https://www.crackmes.one/crackme/67fd376f8f555589f3530b9d

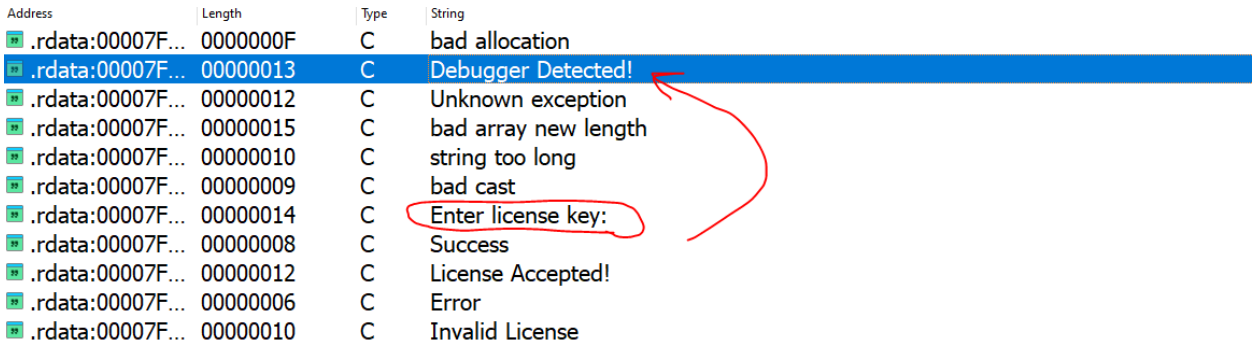
Ionchad's simple crackme

For this one I will be using IDA. And we'll be using the decompiler.

First thing to do is to run the program:

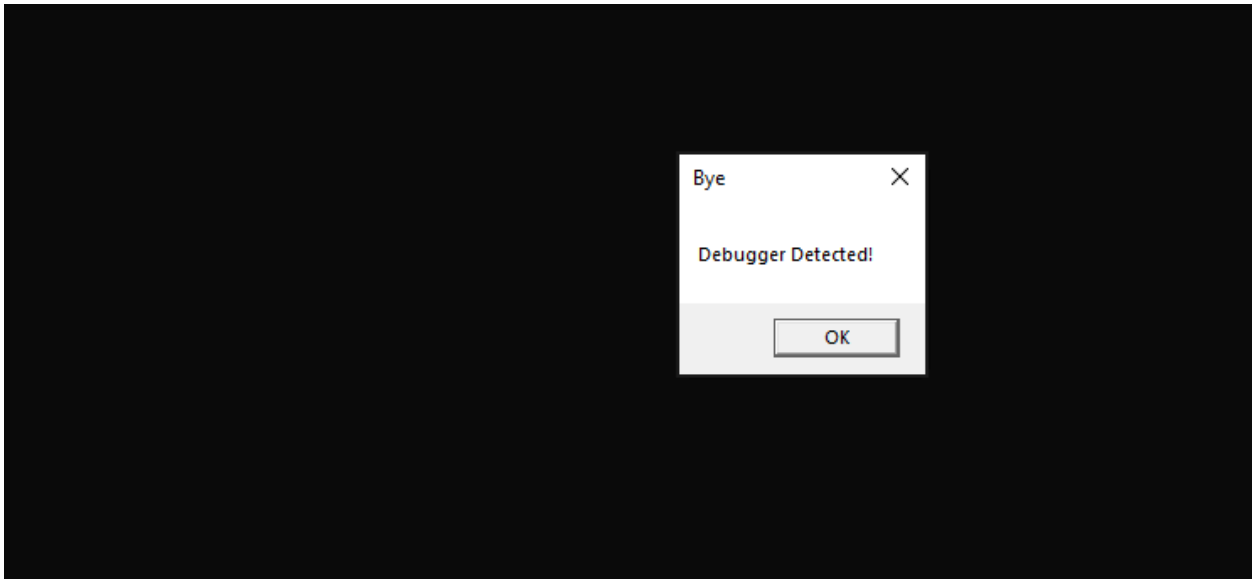
Now we know a string to look for.

Lets boot up IDA and look for strings with the stringview subview.

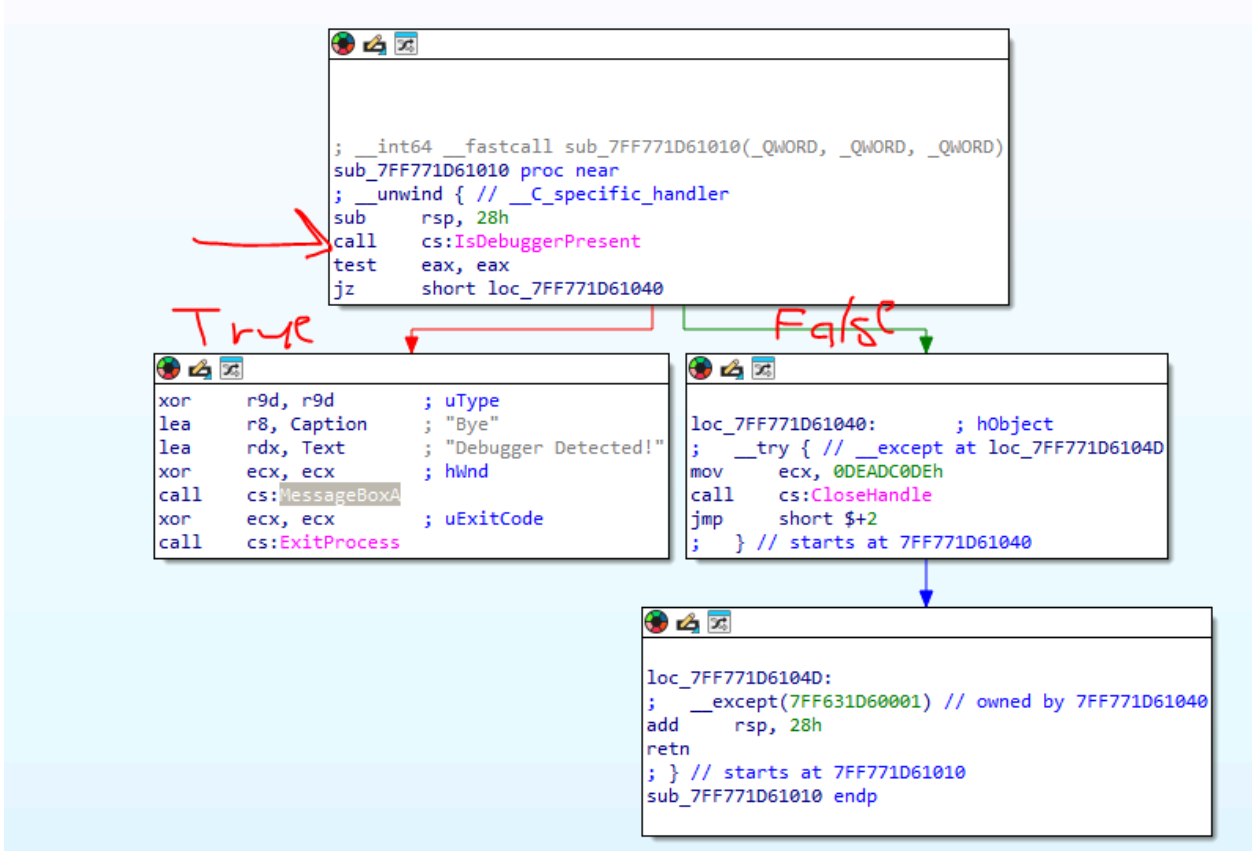


We found what we were looking for, but we also found a string,"Debugger Detected!"
We can infer that this program has anti-debugger measures. Fortunately, we are only decompiling here. By decompiling this program we get a long bit of pseudocode. We also see that there is no obvious string that could be the password here. But if we do run it in a debugger,

we get this:



We can also see the success and license accepted strings, but, I'm going to start there at the debugger detector.



When we go to the main function, we can see that this is the first thing that is called, obviously. Lets open up the decompiler.

Opening it up in the decompiled view:

```
    j_j_free(v22);
  }
  *(_QWORD *)(v13 + 16) = 0LL;
  *(_QWORD *)(v13 + 24) = 15LL;
  *(_BYTE *)v13 = 0;
  if ( v19 )
  {
    v23 = "Success";
    v24 = "License Accepted!";
  }
  else
  {
    v23 = "Error";
    v24 = "Invalid License";
  }
  MessageBoxA(0LL, v24, v23, 0);
```

We see our success and error logic. If (v19).

Lets follow v19 up the pseudocode.

```
sub_140001220(Block, Buf2);
v13 = sub_140001350(v31, Block);
v28 = 0LL;
v29 = 4LL;
v30 = 15LL;
strcpy((char *)&v28, "KIWZ");
sub_140001220(Buf2, &v28);
v14 = Buf2;
v15 = (char *)Buf2[0];
v16 = v34;
if ( v34 > 0xF )
  v14 = (void **)Buf2[0];
v17 = (_QWORD *)v13;
if ( *(_QWORD *)(v13 + 24) > 0xFuLL )
  v17 = *(_QWORD **)v13;
v18 = *(_QWORD *)(v13 + 16);
if ( v18 == v33 )
{
  if ( v18 )
    v19 = memcmp(v17, v14, v18) == 0;
  else
    v19 = 1;
}
else
{
  v19 = 0;
}
```

comparison

Right off the bat, we see a weird string “KIWZ,” this could very well be the licence key were looking for(It is). But let's dig deeper.
We follow v19 up to a comparison where it is set equal to 0 or 1 or 0 again. With the memcmp call, it is comparing two memory blocks, v17 and v14, v18 being the amount of bytes to compare. v19 stores the boolean of the operation, so 0 if the input data and license key are equal, and 1 if they're not. Above that comparison, we see that v14 is set equal to Buf2, which Buf2 is set to v28 in a buffer function. And v28 is equal to “KIWZ”.

And when we try KIWZ. It works!