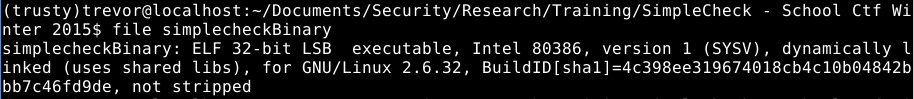
**Simple Check - School Ctf Winter 2015**

**Introduction:** Find The Owner was an easy 100 point forensics challenge that was part of the 2014 Pwnium CTF organized by the SiBears ctf team. This writeup was made possible using resources from the [CTF Wiki](https://github.com/ctfs/write-ups-2015/tree/master/school-ctf-winter-2015/reverse/simple-check-100). During this write up we will improve our reverse engineering skills and learn to bypass simple checks found in binaries. Bypassing simple checks is useful because will allow us to control the execution flow of a program without actually meeting those requirements. This writeup assumes you have basic knowledge with the debugger and disassembler of your choice.

**Task:** Download the provided binary and enter the right key to retrieve the flag.

**Solving:** Once you’ve downloaded the binary go ahead and perform a basic 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”. [Here’s](http://www.computerhope.com/unix/ufile.htm) a link to an article with more information on the command.



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 run this binary and see what this program does. But first we’ll need to chmod the binary so you can execute it on your system ‘chmod 755 binaryName’. Upon execution the program asks for us to input a key. I tried ‘12345’ the program then output “Wrong” and quit. Trying several other inputs resulted in the same behavior. It becomes pretty clear we need to figure out the magic key, or bypass the check that compares our key with the correct key.

Screenshot - 072516 - 23:42:23.png