Lecture 13

More Reversing

Reversing

- FAQ
  - http://www.woodmann.com/fravia/rce-faq.htm
- Unpacking
  - http://www.woodmann.com/fravia/projunpa.htm
- Other resources
  - http://en.wikibooks.org/wiki/Reverse_Engineering

The lists

- http://www.woodmann.com/fravia/packers.htm
- Packers
  - Protect!
  - ICE (COM only)
  - TinyProg, PkTiny
  - Microsoft EXE Pack
  - LZEXE
  - PKLite
  - PROPACKER
  - DIET
  - SEA-AXE
  - PGMPak
  - OPTLink
  - DeltaPack
  - AsPack http://www.aspack.com

The lists

- Packing identifiers
  - PE iDentifier (PEiD)
- Unpackers
  - ProcDump unpacking wizard
  - Tron
  - Xopen
  - Unp
  - StickBuster

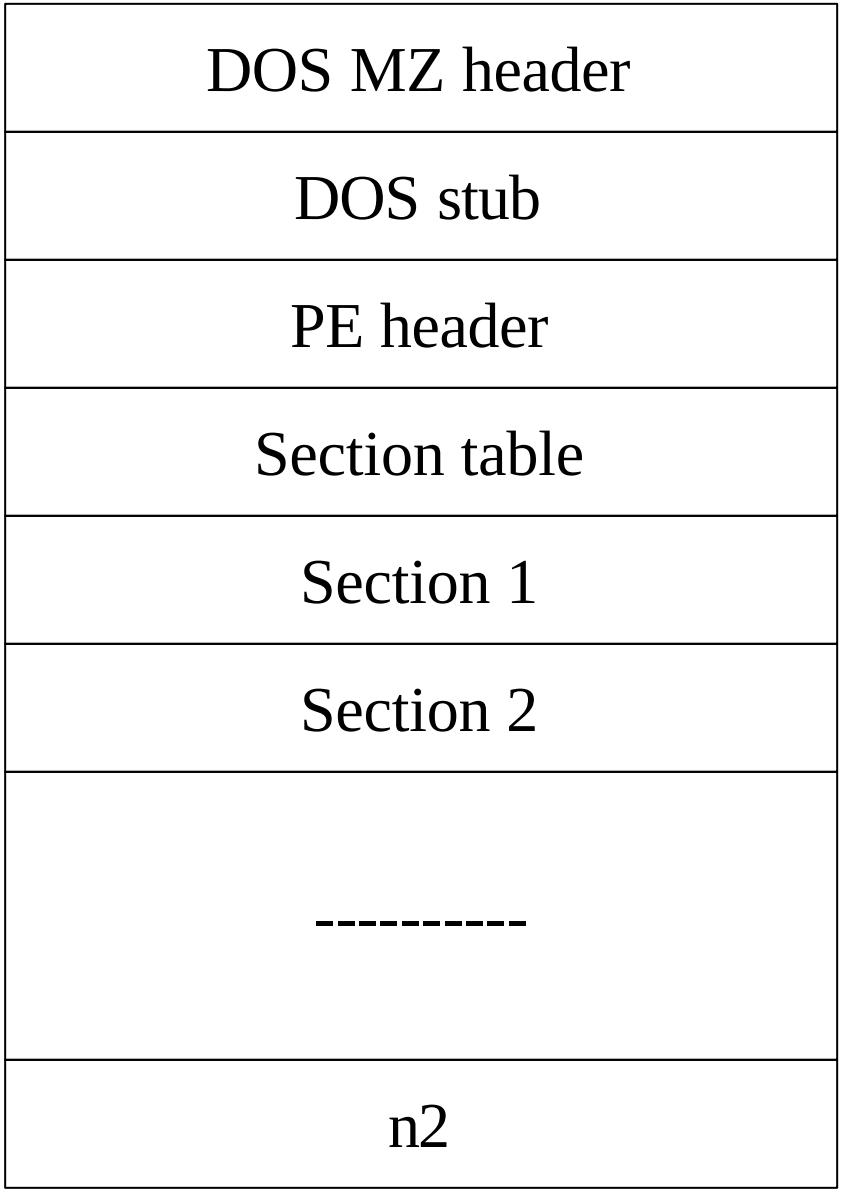
Other tools

- HBGary
  - Reverse engineering software tool suite
  - http://www.hbgary.com/technology.shtml
- Windows Change Analysis Diagnostic tool
  - http://support.microsoft.com/kb/924732

System call analysis

- “Black box” analysis via system call tracing (Windows)
  - Windows system calls
    - http://www.metasploit.com/users/opcode/syscalls.html
  - Strace for Windows
    - http://www.bindview.com/Services/RAZOR/Utilities/Windows/strace_readme.cfm
  - Sebek
    - http://www.honeynet.org/tools/sebek/
  - Snare
    - http://www.intersectalliance.com/projects/SnareWindows/index.html
  - Holodeck (fault injector)
    - http://tejasconsulting.com/open-testware/feature/holodeck-2.0.173.html

PE file format



PE file format

- DOS MZ header
  - So DOS can recognize program
- DOS stub
  - Built-in DOS executable to dixplay “This probram cannot run in MS-DOS mode”
- PE header
  - PE loader uses DOS MZ header to find starting offset of the PE header (skipping stub)
- Sections
  - Blocks of code/data organized on a logical basis

Unpacking

- Generic approach
  - http://www.woodmann.com/fravia/predator_unpacking.htm
- Target
  - http://www.shelltoys.com/files/cmset.exe
  - Packed with ASProtect
- Steps
  - Dumping the App (IceDump, Peditor)
  - Adding a section for new IAT (Peditor)
  - Create new IAT (Revirgin)
  - Combine dump and IAT (HexWorkshop)
  - Update Entrypoint, update IAT rva and size (Peditor)
  - Run (SoftICE)

Highlights

- Find original EntryPoint
  - Useful to find packer entry point
    - Typically outside normal range
    - Examine PE header
      - ASPack 0x00006046
  - Use IceDump's TRACEX to find original entry point
    - Guess original entry point at 0401000 to 0500000
    - IceDump stops at first instruction in this range
  - Trace entire program from here
    - Or set bpx on GetVersion (OS) call

Highlights

- Fix up IAT (middle 3 bullets)
  - ASProtect works by modifying the IAT RVA entries(?)
  - Look for invalid calls
- Fix check for ASProtect running in memory
  - Program will crash if it is not present
  - Must patch this condition up
- More ASProtect links
  - http://www.woodmann.com/fravia/nchant_asprotect.htm
  - http://www.woodmann.com/fravia/crusader_AsProtect%20reversed.html

Example #1

- http://www.woodmann.com/fravia/text/eb_tut31.txt
- Unpacking notepad.exe packed with Shrinker 3.4
  - ProcDump, SoftICE, Symbol Loader
- Packer prevents SoftICE from breaking on entry point of program
  - Edit PE header “Sections”
  - First section is .shrink0
    - Make it “executable” so SoftICE breaks
    - 0xc0000082 changed to 0xE0000020

```
// Section characteristics.
#define IMAGE_SCN_TYPE_NO_PAD 0x00000008 // Reserved.
#define IMAGE_SCN_CNT_CODE 0x00000020 // Section contains code.
#define IMAGE_SCN_CNT_INITIALIZED_DATA 0x00000040 //Section contains initialized data.
#define IMAGE_SCN_CNT_UNINITIALIZED_DATA 0x00000080 //Section contains uninitialized data.
#define IMAGE_SCN_LNK_OTHER 0x00000100 // Reserved.
#define IMAGE_SCN_LNK_INFO 0x00000200 // Section contains comments or some other type of information.
#define IMAGE_SCN_LNK_REMOVE 0x00000800 // Section contents will not become part of image.
#define IMAGE_SCN_LNK_COMDAT 0x00001000 // Section contents comdat.
#define IMAGE_SCN_NO_DEFER_SPEC_EXC 0x00004000 // Reset speculative exceptions handling
                                                                                bits in the TLB entries for this section.
#define IMAGE_SCN_GPREL 0x00008000 // Section content can be accessed relative to GP
#define IMAGE_SCN_MEM_FARDATA 0x00008000
#define IMAGE_SCN_MEM_PURGEABLE 0x00020000
#define IMAGE_SCN_MEM_16BIT 0x00020000
#define IMAGE_SCN_MEM_LOCKED 0x00040000
#define IMAGE_SCN_MEM_PRELOAD 0x00080000
#define IMAGE_SCN_ALIGN_1BYTES 0x00100000 //
#define IMAGE_SCN_ALIGN_2BYTES 0x00200000 //
#define IMAGE_SCN_ALIGN_4BYTES 0x00300000 //
#define IMAGE_SCN_ALIGN_8BYTES 0x00400000 //
#define IMAGE_SCN_ALIGN_16BYTES 0x00500000 // Default alignment if no others are specified.
#define IMAGE_SCN_ALIGN_32BYTES 0x00600000 //
#define IMAGE_SCN_ALIGN_64BYTES 0x00700000 //
#define IMAGE_SCN_ALIGN_128BYTES 0x00800000 //
#define IMAGE_SCN_ALIGN_256BYTES 0x00900000 //
#define IMAGE_SCN_ALIGN_512BYTES 0x00A00000 //
#define IMAGE_SCN_ALIGN_1024BYTES 0x00B00000 //
#define IMAGE_SCN_ALIGN_2048BYTES 0x00C00000 //
#define IMAGE_SCN_ALIGN_4096BYTES 0x00D00000 //
#define IMAGE_SCN_ALIGN_8192BYTES 0x00E00000 //
#define IMAGE_SCN_LNK_NRELOC_OVFL 0x01000000 // Section contains extended relocations.
#define IMAGE_SCN_MEM_DISCARDABLE 0x02000000 // Section can be discarded.
#define IMAGE_SCN_MEM_NOT_CACHED 0x04000000 // Section is not cachable.
#define IMAGE_SCN_MEM_NOT_PAGED 0x08000000 // Section is not pageable.
#define IMAGE_SCN_MEM_SHARED 0x10000000 // Section is shareable.
#define IMAGE_SCN_MEM_EXECUTE 0x20000000 // Section is executable.
#define IMAGE_SCN_MEM_READ 0x40000000 // Section is readable.
#define IMAGE_SCN_MEM_WRITE 0x80000000 // Section is writeable.
.................................

From this we can see that our 0xC0000040 actually means:-

0xC0000040 = IMAGE_SCN_MEM_READ | IMAGE_SCN_MEM_WRITE | IMAGE_SCN_CNT_INITIALIZED_DATA
^^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Read Write Initialized Data
```

Example #1

- Open SoftICE with symbol loader
- Strategy
  - Find call to “main()”
  - Will launch entire application when invoked

Example #1

```
0041454F  FFFF              INVALID
00414556  55                PUSH      EBP
00414557  8BEC              MOV       EBP,ESP
00414559  56                PUSH      ESI
0041455A  57                PUSH      EDI
0041455B  756B              JNZ       004145C8                (NO JUMP)
0041455D  6800010000        PUSH      00000100
00414562  E8D60B0000        CALL      0041513D
00414567  83C404            ADD       ESP,04
0041456A  8B7508            MOV       ESI,[EBP+08]
0041456D  A3B4F14000        MOV       [0040F1B4],EAX
00414572  85F6              TEST      ESI,ESI
00414574  7423              JZ        00414599                (JUMP)
00414599  33FF              XOR       EDI,EDI
0041459B  57                PUSH      EDI
0041459C  893D8C184100      MOV       [0041188C],EDI
004145A2  FF1510224100      CALL      [KERNEL32!GetModuleHandleA]
004145A8  8BF0              MOV       ESI,EAX
004145AA  68FF000000        PUSH      000000FF
004145AF  A1B4F14000        MOV       EAX,[0040F1B4]
004145B4  897D10            MOV       [EBP+10],EDI
004145B7  C7450C01000000    MOV       DWORD PTR [EBP+0C],00000001
004145BE  50                PUSH      EAX
004145BF  56                PUSH      ESI
004145C0  FF15F4214100      CALL      [KERNEL32!GetModuleFileNameA]
004145C6  EB03              JMP       004145CB                (JUMP)
004145CB  E830EAFFFF        CALL      00413000
004145D0  FF7510            PUSH      DWORD PTR [EBP+10]
004145D3  FF750C            PUSH      DWORD PTR [EBP+0C]
004145D6  56                PUSH      ESI
004145D7  E806000000        CALL      004145E2
```

Stop through using F10, notepad will launch after this call

Example #1

```
004145E2  64A100000000      MOV     EAX,FS:[00000000]
004145E8  55                PUSH    EBP
004145E9  8BEC              MOV     EBP,ESP
004145EB  6AFF              PUSH    FF
004145ED  6810E04000        PUSH    0040E010
004145F2  68EC5D4100        PUSH    00415DEC
004145F7  50                PUSH    EAX
004145F8  64892500000000    MOV     FS:[00000000],ESP
004145FF  83EC14            SUB     ESP,14
00414602  C745E401000000    MOV     DWORD PTR [EBP-1C],00000001
00414609  53                PUSH    EBX
0041460A  56                PUSH    ESI
0041460B  57                PUSH    EDI
0041460C  8965E8            MOV     [EBP-18],ESP
0041460F  C745FC00000000    MOV     DWORD PTR [EBP-04],00000000
00414616  8B450C            MOV     EAX,[EBP+0C]
00414619  83F801            CMP     EAX,01
0041461C  7510              JNZ     0041462E                (NO JUMP)
0041461E  E886030000        CALL    004149A9
00414623  FF05C0F14000      INC     DWORD PTR [0040F1C0]
00414629  E882F6FFFF        CALL    00413CB0
0041462E  8B35C0F14000      MOV     ESI,[0040F1C0]
00414634  85F6              TEST    ESI,ESI
00414636  0F848D000000      JZ      004146C9                (NO JUMP)
0041463C  833DC4F1400000    CMP     DWORD PTR [0040F1C4],00
00414643  7526              JNZ     0041466B                (NO JUMP)
00414645  833D6417410000    CMP     DWORD PTR [00411764],00
0041464C  741D              JZ      0041466B                (NO JUMP)
0041464E  A164174100        MOV     EAX,[00411764]
00414653  030588184100      ADD     EAX,[00411888]
00414659  8945DC            MOV     [EBP-24],EAX
0041465C  FF7510            PUSH    DWORD PTR [EBP+10]
0041465F  FF750C            PUSH    DWORD PTR [EBP+0C]
00414662  FF7508            PUSH    DWORD PTR [EBP+08]
00414665  FF55DC            CALL    [EBP-24]
```

BPX on call and use F8 to follow it, notepad will launch after this call

Example #1

- A closer look

```
0041464E  A164174100          MOV       EAX,[00411764]      ; EAX = 0x000010cc
00414653  030588184100        ADD       EAX,[00411888]      ; EAX = 0x004010cc
00414659  8945DC              MOV       [EBP-24],EAX        ; [EBP-24] = 0x004010cc
0041465C  FF7510              PUSH      DWORD PTR [EBP+10]  ; Parameters
0041465F  FF750C              PUSH      DWORD PTR [EBP+0C]
00414662  FF7508              PUSH      DWORD PTR [EBP+08]
00414665  FF55DC              CALL      [EBP-24]            ; Call
```

- Packed program is calling 0x004010cc
  - Entry point into notepad.exe
  - call [EBP-24] used by all shrinker packed apps

Example #1

- Creating an unpacked version
  - Trace program to call [EBP-24] instruction
  - Change call instruction to a jmp
    - “a eip”
    - “jmp eip”
    - F5 to escape
    - ProcDump process
  - Edit PE header
    - Entry point is 0x004010cc (not 0x0001454f)

Example #2

- http://www.woodmann.com/fravia/text/eb_tut32.txt
- Unpacking notepad.exe packed with NeoLite v2.0
  - ProcDump, SoftICE, Symbol Loader
- Try opening with Symbol Loader and SoftICE
  - No breaks happen
- Open notepad.exe using PE editor to fix entry point
  - First section is a .text with characteristics 0xC0000080
  - As with previous, change to 0xE0000020
- Open with Symbol Loader and SoftICE again

Example #2

- SoftICE breaks here

```
0040D17E  E9A6000000          JMP       0040D229                (JUMP)
0040D229  8B442404            MOV       EAX,[ESP+04]
0040D22D  23058FD14000        AND       EAX,[0040D18F]
0040D233  E871030000          CALL      0040D5A9
**unpacking the program in memory this done by the CALL above
  You can trace into it and see what it does if you want. 8)

0040D238  FE0528D24000        INC       BYTE PTR [0040D228]
0040D23E  FFE0                JMP       EAX

**This "JMP EAX" will bring the program to the original Entry Po
```

- Step through until JMP EAX
  - EAX = 0x004010cc
- See previous example

Example #3

- http://www.woodmann.com/fravia/text/eb_tut33.txt
- Unpacking notepad.exe packed with PECompact
  - ProcDump, SoftICE, Symbol Loader
- Open with Symbol Loader and SoftICE

```
0040AC44  FFFF              INVALID
0040AC4C  9C                PUSHFD
0040AC4D  60                PUSHAD
0040AC4E  E802000000        CALL      0040AC55
**If you step over this CALL using F10, the program will run.
  Thus, reload the program and step into this CALL using F8 next time.
```

  - Breaks at entry point of unpacking code
  - First call is unpacking routine
    - Step through it

Example #3

- Lots of conditional loops

```
aaaaaaaa
...
WWWWWWWW
xxxxxxxx JNZ zzzzzzzz              <-- Loop back to aaaaaaaa
yyyyyyyy JMP aaaaaaaa
zzzzzzzz New Instructions
```

- Set bpx on zzzzzzzz

```
0040CA83  8BBD2E744000        MOV       EDI,[EBP+0040742E]
0040CA89  E85E040000          CALL      0040CEEC
0040CA8E  61                  POPAD
0040CA8F  9D                  POPFD
0040CA90  50                  PUSH      EAX
0040CA91  68CC104000          PUSH      004010CC
0040CA96  C20400              RET       0004
```

- POPAD and POPFD used by unpackers a lot
  - POPAD: Pops all registers off stack and restores them
  - POPFD: Pops EFLAGS register from stack
- Push of 0x004010cc like other examples
- Set breakpoint there and dump like previously

Example #4

- http://www.woodmann.com/fravia/volati_s.htm
- Unpacking calc.exe packed with ASPack
  - Much of the same as before
  - Slightly different entry into unpacked code

```
015F:01017554  MOV     [ESP+1C],EAX
015F:01017558  POPAD
015F:01017559  JNZ     01017563                                    (JUMP  )
015F:01017563  PUSH    EAX  *** Take note of the value
of EAX!
015F:01017564  RET          *** Stop here!!!
```

Example #5

- http://www.woodmann.com/fravia/rude45.htm
- Code snippet of decryption routine for ReBirth 1.5

```
loc_0_3576:
                        mov         bx, 10h     ;
                        add         bx, 0F0h    ; 10h + F0h = 100h*
                        push        bx
                        mov         al, 0AAh    ; Encryption key into al
                        add         al, 7           ; change the key by adding 7
                        push        cx
                        mov         cx, 3465h   ; # of bytes to decrypt into cx

loc_0_3586:
                        jmp         loc_0_358A  ; jmp to decryption loop
; ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
                        db 0EAh     ; ?
; ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?

loc_0_358A:
                        xor         cs:[bx], al ; xor the byte at cs:[bx] with our encryption key
                        push        si      ; random code
                        mov         si, 64h     ; ??
                        sub         si, 27DBh   ; ??
                        pop         bp                          ; ??
                        inc         al          ; increment the encryption key
                        inc         bx          ; look at the next byte in the program
                        loop        loc_0_3586 ; loop until all bytes have been processed

* 100h is the starting location in memory of a .com file
```

Example #5

- Removing nag screen

```
                        mov         ds, cs:word_0_2B9       ; setting up a far call
                        push        ds                                ;
                        pop         es                                ;
                        assume es:seg000                    ;
                        call        dword ptr cs:unk_0_2B7  ; NagUser();
```

Trace into the NagUser() function and here is what you will see:

```
                        or          ax, ax          ;Test value in ax
                        jz          loc_0_BA4   ;jump
                        clc
                        retf
loc_0_BA4:
                        xor         ax, ax
                        int         16h             ; KEYBOARD - READ CHAR FROM BUFFER, WAIT IF EMPTY
                                                    ; Return: AH = scan code, AL = character
                        cmp         ah, 49h         ; compare the key pressed to "Page Up"
                        jz          loc_0_BAF   ; jump if equal
```

- Alter to skip keyboard input and to do an unconditional jump

Highlights of other examples

- AsProtect
  - http://www.woodmann.com/fravia/tsehp_asprotect10.htm
  - http://www.woodmann.com/fravia/tsehp_asprotect105.htm
- Anti-SoftICE code in packer
  - Call createfileA on known SoftICE driver
  - Call getlocaltime to see if being debugged
  - SoftICE uses int 3 for breakpoints
    - Check by triggering an int 3
- Getting to entry point
  - Look for POPAD followed by JMP
  - Dump via ProcDump
  - Fix IAT