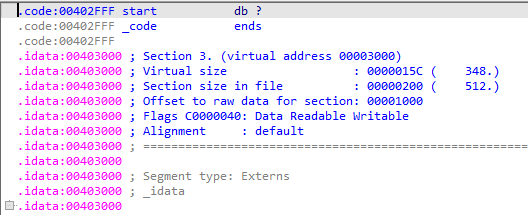
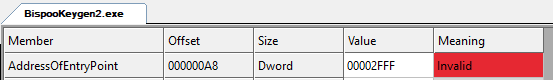
As soon as i load the executable into x32dbg and click on run, i get this error:\



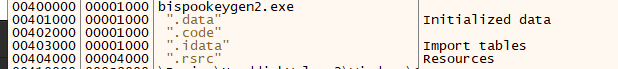
I decided to open the file in IDA and this is where the entry point is:



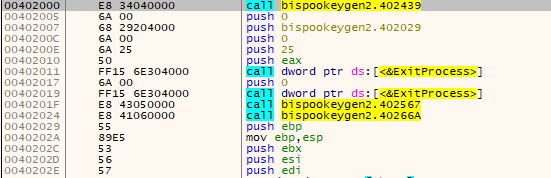
That’s like someone has messed up the entry point. CFF eplorer confirms that:



Back in x32dbg, i take a look in the memory map and notice that the code section actually starts at 402000:

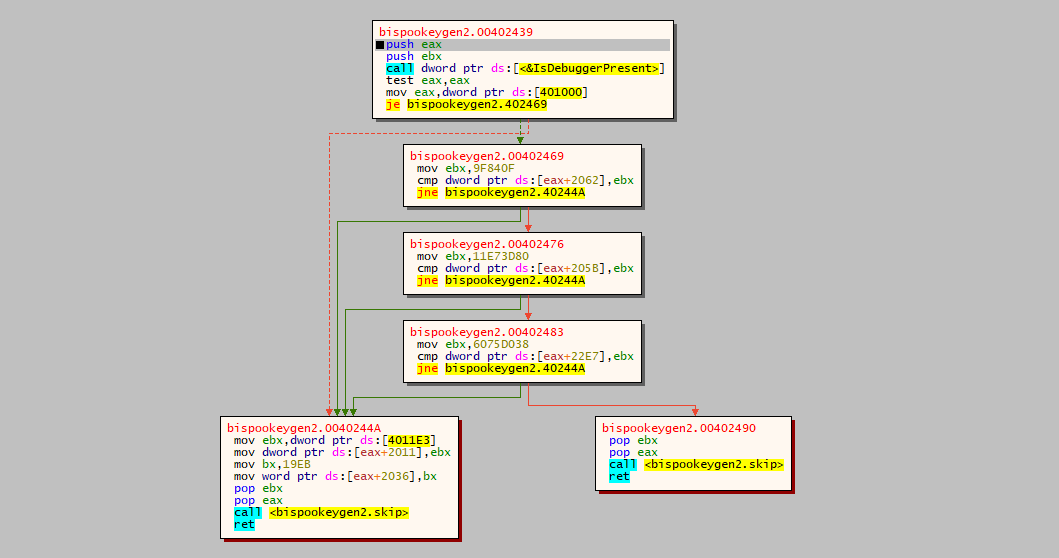


And there is the code:



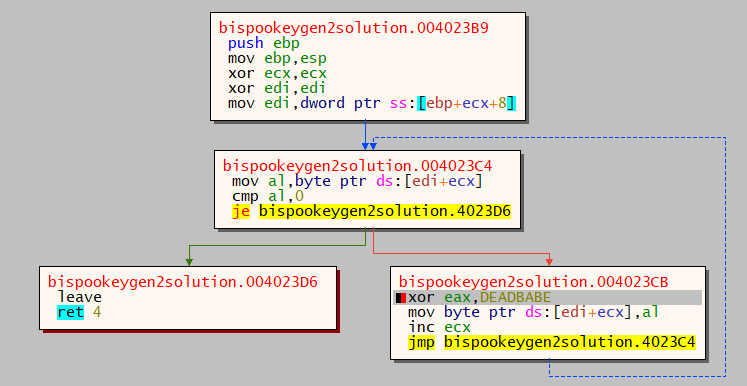
So i modified eip value to 402000. Before we go any further, lets see what the function at 402439 does. Eip keeps getting messed up. But i noticed in IDA the there is a function named TlsCallback\_0 at address 402510. And by placing a breakpoint there, we can confirm that some stuff actually happen here. The tls callback doesn’t stop me from debbuging so...

After the tls callback eip jumps to 402000, and there is a call to the function at 402439. lets see now what happens there. There is a call to IsDebuggerPresent:



And there is a test to make sure that eax is 0, so i will change the call instruction to xor eax,eax. We will just keep running it in steps, and we go into the function named skip. Inside this function there are two loops, we get to the second one only if we succeed at the first one. In order for it to not throw me out, I’m gonna change the jbe for ja. Now that we have passed the anti debug methods, the real fun begins.

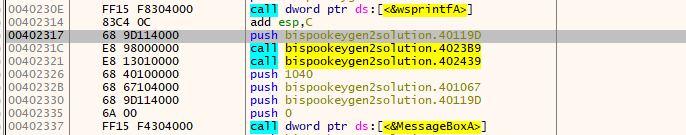
Inside the function at 402029 i notice that there is a call to another function (4023b9) many times in different locations, so i decided to investigate.



This function is doing some hash that works both ways, this is a simulation of it in python:

s = list("Nope... that password is wrong")  
  
i = 0  
while True:  
 if i == len(s):  
 break  
 ch = ord(s[i])  
 ch ^= 0xDEADBABE  
 s[i] = ch & 0xFF  
 i += 1  
  
s = [chr(ch) for ch in s]  
print(''.join(s))

Later I found the bad string in the memory at address 4011c4, so i decrypted it and it was what it should have. I saw another string in the memory above the bad string at address 40119d, and after decrypting it I saw that it was the good string. I searched for uses of the string and i found a usage at 402317:



After patching address 4022e7 from cmp al,dl to cmp,al,al, the program naw throws me away.

At the last test at 402439. so i will skip all these tests and inject jmp 402490 at address 40244a.

Now it doesn’t throw me away.

I run the program, insert ‘shoham’ as username and ‘banana’ as password and i get the popup saying that i solved it.

