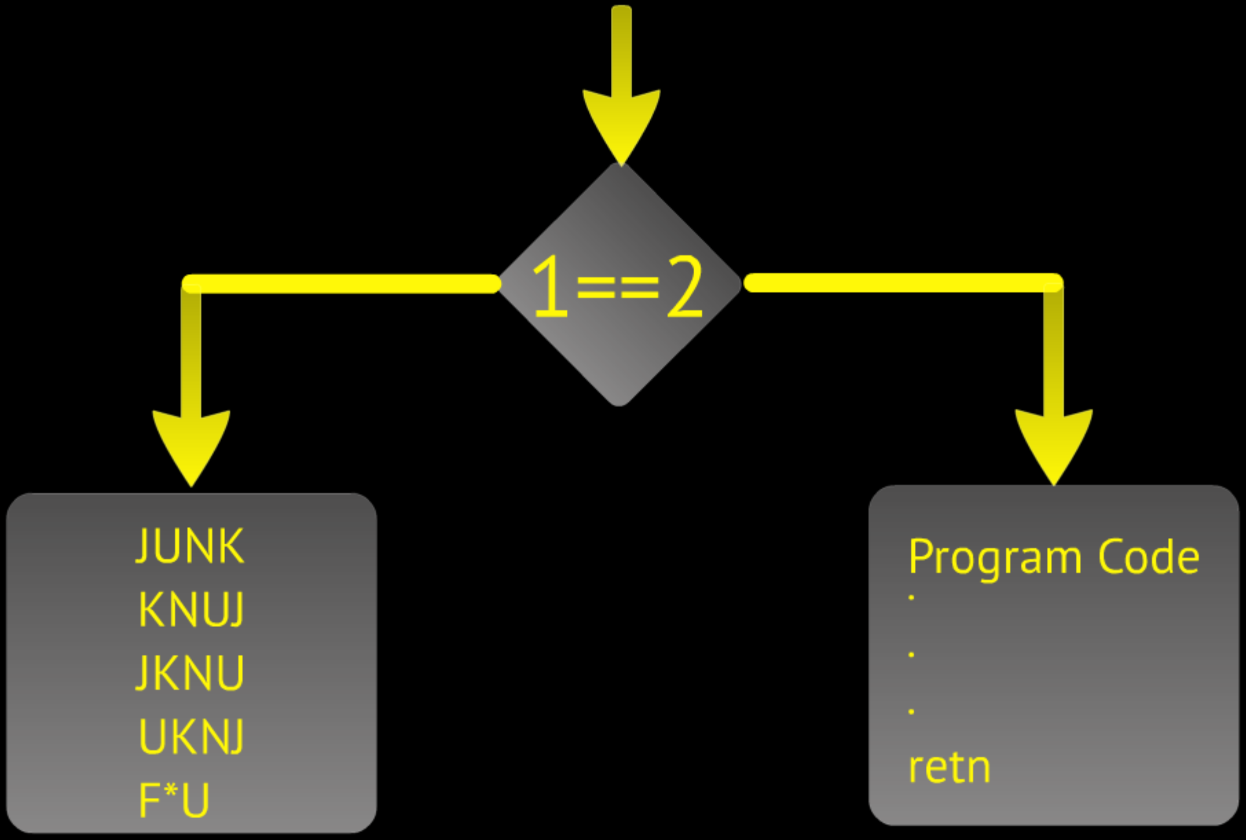
```
0040F2E0 sub     esp, 7Ch
0040F2E3 mov     [esp+7Ch+var_78], ecx
0040F2E7 mov     dword ptr [ecx], offset off_437004
0040F2ED lea     eax, [esp+7Ch+var_78]
0040F2F1 lea     ecx, [esp+7Ch+var_78]
0040F2F5 imul    eax, ecx
0040F2F8 lea     edx, [esp+7Ch+var_78]
0040F2FC lea     ecx, [esp+7Ch+var_78]
0040F300 push    ebx
0040F301 sub     edx, ecx
0040F303 push    ebp
0040F304 push    esi
0040F305 cmp     edx, eax
0040F307 push    edi
0040F308 mov     [esp+8Ch+var_64], 70h
0040F30D mov     [esp+8Ch+var_63], 41h
0040F312 jnz     short loc_40F35A
```

```
.text:0040F2E3 mov   [esp+7Ch+var_78], ecx
.text:0040F2ED lea    eax, [esp+7Ch+var_78]
.text:0040F2F1 lea   ecx, [esp+7Ch+var_78]
.text:0040F2F5 imul  eax, ecx
.text:0040F2F8 lea   edx, [esp+7Ch+var_78]
.text:0040F2FC lea   ecx, [esp+7Ch+var_78]
.text:0040F301 sub   edx, ecx
.text:0040F305 cmp   edx, eax
.text:0040F312 jnz   short loc_40F35A
```

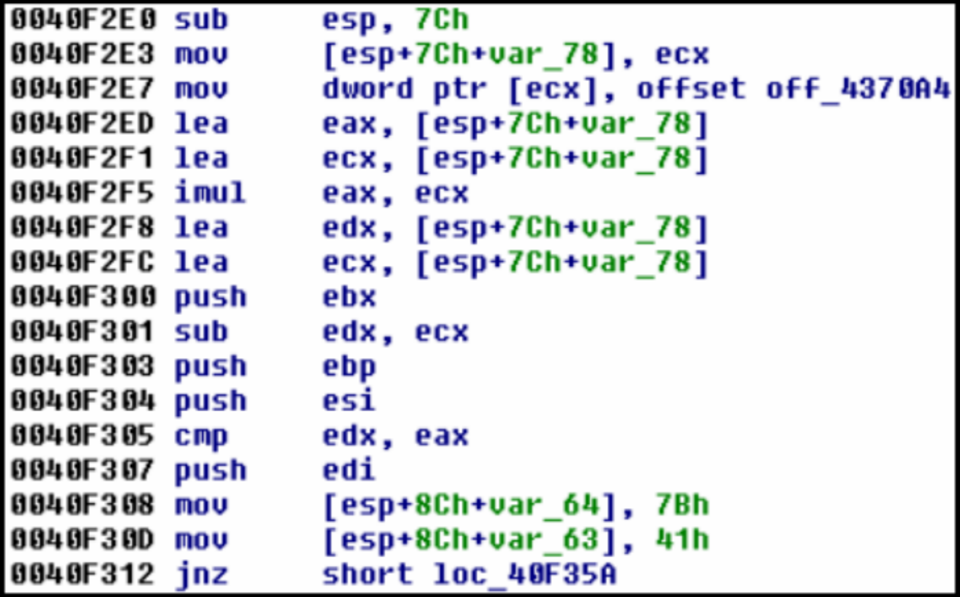


Green Branch
for President!

Opaque Predicates



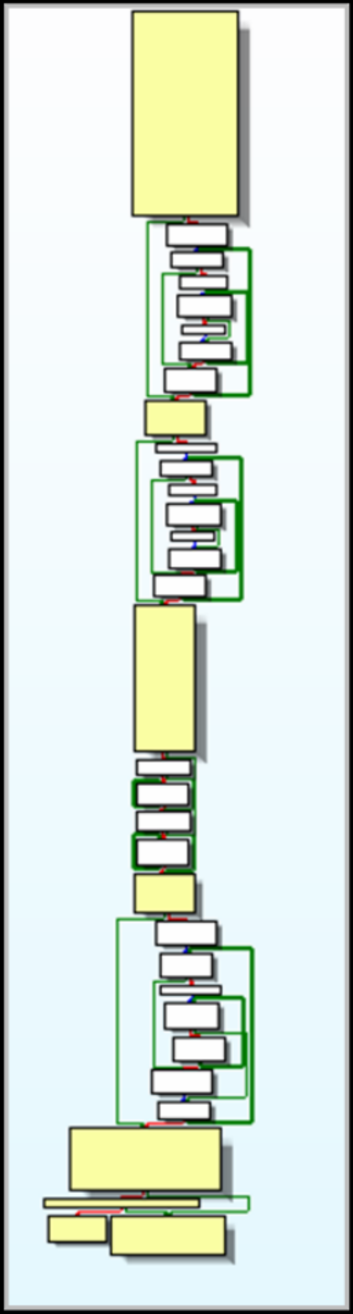
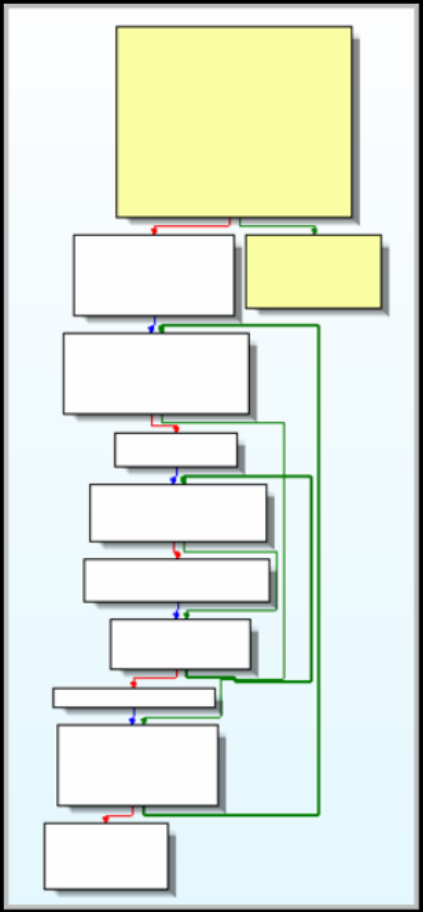
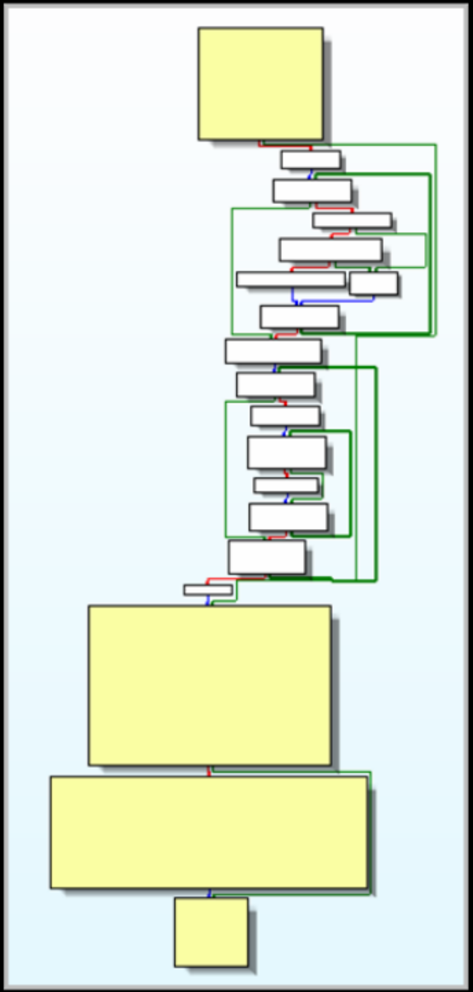
Slightly Obfuscated Opaque Predicates



```
0040F2E0 sub     esp, 7Ch
0040F2E3 mov     [esp+7Ch+var_78], ecx
0040F2E7 mov     dword ptr [ecx], offset off_4370A4
0040F2ED lea     eax, [esp+7Ch+var_78]
0040F2F1 lea     ecx, [esp+7Ch+var_78]
0040F2F5 imul    eax, ecx
0040F2F8 lea     edx, [esp+7Ch+var_78]
0040F2FC lea     ecx, [esp+7Ch+var_78]
0040F300 push    ebx
0040F301 sub     edx, ecx
0040F303 push    ebp
0040F304 push    esi
0040F305 cmp     edx, eax
0040F307 push    edi
0040F308 mov     [esp+8Ch+var_64], 7Bh
0040F30D mov     [esp+8Ch+var_63], 41h
0040F312 jnz     short loc_40F35A
```

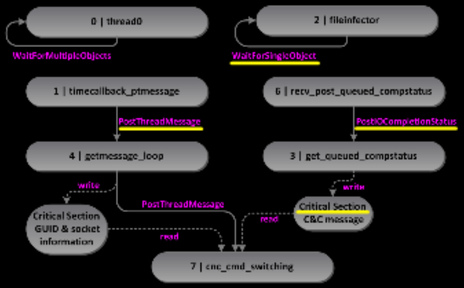
```
.text:0040F2E3 mov   [esp+7Ch+var_78], ecx
.text:0040F2ED lea   eax, [esp+7Ch+var_78]
.text:0040F2F1 lea   ecx, [esp+7Ch+var_78]
.text:0040F2F5 imul  eax, ecx
.text:0040F2F8 lea   edx, [esp+7Ch+var_78]
.text:0040F2FC lea   ecx, [esp+7Ch+var_78]
.text:0040F301 sub   edx, ecx
.text:0040F305 cmp   edx, eax
.text:0040F312 jnz   short loc_40F35A
```

Green Branch for President!

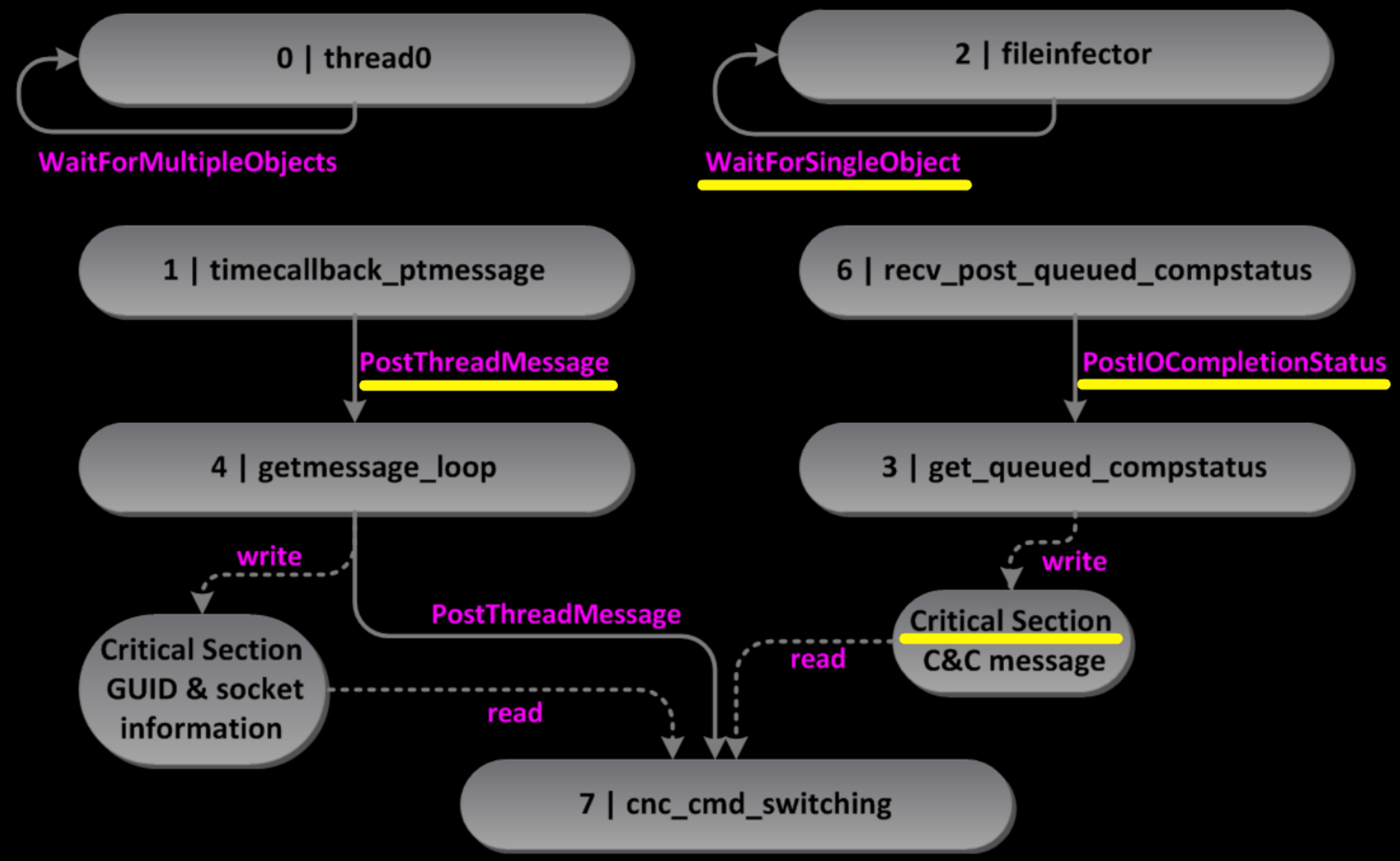
Analysts' Headaches

Thread Me To Hell



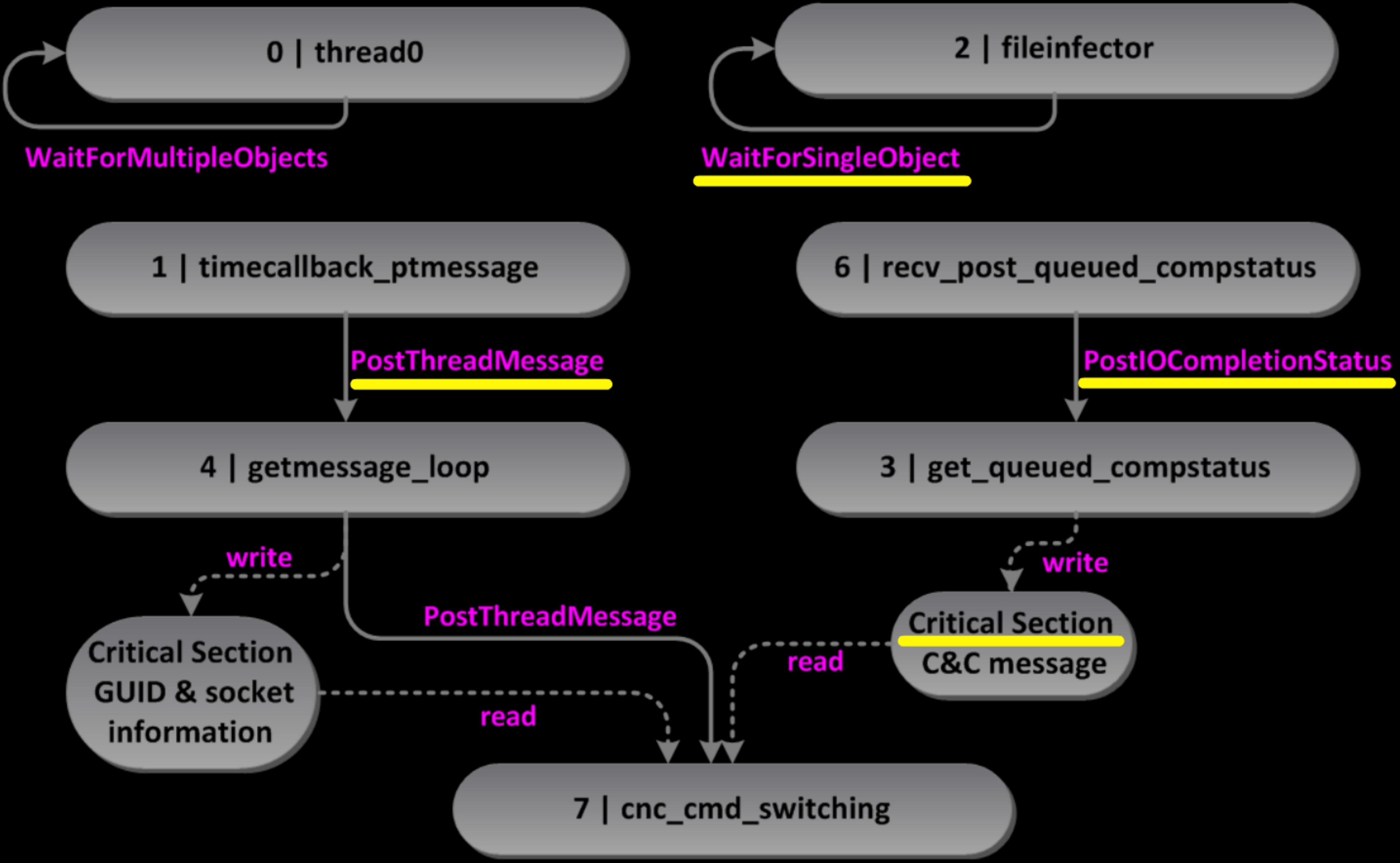
How To Get There

1. Realize there are multiple threads that you have to follow
2. Spot inter-thread communication & synchronization
3. Analyze function bodies with significant functionality
4. Bring down what information is exchanged between threads and how one thread influences the other


0 | thread0
WaitForMultipleObjects
2 | fileinfector
WaitForSingleObject
1 | timecallback_ptmessage
PostThreadMessage
6 | recv_post_queued_compstatus
PostIOCompletionStatus
4 | getmessage_loop
3 | get_queued_compstatus
write
write
PostThreadMessage
Critical Section
C&C message
read
Critical Section
GUID & socket
information
read
7 | cnc_cmd_switching

How To Get There

1. Realize there are multiple threads that you have to follow
2. Spot inter-thread communication & synchronization
3. Analyze function bodies with significant functionality
4. Bring down what information is exchanged between threads and how one thread influences the other

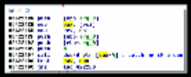

0 | thread0
WaitForMultipleObjects
2 | fileinfector
WaitForSingleObject
1 | timecallback_ptmessage
PostThreadMessage
6 | recv_post_queued_compstatus
PostIOCompletionStatus
4 | getmessage_loop
3 | get_queued_compstatus
write
write
PostThreadMessage
Critical Section
C&C message
read
Critical Section
GUID & socket
information
read
7 | cnc_cmd_switching

C++ or:

Function Calls to Nirvana

C++

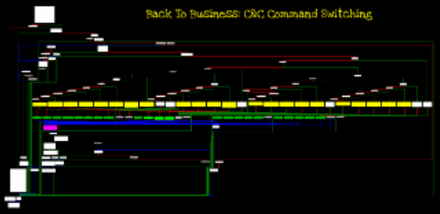
Multiple inheritance
Indirect calls
Binary overhead for "glue code"
Non-linear code
Few documentation for reversers



Special credits to Igor Skochinski & OpenRCE

```
class A                      class B                      class C: public A, public B
{                            {                            {
  int a1;                      int b1;                      int c1;
public:                        int b2;                    public:
  virtual int A_virt1();     public:                        virtual int A_virt2();
  virtual int A_virt2();       virtual int B_virt1();       virtual int B_virt2();
  static void A_static1();     virtual int B_virt2();     };
  void A_simple1();          };
};                                                        class C size(24):
                                                              +---
class A size(8):             class B size(12):                | +--- (base class A)
    +---                         +---                      0  | | {vfptr}
 0  | {vfptr}                 0  | {vfptr}                 4  | | a1
 4  | a1                      4  | b1                         | +---
    +---                      8  | b2                         | +--- (base class B)
                                 +---                      8  | | {vfptr}
A's vftable:                                              12  | | b1
 0  | &A::A_virt1            B's vftable:                 16  | | b2
 4  | &A::A_virt2             0  | &B::B_virt1                | +---
                              4  | &B::B_virt2            20  | c1
                                                              +---

C's vftable for A:
 0  | &A::A_virt1
 4  | &C::A_virt2

C's vftable for B:
 0  | &B::B_virt1
 4  | &C::B_virt2
```

Back To Business: C&C Command Switching



Command: move_file

C++

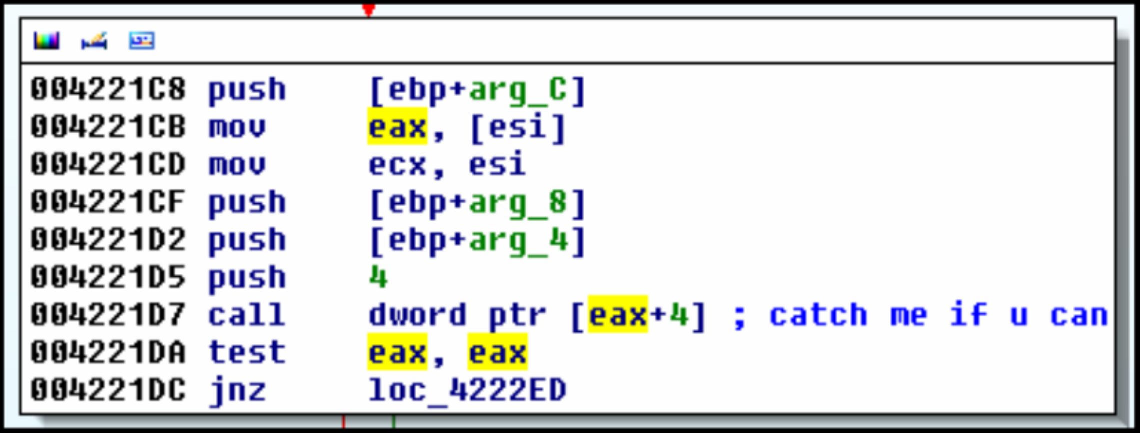
Multiple inheritance

Indirect calls

Binary overhead for "glue code"

Non-linear code

Few documentation for reversers



```
004221C8 push    [ebp+arg_C]
004221CB mov     eax, [esi]
004221CD mov     ecx, esi
004221CF push    [ebp+arg_8]
004221D2 push    [ebp+arg_4]
004221D5 push    4
004221D7 call    dword ptr [eax+4] ; catch me if u can
004221DA test    eax, eax
004221DC jnz     loc_4222ED
```

Special credits to Igor Skochinski & OpenRCE

```
class A
{
  int a1;
public:
  virtual int A_virt1();
  virtual int A_virt2();
  static void A_static1();
  void A_simple1();
};

class A size(8):
    +---
 0  | {vfptr}
 4  | a1
    +---

A's vftable:
 0  | &A::A_virt1
 4  | &A::A_virt2
```

```
class B
{
  int b1;
  int b2;
public:
  virtual int B_virt1();
  virtual int B_virt2();
};

class B size(12):
    +---
 0  | {vfptr}
 4  | b1
 8  | b2
    +---

B's vftable:
 0  | &B::B_virt1
 4  | &B::B_virt2
```

```
class C: public A, public B
{
  int c1;
public:
  virtual int A_virt2();
  virtual int B_virt2();
};

class C size(24):
     +---
     | +--- (base class A)
 0   | | {vfptr}
 4   | | a1
     | +---
     | +--- (base class B)
 8   | | {vfptr}
12   | | b1
16   | | b2
     | +---
20   | c1
```

```
C's v
 0  |
 4  |

C's v
 0  |
 4  |
class B
{
 int b1;
 int b2;
public:
 virtual int B_virt1();
 virtual int B_virt2();
};
```

```
class B size(12):
   +---
 0  | {vfptr}
 4  | b1
 8  | b2
   +---
```

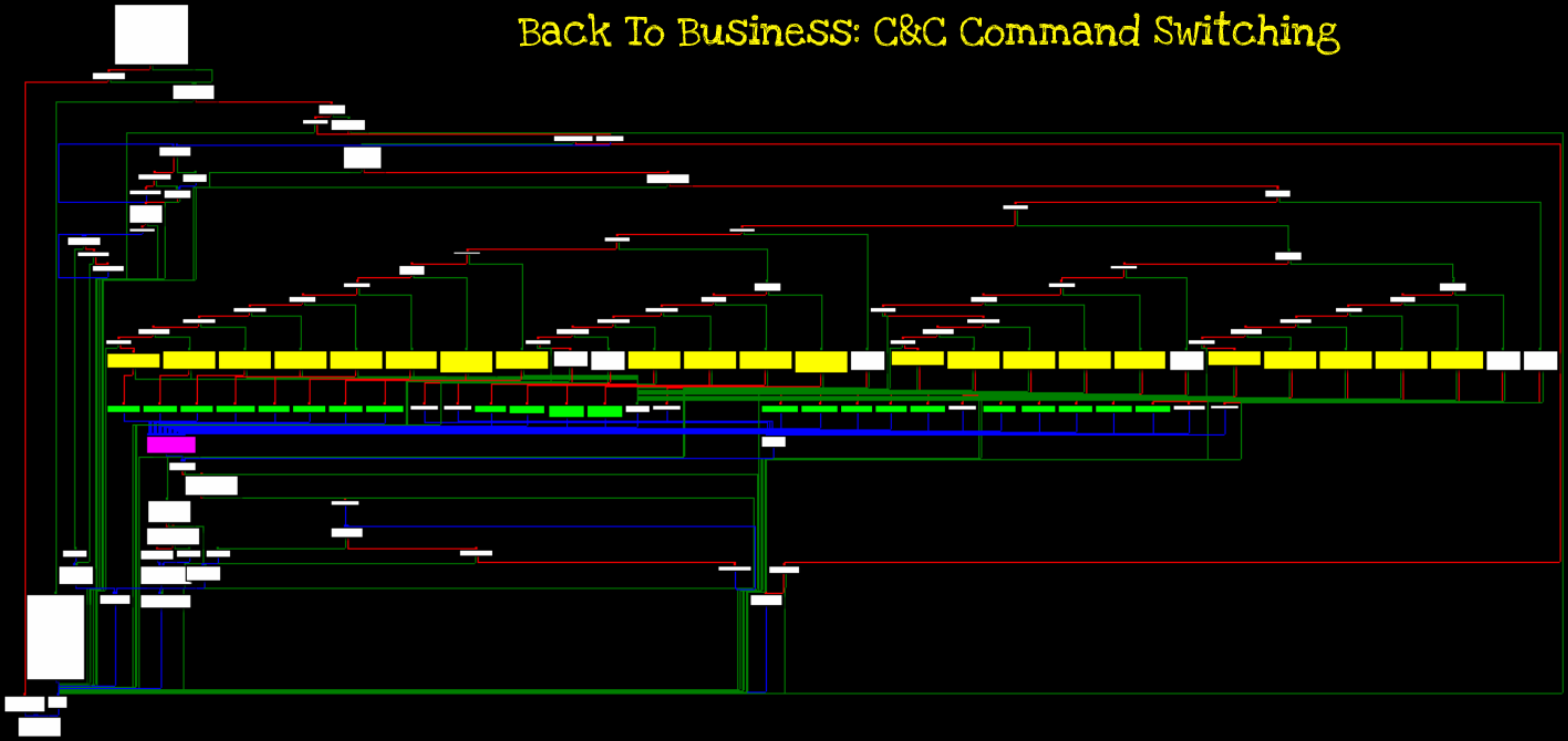
```
B's vftable:
 0  | &B::B_virt1
 4  | &B::B_virt2
```

```
class C: public A, public B
{
 int c1;
public:
 virtual int A_virt2();
 virtual int B_virt2();
};
```

```
class C size(24):
    +---
    | +--- (base class A)
 0  | | {vfptr}
 4  | | a1
    | +---
    | +--- (base class B)
 8  | | {vfptr}
12  | | b1
16  | | b2
    | +---
20  | c1
    +---
```

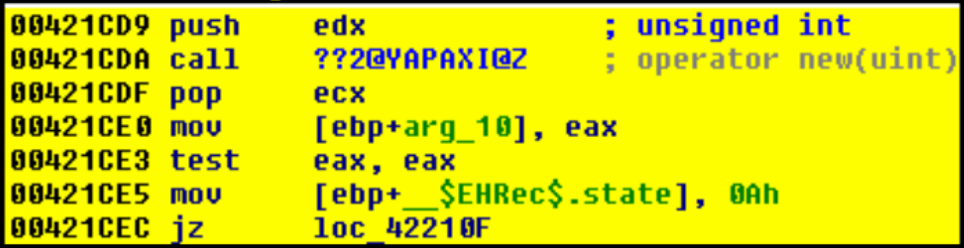
```
C's vftable for A:
 0  | &A::A_virt1
 4  | &C::A_virt2

C's vftable for B:
 0  | &B::B_virt1
 4  | &C::B_virt2
```

Back To Business: C&C Command Switching



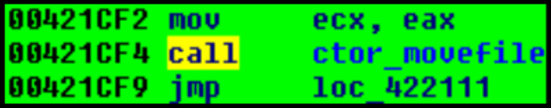
Command: move_file

Memory Allocation



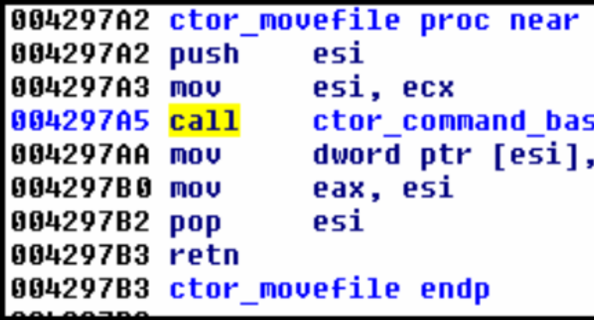
```
00421CD9 push    edx              ; unsigned int
00421CDA call    ??2@YAPAXI@Z     ; operator new(uint)
00421CDF pop     ecx
00421CE0 mov     [ebp+arg_10], eax
00421CE3 test    eax, eax
00421CE5 mov     [ebp+__$EHRec$.state], 0Ah
00421CEC jz      loc_42210F
```

Instantiation

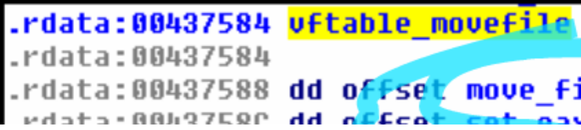


```
00421CF2 mov     ecx, eax
00421CF4 call    ctor_movefile
00421CF9 jmp     loc_422111
```

Constructor



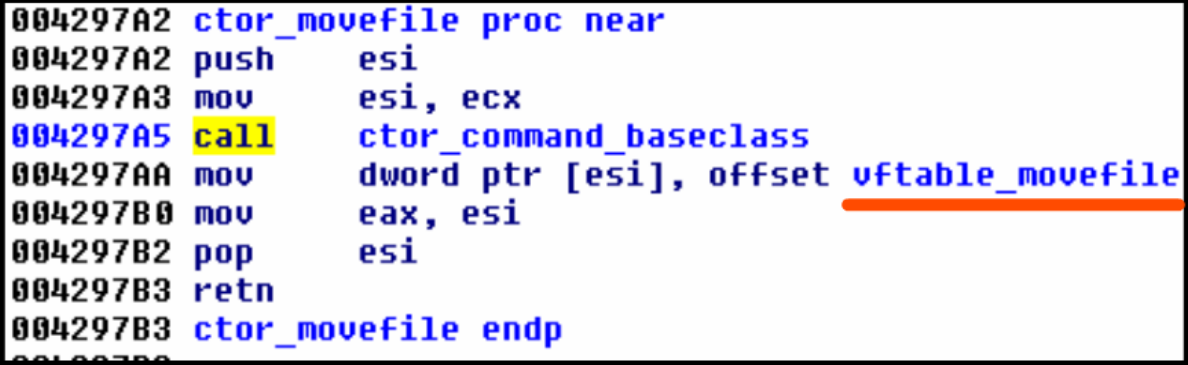
```
004297A2 ctor_movefile proc near
004297A2 push    esi
004297A3 mov     esi, ecx
004297A5 call    ctor_command_bas
004297AA mov     dword ptr [esi],
004297B0 mov     eax, esi
004297B2 pop     esi
004297B3 retn
004297B3 ctor_movefile endp
```



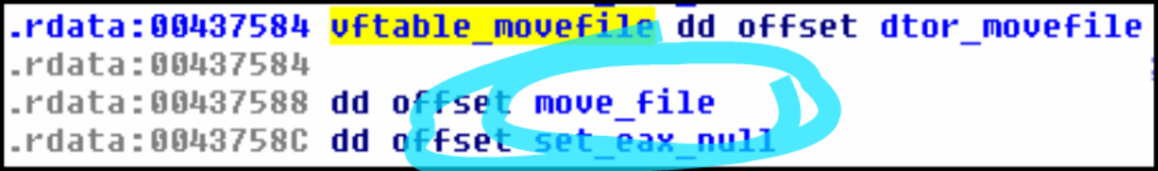
```
.rdata:00437584 vftable_movefile
.rdata:00437584
.rdata:00437588 dd offset move_fi
```

```
x
tate], 0Ah
```

Constructor

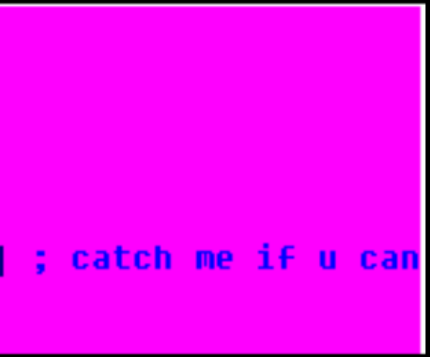


```
004297A2 ctor_movefile proc near
004297A2 push    esi
004297A3 mov     esi, ecx
004297A5 call    ctor_command_baseclass
004297AA mov     dword ptr [esi], offset vftable_movefile
004297B0 mov     eax, esi
004297B2 pop     esi
004297B3 retn
004297B3 ctor_movefile endp
```

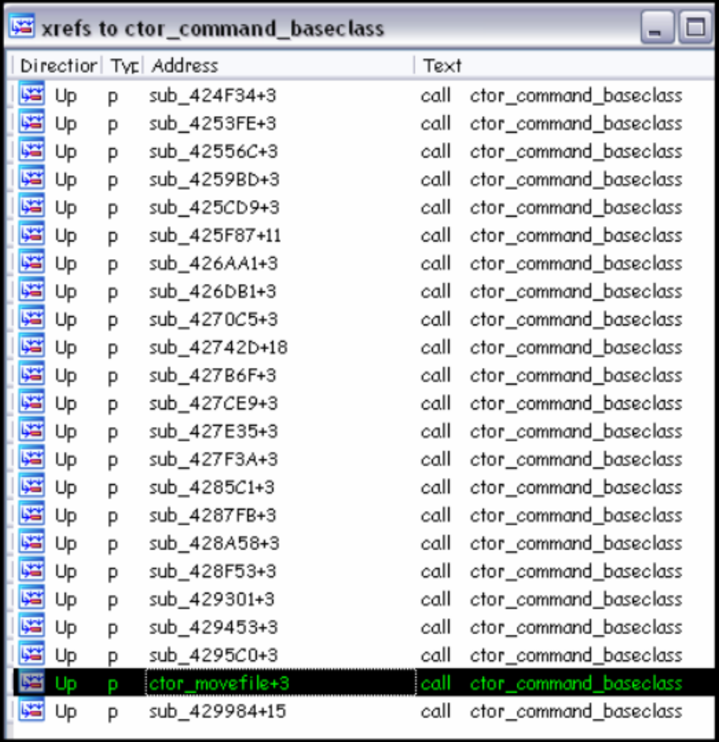


```
.rdata:00437584 vftable_movefile dd offset dtor_movefile
.rdata:00437584
.rdata:00437588 dd offset move_file
.rdata:0043758C dd offset set_eax_null
```

n Call



```
] ; catch me if u can
```

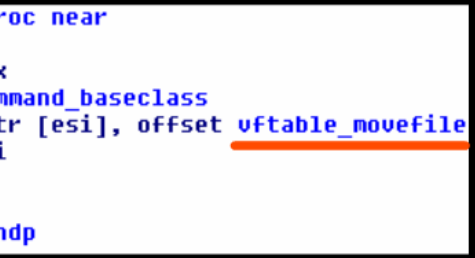


xrefs to ctor_command_baseclass

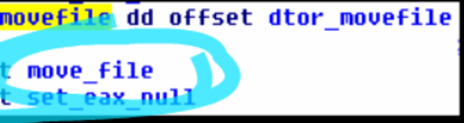
Direction	Typ	Address	Text
Up	p	sub_424F34+3	call ctor_command_baseclass
Up	p	sub_4253FE+3	call ctor_command_baseclass
Up	p	sub_42556C+3	call ctor_command_baseclass
Up	p	sub_4259BD+3	call ctor_command_baseclass
Up	p	sub_425CD9+3	call ctor_command_baseclass
Up	p	sub_425F87+11	call ctor_command_baseclass
Up	p	sub_426AA1+3	call ctor_command_baseclass
Up	p	sub_426DB1+3	call ctor_command_baseclass
Up	p	sub_4270C5+3	call ctor_command_baseclass
Up	p	sub_42742D+18	call ctor_command_baseclass
Up	p	sub_427B6F+3	call ctor_command_baseclass
Up	p	sub_427CE9+3	call ctor_command_baseclass
Up	p	sub_427E35+3	call ctor_command_baseclass
Up	p	sub_427F3A+3	call ctor_command_baseclass
Up	p	sub_4285C1+3	call ctor_command_baseclass
Up	p	sub_4287FB+3	call ctor_command_baseclass
Up	p	sub_428A58+3	call ctor_command_baseclass
Up	p	sub_428F53+3	call ctor_command_baseclass
Up	p	sub_429301+3	call ctor_command_baseclass
Up	p	sub_429453+3	call ctor_command_baseclass
Up	p	sub_4295C0+3	call ctor_command_baseclass
Up	p	ctor_movefile+3	call ctor_command_baseclass
Up	p	sub_429984+15	call ctor_command_baseclass

23 commands,
23 cross references

or

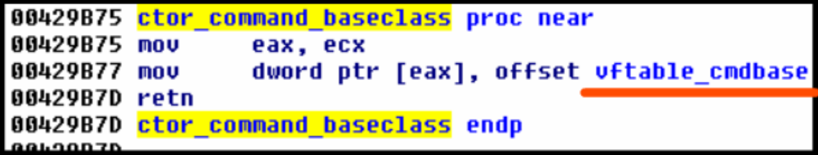


```
roc near

x
mmand_baseclass
tr [esi], offset vftable_movefile
i



ndp
```

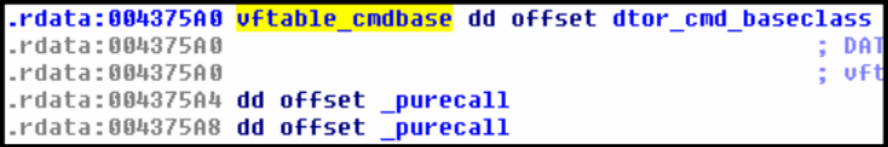


```
movefile dd offset dtor_movefile

t move_file
t set_eax_null
```

Base Class Constructor

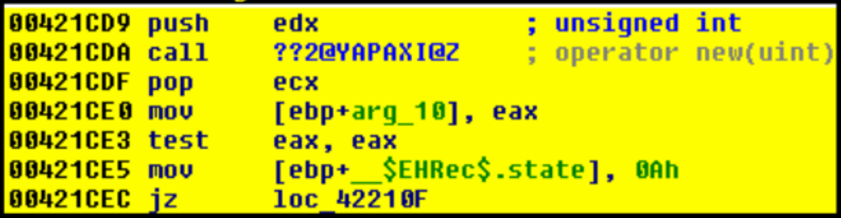


```
00429B75 ctor_command_baseclass proc near
00429B75 mov     eax, ecx
00429B77 mov     dword ptr [eax], offset vftable_cmdbase
00429B7D retn
00429B7D ctor_command_baseclass endp
```



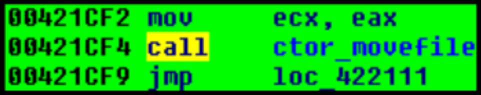
```
.rdata:004375A0 vftable_cmdbase dd offset dtor_cmd_baseclass
.rdata:004375A0                                              ; DAT
.rdata:004375A0                                              ; vft
.rdata:004375A4 dd offset _purecall
.rdata:004375A8 dd offset _purecall
```

Memory Allocation



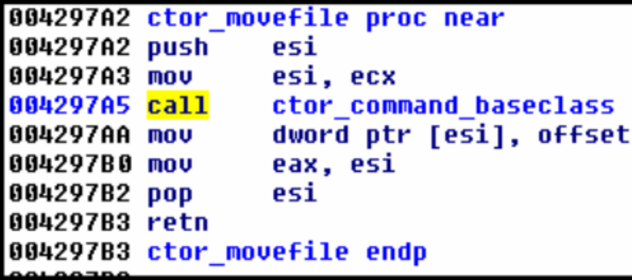
```
00421CD9 push    edx              ; unsigned int
00421CDA call    ??2@YAPAXI@Z     ; operator new(uint)
00421CDF pop     ecx
00421CE0 mov     [ebp+arg_10], eax
00421CE3 test    eax, eax
00421CE5 mov     [ebp+__$EHRec$.state], 0Ah
00421CEC jz      loc_42210F
```

Instantiation

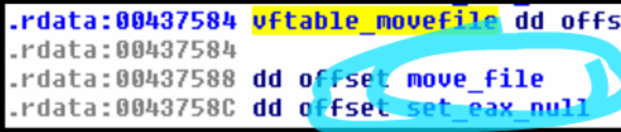


```
00421CF2 mov     ecx, eax
00421CF4 call    ctor_movefile
00421CF9 jmp     loc_422111
```

Constructor

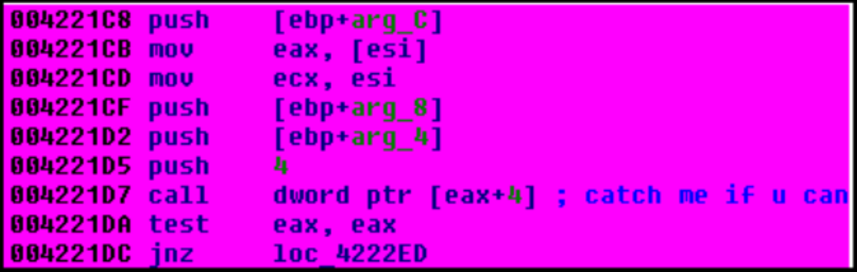


```
004297A2 ctor_movefile proc near
004297A2 push    esi
004297A3 mov     esi, ecx
004297A5 call    ctor_command_baseclass
004297AA mov     dword ptr [esi], offset
004297B0 mov     eax, esi
004297B2 pop     esi
004297B3 retn
004297B3 ctor_movefile endp
```



```
.rdata:00437584 vftable_movefile dd offs
.rdata:00437584
.rdata:00437588 dd offset move_file
.rdata:0043758C dd offset set_eax_null
```

Virtual Function Call



```
004221C8 push    [ebp+arg_C]
004221CB mov     eax, [esi]
004221CD mov     ecx, esi
004221CF push    [ebp+arg_8]
004221D2 push    [ebp+arg_4]
004221D5 push    4
004221D7 call    dword ptr [eax+4] ; catch me if u can
004221DA test    eax, eax
004221DC jnz     loc_4222ED
```

DIY Links

Igor Skochinski
http://www.hexblog.com/wp-content/uploads/2012/06/Recon-2012-Skochinsky-Compiler-Internals.pdf
http://www.openrce.org/articles/full_view/21
http://www.openrce.org/articles/full_view/23

Matt Pietrek
http://www.microsoft.com/msj/0197/Exception/Exception.aspx

Mark Yason & Paul Sabanal
http://www.blackhat.com/presentations/bh-dc-07/Sabanal_Yason/Paper/bh-dc-07-Sabanal_Yason-WP.pdf

Vishal Kochhar
http://www.codeproject.com/Articles/2126/How-a-C-compiler-implements-exception-handling?display=Print

Selvam
http://www.codeproject.com/Articles/7953/Thread-Synchronization-for-Beginners

Josh Haberman
http://blog.reverberate.org/2013/05/deep-wizardry-stack-unwinding.html

Ilfak Guilfanov
http://www.hexblog.com/?p=19

The Malware & Thomas Dulliens Blog
http://addxorrol.blogspot.co.at/2013/01/encouraging-female-reverse-engineers.html