# Notre Malware

BOUCHER, DA SILVA, SMAGGHE

December 2023

## 1 Introduction

Ce document décrit les fonctionnalités implémentées et explique comment nous les utilisons dans notre malware.

## 2 Type et actions en cas de mauvaise entrée

Nous avons écrit un malware de type B. Il fait toujours un "echo" de la chaîne de caractères, et dans le cas où l'entrée est au mauvais format, le programme lance la fonction Switch qui change le bureau de la machine virtuelle, puis supprime le compte Administrateur puis éteind tout les services, ce qui force un redémarrage "lent" de la machine.

## 3 Fonctionnalités implémentées

### 3.1 Les fonctions d'Anti-Debug

- `IsDebugged1` : cette fonction émet un point d'arrêt. Si l'exécution de la fonction résulte d'une exception, cela signifie que le point d'arrêt n'a pas pu être inséré, indiquant ainsi que le programme n'est pas exécuté en mode débogage. Si la fonction réussit, alors le programme est exécuté en mode débogage,
- `IsDebugged2` : Lors d'une execution classique, il y a un Trap Flag dans le registre Flags. Lorsque le Trap Flag est défini, l'exception SINGLE_STEP est levée. Cependant, si nous avons executé le code en mode debogage, le Trap Flag sera effacé par le débogueur donc nous ne verrons pas l'exception,
- `IsDebugged3` : Cette fonction fonctionne de la même manière qu'IsDebugged1, mais avec un autre type de point d'arrêt. L'instruction INT3 est une interruption utilisée comme point d'arrêt,

- `IsDebugged4` : Cette fonction fonctionne de la même manière que IsDebugged1 et IsDebugged3, mais avec un autre type de point d'arrêt. L'instruction INT2D est une interruption utilisée comme point d'arrêt,
- `IsDebuggedChacha` : Cette fonction effectue du chiffrement, de l'auto-modification et de l'antidébogage. La fonction parcourt de manière aléatoire les fichiers présents dans la machine virtuelle en utilisant les fonctions shuffling_list(), get_file_path(), ListFiles(). Elle parcourt les fichiers jusqu'à ce qu'elle trouve le motif de la clé, permettant ainsi de trouver la clé. La clé est utilisée pour déchiffrer un texte avec XChaCha20Poly1305. Ensuite, la fonction vérifie un code d'authentification pour s'assurer de l'exactitude du déchiffrement. Le texte déchiffré est utilisé dans la fonction anti-débogage ci-dessous.

```
char * ppeb;
__asm{
    mov eax, fs:[0x30]
    mov ppeb, eax
}
```

Figure 1: fonction d'antidebug utiliser par la fonction IsDebuggedChacha

Cette fonction s'écrit sur le méthode IsDebuggedChacha (auto-modification) et la fonction IsDebuggedChacha devient une méthode d'anti-débogage.

### 3.2 Les autres fonctions

- `a` : fonction qui exécute les instructions assembleur de la fonction `strlen`,
- `Switch` : cette fonction change le bureau sur Windows XP, bloquant ainsi l'utilisateur de la machine virtuelle sur un nouveau bureau,
- `ListFiles` : cette fonction liste les fichiers,
- `Match` : cette fonction prend en entrée un motif et un fichier, puis cherche la clé en hexadécimal dans le fichier,
- `shuffling_list` : cette fonction mélange aléatoirement une liste de fichiers,
- `get_hexa` : cette fonction lit les nombres hexadécimaux présents dans un fichier,
- `get_file_path` : cette fonction récupère le chemin complet d'un fichier,
- `print_osef` : affiche dans la sortie standard un string en paramètre,
- `print_osef2`,
- `doBarrelRoll` : cette fonction calcule la suite de Fibonacci jusqu'à 67.

# 4 Programme

Tout au long de la fonction `main`, nous avons fait appel aux fonctions définies précédemment. Voici les différentes étapes de notre programme :

- vérification si le programme est actuellement débugué,
- vérification s'il y a bien deux arguments : la commande et le paramètre,
- vérification si le programme est actuellement débugué,
- vérification si le paramètre pris en entrée est bien inférieur à 32 caractères,
- vérification si le programme est actuellement débugué,
- vérification si le paramètre pris en entrée respecte le motif de la regex, qui vérifie que la clé est bien en hexadécimale,
- vérification si le programme est actuellement débugué,
- remplissage d'un tableau de `char` avec la chaîne de caractères "victory",
- calculs qui n'aboutissent à rien de concret,
- XOR qui n'aboutit à rien de concret,
- calculs de 14999 hash SHA256 qui n'aboutissent à rien de concret,
- vérification du 15000$^{\text{ème}}$ hash avec un hash écrit en dur. Les deux hash sont égaux, et donc on rentre dans la boucle `if`. Dans cette boucle `if`, on appelle la fonction `doBarrelRoll()` et la fonction `a("victory")`. La fonction `a` réalise seulement un `strlen` et donc n'affiche rien,
- pour finir, on fait un `echo` de la clé en appelant la fonction `print_osef()` et `while(1)`.