RE #45

I skipped ahead to this problem because it seemed fun. I already had been doing some AVR RE problems I came up with and wanted a harder challenge.

There are two tasks:

1. (Easy) with the help of any debugger, force the program to accept a changed key file.
2. (Medium) your goal is to modify the username to another, without patching the program.

Procedure:

1. For this part of the exercise, I cheated a bit by using IDA PRO to analyze the code. I noticed that only the username and serial number were really being used by the code. Bytes after the terminating zero till offset 0x7F were being ignored. Using a hex editor, I confirmed that bytes terminating zero till offset 0x7F could be edited with no effect. Additionally, I noticed that the serial number could also be changed. I was confused if this was what mean by forcing the debugger to accept a changed file. Dennis Yurichev confirmed that this was the right solution.
2. Loading the exe into IDA PRO we can see the flow where the file is loaded, and the username is checked. What stood out to me what this portion of the code:A computer screen shot of a computer code

   Description automatically generated

Here we see that the output of a function is compared with 0x0E425 and if it fails, we get the keyfile is incorrect message. The call to sub\_4015f0 leads us to a very long function: A screenshot of a computer

Description automatically generated

At first it stumped me because it seemed very odd. I had no idea what algorithim this was and why it was XORING values repeatedly. In this portion, we can see 0x8408 is called multiple times, and in the full code is goes futher. Googleing this value brings up results referencing the CRC-16 algorithim. I’ve never seen this algorithim before, bu after looking at a few implementations it seems like it may be similar. Using this CRC16 calculator (<https://crccalc.com/>) and putting in the username "Dennis Yurichev" if we scroll down to CRC-16/x-25 we see that we get a value of 0x25E4, or 0xE425 in little endian. Looking online it seems like crc-16 is sometimes used as a hashfunction. My idea is then can we get another username that can prodice the same output when put into crc-16, causing a hash collison.

I found a crc-16 implementation for python, and confimed it gave the same value when putting in "Dennis Yurichev". I then gave it a new username “Chris Vega “ whith extra padding to make it the same length. I then wrote a pythonscript to increment the ASCII vlaues for the padded protions, run it through crc-16, and compare it with 0x25E4 until we got a match. After a few second the code retuned the username “Chris Vega (8S” as a match. When the program was run with the modified key file it resulted in the username passing!  
A blue screen with white text

Description automatically generated