Lab 10 (11 in Wi 2020)

Decompiling code

Note this lab was intended to go with a bit of an in class discussion. Decompiling code lets you reconstruct the original source file. But there are some layers of complexity here.

1. Regular debug mode code is actually quote bloated, there’s a lot of stuff there which exists for debugging.
2. Release mode code can be reasonably trimmed down without extra code, but one of the problems is that the original C/C++ code can be reconstructed pretty easily. In fact the Java example below discusses an example of that happening.
3. There are tools for code obfuscation, the idea is that they, at once, optimise your code, but also make it essentially unreadable for humans

~~Build a C/C++ program in release mode. Then upload the executable here:~~

[~~https://retdec.com/decompilation/~~](https://retdec.com/decompilation/) ~~as an input file (you don’t need anything else). Sri Edit: Crap they broke this.~~

~~(Any simple C/C++ program will work, there’s one in a table below)~~

~~Now it requires a fairly extensive installation based on:~~

[~~https://github.com/avast-tl/retdec~~](https://github.com/avast-tl/retdec)

Download: <https://derevenets.com/index.html> and extract it somewhere. Run the snowman.exe

The lab is otherwise the same using this tool, you can load up any .exe (use simple ones you built yourself from a C program in a previous lab, there’s one attached below). I don’t think this works on a mac, make friends with a windows user and get them to send you the screenshots it’s fine.

Possible alternative (I was thinking of trying this for wi2020, it’s here if you want to look but you don’t need to, it’s the NSA decompiling tool): <https://www.ghidra-sre.org/>

Look at the assembly output and compare to the assembly output from your program? That you hopefully know how to do from a previous lab. If not I’ve attached a CPP and an ASM file for you to look at that are related to lab 8 in a table at the end of this document

Can you even find where your program actually is in the decompiled version. My executable is 9Kb, and the output is about 5000 thousand lines long. If you do this in debug mode you’ll get an output that’s about 50 000 lines.   
  
  
This is how you actually reverse engineer a program (such as a virus etc. Though to do that **well** you’d pay for tools that make some of it less terrible than it is with raw code, but those tools see what you just saw).   
  
  
Lastly, have a look at <http://jd.benow.ca/> go here and download the jd-gui-1.6.6.jar, you’ll get a security warning, but it’s fine, run it anyway. Run the program and open the MARS jar file you’ve been using for other assignments, and show me a screenshot of the output. (Yes, that reconstructed the entire source structure from the JAR file, including comments). The JAR file for MARS is deliberately not obfuscated or anything, since it’s an educational tool, not a commercial product. Doing the same on a commercial product is a bit tricky. If you’ve got any JAR files from another course (COIS 2240H or some other course) compare the original and decompiled source.

**SUBMISSION FOR THIS LAB: Show me the decompiled output of a C/C++ executable program (e.g. the one below) and the output from the jd-gui-1.6.6 being run on mars.jar. If you have any other java program (e.g. from 2240H) Run it on that too. Put those screenshots in a word docx, upload to BB.**

|  |
| --- |
| // CpptoASM.cpp : Defines the entry point for the console application.  //  #pragma float\_control(precise)  //#pragma float\_control(fast)  // Code that uses /fp:precise mode  #include "stdafx.h"  #include <iostream>  using namespace std;  int main()  {  cout.precision(10);  float inc = 0.000001, sum = 0.0;  for (float i = 1.0; i <= 1000000.0;)  {  sum += (1.0 / i);  i += 1.0;  }    cout<< sum;  cin.get();  return 0;  } |

|  |
| --- |
| ; Listing generated by Microsoft (R) Optimizing Compiler Version 19.00.23506.0  TITLE **C:**\Users\sir\_s\Desktop\CpptoASM\CpptoASM\CpptoASM.cpp  .686P  .XMM  include listing.inc  .model **flat**  INCLUDELIB MSVCRTD  INCLUDELIB OLDNAMES  PUBLIC ?\_\_empty\_global\_delete@@YAXPAX@Z ; \_\_empty\_global\_delete  PUBLIC ?\_\_empty\_global\_delete@@YAXPAXI@Z ; \_\_empty\_global\_delete  PUBLIC \_main  PUBLIC \_\_real@3f800000  PUBLIC \_\_real@3ff0000000000000  PUBLIC \_\_real@412e848000000000  EXTRN \_\_imp\_?precision@ios\_base@std@@QAE\_J\_J@Z**:**PROC  EXTRN \_\_imp\_??6?$basic\_ostream@DU?$char\_traits@D@std@@@std@@QAEAAV01@M@Z**:**PROC  EXTRN \_\_imp\_?get@?$basic\_istream@DU?$char\_traits@D@std@@@std@@QAEHXZ**:**PROC  EXTRN \_\_imp\_?cin@std@@3V?$basic\_istream@DU?$char\_traits@D@std@@@1@A**:BYTE**  EXTRN \_\_imp\_?cout@std@@3V?$basic\_ostream@DU?$char\_traits@D@std@@@1@A**:BYTE**  EXTRN \_\_fltused**:DWORD**  ; COMDAT \_\_real@412e848000000000  CONST SEGMENT  \_\_real@412e848000000000 DQ 0412e848000000000r ; 1e+06  CONST ENDS  ; COMDAT \_\_real@3ff0000000000000  CONST SEGMENT  \_\_real@3ff0000000000000 DQ 03ff0000000000000r ; 1  CONST ENDS  ; COMDAT \_\_real@3f800000  CONST SEGMENT  \_\_real@3f800000 DD 03f800000r ; 1  CONST ENDS  ; Function compile flags: /Ogtp  ; File c:\users\sir\_s\desktop\cpptoasm\cpptoasm\cpptoasm.cpp  ; COMDAT \_main  \_TEXT SEGMENT  \_main PROC ; COMDAT  ; 15 : cout.precision(10);  **mov** **edx,** **DWORD** PTR \_\_imp\_?cout@std@@3V?$basic\_ostream@DU?$char\_traits@D@std@@@1@A  **push** 0  **push** 10 ; 0000000aH  **mov** **eax,** **DWORD** PTR **[edx]**  **mov** **ecx,** **DWORD** PTR **[eax+**4**]**  **add** **ecx,** **edx**  **call** **DWORD** PTR \_\_imp\_?precision@ios\_base@std@@QAE\_J\_J@Z  ; 16 : float inc = 0.000001, sum = 0.0;  ; 17 : for (float i = 1.0; i <= 1000000.0;)  **movss** **xmm2,** **DWORD** PTR \_\_real@3f800000  **xorps** **xmm3,** **xmm3**  **movsd** **xmm4,** **QWORD** PTR \_\_real@3ff0000000000000  **movsd** **xmm5,** **QWORD** PTR \_\_real@412e848000000000  $LL2@main**:**  ; 18 : {  ; 19 : sum += (1.0 / i);  **cvtps2pd** **xmm0,** **xmm2**  **movaps** **xmm1,** **xmm4**  **divsd** **xmm1,** **xmm0**  **cvtps2pd** **xmm0,** **xmm3**  **xorps** **xmm3,** **xmm3**  **addsd** **xmm1,** **xmm0**  ; 20 : i += 1.0;  **cvtps2pd** **xmm0,** **xmm2**  **cvtsd2ss** **xmm3,** **xmm1**  **addsd** **xmm0,** **xmm4**  **cvtpd2ps** **xmm2,** **xmm0**  **cvtps2pd** **xmm0,** **xmm2**  **comisd** **xmm5,** **xmm0**  **jae** SHORT $LL2@main  ; 21 : }  ; 22 :  ; 23 : cout<< sum;  **push** **ecx**  **mov** **ecx,** **DWORD** PTR \_\_imp\_?cout@std@@3V?$basic\_ostream@DU?$char\_traits@D@std@@@1@A  **movss** **DWORD** PTR **[esp],** **xmm3**  **call** **DWORD** PTR \_\_imp\_??6?$basic\_ostream@DU?$char\_traits@D@std@@@std@@QAEAAV01@M@Z  ; 24 : cin.get();  **mov** **ecx,** **DWORD** PTR \_\_imp\_?cin@std@@3V?$basic\_istream@DU?$char\_traits@D@std@@@1@A  **call** **DWORD** PTR \_\_imp\_?get@?$basic\_istream@DU?$char\_traits@D@std@@@std@@QAEHXZ  ; 25 : return 0;  **xor** **eax,** **eax**  ; 26 : }  **ret** 0  \_main ENDP  \_TEXT ENDS  ; Function compile flags: /Ogtp  ; File c:\users\sir\_s\desktop\cpptoasm\cpptoasm\cpptoasm.cpp  ; COMDAT ?\_\_empty\_global\_delete@@YAXPAXI@Z  \_TEXT SEGMENT  \_\_\_formal**$** **=** 8 ; size = 4  \_\_\_formal**$** **=** 12 ; size = 4  ?\_\_empty\_global\_delete@@YAXPAXI@Z PROC ; \_\_empty\_global\_delete, COMDAT  **ret** 0  ?\_\_empty\_global\_delete@@YAXPAXI@Z ENDP ; \_\_empty\_global\_delete  \_TEXT ENDS  ; Function compile flags: /Ogtp  ; File c:\users\sir\_s\desktop\cpptoasm\cpptoasm\cpptoasm.cpp  ; COMDAT ?\_\_empty\_global\_delete@@YAXPAX@Z  \_TEXT SEGMENT  \_\_\_formal**$** **=** 8 ; size = 4  ?\_\_empty\_global\_delete@@YAXPAX@Z PROC ; \_\_empty\_global\_delete, COMDAT  **ret** 0  ?\_\_empty\_global\_delete@@YAXPAX@Z ENDP ; \_\_empty\_global\_delete  \_TEXT ENDS  END |