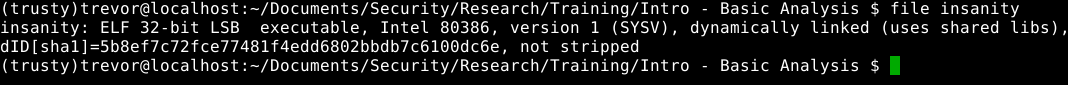
**Intro To Reverse Engineering - DoYouKnowStrings Solution**

**Introduction:** DoYouKnowStrings is an easy challenge designed to teach students how to perform a basic analysis against a target binary to learn more about it. Using the commands discussed in this write up should be one of your first go-tos when trying to learn more about a target executable. This problem binary was originally written by the 9447 CTF team for their 2014 CTF competition.

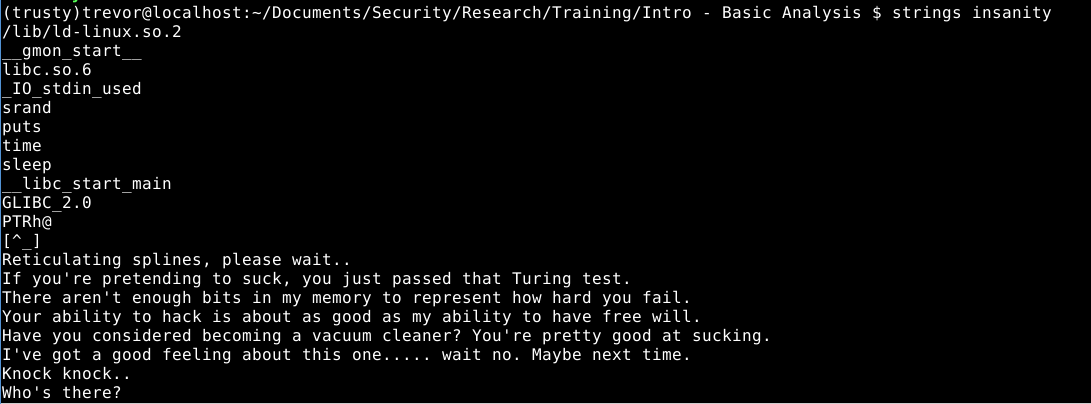
**Challenge:** Retrieve the flags from these binaries, using the commands ‘file’, ‘strings’, ‘strace’, and ‘ltrace’.

**Solving:** Once you’ve downloaded the binary open a terminal and navigate to where you downloaded the ‘insanity’ binary. Let’s begin our analysis against the file. To do this we’ll use two basic commands: ‘file’, and ‘strings’. The file command will provide us with basic information about the file. It’s syntax is: “file filename”.



There are two pieces of information that are important at the moment. First, this file is a 32-bit elf binary. This means that file is executable, and it was compiled on a linux system. Second, this binary is ‘not stripped’ meaning this binary was compiled with debugging info enabled “g++ source.cpp -g”. Compiling binaries with debugging info enabled allows you to read high level code inside of the debugger / disassembler. The alternate is a ‘stripped’ binary that wasn’t compiled with debugging information. This means we would only be able to read assembly / machine code in the debugger / disassembler.

Next let’s learn about the ‘strings’ command, its syntax is: “strings filename”. Strings will print out all printable characters in the files you pass to it. This command is especially useful for attempting to find data that may be hidden in a non-text file, such as this binary. Go ahead and run strings against the ‘insanity’ binary and let’s see what we get.



As you can see from the output, the strings command has given us, well, a whole bunch of strings. This output allows us to see possible functions this program uses during execution. Functions such as: srand(), puts(), time(), and sleep(). Try running the binary on your system to see if sleep() is actually used. The next pieces of data we can see are hard coded strings such as: “Reticulating splines, please wait..”, and “Knock Knock..”. Finally, you can see system level functions used in the program. A good example is “\_\_libc\_start\_main” this function is responsible for calling main() in every program we use.

Sometimes though, flags, and other important data we can use as clues are simply revealed by strings. If we scroll down a ways we’ll find:

Screenshot - 072616 - 00:39:52.png

That’s our first flag in the bag! The strings command can often be combined with another command called [grep](http://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/) if you know beforehand what kind of information you’re looking for. Below is an example.

Screenshot - 072616 - 00:42:28.png