**CS 410 Stepping Stone Seven**

**\*\*\*\*\* \*\*\*\*\*\***

**CS-410 Software Reverse Engineering**

**Southern New Hampshire University**

## Code Functionality.

From the analysis, and using the application, I found out that the main objective of this application is to store students grades and their respective student names. Also the other function that I sow is to secure the data, the reason why a password and user names must be provided for one to view the data.

The program have the following functions.

ReadUserInfo

This functions is used to prompt user for password and the user name.

Check User Permission Access

Once the details are obtained, user name and password, this method is responsible for checking if user provided the correct details. If user has provided user name and password, then this method returns 1, else it will return 0, which will make the program to exit.

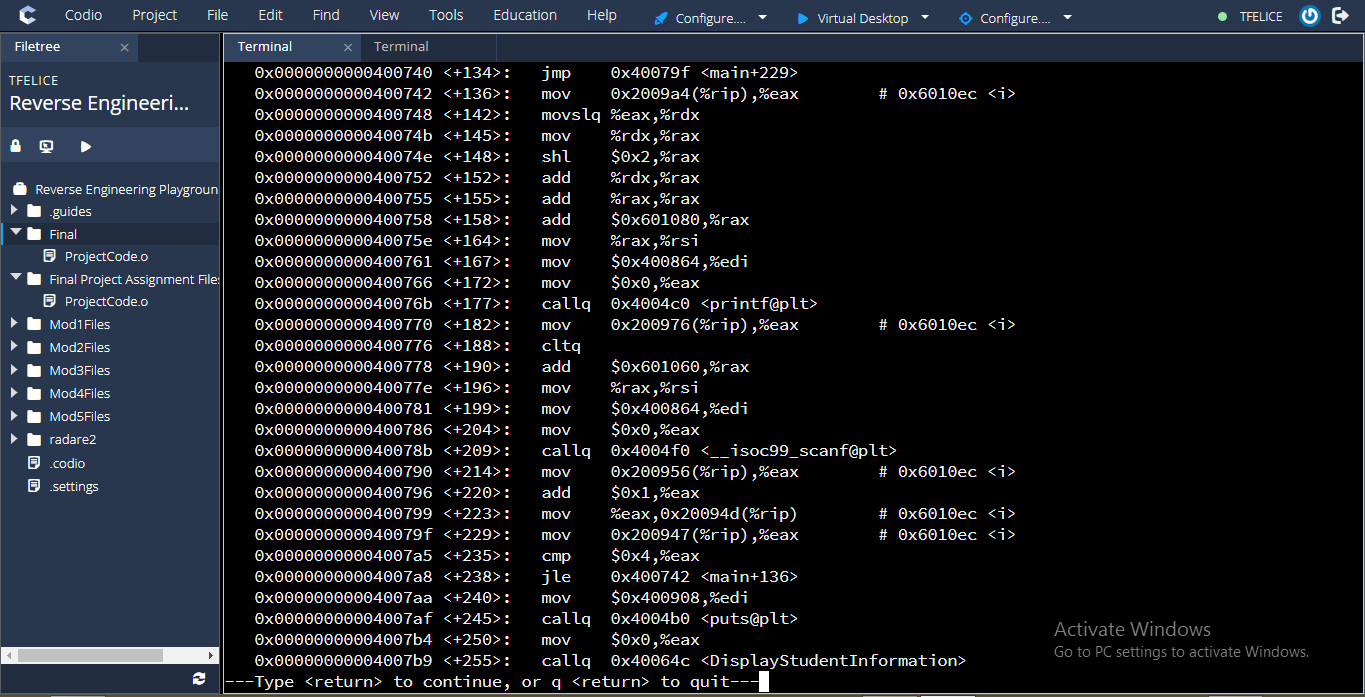
Display Student Information

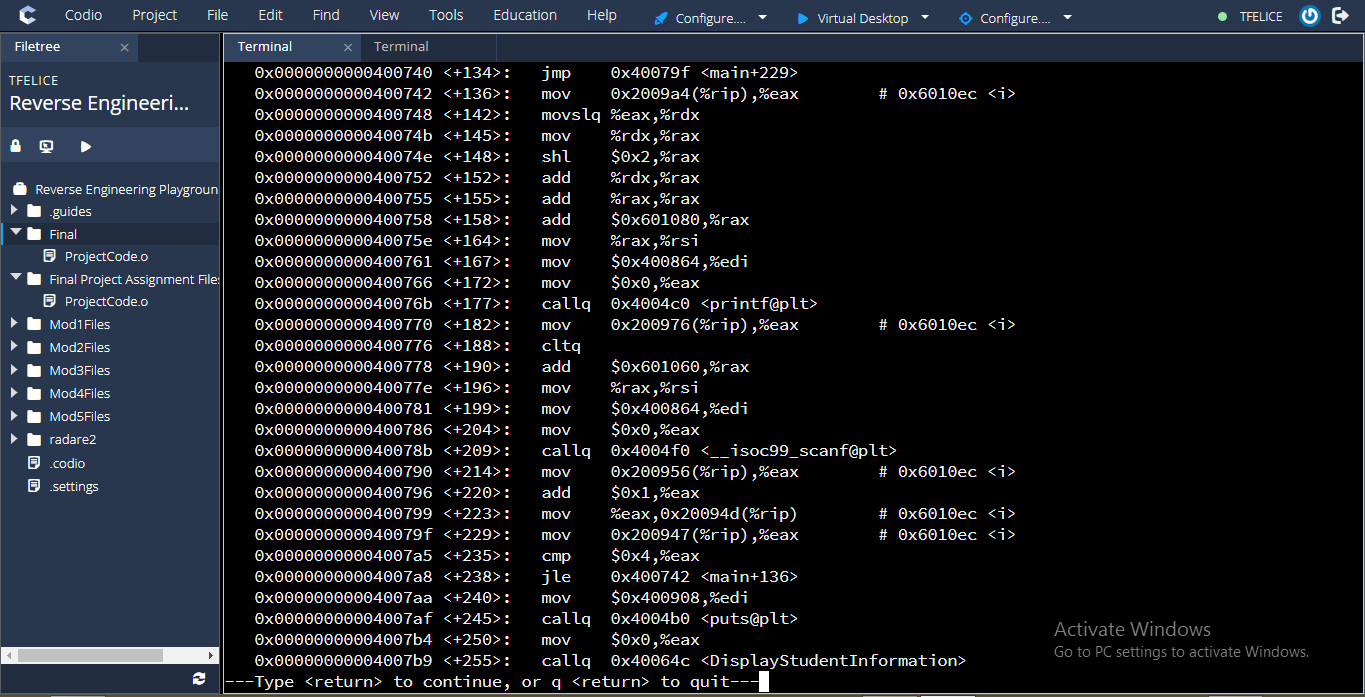
This method is used to display the student’s records. It uses a loop to go through the student array and the string holding the grades. It uses printf in each loop to display the records.

Disassembling Process.

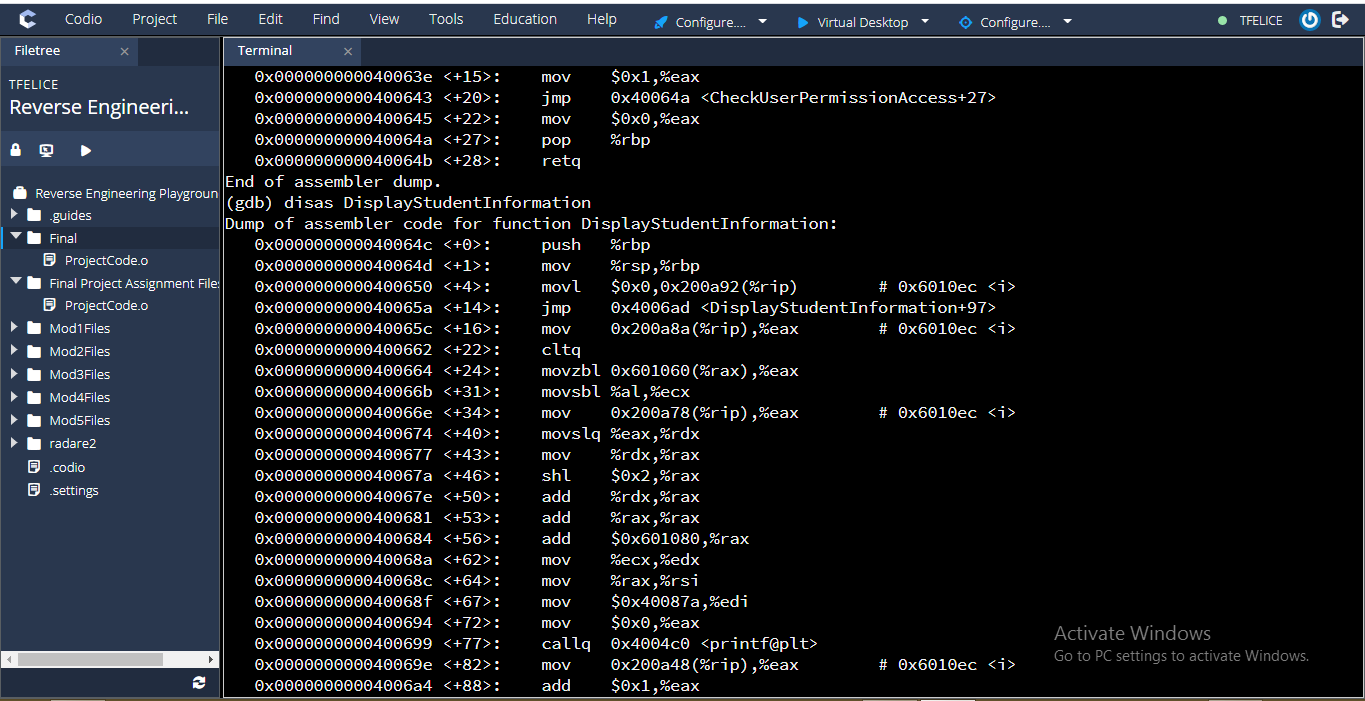
I used a default tool in Linux to check the components of the binary file provided. The gdb tool allow one to disassemble a binary file, check its method and all its code in assembly language. I used the below steps to analyze the binary file.

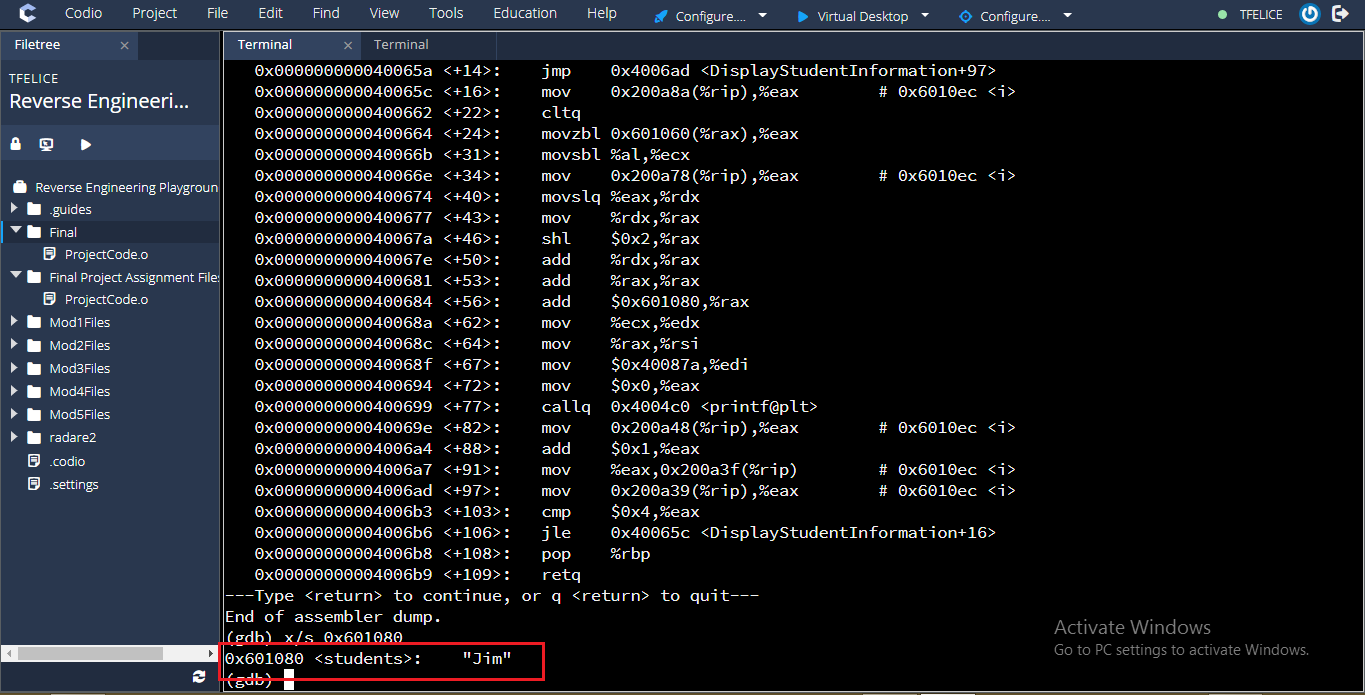
Disassembling main function



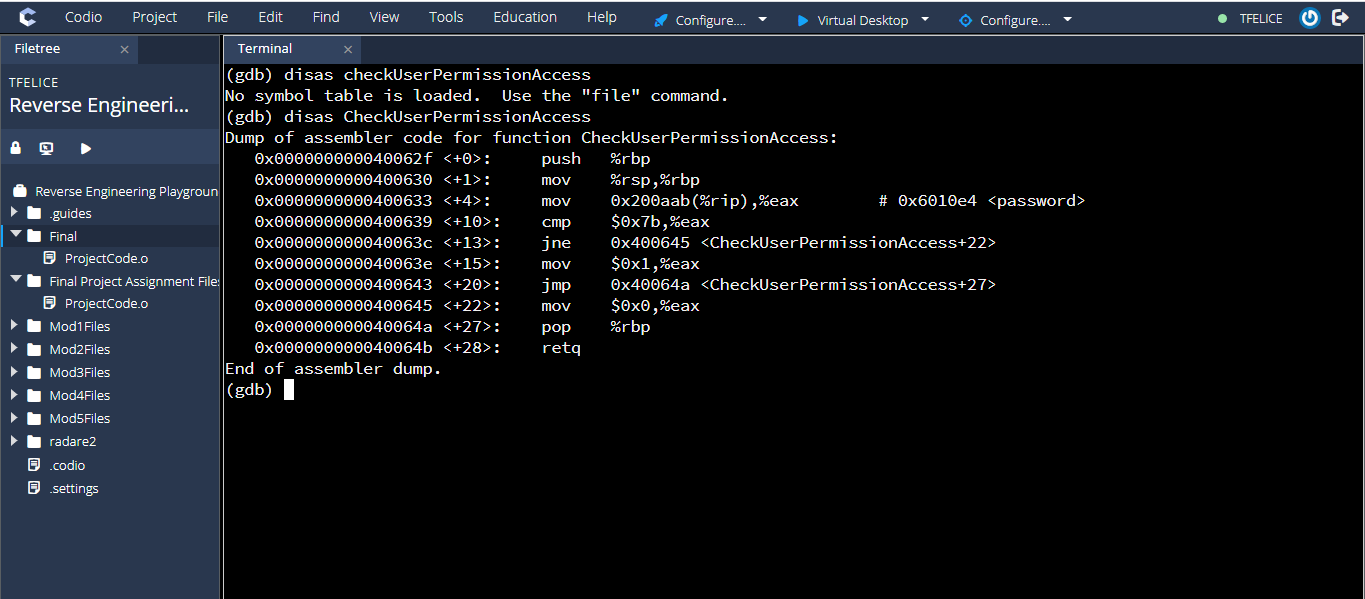


**Disassemble DisplayStudentInformation.**

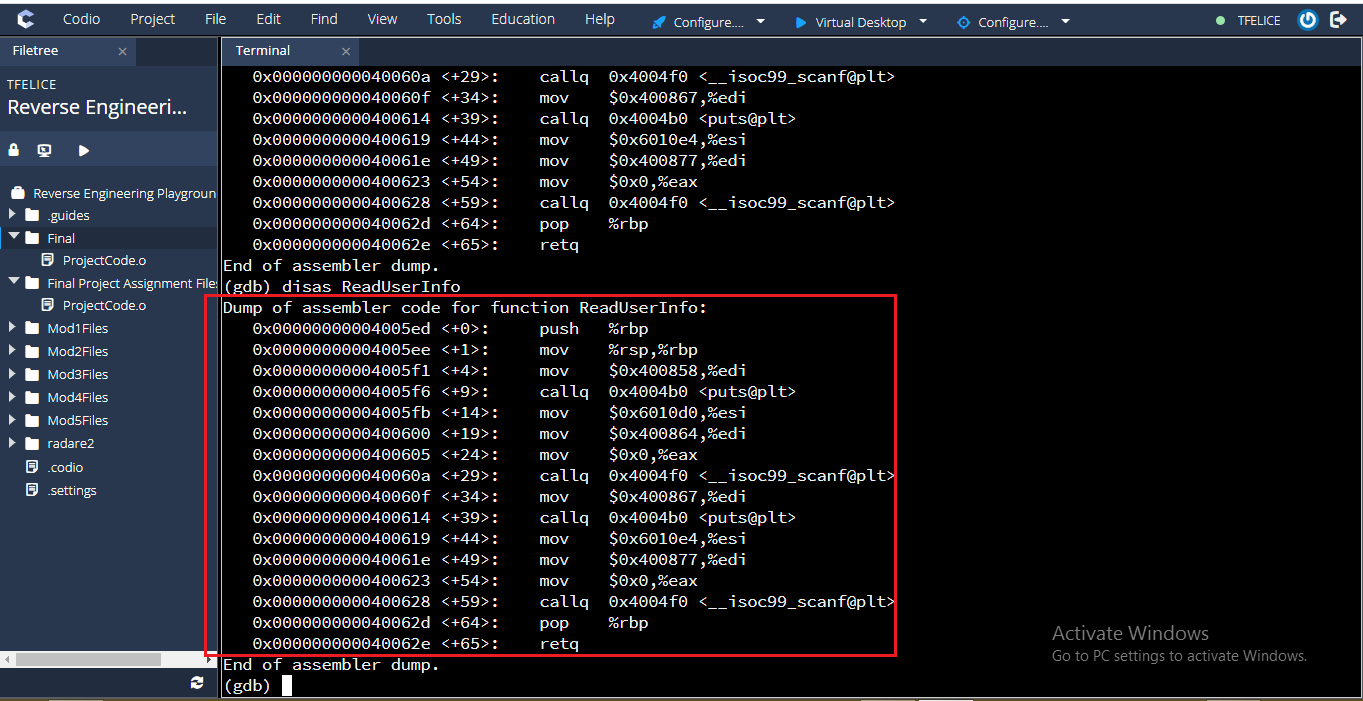




**Disassemble CheckUserPermissionAccess Function.**



**Disassemble ReadUserInfo.**



**Description of exploitable weaknesses and vulnerabilities**.

In most cases, programs stores the security details in a remote database, if not so, then the details will be stored in a separate files but encrypted, they will not be easy to decode or decrypt. However I found out, using the simple gdb tool, that all the passwords and user names are stored as plain texts inside the code. The passwords are not encrypted in any way. This makes the data to be insecure since its very easy to get the password.

Also I found out that to continue to the other processes, skipping the CheckUserPermissionAccess, one just needs to make sure that 1 is supplied or is stored in address that holds the returned value of CheckUserPermissionAccess method, this way the app will allow the execution to continue, when 0 is supplied then the program exits

Basic Recommendations to fix the Weaknesses

There should be a way to encrypt the passwords and user names in the file, to increase the security of the student data.

Also, we may use remote computers to store our security details, there is no need to put the details in the same file, when the app is running it can the access these details from server, which will be encrypted and will be sent once a correct email or user name is sent to the server.

With the current technology, we may user phones to get access to applications and data. This is more secure since the authenticated phone number will receive the password when user wishes to log in the system

When authenticating users, then the app should use both user name and password, this application uses only the password.

**Security Brief Appendix.**

Main function

|  |  |
| --- | --- |
| Assembly Code | Explanation |
| Block One | |
| push %rbp  mov %rsp,%rbp | The start of the main program |
| Block Two | |
| sub $0x10,%rsp  mov %edi,-0x4(%rbp)  mov %rsi,-0x10(%rbp)  mov $0x0,%eax | mov %edi,-0x4(%rbp) Moves value which resides in edi into -0x04(rpb)  mov %rsi,-0x10(%rbp) moves value in address rsi into ,-0x10(%rbp)  mov $0x0,%eax Moves value 0 into eax  char name[] = "";  char password[] = "";  char \*students[] = {"Jim","Tom","Ben","Alice","Ruby"};  char grades[] = "ACCDF"; |
| Block Three | |
| callq 0x4005ed <ReadUserInfo> | This line calls method ReadUserInfor.  This will allow user to enter user name and password  = ReadUserInfor() |
| Block Four | |
| mov $0x0,%eax  callq 0x40062f <CheckUserPermissionAccess> | Move 0 to address eax  This line calls CheckUserPermissionAccess Method  Int access = CheckUserPermissionAccess() |
| Block Five | |
| cmp $0x1,%eax  jne 0x4007be <main+260>  mov $0x400888,%edi  callq 0x4004b0 <puts@plt>  mov $0x0,%eax | As stated earlier, the method CheckUserPermission will return 1 for success, 0 for fail. Therefore, once the method exits, the returned vaue will be compared to 1  cmp $0x1,%eax  If they are equal the the program continues otherwise it will jump to line 260  It will the print string which is in edi, the finally move 0 into address eax  IF(access == 1){  Program continues  }else{  Program exits  } |
| Block 6 | |
| callq 0x40064c <DisplayStudentInformation> | This code calls the function DisplayStudentInformation,  this method display the student names and their grades.  = DisplayStudentInformation() |
| Block 7 | |
| mov $0x4008b8,%edi  callq 0x4004b0 <puts@plt>  mov $0x6010e8,%esi  mov $0x400864,%edi  mov $0x0,%eax | This block of code is responsible for prompting user,  The program will ask user if he/she wishes to adjusts the records,  This part prints out on the screen.  printf("Adjust grades for students?"); |
| Block 8 | |
| callq 0x4004f0 <\_\_isoc99\_scanf@plt>  movzbl 0x2009c9(%rip),%eax # 0x6010e8 <choice> | This block allow user to enter his/her choice.  .  scanf(" %c",&choice); |
| Block 9 | |
| cmp $0x59,%al  jne 0x4007be <main+260> | Then the program compare the value entered with another hardcoded value, if they match then user is allowed to adjust records else the program will jump to line 260, where it will exit.  IF(choice == value){  printf("Enter the GPA for students one at a  time \n");  }ELSE{  NOT ALLOWED  } |
| mov $0x4008d8,%edi  mov $0x0,%eax  callq 0x4004c0 <printf@plt>  movl $0x0,0x2009ac(%rip) | A value is moved to %edi  Using gdb, as I have shown in the image above, it gives the appropriate strings that will be displayed when the printf method is called.  (gdb) x/s 0x4008b8  0x4008b8: "Adjust grades for students?"  (gdb) x/s 0x6010e8  0x6010e8 <choice>: ""  (gdb) x/s 0x4008d8  0x4008d8: "Enter the GPA for students one at a time\n "  This strings are displayed to the user to allow one to change the grades of the students |
| jmp 0x40079f <main+229>  mov 0x2009a4(%rip),%eax  movslq %eax,%rdx  mov %rdx,%rax  shl $0x2,%rax  add %rdx,%rax  add %rax,%rax  add $0x601080,%rax  mov %rax,%rsi  mov $0x400864,%edi | The firs line will make the program to jump to line 229.  This block will also display the student’s records on the screen. Each student data is display in each program loop. |
| mov $0x400864,%edi  mov $0x0,%eax  callq 0x4004c0 <printf@plt>  mov 0x200976(%rip),%eax # 0x6010ec <i>  cltq  add $0x601060,%rax  mov %rax,%rsi  mov $0x400864,%edi  mov $0x0,%eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt>  mov 0x200956(%rip),%eax # 0x6010ec <i>  add $0x1,%eax  mov %eax,0x20094d(%rip) # 0x6010ec <i>  mov 0x200947(%rip),%eax # 0x6010ec <i>  cmp $0x4,%eax  jle 0x400742 <main+136>  mov $0x400908,%edi  callq 0x4004b0 <puts@plt> | This block of code is responsible for allowing a user to change the existing records.  The loop created here iterate thorough the records allowing the user to enter new records.  This bloc of code controls the loop, this will allow all the records to be edited in the string of student’s grades.  for(x = 0; i < 5; i++){  printf("%s",students[i]);  scanf(" %c",&grades[i]);  } |
| mov $0x0,%eax  callq 0x40064c <DisplayStudentInformation> | Once user enters the records for each student, then the DisplayStudentInformation method is called to display the edited records.. |
| Last Block | |
| mov $0x0,%eax  leaveq  retq | The program terminates.  return (EXIT\_SUCCESS); |

**Disassembled ReadUserInfo function**.

|  |  |
| --- | --- |
| Assembly Code | Explanation |
| push %rbp  mov %rsp,%rbp  mov $0x400858,%edi  callq 0x4004b0 <puts@plt> | Start of the function  [mov command] Moves value in $0x400858 into %edi,  The last statement prints the string in address edi  printf("Enter Name:\n"); |
| mov $0x6010d0,%esi  mov $0x400864,%edi  mov $0x0,%eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt> | This block will allow user to enter data into the application when running.  It will prompt user to enter the name and the password at the starting point of the program execution.  The last line calls scanf which allow user to enter an input.  scanf(" %s",&name); |
| mov $0x400867,%edi  callq 0x4004b0 <puts@plt>  mov $0x6010e4,%esi  mov $0x400877,%edi  mov $0x0,%eax | From my analysis in the images above, I found out that the string Enter password is in 0x400867 is enter password. This means that the program is responsible for password from user.  printf("Enter Password:\n"); |
| callq 0x4004f0 <\_\_isoc99\_scanf@plt>  pop %rbp  retq | The scanf allow user or the program to get an input from user. It is called here to get the user password  scanf(" %s",&password);  The function then exit. |

Disassembled CheckUserPermissionAccess.

|  |  |
| --- | --- |
| Assembly Code | Explanation |
| push %rbp  mov %rsp,%rbp | Beginning of the function |
| mov 0x200aab(%rip),%eax # 0x6010e4 <password>  cmp $0x7b,%eax  jne 0x400645 <CheckUserPermissionAccess+22> | mov] the value held in memory with address 0x200aab(%rip) is moved to address eax  [cmp] this is comparison command. It compares value in eax with value $0x7b,%eax  This statements says that jump if not equal, the program execution will jump to line 0x400645 in a case where value $0x7b is not equal to value in %eax |
| mov $0x1,%eax  jmp 0x40064a <CheckUserPermissionAccess+27>  mov $0x0,%eax | This line moves 1 into eax address. This happens when the comparison above is true the values are equal. This means that the method return s one when user enter the correct password, a means of showing that password is correct, otherwise the program exits by returning a zero  if(strcmp(password, pass) == 0){  return 1; }else{  return 0; } |

Disassemble DisplayStudentInformation:

|  |  |
| --- | --- |
| Assembly Code | Description |
| Line 0 to Line 14 | |
| push %rbp  mov %rsp,%rbp  movl $0x0,0x200a92(%rip) # 0x6010ec <i>  jmp 0x4006ad <DisplayStudentInformation+97> | Start of the program  Start of the program  Initialize the counter for the loop  Ask the program to jump, which makes the program to move to line 97 [from the image above]  This lines moves program execution to line 97 [Block 97 Below]  Int i = 0; |
| Line 97 to 106 | |
| mov 0x200a39(%rip),%eax # 0x6010ec <i>  cmp $0x4,%eax  jle 0x40065c <DisplayStudentInformation+16> | A value is moved into address eax.  The value is then compared by 4 [$0x4]  If the value is less than [jle] program will jump to line 16  This reveals a loop as it can be done in C as shown below  WHILE counter < 4:  JUMP TO LINE 16  ADD 1 TO COUNTER  ELSE  LOOP EXITS |
| From Line 16 to 91 | |
| mov 0x200a8a(%rip),%eax # 0x6010ec <i>  cltq  movzbl 0x601060(%rax),%eax  movsbl %al,%ecx  mov 0x200a78(%rip),%eax # 0x6010ec <i  movslq %eax,%rdx  mov %rdx,%rax  shl $0x2,%rax  add %rdx,%rax  add %rax,%rax  add $0x601080,%rax  mov %ecx,%edx  mov %rax,%rsi  mov $0x40087a,%edi  mov $0x0,%eax | Moves value in address 0x200a8a(%rip) into %eax. This represents the array of names as we have seen in the images above.  char \*students[] = {"Jim","Tom","Ben","Alice","Ruby"};  [movsbl %al,%ecx]  They are then moved into ecx  [mov 0x200a78(%rip),%eax ]  Further the grades are loaded into eax  char grades[] = "ACCDF";  From the analysis above, I found out that, during each loop, a student name is displayed. This means that, when counter is 1, the first student at index 1 will be displayed. That way all the students will be displayed |
| callq 0x4004c0 <printf@plt>  mov 0x200a48(%rip),%eax # 0x6010ec <i>  add $0x1,%eax  mov %eax,0x200a3f(%rip) # 0x6010ec <i>  mov 0x200a39(%rip),%eax # 0x6010ec <i>  cmp $0x4,%eax  jle 0x40065c <DisplayStudentInformation+16> | This blocks prints the strings for grades and the student name.  As show in the pseudocode, this block will further increment the counter with 1 as long as it’s less than 4.  [cmp $0x4,%eax]  The counter will then be compared to 4 if it’s less than 4, then the loop continues, otherwise the loops stops by jumping to line 16  for(i =0; i < 5; i++){  printf("%s %c \n",students[i],grades[i]);  } |
| pop %rbp  retq | Program exits |