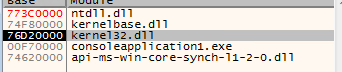
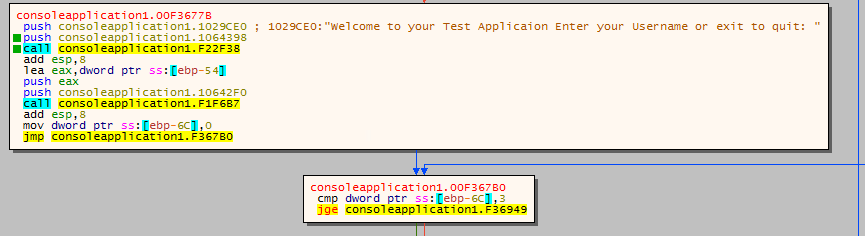
Issues

* Heap
* Naming function calls
* Return values/storage function calls
* Restarting exe/address changes
* Python script with pipes to call your program

Main Questions:

* How can we easily view the Heap?
  + You can search in the symbol table for “malloc” and then set a breakpoint on it, but there are tons of calls to malloc when printing out things and using other standard library functions, so that doesn’t help that much.
* How can we view Global/Static variables?
  + Looks like most number variables are in .data. just translate from hex to decimal.
  + Looks like read-only data, such as literal strings, constants, and debug directory information are in .rdata
* How can we view all custom made function calls?
  + The .pdb file would tell you all of the useful debugging info, but we don’t have that, we just have the executable. This is the Microsoft file type for debugging windows apps.
  + Function calls with lower addresses are user defined ones though:



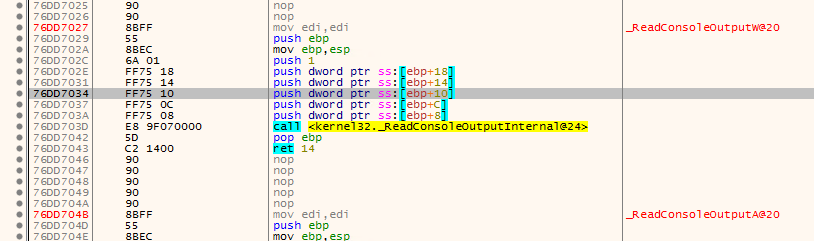
See how the .dll’s are 7 addresses where as the custom.cpp application is at address x00F22F38… 

Well if we look at the memory map, these functions are in the .text section – which contains machine code of all functions we would have written in C sourcefile.



So did Professor Randall just write his own GetConsoleInput functions to screw with me? No.

* + For external libraries used in the application: you can right click when in the external module and chose “download symbols for this module”. This will then fillout the names of all of the functions in the “comment” section of the disassembler.



* + When the application prompts for input, we are in a x777~ memory address which means its an external library function. It always says “kernel32” by EIP
* How can we mark common c library functions so we know what “call \*function:asdf\*” means in the assembly code? I assume there are some string compare calls or something going on here.
* Where do function calls store their results?
  + Points and Dump:
  + Where is my input stored?
  + Where is it being compared to the correct value?
* Is there a better way to “restart” the program, redragging the menu is annoying, and not sure if the addresses change.
* How can I find out how much space was allocated for the username and password? Feeding too many characters to username breaks the application

Goals:

* See if any overflows
* See if nasty strings break
* Find Usernames
* Find Passwords
* Create modified binary to automatically get access