**Attempt Reflection**

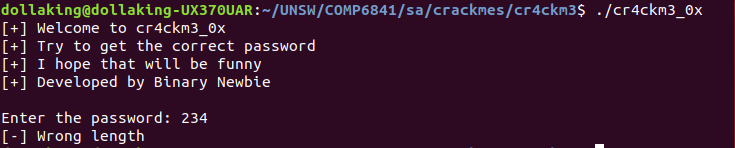
**Crackme:** <https://crackmes.one/crackme/5e51836e33c5d4439bb2dc24>

This problem is another obfuscation problem, but it is a lot harder than the last one even though they are both level 4 in crackmes.one. I can not use the same tricks as I used in Obfuscation1 because just cmp functions alone it took up max page capacity in the terminal. Also unlike Obfuscation, the cmp are comparing random numbers and registers. In Obfuscation, a majority of the cmp where comparing to 0xfff so it was easy to filter out.

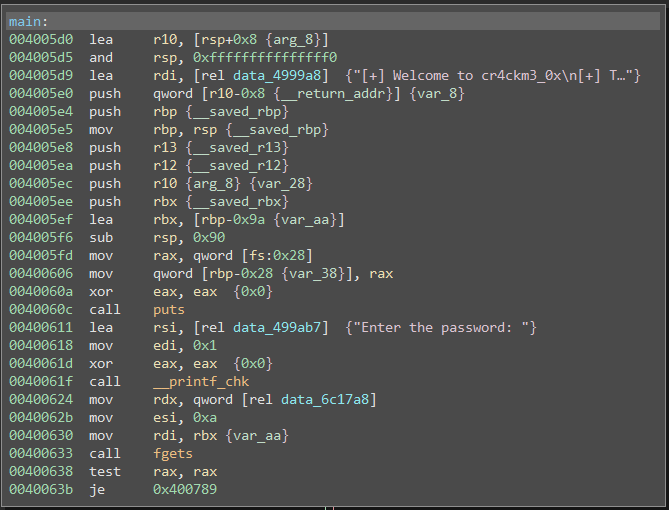
**What I tried:**

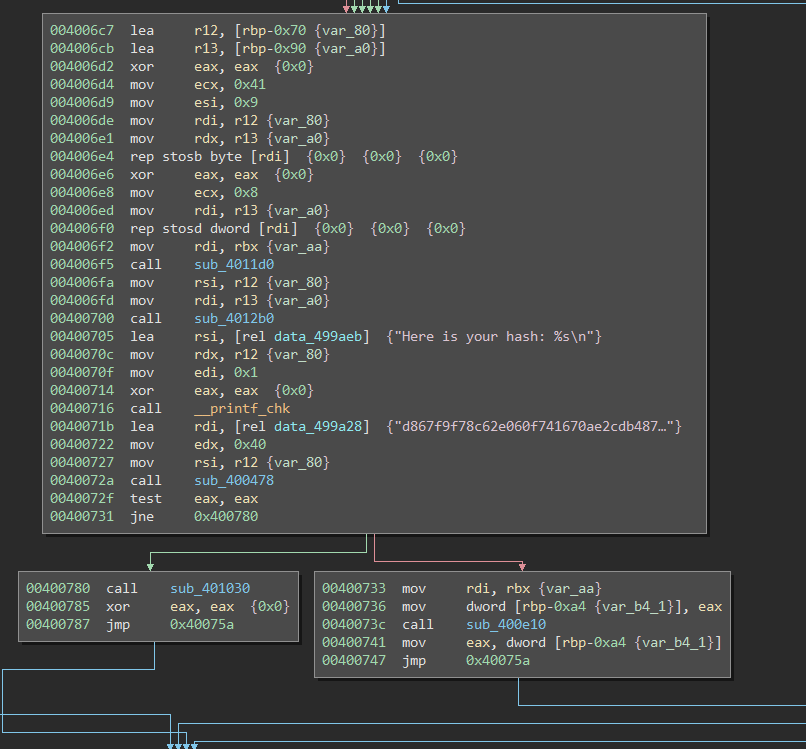
First I tried using strings -a which was very naïve of me because the list was bombarded with random strings to throw me random strings that are irrelevant.

The function requires the correct length in order to proceed to the next stage of the problem. For this I just brute forced the length and it turned out the length was 9. Once I found out the length was 9 I use grep -x ‘.\{9\}’ to filter strings with 9 characters. Unfortunately after trying them all they did not work.

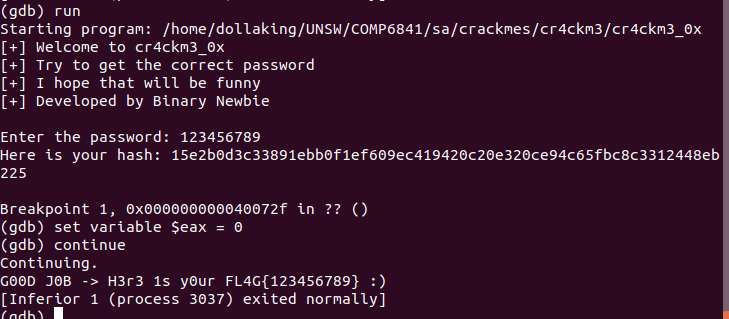


After going pass the length stage it produces a hash then tells me if the password was right or not. This is when I opened up Binary Ninja to see if I could understand the program a bit more.





I realise in there is a clear split off where it is most likely where it decides whether it tells me if I got the password right or wrong. So I made a break at 0x40072f in gdb and set the $eax register to 0 and 1 “set variable $eax = 0”. When I set it to zero it gave me the flag option but I need to get password right in order to get the complete flag.



I tried looking at the sub\_4012b0 call in attempt to understand the hashing component better. I saw a cmp at 0x4013da which is my chance to use to see if it is comparing characters for the password.



So I made a break there and checked the values of $r13 and $rbx. The variable in $rbx remains constant while the $rbx changes. If it is equal it will stop the loop. A block cipher is most likely used here as the has contains 64 letters and the loop ran 4 times meaning that it checked 16 bytes at a time. I have no clue where to look here.