

# Practical Reversing II – Unpacking EXE

Nagareshwar Talekar



[www.SecurityXploded.com](http://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](mailto:tnagareshwar@gmail.com)

# Course Q&A

- ⦿ Keep yourself up to date with latest security news
  - <http://www.securityphresh.com>
  
- ⦿ For Q&A, join our mailing list.
  - <http://groups.google.com/group/securityxploded>



# Contents

- What is EXE Packing?
- Purpose of Packing EXE
- What is Unpacking?
- Detection of Packer
- Execution of Packed EXE Program
- Standard Process of Unpacking EXE
- Unpacking UPX using OllyDbg
- DEMO - Unpacking UPX
- Anti Anti-Debugging Plugins
- References

# 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 - After Packing

```
IDA - C:\Users\Administrator\Desktop\UPX Unpacking\putty_upx.exe - [IDA View-A]
File Edit Jump Search View Debugger Options Windows Help
[Icons] [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 ; 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 0FFFFFFFh, 40348D56h, 8B02E6C1h, 471E648Eh, 68868D00h
UPX1:00447000 ; DATA XREF: UPX1:0047F171j
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, 0BF6FF67Eh, 474567Ch
UPX1:00447000 UPX1:00447000 dd 5EEBC033h, 0C70FF2Bh, 4F60651h, 7BF88C86h, 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, 0FEFF6C68h, 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]
UPX1:00447108 retf 1C8Dh
UPX1:0044710E ; -----
UPX1:00447110 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 007D9CE86h, 40FAEF0h, 6A57293Dh, 72C66203h, 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?

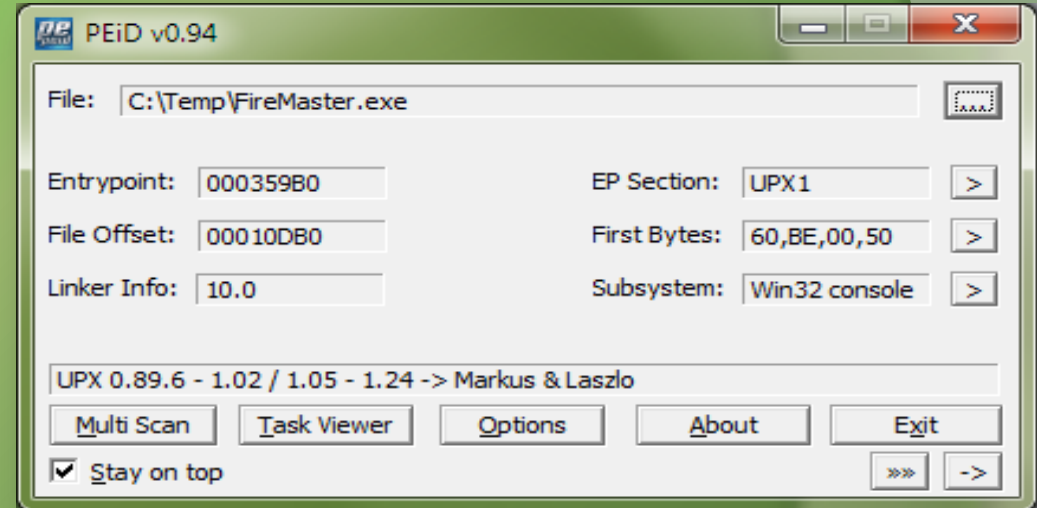
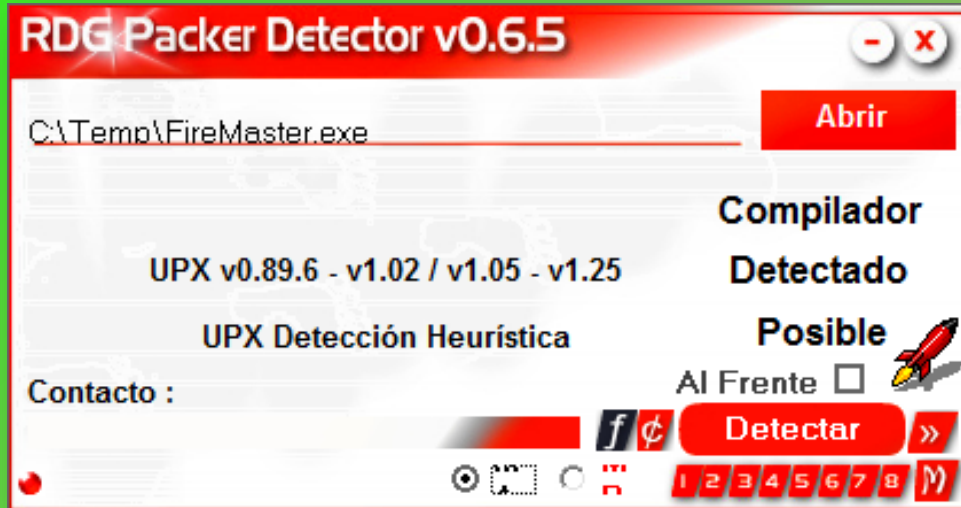
- **Extracting the Original Binary from the Packed Executable File.**
- **Automatic Unpackers available for popular Packers.**
  - **May not work with different versions**
  - **Not available for Complex packers**
- **Involves Live Debugging by Defeating Anti-Debugging techniques**

# 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

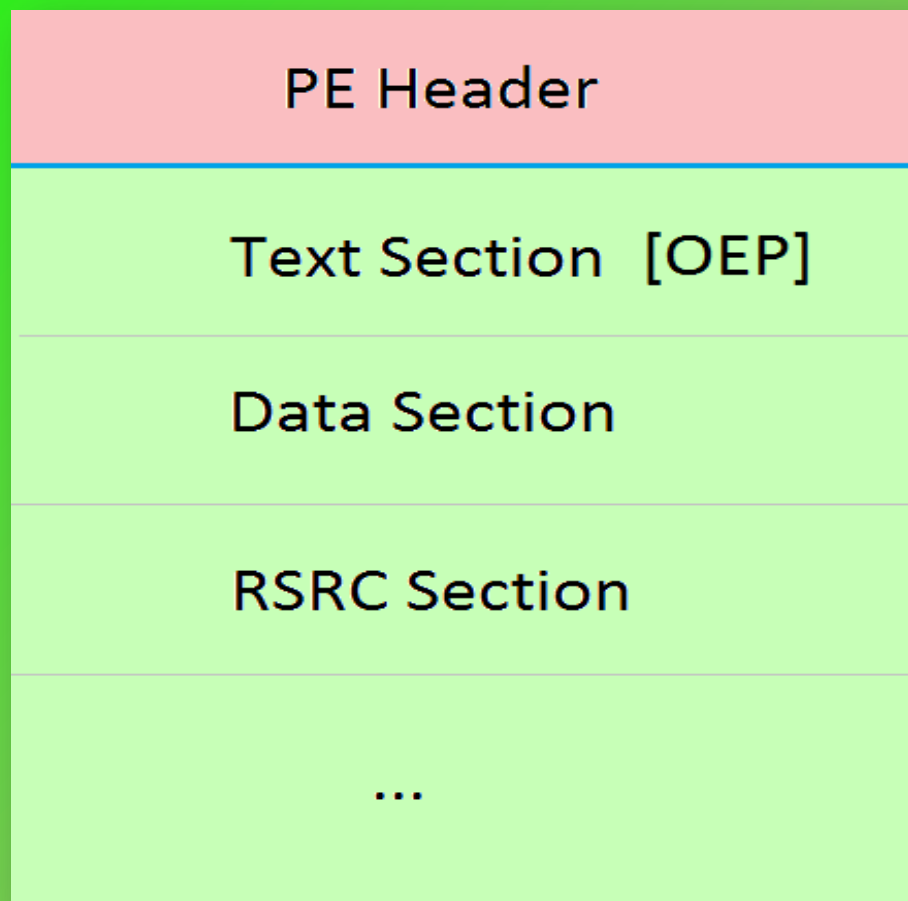


# Packer Detectors

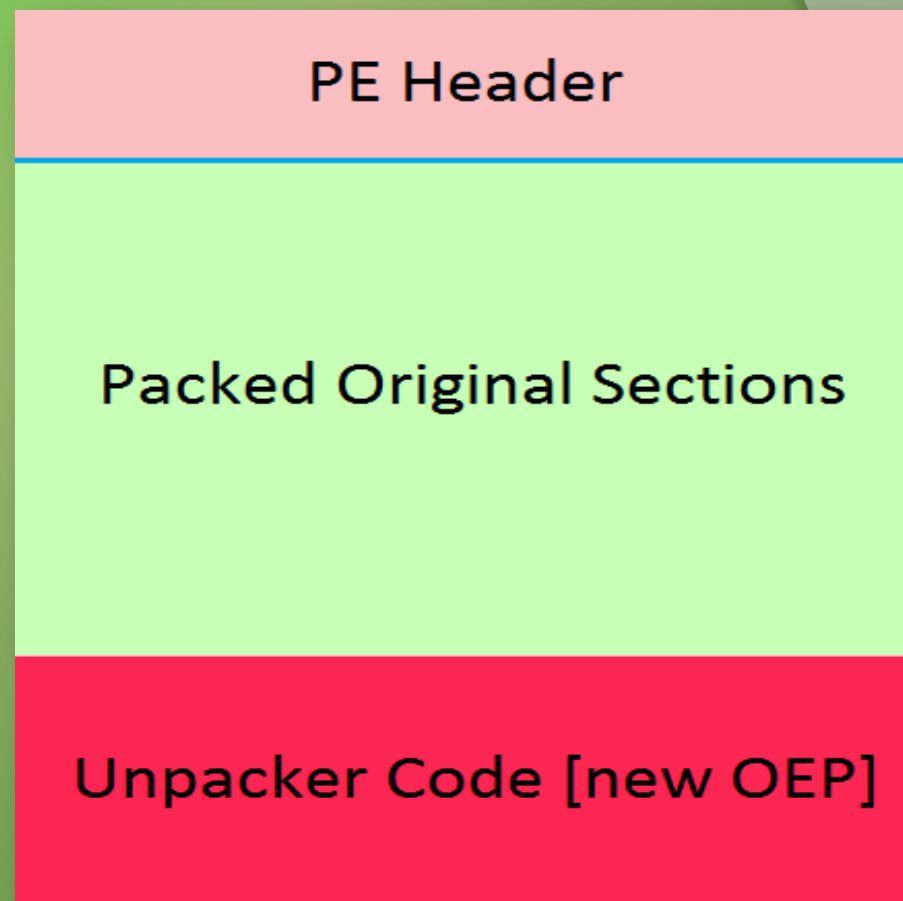




# Structure of Packed EXE



**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

<http://vimeo.com/42197903>

The screenshot shows a debugger window with the following components:

- Assembly View:** A list of instructions with addresses and hex values. The instruction at address 77C41617 is highlighted in red: `CALL DWORD PTR FS:[C0]`.
- Registers (FPU):** A list of registers and their values. The EIP register is highlighted in red and shows the address 77C4161E.
- Memory Dump:** A table with columns for Address, Hex dump, and ASCII. The hex dump shows a sequence of bytes, and the ASCII column shows the corresponding characters.
- Command Line:** A text box containing the command `Debugged program was unable to process exception`.
- Status Bar:** A yellow bar at the bottom right indicating the program is **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](http://www.SecurityXploded.com)**