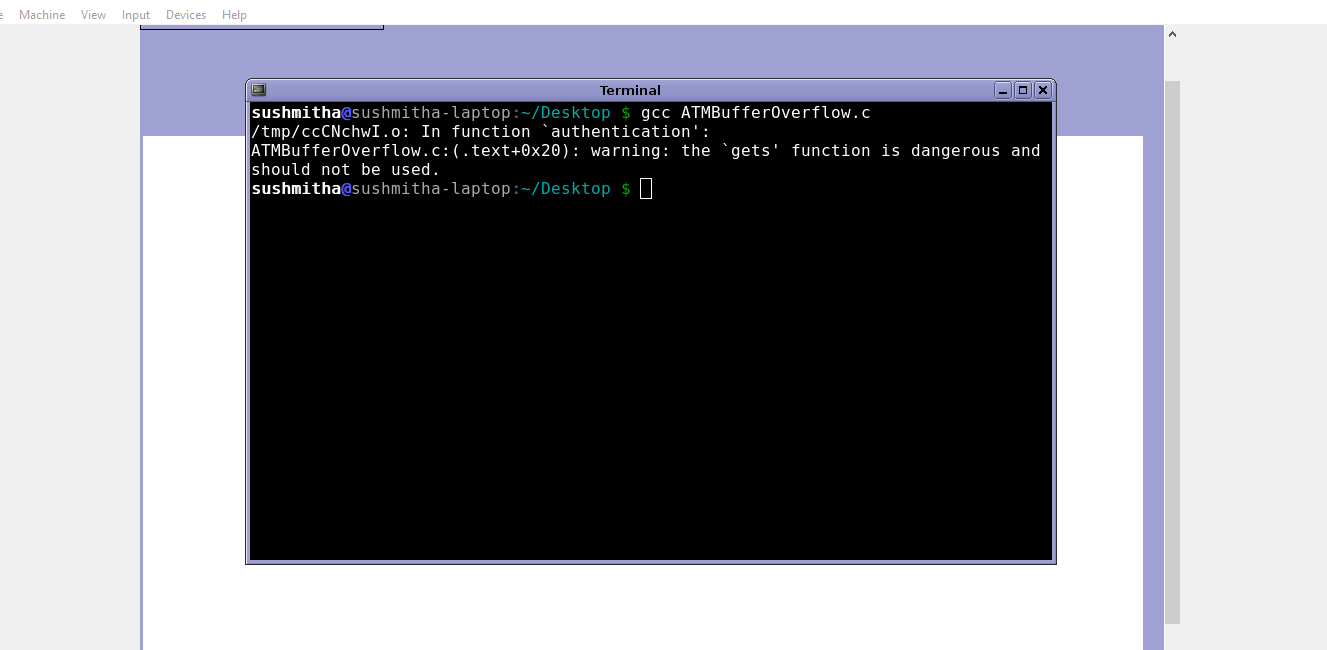
**Read Me:**

* The code is designed for ATM in C language.
* This program accepts user input for Pin authenticates the input and gives three options as ATM transactions.
* We can select any transaction type and perform the required operation.
* The inputs from the user are accepted in command line.

**Program Compilation: -**

Gcc ATMBufferOverflow.c -o ATM

Default correct PIN: - 1234



**Program Flow: -**

* Get the Pin from the User
* Other inputs are not accepted by the program.
* If the password is correct, the ATM gives three options
* 1.Deposit
* 2.Withdraw
* 3.CheckBalance
* 4.Quit
* If the user selects 1, it would ask the amount to be deposited in the bank, the user can enter the amount and it gets added to the balance.
* If the user selects 2, it would ask the amount to be withdrawn from the bank, the user can enter the amount and it would be deducted from the balance.
* If the user selects 3, it would show the available balance in the screen.
* If the user selects 4, it would quit the transactions.

**EXPLOITATION: -**

This program has a basic authentication mechanism to segregate type of users, if the user enters the right password “1234” the person will be authenticated if not the person will have wrong password and cannot have access to the transaction types.

The password has a variable buffer value [5] if the user value exceeds the buffer limit, the program would be exploited to vulnerable attacks. Since the user enters password above buffer values, there is a possibility to stack more values or data in the stack and it would get overwritten in the stack frame thereby allowing attackers to exploit if they enter more user input more than 35.

Eg :- Enter a pin:

1234

Correct Password

Root Access given to the user

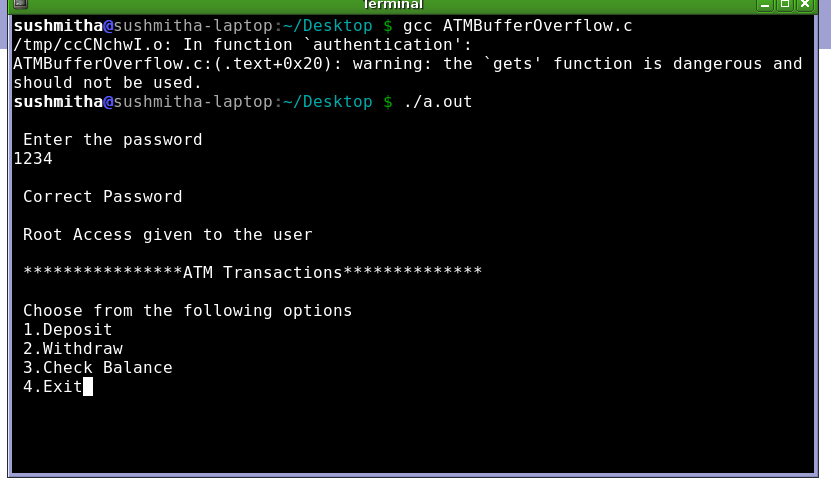
Choose from the options below

1.Deposit

2.Withdraw

3.Check Balance

4.Exit



If the scenario differs

Enter a pin: -

1111111111111111111111111111111111111111

Wrong Password

Root Access given to the user

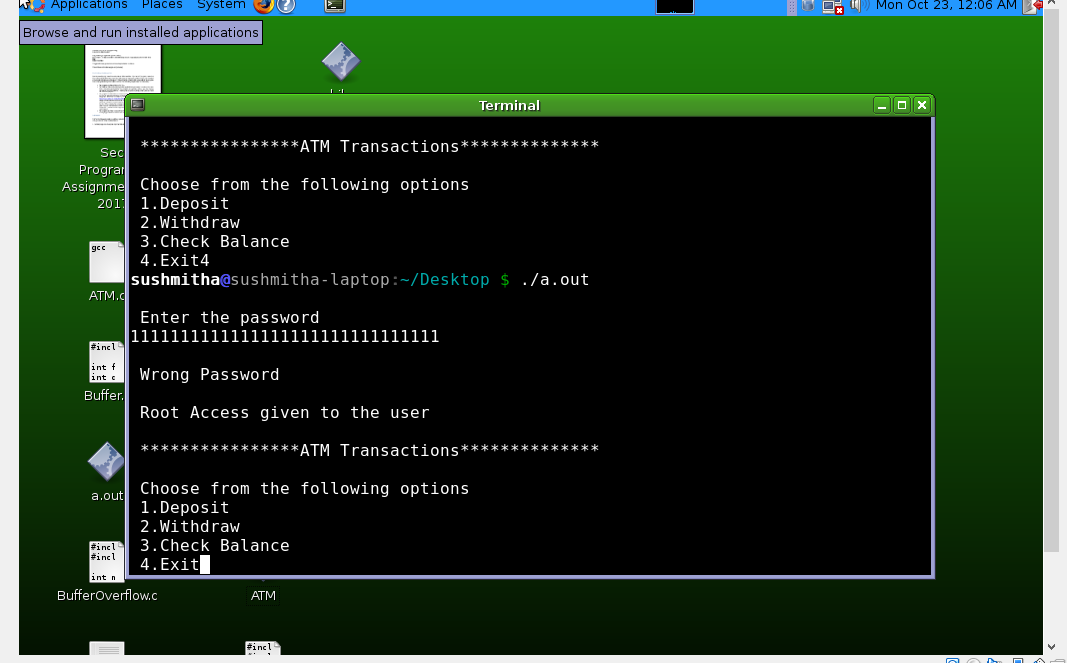
Choose from the options below

1.Deposit

2.Withdraw

3.Check Balance

4.Exit

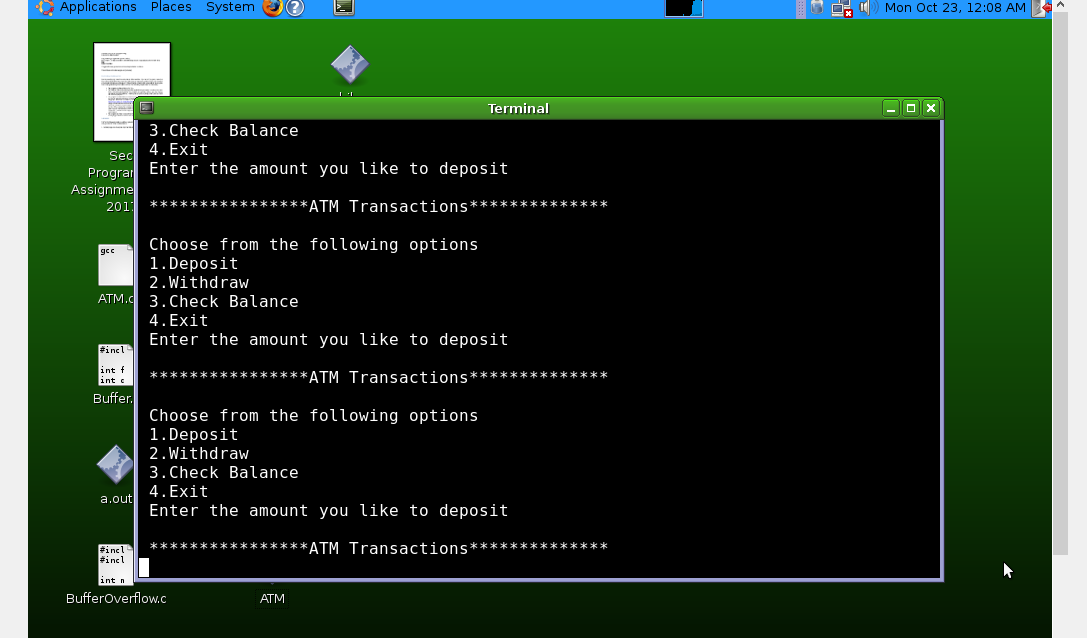


2. If the user enters the transaction input as string instead of integers it would crash the system by buffer overflow.

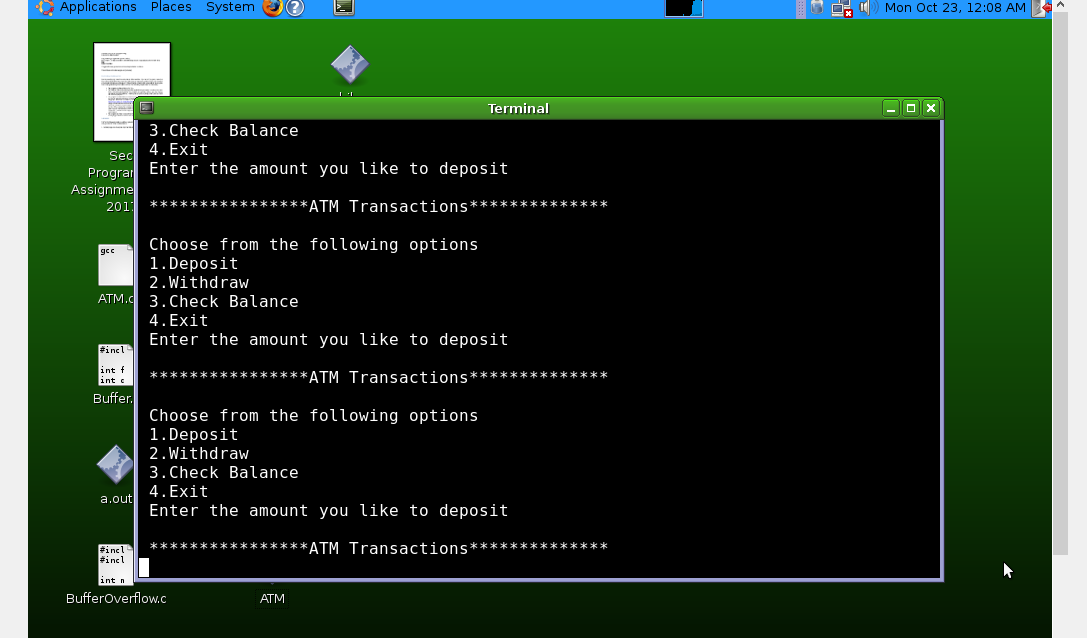
When user takes input from user for deposit amount, if we print alphabets in command line the system would crash due to buffer overflow on scanf statements.

|  |  |
| --- | --- |
|  | The problems with scanf are (at a minimum):   * using %d to get a string from the user, which leads to the possibility that the string may be longer than your buffer, causing overflow. * the possibility of a failed scan leaving your file pointer in an indeterminate location.   Eg: - scanf(“%d” , &depositAmount)  This can be exploited by providing other characters than the buffer length. |

The terminal keeps printing the choose from the below options on and on.



3. When a user enters string input instead of expected integer input in command line for Withdraw transactions, check balance it would crash the system due to buffer overflow in scanf statements.



**REFERENCES:**

<https://www.owasp.org/index.php/Buffer_overflow_attack>

<https://stackoverflow.com/questions/167165/what-c-c-functions-are-most-often-used-incorrectly-and-can-lead-to-buffer-over>

<https://dhavalkapil.com/blogs/Buffer-Overflow-Exploit/>

<https://www.exploit-db.com/docs/28475.pdf>

<https://stackoverflow.com/questions/44469372/exploiting-buffer-overflow-using-gets-in-a-simple-c-program>

Hacking - The art of exploitation (Jon Erickson) – Chapter 3, pg.121 -124.