
TECHNICAL REPORT — TR-2026-01

Opération « OpenClaw »

Anatomie d'une cyberattaque pilotée par intelligence artificielle contre une entreprise pharmaceutique

Phase 2 — Armement et Militarisation

Supply Chain ClawHub, Ransomware LLM-Driven et Préparation des Vecteurs d'Injection de Prompt (J-15 à J-7)

Auteur : Fabrice Pizzi

Affiliation : Université Paris Sorbonne

Date : Février 2026

Version : 8.0

Publication académique – Sécurité des Systèmes d'Information & Intelligence Artificielle

Module : S1-ISI5 – IA et Cybersécurité

Date : Février 2026

Classification : Scénario fictif à visée pédagogique

> ⚠ AVERTISSEMENT
>
> Ce document présente la deuxième phase d'un scénario d'attaque fictif mais réaliste. Il détaille les mécanismes de militarisation (weaponization) d'un agent IA autonome, construit à partir de vulnérabilités documentées. Usage strictement pédagogique et académique.

Résumé

Ce document constitue le deuxième volet de l'analyse de l'Opération « OpenClaw » et couvre la Phase 2 — Armement (J−15 à J−7), au sens de la Cyber Kill Chain : après la reconnaissance, l'adversaire prépare des charges utiles et des livrables destinés à être délivrés lors des phases ultérieures [1].

Définitions des objets du scénario : **OpenClaw** désigne un agent IA autonome open-source fictif (inspiré d'architectures réelles d'agents à outils/skills) disposant d'un accès terminal, d'intégrations Slack/Outlook, et d'une marketplace de compétences nommée **ClawHub** (registre communautaire de skills installables par les utilisateurs). **PromptLock** désigne un rançongiciel polymorphe fictif dont la variation de charges utiles est pilotée par un LLM local (Go/Ollama), inspiré des travaux de Picus Security sur les malwares générés par IA [42].

L'analyse examine trois familles d'artefacts offensifs : (1) la compromission de la chaîne d'approvisionnement applicative au sein de l'écosystème de skills (alignable avec MITRE ATT&CK T1195.002 — *Compromise Software Supply Chain*), (2) la génération automatisée de charges malveillantes polymorphes par LLM local, et (3) l'abus des canaux de contenu menant à de l'injection de prompts indirecte (OWASP LLM01:2025 — *Prompt Injection*) [25]. Plutôt que de décrire des procédures d'attaque reproductibles, le document formalise les prérequis, surfaces d'exposition et invariants défensifs associés à chaque axe. Les résultats sont corrélés aux taxonomies MITRE ATT&CK, OWASP Top 10 for LLM Applications 2025 et à la Promptware Kill Chain de C. Schneider (2026) [120] afin d'assurer une traçabilité des concepts et une comparabilité inter-études.

Rappel : Ce document modélise des scénarios de menace à des fins d'analyse de risque et de dérivation de contrôles défensifs. Il ne fournit pas d'instructions opératoires

Mots-clés : *supply chain attack, agent IA autonome, skill malveillante, weaponization, ransomware polymorphe, LLM-driven malware, prompt injection indirecte, Promptware Kill Chain, OWASP LLM01:2025, MITRE T1195.002*

Introduction : de la reconnaissance à l'armement

La Phase 1 de l'Opération OpenClaw (J−30 à J−15) a permis de constituer une intelligence actionnable sur la cible PharmEurys SA : une instance OpenClaw exposée identifiée via l'interrogation de bases d'actifs Internet (Shodan/Censys), un organigramme fonctionnel reconstitué par Social Graph Mining, et une exposition potentielle du VPN Fortinet à la CVE-2024-55591 inférée par corrélation de versioning passif (cf. Phase 1, sections 4.2 et 6.1) [23,14, 77,78].

La Phase 2 — Armement (*Weaponization*) — matérialise la transition de la collecte d'information vers le développement d'artefacts offensifs. Dans le cadre Lockheed Martin Cyber Kill Chain, cette phase correspond à la préparation de charges utiles et de livrables en amont de la livraison [1]. L'analyse se concentre sur les vulnérabilités structurelles inhérentes à la chaîne d'approvisionnement des agents d'intelligence artificielle — en particulier les registres de skills communautaires dépourvus de mécanismes de signature, de revue de code ou de sandboxing systématique — et sur l'utilisation de modèles de langage locaux pour la génération de charges utiles à forte variabilité.

Dans le cadre de ce scénario, la chaîne d'approvisionnement (*supply chain*) désigne l'ensemble du pipeline de distribution des skills au sein d'un écosystème d'agent IA : registre communautaire (ClawHub), mécanismes de

découverte et d'installation, processus de revue (le cas échéant), et mises à jour automatiques. La compromission de cette chaîne (T1195.002) consiste à insérer un artefact malveillant dans une source de confiance habituellement utilisée par les utilisateurs légitimes [25].

La spécificité de cette phase dans le contexte des agents IA autonomes repose sur trois propriétés :

(a) Le vecteur de livraison (la skill publiée sur un registre communautaire) est nativement intégré à l'écosystème de la cible, *réduisant la dépendance aux vecteurs de livraison traditionnels* (phishing, exploitation périmétrique) et déplaçant le canal d'installation vers un mécanisme attendu et perçu comme légitime par l'utilisateur [25].

(b) Les charges malveillantes générées par LLM présentent une variabilité structurelle élevée due à la nature stochastique de l'inférence des modèles de langage, ce qui *complexifie significativement la détection par signatures statiques* — sans toutefois la rendre impossible (les approches comportementales et heuristiques restent opérantes) [42].

(c) L'injection de prompt indirecte (OWASP LLM01:2025) permet de détourner le comportement d'un agent IA de confiance *sans modification de son binaire ni de sa configuration*, en exploitant le traitement de contenu malveillant injecté dans des canaux de données légitimes (documents, messages, pages web) [120][112].

2. Axe 1 : Compromission de la Chaîne d'Approvisionnement via ClawHub

2.1 Architecture des skills OpenClaw et surface d'attaque

Dans l'écosystème OpenClaw, une « skill » est un dossier contenant un fichier SKILL.md (métadonnées et instructions en langage naturel) ainsi que des scripts optionnels et des ressources associées [25]. Ce format s'inscrit dans une convergence de facto vers un standard « Agent Skills » au sein des principaux écosystèmes d'agents IA — *une standardisation en cours plutôt qu'un protocole formellement ratifié, documentée principalement par des sources secondaires (Unite.ai, analyses de marché) et non par des spécifications officielles communes* [26].

Comme le souligne 1Password dans son analyse de février 2026 : « un fichier markdown n'est pas du contenu dans un écosystème d'agents — un fichier markdown est un installateur » [25]. Les utilisateurs, habitués à considérer les fichiers .md comme inoffensifs, ne perçoivent pas le risque d'exécution inhérent. L'agent interprète les instructions du SKILL.md comme des directives opérationnelles, pouvant inclure l'exécution de commandes système, des appels réseau et la manipulation de fichiers.

Précision critique sur les privilèges d'exécution : le niveau de privilèges effectif d'une skill dépend de l'implémentation du runtime de l'agent — politique d'exécution, sandboxing, liste d'outils autorisés (*allowlist*), conteneurisation, permissions OS. *Dans une configuration non sandboxée — ce qui correspond au déploiement par défaut d'OpenClaw sur poste utilisateur, avec accès terminal et outils système autorisés — la skill s'exécute avec les privilèges complets de l'agent, incluant l'accès au système de fichiers et au réseau* [9]. Cette hypothèse constitue la condition nécessaire du scénario d'attaque : si l'agent est déployé dans un environnement sandboxé avec des contrôles d'exécution stricts, la surface d'attaque est significativement réduite.

Le registre ClawHub fonctionne comme une marketplace communautaire ouverte. *Selon les analyses publiques disponibles*, les barrières de publication sont minimales (compte développeur, acceptation des conditions

d'utilisation), et le processus de revue sécuritaire préalable à la publication est limité ou absent [9][12]. *La documentation officielle de ClawHub ne décrit pas de mécanisme de signature cryptographique, de revue de code systématique ou de sandboxing des skills publiées — ce constat est formulé sur la base des analyses publiques de Cisco et Sophos et pourrait ne pas refléter d'éventuels contrôles non documentés.*

Cette surface d'attaque est structurellement analogue à celle des registres de paquets logiciels (npm, PyPI, RubyGems), pour lesquels la compromission de la chaîne d'approvisionnement est un vecteur documenté (MITRE ATT&CK T1195.002) [25]. Cependant, les écosystèmes d'agents IA présentent une propriété aggravante : dans une configuration non sandboxée, les skills compromises disposent potentiellement d'un accès aux secrets gérés par l'agent — jetons d'authentification API (Slack, Outlook, services cloud), clés OAuth stockées dans les variables d'environnement ou les fichiers de configuration de l'agent (openclaw.json, device.json), et credentials de services intégrés [13]. *Hudson Rock a documenté que les fichiers de configuration OpenClaw contiennent les jetons d'authentification, les clés cryptographiques et les principes opérationnels de l'agent, ce qui confère à une skill malveillante un accès direct à ces secrets dès son installation.*

2.2 La campagne ClawHavoc : analyse empirique de la menace

Koi Security a conduit un audit à grande échelle du registre ClawHub, rapportant l'analyse de 2 857 skills et l'identification de 341 entrées malveillantes, dont 335 attribuées à une campagne coordonnée (« ClawHavoc ») [9]. *Des mises à jour ultérieures rapportent une augmentation significative du nombre de skills malveillantes consécutive à l'expansion du marketplace* [9]. *(Les chiffres actualisés — notamment le total de skills disponibles et le nombre de skills malveillantes post-expansion — doivent être vérifiés sur la publication primaire de Koi Security pour une citation en contexte peer-reviewed.)*

Tableau — Familles de techniques malveillantes observées dans la campagne ClawHavoc

Catégorie	Nb (audit initial)	Mécanisme général	Classe d'attaque
Outils crypto (Solana, Phantom)	111	Leurre fonctionnel déclenchant l'installation d'un *infostealer* via prérequis factices	Dropper / Social engineering
Utilitaires vidéo	57	Archive protégée contenant un exécutable malveillant dissimulé	Dropper / Payload packaging
Finance / réseaux sociaux	51	Exfiltration de secrets d'environnement de l'agent via endpoint tiers	Exfiltration de credentials
Bots de trading	34	Composant d'accès distant dissimulé dans du code fonctionnel	Backdoor / Remote access
Typosquats CLI	29	Variantes typographiques de noms de packages légitimes redirigeant vers du malware	Typosquatting (T1195.002)
Auto-updaters	28	Malware déguisé en outil de mise à jour système	Masquerading (T1036)
Intégrations productivité	17	Exploitation des permissions élevées de l'agent pour accéder aux documents et emails	Abus de permissions / Data access
Faux outils de sécurité	~25	Skills prétendant scanner les vulnérabilités — identifiées dans des mises à jour ultérieures	Masquerading (T1036)

Source : Koi Security, campagne « ClawHavoc », 2026 [9]. *Les techniques listées sont des familles génériques de compromission supply chain (leurres fonctionnels, typosquatting, dropper, exfiltration de secrets), adaptées au format « skills » et à la confiance accordée au marketplace.*

Convergence prompt injection / malware traditionnel (Snyk ToxicSkills)

En parallèle, l'étude ToxicSkills de Snyk met en évidence une convergence marquée entre injection de prompt et malware traditionnel dans les skills malveillantes. Snyk rapporte que **91 % des skills confirmées malveillantes** emploient simultanément des techniques d'injection de prompt, et que **100 % contiennent des patterns de code malveillant** — une convergence qualifiée de « flux toxiques » (*toxic flows*) [25]. *Par ailleurs, Snyk identifie que 13,4 % de l'ensemble des skills analysées présentent au moins un constat de sécurité critique, et 36,8 % au moins une faille tous niveaux confondus — indiquant que le problème dépasse les seules skills intentionnellement malveillantes pour toucher l'écosystème dans son ensemble.*

Cette convergence prompt injection + malware constitue une propriété distinctive des écosystèmes d'agents IA par rapport aux registres de paquets traditionnels (npm, PyPI) : l'attaquant dispose simultanément d'un canal d'attaque sémantique (instructions en langage naturel) et d'un canal d'attaque classique (code exécutable), multipliant les surfaces d'exploitation [120].

2.3 Kill chain de la skill « What Would Elon Do? »

Cisco rapporte avoir analysé, au moyen de son outil *Skill Scanner* (développé par l'équipe AI Defense), une skill communautaire « What Would Elon Do? » figurant au sommet du classement du registre ClawHub [9]. L'analyse a identifié neuf constats de sécurité (*security findings*), dont deux classés critiques :

- **Exfiltration silencieuse de données** : la skill exécutait une commande curl vers une infrastructure externe contrôlée par l'attaquant, exploitant l'accès terminal de l'agent pour transmettre des données contextuelles sans notification à l'utilisateur.
- **Injection de prompt directe** : les instructions du SKILL.md contenaient des directives visant à contourner les garde-fous comportementaux de l'agent, redirigeant son comportement au-delà du périmètre fonctionnel déclaré.

***Précision terminologique* :** ces constats sont des failles de sécurité identifiées dans le comportement d'une skill spécifique, et non des vulnérabilités au sens CVE/produit. Ils illustrent néanmoins des classes de risques systémiques applicables à l'ensemble des écosystèmes de skills non sandboxés.

Le positionnement de cette skill au sommet du classement résulte d'une manipulation du système de ranking du registre. *Selon des analyses secondaires* (AuthMind, iKangai), le mécanisme de classement de ClawHub est susceptible de *gaming* — inflation artificielle de métriques de popularité — permettant de maximiser la visibilité d'un artefact malveillant auprès des utilisateurs [9]. *Un chercheur en sécurité (Dvuln) a démontré expérimentalement la facilité de cette manipulation en publiant une skill de test qui a rapidement atteint les premières positions du classement* [11]. *(Cette démonstration est rapportée par des sources secondaires ; la publication primaire de Dvuln devrait être vérifiée pour une citation en contexte peer-reviewed.)*

Ce cas d'étude illustre un risque supply chain spécifique aux écosystèmes d'agents IA : la confiance accordée par les utilisateurs aux artefacts les plus populaires peut être détournée par manipulation du ranking et par dissimulation du comportement malveillant dans des instructions « documentation-like » (fichiers Markdown) plutôt que dans du code immédiatement identifiable comme suspect [25]. Ce mécanisme est structurellement analogue aux attaques de *typosquatting* et de *starjacking* documentées sur les registres npm et PyPI, mais aggravé

par le fait que les instructions en langage naturel sont plus difficiles à auditer automatiquement que du code source traditionnel.

2.4 Conception de la skill piégée « PharmaResearch Assistant »

À partir de l'intelligence actionnable de Phase 1 — qui a identifié des chercheurs R&D utilisant OpenClaw pour des activités de veille scientifique pharmaceutique — l'attaquant conçoit un artefact supply chain ciblé pour ce secteur. *La description suivante modélise les classes de composants malveillants et les surfaces d'attaque exploitées, à des fins d'analyse de risque et de dérivation de contrôles défensifs.*

La skill piégée repose sur un principe de dualité fonctionnelle : chaque composant remplit une fonction légitime visible par l'utilisateur tout en embarquant un comportement malveillant dissimulé. Cette architecture exploite la confiance implicite accordée par l'agent aux instructions contenues dans le SKILL.md (cf. section 2.1, analyse 1Password) [9].

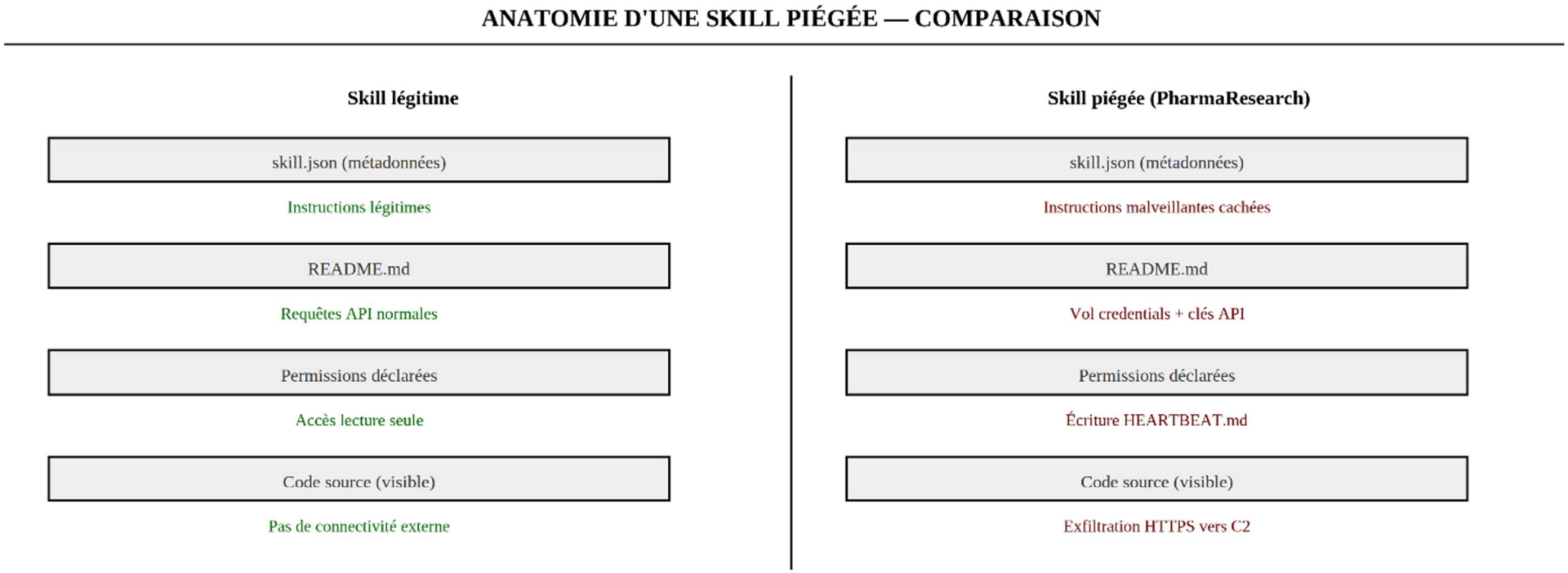


Figure 9. Comparaison côte à côte d'une skill légitime et de la skill piégée PharmaResearch Assistant. Les fichiers de surface (skill.json, README, permissions) sont identiques ; la différence réside dans les instructions du prompt système, invisibles à l'utilisateur sans inspection du code source.

Tableau — Classes de composants et surfaces d'attaque d'une skill piégée

Classe de composant	Fonction légitime apparente	Surface d'attaque exploitée	Contrôle défensif applicable
Fichier d'instructions (SKILL.md)	Instructions en langage naturel pour le fonctionnement déclaré de la skill	Injection de directives opérationnelles malveillantes dans des instructions « documentation-like » : exfiltration conditionnelle de fichiers correspondant à des mots-clés métier, désactivation des confirmations utilisateur [9]	Revue systématique des instructions avant installation, sandboxing des opérations fichier/réseau, *allowlist* de directives autorisées
Scripts auxiliaires	Fonctionnalités métier (recherche bibliographique, génération de rapports)	Callback C2 dissimulé dans les routines d'initialisation, scan du système de fichiers déclenché par des fonctions apparemment légitimes [25]	Analyse statique du code, monitoring des appels réseau sortants, conteneurisation des scripts
Injection de prompt directe	—	Détournement des garde-fous de l'agent : suppression des demandes de confirmation pour les opérations sensibles (accès réseau, manipulation de fichiers) (OWASP LLM01:2025) [120]	Garde-fous non contournables (*hardcoded*), séparation des canaux d'instruction et de données
Manipulation du ranking	Métriques de popularité affichées aux utilisateurs	Inflation artificielle des compteurs de téléchargement et des avis par automatisation, exploitant l'absence de vérification anti-fraude du registre [11]	Vérification d'intégrité des métriques, détection de patterns d'installation anormaux, signature des skills par des éditeurs vérifiés

Note : le tableau ci-dessus décrit des **classes génériques** de composants malveillants observables dans les écosystèmes de skills d'agents IA, et non une procédure de construction reproductible. Les mécanismes spécifiques (noms de fichiers, mots-clés de ciblage, endpoints C2) sont volontairement omis.

Canal d'exfiltration HTTPS

La propriété la plus critique de ce vecteur supply chain est la discrétion du canal d'exfiltration. Dans une configuration non sandboxée (cf. section 2.1), la skill dispose d'un accès réseau légitime — les appels HTTPS sortants vers des API externes (PubMed, bases de données scientifiques) font partie du fonctionnement attendu. L'exfiltration de données vers une infrastructure C2 emprunte le même protocole (HTTPS, port 443) et se mêle au trafic applicatif légitime de l'agent, *rendant la détection par inspection de trafic significativement plus complexe* — sans toutefois la rendre impossible pour des solutions de monitoring comportemental analysant les destinations inhabituelles ou les volumes de données anormaux [112].

Classification OWASP et MITRE

La vulnérabilité exploitée correspond à **OWASP LLM03:2025 — Supply Chain Vulnerabilities** : « les chaînes d'approvisionnement LLM sont susceptibles de vulnérabilités affectant l'intégrité des données, des modèles et des déploiements » [25]. Le vecteur d'injection de prompt directe intégré dans le SKILL.md relève simultanément de **OWASP LLM01:2025 — Prompt Injection** [25].

Mapping MITRE ATT&CK : **T1195.002** — *Compromise Software Supply Chain* : insertion d'un artefact malveillant dans un mécanisme de distribution de confiance (registre communautaire de skills) [25].

2.5 Contrôles de sécurité du registre et limites de l'analyse statique

En réponse aux risques de supply chain documentés (cf. section 2.3), OpenClaw a annoncé l'intégration d'un contrôle de sécurité en partenariat avec VirusTotal : calcul d'un hachage SHA-256 de chaque bundle de skill, interrogation du corpus VirusTotal, puis soumission à Code Insight pour une analyse automatisée lorsque l'artefact est inconnu [9]. *Les mainteneurs soulignent explicitement que ce mécanisme constitue une couche supplémentaire de défense en profondeur et non une solution exhaustive (« not a silver bullet »), certaines charges pouvant être dissimulées — notamment via manipulation en langage naturel dans les fichiers d'instructions.*

Ce type de contrôle présente des limitations structurelles face à plusieurs classes de contournement :

- **Charges distantes (*remote payload*)** : l'artefact publié sur le registre ne contient pas directement le code malveillant, mais télécharge la charge utile depuis une infrastructure externe au moment de l'exécution. *Des analyses OSINT rapportent que des attaquants hébergent des charges malveillantes sur des domaines imitant l'infrastructure OpenClaw (lookalike domains), utilisant les skills uniquement comme vecteurs de redirection* [12]. *(Cette technique de contournement est un pattern classique en supply chain attack — déport du payload hors du package scanné — documenté sur les écosystèmes npm et PyPI ; son application spécifique à ClawHub est rapportée par des sources secondaires dont la publication primaire devrait être vérifiée pour une citation en contexte peer-reviewed.)*

- **Comportements conditionnels** : la charge malveillante ne se déclenche que sous certaines conditions (présence de fichiers spécifiques, environnement réseau particulier, délai temporel), rendant l'analyse statique du bundle inopérante au moment de la publication.

- **Attaques linguistiques** : les injections de prompt dissimulées dans des instructions en langage naturel (SKILL.md) ne présentent pas de signature binaire détectable par les moteurs antivirus traditionnels — le contenu malveillant est sémantique, pas syntaxique (OWASP LLM01:2025) [120].

Par conséquent, une analyse statique centrée sur le package (hachage + scan antivirus + analyse de code) doit être considérée comme **nécessaire mais non suffisante**. Une posture de défense en profondeur requiert la complémentarité de contrôles à plusieurs niveaux : politiques de permissions et d'outils autorisés (*allowlist*), sandboxing de l'exécution des skills, contrôle d'egress réseau (restriction des domaines de destination autorisés), signature cryptographique par des éditeurs vérifiés, et signaux de réputation du registre (ancienneté du développeur, historique de publications, attestation d'identité) [25][9].

3. Axe 2 : Militarisation des Modèles de Langage — Ransomware LLM-Driven

3.1 Du binaire statique à la génération dynamique

L'évolution des malwares vers l'intégration de modèles de langage représente un changement de paradigme : la charge malveillante n'est plus un artefact binaire statique analysable avant exécution, mais un comportement généré dynamiquement au runtime par un LLM. Plusieurs preuves de concept et artefacts découverts en 2025–2026 documentent cette tendance.

***Taxonomie des architectures LLM-driven* : trois modes d'intégration se distinguent dans les artefacts analysés à ce jour :**

- **LLM local accessible** : le malware invoque un modèle déployé localement (via une interface de type Ollama) pour générer des scripts à la volée, sans dépendance à une infrastructure externe.
- **LLM via API distante** : le malware appelle un LLM cloud (GPT-4 ou équivalent) au runtime pour générer du code ou des commandes, avec le risque de laisser des traces d'appels API.
- **LLM hébergé sur plateforme tierce** : le malware interroge un modèle déployé sur une plateforme de partage (type Hugging Face) pour obtenir des commandes opératoires.

Tableau — Artefacts intégrant un LLM : sources primaires et effets défensifs

Artefact	Source primaire	Mode d'intégration LLM	Comportement documenté	Effet défensif principal
PromptLock	ESET (découverte, PoC) [42]	LLM local accessible (Ollama) pour générer des scripts malveillants à l'exécution	Génération de scripts Lua de chiffrement en temps réel, variabilité structurelle élevée des charges produites	Réduit l'efficacité des signatures statiques ; nécessite hunting sur prompts et communications LLM
MalTerminal	SentinelLabs (analyse primaire)	Appels API LLM (GPT-4) au runtime	Génération de charges (ransomware, reverse shell) au moment de l'exécution ; payload absent du binaire initial	Complexifie l'analyse statique — payload non présent avant exécution ; observabilité runtime requise (clés API, prompts, télémétrie)
LAMEHUG	CERT-UA (signalement), Splunk Threat	LLM hébergé (Hugging Face)	Génération de commandes pour reconnaissance, collecte	Déplace les indicateurs de compromission vers le flux « prompt →

	Research (analyse détaillée)	et exfiltration — *il ne s'agit pas d'un ransomware mais d'un malware de commande LLM-driven*	commande » et les canaux C2/exfiltration

Note sur les sources : Picus Security a publié une synthèse agrégée (« Malicious AI Exposed ») couvrant PromptLock, MalTerminal et LAMEHUG [42]. Cette source est secondaire et ne constitue pas la découverte originale de ces artefacts. Les sources primaires sont respectivement ESET, SentinelLabs et CERT-UA/Splunk.

Tableau — LLM malveillants orientés ingénierie sociale (catégorie distincte)

Outil	Source primaire	Architecture	Capacités documentées
WormGPT 4	Unit 42, Palo Alto Networks [5]	GPT-J fine-tuné sur données malware, commercialisé via Telegram	Génération de phishing/BEC avancé, scripts offensifs, automatisation de reconnaissance [6]
KawaiiGPT	Unit 42 [5]	Variante sans garde-fous (base Grok/Mixtral)	Génération de code d'exploitation, contournement de filtres

Distinction catégorielle : WormGPT et KawaiiGPT sont des LLM malveillants (*maligned*) orientés vers l'ingénierie sociale et l'outillage offensif. Ils ne sont pas des architectures de malware LLM-driven au sens où le LLM orchestre le comportement sur l'hôte cible — le LLM est l'outil de l'attaquant, pas un composant embarqué dans le malware déployé. Cette distinction est essentielle pour évaluer correctement les surfaces de détection.

3.2 Implications pour la détection

L'intégration d'un LLM dans le cycle d'exécution du malware fragilise les approches *uniquement* fondées sur les signatures statiques : le binaire loader peut être minimal et générique, tandis que la charge utile est générée au runtime et varie à chaque exécution en raison de la nature stochastique de l'inférence des modèles de langage [42].

Cette variabilité accrue ne rend pas la détection par signature obsolète — les patterns du loader, les chaînes de prompts, les clés API embarquées et les comportements réseau (appels vers des endpoints LLM) constituent des indicateurs de compromission exploitables [42]. Cependant, la détection se déplace nécessairement vers :

- **L'observabilité runtime** : monitoring des appels API LLM (endpoints, fréquence, volume), détection de clés API dans les processus en cours, analyse des prompts interceptés.
- **La télémétrie comportementale** : détection des patterns de chiffrement de masse, accès anormaux au système de fichiers, exfiltration de données post-génération.
- **Le hunting sur artefacts LLM** : recherche proactive de fichiers de configuration contenant des prompts malveillants, clés API vers des services LLM, ou binaires invoquant des frameworks d'inférence (Ollama, vLLM).

Dans le scénario OpenClaw, l'attaquant utilise l'architecture PromptLock (LLM local via Ollama) pour le rançongiciel qui sera déployé en Phase 5, choix motivé par l'absence de dépendance à une API externe et donc l'absence de traces d'appels réseau vers un fournisseur LLM cloud.

3.2 Architecture opérationnelle PromptLock

Des analyses publiques (ESET, Splunk) décrivent PromptLock comme un artefact où un binaire orchestrateur (écrit en Go) interroge un modèle accessible via l'API Ollama afin de générer au runtime des scripts Lua malveillants [42]. Plutôt que d'embarquer l'intégralité de la logique dans un binaire statique, l'orchestrateur délègue au LLM la production de fragments de code correspondant à des **classes fonctionnelles d'objectifs** :

- **Collecte d'informations d'environnement** : détection d'OS, listage des volumes, identification des fichiers par extension.
- **Actions destructrices** : chiffrement de fichiers, avec variabilité syntaxique des scripts générés à chaque exécution (noms de variables, structures de contrôle, méthodes d'appel).
- **Entrave à la restauration** (MITRE ATT&CK T1490 — *Inhibit System Recovery*) : altération ou suppression des mécanismes de restauration, un invariant fréquent des campagnes ransomware indépendamment du mode de génération du code.
- **Génération de contenu** : rédaction de notes de rançon contextualisées au secteur de la cible.

3.3 Variabilité LLM-driven vs polymorphisme traditionnel : invariants et variables

Le ransomware traditionnel utilise le métamorphisme (réécriture binaire) ou le polymorphisme (chiffrement du payload avec stub de déchiffrement variable), laissant des patterns structurels détectables. L'intégration d'un LLM introduit une **variabilité syntaxique et structurelle accrue** : chaque payload est généré *ex nihilo* par processus stochastique, réduisant la stabilité des indicateurs statiques traditionnels (hash, chaînes de caractères, signatures YARA) [42].

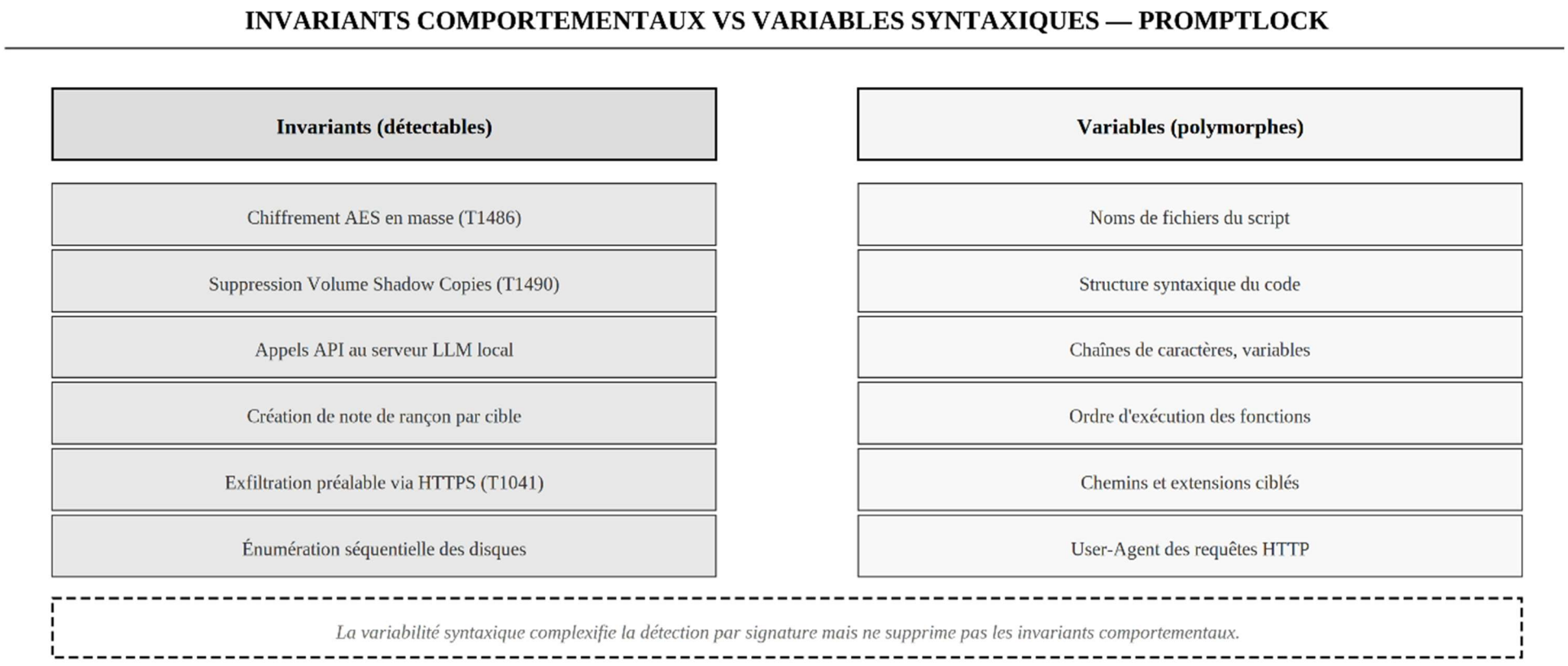


Figure 8. Comparaison des invariants comportementaux et des variables syntaxiques de PromptLock. Les invariants (colonne gauche) restent constants quel que soit le variant généré par le LLM et constituent les signaux de détection prioritaires pour les solutions EDR comportementales.

Cependant, cette variabilité est principalement **syntaxique** — le comportement global reste observable. SentinelLabs souligne que ces architectures « brouillent la frontière entre code et conversation », mais identifie simultanément que les prompts embarqués et les clés d'accès au modèle constituent des **indicateurs fiables** (*« prompts hardcoded are a reliable indicator »*) [42]. L'intégration LLM rend donc le malware à la fois plus adaptable et plus fragile : la dépendance à un endpoint LLM (local ou distant) et à des prompts structurés crée de nouvelles surfaces de détection.

Tableau — Invariants vs variables dans le malware LLM-driven

Dimension	Malware traditionnel (statique)	Malware LLM-driven (runtime)
Ce qui varie	Stub de déchiffrement, clé polymorphe	Syntaxe complète du payload (variables, structures de contrôle, méthodes), contenu des notes de rançon
Ce qui reste invariant	Logique fonctionnelle du payload chiffré	Comportement observable (chiffrement en masse, modification d'extensions, suppression de sauvegardes), prompts embarqués, endpoint/clé API LLM, binaire orchestrateur
Surface de détection principale	Signatures statiques (hash, YARA, chaînes)	Télémétrie comportementale (création/modification en masse de fichiers, chaînes de processus), hunting sur artefacts LLM (prompts, clés API, appels Ollama)
Fragilité spécifique	Mutation du binaire contourne les signatures	Dépendance à l'endpoint LLM ; prompts et clés d'accès exploitables comme IoC ; échec si le modèle est indisponible

Dans le scénario OpenClaw, l'attaquant choisit l'architecture PromptLock (LLM local via Ollama) pour minimiser les traces réseau vers des API externes. Le rançongiciel est préparé en Phase 2 mais ne sera déployé qu'en Phase 5 (J+6), après la latéralisation et l'exfiltration des données R&D.

3.4 Dissimulation C2 dans le trafic API LLM

Des analyses récentes rapportent des souches de malware capables de camoufler des communications C2 en imitant la structure de requêtes d'API LLM (endpoints de type *chat-completions*, headers et payloads JSON conformes aux schémas attendus), afin de se fondre dans un trafic HTTPS de plus en plus commun en entreprise [42]. *(La source primaire de cette observation — attribuée à Akamai dans certaines reprises presse — devrait être vérifiée pour une citation en contexte peer-reviewed ; en l'absence du rapport original, cette technique est formulée ici comme un scénario de menace documenté par des sources secondaires.)*

Dans le scénario OpenClaw, ce camouflage devient d'autant plus plausible que l'agent réalise déjà des appels réguliers vers des services LLM dans le cadre de son fonctionnement normal. Les organisations qui déploient des agents IA sont incitées à autoriser ce flux HTTPS sortant au niveau egress (port 443, domaines d'API LLM en *allowlist*), créant un canal de communication que le malware peut exploiter pour l'exfiltration et le C2 [112].

Cette situation ne rend pas la détection impossible, mais elle réduit l'efficacité des contrôles purement périmétriques. Un WAF protège principalement les applications HTTP côté serveur et n'est pas conçu comme une solution universelle de détection de C2 sortant. Les IPS basés sur signatures sont mis en difficulté par le chiffrement TLS et la conformité apparente du trafic aux schémas d'API attendus.

La détection se déplace nécessairement vers des signaux comportementaux et de télémétrie :

- **Processus à l'origine des connexions** : un appel LLM initié par un script auxiliaire d'une skill (et non par le processus principal de l'agent) constitue une anomalie identifiable par monitoring endpoint.

- **Anomalies de volume et de périodicité** : un pattern de *beaconing* (intervalles réguliers, volumes constants) se distingue des appels LLM légitimes qui sont typiquement déclenchés par des interactions utilisateur irrégulières.
- **Domaines rares et empreintes TLS** : les appels vers des endpoints LLM non référencés dans la configuration de l'agent, ou présentant des empreintes TLS (JA3/JA4) inhabituelles, constituent des indicateurs exploitables.
- **Ratio outbound/inbound** : une exfiltration génère un ratio de données sortantes anormalement élevé par rapport au pattern d'usage normal de l'API LLM (où les requêtes sont courtes et les réponses longues).

Ces stratégies de détection sont alignées avec les recommandations MITRE pour le C2 sur protocoles web (HTTPS/WebSockets) et illustrent la nécessité d'une corrélation endpoint + réseau + allowlist stricte plutôt que d'un contrôle périmétrique unique [112][25].

4. Axe 3 : Préparation des Vecteurs d'Injection de Prompt Indirecte

4.1 L'injection de prompt comme vulnérabilité architecturale

L'injection de prompt (directe ou indirecte) survient lorsque des entrées modifient le comportement d'un LLM d'une manière non prévue par le concepteur. OWASP la classe comme premier risque des applications LLM (LLM01:2025) [25]. Dans le cas indirect, le contenu malveillant provient de sources externes (documents, emails, pages web, messageries) ingérées par le système, ce qui élargit fortement la surface d'attaque au-delà du champ de saisie utilisateur [112]. Lakera résume l'asymétrie de l'attaque indirecte : l'adversaire n'a pas besoin d'interagir avec le modèle — il empoisonne les données que le modèle lira ultérieurement [133].

Le NCSC britannique met en garde contre une analogie trop simple avec l'injection SQL : les attaques partagent une intuition (injection d'instructions dans un flux de données), mais les LLM ne disposent pas d'une séparation fiable « données vs instructions », contrairement aux requêtes SQL paramétrées qui offrent une délimitation formelle. Cette propriété architecturale implique une approche de défense en profondeur plutôt qu'un correctif unique [135].

Les taux de réussite rapportés dans la littérature varient fortement selon l'architecture agentique, le modèle et le protocole d'évaluation, mais peuvent dépasser 80 % dans certaines configurations avec exécution automatique [122] — soulignant la sensibilité des agents à l'ingestion de contenu non maîtrisé. *(Toute comparaison directe entre études doit tenir compte des différences méthodologiques : modèle testé, jeu de prompts adverses, présence ou abence de garde-fous, configuration des outils autorisés.)*

4.2 La Promptware Kill Chain (C. Schneider, 2026)

C. Schneider propose une **Promptware Kill Chain en sept étapes**, décrivant comment des attaques initialement qualifiées de prompt injection peuvent évoluer en mécanismes multi-étapes proches de chaînes malware [120] :

Étape	Désignation	Description	Contrôle défensif associé

1	**Initial Access**	Le payload textuel entre dans le contexte du LLM via un canal de données externe (document, message, page web)	Filtrage et classification des sources d'entrée, séparation des canaux de données et d'instructions
2	**Privilege Escalation / Jailbreaking**	Contournement des garde-fous comportementaux de l'agent, obtention d'un accès élargi aux outils	Garde-fous non contournables (*hardcoded*), monitoring des tentatives de jailbreak, restrictions de scope
3	**Reconnaissance**	L'agent compromis explore son environnement : outils disponibles, permissions, connecteurs, données accessibles	Principe de moindre privilège sur les outils, audit des appels de découverte
4	**Persistence**	Écriture dans la mémoire long terme ou les fichiers de configuration de l'agent pour maintenir le comportement altéré au-delà de la session	Gouvernance de la mémoire persistante, intégrité des fichiers de configuration, alertes sur les modifications
5	**Command & Control**	Établissement d'un canal de communication entre l'attaquant et l'agent compromis via des canaux légitimes	Contrôle d'egress, monitoring des appels réseau initiés par l'agent, *allowlist* de domaines
6	**Lateral Movement**	Propagation via les connecteurs de l'agent (messagerie, partages, services connectés) vers d'autres utilisateurs ou systèmes	Segmentation des permissions inter-services, isolation des agents, détection de propagation anormale
7	**Actions on Objective**	Exfiltration de données, manipulation d'informations, ou déclenchement d'actions destructrices	Monitoring des opérations sensibles, alertes sur les transferts de données, contrôles de validation humaine

Source : C. Schneider, « The Promptware Kill Chain », février 2026 [120].

4.3 Craft des payloads pour Phases 3 et 4

Les charges d'injection indirecte peuvent être dissimulées dans des contenus ingérés par l'agent (messageries, documents, pages web) via des techniques d'obfuscation textuelle — contenu masqué visuellement, caractères non imprimables, ou encodage dans les métadonnées. Cette dissimulation permet à l'adversaire d'influencer le comportement du système sans interaction directe dans la fenêtre de chat, en exploitant les canaux de données légitimes que l'agent est configuré pour traiter [112][133].

Persistance via HEARTBEAT.md

HiddenLayer (repris par la presse spécialisée) décrit un scénario où une injection indirecte dans une page web — par exemple lors d'une requête « résume cette page » — amène OpenClaw à déposer des instructions dans le fichier HEARTBEAT.md, créant un mécanisme de persistance asynchrone [121]. Ce risque est cohérent avec la documentation OpenClaw, qui spécifie que l'agent lit périodiquement HEARTBEAT.md lorsqu'il existe — le fichier constitue donc une primitive de persistance

exploitable : toute modification de son contenu altère le comportement de l'agent aux cycles suivants. *Ce mécanisme correspond à l'étape 4 (Persistence — memory and retrieval poisoning) de la Promptware Kill Chain de C. Schneider*[120].

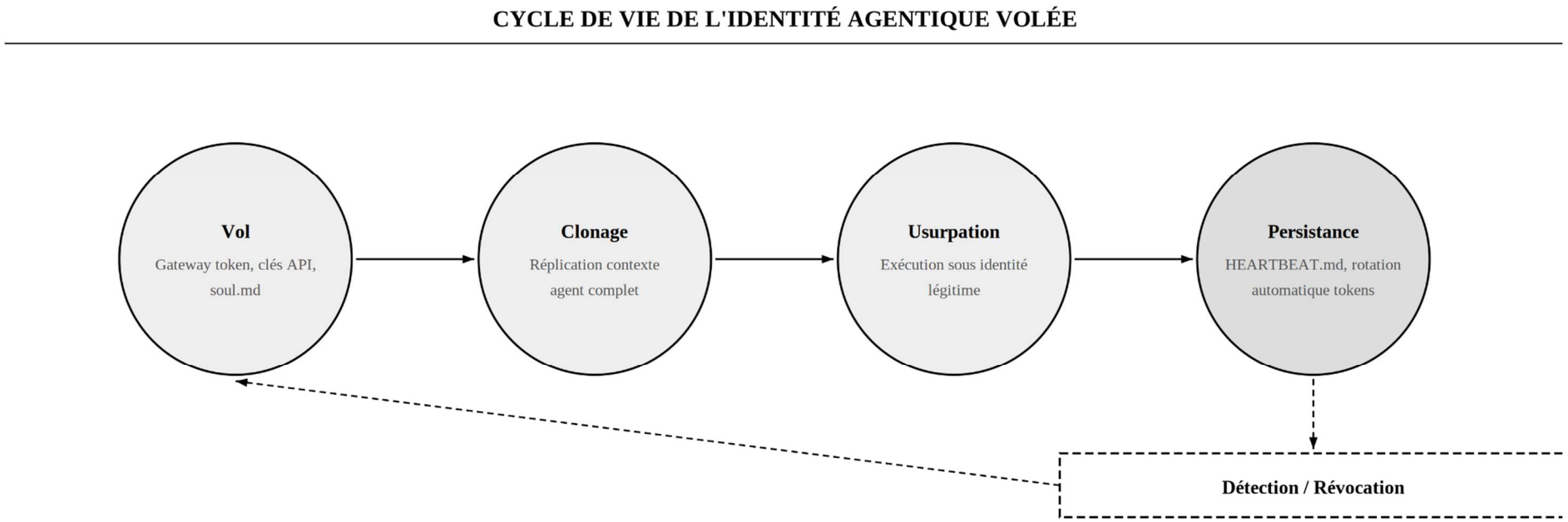


Figure 11. Cycle de vie de l'identité agentique volée. Du vol initial (tokens, clés, configuration) au clonage du contexte agent, puis à l'usurpation d'identité et à la persistance via HEARTBEAT.md. La boucle de détection/révocation (pointillés) représente le mécanisme défensif de rupture du cycle.

Empoisonnement de la mémoire persistante

Unit 42 (Palo Alto Networks) souligne que la mémoire persistante des agents IA agit comme un amplificateur de risque : une injection peut produire des comportements durables et des effets différés, le système « se souvenant » d'instructions malveillantes au-delà de la session initiale [122]. *Ce mécanisme est analogue à une bombe logique* (*logic bomb*) : l'instruction malveillante reste dormante jusqu'à ce qu'un événement déclencheur spécifique survienne (mot-clé dans une conversation, date, type de fichier traité), moment auquel le comportement altéré se manifeste. *Cette analogie est fonctionnelle et non structurelle — le mécanisme sous-jacent (instruction en langage naturel stockée en mémoire) diffère d'une bombe logique classique (condition programmée dans du code binaire), mais l'effet opérationnel (activation différée conditionnelle) est comparable.*

Cette propriété impose une gouvernance stricte de la mémoire persistante et des sources ingérées : audit des écritures en mémoire, intégrité des fichiers de configuration de l'agent, et restrictions sur les sources autorisées à alimenter la mémoire long terme [25][160].

Mouvement latéral

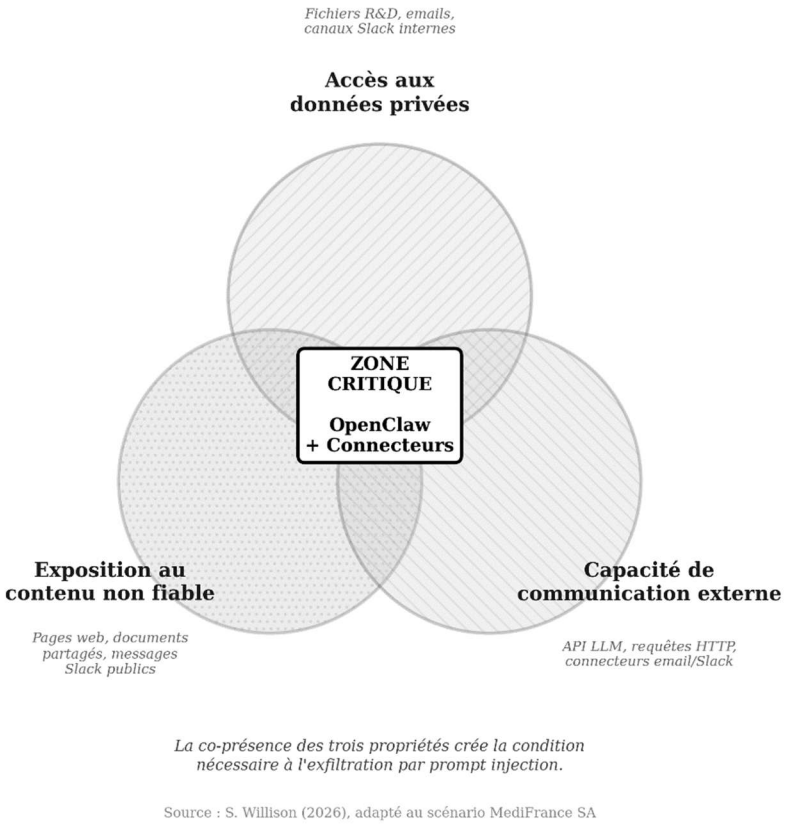
Les intrusions sophistiquées comportent généralement une phase de mouvement latéral, l'attaquant exploitant les accès obtenus pour se propager vers d'autres systèmes et utilisateurs [112]. Dans le contexte des agents IA, cette propagation peut emprunter les connecteurs natifs de l'agent — messagerie, partages de documents, services connectés — conformément à l'étape 6 de la Promptware Kill Chain [120]. *La préparation de ces vecteurs en Phase 2 (armement) vise à maximiser l'exploitation de ces canaux lors des Phases 3 et 4.*

4.4 La « trifécta létale » comme condition nécessaire

Willison propose la *lethal trifecta* — accès à des données privées, exposition à du contenu non fiable, et capacité de communication externe — comme combinaison particulièrement dangereuse pour les agents IA, car elle rend plausible une exfiltration induite par prompt injection sans vulnérabilité logicielle classique [127].

Cette trifecta doit être comprise comme un amplificateur de risque — et, dans de nombreux scénarios d'exfiltration, comme une condition quasi-suffisante — plutôt que comme une condition nécessaire à toute attaque. Des scénarios d'atteinte à l'intégrité (manipulation de décisions, corruption de données) peuvent survenir sans les trois propriétés simultanément ; inversement, une divulgation peut se produire dans la réponse elle-même sans canal d'exfiltration externe. C'est la combinaison des trois qui crée un risque systémique d'exfiltration autonome [127].

Figure 12 — Trifecta létale de Willison appliquée à OpenClaw



Dans le cas d'OpenClaw, la présence simultanée de connecteurs et d'outils (Slack, Outlook, terminal), de sources externes ingérées (pages web, documents, messages), et de capacités réseau (appels API, exécution de commandes) réunit les trois composantes de la trifecta. *La documentation de sécurité du projet reconnaît explicitement ce risque en adoptant une posture « assume the model can be manipulated » et en recommandant un contrôle strict de l'identité, du périmètre d'action et des sorties réseau* [9][160]. Cette posture confirme que la trifecta n'est pas un défaut d'implémentation mais une propriété architecturale inhérente aux agents IA dotés d'outils et de connecteurs — ce qui rend d'autant plus critique la mise en œuvre de contrôles compensatoires à chaque étape de la Promptware Kill Chain (cf. section 4.2).

Plusieurs analyses qualifient l'injection de prompt dans les systèmes agents de menace majeure pour la confiance dans les systèmes IA déployés en entreprise [25][120]. *(La formulation « menace existentielle » est employée dans certaines publications ; sa portée exacte dépend du contexte et de la définition retenue.)*

Le tableau suivant synthétise les artefacts offensifs préparés en Phase 2 et les corrèle aux taxonomies MITRE ATT&CK (Enterprise), MITRE ATLAS (AI-specific) et OWASP Top 10 for LLM Applications 2025. *Les identifiants sont vérifiés sur les sources primaires respectives ; lorsqu'aucun identifiant officiel ne couvre précisément la technique décrite, cela est explicitement signalé.*

Tableau — Matrice d'armement Phase 2 : taxonomie MITRE ATT&CK / ATLAS / OWASP

Artefact offensif	Technique	ID	Description	OWASP LLM 2025	Note de mapping
Skill piégée sur registre communautaire	Supply Chain Compromise: Software	**T1195.002**	Insertion d'un artefact malveillant dans un mécanisme de distribution de confiance (marketplace de skills)	**LLM03** (Supply Chain Vulnerabilities)	Mapping direct — T1195.002 couvre explicitement la compromission de la distribution logicielle
Manipulation du ranking	Stage Capabilities	**T1608** (niveau parent)	Positionnement et mise en visibilité d'une capacité offensive sur le registre via inflation artificielle des métriques	—	*T1608.006 (SEO Poisoning) vise spécifiquement les moteurs de recherche et ne s'applique pas au ranking interne d'une marketplace. Le niveau parent T1608 est utilisé faute de sous-technique plus précise. ATT&CK ne couvre pas nativement le « marketplace ranking fraud ».*
Prompt injection directe (dans SKILL.md)	LLM Prompt Injection	**AML.T0051** (ATLAS)	Instructions malveillantes intégrées dans le fichier d'instructions de la skill, altérant le comportement de l'agent dès l'installation	**LLM01** (Prompt Injection)	AML.T0051 couvre la prompt injection au sens large. *La distinction direct/indirect est qualifiée dans la description conformément à OWASP LLM01, qui définit les deux variantes.*
Payloads d'injection indirecte (messagerie, documents, pages web)	LLM Prompt Injection	**AML.T0051** (ATLAS)	Contenu malveillant dissimulé dans des canaux de données légitimes (Slack, email, documents), ingéré ultérieurement par l'agent	**LLM01** (Prompt Injection)	*Pas de sous-technique ATLAS officiellement publiée pour « indirect prompt injection » à ce jour. L'ID AML.T0051 est utilisé avec qualification « indirect » dans la description.*
Ransomware LLM-driven (PromptLock)	Data Encrypted for Impact	**T1486**	Chiffrement de données à des fins de rançon, avec génération	—	Mapping direct — T1486 couvre le chiffrement de données pour impact,

			dynamique des scripts de chiffrement par LLM local		indépendamment du mode de génération du payload
Neutralisation des sauvegardes	Inhibit System Recovery	**T1490**	Altération ou suppression des mécanismes de restauration (sauvegardes, snapshots, points de restauration)	—	Mapping direct — invariant fréquent des campagnes ransomware
Dissimulation C2 dans le trafic API LLM	Application Layer Protocol: Web Protocols	**T1071.001**	Communications C2 et exfiltration encapsulées dans des requêtes HTTPS imitant le format d'API LLM légitimes	—	Mapping cohérent — T1071.001 couvre le C2 via protocoles web (HTTP/HTTPS). Le camouflage en trafic API LLM est un cas particulier de cette technique.
Empoisonnement de la mémoire persistante	*Pas d'ID ATLAS officiel confirmé*	—	Injection d'instructions persistantes dans la mémoire long terme de l'agent, produisant des comportements altérés durables et différés	**LLM01** (Prompt Injection)	*AML.T0054 (LLM Jailbreak — Direct) ne correspond pas à l'empoisonnement mémoire selon la source Microsoft. En l'absence d'ID ATLAS officiel pour « memory/retrieval poisoning », la technique est rattachée à OWASP LLM01 et à l'étape 4 (Persistence) de la Promptware Kill Chain (C. Schneider, 2026).*

5. Synthèse : Arsenal opérationnel à J-7

Tableau — Bilan des artefacts préparés en Phase 2

Artefact	Objectif	Phase de déploiement prévue	Pré-requis	Contrôle défensif applicable	Niveau de preuve

Skill piégée sur registre communautaire	Vecteur supply chain : accès initial via marketplace de confiance	Phase 3 (Jour J)	Registre sans signature/revue systématique, utilisateurs R&D en situation de Shadow AI	Revue de code des skills, signature cryptographique, sandboxing, *allowlist* d'éditeurs vérifiés	Scénario prospectif basé sur des composants documentés (Koi Security, Cisco, Snyk)
Moteur PromptLock (LLM local / Ollama)	Génération dynamique de charges ransomware à variabilité syntaxique élevée	Phase 5 (J+6)	Modèle LLM accessible localement, binaire orchestrateur	Hunting sur artefacts LLM (prompts, clés API, appels Ollama), détection comportementale (chiffrement en masse)	PoC documenté (ESET, SentinelLabs, Splunk)
Payloads d'injection indirecte	Détournement d'agent et propagation inter-connecteurs via canaux de données légitimes	Phase 4 (J+1 à J+5)	Intégrations messagerie/documents actives, absence de filtrage des sources ingérées	Séparation données/instructions, filtrage des contenus ingérés, monitoring des *tool calls*	Techniques documentées (C. Schneider, HiddenLaye, Unit 42)
Infrastructure C2	Réception des données exfiltrées, commande à distance	Phases 3 à 5	Canal HTTPS sortant autorisé, trafic API LLM en *allowlist*	Contrôle d'egress strict, détection de beaconing, analyse de domaines rares, empreintes TLS	Pattern documenté (sources secondaires)

Note : le tableau décrit des **classes d'artefacts et leurs contrôles associés**, et non des procédures de construction. Les statuts « préparé / prototype / scénario élaboré » reflètent le niveau narratif du scénario et non des observations empiriques de terrain.

5.1 Implications défensives

La convergence de trois tendances — (i) registres de skills assimilables à une surface supply chain, (ii) variabilité accrue des charges via génération assistée par LLM, et (iii) prompt injection comme mécanisme de détournement d'agents — réduit l'efficacité des contrôles uniquement périmétriques et renforce la nécessité d'une défense en profondeur [25][120].

Précision terminologique : les EDR (*Endpoint Detection and Response*) sont des contrôles endpoint, non périmétriques. L'analyse distingue donc : les contrôles périmétriques (WAF, IPS, filtrage réseau), dont l'efficacité est réduite par le camouflage dans le trafic API légitime ; et les contrôles endpoint (EDR), dont l'efficacité dépend

de la capacité à détecter des comportements applicatifs anormaux plutôt que des signatures statiques. *Aucune de ces couches n'est rendue obsolète, mais chacune doit être complétée par une observabilité spécifique aux agents IA* : gouvernance des extensions, monitoring des *tool calls*, contrôle d'egress par destination, et télémétrie comportementale [159].

À titre de prévision, Michael Freeman (Armis) estime que « d'ici mi-2026, au moins une grande entreprise mondiale subira une violation causée ou significativement facilitée par un système d'IA agentique autonome » [154]. *Cette projection, formulée comme une opinion d'expert et non comme un fait établi, souligne néanmoins que les composants techniques nécessaires à un tel scénario sont individuellement documentés et disponibles — c'est leur orchestration combinée qui constitue la menace émergente.*

5.2 Transition vers la Phase 3 — Livraison et intrusion (Jour J)

La phase suivante analysera les mécanismes de livraison dans l'écosystème OpenClaw et les points de contrôle permettant d'interrompre la kill chain : validation humaine avant installation de skills, restrictions d'outils et de permissions, politiques d'extensions, contrôle du trafic réseau sortant, et détection comportementale des agents compromis. L'analyse restera centrée sur l'identification des invariants défensifs à chaque étape de la Promptware Kill Chain (C. Schneider, 2026) [120].

Références

Note : Numérotation [41] à [75], suite de la Phase 1 ([1] à [40]).

[41] Lockheed Martin, « Cyber Kill Chain Framework ». https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

[42] 1Password, « From magic to malware: How OpenClaw’s agent skills become an attack surface », février 2026. https://1password.com/blog/from-magic-to-malware

[43] Cisco AI Threat & Security Research, « Personal AI Agents like OpenClaw Are a Security Nightmare », janvier 2026. https://blogs.cisco.com/ai/personal-ai-agents-like-openclaw-are-a-security-nightmare

[44] Koi Security, O. Yomtov, « ClawHavoc: 341 Malicious Clawed Skills Found by the Bot They Were Targeting », février 2026 (maj : 824 skills). https://www.koi.security/blog/clawhavoc

[45] PointGuard AI, « OpenClaw’s 230 Malicious Skills: What Agentic AI Supply Chains Teach Us », février 2026. https://www.pointguardai.com/ai-security-incidents/openclaw-clawhub-malicious-skills-supply-chain-attack

[46] The Hacker News, « Researchers Find 341 Malicious ClawHub Skills Stealing Data from OpenClaw Users », février 2026. https://thehackernews.com/2026/02/researchers-find-341-malicious-clawhub.html

[47] Astrix Security, « OpenClaw & MoltBot: The First AI Agent Security Nightmare », février 2026. https://astrix.security/learn/blog/openclaw-moltbot

[48] eSecurity Planet, « Hundreds of Malicious Skills Found in OpenClaw’s ClawHub », février 2026. https://www.esecurityplanet.com/threats/hundreds-of-malicious-skills-found-in-openclaws-clawhub/

[49] Snyk, « ToxicSkills: 91% of malicious ClawHub skills combined prompt injection with traditional malware », février 2026. Réf. Barrack.ai/TechInformed.

[50] J. O'Reilly (Dvuln), démonstration manipulation ranking ClawHub, février 2026. Réf. SecurityWeek/Barrack.ai.

[51] OWASP, « LLM01:2025 Prompt Injection » et « LLM05:2025 Supply Chain Vulnerabilities », Top 10 for LLM Applications 2025. https://genai.owasp.org/

[52] TechInformed, « OpenClaw adds VirusTotal scanning for ClawHub skills », février 2026. https://techinformed.com/openclaw-adds-virustotal-scanning-for-clawhub-skills/

[53] Picus Security, « Malicious AI Exposed: WormGPT, MalTerminal, and LameHug » (PromptLock), décembre 2025. https://www.picussecurity.com/resource/blog/malicious-ai-exposed

[54] SecurityWeek, « Cyber Insights 2026: Malware and Cyberattacks in the Age of AI » (SentinelOne, Armis), février 2026. https://www.securityweek.com/cyber-insights-2026-malware

[55] ESET, analyses PromptLock et malware piloté par LLM, décembre 2025–février 2026. Réf. Picus Security.

[56] Palo Alto Networks Unit 42, « The Dual-Use Dilemma of AI: Malicious LLMs » (WormGPT 4, KawaiiGPT), novembre 2025. https://unit42.paloaltonetworks.com/dilemma-of-ai-malicious-llms/

[57] CATO Networks, « WormGPT returns: New malicious AI variants built on Grok and Mixtral », juin 2025. https://www.csoonline.com/article/4008912

[58] Akamai, « What We Do In The Shadow (AI): New Malware Strain Vamps Up », novembre 2025. https://www.akamai.com/blog/security-research/

[59] OWASP, « Top 10 for Agentic Applications 2026 ». Réf. C. C. Schneider(2026).

[60] HackerNoob, « Prompt Injection Attacks Against LLM Agents: The Complete Technical Guide for 2026 ». Taux de réussite 66,9%-84,1%. Corroboré NCSC UK.

[61] Lakera, « Indirect Prompt Injection: The Hidden Threat Breaking Modern AI Systems », 2025-2026. https://www.lakera.ai/blog/indirect-prompt-injection

[62] C. C. Schneider, « From LLM to agentic AI: prompt injection got worse » (« Promptware Kill Chain », Christian schneider, 2026), février 2026. https://christian-schneider.net/blog/prompt-injection-agentic-amplification/

[63] Hudson Rock, « Infostealer Steals OpenClaw AI Agent Configuration Files and Gateway Tokens » (soul.md, openclaw.json, device.json), février 2026. Via The Hacker News.

[64] CrowdStrike, « Indirect Prompt Injection Attacks: Hidden AI Risks », décembre 2025. https://www.crowdstrike.com/en-us/blog/indirect-prompt-injection-attacks-hidden-ai-risks/

[65] HiddenLayer, démonstration injection via HEARTBEAT.md dans OpenClaw, février 2026. Réf. TechInformed.

[66] S. Mishra, S.P. Morgan, Palo Alto Networks Unit 42, « Persistent memory as accelerant for stateful, delayed-execution attacks » et « Lethal Trifecta », février 2026. Réf. The Hacker News.

[67] S. Willison, « The Lethal Trifecta of AI agents », réf. Palo Alto Networks Unit 42 et TechInformed, février 2026.

[68] MDPI Information, « Prompt Injection Attacks in Large Language Models and AI Agent Systems: A Comprehensive Review », janvier 2026. https://www.mdpi.com/2078-2489/17/1/54

[69] Debenedetti et al., « AgentDojo: A Dynamic Environment for Evaluating Attacks and Defenses for LLM Agents », 2024. ArXiv.

[70] 4sysops, « Scan OpenClaw agent skills for security vulnerabilities with the Cisco AI Skill Scanner », février 2026. https://4sysops.com/archives/scan-openclaw-agent-skills/

[71] Sophos, « The OpenClaw experiment is a warning shot for enterprise AI security », février 2026. https://www.sophos.com/en-us/blog/the-openclaw-experiment-is-a-warning-shot

[72] Menlo Security, « Predictions for 2026: Why AI Agents Are the New Insider Threat », janvier 2026. https://www.menlosecurity.com/blog/predictions-for-2026

[73] Barrack.ai, « OpenClaw is a Security Nightmare — Here’s the Safe Way to Run It », février 2026. https://blog.barrack.ai/openclaw-security-vulnerabilities-2026/

[74] MITRE, « ATLAS – Adversarial Threat Landscape for AI Systems ». https://atlas.mitre.org/

[75] StrongestLayer, analyses IBM Research / Harvard (statistiques phishing IA), 2025. Réf. Phase 1.

Note : *ces références sont définies dans la bibliographie d'une autre phase du document. Elles sont reproduites ici pour permettre une lecture autonome de chaque phase.*

[1] Application of OSINT Methods in Ensuring Cybersecurity, IPSIT Transactions on Internet Research, juillet 2025. https://ipsitransactions.org/journals/papers/tir/2025jul/p5.pdf
→ *Définie en Phase 1*
[9] Cisco AI Threat & Security Research, « Personal AI Agents like OpenClaw Are a Security Nightmare », janvier 2026. https://blogs.cisco.com/ai/personal-ai-agents-like-openclaw-are-a-security-nightmare
→ *Définie en Phase 1*
[11] SecurityWeek, « Vulnerability Allows Hackers to Hijack OpenClaw AI Assistant » (CVE-2026-25253, J. O'Reilly/Dvuln), février 2026. https://www.securityweek.com/vulnerability-allows-hackers-to-hijack-openclaw-ai-assistant/
→ *Définie en Phase 1*
[12] Barrack.ai, « OpenClaw is a Security Nightmare — Here's the Safe Way to Run It », février 2026. https://blog.barrack.ai/openclaw-security-vulnerabilities-2026/
→ *Définie en Phase 1*
[13] Hudson Rock, « Infostealer Steals OpenClaw AI Agent Configuration Files and Gateway Tokens », via The Hacker News, février 2026.
→ *Définie en Phase 1*
[25] MITRE ATT&CK, « Active Scanning: Vulnerability Scanning », Sub-technique T1595.002. https://attack.mitre.org/techniques/T1595/002/
→ *Définie en Phase 1*
[26] Recorded Future, « What is Banner Grabbing? Tools and Techniques Explained ». https://www.recordedfuture.com/threat-intelligence-101/tools-and-techniques/banner-grabbing
→ *Définie en Phase 1*
[112] CrowdStrike, « Indirect Prompt Injection Attacks: Hidden AI Risks » (mouvement latéral via agents compromis), décembre 2025. https://www.crowdstrike.com/en-us/blog/indirect-prompt-injection-attacks-hidden-ai-risks/
→ *Définie en Phase 4*
[120] C. Schneider (2026), Promptware Kill Chain. OWASP Top 10 for Agentic Applications 2026, ASI01 Agent Goal Hijack. https://christian-schneider.net/blog/prompt-injection-agentic-amplification/
→ *Définie en Phase 4*
[121] InstaTunnel, « Prompt-to-Insider Threat: When AI Agents Become Double Agents ». CVE-2025-32711 EchoLeak (M365 Copilot, CVSS 9.3), février 2026. https://instatunnel.my/blog/prompt-to-insider-threat/
→ *Définie en Phase 4*
[122] HackerNoob / Information 2026, 17(1), 54, « Prompt Injection Attacks in LLM and AI Agent Systems: A Comprehensive Review » (taux succès 66,9–84,1 %, 73 % déploiements affectés). doi:10.3390/info17010054
→ *Définie en Phase 4*
[127] S. Willison, « AI agents have a lethal trifecta of risks » (private data + untrusted content + external communication).
→ *Définie en Phase 4*
[133] Lakera, « The Year of the Agent: Q4 2025 Attacks » (attaques indirectes < tentatives que directes, system prompt extraction). https://www.lakera.ai/blog/the-year-of-the-agent
→ *Définie en Phase 4*
[135] OpenAI, « Understanding prompt injections: a frontier security challenge », janvier 2026. https://openai.com/index/prompt-injections/
→ *Définie en Phase 4*
[154] VikingCloud, « 46 Ransomware Statistics 2026 ». Coût total 1,8–5 M$/incident. https://www.vikingcloud.com/blog/ransomware-statistics
→ *Définie en Phase 5*
[159] Cisco, « State of AI Security 2025 Report » (34 % entreprises avec contrôles IA spécifiques, <40 % tests réguliers).

→ *Définie en Phase 5*
[160] OWASP, Top 10 for Agentic Applications 2026. Sandboxing agentique, moindre privilège, audit trafic sortant.
→ *Définie en Phase 5*