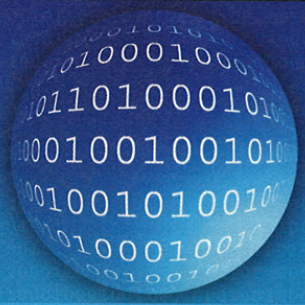

LES RANSOMWARES

Nicolas Brulez – Senior Security Researcher – Kaspersky lab, France

mots-clés : CODES MALICIEUX / REVERSE ENGINEERING / RANSOMWARE / ANALYSE DE CODE

Parmi les codes malicieux que l'on trouve de nos jours, il arrive que l'on rencontre un type de programmes assez particulier : les ransomwares.
Un ransomware est une application qui va modifier la machine pour ensuite demander une rançon au propriétaire, afin de retrouver l'état initial de la machine. En général, tous les fichiers dont l'extension correspond à une liste intégrée au ransomware seront chiffrés à l'aide de la cryptographie (ou d'algorithmes maison plus ou moins performants) et les indications pour obtenir un outil de déchiffrement sont fournies par le malware.
Il existe aussi des ransomwares qui bloquent toutes les applications excepté le navigateur internet pour pouvoir visiter un site web permettant de payer la rançon. Cet article présente l'un d'entre eux, trouvé mi-octobre, et qui est, comme nous allons le voir, très limité techniquement, pour ne pas dire, très basique.
MD5 : FEC60C1E5FBFF580D5391BBA5DFB161A

1 Analyse du ransomware

Notre exécutable n'est pas « *packé* » et ne contient aucune obfuscation. Il semblerait qu'il ait été programmé en assembleur.

Voici ce que l'on obtient après ouverture dans IDA : voir Figure ci-contre.

```
.text:00401290                  public start
.text:00401290 start            proc near
.text:00401290                  push    200000h         ; dwBytes
.text:00401295                  push    40h             ; uFlags
.text:00401297                  call    GlobalAlloc
.text:0040129C                  mov     lpBuffer, eax
.text:004012A1                  push    200h            ; uSize
.text:004012A6                  push    offset Buffer   ; lpBuffer
.text:004012AB                  call    GetWindowsDirectoryA
.text:004012B0                  push    offset aCryptlogfile_t ; "\\CryptLogFile.txt"
.text:004012B5                  push    offset Buffer   ; lpString1
.text:004012BA                  call    lstrcatA
.text:004012BF                  mov     al, Buffer
.text:004012C4                  mov     FileName, al
.text:004012C9                  call    GetCommandLineA
.text:004012CE                  mov     esi, eax
.text:004012D0                  lea     edi, byte_403F28
.text:004012D6
.text:004012D6 loc_4012D6:                              ; CODE XREF: start:loc_4012E0↓j
.text:004012D6                  lodsb
.text:004012D7                  cmp     al, 22h
.text:004012D9                  jz      short loc_4012E0
.text:004012DB                  cmp     al, 0
.text:004012DD                  jz      short loc_4012E2
.text:004012DF                  stosb
.text:004012E0
.text:004012E0 loc_4012E0:                              ; CODE XREF: start+49↑j
```

Fig. 1 : Point d'entrée du ransomware

On notera la référence au fichier `CryptLogFile.txt` qui pourrait être un journal des fichiers « protégés » par le *ransomware*. L'appel des fonctions `GetWindowsDirectoryA` et `lstrcatA` nous indique le chemin complet du fichier journal.

La première fonction de notre ransomeware est de calculer l'offset du fichier BMP qui a été ajouté à la fin du ransomware. Pour ce faire, notre *malware* lit le dernier DWORD (4 octets) de son propre fichier, et soustrait cette valeur à la taille du fichier précédemment récupérée à l'aide de la fonction `GetFileSize`.

```
                        ; CODE XREF: sub_4015B5+AB↑j
push    [ebp+hObject]   ; hObject
call    CloseHandle
push    200h            ; nSize
push    offset path_complet_tmp_wallpaper_bmp ; lpBuffer
push    offset aTmp     ; "TMP"
call    GetEnvironmentVariableA ; Récupère le répertoire temp
push    offset aWallpaper_bmp ; "\\wallpaper.bmp"
push    offset path_complet_tmp_wallpaper_bmp ; lpString1
call    lstrcatA
push    offset path_complet_tmp_wallpaper_bmp ; lpFileName
call    DeleteFileA     ; Efface le fichier avant de le recréer.


push    0               ; hTemplateFile
push    0               ; dwFlagsAndAttributes
push    2               ; dwCreationDisposition
push    0               ; lpSecurityAttributes
push    3               ; dwShareMode
push    0C0000000h      ; dwDesiredAccess
push    offset path_complet_tmp_wallpaper_bmp
call    CreateFileA
cmp     eax, 0FFFFFFFFh
jnz     short create_wallpaper
jmp     loc_4013F0
```

Fig. 2 : Drop du fichier wallpaper.bmp

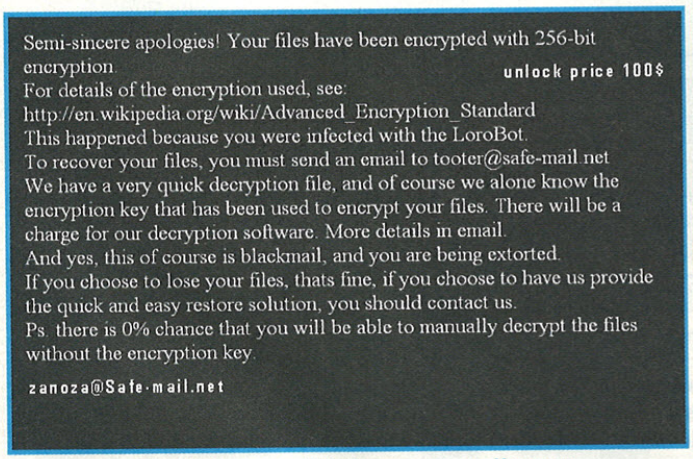

Fig. 3 : Image utilisée comme wallpaper

```
a_db_mp3_waw_jpgjpeg_txt_rtf_p db '.db.mp3.waw.jpgjpeg.txt.rtf.pdf.rar.zip
```

Fig. 4 : Extensions ciblées

```
start_encrypt:                  ; CODE XREF: sub_401263+A↑j
                mov     esi, plain_text_buffer
                mov     edi, esi
                xor     edx, edx

encrypt:                        ; CODE XREF: sub_401263+2A↓j
                cmp     edx, 16         ; Chaque dword utilise une clé différente
                jnz     short loc_401281 ; 4 clés possibles, roulement continue
                xor     edx, edx

loc_401281:                     ; CODE XREF: sub_401263+1A↑j
                lodsd
                xor     eax, key_table[edx] ; Une table de 4 dwords
                stosd
                add     edx, 4
                dec     ecx
                jnz     short encrypt
                retn
sub_401263      endp
```

Fig. 5 : Boucle de chiffrement

Il obtient donc un offset dans son propre fichier, vers une image qu'il va ensuite enregistrer dans le répertoire temporaire, sous le nom de **wallpaper.bmp**. Voici le code responsable de la création du fichier **wallpaper** : voir Figure 2.

Voici d'ailleurs le fameux *wallpaper* qui sera utilisé pour remplacer (à l'aide de la fonction **SystemParametersInfoA**) le wallpaper actuel de l'ordinateur infecté : voir Figure 3.

La seconde étape de notre code malicieux est de parcourir le disque dur à la recherche de fichiers pouvant être chiffrés dans le but d'obtenir une rançon. Il est possible de trouver une liste d'extensions recherchées par l'application. Voici donc cette fameuse liste : voir Figure 4.

Notre malware parcourt les disques de la machine à la recherche de ces fichiers pour les chiffrer. Pour ce faire, les fonctions standards de recherche de fichiers sont utilisées : **FindFirstFileA** et **FindNextFileA**. Pour chaque fichier trouvé sur le disque, l'extension est comparée à celles codées en dur dans le programme, et si celle-ci correspond, alors le fichier sera chiffré.

2 Chiffrement des fichiers

Voici la routine responsable du chiffrement des fichiers. Son fonctionnement est très simple. Elle utilise une table de 4 clés codées en dur. La routine effectue un roulement, et chaque DWORD utilise une clé différente. Tous les 4 DWORDS (à cause des 4 clés possibles), la routine retourne au début de la table des clés, et effectue un roulement continu jusqu'à ce que la totalité du fichier soit chiffré. La routine est un simple XOR et, en aucun cas, l'AES dont nous parle le wallpaper (voir Figure 5).

Une fois le fichier chiffré, la recherche de fichiers à chiffrer continue jusqu'à ce que tout le disque soit chiffré. Une fois tous les fichiers chiffrés, un fichier au nom effrayant (;-)) est créé, puis ouvert par l'application par défaut, contenant quelques mots pour le propriétaire de la machine infectée : voir Figure 6, page suivante.

La personne se rend compte alors que sa machine est infectée, surtout à l'aide du wallpaper modifié : voir Figure 7, page suivante.

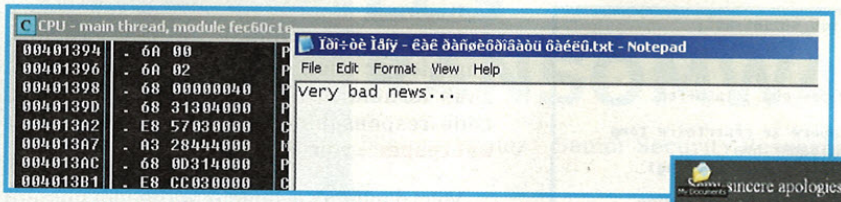

Fig. 6 : Chiffrement des fichiers terminé : Bad news...

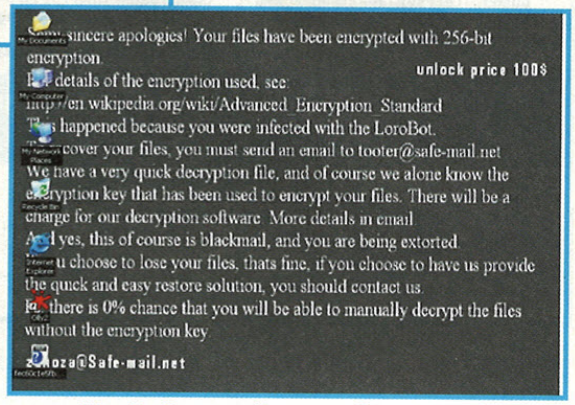

Fig. 7 : Bureau d'une machine infectée

3 Déchiffrement des fichiers

L'utilisateur lambda retrouvera tous ses fichiers chiffrés, et pensera qu'ils sont perdus à jamais, à moins de payer la rançon :

Fig. 8 : Exemple d'un fichier readme chiffré

Le fichier **CryptLogFile** contient tous les fichiers qui ont été chiffrés par notre ransomware. Il est intéressant de noter que notre ransomware vérifie la présence de ce fichier lors d'une seconde exécution et, s'il est présent, il ne continue pas plus loin et se ferme.

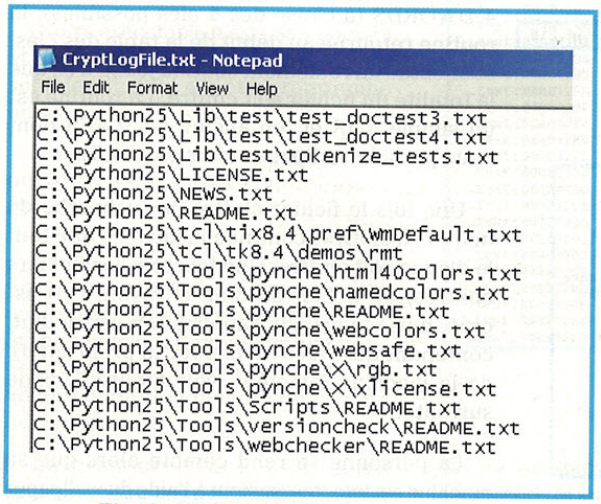

Fig. 9 : Fichier de log du ransomware

Comment déchiffrer nos fichiers ? Etant donné l'absence de contrôle, il suffit tout simplement d'effacer le fichier **CryptLogFile**, de relancer notre ransomware et, comme par magie, tous les fichiers seront déchiffrés automatiquement et gratuitement ;-) Voici un bel exemple de code bien pensé et efficace. En effet, l'utilisation de l'opérateur XOR permet un déchiffrement automatique des données.

Conclusion

En conclusion de cet article, j'aimerais insister sur le fait que nombreux sont les ransomwares qui prétendent utiliser de la cryptographie et, dans la majorité des cas, l'algorithme employé est très faible, voire ridicule. Malgré les messages alarmants que vous pourriez lire suite à une infection, il est préférable d'attendre plutôt que de payer la rançon, car, dans la majorité des cas, il est très simple de déchiffrer les données, et pour pas un centime. Certains ransomwares, comme celui présenté ici, vous fourniront même un moyen de déchiffrement intégré :-)

Cependant, il existe des cas (familles de ransomwares principalement) où la cryptographie est réellement employée et où il est impossible de récupérer ses fichiers. Je vous invite donc à faire des *backups* réguliers et à ne pas payer de rançon, pour éviter d'inciter à ce genre de pratique.