**Security Brief**

**A. Overview and Body of Report**

The program I analyzed was written for teachers to manage students’ grades. The component uses hard-coded names and GPAs as a starting point. To ensure that only faculty can use components, users must enter a password before they are able to view the students’ grades or modify them. Once a user enters the correct password, permission is granted to view the grades. The user is then given a chance to modify the grades one student a time. I was able to disassemble the program and found that there are four functions within the program: main(), DisplayStudentInformation(), CheckUserPermissionAccess(), and ReadUserInfo(). The main() function is the start of the program. This is where the variables are created and initialized. The main() function also calls the other functions within the program, in order to accept user input and determine how the program will behave. The DisplayStudentInformation() function is used to store the names of each student and their grades. The values are stored in arrays and the function loops through each of them. The CheckUserPermissionAccess() is the function that validates user's credentials. ReadUserInfo() is the function responsible for prompting the user to enter their name and password and then reads the user's input.

The first step I took was to compile the program so I could see what it was supposed to do. The first screen displayed a message that prompted me to enter my name, followed by a screen prompting me to enter a password. I used the password provided by Company A++, to continue. The student's names and grades were displayed and there was a prompt to change the grades. I attempted to enter integer values containing decimals. After going through each student, the values I entered were displayed but I noticed the program rounded up these values to a whole number. Next, I used gdb and Info functions to see the functions included in the program. I was able to further disassemble the four functions previously mentioned: the main(), DisplayStudentInformation(), CheckUserPermissionAccess(), and ReadUserInfo(). I was able to see the assembly language for each function where I identified several calls such as call<puts>, call<print> , call<scan> and call<exit> to determine where the program output data and read input from the user, to convert it to C code.

Of the most notable weaknesses and vulnerabilities in the program, is the ability to gain access to the program. The program allows any user to enter a name, which means there is only one authentication process, the password. The password is hard coded into the program which is also an area of vulnerability. This is an easy target for hackers because the values stored in registers can be changed. Passwords should not be included in the program. An alternative would be to change the program to validate user names and passwords, for extra authentication. Passwords should be stored in a separate file or database and permissions should be limited. Below is a table of the assembly code in the left column with explanations for each block, in the middle. There is also corresponding C Code and areas of vulnerability in the right column.

**B. Security Brief Appendix**

|  |  |  |
| --- | --- | --- |
| **Assembly Code** | **Explanation** | **C Code** |
| main()  push %rbp  mov %rsp,%rbp | Beginning of the function that copies the  stack pointer to the base pointer |  |
| sub $0x10,%rsp  mov %edi,-0x4(%rbp)  mov %rsi,-0x10(%rbp)  mov $0x0,%eax  callq 0x4005ed <ReadUserInfo> | Creates and Initializes variables  Utilizes temporary storage  Calls the ReadUserInfo function | ReadUserInfo() |
| mov $0x0,%eax  callq 0x40062f <CheckUserPermissionAccess>  cmp $0x1,%eax  jne 0x4007be <main+260>  mov $0x400888,%edi  callq 0x4004b0 <puts@plt> | Calls the CheckUserPermission function  Compares user input to value stored  Jumps to main if condition is met(user credentials valid)  Calls the data stored once access is granted | int uservalue = CheckUserPermissionAccess()  if(access == 1) {    DisplayStudentInformation();  printf("Adjust the grades for  students?"); |
| mov $0x0,%eax  callq 0x40064c <DisplayStudentInformation>  mov $0x4008b8,%edi  callq 0x4004b0 <puts@plt> | Calls the DisplayStudentInformation function  Calls the data stored to access student info | DisplayStudentInformation() |
| mov $0x6010e8,%esi  mov $0x400864,%edi  mov $0x0,%eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt> | Moves values stored for student grades to temporary registers  Accepts input from user | char userChoice = 'Y';  char userValue = 'X';  scanf(" %c", &uservalue); |
| movzbl 0x2009c9(%rip),%eax # 0x6010e8 <choice>  cmp $0x59,%al  jne 0x4007be <main+260>  mov $0x4008d8,%edi  mov $0x0,%eax  callq 0x4004c0 <printf@plt> | Input from user to choose to edit student info or not  Compares value stored to input to determine what to do next  Jumps to main to edit  Prints the result of user input | if(userChoice == userValue) { |
| movl $0x0,0x2009ac(%rip) # 0x6010ec <i>  jmp 0x40079f <main+229>  mov 0x2009a4(%rip),%eax # 0x6010ec <i>  movslq %eax,%rdx | Students grades are stored in temporary registers  Moves stored values and user input | printf("Enter the GPA for students  one a t a time \n");  int i; |
| mov %rdx,%rax  shl $0x2,%rax  add %rdx,%rax  add %rax,%rax  add $0x601080,%rax  mov %rax,%rsi  mov $0x400864,%edi  mov $0x0,%eax  callq 0x4004c0 <printf@plt> | Values stored and user input are moved to temporary registers  Computation is performed to add new values to existing values  Prints the results of the student's new grade |  |
| mov 0x200976(%rip),%eax # 0x6010ec <i>  cltq | Loop to prompt user to enter new grades |  |
| add $0x601060,%rax  mov %rax,%rsi  mov $0x400864,%edi  mov $0x0,%eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt> | Continues the loop for the number of students  Reads input from user to change grades |  |
| 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> | Continues the loop for the number of students  Compares the number of students to the iterations  Jumps to main if nor more students  Reads input from user to change grades | for(i =0; i <5; i++) {  printf("%s", students[i]);  scanf("%c", &grades[i]); |
| mov $0x0,%eax  callq 0x40064c <DisplayStudentInformation>  mov $0x0,%eax | exits the loop  call the DisplayStudent Information function | printf("You have successfully  updated class grades. The grades  are now as follows:\n")  DisplayStudentInformation(); |
| leaveq  retq | Jumps back to the return address |  |
| DisplayStudentInformation()  push %rbp  mov %rsp,%rbp | Beginning of the function that copies the  stack pointer to the base pointer |  |
| movl $0x0,0x200a92(%rip) # 0x6010ec <i>  jmp 0x4006ad <DisplayStudentInformation+97>  mov 0x200a8a(%rip),%eax # 0x6010ec <i> | Jumps to DisplayStudentInformation |  |
| cltq  movzbl 0x601060(%rax),%eax  movsbl %al,%ecx  mov 0x200a78(%rip),%eax # 0x6010ec <i>  movslq %eax,%rdx  mov %rdx,%rax | Loops through an array of student names and grades |  |
| shl $0x2,%rax  add %rdx,%rax  add %rax,%  add $0x601080,%rax  mov %ecx,%edx  mov %rax,%rsi  mov $0x40087a,%edi  mov $0x0,%eax  callq 0x4004c0 <printf@plt> | Assigns grades to students in corresponding locations  Registers where student's names and grades are stored  Prints the values of student names and corresponding grades | int i;  for(i =0; i < 5; i ++) {  printf("%s %c \n", students[i],  grades[i]);  } |
| 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> | Counter for loop  Compares values stored  Jumps to DisplayStudentInformation if condition is met |  |
| pop %rbp  retq | Beginning of the function that copies the  stack pointer to the base pointer |  |
| CheckUserPermissionAccess()  push %rbp  mov %rsp,%rbp | Beginning of the function that copies the  stack pointer to the base pointer |  |
| mov 0x200aab(%rip),%eax # 0x6010e4 <password>  cmp $0x7b,%eax  jne 0x400645 <CheckUserPermissionAccess+22> | Takes input from user for password  Compares value stored %eax to what was entered  Jumps to verify user's credentials | char pass[] = "123";  char name[] = " "; |
| mov $0x1,%eax  jmp 0x40064a <CheckUserPermissionAccess+27>  mov $0x0,%eax | Returns the value stored in %eax to continue to program or exit if user  password is verified or permission denied | if(strcmp(password, pass) == 0) {  return 1;  } else{  return 0; |
| pop %rbp  retq | Pops the base pointer off the stack and  stores it in %rbp and jumps back to return  address |  |
| ReadUserInfo()  push %rbp  mov %rsp,%rbp | Beginning of the function that copies the  stack pointer to the base pointer |  |
| mov $0x400858,%edi  callq 0x4004b0 <puts@plt>  mov $0x6010d0,%esi  mov $0x400864,%edi  mov $0x0,%eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt> | Calls the function to display "Enter Name"  Reads input for user name | printf("Enter Name: \n");  scanf("%s, &name); |
| mov $0x400867,%edi  callq 0x4004b0 <puts@plt>  mov $0x6010e4,%esi  mov $0x400877,%edi  mov $0x0,%eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt> | Calls the function to display "Enter password"  Reads input for user password | printf("Enter Password: \n");  scanf("%s", &password); |
| pop %rbp  retq | Beginning of the function that copies the  stack pointer to the base pointer |  |