Practical Reversing II – Unpacking EXE

Nagareshwar Talekar

www.SecurityXploded.com

Disclaimer

The Content, Demonstration, Source Code and Programs presented here is "AS IS" without any warranty or conditions of any kind. Also the views/ideas/knowledge expressed here are solely of the trainer’s only and nothing to do with the company or the organization in which the trainer is currently working.

However in no circumstances neither the trainer nor SecurityXploded is responsible for any damage or loss caused due to use or misuse of the information presented here.

Acknowledgement

- Special thanks to **null** & **Garage4Hackers** community for their extended support and cooperation.
- Thanks to all the trainers who have devoted their precious time and countless hours to make it happen.

Reversing & Malware Analysis Training

This presentation is part of our **Reverse Engineering & Malware Analysis** Training program. Currently it is delivered only during our local meet for FREE of cost.

For complete details of this course, visit our Security Training page.

Who am I

Nagareshwar Talekar

- Founder of SecurityXploded
- Reverse Engineering, Malware Analysis, Cryptography, Password Forensics, Secure Coding etc.
- Email: tnagareshwar at gmail.com

Contents

What is EXE Packing/Protecting?

- **EXE Packing:**

  Compressing the Executable to a smaller Size

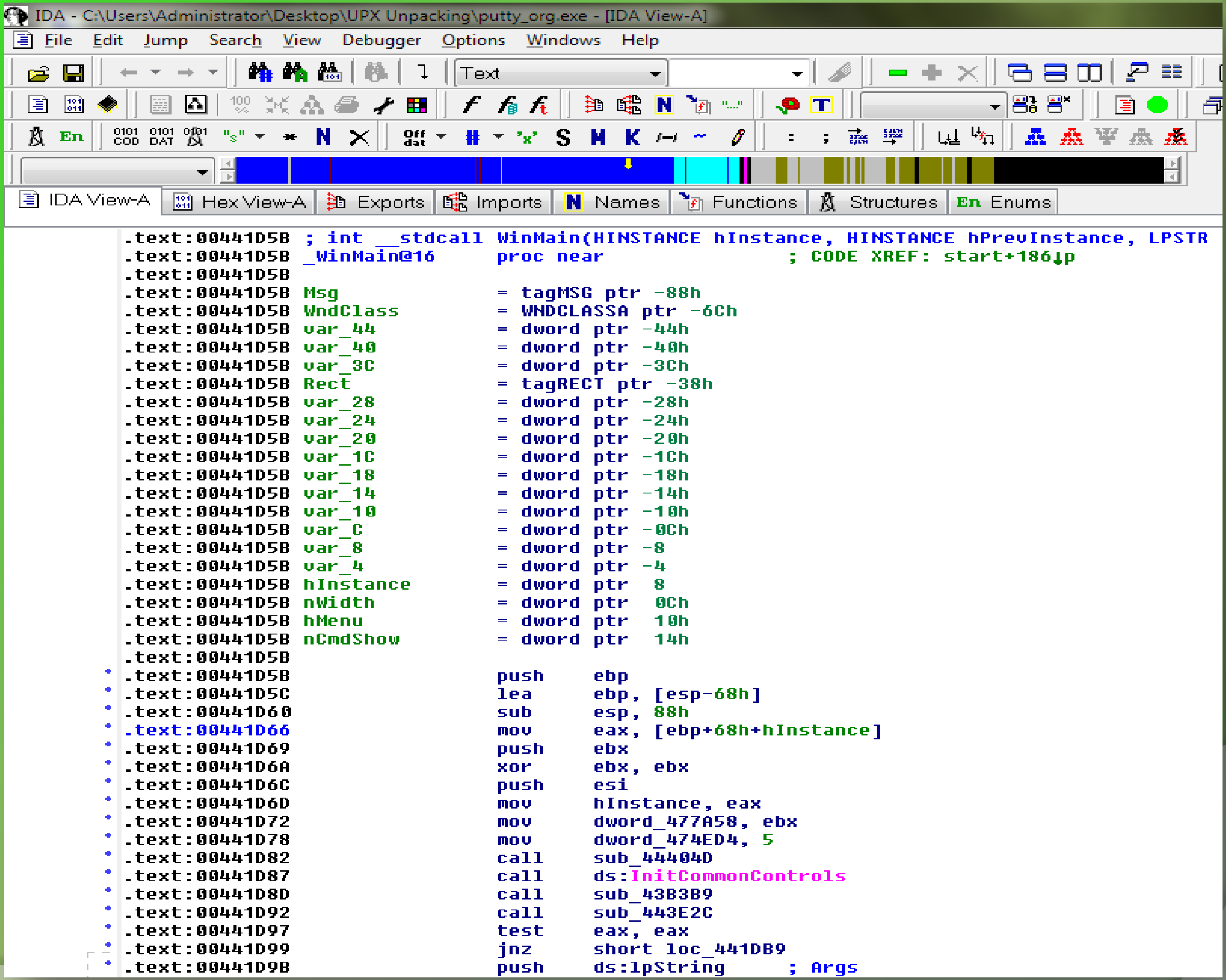
- **EXE Protecting:**

  Encrypting with Anti-Debugging Techniques to prevent Reversing

✓ In Reversing world, both Packer & Protector is commonly referred as **Packer**.

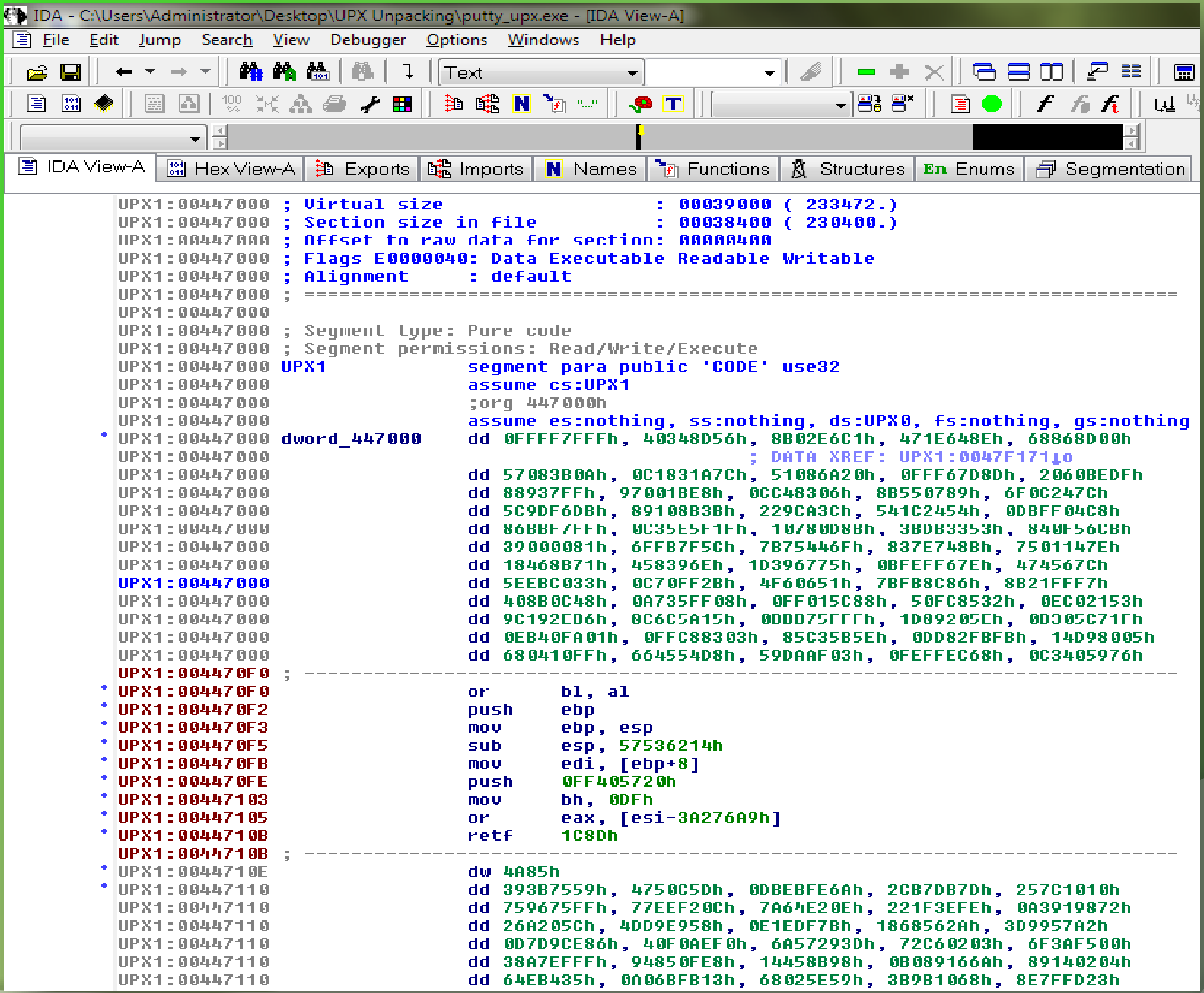
Examples of Packers: UPX, AsProtect, Armadillo etc.

EXE - Before Packing


IDA - C:\Users\Administrator\Desktop\UPX Unpacking\putty_org.exe - [IDA View-A]
File Edit Jump Search View Debugger Options Windows Help
Text
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Structures
Enums

```
.text:00441D5B ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
.text:00441D5B _WinMain@16     proc near               ; CODE XREF: start+186↓p
.text:00441D5B
.text:00441D5B Msg             = tagMSG ptr -88h
.text:00441D5B WndClass        = WNDCLASSA ptr -6Ch
.text:00441D5B var_44          = dword ptr -44h
.text:00441D5B var_40          = dword ptr -40h
.text:00441D5B var_3C          = dword ptr -3Ch
.text:00441D5B Rect            = tagRECT ptr -38h
.text:00441D5B var_28          = dword ptr -28h
.text:00441D5B var_24          = dword ptr -24h
.text:00441D5B var_20          = dword ptr -20h
.text:00441D5B var_1C          = dword ptr -1Ch
.text:00441D5B var_18          = dword ptr -18h
.text:00441D5B var_14          = dword ptr -14h
.text:00441D5B var_10          = dword ptr -10h
.text:00441D5B var_C           = dword ptr -0Ch
.text:00441D5B var_8           = dword ptr -8
.text:00441D5B var_4           = dword ptr -4
.text:00441D5B hInstance       = dword ptr  8
.text:00441D5B nWidth          = dword ptr  0Ch
.text:00441D5B hMenu           = dword ptr  10h
.text:00441D5B nCmdShow        = dword ptr  14h
.text:00441D5B
.text:00441D5B                 push    ebp
.text:00441D5C                 lea     ebp, [esp-68h]
.text:00441D60                 sub     esp, 88h
.text:00441D66                 mov     eax, [ebp+68h+hInstance]
.text:00441D69                 push    ebx
.text:00441D6A                 xor     ebx, ebx
.text:00441D6C                 push    esi
.text:00441D6D                 mov     hInstance, eax
.text:00441D72                 mov     dword_477A58, ebx
.text:00441D78                 mov     dword_474ED4, 5
.text:00441D82                 call    sub_44404D
.text:00441D87                 call    ds:InitCommonControls
.text:00441D8D                 call    sub_43B3B9
.text:00441D92                 call    sub_443E2C
.text:00441D97                 test    eax, eax
.text:00441D99                 jnz     short loc_441DB9
.text:00441D9B                 push    ds:lpString     ; Args
```

EXE - After Packing


IDA - C:\Users\Administrator\Desktop\UPX Unpacking\putty_upx.exe - [IDA View-A]
File Edit Jump Search View Debugger Options Windows Help
Text
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Structures
Enums
Segmentation

```
UPX1:00447000 ; Virtual size                  : 00039000 ( 233472.)
UPX1:00447000 ; Section size in file          : 00038400 ( 230400.)
UPX1:00447000 ; Offset to raw data for section: 00000400
UPX1:00447000 ; Flags E0000040: Data Executable Readable Writable
UPX1:00447000 ; Alignment     : default
UPX1:00447000 ; ===========================================================================
UPX1:00447000
UPX1:00447000 ; Segment type: Pure code
UPX1:00447000 ; Segment permissions: Read/Write/Execute
UPX1:00447000 UPX1            segment para public 'CODE' use32
UPX1:00447000                 assume cs:UPX1
UPX1:00447000                 ;org 447000h
UPX1:00447000                 assume es:nothing, ss:nothing, ds:UPX0, fs:nothing, gs:nothing
UPX1:00447000 dword_447000    dd 0FFFF7FFFh, 40348D56h, 8B02E6C1h, 471E648Eh, 68868D00h
UPX1:00447000                                                    ; DATA XREF: UPX1:0047F171↓o
UPX1:00447000                 dd 57083B0Ah, 0C1831A7Ch, 51086A20h, 0FFF67D8Dh, 2060BEDFh
UPX1:00447000                 dd 88937FFh, 97001BE8h, 0CC48306h, 8B550789h, 6F0C247Ch
UPX1:00447000                 dd 5C9DF6DBh, 89108B3Bh, 229CA3Ch, 541C2454h, 0DBFF04C8h
UPX1:00447000                 dd 86BBF7FFh, 0C35E5F1Fh, 10780D8Bh, 3BDB3353h, 840F56CBh
UPX1:00447000                 dd 39000081h, 6FFB7F5Ch, 7B75446Fh, 837E748Bh, 7501147Eh
UPX1:00447000                 dd 18468B71h, 458396Eh, 1D396775h, 0BFEFF67Eh, 474567Ch
UPX1:00447000                 dd 5EEBC033h, 0C70FF2Bh, 4F60651h, 7BFB8C86h, 8B21FFF7h
UPX1:00447000                 dd 408B0C48h, 0A735FF08h, 0FF015C88h, 50FC8532h, 0EC02153h
UPX1:00447000                 dd 9C192EB6h, 8C6C5A15h, 0BBB75FFFh, 1D89205Eh, 0B305C71Fh
UPX1:00447000                 dd 0EB40FA01h, 0FFC88303h, 85C35B5Eh, 0DD82FBFBh, 14D98005h
UPX1:00447000                 dd 680410FFh, 664554D8h, 59DAAF03h, 0FEFFEC68h, 0C3405976h
UPX1:004470F0 ; ---------------------------------------------------------------------------
UPX1:004470F0                 or      bl, al
UPX1:004470F2                 push    ebp
UPX1:004470F3                 mov     ebp, esp
UPX1:004470F5                 sub     esp, 57536214h
UPX1:004470FB                 mov     edi, [ebp+8]
UPX1:004470FE                 push    0FF405720h
UPX1:00447103                 mov     bh, 0DFh
UPX1:00447105                 or      eax, [esi-3A276A9h]
UPX1:0044710B                 retf    1C8Dh
UPX1:0044710B ; ---------------------------------------------------------------------------
UPX1:0044710E                 dw 4A85h
UPX1:00447110                 dd 393B7559h, 4750C5Dh, 0DBEBFE6Ah, 2CB7DB7Dh, 257C1010h
UPX1:00447110                 dd 759675FFh, 77EEF20Ch, 7A64E20Eh, 221F3EFEh, 0A3919872h
UPX1:00447110                 dd 26A205Ch, 4DD9E958h, 0E1EDF7Bh, 1868562Ah, 3D9957A2h
UPX1:00447110                 dd 0D7D9CE86h, 40F0AEF0h, 6A57293Dh, 72C60203h, 6F3AF500h
UPX1:00447110                 dd 38A7EFFFh, 94850FE8h, 14458B98h, 0B089166Ah, 89140204h
UPX1:00447110                 dd 64EB435h, 0A06BFB13h, 68025E59h, 3B9B1068h, 8E7FFD23h
```

Purpose of Packing EXE

- Prevent Reverse Engineering [Crack License, Secret Code etc.]
  - Defeat Static Disassembling
  - Make Dynamic Debugging Difficult
- Reduce the size of Executable file
- Bypass Anti-virus Detections with multi-level Packing

✓ It is used by Software Vendors to prevent Serial Cracking and Malware Authors to prevent analysis by AV Researchers.

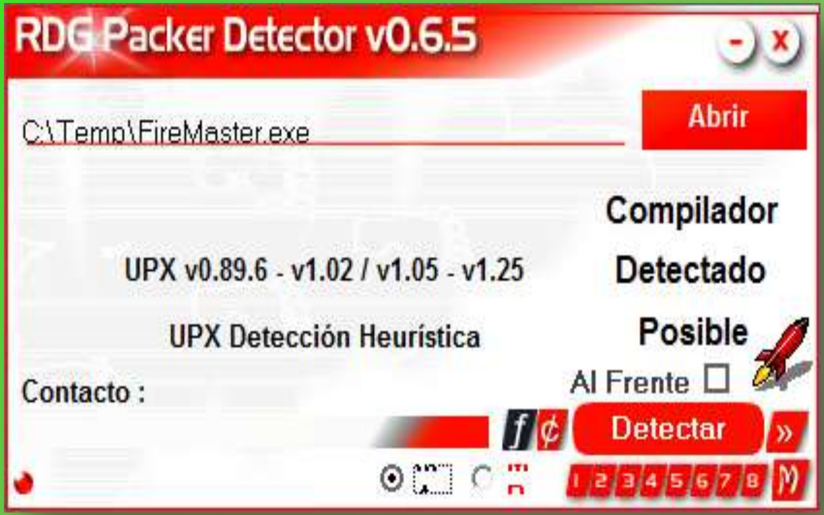
What is Unpacking?

- **Extracting the Original Binary from the Packed Executable File.**
- **Automatic Unpackers available for popular Packers.**
  - **May not work with different versions**
  - **Not available for Complex packers**
- **Involves Live Debugging by Defeating Anti-Debugging techniques**

Detection of Packer

- **Packer Detectors like PEiD, RDG, ExeScan etc**
  - **Detect the popular Packers**
  - **Show the version of Packer also**
- **PE Viewer Tools like PEditor, PEview**
  - **Look at Section Table**
  - **Look at Import Table**

Packer Detectors



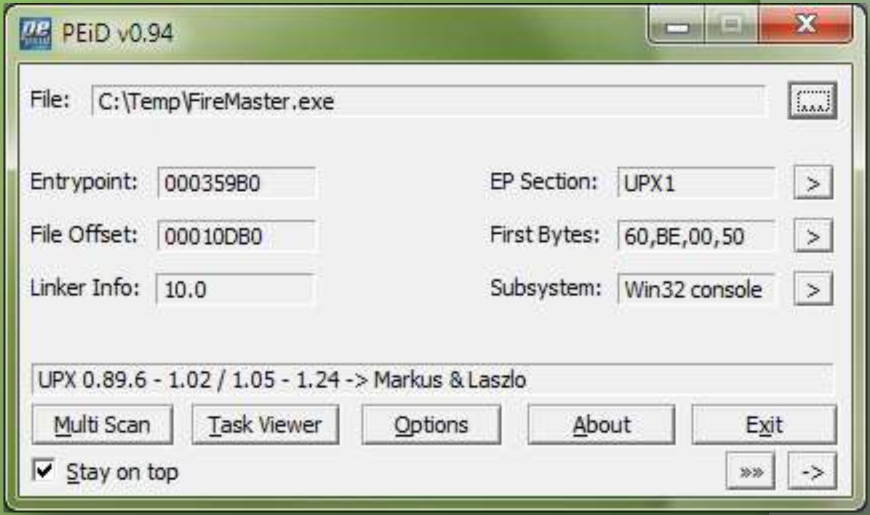
C:\Temp\FireMaster.exe

Compilador

UPX v0.89.6 - v1.02 / v1.05 - v1.25 Detectado

UPX Detección Heurística Posible

Contacto :



File: C:\Temp\FireMaster.exe

Entrypoint: 000359B0

EP Section: UPX1

File Offset: 00010DB0

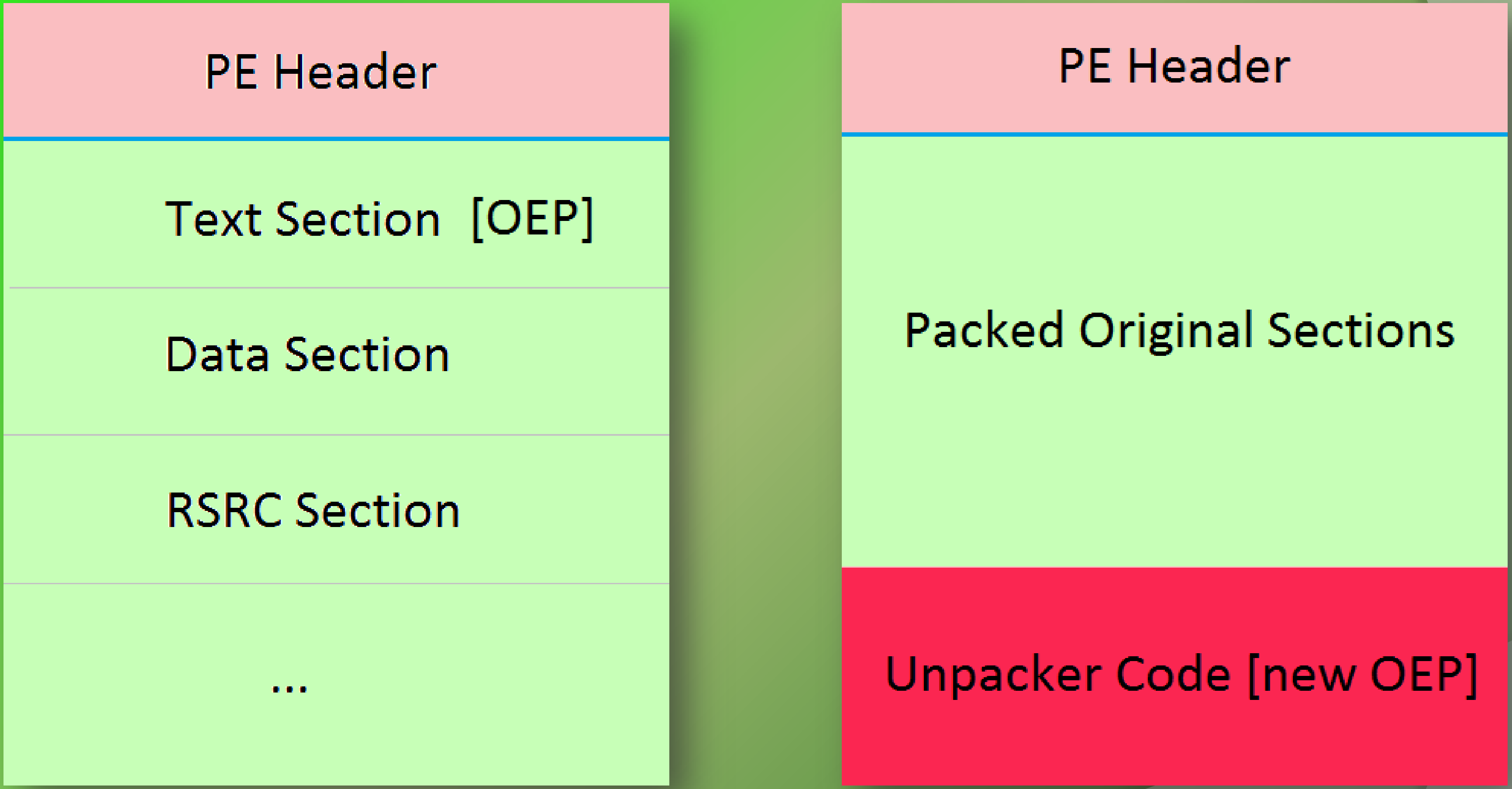
First Bytes: 60,BE,00,50

Linker Info: 10.0

Subsystem: Win32 console

UPX 0.89.6 - 1.02 / 1.05 - 1.24 -> Markus & Laszlo

Structure of Packed EXE



Before Packing

After Packing

Execution of Packed EXE Program

- Execution starts from new OEP
- Saves the Register status using PUSHAD instruction
- All the Packed Sections are Unpacked in memory
- Resolve the import table of original executable file.
- Restore the original Register Status using POPAD instruction
- Finally Jumps to Original Entry point to begin the actual execution

Standard Process of Unpacking EXE

- **Debug the EXE to find the real OEP (Original Entry Point)**
- **At OEP, Dump the fully Unpacked Program to Disk**
- **[?] Fix the Import Table using ImpRec Tool**
- **[?] Fix the PE Header**

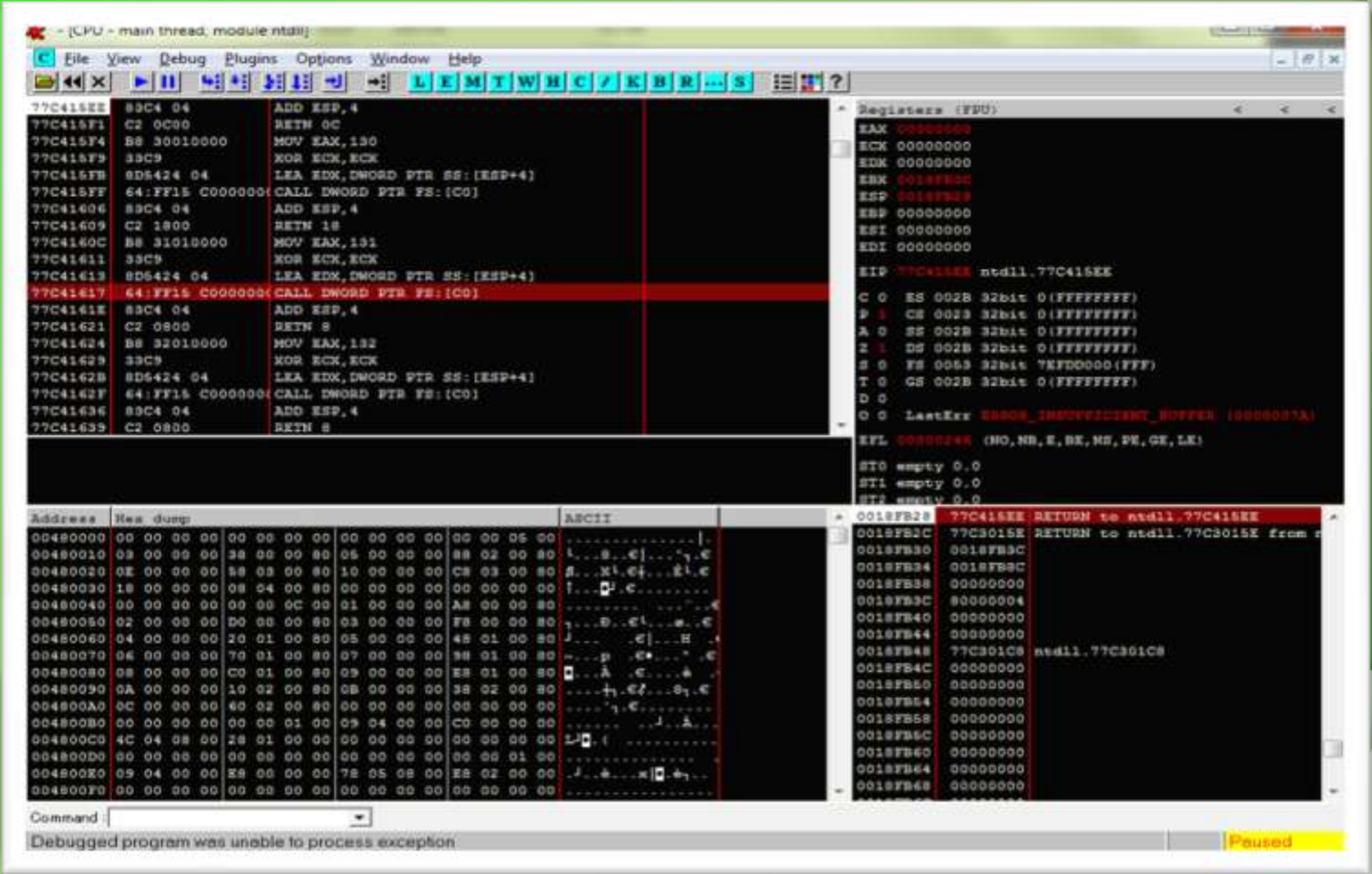
Unpacking UPX using OllyDbg

- Load the UPX packed EXE file into the OllyDbg
- Start tracing the EXE, until you encounter a PUSHAD instruction.
- At this stage, put the Hardware Breakpoint (type 'hr esp-4' at command bar) so as to stop at POPAD instruction.
- Other way is to manually search for POPAD (Opcode 61) instruction and then set Breakpoint on it.

Unpacking UPX using OllyDbg (contd)

- Next press F9 to continue the Execution.
- You will break on the instruction which is immediately after POPAD or on POPAD instruction [based on the method you have chosen]
- Now start tracing with F7 and soon you will encounter a JMP instruction which will Jump to OEP in the original program.
- At OEP, dump the whole program using OllyDmp plugin.

DEMO - Unpacking UPX


[CPU - main thread, module ntdll]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S

```
77C415EE   83C4 04          ADD ESP,4
77C415F1   C2 0C00          RETN 0C
77C415F4   B8 30010000      MOV EAX,130
77C415F9   33C9             XOR ECX,ECX
77C415FB   8D5424 04        LEA EDX,DWORD PTR SS:[ESP+4]
77C415FF   64:FF15 C0000000 CALL DWORD PTR FS:[C0]
77C41606   83C4 04          ADD ESP,4
77C41609   C2 1800          RETN 18
77C4160C   B8 31010000      MOV EAX,131
77C41611   33C9             XOR ECX,ECX
77C41613   8D5424 04        LEA EDX,DWORD PTR SS:[ESP+4]
77C41617   64:FF15 C0000000 CALL DWORD PTR FS:[C0]
77C4161E   83C4 04          ADD ESP,4
77C41621   C2 0800          RETN 8
77C41624   B8 32010000      MOV EAX,132
77C41629   33C9             XOR ECX,ECX
77C4162B   8D5424 04        LEA EDX,DWORD PTR SS:[ESP+4]
77C4162F   64:FF15 C0000000 CALL DWORD PTR FS:[C0]
77C41636   83C4 04          ADD ESP,4
77C41639   C2 0800          RETN 8
```

```
Registers (FPU)
EAX [illegible]
ECX 00000000
EDX 00000000
EBX [illegible]
ESP [illegible]
EBP 00000000
ESI 00000000
EDI 00000000

EIP [illegible] ntdll.77C415EE

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_INSUFFICIENT_BUFFER (0000007A)

EFL [illegible] (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
```

```
Address   Hex dump                                          ASCII
00480000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 00
00480010  03 00 00 00 38 00 00 80 05 00 00 00 88 02 00 80
00480020  0E 00 00 00 58 03 00 80 10 00 00 00 C8 03 00 80
00480030  18 00 00 00 08 04 00 80 00 00 00 00 00 00 00 00
00480040  00 00 00 00 00 00 0C 00 01 00 00 00 A8 00 00 80
00480050  02 00 00 00 D0 00 00 80 03 00 00 00 F8 00 00 80
00480060  04 00 00 00 20 01 00 80 05 00 00 00 48 01 00 80
00480070  06 00 00 00 70 01 00 80 07 00 00 00 98 01 00 80
00480080  08 00 00 00 C0 01 00 80 09 00 00 00 E8 01 00 80
00480090  0A 00 00 00 10 02 00 80 0B 00 00 00 38 02 00 80
004800A0  0C 00 00 00 60 02 00 80 00 00 00 00 00 00 00 00
004800B0  00 00 00 00 00 00 01 00 09 04 00 00 C0 00 00 00
004800C0  4C 04 08 00 28 01 00 00 00 00 00 00 00 00 00 00
004800D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00
004800E0  09 04 00 00 E8 00 00 00 78 05 08 00 E8 02 00 00
004800F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
0018FB28   77C415EE  RETURN to ntdll.77C415EE
0018FB2C   77C3015E  RETURN to ntdll.77C3015E from
0018FB30   0018FB3C
0018FB34   0018FB8C
0018FB38   00000000
0018FB3C   80000004
0018FB40   00000000
0018FB44   00000000
0018FB48   77C301C8  ntdll.77C301C8
0018FB4C   00000000
0018FB50   00000000
0018FB54   00000000
0018FB58   00000000
0018FB5C   00000000
0018FB60   00000000
0018FB64   00000000
0018FB68   00000000
```

Command

Debugged program was unable to process exception

Paused

Anti Anti-Debugging Plugins

Here are useful OllyDbg Plugins for Anti Anti-Debugging

- **Olly Advanced**
- **Hide Debugger**
- **NtGlobalFlag**
- **Anti Anti BPM**

Useful Tips

- Always use simple EXE for Unpacking exercises
- Use same EXE for all – You will know the OEP & other magic numbers
- Use Windows XP for better (less annoying) debugging experience.
- Have Patience, Its an Art and takes time.
- For best results, do it in the Moon Light ☺

What's Next?

- **Try Unpacking AsPack, AsProtect, PESpin, YodaP etc**
- **Try Unpacking Packed DLL (Google - Neolite DLL Unpacking)**
- **Try Advanced Packers: Armadillo ☺**

Reference

- Complete Reference Guide for Reversing & Malware Analysis Training

Thank You !

www.SecurityXploded.com