**PES UNIVERSITY**

Electronic City Campus,1 KM before Electronic City, Hosur Road,

Bangalore-100



**PROJECT REPORT**

**on**

**“**Stock purchasing application: Stoxx**”**

Submitted in partial fulfillment of the requirements for the IV Semester

Secure programming with C (UE19CS257C)

**Bachelor of Engineering**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**For the Academic year**

**2020-2021**

**BY**

**Student Name Achyut Jagini SRN PES2UG19CS013**

**Student Name Devprakash Bisoi SRN PES2UG19CS110**

**Under the Guidance of**

**Vishwachetan D**

**Assistant Professor**

**Department of Computer Science and Engineering**

**PES UNIVERSITY EC CAMPUS**

**Hosur Road, Bengaluru -560100**

**PES UNIVERSITY EC CAMPUS**

**Hosur Road, Bangalore -560100**

**Department of Computer Science and Engineering**



**CERTIFICATE**

Certified that the project work entitled **“**Stoxx**”** is a bonafide work carried out by **Devprakash Bisoi** bearing USN:PES2201900475 and **Achyut Jagini** bearing USN:PES2201900of **PES University EC CAMPUS in** partial fulfillment for the special topic course **Secure Programming with C** of IV Semester in **Computer Science and Engineering** of the **Pes University**, **Bangalore** during the year 2020-2021.

***Signatures:***

|  |  |
| --- | --- |
| Project Guide:  **Vishwachetan D**  **Assistant Professor, Dept. of CSE,**  **PES UNIVERSITY EC CAMPUS, Bengaluru** | **Dr. Sandesh B J**  **Head, Dept of CSE**  **PES UNIVERSITY EC CAMPUS,Bengaluru** |

**Declaration**

I hereby declare that the project entitled **“**Stock purchasing application: Stoxx**”** submitted for the special topic course **Secure Programming with C** of IV Semester in **Computer Science and Engineering** of the **Pes University**, **Bangalore**, Bangalore is my original work.

**Signature of the Student: Devprakash Bisoi**

**Place: Bengaluru**

**Date: 11/04/21**

**Signature of the Student: Achyut Jagini**

**Place: Bengaluru**

**Date: 11/04/21**

Stock purchasing application: Stoxx

Authors

1. **Devprakash Bisoi** bearing USN: PES2201900475 of **PES University EC CAMPUS**

2. **Achyut Jagini** bearing USN:PES2201900086of **PES University EC CAMPUS**

**Table of Content**

1.Problem definition 7

2.Requirement Specification 8

3.Compilation and Execution 9

4.Security Requirement Specification 11

5.Safety recommendations followed 15

6.Data flow diagram 19

7.implementations 20

8.Testing 32

9.Static analysis report 37

10.Maintenance 39

11. Conclusion 42

12. References 43

**Problem definition**

To create a stock purchasing application that will allow the user to create his/her account to buy shares of stock.

About the project: The application allows new users to register by creating an account. After registering the details of user are stored in a CSV file. Next the user should enter his account balance. Finally the user can purchase stocks of his choice and the cost is reduced from his account balance.

Also the user can see his/her portfolio where all the quantity of the stocks purchased can be seen and the account balance.

**Requirement Specification**

1: C Programming Language

The project uses the C11 programming standard.

2.Compiler

The Project is run using gcc compiler.

3.Static analysis tool

Splint and flawfinder were the static analysis tools used in coding the project.

**Compilation and Execution**

To compile and run the files

gcc header.h main.c vulner.c

secure file

gcc header.h main.c secure.c

Next, execute the outfile using ./a.out

A picture containing text, indoor

Description automatically generated

A database file is created upon first run when first user creates an account and all user credentials are stored in this file.

Text

Description automatically generated

**Security Requirement Specification**

1.Data types used

* Structure pointer-Used to create the account.It is dynamically allocated memory on execution.
* File pointer-Used for file handling operations.
* Int
* Boolean
* Char
* Pointer

2.Functions used

Built in functions

* malloc
* free
* fopen
* fwrite
* fclose
* strcmp
* printf
* scanf

User defined functions

* ACCOUNT\* init(int\*);
* void disp\_acc(ACCOUNT\*);
* void disp(ACCOUNT\*);
* void login();
* void reg\_acc();
* void activity();
* void update();

Header files used

* #include <stdio.h>
* #include <stdlib.h>
* #include <string.h>
* #include <stdbool.h>
* #include <limits.h>

**Safety recommendations followed**

1. MEM30-C. Do not access freed memory

 2.MEM31-C. Free dynamically allocated memory when no longer needed

After memory of accs is freed it is not accessed again.

Graphical user interface, application

Description automatically generated

3. MEM36-C. Do not modify the alignment of objects by calling realloc()

4.STR30-C. Do not attempt to modify string literals

String literals are not modified in the application

5.STR32-C. Do not pass a non-null-terminated character sequence to a library function that expects a string

Graphical user interface, text, application, email

Description automatically generated

6.FLP30-C. Do not use floating-point variables as loop counters

floating-point variables are not used as loop counters in the application.

7.FIO42-C. Close files when they are no longer needed

8.FIO46-C. Do not access a closed file

Graphical user interface, text, application

Description automatically generatedFile db.csv closed after it is no longer needed.It is not accessed after closing.

**Data flow diagram**

Allocate memory for user’s account

Allow user to register or login

register login

Check if details are there in database.It is there only for valid user.

Store details in database file .Also store current balance.

Ask user which operation to perform.Check balance or buy stock.

Check balance Buy stock

Display balance

subtract cost from current balance

**implementations**

header.h

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

#include <limits.h>

struct acc

{

char usrn[20],pwd[20];

int bal,stock,ts,ap,k,wd,nf;

};

typedef struct acc ACCOUNT;

int main();

ACCOUNT\* init(int\*);

void disp\_acc(ACCOUNT\*);

void disp(ACCOUNT\*);

void login();

void reg\_acc();

void activity();

void update();

int m;

bool flag;

ACCOUNT\* accs;

ACCOUNT\* curr;

vulner.c

#include "header.h"

int i;

ACCOUNT\* init(int \*position)

{

\*position = 0;

FILE\* f = fopen("db.csv","r");

if (f==NULL)

return NULL;

else

{

ACCOUNT t;

while (fread(&t,sizeof(ACCOUNT),1,f))

{

accs[\*position] = t;

(\*position)++;

}

fclose(f);

if(\*position==0)

return NULL;

else

return accs;

}

}

void reg\_acc()

{

printf("\nEnter username:");

scanf("%s",accs[m].usrn);

printf("Enter password: ");

scanf("%s",accs[m].pwd);

printf("Enter balance(only usd accepted): ");

scanf("%d",&accs[m].bal);

accs[m].ts = 0;

accs[m].ap = 0;

accs[m].k = 0;

accs[m].wd = 0;

accs[m].nf = 0;

curr = &accs[m];

FILE\* f = fopen("db.csv","a");

fwrite(&accs[m],sizeof(ACCOUNT),1,f);

fclose(f);

m++;

}

char user[20];

void login()

{

char password[20];

printf("\nEnter username: ");

scanf("%s",user);

printf("Enter password: ");

scanf("%s",password);

bool found = false;

for(i=0;i<m;i++)

{

if(!strcmp(accs[i].usrn,user) && !strcmp(accs[i].pwd,password))

{

curr = &accs[i];

found = true;

break;

}

}

if(!found)

{

printf("\nDetails are invalid.Try Again.\n");

free(accs);

main();

}

else

printf("\nHello %s!What would you like to do?\n",user);

}

void activity()

{

int c,s,tot,choice;

char ch;

while(true)

{

printf("1.Check account balance\n2.Buy Stocks\n3.Check my portfolio\n4.Exit\nEnter your choice: ");

scanf("%d",&c);

printf("\n");

if (c==1)

printf("%s Your current balance is $%d\n",user,curr->bal);

else if (c==2)

{

printf("Here are your options:\n\n");

printf("1. Tesla(TSLA) for $840\n");

printf("2. Apple(AAPL) $134\n");

printf("3. The Coca-Cola Company(KO) $50\n");

printf("4. The Walt Disney Company(DIS) $171\n");

printf("5. Netflix(NFLX) $539\n");

printf("What would you like to buy(If you would like to buy apple type 2):");

scanf("%d",&choice);

printf("How many stocks do you want to buy? ");

scanf("%d",&s);

if(s<=0)

{

printf("Invalid quantity\n");

break;

}

if(choice ==1)

{

tot = s\*840;

curr->ts =+s;

curr->bal-=tot;

}

else if(choice ==2)

{

tot = s\*134;

curr->ap =+s;

curr->bal-=tot;

}

else if(choice ==3)

{

tot = s\*50;

curr->k = +s;

curr->bal-=tot;

}

else if(choice ==4)

{

tot = s\*171;

curr->wd =+s;

curr->bal-=tot;

}

else if(choice ==5)

{

tot = s\*539;

curr->nf =+s;

curr->bal-=tot;

}

else

{

printf("Invalid choice\n");

}

printf("Total cost: $%d\n",tot);

printf("Your transaction is successful!\n");

update();

}

else if(c==3)

{

printf("Here is your portfolio:\n");

printf("Tesla(TSLA): %d\n",curr->ts);

printf("Apple(AAPL): %d\n",curr->ap);

printf("The Coca-Cola Company(KO): %d\n",curr->k);

printf("The Walt Disney Company(DIS): %d\n",curr->wd);

printf("Netflix(NFLX): %d\n",curr->nf);

}

else if (c==4)

{

free(accs);

printf("Thank you %s!\n",user);

exit(0);

}

else

printf("Try again.\n");

printf("\n");

}

}

void update()

{

FILE\* f = fopen("db.csv","w");

for(i=0;i<m;i++)

fwrite(&accs[i],sizeof(ACCOUNT),1,f);

fclose(f);

}

Main.c

#include "header.h"

int main()

{

accs = (ACCOUNT\*)malloc(sizeof(ACCOUNT)\*10000);

curr = NULL;

init(&m);

flag = false;

if (m!=0)

flag = true;

printf("1.Create a new account\n2.Login\n3.Exit\nEnter your choice:");

int c;

scanf("%d",&c);

if(c==1)

{

reg\_acc();

printf("\nYou have registered successfully!\n\n");

activity();

}

else if (c==2 && flag)

{

login();

activity();

}

else if (c==3)

{

printf("Thank you!\n");

exit(0);

}

else

{

printf("Invalid choice. Try Again!\n");

free(accs);

main();

}

}

secure.c

#include "header.h"

ACCOUNT\* init(int \*position)

{

\*position = 0;

FILE\* f = fopen("db.csv","r");

if (f==NULL)

return NULL;

else

{

ACCOUNT t;

while (fread(&t,sizeof(ACCOUNT),1,f))

{

accs[\*position] = t;

(\*position)++;

}

fclose(f);

if(\*position==0)

return NULL;

else

return accs;

}

}

void display\_account(ACCOUNT\* account)

{

printf("\nUsername: %s\n",account->usrn);

printf("Password: %s\n",account->pwd);

printf("Balance: %d\n",account->bal);

printf("Stocks: %d\n",account->stock);

}

void display(ACCOUNT\* account)

{

for(int i=0;i<m;i++)

{

printf("\nUsername: %s\n",account[i].usrn);

printf("Password: %s\n",account[i].pwd);

printf("Balance: %d\n",account[i].bal);

printf("Stocks: %d\n\n",account[i].stock);

}

}

void reg\_acc()

{

printf("\nEnter username: ");

scanf("%s",accs[m].usrn);

if(strlen(accs[m].usrn)>=20)

{

printf("Length of username must be less than 20 characters.\n");

main();

}

printf("Enter password: ");

scanf("%s",accs[m].pwd);

printf("Enter balance: ");

scanf("%d",&accs[m].bal);

accs[m].stock = 0;

curr = &accs[m];

FILE\* f=fopen("db.csv","a");

fwrite(&accs[m],sizeof(ACCOUNT),1,f);

fclose(f);

m++;

}

char user[20];

void login()

{

char password[20];

printf("\nEnter username: ");

scanf("%s",user);

printf("Enter password: ");

scanf("%s",password);

bool found = false;

for(int i=0;i<m;i++)

{

if(!strcmp(accs[i].usrn,user) && !strcmp(accs[i].pwd,password))

{

curr = &accs[i];

found = true;

break;

}

}

if(!found)

{

printf("\nDetails are invalid.Try Again!\n\n");

free(accs);

main();

}

else

printf("\nYou have logged in!\n\n");

}

void activity()

{

int c,s,tot,choice;

char ch;

while(true)

{

printf("1.Check account balance\n2.Buy Stocks\n3.Check my portfolio\n4.Exit\nEnter your choice: ");

scanf("%d",&c);

if(c>4)

{

printf("Invalid choice\n");

activity();

}

printf("\n");

if (c==1)

printf("%s Your current balance is $%d\n",user,curr->bal);

else if (c==2)

{

printf("Here are your options:\n\n");

printf("1. Tesla(TSLA) for $840\n");

printf("2. Apple(AAPL) $134\n");

printf("3. The Coca-Cola Company(KO) $50\n");

printf("4. The Walt Disney Company(DIS) $171\n");

printf("5. Netflix(NFLX) $539\n");

printf("What would you like to buy(If you would like to buy apple type 2):");

scanf("%d",&choice);

printf("How many stocks do you want to buy? ");

scanf("%d",&s);

if(s<=0)

{

printf("Invalid quantity\n");

break;

}

if(s>INT\_MAX || tot<0)

{

printf("Invalid.\n");

continue;

}

if(tot>curr->bal)

{

printf("Your balance is low.\n");

continue;

}

if(choice ==1)

{

tot = s\*840;

curr->ts =+s;

curr->bal-=tot;

}

else if(choice ==2)

{

tot = s\*134;

curr->ap =+s;

curr->bal-=tot;

}

else if(choice ==3)

{

tot = s\*50;

curr->k = +s;

curr->bal-=tot;

}

else if(choice ==4)

{

tot = s\*171;

curr->wd =+s;

curr->bal-=tot;

}

else if(choice ==5)

{

tot = s\*539;

curr->nf =+s;

curr->bal-=tot;

}

else

{

printf("Invalid choice\n");

}

printf("Total cost: $%d\n",tot);

printf("Your transaction is successful!\n");

update();

}

else if(c==3)

{

printf("Here is your portfolio:\n");

printf("Tesla(TSLA): %d\n",curr->ts);

printf("Apple(AAPL): %d\n",curr->ap);

printf("The Coca-Cola Company(KO): %d\n",curr->k);

printf("The Walt Disney Company(DIS): %d\n",curr->wd);

printf("Netflix(NFLX): %d\n",curr->nf);

}

else if (c==4)

{

free(accs);

printf("Thank you %s!\n",user);

exit(0);

}

else

printf("Try again.\n");

printf("\n");

}

}

void update()

{

FILE\* f = fopen("db.csv","w");

for(int i=0;i<m;i++)

fwrite(&accs[i],sizeof(ACCOUNT),1,f);

fclose(f);

}

**Testing**

Vulnerable test cases

1.If total cost of stocks more than remaining balance it results in balance to become negative.

A picture containing text, indoor

Description automatically generated

2.If the balance entered is larger than value

of INT\_Max the balance becomes negative value of -1.

Graphical user interface, text

Description automatically generated

3.if username entered more than 20 characters.

Then the user cannot login again.

A screenshot of a computer

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with low confidence

**Non-vulnerable test cases**

1.If total cost of stocks less than balance transaction is successful.

Graphical user interface, text, chat or text message

Description automatically generated

2.Balance entered should be less than value of INT\_MAX.Then balance won’t become $-1.Text

Description automatically generated

3.Username must be less than 20 characters.

Text

Description automatically generated

**Static analysis report**

1.Using splint

Text

Description automatically generated

2.Using flawfinder

Text

Description automatically generated

Text

Description automatically generated

**Maintenance**

**Modification**

The design philosophy used is that no two cases of the application are dependent on each other, this independency makes it easy for the modification of the code in the future. We have made options for reusability like in the case of ADTs and password processing procedures, but this shouldn’t stop us from wanting to modify our code, the connection of any procedure with these common platforms is highly transparent and it shouldn’t be a matter of concern for developers in modifying the code.

We have made use of defined constants declared in the beginning of the code, so if you have to change the value of a constant you just have to change the value of the constant in the beginning and this is so much easier when compared to changing the value at every occurrence of the constant value.

**Creating Multiple Versions**

The architecture of the code favours creation of multiple versions.

The architecture supports modular programming, the problem can be divided into smaller parts and various developers can develop their version, we will then have multiple versions and all these versions can be integrated in the end to get the final software.

If there are multiple uses of recommendations, multiple versions can be used to show the various recommendations.

If a developer wishes to create a new version, then he can look at the explanation we’ve given with every recommendation used and the developer can make use of the recommendations with a lot more ease.

### **Scalability**

Scalability is the property of a system to handle a growing amount of work by adding resources to the system. Because of the architecture we’ve developed, the levels of abstraction we’ve given scalability is easy.

**Conclusion**

A stock purchasing application was developed successfully using secure programming.

But why did we use CERT recommendations in our project? To prevent hackers from hacking our software. How will they hack it? They would give vulnerable inputs. How did we get to know to avoid this? This is where CERT recommendations come in, they have figured out such bugs and have given solutions to them, we just had to adapt their solutions to our code. There are errors other than syntactical, memory allocation, run-time and logical errors. These errors are not because of a mistake by us, but we are still open for vulnerabilities.

Despite writing correct code logically and syntactically we have to protect ourselves from vulnerabilities, this is the purpose of implementing CERT recommendations in our project.

**References**

1) Secure Coding in C and C++, Second Edition, Robert C. Seacord

2) SEI CERT, C Coding Standard, Rules for Developing Safe, Reliable, and Secure Systems, 2016 Edition, Carnegie Mellon University

3) CERT C Programming Language Secure Coding Standard, Document No. N1255

4) SEI CERT C Coding Standard, wiki.sei.cmu.edu, Official website of Carnegie Mellon University