

Projet Sécurité :

Virus compagnon

Table des matières

[1. Grandes fonctions d’un virus 3](#_Toc130312528)

[2. Partie affichage de MediaPlayer 3](#_Toc130312529)

[3. Programmes utilitaires du projet 5](#_Toc130312530)

[4. Fonctions d’un virus compagnon dans le cadre du projet 6](#_Toc130312531)

[a) Fonction infection() 6](#_Toc130312532)

[b) Fonction creationOld(struct dirent \*files) 7](#_Toc130312533)

[c) Fonction copyMediaPlayer(struct dirent \*files) 8](#_Toc130312534)

[d) Fonction Main() 9](#_Toc130312535)

[5. Une vérification double 10](#_Toc130312536)

[6. Copie d’une version exécutable 10](#_Toc130312537)

[7. Amplifier l’infection 10](#_Toc130312538)

[8. Oublie du transfert d’exécution 10](#_Toc130312539)

[9. Sources 10](#_Toc130312540)

# Grandes fonctions d’un virus

Pour commencer un virus est un programme indésirable qui a besoin d’un autre programme ou fichier qui joue le rôle de l’hôte du virus et sans lequel le virus ne peut pas se propager.

Pour se faire il met en œuvre différents mécanismes d’attaques :

* S’introduire dans la machine ( via internet, Mails, clé USB …)
* Se copier
* Rechercher les fichiers cibles
* Se propager

Pour cela il faudra implémenter des fonctions de :

* Recherches ( lutte contre la surinfection )
* Fonctions de reproductions
* Destruction

# Partie affichage de MediaPlayer

L’affichage de MediaPlayer.exe devant être simple et intuitive pour l’utilisateur qui pense utiliser ce programme pour afficher les photos et vidéos présentes dans le dossier a été conçu avec ***GTK*** qui est un ensemble complet d'éléments d'interface utilisateur compatible avec plusieurs langages dont le langage C utilisé pour ce projet.

Avant toutes choses il faut correctement configurer notre machine et la mettre à jour pour éviter tout problèmes.

1. Mise à jour de notre machine

**sudo apt upgrade**

**sudo apt upgrade**

**sudo apt install build-essential**

1. Il faudra vérifier la présence de la librairie GTK :

**dpkg-query -l**

Si GTK n’est pas installer :

**sudo apt-get install libgtk-3-dev**

**sudo apt-get install libcanberra-gtk-dev**

1. Compiler un programme C :

Exemple avec la compilation de *MediaPlayer.c*

**gcc -Wall ./MediaPlayer.c -o ./MediaPlayer.exe `pkg-config --cflags --libs gtk+-3.0`**

Exemple avec un programme utilitaire :

**gcc -Wall ./MonPG1.c -o ./MonPG1.exe `pkg-config --cflags --libs gtk+-3.0`**

Utilisation de pkg-config pour pouvoir utiliser les librairies GTK

1. Exécuter un programme .exe :

Penser au préalable si nécessaire cette commande pour donner les droits d’exécutions

**chmod +x ./MediaPlayer.exe**

Exemple avec l’exécution de *MediaPlayer.exe*

**./MediaPlayer.exe**

Mais pour le cadre du projet nous avons créé des fichiers *installeur.sh* qui va installer toutes les dépendances/librairies et *compiler.sh* qui permettent de générer un utilisateur lambda qui aura juste à utiliser les « .exe ».

**Une image contenant texte, nature

Description générée automatiquement**

Fig.1 : Capture de MediaPlayer.exe

Au lancement du programme *MediaPlayer.exe*affiche simplement une image [Fig.1], l’utilisateur peut visionner les autres photos présentent dans le dossier courant ici référentiel, en sélectionnant tout simplement le bouton Open qui va ouvrir une fenêtre du répertoire afin de pouvoir ouvrir d’autres images [Fig.2]. Il peut quitter le programme en cliquant sur le bouton Quit ou la croix de la fenêtre de l’application.

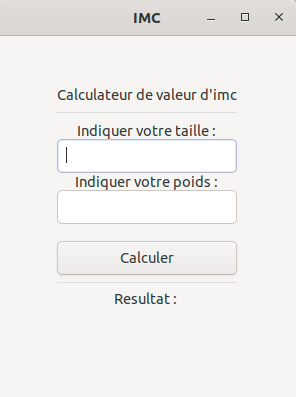
Une image contenant texte

Description générée automatiquement

Fig.2 : Open file MediaPlayer.exe

# Programmes utilitaires du projet

Le dossier référentiel est composé de 6 images avec plusieurs extensions (.JPG, .PNG, .GIF), du fichier *MediaPlayer.c* et de son exécutable *MediaPlayer.exe*, ainsi que 6 fichiers portant les noms *MonPG1-6.c* représentent des petits programmes utilitaires pouvant être tout comme MediaPlayer lancé par l’utilisateur, qui seront donc la cible de notre virus compagnon.

Une image contenant texte, Appareils électroniques, léger, clavier

Description générée automatiquement

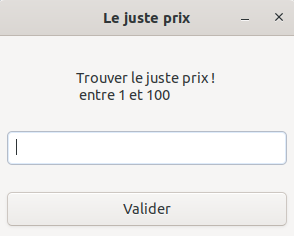


Fig.3 : MonPG1.exe Fig.4 : MonPG2.exe Fig.5 : MonPG3.exe

Une image contenant texte

Description générée automatiquement

Fig.6 : MonPG4.exe

Une image contenant texte, Appareils électroniques, clavier

Description générée automatiquementUne image contenant table

Description générée automatiquement

Fig.7 : MonPG5.exe Fig.8 : MonPG6.exe

# Fonctions d’un virus compagnon dans le cadre du projet

## Fonction infection()

Le virus va commencer par scanner le répertoire courant en recherche des fichiers cibles à attaquer grâce à la fonction *infection()*, pour ce faire il devra parcourir l’ensemble du répertoire courant ou il se trouve, pour chaque fichier il devra récupérer leurs statuts ici on veut vérifier que les fichiers possèdent bien le statut exécutables (S\_IXUSR) et réguliers (S\_IFREG) disponible avec le champ st\_mode du fichier.

La fonction va donc examiner chaque fichier tant que le dossier n’est pas vide, pour renommer un fichier nous utilisons la variable old de type char qui permet de créer le nouveau nom du fichier/programme en les concaténant. Avant de lui donner sa nouvelle extension nous commençons par ajouter un « . » devant, ceci permet de rendre le programme caché dans le dossier [Fig9] ici nous ne voyons pas « .*MonPG1.exe.old* », afin que l’utilisateur ne voie pas deux *MonPGX.exe* car l’ancien fichier est bien présent dans le dossier on peut le vérifier avec la commande ll du terminal [Fig10] avec cette commande nous pouvons facilement vérifier la présence de « .*MonPG1.exe.old »*. C’est lui qui sera exécuté plus tard si l’utilisateur souhaite utiliser un utilitaire. Ensuite il concatène avec le « . » le nom du fichier original et l’extension« .old ». La variable old donne donc « **.MonPG1.exe.old** » qui créé un nouveau fichier portant ce nom.

Une image contenant texte

Description générée automatiquement

Fig.9 : Aperçu du dossier avec la commande ls

Une image contenant texte

Description générée automatiquement

Fig.10 : Aperçu du dossier avec la commande ll

Ensuite il lui faudra vérifier quels fichiers réellement cibler. En effet pour éviter la surinfection il ne devra pas infecter un fichier « .exe » possédant déjà un « .old », si la condition est vérifiée le programme fait appelle à la fonction *creationOld(struct dirent \*files)* puis *copyMediaPlayer(struct dirent \*files).*

## Fonction creationOld(struct dirent \*files)

Cette fonction appelée dans *infection()* va permettre de créer la copie du fichier/programme original passé en paramètre dans le fichier« .old » créé précédemment dans la fonction *infection()*. La variable commande de type char permet de créer la commande que la fonction *system()* exécuteraà la fin de la fonction *creationOld()* en concaténant les valeurs passées à *strcat* et *strpy.* La variable commande se traduire par « **cp MonPG1.exe .MonPG1.exe.old** »

Une image contenant texte

Description générée automatiquement

Fig.11 : Fonction creationOld

## Fonction copyMediaPlayer(struct dirent \*files)

Une fois toutes les vérifications établies on va pouvoir copier et transférer le code d’exécution au code hôte, ce que permet la fonction *copyMediaPlayer(struct dirent \*files)*qui va simplement copier le code de MediaPlayer.exe dans le code du nouveau fichier créé précédemment avec **files->d\_name** quicorrespond au nom du fichier passé en paramètre.

Evidemment si l’utilisateur veut ouvrir *MediaPlayer.exe* après infection, rien ne se passe car pas besoin de passer la main à un autre programme en effet aux yeux de l’utilisateur MediaPlayer apparait comme voulu c’est-à-dire juste afficher les images ou vidéos du dossier.

Cependant si l’utilisateur veut ouvrir un des programmes utilitaires infectés il est nécessaire de transférer l’exécution au programme original pour ne pas alerter l’utilisateur sur le mauvais fonctionnement ou une différence du programme original qu’il voulait utiliser.

Une image contenant texte

Description générée automatiquement

Fig.12 : Fonction copyMediaPlayer

## Fonction Main()

Au lancement on lance l’infection avec la fonction *infection()* vue précédemment, ensuite il faut regarder si le programme demandé est *MediaPlayer.exe* pour ce faire nous vérifions que argv[0] est égal à « *./MediaPlayer.exe* »si c’est le cas c’est que l’utilisateur souhaite lancer ce programme et on donne la main à la fonction *MediaPlayer()* qui est le cœur du programme de *MediaPlayer*.

Dans le cas contraire si argv[0] est égal à « *./MonPg1.exe* » par exemple ou un programme différent de *MediaPlayer* alors on transfère l’exécution du programme a son original à savoir le « .old » dans l’exemple ici l’exécution du programme ./MonPg1.exe donne ici la main à « *./.MonPg1.exe.old* ».

La variable commande de type char est la variable qui prend la chaine de caractères permettant d’écrire la commande que la fonction system() exécutera pour lancer le programme original.

Une image contenant texte

Description générée automatiquement

Fig.13 : Fonction main()

# Une vérification double

Le principe de vérification double est de vérifié si dans le répertoire commun il existe bien un «.exe » et qu’il est différent de notre média player et aussi de vérifié si un .old de ce fichier existe déjà. S’il existe cela veut dire que le fichier a déjà été infecté.

# Copie d’une version exécutable

Lorsque le virus se duplique il doit bien être en version exécutable afin que l’utilisateur puisse ouvrir le programme utilitaire normalement et ne pas éveiller les soupçons d’un changement dans les fichiers sources.

# Amplifier l’infection

Pour amplifier l’infection lors de l’étape 6 le virus pourrait aussi cibler les fichiers d’une autre extension que « .exe »

# Oublie du transfert d’exécution

Comme lors de la question portant sur la copie d’une version exécutable, l’étape 7 est d’autant plus importante car c’est elle qui va passer la main au programme original lorsque l’utilisateur souhaite utiliser un utilitaire autre que *MediaPlayer,* il ne verra pas que le programme aura été renommé et remplacé par notre copie de *MediaPlayer*

# 9. Sources

[1] http://turrier.fr/articles/ubuntu-gcc-gtk/programmer-en-c-avec-gtk.php