

Analyse de Locky

Auteur: Adrien Couëron

Introduction

Contexte

Ce document a été créé pour présenter une analyse du ransomware Locky afin d'améliorer les compétences personnels de l'auteur et de fournir une analyse approfondie du malware.

Le document a été réalisé alors que l'analyse n'était pas complète. Il permet de présenter les premiers résultats de l'analyse. Des mises à jour du document seront effectués tout au long de l'avancée de l'analyse.

Objectifs

Les principaux objectif de l'analyse est de trouver des moyens de bloquer le processus de chiffrement de fichiers (Communication avec le C&C, détection, ...) et de restaurer des fichiers déjà chiffrés (faiblesse de l'implémentation cryptographique, du partage de clés).

Structure du document

Le document se structure suivant les chapitres suivants:

- Informations sur les fichiers exécutables étudiés du malware
- Dépackage du malware
- Processus d'infection

- Détails de fonctionnement

Annotation

Des annotations du type ? pourront suivre des explications durant le rapport. Il signale un doute de l'auteur sur la véracité d'un fait. Il préfère de vous le signaler plutôt que de vous induire en erreur.

Si vous voulez affirmer ou corriger des doutes laissés dans ce rapport, il vous est possible d'entrer en contact avec l'auteur (adrien.coueron@wanadoo.fr). Tout doute levé améliorera la qualité de ce rapport et aidera les futurs lecteurs.

I-Informations sur les fichiers

Fichier packer

Hash

- MD5: 73304ca4e455286b7a63ed71af48390a
- SHA1: e8ea52e0d43f9420a65993a4123fc15d64bc880e
- SHA256:
3dc979164206c86823cab9684e662f84528d40a92027f48d31970c3d8f9f5114
- SHA512:
9d80839100d20c334a4c0f74bb8a2d4dc121c14bbd09d50a80eed7e94e514c8feb28c393f5fd90087b08e387bf83bffb7ac337a48578b86dcdb9b58d90a903c
- SSDEEP:
3072:wOM5W8c5FAswIJPY/ePTkflEVE/3WhKoxasMvzzzFVy0lv4p7RhPu/O3iXgOYbL:eW8c5KIJPY2LkflEVEPWhKnl+A6

Type de fichier

locky_packed: PE32 executable (GUI) Intel 80386, for MS Windows

Informations PE

- Date de compilation: 24/02/2016 10:53:51
- Machine cible: 0x14C Intel 386 et processeurs précédents compatibles
- Point d'entrée: 0x00417430

Sections

Nom	Adresse virtuelle	Taille virtuelle	Taille original	MD5
.text	0x00401000	0x17956	0x17A00	975653a6c2bd9ef08e
.core	0x00419000	0x200	0x17E00	e1596859847c1e1a1
.rsc	0x0041A000	0x108DC	0x10A00	51e53af42d7e5639c2

Imports

- ntdll.dll
 - RtlZeroMemory
- KERNEL32.dll
 - InitializeCriticalSection
 - Sleep
 - LeaveCriticalSection
 - GetProcAddress
 - EnterCriticalSection
 - LoadLibraryA
 - LocalAlloc
 - DeleteCriticalSection

- ReleaseMutex
- CloseHandle
- LocalFree
- CreateThread
- lstrcpA
- ExitProcess
- GetLastError
- ADVAPI32.dll
 - RegCreateKeyExA
 - SetSecurityDescriptorDacl
 - RegCloseKey
 - FreeSid
 - SetEntriesInAclA
 - InitializeSecurityDescriptor
 - AllocateAndInitializeSid
- COMCTL32.dll
 - InitCommonControlsEx
 - ImageList_Add

Fichier unpacker

Hash

- MD5: 45f4c705c8f4351e925aea2eb0a7f564
- SHA1: dc04128fd3e916e56ce734c06ff39653c32ade50
- SHA256:
034af3eff0433d65fe171949f1c0f32d5ba246d468f3cf7826c42831a1ef4031
- SHA512:
a4462f7d98ef88e325aac54d1acffd4b8f174baa77efd58f85cdd14520
1a99e7b03f9ba6f25bdd25265714aa25070a26f72d18401de5463a91

b3d21b47d17b13

- SSDEEP:

3072:3072:pjNaly6K25gyi4x3gS6Y1TcVbrkijMziie:pAsah1wtLjMPe

Type de fichier

locky_packed: PE32 executable (GUI) Intel 80386, for MS Windows

Informations PE

- Date de compilation: 07:26:59 29/01/2002
- Machine cible: 0x14C Intel 386 et processeurs précédents compatibles
- Point d'entrée: 0x0040A344

Sections

Nom	Adresse virtuelle	Taille virtuelle	Taille original	MD5
.text	0x00401000	0xF28B	0xF400	009d0d91d06f2b87817
.rdata	0x00411000	0x60B8	0x6200	fd8ac6be745acedaca4
.data	0x00418000	0x1B64	0xE00	2eba3ead215cf9594aæ
.reloc	0x0041A000	0x21CA	0x2200	0107bbcaa901e0b260

Imports

- KERNEL32.dll
 - LeaveCriticalSection
 - GetCurrentThread
 - FindNextFileW
 - GetDiskFreeSpaceExW
 - GetVolumeInformationW

- GetLogicalDrives
- GetDriveTypeW
- EnterCriticalSection
- LoadLibraryW
- HeapReAlloc
- DeleteCriticalSection
- InitializeCriticalSection
- GetSystemTime
- GetTempFileNameW
- CreateProcessW
- GetModuleHandleA
- GetProcAddress
- GetCurrentProcess
- FindClose
- GetVolumeNameForVolumeMountPointA
- GetWindowsDirectoryA
- GetLocaleInfoA
- FindFirstFileW
- MultiByteToWideChar
- WideCharToMultiByte
- WaitForSingleObject
- CreateThread
- CopyFileW
- GetTempPathW
- Sleep
- GetUserDefaultUILanguage
- GetUserDefaultLangID
- GetSystemDefaultLangID
- SetUnhandledExceptionFilter
- SetErrorMode
- MulDiv

- GetVersionExA
- ExitProcess
- GetModuleFileNameW
- GetLastError
- FlushFileBuffers
- SetFileTime
- GetSystemTimeAsFileTime
- SetFilePointer
- ReadFile
- SetFileAttributesW
- GetFileAttributesExW
- DeleteFileW
- MoveFileExW
- WriteFile
- GetFileSizeEx
- CreateFileW
- CloseHandle
- RtlUnwind
- GetCurrentProcessId
- GetTickCount
- QueryPerformanceCounter
- GetFileType
- InitializeCriticalSectionAndSpinCount
- SetHandleCount
- GetEnvironmentStringsW
- FreeEnvironmentStringsW
- GetModuleFileNameA
- GetStringTypeW
- LCMapStringW
- HeapCreate
- GetStdHandle

- TerminateProcess
- IsDebuggerPresent
- UnhandledExceptionFilter
- GetCurrentThreadId
- SetLastError
- TlsFree
- TlsSetValue
- TlsGetValue
- TlsAlloc
- HeapAlloc
- HeapFree
- GetCommandLineA
- HeapSetInformation
- GetStartupInfoW
- RaiseException
- IsProcessorFeaturePresent
- HeapSize
- GetModuleHandleW
- GetCPInfo
- InterlockedIncrement
- InterlockedDecrement
- GetACP
- GetOEMCP
- IsValidCodePage
- USER32.dll
 - DrawTextW
 - SystemParametersInfoW
 - ReleaseDC
 - FrameRect
 - FillRect
 - GetSystemMetrics

- GetDC
- GDI32.dll
 - SetTextColor
 - GetDIBits
 - GetObjectA
 - SetBkMode
 - CreateSolidBrush
 - CreateCompatibleBitmap
 - SelectObject
 - CreateFontA
 - DeleteObject
 - GetDeviceCaps
 - CreateCompatibleDC
 - DeleteDC
- ADVAPI32.dll
 - CryptGetHashParam
 - AccessCheck
 - MapGenericMask
 - DuplicateToken
 - OpenThreadToken
 - GetFileSecurityW
 - CryptHashData
 - SetTokenInformation
 - OpenProcessToken
 - CryptDestroyHash
 - CryptCreateHash
 - RegSetValueExW
 - RegQueryValueExA
 - RegDeleteValueA
 - RegSetValueExA
 - RegCreateKeyExA

- RegCloseKey
- RegOpenKeyExA
- CryptAcquireContextA
- CryptGenRandom
- CryptReleaseContext
- CryptEncrypt
- CryptSetKeyParam
- CryptImportKey
- CryptDestroyKey
- SHELL32.dll
 - SHGetFolderPathW
 - ShellExecuteW
- WININET.dll
 - InternetOpenA
 - InternetCloseHandle
 - InternetSetOptionA
 - HttpOpenRequestA
 - InternetQueryOptionA
 - HttpSendRequestExA
 - InternetWriteFile
 - HttpEndRequestA
 - HttpSendRequestA
 - HttpQueryInfoA
 - InternetCrackUrlA
 - InternetReadFile
 - InternetConnectA
- MPR.dll
 - WNetEnumResourceW
 - WNetCloseEnum
 - WNetAddConnection2W
 - WNetOpenEnumW

- NETAPI32.dll
 - DsRoleGetPrimaryDomainInformation
 - DsRoleFreeMemory

II-Unpack

Le fichier unpacké étant disponible et l'analyse du fonctionnement du malware étant la priorité, la procédure de dépackage sera réalisée et expliquée dans l'avenir.

III-Processus d'infection

Vue globale

Locky est un ransomware. Son but est de chiffrer les fichiers personnels de l'utilisateur d'un poste infecté. Par la suite, des instructions sont données à l'utilisateur pour lui expliquer comment payer et récupérer ses données. Les instructions sont présentées grâce à un fond d'écran et des fichiers textes dans le système de fichiers.

Une vue globale du processus d'infection analyse couvert par l'analyse est présenté dans l'illustration 1.

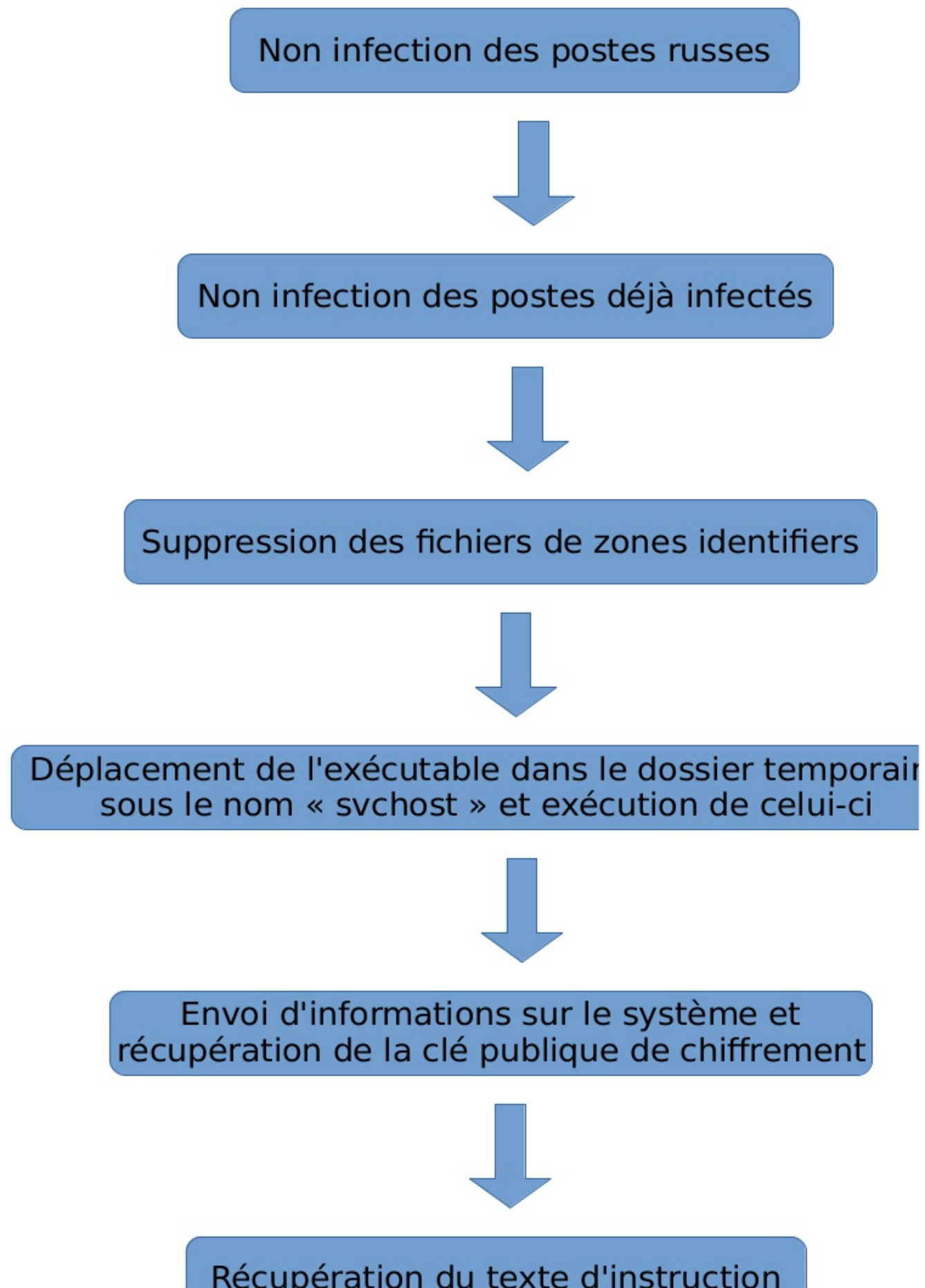


Illustration 1: Vue d'ensemble du processus d'infection

Le reste de ce chapitre présente les actions réalisées par le malware dans l'ordre chronologique.

Initialisation

Désactivation de la virtualisation

Le malware commence par changer les propriétés du token de son processus. Il désactive la "virtualisation". Cela permet au malware d'accéder aux fichiers et aux clés de registres globaux à la machine et non restreint à l'utilisateur?.

```
lea    eax, [ebp+accesstoken]
push   eax                ; TokenHandle
xor     ebx, ebx
push   TOKEN_ADJUST_DEFAULT ; DesiredAccess
                        ; Required to change default owner, primary group or DACL of access token
mov     [ebp+accessTokenInformation], ebx ; accessTokenInformation = 0
call    ds:GetCurrentProcess ; Get handle to the current process
push   eax                ; ProcessHandle
call    ds:OpenProcessToken ; Get current process access token
test    eax, eax
jz      short getAccessTokenFailed
```

```
push   INT_LENGTH        ; TokenInformationLength
lea     eax, [ebp+accessTokenInformation]
push   eax                ; TokenInformation
push   TokenVirtualizationEnabled ; TokenInformationClass
push   [ebp+accesstoken] ; TokenHandle
call    ds:SetTokenInformation ; unsetVirtualizationToThisProcessus
push   [ebp+accesstoken] ; hObject
call    ds:CloseHandle    ; stopUseAccessToken
```

Illustration 2: Désactivation de la virtualisation du processus

Désactivation des redirections WoW64

Ensuite le malware désactive les redirections WoW64 (Windows 32-bits on Windows 64-bits) du système de fichiers. Cela enlève les redirections transparentes vers les dossiers de compatibilité 32-bits sur les systèmes 64-bits.



Illustration 3: Désactivation des redirections WoW64

Initialisation de la liste des adresses IP de C&C

Le malware contient dans sa configuration une liste d'adresse IP de C&C à contacter. Lors de cette étape, il crée un vecteur contenant chacun de ces adresses en vue de les utiliser plus tard.

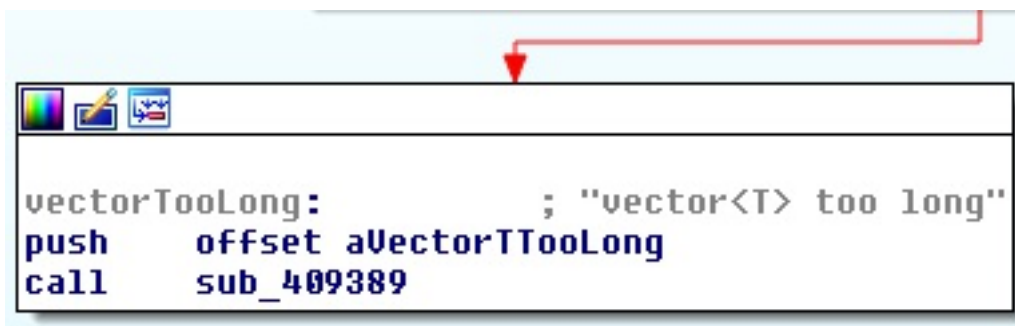
```

:004137E0 configuration    T_configuration <3, 7, 1Eh, 0, 0, \
:004137E0                                ; DATA XREF: getPubkey+298↑r
:004137E0                                ; WinMain(x,x,x,x):listIpAddressesEmpty↑r ...
:004137E0                                '31.41.47.37,188.138.88.184,91.121.97.170,5.34.183.136'>

```

Illustration 4: Configuration du sample

Nous voyons ici la structure des données de configuration dont le troisième champs est une liste des adresses IP de C&C séparés par des virgules. Ce sont ces adresse IP qui sont extraites et rentrées dans une structure de données de type vector (Vraisemblablement une structure standard au langage de programmation utilisé).



```
vectorTooLong:                ; "vector<T> too long"
push    offset aVectorTTooLong
call    sub_409389
```

Illustration 5: Trace d'utilisation de structure de données de type vector

Vérification de non infection de postes russes

Le malware vérifie à travers trois paramètres du systèmes s'il n'est pas sur un poste russe (La langue du système, la langue de l'utilisateur et la langue de l'interface graphique). Si pour l'un, il s'avère que c'est le cas, le malware n'effectue pas la procédure de chiffrement.

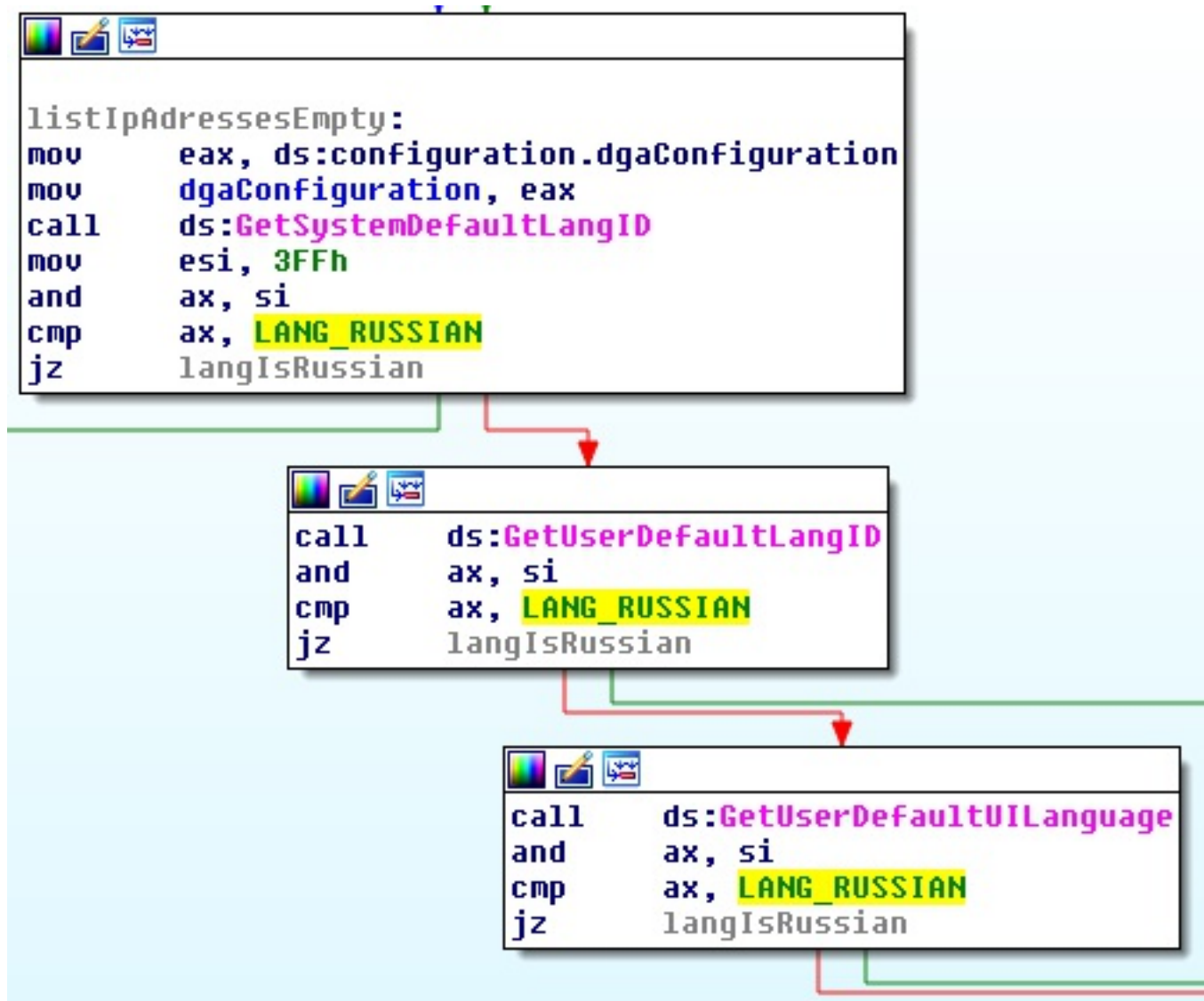


Illustration 6: Vérification de la non infection de poste russe (Vue précise)

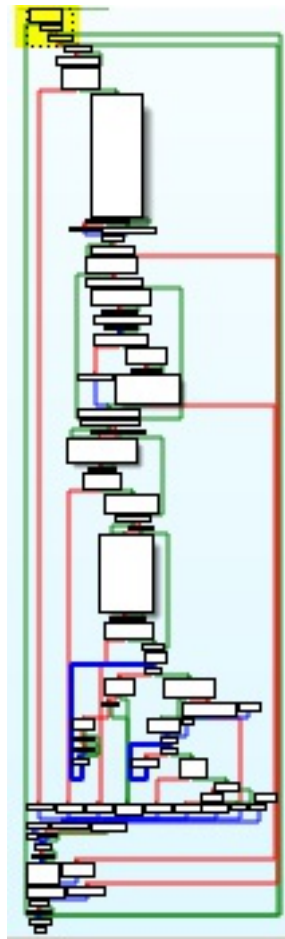


Illustration 7: Vérification de la non infection de poste russe (Vue macro)

L'illustration 7 montre que le flux d'exécution est dévié en fin de programme si le poste est Russe (Trois traits verts partant de la zone jaune).

Ceci permet de configurer le système pour le protéger d'une infection (cf Partie V).

Attente avant l'activation du malware

Le malware contient dans sa configuration une donnée qui définit le temps qu'il attendra avant de se déclencher. Il pourra ainsi attendre entre 0 et 9 heures pour s'activer.

```
004137E0 configuration    T_configuration <3, 7, 1Eh, 0, 0, \
004137E0                                ; DATA XREF: getPubkey+298↑r
004137E0                                ; WinMain(x,x,x,x):listIpAdressesEmpty↑r ...
004137E0                                '31.41.47.37,188.138.88.184,91.121.97.170,5.34.183.136'>
```

Illustration 8: Configuration du malware et temps d'attente

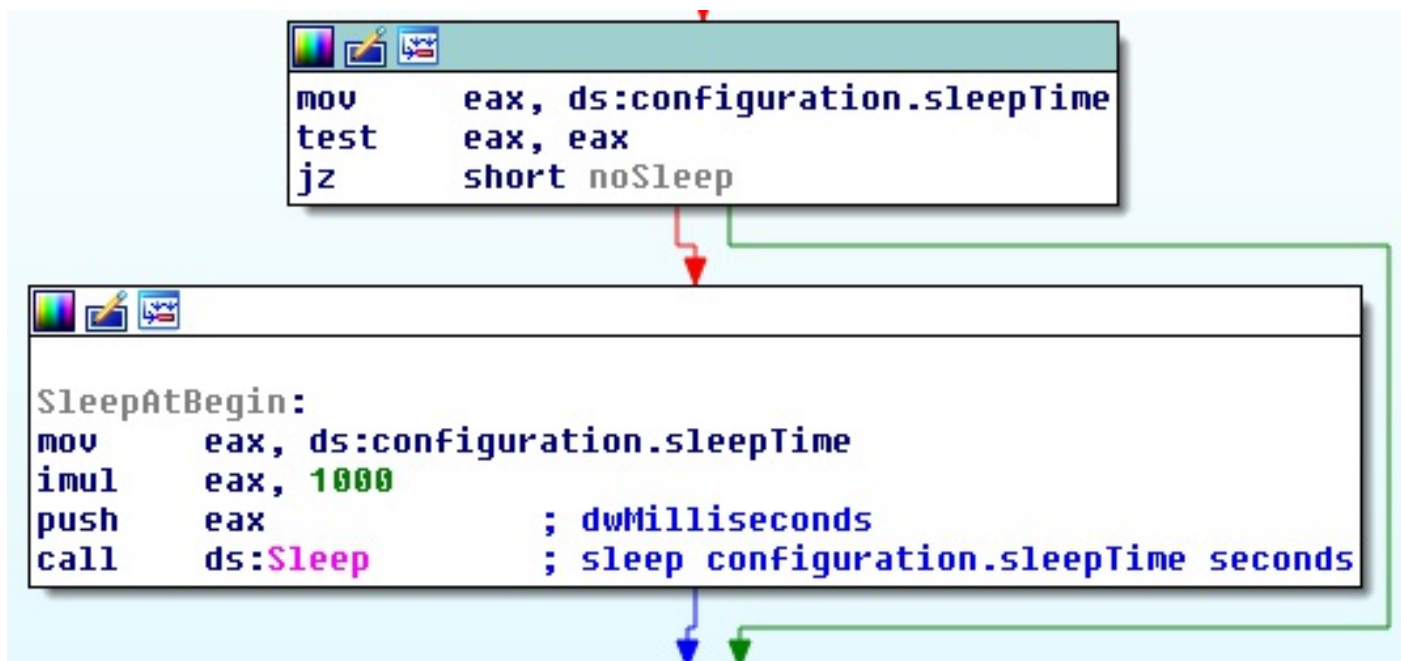


Illustration 9: Attente

Le sample étudié attendra 30 secondes.

Ouverture de la clé de registre principale

Le malware utilise une clé de registre du nom de "Locky" dans HKEY_USER\Software. Il y stocke l'identifiant de la victime, le texte d'explication pour la rançon, la clé publique pour le chiffrement et un marqueur de réalisation passée de l'attaque. Lorsqu'il ouvre la clé principale, si une erreur apparaît, il n'effectue pas le chiffrement.

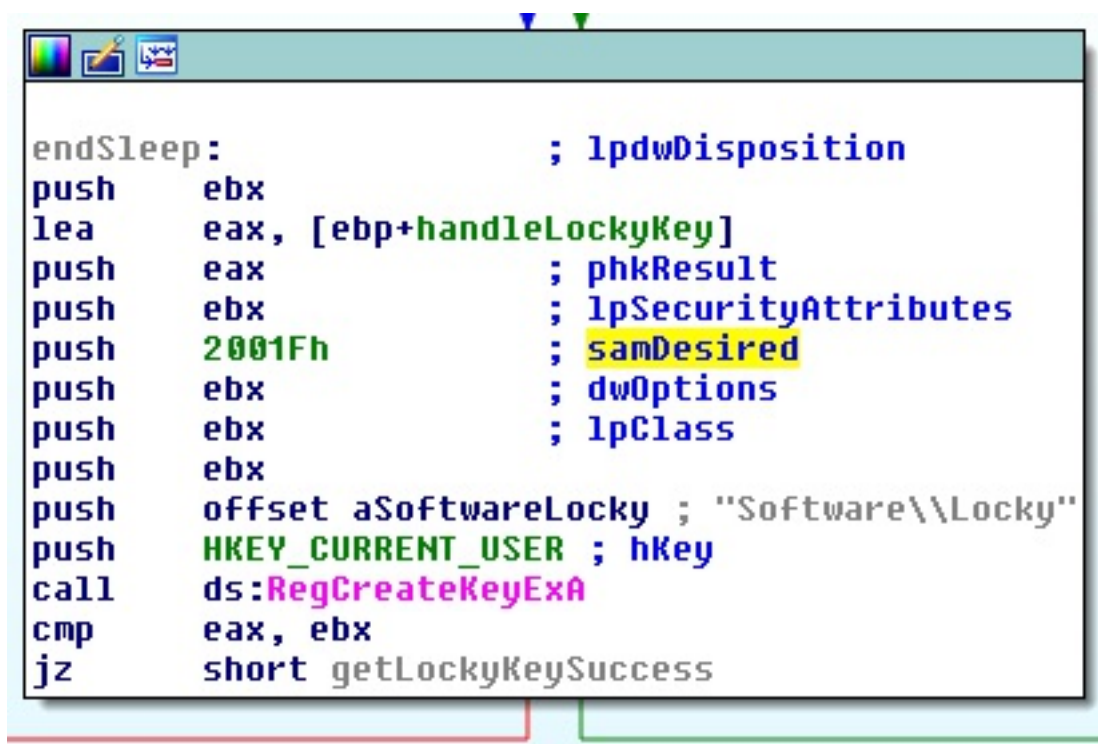


Illustration 10: Ouverture de la clé HKEY_USER\Software\Locky (Vue précise)

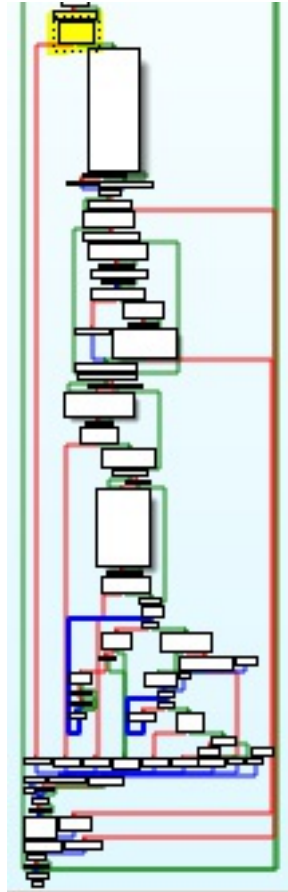


Illustration 11: Ouverture de la clé HKEY_USER\Software\Locky (Vue macro)

L'illustration 11 montre que si l'ouverture de la clé de registre n'est pas possible, le flux d'exécution est dévié jusqu'à la fin du programme (Trait rouge partant de la zone jaune).

Ceci permet de configurer le système pour le protéger d'une infection (cf Chapitre V).

Récupération des valeurs des sous-clés de registre

Locky prend les valeurs présentes dans les sous-clés de registre de clé publique, de texte d'explication, d'identifiant et le marqueur de fin avant de les sauvegarder dans des variables globales.

Calcul de l'identifiant de la victime

Le malware définit un identifiant à chaque victime suivant le GUID du disque contenant le système Windows. Celui-ci lui sert par la suite.

La procédure de génération d'identifiant est détaillée dans la partie IV-1.

Recherche des traces d'infection passée

Vérification de la validité du marqueur de fin

Le malware récupère la sous-clé, vérifie qu'il contient bien un entier. Si la clé n'existe pas, ne contient rien ou une valeur nulle, le malware s'exécutera. N'importe quelle valeur entière est donc un marqueur qui montre que Locky a déjà infecté le poste.

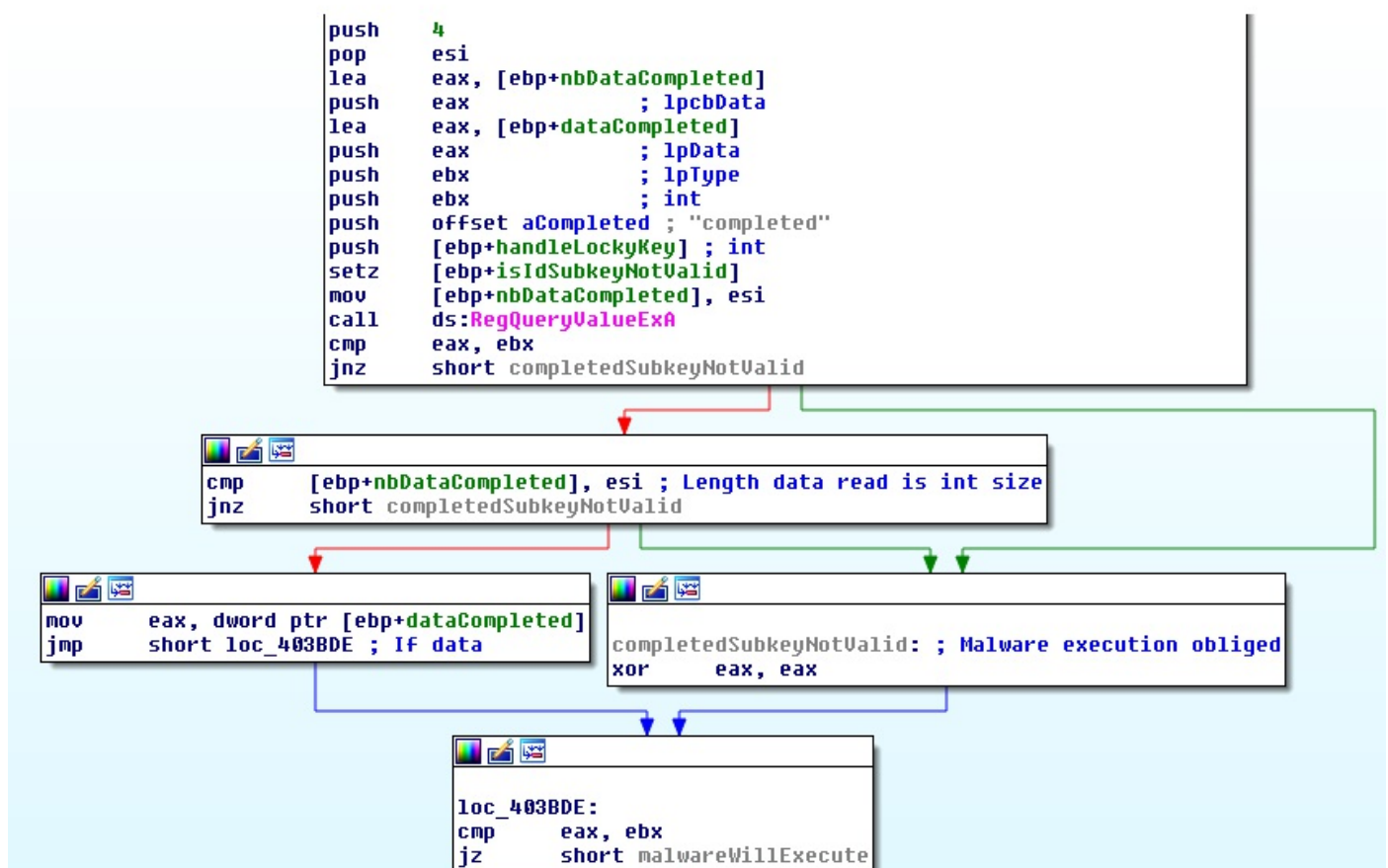


Illustration 12: Vérification de la validité de la sous-clé completed

Vérification de la validité de l'identifiant sauvegardé

Le malware compare la valeur de l'identifiant sauvegardé dans la sous-clé de registre id avec la valeur calculée et sauvegarde le résultat de la

comparaison dans une variable locale.

```
mov     edi, offset stringStrct_idSubkeyValue
lea     eax, [ebp+stringStrct_computedValueIdentifiant]
mov     edx, edi
mov     byte ptr [ebp+var_4], 9
call    compareStringStrct ; Compare id computed and id subkey value
xor     ebx, ebx
cmp     al, bl
push    4
pop     esi
lea     eax, [ebp+nbDataCompleted]
push    eax ; lpcbData
lea     eax, [ebp+dataCompleted]
push    eax ; lpData
push    ebx ; lpType
push    ebx ; int
push    offset aCompleted ; "completed"
push    [ebp+handleLockyKey] ; int
setz    [ebp+isIdSubkeyNotValid] ; 1 if idSubkey.value != valueComputed
; 0 else
```

Illustration 13: Comparaison des identifiants sauvegardés dans la sous-clé de registre et celui calculé

Ensuite si le marqueur de fin est valide, le malware effectue une deuxième vérification d'égalité entre les deux identifiants. Le flux d'exécution contourne toute la charge malicieuse en cas de validité de l'identifiant de la sous-clé.

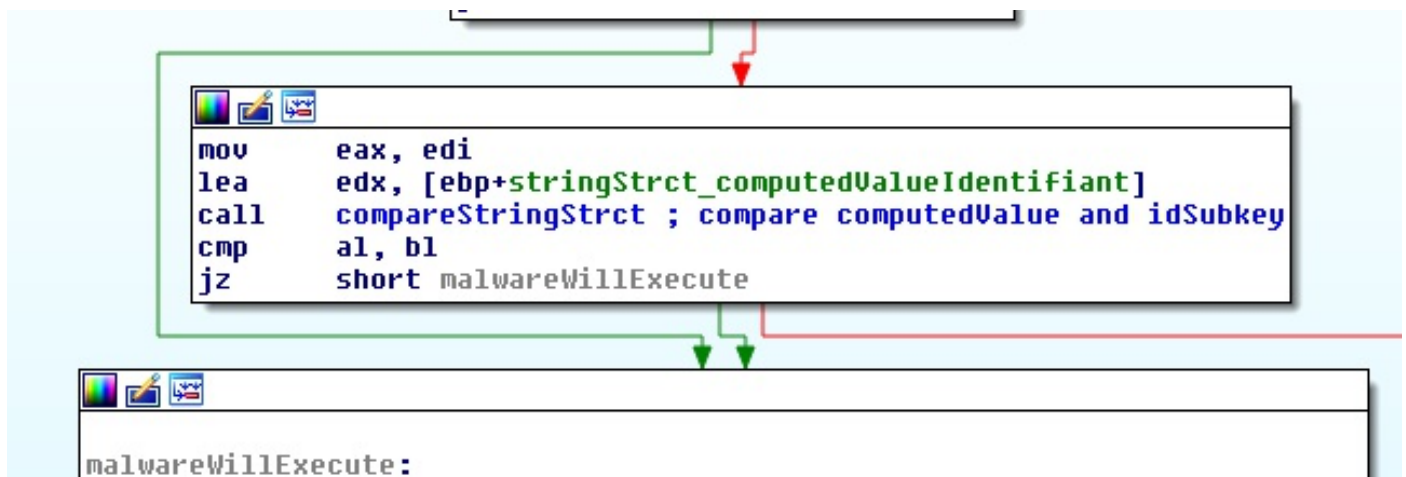


Illustration 14: Vérification de la validité de la sous-clé id

Contournement de la charge utile

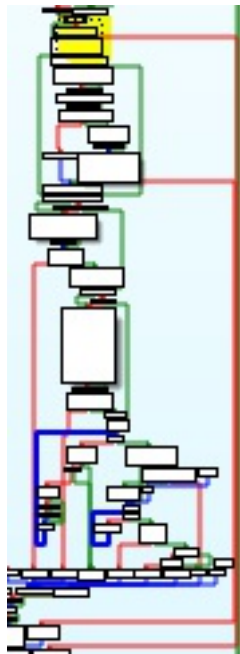


Illustration 15: Vérification de la validité de la sous-clé id (Macro)

L'illustration 15 montre que si les clés completed et id sont valides, le flux d'exécution est jusqu'à la fin du programme (Trait rouge partant du carré jaune).

Furtivité de l'exécution

Pour améliorer la furtivité, le malware se déplace dans un dossier temporaire sous le nom svchost.exe.

Choix de l'activation ou non du déplacement de l'exécutable

La procédure est conditionnée par la configuration du sample. Un champs de la configuration permet de contourner le déplacement de l'exécutable si cette valeur est nulle.

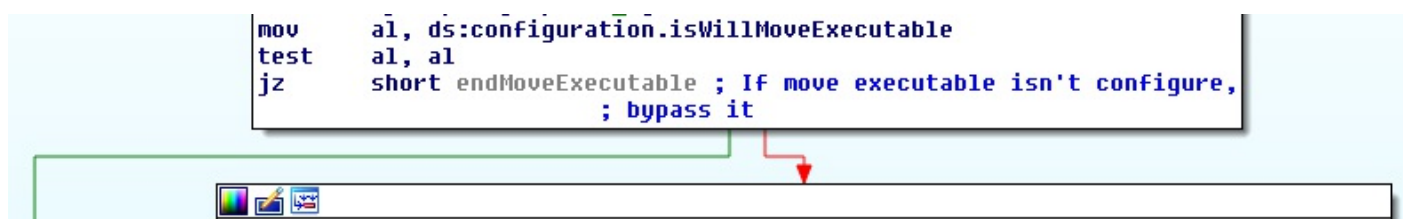


Illustration 16: Contournement du processus de furtivité suivant la configuration

```

004137E0 configuration    T_configuration <3, 7, 1Eh, 0, 0, \
004137E0                ; DATA XREF: getPubkey+298↑r
004137E0                ; WinMain(x,x,x,x):listIpAdressesEmpty↑r ...
004137E0                '31.41.47.37,188.138.88.184,91.121.97.170,5.34.183.136'>

```


Illustration 17: Configuration de la furtivité

Dans le sample étudié, la configuration désactive le déplacement de l'exécutable.

Récupération de l'actuel et du futur emplacements de l'exécutable

En premier, l'emplacement actuel de l'exécutable est récupéré via un simple appel à l'API Windows.

Ensuite, le chemin du futur exécutable est créé en concaténant le chemin du dossier temporaire et la chaîne "svchost.exe".

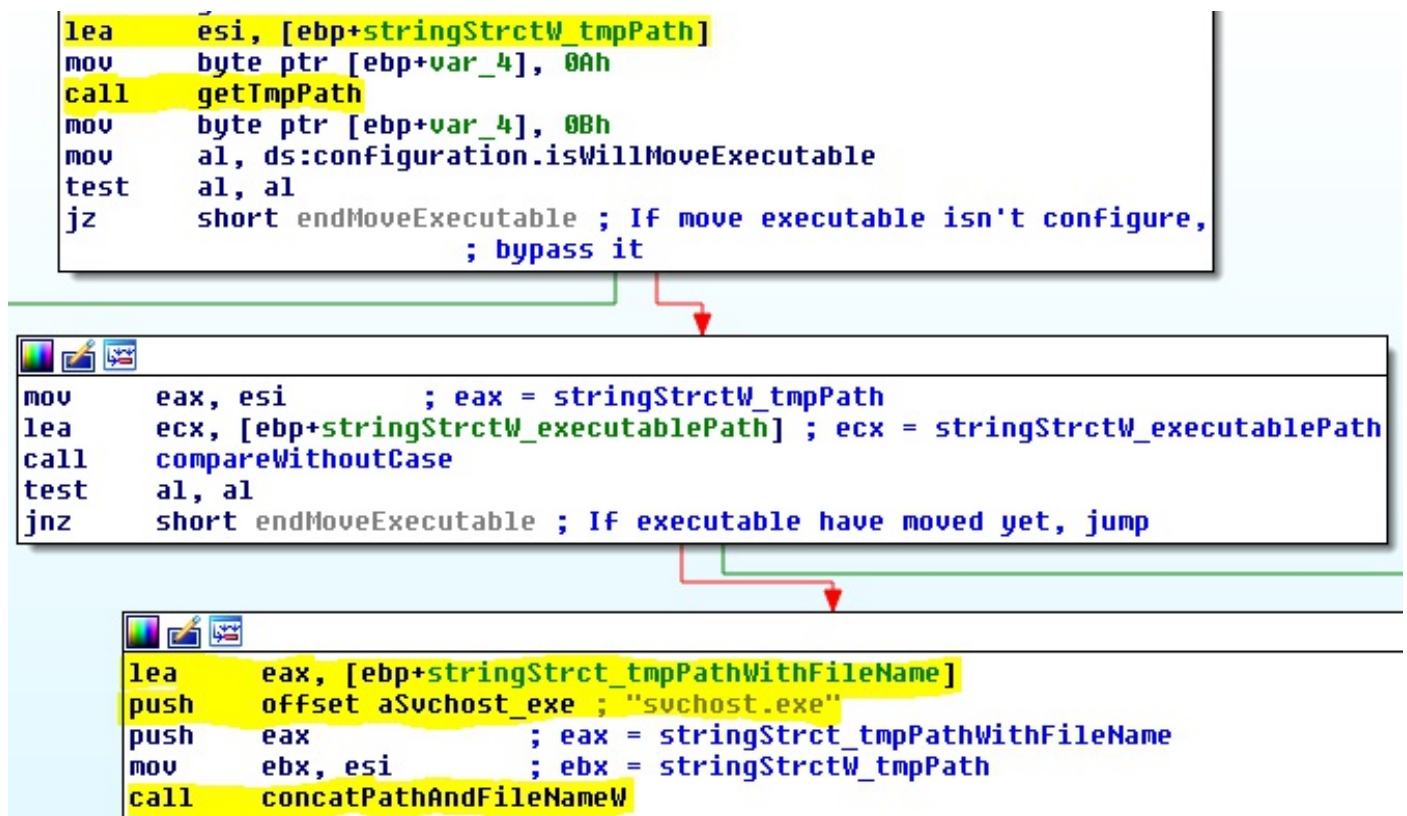
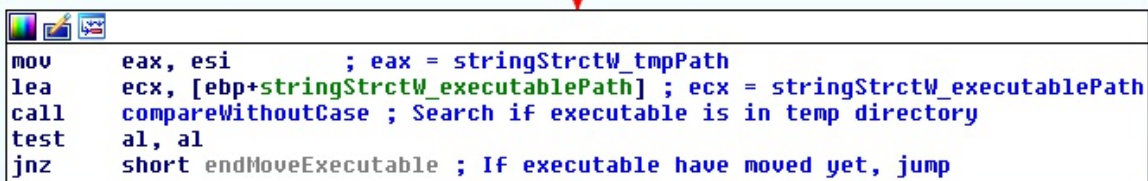


Illustration 18: Création du chemin de destination

Vérification de l'emplacement de l'exécutable

Le malware fait une vérification pour savoir s'il est déjà placé dans le dossier temporaire donc à l'emplacement voulu. Il compare le chemin de l'exécutable et celui du dossier temporaire. La comparaison s'arrête à la longueur du dossier temporaire pour exclure le nom du fichier. Si l'exécutable est déjà dans le bon dossier, la procédure de déplacement

n'est pas activée.



```
mov     eax, esi      ; eax = stringStrctW_tmpPath
lea     ecx, [ebp+stringStrctW_executablePath] ; ecx = stringStrctW_executablePath
call    compareWithoutCase ; Search if executable is in temp directory
test    al, al
jnz     short endMoveExecutable ; If executable have moved yet, jump
```

Illustration 19: Vérification que l'exécutable n'est pas dans le dossier temporaire

Copie de l'exécutable

Le fichier exécutable du malware est copié dans le dossier temporaire sous le nom "svchost.exe"

Suppression du fichier de zone identifier

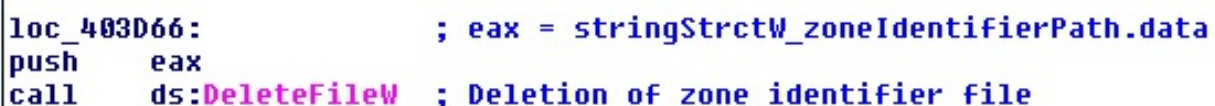
Avant le lancement du nouvel exécutable, le malware supprime le fichier zone identifier qui ferai apparaitre un message d'avertissement à l'utilisateur.



```
lea     eax, [ebp+stringStrctW_zoneIdentifierPath]
push    offset aZone_identifie ; ":Zone.Identifier"
push    eax ; int
lea     ebx, [ebp+stringStrctW_tmpPathWithFileName]
call    concatPathAndFileNameW
cmp     [eax+T_strStrctW.spaceReserved], LIMIT_MAX_DATAW_INTERNAL
pop     ecx
pop     ecx
jnb     short loc_403D66
```



```
mov     eax, dword ptr [eax+T_strStrct.data]
```



```
loc_403D66: ; eax = stringStrctW_zoneIdentifierPath.data
push     eax
call     ds:DeleteFileW ; Deletion of zone identifier file
```

Illustration 20: Suppression du fichier de zone identifier

Lancement du nouvel exécutable

Le fichier exécutable copié, le fichier de zone identifier supprimé, le malware peut se relancer sans éveiller les soupçons de l'utilisateur et se faire passer pour le processus légitime svchost.

Arret du processus parent

Si le nouveau processus s'est bien lancé, le processus parent va en fin de programme et laisse la main au fils.

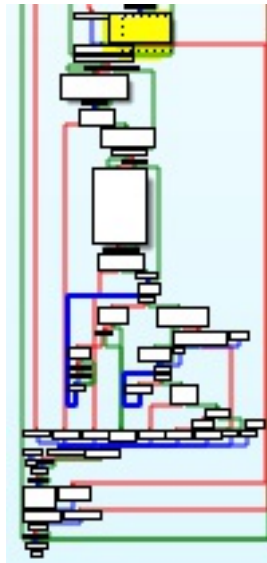


Illustration 21: Arret du processus parent

IV-Détails de fonctionnement

Génération de l'identifiant de victime

- Récupération du chemin du dossier Windows
- Recherche du nom du volume du point de montage
- Extraction du GUID du volume (Global Unique Identifier)
- Hash MD5 du GUID
- Transformation du hash en caractères hexadécimaux (Majuscules)
- Selection des 16 premiers caractères du hash

Sources

[MSDN](#)

[Wikipedia](#)