Packers and Unpacking

Malware Analysis
CSCI 4976 - Fall 2015
Branden Clark

Overview

- Introduction
- Packer techniques
- Automated Unpacking
- Manual Unpacking

```
        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
```

What is a packed program?

- A program that is wrapped by another program
  - Typically also encrypted and/or compressed

- We've seen UPX before

Why bother?

- Signature evasion
  - Old Anti-Virus technique
  - Packed <-/-> malware
- Reduces payload size
- Increases difficulty of analysis

Detection

- We've been over some methods already
  - PEiD
  - PE Header: Size of Raw Data vs Virtual Size
  - limited strings / imports

Detection

- Another common method
  - Entropy: Measure of randomness
  - Has many uses
    - Check if address had high entropy data, but is now executable
    - Can track it through memory
      - What routines touch this data?
      - Do they move it? Unpack it in place? Jump to it?

Overview

- Introduction
- Packer techniques
- Automated Unpacking
- Manual Unpacking

```
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
```

The process

- Decrypts/Decompresses packed 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_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
```

The process

- Decrypts/Decompresses packed code
- Loads code into memory

The process

- Decrypts/Decompresses packed code
- Loads code into memory
- Resolves imports of original executable

```
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
                            _0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push
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
```

The process

- Decrypts/Decompresses packed code
- Loads code into memory
- Resolves imports of original executable
- Transfer execution to OEP (Original Entry Point)

The process

- Decrypts/Decompresses packed code
- Loads code into memory
- Resolves imports of original executable
- Transfer execution to OEP

Usually anti-analysis at one or all of these steps

Unpacking the code

- Any method you can imagine
  - Plaintext
  - Compression
  - XOR
  - AES
  - ...
- Sometimes this is as far as you need

```
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
```

```
                                ; 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
```

Unpacking the code

- Location of unpacked code
  - Same section (as stub)
  - New section
  - Heap
  - ???

```
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
```

Resolving Imports - Lame

- Keep original import table intact
- Windows' Loader will resolve imports

Resolving Imports - Easy

- Packed stub only imports GetProcAddress and LoadLibrary
- Iterates original executables import table
- Creates the import table from results
- The most common method

Resolving Imports - Hard

- No imports

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

Resolving Imports - Hard

- No imports
- Get address of kernel32 from PEB

```
// find base addr of kernel32.dll
    mov ebx, fs:[edx + 0x30]  ;ebx = ADDRESS of PEB
    mov ecx, [ebx + 0xc]      ;ecx = pointer to loader data
    mov ecx, [ecx + 0x1c]     ;ecx = first entry in initialization order list
    mov ecx, [ecx]            ;ecx = second entry in list (kernelab.dll)
    mov ecx, [ecx]            ;ecx = third entry in list (kernel32.dll)
    mov ebp, [ecx + 0x08]     ;ebp = base ADDRESS of kernel32.dll
```

Resolving Imports - Hard

- Walk kernel32.dll's export table
- Find address of GetProcAddress and LoadLibrary
- Two options
  - Repeat common method using newfound function pointers
  - Use LoadLibrary on required DLLs
    - Walk their export tables, too

Overview

- Introduction
- Packer techniques
- Automated Unpacking
- Manual Unpacking

Automated Unpacking

- Sometimes an official tool is available
  - e.g. UPX
- Usually reliable unless tampered with
  - Change metadata
  - Corkami UPX trap

Automated Unpacking

- Other times, people write scripts or custom tools
  - OllyScript / ODbgScript
    - Olly-scripting.zip
  - Python
  - PEExplorer
    - NSPack, UPack, UPX
- Don't always work
- Sometimes tailored towards a specific executable or packer version

Overview

- Introduction
- Packer techniques
- Automated Unpacking
- Manual Unpacking

```
            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
```

When automated techniques fail

- You can't always just run a tool and get the flag

Packers and Unpacking

When automated techniques fail

- You can't always just run a tool and get the flag
- Difficulty and time required varies greatly
  - Packer
  - Packer options chosen
  - Unpacking method
  - Multiple levels of packing?

When automated techniques fail

- Two methods
  - Write a tool to undo each step performed by the packer
    - No one does this
  - Run the packed program to OEP, dump the process from memory
    - Much, much more common

The process

- Find OEP
- Reconstruct import table

```
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
```

The process

- Find OEP
- Reconstruct import table

Easy, right?

```
                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
```

Finding OEP

- Tail jumps
  - Jump from unpacking stub to OEP
  - Really visible in simple unpacking stubs
    - jmp BB goes nowhere
  - Doesn't have to be a jmp
    - push/ret
    - SEH
    - call
    - ???

```
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
```

Finding OEP

- Section hop
  - OEP can be in a different section than the unpacking stub

```
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
        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_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
```

Finding OEP

- Section hop
  - OEP can be in a different section than the unpacking stub
  - Olly can set a memory breakpoint on the other sections
  - Will automatically break as soon as execution changes sections

Finding OEP

- PUSHA/POPA
  - PUSHA is used to save register state at beginning of stub
  - POPA is used to restore register state at end of stub

```
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
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
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
```

Finding OEP

- PUSHA/POPA
  - PUSHA is used to save register state at beginning of stub
  - POPA is used to restore register state at end of stub
  - Put a memory breakpoint on one of the register values on the stack
  - Will break when POPA pops the value
  - Usually really close to OEP

```
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

lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi

call    sub_314623
test    eax, eax
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
```

Finding OEP

- API call
  - Fixing the import table is usually close to the end of the stub
  - Break on LoadLibrary/GetProcAddress
  - Search nearby

```
                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
                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
```
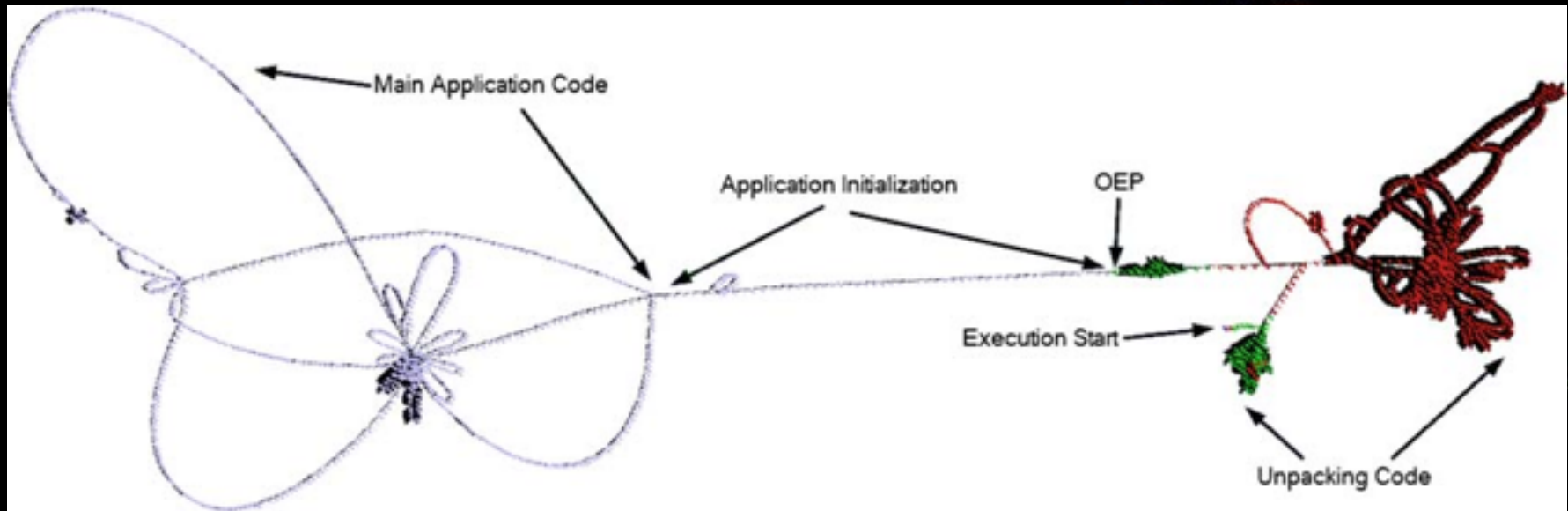
Finding OEP

- API call
  - Windows makes identifiable API calls in start
    - e.g. GetModuleHandle, GetCommandLine
  - Break on one of these functions
  - OEP is usually at the top of the parent

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

Finding OEP

- [VERA](#)
  - Awesome visualization tool that uses PIN

Fixing Imports

- OllyDbg is pretty good at it
- ImpREC is *very* good at it
- You *really* want to avoid doing it by hand
  - Have to look up each address and fill in the pointer in the IAT

Fixing PE header

- Sometimes the PE header doesn't dump right
- Sometimes you have no PE header
  - e.g. Dumping from heap
- LordPE is a great tool for dealing with this
  - Fix, compare, analyze headers

```
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
call    sub_31466A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
        esi, 1D0h
        esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D

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
```

Tips

- PMA Chapter 18 has some great tips for several common packers

References

1. Sikorski, Michael, and Andrew Honig. Practical Malware Analysis the Hands-on Guide to Dissecting Malicious Software. San Francisco: No Starch, 2012. Print.