**HW2 - Assembly Analysis**

**Question 1 (15 points)**

start:

push ebp

mov ebp,esp

sub esp,0x10

mov eax,DWORD PTR [esp] ; what’s in esp? 0?

add eax,0x2e48 ; eax = 11848, the result of add isn’t used, I think.

mov DWORD PTR [ebp-0x8],0x30 ; y = 48, hex 30 means ‘0’ in ASCII

mov DWORD PTR [ebp-0x4],0x0 ; x = 0

jmp loc\_2

loc\_1:

mov edx,DWORD PTR [ebp-0x4]

mov eax,DWORD PTR [ebp+0x8]

add eax,edx

movzx eax,BYTE PTR [eax]

movsx eax,al ; what does that mean?

xor DWORD PTR [ebp-0x8],eax ; what does that mean?

add DWORD PTR [ebp-0x4],0x1

loc\_2:

mov eax, DWORD PTR [ebp-0x4]

cmp eax,DWORD PTR [ebp+0xc] ; what’s in ebp + 0xc? Arg\_4?

jl loc\_1

loc\_3:

mov eax,DWORD PTR [ebp-0x8]

leave

ret

1.1) In a few sentences, explain what this function does. (5 pts)

--------------------------------------------------------------------------------------------

I think the function XORing on an array with ‘0x30’ until it hits user’s input number(arg\_4). And it tries to XOR on each byte in an input array.

--------------------------------------------------------------------------------------------

1.2) Write a function in C that is equivalent to the assembly above. (5 pts)

--------------------------------------------------------------------------------------------

int func3(char\* arg\_0, int arg\_4) {

int x = 0;

int ret = 0x30;

while( x < arg\_4 ) {

ret = ret ^ arg\_0[x];

x = x + 1;

}

return ret;

}

--------------------------------------------------------------------------------------------

1.3) Let arg\_0 be a pointer to the string "x64 is better than x86" and let arg\_4 be 22.

What does the function return? (5 pts)

--------------------------------------------------------------------------------------------

**The function will return ‘0x25’ or 37(integer).**

Below is my code for a reference.

#include <stdio.h>

void print\_bits(unsigned int x)

{

int i;

for(i=8\*sizeof(x)-1; i>=0; i--) {

(x & (1 << i)) ? putchar('1') : putchar('0');

}

printf("\n");

}

int main()

{

const char \* input = "x64 is better than x86";

int a = 0x30;

for(int i=0;i<22;i++) {

// printf("%u,", input[i]);

printf("iteration:%d,", i);

printf("in char %c,", input[i]);

printf("in hex %x\n", input[i]);

// print\_bits(input[i]);

// printf("%u,", a);

printf(" input1:"); print\_bits(a);

printf(" input2:"); print\_bits(input[i]);

a = a ^ input[i];

printf("XOR bits:");print\_bits(a);

printf("XOR hex:%x",a);

printf("\n");

}

printf("\n");

return 0;

}

--------------------------------------------------------------------------------------------

**Question 2 (15 points)**

start:

PUSH EBP

MOV EBP, ESP

MOV ECX, [EBP+arg\_0] ; ecx = arg\_0

MOV ESI, [EBP+arg\_4] ; esi = arg\_4

MOV [EBP+var\_1], 0 ; var\_1 = 0

JMP loc\_2

loc\_1:

MOV EAX, [EBP+var\_1] ; eax = var\_1

ADD EAX, ECX ; eax = ecx + var\_1

MOV EDX, byte ptr [EAX] ; edx = [eax]

XOR EDX, ESI ; edx = edx ^ esi

MOV [EAX], DL ; what is DL?

ADD [EBP+var\_1], 0x1 ; var\_1 += 1

loc\_2:

MOV EAX, [EBP+var\_1] ; eax = var\_1

CMP byte ptr [ECX + EAX], 0 ; if( ecx + eax != 0 )

JNZ loc\_1 ; jump when values are diff

MOV ESP, EBP

POP EBP

RETN

I think given arg\_0 = [‘z’,’x’,’c’,’d’], arg\_4 = 57

arg\_0 = [1,1,1,0];

2.1) In a few sentences, explain what this function does. (5 pts)

--------------------------------------------------------------------------------------------

I think this is the code for XORing an array(arg\_0) with the value of second parameter(arg\_4) until it encounters 0. It converts an array’s each byte value using XOR operation with second parameter.

--------------------------------------------------------------------------------------------

2.2) Write a function in C that is equivalent to the assembly above. (5 pts)

--------------------------------------------------------------------------------------------

int func2(char\* arg\_0, int arg\_4) {

// Your code goes here

Int var\_1 = 0;

while( arg\_0[var\_1] != 0 ) {

arg\_0[var\_1] = arg\_0[var\_1] ^ arg\_4

var\_1 = var\_1 + 1

}

}

--------------------------------------------------------------------------------------------

2.3) Let arg\_0 be a pointer to the null-terminated string "\xa7\xa4\xe2\xaf\xcf\xd2\xc6\xf1\xe1\xe3\xf3\xcc\xaf\xd8\xef\xdb\xf1\xe1\xa3\xa7\xaf\xe6\xa1\xd7\xd7\xae\xff\xe5\xfc\xe4\xa0\xc5\xdb\xe1" and let arg\_4 be the integer 0x96.

What is the value of the string pointed to by arg\_0 when the function completes? (Hint: It will decode to a bitcoin wallet) What value does the function return? (5 pts)

--------------------------------------------------------------------------------------------

**The result would be, “12t9YDPgwueZ9NyMgw519p7AA8isjr6SMw”.**

Below is a program which prints out the result in a character for my reference.

#include <stdio.h>

int main()

{

const char \*input = "\xa7\xa4\xe2\xaf\xcf\xd2\xc6\xf1\xe1\xe3\xf3\xcc\xaf\xd8\xef\xdb\xf1\xe1\xa3\xa7\xaf\xe6\xa1\xd7\xd7\xae\xff\xe5\xfc\xe4\xa0\xc5\xdb\xe1";

for(int i=0;i<34;i++) {

printf("%c", input[i] ^ 0x96);

}

printf("\n");

return 0;

}

--------------------------------------------------------------------------------------------

**Question 3 (10 points):**

public main

main proc near

var\_20 = dword ptr -20h

var\_1C = dword ptr -1Ch

var\_18 = dword ptr -18h

var\_10 = dword ptr -10h

var\_8 = dword ptr -8

push ebp

mov ebp, esp

and esp, 0FFFFFFF0h

sub esp, 20h

lea eax, [esp+20h+var\_8]

mov [esp+20h+var\_20], eax

call \_PQR1cC1Ev

mov [esp+20h+var\_18], 12

mov [esp+20h+var\_1C], 4

lea eax, [esp+20h+var\_10]

mov [esp+20h+var\_20], eax

call \_PQR1cC2Eii

lea eax, [esp+20h+var\_8]

mov [esp+20h+var\_20], eax

call \_PQR1c4dumpEv

lea eax, [esp+20h+var\_10]

mov [esp+20h+var\_20], eax

call \_PQR1c4dumpEv

mov eax, 0

leave

retn

main endp

The code above is the assembly language of the main function of a C++ program. The program code was compiled with gcc. The C++ program defines one class that consists of two constructors and one method. Both constructors and methods are being used in the main function above.

Write the main function in C++ that is equivalent to the assembly above. Note that you are not asked to explain the functionality of the class, just to provide the abstract C++ main of the code above. Don’t get caught in the details!

--------------------------------------------------------------------------------------------

class ABC

{

     private:

        int x,y;

     public:

        ABC ()       //constructor 1 with no arguments

       {

            x = 4;

y = 12;

        }

        ABC(int a, int b)    //constructor 2 with one argument

       {

            x = a

y = b;

        }

        void dump()

        {

system.print.out(‘%d %d’, x, y);

        }

};

int main()

{

ABC cc1 = new ABC(); //constructor without parameter

ABC cc2 = new ABC(1,2); //constructor with parameters

cc1.dump();

cc2.dump();

return 0;

}  //end of program

--------------------------------------------------------------------------------------------