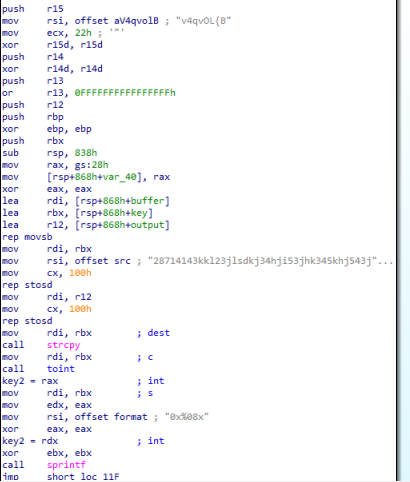
Disassemble the file, we get:



(buffer, key, output are named by me, just for convenient)

Setup:

-firstly, it write into buffer a string “v4q…….” length is given in ecx (22h)

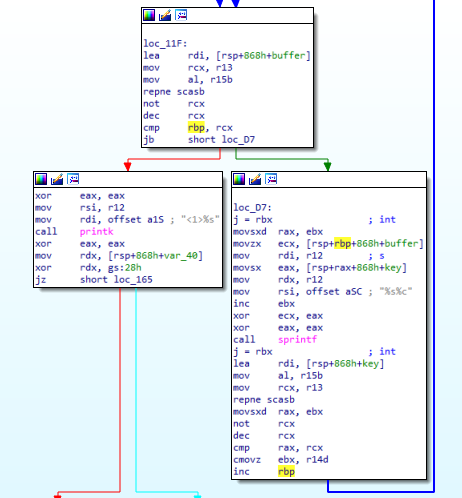
-second, it writes into the key a string “28714143….”

-then output is cleared

(notice that at very first, ebp and r14d are set to 0 by xor)

Then:

-call toint(key) then write back to key inform “0x%08x” (toint() not built in function from any library)



easy found that it is a loop with i = rbp, with loop condition is i < rcx = len(buffer) (repne, not then dec stub is common routine for length of a string)

The output(flag) is generated by buffer[i] ^ key[i%len(key)]

so I have the script (the toInt is not built in function so i code the same function with the code of toInt)

def toInt(key):

result = 1

tmp = 0

if key[0] == '-':

result = -1

if key[0] == '+' or key[0] == '-':

key = key[1:]

for i in key:

x = ord(i) - 48

if x > 9:

break

tmp = 10 \* tmp + x

return "0x" + "{0:08x}".format(result \* tmp)

key = "28714143kkl23jlsdkj34hji53jhk345khj543jhk354h354jh354jhkl354jhkl354hjk345hjk3h4i5h3l4h5iul34u6h4e5uh7ui5h7uilyhhiuyhuileyhlui6yhuilyuhil55hhuilhiw543uhiw34uhihuiuh6iwl354h"

key = toInt(key)

i = 0

j = 0

flag = ""

buffer = "v4qvOL{B\x0E^\x06N\x02rP\x01S\a\f4\x06K\a\x04\*a\nfsS\x03N\x04"

aBuffer = [] #array of buffer inform int

for i in buffer:

aBuffer.append(ord(i))

i = 0

while i < len(buffer):

x = key[j]

j = j + 1

flag += chr( ord(x) ^ aBuffer[i])

if j == 10:

j = 0

i = i + 1

print(flag)

FLAG-zIv78662C27a35R6375HW8RJ5364