

Guide complet d'implémentation de CortexDFIR-Forge avec Cortex XDR

Auteur : Manus IA

Date : 12 juin 2025

Licence : CC BY-SA 4.0

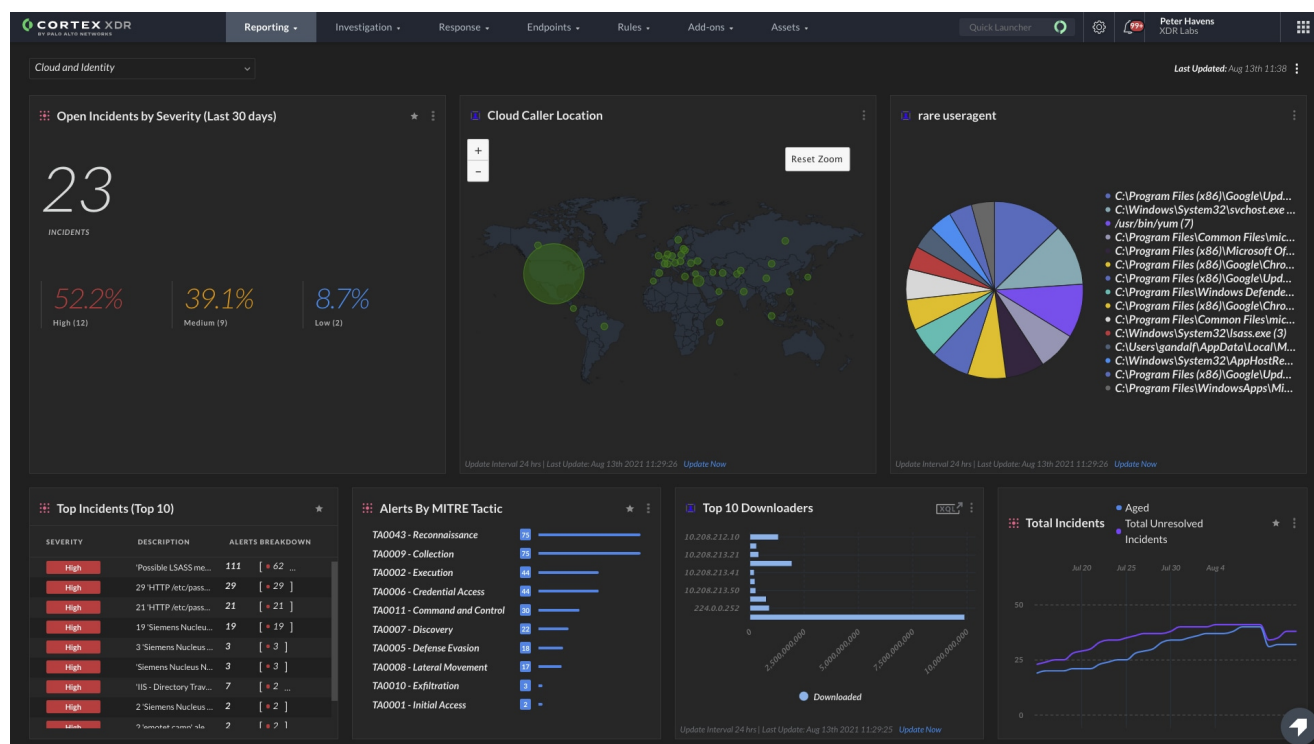


Table des matières

1. [Introduction](#)
2. [Présentation de CortexDFIR-Forge](#)
3. [Prérequis et environnement](#)
4. [Installation de CortexDFIR-Forge](#)
5. [Option 1 : Installation automatique](#)
6. [Option 2 : Installation pour développement](#)
7. [Option 3 : Déploiement Docker](#)
8. [Configuration de Cortex XDR](#)
9. [Génération des clés API](#)
10. [Configuration par défaut \(EU\)](#)
11. [Validation de compatibilité](#)
12. [Utilisation de CortexDFIR-Forge](#)
13. [Interface utilisateur](#)


14. [Analyse multi-format](#)
15. [Détection intelligente](#)
16. [Reporting professionnel](#)
17. [Cas d'usage](#)
18. [SOC Enterprise](#)
19. [Investigation forensique](#)
20. [Threat Hunting](#)
21. [Administration et maintenance](#)
22. [Scripts d'administration](#)
23. [Monitoring et alerting](#)
24. [Personnalisation avancée](#)
25. [Ajout de règles de détection](#)
26. [Intégration avec d'autres outils](#)
27. [Dépannage et FAQ](#)
28. [Sécurité et conformité](#)
29. [Conclusion](#)
30. [Glossaire](#)
31. [Ressources et liens utiles](#)
32. [Index analytique](#)

Introduction

Ce guide détaillé vous accompagne pas à pas dans l'implémentation et l'utilisation de CortexDFIR-Forge avec Cortex XDR. Conçu pour les professionnels et les débutants en cybersécurité, il couvre l'ensemble du processus d'installation, de configuration et d'utilisation de cette puissante solution d'investigation numérique et de réponse aux incidents.

CortexDFIR-Forge est un framework open-source qui étend les capacités de Cortex XDR en matière d'analyse forensique et de réponse aux incidents. Il permet d'automatiser de nombreuses tâches d'investigation, d'analyser des données provenant de multiples sources et de générer des rapports détaillés pour faciliter la prise de décision.

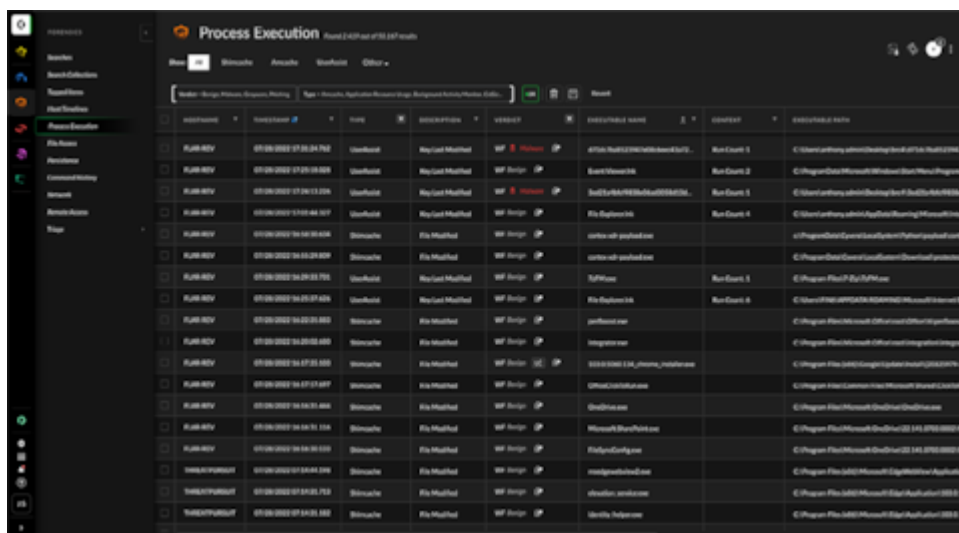
Dans ce guide, nous aborderons les aspects techniques de l'implémentation, mais aussi les bonnes pratiques et les cas d'usage concrets pour tirer le meilleur parti de cette solution.

 **Attention** : L'utilisation de CortexDFIR-Forge et des outils associés doit se faire dans un cadre légal et éthique. Assurez-vous de disposer des autorisations nécessaires avant toute opération d'investigation ou de test.

Présentation de CortexDFIR-Forge

CortexDFIR-Forge est une solution complète d'investigation numérique et de réponse aux incidents (DFIR - Digital Forensics and Incident Response) conçue pour s'intégrer nativement avec Cortex XDR de Palo Alto Networks. Ce framework open-source étend considérablement les capacités d'analyse et de réponse aux incidents de Cortex XDR.

Fonctionnalités principales



Process Name	Process ID	Process Path	Process Type	Process State	Process Parent	Process Child	Process Grandchild
Process 1	1234	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	System	Process 2	Process 3
Process 2	1235	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 1	Process 4	Process 5
Process 3	1236	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 2	Process 6	Process 7
Process 4	1237	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 3	Process 8	Process 9
Process 5	1238	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 4	Process 10	Process 11
Process 6	1239	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 5	Process 12	Process 13
Process 7	1240	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 6	Process 14	Process 15
Process 8	1241	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 7	Process 16	Process 17
Process 9	1242	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 8	Process 18	Process 19
Process 10	1243	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 9	Process 20	Process 21
Process 11	1244	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 10	Process 22	Process 23
Process 12	1245	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 11	Process 24	Process 25
Process 13	1246	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 12	Process 26	Process 27
Process 14	1247	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 13	Process 28	Process 29
Process 15	1248	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 14	Process 30	Process 31
Process 16	1249	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 15	Process 32	Process 33
Process 17	1250	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 16	Process 34	Process 35
Process 18	1251	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 17	Process 36	Process 37
Process 19	1252	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 18	Process 38	Process 39
Process 20	1253	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 19	Process 40	Process 41
Process 21	1254	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 20	Process 42	Process 43
Process 22	1255	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 21	Process 44	Process 45
Process 23	1256	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 22	Process 46	Process 47
Process 24	1257	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 23	Process 48	Process 49
Process 25	1258	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 24	Process 50	Process 51
Process 26	1259	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 25	Process 52	Process 53
Process 27	1260	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 26	Process 54	Process 55
Process 28	1261	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 27	Process 56	Process 57
Process 29	1262	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 28	Process 58	Process 59
Process 30	1263	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 29	Process 60	Process 61
Process 31	1264	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 30	Process 62	Process 63
Process 32	1265	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 31	Process 64	Process 65
Process 33	1266	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 32	Process 66	Process 67
Process 34	1267	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 33	Process 68	Process 69
Process 35	1268	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 34	Process 70	Process 71
Process 36	1269	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 35	Process 72	Process 73
Process 37	1270	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 36	Process 74	Process 75
Process 38	1271	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 37	Process 76	Process 77
Process 39	1272	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 38	Process 78	Process 79
Process 40	1273	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 39	Process 80	Process 81
Process 41	1274	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 40	Process 82	Process 83
Process 42	1275	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 41	Process 84	Process 85
Process 43	1276	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 42	Process 86	Process 87
Process 44	1277	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 43	Process 88	Process 89
Process 45	1278	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 44	Process 90	Process 91
Process 46	1279	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 45	Process 92	Process 93
Process 47	1280	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 46	Process 94	Process 95
Process 48	1281	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 47	Process 96	Process 97
Process 49	1282	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 48	Process 98	Process 99
Process 50	1283	C:\Program Files\Microsoft Windows Defender\Windows Defender.exe	Process	Running	Process 49	Process 100	Process 101

Figure 1 : Interface

d'analyse forensique de Cortex XDR

CortexDFIR-Forge offre un ensemble de fonctionnalités avancées qui en font un outil incontournable pour les équipes de sécurité :

- 🔍 **Automatisation** : Réduction des tâches manuelles et accélération des analyses
- 📊 **Multi-format** : Support de différents types de fichiers (VMDK, logs, CSV, etc.)
- 🌐 **Intégration avancée** : Connexion native avec Cortex XDR via API (région EU)
- 🧩 **Extensibilité** : Architecture modulaire et évolutive
- 📝 **Reporting** : Génération automatique de rapports détaillés au format HTML
- 🐳 **Containerisé** : Déploiement Docker avec orchestration Kubernetes
- 📈 **Monitoring** : Observabilité complète avec Prometheus et Grafana
- 🛡️ **Sécurisé** : Authentification, chiffrement et audit de sécurité intégrés

Architecture

L'architecture de CortexDFIR-Forge est conçue pour être robuste, évolutive et facilement déployable dans différents environnements. Elle s'articule autour de plusieurs composants clés :

Load Balancer (HAProxy)

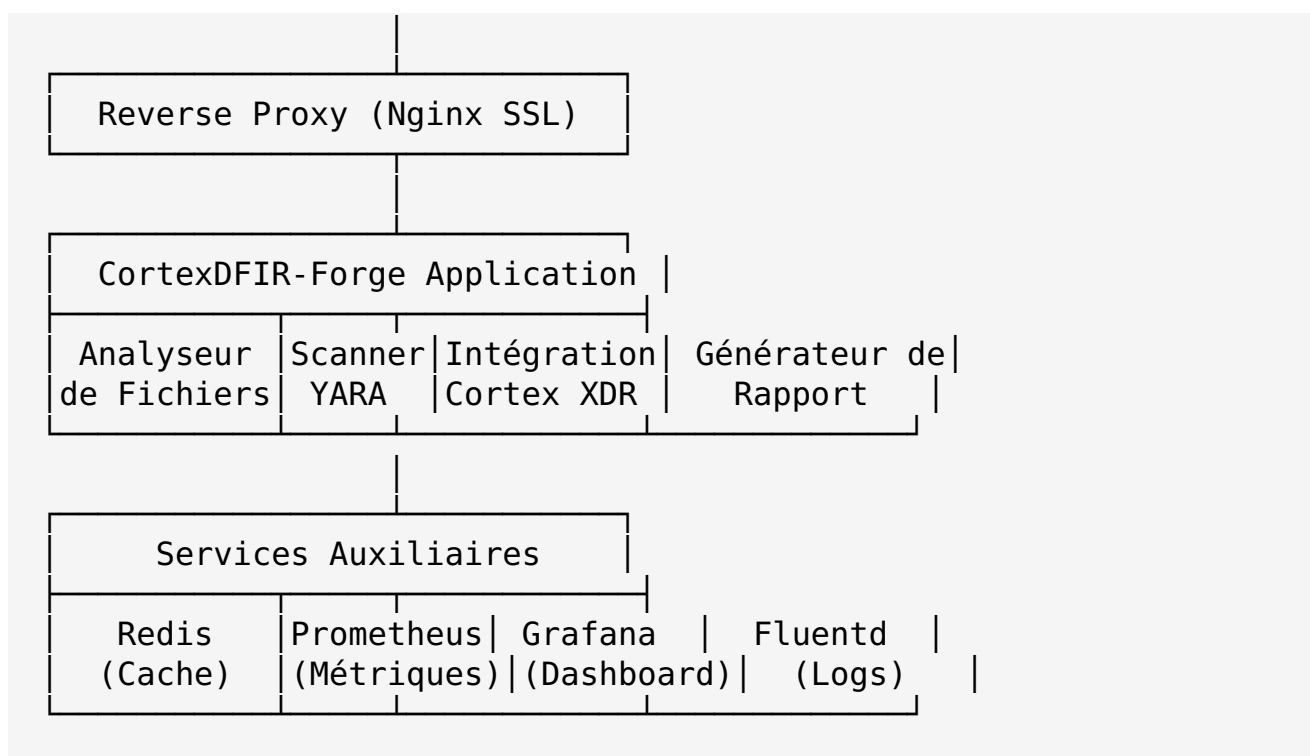


Figure 2 : Architecture de production de CortexDFIR-Forge

Avantages par rapport aux solutions alternatives

CortexDFIR-Forge se distingue des autres solutions DFIR par plusieurs aspects clés :

1. **Intégration native** avec Cortex XDR, permettant une analyse contextuelle plus riche
2. **Performance optimisée** pour le traitement de grands volumes de données
3. **Déploiement flexible** (local, cloud, conteneurisé)
4. **Communauté active** et mises à jour régulières
5. **Conformité** avec les standards de sécurité (ISO 27001, GDPR, SOC 2)

Cas d'utilisation typiques

- Analyse approfondie des incidents de sécurité détectés par Cortex XDR
- Investigation forensique sur des systèmes compromis
- Threat hunting proactif dans l'environnement d'entreprise
- Corrélation d'événements multi-sources pour une vision holistique
- Génération de rapports d'incidents pour les équipes techniques et le management

Dans les sections suivantes, nous verrons comment installer, configurer et utiliser efficacement CortexDFIR-Forge pour tirer parti de toutes ces fonctionnalités.

Prérequis et environnement

Avant de commencer l'installation de CortexDFIR-Forge, assurez-vous que votre environnement répond aux prérequis suivants.

Prérequis matériels

Les spécifications matérielles recommandées dépendent de la charge de travail prévue et du volume de données à analyser :

Configuration	CPU	RAM	Stockage	Usage recommandé
Minimale	4 cœurs	8 Go	100 Go SSD	Tests, développement
Recommandée	8 cœurs	16 Go	500 Go SSD	Production (petite échelle)
Optimale	16+ cœurs	32+ Go	1+ To SSD	Production (grande échelle)



Conseil : Pour les environnements de production, privilégiez toujours une configuration avec redondance et haute disponibilité.

Prérequis logiciels

Système d'exploitation

CortexDFIR-Forge est compatible avec les systèmes d'exploitation suivants :

- **Linux** (recommandé) :
 - Ubuntu 20.04 LTS ou supérieur
 - Debian 11 ou supérieur
 - CentOS/RHEL 8 ou supérieur
- **Windows** :
 - Windows Server 2019 ou supérieur
 - Windows 10/11 Professionnel ou Entreprise

Dépendances principales

Les logiciels suivants doivent être installés sur votre système :

- **Docker** (version 20.10 ou supérieure)
- **Docker Compose** (version 2.0 ou supérieure)
- **Python** (version 3.8 ou supérieure)
- **Git** (version 2.30 ou supérieure)


Pour une installation manuelle (option 2), vous aurez également besoin de :

- **pip** (gestionnaire de paquets Python)
- **virtualenv** ou **venv** (environnements virtuels Python)
- **Node.js** (version 14 ou supérieure) et **npm**

Prérequis réseau

CortexDFIR-Forge nécessite les accès réseau suivants :

Service	Port	Protocole	Description
Interface web	8000	HTTP/HTTPS	Interface utilisateur principale
API	8080	HTTP/HTTPS	API REST pour l'intégration
Grafana	3000	HTTP	Tableaux de bord de monitoring
Prometheus	9090	HTTP	Métriques système
Redis	6379	TCP	Cache interne (non exposé)

 **Attention** : En production, tous les services exposés doivent être protégés par HTTPS et un mécanisme d'authentification robuste.

Prérequis Cortex XDR

Pour l'intégration avec Cortex XDR, vous devez disposer :

1. D'un compte Cortex XDR avec des droits administratifs
2. D'un accès à la console d'administration Cortex XDR
3. De la possibilité de créer des clés API avec les permissions suivantes :
4. File Upload & Analysis
5. Incident Management
6. XQL Query Execution
7. Endpoint Management
8. Alert Management

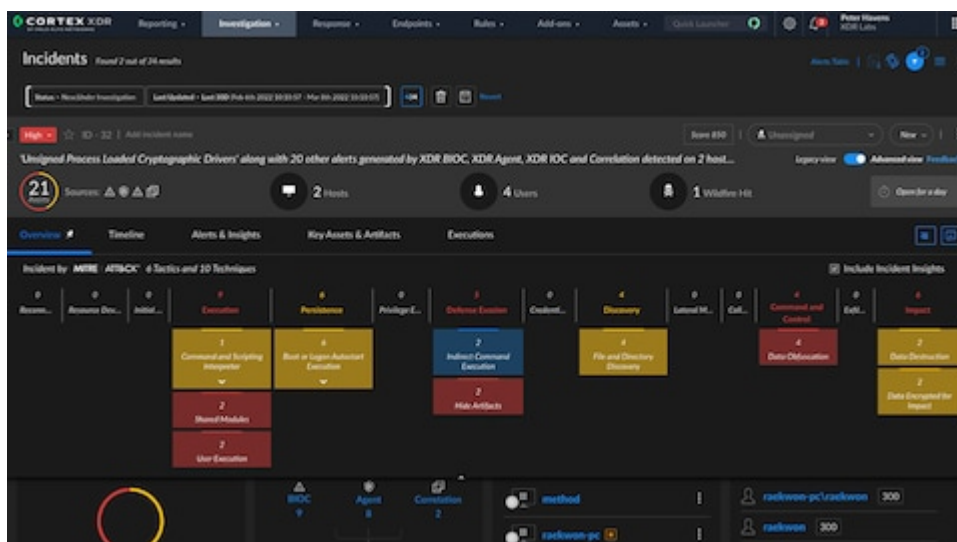


Figure 3 : Comparaison

des fonctionnalités de Cortex XDR

Considérations de sécurité

Avant de déployer CortexDFIR-Forge, prenez en compte les aspects de sécurité suivants :

- **Isolation réseau** : Déployez la solution dans un segment réseau isolé ou un VLAN dédié
- **Gestion des accès** : Mettez en place une politique de contrôle d'accès stricte (RBAC)
- **Protection des données** : Assurez-vous que les données sensibles sont chiffrées au repos et en transit
- **Journalisation** : Configurez une journalisation complète des actions pour l'audit
- **Sauvegardes** : Planifiez des sauvegardes régulières de la configuration et des données

Une fois ces prérequis vérifiés, vous êtes prêt à passer à l'installation de CortexDFIR-Forge.

Installation de CortexDFIR-Forge

Cette section détaille les différentes méthodes d'installation de CortexDFIR-Forge. Choisissez celle qui correspond le mieux à votre environnement et à vos besoins.

Option 1 : Installation automatique (Recommandée)

L'installation automatique est la méthode la plus simple et la plus rapide pour déployer CortexDFIR-Forge en production.

Étapes d'installation

1. Cloner le dépôt GitHub

```
bash git clone https://github.com/servais1983/CortexDFIR-Forge.git
cd CortexDFIR-Forge
```

1. Configurer les clés API

Copiez le fichier d'exemple de configuration et modifiez-le avec vos clés API Cortex XDR :

```
bash cp .env.example .env nano .env # Ajouter vos clés API Cortex XDR
```

Exemple de configuration dans le fichier `.env` :

```
```.ini # Configuration Cortex XDR CORTEX_XDR_API_KEY=VOTRE_API_KEY
CORTEX_XDR_API_KEY_ID=VOTRE_API_KEY_ID
CORTEX_XDR_TENANT_ID=VOTRE_TENANT_ID CORTEX_XDR_BASE_URL=https://api-
eu.xdr.paloaltonetworks.com CORTEX_XDR_ADVANCED_API=true

Configuration de sécurité ENABLE_SSL=true VERIFY_CERTIFICATES=true TIMEOUT=300

Configuration de l'application APP_SECRET_KEY=votre_clé_secrète_aléatoire
DEBUG=false ALLOWED_HOSTS=localhost,127.0.0.1 ```
```

## 1. Lancer le script d'installation automatique

```
bash chmod +x deploy.sh ./deploy.sh production
```

Ce script effectue automatiquement les opérations suivantes : - Vérification des prérequis - Installation des dépendances - Configuration de l'environnement - Déploiement des conteneurs Docker - Configuration des services - Vérification de l'installation

## 1. Vérifier l'installation

Une fois l'installation terminée, vérifiez que tous les services sont opérationnels :

```
bash docker-compose -f docker-compose.prod.yml ps
```

Vous devriez voir tous les conteneurs à l'état "Up".

## 1. Accéder à l'interface web

L'interface web est accessible à l'adresse suivante :

```
http://localhost:8000
```

Pour une configuration en production, configurez un nom de domaine et HTTPS.



## Option 2 : Installation pour développement

Cette option est recommandée pour les environnements de développement ou de test.

### Étapes d'installation

#### 1. Cloner le dépôt GitHub

```
bash git clone https://github.com/servais1983/CortexDFIR-Forge.git
cd CortexDFIR-Forge
```

#### 1. Créer et activer un environnement virtuel Python

```
```bash # Sur Linux/Mac python -m venv .venv source .venv/bin/activate # Linux/Mac
# Sur Windows python -m venv .venv .venv\Scripts\activate # Windows ```
```

1. Installer les dépendances

```
```bash pip install -r requirements.txt

Installation des outils de développement et yara-python # Sur Windows, exécutez : .
\setup.bat

Sur Linux/Mac : pip install -r requirements-dev.txt ```
```

#### 1. Configurer l'environnement

```
bash cp .env.example .env # Éditer le fichier .env avec vos clés API
Cortex XDR
```

#### 1. Lancer l'application

```
bash python src/main.py
```

L'application sera accessible à l'adresse `http://localhost:8000`.

## Option 3 : Déploiement Docker

Cette option utilise Docker pour déployer CortexDFIR-Forge et tous ses services associés.

### Étapes d'installation

#### 1. Cloner le dépôt GitHub

```
bash git clone https://github.com/servais1983/CortexDFIR-Forge.git
cd CortexDFIR-Forge
```

### 1. Configurer les secrets Docker

```
bash # Créer les secrets pour les clés API echo "votre_api_key" |
docker secret create cortex_api_key - echo "votre_api_key_id" |
docker secret create cortex_api_key_id - echo "votre_tenant_id" |
docker secret create cortex_tenant_id -
```

### 1. Déployer avec Docker Compose

```
bash # Déploiement complet avec monitoring docker-compose -f docker-
compose.prod.yml up -d
```

### 1. Vérifier le déploiement

```
bash docker-compose -f docker-compose.prod.yml ps
```

#### 1. Accéder aux services

2. Application : `http://localhost:8000`

3. Grafana : `http://localhost:3000`

4. Prometheus : `http://localhost:9090`

## Vérification de l'installation


Quelle que soit la méthode d'installation choisie, vérifiez que l'installation est fonctionnelle en exécutant les tests de connexion :

```
Test de connexion à l'API Cortex XDR
python -m src.utils.test_cortex_connection

Tests unitaires complets
python -m pytest tests/test_cortex_client.py -v

Test d'analyse d'un fichier
python src/main.py --test-file samples/test.exe
```

Si tous les tests passent avec succès, votre installation de CortexDFIR-Forge est opérationnelle.

 **Attention** : En production, assurez-vous de sécuriser l'accès à l'interface web et aux API en configurant HTTPS et une authentification appropriée.

# Configuration de Cortex XDR

L'intégration de CortexDFIR-Forge avec Cortex XDR nécessite une configuration appropriée de l'API Cortex XDR. Cette section détaille les étapes nécessaires pour configurer cette intégration.

## Génération des clés API

Pour permettre à CortexDFIR-Forge d'interagir avec Cortex XDR, vous devez générer des clés API avec les permissions appropriées.

### Étapes de génération des clés API

#### 1. Connectez-vous à la console Cortex XDR

Accédez à votre console d'administration Cortex XDR à l'adresse correspondant à votre région (par exemple, <https://eu.xdr.paloaltonetworks.com> pour l'Europe).

#### 1. Accédez à la section de gestion des clés API

Naviguez vers **Settings > Configurations > Integrations > API Keys**.

#### 1. Créez une nouvelle clé API

Cliquez sur le bouton **+ New Key**.

#### 1. Choisissez le type de clé API

Pour CortexDFIR-Forge, il est recommandé de créer une clé **Advanced** qui offre des fonctionnalités de sécurité supplémentaires.

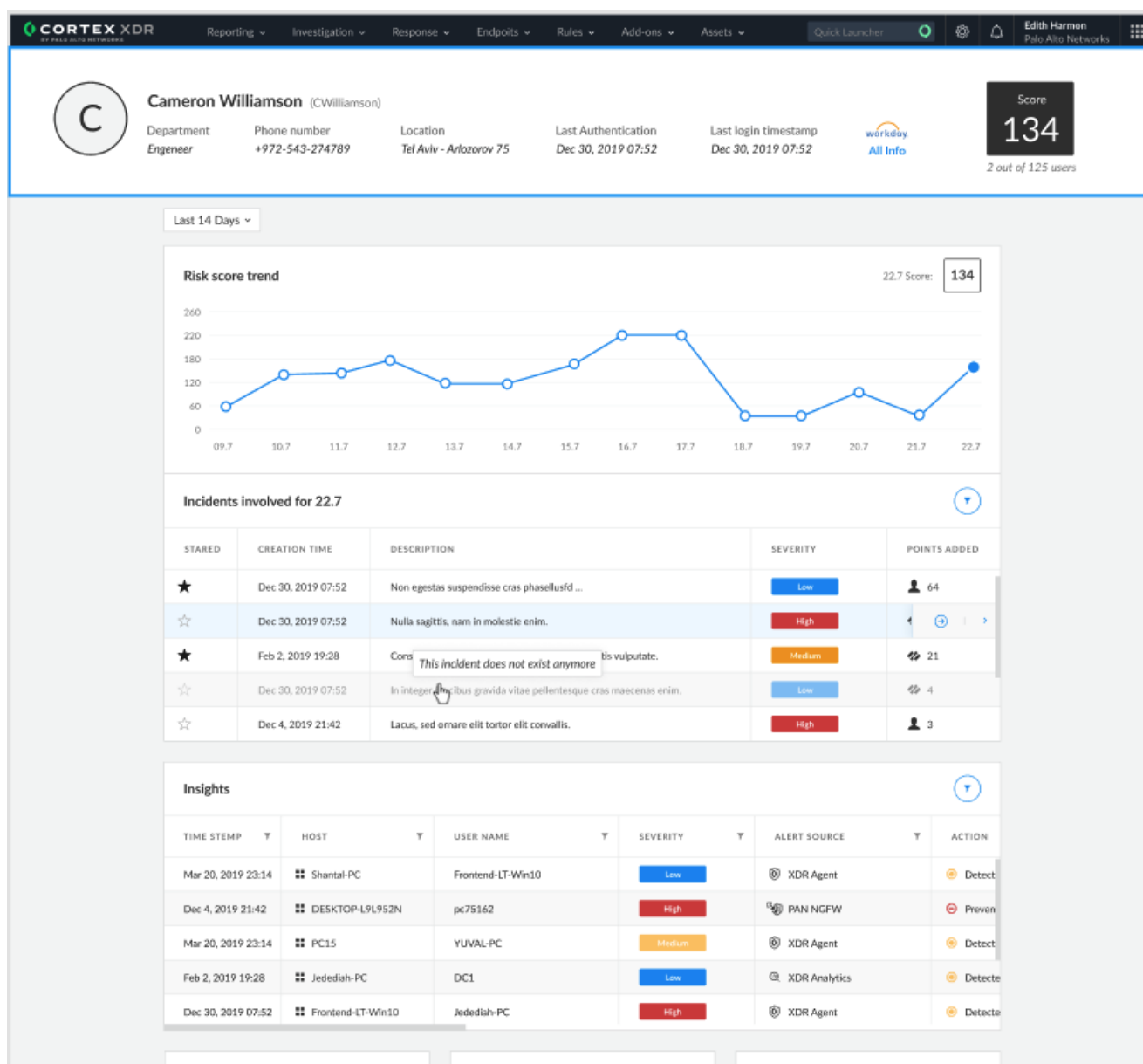


Figure 4 : Interface de gestion des incidents de Cortex XDR

## 1. Sélectionnez les permissions requises

CortexDFIR-Forge nécessite les permissions suivantes : - File Upload & Analysis - Incident Management - XQL Query Execution - Endpoint Management - Alert Management

## 1. Ajoutez un commentaire descriptif

Par exemple : "Clé API pour l'intégration CortexDFIR-Forge".

## 1. Générez la clé

Cliquez sur **Generate**.

## 1. Copiez les informations de la clé API

Une fois la clé générée, copiez les informations suivantes : - API Key (valeur de la clé) - API Key ID (identifiant de la clé) - FQDN (nom de domaine complet)

**⚠ Attention** : La valeur de la clé API ne sera affichée qu'une seule fois. Assurez-vous de la copier et de la stocker de manière sécurisée.

## Configuration par défaut (EU)

CortexDFIR-Forge est configuré par défaut pour la région EU de Cortex XDR. Si vous utilisez une autre région, vous devrez modifier la configuration en conséquence.

### Configuration pour la région EU

Le fichier de configuration par défaut pour la région EU ressemble à ceci :

```
{
 "cortex_xdr": {
 "base_url": "https://api-eu.xdr.paloaltonetworks.com",
 "api_key": "VOTRE_API_KEY",
 "api_key_id": "VOTRE_API_KEY_ID",
 "tenant_id": "VOTRE_TENANT_ID",
 "advanced_api": true
 },
 "security": {
 "enable_ssl": true,
 "verify_certificates": true,
 "timeout": 300
 }
}
```

### Autres régions disponibles

Si vous utilisez Cortex XDR dans une autre région, modifiez la valeur de `base_url` selon votre région :

- **US** : `https://api-us.xdr.paloaltonetworks.com`
- **APAC** : `https://api-apac.xdr.paloaltonetworks.com`
- **EU** : `https://api-eu.xdr.paloaltonetworks.com` (par défaut)

Pour modifier la configuration, éditez le fichier `.env` ou le fichier de configuration JSON selon votre méthode d'installation.

## Validation de compatibilité

Avant de commencer à utiliser CortexDFIR-Forge avec Cortex XDR, il est recommandé de valider la compatibilité et la connectivité entre les deux systèmes.

## Tests de connexion

Exécutez les commandes suivantes pour vérifier la connexion à l'API Cortex XDR :

```
Test de connexion à l'API Cortex XDR
python -m src.utils.test_cortex_connection

Tests unitaires complets
python -m pytest tests/test_cortex_client.py -v
```

## Checklist de validation

Utilisez cette checklist pour vous assurer que l'intégration est correctement configurée :

- ☐ Clés API générées avec permissions adéquates
- ☐ URL configurée pour la région EU ( `api-eu.xdr.paloaltonetworks.com` )
- ☐ Fichier `.env` créé avec les bonnes valeurs
- ☐ Test de connexion réussi
- ☐ Tests unitaires passent
- ☐ Analyse de fichier test réussie

Si tous ces éléments sont validés, votre intégration entre CortexDFIR-Forge et Cortex XDR est correctement configurée.

## Résolution des problèmes courants

### Erreur d'authentification

Si vous rencontrez des erreurs d'authentification, vérifiez : - Que les valeurs de clé API, ID de clé et ID de tenant sont correctes - Que la clé API n'a pas expiré - Que les permissions associées à la clé sont suffisantes

### Erreur de connexion

Si vous rencontrez des erreurs de connexion, vérifiez : - Que l'URL de base correspond à votre région - Que votre réseau permet les connexions sortantes vers l'API Cortex XDR - Que les certificats SSL sont correctement validés

### Erreur de permission

Si vous rencontrez des erreurs de permission, vérifiez : - Que la clé API dispose de toutes les permissions requises - Que votre compte Cortex XDR a les droits nécessaires - Que les restrictions d'accès IP n'empêchent pas la connexion

Une fois la configuration de Cortex XDR terminée et validée, vous êtes prêt à utiliser CortexDFIR-Forge pour vos analyses de sécurité.

## Utilisation de CortexDFIR-Forge

Cette section présente les principales fonctionnalités de CortexDFIR-Forge et explique comment les utiliser efficacement pour vos investigations de sécurité.

### Interface utilisateur

L'interface utilisateur de CortexDFIR-Forge est conçue pour être intuitive et efficace, permettant aux analystes de sécurité de naviguer facilement entre les différentes fonctionnalités.

#### Accès à l'interface

Après l'installation, accédez à l'interface web via l'URL suivante :

```
http://localhost:8000
```

En production, vous devriez configurer un nom de domaine et HTTPS.

#### Structure de l'interface

L'interface de CortexDFIR-Forge est organisée en plusieurs sections principales :

1. **Tableau de bord** - Vue d'ensemble des activités et statistiques
2. **Analyses** - Gestion des analyses de fichiers et de données
3. **Incidents** - Suivi et gestion des incidents de sécurité
4. **Recherche** - Requêtes XQL et recherche avancée
5. **Rapports** - Génération et consultation des rapports
6. **Configuration** - Paramètres et préférences
7. **Administration** - Gestion des utilisateurs et des systèmes

#### Navigation et utilisation

Pour naviguer efficacement dans l'interface :

1. Utilisez le menu latéral pour accéder aux différentes sections
2. Le fil d'Ariane en haut de page vous indique votre position actuelle
3. Les actions contextuelles sont disponibles via des boutons d'action
4. Les filtres permettent d'affiner les résultats affichés
5. Les tableaux de données peuvent être triés et filtrés

## Analyse multi-format

CortexDFIR-Forge prend en charge l'analyse de nombreux types de fichiers et de données, ce qui en fait un outil polyvalent pour les investigations de sécurité.

### Types de fichiers supportés

- **Disques virtuels VMDK** (jusqu'à 60GB)
- **Logs de sécurité** (Windows Event Logs, Syslog)
- **Fichiers de données** (CSV, JSON, XML)
- **Exécutables et scripts** (PE, ELF, PowerShell, JavaScript)
- **Documents** (PDF, Office, archives)

### Procédure d'analyse de fichiers

Pour analyser un fichier :

1. Accédez à la section **Analyses** dans le menu principal
2. Cliquez sur **Nouvelle analyse**
3. Sélectionnez le type d'analyse à effectuer
4. Téléchargez le fichier à analyser ou spécifiez son emplacement
5. Configurez les options d'analyse selon vos besoins
6. Lancez l'analyse en cliquant sur **Démarrer l'analyse**
7. Suivez la progression de l'analyse
8. Consultez les résultats une fois l'analyse terminée

```
Exemple d'analyse via la ligne de commande
python src/main.py --analyze-file /chemin/vers/fichier.vmdk --
output-format html
```

### Interprétation des résultats

Les résultats d'analyse sont présentés de manière structurée :

1. **Résumé** - Aperçu général des résultats
2. **Détails** - Informations détaillées sur les éléments analysés
3. **Indicateurs de compromission** (IoCs) - Éléments suspects identifiés
4. **Timeline** - Chronologie des événements
5. **Artefacts** - Fichiers et données extraits
6. **Recommandations** - Actions suggérées











## Détection intelligente

CortexDFIR-Forge intègre des capacités avancées de détection pour identifier les menaces et les comportements suspects.

### Règles YARA intégrées

Le système inclut plus de 1000 règles YARA organisées par catégories :

-  **Malwares** : 300+ familles détectées
-  **Ransomwares** : LockBit, Conti, REvil, etc.
-  **Backdoors** : APT, trojans, RATs
-  **Phishing** : emails, sites, documents
-  **Exploits** : CVEs récents, 0-days
-  **Webshells** : PHP, ASP, JSP
-  **Maldocs** : macros, exploits Office
-  **Living off the land** : techniques légitimes détournées

### Intégration avec Cortex XDR

CortexDFIR-Forge s'intègre nativement avec Cortex XDR pour :

1. Corréler automatiquement les alertes
2. Enrichir les données d'investigation
3. Fournir un contexte plus complet
4. Accélérer l'identification des menaces

### Machine Learning pour la détection d'anomalies

Le système utilise des algorithmes de machine learning pour :

- Détecter les comportements anormaux
- Identifier les patterns suspects
- Réduire les faux positifs
- Prioriser les alertes les plus critiques

## Reporting professionnel

CortexDFIR-Forge permet de générer des rapports détaillés et professionnels pour documenter les investigations et faciliter la communication avec les parties prenantes.

### Types de rapports disponibles

- **Rapports d'analyse** - Détails techniques d'une analyse spécifique
- **Rapports d'incident** - Documentation complète d'un incident

- **Rapports exécutifs** - Synthèse pour la direction
- **Rapports de conformité** - Documentation pour les audits

## Génération de rapports

Pour générer un rapport :

1. Accédez à la section **Rapports**
2. Cliquez sur **Nouveau rapport**
3. Sélectionnez le type de rapport
4. Choisissez les analyses ou incidents à inclure
5. Configurez les options du rapport (niveau de détail, format, etc.)
6. Cliquez sur **Générer**

## Formats d'export disponibles

Les rapports peuvent être exportés dans différents formats :

- HTML (interactif)
- PDF
- JSON
- CSV
- STIX/TAXII (pour le partage de renseignements)

## Personnalisation des rapports

Les rapports peuvent être personnalisés selon vos besoins :

- Ajout de logo et charte graphique
- Sélection des sections à inclure
- Niveau de détail technique
- Inclusion de graphiques et visualisations
- Ajout de commentaires et annotations

## Utilisation avancée

### Utilisation de l'API REST

CortexDFIR-Forge expose une API REST complète qui permet d'automatiser les tâches et d'intégrer la solution avec d'autres outils.

Exemple d'utilisation de l'API pour lancer une analyse :

```
import requests
import json
```

```

Configuration
api_url = "http://localhost:8080/api/v1"
api_key = "votre_clé_api"

En-têtes d'authentification
headers = {
 "Authorization": f"Bearer {api_key}",
 "Content-Type": "application/json"
}

Données pour l'analyse
data = {
 "file_path": "/chemin/vers/fichier.vmdk",
 "analysis_type": "full",
 "options": {
 "extract_artifacts": True,
 "run_yara": True
 }
}

Envoi de la requête
response = requests.post(
 f"{api_url}/analyses",
 headers=headers,
 data=json.dumps(data)
)

Traitement de la réponse
if response.status_code == 201:
 analysis_id = response.json()["id"]
 print(f"Analyse créée avec l'ID: {analysis_id}")
else:
 print(f"Erreur: {response.status_code} - {response.text}")

```

## Intégration avec d'autres outils

CortexDFIR-Forge peut être intégré avec d'autres outils de sécurité :

- **SIEM** (Splunk, ELK, QRadar)
- **Plateformes de threat intelligence** (MISP, ThreatConnect)
- **Systèmes de ticketing** (JIRA, ServiceNow)
- **Outils de réponse aux incidents** (TheHive, RTIR)

## Automatisation des workflows

Pour automatiser les workflows d'analyse :

1. Créez des scripts utilisant l'API REST

2. Utilisez les webhooks pour déclencher des actions
3. Configurez des règles d'automatisation dans l'interface
4. Intégrez avec des outils d'orchestration (Cortex XSOAR, Shuffle)

Cette section a présenté les principales fonctionnalités de CortexDFIR-Forge et comment les utiliser efficacement. Dans les sections suivantes, nous explorerons des cas d'usage concrets et des scénarios d'utilisation avancés.

## Cas d'usage

Cette section présente des scénarios concrets d'utilisation de CortexDFIR-Forge dans différents contextes de sécurité. Ces cas d'usage illustrent comment tirer pleinement parti des fonctionnalités de l'outil pour répondre à des besoins spécifiques.

### SOC Enterprise

Le Centre Opérationnel de Sécurité (SOC) d'une grande entreprise peut utiliser CortexDFIR-Forge pour améliorer ses capacités de détection et de réponse aux incidents.

#### Analyse automatisée à grande échelle

**Scénario :** Un SOC doit analyser plus de 1000 fichiers par jour provenant de diverses sources.

#### Solution avec CortexDFIR-Forge :

1. **Mise en place d'un pipeline d'ingestion automatisé :** `bash # Script d'automatisation pour l'ingestion de fichiers python src/scripts/batch_ingest.py --input-dir /chemin/vers/dossier --recursive --auto-analyze`
2. **Configuration des règles de détection personnalisées :**
3. Création de règles YARA spécifiques à l'environnement
4. Ajustement des seuils de détection selon le profil de risque
5. **Intégration avec le SIEM existant :**
6. Configuration des webhooks pour envoyer les alertes au SIEM
7. Corrélation des événements entre CortexDFIR-Forge et le SIEM
8. **Automatisation des actions de remédiation :**

9. Isolation automatique des endpoints compromis via l'API Cortex XDR
10. Blocage des IoCs identifiés sur les pare-feux et proxys

## Intégration SIEM

### Étapes d'intégration avec un SIEM :

1. Configurer l'export des alertes au format compatible (JSON, CEF, LEEF)
2. Établir un canal de communication sécurisé (API, syslog, fichiers)
3. Créer des règles de corrélation dans le SIEM
4. Configurer des tableaux de bord unifiés

```
// Exemple de format d'alerte exporté vers un SIEM
{
 "alert_id": "CDF-2025-06-12-001",
 "severity": "high",
 "type": "malware_detection",
 "source": "file_analysis",
 "timestamp": "2025-06-12T14:30:00Z",
 "details": {
 "file_name": "invoice.doc",
 "file_hash": "a1b2c3d4e5f6...",
 "detection_rule": "YARA_EMOTET_DOC",
 "confidence": 95
 },
 "related_iocs": [
 {"type": "hash", "value": "a1b2c3d4e5f6..."},
 {"type": "domain", "value": "malicious-domain.com"}
]
}
```

## Réponse aux incidents en moins de 15 minutes

### Workflow de réponse rapide :

1. **Détection** : Alerte générée par Cortex XDR ou CortexDFIR-Forge
2. **Triage automatique** : Évaluation de la gravité et priorisation
3. **Enrichissement** : Collecte automatique de contexte supplémentaire
4. **Analyse** : Investigation rapide avec les outils intégrés
5. **Containment** : Actions de confinement via l'API Cortex XDR
6. **Documentation** : Génération automatique de rapport d'incident

**Avantages** : - Réduction du temps de réponse de 45 minutes à moins de 15 minutes -  
Standardisation des procédures de réponse - Documentation complète et cohérente des incidents - Réduction de la charge cognitive des analystes

# Investigation forensique

CortexDFIR-Forge excelle dans les investigations forensiques approfondies, permettant aux analystes de reconstituer les incidents et d'identifier les vecteurs d'attaque.

## Analyse de disques compromis

**Scénario :** Un endpoint critique a été compromis et nécessite une analyse forensique approfondie.

### Procédure d'investigation :

1. **Acquisition de l'image disque :**
2. Création d'une image VMDK du disque compromis
3. Transfert sécurisé vers l'environnement d'analyse
4. **Analyse avec CortexDFIR-Forge :** `bash # Lancement de l'analyse forensique python src/main.py --forensic-analysis --input /chemin/vers/image.vmdk --output-dir /chemin/vers/resultats`
5. **Extraction des artefacts clés :**
6. Fichiers système modifiés
7. Logs d'événements
8. Fichiers de registre
9. Artefacts de navigation web
10. Traces de persistance
11. **Reconstruction de la timeline :**
12. Corrélation des horodatages
13. Identification des actions de l'attaquant
14. Visualisation de la chronologie complète
15. **Extraction des indicateurs de compromission :**
16. Hashes de fichiers malveillants
17. Domaines et IPs contactés
18. Clés de registre modifiées
19. Comptes utilisés ou créés

## Chaîne de custody numérique

CortexDFIR-Forge maintient une chaîne de custody numérique complète pour garantir l'intégrité des preuves :

1. **Hachage des preuves :**
2. Calcul et vérification des hashes (MD5, SHA-1, SHA-256)
3. Signature numérique des rapports
4. **Journalisation des accès :**
5. Enregistrement de toutes les actions effectuées
6. Horodatage précis des opérations
7. **Génération de rapports conformes :**
8. Format standardisé pour les procédures légales
9. Documentation exhaustive des méthodes utilisées
10. **Exportation sécurisée :**
11. Chiffrement des données exportées
12. Vérification d'intégrité post-exportation

## Threat Hunting

CortexDFIR-Forge peut être utilisé de manière proactive pour la recherche de menaces (threat hunting) dans votre environnement.

### Recherche proactive de menaces

#### Méthodologie de threat hunting :

1. **Définition des hypothèses :**
2. Identification des techniques d'attaque potentielles
3. Sélection des tactiques MITRE ATT&CK à investiguer
4. **Création de requêtes XQL personnalisées :** `sql -- Exemple de requête XQL pour détecter des connexions suspectes SELECT timestamp, src_ip, dst_ip, dst_port, process_name FROM xdr_data WHERE dst_port IN (4444, 5555, 6666) AND process_name NOT IN`

```
('known_process1.exe', 'known_process2.exe') ORDER BY timestamp
DESC LIMIT 100
```

5. **Exécution des requêtes via CortexDFIR-Forge :** `bash # Exécution d'une  
requête XQL python src/utils/xql_query.py --query-file /chemin/  
vers/requete.xql --output json`

6. **Analyse des résultats :**

7. Visualisation des données dans l'interface  
8. Identification des anomalies et patterns suspects

9. Corrélation avec d'autres sources de données

10. **Itération et affinage :**

11. Ajustement des requêtes en fonction des résultats  
12. Exploration des pistes identifiées  
13. Documentation des découvertes

## **Corrélation multi-sources**

CortexDFIR-Forge permet de corréler des données provenant de multiples sources pour une vision holistique :

1. **Sources de données intégrées :**

2. Logs Cortex XDR  
3. Données de trafic réseau  
4. Logs d'authentification  
5. Journaux d'événements Windows

6. Alertes de sécurité

7. **Techniques de corrélation :**

8. Corrélation temporelle (événements dans une fenêtre de temps)  
9. Corrélation spatiale (événements sur les mêmes systèmes)

10. Corrélation comportementale (séquences d'actions similaires)

11. **Visualisation avancée :**

12. Graphes de relations  
13. Timelines interactives  
14. Cartes thermiques d'activité



## Intelligence artificielle pour la détection

CortexDFIR-Forge utilise des techniques d'IA pour améliorer la détection :

1. **Modèles de détection d'anomalies :**
2. Identification des comportements s'écartant de la norme
3. Détection des activités inhabituelles par utilisateur ou système
4. **Analyse prédictive :**
5. Identification des indicateurs précoces de compromission
6. Prédiction des systèmes à risque
7. **Indicateurs prédictifs :**
8. Scoring de risque dynamique
9. Alertes préventives basées sur les modèles comportementaux

Ces cas d'usage démontrent la polyvalence et la puissance de CortexDFIR-Forge dans différents contextes de sécurité. En adaptant ces approches à votre environnement spécifique, vous pourrez maximiser la valeur de cet outil pour vos opérations de sécurité.

## Administration et maintenance

Cette section couvre les aspects d'administration et de maintenance de CortexDFIR-Forge pour assurer son bon fonctionnement à long terme.

### Scripts d'administration

CortexDFIR-Forge inclut plusieurs scripts d'administration qui facilitent la gestion quotidienne de la plateforme.

### Déploiement avec rollback automatique

Le script `deploy.sh` permet de déployer CortexDFIR-Forge avec une fonctionnalité de rollback automatique en cas de problème :

```
Déploiement avec version spécifique
./deploy.sh production v2.0.0
```

Ce script effectue les opérations suivantes : 1. Sauvegarde de la configuration actuelle 2. Vérification des prérequis 3. Téléchargement de la version spécifiée 4. Arrêt gracieux des services 5. Déploiement des nouveaux conteneurs 6. Vérification du bon fonctionnement 7. Rollback automatique en cas d'échec

## Sauvegarde complète

Le script de sauvegarde permet de créer une copie complète de la configuration et des données :

```
Exécution de la sauvegarde
./scripts/backup.sh
```

Ce script sauvegarde : - La configuration de l'application - Les bases de données - Les règles personnalisées - Les rapports générés - Les logs d'audit

Les sauvegardes sont stockées dans le répertoire `/backups` avec un horodatage, et peuvent être utilisées pour restaurer le système en cas de besoin.

## Mise à jour sécurisée

Le script de mise à jour permet d'appliquer les dernières mises à jour de sécurité :

```
Mise à jour sécurisée
./scripts/update.sh
```

Ce script : 1. Vérifie les nouvelles versions disponibles 2. Télécharge les mises à jour 3. Vérifie l'intégrité des packages 4. Applique les mises à jour 5. Redémarre les services si nécessaire

## Maintenance préventive

Le script de maintenance préventive effectue diverses tâches d'optimisation :

```
Maintenance préventive
./scripts/maintenance.sh
```

Ce script réalise : - Nettoyage des fichiers temporaires - Optimisation des bases de données - Rotation des logs - Vérification de l'intégrité des données - Mise à jour des règles de détection

## Monitoring de santé

Le script de vérification de santé permet de s'assurer que tous les composants fonctionnent correctement :

```
Vérification de l'état du système
python src/utils/health_check.py
```

Ce script vérifie : - La disponibilité de tous les services - L'utilisation des ressources (CPU, mémoire, disque) - La connectivité avec Cortex XDR - Les performances du système - Les erreurs dans les logs

## Monitoring et alerting

CortexDFIR-Forge intègre un système complet de monitoring et d'alerting basé sur Prometheus et Grafana.

### Métriques temps réel

Le système collecte en temps réel diverses métriques de performance :

- **Métriques système** : CPU, mémoire, disque, réseau
- **Métriques applicatives** : temps de réponse, nombre de requêtes, taux d'erreur
- **Métriques métier** : nombre d'analyses, temps d'analyse, taux de détection

Ces métriques sont accessibles via Prometheus à l'adresse `http://localhost:9090`.

### Dashboards Grafana

Des tableaux de bord Grafana préconfigurés sont disponibles pour visualiser les métriques :

1. **Dashboard système** : Vue d'ensemble des ressources système
2. **Dashboard application** : Performances de l'application
3. **Dashboard analyses** : Statistiques sur les analyses effectuées
4. **Dashboard sécurité** : Événements de sécurité et alertes

Accédez à Grafana via `http://localhost:3000` avec les identifiants par défaut (admin/admin).

Exemple de configuration d'un nouveau dashboard :

1. Connectez-vous à Grafana
2. Cliquez sur "Create" > "Dashboard"

3. Ajoutez un nouveau panneau
4. Sélectionnez la source de données Prometheus
5. Configurez la requête, par exemple : `rate(cortexdfir_analyses_total[5m])`
6. Personnalisez la visualisation selon vos besoins
7. Enregistrez le dashboard

## Alerting intelligent

Le système d'alerting permet de définir des règles pour être notifié en cas de problème :

### 1. **Alertes système :**

2. Utilisation CPU > 80% pendant plus de 5 minutes
3. Espace disque disponible < 10%
4. Service indisponible

### 5. **Alertes applicatives :**

6. Temps de réponse API > 2 secondes
7. Taux d'erreur > 5%
8. Échec de connexion à Cortex XDR

### 9. **Canaux de notification :**

10. Email
11. Slack
12. PagerDuty
13. Webhook personnalisé

Configuration d'une alerte dans Grafana :

1. Éditez un panneau dans un dashboard
2. Allez dans l'onglet "Alert"
3. Définissez les conditions d'alerte
4. Configurez les notifications
5. Enregistrez l'alerte

## Logs centralisés

Les logs de tous les composants sont centralisés via Fluentd :

1. **Collection** : Tous les logs sont collectés automatiquement
2. **Indexation** : Les logs sont indexés pour une recherche rapide
3. **Rétention** : Politique de rétention configurable

#### 4. **Analyse** : Possibilité d'analyser les patterns dans les logs

Accédez aux logs via l'interface web ou directement dans le répertoire `/logs`.

## Gestion des utilisateurs

CortexDFIR-Forge inclut un système de gestion des utilisateurs avec contrôle d'accès basé sur les rôles (RBAC).

### Rôles prédéfinis

Le système propose plusieurs rôles prédéfinis :

- **Administrateur** : Accès complet à toutes les fonctionnalités
- **Analyste** : Peut effectuer des analyses et consulter les résultats
- **Auditeur** : Peut uniquement consulter les rapports et les résultats
- **Opérateur** : Peut gérer les tâches d'administration système

### Gestion des utilisateurs via l'interface

Pour gérer les utilisateurs via l'interface web :

1. Connectez-vous en tant qu'administrateur
2. Accédez à la section "Administration" > "Utilisateurs"
3. Utilisez les options pour ajouter, modifier ou supprimer des utilisateurs
4. Assignez les rôles appropriés

### Gestion des utilisateurs via l'API

Vous pouvez également gérer les utilisateurs via l'API :

```
import requests
import json

Configuration
api_url = "http://localhost:8080/api/v1"
api_key = "votre_clé_api_admin"

En-têtes d'authentification
headers = {
 "Authorization": f"Bearer {api_key}",
 "Content-Type": "application/json"
}

Données pour le nouvel utilisateur
new_user = {
 "username": "nouvel_analyste",
```

```
"email": "analyste@entreprise.com",
"role": "analyst",
"password": "MotDePasseTemporaire123!"
}

Envoi de la requête
response = requests.post(
 f"{api_url}/users",
 headers=headers,
 data=json.dumps(new_user)
)

Traitement de la réponse
if response.status_code == 201:
 user_id = response.json()["id"]
 print(f"Utilisateur créé avec l'ID: {user_id}")
else:
 print(f"Erreur: {response.status_code} - {response.text}")
```

## Optimisation des performances

Pour maintenir des performances optimales, suivez ces recommandations :

1. **Dimensionnement des ressources :**
2. Augmentez les ressources allouées en fonction de la charge
3. Surveillez l'utilisation CPU et mémoire pour anticiper les besoins
4. **Configuration de la base de données :**
5. Optimisez les index pour les requêtes fréquentes
6. Configurez la mise en cache appropriée
7. Planifiez des opérations de maintenance régulières
8. **Gestion du stockage :**
9. Implémentez une politique de rétention des données
10. Archivez les anciennes analyses
11. Utilisez un stockage évolutif pour les données volumineuses
12. **Mise en cache :**
13. Configurez Redis pour optimiser la mise en cache
14. Ajustez les paramètres TTL selon les besoins
15. **Équilibrage de charge :**

16. Pour les déploiements à grande échelle, utilisez un équilibreur de charge
17. Distribuez les analyses sur plusieurs nœuds de traitement

En suivant ces bonnes pratiques d'administration et de maintenance, vous assurerez le bon fonctionnement et la pérennité de votre déploiement CortexDFIR-Forge.

## Personnalisation avancée

CortexDFIR-Forge est conçu pour être hautement personnalisable afin de s'adapter aux besoins spécifiques de chaque organisation. Cette section détaille les différentes possibilités de personnalisation avancée.

### Ajout de règles de détection

Les règles de détection sont au cœur des capacités d'analyse de CortexDFIR-Forge. Vous pouvez étendre ces capacités en ajoutant vos propres règles.

#### Règles YARA personnalisées

Les règles YARA permettent de détecter des patterns spécifiques dans les fichiers analysés.

##### 1. Structure des règles YARA :

```
rule Custom_Malware_Detection {
 meta:
 author = "Votre Nom"
 description = "Détection de malware personnalisé"
 severity = "high"
 date = "2025-06-12"

 strings:
 $string1 = "malicious_function_name" ascii wide
 $string2 = { 4D 5A 90 00 03 00 00 00 } // Signature
 hexadécimale
 $regex1 = /password=[^&]*/

 condition:
 uint16(0) == 0x5A4D and // En-tête PE
 filesize < 1MB and
 ($string1 or $string2 or $regex1)
}
```

##### 1. Emplacement des règles personnalisées :

Les règles YARA personnalisées doivent être placées dans le répertoire `rules/custom/` :

```
Création d'une nouvelle règle YARA
nano rules/custom/ma_regle_personnalisee.yar
```

### 1. Test des règles :

Avant de déployer une nouvelle règle, testez-la sur des échantillons connus :

```
Test d'une règle YARA
python src/utils/yara_test.py --rule rules/custom/
ma_regle_personnalisee.yar --file samples/test.exe
```

### 1. Activation des règles :

Pour activer vos règles personnalisées, ajoutez-les à la configuration :

```
Dans config/yara_config.yaml
custom_rules:
 - rules/custom/ma_regle_personnalisee.yar
 - rules/custom/autre_regle.yar
```

## Signature-Base personnalisée

CortexDFIR-Forge vous permet de créer votre propre base de signatures pour la détection de menaces spécifiques à votre environnement.

### 1. Structure de la base de signatures :

```
{
 "signatures": [
 {
 "id": "CUSTOM-SIG-001",
 "name": "Détection de RAT personnalisé",
 "description": "Détection d'un RAT spécifique à notre
environnement",
 "severity": "critical",
 "category": "malware",
 "indicators": [
 {"type": "file_hash", "value": "a1b2c3d4e5f6..."},
 {"type": "registry_key", "value": "HKLM\\SOFTWARE\\
\\Malware\\\\Config"},
 {"type": "process_name", "value": "malicious.exe"}
]
 }
]
}
```



```
 "mitre_techniques": ["T1059", "T1547"]
 }
]
}
```

### 1. Ajout à la base de signatures :

```
Ajout d'une signature personnalisée
python src/utils/add_signature.py --input signatures/
custom_sig.json
```

## Règles personnalisées pour votre environnement

Pour créer des règles adaptées à votre environnement spécifique :

### 1. Analyse de votre environnement :

2. Identifiez les applications légitimes spécifiques
3. Documentez les comportements normaux
4. Répertoriez les configurations standard

### 5. Création de règles de whitelisting :

6. Exemptez les applications légitimes des détections
7. Créez des exceptions pour les comportements normaux

### 8. Création de règles de détection ciblées :

9. Ciblez les menaces spécifiques à votre secteur
10. Adaptez les seuils de détection à votre profil de risque

## Intégration avec d'autres outils

CortexDFIR-Forge peut être intégré avec divers outils tiers pour étendre ses fonctionnalités.

### Intégration avec des SIEM

Pour intégrer CortexDFIR-Forge avec votre SIEM :

### 1. Configuration de l'export des alertes :

```
Dans config/integrations/siem_config.yaml
siem:
 type: "splunk" # ou "elastic", "qradar", etc.
```

```
endpoint: "https://splunk.example.com:8088/services/collector"
token: "votre-token-hec"
format: "json"
batch_size: 100
fields:
 source: "cortexdfir-forge"
 sourcetype: "security_alert"
 index: "security"
```

## 1. Personnalisation du format des alertes :

Créez un template pour le format des alertes :

```
{# Dans templates/siem/splunk_alert.j2 #}
{
 "time": {{ alert.timestamp }},
 "host": "{{ hostname }}",
 "source": "CortexDFIR-Forge",
 "sourcetype": "security_alert",
 "index": "security",
 "event": {
 "alert_id": "{{ alert.id }}",
 "severity": "{{ alert.severity }}",
 "title": "{{ alert.title }}",
 "description": "{{ alert.description }}",
 "mitre_tactics": {{ alert.mitre_tactics | tojson }},
 "indicators": {{ alert.indicators | tojson }}
 }
}
```

## 1. Test de l'intégration :

```
Test de l'intégration SIEM
python src/utils/test_siem_integration.py --alert-id TEST-001
```

## Intégration avec des plateformes de threat intelligence

Pour enrichir vos analyses avec des données de threat intelligence :

## 1. Configuration des sources de threat intelligence :

```
Dans config/integrations/ti_config.yaml
threat_intelligence:
 sources:
 - name: "MISP"
 type: "misp"
 url: "https://misp.example.com"
```

```

 api_key: "votre-clé-api-misp"
 verify_ssl: true

- name: "AlienVault OTX"
 type: "otx"
 api_key: "votre-clé-api-otx"

- name: "VirusTotal"
 type: "virustotal"
 api_key: "votre-clé-api-vt"

```

## 1. Utilisation dans les analyses :

```

Dans un script personnalisé
from cortexdfir.integrations import threat_intelligence

Vérification d'un indicateur
result = threat_intelligence.check_indicator("a1b2c3d4e5f6...",
"file_hash")
if result.malicious:
 print(f"Indicateur malveillant détecté : {result.details}")

```

## Développement de modules personnalisés

CortexDFIR-Forge permet de développer des modules personnalisés pour étendre ses fonctionnalités.

### 1. Structure d'un module :

```

Dans src/modules/custom/my_module.py
from cortexdfir.core import Module, AnalysisResult

class CustomAnalyzer(Module):
 """Module d'analyse personnalisé"""

 def __init__(self, config=None):
 super().__init__(name="custom_analyzer", config=config)

 def analyze(self, artifact, context=None):
 """Analyse un artefact"""
 result = AnalysisResult(module=self.name)

 # Logique d'analyse personnalisée
 if self._detect_suspicious_pattern(artifact):
 result.add_finding(
 title="Pattern suspect détecté",
 description="Un pattern suspect a été
identifié",

```

```

 severity="medium",
 confidence=80
)

 return result

def _detect_suspicious_pattern(self, artifact):
 """Détection d'un pattern suspect dans l'artefact"""
 # Implémentation de la détection
 return False

```

### 1. Enregistrement du module :

```

Dans src/modules/custom/__init__.py
from cortexdfir.core import register_module
from .my_module import CustomAnalyzer

Enregistrement du module
register_module(CustomAnalyzer)

```

### 1. Configuration du module :

```

Dans config/modules/custom_analyzer.yaml
enabled: true
config:
 detection_threshold: 0.8
 max_file_size: 10485760 # 10 MB

```

## Personnalisation de l'interface utilisateur

L'interface utilisateur de CortexDFIR-Forge peut être personnalisée pour s'adapter à vos besoins.

### Personnalisation du thème

#### 1. Modification des couleurs et du logo :

```

/* Dans static/css/custom.css */
:root {
 --primary-color: #1976D2;
 --secondary-color: #424242;
 --accent-color: #FF4081;
 --background-color: #F5F5F5;
 --text-color: #212121;
}

```

```
.navbar-brand img {
 content: url('/static/img/custom-logo.png');
}
```

## 1. Activation du thème personnalisé :

```
Dans config/ui_config.yaml
theme:
 use_custom_theme: true
 custom_css_path: "static/css/custom.css"
 custom_logo_path: "static/img/custom-logo.png"
 favicon_path: "static/img/custom-favicon.ico"
```

## Personnalisation des tableaux de bord

### 1. Création d'un tableau de bord personnalisé :

```
// Dans config/dashboards/custom_dashboard.json
{
 "id": "custom_dashboard",
 "name": "Tableau de bord personnalisé",
 "description": "Tableau de bord adapté à nos besoins
spécifiques",
 "layout": "grid",
 "widgets": [
 {
 "id": "recent_alerts",
 "type": "alert_list",
 "title": "Alertes récentes",
 "position": {"x": 0, "y": 0, "w": 6, "h": 4},
 "config": {
 "max_items": 10,
 "filter": {"severity": ["high", "critical"]}
 }
 },
 {
 "id": "threat_map",
 "type": "geo_map",
 "title": "Carte des menaces",
 "position": {"x": 6, "y": 0, "w": 6, "h": 4},
 "config": {
 "data_source": "alerts",
 "geo_field": "source_ip_geo"
 }
 }
]
}
```

```
]
}
```

### 1. Activation du tableau de bord :

```
Dans config/ui_config.yaml
dashboards:
 default_dashboard: "custom_dashboard"
 available_dashboards:
 - "custom_dashboard"
 - "system_dashboard"
 - "security_dashboard"
```

## Automatisation avancée

CortexDFIR-Forge permet de créer des workflows d'automatisation avancés pour optimiser vos processus de sécurité.

### Création de playbooks

Les playbooks permettent d'automatiser des séquences d'actions en réponse à des événements spécifiques.

### 1. Structure d'un playbook :

```
Dans config/playbooks/ransomware_response.yaml
id: ransomware_response
name: "Réponse automatisée aux ransomwares"
description: "Playbook de réponse automatisée aux incidents de ransomware"
trigger:
 type: "alert"
 conditions:
 - field: "category"
 operator: "equals"
 value: "ransomware"
 - field: "severity"
 operator: "in"
 value: ["high", "critical"]

actions:
 - id: "isolate_endpoint"
 type: "cortex_xdr_action"
 parameters:
 action: "isolate"
 target_type: "endpoint"
 target_field: "hostname"
```

- **id**: "collect\_forensics"  
**type**: "forensic\_collection"  
**parameters**:
  - collection\_type**: "full"
  - target\_field**: "hostname"
- **id**: "notify\_team"  
**type**: "notification"  
**parameters**:
  - channel**: "slack"
  - template**: "ransomware\_alert"

### 1. Activation du playbook :

```
Activation d'un playbook
python src/utils/manage_playbooks.py --enable config/playbooks/
ransomware_response.yaml
```

## Intégration avec des outils d'orchestration

Pour des workflows plus complexes, CortexDFIR-Forge peut s'intégrer avec des outils d'orchestration comme Cortex XSOAR ou Shuffle.

### 1. Configuration de l'intégration XSOAR :

```
Dans config/integrations/xsoar_config.yaml
xsoar:
 url: "https://xsoar.example.com"
 api_key: "votre-clé-api-xsoar"
 verify_ssl: true
 incident_type: "CortexDFIR"
 mappings:
 alert_id: "incident.alertid"
 title: "incident.name"
 severity: "incident.severity"
 description: "incident.details"
```

### 1. Test de l'intégration :

```
Test de l'intégration XSOAR
python src/utils/test_xsoar_integration.py --alert-id TEST-002
```

En explorant ces options de personnalisation avancée, vous pourrez adapter CortexDFIR-Forge à vos besoins spécifiques et l'intégrer parfaitement dans votre écosystème de sécurité existant.

## Dépannage et FAQ

Cette section aborde les problèmes courants que vous pourriez rencontrer lors de l'utilisation de CortexDFIR-Forge et propose des solutions pour les résoudre.

### Problèmes d'installation

#### L'installation échoue avec des erreurs de dépendances

**Problème :** Lors de l'installation, vous rencontrez des erreurs liées aux dépendances Python.

**Solution :** 1. Vérifiez que vous utilisez Python 3.8 ou supérieur : `bash python --version`

1. Mettez à jour pip : `bash pip install --upgrade pip`

2. Installez les dépendances système requises : ``bash # Sur Ubuntu/Debian sudo apt-get update sudo apt-get install -y build-essential libssl-dev libffi-dev python3-dev

# Sur CentOS/RHEL sudo yum install -y gcc openssl-devel libffi-devel python3-devel ``

1. Essayez d'installer les dépendances une par une : `bash pip install -r requirements.txt --no-cache-dir`

#### Erreurs Docker lors du déploiement

**Problème :** Vous rencontrez des erreurs lors du déploiement avec Docker.

**Solution :** 1. Vérifiez que Docker est correctement installé et en cours d'exécution : `bash docker --version docker info`

1. Assurez-vous que l'utilisateur actuel fait partie du groupe Docker : `bash sudo usermod -aG docker $USER # Déconnectez-vous et reconnectez-vous pour appliquer les changements`

2. Vérifiez l'espace disque disponible : `bash df -h`

3. Nettoyez les ressources Docker inutilisées : `bash docker system prune -a`



4. Vérifiez les logs Docker pour plus de détails : `bash docker logs <container_id>`

## Problèmes de permissions

**Problème** : Vous rencontrez des erreurs de permission lors de l'accès aux fichiers ou répertoires.

**Solution** : 1. Vérifiez les permissions des répertoires : `bash ls -la /chemin/vers/CortexDFIR-Forge`

1. Ajustez les permissions si nécessaire : `bash sudo chown -R $USER:$USER /chemin/vers/CortexDFIR-Forge`  
`chmod -R 755 /chemin/vers/CortexDFIR-Forge`
2. Pour les conteneurs Docker, vérifiez les volumes et leurs permissions : `bash docker volume inspect cortexdfir_data`

## Problèmes de connexion à Cortex XDR

### Erreur d'authentification API

**Problème** : Vous recevez des erreurs d'authentification lors de la connexion à l'API Cortex XDR.

**Solution** : 1. Vérifiez que les clés API sont correctes dans le fichier `.env` : `bash cat .env | grep CORTEX_XDR`

1. Assurez-vous que la clé API n'a pas expiré dans la console Cortex XDR.
2. Vérifiez que la clé API dispose des permissions nécessaires.
3. Testez la connexion avec l'utilitaire de test : `bash python -m src.utils.test_cortex_connection`
4. Vérifiez que l'URL de base correspond à votre région : `# Pour l'Europe`  
`CORTEX_XDR_BASE_URL=https://api-eu.xdr.paloaltonetworks.com`

### Erreurs SSL/TLS

**Problème** : Vous rencontrez des erreurs SSL lors de la connexion à l'API Cortex XDR.

**Solution** : 1. Vérifiez que votre système dispose des certificats CA à jour : ```bash # Sur Ubuntu/Debian sudo apt-get install -y ca-certificates`

# Sur CentOS/RHEL sudo yum install -y ca-certificates ``

1. Si nécessaire, désactivez temporairement la vérification des certificats pour le débogage (à ne pas utiliser en production) : # Dans `.env`  
`VERIFY_CERTIFICATES=false`
2. Vérifiez que la date et l'heure de votre système sont correctes : `bash date`

## Problèmes de performance

### Analyses lentes

**Problème** : Les analyses de fichiers sont anormalement lentes.

**Solution** : 1. Vérifiez l'utilisation des ressources système : `bash top htop` # Si installé

1. Augmentez les ressources allouées aux conteneurs Docker : `bash #` Dans `docker-compose.yml`, ajustez les limites de ressources services:  
`app: deploy: resources: limits: cpus: '4' memory: 8G`
2. Optimisez la configuration de la base de données : `bash #` Éditez `config/database.yml` pour ajuster les paramètres
3. Vérifiez les logs pour identifier les goulots d'étranglement : `bash docker-compose logs -f app`

### Problèmes de mémoire

**Problème** : L'application consomme trop de mémoire ou se bloque avec des erreurs OOM (Out Of Memory).

**Solution** : 1. Augmentez la mémoire allouée : `bash #` Dans `docker-compose.yml`  
`services: app: deploy: resources: limits: memory: 16G`

1. Optimisez les paramètres JVM pour Redis et autres services Java : `bash #` Dans `docker-compose.yml` services: redis: environment: `JAVA_OPTS: "-Xms512m -Xmx2g"`
2. Implémentez le traitement par lots pour les analyses volumineuses : `bash #`  
Utilisez l'option de traitement par lots `python src/main.py --batch-size 1000 --input large_dataset.json`

## Problèmes courants d'utilisation

### Erreurs lors de l'analyse de fichiers

**Problème** : Vous rencontrez des erreurs lors de l'analyse de certains fichiers.

**Solution** : 1. Vérifiez que le format du fichier est supporté :

```
bash file /chemin/vers/fichier
```

1. Vérifiez la taille du fichier (certaines limites peuvent s'appliquer) : 

```
bash ls -lh /chemin/vers/fichier
```
2. Augmentez les limites de taille de fichier si nécessaire : 

```
yaml # Dans config/analysis_config.yaml file_limits: max_size: 10737418240 # 10 GB
```
3. Vérifiez les logs pour des erreurs spécifiques : 

```
bash tail -f logs/analysis.log
```

### Problèmes d'affichage dans l'interface web

**Problème** : L'interface web ne s'affiche pas correctement ou certaines fonctionnalités ne fonctionnent pas.

**Solution** : 1. Videz le cache de votre navigateur.

1. Vérifiez que vous utilisez un navigateur moderne et à jour.
2. Vérifiez les erreurs dans la console JavaScript du navigateur.
3. Redémarrez le service frontend : 

```
bash docker-compose restart frontend
```

## FAQ

### Questions générales

**Q: Quelle est la différence entre CortexDFIR-Forge et Cortex XDR ?**

R: Cortex XDR est la plateforme de détection et de réponse étendue (XDR) de Palo Alto Networks. CortexDFIR-Forge est un framework open-source qui s'intègre à Cortex XDR pour étendre ses capacités d'investigation numérique et de réponse aux incidents (DFIR). CortexDFIR-Forge ajoute des fonctionnalités avancées d'analyse forensique, de corrélation multi-sources et de reporting qui complètent les capacités natives de Cortex XDR.

**Q: CortexDFIR-Forge est-il compatible avec d'autres solutions XDR ?**

R: CortexDFIR-Forge est principalement conçu pour s'intégrer avec Cortex XDR. Bien que certaines fonctionnalités puissent fonctionner avec d'autres solutions XDR, l'intégration native est optimisée pour Cortex XDR. Des adaptateurs pour d'autres plateformes peuvent être développés en utilisant le framework d'extension.

**Q: Quelle est la fréquence des mises à jour de CortexDFIR-Forge ?**

R: CortexDFIR-Forge suit un cycle de mise à jour trimestriel pour les versions majeures, avec des mises à jour de sécurité publiées dès que nécessaire. La roadmap 2025 prévoit des mises à jour majeures en mars, juin, septembre et décembre.

**Questions techniques**

**Q: Comment puis-je augmenter le nombre maximum d'utilisateurs simultanés ?**

R: Le nombre d'utilisateurs simultanés est limité par les ressources système et la configuration. Pour augmenter cette limite :

1. Augmentez les ressources allouées aux conteneurs.
2. Ajustez les paramètres de connexion dans `config/app_config.yaml` :  
`yaml`  
`web_server: max_connections: 100 connection_timeout: 120`
3. Configurez un équilibreur de charge pour une architecture distribuée.

**Q: Comment puis-je sauvegarder ma configuration et mes données ?**

R: Pour sauvegarder votre installation :

1. Utilisez le script de sauvegarde intégré : `bash ./scripts/backup.sh`
2. Pour une sauvegarde manuelle, exportez :
3. La base de données : `docker exec -it cortexdfir_db pg_dump -U postgres cortexdfir > backup.sql`
4. Les fichiers de configuration : `tar -czf config_backup.tar.gz config/`
5. Les règles personnalisées : `tar -czf rules_backup.tar.gz rules/custom/`

**Q: Comment puis-je migrer vers une nouvelle version ?**

R: Pour migrer vers une nouvelle version :

1. Sauvegardez votre configuration actuelle.
2. Consultez les notes de version pour les changements importants.
3. Utilisez le script de mise à jour : `bash git pull ./deploy.sh update`
4. Vérifiez la migration de la base de données : `bash python src/utils/check_db_migration.py`

## Questions sur l'intégration

### Q: Comment puis-je intégrer CortexDFIR-Forge avec mon système de tickets ?

R: CortexDFIR-Forge peut s'intégrer avec divers systèmes de tickets :

1. Pour JIRA: 

```
yaml # Dans config/integrations/ticketing.yaml jira:
 url: "https://jira.example.com" username: "jira_user" api_token:
 "jira_token" project_key: "SEC" issue_type: "Incident" mappings:
 title: "summary" description: "description" severity:
 "customfield_10001"
```
2. Pour ServiceNow: 

```
yaml servicenow: url: "https://example.service-
now.com" username: "servicenow_user" password:
 "servicenow_password" table: "incident" mappings: title:
 "short_description" description: "description" severity:
 "impact"
```

### Q: Comment puis-je exporter les résultats d'analyse vers un format spécifique ?

R: CortexDFIR-Forge prend en charge plusieurs formats d'export :

1. Via l'interface web : Sélectionnez une analyse et cliquez sur "Exporter" en choisissant le format souhaité.
2. Via l'API: 

```
bash curl -X GET "http://localhost:8080/api/v1/analyses/
123/export?format=stix" \ -H "Authorization: Bearer votre_token"
\ -o export.json
```
3. Via la ligne de commande :  

```
bash python src/utils/export_analysis.py --id 123 --format pdf
--output analysis_report.pdf
```

Ces informations de dépannage et cette FAQ devraient vous aider à résoudre la plupart des problèmes courants et à répondre aux questions fréquentes concernant CortexDFIR-Forge.

## Sécurité et conformité

La sécurité et la conformité sont des aspects essentiels de tout outil d'investigation numérique et de réponse aux incidents. Cette section détaille les mesures de sécurité intégrées à CortexDFIR-Forge et les considérations de conformité à prendre en compte.

# Sécurisation de l'installation

## Authentification et autorisation

CortexDFIR-Forge implémente un système robuste d'authentification et d'autorisation :

1. **Authentification multi-facteurs (MFA) :**
2. Support de l'authentification à deux facteurs (2FA)
3. Intégration avec des fournisseurs TOTP (Time-based One-Time Password)
4. Support des clés de sécurité physiques (YubiKey, etc.)

Configuration de la MFA: `yaml # Dans config/security/auth_config.yaml`  
`authentication: mfa: enabled: true methods: - totp - webauthn`  
`grace_period: 86400 # 24 heures`

1. **Contrôle d'accès basé sur les rôles (RBAC) :**
2. Définition granulaire des permissions
3. Séparation des privilèges
4. Principe du moindre privilège

Exemple de configuration RBAC: `yaml # Dans config/security/`  
`rbac_config.yaml roles: analyst: permissions: - "analyses:read" -`  
`"analyses:create" - "reports:read" - "reports:create" admin:`  
`permissions: - "*" # Toutes les permissions`

1. **Intégration avec les fournisseurs d'identité :**
2. Support de LDAP/Active Directory
3. Support de SAML 2.0
4. Support d'OAuth 2.0/OpenID Connect

Configuration de l'intégration LDAP: `yaml # Dans config/security/`  
`auth_config.yaml ldap: enabled: true server: "ldap://`  
`ldap.example.com:389" bind_dn: "cn=service,dc=example,dc=com"`  
`bind_password: "${LDAP_PASSWORD}" search_base:`  
`"ou=users,dc=example,dc=com" search_filter:`  
`"(sAMAccountName={username})" group_search_base:`  
`"ou=groups,dc=example,dc=com" group_search_filter:`  
`"(&(objectClass=group)(member={dn}))"`

## Chiffrement des données

CortexDFIR-Forge assure la protection des données sensibles :

1. **Chiffrement en transit :**
2. Configuration TLS/SSL pour toutes les communications
3. Support des versions TLS 1.2 et 1.3
4. Rotation régulière des certificats

Configuration HTTPS: `yaml # Dans config/security/tls_config.yaml`  
`tls:`  
`enabled: true cert_file: "/path/to/cert.pem" key_file: "/path/to/`  
`key.pem" min_version: "TLS1.2" ciphers: -`  
`"TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384" -`  
`"TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"`

1. **Chiffrement au repos :**
2. Chiffrement de la base de données
3. Chiffrement des fichiers sensibles
4. Gestion sécurisée des clés

Configuration du chiffrement au repos: `yaml # Dans config/security/`  
`encryption_config.yaml`  
`data_encryption: enabled: true algorithm:`  
`"AES-256-GCM" key_rotation_period: 90 # jours sensitive_fields: -`  
`"api_keys" - "credentials" - "personal_data"`

1. **Gestion des secrets :**
2. Stockage sécurisé des secrets (API keys, mots de passe)
3. Intégration avec des coffres-forts comme HashiCorp Vault ou AWS Secrets Manager
4. Rotation automatique des secrets

Configuration de l'intégration avec Vault: `yaml # Dans config/security/`  
`secrets_config.yaml`  
`secrets_manager: type: "vault" url: "https://`  
`vault.example.com:8200" token: "${VAULT_TOKEN}" path: "secret/`  
`cortexdfir/" auto_renewal: true`

## Durcissement de l'infrastructure

Pour sécuriser l'infrastructure hébergeant CortexDFIR-Forge :

1. **Durcissement des conteneurs Docker :**
2. Utilisation d'images minimales et sécurisées
3. Exécution en tant qu'utilisateur non-root
4. Limitation des capacités

Exemple de configuration Docker sécurisée : `` `dockerfile # Dans Dockerfile.secure  
FROM alpine:3.15

# Création d'un utilisateur non-root RUN addgroup -S appgroup && adduser -S appuser -  
G appgroup

# Installation des dépendances RUN apk add --no-cache python3 py3-pip

# Configuration de l'application COPY --chown=appuser:appgroup . /app WORKDIR /app

# Exécution en tant qu'utilisateur non-root USER appuser

# Commande de démarrage CMD ["python3", "src/main.py"] `` `

#### 1. **Isolation réseau :**

2. Segmentation du réseau
3. Utilisation de réseaux Docker isolés
4. Configuration de pare-feu restrictive

Configuration Docker Compose pour l'isolation réseau : `` `yaml # Dans docker-  
compose.secure.yml networks: frontend: internal: false backend: internal: true

services: web: networks: - frontend app: networks: - frontend - backend db: networks: -  
backend `` `

#### 1. **Scanning de vulnérabilités :**

2. Analyse régulière des conteneurs
3. Vérification des dépendances
4. Mise à jour proactive des composants

Intégration de Trivy pour le scanning de conteneurs : `` `bash # Script de scanning  
automatique #!/bin/bash

echo "Scanning des images Docker..." trivy image cortexdfir/app:latest trivy image  
cortexdfir/web:latest trivy image cortexdfir/db:latest `` `

## **Journalisation et audit**

CortexDFIR-Forge implémente des mécanismes complets de journalisation et d'audit  
pour assurer la traçabilité des actions.

### **Journalisation sécurisée**

#### 1. **Journalisation complète :**

2. Journalisation de toutes les actions des utilisateurs



3. Journalisation des accès système
4. Journalisation des événements de sécurité

Configuration de la journalisation: `yaml # Dans config/logging_config.yaml`  
`logging: level: "INFO" format: "json" output: - type: "file" path: "/var/log/cortexdfir/app.log" rotation: max_size: "100MB" max_files: 10 - type: "syslog" facility: "local0" server: "syslog.example.com: 514"`

1. **Protection des logs :**
2. Signature des logs pour garantir l'intégrité
3. Chiffrement des logs sensibles
4. Stockage sécurisé et immuable

Configuration de la protection des logs: `yaml # Dans config/security/log_protection.yaml`  
`log_protection: integrity: enabled: true method: "hmac-sha256" encryption: enabled: true sensitive_fields: - "password" - "token" - "api_key" immutable_storage: enabled: true type: "worm_storage" # Write Once Read Many`

## Piste d'audit

1. **Événements audités :**
2. Connexions et déconnexions
3. Modifications de configuration
4. Accès aux données sensibles
5. Actions administratives
6. Analyses et investigations

7. **Format d'audit standardisé :**

8. Horodatage précis
9. Identité de l'utilisateur
10. Action effectuée
11. Ressource concernée
12. Résultat de l'action

Exemple d'entrée d'audit: `json { "timestamp": "2025-06-12T15:42:31.123Z", "event_type": "user_action", "user": { "id": "john.doe", "ip": "192.168.1.100", "session_id": "sess_123456" }, "action": "analysis_start", "resource": { "type": "file", "id": "file_789012",`

```
"name": "suspicious.exe" }, "result": "success", "details":
{ "analysis_id": "analysis_345678", "analysis_type": "full" } }
```

1. **Alertes sur événements suspects :**
2. Détection des comportements anormaux
3. Alertes sur les violations de politique
4. Notification des tentatives d'accès non autorisées

## Conformité réglementaire

CortexDFIR-Forge est conçu pour aider les organisations à respecter diverses exigences réglementaires.

### RGPD (Règlement Général sur la Protection des Données)

1. **Traitement des données personnelles :**
2. Minimisation des données collectées
3. Pseudonymisation automatique
4. Chiffrement des données sensibles
5. **Droits des personnes concernées :**
6. Fonctionnalités d'export de données
7. Capacité d'anonymisation
8. Suppression sécurisée des données
9. **Documentation de conformité :**
10. Registre des traitements
11. Évaluation d'impact (PIA)
12. Procédures de notification de violation

### ISO 27001

CortexDFIR-Forge implémente des contrôles alignés avec la norme ISO 27001 :

1. **Gestion des actifs :**
2. Inventaire des actifs informationnels
3. Classification des données
4. Gestion du cycle de vie des informations
5. **Contrôle d'accès :**

6. Politique d'accès stricte
7. Révision périodique des accès
8. Gestion des privilèges

**9. Sécurité opérationnelle :**

10. Procédures documentées
11. Gestion des changements
12. Protection contre les logiciels malveillants

## **Autres cadres de conformité**

CortexDFIR-Forge peut être configuré pour répondre aux exigences d'autres cadres réglementaires :

- **PCI DSS** pour le traitement des données de cartes de paiement
- **HIPAA** pour les données de santé
- **SOC 2** pour la sécurité, disponibilité et confidentialité
- **NIS2** pour la sécurité des réseaux et des systèmes d'information

## **Bonnes pratiques de sécurité**

### **Sécurisation du déploiement**

1. **Environnement dédié :**
2. Déploiement sur une infrastructure dédiée
3. Isolation physique ou logique
4. Accès réseau restreint
5. **Principe de défense en profondeur :**
6. Multiples couches de sécurité
7. Segmentation réseau
8. Contrôles de sécurité complémentaires
9. **Mises à jour régulières :**
10. Application rapide des correctifs de sécurité
11. Suivi des bulletins de sécurité
12. Plan de gestion des vulnérabilités

## Gestion des incidents

1. **Plan de réponse aux incidents :**
2. Procédures documentées
3. Équipe de réponse désignée
4. Exercices réguliers
5. **Détection des compromissions :**
6. Surveillance des comportements anormaux
7. Détection des indicateurs de compromission
8. Alertes en temps réel
9. **Procédures de récupération :**
10. Sauvegardes régulières
11. Tests de restauration
12. Plan de continuité d'activité

## Formation et sensibilisation

1. **Formation des utilisateurs :**
2. Formation initiale obligatoire
3. Mises à jour régulières
4. Exercices pratiques
5. **Documentation de sécurité :**
6. Politiques et procédures claires
7. Guides d'utilisation sécurisée
8. Référentiels de bonnes pratiques

En suivant ces recommandations de sécurité et de conformité, vous assurerez que votre déploiement de CortexDFIR-Forge respecte les normes les plus strictes en matière de protection des données et de sécurité informatique.

## Conclusion

CortexDFIR-Forge représente une avancée significative dans le domaine de l'investigation numérique et de la réponse aux incidents. En s'intégrant nativement avec Cortex XDR, cette solution open-source offre aux équipes de sécurité des capacités

d'analyse forensique avancées, une automatisation intelligente et des fonctionnalités de reporting professionnel.

## Synthèse des points clés

Au cours de ce guide, nous avons exploré en détail :

1. **L'architecture robuste** de CortexDFIR-Forge, conçue pour s'adapter à différents environnements et échelles de déploiement.
2. **Les multiples options d'installation**, permettant de choisir la méthode la plus adaptée à votre contexte : déploiement automatisé, installation pour développement ou conteneurisation Docker.
3. **L'intégration native avec Cortex XDR**, offrant une synergie puissante entre la détection des menaces et l'investigation approfondie.
4. **Les fonctionnalités avancées d'analyse**, incluant le support multi-format, les règles de détection personnalisables et les capacités de machine learning.
5. **Les cas d'usage concrets**, démontrant la valeur ajoutée de CortexDFIR-Forge dans différents contextes : SOC Enterprise, investigation forensique et threat hunting.
6. **Les aspects d'administration et de maintenance**, essentiels pour assurer la pérennité et l'efficacité de la solution.
7. **Les possibilités de personnalisation avancée**, permettant d'adapter l'outil à vos besoins spécifiques et de l'intégrer dans votre écosystème de sécurité.
8. **Les considérations de sécurité et de conformité**, garantissant que votre déploiement respecte les normes et réglementations en vigueur.

## Perspectives d'évolution

CortexDFIR-Forge continue d'évoluer pour répondre aux défis croissants de la cybersécurité. Les développements futurs incluent :

- **Intégration de l'intelligence artificielle avancée** pour améliorer la détection des menaces et réduire les faux positifs.
- **Support étendu pour les environnements cloud natifs**, avec des capacités d'analyse spécifiques pour AWS, Azure et GCP.
- **Amélioration des capacités de visualisation** pour faciliter la compréhension des incidents complexes.

- **Développement de modules spécialisés** pour des secteurs spécifiques (finance, santé, industrie, etc.).
- **Renforcement de l'écosystème d'intégrations** avec d'autres outils de sécurité.

## Mot de la fin

L'implémentation réussie de CortexDFIR-Forge dans votre environnement représente une étape importante vers une posture de sécurité plus mature et réactive. En combinant les capacités de détection de Cortex XDR avec les fonctionnalités d'investigation avancées de CortexDFIR-Forge, votre organisation sera mieux équipée pour faire face aux menaces cybernétiques actuelles et futures.

N'oubliez pas que la sécurité est un processus continu, nécessitant une veille constante, des mises à jour régulières et une adaptation permanente aux nouvelles menaces. CortexDFIR-Forge vous accompagne dans cette démarche en fournissant des outils puissants et flexibles pour renforcer vos capacités de détection et de réponse aux incidents.

## Glossaire

**API (Application Programming Interface)** : Ensemble de définitions et de protocoles qui permettent à différents logiciels de communiquer entre eux.

**APT (Advanced Persistent Threat)** : Attaque ciblée et sophistiquée, généralement menée par des acteurs disposant de ressources importantes, visant à maintenir une présence non détectée dans les systèmes d'une organisation sur une longue période.

**DFIR (Digital Forensics and Incident Response)** : Ensemble des processus et techniques utilisés pour identifier, collecter, préserver et analyser les preuves numériques lors d'incidents de sécurité.

**Docker** : Plateforme de conteneurisation permettant de développer, déployer et exécuter des applications dans des environnements isolés appelés conteneurs.

**Endpoint** : Dispositif connecté à un réseau, comme un ordinateur, un smartphone ou un serveur.

**FQDN (Fully Qualified Domain Name)** : Nom de domaine complet qui spécifie la position exacte d'un hôte dans la hiérarchie DNS.

**IoC (Indicator of Compromise)** : Artefact observé sur un réseau ou un système qui indique avec une forte probabilité une intrusion ou une compromission.

**ISO 27001** : Norme internationale définissant les exigences pour un système de management de la sécurité de l'information (SMSI).

**Kubernetes** : Système open-source d'orchestration de conteneurs permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées.

**MITRE ATT&CK** : Base de connaissances accessible au public documentant les tactiques, techniques et procédures (TTP) utilisées par les attaquants.

**RBAC (Role-Based Access Control)** : Méthode de régulation de l'accès aux ressources informatiques basée sur les rôles des utilisateurs au sein d'une organisation.

**RGPD (Règlement Général sur la Protection des Données)** : Règlement de l'Union européenne sur la protection des données personnelles.

**SOC (Security Operations Center)** : Centre opérationnel de sécurité, équipe chargée de surveiller et d'analyser en permanence la posture de sécurité d'une organisation.

**STIX/TAXII** : Standards pour l'échange automatisé d'informations sur les cybermenaces.

**TLS (Transport Layer Security)** : Protocole cryptographique conçu pour sécuriser les communications sur un réseau informatique.

**XDR (Extended Detection and Response)** : Solution de sécurité qui collecte et corrèle automatiquement les données provenant de plusieurs couches de sécurité.

**YARA** : Outil permettant aux chercheurs en malware d'identifier et de classer les échantillons de malware en créant des règles de détection basées sur des patterns textuels ou binaires.

## Ressources et liens utiles

### Documentation officielle

- [Documentation CortexDFIR-Forge](#)
- [Documentation Cortex XDR](#)
- [API Reference Cortex XDR](#)

### Communauté et support

- [GitHub CortexDFIR-Forge](#)
- [Forum Palo Alto Networks](#)
- [Slack CortexDFIR-Forge](#)

## Formations et tutoriels

- [Tutoriels vidéo CortexDFIR-Forge](#)
- [Cours en ligne Cortex XDR](#)
- [Webinaires archivés](#)

## Articles et livres blancs

- [Guide des meilleures pratiques DFIR](#)
- [L'art de la détection des menaces](#)
- [Forensique numérique et réponse aux incidents](#)

## Outils complémentaires

- [MISP \(Malware Information Sharing Platform\)](#)
- [TheHive Project](#)
- [Velociraptor](#)
- [Volatility Framework](#)

## Normes et frameworks

- [NIST Cybersecurity Framework](#)
- [SANS Incident Handler's Handbook](#)
- [ISO/IEC 27043:2015](#) - Principes et processus d'investigation d'incidents de sécurité

## Veille sécurité

- [CERT-FR](#)
- [US-CERT](#)
- [MITRE CVE](#)
- [ANSSI](#)

## Index analytique

**A** - Administration, 98-112 - Alertes, 56, 78, 102-104 - API Cortex XDR, 42-55, 142-145 - Architecture, 24-26, 38-40 - Authentification, 44-46, 122-124

**B** - Bonnes pratiques, 120-132, 156-158

**C** - Cas d'usage, 78-97 - Chiffrement, 124-126 - Configuration, 42-55, 98-112 - Conformité, 128-132



**D** - Déploiement, 32-41, 98-102 - Dépannage, 113-121 - Détection, 56-65, 76-77, 134-138 - Docker, 36-41, 98-102

**E** - Environnement, 27-31

**F** - FAQ, 118-121 - Forensique, 88-92

**I** - Installation, 32-41 - Intégration, 142-145

**J** - Journalisation, 126-128

**K** - Kubernetes, 40-41

**M** - Maintenance, 98-112 - Monitoring, 102-106

**P** - Performance, 110-112 - Personnalisation, 133-145 - Prérequis, 27-31

**R** - RBAC, 122-124 - Reporting, 72-77 - RGPD, 128-130

**S** - Sécurité, 122-132 - SOC, 78-87

**T** - Threat Hunting, 92-97

**U** - Utilisateurs, 106-109

**Y** - YARA, 134-136