CMPSC 443 Lab 10: Unpacking

# Lab Description

Note: you are supposed to use a Windows VM to finish all these labs

The ollydump plugin and PEid need to be installed to accomplish this lab.

Your goal for the labs in this chapter is simply to unpack the code for further analysis. For each lab, you should try to unpack the code so that other static analysis techniques can be used. While you may be able to find an automated unpacker that will work with some of these labs, automated unpackers won’t help you learn the skills you need when you encounter custom packers. Also, once you master unpacking, you may be able to manually unpack a file in less time than it takes to find, download, and use an automated unpacker.

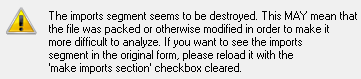
## Task 1

Lab18-01.exe

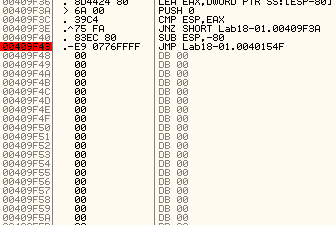
Loading this executable into PEiD, it reports that nothing is found which means it can’t detect what packing algorithm was used.

P:\CMPSC443\Git\CMPSC443\Lab10\NothingFound.PNG

Loading the executable into IDA Pro, it reports that “The imports segment seems to be destroyed. This MAY mean that the file was packed or otherwise modified in order to make it more difficult to analyze.” The assembly instructions that it is showing us are the unpacking assembly instructions.



Loading the executable into OllyDbg, it reports that “its code section is either compressed, encrypted, or contains large amount of embedded data”. Basically acknowledging that it is packed. Continuing with the analysis, we see that OllyDbg starts the program (its entry point) at 0x00409DC0. This is odd because (as said in class) the usual entry point is something along the lines of 0x00401xxx. I noticed that a few instructions down is filled with “DB 00”, so this program probably jumps before it reaches that.



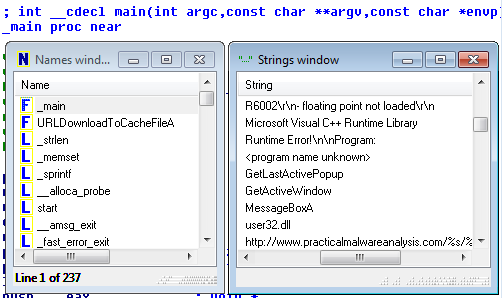
Sure enough, the instruction before “DB 00” it jumps to 0x0040154F. The part before the jump must be the unpacking algorithm. Because the first instruction at 0x0040154F is “push ebp”, that’s a tell that this is most likely the start of the real program.

P:\CMPSC443\Git\CMPSC443\Lab10\pushebp.PNG

From here we can use the OllyDbg plugin, “OllyDump 🡪 Dump debugged process”.

We can now take this dumped process and analyze it with PEiD. PEiD now recognizes that this sample is compiled with “Microsoft Visual Studio C++ 6.0”.

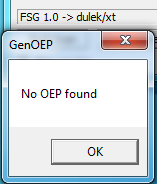
We can also put this dumped process into IDA Pro. Although it still gives us the warning that “its code section is either compressed, encrypted, or contains large amount of embedded data”, this time we are presented with the “main()” function. Combine this with the plethora of function names and strings IDA Pro now finds, we can be sure that it is now unpacked.



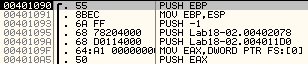
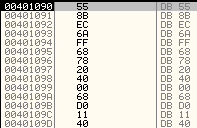
## Task 2:

Lab18-02.exe

Loading this file into PEiD, it tells us that this was packed with “FSG 1.0 -> dulek/xt”. Attempting to find a “generic OEP”, PEiD alerts us that “No OEP found”.



Loading the file now into OllyDbg, it tells us that the process “has entry point outside the code (as specified in the PE header).” OllyDbg loads the file at address 0x004050000. Using the OllyDbg plugin OllyDump 🡪 Find OEP by Section Hop (Trace over), it ends at 0x00401090. However, at this location the instructions are unreadable: “DB 55, DB 8B, …”. However, forcing OllyDbg to disassemble these instructions (analysis 🡪 analyze code), we see normal x86 instructions. Being that the first instruction is once again “push ebp”, it’s not a bad idea to dump the process at this point.



Loading the dumped process into PEiD results in recognition of being compiled with “Microsoft Visual C++ 6.0”.

Although IDA Pro once again gives us a warning, we now see the main function which verifies that the process is now unpacked.

## Task 3:

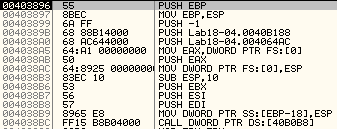
Lab18-03.exe [in-class demo]: you need to repeat it in the lab, but do not have to include it in your report.

Lab18-04.exe: you should unpack this sample and write the steps in your report.

As normal, we’ll start by loading this file into PEiD, which reports back that it is packed with “ASPack 2.12 -> Alexey Solodovnikov”. Generic OEP Finder alerts us that no OEP is found.

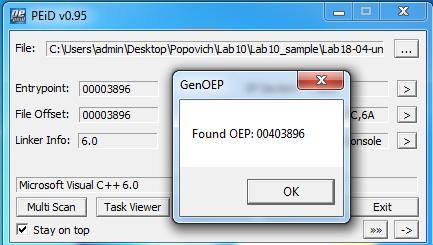
Loading the file into OllyDbg once again alerts us that the process “has entry point outside the code (as specified in the PE header).”

OllyDbg starts the process at 0x00411001. The first instruction at this point is “pushad”. “pushad” pushes all of the registers to the stack. It’s likely that the packing program will eventually restore all the registers once it is done with the unpacking algorithm. After the push is complete, 0x0012FF6C is left on the esp, or stack pointer. We can set a breakpoint whenever the memory accesses this address. Once that address is reached, we will notice that it “retn 0C” two lines below. Following this return, we arrive at 0x00403896. Although it is initially unreadable, we can force OllyDbg to disassemble the instructions as we have before. After this, the instructions will look how they do below:



Being that the first instruction is “push ebp”, that’s a great sign. Time to dump the process and load it into PEiD to see if it can identify the compiler.

PEiD successfully identifies it in the image below, with the OEP being exactly what we expected.



# Submission

|  |
| --- |
| * For each sample, please describe the unpacking process with a few sentences and screenshots in the lab report. * Please only submit the lab report. |