Projet 8 : Maitrise du full stack (C,C++)

Damian BAERTS

February 14, 2024

1 Introduction

Avant d'expliquer comment j'ai réalisé ce projet, je voudrai m'introduir rapidement. Je m'appelle Damian BAERTS. Je suis actuellement en première année de cybersécurité à l'école Guardia Cybersecurity. Par rapport au language C et C++, je n'avais aucunes connaissances sur ces languages avant le début du projet ainsi que quelques craintes vis à vis de l'apprentissage de ces languages car j'avais entendu que ce n'était pas un language simple.
Au fur et à mesure de l'avancement du projet, j'ai pu découvrir les bases du language C et voir qu'au final, non, ce n'était pas si compliqué. Nous avons prit le temps d'apprendre les bases du language (2 semaines) puis nous avons commencé à réaliser le projet demandé (1 semaine).

2 Projet demandé :

Le projet a consisté à réaliser un programme en C pour déterminer le type de hashage utilisé sur, soit un hash donné en dur, soit un hash contenu dans un fichier externe.

2.1 Les hashs gérés

Le programme devait au minimum reconnaitre les hash suivants :
- MD5
- SHA-1
- SHA-256
- SHA-512

2.2 Les arguments

Le programme devait accepter deux arguments :
- "-f" qui permet de mettre le chemin d'accès d'un document externe qui contient un hash.

```
PS C:\Users\dbaer\Nextcloud\Guardia\Projet 8\Compte rendu> .\hash.exe -f .\fichier_test.txt
```

- "-h" qui permet de passer directement un hash en dur.

```
PS C:\Users\dbaer\Nextcloud\Guardia\Projet 8\Compte rendu> .\hash.exe -h d41d8cd98f00b204e9800998ecf8427e
```

2.3 Les librairies

Les librairies autorisées sont :
- stdio.h
- stdlib.h
- time.h

En cas de besoin, nous pouvions utiliser d'autres librairies mais il faudra justifier leurs utilisations.

3 Mon programme

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

// On defini les tailles des hashs
#define MD5_LENGTH 32
#define SHA1_LENGTH 40
#define SHA224_LENGTH 56 // Nouvelle constante pour SHA-224
#define SHA256_LENGTH 64
#define SHA384_LENGTH 96 // Nouvelle constante pour SHA-384
#define SHA512_LENGTH 128
#define NOMBRE_TESTS 1000000

// Fonction pour calculer la longueur d'une chaine de caractères
int calculerLongueurChaine(char *chaine) {
    int longueur = 0;
    while (chaine[longueur] != '\0') {
        longueur++;
    }
    return longueur;
}

// Fonction pour determiner le type de hachage et le renvoyer
void determinerType(int hash_length) {
    switch (hash_length) {
        case MD5_LENGTH:
            printf("Le hash est de type MD5.\n");
            break;
        case SHA1_LENGTH:
            printf("Le hash est de type SHA-1.\n");
            break;
        case SHA224_LENGTH:
            printf("Le hash est de type SHA-224.\n");
            break;
        case SHA256_LENGTH:
            printf("Le hash est de type SHA-256.\n");
            break;
        case SHA384_LENGTH:
            printf("Le hash est de type SHA-384.\n");
            break;
        case SHA512_LENGTH:
            printf("Le hash est de type SHA-512.\n");
            break;
        default:
            printf("Type de hash indetermine.\n");
            break;
    }
}

// Fonction pour determiner le type de hachage et mesurer le temps d'execution
void determinerTypeEtMesurerTemps(char *hash) {
    struct timespec debut, fin;
    double tempsExecution = 0.0;
    int hash_length = calculerLongueurChaine(hash);

    // Debut de la mesure du temps
    clock_gettime(CLOCK_MONOTONIC, &debut);

    // Determination du type de hachage
    determinerType(hash_length);

    // Fin de la mesure du temps
    clock_gettime(CLOCK_MONOTONIC, &fin);
    tempsExecution = (fin.tv_sec - debut.tv_sec) + (fin.tv_nsec - debut.tv_nsec) / 1e9;

    printf("Temps d'execution pour determiner le type de hachage : %.10lf secondes\n", tempsExecution);
}

```

```c
// Fonction principale + gestion des arguments
int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage : %s [-f | -h] <hash>\n", argv[0]);
        return 1;
    }

    if (argv[1][0] == '-' && argv[1][1] == 'h') {
        // Appel de la fonction pour determiner le type de hachage et mesurer le temps d'execution
        determinerTypeEtMesurerTemps(argv[2]);
    } else if (argv[1][0] == '-' && argv[1][1] == 'f') {
        // Lire le fichier et traiter chaque hachage
        FILE *fichier = fopen(argv[2], "r");
        // Gestion des erreurs
        if (fichier (char [44])"Erreur : Impossible d'ouvrir le fichier %s\n"
            printf("Erreur : Impossible d'ouvrir le fichier %s\n", argv[2]);
            return 1;
        }

        char hash[129]; // Pour stocker le hachage (128 caractères + 1 pour le caractère de fin de chaine)
        while (fscanf(fichier, "%128[^;];", hash) == 1) {
            printf("Hachage lu depuis le fichier : %s\n", hash);
            // Appel de la fonction pour determiner le type de hachage et mesurer le temps d'execution
            determinerTypeEtMesurerTemps(hash);
        }

        fclose(fichier);
    } else {
        // Gestion des erreurs
        printf("Option non reconnue : %s\n", argv[1]);
        return 1;
    }
    return 0;
}
```

4 Explication du code

4.1 Inclusions des bibliothèques

Les inclusions des bibliothèques sont effectuées au début du programme pour importer les fonctionnalités nécessaires. Les bibliothèques incluses sont :

- `stdio.h` : Fournit les fonctions d'entrée/sortie standard.
- `stdlib.h` : Contient les fonctions d'allocation de mémoire dynamique.
- `time.h` : Utilisée pour la mesure du temps.

4.2 Définition des tailles des hashs

```c
// On defini les tailles des hashs
#define MD5_LENGTH 32
#define SHA1_LENGTH 40
#define SHA224_LENGTH 56 // Nouvelle constante pour SHA-224
#define SHA256_LENGTH 64
#define SHA384_LENGTH 96 // Nouvelle constante pour SHA-384
#define SHA512_LENGTH 128
#define NOMBRE_TESTS 1000000
```

Les longueurs attendues des différents types de hachage sont définies comme des constantes. Ces constantes sont utilisées pour identifier le type de hachage en fonction de sa longueur.

4.3 Fonction pour calculer la longueur d'une chaîne de caractères

```c
// Fonction pour calculer la longueur d'une chaine de caractères
int calculerLongueurChaine(char *chaine) {
    int longueur = 0;
    while (chaine[longueur] != '\0') {
        longueur++;
    }
    return longueur;
}
```

La fonction `calculerLongueurChaine` prend une chaîne de caractères en entrée et retourne sa longueur en comptant le nombre de caractères jusqu'à rencontrer le caractère nul `'\0'`.

4.4 Fonction pour déterminer le type de hachage

```c
// Fonction pour determiner le type de hachage et le renvoyer
void determinerType(int hash_length) {
    switch (hash_length) {
        case MD5_LENGTH:
            printf("Le hash est de type MD5.\n");
            break;
        case SHA1_LENGTH:
            printf("Le hash est de type SHA-1.\n");
            break;
        case SHA224_LENGTH:
            printf("Le hash est de type SHA-224.\n");
            break;
        case SHA256_LENGTH:
            printf("Le hash est de type SHA-256.\n");
            break;
        case SHA384_LENGTH:
            printf("Le hash est de type SHA-384.\n");
            break;
        case SHA512_LENGTH:
            printf("Le hash est de type SHA-512.\n");
            break;
        default:
            printf("Type de hash indetermine.\n");
            break;
    }
}

```

La fonction `determinerType` détermine le type de hachage en fonction de sa longueur. On utilise `switch` pour afficher le cas correspondant.

4.5 Fonction pour déterminer le type de hachage et mesurer le temps d'exécution

```c
// Fonction pour déterminer le type de hachage et mesurer le temps d'exécution
void determinerTypeEtMesurerTemps(char *hash) {
    struct timespec debut, fin;
    double tempsExecution = 0.0;
    int hash_length = calculerLongueurChaine(hash);

    // Début de la mesure du temps
    clock_gettime(CLOCK_MONOTONIC, &debut);

    // Détermination du type de hachage
    determinerType(hash, hash_length);

    // Fin de la mesure du temps
    clock_gettime(CLOCK_MONOTONIC, &fin);
    tempsExecution = (fin.tv_sec - debut.tv_sec) + (fin.tv_nsec - debut.tv_nsec) / 1e9;

    printf("Temps d'exécution pour déterminer le type de hachage : %.10lf secondes\n", tempsExecution);
```

La fonction `determinerTypeEtMesurerTemps` mesure le temps d'exécution pour déterminer le type de hachage d'un hash donné. Elle appelle la fonction `calculerLongueurChaine` pour obtenir la longueur du hash, puis utilise la fonction `determinerType` pour déterminer le type de hachage.

Détails :

La fonction `determinerTypeEtMesurerTemps(char *hash)` est déclarée pour prendre un pointeur vers une chaîne de caractères représentant le hash dont on veut déterminer le type.

Les variables `struct timespec debut` et `fin` sont utilisées pour stocker les temps de début et de fin de l'opération de détermination du type de hachage.

La variable `tempsExecution` est utilisée pour stocker la durée de l'opération en secondes.

La ligne `int hash_length = calculerLongueurChaine(hash);` appelle la fonction `calculerLongueurChaine` pour déterminer la longueur de la chaîne de caractères `hash`, ce qui correspond à la longueur du hachage.

L'instruction `clock_gettime(CLOCK_MONOTONIC, &debut);` obtient le temps actuel et le stocke dans la variable `debut`, marquant ainsi le début de l'opération de détermination du type de hachage.

La ligne `determinerType(hash_length);` appelle la fonction `determinerType` pour déterminer le type de hachage du hash en fonction de sa longueur.

L'instruction `clock_gettime(CLOCK_MONOTONIC, &fin);` obtient à nouveau le temps actuel et le stocke dans la variable `fin`, marquant ainsi la fin de l'opération de détermination du type de hachage.

La ligne `tempsExecution = (fin.tv_sec - debut.tv_sec) + (fin.tv_nsec - debut.tv_nsec) / 1e9;` calcule la durée de l'opération en secondes en soustrayant le temps de début du temps de fin et en tenant compte de la précision des nanosecondes.

4.6 Fonction principale + gestion des arguments

```c
// Fonction principale + gestion des arguments
int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage : %s [-f | -h] <hash>\n", argv[0]);
        return 1;
    }

    if (argv[1][0] == '-' && argv[1][1] == 'h') {
        // Appel de la fonction pour déterminer le type de hachage et mesurer le temps d'exécution
        determinerTypeEtMesurerTemps(argv[2]);
    } else if (argv[1][0] == '-' && argv[1][1] == 'f') {
        // Lire le fichier et traiter chaque hachage
        FILE *fichier = fopen(argv[2], "r");
        // Gestion des erreurs
        if (fichier == NULL) {
            printf("Erreur : Impossible d'ouvrir le fichier %s\n", argv[2]);
            return 1;
        }

        char hash[129]; // Pour stocker le hachage (128 caractères + 1 pour le caractère de fin de chaîne)
        while (fscanf(fichier, "%128[^;];", hash) == 1) {
            printf("Hachage lu depuis le fichier : %s\n", hash);
            // Appel de la fonction pour déterminer le type de hachage et mesurer le temps d'exécution
            determinerTypeEtMesurerTemps(hash);
        }

        fclose(fichier);
    } else {
        // Gestion des erreurs
        printf("Option non reconnue : %s\n", argv[1]);
        return 1;
    }
    return 0;
}
```

Cette partie du code est la fonction principale (`main`) du programme, elle est responsable de la gestion des arguments fournis en ligne de commande lors de l'exécution du programme.

La fonction `main` prend deux paramètres : `argc`, qui représente le nombre d'arguments passés à la ligne de commande, et `argv`, un tableau de chaînes de caractères contenant les arguments eux-mêmes.

Le premier bloc `if` vérifie si le nombre d'arguments est différent de 3. Si ce n'est pas le cas, cela signifie que l'utilisateur n'a pas fourni les arguments attendus, donc le programme affiche un message d'erreur expliquant comment utiliser le programme et se termine avec un code de retour de 1, indiquant une erreur.

Ensuite, le programme vérifie si le premier argument (`argv[1]`) commence par un tiret (-) et si le deuxième caractère de cet argument est `h`. Cela indique que l'utilisateur a choisi l'option `-h` pour passer un seul hachage à analyser. Dans ce cas, le programme appelle la fonction `determinerTypeEtMesurerTemps` avec le hachage passé comme deuxième argument (`argv[2]`). Cette fonction détermine le type de hachage et mesure le temps d'exécution.

Si le premier argument commence par un tiret et que le deuxième caractère est `f`, cela signifie que l'utilisateur a choisi l'option `-f` pour spécifier un fichier contenant plusieurs hachages à analyser. Dans ce cas, le programme ouvre le fichier spécifié (`argv[2]`) en mode lecture. Si le fichier ne peut pas être ouvert, il affiche un message d'erreur et se termine avec un code de retour de 1. Sinon, il lit chaque hachage du fichier, appelle la fonction `determinerTypeEtMesurerTemps` pour chaque hachage et affiche le résultat.

Si aucun des cas précédents n'est vérifié, cela signifie que l'option fournie n'est pas reconnue. Le programme affiche donc un message d'erreur indiquant l'option non reconnue et se termine avec un code de retour de 1.

Enfin, la fonction `main` se termine en renvoyant 0 si tout s'est déroulé correctement, indiquant ainsi que le programme a été exécuté avec succès.

5 Amélioration

Pour les améliorations, j'ai pensé à calculer le temps nécessaire pour déterminer le type de hash car tous les hashs ne mettent pas le même temps à être trouvé. Cela rajoute donc une vérification sur le type de hash trouvé. De plus j'ai rajouter le traitement de 2 types de hashages : le SHA-224 et le

SHA-384.

6 Conclusion

Ce projet a été très interressant. Il m'a permis de découvrir le language C, de comprendre comment il fonctionne, à quoi il sert et comment l'utiliser. Il m'a permis d'oublier tous les craintes que j'avais par rapport à ce language.