# **CS 410 Binary to C++ With Security Vulnerabilities Activity Template**

**Step 1:** Convert the binary file to assembly code.

**DisplayMenu**

push %rbp

mov %rsp,%rbp

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0x17 <\_Z11DisplayMenuv+23>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0x2a <\_Z11DisplayMenuv+42>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0x3d <\_Z11DisplayMenuv+61>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0x50 <\_Z11DisplayMenuv+80>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0x63 <\_Z11DisplayMenuv+99>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0x76 <\_Z11DisplayMenuv+118>

nop

pop %rbp

ret

**main**

push %rbp

mov %rsp,%rbp

sub $0x20,%rsp

mov %fs:0x28,%rax

mov %rax,-0x8(%rbp)

xor %eax,%eax

movl $0x0,-0x14(%rbp)

mov -0x14(%rbp),%eax

cmp $0x5,%eax

je 0x308 <main+655>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0xb6 <main+61>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0xc9 <main+80>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0xdc <main+99>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0xef <main+118>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0x102 <main+137>

lea 0x0(%rip),%rsi

lea 0x0(%rip),%rdi

call 0x115 <main+156>

lea -0x14(%rbp),%rax

mov %rax,%rsi

lea 0x0(%rip),%rdi

call 0x128 <main+175>

mov -0x14(%rbp),%eax

cmp $0x1,%eax

jne 0x1c9 <main+336>

lea -0x10(%rbp),%rax

mov %rax,%rsi

lea 0x0(%rip),%rdi

call 0x147 <main+206>

mov %rax,%rdx

lea -0xc(%rbp),%rax

mov %rax,%rsi

mov %rdx,%rdi

call 0x159 <main+224>

mov -0x10(%rbp),%eax

mov %eax,%esi

lea 0x0(%rip),%rdi

call 0x16a <main+241>

lea 0x0(%rip),%rsi

mov %rax,%rdi

call 0x179 <main+256>

mov %rax,%rdx

mov -0xc(%rbp),%eax

mov %eax,%esi

mov %rdx,%rdi

call 0x189 <main+272>

lea 0x0(%rip),%rsi

mov %rax,%rdi

call 0x198 <main+287>

mov %rax,%rcx

mov -0x10(%rbp),%edx

mov -0xc(%rbp),%eax

sub %eax,%edx

mov %edx,%eax

mov %eax,%esi

mov %rcx,%rdi

call 0x1af <main+310>

mov %rax,%rdx

mov 0x0(%rip),%rax

mov %rax,%rsi

mov %rdx,%rdi

call 0x1c4 <main+331>

jmp 0x97 <main+30>

mov -0x14(%rbp),%eax

cmp $0x2,%eax

jne 0x268 <main+495>

lea -0x10(%rbp),%rax

mov %rax,%rsi

lea 0x0(%rip),%rdi

call 0x1e8 <main+367>

mov %rax,%rdx

lea -0xc(%rbp),%rax

mov %rax,%rsi

mov %rdx,%rdi

call 0x1fa <main+385>

mov -0x10(%rbp),%eax

mov %eax,%esi

lea 0x0(%rip),%rdi

call 0x20b <main+402>

lea 0x0(%rip),%rsi

mov %rax,%rdi

call 0x21a <main+417>

mov %rax,%rdx

mov -0xc(%rbp),%eax

mov %eax,%esi

mov %rdx,%rdi

call 0x22a <main+433>

lea 0x0(%rip),%rsi

mov %rax,%rdi

call 0x239 <main+448>

mov %rax,%rcx

mov -0x10(%rbp),%edx

mov -0xc(%rbp),%eax

add %edx,%eax

mov %eax,%esi

mov %rcx,%rdi

call 0x24e <main+469>

mov %rax,%rdx

mov 0x0(%rip),%rax

mov %rax,%rsi

mov %rdx,%rdi

call 0x263 <main+490>

jmp 0x97 <main+30>

mov -0x14(%rbp),%eax

cmp $0x3,%eax

jne 0x97 <main+30>

lea -0x10(%rbp),%rax

mov %rax,%rsi

lea 0x0(%rip),%rdi

call 0x287 <main+526>

mov %rax,%rdx

lea -0xc(%rbp),%rax

mov %rax,%rsi

mov %rdx,%rdi

call 0x299 <main+544>

mov -0x10(%rbp),%eax

mov %eax,%esi

lea 0x0(%rip),%rdi

call 0x2aa <main+561>

lea 0x0(%rip),%rsi

mov %rax,%rdi

call 0x2b9 <main+576>

mov %rax,%rdx

mov -0xc(%rbp),%eax

mov %eax,%esi

mov %rdx,%rdi

call 0x2c9 <main+592>

lea 0x0(%rip),%rsi

mov %rax,%rdi

call 0x2d8 <main+607>

mov %rax,%rcx

mov -0x10(%rbp),%eax

mov -0xc(%rbp),%esi

cltd

idiv %esi

mov %eax,%esi

mov %rcx,%rdi

call 0x2ee <main+629>

mov %rax,%rdx

mov 0x0(%rip),%rax

mov %rax,%rsi

mov %rdx,%rdi

call 0x303 <main+650>

jmp 0x97 <main+30>

mov $0x0,%eax

mov -0x8(%rbp),%rcx

xor %fs:0x28,%rcx

je 0x321 <main+680>

call 0x321 <main+680>

leave

ret

**Step 2:** Explain the functionality of the blocks of assembly code.

**DisplayMenu**

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp | Reserve stack pointer |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x17 <\_Z11DisplayMenuv+23> | Get the address of string “----------------” and print it out |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x2a <\_Z11DisplayMenuv+42> | Get the address of string “- 1)Add -” and print it out |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x3d <\_Z11DisplayMenuv+61> | Get the address of string “- 2)Subtract -” and print it out |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x50 <\_Z11DisplayMenuv+80> | Get the address of string “- 3)Multiply -” and print it out |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x63 <\_Z11DisplayMenuv+99> | Get the address of string “- 4)Exit -” and print it out |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x76 <\_Z11DisplayMenuv+118> | Get the address of string “----------------” and print it out |
| nop  pop %rbp  ret | Exit the proc |

**main**

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax | Reserve Stack Pointer and Reserve 32 bytes (4 int variables)  Set EAX register to zero |
| movl $0x0,-0x14(%rbp)  mov -0x14(%rbp),%eax  cmp $0x5,%eax  je 0x308 <main+655>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xb6 <main+61>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xc9 <main+80>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xdc <main+99>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xef <main+118>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x102 <main+137>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x115 <main+156>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x128 <main+175>   mov -0x14(%rbp),%eax | Start a for loop with checking the variable value to be 5 however, we don’t see inc instruction which means the program will run forever.  Get the address of string “----------------” and print it out  Get the address of string “- 1)Add -” and print it out  Get the address of string “- 2)Subtract -” and print it out  Get the address of string “- 3)Multiply -” and print it out  Get the address of string “- 4)Exit -” and print it out  Get the address of string “----------------” and print it out  Get a value from the user |
| cmp $0x1,%eax  jne 0x1c9 <main+336> | Check if the value if 1 then |
| lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x147 <main+206>  mov %rax,%rdx  lea -0xc(%rbp),%rax | Fetch the num1 and num2 |
| mov %rax,%rsi  mov %rdx,%rdi  call 0x159 <main+224>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi  call 0x16a <main+241>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x179 <main+256>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x189 <main+272>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x198 <main+287>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  sub %eax,%edx  mov %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  call 0x1af <main+310> | Get the address of “ – “ and get the address of “ = “  And print out the num1 followed by “-“ and num2 and then “ = “ print out the subtraction |
| mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x1c4 <main+331> | Move back to the loop print out the menu and fetch user choice. |
| cmp $0x2,%eax  jne 0x268 <main+495>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x1e8 <main+367> | If choice is 2 then, get num1 from user |
| mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x1fa <main+385>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi  call 0x20b <main+402>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x21a <main+417>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x22a <main+433>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x239 <main+448>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  add %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  call 0x24e <main+469>  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x263 <main+490>  jmp 0x97 <main+30>  mov -0x14(%rbp),%eax | Get num2 from the user,  Print out num1 followed by “ – “ and num2,  “ = “ the summation of num1, num2  Jump back to the beginning of the loop.  Fetch the choice from the user |
| cmp $0x3,%eax  jne 0x97 <main+30>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x287 <main+526>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x299 <main+544>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi  call 0x2aa <main+561>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x2b9 <main+576>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x2c9 <main+592>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x2d8 <main+607>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi  cltd  idiv %esi  mov %eax,%esi  mov %rcx,%rdi  call 0x2ee <main+629>  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x303 <main+650> | If the choice is 3 get num1, num2 from the user and the reference of “ – “ and “ = “  Print out num1 divided num2 |
| jmp 0x97 <main+30>  mov $0x0,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx  je 0x321 <main+680>  call 0x321 <main+680>  lea | Jump back to the beginning of the loop |
|  |  |

**Step 3:** Convert the assembly code to binary.

**Attached**

**Step 4:** Convert the assembly code to C++ code.

**DisplayMenu**

| **Blocks of Assembly Code** | **C++ Code** |
| --- | --- |
| push %rbp  mov %rsp,%rbp | Void DisplayMenu(void){ |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x17 <\_Z11DisplayMenuv+23> | cout << “---------------- ” << endl; |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x2a <\_Z11DisplayMenuv+42> | cout << “- 1)Add -” << endl; |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x3d <\_Z11DisplayMenuv+61> | cout << “- 2)Subtract -” << endl; |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x50 <\_Z11DisplayMenuv+80> | cout << “- 3)Multiply -” << endl; |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x63 <\_Z11DisplayMenuv+99> | cout << “- 4)Exit -” << endl; |
| lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x76 <\_Z11DisplayMenuv+118> | cout << “----------------” << endl; |
| nop  pop %rbp  ret | Exit the proc |

**main**

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| push %rbp  mov %rsp,%rbp  sub $0x20,%rsp  mov %fs:0x28,%rax  mov %rax,-0x8(%rbp)  xor %eax,%eax | Reserve Stack Pointer and Reserve 32 bytes (4 int variables)  Set EAX register to zero  int main(void){  int choice, num1, num2,I; |
| movl $0x0,-0x14(%rbp)  mov -0x14(%rbp),%eax  cmp $0x5,%eax  je 0x308 <main+655>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xb6 <main+61>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xc9 <main+80>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xdc <main+99>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0xef <main+118>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x102 <main+137>  lea 0x0(%rip),%rsi  lea 0x0(%rip),%rdi  call 0x115 <main+156>  lea -0x14(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x128 <main+175>   mov -0x14(%rbp),%eax | for(i = 0 ; i < 5 ; )  {  cout << “---------------- ” << endl;  cout << “- 1)Add -” << endl;  cout << “- 2)Subtract -” << endl;  cout << “- 3)Multiply -” << endl;  cout << “- 4)Exit -” << endl;  cout << “----------------” << endl;  cin >> choice; |
| cmp $0x1,%eax  jne 0x1c9 <main+336> | If( choice == 1) { |
| lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x147 <main+206>  mov %rax,%rdx  lea -0xc(%rbp),%rax | cin >> num1;  cin >> num2; |
| mov %rax,%rsi  mov %rdx,%rdi  call 0x159 <main+224>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi  call 0x16a <main+241>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x179 <main+256>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x189 <main+272>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x198 <main+287>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  sub %eax,%edx  mov %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  call 0x1af <main+310> | cout << num1 << “ – “ << num2 << “ = “ << num1-num2 << endl;  } |
| mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x1c4 <main+331> |  |
| cmp $0x2,%eax  jne 0x268 <main+495>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x1e8 <main+367> | if (choice == 2){  cin >> num1 ;  cin >> num2 ;  cout << num1 << “ – “ << num2 << “ = “ << num1+num2 << endl;  } |
| mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x1fa <main+385>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi  call 0x20b <main+402>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x21a <main+417>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x22a <main+433>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x239 <main+448>  mov %rax,%rcx  mov -0x10(%rbp),%edx  mov -0xc(%rbp),%eax  add %edx,%eax  mov %eax,%esi  mov %rcx,%rdi  call 0x24e <main+469>  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x263 <main+490>  jmp 0x97 <main+30>  mov -0x14(%rbp),%eax |  |
| cmp $0x3,%eax  jne 0x97 <main+30>  lea -0x10(%rbp),%rax  mov %rax,%rsi  lea 0x0(%rip),%rdi  call 0x287 <main+526>  mov %rax,%rdx  lea -0xc(%rbp),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x299 <main+544>  mov -0x10(%rbp),%eax  mov %eax,%esi  lea 0x0(%rip),%rdi  call 0x2aa <main+561>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x2b9 <main+576>  mov %rax,%rdx  mov -0xc(%rbp),%eax  mov %eax,%esi  mov %rdx,%rdi  call 0x2c9 <main+592>  lea 0x0(%rip),%rsi  mov %rax,%rdi  call 0x2d8 <main+607>  mov %rax,%rcx  mov -0x10(%rbp),%eax  mov -0xc(%rbp),%esi  cltd  idiv %esi  mov %eax,%esi  mov %rcx,%rdi  call 0x2ee <main+629>  mov %rax,%rdx  mov 0x0(%rip),%rax  mov %rax,%rsi  mov %rdx,%rdi  call 0x303 <main+650> | if(choice == 3)  {  cin >> num1 ;  cin >> num2 ;  cout << num1 << “ – “ << num2 << “ = “ << num1/num2 << endl;  } |
| jmp 0x97 <main+30>  mov $0x0,%eax  mov -0x8(%rbp),%rcx  xor %fs:0x28,%rcx  je 0x321 <main+680>  call 0x321 <main+680>  lea | return 0;  } |