# CMSC 491/691 Malware Analysis HW 6

Name: Daniel Roh
Assigned: 4/20/2020
Due: 4/29/2020 by 5:00pm

Download the OllyDumpEx plugin and ImportREC.7z from the course website onto your Flare VM. Place OllyDumpEx_Imm18.dll in C:\Progam Files (x86)\Immunity Inc\Immunity Debugger\. Unzip ImportREC.7z. The password is "imprec". Download hw6.7z to your Flare VM and extract it. The password is "infected".

**Place your VM into internal network mode.** Take a snapshot of your VM so you can easily revert to a clean state.
There are 105 points available on this homework and it is graded out of 100.
*Hint: Chapter 18 of PMA is a great reference for this homework!*

**Note to self:** ***Run ImportREC as admin*** *to avoid wasting a hour (T_T)*

**Part 1: Unpacking hw6_1.exe (35 pts)**

1) **Answer the following questions about hw6_1.exe:**
    a. **What is the address of the entrypoint? (3 pts)**
    Entry Point: 0041d001

    b. **What section do you believe contains the unpacking stub? Why? (4 pts)**
    Unpacking stub section: .aspack

    I believe this is where the stub is because in ghidra, the entrypoint (calculated by the RVA + Offset) falls inside the address range that ghidra reports for the section called .aspack

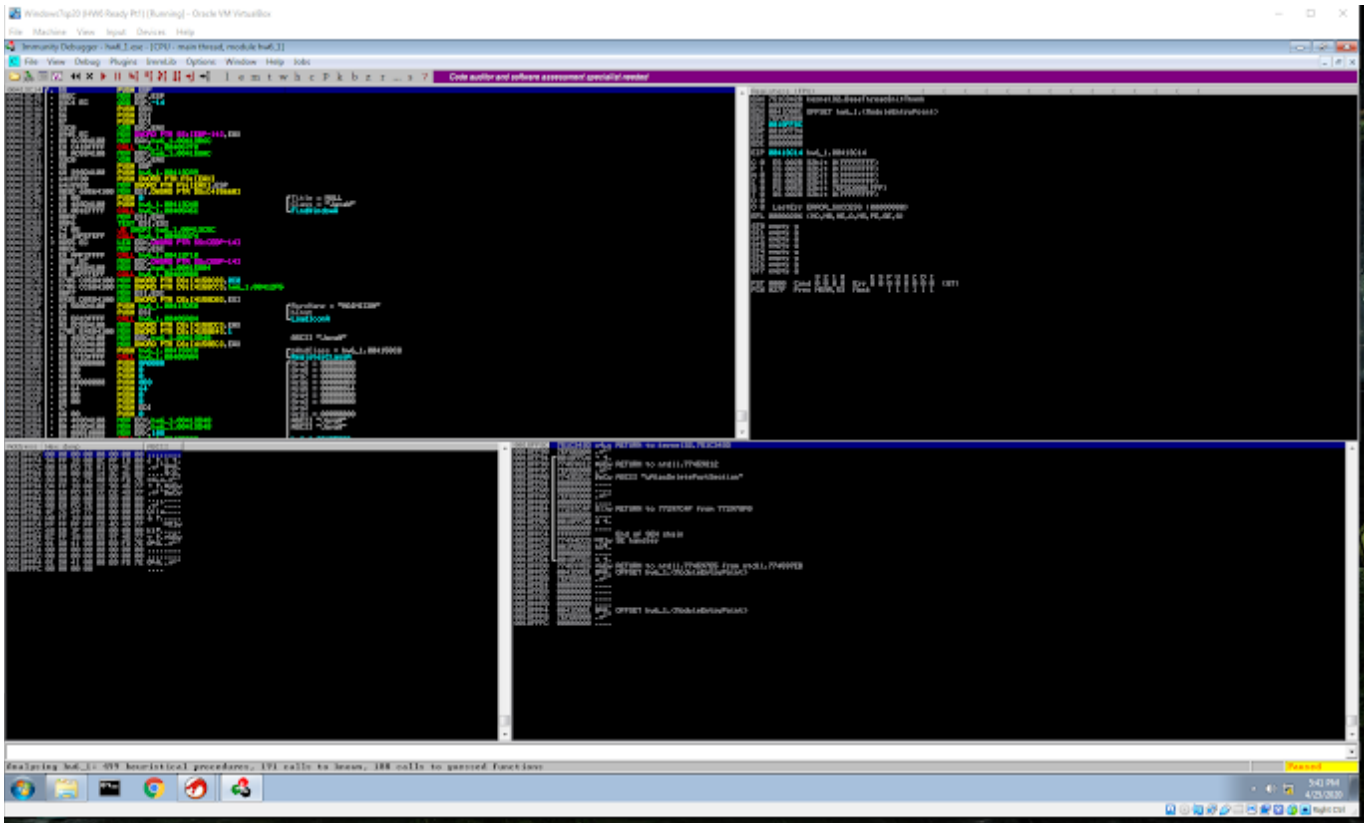
    c. **What section do you believe contains the packed data? Why? (4 pts)**
    I believe that the Section 0 "CODE" contains the packed data.

    This is because in Die, the section CODE has an entropy of 7.98, which is higher than 7, indicating packed data. But the sections .idata, and .rsrc likely also contain packed data due to the entropy also being higher than 7 for these sections.

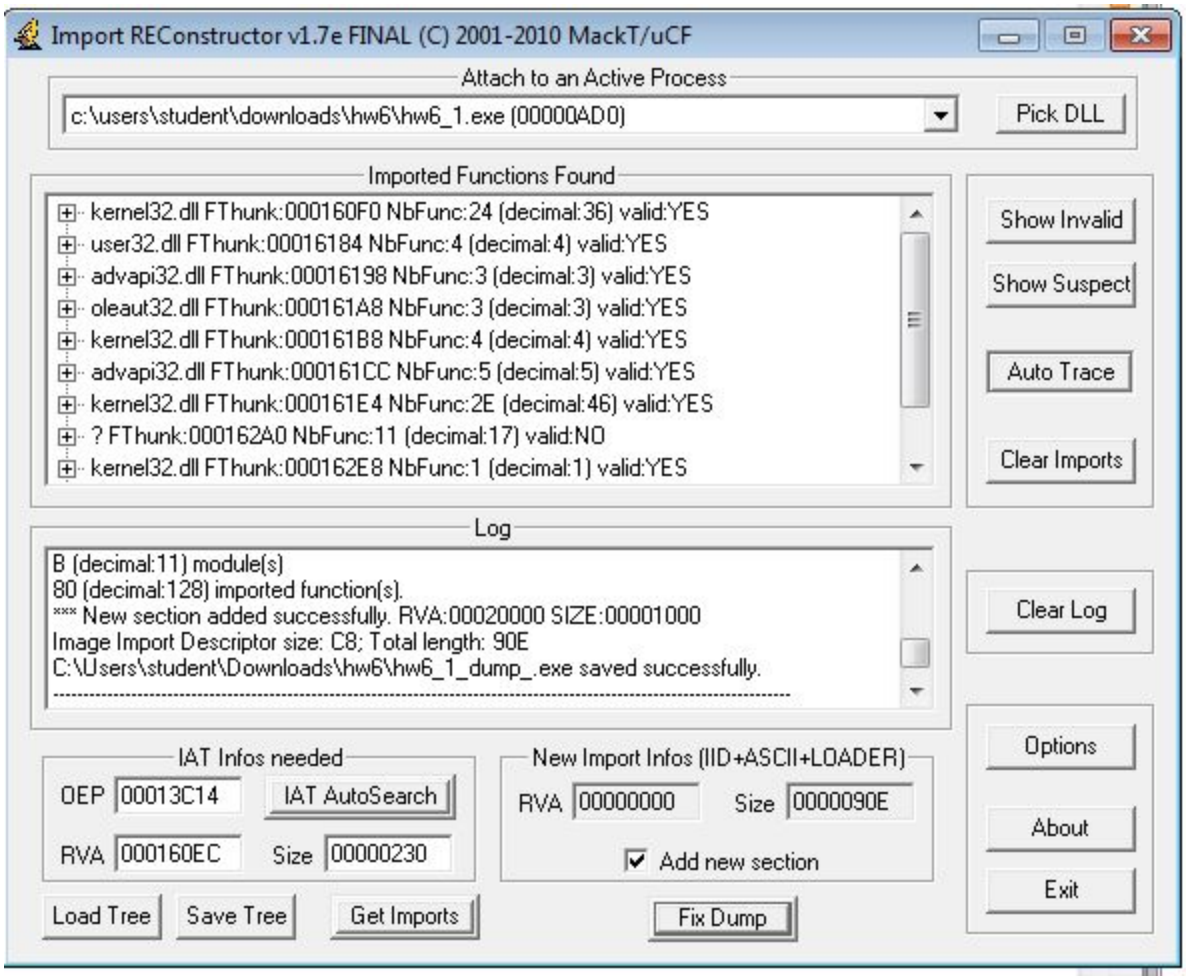
2) **Using one of the methods described in class, find the OEP of hw6_1.exe. In a few sentences, describe how you did this. Provide a screenshot of Immunity debugger with execution paused at the OEP. (4 pts)**
    OEP: 00413C14

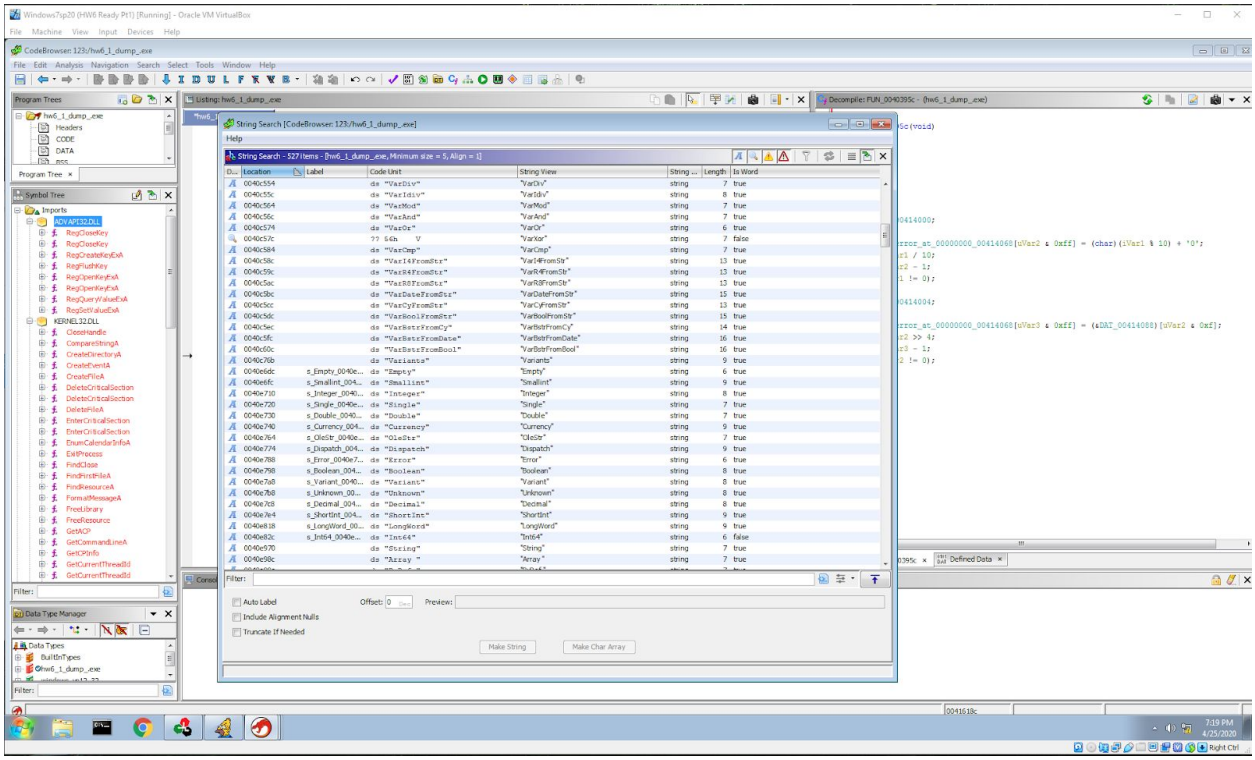
    This was found by taking a look at the disassembled code in ghidra. From the entry point, only two function calls were found, one of which likely the unpacking code. Taking a look at this point on immunity, a call "PUSHAD" was found which after taking a look at the ESP register. Indicated that possibly that 18FF6C would be around where the unpacking is completed. Then going to the

memory view, and finding the address, placed a hardware breakpoint. After reaching the breakpoint, I stepped though, until a jump to what look was data was made. Then I asked immunity to analyze this data to then turn the “data” back into code.



3) **Use OllyDumpEx to dump the unpacked hw6_1.exe and then fix its IAT using ImpRec. Open the unpacked malware in Ghidra and provide screenshots showing that it has been unpacked and that its IAT has been fixed. How many imports does the unpacked program have? (4 pts)**
Imports: 5 DLL's
   - Functions: 128





## Part 2: Unpacking hw6_2.exe (35 pts)

1) **Answer the following questions about hw6_2.exe:**
   a. **What is the RVA of the entrypoint? (3 pts)**
   RVA: 00018000

   b. **What section do you believe contains the unpacking stub? Why? (4 pts)**
   Unpacking stub section: sunray

   This is because the entry point of the code (calculated by the RVA + Offset) starts here and a unpacking stub is normally the first thing that is run in the program. Ghidra also shows that the entry point begins at a section called “sunray”

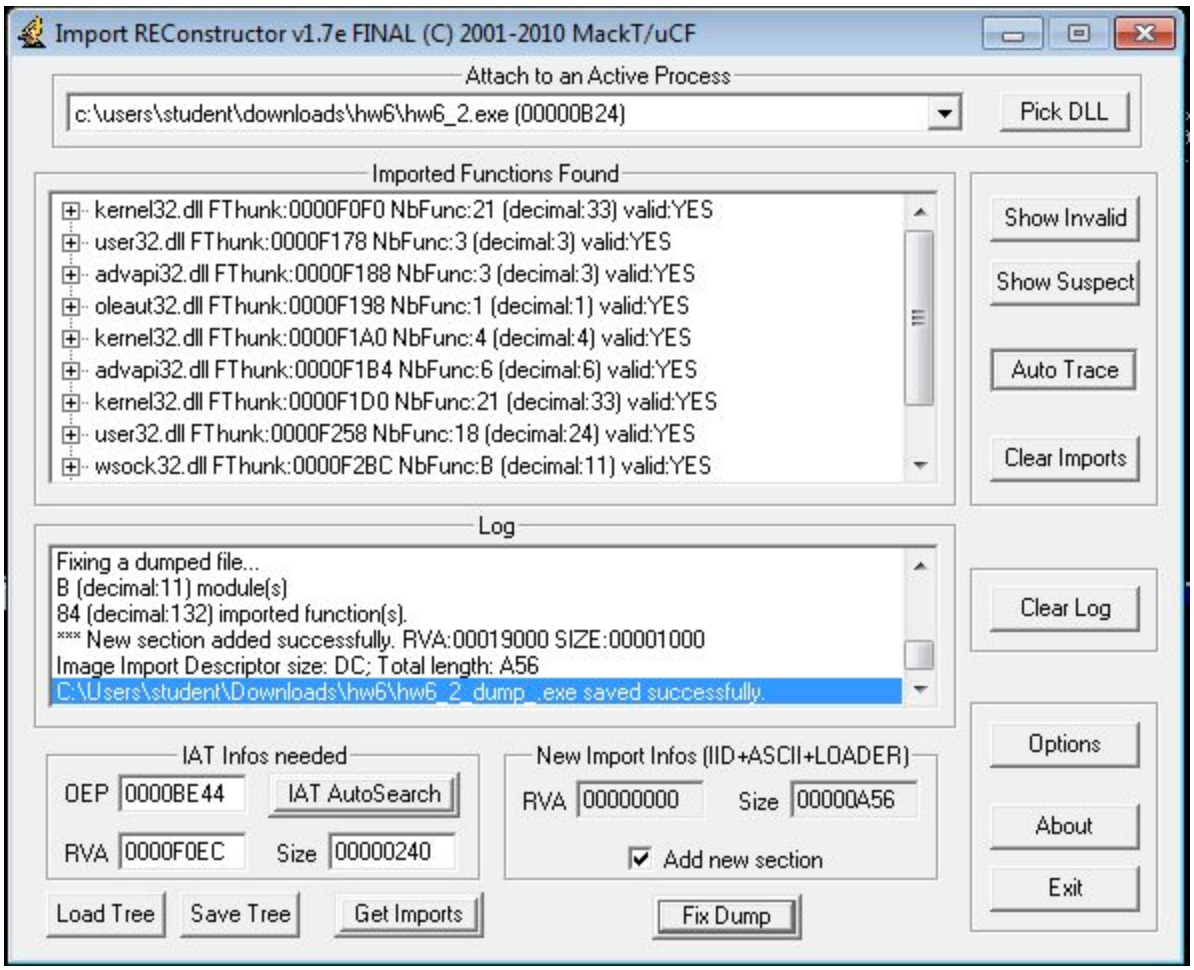
   c. **What section do you believe contains the packed data? Why? (4 pts)**
   Section 1 “UPX1”

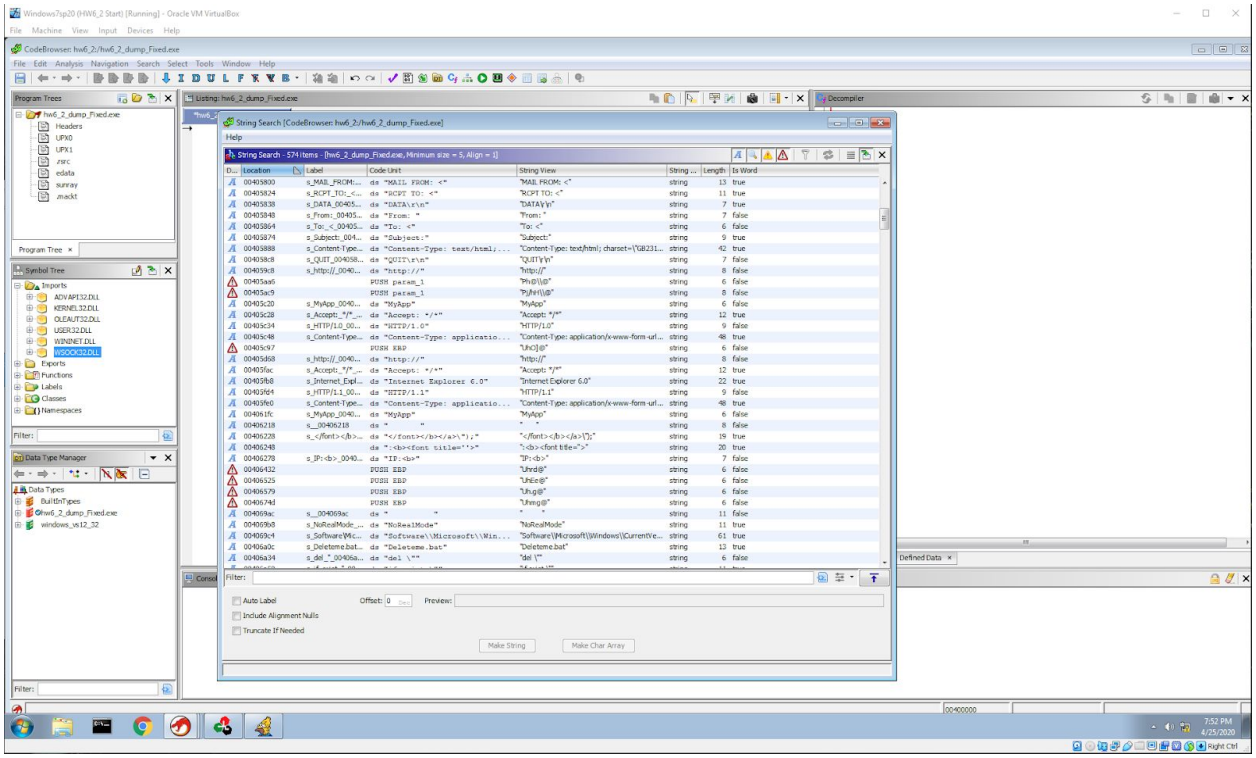
   This is because UPX1 has a high entropy of 7.89, which a indicator of packing

2) **Using one of the methods described in class, find the OEP of hw6_2.exe. In a few sentences, describe how you did this. Provide a screenshot of Immunity debugger with execution paused at the OEP. (12 pts)**
   OEP: 0000BE44

To find this, I stepped into the code one line at a time until I found a PUSHAD instruction. After finding the address that the PUSHAD stored and placing a hardware breakpoint at the memory address 18FF58 that was stored in hopes of stopping after the code was unpacked. Then after running the program, the hardware breakpoint was hit, and I stepped through until the jmp instruction to a far away address “0000BE44” was taken in which that was where the OEP was located

3) **Use OllyDumpEx to dump the unpacked hw6_2.exe and then fix its IAT using ImpRec. Open the unpacked malware in Ghidra and provide screenshots showing that it has been unpacked and that its IAT has been fixed. How many imports does the unpacked program have? (12 pts)**
Imports: 6
   - Functions: 132





## Part 3: Unpacking hw6_3.exe (35 pts)

1) **Answer the following questions about hw6_3.exe:**
   a. **What is the RVA of the entrypoint? (3 pts)**
   RVA: 00001018

   b. **What section do you believe contains the unpacking stub? Why? (4 pts)**
   The section called “PS_____”

   This is because the RVA + the offset gives the address 321018 which is inside the section “PS____”. Given that the unpacking stub is called in the beginning, this is where I believe the stub is contained

   c. **What section do you believe contains the packed data? Why? (4 pts)**
   Section 1 “” is where i believe contains the packed data

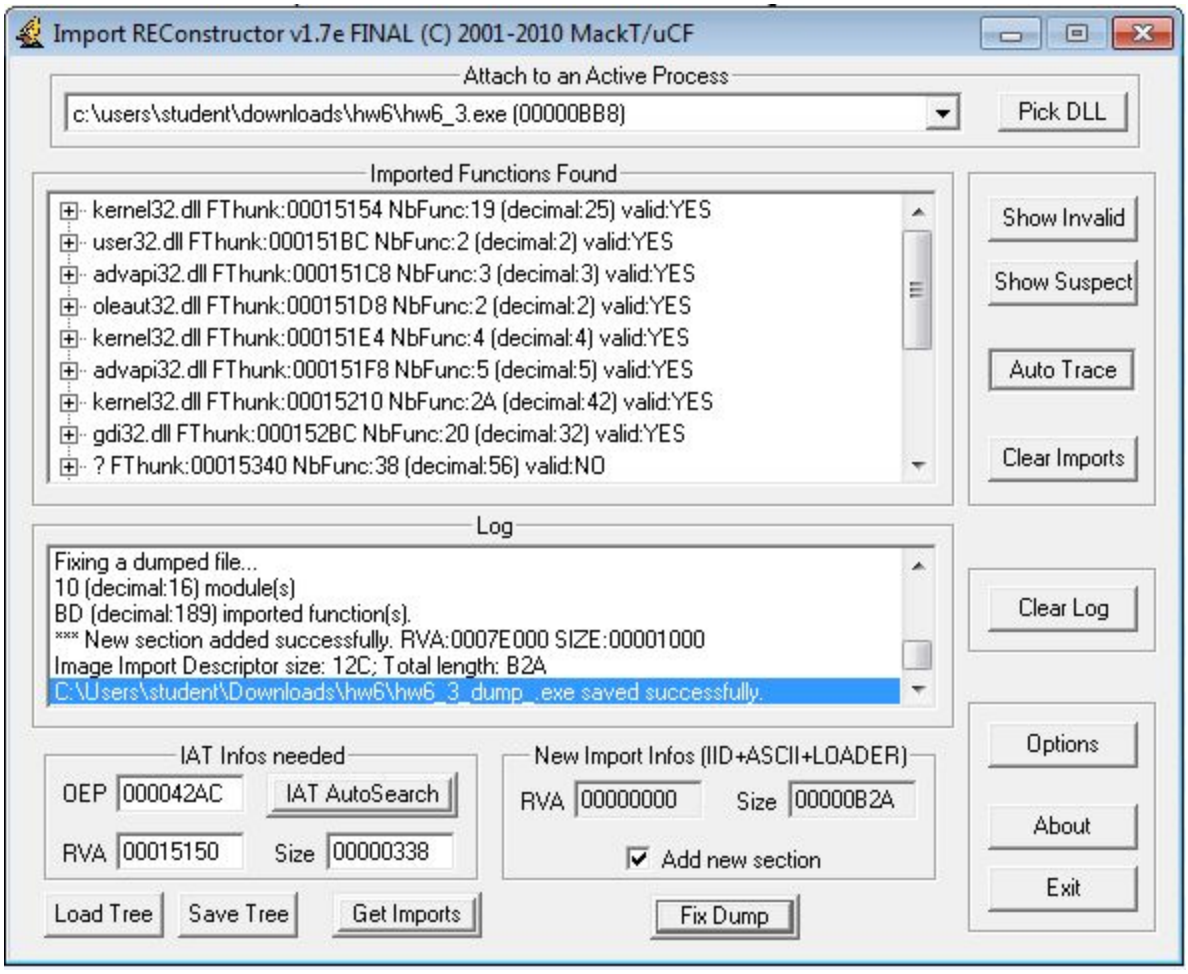
   This is because “” ‘s entropy is 7.87, which is a indicator that this is where the data has been packed

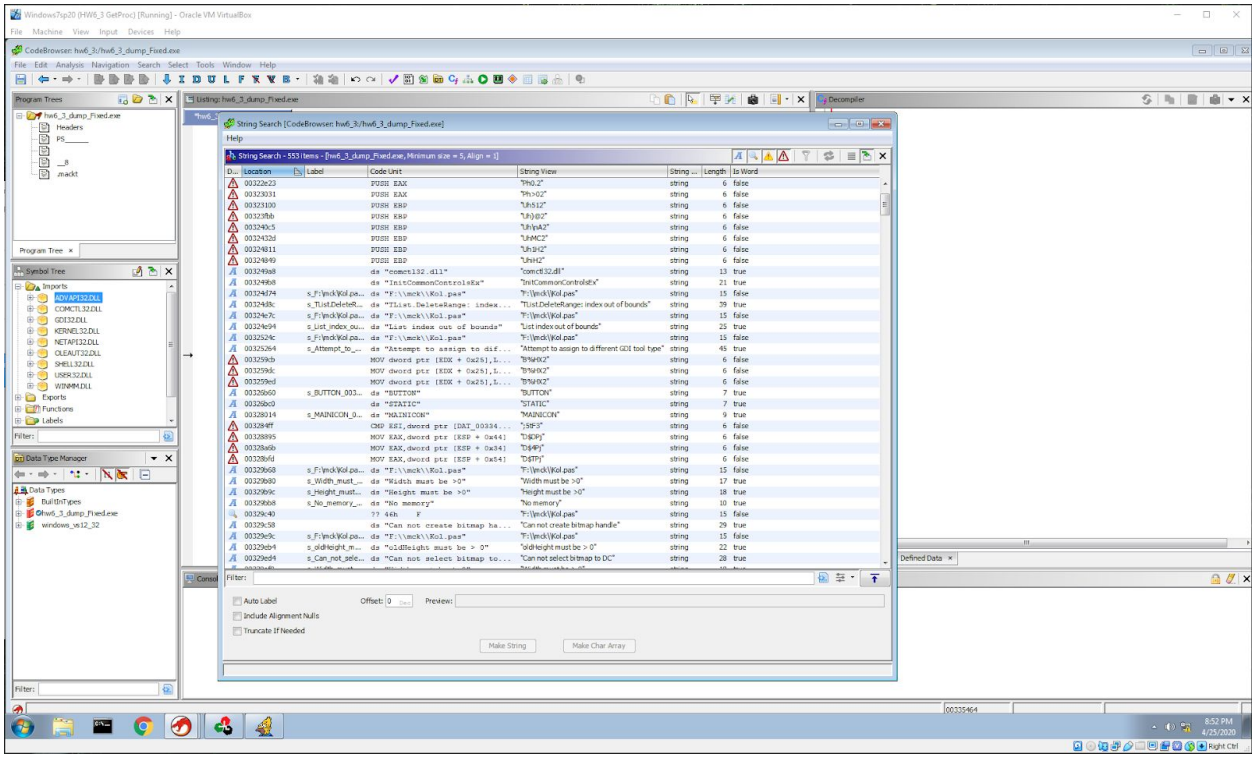
2) **Using one of the methods described in class, find the OEP of hw6_3.exe. In a few sentences, describe how you did this. Provide a screenshot of Immunity debugger with execution paused at the OEP. (12 pts)**
   ***Hint: The last call to LoadLibraryA before the OEP is made from hw6_3.00395D87 when the FileName argument is WINMM.DLL. After this, you will only need to break once on GetProcAddress.***
   OEP: 000042AC

   To find this, I started by finding the address of LoadLibraryA and LoadLibrary W and setting a

breakpoint for both. Then I ran the program until I hit the LoadLibraryA call for WINMM.DLL (due to a hint given from RJ). From there, I removed the breakpoints set for LoadLibraryA and LoadLibraryW, and set a breakpoint for GetProcAddress. Then I ran the program once to break on GetProcAddress (as the hint suggested). Then I stepped through the GetProcAddress until I reached the RET which brought me to an address that was outside of the setup code. From there, I ran single stepping through the code until I found a JMP instruction that was far away. After going through some JMP's, a JMP instruction took me to 000042AC which is relatively far from the location from where I was currently at. This JMP to 000042AC ended up being the OEP

3) **Use OllyDumpEx to dump the unpacked hw6_3.exe and then fix its IAT using ImpRec. Open the unpacked malware in Ghidra and provide screenshots showing that it has been unpacked and that its IAT has been fixed. How many imports does the unpacked program have? (12 pts)**
Imports: 9
   - Functions: 189




Windows7sp20 (HW6_3 GetProc) [Running] - Oracle VM VirtualBox
CodeBrowser: hw6_3:/hw6_3_dump_Fixed.exe
String Search [CodeBrowser: hw6_3:/hw6_3_dump_Fixed.exe]
String Search - 553 items - [hw6_3_dump_Fixed.exe, Minimum size = 5, Align = 1]
Program Trees
Symbol Tree
Data Type Manager
Listing: hw6_3_dump_Fixed.exe
Decompiler
Defined Data
Auto Label
Include Alignment Nulls
Truncate If Needed
Offset:
Preview:
Make String
Make Char Array
8:52 PM
4/25/2020