Program name: Password generator

individual project

Module Name: ST4061CEM

Programming and Algorithms 1

BSC. (Hons) Ethical Hacking and Cyber Security

Softwarica College of IT and E-commerce, Coventry University

Submitted by:

Diamond Gurung

Submitted to:

Module leader

Suman Shrestha

Abstract

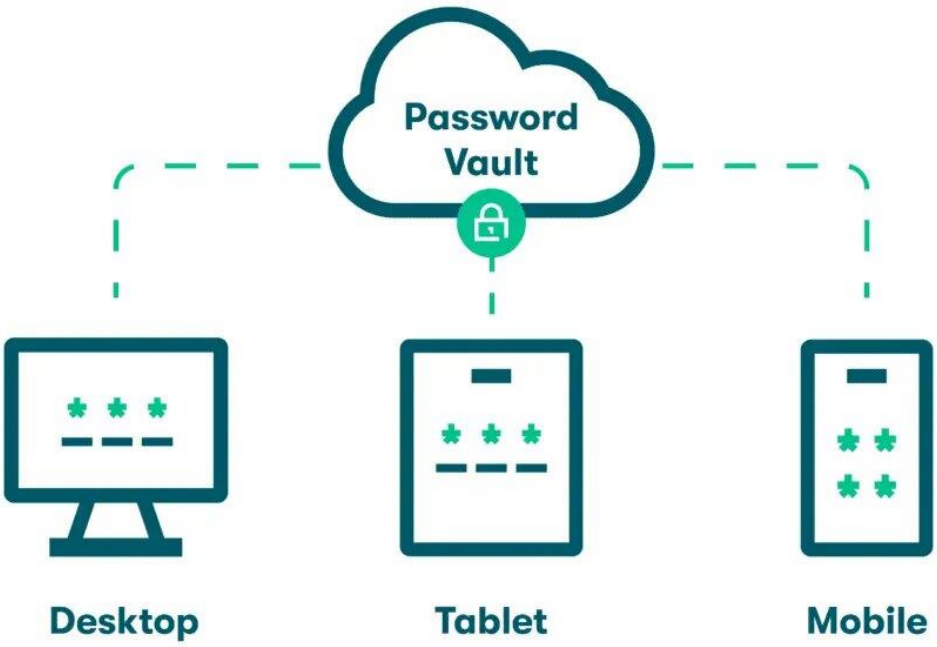
This coursework is centred on the development of a basic network packet processing application utilizing the C programming language. The application is designed to analyze incoming data packets and make routing decisions predicated on specified criteria. To establish a theoretical framework, a succinct literature review is incorporated, which contextualizes existing research pertaining to networking and packet processing. This review serves to elucidate the current state of knowledge in the field and the relevance of the proposed program. The methodology section is dedicated to elucidating the algorithmic approach and the underlying logic employed in the application's design. A detailed exposition of the methodologies adopted reveals the systematic steps taken to achieve the desired functionality. Furthermore, the execution and output segment illustrate the performance of the application across a range of scenarios, ensuring that the outputs are

Table of contents

Introduction

1.1) BACKGROUND

A password manager is packed with helpful features that make managing your online security a lot easier. For starters, it offers **password backup and recovery**, so if you ever lose access to an account, you can quickly recover your credentials without stress. It also helps you create **unique passwords** for each account, which are strong, random, and much harder for hackers to crack. To make your accounts even safer, many password managers support **two-factor authentication (2FA)**, which adds an extra layer of security beyond just your password. The **autofill** feature automatically fills in your login details on websites and apps, so you don't have to waste time typing everything out. When it comes to storage, your passwords are kept safe with **strong encryption**, making sure no one can access them without permission. Plus, with **password syncing**, you can easily access your passwords across all your devices, whether you're on your.



1.2 **Problem Statement:**

Problem Statement:

As we increasingly rely on digital platforms, people are required to manage an ever-growing number of online accounts, each with its own set of login credentials. Keeping track of all these passwords has become a major challenge. Many users resort to weak or repetitive passwords, putting their personal and professional accounts at risk of cyber-attacks and identity theft. Moreover, storing passwords in unsecured places like written notes, browsers, or unprotected files only further heightens the potential for breaches.

1.3 Objectives and Goals of the Project

The primary project is to design and develop a password manager while using c programming and it is user friendly tool and make strong password.

The objectives of password manager are;

- **Generate Strong and Secure Passwords:** The tool will help users create **unique and robust passwords** for each account, ensuring their data is protected through encryption.
- **Safe and Accessible Storage:** The password manager will provide a **secure location** to store passwords, making them easy to retrieve whenever needed, while keeping them protected from unauthorized access.
- **Automatic Password Generation:** It will include a **password generator** that creates **random and strong passwords** to ensure users are always using the best possible security for their accounts.

2.Literature Review

A password manager functions as a software application that facilitates the secure storage and management of passwords, thereby aiding users in the maintenance of strong and unique passwords across various accounts. The existing literature pertaining to password managers emphasizes their role in enhancing security provisions, as they store passwords within an encrypted vault and necessitate the use of a master password for access. Empirical studies indicate that the utilization of password managers significantly diminishes the likelihood of employing weak or redundant passwords, which are frequently identified as notable security vulnerabilities.

Moreover, the incorporation of features such as password generation, two-factor authentication (2FA), and cross-platform synchronization contributes to their effectiveness in protecting online identities. Nonetheless, it has been articulated by researchers that the selection of a reliable and user-friendly password manager is of paramount importance. This aspect ensures that users can efficiently navigate the software while benefiting from its security advantages.

2.3 algorithm

Algorithm of a Password Manager

1. User Setup/Registration

- **Step 1**: The user creates an initial **master password** to secure the manager.
- **Step 2**: Encrypt the master password using a strong encryption algorithm (e.g., AES or RSA).
- **Step 3**: Initialize the password manager and create a secure vault for storing all future passwords.
- **Step 4**: Present the user with options to either create, retrieve, or manage passwords.

2. Password Creation

- **Step 1**: The user requests a new password.
- **Step 2**: The password manager generates a **random password** based on user preferences (e.g., length, inclusion of symbols and numbers).
- **Step 3**: Encrypt the generated password for storage.
- **Step 4**: Save the encrypted password in the vault/database.
- **Step 5**: Provide the user with the new, generated password (and optionally copy it for use).

3. Password Storage

- **Step 1**: The user inputs a password for a new account.
- **Step 2**: Encrypt the password before storing it to protect it from unauthorized access.
- **Step 3**: Save the encrypted password along with the associated website or service name.

4. Password Retrieval

- **Step 1**: The user selects a website or service and requests the password.
- **Step 2**: The system prompts the user for **authentication** (e.g., master password or biometric input).
- **Step 3**: Decrypt the password stored in the vault.

- **Step 4**: Display the decrypted password to the user or automatically fill it into a login form.

5. Backup & Synchronization

- **Step 1**: The user can back up their password vault.
- **Step 2**: Encrypt the backup and store it securely, either locally or in the cloud.
- **Step 3**: Sync the encrypted vault across multiple devices to ensure all devices have up-to-date password data.

6. Two-Factor Authentication (2FA)

- **Step 1**: The user can enable **2FA** for accounts, adding an extra layer of protection.
- **Step 2**: Store the 2FA method (e.g., time-based one-time password or SMS) securely.
- **Step 3**: When logging in, the password manager prompts the user to enter the 2FA code before granting access.

7. Session Management

- **Step 1**: After the user logs out or after a set time, all session data (e.g., decrypted passwords) is cleared from memory.
- **Step 2**: Ensure that the password vault remains encrypted until the user logs in again with proper authentication.

3.procedure

3.1 Header Files and Macros

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define MAX_PASSWORDS 100
#define PASSWORD_LENGTH 12
```

Input/Output includes <stdio.h> is utilized for input and output operations like printf and scanf.

#include <stdlib.h> provides functions such as rand() to generate random numbers.

#include <string.h> This would be used for string manipulation functions, such as strcpy or strcmp.

#include <time.h> - Header used to get the current time to generate random numbers using time(NULL).

3.2 Struct Definition

```
typedef struct {
    char website[50];
    char username[50];
    char password[50];
} Credential;
```

t names the structure Credential and contains each user's website, username, and password.
website[50]: The name of the website up to 50 characters.
username[50]: This stores the username. Maximum 50 characters.
password[50]: Stores the password (max 50 characters).

3.3 Global Variables

```
Credential passwordList[MAX_PASSWORDS];
int credentialCount = 0;
```

passwordList[MAX_PASSWORDS]: This array of Credential structures will hold all the credentials of the user. The size of the array is constrained by MAX_PASSWORDS, which here is taken to be 100.
credentialCount - To maintain the record of credentials being stored currently.

4. Generate Password Function

```
void generatePassword(char *password) {
    const char charset[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01
    for (int i = 0; i < PASSWORD_LENGTH; i++) {
        password[i] = charset[rand() % (sizeof(charset) - 1)];
    }
    password[PASSWORD_LENGTH] = '\0';
}
```

The function generates Password(char *password) is designed to create a random password. The permissible characters for inclusion in the password are specified by the charset array, which encompasses uppercase letters, lowercase letters, digits, and special characters.
To achieve the desired length of the password, which is defined as PASSWORD_LENGTH (12), a loop is executed that iterates exactly twelve times. During each iteration, a random character is selected from the charset array through the use of the rand () function.
Upon the completion of populating the password array, a null terminator ('\\0') is appended to ensure the result conforms to the requirements of a valid C string.

Add Credential Function

```
void addCredential() {
    if (credentialCount >= MAX_PASSWORDS) {
        printf("Password list is full. Cannot add more credentials.\n");
        return;
    }

    Credential newCredential;

    printf("Enter website: ");
    scanf("%s", newCredential.website);

    printf("Enter username: ");
    scanf("%s", newCredential.username);

    generatePassword(newCredential.password);
    passwordList[credentialCount++] = newCredential;

    printf("Password for %s (%s) has been generated and saved.\n", newCredential.websit
}
```

addCredential(): This is for adding a new credential - that is, a website and username.

If the list of passwords is full, it prints a message and then exits.

Otherwise, it asks the user to input a website and a username.

This subsequently calls the function generate Password, returning a password for this credential.

The new credential is added to the pass wordlist array, after which the credential Count is incremented.

6. View Credentials Function

```
void viewCredentials() {
    if (credentialCount == 0) {
        printf("No credentials saved yet.\n");
        return;
    }

    printf("\nStored Credentials:\n");
    printf("--------------------\n");
    for (int i = 0; i < credentialCount; i++) {
        printf("Website: %s\n", passwordList[i].website);
        printf("Username: %s\n", passwordList[i].username);
        printf("Password: %s\n", passwordList[i].password);
        printf("--------------------\n");
    }
}
```

view Credentials(): This method shows all the saved credentials.
In the event of no saved credentials, it prints a message that no credentials are saved.
Otherwise, it iterates through the password List array and prints out a website, username, and password for each stored credential.

7. Main Function

```
void viewCredentials() {
    if (credentialCount == 0) {
        printf("No credentials saved yet.\n");
        return;
    }

    printf("\nStored Credentials:\n");
    printf("--------------------\n");
    for (int i = 0; i < credentialCount; i++) {
        printf("Website: %s\n", passwordList[i].website);
        printf("Username: %s\n", passwordList[i].username);
        printf("Password: %s\n", passwordList[i].password);
        printf("--------------------\n");
    }
}
```

main (): This is the entry point of the program. It starts by initializing a random seed-stand (time (NULL))-for the random number generator. The program runs in an infinite while (1) loop, presenting the user with a menu. Depending on the user's choice, it: Calls add Credential () to add a new credential. Calls view Credentials () to display all stored credentials. Exits the program if the user chooses option 3.

In case of invalid input, the program will ask the user to retr

```c
int main() {
    int choice;

    srand(time(NULL));

    while (1) {
        printf("\nPassword Manager Menu:\n");
        printf("1. Add a new credential\n");
        printf("2. View stored credentials\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                addCredential();
                break;
            case 2:
                viewCredentials();
                break;
            case 3:
                printf("Exiting Password Manager. Goodbye!\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```

Output

Example 1: Add a New Credential

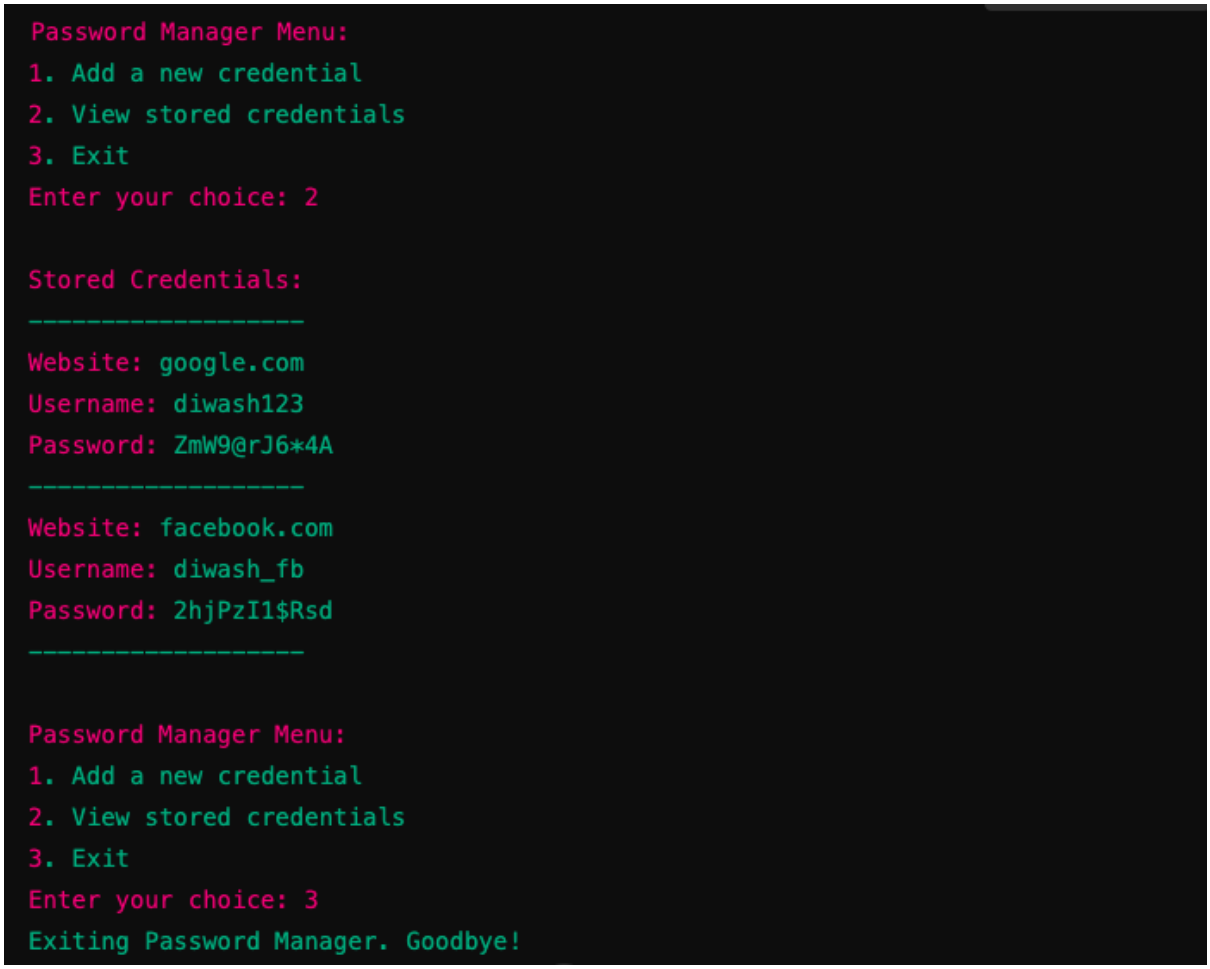
```
Password Manager Menu:
1. Add a new credential
2. View stored credentials
3. Exit
Enter your choice: 1
Enter website: google.com
Enter username: diwash123
Password for google.com (diwash123) has been generated and saved.

Password Manager Menu:
1. Add a new credential
2. View stored credentials
3. Exit
Enter your choice: 1
Enter website: facebook.com
Enter username: diwash_fb
Password for facebook.com (diwash_fb) has been generated and saved.
```

The functionality of a basic password manager through a command-line interface is illustrated in the output presented. Initially, a menu is displayed, offering three distinct options: the addition of a new credential, the viewing of saved credentials, or the termination of the program. The procedure commences with the selection of the option to add a new credential, wherein information pertaining to the website, specifically google.com, and the corresponding username, diwash123, is submitted by the user. Subsequently, a password is generated for this account, followed by a confirmation of its successful saving.

The user proceeds to repeat this process, this time entering facebook.com as the designated website and diwash_fb as the username. Again, the program generates a password and provides confirmation of its storage. Through this sequence of actions,

2: View Stored Credentials



The Password Manager program is designed to facilitate the secure management of user credentials. It features a menu that presents three distinct options: the addition of a new credential, the viewing of existing credentials, and the option to exit the program.

In the course of utilizing the program, the user chooses the first option on two separate occasions. Initially, the user inputs the website google.com along with the associated username diwash123. Subsequently, the user enters the website facebook.com with the corresponding username diwash_fb. For each credential entered, unique passwords are generated by the system and confirmation is provided regarding the successful storage of these credentials.

This program provides an effective solution for users seeking to securely store and organize their credentials associated with various websites. By doing so, it ensures both convenience and security in the management of sensitive information

4.1Discussion and conclusion

Conclusion

In the contemporary digital landscape, the utilization of password management tools has become increasingly essential. Such tools serve the purpose of consolidating numerous passwords into a single, encrypted cloud vault, thereby facilitating user convenience and security. Password managers are engineered to generate robust and unique passwords, thereby simplifying the user's experience by minimizing the need to memorize multiple credentials.The security of these password managers is primarily contingent upon the requirement of a master password, which must be entered diligently to gain access to the stored accounts. This feature significantly mitigates the risks associated with unauthorized access to sensitive information. Given the escalating complexity of cyber threats, the implementation of a password manager represents a proactive measure in the defense of both personal and professional data in the online realm.Moreover, the role of password managers extends beyond mere convenience; their importance in safeguarding personal information cannot be overstated. Unauthorized individuals gaining entry into a password manager can potentially access a wide array of personal and sensitive information, thus underscoring the necessity for robust security measures and prudent password management practices.

Discussion

A password manager is a handy tool that helps you keep track of all your passwords, making it easier to log into your accounts. It plays an important role in online security. To use a password manager, you first need to enter a master password. Think of this master password as the key to your digital vault. This vault is where all your passwords are safely stored, so you don’t need to remember each one. Instead, you only need to remember that single master password to access everything. All the passwords inside are protected through encryption, which means only someone with the master password can see them. This simple but effective tool keeps your online accounts secure and organized, giving you peace of mind as you manage your different accounts.

4.2Further improvements

The only thing that we can do with this password manager is to include those features to make it safer and more comfortable to use. For instance, the inclusion of 2FA provides an added layer of protection against logging in, asking something above and beyond just your master password. A password strength checker would ensure your passwords are strong enough to defend against hackers, and the ability to automatically generate random, strong passwords would save you time and effort. It would be great to sync your passwords across all devices so that you can easily access them from your phone, tablet, or computer. Regular security checks would help spot weak or reused passwords and keep your accounts safe. Above all, safely sharing passwords with people you trust, logging in with your fingerprint or face, and backing up and recovering all data in a straightforward way would make everything function in a much more seamless and secure manner. These enhancements will make a password manager even more secure but also more convenient for managing your life online with confidence.

4.3 References

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language* (2nd ed.). Prentice-Hall.