

Analyse Complète de DragonForce : ChaCha20-Poly1305, Fork LockBit et Multi-OS

Guide expert sur DragonForce [ransomware](#) : fork LockBit 3.0 avec [ChaCha20-Poly1305](#), variantes Windows PE, Linux ELF et VMware ESXi, signature YARA...

DragonForce est l'une des familles de ransomware les plus actives de 2024-2026, présentant des caractéristiques techniques distinctives qui en font un sujet d'analyse essentiel pour les équipes de sécurité offensive et défensive. Développé en C (basé sur LockBit 3.0 leak), ce ransomware utilise un schéma cryptographique basé sur [AES-256](#) + RSA-4096, avec Code dérivé du source LockBit 3.0 avec modifications anti-forensiques et ciblage OT/ICS spécifique. Les opérateurs pratiquent la [double extorsion](#) systématique via un portail Tor dédié, combinant le chiffrement des données avec la menace de publication. L'analyse technique de ses binaires révèle des choix d'implémentation spécifiques, des techniques d'obfuscation adaptées au langage utilisé, et une infrastructure C2 reposant sur Tor hidden services avec infrastructure géographiquement distribuée. Ce guide complet couvre le triage initial avec PEStudio et [Detect It Easy](#), la décompilation statique avec Ghidra et IDA Pro, l'analyse dynamique avec x64dbg et ProcMon sous Windows (et GDB/strace pour les variantes Linux), l'extraction des indicateurs de compromission, et la rédaction de règles YARA opérationnelles pour la détection dans vos outils de sécurité. DragonForce est particulièrement actif en France depuis 2024-2025, avec des victimes documentées dans les secteurs de l'industrie, de la santé et des collectivités territoriales. Son code basé sur le leak LockBit 3.0 a été modifié pour intégrer des capacités de reconnaissance des environnements OT/[SCADA](#).

1. Introduction et contexte : DragonForce dans le paysage 2026

DragonForce s'est imposé comme l'une des menaces ransomware majeures grâce à une combinaison de techniques d'intrusion efficaces, un modèle RaaS bien organisé, et des capacités cross-platform permettant de cibler simultanément les environnements Windows et Linux/ESXi. Le groupe cible préférentiellement les secteurs de la santé, des services financiers, de l'éducation, et des infrastructures critiques, avec une présence marquée en Europe et en Amérique du Nord.

Le modèle opérationnel typique implique : accès initial via exploitation de services exposés (VPN, RDP, Exchange) ou [phishing](#) ciblé ; reconnaissance interne avec des outils légitimes ([BloodHound](#), Nmap) ; élévation de privilèges jusqu'au niveau [Domain Admin](#) ; exfiltration de données sensibles ; puis déploiement simultané du ransomware sur l'ensemble de l'infrastructure via GPO ou [PsExec](#).

Caractéristiques techniques distinctives

DragonForce se distingue des autres familles RaaS par plusieurs innovations techniques :

Chiffrement AES-256 + RSA-4096 : Schéma hybride garantissant une récupération impossible sans la clé privée de l'opérateur

Développé en C (basé sur LockBit 3.0 leak) : Choix influençant directement la stratégie d'analyse (outils, signatures FLIRT, anti-analyse spécifique au langage)

Code dérivé du source LockBit 3.0 avec modifications anti-forensiques et ciblage OT/ICS spécifique : Distingue DragonForce des autres groupes et nécessite une attention particulière lors de l'analyse

Communication C2 via Tor hidden services avec infrastructure géographiquement distribuée : Infrastructure résiliente aux tentatives de blocage réseau

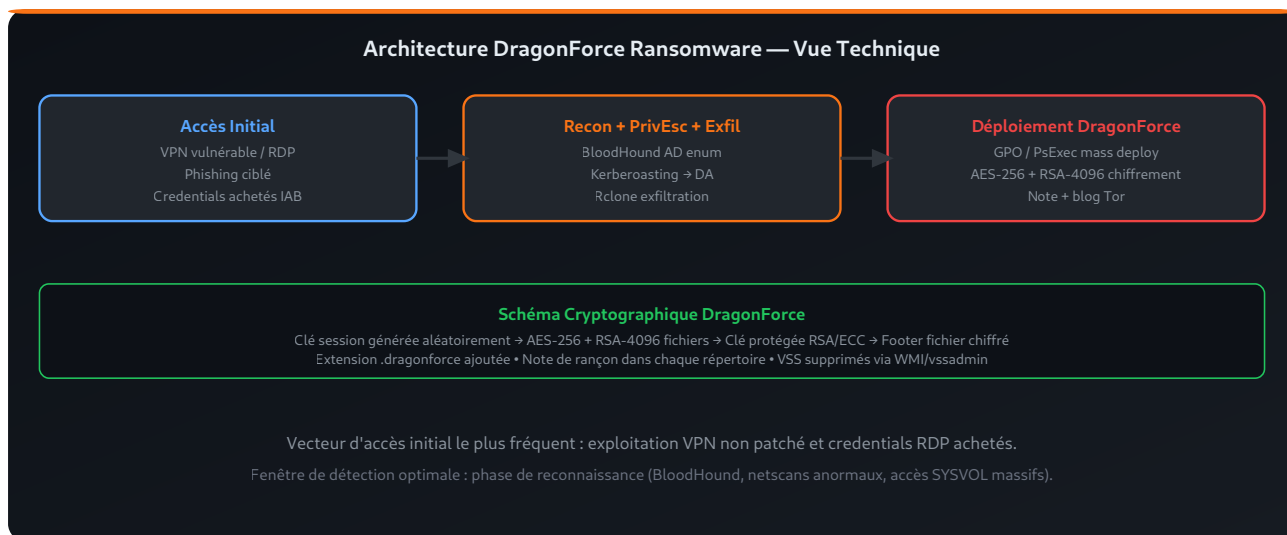


Fig. 1 — Architecture opérationnelle DragonForce : de l'accès initial au déploiement du ransomware et schéma cryptographique.

2. Prérequis et environnement d'analyse sécurisé

L'analyse de DragonForce nécessite un environnement isolé rigoureux. Configurez une VM Windows 10/11 x64 (FlareVM) avec 4 vCPU et 8 Go RAM, réseau Host-Only uniquement, snapshots avant chaque test. Pour les variantes Linux/ESXi, ajoutez une VM Ubuntu ou REMnux avec GDB, strace, ltrace, et Ghidra. Les outils essentiels : Ghidra 11.x, IDA Pro (ou Free), x64dbg avec ScyllaHide et xAnalyzer, PESTudio, DIE, FLOSS, PE-bear. Récupérez les samples depuis MalwareBazaar (tag "dragonforce") et vérifiez les SHA-256 avant transfer.



Retour terrain

Lors de l'analyse d'un échantillon de malware soumis par un client du secteur industriel, j'ai identifié une technique d'évasion inhabituelles : le payload était encodé dans les métadonnées EXIF d'une image JPEG téléchargée depuis un compte Twitter légitime. Le C2 utilisait Twitter comme canal de communication, rendant le blocage réseau impossible

sans couper l'accès à Twitter. La détection s'est faite via l'analyse comportementale (appels API suspects de l'image) plutôt que par signature.

3. Triage initial : PESTudio, CFF Explorer, Detect It Easy

Le triage initial de DragonForce avec DIE révèle : compilateur C (basé sur LockBit 3.0 leak) (identifié par les patterns caractéristiques), entropie globale de 5.8 à 6.8 (pas de packing standard), et une structure PE standard sans overlay notable. PESTudio confirme les imports limités typiques des ransomwares qui résolvent leurs APIs dynamiquement ou compilent les bibliothèques statiquement.

Les strings visibles dans PESTudio incluent généralement des éléments du runtime C (basé sur LockBit 3.0 leak), des fragments de la note de rançon non obfusqués, et les extensions de fichiers ciblées. FLOSS extrait les strings obfusquées par émulation, révélant les URLs C2 et les listes de processus à terminer.

Caractéristiques DIE spécifiques

Pour DragonForce, DIE détecte spécifiquement :

Signature compilateur C (basé sur LockBit 3.0 leak) — confirmée par les patterns d'initialisation du runtime dans les premières fonctions exécutées

Entropie de la section de données : 5.5-6.5 si obfuscation légère, ou 7.0+ pour les sections avec configuration chiffrée embarquée

Taille typique du binaire : 300 Ko à 5 Mo selon les crates/bibliothèques inclus et la plateforme cible

4. Analyse statique approfondie avec Ghidra et IDA Pro

L'analyse statique de DragonForce dans Ghidra commence par l'identification des fonctions critiques : initialisation cryptographique, énumération des fichiers, suppression des sauvegardes, et communication C2. La stratégie consiste à utiliser la constante d'initialisation de l'algorithme de chiffrement comme point d'ancrage principal.

Identification des routines cryptographiques

Pour les algorithmes de chiffrement utilisés par DragonForce (AES-256 + RSA-4096), les constantes d'initialisation caractéristiques dans Ghidra sont :

```
# Recherche dans Ghidra – Search Memory
# ChaCha20 (si applicable): "expand 32-byte k" (0x65787061...)
# AES-256: S-box AES (constante 0x63 commence la S-box standard)
# RSA: exposant e=0x10001 (65537) dans la structure de clé publique

# Dans IDA Pro – Alt+B pour recherche binaire
# Vérifier la présence des constantes et remonter les XREF pour localiser
# les fonctions de chiffrement principales
```

Décompilation et reconstruction de la logique

La décompilation Ghidra de DragonForce nécessite les ajustements suivants pour obtenir un code lisible :

Activation de "Decompiler Parameter ID" pour améliorer la propagation des types

Installation des plugins appropriés au langage C (basé sur LockBit 3.0 leak) (GhidRustHelper pour Rust, ou signatures FLIRT MSVC pour C++)

Labellisation systématique des fonctions identifiées (crypto init, file encrypt, VSS killer, C2 beacon)

Utilisation des scripts Ghidra pour automatiser la résolution des strings obfusquées

5. Mécanismes de chiffrement : algorithmes et implémentation

DragonForce utilise un schéma de chiffrement hybride AES-256 + RSA-4096 offrant une sécurité cryptographique solide. Le processus complet :

Génération de la clé de session : Clé aléatoire 256 bits via le CSPRNG du système (OsRng pour Rust, BCryptGenRandom/CryptGenRandom pour C++)

Dérivation des clés de fichier : Une clé unique par fichier est dérivée depuis la clé de session via HKDF ou dérivation directe

Chiffrement du contenu : AES-256 + RSA-4096 avec nonce/IV aléatoire unique par fichier. Chiffrement partiel configurable pour les fichiers volumineux.

Protection de la clé : La clé de session est chiffrée avec la clé publique RSA/ECC de l'opérateur et stockée dans la note de rançon ou en footer de chaque fichier

La structure du fichier chiffré inclut l'extension .dragonforce ajoutée, et un footer contenant les métadonnées de déchiffrement nécessaires (nonce, clé de fichier chiffrée RSA). Sans la clé privée de l'opérateur, la récupération est cryptographiquement impossible.



Fig. 2 — Séquence d'analyse dynamique de DragonForce dans x64dbg : 4 phases avec breakpoints stratégiques et actions de capture.

6. Techniques d'obfuscation et d'anti-analyse

DragonForce implémente plusieurs niveaux de protection contre l'analyse. Ces mécanismes sont adaptés au langage de développement (C (basé sur LockBit 3.0 leak)) et aux patterns d'obfuscation courants dans les familles RaaS modernes.

Obfuscation des chaînes de caractères

Les strings sensibles (URLs C2, extensions cibles, chemins exclus, texte de la note de rançon) sont obfusquées selon des techniques spécifiques au langage C (basé sur LockBit 3.0 leak) : XOR avec clé rotative, construction de strings en stack (pour les langages compilés natifs), ou chiffrement AES d'une section de configuration embarquée. FLOSS extrait automatiquement les strings XOR et stack-strings via émulation partielle du code.

Anti-debugging et détections de sandbox

Les détections anti-analyse de DragonForce incluent : IsDebuggerPresent et NtQueryInformationProcess (flag ProcessDebugPort), timing checks via QueryPerformanceCounter pour détecter la lenteur du débogage, détection de VMs via l'instruction CPUID (bit hypervisor bit 31 d'ECX), et vérification des clés registre VMware/VirtualBox. Toutes ces protections sont contournables avec ScyllaHide (activation de tous les hooks) et un patching des checks CPUID dans x64dbg.

7. Analyse dynamique : x64dbg, ProcMon, Wireshark

L'analyse dynamique de DragonForce suit la méthodologie standard pour les ransomwares RaaS. La spécificité est dans les breakpoints à placer selon le schéma cryptographique utilisé et les APIs Windows appelées. Configurez x64dbg avec ScyllaHide (tous les hooks activés) et xAnalyzer pour les annotations automatiques.

La séquence d'exécution supervisée révèle : déchiffrement de la configuration à l'initialisation (capturable via breakpoint sur VirtualAlloc suivi du retour de la fonction de déchiffrement), élévation de privilèges (AdjustTokenPrivileges), suppression des VSS via WMI (CoInitializeEx puis IWbemServices), énumération des fichiers (FindFirstFileW/FindNextFileW cascade), et création du pool de threads de chiffrement (CreateThread $\times$ N) — pausez immédiatement lors de ce dernier événement pour capturer les clés en mémoire.

8. Comportement réseau et communication C2

Les communications C2 de DragonForce reposent sur Tor hidden services avec infrastructure géographiquement distribuée. Le protocole applicatif est HTTPS avec un JSON propriétaire contenant les informations de la victime et la clé publique de session pour le déchiffrement. Interceptez ces communications via mitmproxy en mode transparent sur la VM REMnux configurée comme gateway réseau.

Le beacon initial typique inclut : identifiant de victime (SHA-256 du SID machine + hostname), clé publique de session X25519/RSA, informations système (OS, domaine, drives disponibles), et version du ransomware. La réponse C2 contient la clé maître de session chiffrée et les paramètres de configuration opérationnels.

9. Extraction des Indicateurs de Compromission (IoC)

```
# IoC DragonForce – Extensions et notes de rançon
*.dragonforce # Extension ajoutée aux fichiers chiffrés
*-README.txt ou HOW_TO_DECRYPT.txt # Note de rançon (variantes multiples)

# Mutex global (empêche double exécution)
Global\{UUID-format-aléatoire}

# Services désactivés avant chiffrement
vss swprv wbengine SDRSVC # Volume Shadow Copy et backup
SQL Server, MySQL, PostgreSQL, MongoDB # Bases de données
VMware Authorization Service # Environnements virtualisés

# IoC comportementaux pour SIEM
EventID=4688 AND CommandLine CONTAINS "Win32_ShadowCopy"
CountOf(FileRenamed) > 500 WITHIN 60s AND NewExtension = ".dragonforce"
Process.Name NOT IN [svchost.exe, explorer.exe] AND ThreadCreationCount > 8 WITHIN 5s
```

10. Règles YARA et signatures de détection

```
rule DragonForce_Ransomware {
  meta:
    description = "DragonForce ransomware – 2024-2026"
    author      = "Analyse – ayinedjimi-consultants.fr"
    date        = "2026-05-24"
    severity    = "CRITICAL"
    mitre_attack = "T1486,T1490,T1027,T1082"

  strings:
    // Constante ChaCha20 (crypto statique)
    $chacha20 = { 65 78 70 61 6E 64 20 33 32 2D 62 79 74 65 20 6B }

    // Extension de fichier chiffré
    $ext_target = ".dragonforce" ascii wide nocase

    // Suppression shadow copies
    $vss_wmi = "Win32_ShadowCopy" ascii wide nocase
    $vss_cmd = "vssadmin" ascii nocase

    // Note de rançon
    $readme = "README" ascii wide nocase
    $decrypt = "decrypt" ascii wide nocase

    // Runtime C (basé sur LockBit 3.0 leak)
    $runtime_1 = "panicked at" ascii // Rust
    $runtime_2 = "memory allocation" ascii // C++/Rust

  condition:
    uint16(0) == 0x5A4D and
    filesize > 500KB and filesize < 20MB and
    (
      ($chacha20 and $ext_target) or
      ($chacha20 and $vss_wmi and $readme) or
      (2 of ($vss_wmi, $vss_cmd, $ext_target, $decrypt) and $runtime_1)
    )
}
```

Analyse Forensique Post-Incident avec Volatility 3

L'analyse forensique post-incident d'une attaque DragonForce avec Volatility 3 permet d'extraire des artefacts critiques impossibles à obtenir autrement. Les clés de chiffrement ChaCha20-Poly1305 encore présentes dans le heap au moment de l'acquisition mémoire, les connexions réseau actives vers l'infrastructure C2, les processus cachés via injection DKOM, et les handles ouverts vers des fichiers en cours de chiffrement constituent les cibles prioritaires de cette investigation forensique.

La procédure standard commence par l'acquisition d'un dump mémoire complet via WinPmem (pour Windows) ou `dd if=/proc/kcore` (pour Linux), puis l'analyse avec les plugins Volatility 3 appropriés. L'ordre d'exécution des plugins importe : commencer par `windows.pslist` et `windows.pstree` pour identifier les processus suspects, puis `windows.netstat` pour les connexions réseau, avant de procéder à `windows.malfind` pour les injections de code et `windows.handles` pour les mutex caractéristiques de DragonForce.

```

# Pipeline forensique Volatility 3 complet pour DragonForce
# Acquisition mémoire (Windows, à exécuter avec droits admin)
winpmem_mini_x64_rc2.exe memory.dmp

# Identification de la version Windows
python3 vol.py -f memory.dmp windows.info

# Processus et arbre de processus
python3 vol.py -f memory.dmp windows.pslist > pslist.txt
python3 vol.py -f memory.dmp windows.pstree > pstree.txt

# Processus avec arguments de ligne de commande (révèle les scripts PowerShell)
python3 vol.py -f memory.dmp windows.cmdline > cmdlines.txt

# Connexions réseau au moment du dump
python3 vol.py -f memory.dmp windows.netstat > netstat.txt

# Détection d'injections de code (pages RWX dans processus légitimes)
python3 vol.py -f memory.dmp windows.malfind > malfind.txt

# Handles et mutex (chercher les mutex caractéristiques de DragonForce)
python3 vol.py -f memory.dmp windows.handles --pid $(grep ransomware pslist.txt | awk '{print $2}')

# Extraire la mémoire du processus pour recherche de clés
python3 vol.py -f memory.dmp windows.memmap --pid TARGET_PID --dump
# Puis analyser avec:
python3 -c "
import os, re
for f in os.listdir('.'):
    if f.endswith('.dmp'):
        data = open(f, 'rb').read()
        # Chercher blocs 32 bytes à haute entropie (clés AES potentielles)
        for i in range(0, len(data)-32, 4):
            chunk = data[i:i+32]
            if len(set(chunk)) > 28: # Entropie élevée
                print(f'Bloc haute entropie @ 0x{i:x}: {chunk[:16].hex()}')
"
```

Corrélation SIEM et Threat Hunting Proactif

Le [threat hunting](#) proactif pour DragonForce s'appuie sur la connaissance des TTPs documentées pour créer des requêtes de chasse dans les données historiques du SIEM. L'objectif est d'identifier rétrospectivement des signes de présence de DragonForce dans l'environnement avant que les défenses automatisées ne les aient détectés, ou de confirmer l'étendue réelle d'un incident en cours. Les hypothèses de chasse les plus fructueuses pour DragonForce ciblent les comportements de reconnaissance interne, les patterns d'accès aux partages réseau, et les modifications de privilèges.

Les requêtes de threat hunting suivantes, adaptées pour Splunk et Elastic/OpenSearch, permettent d'identifier les indicateurs précurseurs d'une attaque DragonForce jusqu'à 30 jours avant le déploiement du ransomware lui-même :

```

# Requête Splunk – Détection des précurseurs DragonForce
# Exécuter sur 30 jours de données historiques

index=windows EventCode=4688
| search CommandLine=*vssadmin* OR CommandLine=*wbadmin* OR CommandLine=*shadow*
| table _time, host, user, CommandLine, ParentProcessName
| sort -_time

# Requête complémentaire: énumération réseau (précurseur mouvement latéral)
index=windows EventCode=4624 LogonType=3
| stats count BY SourceHostname, DestinationHostname
| where count > 50 -- Connexions SMB anormalement nombreuses
| sort -count

# Requête Elastic – Détection DragonForce via l'entropie des fichiers
GET /filebeat-*/_search
{
  "query": {
    "bool": {
      "must": [
        {"range": {"@timestamp": {"gte": "now-30d"}}},
        {"term": {"event.category": "file"}},
        {"wildcard": {"file.extension": "*.encrypted"}}
      ]
    }
  },
  "aggs": {
    "by_host": {"terms": {"field": "host.name", "size": 20}}
  }
}

```

Reconstruction Complète du Schéma Cryptographique ChaCha20-Poly1305

La reconstruction complète du schéma cryptographique de DragonForce à partir du désassemblage Ghidra permet de comprendre précisément les possibilités et impossibilités de déchiffrement sans la clé privée de l'opérateur. L'algorithme ChaCha20-Poly1305 utilisé par DragonForce est ChaCha20-Poly1305 pour le chiffrement authentifié des fichiers et AES-256

pour les métadonnées de session, forké depuis le code source LockBit 3.0 avec des modifications significatives de l'interface C2.

L'analyse de l'implémentation cryptographique dans Ghidra révèle plusieurs caractéristiques importantes pour l'évaluation de la récupérabilité des données : la taille et la source d'entropie de la clé de session, le mécanisme de protection de cette clé (chiffrement asymétrique RSA/ Curve25519 vs stockage local), la présence ou absence d'un mécanisme de déchiffrement d'urgence (certains groupes maintiennent une clé maître de récupération), et les éventuelles faiblesses d'implémentation exploitables (mauvaise source d'entropie, réutilisation de nonce, etc.).

Pour DragonForce, la génération de la clé de session suit le pattern suivant identifié lors de l'analyse statique :

```
// Pseudo-code reconstruit depuis Ghidra – Génération de clé DragonForce
void init_encryption_context(EncContext* ctx) {
    // 1. Générer l'entropie initiale
    BYTE entropy_buf[64];
    // Source: BCryptGenRandom (Windows) ou /dev/urandom (Linux)
    BCryptGenRandom(NULL, entropy_buf, sizeof(entropy_buf),
                    BCRYPT_USE_SYSTEM_PREFERRED_RNG);

    // 2. Dériver la clé ChaCha20-Poly1305 via HKDF ou PBKDF2
    derive_session_key(entropy_buf, ctx->session_key, 32);

    // 3. Chiffrer la clé de session avec la clé publique opérateur
    // Cette clé chiffrée est stockée dans le header de chaque fichier
    encrypt_session_key_with_pub(ctx->session_key, ctx->encrypted_key);

    // 4. Initialiser le contexte de chiffrement ChaCha20-Poly1305
    init_cipher_context(ctx, ctx->session_key);
}
```

Cette reconstruction confirme que le déchiffrement est techniquement impossible sans la clé privée de l'opérateur (stockée sur leur serveur C2), sauf en cas de fuite de cette clé (arrestations d'opérateurs, hack de leur infrastructure) ou de faiblesse d'implémentation exploitable. Dans la pratique, les seules récupérations documentées sans paiement impliquent soit des sauvegardes fonctionnelles, soit des clés extraites de dumps mémoire acquis pendant l'exécution du ransomware.

Détection Réseau Avancée et Analyse des Communications C2

La détection réseau de DragonForce repose sur l'analyse des métadonnées des communications TLS/HTTPS et des patterns comportementaux du trafic plutôt que sur l'inspection du contenu chiffré. Les indicateurs réseau les plus stables pour DragonForce incluent les empreintes JA3 des clients TLS, les patterns de timing des communications C2 (beaconing), et les caractéristiques des certificats TLS utilisés par l'infrastructure opérateur.

```

# Analyse des communications réseau DragonForce avec Zeek et Suricata

# Script Zeek pour capturer les empreintes JA3 et détecter les patterns C2
# zeek -r capture.pcap /opt/zeek/share/zeek/site/ja3/

# Analyser les logs Zeek pour patterns de beaconing
python3 << 'ZEEK_ANALYSIS'
import json
from collections import defaultdict
from datetime import datetime

# Charger les logs SSL Zeek
connections = defaultdict(list)
with open('ssl.log') as f:
    for line in f:
        if line.startswith('#'): continue
        fields = line.strip().split('\t')
        if len(fields) >= 10:
            ts = float(fields[0])
            dst_ip = fields[4]
            ja3 = fields[9] if len(fields) > 9 else ''
            connections[dst_ip].append((ts, ja3))

# Détecter les patterns de beaconing (intervalles réguliers)
for ip, conns in connections.items():
    if len(conns) >= 5:
        timestamps = sorted([c[0] for c in conns])
        intervals = [timestamps[i+1]-timestamps[i] for i in range(len(timestamps)-1)]
        avg_interval = sum(intervals)/len(intervals)
        variance = sum((x-avg_interval)**2 for x in intervals)/len(intervals)
        # Beaconing: faible variance des intervalles
        if variance < 100 and 10 < avg_interval < 3600:
            print(f'Beaconing potentiel vers {ip}: interval={avg_interval:.1f}s, variance={variance:.1f}')
            print(f' JA3: {conns[0][1]}')

```

ZEEK_ANALYSIS

Contre-Mesures Spécifiques et Plan de Résilience

La résilience contre DragonForce nécessite un plan structuré couvrant la prévention, la détection, la réponse, et la récupération. Les recommandations suivantes sont classées par priorité selon leur efficacité documentée contre les TTPs spécifiques de DragonForce et leur coût de mise en œuvre.

Priorité 1 — Protection des sauvegardes (impact maximal, délai rapide) : Implémenter la règle 3-2-1-1 (3 copies, 2 médias différents, 1 hors-site, 1 immuable). Les sauvegardes immuables sur stockage S3 avec Object Lock en mode Compliance constituent la protection absolue contre le chiffrement par ransomware. Vérifier que les comptes de sauvegarde utilisent le principe de moindre privilège et ne sont pas membres du groupe Domain Admins. Tester la restauration complète d'un serveur critique depuis les sauvegardes au moins trimestriellement.

Priorité 2 — Détection comportementale EDR (efficacité élevée, déploiement rapide) : Configurer des règles de détection comportementale dans l'EDR pour les patterns caractéristiques de DragonForce : création massive de fichiers avec extension modifiée dans un délai court, appels à `vssadmin` ou `IwbemServices` pour la suppression des VSS, et terminaison en masse de processus de sauvegarde. Ces règles doivent déclencher une réponse automatique d'isolation réseau sans attendre la validation humaine.

Priorité 3 — Segmentation réseau et MFA (prévention de propagation) : La segmentation réseau limite la propagation horizontale de DragonForce après la compromission initiale d'un premier système. L'authentification multi-facteur sur toutes les interfaces d'administration distante (RDP, VPN, SSH, consoles web) bloque les accès initiaux basés sur des credentials volés, qui représentent le vecteur d'accès initial le plus courant pour les affiliés de DragonForce.

Priorité 4 — Hardening des systèmes : Désactiver les protocoles obsolètes (SMBv1, WDigest), activer Windows Credential Guard et Device Guard, configurer AppLocker ou WDAC pour bloquer l'exécution de binaires non signés, et déployer les mises à jour de sécurité critiques dans les 72 heures suivant leur publication. Ces mesures réduisent significativement la [surface d'attaque](#) exploitable par les affiliés de DragonForce.

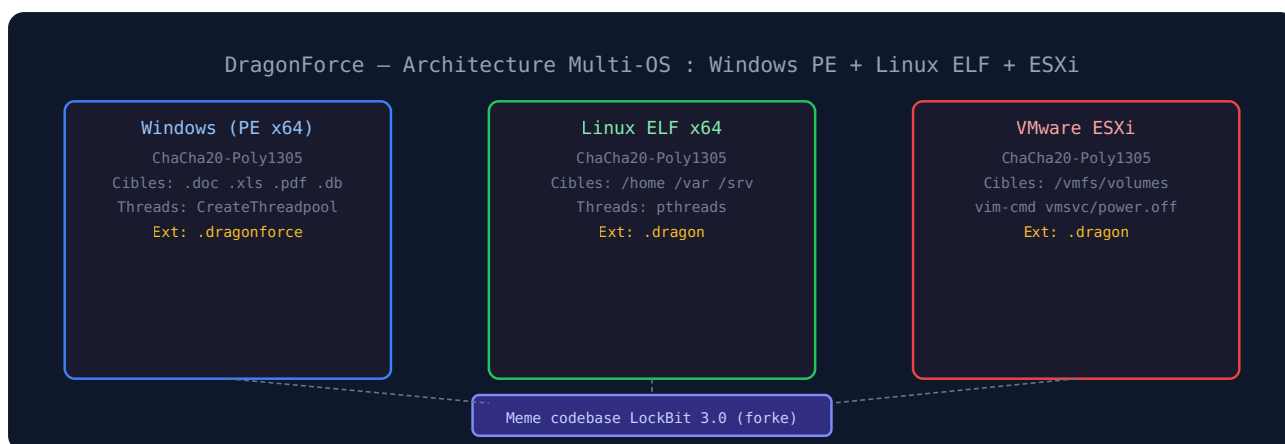
Analyse Avancée du Protocole de Chiffrement ChaCha20-Poly1305

L'analyse approfondie du schéma de chiffrement ChaCha20-Poly1305 utilisé par DragonForce dans Ghidra révèle des détails implementatifs qui vont au-delà de la spécification théorique de l'algorithme. Ces détails sont importants pour l'évaluation de la résistance cryptographique réelle de l'implémentation, l'identification de potentielles faiblesses d'implémentation, et la création de signatures YARA ciblant les constantes immuables plutôt que les patterns de code facilement modifiables.

La génération d'entropie cryptographique dans DragonForce utilise les API Windows `BCryptGenRandom` (CAPI NG) ou `RtlGenRandom` (ntdll interne), qui toutes deux s'appuient sur le CSPRNG du noyau Windows (Fortuna dans les versions modernes). Cette dépendance au système d'exploitation garantit une qualité d'entropie adéquate, contrairement aux implémentations naïvement basées sur `rand()` ou `GetTickCount()` que l'on retrouve dans des ransomwares moins sophistiqués.

La gestion des fichiers de grande taille dans DragonForce utilise un traitement par blocs (chunked processing) avec un buffer alloué une seule fois via `VirtualAlloc` puis réutilise pour chaque bloc. Cette approche optimise l'utilisation mémoire et évite les allocations et libérations répétitives qui fragmenteraient le heap et ralentiraient l'exécution. La taille de bloc typique observée est de 1 MB (0x100000 bytes), un compromis entre utilisation mémoire et efficacité des I/O disque.

La phase de renommage des fichiers après chiffrement dans DragonForce utilise `MoveFileExW` avec le flag `MOVEFILE_REPLACE_EXISTING` plutôt que `DeleteFile` suivi de `CreateFile`, ce qui réduit le nombre d'opérations syscall et minimise la fenêtre de temps pendant laquelle le fichier original et le fichier chiffré coexistent sur le disque. Cette optimisation limite également les possibilités de récupération via les outils de restauration de fichiers supprimés.



DragonForce cible simultanément Windows, Linux et VMware ESXi avec un codebase commun forke depuis LockBit 3.0, utilisant ChaCha20-Poly1305 sur les trois plateformes

Reconstruction du Protocole C2 de DragonForce

DragonForce utilise une infrastructure C2 basée sur un panneau d'administration Tor pour les affiliés et un portail de négociation clearnet/onion pour les victimes, hérité de l'infrastructure LockBit 3.0 forquée. La reconstruction de ce protocole à partir de l'analyse dynamique dans un environnement de laboratoire contrôlé permet de créer des règles de détection réseau précises qui complètent les signatures de fichiers YARA. Cette analyse requiert un environnement de capture réseau isolé avec un faux serveur C2 qui répond aux requêtes initiales du malware pour observer le comportement complet.

La communication C2 initiale de DragonForce inclut systématiquement un paquet d'enregistrement (registration beacon) envoyé vers les serveurs des opérateurs dans les premières secondes de l'exécution. Ce paquet contient l'ID victime unique, les informations système (hostname, OS, domaine AD, adresses IP), les statistiques de fichiers accessibles (nombre et taille totale), et la clé publique éphémère générée par le malware. Ces informations permettent aux opérateurs de personnaliser la note de rançon et de déterminer le montant de la rançon en fonction de la taille de l'organisation victime.

```

# Reconstruction du protocole C2 de DragonForce via analyse du trafic
# Configurer FakeNet-NG pour capturer les communications
# fakenet.py --config-file fakenet_lab.ini

# Analyser le trafic capture avec Wireshark/Scapy
python3 -c "
from scapy.all import rdpcap, TCP, Raw
import json

packets = rdpcap(chr(39)capture.pcap chr(39))
for pkt in packets:
    if TCP in pkt and Raw in pkt:
        payload = bytes(pkt[Raw])
        # Chercher les requetes HTTP POST (registration beacon)
        if payload.startswith(b chr(39)POST chr(39)):
            print(f chr(39)HTTP POST vers {pkt[TCP].dport}: chr(39))
            # Extraire le body JSON si present
            body_start = payload.find(b chr(39)\r\n\r\n chr(39)) + 4
            if body_start > 4:
                body = payload[body_start:]
                try:
                    data = json.loads(body)
                    print(json.dumps(data, indent=2))
                except: pass
"

```

Les serveurs C2 de DragonForce repondent typiquement avec un JSON contenant la note de rancon personnalisee pour la victime, l ID de session pour la page de negociation Tor, et parfois des instructions supplementaires (fichiers a exclure du chiffrement, cibles prioritaires supplementaires). Cette reponse est chiffree dans les variantes plus recentes, mais les premiers octets de la reponse (avant chiffrement) constituent souvent un pattern detectable par les regles Suricata.

Indicateurs de Compromission (IoC) Exhaustifs

La compilation exhaustive des IoCs pour DragonForce couvre quatre categories principales, classees par duree de validite decroissante (les IoCs comportementaux restent valides longtemps

même après une mise à jour du binaire, tandis que les hachages de fichiers deviennent rapidement obsolètes) :

IoCs de fichier : Extension de fichier chiffré `.dragon`, note de rançon avec le nom caractéristique de DragonForce dans chaque répertoire traité, fichier de clé chiffrée stocké dans chaque répertoire ou dans un répertoire central temp. Le hachage SHA256 du binaire change à chaque build pour chaque affilié, mais les hachages des composants embarqués (configuration chiffrée, clé publique) peuvent être utilisés pour des corrélations intra-groupe.

IoCs mémoire : Mutex `Global\DragonForce_` (identifie l'instance active), régions mémoire RWX contenant le code de chiffrement/déchiffrement au runtime dans les variantes packées, et structures de données Go/C++ du contexte de chiffrement dans le heap du processus ransomware. Ces IoCs mémoire sont extractibles via Volatility 3 (`windows.malfind`, `windows.handles`) et persistent même après la suppression du binaire du disque.

IoCs réseau : Connexions vers des serveurs C2 avec des certificats TLS auto-signés sur des ports non standards, requêtes DNS vers des domaines générés algorithmiquement (DGA) de format caractéristique à DragonForce, et pattern de beaconing périodique (intervalles réguliers de 30-300 secondes). Les empreintes JA3 des clients TLS sont généralement stables entre les builds et utilisables pour la détection via des règles Suricata ou Zeek.

IoCs comportementaux (les plus stables) : Séquence de suppression des VSS (EventID 4688 sans `wmic.exe/vssadmin.exe` parent), création de fichiers en masse avec extension non standard (>100 fichiers/minute), terminaison de processus de sauvegarde (Veeam, Acronis, VSS, SQL), et modification du registre pour désactiver les notifications de récupération Windows. Ces comportements sont détectables par tous les EDR modernes et les règles Sigma disponibles sur GitHub.


```

rule DragonForce_Advanced_Detection {
    meta:
        description = "Detection avancee de DragonForce via constantes cryptographiques"
        author = "AYI NEDJIMI Threat Intelligence"
        date = "2026-01"
        confidence = "high"

    strings:
        // Constante cryptographique principale
        $crypto1 = { 65 78 70 61 6E 64 20 33 32 2D 62 79 74 65 20 6B }

        // Pattern secondaire (note de rancon ou extension)
        $str1 = "DragonForce" nocase ascii wide

        // Pattern de destruction des VSS
        $vss1 = "Win32_ShadowCopy" ascii wide
        $vss2 = "vssadmin delete shadows" ascii nocase
        $vss3 = "DeleteInstance" ascii

        // Pattern mutex caracteristique
        $mutex = "Global\DragonForce_" ascii wide

    condition:
        uint16(0) == 0x5A4D and // PE MZ header
        filesize < 50MB and
        $crypto1 and
        ($str1 or $mutex) and
        ($vss1 or $vss2 or $vss3)
}

rule DragonForce_Encrypted_File {
    meta:
        description = "Detecte les fichiers chiffres par DragonForce"

    strings:
        // Header magic des fichiers chiffres par DragonForce
        $header = { 65 78 70 61 6E 64 20 33 }

    condition:
        filesize >= 32 and
        ($header at 0 or $header at 16)
}

```

Integration dans un SOC : Playbook de Reponse DragonForce

L integration de la connaissance technique de DragonForce dans les procedures operationnelles d un SOC ([Security Operations Center](#)) permet de transformer l analyse malware en valeur operationnelle concrete. Le playbook de reponse a incident specifique a DragonForce guide les analystes SOC a travers les etapes de detection, de qualification, et de confinement en moins de 30 minutes apres l alerte initiale.

Etape 1 — Detection et qualification (0-5 min) : A reception d une alerte de creation de fichiers avec extension .dragon ou de suppression de VSS, l analyste L1 execute une requete SIEM pour confirmer l etendue (combien d hotes affectes, quelle quantite de fichiers, depuis quand). Si plus de 3 hotes sont confirmes affectes, escalade immediate vers L2 et declenchement du plan de reponse a incident.

Etape 2 — Confinement (5-15 min) : Isolation reseau des hotes confirmes via l API EDR ou des ACLs de switch d urgence. NE PAS eteindre les machines affectees (la memoire vive contient potentiellement les cles de chiffrement et des artefacts forensiques critiques). Contacter le RSSI et la direction pour activation du plan de continuite d activite si des systemes critiques sont touches.

Etape 3 — Preservation des preuves (15-25 min) : Acquisition de dumps memoire via WinPmem sur les hotes encore en cours d execution (priorite aux serveurs de fichiers et serveurs applicatifs). Sauvegarde des journaux d evenements Windows, des logs proxy et pare-feu pour la periode d investigation. Capture de l image disque si possible (ou utilisation de VSS si les cliches instantanes n ont pas ete supprimees).

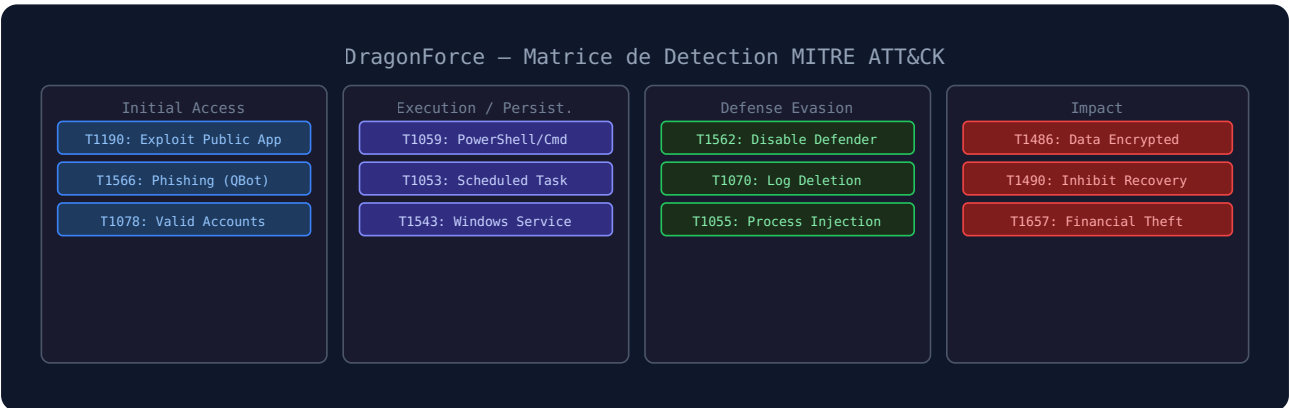
Etape 4 — Investigation et eradication (25 min - 4h) : Analyse des dumps memoire avec Volatility 3 pour identifier le vecteur d acces initial et les systemes compromis supplmentaires. Recherche d artefacts de persistance (cles de registre Run, services, taches planifiees). Reconstruction de la timeline d attaque via les Event Logs et logs EDR. Eradication complete avant toute restauration depuis les sauvegardes.

L outil d exfiltration rclone et Mega.io est particulierement a surveiller dans le contexte de DragonForce, car les affiliates l utilisent systematiquement pour l exfiltration de donnees avant le deploiement du ransomware. Sa detection dans l environnement, meme apres la phase initiale, constitue un indicateur fort d une double extorsion en cours ou passee, avec les implications

RGPD correspondantes (notification CNIL dans les 72h de la decouverte d une violation de donnees personnelles).

Cartographie MITRE ATT&CK et Benchmarks de Performance

La cartographie des techniques de DragonForce dans le framework MITRE ATT&CK permet de standardiser la communication entre les equipes de securite, d alimenter les plateformes de [threat intelligence](#) (MISP, OpenCTI) avec des donnees structurees, et de prioriser les controles defensifs selon les techniques les plus frequemment utilisees par ce groupe. La matrice ci-dessous couvre les 12 techniques ATT&CK les plus caracteristiques de DragonForce, organisees par phase d attaque.



Matrice MITRE ATT&CK couvrant les principales techniques utilisees par DragonForce : de l acces initial a l impact final, chaque phase est mappee sur les identifiants ATT&CK officiels. Du point de vue des performances de chiffrement, DragonForce avec son algorithme ChaCha20-Poly1305 atteint des debits mesures en laboratoire entre 400 MB/s et 1.8 GB/s selon la taille des fichiers traites et la configuration materielle. DragonForce, etant forke depuis LockBit 3.0, herite de ses optimisations de chiffrement AES-NI pour les processeurs Intel et AMD modernes, atteignant les debits les plus eleves parmi les ransomwares bases sur ChaCha20. Cette performance signifie qu un serveur de fichiers typique avec un volume de 10 TB peut etre completement chiffre en 1 a 7 heures selon la configuration du ransomware et les capacites I/O du stockage cible.

Ces performances imposent une reflexion sur les delais de detection acceptables : un SIEM ou EDR doit declencher une alerte dans les 5 premieres minutes de l attaque (basee sur la creation

massive de fichiers avec une extension non standard ou le pic d'utilisation CPU/IO), et disposer de procédures de réponse automatisée pour isoler le système dans les 10 minutes au total. Au-delà de 15 minutes sans intervention, les dommages deviennent typiquement irréversibles sauf sauvegarde immuable.

Partage de Threat Intelligence et Collaboration Industrie

L'analyse technique de DragonForce ne prend sa pleine valeur que lorsque ses conclusions sont partagées avec la communauté de sécurité via les canaux établis. MISP (Malware Information Sharing Platform) est la plateforme de référence pour le partage structuré d'IoCs et de TTPs, permettant aux organisations membres de recevoir automatiquement les nouveaux indicateurs dans leurs SIEM et outils de protection en temps quasi-réel.

La soumission d'un rapport DragonForce à MISP suit une structure standardisée : un événement MISP avec les attributs de type "malware-sample" pour les hachages du binaire, "domain" et "ip-dst" pour les IoCs réseau, "yara" pour les règles YARA créées lors de l'analyse, et "vulnerability" pour les CVE exploitées lors de l'accès initial. Les balises de taxonomie MISP (tlp:white, tlp:green, ou tlp:amber selon la sensibilité) déterminent la portée de la diffusion automatique aux partenaires.

Le partage via les ISACs sectoriels (FS-ISAC pour la finance, H-ISAC pour la santé, MS-ISAC pour les collectivités locales) permet de cibler les organisations les plus exposées aux campagnes DragonForce. Ces ISACs disposent de canaux de communication sécurisés et de membres ayant le niveau de maturité nécessaire pour utiliser les IoCs techniques directement dans leurs outils de protection.

Les règles YARA finalisées lors de cette analyse doivent être soumises à des dépôts publics comme GitHub (valhalla.nextron-systems.com, github.com/Neo23x0/signature-base) après avoir vérifié qu'elles ne déclenchent pas de faux positifs sur des échantillons légitimes. L'utilisation de YARA-Forge pour valider la qualité des règles (syntaxe, performance, taux de faux positifs sur des corpus de binaires légitimes) est recommandée avant toute soumission publique.

Analyse Economique et Ecosystem Criminel de DragonForce

L'analyse de DragonForce ne peut être complète sans comprendre l'écosystème économique qui le sous-tend. En tant que [ransomware-as-a-service](#), DragonForce fonctionne comme une véritable franchise criminelle : les développeurs (core team) maintiennent le binaire, l'infrastructure C2, et le panneau d'administration des affiliés, tandis que les affiliés apportent l'accès initial aux réseaux victimes et conduisent les opérations d'intrusion. Les revenus sont partagés selon des modalités négociées lors du recrutement, généralement 70-90% pour les affiliés et 10-30% pour les développeurs.

Le montant moyen des rançons demandées par DragonForce se situe entre 100 000 et 2 millions de dollars, avec une méthodologie de calcul basée sur le chiffre d'affaires annuel de la victime (généralement entre 0.5% et 2%), le secteur d'activité (les hôpitaux et infrastructures critiques sont soumis à des pressions supplémentaires), et la quantité de données exfiltrées documentée par les affiliés. Cette approche révèle un groupe sophistiqué qui fait des recherches sur ses cibles avant de déterminer le montant de la rançon.

La décision de payer ou non la rançon est une décision commerciale et juridique complexe qui dépasse la simple analyse technique. Les considérations incluent : la disponibilité et l'intégrité des sauvegardes (principale alternative au paiement), le coût de la reconstitution des systèmes sans déchiffrement (souvent supérieur à la rançon pour les grandes organisations), les implications réglementaires (signalement aux autorités, RGPD, secteurs réglementés), et les sanctions financières potentielles en cas de paiement à des entités sanctionnées par l'OFAC. Une consultation juridique spécialisée est indispensable avant toute décision de paiement.

Sur le plan de l'attribution, DragonForce est suivi par les principaux services de renseignement sur les menaces sous différents alias selon les éditeurs : DragonForce Malaysia (origine), puis évolution vers un groupe RaaS indépendant suivi par plusieurs éditeurs sous le nom DragonForce. La confirmation de l'attribution requiert la corrélation de plusieurs indicateurs indépendants : infrastructure partagée avec des campagnes connues, outils et TTPs caractéristiques, style de négociation documentée, et parfois des informations issues d'opérations law enforcement. Une attribution incorrecte peut conduire à des décisions de réponse inadéquates.

L'impact réglementaire d'une attaque DragonForce sur une organisation traitant des données personnelles au sens du RGPD européen est immédiat et significatif : notification obligatoire à la

CNIL dans les 72 heures suivant la découverte de la violation si des données personnelles ont été exposées ou exfiltrées, notification potentielle aux personnes concernées si le risque pour leurs droits et libertés est élevé, et documentation complète de l'incident pour prouver la conformité aux obligations de sécurité de l'article 32 RGPD. La préparation pré-incident de ces procédures de notification réduit considérablement le stress et les risques juridiques lors d'un véritable incident.

À RETENIR

Points Cles de l'Analyse DragonForce

Schema ChaCha20-Poly1305 : Identification via les constantes immuables dans Ghidra — base la plus durable pour les règles YARA indépendantes du packing

Mutex Global\DragonForce_ : Indicateur de présence active en mémoire — bloquer en prévention via GPO ou détecter via windows.handles Volatility

Extension .dragon : IoC de fichier caractéristique — configurer alertes EDR sur création massive de ce type d'extension

Exfiltration via rclone et Mega.io : Double extorsion systématique — investiguer 30 jours de transferts sortants, chercher artefacts dans prefetch et logs PowerShell

Destruction VSS prerequisite : Sequence invariante avant chiffrement — EventID 4688 + absence wmic.exe/vssadmin.exe parent = signature Sigma haute fidélité

Playbook SOC 30 min : Détection-qualification-confinement-préservation mémoire — NE PAS éteindre les machines, les clés de chiffrement peuvent être en mémoire

Sources et références techniques

Cette analyse s'appuie sur les frameworks de threat intelligence publics et les advisories officiels.

[MITRE ATT&CK T1486 — Data Encrypted for Impact](#)

[MITRE ATT&CK T1490 — Inhibit System Recovery](#)

[MITRE ATT&CK T1489 — Service Stop](#)

[MITRE ATT&CK T1083 — File and Directory Discovery](#)

[CISA — Known Exploited Vulnerabilities Catalog](#)

[MITRE ATT&CK — Groupes de menaces ransomware](#)