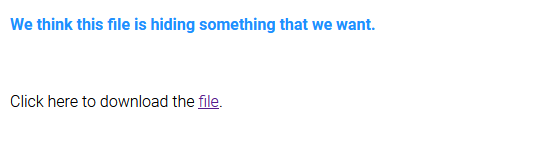
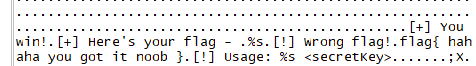
**Challenge 6.**



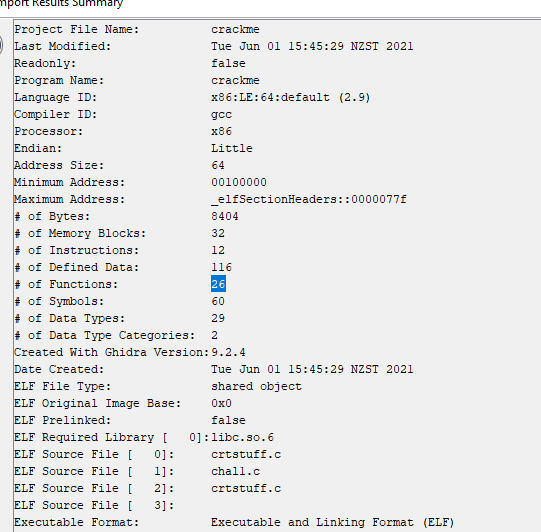
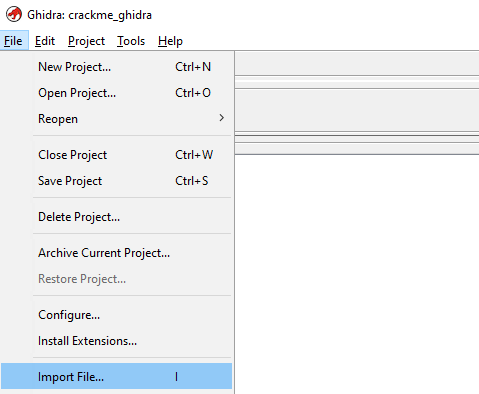
1. I downloaded the file and had a look with a hex editor. Analysis of the file returned a funny answer:



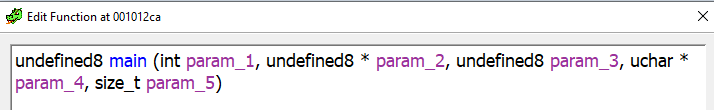
1. It did however look like a file which was meant to be executed. And potentially the <secretkey> was something that could be retrieved.
2. I went through a whole number of steps to try to run this file. Compiling via gcc, running as an executable, running on a 16bit VM. Long story short:



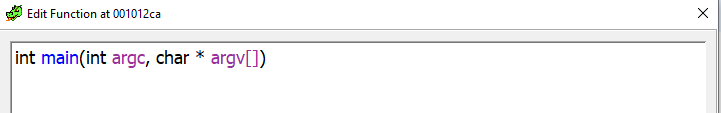
1. At this point seeing a binary that needs a password I think it’s time to bring out ghidra. I created a new Ghidra project and imported the binary. Using the default recommendations I got an output seen below. There were a few errors (missing libc) but we don’t worry about those.



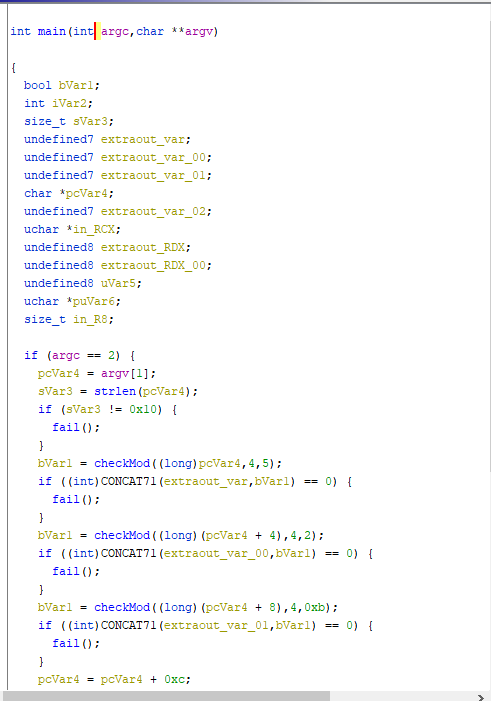
1. At this point we use codebrowserto analyze the binary file. We choose to use the Ghidra analyzers and leave them all on default except “decompiler parameter id” which we turn on, as it often improves our decompilation parameter results.
2. The next step is to search our symbol tree for our “main” function and then edit the function signature to clean it up.



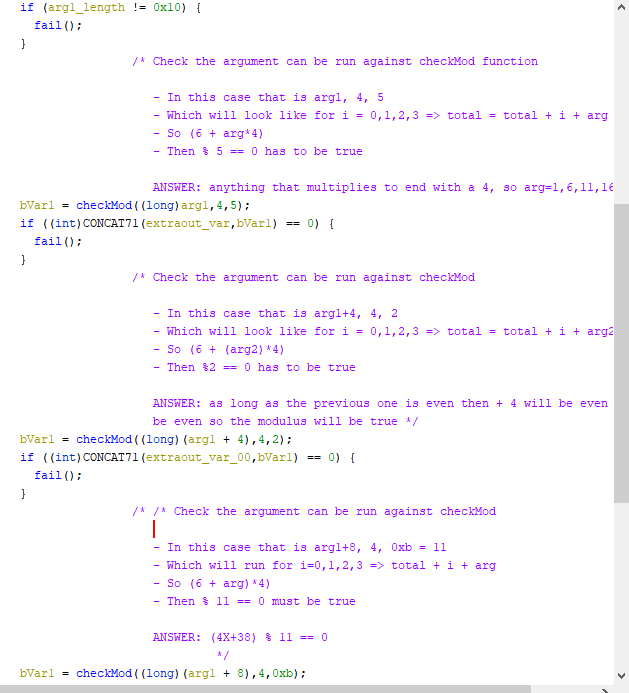
**TO:**



1. This cleaned up our function results quite a bit:



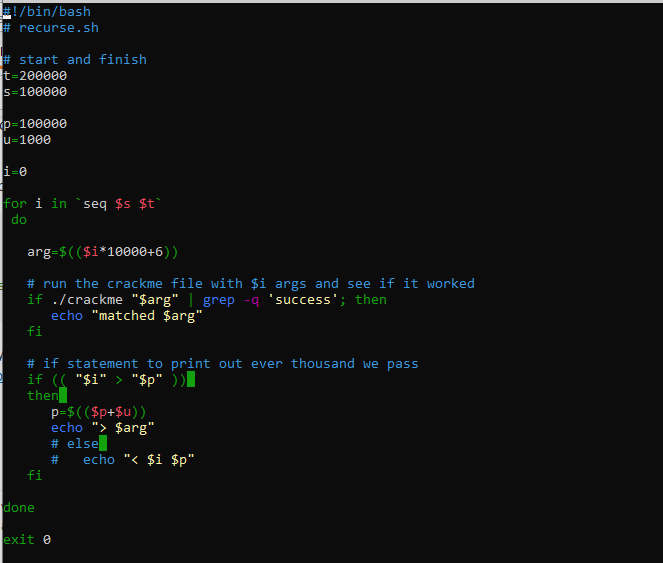
Then I spent some time working out what algorithms they were using. It looked like the answer required a very specific number to be input.



1. It got a bit too complicated for me to work out mathematically so I went looking for the flag another way. A method entitled “success” seemed a good place to look:



1. At this point I figured it was worth trying a bash script to see if I could just do every possible combination. Using my knowledge I refined the scope to only even numbers ending with 6.



1. While waiting for the script, I decided to see if I could output a modified binary via Ghidra. I first attempted to modify the exit(1) instruction. This produced a runtime error. I had a look at modifying the checkMod() function with no success. Finally I modified the fail() instruction to instead be a success()



This had the desired output when I ran the program:

