Reverse Enginering

Section 1: Introduction

1.1 Initial Analysis with Detect It Easy (DIE)

1.2 Identifying UPX Packing

1.3 Entropy Analysis for UPX Confirmation

Section 2: Main Content

2.1 Setting Up x64dbg for Debugging

2.1.1 Configuring Debugging Options

2.1.2 Running the Program to First Entry Point

2.2 Analyzing the Unpacked Code

2.2.1 Identifying the End of UPX Compression

2.2.2 Locating and Utilizing the 'jmp' Instruction

2.3 Extracting and Modifying the File

2.3.1 Using 'IAT Autosearch' and 'Get Imports'

2.3.2 Removing the 'Fthunk' Import

2.3.3 Dumping and Fixing the Modified File

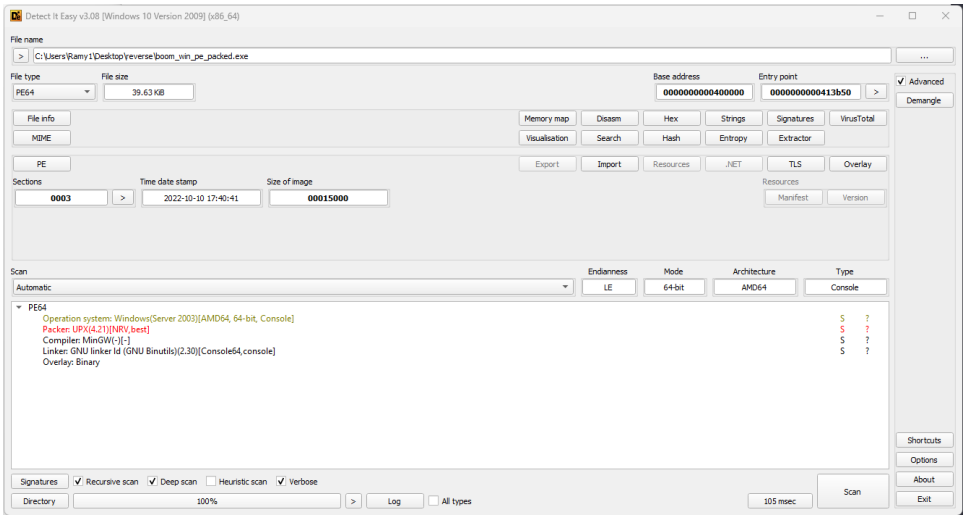
2.4 Verifying the Removal of UPX

2.4.1 Reanalysis with Detect It Easy

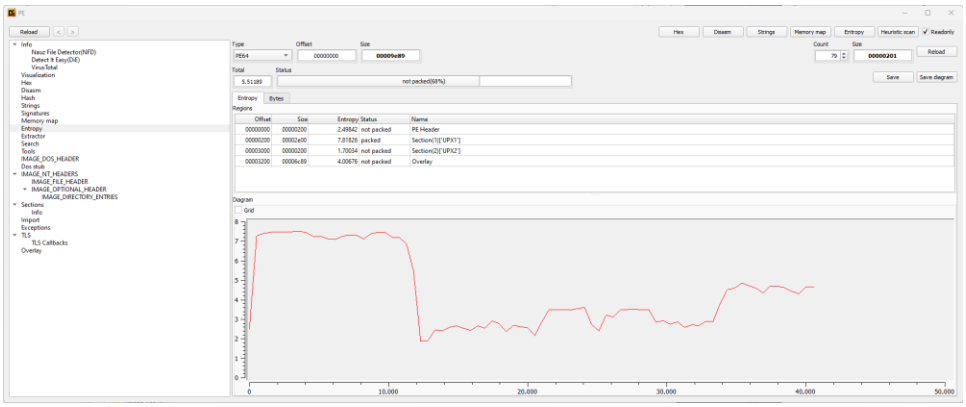
Section 3: Conclusion

3.1 Summary of Key Findings

Section 1: Introduction

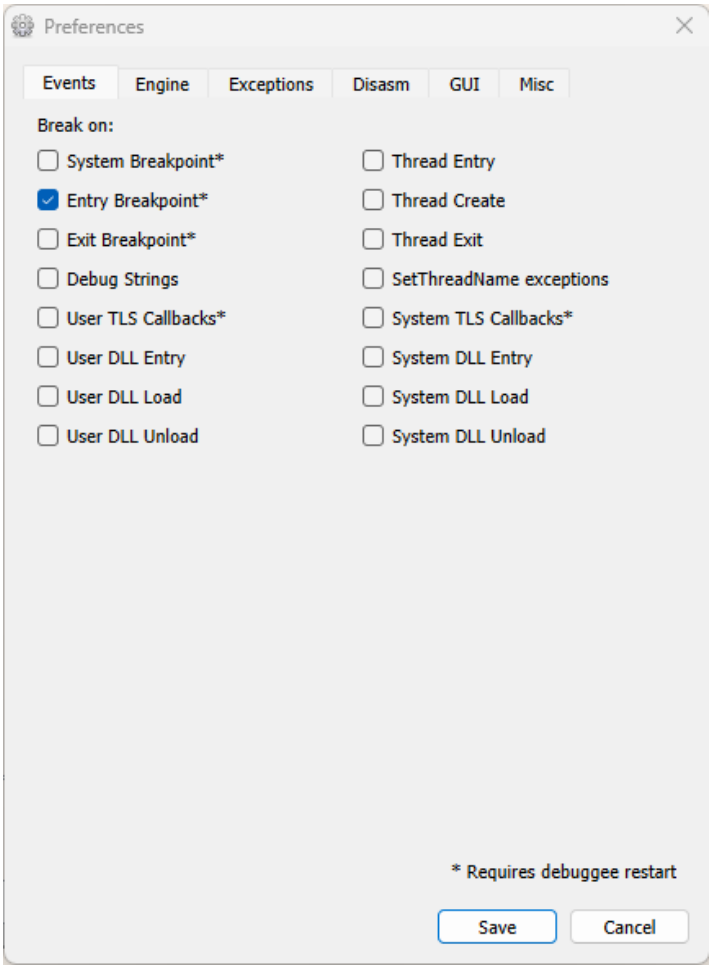


Upon opening the .exe file in Detect It Easy (DIE), a tool adept at identifying file properties, we quickly ascertained that the file was packed using UPX (Ultimate Packer for eXecutables). DIE's analysis revealed characteristic UPX signatures, such as specific headers and a high entropy level, indicating compression and potential obfuscation. This initial finding was crucial, as it informed us that the file needed to be unpacked to reveal its original code and structure for further analysis.

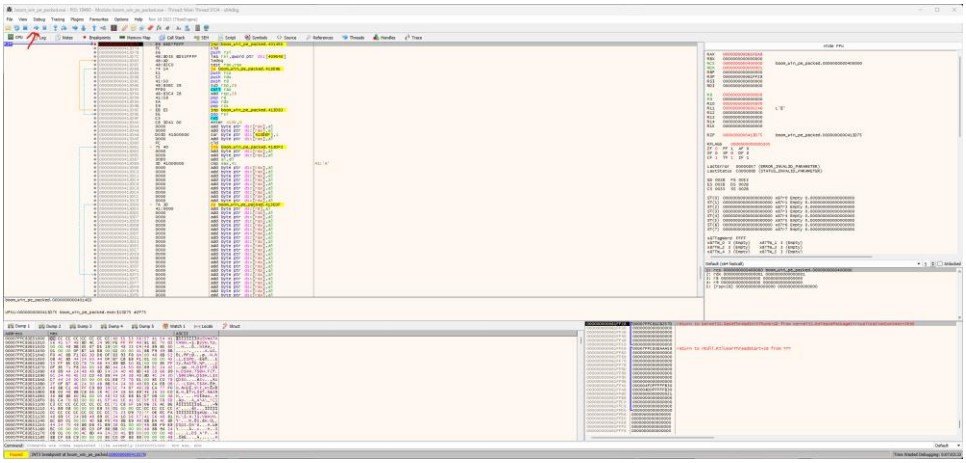


Further investigation using DIE's entropy analysis tool confirmed that Section 1, labeled "UPX 1," was indeed packed. The entropy option highlighted this section's high compression level, typical of UPX packing, reinforcing our initial findings and guiding our next steps in the unpacking process.

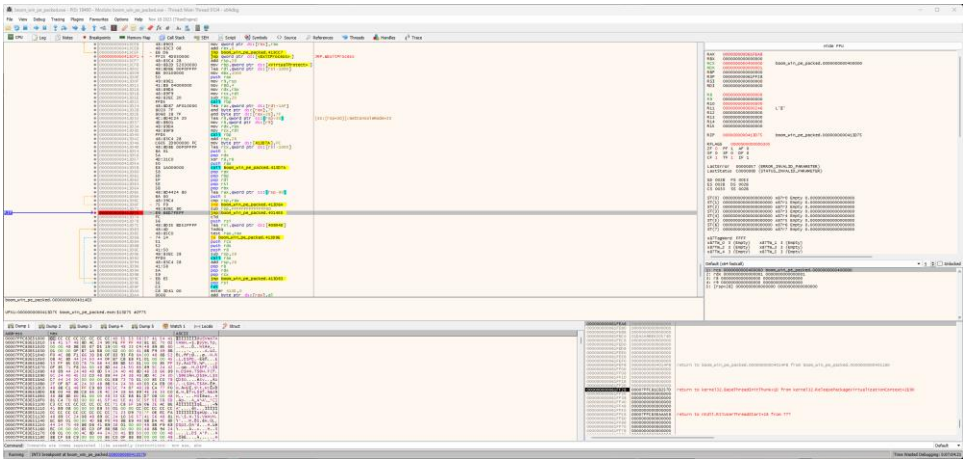
Section 2: Main Content



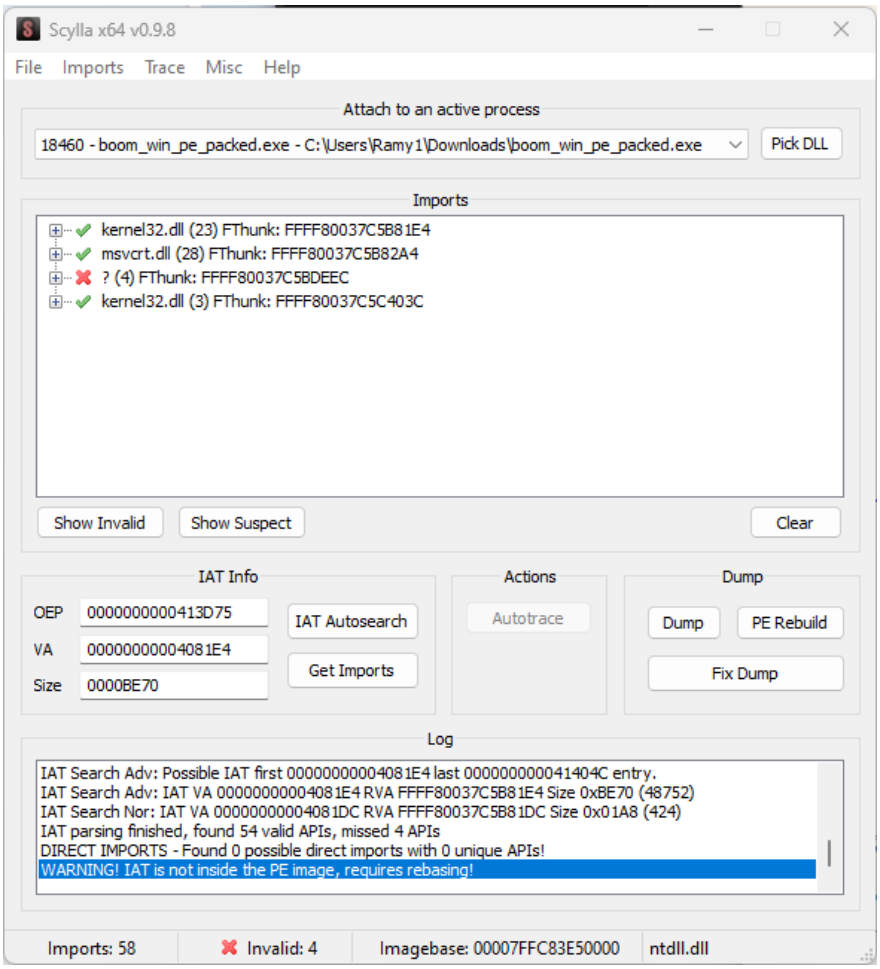
Next, we switched to x64dbg, a powerful debugger, where we accessed the options menu to disable all settings except for 'Entry Breakpoint'. This configuration was crucial for focusing our analysis on the program's initial execution point.



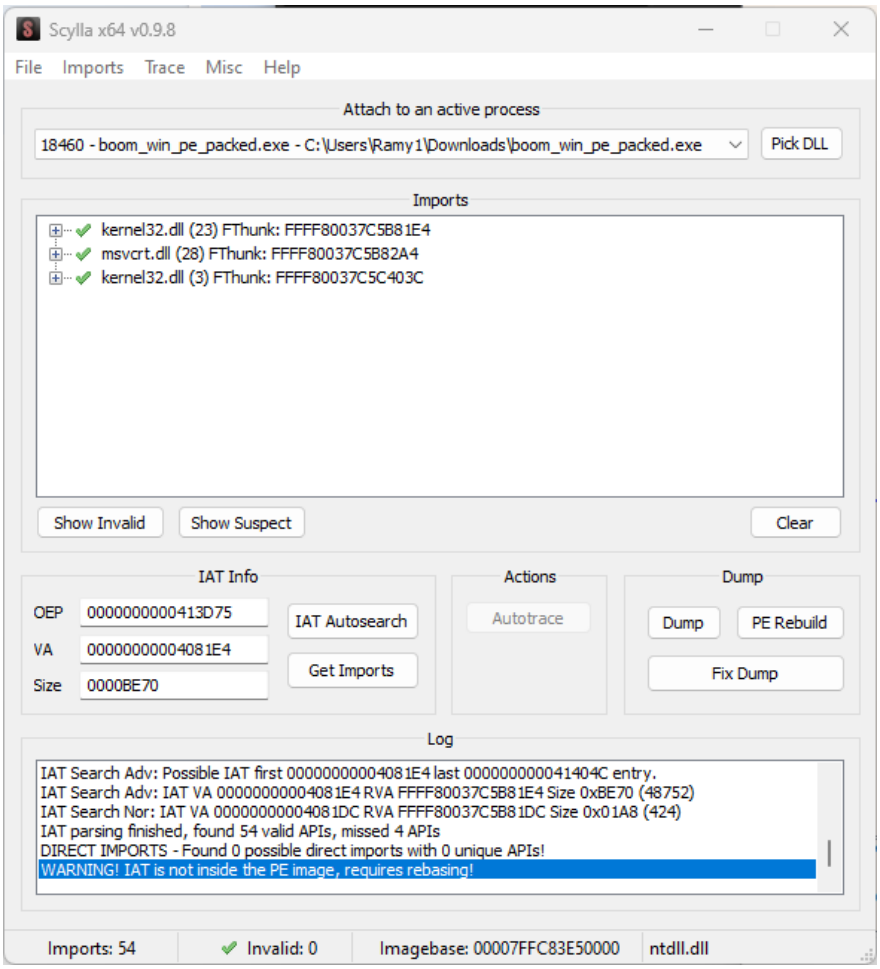
With the options configured in x64dbg, we proceeded to run the program, aiming to pause it at the first entry point for detailed analysis.



After seeing the first breakpoint we are going to manualy scroll down till we see a lot of POP commands this indicates that the upx compression has stopped. We are going to search for a jmp, we are then going to use the address of the jmp for the next step

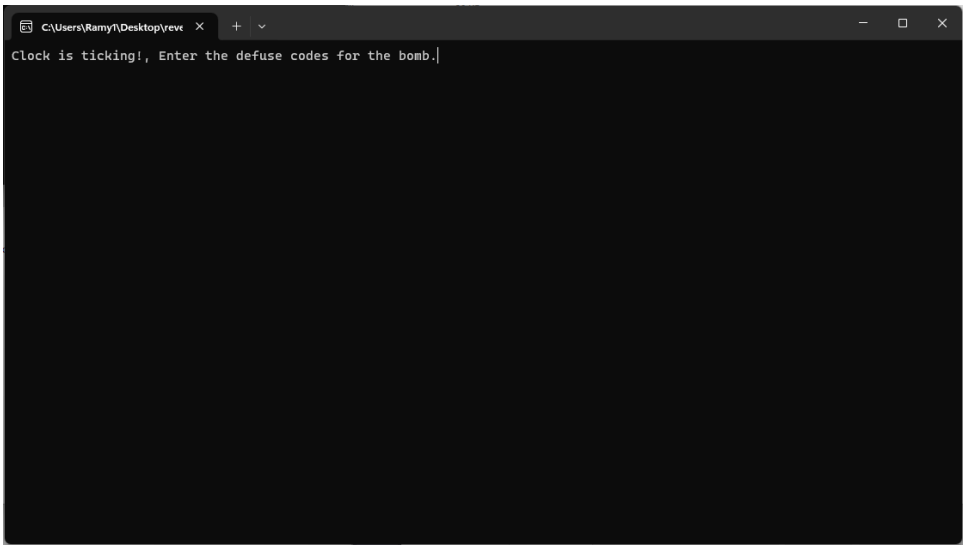


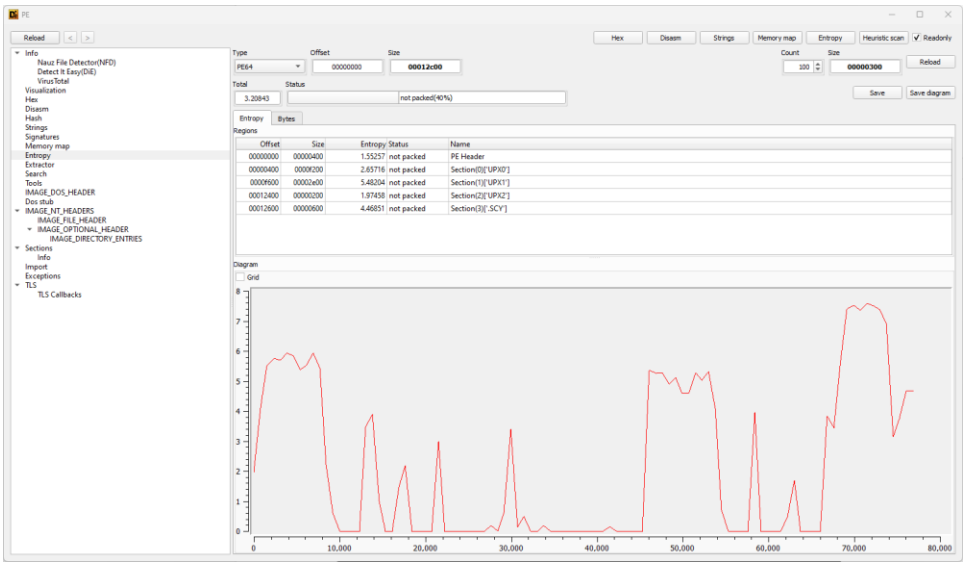
Next, we used the copied address in x64dbg to utilize the 'IAT Autosearch' feature, followed by 'Get imports' to retrieve all import addresses, focusing specifically on locating the 'Fthunk'.



After removing the 'Fthunk' import, we dumped the modified file to our folder. Encountering functionality issues with the dumped file, we then utilized the 'Fix dump' feature in x64dbg to rectify these issues.

The modified file, now named with the original name plus '_SCY', indicates successful alteration by x64dbg. This new file is free of UPX packing, confirming the effectiveness of our unpacking process.





We can also verify that the upx is removed from the file using DIE again but this time there is no .upx anymore

Section 3: Conclusion

- **UPX Packing Identified**: The initial analysis using Detect It Easy (DIE) revealed that the .exe file was packed with UPX (Ultimate Packer for eXecutables). This was a crucial discovery, as UPX packing often obscures the true content of the file, necessitating unpacking for further analysis.
- **Entropy Analysis Confirmation**: Further investigation with DIE's entropy analysis tool confirmed the presence of UPX packing in Section 1, labeled "UPX 1." This reinforced the initial

findings and provided a clear direction for the subsequent unpacking process.

- **Effective Use of x64dbg**: The transition to x64dbg, a powerful debugger, allowed for a more in-depth analysis. By configuring the tool to focus on the program's initial execution point and running the program to the first entry point, we could closely examine the unpacking process.

- **Unpacking Process Detailed**: The analysis revealed a series of 'POP' commands in x64dbg, indicating the end of UPX compression. A 'jmp' instruction was identified and used for further steps, showcasing a methodical approach to reverse engineering.

- **Modification and Extraction**: Utilizing the 'IAT Autosearch' and 'Get imports' features in x64dbg, we successfully located and removed the 'Fthunk' import. This step was crucial in modifying the file, which was then dumped and fixed using x64dbg's 'Fix dump' feature.

- **Successful Removal of UPX**: The final file, renamed with '_SCY' to indicate its modification by x64dbg, was confirmed to be free of UPX packing. This was verified through a reanalysis with DIE, ensuring the effectiveness of the unpacking process.

- **Preservation of File Integrity**: Throughout the reverse engineering process, the integrity of the original file was maintained. This was crucial for ensuring that the file remained functional and true to its original design, post-modification.