**Aarch64 challenges**

**Binaries Link:** [**https://github.com/raravind7/ARM-challenges-and-writeups-for-ctf**](https://github.com/raravind7/ARM-challenges-and-writeups-for-ctf)

**Setting up the environment**

sudo apt-get install -qy qemu-user libc6-arm64-cross gdb-multiarch

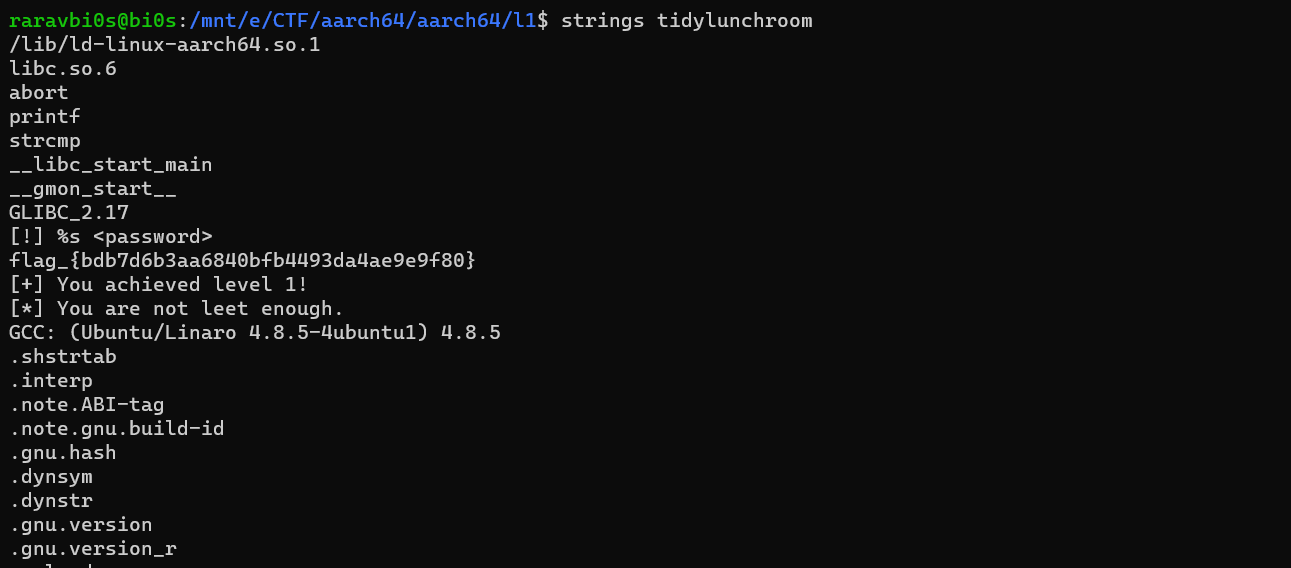
download lbc.so.6 and ld-linux-aarch64.so.1

and copy both to /lib

qemu-aarch64 -L /usr/aarch64-linux-gnu <filename> #to run

qemu-aarch64 -g <port> <filename> <input> #run with gdbserver

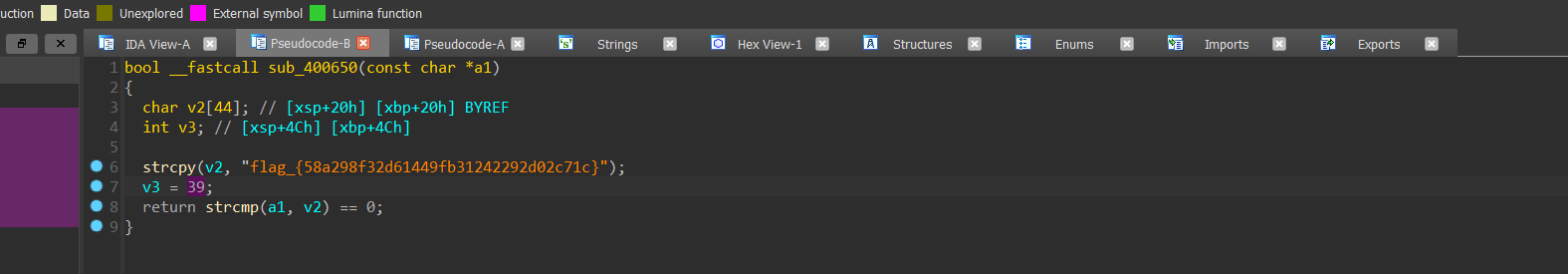
**Level 1:**

****

flag\_{bdb7d6b3aa6840bfb4493da4ae9e9f80}

**Level 2:**

Just opened in ida and we got the flag

****

flag\_{58a298f32d61449fb31242292d02c71c}

**Level 3:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Text

Description automatically generated**

****

Here in the function v3 is xored with 120 every time and u will get the flag

v3=[0]\*40

v3[0] = 30

v3[1] = 20

v3[2] = 25

v3[3] = 31

v3[4] = 39

v3[5] = 3

v3[6] = 73

v3[7] = 74

v3[8] = 27

v3[9] = 76

v3[10] = 65

v3[11] = 76

v3[12] = 77

v3[13] = 25

v3[14] = 72

v3[15] = 64

v3[16] = 78

v3[17] = 79

v3[18] = 76

v3[19] = 75

v3[20] = 75

v3[21] = 64

v3[22] = 26

v3[23] = 27

v3[24] = 76

v3[25] = 74

v3[26] = 77

v3[27] = 29

v3[28] = 75

v3[29] = 28

v3[30] = 75

v3[31] = 27

v3[32] = 72

v3[33] = 27

v3[34] = 65

v3[35] = 74

v3[36] = 79

v3[37] = 27

v3[38] = 5

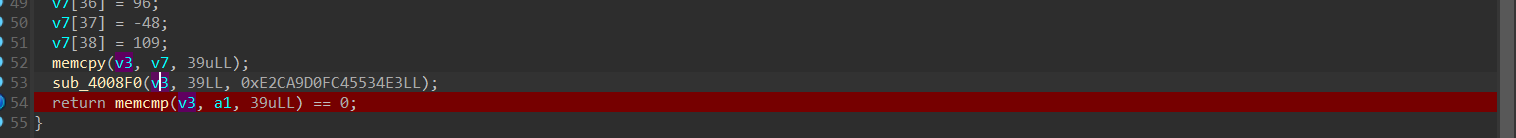
v3[39] = 0

for i in range(39):

  print(chr(v3[i]^120), end='')

flag\_{12c4945a08674338bc425e3d3c0c927c}

**Level 4:**

****

Created breakpoint here and checked what’s v3

flag\_{2a14e64b49b74a4792f9101577e7c946}

**LEVEL 5:**

**A picture containing rectangle

Description automatically generated**

Found v7 by debugging, v6 is the encrypted array

v6=[0x5B, 0xC7, 0xB6, 0x23, 0xDA, 0x7C, 0xB5, 0x44, 0x6A, 0xB3, 0xFC, 0x5E,0xD2, 0x15, 8, 0x8C, 0xAB, 0x9A, 0x8E, 0x97, 0x92, 0xFA, 0x46, 0xD1,0x17, 0x9B, 0x36, 0x83, 0x6A, 0x71, 0x91, 0xF7, 0xA1, 0x27, 0xCF, 0x57,0xFC, 0xC6, 0x9E]

#input

i=0

v7=[0x3D, 0xAB, 0xD7, 0x44, 0x85, 7, 0x81, 0x22, 0xE, 0xD5, 0xCB, 0x6A,0xE4, 0x76, 0x38, 0xB8, 0xCD, 0xAD, 0xBA, 0xA1, 0xF4, 0x9C, 0x24, 0xE5,0x27, 0xA2, 0x53, 0xB0, 0x52, 0x14, 0xF0, 0xC6, 0x91, 0x44, 0xFD, 0x35,0x9E, 0xFF, 0xE3]

while(i<=39):

    v6[i]^= v7[i]

    i+=1

for i in v6:

    print(chr(i),end='')

flag\_{4fdf746c04f746ffb409e38ea10c2bb9}]

**LEVEL 6:**

**Graphical user interface, text, application

Description automatically generated**

Found v7 and v6.

v7=[0xEC, 0x23, 0x91, 0xBC, 0x19, 0xA, 0x72, 0xD6, 0x31, 0x23, 0x49, 0x9B,0x11, 0xAF, 0x8F, 0x9E, 0xF8, 0x16, 0x8B, 0xBC, 0xF, 0xAF, 0xC1, 0xF3,0x13, 0xD1, 0x25, 0xCE, 0x95, 0x9F, 0x53, 0x11, 0x1D, 0x3C, 0xD0, 0x3D,0x94, 0x60]

l=[0x8A, 0x4F, 0xF0, 0xDB, 0x46, 0x71, 0x46, 0xB3, 0x53, 0x12, 0x78, 0xFA,0x25, 0x9D, 0xEC, 0xFA, 0xC9, 0x22, 0xBF, 0xDD, 0x38, 0xCB, 0xA3, 0xC2,0x77, 0xB0, 0x1D, 0xFD, 0xF3, 0xA9, 0x6B, 0x20, 0x2A, 0x5F, 0xE0, 0xA,0xA5, 0x53]

print(len(v7))

print(len(l))

i=0

while(i<38):

    l[i]^= v7[i]

    i+=1

l.append(ord('}'))

for i in l:

    print(chr(i),end='')

flag\_{4eb11a42cd144a7db1da83f6817c0713}

**Level 7:**

**Text

Description automatically generated**

If condition is patched to if not equal to , changed beq to bne

Text

Description automatically generated

Here v6 is given as the encrypted key , v7 both found by debugging

v7=[7, 0xF3, 0xF4, 0x22, 0xAC, 4, 0xE8, 0xC6, 0xFE, 0xE, 0x98, 0xD3, 0x40, 0xEC, 0xE1, 0xAA, 0x7F, 0xE3, 0xC5, 0x98, 0x8F, 0x59, 0x9B, 0xC9, 0xD8,0x13, 0x73, 0x97, 0xFD, 0xBB, 0xCE, 0x50, 0xF5, 0xEB, 0x3A, 0x2B, 0xC3,0x3F, 0x64]

l=[0x61, 0x9F, 0x95, 0x45, 0xF3, 0x7F, 0x8A, 0xA3, 0xCC, 0x3A, 0xAB, 0xEB,0x73, 0x8E, 0x85, 0xCF, 0x4B, 0xD5, 0xF1, 0xA8, 0xB7, 0x3C, 0xA2, 0xA8,0xEA, 0x72, 0x43, 0xA7, 0x9C, 0x8A, 0xA8, 0x66, 0xC6, 0xDD, 0x5E, 0x48,0xA1, 7, 0x19]#l=v6

i=0

while(i<39):

    l[i]^= v7[i]

    i+=1

for i in l:

    print(chr(i),end='')

flag\_{be24383bde46408e9a2a00a1f636dcb8}

**Level 8:**

**Text

Description automatically generated**

Here found the v6 the encrypted key, then byte\_4121A9

l=[0xEB, 0xB0, 0xEA, 0x8F, 0x3B, 0x3B, 0xF3, 0xFA, 0xCF, 0x57, 0xC9, 0xB4,0x2A, 0xDF, 6, 0xC8, 0xB3, 6, 0x1C, 1, 0x6B, 0x67, 0xCB, 0x7D, 0x9E,0xD5, 0x37, 0x36, 0x77, 0x76, 2, 0x17, 0xFA, 0xF6, 0x38, 0x59, 0xCD,0x7B, 0xDE]

v7=[0x8D, 0xDC, 0x8B, 0xE8, 0x64, 0x40, 0x97, 0xC2, 0xF7, 0x32, 0xFF, 0xD7, 0x4B, 0xEE, 0x36, 0xAA, 0x83, 0x36, 0x28, 0x33, 0x58, 0x56, 0xAA, 0x18, 0xAE, 0xB7, 0x05, 0x07, 0x13, 0x10, 0x64, 0x2E, 0x99, 0xCF, 0x5A, 0x68, 0xFA, 0x1E, 0xA3]

#byte

i=0

while(i<39):

    l[i]^= v7[i]

    i+=1

for i in l:

    print(chr(i),end='')

flag\_{d88e6ca10b004231ae0b21dff9c9b17e}

Level 9:

Graphical user interface

Description automatically generated

This at the end of main

The for loop is what we have to reverse, giving v55 as encrypted key and finding v7 by debugging

l=[0xE4, 0x68, 0x2B, 0x7E, 0xCB, 0xF0, 0xFF, 0x44, 0x7F, 0xEA,

  0x7D, 0xD0, 0x76, 0x81, 0x1C, 0x0A, 0x26, 0x46, 0x09, 0x79,

  0xEF, 0xFF, 0x58, 0xEF, 0xCE, 0x63, 0xEE, 0x2C, 0x28, 0x84,

  0xA2, 0xAF, 0x28, 0xD4, 0x15, 0x62, 0xDB, 0x47, 0x27]

v7=[

  0x82, 0x04, 0x4A, 0x19, 0x94, 0x8B, 0xC7, 0x76, 0x4F, 0x8C,

  0x4E, 0xB4, 0x44, 0xE2, 0x2F, 0x69, 0x43, 0x7F, 0x3D, 0x4B,

  0xDE, 0x9D, 0x3A, 0xDC, 0xF6, 0x54, 0xDD, 0x1A, 0x11, 0xB5,

  0xC4, 0xCC, 0x4B, 0xEC, 0x23, 0x00, 0xE8, 0x22, 0x5A

]

i=0

while(i<39):

    l[i]^= v7[i]

    i+=1

for i in l:

    print(chr(i),end='')

flag\_{820f3d2c3ce9421bb3873691fcc86b3e}

Level 10:

During the runtime they are creating set of assembly from ‘dest’, dest is the function that checks our input. So set a breakpoint before dest. And went to dest. Initially it was of just hex bytes in ida, changed to assembly. Then used create function. Then disassembled dest and took two lists that needs to be xored to get the flag

l=[

  0x9F, 0xED, 0x5D, 0x30, 0xC7, 0xFB, 0x35, 0x3B, 0x84, 0xFB,

  0x34, 0xAE, 0x19, 0xBD, 0x7C, 0x69, 0x51, 0xF6, 0xE8, 0xBA,

  0x5E, 0xC2, 0x63, 0xF6, 0x2F, 0xB2, 0xF8, 0xBC, 0xC6, 0x32,

  0x0B, 0x2E, 0x0D, 0x04, 0x6E, 0x87, 0x39, 0x0F, 0x78

]

x=[

  0xF9, 0x81, 0x3C, 0x57, 0x98, 0x80, 0x51, 0x0D, 0xB4, 0x9F,

  0x01, 0xCF, 0x2C, 0xDF, 0x1F, 0x5A, 0x33, 0x95, 0xDC, 0xD9,

  0x38, 0xF2, 0x02, 0xC0, 0x1B, 0x82, 0xCC, 0xDF, 0xF4, 0x54,

  0x32, 0x1E, 0x39, 0x3D, 0x5C, 0xB5, 0x5C, 0x39, 0x05

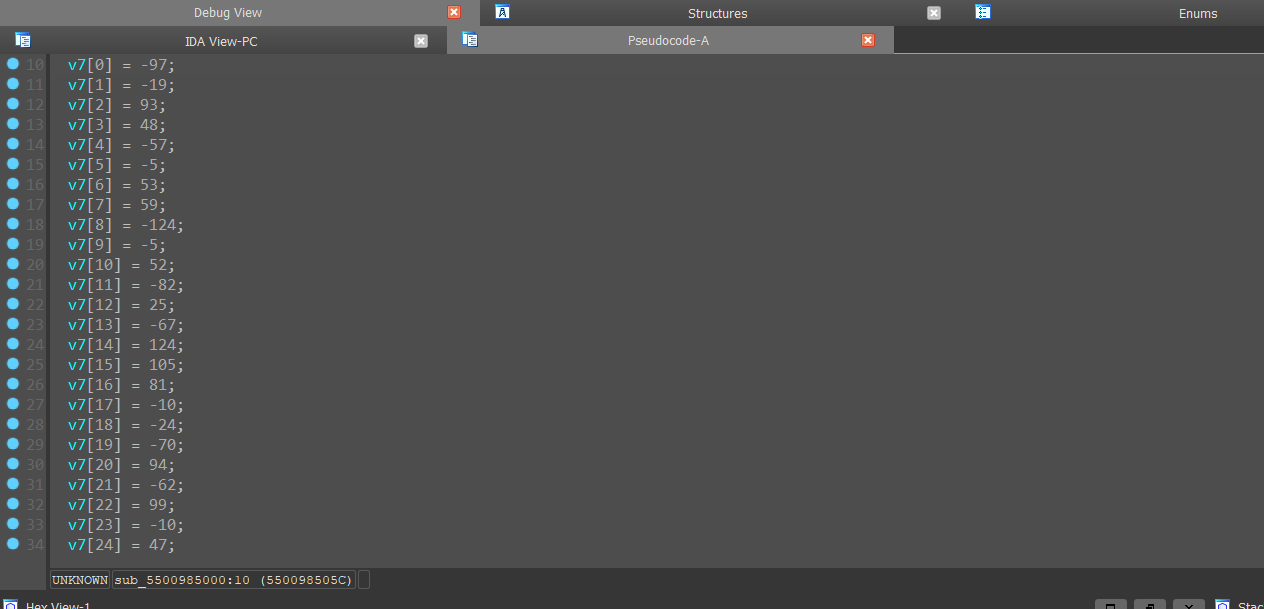
]

X=[]

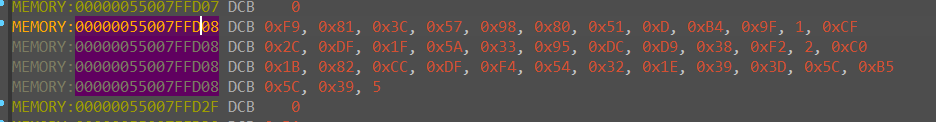
for i in range(39):

  print(chr(l[i]^x[i]), end='')

first array



Second array



flag\_{d60d5a5bc3bc4cf0a6404c2f904922e6}

Level 11:

Text

Description automatically generated

Patched and made the all returns inside sub\_4010B4 as 1 lol ,ya to bypass our wrong inputs

l=[

  0x24, 0xD1, 0x49, 0xCA, 0x35, 0x43, 0xDD, 0xFF, 0xA4, 0x58,

  0xD4, 0x59, 0x5A, 0x5A, 0x33, 0xD5, 0x9A, 0x85, 0xEE, 0x4C,

  0x40, 0xFD, 0x78, 0x4D, 0xE2, 0xA2, 0xEC, 0x56, 0x5A, 0xDF,

  0xBF, 0x7B, 0x98, 0xAC, 0xBC, 0x90, 0x75, 0xFD, 0x7C

]

v7=[

  0x42, 0xBD, 0x28, 0xAD, 0x6A, 0x38, 0xE9, 0xCC, 0xC6, 0x6C,

  0xE1, 0x6A, 0x6C, 0x6B, 0x02, 0xB0, 0xAF, 0xB3, 0xDA, 0x29,

  0x73, 0x9B, 0x41, 0x7F, 0xD2, 0x97, 0x88, 0x60, 0x62, 0xEC,

  0x89, 0x4F, 0xA8, 0xCA, 0xDD, 0xF4, 0x4D, 0xCC, 0x01, 0x00

]

i=0

while(i<39):

    l[i]^= v7[i]

    i+=1

for i in l:

    print(chr(i),end='')

flag\_{43b453611e564e3f9205d683640fad81}