

Infrastructure de Red Team

Ce document technique détaille la mise en place et la configuration d'une **infrastructure Red Team**, en mettant l'accent sur les serveurs de **commande et de contrôle (C2)** comme le framework Mythic. Il explique comment utiliser des services **cloud**, des **réseaux de diffusion de contenu (CDN)** et des **redirecteurs** pour masquer l'origine des attaques et imiter le trafic légitime. Différents **canaux de communication**, tels que le HTTPS et le DNS over HTTPS, sont présentés pour assurer la discrétion des opérations. Enfin, le guide aborde l'installation d'**agents spécialisés** et l'utilisation de **proxys SOCKS** pour mener à bien les missions d'évaluation de la sécurité tout en protégeant les serveurs principaux contre la détection.

Comment fonctionne l'infrastructure des équipes Red Team pour simuler du trafic légitime ?

Pour simuler du trafic légitime et échapper à la détection des équipes de défense (Blue Teams), l'infrastructure d'une Red Team s'appuie sur plusieurs techniques et composants d'obfuscation :

1. L'exploitation des plateformes cloud et des réseaux CDN

- **Mélange avec le trafic de fournisseurs réputés** : Les équipes Red Team déploient de plus en plus leurs infrastructures sur des plateformes cloud (comme Azure ou Google Cloud) associées à des réseaux de diffusion de contenu (CDN) tels que Cloudflare ou Azure CDN.
- **Bénéfice de réputation** : Les entreprises légitimes utilisant elles-mêmes ces infrastructures, le trafic provenant des grands fournisseurs cloud est généralement considéré comme fiable et n'est pas bloqué par les filtres de sécurité au périmètre du réseau. Cela empêche les analystes de distinguer aisément les communications malveillantes du trafic légitime.

2. L'usage de redirecteurs et de proxys inverses (*Reverse Proxies*)

- **Masquage du serveur C2** : Des redirecteurs sont placés entre la cible et le serveur de commande et contrôle (C2) principal afin de relayer les flux entrants, de masquer l'adresse réelle du serveur backend et de protéger ce dernier en cas de détection du point de contact externe.
- **Filtrage et réécriture du trafic** : À la différence de relais basiques (« *dumb pipe* » avec socat ou iptables) qui transmettent tout le trafic aveuglément, des proxys inverses sophistiqués (Apache ou Nginx) filtrent les connexions. Ils peuvent vérifier des *User-Agents* spécifiques, filtrer les adresses IP autorisées ou appliquer des réécritures d'URL à la volée afin de ne transférer que le trafic des agents et d'écarter les scanners de sécurité Internet.
- **Cloisonnement réseau**: Les règles de pare-feu sont restreintes pour n'autoriser les flux vers le redirecteur et le backend C2 que s'ils proviennent des adresses IP sources du CDN (par exemple via l'étiquette de service AzureFrontDoor.Backend).

3. Les protocoles standards et chiffrés

- **HTTPS privilégié** : L'utilisation de **HTTPS** avec des certificats SSL valides (comme Let's Encrypt) constitue la méthode standard, car ce protocole est autorisé sur la quasi-totalité des réseaux d'entreprise et chiffre le canal de bout en bout.
- **DNS over HTTPS (DoH)** : Les requêtes DNS transitent au sein de flux HTTPS vers des résolveurs connus (tels que cloudflare-dns.com), contournant ainsi les mécanismes traditionnels de surveillance et d'inspection DNS.

4. La personnalisation des profils C2 et des en-têtes HTTP

- **Imitation de serveurs web courants** : Les frameworks C2 (comme Mythic ou Cobalt Strike) permettent de configurer des profils de communication sur mesure. Les opérateurs personnalisent les URI, les requêtes GET/POST et les en-têtes HTTP pour imiter des technologies d'entreprise légitimes, par exemple en simulant des en-têtes de serveur **Microsoft-IIS/10.0**.

5. La randomisation temporelle grâce au « jitter »

- **Rupture de la périodicité** : Les anciens systèmes C2 utilisaient des intervalles de rappel (*sleep intervals*) fixes et réguliers, facilement repérables par les outils de surveillance comportementale.
- **Variation aléatoire** : L'application d'un pourcentage de **jitter** (gigue) applique un délai aléatoire entre chaque rappel de l'agent, rendant les connexions imprévisibles et indissociables d'une activité humaine ou de requêtes réseau ordinaires.

Explorer plus en détail les étapes de configuration d'un redirecteur Apache combiné au CDN Azure

La mise en place d'un **redirecteur Apache combiné au CDN Azure** repose sur une architecture conçue pour masquer le serveur de commande et de contrôle (C2) derrière des adresses et des services de confiance. Voici les étapes détaillées de déploiement et de configuration :

1. Préparation de l'environnement et prérequis

Pour cette infrastructure, le déploiement s'articule autour des éléments suivants :

- **Serveur redirecteur** : Une instance Azure sous Ubuntu (taille standard B1s).
- **Serveur C2 backend** : Une instance Azure sous Ubuntu (taille standard B2ms) hébergeant le framework C2 (comme Mythic).
- **CDN Azure** : Profil CDN classique associé au domaine du redirecteur.
- **Enregistrement DNS** : Configuration des enregistrements de type **A** pointant initialement vers les serveurs pour valider la résolution de nom et la génération des certificats.

2. Installation d'Apache et activation des modules

Sur le serveur redirecteur, les modules nécessaires pour la réécriture d'URL, la gestion SSL et le proxy inverse doivent être installés et activés :

```
sudo apt update
sudo apt full-upgrade
apt-get install apache2
systemctl stop apache2.service
a2enmod ssl proxy remoteip rewrite proxy_http
```

Ces modules permettent notamment de gérer le chiffrement SSL (ssl), d'agir comme un reverse proxy (proxy, proxy_http), de réécrire les flux à la volée (rewrite) et de transmettre les véritables IP sources via l'en-tête X-Forwarded-For (remoteip).

3. Génération des certificats SSL (Let's Encrypt)

Avant de verrouiller le réseau, les ports **80** et **443** doivent être temporairement ouverts au trafic entrant externe pour permettre la validation des challenges Let's Encrypt par Certbot :

```
apt-get install certbot
certbot certonly --standalone --preferred-challenges http -d
exampledomain.com --register-unsafely-without-email --agree-tos
```

4. Configuration du VirtualHost Apache

Le fichier de configuration d'Apache configure la redirection systématique du flux HTTP non sécurisé vers HTTPS, ainsi que le relais du trafic chiffré vers le backend :

```
<VirtualHost *:80>
    ServerName exampledomain.com
    RewriteEngine On
    RewriteCond %{HTTPS} off
    RewriteRule ^(.*)$ https://%{HTTP\_HOST}%{REQUEST\_URI} [L, R=301]

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

<VirtualHost *:443>
```

ServerName exampledomain.com
RemoteIPHeader X-Forwarded-For

ProxyPass / <https://exampledomain.com/>
ProxyPassReverse / <https://exampledomain.com/>

SSLProxyEngine On
SSLProxyVerify none
SSLProxyProtocol TLSv1.2

SSLCertificateFile
/etc/letsencrypt/live/exampledomain.com/fullchain.pem
SSLCertificateKeyFile
/etc/letsencrypt/live/exampledomain.com/privkey.pem

ErrorLog \${APACHE_LOG_DIR}/error.log
CustomLog \${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

5. Résolution interne vers le serveur C2

Pour que le redirecteur relaie le trafic vers le véritable serveur C2 tout en conservant le nom de domaine HTTPS, l'adresse IP du C2 est ajoutée localement dans le fichier /etc/hosts du redirecteur :

```
echo <C2ServerIP> <C2Domain> >> /etc/hosts
```

*Une fois cette correspondance en place, l'enregistrement public **A** pointant vers le serveur C2 peut être supprimé pour masquer entièrement l'adresse publique du backend aux yeux des scanners Internet.*

6. Filtrage réseau (NSG Azure / Pare-feu)

Dès que les certificats sont générés, l'exposition directe du redirecteur à Internet est bloquée :

- **Sur le redirecteur** : Les règles du groupe de sécurité réseau (NSG) sont restreintes pour n'autoriser les flux entrants sur les ports 80 et

443 que depuis l'étiquette de service Azure (**Service Tag**) :
AzureFrontDoor.Backend.

- **Sur le serveur C2** : Le pare-feu bloque tout accès direct depuis Internet et autorise uniquement le port 443 provenant de l'adresse IP unique du redirecteur Apache.

7. Configuration du point de terminaison CDN Azure

Dans la console Azure, un point de terminaison CDN classique est créé :

- **Type d'origine (Origin type)** : *Custom origin*.
- **Nom d'hôte de l'origine (Origin hostname)** : Le domaine configuré (yourdomain.com).
- **En-tête de l'hôte d'origine (Origin host header)** : yourdomain.com.
- **Ports** : Ports 80 (HTTP) et 443 (HTTPS) activés.
- **Compression** : Désactivée (**Off**).
- **Règles de mise en cache (Caching rules)** : Définies sur « **Bypass caching for query strings** ».

La désactivation de la compression et le contournement de la mise en cache sont essentiels pour ne pas corrompre ou altérer les communications dynamiques entre l'agent et le serveur C2.

8. Validation de la chaîne de communication

Pour vérifier que toute la chaîne (CDN Azure → Redirecteur Apache → Serveur C2) fonctionne correctement de bout en bout, une requête curl est envoyée vers l'URL fournie par Azure CDN :

curl -I <https://<nom-point-de-terminaison>.azureedge.net>

La réception d'un code de statut **HTTP/2 405** confirme que le trafic transite correctement jusqu'au backend (la méthode n'étant pas autorisée par défaut sans URI valide pour l'agent), sans pour autant exposer la présence du C2 aux requêtes génériques.

Comment configurer l'agent Apollo et le profil HTTP de Mythic ?

La configuration de l'agent **Apollo** et du profil **HTTP** au sein de Mythic s'effectue en plusieurs étapes : l'installation des composants, la personnalisation du profil C2 pour le relier à l'infrastructure CDN, puis la génération de la charge utile (*payload*).

1. Installation de l'agent Apollo et du profil HTTP

Par défaut, Mythic est déployé sans aucun agent ni profil C2 préinstallé. L'installation s'effectue en ligne de commande sur le serveur C2 via l'utilitaire mythic-cli :

Installation de l'agent Apollo (.NET) et du profil HTTP

```
sudo ./mythic-cli install github https://github.com/MythicAgents/Apollo.git
```

```
sudo ./mythic-cli install github https://github.com/MythicC2Profiles/http
```

Démarrage des conteneurs Mythic

```
sudo -E ./mythic-cli start
```

```
``` [5]
```

Pour accéder à l'interface graphique de Mythic en toute sécurité, un tunnel SSH local est créé depuis le poste de l'opérateur [6] :

```
```bash
```

```
ssh -L 7443:127.0.0.1:7443 user@mythicserver
```

```
``` [6]
```

La connexion se fait ensuite sur `https://127.0.0.1:7443` (le mot de passe initial du compte `mythic\_admin` se trouve dans le fichier `.env` du répertoire Mythic) [6].

---

## ### 2. Configuration du profil C2 HTTP

Avant d'associer le profil à un agent, il est indispensable de le configurer pour qu'il prenne en charge le chiffrement SSL et qu'il simule un serveur web légitime [2, 3, 7] :

### 1. **\*\*Intégration des certificats SSL\*\*** :

\* Accédez au menu **\*\*Agents & C2\*\*** dans le panneau latéral gauche [3,

6].

- \* Cliquez sur l'icône de presse-papiers associée au profil `http` pour gérer ses fichiers [3, 7].

- \* Copiez le contenu des certificats Let's Encrypt du serveur (`fullchain.pem` et `privkey.pem`) directement dans les fichiers correspondants du profil [3, 7].

## 2. **\*\*Personnalisation des paramètres et des en-têtes HTTP\*\*** :

- \* Modifiez la configuration du profil (`View/Edit Config`) pour masquer les signatures par défaut [4, 7].

- \* Activez le SSL (`"use\_ssl": true`), définissez le port d'écoute sur **\*\*443\*\***, et personnalisez les en-têtes serveur pour imiter une technologie connue telle que **\*\*Microsoft-IIS/10.0\*\*** [4, 7] :

```
```json
{
  "instances": [
    {
      "ServerHeaders": {
        "Server": "Microsoft-IIS/10.0",
        "Cache-Control": "max-age=0, no-cache",
        "Pragma": "no-cache",
        "Connection": "keep-alive",
        "Content-Type": "text/html; charset=utf-8"
      },
      "port": 443,
      "key_path": "privkey.pem",
      "cert_path": "fullchain.pem",
      "debug": false,
      "use_ssl": true
    }
  ]
}
``` [4]
```

3. **\*\*Redémarrage du profil\*\*** : Cliquez sur **\*\*Stop Profile\*\*** puis **\*\*Start Profile\*\*** afin que les modifications soient appliquées [4].

---

### ### 3. Création et configuration du payload Apollo

La création de l'agent s'effectue dans le menu **Create Payload** de Mythic en suivant l'assistant de configuration [3, 4] :

- \* **Étape 1 (Select Target OS)** : Sélectionnez **Windows** [4].
- \* **Étape 2 (Payload Type)** : Choisissez **apollo** comme type d'agent et sélectionnez le format de sortie désiré (par exemple **WinExe** pour un binaire exécutable) [4].
- \* **Étape 3 (Select Commands)** : Cochez les commandes que vous souhaitez intégrer à l'agent (``download``, ``load``, ``ps``, ``ifconfig``, etc.) [4, 8, 9].
- \* **Étape 4 (Select C2 Profiles)** : Sélectionnez le profil **http** et définissez les paramètres de communication [8, 10] :
  - \* **Callback Host** : L'adresse URL de contact externe, soit l'adresse du point de terminaison CDN (ex. : ``https://<nom-cdn>. azureedge.net``) [8, 10].
  - \* **Callback Port** : Port **443** [8, 10].
  - \* **Callback Interval** : L'intervalle de temps fixe entre chaque rappel (ex. : ``30`` secondes) [8, 10].
  - \* **Callback Jitter** : Le pourcentage de variation aléatoire appliqué au délai (ex. : ``20%``) pour éviter une périodicité repérable [8, 10].
  - \* **Encryption Type** : La méthode de chiffrement retenue pour les échanges [8, 10].
  - \* **Kill Date** : La date de fin de validité au-delà de laquelle l'agent s'arrête de communiquer [8, 10].
- \* **Étape 5 (Payload Review)** : Définissez le nom du fichier généré (ex. : ``testing.exe``) ainsi qu'une description descriptive avant de cliquer sur **Create Payload** [10, 11].

Une fois compilée, la charge utile peut être téléchargée depuis la section ``/new/payloads`` de Mythic pour être déployée sur la cible de test [10, 11]. Dès son exécution, la première connexion entrante apparaîtra dans le

tableau de bord de Mythic via le canal HTTPS transitant par le CDN et le redirecteur [11, 12].

---

## Comment configurer un tunnel SOCKS5 avec proxychains4 ?

La configuration d'un tunnel **SOCKS5** avec **proxychains4** permet à l'opérateur d'exploiter la connexion réseau de la machine cible pour y acheminer son trafic sans exécuter d'outils directement sur l'hôte compromis.

Voici les étapes de configuration et d'utilisation :

### 1. Démarrage du proxy SOCKS5 dans Mythic

Dans la console d'interaction avec l'agent Mythic (accessible en cliquant sur l'icône de clavier du callback) :

1. **Lancement du proxy** : Spécifiez le port d'écoute souhaité (par exemple 7006) :

```
socks 7006
```

1. **Passage en mode interactif** : Supprimez les délais de rappel pour permettre une transmission fluide et immédiate des paquets à travers le tunnel :

```
sleep 0 0
```

1. **Suivi des connexions** : L'état du proxy SOCKS est consultable directement depuis l'interface web en cliquant sur l'icône **Socks** ou via l'URL <https://127.0.0.1:7443/new/search?tab=socks>.

### 2. Installation et paramétrage de proxychains4

Sur la machine effectuant les requêtes :

1. **Installation du paquet** :

```
apt-get install proxychains4
```

1. **Édition du fichier de configuration** : Modifiez le fichier  
/etc/proxychains4.conf et ajoutez la ligne de configuration à la fin :  
socks5 127.0.0.1 7006

**Recommandation OPSEC pour les opérations réelles** : Dans un cadre opérationnel, il est conseillé d'installer et d'exécuter proxychains4 depuis un **poste d'opérateur dédié** distinct du serveur C2. Dans ce cas, remplacez 127.0.0.1 par l'adresse IP (publique ou privée) de votre serveur C2 et ouvrez ce port dans les règles de pare-feu du serveur C2 pour l'IP de l'opérateur, évitant ainsi toute attribution directe au serveur C2 en cas de fuite d'adresse IP.

### 3. Vérification du bon acheminement du trafic

Pour confirmer que le flux transite bien par la machine cible compromise, comparez une requête directe et une requête relayée vers un service d'identification d'adresse IP :

# Requête directe (affiche l'IP de votre machine/serveur)  
sudo curl ipconfig.io

# Requête via le tunnel (doit afficher l'IP publique de la machine cible)  
sudo proxychains4 curl ipconfig.io

La sortie de proxychains4 doit indiquer l'établissement d'une chaîne stricte (*Strict chain*) vers 127.0.0.1:7006 et la résolution de la cible avec succès.

### 4. Utilisation pour la post-exploitation

Une fois le tunnel établi, vous pouvez relayer les outils d'attaque usuels à travers la cible tout en maintenant une excellente sécurité opérationnelle (OPSEC), notamment :

- **Impacket**
- **Certipy**
- **SCCMHunter**
- **Idapper**
- **Vibe**

## Comment cartographier et rapporter les actions de votre agent dans Mythic à l'aide de la matrice MITRE ATT&CK

Dans **Mythic**, la cartographie des activités sur la matrice **MITRE ATT&CK** et la génération des rapports s'effectuent directement depuis l'interface web grâce à des fonctionnalités intégrées.

### 1. Accéder à la matrice MITRE ATT&CK dans l'interface

- **Navigation** : Cliquez sur l'icône de tableau en colonnes dans la barre de navigation supérieure ou accédez directement à l'adresse URL <https://127.0.0.1:7443/new/mitre>.
- **Menu « Actions »** : En haut à droite de la matrice, le menu déroulant **Actions** permet de charger deux types de correspondances :
  - **Fetch All Commands Mapped to MITRE** : Affiche toutes les correspondances théoriques associées aux commandes intégrées dans votre agent (par exemple, la commande ifconfig associée à la technique T1590.005 – IP Addresses), même si ces commandes n'ont pas encore été exécutées.
  - **Fetch All Issued Tasks Mapped to MITRE** : Affiche uniquement les techniques correspondant aux **tâches réellement émises et exécutées** au cours de l'opération. Les techniques associées apparaissent alors surlignées en rouge dans la matrice (par exemple, la technique T1029 – Scheduled Transfer pour les ajustements de délai sleep).

### 2. Configurer et générer le rapport opérationnel

- **Accès au module de rapport** : Cliquez sur l'icône de **drapeau (flag)** dans le menu supérieur ou rendez-vous sur <https://127.0.0.1:7443/new/reporting>.
- **Paramètres du rapport** :
  - **Format de sortie (Output Format)** : Choisissez entre un format **HTML** ou **JSON**.
  - **Options MITRE** : Activez les cases à cocher « **Include MITRE ATT&CK Coverage Per Task** » (pour associer chaque commande exécutée à sa technique spécifique) et « **Include**

**MITRE ATT&CK Coverage Overview** » (pour obtenir un récapitulatif global de la couverture).

- **Filtres et exclusions** : Il est possible d'exclure certains utilisateurs, noms d'hôtes ou identifiants de rappel (*Callback IDs*) si vous souhaitez épurer les données.
- **Téléchargement** : Cliquez sur **GENERATE**, puis sur **Download here** une fois le fichier compilé.

### 3. Contenu et exploitation du rapport généré

Le rapport généré fournit une vue d'ensemble détaillée de la mission :

- **Métriques de l'opération** : Liste des opérateurs assignés, utilisateurs compromis, hôtes touchés, nombre de callbacks et total des tâches émises.
- **Journalisation par tâche** : Chaque commande exécutée est horodatée et étiquetée avec l'identifiant MITRE correspondant (comme T1029).
- **Section « MITRE ATT&CK Overview »** : Un tableau récapitule l'ensemble des techniques couvertes, indiquant l'identifiant ATT&CK, le nom de la technique et le nombre d'occurrences durant la mission.

## Quels sont les avantages des frameworks C2 comme Sliver ou Havoc ?

Dans votre document, **Sliver** et **Havoc** sont identifiés parmi les frameworks C2 publics les plus répandus et activement soutenus par la communauté (aux côtés de Mythic et BadRats).

En tant que solutions publiques et open source, leurs principaux avantages par rapport aux solutions commerciales sont les suivants :

- **Une grande flexibilité de personnalisation** : L'avantage majeur de ces frameworks réside dans la liberté de configuration qu'ils offrent aux opérateurs, permettant d'adapter précisément les composants pour une efficacité opérationnelle optimale, là où les solutions commerciales peuvent s'avérer plus restrictives.

- **La réduction des risques de détection par signature EDR** : Les frameworks commerciaux (comme Cobalt Strike) risquent davantage d'être détectés par signature via les EDRs lorsqu'ils sont utilisés avec leurs paramètres d'usine par défaut. La personnalisation poussée permise par les frameworks publics aide à contourner ces signatures génériques, bien qu'ils demandent en contrepartie davantage de travail de configuration initial pour être pleinement opérationnels.

**Ce que vos sources ne couvrent pas** : Vos sources mentionnent Sliver et Havoc dans cette vue d'ensemble comparative, mais **ne détaillent pas les caractéristiques techniques individuelles spécifiques à chacun de ces deux outils** (le reste du document se concentre sur le fonctionnement de **Mythic** avec l'agent Apollo). Le document renvoie toutefois à la référence en ligne « **C2 Matrix** » (<https://howto.thec2matrix.com>) pour comparer en détail les types d'agents et les fonctionnalités propres à chaque framework.

## **Quels sont les autres frameworks C2 open source ?**

Dans votre document, en plus de **Mythic**, **Sliver** et **Havoc**, le seul autre framework C2 public explicitement cité parmi les plus répandus et soutenus par la communauté est **BadRats**.

Le document souligne qu'il existe une multitude d'options (tant commerciales qu'open source) et renvoie vers la ressource de référence « **C2 Matrix** » (<https://howto.thec2matrix.com>), qui permet de passer en revue l'ensemble des solutions C2 disponibles, leurs agents et leurs fonctionnalités (comme la prise en charge de SOCKS).

En dehors de cette liste, vos sources ne mentionnent pas d'autres frameworks C2 open source spécifiques, le reste de l'ouvrage étant axé sur la mise en œuvre de Mythic.



Vos sources ne détaillant pas d'autres outils au-delà de ces mentions, souhaitez-vous que j'effectue des recherches sur le web pour répertorier et comparer d'autres frameworks C2 open source reconnus ?

## Effectue des recherches sur le web pour répertorier et comparer d'autres frameworks C2 open source reconnus

Voici ce que j'ai trouvé sur les **frameworks C2 open source pour Red Team** :

L'écosystème open source propose aujourd'hui des alternatives très matures et diversifiées aux solutions commerciales comme Cobalt Strike, avec des spécialisations marquées selon les priorités opérationnelles : l'évasion EDR, la résilience réseau ou l'automatisation de scénarios d'attaque.

### Thèmes clés identifiés :

1. **Évasion avancée et masquage en mémoire** : Des frameworks modernes comme **Havoc** et **Sliver** intègrent des techniques de pointe (*sleep obfuscation*, *indirect syscalls*, loaders COFF/BOF) spécialement conçues pour contourner les défenses et EDR récents sur Windows.
2. **Diversité et robustesse des protocoles de transport** : L'écosystème s'est affranchi du simple HTTP(S) traditionnel en adoptant nativement des canaux chiffrés et furtifs comme WireGuard, Mutual TLS (mTLS), le DNS chiffré ou le tunneling adaptatif.
3. **Automatisation et cartographie MITRE ATT&CK** : Des plateformes comme **MITRE Caldera** et **BC-Security Empire** mettent l'accent sur l'émulation reproductible des menaces APT et la corrélation directe de chaque action avec la matrice ATT&CK pour faciliter les exercices Purple Team.
4. **Adaptation aux réseaux d'entreprise complexes** : Des projets spécialisés comme **PoshC2** sont conçus pour naviguer au travers des proxys d'entreprise stricts et génèrent automatiquement des règles pour redirecteurs (Apache mod\_rewrite).



- 1. Évasion EDR et comparaison technique des implants
  - **Havoc vs Apollo** : *Comment fonctionnent les mécanismes d'évasion avancés de l'implant Demon de Havoc (notamment les appels*

*système indirects / indirect syscalls et l'obfuscation du sommeil / sleep obfuscation) pour contourner les EDR modernes ?*

- **Transports furtifs avec Sliver** : *Quels sont les avantages comparatifs des protocoles mTLS, WireGuard et DNS pris en charge par Sliver par rapport à un profil HTTP(S) classique derrière un CDN ?*



## 2. Contournement de proxy et déploiement d'entreprise

- **Résilience réseau avec PoshC2** : *En quoi l'architecture de PoshC2 est-elle particulièrement adaptée aux environnements d'entreprise stricts protégés par des proxys authentifiés ?*
- **Automatisation des redirecteurs** : *Comment générer et durcir automatiquement les règles de réécriture Apache (mod\_rewrite) pour s'assurer que seuls les agents C2 légitimes atteignent vos serveurs d'équipe ?*



## 3. Émulation d'adversaires et automatisation Purple Team

- **Simulations avec MITRE Caldera** : *Comment configurer des opérations automatisées dans Caldera pour reproduire les tactiques d'un groupe APT spécifique et évaluer l'efficacité de la détection Blue Team ?*
- **Post-exploitation avec Empire & Starkiller** : *Comment exploiter les modules Active Directory et l'interface multi-opérateurs Starkiller d'Empire lors de la phase de déplacement latéral ?*



## 4. Détection et contre-mesures (perspective Blue Team)

- **Détection du trafic relayé** : *Quelles signatures ou anomalies comportementales permettent aux analystes SOC et aux EDR de repérer l'utilisation d'un tunnel SOCKS5 ou d'un redirecteur hébergé sur Azure CDN ?*

Voici les réponses détaillées aux quatre questions proposées :

## 1. Évasion EDR et protocoles avancés (Havoc Demon et Sliver)

- **Mécanismes d'évasion de Havoc (Implant Demon) :**

- **Architecture modulaire** : Havoc sépare son serveur d'équipe (*Teamserver*, hébergé sur VPS pour centraliser les opérateurs et les écouteurs) de son interface graphique (*Client* en Qt).
- **Techniques anti-EDR** : Son implant Demon est écrit pour déjouer la détection en mémoire grâce à l'**obfuscation de mise en veille (sleep obfuscation)**, qui masque ou chiffre la mémoire de l'agent pendant les périodes d'inactivité afin d'échapper aux scans de mémoire périodiques des EDR. Il exploite également des **appels système indirects (indirect syscalls)**, ce qui permet d'exécuter des actions sensibles sans passer par les hooks placés par les EDR dans les DLL système en espace utilisateur. Enfin, il supporte le chargement dynamique de fichiers objets (*Object Files / BOF*).

- **Protocoles de transport furtifs dans Sliver :**

- **Diversité réseau** : Développé en Go par Bishop Fox, Sliver gère nativement quatre canaux de transport : **Mutual TLS (mTLS)**, **WireGuard**, **HTTP(S)** et **DNS**.
- **Avantages par rapport au HTTP(S) classique** : Le recours à **mTLS** et **WireGuard** utilise des certificats X.509 et des clés asymétriques uniques par binaire généré. Ces flux chiffrés point à point empêchent l'inspection par des proxys d'interception SSL traditionnels. Le protocole **DNS** permet quant à lui de maintenir le contact ou d'exfiltrer des données même lorsque les sorties web directes sont bloquées.
- **Polymorphisme et fonctionnalités Windows** : Chaque implant Sliver est compilé dynamiquement avec une obfuscation au moment de la compilation et des profils HTTP(S) générés de manière procédurale. Il intègre la détection de « canaris » DNS, l'injection de processus, la manipulation de tokens Windows et l'exécution d'assemblies .NET ou de loaders COFF/BOF en mémoire.

## 2. Contournement de proxy d'entreprise et automatisation des redirecteurs (PoshC2)

- **Résilience réseau face aux proxys stricts :**
  - PoshC2 est conçu dès le départ pour être **conscient des proxys (proxy-aware)** au sein des réseaux d'entreprise.
  - Il propose des implants en C#, Python et PowerShell capables de fonctionner sans recourir à `System.Management.Automation.dll` (*implants PowerShell-less*), réduisant drastiquement les alertes liées au moteur PowerShell classique.
  - Ses shellcodes intègrent d'office un **contournement d'AMSI** et le **patching d'ETW (Event Tracing for Windows)** pour empêcher la télémétrie locale d'enregistrer ses activités.
  - Pour les hôtes internes dépourvus de connexion Internet directe, PoshC2 gère le **chaînage d'implants en cascade (daisy-chaining)** via le protocole SMB et des canaux nommés (*named pipes*), permettant de relayer les communications de machine à machine jusqu'au point de sortie autorisé.
- **Génération automatique des redirecteurs Apache :**
  - PoshC2 génère automatiquement les règles de réécriture **Apache (mod\_rewrite)** nécessaires au déploiement de proxys inverses.
  - Ces règles filtrent les requêtes entrantes pour ne transmettre au serveur C2 que celles qui respectent les URI ou en-têtes exacts attendus de l'implant. Toute requête tierce (comme celles émises par des scanners Internet ou des analystes SOC) est bloquée ou redirigée vers une ressource inoffensive, préservant ainsi la sécurité opérationnelle (OPSEC) du backend C2.

## 3. Émulation d'adversaires et automatisation Purple Team (MITRE Caldera et Empire / Starkiller)

- **Automatisation avec MITRE Caldera :**

- Conçu par le MITRE, Caldera est une plateforme d'évaluation et de simulation automatisée d'attaques (*breach-and-attack simulation*) nativement articulée autour du cadre **MITRE ATT&CK**.
- Il s'appuie sur une architecture composée d'un serveur central (avec API REST et interface web) et d'un écosystème de greffons (*plugins*).
- Une simulation repose sur des **profils d'adversaires (Adversary Profiles)** regroupant des **capacités (Abilities)** correspondant aux TTPs d'un groupe APT. Des **planificateurs autonomes (Planners)** évaluent les faits découverts en temps réel (identifiants, hôtes, configurations) pour déclencher logiquement les étapes suivantes (découverte, mouvement latéral, collecte), ce qui permet d'éprouver automatiquement la réactivité des mécanismes de détection Blue Team.
- **Post-exploitation collaborative avec Empire et Starkiller :**
  - Empire repose sur une architecture client-serveur multijoueur écrite en Python 3. Il intègre nativement l'interface graphique web **Starkiller**, permettant à plusieurs opérateurs de coordonner simultanément leurs actions lors d'un engagement.
  - Sa bibliothèque compte plus de **400 modules** en PowerShell, C# et Python.
  - Lors des phases de déplacement latéral et d'audit Active Directory, il donne un accès direct à des outils reconnus exécutables en mémoire (.NET / BOF) tels que **Mimikatz**, **Rubeus**, **Seatbelt**, **Certify** et **SharpSploit**, tout en appliquant une obfuscation intégrée via *ConfuserEx 2* et *Invoke-Obfuscation*.

#### 4. Détection et contre-mesures (Perspective Blue Team)

Pour les équipes de défense (SOC/Blue Team), la détection d'infrastructures C2 relayées et de tunnels s'articule autour de plusieurs signaux :

- **Analyse comportementale des balises (Beacons)** : Même lorsque les opérateurs appliquent un pourcentage de gigue (*jitter*) pour faire varier les délais entre chaque rappel de l'agent, l'analyse temporelle et statistique des connexions sortantes (fréquence cumulée, volume de données échangé et persistance des contacts vers un même point de terminaison) permet d'isoler des comportements de rappel automatisés.
- **Empreintes cryptographiques (JA3 / JA3S et JARM)** : Les bibliothèques et clients utilisés par les agents génèrent des empreintes TLS identifiables lors de la phase de négociation SSL/TLS. Le profilage des empreintes JA3/JA3S (côté client) et JARM (côté serveur) permet de discriminer un binaire C2 d'un navigateur légitime, ce qui pousse d'ailleurs des frameworks comme Empire à implémenter des contournements d'empreintes JA3 et JARM.
- **Analyse des en-têtes et du routage CDN** : Lors de l'utilisation d'un CDN cloud (Azure CDN, Cloudflare) ou d'un redirecteur Apache, le trafic traverse des infrastructures légitimes. La détection repose alors sur la détection d'incohérences dans les en-têtes HTTP (par exemple, des en-têtes simulant Microsoft-IIS/10.0 alors que le comportement réel de la négociation réseau trahit un serveur sous Linux ou une divergence d'en-tête X-Forwarded-For).
- **Surveillance du trafic généré par les tunnels SOCKS5** : Le proxy SOCKS5 évite d'exécuter des outils directement sur le disque de la victime, mais tout le trafic réseau intermédiaire transite par son interface. Les analystes peuvent repérer ces pivots grâce à des flux inhabituels émis depuis ce poste client vers le réseau interne (requêtes Kerberos avec Rubeus, appels LDAP massifs via ldapper ou flux SMB vers des partages administratifs) sans qu'aucun processus utilisateur légitime ne soit associé à ces sockets au niveau de la machine compromise.

**Comment configurer les redirections DNS et les enregistrements A ?**

Dans le cadre d'une infrastructure Red Team, la gestion des **enregistrements A** et de la **résolution DNS** répond à un double impératif : permettre la délivrance initiale des certificats SSL et masquer définitivement le serveur backend aux yeux des scanners Internet.

## 1. Configuration initiale des enregistrements A publics

- **Pointage vers les instances** : Lors de la phase initiale de mise en place, des enregistrements de type **A** doivent être créés auprès du bureau d'enregistrement (*registrar*) pour faire pointer publiquement le ou les domaines utilisés par l'infrastructure vers les adresses IP correspondantes du redirecteur et du serveur C2.
- **Génération des certificats SSL (Let's Encrypt)** : Cette résolution DNS publique est indispensable pour valider les défis HTTP émis par l'autorité de certification Let's Encrypt (certbot). Durant cette opération, les ports **80** et **443** sont temporairement ouverts au trafic entrant externe.

## 2. Suppression de l'enregistrement A public et redirection locale (/etc/hosts)

Une fois les certificats SSL récupérés et stockés, l'architecture durcit son anonymat pour respecter les impératifs d'OPSEC :

- **Suppression de l'enregistrement A du C2** : L'enregistrement A public pointant vers l'adresse IP du serveur C2 est **supprimé** de la zone DNS publique. Cela empêche les scanners Internet et les analystes d'identifier directement l'hôte hébergeant le framework C2 ou de provoquer une suspension de l'instance par le fournisseur de cloud.
- **Résolution interne sur le redirecteur** : Pour que le redirecteur Apache puisse continuer à router les flux vers le serveur C2 sous son nom de domaine HTTPS valide, une entrée statique est ajoutée directement dans le fichier /etc/hosts du redirecteur :

```
echo <C2ServerIP> <C2Domain> >> /etc/hosts
```

- **Relais proxy** : Le fichier de configuration du VirtualHost Apache utilise cette correspondance locale pour transférer les requêtes entrantes vers <https://<C2Domain>/>.

### 3. Acheminement DNS via le réseau CDN

- **Origine du CDN** : Dans la console du CDN (comme Azure Classic CDN), le domaine du redirecteur est renseigné dans les paramètres d'origine (**Origin hostname** et **Origin host header**).
- **Nom de domaine de rappel (Callback Host)** : Au lieu de faire pointer l'agent directement vers le domaine du redirecteur, l'agent C2 est configuré pour communiquer avec le point de terminaison fourni par le CDN (ex. : <https://<nom-endpoint>.azureedge.net>). Les requêtes DNS des postes cibles résolvent ainsi les adresses IP légitimes du réseau de diffusion de contenu plutôt que vos propres serveurs.

### 4. Utilisation du canal DNS pour les communications C2

Au-delà de la résolution web, le protocole DNS est parfois utilisé directement comme canal de transport C2 :

- **Profil C2 DNS (Requêtes TXT)** : Certains profils (comme le profil dns de Mythic) établissent des communications directes en configurant un serveur DNS personnalisé qui écoute et répond aux requêtes de type **TXT** générées par l'agent.
- **DNS over HTTPS (DoH)** : Pour contourner les mécanismes de surveillance du trafic DNS non chiffré au sein des réseaux d'entreprise, les flux DNS peuvent transiter via des connexions chiffrées HTTPS vers des résolveurs publics réputés (comme [cloudflare-dns.com](https://cloudflare-dns.com)), garantissant le chiffrement de bout en bout.

## Quelles sont les fonctionnalités de MITRE Caldera ?

**MITRE Caldera** est une plateforme open source d'émulation d'adversaires et de simulation d'attaques conçue par le MITRE et nativement articulée autour de la matrice **MITRE ATT&CK**. Elle est développée pour orchestrer des exercices autonomes de simulation de brèches (*breach-and-attack*

*simulation*), mener des opérations manuelles de Red Team ou automatiser la réponse aux incidents de sécurité.

L'ensemble de ses fonctionnalités repose sur une architecture modulaire composée d'un **système central (Core system)** et d'un écosystème extensible de **greffons (Plugins)** :

## 1. Architecture et moteur opérationnel

- **Serveur C2 asynchrone** : Le cœur de Caldera intègre un serveur de commande et de contrôle asynchrone doté d'une interface web et d'une API REST complète (API v2) permettant de tout piloter programmatically.
- **Profils d'adversaires (Adversary Profiles) et Capacités (Abilities)** : Les attaques sont modélisées sous forme de profils d'adversaires regroupant des capacités offensives concrètes, chacune étant directement liée à une technique de la matrice ATT&CK.
- **Planificateurs autonomes (Planners)** : Caldera intègre des planificateurs logiques qui décident dynamiquement des prochaines actions à déclencher selon les conditions observées.
- **Gestion des faits (Facts), relations et règles** : La plateforme analyse les sorties de commandes à l'aide de parseurs pour stocker des faits (identifiants découverts, adresses IP, chemins de fichiers) et établir des relations qui alimentent automatiquement les actions suivantes.
- **Objectifs de mission (Objectives & Goals)** : Les opérations peuvent être configurées avec des buts précis à atteindre, servant de critères de succès ou d'arrêt pour l'exercice.

## 2. Gestion des agents et des charges utiles

- **Diversité des agents** : Caldera fournit plusieurs types d'agents selon le contexte :
  - **Sandcat** : L'agent par défaut développé en Go, déployable sous forme d'exécutable ou de DLL.

- **Manx et Mock** : Agents interactifs ou agents de simulation virtuelle pour tester des scénarios sans impacter de vrais systèmes.
- **Caldera for OT** : Extension spécialisée pour les environnements de technologies opérationnelles et industrielles.
- **Compilation dynamique de charges utiles** : Prise en charge de la compilation dynamique de charges utiles personnalisées avant déploiement.
- **Relais Peer-to-Peer (P2P) et tunneling** : Les agents Sandcat peuvent établir des connexions de pair à pair et relayer leurs communications via des proxys P2P ou des tunnels SSH, permettant d'opérer dans des segments réseau isolés d'Internet.

### 3. Canaux de contact et protocoles de transport

Le sous-système de contacts offre une variété de protocoles de communication pour transmettre les ordres et recevoir les résultats des agents :

- **Protocoles réseau standards** : HTTP/HTTPS, WebSockets, TCP brut et UDP.
- **Canaux alternatifs et tiers** : Échanges via requêtes DNS, serveurs FTP, canaux Slack ou extraits GitHub Gist (contact\_gist, contact\_slack).
- **Encodeurs et obfuscateurs** : Modules de masquage de code et de chiffrement des données (comme l'obfuscation Base64 ou en texte brut) pour dissimuler les commandes exécutées.

### 4. Bibliothèque de greffons (*Plugins*)

La richesse fonctionnelle de Caldera provient de son catalogue de plugins :

- **Stockpile** : Référentiel contenant des centaines de capacités offensives préconfigurées et des modules d'exfiltration vers des services variés (AWS, FTP, Dropbox, dépôts GitHub).
- **Compass et Caltack** : Outils de cartographie et de visualisation de la couverture ATT&CK.

- **Atomic** : Intégration directe des tests d'émulation issus du projet *Atomic Red Team*.
- **GameBoard et Human** : Outils pour les exercices Blue vs Red et simulation d'activité humaine sur les postes cibles.
- **Response** : Module dédié à l'exécution de scénarios automatisés de réponse aux incidents pour la défense.
- **Debrief et Builder** : Modules pour l'analyse après action, l'assemblage de composants et le suivi de formation (*Training*).

## 5. Rapports et journalisation

- **Rapports d'opérations (Operation Reports)** : Restitution complète des actions exécutées, des faits découverts et de la chronologie des événements sous forme de rapports exportables.
- **Journaux d'événements (Event Logs)** : Traçabilité détaillée de chaque commande et résultat pour faciliter la corrélation avec les alertes des équipes défensives (Blue Teams).

## Comment auditer la sécurité de mon infrastructure C2 ?

Pour auditer la sécurité et la robustesse opérationnelle (**OPSEC**) d'une infrastructure C2, vous devez vous assurer que le serveur backend reste totalement invisible, que le trafic ne peut pas être attribué à vos opérateurs et que les flux émulent parfaitement des communications légitimes.

Voici les points de contrôle et tests d'audit à effectuer sur votre infrastructure :

### 1. Audit de l'isolation réseau et du cloisonnement (Pare-feu / NSG)

- **Suppression de l'exposition publique du serveur C2** : Vérifiez que l'enregistrement DNS public de type **A** pointant vers l'adresse IP de votre serveur C2 a bien été supprimé une fois les certificats générés. La résolution vers le C2 ne doit être assurée que localement par le fichier `/etc/hosts` de votre redirecteur.
- **Filtrage des flux vers le C2** : Contrôlez les règles de pare-feu du serveur backend pour vérifier qu'il **n'écoute que sur le port 443** et

que les connexions entrantes sont **strictement restreintes à l'adresse IP unique de votre redirecteur**. Si le serveur C2 reste accessible publiquement depuis Internet, il s'expose aux scanners Internet et risque d'être signalé ou suspendu par le fournisseur de cloud.

- **Filtrage d'entrée du redirecteur** : Sur le pare-feu du redirecteur, auditez les règles d'accès pour vous assurer que le trafic HTTP/HTTPS extérieur n'est accepté que s'il provient des adresses IP légitimes du CDN (par exemple via l'étiquette de service Azure AzureFrontDoor.Backend).
- **Ségrégation du poste opérateur** : Assurez-vous que l'administration du C2 ou les ports de rebond (comme les tunnels SOCKS5) ne sont pas gérés directement depuis le serveur C2 lui-même, mais depuis une machine d'opérateur dédiée dont l'IP est explicitement autorisée dans les règles de pare-feu, évitant ainsi toute attribution en cas de fuite d'adresse IP.

## 2. Audit du comportement du redirecteur et du CDN

- **Vérification du filtrage applicatif (anti-« dumb pipe »)** : Assurez-vous que le redirecteur n'utilise pas de simple transfert aveugle de ports (socat ou simples règles iptables), car ceux-ci transmettent aveuglément les requêtes des scanners vers le C2. Le redirecteur Apache ou Nginx doit filtrer les requêtes à l'aide de règles de réécriture (mod\_rewrite) basées sur des *User-Agents* spécifiques ou des chemins d'URI précis pour bloquer les sondes extérieures.
- **Paramétrage du CDN** : Dans la console de votre CDN (comme Azure CDN), vérifiez impérativement que la **compression est désactivée (Off)** et que les règles de mise en cache sont définies sur « **Bypass caching for query strings** ». Une mise en cache active ou une compression risquerait de corrompre les flux dynamiques entre l'agent et le serveur.

## 3. Audit des profils C2 et de l'obfuscation de l'agent

- **Bannissement des signatures d'usine par défaut** : Vérifiez que vous n'utilisez pas la configuration par défaut du profil C2. Les en-têtes serveur renvoyés doivent être personnalisés pour imiter des technologies web légitimes (comme simuler les en-têtes d'un serveur Microsoft-IIS/10.0).
- **Chiffrement du transport** : Vérifiez que l'ensemble des échanges s'effectue via un protocole chiffré (HTTPS avec TLSv1.2 et certificats SSL valides) pour empêcher toute analyse en clair par les mécanismes de surveillance réseau de l'entreprise.
- **Gigue (Jitter) et caractère aléatoire des délais** : Assurez-vous que l'agent n'utilise pas un intervalle de rappel statique qui trahirait une balise automatisée (*beaconing*). Une variation aléatoire (*jitter*, par exemple 20 %) doit être configurée sur les rappels.
- **Présence d'une date d'expiration (Kill Date)** : Contrôlez que la charge utile intègre une date limite d'arrêt au-delà de laquelle l'agent cesse définitivement toute communication, évitant ainsi de laisser des agents actifs une fois l'exercice terminé.

#### 4. Tests de validation technique (Validation pratique)

- **Test de la chaîne de redirection** : Effectuez un appel curl vers l'URL externe de votre CDN :

curl -I <https://<nom-point-de-terminaison>.azureedge.net>

La réception d'un code de statut **HTTP/2 405** valide que la requête transite de bout en bout à travers le CDN et le redirecteur jusqu'au C2, tout en refusant les requêtes non sollicitées.

- **Test d'étanchéité du tunnel SOCKS / Proxychains** : Si vous utilisez un tunnel pour pivoter sur le réseau interne, validez l'absence de fuite d'adresse IP en comparant une requête directe et une requête relayée :

```
sudo curl ipconfig.io
```

```
sudo proxychains4 curl ipconfig.io
```

La seconde commande doit obligatoirement renvoyer l'adresse IP de la cible compromise et non l'IP de votre serveur C2 ou de votre poste d'attaque [78, Figure 3-21].

#### 5. Audit de la traçabilité et de la journalisation

- **Historisation des opérations** : Vérifiez que votre infrastructure enregistre de manière exhaustive et horodatée chaque tâche, commande, hôte et utilisateur dans sa base de données (comme le font Mythic ou PoshC2). Cette traçabilité est indispensable pour la redevabilité lors de l'élaboration du rapport de mission et la corrélation avec les équipes de défense.

### Quels sont les détails de configuration du profil HTTP de Mythic ?

La configuration du profil **HTTP** de Mythic repose sur deux volets distincts : le paramétrage du service d'écoute sur le serveur C2 et la définition des paramètres de communication lors de la création de la charge utile (*payload*).

#### 1. Configuration du service côté serveur C2

Le profil HTTP communique à l'aide de requêtes GET/POST et prend en charge le chiffrement SSL, des en-têtes personnalisés ainsi que des paramètres de proxy. Sa configuration s'effectue dans l'interface de Mythic sous le menu **Agents & C2** :

- **Intégration des certificats SSL** :
  - Dans la gestion des fichiers du profil (icône de presse-papiers), le contenu des certificats du domaine (obtenus par exemple avec Let's Encrypt) doit être copié directement dans les fichiers fullchain.pem et privkey.pem.
- **Fichier de configuration du service (config.json / Exemple 3-10)** :
  - Pour éviter les signatures par défaut, la configuration du profil est modifiée pour imiter une technologie serveur courante, comme **Microsoft-IIS/10.0**.
  - Les paramètres de l'instance comprennent :

- **port** : Défini sur 443.
- **use\_ssl** : Activé (true).
- **cert\_path** : Chemin pointant vers "fullchain.pem".
- **key\_path** : Chemin pointant vers "privkey.pem".
- **debug** : Positionné sur false.
- **ServerHeaders** : Personnalisation des en-têtes HTTP retournés par le serveur pour simuler un environnement IIS :
  - "Server": "Microsoft-IIS/10.0"
  - "Cache-Control": "max-age=0, no-cache"
  - "Pragma": "no-cache"
  - "Connection": "keep-alive"
  - "Content-Type": "text/html; charset=utf-8"
- **Application des changements** :
  - Le profil doit ensuite être redémarré en cliquant sur **Stop Profile** puis **Start Profile** afin que les modifications prennent effet.

## 2. Paramètres du profil HTTP lors de la création de la charge utile

Lors de la génération d'un agent (comme l'agent Apollo), l'étape **Select C2 Profiles** demande de préciser les options de communication de l'agent pour ce profil HTTP :

- **Callback Host** : L'adresse URL vers laquelle l'agent doit se connecter depuis l'extérieur (le point de contact externe, typiquement le point de terminaison du CDN comme <https://<nom-cdn>.azureedge.net>).
- **Callback Port** : Le port d'écoute externe utilisé pour les rappels (généralement 443).
- **Callback Interval** : L'intervalle de temps fixe (en secondes) entre chaque tentative de connexion de l'agent vers le serveur.
- **Callback Jitter** : Le pourcentage de variation aléatoire appliqué à l'intervalle de rappel afin d'introduire de l'imprévisibilité et de briser les périodicités régulières facilement détectables.

- **Kill Date** : La date limite au-delà de laquelle l'agent cesse définitivement d'émettre des requêtes et de fonctionner.
- **Encryption Type** : La méthode de chiffrement retenue pour protéger le contenu des données échangées.
- **Headers et URI personnalisés** : La personnalisation des en-têtes de messages GET et POST ainsi que des URI pour simuler des requêtes d'applications légitimes plutôt que de conserver les valeurs par défaut.

## Comment configurer les redirections DNS et les enregistrements A ?

La configuration des enregistrements **A** et du routage **DNS** au sein d'une infrastructure Red Team s'articule autour de la dissimulation du serveur backend et de la sécurisation des communications :

### 1. Création initiale des enregistrements A publics (Validation SSL)

- **Pointage public initial** : Lors de la phase de préparation, des enregistrements DNS de type **A** doivent être configurés pour faire pointer les noms de domaine vers les adresses IP publiques respectives du serveur redirecteur et du serveur C2.
- **Prérequis pour le serveur C2** : Avant d'initialiser Mythic, l'enregistrement A du domaine associé doit impérativement pointer vers l'adresse IP de l'instance C2.
- **Génération des certificats** : Cette résolution publique est nécessaire pour valider les défis HTTP émis par Let's Encrypt via Certbot. Durant cette opération, les ports 80 et 443 sont temporairement ouverts au trafic externe avant d'être à nouveau restreints.

### 2. Suppression de l'enregistrement A public et redirection locale (/etc/hosts)

- **Suppression de l'enregistrement A du C2** : Dès que les certificats SSL sont générés et déplacés dans le répertoire du framework, l'enregistrement A public pointant vers le serveur C2 est **supprimé**

de la zone DNS publique afin de ne pas exposer le serveur principal aux scanners Internet.

- **Résolution statique locale** : Pour que le redirecteur sache vers où transférer les flux sans recourir à un DNS public, l'adresse IP et le domaine du C2 sont ajoutés localement dans son fichier `/etc/hosts` :  
echo <C2ServerIP> <C2Domain> >> /etc/hosts
- **Relais par le VirtualHost** : Le proxy inverse Apache relaie les requêtes entrantes vers cette destination locale chiffrée (ProxyPass / <https://exampledomain.com/>).
- **Verrouillage réseau** : Le pare-feu du serveur C2 est ensuite restreint pour n'autoriser les connexions entrantes sur le port 443 que depuis l'adresse IP unique du redirecteur.

### 3. Résolution DNS via le réseau CDN

- **Origine du CDN** : Dans la configuration du CDN (comme le CDN classique d'Azure), le domaine de votre redirecteur est configuré dans les champs de l'hôte d'origine (*Origin hostname*) et de l'en-tête de l'hôte (*Origin host header*).
- **Hôte de rappel (Callback Host)** : Lors de la génération de la charge utile (payload), l'opérateur configure l'hôte de rappel pour cibler l'adresse fournie par le CDN (ex. : <https://<nom-point-de-terminaison>.azureedge.net>).
- **Bénéfice de réputation** : Les requêtes DNS initiées depuis le poste cible résolvent ainsi les adresses IP du réseau de distribution du fournisseur cloud plutôt que vos serveurs réels, ce qui permet de se fondre dans le trafic légitime.

### 4. Utilisation du canal DNS pour les flux de commande et contrôle

- **Profil DNS (Requêtes TXT)** : Au-delà de la résolution de noms web, des profils C2 comme le profil dns de Mythic utilisent un serveur DNS personnalisé configuré pour écouter et répondre aux requêtes de type **TXT** transmises par l'agent.

- **DNS over HTTPS (DoH)** : Pour éviter la détection ou l'interception sur les réseaux surveillés, les requêtes et réponses DNS peuvent transiter via des connexions chiffrées HTTPS vers des résolveurs publics réputés (comme [cloudflare-dns.com](https://cloudflare-dns.com)).

## **Comment tester et valider la résolution de ces flux à l'aide d'outils comme dig ou curl à chaque étape du déploiement**

Pour valider méthodiquement chaque maillon de l'infrastructure et garantir à la fois son bon fonctionnement technique et sa sécurité opérationnelle (OPSEC), les tests avec **dig** et **curl** s'organisent en cinq étapes successives :

Étape 1 : Validation de la résolution initiale et des certificats SSL

Avant de restreindre les accès pare-feu, il faut s'assurer que les enregistrements DNS publics de type **A** pointent correctement vers vos serveurs pour permettre l'émission des certificats Let's Encrypt.

- **Test de résolution DNS avec dig** :

# Vérifier que le domaine résout bien l'adresse IP publique de votre instance (redirecteur ou C2)  
dig +short exampledomain.com A

*Résultat attendu* : L'adresse IP publique de la machine cible s'affiche immédiatement. Vous pouvez forcer l'interrogation d'un résolveur public (dig @8.8.8.8 exampledomain.com) pour vous assurer que la propagation DNS est effective.

- **Test d'accessibilité HTTP (port 80) avec curl** :

curl -I <http://exampledomain.com>

*Résultat attendu* : Une réponse HTTP confirmant que le port 80 est bien ouvert temporairement vers l'extérieur pour valider les défis de Certbot.

Étape 2 : Validation de la redirection locale (/etc/hosts) et du proxy Apache

Dès que les certificats sont générés, l'adresse IP et le domaine du C2 sont ajoutés localement dans /etc/hosts sur le serveur redirecteur :

- **Test de résolution locale sur le redirecteur :**

# Vérifier la résolution locale interne

getent hosts <C2Domain>

*Résultat attendu :* La ligne configurée dans /etc/hosts est renvoyée avec l'adresse IP interne ou directe du serveur C2.

- **Test de liaison HTTPS locale entre le redirecteur et le C2 :** Depuis le redirecteur, validez la connexion vers le backend C2 :

curl -k -I <https://<C2Domain>/>

*Résultat attendu :* Une négociation TLS réussie confirmant que le port 443 du serveur C2 est joignable et accepte la connexion chiffrée depuis l'IP du redirecteur.

Étape 3 : Contrôle OPSEC de la suppression de l'enregistrement A public du C2

Pour éviter que des scanners Internet ne découvrent et ne signalent votre serveur backend C2, son enregistrement public doit être supprimé.

- **Vérification de la disparition de l'enregistrement avec dig :**

dig +short <C2Domain> @8.8.8.8

*Résultat attendu :* **Aucune adresse IP** ne doit être retournée (réponse vide ou code NXDOMAIN). Cela confirme que le serveur C2 n'est plus directement repérable sur les zones DNS publiques.

Étape 4 : Validation du point de terminaison CDN et du flux de bout en bout  
Cette étape confirme que le trafic traverse l'ensemble de la chaîne : **Point de terminaison CDN → Redirecteur Apache → Serveur C2 backend.**

- **Test de résolution DNS du point de terminaison CDN avec dig :**

dig <https://<nom-point-de-terminaison>.azureedge.net>

*Résultat attendu* : La commande résout les adresses IP Anycast appartenant aux plages réseau légitimes de Microsoft/Azure CDN.

- **Test du flux applicatif de bout en bout avec curl** : Sollicitez l'en-tête du point de terminaison CDN :

curl -I <https://<nom-point-de-terminaison>.azureedge.net>

*Résultat attendu* :

HTTP/2 405

date: Sun, 23 Jun 2024 20:20:25 GMT

content-type: text/plain

content-length: 22

x-azure-ref: <snipped>

x-fd-int-roxy-purgeid: 0

x-cache: TCP\_MISS

- **Interprétation du code 405** : Le code de statut **HTTP/2 405 (Method Not Allowed)** confirme que la requête a traversé le CDN, a été relayée par Apache et a bien été traitée par le profil HTTP de Mythic sur le serveur backend (qui rejette une requête générique sans URI d'agent valide).
- Les en-têtes x-azure-ref et x-cache: TCP\_MISS valident que le CDN a correctement transféré la requête sans la mettre en cache.
- **Test du verrouillage du redirecteur avec curl** : Tentez d'accéder directement à l'adresse IP publique du redirecteur depuis votre propre machine :

curl -I <https://<IP-du-redirecteur>> --connect-timeout 5

*Résultat attendu* : La connexion doit échouer par dépassement de délai (*timeout*), confirmant que le groupe de sécurité (NSG) bloque tout accès n'émanant pas de l'étiquette de service AzureFrontDoor.Backend.

Étape 5 : Validation de l'étanchéité du tunnel SOCKS5

Lorsque l'agent est déployé et que le proxy SOCKS5 est activé dans Mythic, il faut valider l'absence totale de fuite d'adresse IP lors du pivotement :

- **Test comparatif d'adresse IP sortante :**

# 1. Requête directe (affiche l'adresse IP de votre serveur ou poste d'attaque)

```
sudo curl ipconfig.io
```

# 2. Requête relayée via le tunnel SOCKS5

```
sudo proxychains4 curl ipconfig.io
```

*Résultat attendu :* La sortie de proxychains4 affiche Strict chain ... 127.0.0.1:7006 ... ipconfig.io:80 ... OK et l'adresse IP renvoyée correspond impérativement à **l'adresse IP publique de la machine cible compromise**, confirmant l'absence de fuite d'identité de votre infrastructure.

## Comment cartographier et rapporter les actions dans Mythic ?

Dans **Mythic**, la cartographie des activités sur le cadre **MITRE ATT&CK** et la génération des rapports s'effectuent directement depuis l'interface web d'administration.

### 1. Cartographier les actions sur la matrice MITRE ATT&CK

- **Accès à la matrice :** Vous pouvez accéder au tableau de correspondance en cliquant sur l'icône « **Colonne** » dans la barre de navigation supérieure ou en vous rendant manuellement à l'URL <https://127.0.0.1:7443/new/mitre>.
- **Options de chargement (Menu « Actions ») :** En haut à droite, l'onglet déroulant **Actions** propose deux modes principaux d'affichage :
  - **Fetch All Commands Mapped to MITRE :** Cette option liste l'ensemble des correspondances théoriques possibles associées aux commandes intégrées dans l'agent utilisé, même

si celles-ci n'ont pas encore été exécutées (par exemple, la commande ifconfig de l'agent Apollo mappée vers la sous-technique T1590.005 – Gather Victim Network Information: IP Addresses).

- **Fetch All Issued Tasks Mapped to MITRE** : Cette option extrait uniquement les techniques correspondant aux **tâches réellement émises et exécutées** durant la mission. Les techniques correspondantes s'affichent alors **en rouge** dans la matrice (comme la technique T1029 – Scheduled Transfer surlignée lors de l'utilisation des commandes d'ajustement de délai sleep).

## 2. Configurer et générer le rapport opérationnel

- **Accès au module de rapport** : Cliquez sur l'icône de **drapeau (flag)** dans la barre d'outils supérieure ou accédez directement à <https://127.0.0.1:7443/new/reporting>.
- **Paramètres de configuration du rapport** :
  - **Format de sortie (Output Format)** : Sélectionnez le format désiré, soit **HTML**, soit **JSON**.
  - **Options de couverture MITRE** : Activez les commutateurs « **Include MITRE ATT&CK Coverage Per Task** » (pour associer chaque tâche spécifique à son identifiant ATT&CK) et « **Include MITRE ATT&CK Coverage Overview** » (pour intégrer une synthèse globale de la couverture).
  - **Filtres et exclusions optionnels** : Si vous souhaitez nettoyer ou cibler les données, vous pouvez exclure des utilisateurs spécifiques (*Exclude Users*), des noms d'hôtes (*Exclude Hosts*) ou des identifiants précis (*Exclude IDs / Callbacks*).
- **Génération et téléchargement** : Cliquez sur le bouton vert **GENERATE**, puis sur « **Download here** » lorsque la boîte de dialogue indique que le fichier est prêt.

## 3. Exploitation des informations du rapport généré

Le rapport consolidé (notamment au format HTML) regroupe des métriques précises pour la restitution de mission :

- **Opérateurs assignés (Assigned Operators)** : Liste des comptes opérateurs ayant participé à la mission ainsi que leur rôle (ex. : lead).
- **Métriques opérationnelles (Operation Metrics)** : Tableau quantifiant le nombre d'utilisateurs et d'hôtes compromis, les domaines accédés, le total des callbacks, les callbacks avec privilèges élevés (*High Integrity*) et le nombre total de tâches émises.
- **Détails chronologiques des callbacks et tâches** : Historique horodaté de chaque connexion (utilisateur, PID, adresse IP, intégrité, canal C2 utilisé) et association de chaque commande exécutée avec son identifiant MITRE ATT&CK (ex. : T1029 pour sleep 0 0).
- **Synthèse « MITRE ATT&CK Overview »** : Tableau récapitulatif listant chaque identifiant technique ATT&CK, son nom officiel et le nombre d'occurrences enregistrées durant l'évaluation