EL-ESS

We have already seen some basic tools and techniques used in binary exploitation. Now moving on to the problem named El-ess.

By simple execution we can find that the program takes an Input and runs a ‘ls’ command on the system with it as the argument next thing we noticed is that we get no other messages even if the input is wrong. By giving on the server like “.” we can find out the Containment of the directory where the program is being executed from. Also, we can find that there is a ‘flag.txt’ file along with the program. This is our target. Along with it we also have a program named ‘showflag’

now let's disassemble the program and read the Assembly.

* The scanf that takes your input has string formatting as “%16s”
* Also, we find out that the string “/bin/ls” has been loaded to the stack. Evidently it would be used for making a system call.
* later in the program input sanitization is taking place to make our input valid for /bin/ls character by character. [ this can be observed through the through giving an invalid input while in GDB]
* the program pushes “/ls” above the input string and the “/bin” making it “/bin/ls <input>”.
* The final string formed at the time of system call would be “/bin/ls <input> 2> /dev/null”.
* 2> means Directing areas only to
* /dev/null Is a directory called black hole. it means anything directed to it would be nothing and would be lost forever. <[read about it](https://en.wikipedia.org/wiki/Null_device)>
* An interesting thing about the program, on moving an input longer than 15 characters it gives no output even for a valid input
* On inspecting we find that the string built was “<16 characters from input> 2 /dev/null”
* As is obvious 16 only because 16 hour the maximum number of characters that can be read by the scanner

Therefore using this vulnerability we can make any system call that would be 16 characters long. All we need to do now is to execute the showflag program. Showflag is 8 bytes long thus we need 8 bytes more. We use ‘./’ to fill rest of our 8 bytes. './', however many times written, will return us to the same directory.

Now our input would be “././././showflag”. This would execute the show flag program

and give us required flag...