Dennis Persaud

CS-410-J5821 Software Reverse Engineering

Southern New Hampshire University

20EW5

Final Project: Security Brief

**Code Functionality**

The purpose of this code is to allow a user to login with a name and password to view and change the grades of an array of students. This program consists of four functions: main, ReadUserInfo, CheckUserPermissionAccess, and DisplayStudentInformation. The ReadUserInfo function is the first method called by the main function, it does not have any parameters. The purpose of this function is to prompt the user with a message asking them to enter their name and password. The CheckUserPermissionAcces function is the second function called by the main function. This function also does not take any parameters but returns a binary value. The purpose of this function is to compare the password entered by the user in the ReadUserInfo function to a predefined password set by the developers of the program. When comparing the password entered by the user to the preset password, if the values match the function returns a one, if they do not match the function returns a zero. The DisplayStudentInformation function is called twice in the main function. It first called to display an initial list of the students and their grades and the second time it is called is after the grades have been adjusted by the user. This function uses a loop to display the elements of the array, which includes the name and grade of each student.

**Disassembling Process**

To disassemble this program, I used GNU debugger (GDB) to identify each function and disassemble them individually. The GDB commands that I used to disassemble this program were the “info” and “disassemble” commands in conjunction with each function name. In using these two commands I was able to view all of the functions for this program. In addition to other actions, gdb allows us to run the program while setting breakpoints to examine code. Using breakpoints offers an opportunity to step into each instruction were it can be seen where the variables are stored and the memory location. The gbd commands that support the examination of these steps while the program is running are si and ni. Also, p or x/s commands show the value in the memory location. To gather more information about the details of the program I used a hex editor to view a binary dump of the program. This dump revealed strings, string modifiers and the contents of some of the variables contained within the program.

**Exploitable Weaknesses and Vulnerabilities**

One of the weaknesses in this program is that the value that is used for comparing the password is exposed in the main function. Simply using a tool like the GNU debugger will reveal that the result of the CheckUserPermissionAccess compares the user input with the value one. An attacker could theoretically implement a buffer overflow attack to insert the value one into the location in memory and gain access to the program. A second vulnerability is that the program only verifies the user’s password but not the user’s name. This vulnerability is also easily revealed by the GNU Debugger. This vulnerability can’t really be exploited by the attacker, it just makes it easier for them to get in because they only need to worry about getting the password correct.

**Basic Recommendations**

This program should be modified with the vulnerabilities and weakness outlined in this report in mind. The first modification to this program should be to verify both the user’s password and their name in the CheckUserPermissionAccess function. The second modification should be to hide the value returned by the comparison in the CheckUserPermissionAccess using a hashing algorithm.

Security Brief Appendix

|  |  |  |
| --- | --- | --- |
| **main()** | | |
| **Assembly Code** | **Explanation** | **C Code** |
| push %rbp  mov %rsp, %rbp  sub $0x10, %rsp  mov %edi, -0x4(%rbp)  mov %rsi, -0x10(%rbp) | Initializes the main function’s stack frame. | Function Prologue. |
| mov $0x0, %eax  callq 0x4005ed <ReadUserInfo> | Call the ReadUserInfo function to get the user’s login information. | ReadUserInfo(); |
| mov $0x0, %eax  callq 0x40062f <CheckUserPermissionAccess> | Call the CheckUserPermissionAccess function to check the user’s access to the program. | CheckUserPermissionAccess(); |
| cmp $0x1, %eax  jne 0x4007be <main+260> | Compare value returned by CheckUserPermissionAccess to 1, continue if equal, exit if not. | if (password == 1) |
| mov $0x400888, %edi  callq 0x4004b0 <puts@plt> | Display welcome message to the professor. | printf(“Welcome Professor. Below are all student grades.”); |
| mov $0x0, %eax  callq 0x40064c <DisplayStudentInformation> | Call the DisplayStudentInformation function to display student information. | DisplayStudentInformation(); |
| mov $0x4008b8, %edi  callq 0x4004b0 <puts@plt> | Display the string “Adjust grades for students?” to the user. | printf(“Adjust grades for students?”); |
| mov $0x6010e8, %esi  mov $0x400864, %edi  mov $0x0, %eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt> | Store the user’s input in the variable “choice”. | scanf(“%s”, choice); |
| movzbl 0x2009c9(%rip), %eax # 0x6010e8 <choice>  cmp $0x59, %al  jne 0x4007be <main+260> | If the choice entered by the user is equal to the value continue to the next line, if not exit the program. | if(choice == value) |
| mov $0x4008d8, %edi  mov $0x0, %eax  callq 0x4004c0 <printf@plt> | Display string asking user to enter the GPA for each student one at a time. | printf(“Enter the GPA for students one at a time ”); |
| movl $0x0,0x2009ac(%rip) # 0x6010ec <i>  jmp 0x40079f <main+229>  mov 0x2009a4(%rip), %eax # 0x6010ec <i>  movslq %eax, %rdx  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>  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> | Loop through each of the students stored in the array. For each loop print the student’s name and get the new GPA from the user’s input. This loop will run 5 times, since there are only 5 students in the array. | for(int i=0; i<=4; i++){  printf(“%s”, students[i]);  scanf(“%s”, &grades[i]);  } |
| mov $0x400908, %edi  callq 0x4004b0 <puts@plt> | Print string to the user confirming that the grades have been successfully updated. | printf(“You have successfully updated class grades. The grades are now as follows: ”); |
| mov $0x0, %eax  callq 0x40064c <DisplayStudentInformation> | Call the DisplayStudentInformation function to display newly updated grades. | DisplayStudentInformation(); |
| mov $0x0, %eax  leaveq  retq | Exit the main function. | return 0; |

|  |  |  |
| --- | --- | --- |
| **ReadUserInfo()** | | |
| **Assembly Code** | **Explanation** | **C Code** |
| push %rbp  mov %rsp, %rbp | Initializes the ReadUserInfo function’s stack frame. | Function Prologue. |
| mov $0x400858, %edi  callq 0x4004b0 <puts@plt> | Display message to the user asking them to enter their name. | printf(“Enter name: ”); |
| mov $0x400864, %edi  mov $0x0, %eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt> | Get the user input and store it in the variable “name”. | scanf(“%s”, name); |
| mov $0x400867, %edi  callq 0x4004b0 <puts@plt> | Display message to the user asking them to enter their password. | printf(“Enter password: ”); |
| mov $0x6010e4, %esi  mov $0x400877, %edi  mov $0x0, %eax  callq 0x4004f0 <\_\_isoc99\_scanf@plt> | Get the user input and store it in the variable “Password”. | scanf(“%d”, password); |
| pop %rbp  retq | Pop base pointer off stack frame and return to the main function. | return; |

|  |  |  |
| --- | --- | --- |
| **CheckUserPermissionAccess()** | | |
| **Assembly Code** | **Explanation** | **C Code** |
| push %rbp  mov %rsp, %rbp | Initializes the CheckUserPermissionAccess function’s stack frame. | Function prologue. |
| mov 0x200aab(%rip), %eax # 0x6010e4 <password>  cmp $0x7b, %eax  jne 0x400645 <CheckUserPermissionAccess+22>  mov $0x1, %eax  jmp 0x40064a <CheckUserPermissionAccess+27>  mov $0x0, %eax | Compare the password entered by the user in the ReadUserInfo function to the value “123” which is the preset password set by Company A++. If the password is equal to “123” set the password variable to “1”, if not set the password variable to “0”. | if (password == “123”) {  password = 1;  } else {  password = 0;  } |
| pop %rbp  retq | Pop base pointer off stack frame and return to the main function. | return; |

|  |  |  |
| --- | --- | --- |
| **DisplayStudentInformation()** | | |
| **Assembly Code** | **Explanation** | **C Code** |
| push %rbp  mov %rsp, %rbp | Initializes the DisplayStudentInformation function’s stack frame. | Function prologue. |
| movl $0x0, 0x200a92(%rip) # 0x6010ec <i>  jmp 0x4006ad <DisplayStudentInformation+97>  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  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> | Loop through each of the students stored in the “students” array and print the name and grade for each student in the array. | for(i=0; i<=4; i++){  printf(“%s %c \n), students[i], grades[i];  } |
| pop %rbp  retq | Pop base pointer off stack frame and return to the main function. | return; |