REPORT ON UNPACKING OF BOOM_WIN_PE_PACKED.EXE

Ali Abdelhamid El Fekki
Aa2100274@tkh.edu.eg

Table of Contents

Introduction:

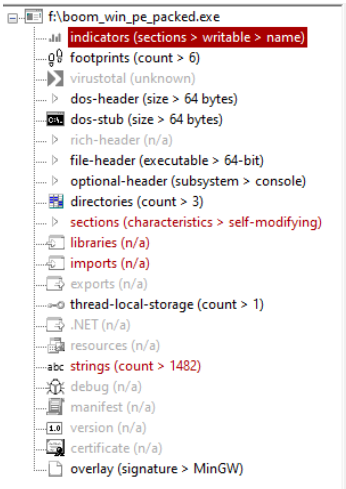
This report provides a comprehensive description of the manual unpacking process for the executable file named 'boom_win_pe_packed.exe', which was suspected to be packed using UPX. The process documented here outlines the identification of the packing method and the subsequent steps taken to unpack the file manually without relying on the automated UPX tool.

Methodology and Tools:

The unpacking process utilized several tools known for their efficacy in reverse engineering and debugging executables. PEStudio was initially used to inspect the file, suggesting a UPX packing signature. DiE (Detect it Easy) provided a secondary confirmation of the UPX packer. The primary debugging tool used was x64dbg, which facilitated a dynamic analysis of the executable. The Scylla plugin within x64dbg was crucial in reconstructing the Import Address Table (IAT) and finalizing the unpacking process.

Initial Identification of Packing Method

1. Analysis with PEStudio: Initial examination of the executable was conducted using PEStudio. This tool provided an overview of the file structure and indicated signs typical of a UPX packed file, such as:
    - Writable sections named **UPX0**.
    - An entry point is located at **413D75**.
    - No imports or exports, which is common in packed executables to evade analysis.
    - An overlay size of **27785 bytes**, which can sometimes contain additional packed data.



indicator (23)	detail	level
sections > writable > name	UPX0	1
entry-point > location	0x0902A093	1
sections > executable > count	2	1
sections > self-modifying > name	UPX0 + UPX1	1
libraries > count	0	1
imports > count	0	1
overlay > size	27785 bytes	2
overlay > entropy	4.007	2
file > checksum	0x00000000	2
sections > virtualized	UPX0	2
groups > API	execution, dynamic-library, memory, file, exception, diagnostic, reconn...	2
mitre > technique	T1045, T1106, T1055, T1497, T1124, T1057	2
file > entropy	5.512	3
file > footprint	4AB13F66C9BBB05404D669E901C199A4A4899E6F230D19C8D3A4F3AA71...	3
file > size	40585 bytes	3
file > score > error	The requested resource is not among the finished, queued or pending s...	3
file > tooling	MinGW	3
security > protection	data-execution-prevention (DEP) > OFF	3
security > protection	control-flow-guard (CFG) > OFF	3
security > protection	address-space-layout-randomization (ASLR) > OFF	3
file > subsystem	console	3
thread-local-storage > callbacks	1	3
security > protection	code-integrity (CI) > OFF	3

2. Entropy Check with DiE (Detect it Easy):

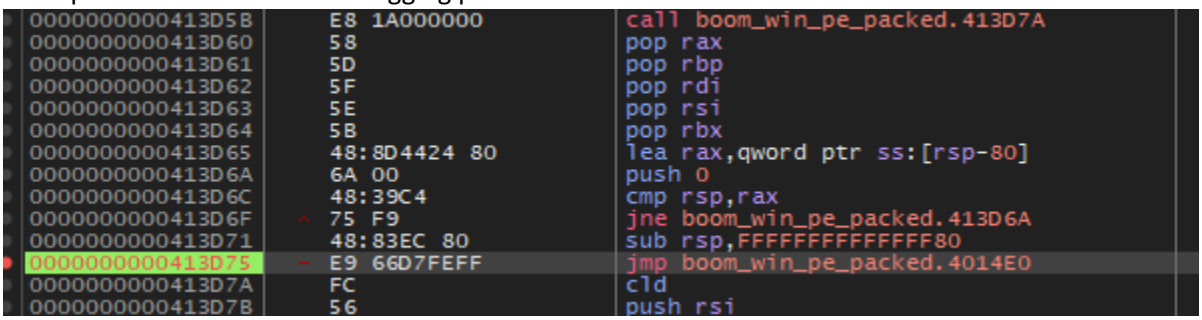
   Entropy analysis was performed using DiE, which showed a value of **4.794**. High entropy is indicative of compressed or encrypted data, reinforcing the suspicion that the executable was packed.

3. Further Analysis with PEStudio: The executable was scanned again with PEStudio, revealing:

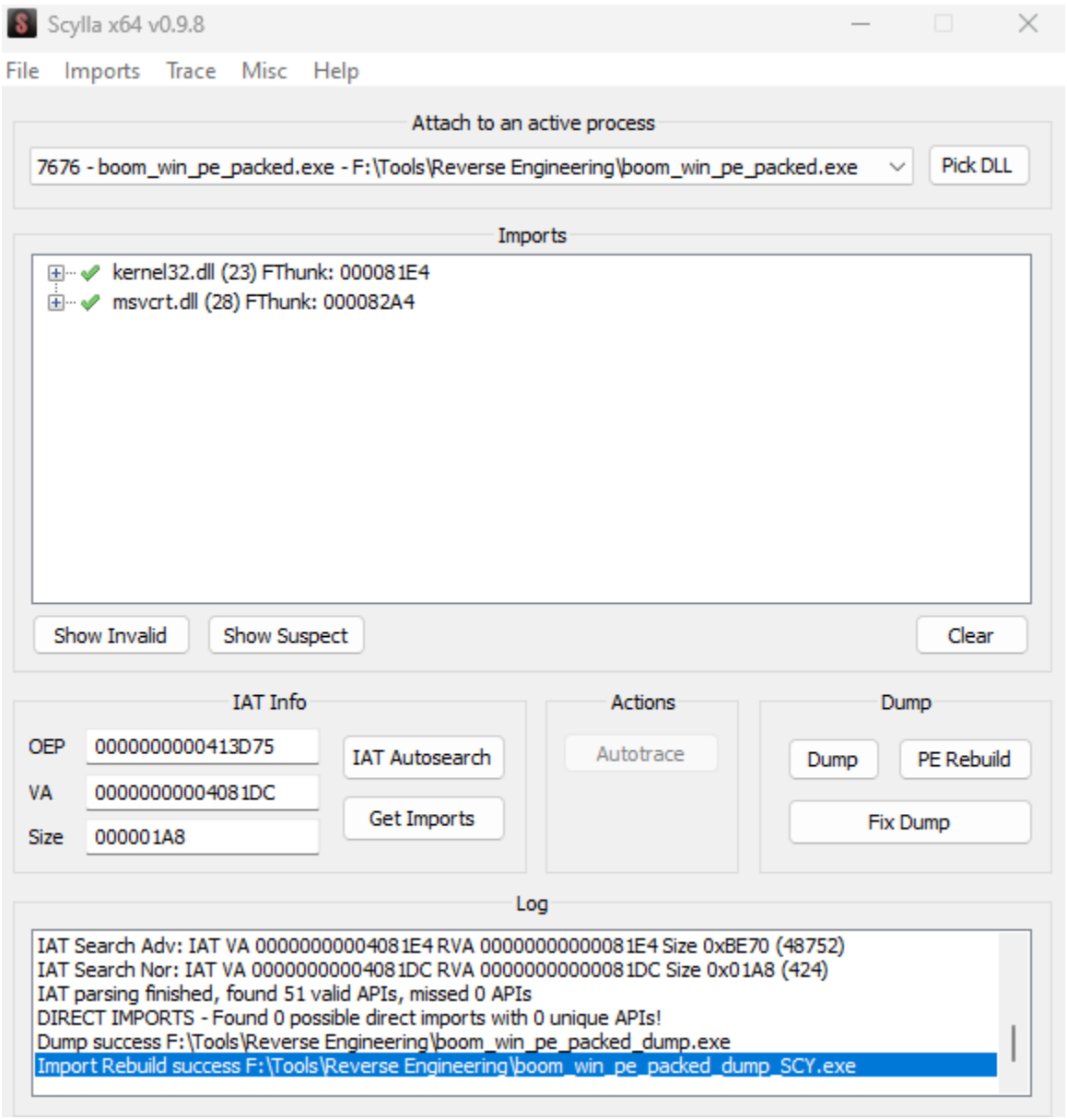
   - The absence of libraries typically used by unpacked executables.
   - A total of two executable sections, with self-modifying code present in **UPX0** and **UPX1**.
   - The lack of direct imports, which is a characteristic of a UPX packed file.

Manual Unpacking Process with x64dbg

1. **Loading the Executable**: The file "boom_win_pe_packed.exe" was loaded into x64dbg for debugging.
2. **Identifying the Entry Point**: The entry point was identified as a critical location to set a breakpoint and initiate the debugging process.



3. **Setting a Breakpoint and Running the Executable**: A breakpoint was set at the identified entry point, and the executable was run until it hit the breakpoint.
4. **Using Scylla Plugin for IAT Reconstruction**:
   - The Scylla plugin within x64dbg was utilized to locate the Original Entry Point (OEP).
   - An IAT autosearch was conducted to rebuild the Import Address Table (IAT).



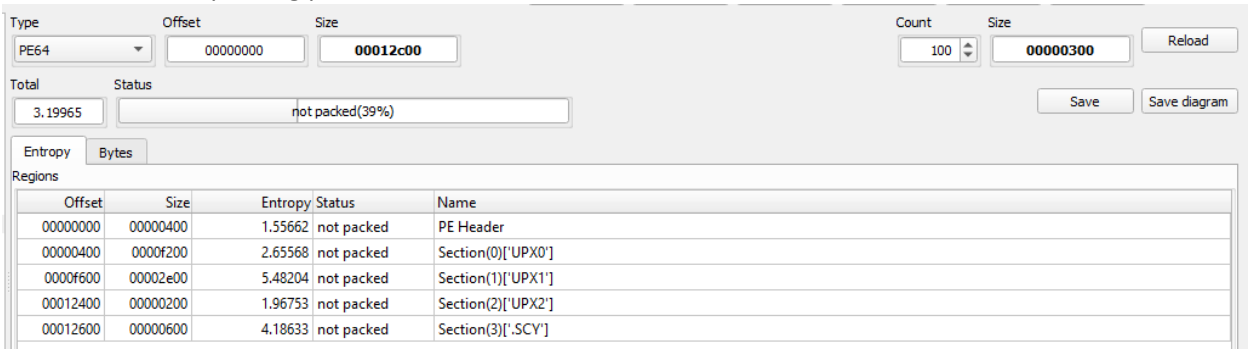
- Any suspicious imports marked by Scylla were removed to clean the IAT.

5. **Dumping the Process Memory**: With the correct OEP identified and the IAT cleaned, the process memory was dumped to create a new executable file.
6. **Repairing the Dumped File**: Scylla's 'Fix Dump' feature was used to repair the newly dumped file, resulting in a restored and functional executable named "boom_win_pe_packed_dump_SCY.exe".

Confirmation of Unpacking

1. **Comparing File Sizes**: The original packed executable, the dumped file, and the repaired file were compared. The increase in file size from the original to the repaired file confirmed the expansion post-unpacking.

2. **SHA-256 Hash Calculation**: The SHA-256 hash of the unpacked file was calculated to verify its integrity and confirm that the file contents had been altered through the unpacking process.
3. **Final Verification with PEStudio**: The fully unpacked executable was scanned one last time with PEStudio to ensure all signs of UPX packing had been removed, confirming the success of the manual unpacking process.



Conclusion

The boom_win_pe_packed.exe file was conclusively identified as a UPX-packed executable using PEStudio. This was confirmed by several indicators, including specific section names, high entropy values, and an empty import table. Through the detailed steps outlined above, "boom_win_pe_packed.exe" was successfully unpacked manually. This process required a deep understanding of the executable structure, UPX packing mechanisms, and the use of advanced debugging tools to reverse-engineer the file to its original state.