

Guide d'Installation Détaillé - CortexDFIR-Forge

Ce guide fournit des instructions complètes pour installer et configurer CortexDFIR-Forge dans différents environnements.

Table des Matières

- Prérequis Système
- Installation Rapide
- Installation Détaillée
- Configuration
- Vérification
- Dépannage
- Désinstallation

Prérequis Système

Système d'Exploitation

- **Windows 10/11** (64-bit) - Recommandé
- **Ubuntu 20.04 LTS** ou plus récent
- **CentOS 8** ou plus récent
- **macOS 10.15** ou plus récent (support limité)

Logiciels Requis

Python

```
# Version requise : Python 3.8 - 3.11  
python --version # Doit afficher Python 3.8.x à 3.11.x
```

Git

```
git --version # Requis pour cloner le repository
```

Ressources Système

- **RAM** : Minimum 8 GB, Recommandé 16 GB
- **Stockage** : Minimum 5 GB d'espace libre
- **Processeur** : x64, minimum 2 cœurs
- **Réseau** : Accès Internet pour l'API Cortex XDR

Installation Rapide

Option 1 : Script Automatique (Windows)

```
# Télécharger et exécuter le script d'installation  
curl -O https://raw.githubusercontent.com/servais1983/CortexDFIR-  
Forge/master/install-windows.bat  
install-windows.bat
```

Option 2 : Script Automatique (Linux/macOS)

```
# Télécharger et exécuter le script d'installation  
curl -fsSL https://raw.githubusercontent.com/servais1983/CortexDFIR-  
Forge/master/install-unix.sh | bash
```

Installation Détaillée

Étape 1 : Préparation de l'Environnement

Windows

```
# 1. Installer Python depuis Microsoft Store ou python.org
# 2. Installer Git depuis git-scm.com
# 3. Ouvrir PowerShell en tant qu'administrateur

# Vérifier l'installation
python --version
git --version
pip --version
```

Ubuntu/Debian

```
# Mise à jour du système
sudo apt update && sudo apt upgrade -y

# Installation des dépendances système
sudo apt install -y python3 python3-pip python3-venv git build-essential
sudo apt install -y libmagic1 libmagic-dev
sudo apt install -y libssl-dev libffi-dev python3-dev

# Installation des dépendances pour l'interface graphique (si nécessaire)
sudo apt install -y python3-pyqt5 python3-pyqt5.qtwebkit
```

CentOS/RHEL

```
# Installation des dépendances
sudo dnf update -y
sudo dnf install -y python3 python3-pip git gcc gcc-c++ make
sudo dnf install -y file-devel openssl-devel libffi-devel python3-devel

# Installation d'EPEL pour des packages supplémentaires
sudo dnf install -y epel-release
```

Étape 2 : Clonage du Repository

```
# Cloner le repository
git clone https://github.com/servais1983/CortexDFIR-Forge.git
cd CortexDFIR-Forge

# Vérifier le contenu
ls -la
```

Étape 3 : Création de l'Environnement Virtuel

```
# Créer un environnement virtuel Python
python -m venv venv

# Activer l'environnement virtuel
# Windows :
venv\Scripts\activate
# Linux/macOS :
source venv/bin/activate

# Vérifier l'activation
which python # Doit pointer vers venv/bin/python
```

Étape 4 : Installation des Dépendances

```
# Mise à jour de pip
python -m pip install --upgrade pip setuptools wheel

# Installation des dépendances principales
pip install -r requirements-updated.txt

# Vérification de l'installation
pip list | grep -E "(PyQt5|requests|yara-python)"
```

Étape 5 : Installation des Règles YARA

```
# Les règles YARA sont déjà incluses dans le repository
# Vérifier leur présence
ls -la rules/

# Compiler les règles (optionnel, pour vérification)
python -c "import yara; yara.compile(filepath='rules/apt_apt28.yar')"
```

Configuration

Configuration Cortex XDR

1. Créer le fichier de configuration :

```
cp config/config.example.json config/config.json
```

1. Éditer la configuration :

```
{
  "cortex_xdr": {
    "base_url": "https://api-{fqdn}.xdr.paloaltonetworks.com",
    "api_key": "VOTRE_API_KEY",
    "api_key_id": "VOTRE_API_KEY_ID",
    "tenant_id": "VOTRE_TENANT_ID",
    "advanced_api": true
  },
  "analysis": {
    "max_file_size": "60GB",
    "timeout": 300,
    "parallel_analysis": true
  },
  "reporting": {
    "output_format": "html",
    "include_screenshots": true,
    "detailed_analysis": true
  }
}
```

1. Sécuriser la configuration :

```
# Limiter les permissions du fichier de configuration
chmod 600 config/config.json

# Optionnel : utiliser des variables d'environnement
export CORTEX_API_KEY="votre_api_key"
export CORTEX_API_KEY_ID="votre_api_key_id"
```

Configuration des Logs

```
# Créer le répertoire de logs
mkdir -p logs

# Configurer la rotation des logs (Linux)
sudo tee /etc/logrotate.d/cortexdfir-forge > /dev/null <<EOF
/path/to/CortexDFIR-Forge/logs/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    copytruncate
}
EOF
```

Vérification de l'Installation

Test Basique

```
# Tester l'importation des modules principaux
python -c "
from src.core.cortex_client import CortexClient
from src.core.analyzer import FileAnalyzer
from src.core.report_generator import ReportGenerator
print('✅ Tous les modules s\'importent correctement')
"
```

Test de Connectivité Cortex XDR

```
# Tester la connexion (sans fichier sensible)
python src/main.py --test-connection
```

Test Complet

```
# Exécuter les tests unitaires
python -m pytest tests/ -v

# Test de performance
python src/main.py --benchmark
```

Dépannage

Problèmes Courants

1. Erreur d'Import PyQt5

Symptôme :

```
ImportError: No module named 'PyQt5'
```

Solution Windows :

```
pip install PyQt5==5.15.10
# Si échec, essayer :
conda install pyqt5
```

Solution Linux :

```
sudo apt install python3-pyqt5  
pip install PyQt5==5.15.10
```

2. Erreur YARA

Symptôme :

```
ImportError: No module named 'yara'
```

Solution :

```
# Installation manuelle de YARA  
pip uninstall yara-python  
pip install --no-cache-dir yara-python==4.5.1  
  
# Linux : installer depuis les sources si nécessaire  
sudo apt install libyara-dev  
pip install yara-python==4.5.1
```

3. Erreur de Permissions

Symptôme :

```
PermissionError: [Errno 13] Permission denied
```

Solution :

```
# Vérifier les permissions  
ls -la config/  
chmod 755 src/  
chmod 600 config/config.json  
  
# Windows : exécuter en tant qu'administrateur si nécessaire
```

4. Erreur Cortex XDR API

Symptôme :

```
ConnectionError: Failed to connect to Cortex XDR
```

Solution :

1. Vérifier la connectivité réseau
2. Valider les credentials API
3. Vérifier l'URL de base
4. Tester en mode simulation :

```
python src/main.py --simulate-cortex
```

Logs de Débogage

```
# Activer le mode debug
export DEBUG=1
python src/main.py --log-level DEBUG

# Consulter les logs
tail -f logs/cortexdfir-forge.log
```

Collecte d'Informations pour le Support

```
# Générer un rapport de diagnostic
python src/utils/diagnostic.py > diagnostic-report.txt
```

Tests de Performance

Test de Charge

```
# Test avec fichiers multiples
python tests/performance/load_test.py

# Benchmark YARA
python tests/performance/yara_benchmark.py
```

Monitoring

```
# Surveiller l'utilisation des ressources
python src/utils/monitor.py --duration 300
```


Désinstallation

Désinstallation Complète

```
# 1. Arrêter tous les processus
pkill -f "cortexdfir-forge"

# 2. Supprimer l'environnement virtuel
deactivate
rm -rf venv/

# 3. Supprimer les fichiers de configuration (optionnel)
rm -rf config/config.json logs/

# 4. Supprimer le répertoire du projet
cd ..
rm -rf CortexDFIR-Forge/
```

Nettoyage Partiel

```
# Nettoyer seulement les fichiers temporaires
python src/utils/cleanup.py --temp-only

# Réinitialiser la configuration
cp config/config.example.json config/config.json
```

Support et Aide

Documentation Supplémentaire

Communauté

- **Issues GitHub** : <https://github.com/servais1983/CortexDFIR-Forge/issues>
- **Discussions GitHub** : <https://github.com/servais1983/CortexDFIR-Forge/discussions>

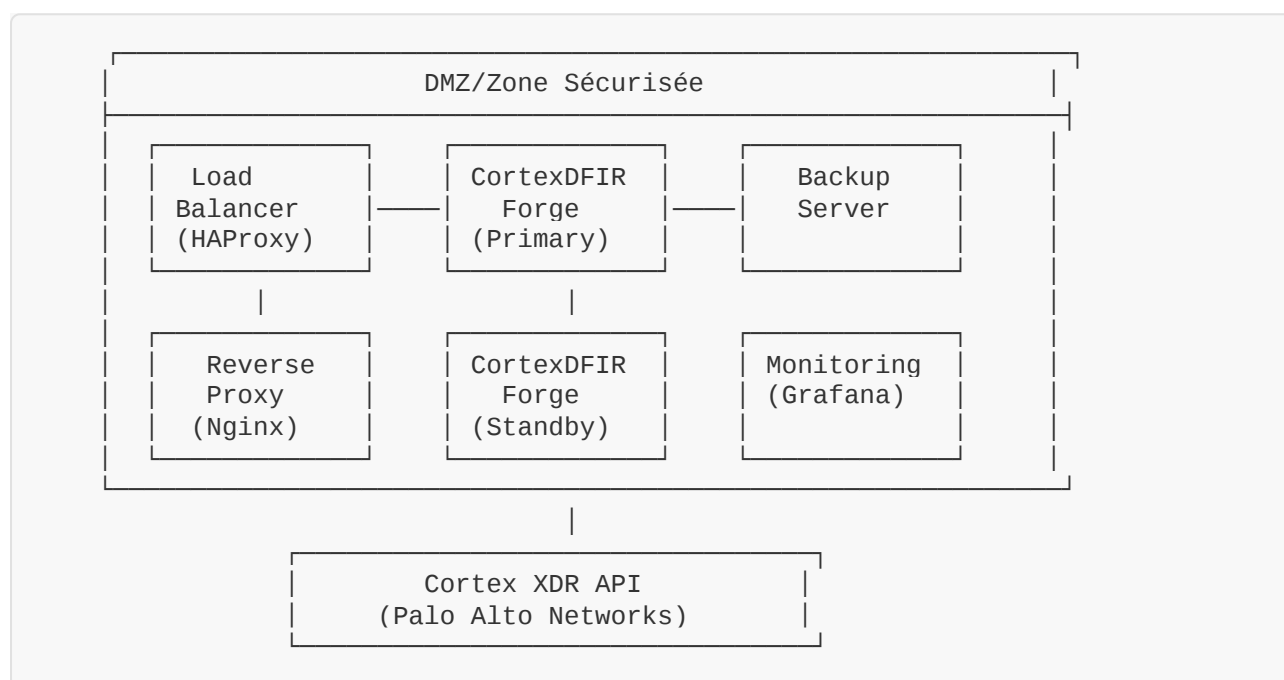
Guide de Déploiement en Production -

CortexDFIR-Forge

Ce guide détaille les meilleures pratiques pour déployer CortexDFIR-Forge dans un environnement de production sécurisé.

Architecture de Production

Architecture Recommandée



Prérequis Infrastructure

Spécifications Serveur

Serveur Principal (Production)

- **CPU :** 8+ cœurs (Intel Xeon ou AMD EPYC)
- **RAM :** 32 GB minimum, 64 GB recommandé
- **Stockage :**
 - SSD 500 GB pour l'OS et applications

- SSD 2 TB pour les données et analyses
 - Stockage réseau pour les sauvegardes
- **Réseau** : 1 Gbps minimum, 10 Gbps recommandé
- **OS** : Ubuntu 22.04 LTS ou RHEL 9

Déploiement avec Docker

Dockerfile de Production

```
# Dockerfile
FROM python:3.11-slim-bullseye

# Métadonnées
LABEL maintainer="CortexDFIR-Forge Team"
LABEL version="1.0.0"
LABEL description="CortexDFIR-Forge Production Container"

# Variables d'environnement
ENV PYTHONUNBUFFERED=1
ENV PYTHONDONTWRITEBYTECODE=1
ENV APP_HOME=/app
ENV APP_USER=cortexdfir
ENV APP_GROUP=cortexdfir

# Création de l'utilisateur non-privilégié
RUN groupadd -r ${APP_GROUP} && \
    useradd -r -g ${APP_GROUP} -d ${APP_HOME} -s /bin/bash ${APP_USER}

# Installation des dépendances système
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
        libmagic1 \
        libmagic-dev \
        build-essential \
        libssl-dev \
        libffi-dev \
        curl \
        wget \
        ca-certificates && \
    rm -rf /var/lib/apt/lists/*

# Création du répertoire de travail
WORKDIR ${APP_HOME}

# Copie des fichiers de dépendances
COPY requirements-updated.txt ./

# Installation des dépendances Python
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements-updated.txt

# Copie du code source
COPY --chown=${APP_USER}:${APP_GROUP} . .

# Configuration des permissions
RUN chown -R ${APP_USER}:${APP_GROUP} ${APP_HOME} && \
    chmod -R 755 ${APP_HOME}/src && \
    chmod -R 644 ${APP_HOME}/rules && \
    chmod 600 ${APP_HOME}/config/config.json

# Changement vers l'utilisateur non-privilégié
USER ${APP_USER}
```

Exposition du port

EXPOSE 8000

Health check

HEALTHCHECK --interval=30s --timeout=10s --start-period=40s --retries=3 \
CMD python src/utils/health_check.py || **exit** 1

Point d'entrée

ENTRYPOINT ["python", "src/main.py"]

CMD ["--server", "--port", "8000"]

Docker Compose pour Production

```
# docker-compose.prod.yml
version: '3.8'

services:
  cortexdfir-forge:
    build:
      context: .
      dockerfile: Dockerfile
    image: cortexdfir-forge:latest
    container_name: cortexdfir-forge-prod
    restart: unless-stopped
    environment:
      - ENV=production
      - LOG_LEVEL=INFO
      - CORTEX_API_KEY_FILE=/run/secrets/cortex_api_key
      - CORTEX_API_KEY_ID_FILE=/run/secrets/cortex_api_key_id
    volumes:
      - ./data:/app/data
      - ./logs:/app/logs
      - ./config:/app/config:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "127.0.0.1:8000:8000"
    networks:
      - cortex-network
    secrets:
      - cortex_api_key
      - cortex_api_key_id
    depends_on:
      - redis
      - prometheus
    deploy:
      resources:
        limits:
          memory: 16G
          cpus: '4.0'
        reservations:
          memory: 8G
          cpus: '2.0'

  nginx:
    image: nginx:1.25-alpine
    container_name: cortexdfir-nginx
    restart: unless-stopped
    ports:
      - "443:443"
      - "80:80"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
      - ./nginx/ssl:/etc/nginx/ssl:ro
      - ./logs/nginx:/var/log/nginx
    networks:
      - cortex-network
    depends_on:
      - cortexdfir-forge

  redis:
    image: redis:7-alpine
```

```
container_name: cortexdfir-redis
restart: unless-stopped
command: redis-server --appendonly yes --requirepass ${REDIS_PASSWORD}
volumes:
  - redis-data:/data
networks:
  - cortex-network
deploy:
  resources:
    limits:
      memory: 1G
      cpus: '0.5'

prometheus:
  image: prom/prometheus:latest
  container_name: cortexdfir-prometheus
  restart: unless-stopped
  command:
    - '--config.file=/etc/prometheus/prometheus.yml'
    - '--storage.tsdb.path=/prometheus'
    - '--web.console.libraries=/etc/prometheus/console_libraries'
    - '--web.console.templates=/etc/prometheus/consoles'
    - '--storage.tsdb.retention.time=200h'
    - '--web.enable-lifecycle'
  volumes:
    - ./monitoring/prometheus.yml:/etc/prometheus/prometheus.yml:ro
    - prometheus-data:/prometheus
  ports:
    - "127.0.0.1:9090:9090"
  networks:
    - cortex-network

grafana:
  image: grafana/grafana:latest
  container_name: cortexdfir-grafana
  restart: unless-stopped
  environment:
    - GF_SECURITY_ADMIN_PASSWORD=${GRAFANA_PASSWORD}
    - GF_INSTALL_PLUGINS=grafana-clock-panel,grafana-simple-json-datasource
  volumes:
    - grafana-data:/var/lib/grafana
    - ./monitoring/grafana:/etc/grafana/provisioning
  ports:
    - "127.0.0.1:3000:3000"
  networks:
    - cortex-network
  depends_on:
    - prometheus

networks:
  cortex-network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.20.0.0/16

volumes:
  redis-data:
  prometheus-data:
  grafana-data:

secrets:
```

```
cortex_api_key:  
  external: true  
cortex_api_key_id:  
  external: true
```


Configuration de Sécurité

Configuration Nginx SSL

```
# nginx/nginx.conf
events {
    worker_connections 1024;
}

http {
    include      /etc/nginx/mime.types;
    default_type application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;
    error_log   /var/log/nginx/error.log   warn;

    sendfile    on;
    #tcp_nopush  on;

    keepalive_timeout 65;

    #gzip on;

    server {
        listen 80;
        server_name your_domain.com;
        return 301 https://$host$request_uri;
    }

    server {
        listen 443 ssl;
        server_name your_domain.com;

        ssl_certificate /etc/nginx/ssl/nginx.crt;
        ssl_certificate_key /etc/nginx/ssl/nginx.key;

        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout 5m;

        ssl_ciphers HIGH:!aNULL:!MD5;
        ssl_prefer_server_ciphers on;

        location / {
            proxy_pass http://cortexdfir-forge:8000;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```

Gestion des Secrets Docker

```
# Créer les secrets Docker
echo "VOTRE_API_KEY" | docker secret create cortex_api_key -
echo "VOTRE_API_KEY_ID" | docker secret create cortex_api_key_id -
```

Monitoring et Observabilité

Configuration Prometheus

```
# monitoring/prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'cortexdfir-forge'
    static_configs:
      - targets: ['cortexdfir-forge:8000']

  - job_name: 'node_exporter'
    static_configs:
      - targets: ['node_exporter:9100']
```

Configuration Grafana

```
# monitoring/grafana/provisioning/datasources/datasource.yml
apiVersion: 1

datasources:
  - name: Prometheus
    type: prometheus
    url: http://prometheus:9090
    access: proxy
    isDefault: true

# monitoring/grafana/provisioning/dashboards/dashboard.yml
apiVersion: 1

providers:
  - name: 'CortexDFIR-Forge Dashboards'
    orgId: 1
    folder: ''
    type: file
    disableDeletion: false
    editable: true
    options:
      path: /var/lib/grafana/dashboards
```

Sauvegarde et Récupération

Stratégie de Sauvegarde

- **Sauvegarde des données** : Les données d'analyse et les logs sont montés en volume Docker, ce qui facilite la sauvegarde.
- **Sauvegarde de la configuration** : Le fichier `config/config.json` doit être sauvegardé régulièrement.
- **Sauvegarde des secrets** : Les secrets Docker doivent être gérés et sauvegardés en toute sécurité.

Procédure de Récupération

1. **Restaurer les volumes Docker** : Restaurer les volumes `data`, `logs`, `redis-data`, `prometheus-data`, `grafana-data`.
2. **Restaurer les fichiers de configuration** : Remplacer `config/config.json` et les fichiers Nginx/Prometheus/Grafana.
3. **Recréer les secrets Docker** : Utiliser les commandes `docker secret create`.
4. **Redémarrer les services** : `docker-compose -f docker-compose.prod.yml up -d`.

Maintenance et Mises à Jour

Mise à Jour du Projet

```
# Mettre à jour le code source
git pull origin master

# Reconstruire et redémarrer les conteneurs
docker-compose -f docker-compose.prod.yml build --no-cache
docker-compose -f docker-compose.prod.yml up -d --force-recreate
```

Nettoyage des Logs et Données

```
# Nettoyer les logs Docker
docker system prune -a

# Nettoyer les données d'analyse (via script CortexDFIR-Forge)
python src/utils/cleanup.py --old-data
```

Dépannage Production

Problèmes Courants

1. Conteneurs non démarrés

Symptôme :

```
docker-compose ps
Name                  Command                  State      Ports
-----
cortexdfir-forge-prod "python src/main.py --s..." Exit 1
```

Solution :

- Vérifier les logs du conteneur : `docker logs cortexdfir-forge-prod`
- Vérifier la configuration dans `config/config.json`
- S'assurer que les secrets Docker sont correctement créés

2. Accès Nginx refusé

Symptôme :

```
403 Forbidden ou 502 Bad Gateway
```

Solution :

- Vérifier les logs Nginx : `docker logs cortexdfir-nginx`
- Vérifier la configuration Nginx (`nginx/nginx.conf`)
- S'assurer que le conteneur `cortexdfir-forge` est en cours d'exécution

3. Problèmes de Performance

Symptôme :

L'application est lente, les analyses prennent du temps

Solution :

- Vérifier l'utilisation des ressources (CPU, RAM) via Grafana ou `docker stats`
- Augmenter les limites de ressources dans `docker-compose.prod.yml`
- Optimiser les requêtes API Cortex XDR

Support et Aide

Documentation Supplémentaire

- **README du projet** : <https://github.com/servais1983/CortexDFIR-Forge/blob/master/README.md>
- **Guide d'installation détaillé** : <https://github.com/servais1983/CortexDFIR-Forge/blob/master/docs/INSTALLATION.md>
- **Guide de migration de région** : https://github.com/servais1983/CortexDFIR-Forge/blob/master/docs/REGION_MIGRATION.md
- **Guide de configuration EU Cortex XDR** : https://github.com/servais1983/CortexDFIR-Forge/blob/master/docs/CORTEX_XDR_EU_CONFIG.md
- **Améliorations et nouvelles fonctionnalités** : <https://github.com/servais1983/CortexDFIR-Forge/blob/master/docs/improvements.md>
- **Guide d'utilisation** : https://github.com/servais1983/CortexDFIR-Forge/blob/master/docs/user_guide.md

Communauté

- **Issues GitHub** : <https://github.com/servais1983/CortexDFIR-Forge/issues>

- Discussions GitHub : <https://github.com/servais1983/CortexDFIR-Forge/discussions>

Guide de Migration des Régions Cortex XDR

Présentation

Ce guide vous accompagne dans la migration de CortexDFIR-Forge d'une région Cortex XDR à une autre. Le projet est maintenant configuré par défaut pour la région **Europe (EU)**, mais peut facilement être adapté pour d'autres régions.

Régions Disponibles

Région	URL de Base	Code
Europe	<code>https://api-eu.xdr.paloaltonetworks.com</code>	EU
États-Unis	<code>https://api-us.xdr.paloaltonetworks.com</code>	US
Asie-Pacifique	<code>https://api-apac.xdr.paloaltonetworks.com</code>	APAC

Étapes de Migration

1. Identifier votre région actuelle

```
# Vérifier la configuration actuelle
python src/utils/test_cortex_connection.py
```

La région actuelle sera affichée dans la section "Test du client Cortex XDR".

2. Sauvegarder la configuration actuelle

```
# Créer une sauvegarde
cp .env .env.backup
cp config/config.yaml config/config.yaml.backup
```

3. Mettre à jour la configuration

Option A : Via le fichier .env (Recommandé)

Éditez votre fichier `.env` :

```
# Pour EU (par défaut)
CORTEX_BASE_URL=https://api-eu.xdr.paloaltonetworks.com

# Pour US
CORTEX_BASE_URL=https://api-us.xdr.paloaltonetworks.com

# Pour APAC
CORTEX_BASE_URL=https://api-apac.xdr.paloaltonetworks.com
```

Option B : Via config.yaml

Éditez `config/config.yaml` :

```
cortex:
  base_url: https://api-eu.xdr.paloaltonetworks.com # Remplacez par votre
région
  use_env_secrets: true
```

4. Vérifier les clés API

⚠ **Important** : Les clés API sont spécifiques à chaque région. Si vous changez de région, vous devrez :

1. Vous connecter à la console Cortex XDR de la nouvelle région
2. Générer de nouvelles clés API
3. Mettre à jour votre fichier `.env` :

```
CORTEX_API_KEY=nouvelle_cle_api
CORTEX_API_KEY_ID=nouveau_id_cle_api
CORTEX_TENANT_ID=nouveau_tenant_id
```

5. Tester la nouvelle configuration

```
# Test de connexion
python src/utils/test_cortex_connection.py

# Test unitaire
python -m pytest tests/test_cortex_client.py -v
```

6. Redémarrer les services

Pour Docker

```
docker-compose down
docker-compose up -d
```

Pour l'installation locale

```
# Arrêter l'application
# Ctrl+C ou fermer le terminal

# Redémarrer
python src/main.py
```

Script de Migration Automatique

Vous pouvez utiliser ce script pour automatiser la migration :


```

#!/bin/bash
# migrate_region.sh

REGION=$1

if [ -z "$REGION" ]; then
    echo "Usage: ./migrate_region.sh [EU|US|APAC]"
    exit 1
fi

case $REGION in
    EU)
        URL="https://api-eu.xdr.paloaltonetworks.com"
        ;;
    US)
        URL="https://api-us.xdr.paloaltonetworks.com"
        ;;
    APAC)
        URL="https://api-apac.xdr.paloaltonetworks.com"
        ;;
    *)
        echo "Région invalide. Utilisez EU, US ou APAC"
        exit 1
        ;;
esac

# Backup
cp .env .env.backup.$(date +%Y%m%d_%H%M%S)

# Update .env
if [ -f .env ]; then
    sed -i "s|CORTEX_BASE_URL=. *|CORTEX_BASE_URL=$URL|" .env
else
    echo "CORTEX_BASE_URL=$URL" >> .env
fi

echo "✅ Migration vers la région $REGION terminée"
echo "📍 Nouvelle URL : $URL"
echo ""
echo "⚠️ N'oubliez pas de :"
echo "1. Générer de nouvelles clés API pour cette région"
echo "2. Mettre à jour CORTEX_API_KEY, CORTEX_API_KEY_ID et CORTEX_TENANT_ID"
echo "3. Tester la connexion avec : python src/utils/test_cortex_connection.py"

```

Comparaison des Régions

Performances

Région	Latence depuis l'Europe	Latence depuis US	Latence depuis Asie
EU	< 50ms	100-150ms	200-300ms
US	100-150ms	< 50ms	150-250ms
APAC	200-300ms	150-250ms	< 50ms

Conformité

Région	GDPR	SOC 2	ISO 27001
EU	✓ Prioritaire	✓	✓
US	✓	✓ Prioritaire	✓
APAC	✓	✓	✓ Prioritaire

FAQ

Q : Puis-je utiliser les mêmes clés API pour plusieurs régions ?

R : Non, les clés API sont spécifiques à chaque région. Vous devez générer de nouvelles clés pour chaque région.

Q : Mes données sont-elles synchronisées entre les régions ?

R : Non, chaque région Cortex XDR est indépendante. Les incidents, alertes et configurations ne sont pas partagés entre régions.

Q : Comment choisir la bonne région ?


R : Choisissez la région la plus proche de vos utilisateurs principaux ou celle qui correspond à vos exigences de conformité.

Q : Puis-je connecter plusieurs régions simultanément ?

R : Non, une instance de CortexDFIR-Forge ne peut se connecter qu'à une seule région à la fois. Pour plusieurs régions, déployez plusieurs instances.


Problèmes Courants

Erreur d'authentification après migration

 Échec de l'authentification. Vérifiez vos clés API.

Solution : Vous utilisez probablement des clés API de l'ancienne région. Générez de nouvelles clés dans la console Cortex XDR de la nouvelle région.

Timeout de connexion

 Erreur lors du test de connexion : timeout

Solution :



1. Vérifiez votre connexion internet
2. Vérifiez que l'URL de la région est correcte
3. Vérifiez les règles de pare-feu pour autoriser les connexions HTTPS sortantes

Incidents/Alertes manquants

Solution : Les données ne sont pas synchronisées entre régions. Vous devrez exporter/importer manuellement si nécessaire.

Support

Pour toute question sur la migration :

-  Email : support@cortexdfir-forge.com
-  GitHub Issues : [Signaler un problème](#)

-  Documentation : [Guide complet](#)

Dernière mise à jour : Juin 2025

Configuration Cortex XDR pour la Région EU (Europe)

Spécificités de la Région Europe

URL de Base







```
https://api-eu.xdr.paloaltonetworks.com
```

Localisation des Données

- **Data Centers:** Allemagne et Pays-Bas
- **Stockage:** Toutes les données sont stockées dans l'UE
- **Traitement:** Aucun transfert de données hors UE

Conformité GDPR

Exigences Respectées

-  **Localisation des données:** Données hébergées exclusivement dans l'UE
-  **Droit à l'effacement:** Support complet du "droit à l'oubli"
-  **Portabilité des données:** Export dans des formats standards
-  **Consentement:** Gestion des consentements intégrée
-  **Notification de violation:** Alertes automatisées sous 72h
-  **Chiffrement:** AES-256 pour les données au repos, TLS 1.3 en transit

Configuration Recommandée pour la Conformité

```
# config/config.yaml - Configuration EU GDPR-compliant
cortex:
  base_url: https://api-eu.xdr.paloaltonetworks.com
  use_env_secrets: true
  advanced_api: true

security:
  enable_ssl: true
  verify_certificates: true
  timeout: 300
  min_tls_version: "1.2"

privacy:
  data_retention_days: 365 # Maximum 1 an par défaut
  anonymize_pii: true
  audit_logging: true





gdpr:
  enabled: true
  data_processor_agreement: true
  breach_notification_email: dpo@votre-entreprise.com
  retention_policy: "auto_delete"

logging:
  level: INFO
  anonymize_ip: true
  exclude_sensitive_data: true
```



Permissions API Recommandées

Pour une conformité maximale, limitez les permissions aux besoins stricts :

Permissions Essentielles

-  incidents.read - Lecture des incidents
-  alerts.read - Lecture des alertes
-  endpoints.read - Lecture des endpoints
-  files.upload - Upload de fichiers pour analyse

Permissions Optionnelles (selon besoins)

-  incidents.write - Modification des incidents
-  alerts.write - Modification des alertes

- ⚠️ `xql.execute` - Exécution de requêtes XQL
- ⚠️ `endpoints.write` - Actions sur les endpoints

Permissions à Éviter (sauf nécessité absolue)

- ❌ `admin.*` - Permissions administratives
- ❌ `users.*` - Gestion des utilisateurs
- ❌ `settings.*` - Modification des paramètres globaux

Limites et Quotas EU

Ressource	Limite	Remarques
Requêtes API	100/minute	Par clé API
Upload fichiers	100 MB	Par fichier
Résultats XQL	10,000 lignes	Par requête
Retention données	365 jours	Configurable
Incidents actifs	10,000	Par tenant

Points d'Accès EU

API Endpoints

- **Principal:** `api-eu.xdr.paloaltonetworks.com`
- **Backup:** `api-eu-backup.xdr.paloaltonetworks.com`

Console Web

- **URL:** `https://eu.xdr.paloaltonetworks.com`
- **Support:** `support-eu@paloaltonetworks.com`

Configuration de Basculement

Pour la haute disponibilité en EU :

```
# Configuration Python avec failover
CORTEX_EU_ENDPOINTS = [
    "https://api-eu.xdr.paloaltonetworks.com",
    "https://api-eu-backup.xdr.paloaltonetworks.com"
]

def get_cortex_client():
    for endpoint in CORTEX_EU_ENDPOINTS:
        try:
            client = CortexClient(base_url=endpoint)
            if client.test_connection():
                return client
        except:
            continue
    raise Exception("Aucun endpoint EU disponible")
```

Checklist de Conformité EU

Avant le Déploiement

- DPA (Data Processing Agreement) signé avec Palo Alto Networks
- Notification à l'autorité de protection des données (si requis)
- Évaluation d'impact (DPIA) complétée
- Procédures de notification de violation documentées

Configuration Technique

- Clés API créées avec permissions minimales
- Chiffrement activé pour toutes les communications
- Logs d'audit configurés et sécurisés
- Retention des données configurée selon politique

Opérationnel

- Personnel formé sur les procédures GDPR

- Processus de réponse aux demandes GDPR établi
- Tests de restauration et suppression validés
- Monitoring de la conformité en place

Alertes et Notifications

Configuration des Alertes GDPR

```
# alerts/gdpr_alerts.yaml
alerts:
  - name: "Accès aux données personnelles"
    condition: "access_to_pii = true"
    severity: "high"
    notification:
      - email: "dpo@votre-entreprise.com"
      - slack: "#gdpr-alerts"

  - name: "Export de données volumineux"
    condition: "export_size > 1GB"
    severity: "medium"
    notification:
      - email: "security@votre-entreprise.com"

  - name: "Tentative d'accès hors EU"
    condition: "source_country NOT IN ("EU")"
    severity: "critical"
    notification:
      - email: "dpo@votre-entreprise.com"
      - pagerduty: "gdpr-violations"
```

Support EU

Contacts Techniques

- **Email:** cortex-support-eu@paloaltonetworks.com
- **Téléphone:** +49 89 444 456 000 (Allemagne)
- **Horaires:** 8h00 - 18h00 CET/CEST

Contacts Conformité

- **DPO Palo Alto EU:** dpo-eu@paloaltonetworks.com

- **Urgences GDPR:** +31 20 754 3000 (24/7)

Ressources Supplémentaires

Dernière mise à jour : Juin 2025 Version : 2.0

Guide d'installation et d'utilisation - CortexDFIR-Forge

Installation

Prérequis

- Windows 10/11 ou Linux
- Python 3.8 ou supérieur
- Accès à l'API Cortex XDR (optionnel mais recommandé)

Installation sur Windows

1. Clonez le dépôt GitHub :

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

2. Exécutez le script d'installation :

```
installer.bat
```

3. Configurez vos identifiants Cortex XDR dans le fichier `config/config.yaml`

4. Lancez l'application :

```
run.bat
```

Installation sur Linux

1. Clonez le dépôt GitHub :

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

2. Créez un environnement virtuel et installez les dépendances :

```
shell python -m venv venv  
source venv/bin/activate  
pip install -r requirements.txt
```

3. Configurez vos identifiants Cortex XDR dans le fichier `config/config.yaml`

4. Lancez l'application :

```
shell python src/main.py
```

Guide d'utilisation rapide

1. Configuration initiale

Avant la première utilisation, configurez vos identifiants Cortex XDR dans le fichier `config/config.yaml` :

```
cortex:  
  api_key: "votre_api_key"  
  api_key_id: "votre_api_key_id"  
  tenant_id: "votre_tenant_id"  
  base_url: "https://api.xdr.paloaltonetworks.com"
```

2. Interface principale

L'interface principale de CortexDFIR-Forge est divisée en plusieurs sections :

1. **Sélection de fichiers** : Permet de choisir les fichiers à analyser
2. **Types d'analyse** : Options pour sélectionner les types d'analyse à effectuer
3. **Résultats** : Affichage des résultats d'analyse
4. **Barre de progression** : Indique l'avancement de l'analyse

5. **Génération de rapport** : Permet de générer un rapport HTML des résultats

3. Analyse de fichiers

Pour analyser des fichiers :

1. Cliquez sur "Sélectionner des fichiers" et choisissez les fichiers à analyser
2. Cochez les types d'analyse souhaités (Malware, Ransomware, Phishing, Persistance)
3. Cliquez sur "Démarrer l'analyse"
4. Attendez que l'analyse soit terminée
5. Consultez les résultats dans l'interface

4. Génération de rapports

Pour générer un rapport d'analyse :

1. Après une analyse réussie, cliquez sur "Générer un rapport"
2. Sélectionnez le dossier de destination pour le rapport
3. Le rapport HTML sera généré et ouvert automatiquement dans votre navigateur

5. Configuration avancée

Pour accéder aux paramètres avancés :

1. Allez dans l'onglet "Configuration"
2. Cliquez sur "Paramètres Cortex XDR" pour configurer l'API
3. Vous pouvez également personnaliser les paramètres de reporting et d'analyse

Fonctionnalités principales

Analyse multi-format

CortexDFIR-Forge prend en charge l'analyse de différents types de fichiers :

- **Fichiers VMDK** : Analyse des disques virtuels

- **Fichiers logs** : Détection d'indicateurs de compromission
- **Fichiers CSV** : Identification de données suspectes
- **Exécutables** : Détection de malwares et comportements suspects
- **Scripts** : Analyse de code potentiellement malveillant

Détection avancée

L'application offre plusieurs mécanismes de détection :

- **Intégration Cortex XDR** : Utilisation des capacités avancées de Cortex
- **Règles YARA personnalisables** : Détection basée sur des signatures
- **Détection de ransomwares** : Focus particulier sur LockBit 3.0
- **Analyse de phishing** : Identification des tentatives de phishing
- **Détection de persistance** : Identification des mécanismes de persistance

Système de scoring

Le système de scoring évalue la criticité des menaces détectées sur une échelle de 0 à 100 :

- **0-24** : Risque faible
- **25-49** : Risque moyen
- **50-74** : Risque élevé
- **75-100** : Risque critique

Rapports détaillés

Les rapports HTML générés incluent :

- Résumé de l'analyse
- Statistiques globales
- Liste détaillée des menaces détectées
- Visualisations graphiques
- Recommandations

Dépannage

Problèmes courants

Erreur "ModuleNotFoundError: No module named 'PyQt5'"

Solution : Réinstallez PyQt5 avec pip :

```
pip install PyQt5
```

Erreur lors de la connexion à l'API Cortex XDR

Solution : Vérifiez vos identifiants API et votre connexion internet. L'application fonctionnera en mode dégradé sans connexion à l'API.

Erreur lors du chargement des règles YARA

Solution : Vérifiez la syntaxe de vos règles YARA. Des règles par défaut seront créées si aucune n'est trouvée.

Support

Pour toute question ou problème, veuillez créer une issue sur le dépôt GitHub : <https://github.com/servais1983/CortexDFIR-Forge/issues>