STREAMSCAN

ANALYSE

RANCONGICIEL MEDUSALOCKER

NOV 2025 //

Introduction

Le rançongiciel **MedusaLocker** est apparu pour la première fois en septembre 2019. Il cible principalement le secteur de la santé, mais a également été observé dans d'autres secteurs.

Il existe plusieurs variantes de MedusaLocker utilisant différentes extensions pour crypter les fichiers. Il fonctionne selon un modèle de Ransomware-as-a-Service (RaaS) qui utilise les algorithmes RSA et AES pour crypter les fichiers.

Nous avons réalisé une rétro-ingénierie de MedusaLocker, et nous avons constaté qu'il utilise l'algorithme **Chacha20** pour chiffrer les données.

Compte tenu de sa structure de configuration modifiée, de son mécanisme de persistance mis à jour et de ses nouvelles fonctionnalités intégrées par rapport aux variantes précédentes de MedusaLocker, la version que nous avons analysée peut-être classée comme une version 3 de la famille de rancongiciels MedusaLocker.

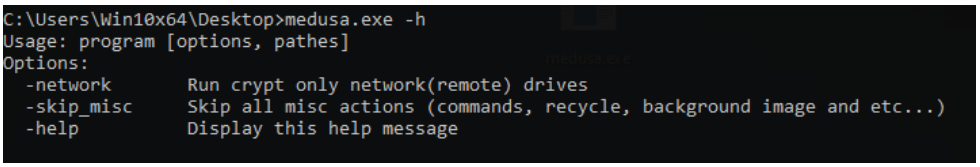


Figure1 : Paramètres de ligne de commande

Paramètres de ligne de commande

Paramètres	Info
-network	Crypter uniquement les lecteurs réseau/distants
-skip_misc	Ignorer toutes les actions diverses • Commandes avant et après • Vider les corbeilles • Définir l'image d'arrière-plan • Ajout d'une clé de registre Autorun • Arrêt de explorer.exe lorsque VK_HOME (touche Windows) est enfoncé • Remplacer tout l'espace inutilisable
-help	Afficher les options d'aide

Retro-ingénierie du rancongiciel MedusaLocker

Details sur la variante MedusaLocker analysée

Paramètres	Info
Hash MD5	8ba3a306f5550374490030ea472f2f92
Hash SHA-256	6d000a159fe10af1b29ddf4e4015931a9e9d0a020aeef0c602d8c5419b5966e6
Taille du fichier	738.00 KB (755712 bytes)
Date de création	2025-05-17 11:10:52 UTC

Configuration

La plupart des rançongiciels modernes ont tendance à utiliser des configurations pour personnaliser leur comportement et s'adapter à leurs besoins. MedusaLocker en fait partie.

Au départ, la configuration est chiffrée à l'aide du chiffrement par flux Chacha20. Cela nécessite la clé et le nonce pour déchiffrer la configuration intégrée dans les données de ressources.

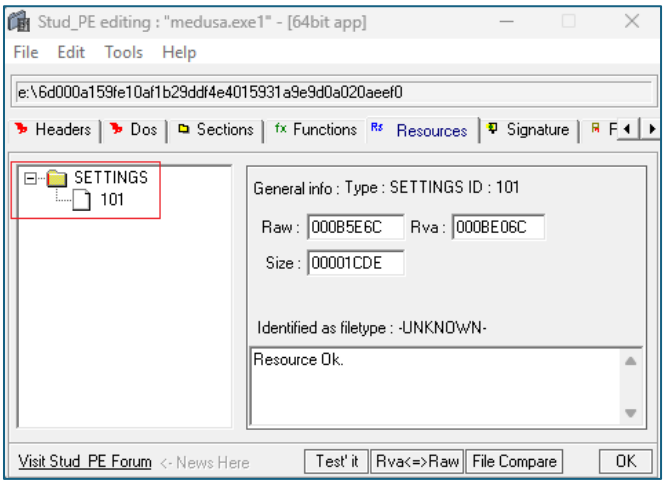


Figure2 : Données intégrées dans la section Resource avec une taille de 0x1CDE

```
size[0] = 0;
sub_140003E70(&___buf);
__copyS(_pointer, L"SETTINGS");
__readResource(&___buf, v0, _pointer);
__releaseObject(_pointer);
__initSalsa20(v194, "TRUMPTRUMPTRUMPTRUMPTRUMPTRUMPTRUMP", "PUTLERPUTLER");
__decryptStream(v194, ___buf, v192 - ___buf);
v1 = sub_140008FD0(v177);
sub_14000AC20(v1, "--------------------------------");
```

Figure3 : Pseudocode qui lit et décrypte la configuration à partir de la section Ressource

Cela permettra de lire le contenu du type de ressource défini par l'utilisateur « SETTINGS » où se trouvent les données intégrées. Une fois la lecture effectuée, le chiffrement par flux Chacha20 est initialisé à l'aide de la clé « **TRUMPTRUMPTRUMPTRUMPTRUMPTRUMPTRUMP** » tronquée à 32 caractères, puis celle avec **« PUTLERPUTLER** » qui sera tronquée à 8 caractères seulement.

Nous continuons à pousser nos recherches pour comprendre pourquoi ce gang de rançongiciel a décidé d'utiliser les mots TRUMP et PUTLER (qui selon certaines interprétations fait référence à Vladimir PUTIN comparé à Hitler). Nous restons convaincus que ce n'est pas anodin. Nous publierons très certainement un article à ce sujet.

Address	Hex	ASCII
000000A7766FF200	65 78 70 61 6E 64 20 33 32 2D 62 79 74 65 20 6B	expand 32-byte k
000000A7766FF210	54 52 55 4D 50 54 52 55 4D 50 54 52 55 4D 50 54	TRUMPTRUMPTRUMPT
000000A7766FF220	52 55 4D 50 54 52 55 4D 50 54 52 55 4D 50 54 52	RUMPTRUMPTRUMPTR
000000A7766FF230	00 00 00 00 00 00 00 00 50 55 54 4C 45 52 50 55	........PUTLERPU

Figure4 : Initialisation du bloc de clés Chacha20 avec la constante « expand 32-byte k »

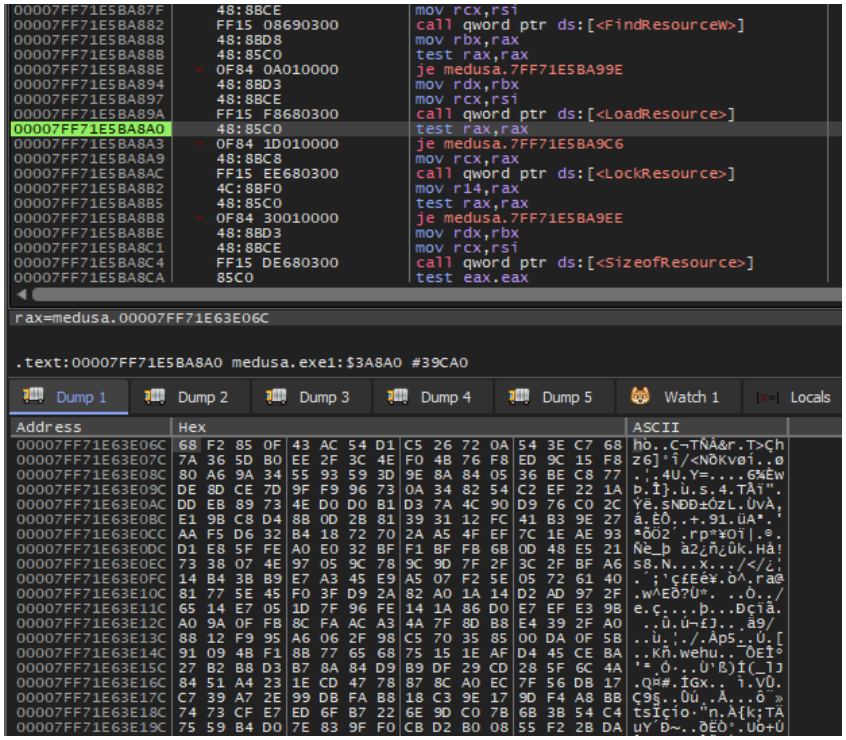


Figure5 : Les codes d'assemblage sont destinés à trouver et à lire le contenu des données de ressources. Vous trouverez ci-dessous les données de ressources non chiffrées dans le vidage de mémoire.

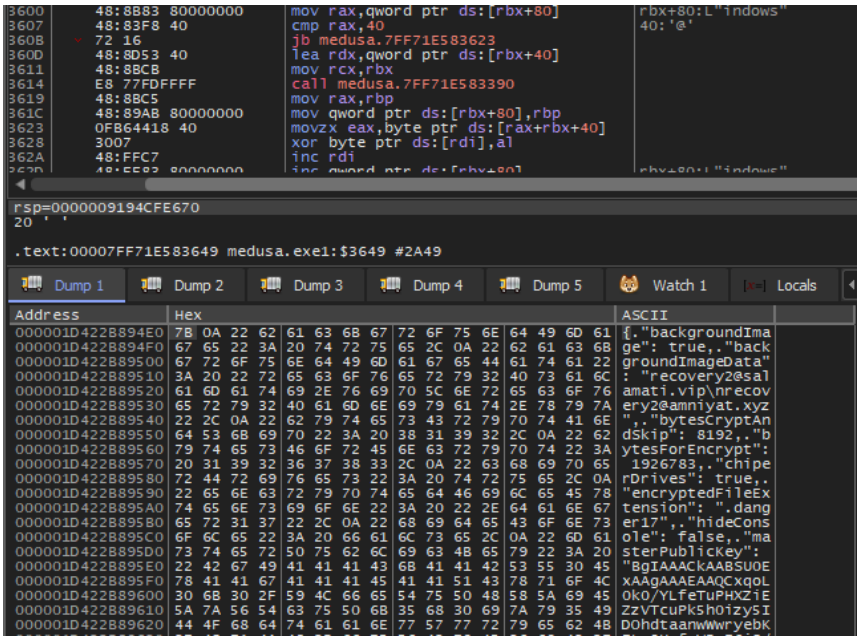


Figure6 : Configuration décryptée révélant la structure JSON intégrée

Figure7 : aperçu de la configuration JSON

Clés de configuration

Clé	Info
backgroundImage : booléen	Indicateur permettant de modifier l'arrière-plan du bureau
backgroundImageData : chaîne	Texte à afficher sur le bureau
bytesCryptAndSkip : nombre	Chiffrer le fichier en alternance (d'abord 0x2000, puis sauter les 0x2000 suivants, et ainsi de suite)
chiperDrives : booléen	Mettre à zéro les octets inutilisables dans les lecteurs afin d'empêcher la récupération des fichiers à l'aide d'outils
encryptedFileExtension : chaîne	Extension pour fichier crypté
hideConsole : booléen	Masquer la fenêtre de console
masterPublicKey : chaîne	Clé principale pour crypter la clé privée
openRequirementsOnFinish : booléen	Drapeau à l'ouverture de la note de rançon à la fin
postRunCommands : tableau	Commandes à la fin
preRunCommands : tableau	Commandes initiales
regenerateKeyAlways : booléen	Indique si les clés doivent toujours être régénérées
removeRecycle : booléen	Indicateur pour vider la corbeille
requirementsFileDataUTF8 : chaîne	Contenu de la demande de rançon

requirementsFileName : chaîne	Nom du fichier de la demande de rançon
skipExtensions : tableau	Liste des extensions autorisées
skipPathes : tableau	Liste des répertoires autorisés
startPathes : tableau	Chemin initial
threadPool : booléen	Indicateur de priorité
threadPoolMaxThreads : nombre	Nombre de threads
ThreadPoolPriorityExtensions : tableau	Liste des extensions à prioriser

Une fois la configuration analysée avec succès, le système procède à la génération des clés RSA. Ces clés sont stockées dans le registre, mais seule la clé privée est cryptée à l'aide du standard DES (Data Encryption Standard), un algorithme de chiffrement par blocs symétrique.

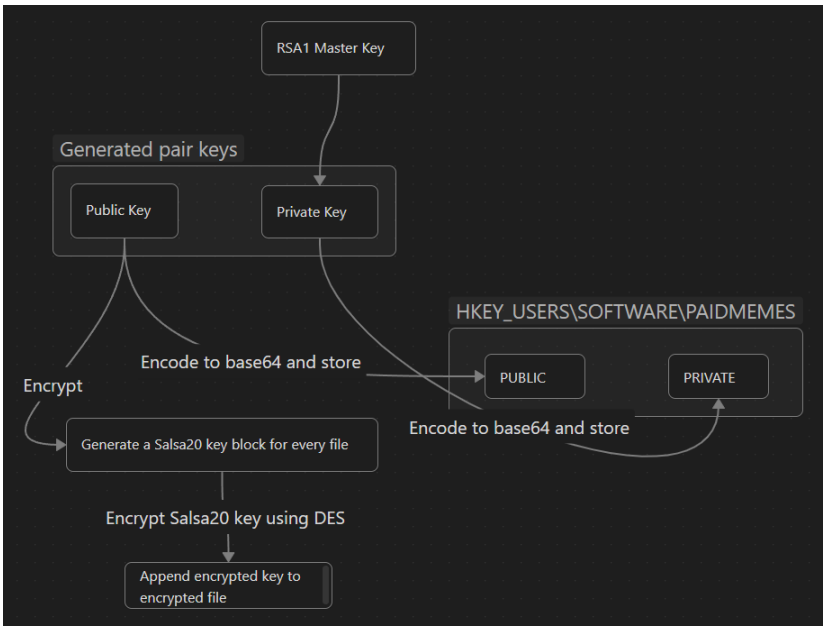


Figure8 : Hiérarchie des clés et flux de chiffrement illustrant un schéma hybride dans lequel une clé principale RSA protège les clés RSA générées, qui à leur tour sécurisent les clés de chiffrement Chacha20 par fichier. La clé privée est chiffrée avec DES et stockée dans le registre, tandis que la clé Chacha20 de chaque fichier est chiffrée et ajoutée au fichier chiffré pour une récupération ultérieure.

```
__generatePairKeys();
__encryptPrivateKey(v27, &__encryptedPrivateKey);
if ( *(&__encryptedPrivateKey + 1) - __encryptedPrivateKey != 0x500 )
{
  v9 = sub_1400096A0(&__publicKey);
  v10 = sub_140003710(&v16, v9);
  v11 = sub_14000B710(v19, "Bad private key size: ", v10);
  sub_140001BA0(pExceptionObject, v11);
  throw pExceptionObject;
}
```

Figure9 : Pseudocode pour la génération et le chiffrement de la clé privée

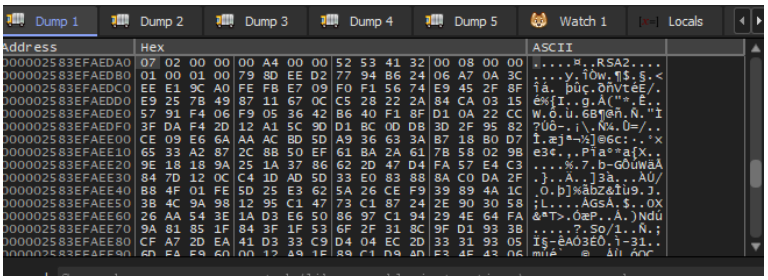


Figure10: Clé privée

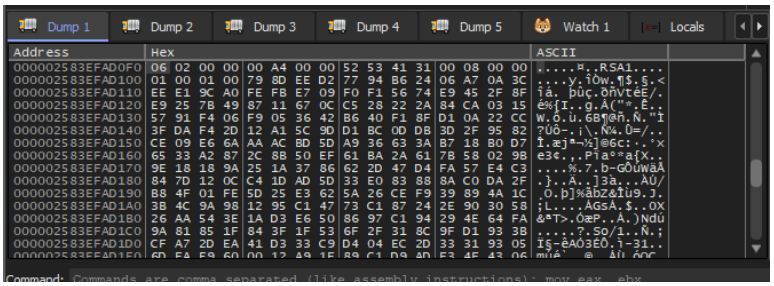


Figure11: Clé publique

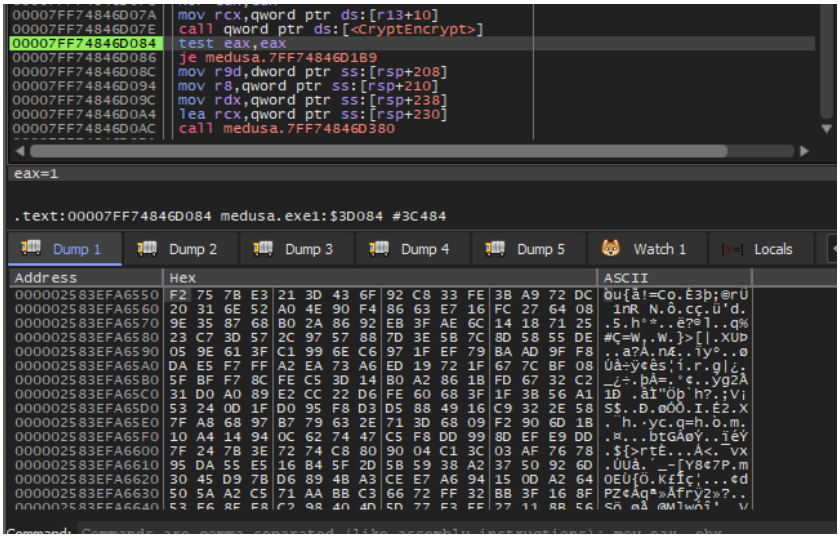


Figure12: CryptEncrypt() est utilisé pour chiffrer la clé privée

Cette routine crypte la clé privée en blocs de 0x100 octets à l'aide de CryptEncrypt() avec RSA de la clé principale, complétée si nécessaire. Cela permettra de valider que la taille totale du conteneur de clés cryptées est de 0x500 octets.

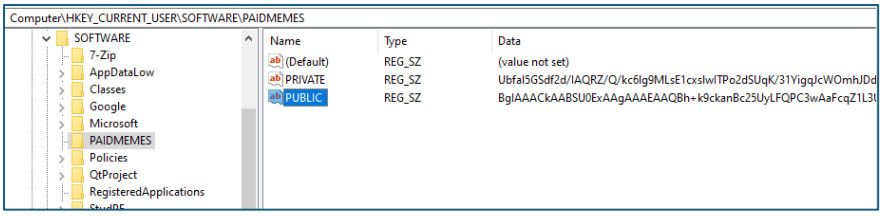


Figure13 : Stockage des paires de clés générées dans le registre

Ensuite, les clés de la paire seront encodées en Base64 à l'aide de l'API CryptStringToBinaryW et stockées dans le registre **HKEY_USERS\SOFTWARE**\PAIDMEMES avec les noms de valeur PUBLIC et PRIVATE pour les deux clés.

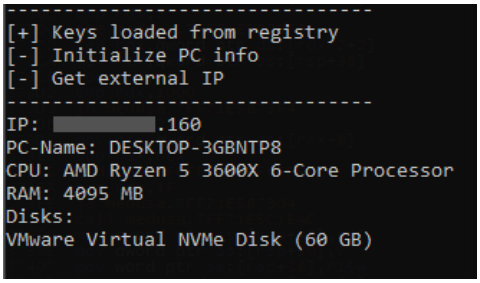
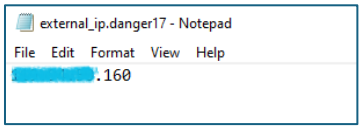


Figure14 : Informations sur la machine victime

Ensuite, les informations sur la machine victime seront collectées et cryptées à l'aide de la clé principale. Cela servira à identifier l'appareil infecté. Les informations suivantes sont incluses :

- Le nom de l'ordinateur est obtenu à l'aide de *GetComputerNameA*
- Le nom de domaine est obtenu *à l'aide de GetComputerNameExA*
- Détermine le type de processeur en lisant la clé de registre *HKLM\HARDWARE\DESCRIPTION\System\CentralProcessor\0* sous le nom de valeur *ProcessorNameString*.
- L'adresse IP externe est obtenue en envoyant une requête à https://api.ipify.org avec *URLDownloadToFileW*. L'adresse IP est temporairement stockée dans le fichier nommé « external_ip.danger17 ».



- La taille de la RAM est obtenue à l'aide de *GlobalMemoryStatusEx*
- Les lecteurs physiques sont énumérés à l'aide de requêtes *DeviceIoControl* vers les propriétés IOCTL_STORAGE et la géométrie du lecteur IOCTL_DISK.

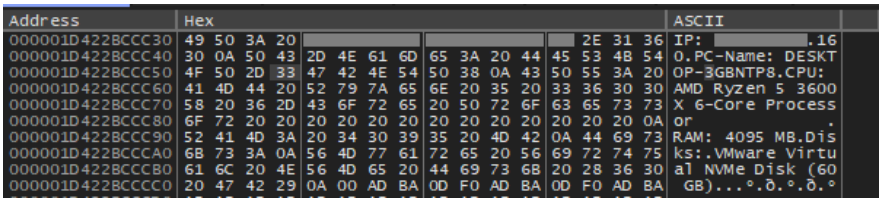


Figure15 : informations collectées sur la machine victime

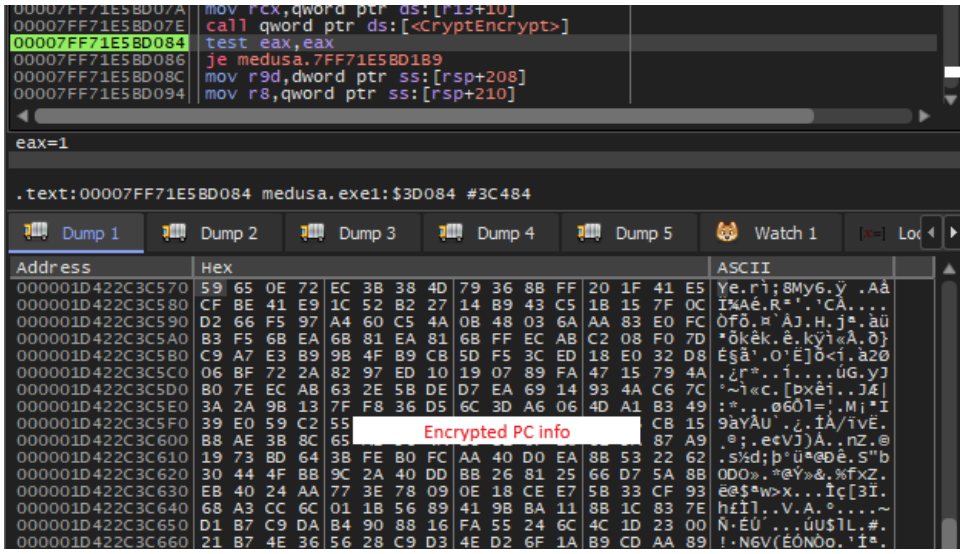


Figure16 : PC chiffré après appel de l'API CryptEncrypt

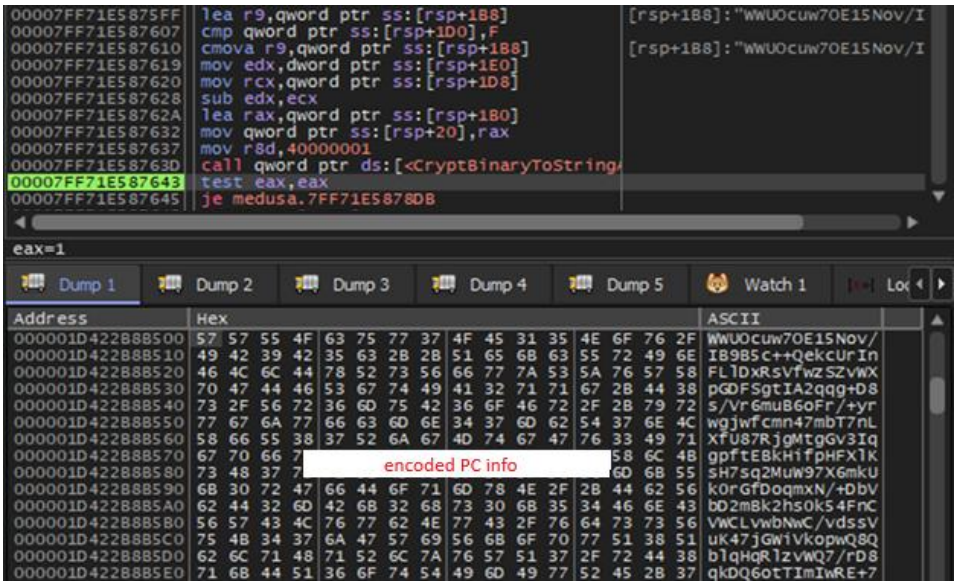


Figure17 : Encoding of encrypted information

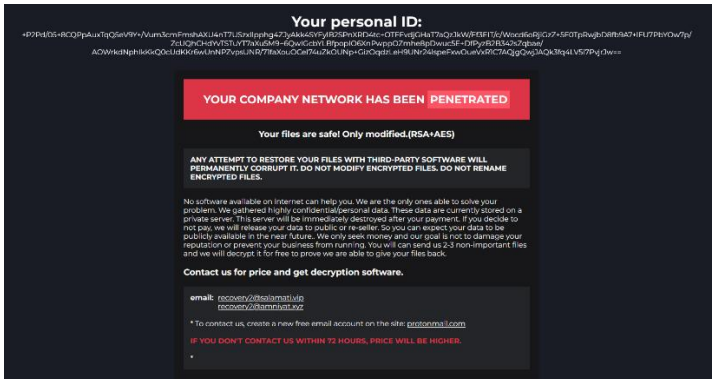


Figure18 : Les informations cryptées précédemment serviront d'identifiant à la victime. Notez que l'identifiant dans cette image est différent de celui de l'image précédente.

Une fois la configuration analysée avec succès, plusieurs threads seront créés avant l'exécution des précommandes.

Thread de routine de surveillance

L'un des threads créés sert à surveiller certaines touches virtuelles empêchant l'ouverture d'une fenêtre et le nombre d'utilisations du processeur afin de modifier dynamiquement le nombre de threads dans son pool de threads.

À l'aide de *GetAsyncKeyState*, si la touche INSERT est enfoncée, la fenêtre de console du logiciel malveillant sera activée (affichée ou masquée) en fonction des états actuels.

```
while ( !__stopFlag )
{
  secondTick = Xtime_get_ticks();
  minutesPassed = (secondTick - firstTick) / 600000000;
  if ( GetAsyncKeyState(VK_INSERT) && secondTick != thirdTick && secondTick >= thirdTick )
  {
    LODWORD(lParam) = GetCurrentProcessId();
    EnumWindows(EnumFunc, &lParam);
    v8 = IsWindowVisible(0LL);
    v9 = SW_SHOW;
    if ( v8 )
      v9 = SW_HIDE;
    ShowWindow(0LL, v9);
    thirdTick = secondTick + 2000000;
  }
```

Figure19 : Vérification de l'appui sur la touche INSERT.

Lorsque la touche HOME est enfoncée, le code vérifie si la fenêtre actuellement sélectionnée est la même fenêtre de console que celle du logiciel malveillant. Si c'est le cas, il exécute deux commandes à la suite.

1. taskill /f /im explorer.exe – force la fermeture de l'Explorateur Windows.
2. start explorer.exe – redémarre l'Explorateur.

```
if ( GetAsyncKeyState(VK_HOME) )            // Windows key
{
  fourthTick = Xtime_get_ticks();
  if ( fourthTick != fifthTick && fourthTick >= fifthTick )
  {
    ForegroundWindow = GetForegroundWindow();
    if ( ForegroundWindow == GetConsoleWindow() )
    {
      v43 = 0LL;
      v44 = 0LL;
      v45 = 0LL;
      __copy_s(&v43, L"taskkill /f /im explorer.exe", 0x1CuLL);
      v46 = 0LL;
      v47 = 0LL;
      v48 = 0LL;
      __copy_s(&v46, L"start explorer.exe", 0x12uLL);
      v57[0] = &v43;
      v57[1] = &v49;
      sub_140004530(&v40, v57, v12, v13);
      __executeCommands(&v40, v14, v15, v16);
      sub_140009330(&v40, v17, v18);
      `eh vector destructor iterator'(&v43, 0x20uLL, 2uLL, __releaseObject);
      fifthTick = Xtime_get_ticks() + 10000000;
    }
    cpuUsage = v36;
  }
}
```

Figure20 : si la touche HOME est enfoncée à l'aide de GetAsyncKeyState alors que la fenêtre de console du programme est sélectionnée, l'explorateur sera redémarré de force.

Enfin, il a la capacité d'ajuster dynamiquement le nombre de threads dans son pool de threads en fonction de l'utilisation du processeur.

- Si l'utilisation du processeur est trop élevée (>90 %), il réduit le nombre de threads.
- Si l'utilisation du processeur est faible (< 90 %) et n'atteint pas son niveau maximal, il augmente le nombre de threads.

Thread de création d'exécution automatique

S'il n'est pas élevé, le processus ajoute une entrée persistante sous la clé Autorun de l'utilisateur actuel (par exemple HKCU\Software\Microsoft\Windows\CurrentVersion\Run) avec le nom de valeur « BabyLockerKZ » et les données définies sur le chemin d'accès de l'exécutable.

```
__copy_s(lpSubKey, L"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", 0x2DuLL);
hKey = 0LL;
v0 = lpSubKey;
if ( v33.m128i_i64[1] > 7uLL )
  v0 = lpSubKey[0];
v1 = RegCreateKeyExW(HKEY_CURRENT_USER, v0, 0, 0LL, 0, 2u, 0LL, &hKey, 0LL);
```

Figure21 : ouverture de la clé de registre Autorun à l'aide de RegCreateKeyExW

```
*lpSubKey = 0LL;
v33 = 0uLL;
__copy_s(lpSubKey, L"BabyLockerKZ", 0xCuLL);
v25 = 2 * v44[0] + 2;
if ( v25 > 0xFFFFFFFF )
{
  sub_140001C80(v40, "Input size_t value is too big: size_t value doesn't fit into a DWORD.");
  throw v40;
}
filePath = lpData;
if ( v44[1] > 7uLL )
  filePath = lpData[0];
babyLockerKZ = lpSubKey;
if ( v33.m128i_i64[1] > 7uLL )
  babyLockerKZ = lpSubKey[0];
v28 = RegSetValueExW(hKeyAutorun, babyLockerKZ, 0, REG_SZ, filePath, v25);
if ( v28 )
{
  sub_140004150(v42, v28, "Cannot write string value: RegSetValueExW failed.");
  throw v42;
}
```

Figure22 : Ajout d'un nom de valeur « BabyLockerKZ » avec les données de son chemin d'accès.

Effacement du thread Corbeille

Ce thread garantit qu'aucun fichier ne sera restauré après le chiffrement à l'aide de la corbeille en appelant SHEmptyRecycleBinW sans boîte de dialogue de confirmation, sans interface utilisateur de progression et sans notification sonore.

```
v0 = SHEmptyRecycleBinW(0LL, 0LL, SHERB_NOCONFIRMATION|SHERB_NOPROGRESSUI|SHERB_NOSOUND);
if ( v0 )
{
  __sprintf(&v2, L"[!] Recycle bin not cleaned, error 0x");
  *&v3[*(v2 + 4) + 16] = *&v3[*(v2 + 4) + 16] & 0xFFFFF1FF | 0x800;
  sub_140031820(&v2, v0);
}
else
{
  __sprintf(&v2, L"[+] Recycle bin cleaned");
}
sub_14000B000(&v2);
__consolePrint_0(v3);
```

Figure23 : Efface la corbeille sans notification ni indication à l'aide de SHEmptyRecycleBinW avec plusieurs indicateurs. Notez que certains codes ont été supprimés pour plus de clarté.

Configuration du fond d'écran avec le thread note

Si le drapeau *backgroundImage* est activé dans la configuration, le fond d'écran sera remplacé par une note contenant les adresses de courriels des auteurs de la menace.

Le processus commence par récupérer le contenu de backgroundImageData (config) et le rendre sous forme d'image. Celle-ci sera enregistrée temporairement dans le répertoire actuel du programme. Après l'enregistrement, à l'aide de SystemParametersInfoW, le fond d'écran est défini avec le drapeau SPI_SETDESKWALLPAPER.

```
BOOL __setBackgroundImage()
{
  PVOID *v0; // r8
  BOOL result; // eax
  if ( xmmword_1400B0DE0 )
  {
    sub_140039210();   // Uses the content of backgroundImageData from config and
                       // convert to image and save to output.bmp
    v0 = &pvParam;
    if ( *(&xmmword_1400B0DE0 + 1) > 7uLL )
      v0 = pvParam;
    return SystemParametersInfoW(SPI_SETDESKWALLPAPER, 0, v0, 3u);
  }
  return result;
}
```

Figure24 : Appelle l'API SystemParametersInfoW pour définir le fond d'écran avec les notes.

Figure25 : Fond d'écran contenant les contacts e-mail de l'acteur malveillant

Exécution des précommandes/postcommandes

La liste preRunCommands dans la configuration est exécutée via CreateProcessW, qui lance des utilitaires Windows intégrés pour supprimer les sauvegardes locales et empêcher la récupération. Les commandes exécutées comprennent :

- vssadmin.exe Delete Shadows /All /Quiet
- wbadmin delete backup -keepVersion:0 -quiet
- wmic.exe SHADOWCOPY /nointeractive
- bcdedit.exe /set {default} recoverynabled No

La liste postRunCommands dans la configuration est vide.

Montage de lecteurs sans lettre attribuée

Le programme appelle FindFirstVolumeW et FindNextVolumeW pour énumérer tous les chemins d'accès GUID des volumes sur l'hôte, s'assurant ainsi que tous les lecteurs disponibles sont montés avant de procéder au chiffrement des fichiers.

```
FirstVolumeW = FindFirstVolumeW(szVolumeName, 0x104u);
if ( FirstVolumeW != -1LL )
{
  do
  {
    v76 = -1LL;
    do
      ++v76;
    while ( szVolumeName[v76] );
    if ( szVolumeName[0] != '\\' )
      break;
    if ( szVolumeName[1] != '\\' )
      break;
    if ( szVolumeName[2] != '?' )
      break;
    if ( szVolumeName[3] != '\\' )
      break;
    v77 = 2 * v76 - 2;
    if ( *(szVolumeName + v77) != '\\' )
      break;
    if ( v77 >= 0x208 )
      sub_140041FE8();
    *(szVolumeName + v77) = 0;
    DosDeviceW = QueryDosDeviceW(DeviceName, TargetPath, 0x104u);
    *(szVolumeName + v77) = '\\';
    if ( !DosDeviceW )
      break;
    __getLogicalDrives(szVolumeName);
  }
  while ( FindNextVolumeW(FirstVolumeW, szVolumeName, 0x104u) );
  FindVolumeClose(FirstVolumeW);
}
```

Figure26 : L'extrait de code énumère tous les lecteurs montés et tente de résoudre leurs chemins d'accès associés à l'aide de QuestDosDeviceW. Cela garantit que tous les lecteurs sont répertoriés avant le chiffrement.

Tous les lecteurs logiques, y compris les lecteurs amovibles, fixes, RAM et distants, sont énumérés à l'aide de GetLogicalDrives. Pour les lecteurs distants, WNetGetConnectionW est utilisé pour obtenir leurs chemins d'accès réseau correspondants.

```
if ( GetDriveTypeW(v5) == DRIVE_REMOTE )
{
  nLength[0] = 1028;
  v6 = lpRootPathName;
  if ( v19 > 7 )
    v6 = lpRootPathName[0];
  ConnectionW = WNetGetConnectionW(v6, RemoteName, nLength);
  if ( ConnectionW )
  {
    sub_14003C320(L"[!] WNetGetConnection failed. Error: 0x%X\n", ConnectionW);
  }
```

Figure27 : Si le type de lecteur est DRIVE_REMOTE, WNetGetConnectionW est utilisé pour récupérer le nom distant du lecteur.

Si l'indicateur *chiperDrives* est vrai, la commande « cipher \w:<répertoire cible> » est exécutée, rendant définitivement irrécupérables les fichiers supprimés en écrasant l'espace non alloué (libre) par des zéros.

```
if ( Src[3] > 7uLL )
  v6 = *Src;
__toWideChar(v11, a2, a3, L"cipher /w:", 10LL, v6, v5);
sub_14003B120(v13, v11);
```

Figure28: Commande cipher

Chiffrement des fichiers

Une fois que tous les lecteurs disponibles sont obtenus, y compris ceux nouvellement montés, le chiffrement commence. Un thread dédié est créé uniquement pour l'énumération des fichiers et l'opération de chiffrement.

Au départ, la clé publique générée est importée à l'aide de *CryptImportKey()*, puis la longueur du bloc de la clé publique est récupérée avec *CryptGetKeyParam()*.

Tous les fichiers sont énumérés de manière récursive à l'aide de *FindFirstFileW* et *FindNextFileW*. Chaque chemin d'accès du fichier est comparé à la liste des sous-chaînes et des répertoires, ce qui empêche le chiffrement des répertoires liés au système. Cela garantit que le système reste opérationnel pendant le chiffrement des fichiers.

```
ExtensionW = PathFindExtensionW(FindFileData.cFileName);
v50 = __whitelistedExtensions;
v51 = qword_1400B7BD0;
while ( v50 != v51 )
{
  v52 = v50;
  if ( *(v50 + 24) > 7uLL )
    v52 = *v50;
  if ( !lstrcmpiW(ExtensionW, v52) )
  {
    sub_14000A030(&v66, *(&qword_1400B6030 + *(qword_1400B6030 + 4) + 72));
    __sprintf(&v66, L"[-] Skipping file: ");
    v53 = v75;
    if ( *(&v76 + 1) > 7uLL )
      v53 = v75[0];
    sub_140010A40(&v66, v53, v76);
    __sprintf(&v66, L", found: ");
```

Figure29 : l'extension est récupérée à l'aide de PathFindExtensionW et comparée aux extensions figurant sur la liste blanche à l'aide de lstrcmpiW

Enfin, les extensions figurant sur la liste blanche de la configuration sont comparées à l'extension de chaque fichier, qui comprend également les fichiers liés au système, le nom de la note de rançon et les extensions connues des ransomwares liés à MedusaLocker

```
if ( (FindFileData.nFileSizeLow | (FindFileData.nFileSizeHigh << 32)) >= 1000 )
{
  v57 = lpString2;
  if ( qword_1400B0DC8 > 7 )
    v57 = lpString2[0];
  if ( lstrcmpiW(ExtensionW, v57) )       // skipped extensions
  {
    v58 = &__ransomNoteFileName;
    if ( qword_1400B7BA8 > 7 )
      v58 = __ransomNoteFileName;         // HOW_TO_RECOVER_DATA.html
    if ( lstrcmpiW(FindFileData.cFileName, v58) )
      break;
  }
}
```

Figure30 : La taille du fichier doit être supérieure ou égale à 1 000 octets

Il ne crypte que les fichiers dont la taille est supérieure ou égale à 1 000 octets en décimal, empêchant ainsi le cryptage des petits fichiers.

Outre les conditions ci-dessus, avant de lancer le processus de chiffrement, **MedusaLocker utilise Windows Restart Manager pour vérifier si le fichier cible à chiffrer est verrouillé ou ouvert par un autre processus, empêchant ainsi le chiffrement.** Cette technique est également utilisée par plusieurs groupes de ransomware dans le même but.

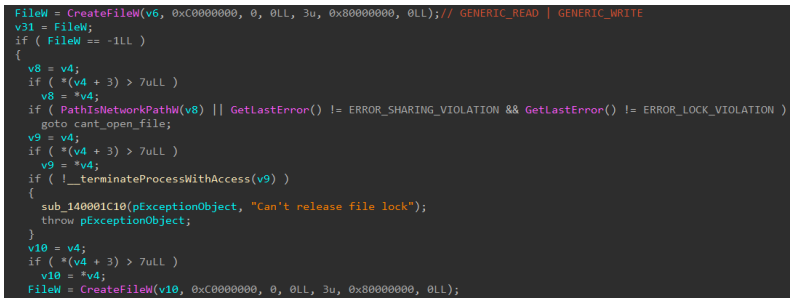


Figure31 : si le fichier est actuellement ouvert par un autre processus, le programme appelle __terminateProcessWithAccess pour mettre fin au processus qui détient son descripteur, empêchant ainsi le chiffrement. (Par exemple, si notepad.exe détient le descripteur d'un fichier, il sera fermé).

Figure32 : utilise les API RmStartSession, RmRegisterResources, RmGetList, RmShutdown et RmEndSession pour arrêter les processus qui détiennent le descripteur de fichier, empêchant ainsi le chiffrement.

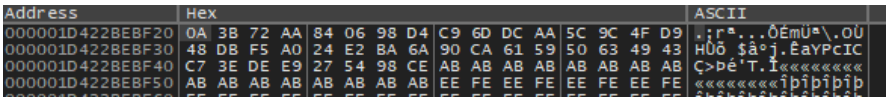


Figure33 : 0x28 octets sécurisés générés de manière aléatoire par CryptGenRandom().

Une fois le chiffrement lancé, le rançongiciel MedusaLocker génère 0x28 octets aléatoires sécurisés à l'aide de CryptGenRandom(). Ces blocs de données serviront de matériel d'initialisation du bloc de flux de clés Chacha20.

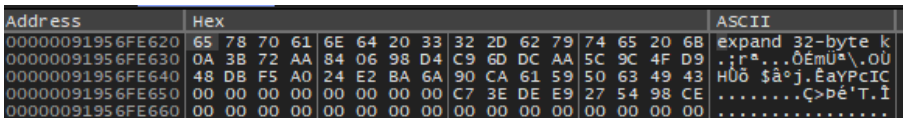


Figure34 : Initialisation du bloc de flux de clés Chacha20 avec la constante reconnaissable « expand 32-byte k »

Les données générées sont ensuite chiffrées à l'aide de la clé publique générée à partir des clés appariées, qui est également stockée dans le registre du système.

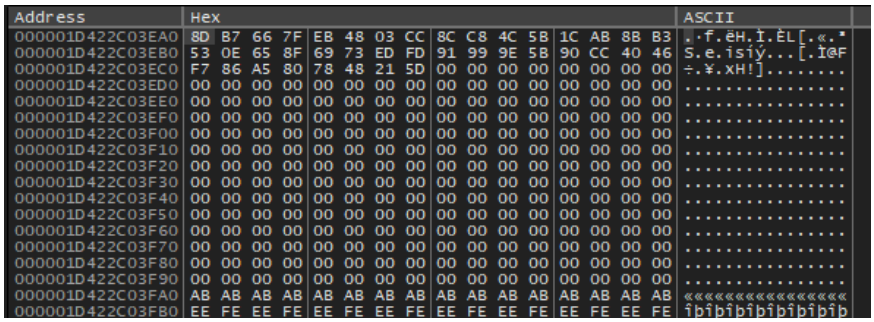


Figure35: Les 0x28 octets générés sont copiés dans un conteneur de 0x100 octets avant d'être chiffrés à l'aide de la clé publique

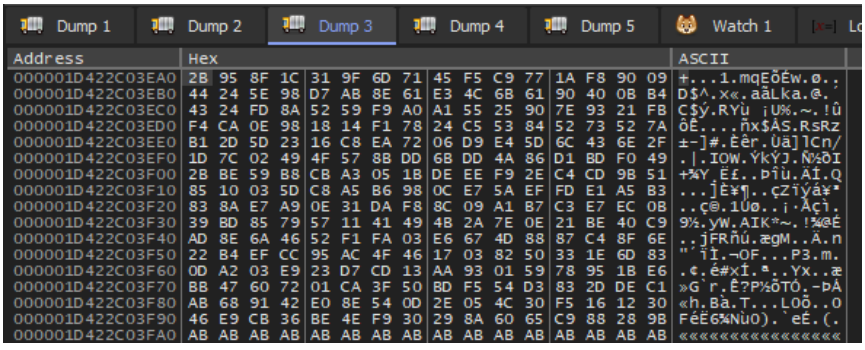


Figure36: Après le chiffrement

Une fois le flux de clés entièrement initialisé, le rançongiciel procède au chiffrement du fichier par segments de 0x2000 octets, laissant un espace équivalent de 0x2000 octets entre les zones chiffrées. Cette technique est couramment utilisée pour accélérer le processus de chiffrement tout en rendant les fichiers illisibles.

```
Src[1] = v11 + 0x28;
__generateSecureRandomBytes(Src);             // Generate cryptographical secure random bytes
__initSalsaKeyStreamBlock(v49, Src[0], Src[0] + 32);// Initialize Salsa20 Keystream block
v51 = 64LL;
v50 = 0LL;
__encryptPrivateKey(a1, Src);                 // Encrypt Salsa20 Key using RSA
if ( (Src[1] - Src[0]) != 0x100 )             // Check size of key
{
  v26 = sub_1400096A0(Src);
  v27 = sub_14000710(&Overlapped, v26);
  v28 = sub_14000B710(lpNewFileName1, "Bad password size: ", v27);
  sub_140001BA0(v42, v28);
  throw v42;
}
__chacha20Encrypt(FileW, v49, lpExistingFileName, a3);// Chacha20 encrypt
v13 = *(&__encryptedPrivateKey + 1);
```

Figure37 : l'image montre la génération de 0x28 octets pour l'initialisation du bloc de flux de clés Chacha20 jusqu'au chiffrement du fichier avec Chacha20.

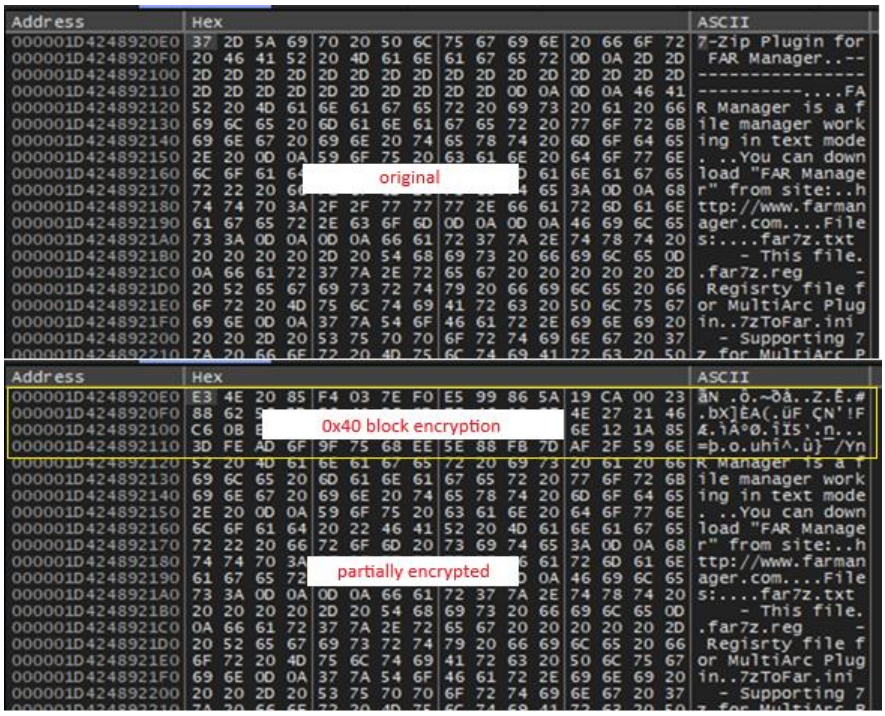


Figure38: Image montrant le chiffrement partiel de fichiers mis en œuvre par le ransomware à l'aide de Chacha20 en blocs de 0x40 octets.

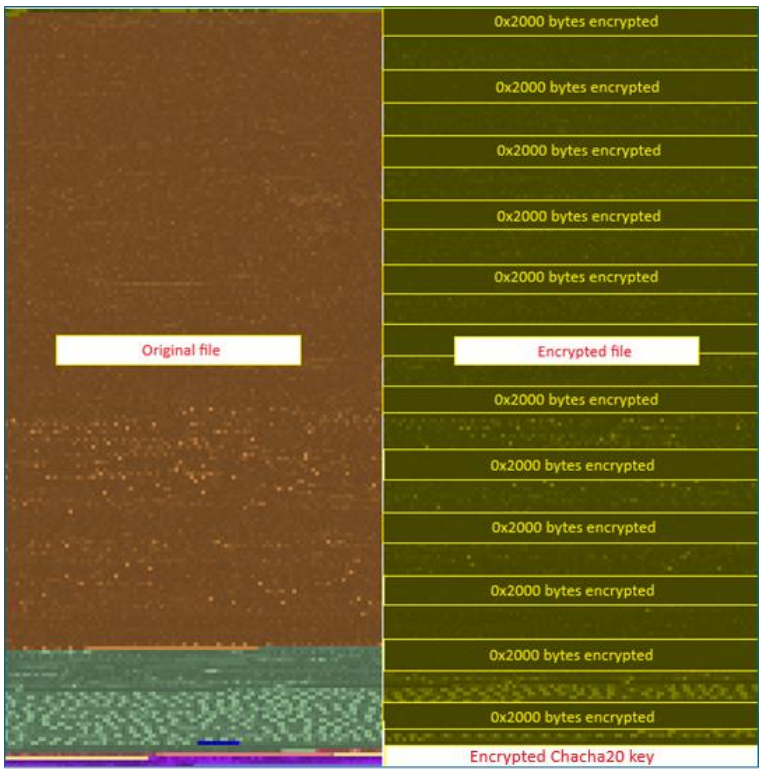


Figure39 : Comparaison entre un fichier original (à gauche) et son équivalent chiffré (à droite). La version chiffrée présente un motif de chiffrement alternatif, dans lequel chaque bloc de 0x2000 octets de données est chiffré, laissant les blocs intermédiaires intacts. Cette technique de chiffrement sélectif par blocs, combinée à un segment de clé Chacha20 chiffré de 0x100 octets à la fin du fichier, est caractéristique des stratégies de chiffrement partiel rapide souvent utilisées par les ransomwares modernes pour accélérer le chiffrement tout en rendant le fichier inutilisable.

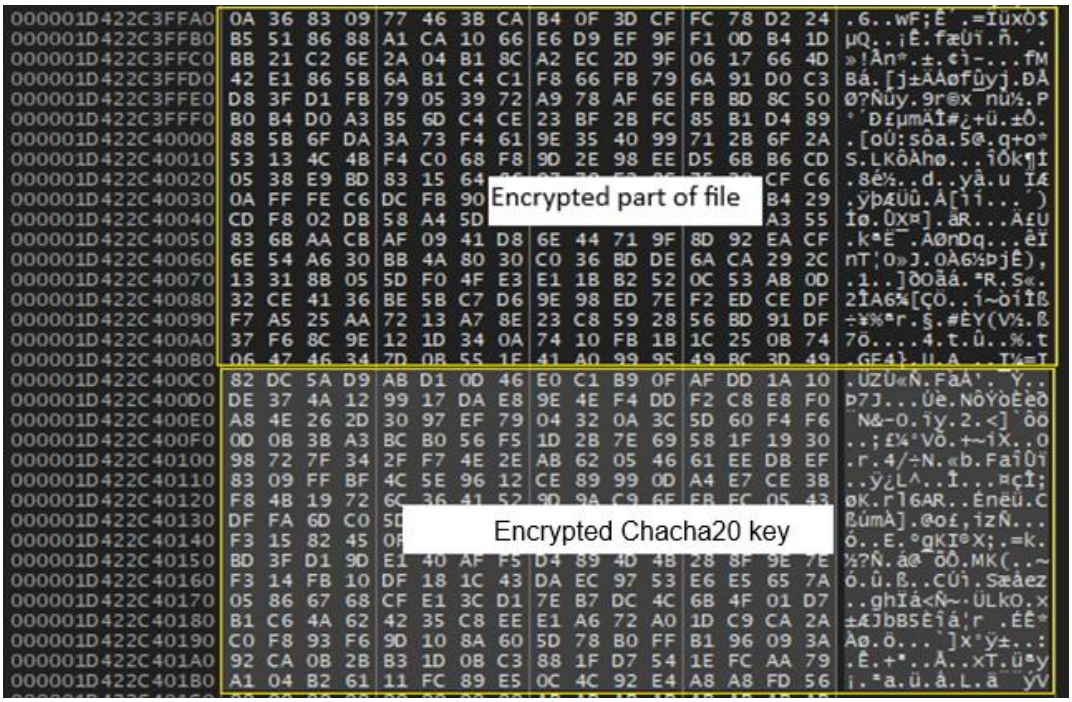


Figure40: La clé cryptée est intégrée à la fin du fichier crypté, ce qui permet la récupération puisque le composant de décryptage unique reste attaché à la charge utile.

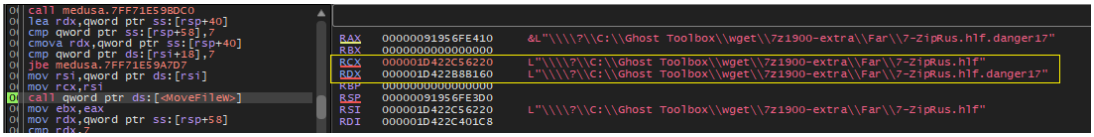
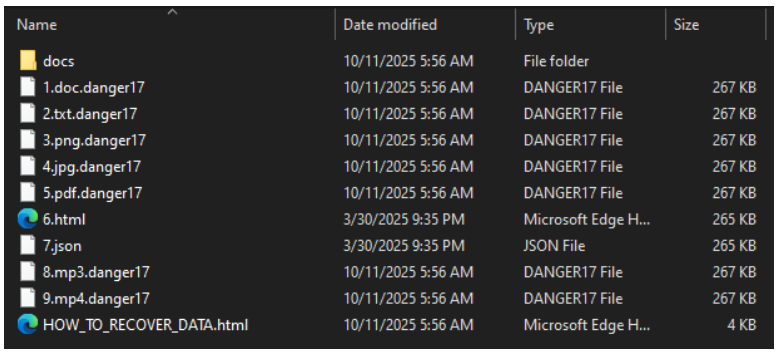


*Figure41: Renommer les fichiers chiffrés à l'aide de MoveFileW en ajoutant l'extension *.danger17.*

L'échantillon MedusaLocker analysé ressemble à la structure et aux caractéristiques de MedusaLockerLocker version 3, également connu sous le nom de variante ***BabyLockerKZ***, et cette analyse renforce la conclusion selon laquelle ils ont la même origine.

L'image ci-dessous montre les fichiers chiffrés auxquels a été ajoutée l'extension « .danger17 » et une note contenant l'identifiant personnel et les adresses e-mail de contact de l'auteur de la menace.



Name	Date modified	Type	Size
docs	10/11/2025 5:56 AM	File folder	
1.doc.danger17	10/11/2025 5:56 AM	DANGER17 File	267 KB
2.txt.danger17	10/11/2025 5:56 AM	DANGER17 File	267 KB
3.png.danger17	10/11/2025 5:56 AM	DANGER17 File	267 KB
4.jpg.danger17	10/11/2025 5:56 AM	DANGER17 File	267 KB
5.pdf.danger17	10/11/2025 5:56 AM	DANGER17 File	267 KB
6.html	3/30/2025 9:35 PM	Microsoft Edge H...	265 KB
7.json	3/30/2025 9:35 PM	JSON File	265 KB
8.mp3.danger17	10/11/2025 5:56 AM	DANGER17 File	267 KB
9.mp4.danger17	10/11/2025 5:56 AM	DANGER17 File	267 KB
HOW_TO_RECOVER_DATA.html	10/11/2025 5:56 AM	Microsoft Edge H...	4 KB

*Figure42: Les fichiers cryptés ont été renommés avec l'extension ***.danger17** et une note de rançon nommée « HOW_TO_RECOVER_DATA.html » a été déposée.*

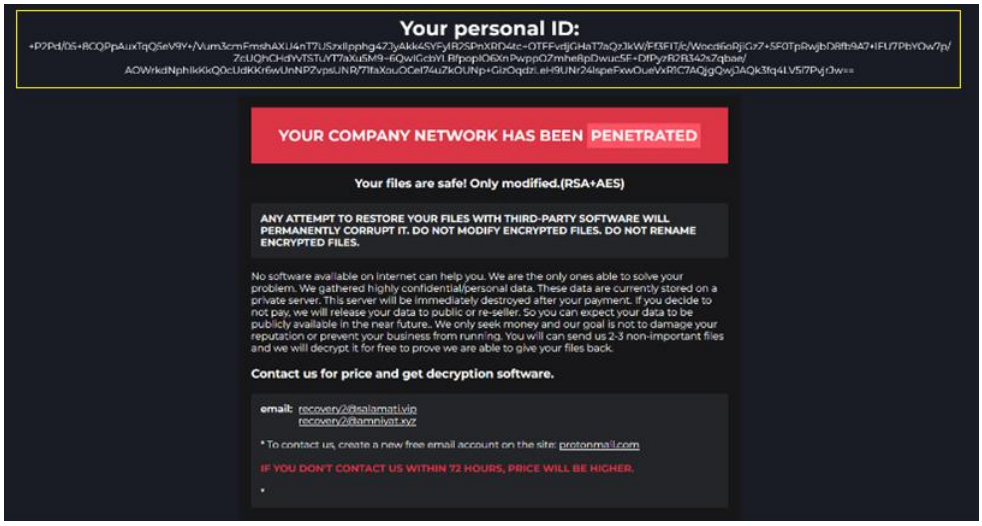


Figure43: La note de rançon contient les informations cryptées du PC qui serviront à identifier la victime pour la récupération. Elle comprend également les instructions pour contacter l'auteur de la menace à l'aide de ProtonMail.

`

Chiffrement du lecteur distant

Si le paramètre « -network » n'est pas présent dans les paramètres de la ligne de commande, le rançongiciel MedusaLocker crée son propre processus avec les paramètres « -network » et « -skip_misc », ignorant les tâches diverses pour se concentrer sur les lecteurs réseau.

```
__networkDriveOnly_1 = __networkDriveOnly;
if ( !__networkDriveOnly && __isElevated )
{
  if ( (0x7FFFFFFFFFFFFFELL - xmmword_1400B0E20) < 0x14 )
    unknown_libname_3(xmmword_1400B0E20, v86, v87, v88);
  v90 = &xmmword_1400B0E10;
  if ( *(&xmmword_1400B0E20 + 1) > 7 )
    v90 = xmmword_1400B0E10;
  __toWideChar(v176, v86, v87, v90, xmmword_1400B0E20, L" -network -skip_misc", 20LL);
  __createProcess(v176);
}
```

Figure44: S'il n'y a pas de paramètre « -network », un processus distinct est créé avec « -network » et « -skip_misc ».

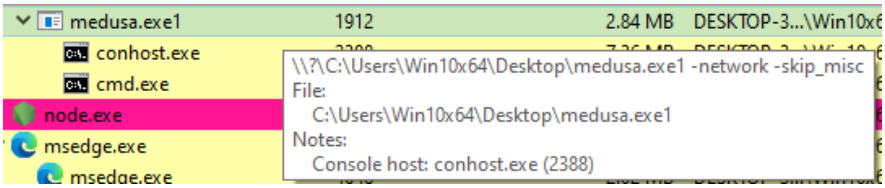


Figure45: Création d'un processus de lui-même avec les paramètres « -network » et « -skip_misc ».

Répertoires ignorés par le rançongiciel MedusaLocker

Le contenu des répertoires suivants n'est pas chiffré par le rançongiciel MedusaLocker.

- « :\\$Recycle.Bin\\ »
- « :\\Recovery\\ »
- « :\\System Volume Information\\ »
- « :\\Windows\\ »
- « \\AppData\\ »
- « :\\$WinREAgent »
- « :\\$Windows.~WS »
- « :\\$WINDOWS.~BT »
- « \\Google\\Chrome\\ »
- « %AppData% »
- « %LocalAppData% »
- "\\\\?\\C:\\Users\\Public\\Documents"
- "\\\\?\\C:\\ProgramData"
- « C:\\perflogs »
- « C:\\Intel »
- « C:\\Drivers »
- « C:\\inetpub »
- « C:\\ProgramData\\AnyDesk »

- « C:\\AMD »
- « B:\\EFI »
- « A\\Boot »

Extensions ignorées par le rancongiciel MedusaLocker

MedusaLocker ne chiffre pas les fichiers ayant les extensions suivantes. Il se concentre sur les fichiers qui sont susceptibles de contenir des données.

.exe, .dll, .sys, .ini, .rdp, .lnk, .bmp, .mov, .cab, .url, .vsix, .msi, .pyc, .pyd, .vdm, .json,

.log,.xrm-
ms, .danger21, .html, .cyberhazard32, .cyberhazard33, .cyberhazard34, .warning!, .infe
ction, .warning!_2, .warning!_3, .warning!_4, .danger, .warning!_5, .warning!_6, .warnin
g!_7, .warning!_8, .warning!_9, .warning!_10, .warning!_11, .warning!_12, .warning!_13,
.warning!_14, .warning!_15, .warning!_16, .warning!_17, .hazard, .hazard2, .explorer, .w
arninrg!, .danger3, .danger4, .hazard4, .danger5, .danger6, .danger7, .danger10, .dang
er12, .danger15, .hazard19, .danger230, .danger227, .danger237, .danger238, .danger
239, .cybertron13, .cybertron18, .cybertron22, .cybertron23, .cybertron24, .cybertron26,
.cybertron28, .cybertron32, .cybertron40, .cybertron41, .cybertron42, .danger61, .cybertr
on49, .cybertron50, .cybertron52, .cybertron54, .cybertron56, .cybertron57, .cybertron58
, .cybertron59, .cybertron62, .cyberhazard4, .cyberhazard6, .cyberhazard7, .hazard3, .d
anger13, .hazard4, .hazard5, .hazard7, .danger21, .hazard8, .hazard9, .danger26, .dan
ger27, .danger30, .danger31, .danger33, .hazard11, .danger43, .hazard13, .hazard14, .
hazard15, .danger45, .danger50, .danger51, .hazard16, .danger53, .danger54, .hazard1
8, .hazard19, .hazard20, .hazard21, .danger58, .danger60, .danger61, .hazardick, .haza
rdick2, .hazardick3, .hazardick3, .hazardick7, .hazardick8, .hazardick9, .hazardick10, .h
azardick11, .hazardick12, .hazardick14, .hazardick13, .hazardick15, .hazardick16, .haza
rdick17, .hazardick18, .hazard20, .hazardick21, .hazardick22, .hazardick23, .hazardick2
4, .hazardick25, .hazardick26, .hazardick27, .hazardick28, .hazardick30, .hazardick35, .
hazardick38, .lckfile, .revenge, .hazard2, .danger11, .danger12, .danger13, .danger14, .
danger15, .danger16, .**danger17.**

Detection et blocage de MedusaLocker par le EDR de Streamscan

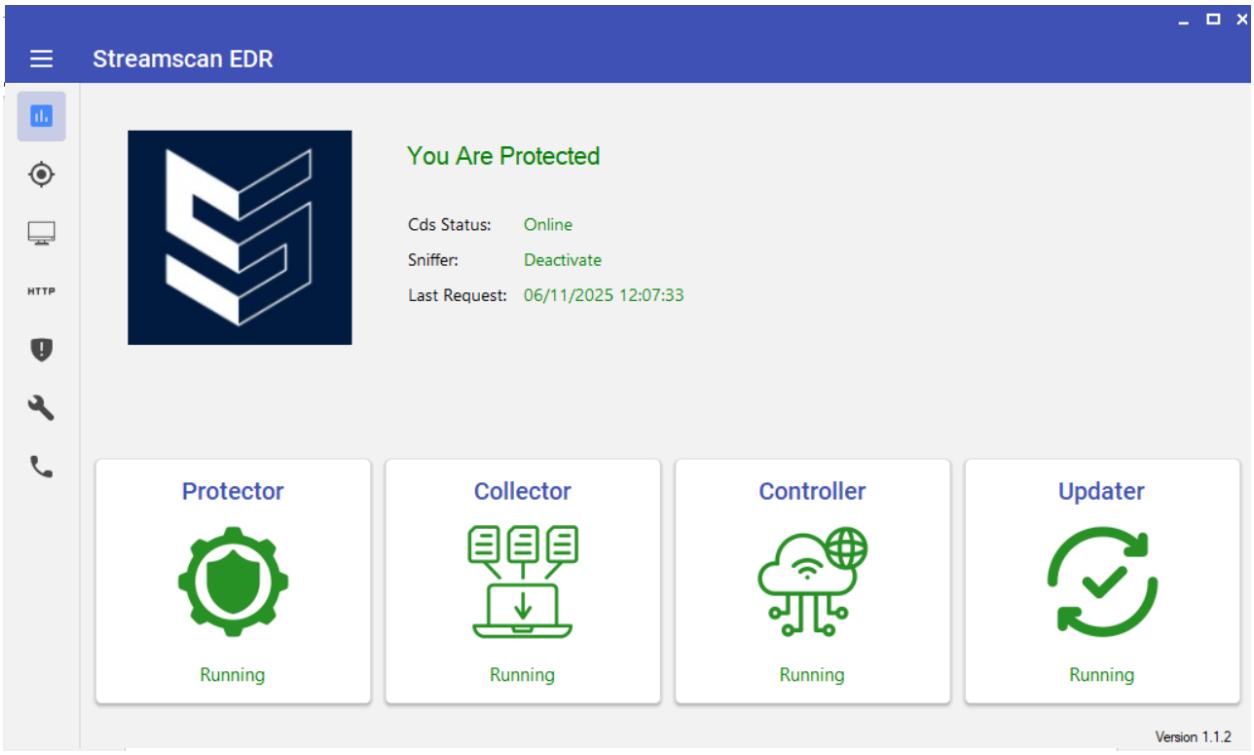
En plus d'offrir un service de surveillance à distance de la sécurité SOC/MDR et un service de réponse aux incidents, StreamScan est éditeur de technologies de cybersécurité.

Notre technologie XDR nommé CDS (*Cyberthreat Detection System*) combine des fonctionnalités de IDS/IPS/NDR, EDR, Gestion des logs, etc.). Nous disposons d'un brevet américain (US Patent US10218731B2) pour le CDS.

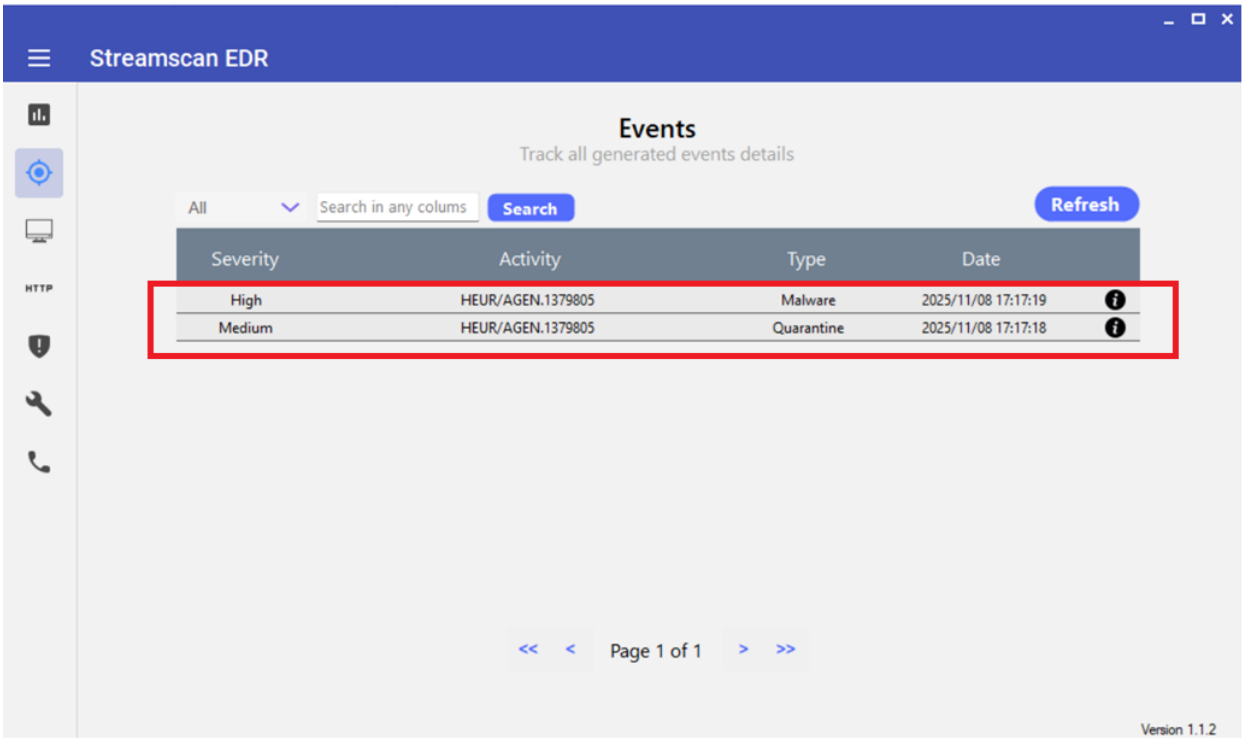
Pour rappel, notre CDS a été sélectionnée comme Innovation par le gouvernement fédéral du Canada.

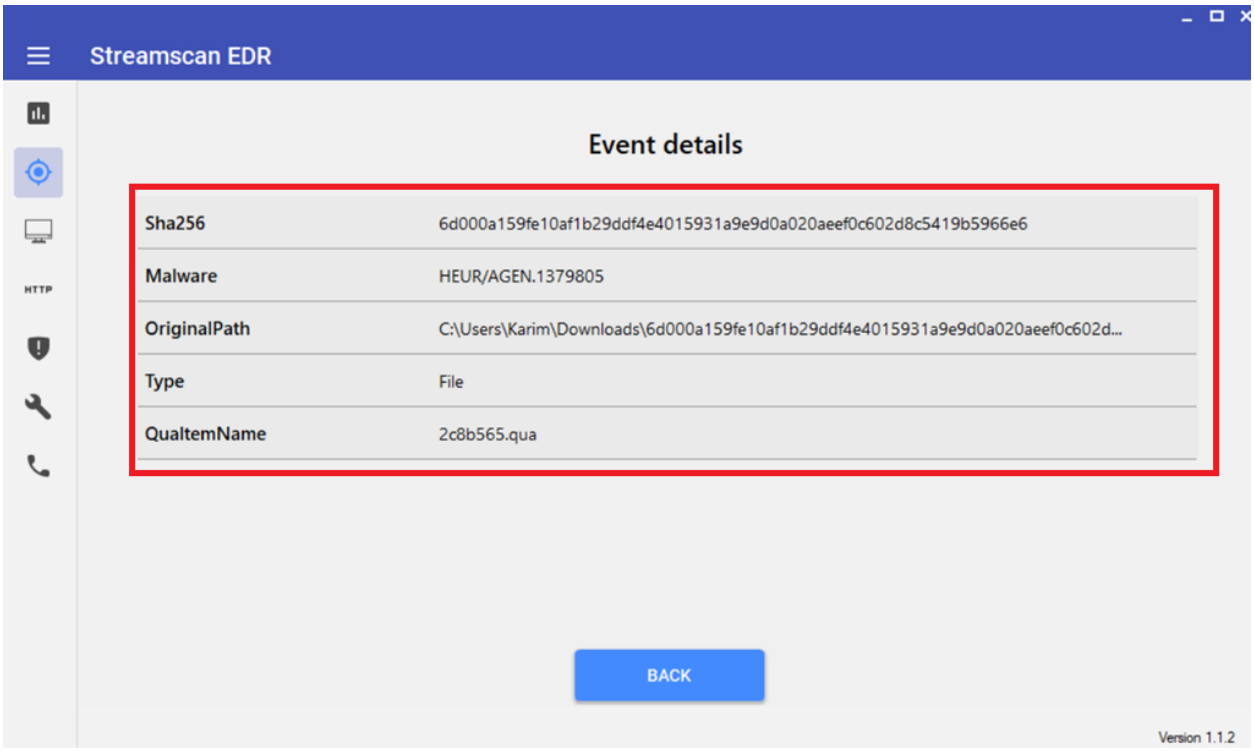
Pour en savoir plus sur notre EDR : https://streamscan.ai/nos-technologies/edr/

Nous avons vérifié la capacité de notre EDR à détecter le rancongiciel MedusaLocker. MedusaLocker :



- Lorsqu'analysé par notre Streamscan EDR, le rançongiciel MedusaLocker est détecté et bloqué.


Streamscan EDR
HTTP
Events
Track all generated events details
All
Search in any colums
Search
Refresh
Severity
Activity
Type
Date
High
HEUR/AGEN.1379805
Malware
2025/11/08 17:17:19
Medium
HEUR/AGEN.1379805
Quarantine
2025/11/08 17:17:18
<< < Page 1 of 1 > >>
Version 1.1.2


Streamscan EDR
HTTP
Event details
Sha256
6d000a159fe10af1b29ddf4e4015931a9e9d0a020aeef0c602d8c5419b5966e6
Malware
HEUR/AGEN.1379805
OriginalPath
C:\Users\Karim\Downloads\6d000a159fe10af1b29ddf4e4015931a9e9d0a020aeef0c602d...
Type
File
QualtemName
2c8b565.qua
BACK
Version 1.1.2

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

-