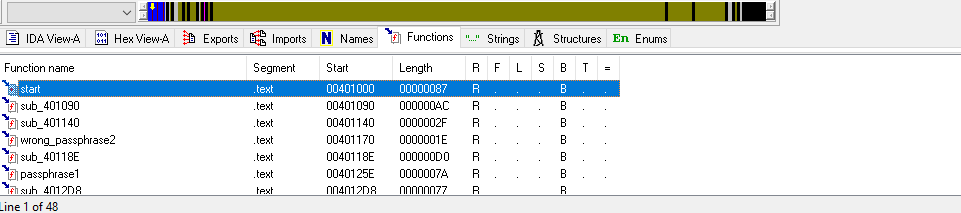
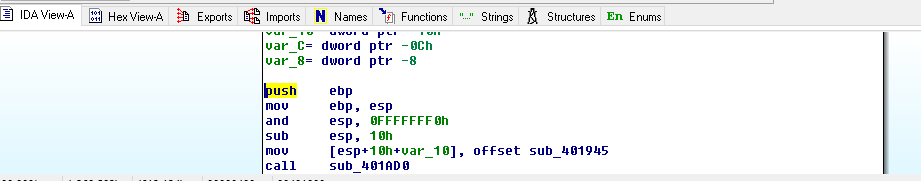
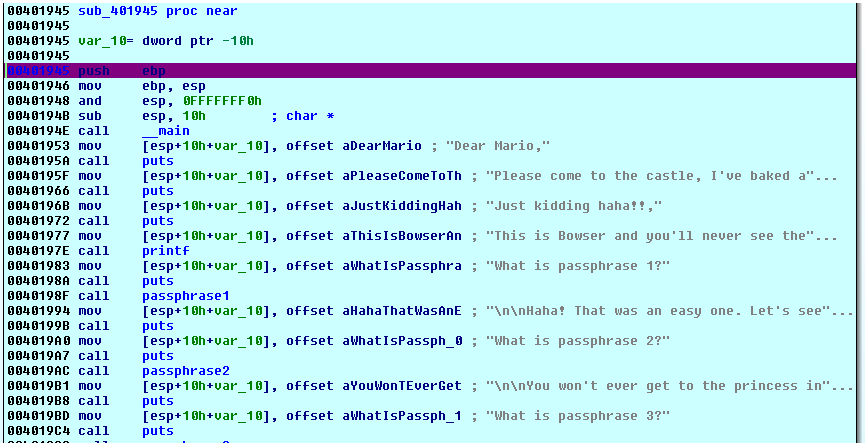
Group member:

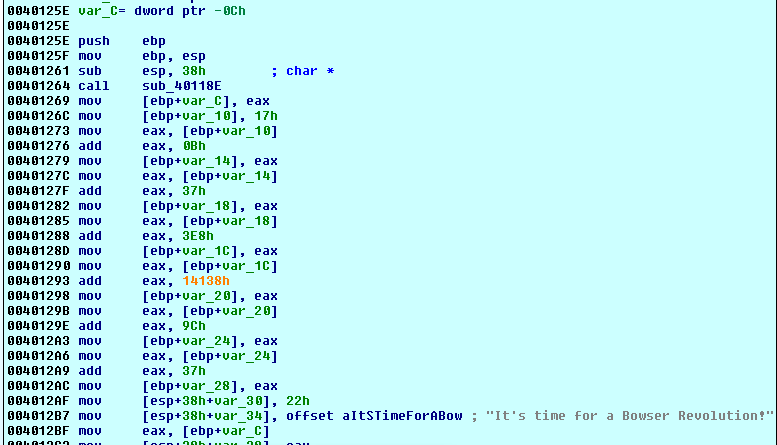
1. Md Abu Sayed, 80718658
2. Siyu Deng, 80741923

How we find start:

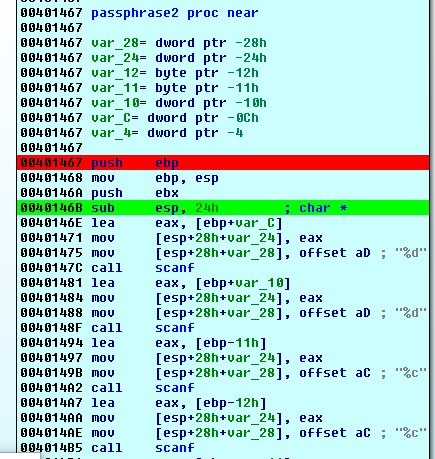




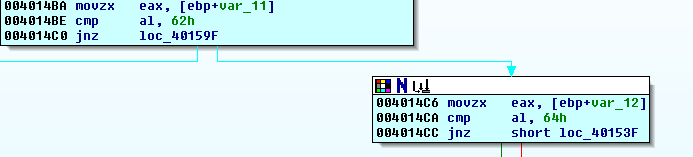




1. Passphrase1: It's time for a Bowser Revolution!
2. After looking at code passphrase2 and initial 4 scanf I realized that 4 local variable is used (var\_c, var\_10, var\_11, var\_12).



But do not understand how this function takes input. Then look at function and find something important (62h and 64h hex of ‘b’ and ‘d‘).



In debug mode try input ‘bd’, then found var\_11 and var\_12 holds ‘b’ and ‘d’. Then start input number try different input:

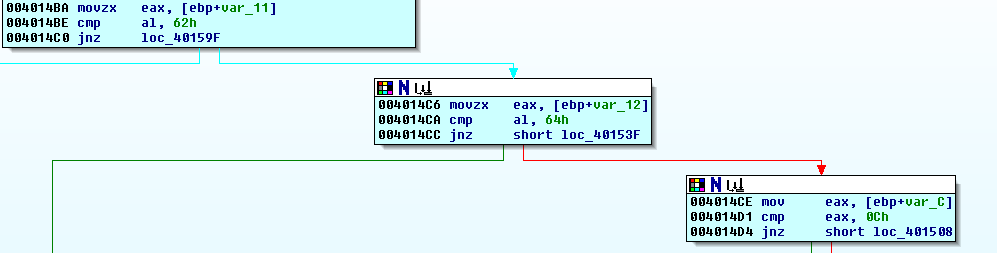
12bd: var\_c contains 12, var\_10 contains nothing, var\_11 and var\_12 holds ‘b’ and ‘d’

1 2 bd: var\_c contains 1, var\_10 contains 2, var\_11 and var\_12 holds space and ‘b’

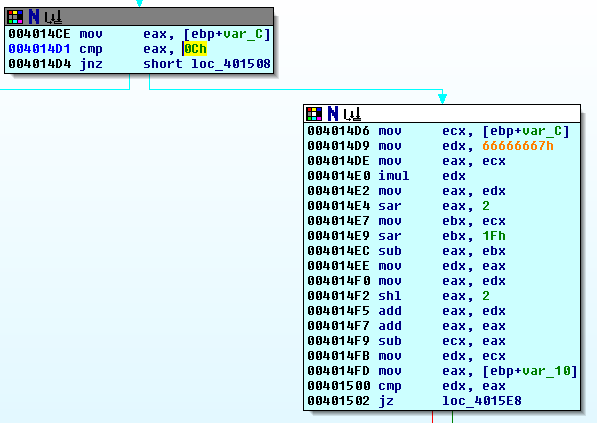
Finally try 20 10bd it works as input and var\_c contains 20, var\_10 contains 10, var\_11 and var\_12 holds ‘b’ and ‘d’. This is very explicit that’s why do not give screenshot.

A lot working 2nd secreet passphrase: **12 2bd, 234 4bd, 11 1br, 21 21br, 32 -10br, 2 22dr, 5 15dn**. I am going to explain how I got each of them.

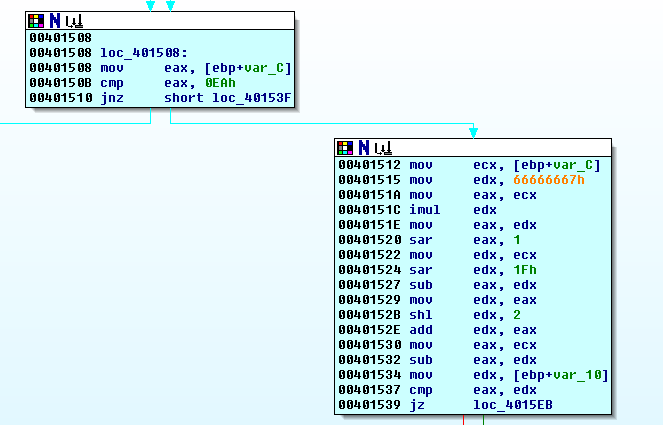
**12 2bd**: After checking var\_11 and var\_12 contain ‘b’ and ’d’, code check weather var\_c contain 0ch.



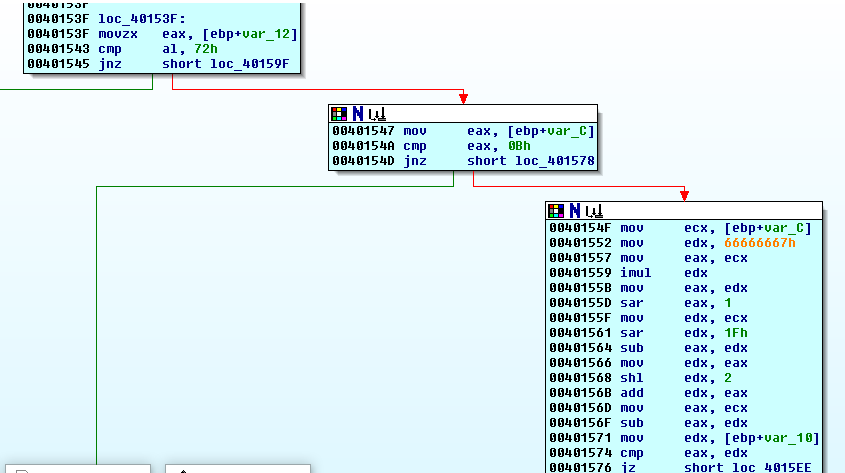
If that’s true then it check var\_10 equal to 2(0c % 10 = 2)(this is the meaning of below code). As that match then 12 2bd is valid passphrase.



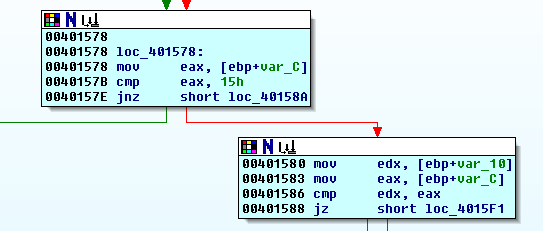
**234 4bd**: This follow the same pattern like previous one. Now var\_c = EAh = 234, var\_10 must be 4(234 %10 = 4).



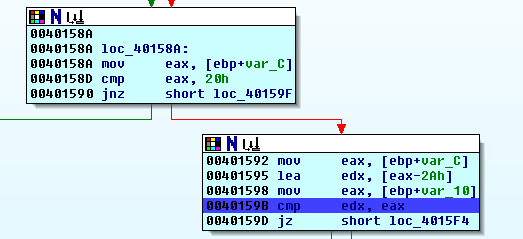
**11 1br**: This follow the same pattern like previous one. Now var\_c = Bh = 11, var\_10 must be 1(11 %10 = 1). This is for different var\_12 value(‘r’)



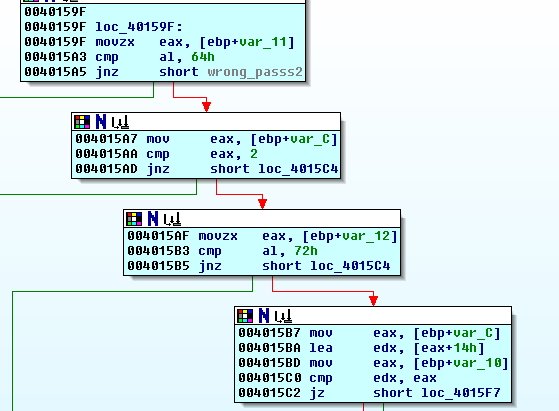
**21 21br**: var\_12 is same as previous one(‘r’). var\_c = 15h = 21. As directly compare var\_c and var\_10, so var\_10 must be 21.



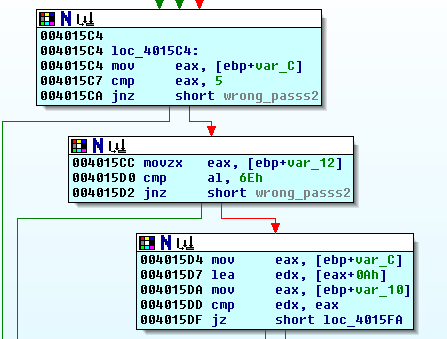
**32 -10br**: var\_12 is same as previous one(‘r’). var\_c = 20h = 32. As var\_c – 42 = 32- 42 = -10, so var\_10 must be -10.



**2 22dr**: var\_12 = ‘r’, var\_11 = ‘d’ var\_c = 2h = 2. var\_10 must be var\_c + 20 = 2 + 20 = 22.

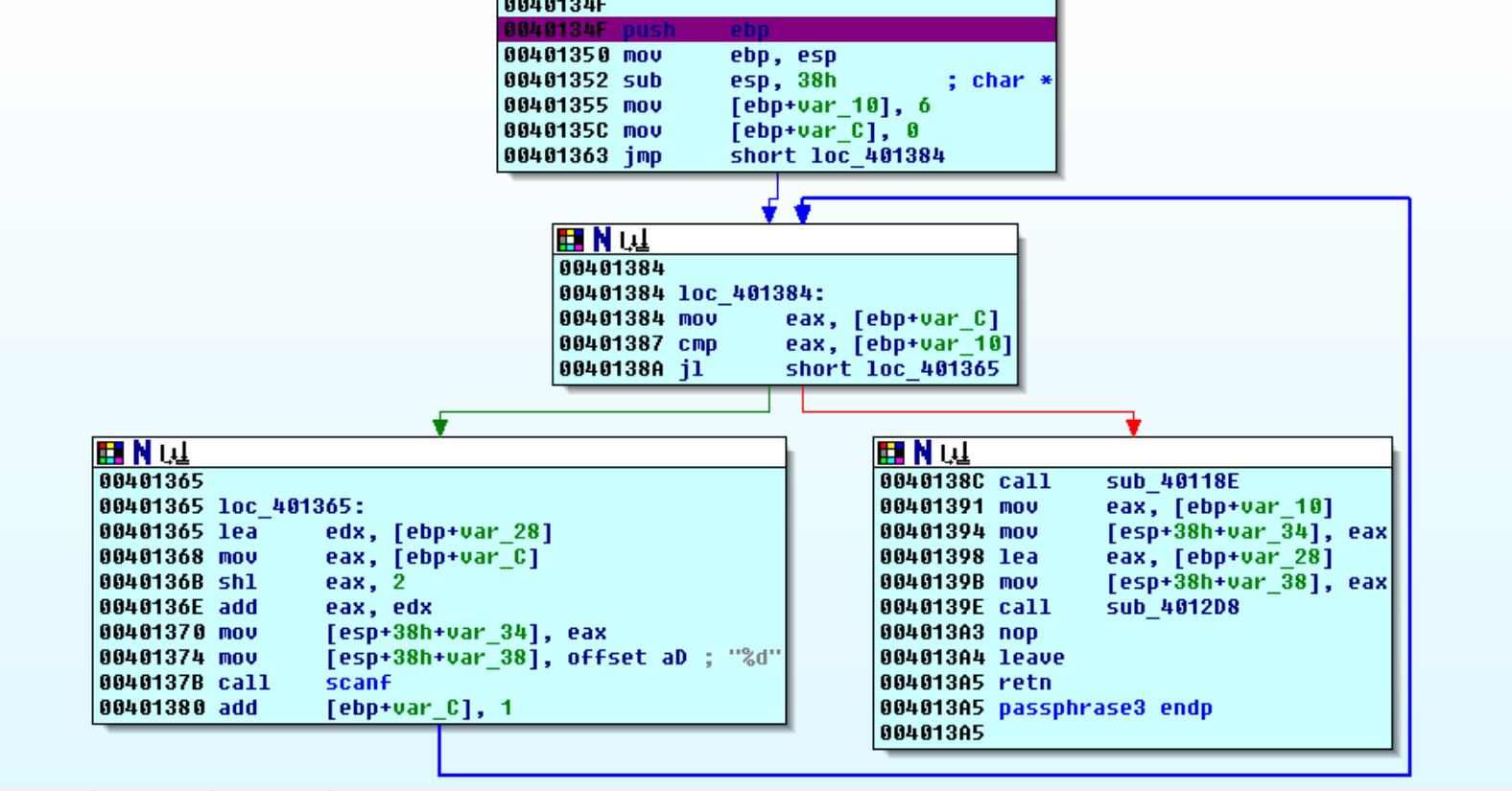


**5 15dn**: var\_12 = ‘n’, var\_11 = ‘d’, var\_c = 5h = 5. var\_10 must be var\_c + 10 = 5 + 10 = 15.

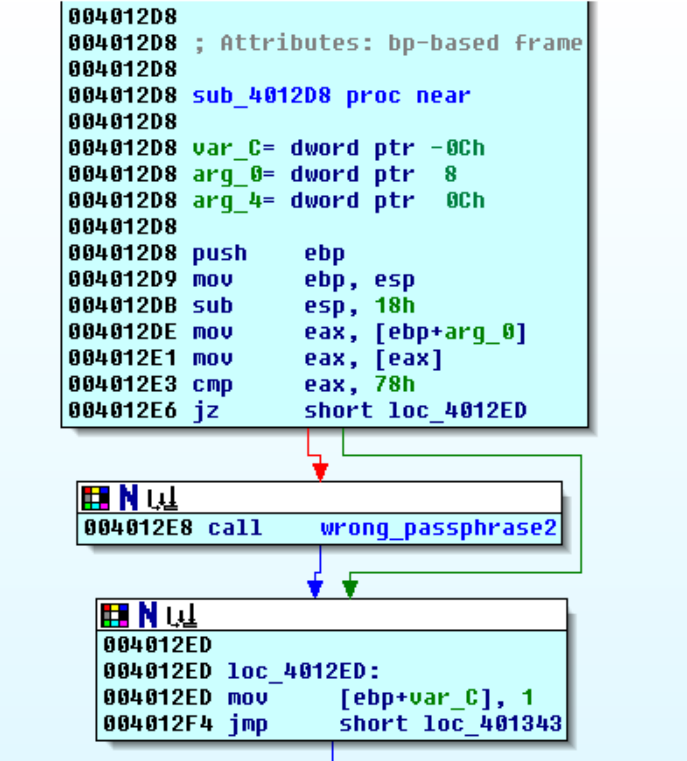


1. Passphrase : **120 35 91 323 1571 10083**

After seeing function passphrase3, I found using loop this function takes 6 numbers input and store them in var\_28 variable(size 24 bytes). After completing taking input, function call to sub\_40118E(not necessary for finding passphrase3 as it allocates some spaces for 6 numbers that accessed by next Call (sub\_4012D8)) and sub\_4012D8. Let’s focus on sub\_4012D8.



I found first number 78h = 120, by looking at this code, arg\_0 holds starting address of var\_28, thus eax contain first number and as check eax equal to 120, makes first number must be 120.



Checking of next number based on below code(loc\_4012F6).

var\_c = 1

Prev = 120, ecx = calculated number = **35**

Eax = 120 \*CCCCCCCDh = , eax = 00000018h, edx = 60h

Edx = 60h = 96 = 48 = 24

Eax = 1 = 0 = 0\*24 = 0 = 0 = 0 = 0 +35 = 35(next number)

var\_c = 2

Prev = 35, ecx = calculated number = **91**

Eax = 35 \*CCCCCCCDh = , eax = 00000007h, edx = 1Ch

Edx = 1Ch = 28 = 14 = 7

Eax = 2 = 1 = 1\*7 = 7 = 14 = 28 = 56 +35 = 91(next number)

var\_c = 3

Prev = 91, ecx = calculated number = **323**

Eax = 91 \*CCCCCCCDh = , eax = CCCCCCDFh, edx = 48h

Edx = 48h = 72 = 36 = 18

Eax = 3 = 2 = 2\*18 = 36 = 72 = 144 = 288 +35 = 323(next number)

var\_c = 4

Prev = 323, ecx = calculated number = **1571**

Eax = 323 \*CCCCCCCDh = , eax = 666666A7h, edx = 102h

Edx = 102h = 258 = 129 = 64

Eax = 4 = 3 = 3\*64 = 192 = 384 = 768 = 1536 +35 = 1571(next number)

var\_c = 5

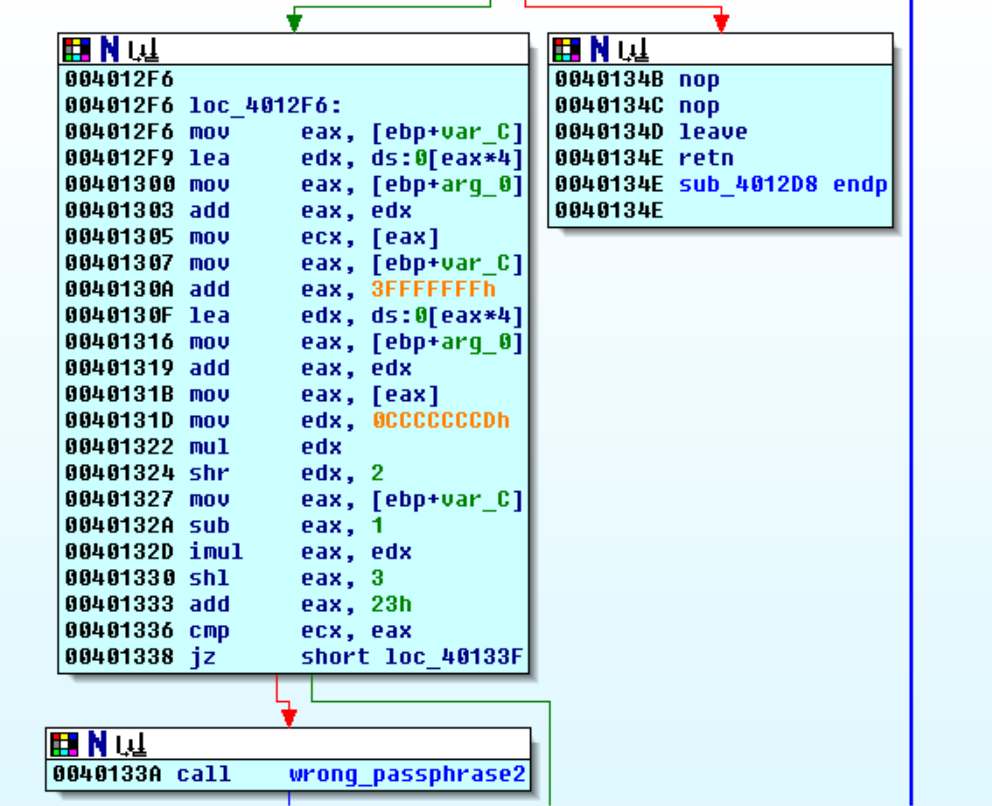
Prev = 1571, ecx = calculated number = **10083**

Eax = 1571 \*CCCCCCCDh = , eax = CCCCCE07h, edx = 4E8h

Edx = 4E8h = 1256 = 628 = 314

Eax = 5 = 4 = 4\*314 = 1256 = 2512 = 5024 = 10048 +35 = 10083(next number)

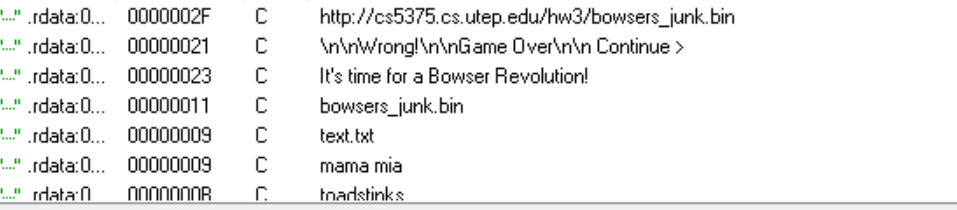
All numbers: 120 35 91 323 1571 10083



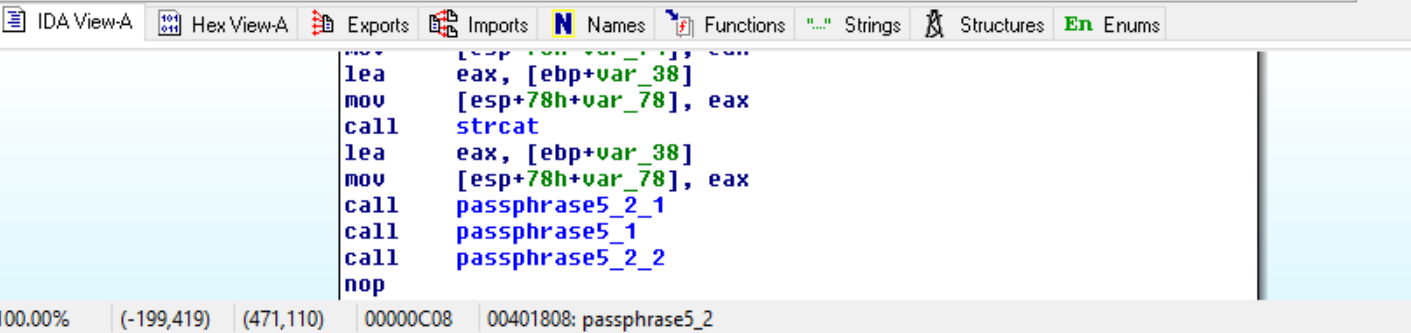
1. No Passphrase. We need to manipulate something. After look into function ‘passphrase4’, we saw it trying to open file ‘bowsers\_junk.bin’. But in debug mode, after call \_fpopen64, eax value(file pointer) is zero that means can not open that file.



Then we look at string and find a link to download ‘bowsers\_junk.bin’ file. we download that file and keep that file in same folder of trigger.exe and it open that file when we run trigger.exe.



1. Passphrase5 is little bit tricky. We need to manipulate a file that create when exe is running. ‘passphrase5\_2\_1’ create a ‘text.txt’ file and write ”mama mia”.



‘passphrase5\_2\_2’ read “text.txt” upto space and store it to var\_10B(“mama”). Then check whether var\_10B contains substring “toadstinks”. So if we change ‘text.txt’ content (‘mama’ to ‘toadstinks’) that actually solve passphrase5.

