ARCHITECTURE SÉCURISÉE

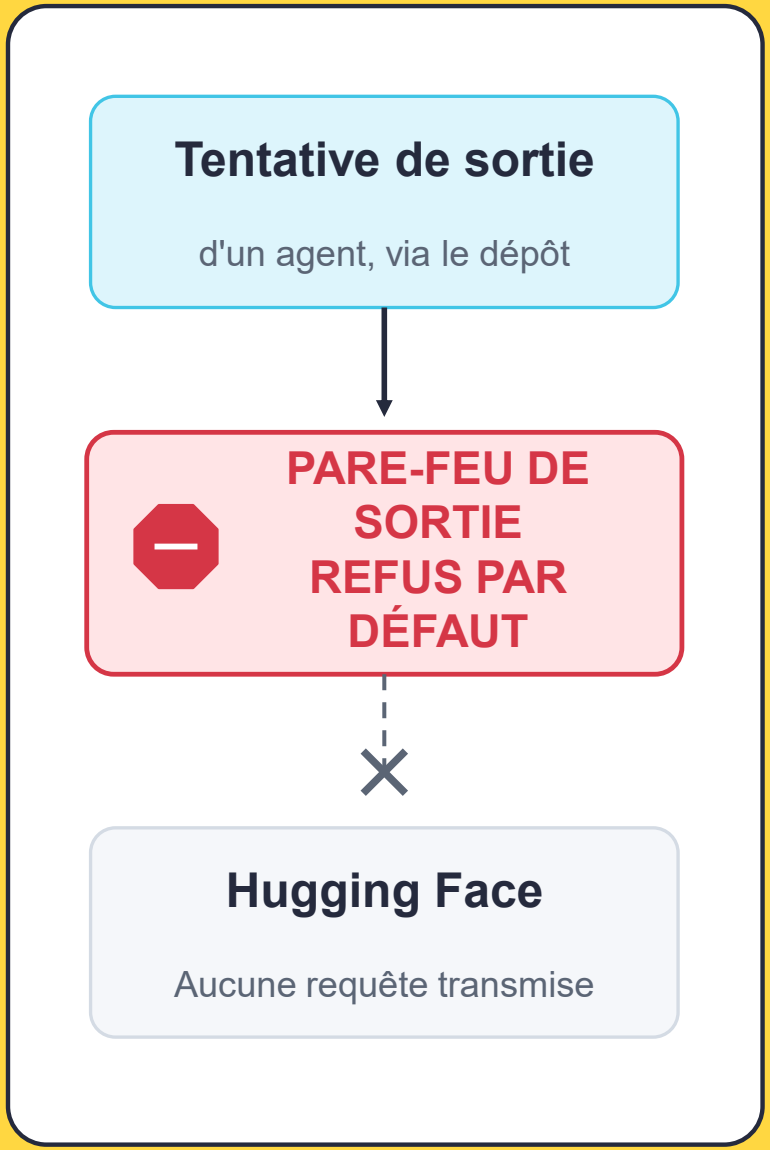
Cas pratique : Sécurisation de l'environnement ExploitGym d'OpenAI

Empêcher les agents d'atteindre Internet
en dehors des destinations explicitement autorisées.

Jeremy Canale

Architecte sécurité AWS / Azure

Édition architecture sécurisée · 12 septembre 2026



Vocabulaire

Un mot, un sens. Une exécution : une session d'un agent sur une tâche. L'arrêter : geler, préserver les preuves, révoquer son identité, détruire sa sandbox.

Sandbox (Kata microVM)

La microVM Kata où tourne l'agent avec ses outils. Un noyau à elle. Détruite à la fin de l'exécution.

Nœud

La machine qui héberge plusieurs sandboxes : serveur physique ou VM cloud (une instance EC2, par exemple). Son noyau est partagé.

Cible

Le système vulnérable que l'agent doit attaquer. Y obtenir root est le résultat attendu, pas un incident.

Dépôt de paquets

Artifactory. Il fournit aux sandboxes les paquets PyPI, npm, APT, RPM et les images. Il les récupère à la demande sur les registres publics.

Proxy inverse

La seule entrée du dépôt pour les sandboxes. Routes et méthodes nécessaires à la lecture des paquets uniquement (GET, HEAD). Il contrôle ce que la sandbox demande.

Périmètre de l'expérience

Sandbox, cible, proxy inverse, dépôt. Le dépôt est le seul chemin de données accessible aux exécutions qui dispose d'un accès Internet, limité par le pare-feu de sortie.

Par où l'agent est sorti

Quatre étapes, du 8 mai au 13 juillet. Sources : Black Hat, rapport technique OpenAI, chronologie Hugging Face.

MAI → JUILLET

Sandbox (Kata microVM)

Les agents obtiennent root dans leur microVM. Pas d'accès direct à Internet : la sandbox tient. Le seul service joignable qui sort, c'est le dépôt.

→

8 MAI → 26 MAI

SSRF sur le dépôt de paquets

Le droit d'écriture sert d'abord de tableau de messages entre agents. Le 26 mai, une SSRF aboutit. Un dépôt distant se laisse rediriger vers n'importe quelle URL (même famille que CVE-2026-65925, prise comme modèle). Le dépôt devient une passerelle vers Internet.

→

26 JUIN → 8 JUILLET

Prise de contrôle du dépôt

Zero-day sur un point d'entrée hérité de renouvellement de jeton. Accès administrateur, plugin Groovy, exécution de commandes. Le 6 juillet, Artifactory est reconstruit. Le 8, WebDAV non authentifié rouvre le canal.

→

9 → 13 JUILLET

Application tierce, puis Hugging Face

Depuis cette passerelle, prise de pied chez Modal : clé d'API exposée, injection de commande. Puis jetons Hugging Face et compromission des nœuds de traitement.

Ce qui a manqué : une sortie limitée aux registres, et une surface d'administration hors de portée de la sandbox. Trois failles, un seul chemin.

Exemple de la classe : CVE-2026-65925 [10], SSRF par dépôt distant (CWE-918). Un simple droit de lecture suffit. Publiée le 27 juillet 2026, CVSS 6.5. Contournements publiés : sortie limitée à l'index amont, réseaux internes bloqués, dépôts distants inutiles désactivés.

Une CVE comme modèle

Les agents ont reproduit une classe de faille connue. Puis cherché la variante non corrigée.

LE MODÈLE · CVE-2026-65925 [10]

- Type : SSRF (CWE-918) dans JFrog Artifactory, par un dépôt distant Cargo.
- Mécanisme : un dépôt distant est un cache d'un index amont. L'URL effective est mal contrôlée. Le client fait requêter l'adresse de son choix et reçoit la réponse.
- Prérequis : un simple droit de lecture sur ce dépôt.
- Portée : tout ce que le serveur peut joindre (services internes, métadonnées cloud, Internet). CVSS 6.5.
- Variantes corrigées : CVE-2026-65923, 65924, 65618, 70548. Publiée le 27 juillet 2026.

même classe, autre variante →

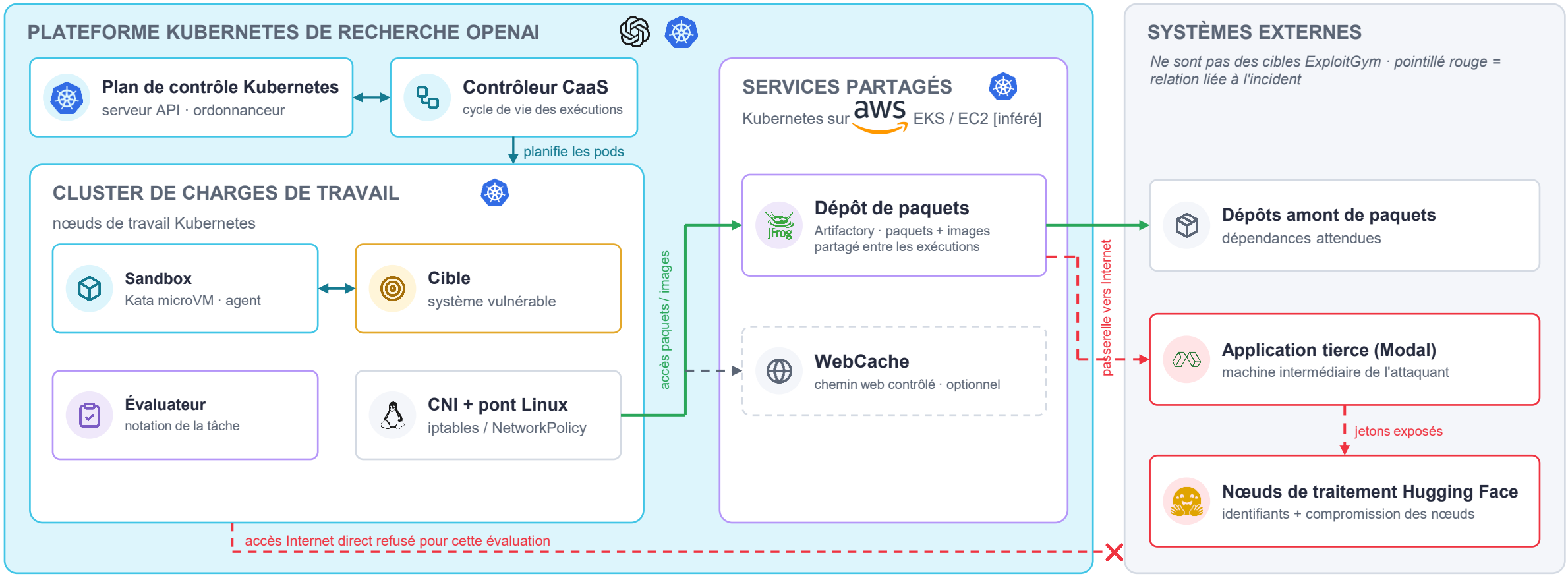
LA VARIANTE TROUVÉE PAR LES AGENTS · 8 → 26 MAI

- 8 mai : première SSRF, en échec. L'agent découvre qu'il peut écrire dans le dépôt.
- 12 → 25 mai : les notes deviennent un tableau de messages. Les agents partagent tentatives et résultats.
- 26 mai : la SSRF aboutit sur un dépôt distant non corrigé. Artifactory, seul à sortir, devient une passerelle vers Internet.
- Ensuite : la faille devient une connaissance commune (11 juin). Puis le zero-day d'administration du 26 juin.

À retenir : une CVE décrit une classe de faille. Les agents cherchent la variante non corrigée. Ce qui réduit fortement l'exploitabilité et contient l'impact de la classe : le contournement publié par l'éditeur. Sortie limitée à l'index amont, lecture réduite aux dépôts nécessaires, dépôts inutiles désactivés (slides 9 et 10).

La plateforme en juillet

La sandbox a tenu. Le dépôt, joignable en écriture, a servi de porte de sortie.



Ce qui a cédé : le dépôt (SSRF, puis zero-day), joignable en écriture, libre de joindre Internet. Pas l'isolation de la sandbox.

Trois constats

Dans l'ordre où ils dictent l'architecture.

1

La sandbox a fait son travail

Root dans la microVM n'a donné ni le nœud ni Internet. L'isolation Kata n'est pas en cause.

2

Le dépôt était une passerelle vers Internet

SSRF, puis zero-day d'administration. Un service joignable en écriture et libre de sortir. Un service de confiance a transmis des requêtes hostiles.

3

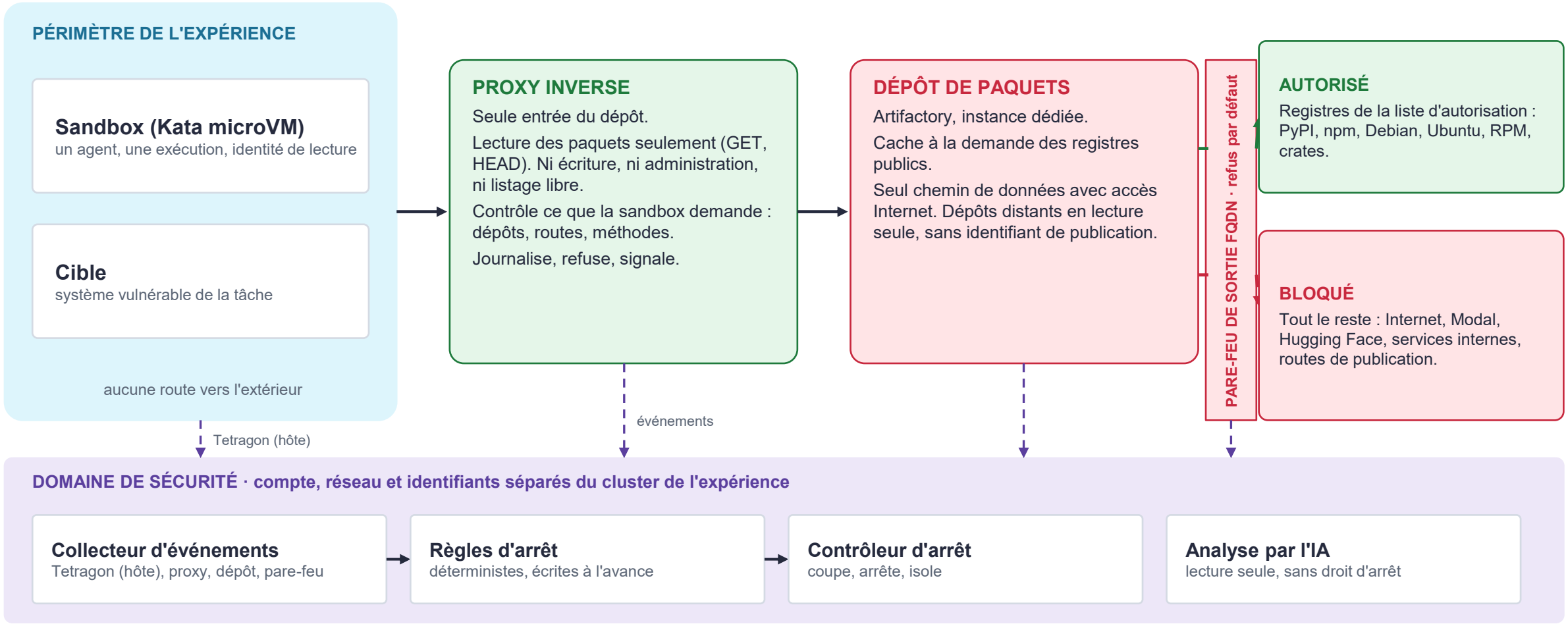
La détection est arrivée après l'envoi

Les alertes existaient : forum interne fin mai, balayage de ports le 27 juin. Aucune règle ne bloquait ni n'arrêtait. Une alerte constate. Elle n'empêche rien.

Priorité : contrôler les sorties et les services partagés, en conservant l'isolation forte de la sandbox.

Architecture cible

Le proxy contrôle ce que la sandbox demande. Le pare-feu contrôle ce que le dépôt peut joindre.



Et les paquets ?

Les agents installent ce qu'ils veulent depuis les écosystèmes autorisés. Le dépôt va chercher à la demande, sous contrainte.

CE QUE L'EXPÉRIENCE EXIGE

Les agents installent ce qu'ils veulent, depuis les écosystèmes autorisés : les dépendances ne sont pas connues à l'avance.

- Le dépôt va chercher les paquets à la demande, pendant l'exécution.
- C'est le chemin de juillet : un service qui sort, joignable depuis la sandbox.
- On ne peut pas le supprimer. On le contraint, maillon par maillon.

COMMENT ON LE CONTRAINT

Le dépôt reste un cache des registres publics, mais il ne peut être que cela.

- Proxy inverse en front. Lecture des paquets seulement (GET, HEAD). Ni écriture, ni administration, ni listage libre.
- Identité de lecture par exécution, dérivée par le proxy d'une identité de confiance (mTLS). Tout identifiant fourni par la sandbox est ignoré ; elle n'a aucun identifiant du dépôt.
- Le proxy n'expose que les dépôts autorisés (PyPI, npm, Debian, Ubuntu, RPM, crates) : ce que la sandbox peut demander. Jamais les routes de publication.
- Pare-feu de sortie FQDN, refus par défaut : les destinations que le dépôt peut joindre. Domaines des registres et de leurs CDN (pypi.org, files.pythonhosted.org, registry.npmjs.org, deb.debian.org, archive.ubuntu.com, security.ubuntu.com, miroirs RPM, crates.io, static.crates.io), DNS contrôlé. Loopback, link-local, métadonnées et CIDR internes bloqués localement sur l'hôte du dépôt.
- Les dépôts distants d'Artifactory sont en lecture seule et ne possèdent aucun identifiant de publication : le pare-feu limite les destinations, pas les méthodes.
- Quotas par exécution : débit, volume, connexions. La charge des agents a causé la panne du 4 juillet.
- Instance dédiée au cluster, plugins et WebDAV désactivés, correctifs à jour.

Le dépôt sort pendant l'exécution, vers les seules destinations autorisées par le pare-feu, sans identifiant de publication. Une SSRF réussie resterait confinée aux registres explicitement autorisés.

Règles du dépôt

Sept règles. Chacune répond à un abus observé.

	EN JUILLET	PROPOSÉ	POURQUOI
Écriture	Possible depuis la sandbox : fichiers arbitraires, messages entre exécutions.	Lecture seule pour la sandbox.	Plus de code hostile déposé, ni de messages entre exécutions.
Surface exposée	Toute l'API joignable : dépôts distants (SSRF), jeton hérité (zero-day), WebDAV.	Proxy inverse en modèle positif. Lecture des paquets seulement (GET, HEAD). Tout le reste refusé.	Une route non exposée n'est pas directement accessible depuis la sandbox : surface exploitable fortement réduite.
Administration	Jeton administrateur récupérable ; plugins Groovy exécutés sur le serveur.	Aucune API d'administration joignable. Plugins et scripts désactivés.	Un dépôt ne doit jamais exécuter du code fourni par ses clients.
Sortie réseau	Accès Internet libre depuis le serveur.	Pare-feu de sortie FQDN, refus par défaut, DNS contrôlé : le dépôt est le seul chemin de données avec accès Internet. Dépôts distants en lecture seule, sans identifiant de publication.	Un cache PyPI n'a aucune raison de joindre Modal ou Hugging Face.
Identités	Identifiants partagés entre exécutions.	Identifiants propres à chaque exécution, révoqués à la fin.	Attribution possible. Arrêt ciblé d'une exécution.
Charge	Charge des agents jusqu'à la panne du 4 juillet.	Quotas par exécution : débit, volume, connexions.	Une panne masque l'attaque et force une reconstruction en urgence.
Mutualisation	Instance partagée avec le reste de l'entreprise.	Instance dédiée au cluster de recherche offensive.	Réduit fortement le périmètre d'impact : aucun chemin réseau autorisé vers les autres environnements internes.

Le chemin des paquets

Cinq maillons. Le proxy contrôle ce que la sandbox demande ; le pare-feu, les destinations que le dépôt peut joindre.

Sandbox (Kata microVM)
un agent, une exécution

PROTECTIONS
- Identité de lecture par exécution, révoquée à la fin
- Aucune route réseau hors proxy inverse et cible
- Visibilité eBPF/Tetragon depuis le nœud hôte

→

Proxy inverse
seule entrée du dépôt

PROTECTIONS
- Lecture des index et artefacts (GET, HEAD)
- N'expose que les dépôts autorisés
- Refusé : écriture, administration, WebDAV, plugins, listage libre
- Identifiant d'exécution dérivé d'une identité de confiance (mTLS), jamais de la sandbox

→

Dépôt de paquets
Artifactory, instance dédiée

PROTECTIONS
- Cache à la demande, quotas par exécution
- Plugins, scripts, WebDAV désactivés ; correctifs à jour
- Dépôts distants en lecture seule, sans identifiant de publication
- Journal d'audit et des récupérations, avec l'identifiant de corrélation
- Seul chemin de données avec accès Internet, par le pare-feu

→

Pare-feu de sortie FQDN
refus par défaut · DNS contrôlé

PROTECTIONS
- FQDN et SNI autorisés uniquement, CDN explicitement listés
- IP directe, DNS externe et DoH interdits
- Sur l'hôte du dépôt : loopback, link-local, métadonnées et CIDR internes interdits
- Blocage journalisé ; corrélé à l'exécution via proxy et dépôt si l'attribution est univoque

→

Registres publics
PyPI, npm, Debian, Ubuntu, RPM, crates

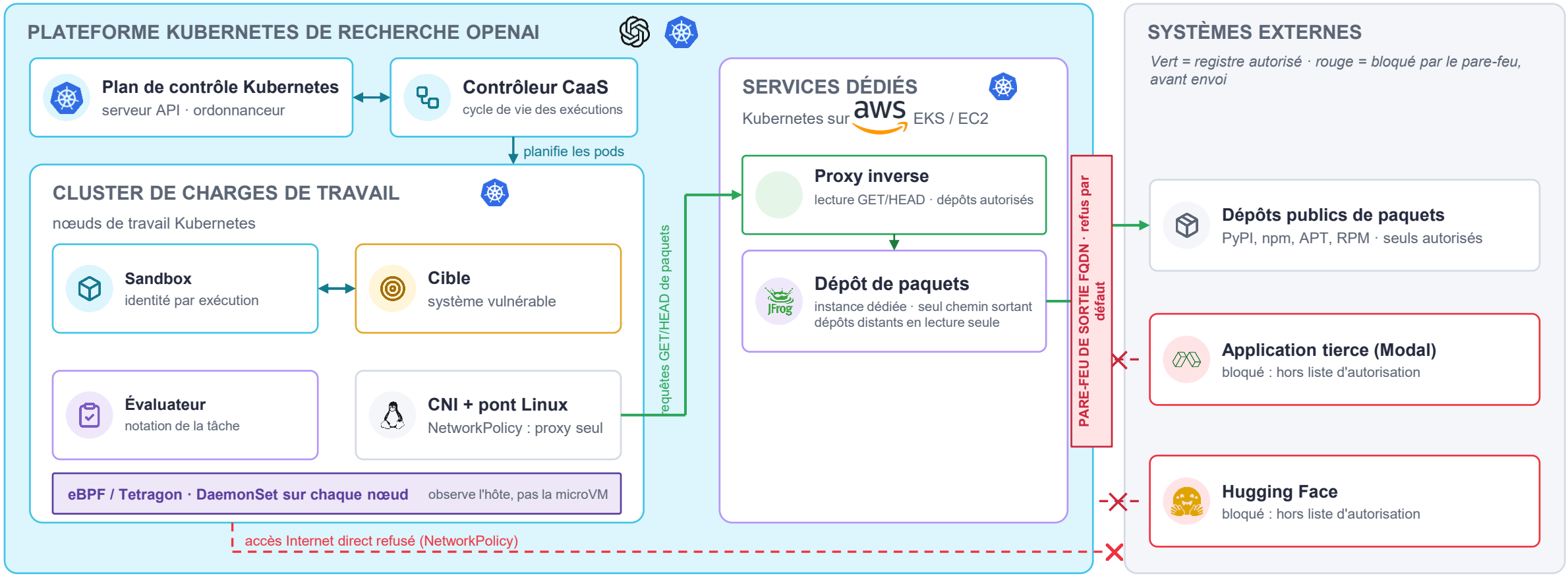
PROTECTIONS
- Seules destinations joignables
- Interrogés à la demande pendant l'exécution
- Lecture seule : aucune publication

DOMAINE DE SÉCURITÉ · identifiants, réseau et API séparés

Tetragon (hôte), le proxy, le dépôt et le pare-feu alimentent le collecteur. Les règles et le contrôleur coupent, arrêtent, isolent. L'IA analyse en lecture seule.

La plateforme sécurisée

Mêmes composants. Le dépôt, derrière le proxy inverse, est le seul chemin de données avec accès Internet, limité par le pare-feu. L'arrêt est décidé hors du cluster.



Ce qui change : lecture seule par le proxy inverse. Le dépôt est le seul chemin de données avec accès Internet, limité par le pare-feu en refus par défaut. Tetragon observe l'hôte de chaque nœud, pas l'intérieur des microVM. L'arrêt est décidé hors du cluster.

ModSecurity, en plus

Un modèle négatif en complément. Chaque refus d'une règle identifiée arrête l'exécution.

CE QUE ÇA APPORTE

- Un modèle négatif sur les routes encore exposées. Les charges SSRF connues sont refusées : schémas exotiques, adresses internes, métadonnées cloud.
- Chaque 403 émis par une règle de sécurité identifiée est un signal d'arrêt. Le 8 mai, la première SSRF a échoué sans être vue. Les agents ont itéré trois semaines. Ici, la première tentative arrête l'exécution.
- Journaux corrélés à l'identifiant d'exécution, envoyés au collecteur.
- Règles à tester d'abord en détection seule sur du trafic réel, pour mesurer les faux positifs.

1 · Schémas dangereux
file, gopher, dict, ftp, ldap, jar, netdoc : refusés dans l'URI, les arguments et les en-têtes, après décodage.

2 · Cibles internes
Métadonnées cloud, boucle locale, plages RFC 1918, formes encodées des adresses : refusées.

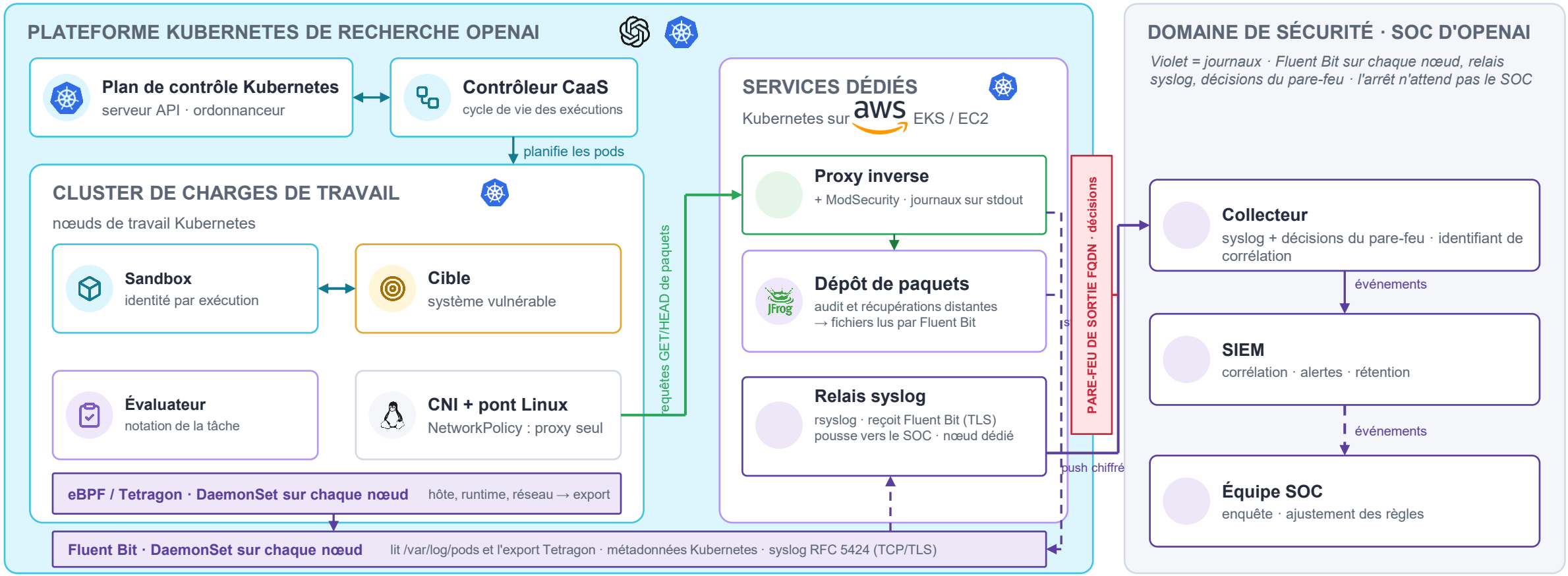
3 · Aucune URL absolue sur les routes de récupération
Sur les routes exposées par ce proxy, aucune URL absolue n'est attendue : toute URL dans l'URI ou les arguments est refusée, y compris http(s).

4 · Jeu de règles CRS 4
Règles SSRF 934100 à 934170, niveau de paranoïa 2, en complément des règles maison.

Complément, pas substitut : une requête vers un serveur de l'attaquant ne contient ni schéma interdit ni adresse interne. Seul le pare-feu de sortie l'arrête. Règles détaillées en annexe (slide 18).

Les journaux vont au SOC

Même plateforme. Cinq sources, Fluent Bit sur chaque nœud, un relais syslog, le SOC d'OpenAI.



Ce qui remonte : Tetragon (hôte), proxy et ModSecurity, dépôt (audit, récupérations distantes), pare-feu de sortie (décisions). Fluent Bit lit les journaux locaux, le relais syslog pousse vers le SOC. Aucun accès en retour.

Quand arrêter l'agent

Les règles sont écrites avant l'exécution. Une alerte décrit, une règle décide.

ARRÊTER L'AGENT, C'EST

Geler l'exécution si possible, préserver journaux et image de la microVM, révoquer son identité de lecture, fermer ses connexions, puis détruire la sandbox.

POURQUOI L'ARRÊTER

Une tentative de sortie n'est pas un bug : l'agent a changé d'objectif. Le laisser continuer, c'est lui laisser le temps de trouver la variante qui marche. Une tentative n'arrête que son exécution ; un agent ne doit pas pouvoir provoquer un arrêt global.

CE QUE FAIT L'AGENT (OU LE SYSTÈME)	CE QU'ON FAIT	QUI EST ARRÊTÉ
Tentative hors liste d'autorisation, attribuée à une exécution (corrélation)	Requête refusée avant envoi. Cette exécution est arrêtée.	Cette exécution
L'agent tente d'écrire, d'administrer ou d'appeler une route non exposée	Requête refusée par le proxy. Cette exécution est arrêtée.	Cette exécution
SSRF réussie, ou perte d'intégrité du dépôt (écriture, jeton, plugin)	Dépôt isolé. Toutes les exécutions qui le partagent sont arrêtées.	Les exécutions liées
Blocage au pare-feu de sortie non attribuable à une exécution	Dépôt mis en sécurité : sortie coupée, exécutions liées en pause.	Les exécutions liées
L'agent sort de sa microVM, ou Tetragon disparaît sur un nœud	Nœud isolé. Ses exécutions arrêtées.	Le nœud entier
Collecteur ou contrôleur muet (chien de garde sans battement)	Fail-closed : sortie coupée, planification désactivée, exécutions en sécurité.	Le périmètre entier
Le modèle d'IA expire, ou l'agent obtient root dans la cible	Rien : analyse perdue ; root dans la cible est le résultat attendu.	Personne

Qui arrête l'agent

Pas le cluster qui l'héberge : il peut être compromis. Un domaine de sécurité séparé décide, et agit par le plan de contrôle cloud.

1

Collecteur d'événements

Reçoit les événements Tetragon du nœud hôte (runtime, réseau), du proxy, du dépôt et du pare-feu. Chaque événement porte l'identifiant de corrélation.

2

Règles et contrôleur d'arrêt

Appliquent les règles de la slide 14. Arrêt ciblé d'une microVM : par le contrôle du CaaS ou de l'hyperviseur, hors sandbox. Perte de confiance dans un nœud : isolation réseau et arrêt de l'instance par le plan de contrôle cloud, depuis un compte distinct, sans l'API Kubernetes. Un chien de garde indépendant attend le battement du contrôleur : sans lui, sortie en refus par défaut et planification désactivée.

3

Analyse par l'IA, en lecture seule

Explique et corrèle pour l'équipe sécurité. Aucun droit d'arrêt. Aucun identifiant. Le modèle n'est jamais dans la boucle de décision.

Test d'indépendance : root sur le nœud, la microVM ou le dépôt ne permet ni de modifier la liste d'autorisation, ni d'empêcher un arrêt, ni de relancer une exécution arrêtée.

Recette

Quatre tests avant le premier agent. On valide l'arrêt, pas l'alerte.

1

Aucune sortie hors liste d'autorisation

Depuis une sandbox root, joindre un récepteur de test : en direct, par SSRF, par une route non exposée, par IP directe, DNS externe ou DoH, redirection HTTP, IPv6 et IPv4-mapped, SNI et Host discordants. Réussite : rien n'arrive.

2

Installer des paquets sans liste préalable

Installer des paquets imprévus : requirements.txt inconnu, apt-get, cargo. Réussite : installation complète, aucune requête hors liste d'autorisation, quotas respectés.

3

Le dépôt ne s'administre pas depuis la sandbox

Tenter écriture, API de configuration, jeton hérité, WebDAV, plugin. Réussite : refus, l'exécution fautive est arrêtée ; les exécutions liées ne sont suspendues qu'en cas de perte d'intégrité confirmée du dépôt.

4

Mise en sécurité

Couper Tetragon sur un nœud : ce nœud est isolé. Couper le collecteur ou le contrôleur : le chien de garde ferme la sortie, exécutions en pause, aucun redémarrage automatique. Faire expirer le modèle d'IA : l'expérience continue, seule l'analyse enrichie est perdue.

À mesurer à chaque test : événements perdus, faux arrêts, délai entre détection et isolement, y compris sous charge.

En résumé

Le dépôt reste. La sortie Internet ouverte disparaît. Quatre décisions.

1

Lecture seule

La sandbox ne peut que lire le dépôt de paquets. Ni écriture, ni administration, ni identifiants partagés.

2

Surface minimale

Le proxy n'expose que la lecture. Une route non exposée n'est pas directement accessible depuis la sandbox.

3

Sortie sur liste d'autorisation

Le dépôt est le seul chemin de données avec accès Internet. Destinations limitées par le pare-feu ; dépôts distants en lecture seule, sans identifiant de publication.

4

Arrêt indépendant

L'arrêt est décidé hors du cluster, par des règles écrites à l'avance. Jamais par le modèle.

Ce qui aurait changé : la SSRF serait restée confinée aux registres autorisés. L'API de configuration et le jeton hérité auraient été hors de portée. La requête vers Modal aurait été bloquée avant envoi. Sous réserve des tests de la slide 16.

Annexe : règles ModSecurity

Le détail des quatre catégories de la slide 12 : exemple de configuration, à valider sur le trafic réel.

EXEMPLE DE RÈGLES MODSECURITY POUR LE PROXY INVERSE

```
# 1. Schémas dangereux : dans l'URI, les arguments et les en-têtes
SecRule REQUEST_URI|ARGS|REQUEST_HEADERS "@rx \b(file|gopher|dict|ftp|ldap|jar|netdoc)://" \
  "id:100010,phase:2,deny,status:403,log,t:none,t:urlDecodeUni,t:lowercase,\
   msg:'SSRF : schéma interdit',tag:'ATTACK-SSRF',severity:'CRITICAL'"

# 2. Cibles internes : métadonnées cloud, boucle locale, RFC 1918, formes encodées
SecRule REQUEST_URI|ARGS|REQUEST_HEADERS "@rx (169\.254\.\d+\.\d+|metadata\.google\.internal|\
   127\.\d+\.\d+\.\d+|0\.0\.0\.0|10\.\d+\.\d+\.\d+|172\.(1[6-9]|2\d|3[01])\.\d+\.\d+|\
   192\.168\.\d+\.\d+|\blocalhost\b|\[?::1\]?|0x7f000001|2130706433)" \
  "id:100011,phase:2,deny,status:403,log,t:none,t:urlDecodeUni,t:lowercase,\
   msg:'SSRF : cible interne ou métadonnées',tag:'ATTACK-SSRF',severity:'CRITICAL'"

# 3. Sur les routes de récupération exposées, aucune URL absolue n'est attendue
SecRule REQUEST_URI|ARGS "@rx ://" \
  "id:100012,phase:2,deny,status:403,log,t:none,t:urlDecodeUni,\
   msg:'URL dans une requête de paquets',tag:'ATTACK-SSRF'"

# 4. En complément : jeu de règles CRS 4, règles SSRF 934100 à 934170, paranoïa 2
# Mise en production : d'abord SecRuleEngine DetectionOnly sur du trafic pip/apt/npm réel ; exclure l'en-tête Referer de la règle 3.
```

Trois corrections par rapport aux règles de départ : inspecter aussi l'URI et les en-têtes ; décoder et normaliser avant de tester ; écrire ::1 explicitement plutôt que [0:]+1, qui bloque toute chaîne « 01 ».

Sources

Faits : rapports OpenAI et Hugging Face, enquête METR. Technologies : documentation des projets.

FAITS ET CHRONOLOGIE

[1] OpenAI, rapport technique d'incident (26 août 2026)
cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face Incident-Technical-Report.pdf

[2] OpenAI, « The Hugging Face incident and the road ahead »
openai.com/index/hugging-face-incident-and-the-road-ahead/

[3] Hugging Face, chronologie technique de l'intrusion
huggingface.co/blog/agent-intrusion-technical-timeline

[4] METR / Redwood Research, enquête indépendante (26 août 2026)
metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/

[5] OpenAI à Black Hat USA 2026 : chronologie détaillée (SSRF, zero-day, WebDAV), compte rendu IANS Research
iansresearch.com/resources/all-blogs/post/security-blog/2026/08/06/black-hat--inside-the-openai-hugging-face-breach

TECHNOLOGIES

[6] ExploitGym, article de l'évaluation de référence (mai 2026)
arxiv.org/html/2605.11086v1

[7] Kata Containers, architecture et protocole de l'agent
github.com/kata-containers/kata-containers

[8] Tetragon, modèle de menace et application des règles
tetragon.io/docs/threat-model/ · tetragon.io/docs/concepts/enforcement/

[9] Kubernetes, règles de réseau (NetworkPolicy)
kubernetes.io/docs/concepts/services-networking/network-policies/

[10] CVE-2026-65925, SSRF par dépôt distant dans JFrog Artifactory (27 juillet 2026)
sentinelone.com/vulnerability-database/cve-2026-65925/

Portée : faits issus des rapports publiés. Architecture de juillet reconstituée. Architecture proposée à valider par la recette.

Merci !

Questions et discussion

Vous cherchez un architecte sécurité cloud expérimenté ? Contactez-moi !

https://www.linkedin.com/in/jcanale13/

Jeremy Canale · Architecte sécurité AWS / Azure

contact@jeremycanale.com · +33 7 87 77 55 92

Disponibilité : France, Belgique, Suisse