Apprentissage par renforcement pour une cybersécurité résiliente

par

Khaled EL JIZI

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE EN GENIE DES TECHNOLOGIES DE L'INFORMATION
AVEC MÉMOIRE
M. Sc. A.

MONTRÉAL, LE 17 DECEMBRE, 2025

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Chamseddine Talhi, directeur de mémoire
Département de génie logiciel et des TI à l'École de Technologie Supérieure

Mme. Hakima Ould-Slimane, codirectrice
Département de Mathématiques et d'Informatique à l'Université du Québec à Trois-Rivières

M. Abdelouahed Gherbi, président du jury
Département de génie logiciel et des TI à l'École de Technologie Supérieure

M. Kaiwen Zhang, examinateur externe
Département de génie logiciel et des TI à l'École de Technologie Supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 10 DECEMBRE, 2025

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je souhaite exprimer ma profonde gratitude pour l'opportunité qui m'a été offerte de poursuivre une maîtrise en génie des technologies de l'information à l'École de technologie supérieure.

Je tiens à remercier chaleureusement **le Professeur Chamseddine Talhi** pour son encadrement, son mentorat et son soutien constant tout au long de ce parcours. Ses conseils avisés et ses commentaires constructifs ont grandement contribué à la réalisation de ce mémoire.

Mes remerciements s'adressent aussi à **la Professeure Hakima Ould Slimane**, ma codirectrice, pour ses conseils précieux et son accompagnement tout au long de ce projet.

J'aimerais également exprimer ma reconnaissance envers **le Dr Hani Sami** pour ses orientations et ses suggestions qui m'ont aidé à surmonter les difficultés rencontrées au cours de ce travail.

Je remercie également l'ensemble des professeurs qui m'ont accompagné durant ce cheminement académique et qui ont contribué à ma réussite.

Enfin, j'aimerais remercier mes amis et ma famille pour leur soutien indéfectible, leur patience et leurs encouragements tout au long de mes études de maîtrise.

Merci à tous pour l'aide et les conseils que vous m'avez apportés.

Apprentissage par renforcement pour une cybersécurité résiliente

Khaled EL JIZI

RÉSUMÉ

À mesure que la dépendance mondiale aux dispositifs de l'IoT et aux technologies interconnectées s'intensifie, les menaces en cybersécurité évoluent à un rythme sans précédent. Les mécanismes de défense traditionnels basés sur des règles, ainsi que les modèles classiques d'apprentissage automatique, montrent une efficacité décroissante face à la nature dynamique et adaptative des cyberattaques modernes. Pour combler cette lacune, ce mémoire propose un cadre innovant fondé sur l'apprentissage par renforcement dans le but d'améliorer la résilience en cybersécurité. Plus précisément, nous développons des agents RL capables de simuler des comportements adverses en lançant des cyberattaques adaptatives et en évaluant la robustesse des réseaux cibles dans différents scénarios.

L'étude débute par une contextualisation de l'apprentissage par renforcement dans le paysage plus large de l'intelligence artificielle appliquée à la cybersécurité. Nous concevons ensuite un modèle RL intégré à un algorithme d'apprentissage méta-indépendant du modèle, permettant une adaptation rapide et une meilleure généralisation à travers des environnements réseaux hétérogènes. Contrairement aux travaux existants, souvent limités à des contextes spécifiques, notre modèle démontre une capacité de généralisation, rendant possible une évaluation plus efficace des défenses sur un large éventail d'infrastructures cybernétiques.

Nos résultats expérimentaux montrent que l'apprentissage par renforcement, renforcé par l'apprentissage méta, constitue une approche évolutive et adaptative pour les opérations de cybersécurité offensives. En permettant aux systèmes d'apprendre de leurs interactions avec l'environnement cybernétique, cette méthode offre une solution intelligente et résiliente pour l'architecture de cybersécurité moderne.

Mots-clés: Apprentissage par renforcement (RL), Cybersécurité, Apprentissage automatique, Détection d'intrusion,Internet of Things(IoT).

Reinforcement Learning For Resilient Cybersecurity

Khaled EL JIZI

ABSTRACT

As global reliance on IoT devices and interconnected technologies intensifies, cybersecurity threats are evolving at an unprecedented pace. Traditional rule-based defense mechanisms and even conventional machine learning models have shown diminishing effectiveness against the dynamic and adaptive nature of modern cyber threats. To address this gap, this thesis introduces a novel RL based framework aimed at enhancing cybersecurity resiliency. Specifically, we propose the use of RL agents to simulate adversarial behavior by launching adaptive cyberattacks and evaluating the robustness of target networks under varying conditions.

The study begins by positioning reinforcement learning within the broader landscape of artificial intelligence in cybersecurity, rule-based approaches. We then design and implement a reinforcement learning model integrated with model-agnostic meta-learning to enable rapid adaptation and generalization across heterogeneous network environments. Unlike previous works that are limited to narrowly scoped settings, our RL model demonstrates generalization capability, enabling more effective penetration testing.

Our experimental results show that reinforcement learning, augmented with meta-learning, provides a scalable and adaptive methodology for offensive and cybersecurity operations. By empowering systems to learn from cyber interactions, this approach contributes a resilient, intelligent layer to modern cybersecurity architectures.

Keywords: Reinforcement Learning (RL), Cybersecurity, Machine Learning, Intrusion Detection, Internet of Things (IoT), Area Under the Curve (AUC).

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AI	Intelligence artificielle
API	Interface de programmation d'applications
APT	Menace persistante avancée (Advanced Persistent Threat)
ATT&CK	Adversarial Tactics, Techniques and Common Knowledge (Cadre MITRE)
AUC	Aire sous la courbe (Area Under the Curve)
Caldera	Plateforme d'émulation d'adversaires développée par MITRE
CPU	Unité centrale de traitement (Central Processing Unit)
DQN	Deep Q-Network
DRL	Apprentissage profond par renforcement (Deep Reinforcement Learning)
GAN	Réseau antagoniste génératif (Generative Adversarial Network)
IoT	Internet des objets (Internet of Things)
MDP	Processus de décision markovien (Markov Decision Process)
MAML	Apprentissage méta modèle-agnostique (Model-Agnostic Meta-Learning)
PPO	Optimisation de politique proximale (Proximal Policy Optimization)
Q-MAML	Q-learning avec méta-apprentissage modèle-agnostique
Q-learning	Apprentissage par renforcement tabulaire basé sur la fonction Q
RL	Apprentissage par renforcement (Reinforcement Learning)
SDN	Réseau défini par logiciel (Software-Defined Network)
UFW	Pare-feu Uncomplicated Firewall (Linux)
VM	Machine virtuelle (Virtual Machine)
MDP	Processus de Décision de Markov (Markov Decision Process)

LISTE DES SYMBOLES ET UNITÉS DE MESURE

α	Taux d'apprentissage
γ	Facteur d'actualisation
ϵ	Taux d'exploration
r	Récompense immédiate
s	État courant de l'environnement
a	Action choisie par l'agent
$Q(s, a)$	Valeur de la fonction Q pour l'état s et l'action a
Φ	Potentiel de récompense
K	Nombre d'épisodes d'adaptation interne
M	Nombre d'itérations méta d'entraînement
η	Taux d'apprentissage méta

INTRODUCTION

Le domaine de la cybersécurité est caractérisé par une compétition incessante et asymétrique, souvent décrite comme une véritable course aux armements. Les mécanismes conventionnels, principalement fondés sur des signatures ou des règles statiques, peinent à suivre l'évolution rapide des attaques zéro-day, des stratégies polymorphes ou encore des comportements émergents liés à l'automatisation offensive. Dans ce contexte, un changement de paradigme vers des approches proactives, adaptatives et autonomes s'impose. L'apprentissage par renforcement apparaît comme une avenue prometteuse, permettant à un agent d'interagir avec un environnement cybernétique, d'explorer diverses stratégies, d'évaluer leurs conséquences et d'optimiser progressivement ses décisions.

Le RL a démontré son efficacité dans plusieurs domaines et attire de plus en plus l'attention dans la cybersécurité. Sa capacité à apprendre par essais-erreurs sans dépendre de données préalablement étiquetées, le rend particulièrement adapté aux environnements dynamiques, partiellement observables et hostiles. Huang *et al.* (2022) soulignent notamment son utilité pour mettre en place des défenses dynamiques, telles que la déception ou les mécanismes de Moving Target Defense Huang *et al.* (2022).

La sophistication des adversaires transforment la cybersécurité en un champ de bataille où les mécanismes statiques montrent leurs limites. Les approches classiques d'apprentissage automatique restent dépendantes de données labellisées, manquent de flexibilité et généralisent difficilement à des scénarios inédits. De ce fait, la recherche se tourne vers des modèles plus autonomes, capables de s'adapter rapidement à des situations nouvelles.

Plusieurs travaux ont tenté d'intégrer le RL avec des environnements d'émulation réseau, des plateformes d'automatisation offensive et des agents intelligents. Parmi eux, CyGIL (Li *et al.* (2021)) propose une architecture liant un environnement cybernétique, la plateforme MITRE CALDERA et un agent RL. Toutefois, ces approches demeurent souvent partielles, peu

instrumentées ou difficilement reproductibles, limitant l'évaluation empirique et la comparaison systématique.

Le présent mémoire s'inscrit dans cette continuité tout en élargissant la portée méthodologique et expérimentale. Nous proposons une approche plus complète, opérationnelle et reproductible comprenant :

- la conception d'un environnement d'émulation réaliste reposant sur un réseau virtualisé sous VMware et intégrant MITRE CALDERA ;
- la formalisation d'un environnement RL compatible OpenAI Gym avec un espace d'actions opérationnel, un espace d'observations structuré et une fonction de récompense adaptée aux opérations offensives ;
- l'évaluation de méthodes d'apprentissage avancées, notamment le méta-apprentissage à travers un agent Q-MAML comparé à un agent Q-learning standard.

Cette plateforme a été conçue pour étudier la capacité d'agents RL à reconnaître, anticiper et contrer des cybermenaces simulées, en particulier dans des scénarios d'attaque modulaires inspirés du cadre MITRE ATT&CK. En reliant des attaques réelles issues de la Red Team automatisée de Caldera à un agent RL, nous élaborons un cadre expérimental permettant de mesurer l'adaptabilité, la robustesse et la capacité de généralisation des politiques apprises.

Malgré les avancées du RL, une limite persistante demeure : les agents traditionnels généralisent mal hors de leur distribution d'entraînement et deviennent fragiles face à des variations de tactiques adverses. Pourtant, dans un contexte opérationnel, un agent de sécurité doit atteindre une résilience réelle, c'est-à-dire s'adapter rapidement à des menaces diverses et inédites. Le méta-apprentissage répond à cette problématique en permettant une adaptation rapide à de nouveaux scénarios.

Ce mémoire vise ainsi à répondre aux deux questions de recherche suivantes :

1. **Comment concevoir une plateforme d'émulation réaliste, reproductible et suffisamment instrumentée pour entraîner des agents RL dans un contexte d'opérations offensives ?**
2. **Dans quelle mesure le méta-apprentissage, via une approche Q-MAML, permet-il à un agent d'améliorer sa capacité d'adaptation et de généralisation face à de nouveaux scénarios d'attaque ?**

Pour guider le lecteur à travers les différentes étapes de notre démarche, ce mémoire est structuré comme suit :

- **Chapitre 1 — Concept théorique** — Présente les notions fondamentales nécessaires à la compréhension du mémoire : résilience cyber, cybersécurité, modèles d'apprentissage par renforcement et bases mathématiques comme les MDP et les équations de Bellman.
- **Chapitre 2 — Revue de la littérature** — Analyse critique des principales plateformes d'entraînement RL pour la cybersécurité, leurs contributions, leurs limites, et met en évidence les écarts méthodologiques qui motivent la conception de notre propre cadre expérimental.
- **Chapitre 3 — Approche proposée** — Décrit l'architecture du système, la topologie réseau, l'intégration avec Caldera, la définition des espaces d'actions et d'observations, les contraintes spécifiques à chaque scénario et la fonction de récompense.
- **Chapitre 4 — Expérimentations et résultats** — Présente le protocole expérimental, les métriques utilisées, les résultats obtenus avec Q-learning et Q-MAML, et analyse la capacité de généralisation inter-scénarios.

En somme, ce mémoire propose une plateforme d'émulation fidèle, un cadre d'apprentissage rigoureux et une étude comparative incluant le méta-apprentissage, afin de jeter les bases d'agents

capable de s’adapter rapidement à des attaques variées et imprévues, contribuant ainsi à une résilience cybernétique accrue.

CHAPITRE 1

CONCEPT THÉORIQUE

1.1 La résilience cyber

Un mécanisme de cybersécurité résilient comporte quatre étapes : la préparation, la prévention, la réponse et la récupération.Huang *et al.* (2022).

Les quatre étapes de la résilience cyber mentionnées dans Huang *et al.* (2022) sont celles qui nous guideront dans ce mémoire, car nous allons aborder l'une de ces étapes, et notre modèle proposé sera une solution qui aide à la mise en œuvre de cette étape.

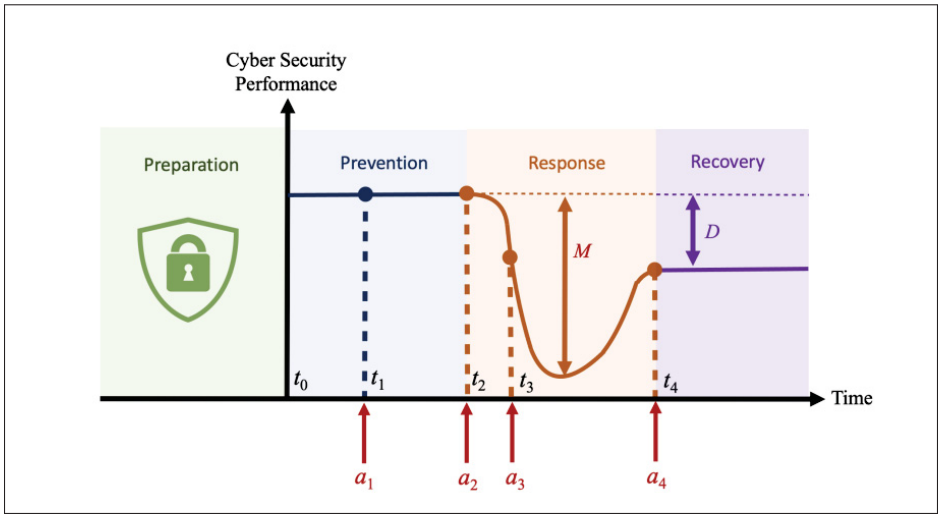


Figure 1.1 Les différentes étapes d'un mécanisme cyber-résilient
Huang *et al.* (2022)

Dans la première étape, nous évaluons les risques présents dans le système cybernétique et créons des politiques appropriées Huang *et al.* (2022). Dans la deuxième étape, nous implémentons les politiques conçues pour protéger notre système Huang *et al.* (2022). Pour la troisième étape, il est impératif d'acquérir des informations sur l'attaquant afin de reconfigurer notre système cybernétique et minimiser le risque autant que possible après un échec dans l'étape de prévention Huang *et al.* (2022). En ce qui concerne la quatrième étape, notre objectif est de réduire les

effets secondaires après l'attaque et de restaurer les performances du système autant que possible Huang *et al.* (2022).

1.2 Cybersécurité

La cybersécurité regroupe l'ensemble des pratiques, politiques et technologies visant à protéger les systèmes informatiques contre les menaces pouvant compromettre la confidentialité, l'intégrité et la disponibilité de l'information, Standards & Technology (2024). Elle constitue aujourd'hui un pilier fondamental de la transformation numérique et de la protection des infrastructures critiques, intégrant la gestion du risque, la gouvernance, et la résilience opérationnelle face à des menaces en constante évolution Admass, Munaye & Diro (2024).

Dans le contexte de ce mémoire, la cybersécurité est abordée sous un angle expérimental : l'objectif est de simuler des attaques et des contre-mesures afin d'évaluer la résilience d'un réseau et la capacité d'un agent d'apprentissage à s'y adapter Kott, Weisman & Vandekerckhove (2022). Les sous-sections suivantes présentent les concepts fondamentaux liés à la sécurité de l'information : les politiques de sécurité, les attaques, les mécanismes de défense et la résilience des systèmes.

1.2.1 Politiques de sécurité

Les politiques de sécurité constituent la base organisationnelle et technique de la cybersécurité. Elles définissent les règles, les procédures et les responsabilités relatives à la protection des actifs informationnels Magnusson, Iqbal, Elm & Dalipi (2025). Une politique de sécurité précise les niveaux d'accès, les obligations de surveillance, les conditions de mise à jour des systèmes ainsi que les comportements autorisés sur le réseau Rostami, Karlsson, Kolkowska & Gao (2025).

Dans les environnements simulés, ces politiques se traduisent par des configurations et des restrictions implémentées au niveau des hôtes virtuels : segmentation des sous-réseaux, filtrage des ports, privilèges utilisateurs ou journalisation obligatoire. Elles encadrent les interactions

possibles entre l'agent et l'environnement d'apprentissage, en définissant ce qui est considéré comme un comportement acceptable ou malveillant.

Dans notre cadre expérimental, chaque scénario Caldera s'appuie implicitement sur une politique de sécurité simulée. Par exemple, un pare-feu actif, un service critique protégé, ou des privilèges d'accès restreints constituent autant de règles que l'agent doit contourner ou préserver selon son rôle.

1.2.2 Attaques et menaces

Une **attaque informatique** désigne toute action intentionnelle visant à compromettre la sécurité d'un système d'information. Elle cherche à altérer l'un des trois piliers de la sécurité :

- La *confidentialité* : accès non autorisé aux données.
- L'*intégrité* : modification, suppression ou corruption d'informations.
- La *disponibilité* : interruption ou dégradation de services.

Les attaques peuvent prendre de nombreuses formes, allant de l'exploitation de failles logicielles à l'ingénierie sociale. Elles sont souvent classées selon leur objectif ou leur mode opératoire :

- les **attaques passives**, qui consistent à observer ou intercepter des communications sans modifier le système ;
- les **attaques actives**, qui impliquent une altération directe du système (ex. injection de code, sabotage, déni de service) ;
- les **attaques internes**, menées par un acteur disposant déjà d'un accès légitime au réseau ;
- les **attaques externes**, provenant d'un attaquant situé en dehors du périmètre de sécurité.

Ces modèles conceptuels facilitent la compréhension, la détection et la reproduction des comportements adverses dans un cadre d'expérimentation ou d'analyse
Pashamokhtari, Batista & Habibi Gharakheili (2023).

1.2.3 Mécanismes de défense et contre-mesures

Les mécanismes de défense regroupent l'ensemble des dispositifs techniques et procéduraux destinés à prévenir, détecter et corriger les intrusions. Ils constituent la composante opérationnelle de la politique de sécurité Huang *et al.* (2022). Parmi les principaux mécanismes, on retrouve :

- les **pare-feux** pour contrôler les flux réseau ;
- les **systèmes de détection d'intrusion** pour signaler les comportements suspects ;
- les **protocoles d'authentification et de chiffrement** pour protéger l'accès et les communications ;
- les **mécanismes de récupération** tels que le redémarrage automatique ou la restauration d'état.

Dans un cadre expérimental, ces défenses peuvent être activées ou désactivées par l'agent bleu afin d'observer leur impact sur la dynamique de l'attaque et sur la disponibilité du système. L'interaction entre les capacités offensives et défensives constitue un jeu d'adaptation qui se prête naturellement à l'approche de l'apprentissage par renforcement.

Ainsi, la gestion des contre-mesures n'est plus statique : elle devient une fonction d'optimisation où le défenseur apprend à réagir de manière optimale aux tactiques adverses.

1.2.4 Résilience des systèmes

La résilience désigne la capacité d'un système à résister, absorber et se rétablir après une perturbation. Dans le domaine de la cybersécurité, elle se traduit par la capacité à maintenir les fonctions essentielles malgré les attaques, ou à les restaurer rapidement après compromission Weisman *et al.* (2023).

Les approches traditionnelles évaluent la résilience par la redondance, la robustesse et les plans de reprise après sinistre. Cependant, l'émergence des environnements dynamiques et autonomes nécessite des méthodes plus adaptatives. L'intégration de l'apprentissage par renforcement (RL) permet d'étudier cette adaptabilité : un agent peut apprendre à réagir à des attaques en fonction de l'état courant du réseau et de l'efficacité passée de ses actions Huang *et al.* (2022).

Ces mesures traduisent la capacité du système à s'adapter, à se stabiliser et à maintenir un niveau de service acceptable face à des attaques répétées.

1.3 L'apprentissage par renforcement

L'apprentissage par renforcement est une branche de l'intelligence artificielle qui étudie la manière dont un agent autonome peut apprendre à prendre des décisions optimales par interaction avec un environnement. Contrairement à l'apprentissage supervisé, où les données d'entraînement sont étiquetées, le RL repose sur un processus d'essais et d'erreurs. L'agent observe un état s, choisit une action a, reçoit une récompense r ou une loss l en retour, puis met à jour sa politique afin de maximiser la somme des récompenses cumulées Yang, Acuto, Zhou & Wojtczak (2024).

L'apprentissage par renforcement s'impose comme un paradigme central pour la prise de décision adaptative, autonome et résiliente dans les systèmes de cybersécurité et les environnements autonomes. En interagissant continuellement avec leur environnement, les agents RL apprennent des politiques optimales leur permettant de répondre aux menaces, d'optimiser la défense et de s'adapter à des contextes complexes où les approches statiques atteignent leurs limites Huang *et al.* (2022). Les sous-sections suivantes illustrent plusieurs domaines d'application du RL, suivies d'une discussion sur son étendue dans d'autres secteurs.

1.3.1 Processus de Décision de Markov

L'apprentissage par renforcement repose sur le cadre formel du **Processus de Décision de Markov**, qui permet de modéliser les problèmes de prise de décision séquentielle. Un MDP est généralement défini par un quintuplet $(\mathcal{S}, \mathcal{A}, \rho, \mathcal{P}, \gamma)$ Wang *et al.* (2024), où :

- $\mathcal{S}$ est l'ensemble des états observables de l'environnement.
- $\mathcal{A}$ est l'ensemble des actions possibles, discrètes ou continues.
- $\rho : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ est la fonction de récompense immédiate.
- $\mathcal{P}(s' \mid s, a)$ est la fonction de transition, qui donne la probabilité de passer à l'état s' après avoir exécuté l'action a dans l'état s.

- $\gamma \in [0, 1]$ est le facteur d'actualisation qui détermine l'importance des récompenses futures.

L'hypothèse centrale du MDP est la propriété de *markovianité* : l'état futur dépend uniquement de l'état actuel et de l'action exécutée, et non de la trajectoire complète Wang *et al.* (2024). Dans ce contexte, l'objectif est d'apprendre une politique $\pi : \mathcal{S} \rightarrow \mathcal{A}$ qui maximise la somme des récompenses cumulées actualisées :

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Cette formulation permet de concevoir des agents capables de prendre des décisions optimales dans un environnement dynamique. Dans les cas où l'environnement n'est pas complètement observable, des variantes telles que les *Partially Observable Markov Decision Processes* peuvent être employées Wang *et al.* (2024).

Les MDP fournissent ainsi une base mathématique robuste permettant de formaliser les interactions entre un agent autonome et son environnement, en capturant à la fois la dynamique de transition, les incitations via les récompenses, et le comportement optimal attendu. Cette structure est particulièrement utile dans les contextes complexes comme la cybersécurité, où les états peuvent représenter des configurations réseau, les actions des choix d'attaque ou de défense, et les récompenses des mesures de succès ou d'échec dans un scénario donné.

Il est important de noter que la recherche actuelle explore également des extensions aux MDP classiques pour mieux refléter les contraintes du monde réel, notamment :

- **MDP partiellement observables (POMDP)** : utilisés lorsque l'agent ne dispose que d'une information incomplète ou bruitée sur l'état réel du système.
- **MDP stochastiques** : qui incorporent de l'aléa non seulement dans les transitions, mais aussi dans les récompenses.

- **MDP à plusieurs agents** : pour modéliser des environnements compétitifs ou coopératifs avec plusieurs agents intelligents.

Dans le cadre de l'apprentissage par renforcement profond, les MDP servent de base à des algorithmes tels que le Deep Q-Network ou les méthodes acteur-critique, qui utilisent des réseaux de neurones pour approximer les politiques et les fonctions de valeur dans des environnements de grande dimension Wang *et al.* (2024).

1.3.2 Équations de Bellman

La résolution des MDP repose fortement sur les **équations de Bellman**, qui expriment les fonctions de valeur en forme récursive. La fonction de valeur d'un état sous une politique π est définie comme :

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} R_{t+k+1} \mid s_t = s\right]$$

Cette équation peut être réécrite sous forme récursive :

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi, s'\sim\mathcal{P}}\left[R(s,a) + \gamma V^{\pi}(s')\right]$$

De manière analogue, la **fonction de valeur état-action** (ou fonction Q) est définie comme :

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[R_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a\right]$$

Les équations de Bellman sont centrales pour les algorithmes d'apprentissage par renforcement, car elles permettent de calculer les fonctions de valeur et d'améliorer les politiques en conséquence Wang *et al.* (2024).

Ces équations expriment une relation fondamentale : la valeur d'un état ou d'une action peut être estimée en fonction de la récompense immédiate et de la valeur des états futurs, pondérés par leur probabilité d'occurrence. Cela permet d'établir un principe d'optimisation récursive, qui est exploité dans les algorithmes de type *Dynamic Programming*, *Monte Carlo*, et *Temporal Difference* Wang *et al.* (2024).

Dans le contexte de la cybersécurité, l'utilisation des équations de Bellman permet de raisonner sur la *valeur stratégique* d'un état réseau donné : un état vulnérable mais qui mène rapidement à une récompense élevée peut avoir une valeur supérieure à un état temporairement sûr. Cela guide les décisions de l'agent de manière dynamique et adaptative, ce qui est crucial dans des environnements hostiles ou changeants.

1.3.3 Q-learning tabulaire

Une approche fondatrice de l'apprentissage par renforcement est le **Q-learning tabulaire**, qui repose sur la mise à jour d'une table de valeurs $Q(s, a)$ représentant l'utilité d'une action a dans un état s. Ce paradigme simple mais puissant permet à un agent d'apprendre une stratégie optimale par itérations successives, même dans des environnements non modélisés. Le **Q-learning** est l'un des algorithmes les plus fondamentaux et les plus largement utilisés Wang *et al.* (2024). Le Q-learning met à jour la fonction $Q(s, a)$ en fonction de la récompense immédiate et de la meilleure estimation de la valeur future :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

où α est le taux d'apprentissage, γ est le facteur d'actualisation, et $\max_{a'} Q(s_{t+1}, a')$ représente la meilleure estimation de la récompense future.

L'approche Q-learning est dite *hors-politique*, car elle apprend la politique optimale indépendamment de la politique utilisée pour explorer l'environnement. Cela permet une utilisation efficace des expériences passées grâce à des techniques comme la *relecture d'expérience* (*experience*

replay) Wang *et al.* (2024). Ce principe est crucial dans les algorithmes profonds tels que le **Deep Q-Network (DQN)**, qui approximent la fonction Q à l'aide de réseaux de neurones profonds, permettant ainsi de gérer des espaces d'états et d'actions de haute dimension.

Dans notre approche, Q-learning joue un rôle fondamental en tant que mécanisme d'optimisation des actions dans un environnement cybernétique simulé. Il permet à l'agent d'identifier les séquences d'actions les plus efficaces pour atteindre ses objectifs tout en s'adaptant aux politiques dynamiques de défense.

Dans le domaine de la cybersécurité, cette approche a montré son efficacité pour entraîner des agents à identifier et exploiter des vulnérabilités dans des environnements simulés. Cai *et al.* (2024) proposent une méthode basée sur Q-learning tabulaire pour entraîner des agents cybernétiques autonomes capables de généraliser à de nouveaux scénarios en s'appuyant sur une base de vulnérabilités connue et des environnements simulés variés générés dynamiquement.

1.3.4 Model-Agnostic Meta-Learning

Le méta-apprentissage regroupe des méthodes où l'on n'entraîne plus un agent sur une seule tâche, mais sur une distribution de tâches, afin d'apprendre une initialisation ou une règle de mise à jour capable de s'adapter rapidement à une nouvelle tâche. Dans ce cadre, les approches dites model-agnostic n'imposent pas de structure particulière au modèle : elles appliquent quelques pas de gradient sur chaque tâche, puis ajustent les paramètres initiaux en fonction de ces adaptations. Reptile Nichol, Achiam & Schulman (2018) est une variante simplifiée de MAML où l'on remplace le méta-gradient explicite par une simple interpolation entre les paramètres partagés et ceux obtenus après adaptation sur une tâche. Cette mise à jour rapproche alors les paramètres globaux des solutions propres à chaque tâche, ce qui fournit une initialisation réutilisable sur de nouvelles tâches. Dans notre mémoire, les différents scénarios Caldera sont justement considérés comme des tâches, ce qui motive l'emploi d'un schéma de méta-apprentissage de type Reptile.

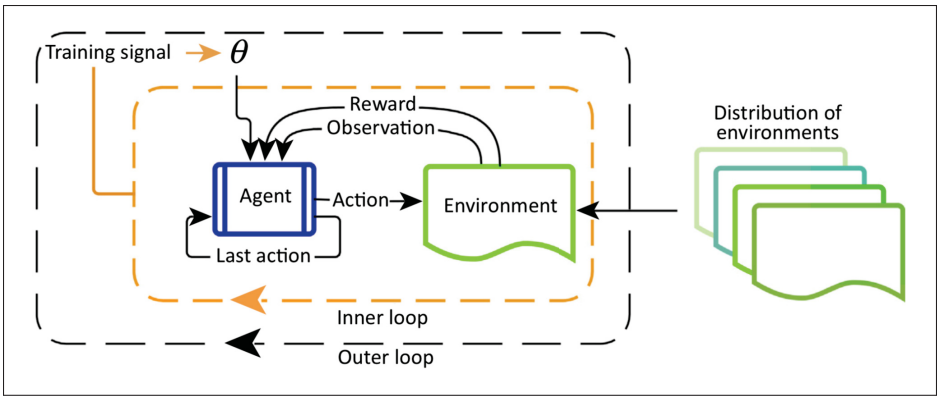


Figure 1.2 Schéma du méta-apprentissage par renforcement, illustrant les boucles interne et externe de l'entraînement Botvinick *et al.* (2019)

La Figure 1.2 illustre le cadre de méta-apprentissage par renforcement. L'agent interagit avec un environnement spécifique, recevant des observations et des récompenses, et ajustant ses actions en conséquence. Cette interaction constitue la boucle interne, où l'apprentissage rapide sur une tâche unique se produit. En parallèle, une boucle externe adapte les paramètres initiaux à partir d'une distribution d'environnements, en intégrant les signaux de mise à jour issus des adaptations locales. Ce schéma correspond à l'approche Reptile, dans laquelle les paramètres globaux sont progressivement rapprochés des solutions propres à chaque tâche, sans calcul explicite de méta-gradient.

L'algorithme **Model-Agnostic Meta-Learning** proposé par Finn *et al.* (2017) constitue une avancée majeure en apprentissage par transfert, MAML introduit un cadre général compatible avec tout modèle entraîné par descente de gradient.

Dans l'approche MAML, proposée par Finn et al. Finn *et al.* (2017), l'agent effectue une mise à jour interne par descente de gradient sur chaque tâche, puis un ajustement externe des paramètres initiaux afin de favoriser une adaptation rapide aux nouvelles tâches. La descente de gradient est une méthode d'optimisation itérative utilisée pour ajuster les paramètres d'un modèle d'apprentissage en minimisant une fonction de coût. Cette méthode permet au modèle

de réduire progressivement l'erreur en apprenant les paramètres qui minimisent la fonction de perte.

L'idée centrale consiste à apprendre une **initialisation partagée des paramètres** telle qu'un petit nombre d'étapes de gradient, à partir d'un jeu de données limité provenant d'une nouvelle tâche, puisse conduire à des gains de performance significatifs.

L'un des points forts de MAML réside dans son caractère *agnostique au modèle* et sa large applicabilité à l'apprentissage supervisé, non supervisé et par renforcement.

Algorithm 3 MAML for Reinforcement Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using f_θ in $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
11: **end while**

Figure 1.3 Schéma de MAML appliqué à l'apprentissage par renforcement
Finn *et al.* (2017)

L'article formalise cette approche à travers trois instanciations : l'apprentissage supervisé, la régression, et surtout l'**apprentissage par renforcement**, décrit dans l'**Algorithme 3**, que nous illustrons dans la Figure 1.3 ci-dessous. Dans cette formulation en apprentissage par

renforcement, chaque tâche est modélisée comme un **processus de décision markovien (MDP)**, et l'objectif est d'entraîner une politique capable de s'adapter rapidement via des mises à jour de gradient de politique.

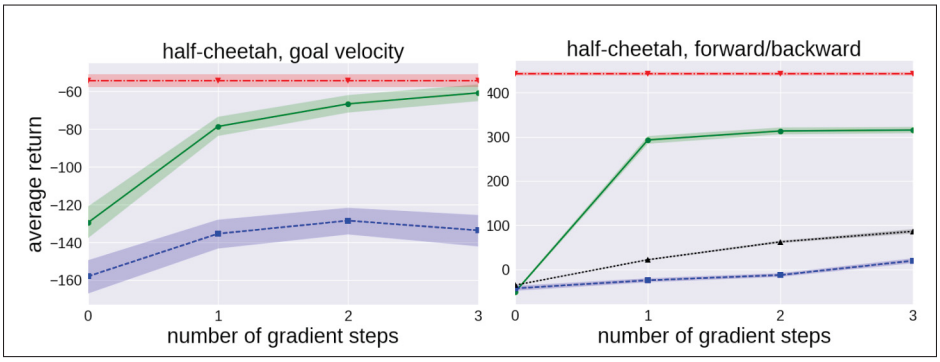


Figure 1.4 Résultats de l'apprentissage par renforcement pour les tâches de locomotion *half-cheetah* et *ant*, les tâches étant illustrées à droite
Finn *et al.* (2017)

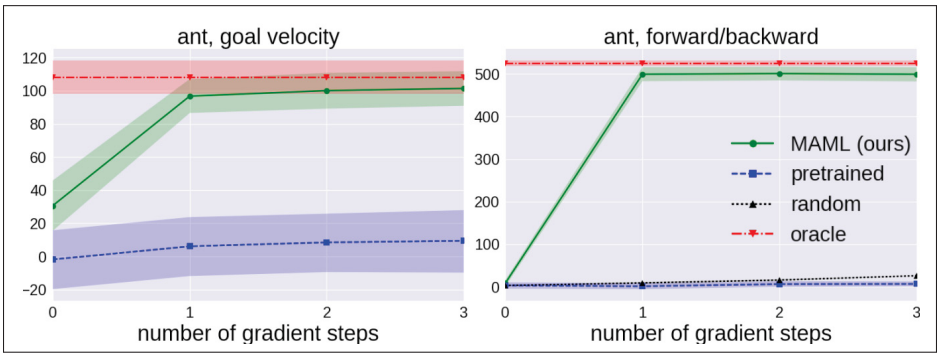


Figure 1.5 Résultats complémentaires de l'apprentissage par renforcement pour les tâches de locomotion *half-cheetah* et *ant*
Finn *et al.* (2017)

Dans ce mémoire, nous utilisons un schéma de méta-apprentissage model-agnostic avec l'objectif d'apprendre une bonne initialisation de politique qui puisse s'adapter rapidement à un nouveau scénario avec quelques itérations d'apprentissage seulement. Reptile, proposé par OpenAI, est une variante simple de ce cadre : il consiste à entraîner temporairement le modèle sur une tâche,

puis à ajuster les paramètres initiaux dans la direction de ces paramètres adaptés. En pratique, Reptile offre un comportement proche de MAML, tout en restant plus léger à implémenter et à optimiser, ce qui en fait un choix naturel pour notre banc d'essai multi-scénarios basé sur Caldera Nichol *et al.* (2018).

1.4 Domaines d'application de l'apprentissage par renforcement en cybersécurité

1.4.1 Détection d'hameçonnage et reconnaissance adaptative des menaces

Tang & Mahmoud (2022) ont proposé un cadre d'apprentissage profond pour la détection de sites d'hameçonnage, démontrant la capacité des modèles intelligents à s'adapter de manière autonome à l'évolution des comportements malveillants. Bien que cette approche ne soit pas exclusivement fondée sur le RL, elle illustre la logique d'apprentissage adaptatif et itératif, fondement même du RL, où les agents améliorent continuellement leurs performances grâce à des retours d'expérience successifs.

1.4.2 Sécurisation du Mobile Edge Caching et stratégies anti-brouillage

Xiao *et al.* (2018) ont appliqué le Q-learning et les réseaux de neurones profonds (DQN) à la sécurisation des systèmes de *mobile edge caching* (MEC) dans les réseaux 5G. Leur approche a permis de contrer les attaques de brouillage, d'usurpation et de déni de service grâce à des agents capables d'optimiser dynamiquement les paramètres de communication et les seuils d'authentification. Cette adaptation en temps réel démontre la capacité du RL à offrir une défense autonome et intelligente dans des environnements distribués à ressources limitées.

1.4.3 Apprentissage profond par renforcement pour la conduite autonome sécurisée

Rasheed, Hu & Zhang (2020) ont proposé une architecture d'apprentissage profond par renforcement combinant des réseaux de mémoire à long court terme et des réseaux antagonistes génératifs afin d'assurer la sécurité et la sûreté des véhicules autonomes. Ce modèle hybride

LSTM-GAN permet au véhicule de détecter, prédire et répondre en temps réel à des menaces cyber-physiques telles que l'usurpation ou la falsification de capteurs. L'intégration de l'apprentissage temporel et de la génération adversariale améliore la robustesse décisionnelle dans des conditions de conduite incertaines et hostiles, confirmant l'efficacité du DRL pour la cybersécurité des systèmes embarqués intelligents.

1.4.4 Véhicules autonomes et apprentissage profond pour la résilience

Giannaros *et al.* (2023) ont examiné les vulnérabilités des véhicules autonomes et souligné le rôle du *Deep Reinforcement Learning* (DRL) pour renforcer leur résilience. Les approches DRL permettent aux véhicules d'ajuster leurs décisions en temps réel face à des manipulations de capteurs, des attaques LiDAR ou des intrusions réseau. Cette capacité d'adaptation renforce les principes fondamentaux de la sécurité — confidentialité, intégrité et disponibilité — dans des contextes opérationnels critiques.

1.4.5 Réponse autonome aux incidents dans les systèmes militaires

Madsen, Grov, Mancini, Baksaas & vald Å slaugson Sommervoll (2024) ont démontré une mise en œuvre réaliste du RL pour la défense cybernétique autonome dans les systèmes physiques. Leur travail a appliqué les algorithmes de Q-learning et de DQN à des véhicules militaires terrestres autonomes afin de permettre une réponse indépendante aux incidents en conditions adverses. Le cadre *sim-to-real* proposé a validé le transfert de politiques apprises en simulation vers le monde réel, confirmant la faisabilité opérationnelle du RL pour la cybersécurité de défense.

1.4.6 Méthodes à gradient de politique pour la défense autonome

Wang & Dechene (2024) ont étudié les méthodes à gradient de politique, telles que A2C et PPO, appliquées à la cybersécurité. Ces approches optimisent directement la fonction de politique, ce qui favorise un apprentissage continu et une meilleure stabilité dans des espaces d'action vastes

et dynamiques. Elles complètent les approches à base de valeur en permettant une adaptation constante face à des menaces non stationnaires.

1.4.7 Applications étendues de l'apprentissage par renforcement

Au-delà de ces exemples représentatifs, le RL a été appliqué à de nombreux domaines : détection d'intrusion, confinement de logiciels malveillants, routage sécurisé, protection des systèmes industriels et authentification adaptative. Dans chacun de ces cas, le RL permet aux systèmes de tirer parti de l'expérience pour détecter, anticiper et contrer les menaces de manière autonome. Cette diversité d'applications confirme le rôle du RL comme pilier des systèmes cyber-physiques intelligents et résilients de prochaine génération.

CHAPITRE 2

REVUE DE LA LITTÉRATURE

Dans ce chapitre, nous présentons les travaux les plus significatifs dans le domaine ciblé par notre étude, à savoir l'application de l'apprentissage par renforcement à la cybersécurité. Cette revue met en lumière les environnements de simulation, les cadres d'expérimentation et les modèles d'agents RL utilisés dans la littérature. Nous analysons plusieurs plateformes clés en identifiant leurs contributions, leurs limites, ainsi que leur pertinence par rapport à notre approche. Enfin, une étude comparative synthétise les caractéristiques de ces travaux afin de dégager les éléments différenciateurs qui motivent la conception de notre propre cadre expérimental.

2.1 CyberBattleSim

Dans l'article Kunz *et al.* (2023), les auteurs améliorent et testent deux algorithmes d' apprentissage par renforcement dans **Microsoft's CyberBattleSim Microsoft (2021)**, un environnement d'entraînement initialement conçu pour entraîner des agents rouges. Les auteurs l'ont amélioré afin de pouvoir entraîner des agents bleus simultanément, en ajoutant une classe *defender* à l'environnement existant.

Le nouvel environnement amélioré s'appelle **Marlon : Multi-Agent Training ON CyberBattleSim**. Les auteurs ont utilisé **Actor Critic** et **Proximal Policy Optimisation** pour tester l'environnement.

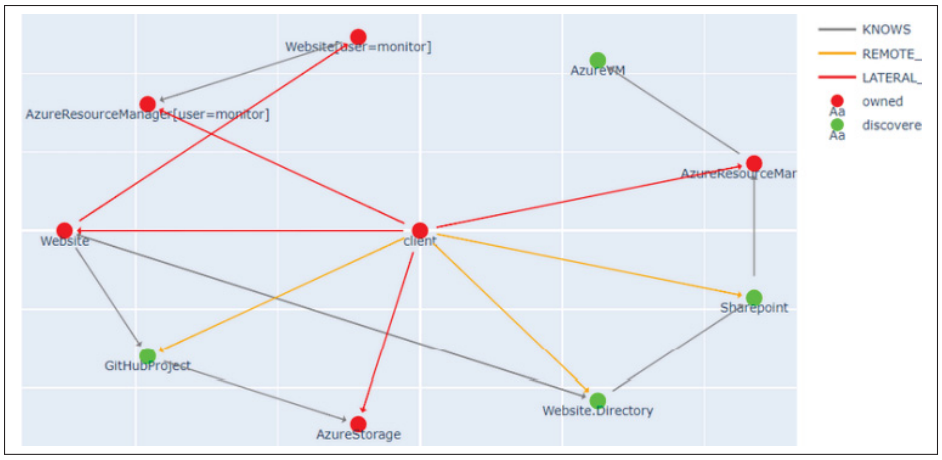


Figure 2.1 Exemple de topologie réseau dans CyberBattleSim
Kunz *et al.* (2023)

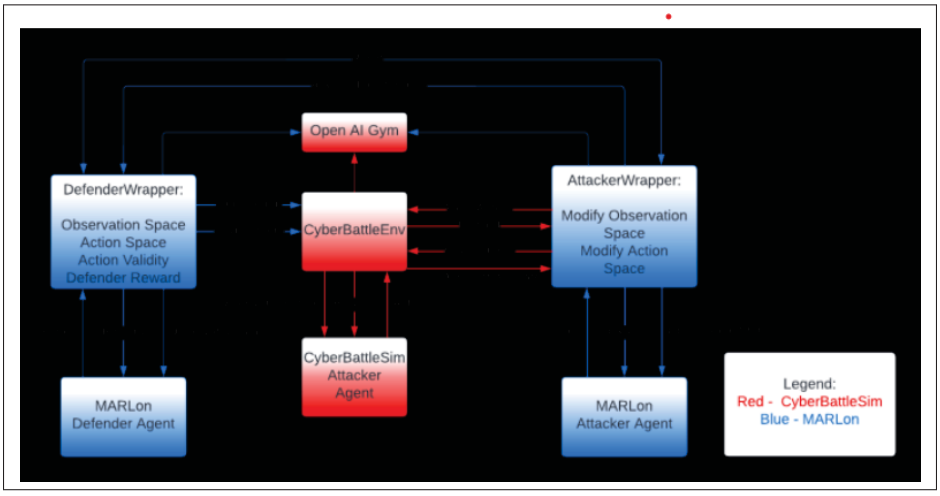


Figure 2.2 Diagramme conceptuel de haut niveau de MARLon
Kunz *et al.* (2023)

Voici un schéma expliquant comment **Marlon** a été construit sur **CyberBattleSim**, en ajoutant des agents bleus. Les améliorations clés incluent :

- Une nouvelle classe `DefenderWrapper` qui permet aux agents bleus d'opérer dans le cadre du apprentissage par renforcement, en fournissant des espaces d'observation et d'action conformes aux standards du RL.
- L'espace d'action du *defender* inclut des actions telles que le reformatage des nœuds, le blocage ou l'autorisation du trafic, ainsi que la gestion des services.

Les agents bleus, lorsqu'ils sont entraînés conjointement avec les agents rouges, ont démontré des défenses plus solides. Plus précisément, les agents bleus entraînés avec l'algorithme **PPO** (appelés *PPO MARL Blue Agents*) étaient plus efficaces pour contrer les agents rouges sophistiqués que les agents bleus entraînés de manière isolée.

Pendant le processus d'entraînement des deux types d'agents, lorsqu'ils étaient entraînés seuls :

- Les agents rouges ont obtenu de bien meilleures performances, puisqu'aucun agent bleu n'était là pour les stopper.
- Les agents bleus ont mieux performé, car ils étaient confrontés à des scénarios en temps réel.

L'article apporte une contribution significative en étendant **CyberBattleSim** pour permettre l'entraînement conjoint des agents rouges et bleus. Il démontre que les agents bleus entraînés de cette manière surpassent ceux entraînés isolément. Cependant, cela a également mené à une nouvelle découverte : chaque fois que nous entraînons des agents rouges et bleus simultanément, l'un d'eux sera sacrifié, car l'un surpassera l'autre. Ce sont généralement les agents rouges qui sont affectés, puisque la plupart de leurs actions sont bloquées par les agents bleus.

2.2 Atmos

Dans l'article intitulé *Autonomous Threat Mitigation in SDN using Reinforcement Learning* Akbari *et al.* (2020), les auteurs présentent le cadre **ATMoS**, qui permet la conception rapide d'applications de Reinforcement Learning pour la sécurité des réseaux, ainsi qu'une formulation concrète de l'atténuation des menaces comme un problème d'apprentissage par renforcement Akbari *et al.* (2020). Le cadre **ATMoS** exploite l'infrastructure **SDN** et le profilage du comportement des hôtes. Il utilise **Neural Fitted Q-learning** pour détecter et atténuer des

APT furtifs, et introduit également un espace d'action basé sur un réseauvirtuel Akbari *et al.* (2020).

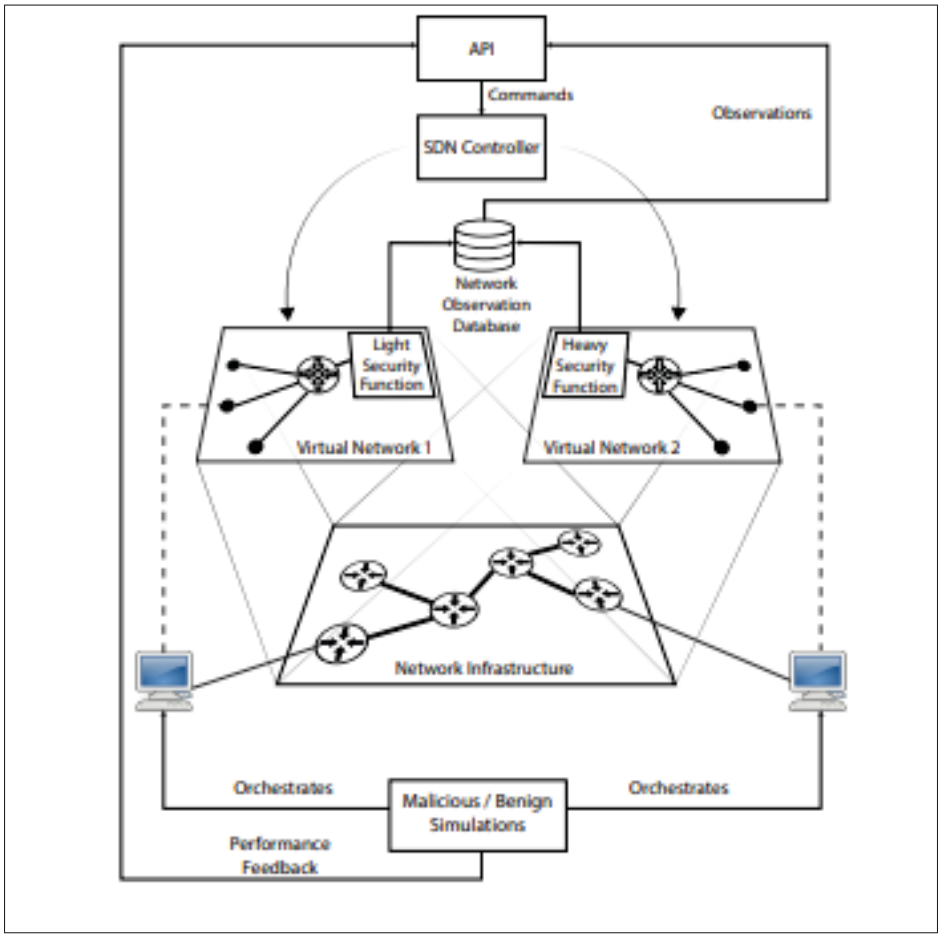


Figure 2.3 Implémentation type de l'architecture ATMoS : les hôtes sont déplacés entre deux réseaux virtuels tout en demeurant transparents à ces changements
Akbari *et al.* (2020)

La figure ci-dessus montre comment le cadre ATMoS est structuré et comment tous ses composants communiquent entre eux. Pour simplifier le fonctionnement du cadre : lorsqu'une menace est détectée sur un nœud du réseau, celui-ci est transféré de la fonction de sécurité légère à la fonction de sécurité lourde.

No. Hosts	Attack	Loss State	Iters. to Convergence
2	SYN	No	150
2	APT	No	1400
2	APT	Yes	1000
3	APT	No	1800

Figure 2.4 Résumé des résultats : itérations nécessaires à la convergence dans les expériences d'ATMoS
Akbari *et al.* (2020)

Ce tableau 2.4 montre les différentes expériences réalisées par les auteurs :

- **2 hosts** indique un hôte bénin et un hôte malveillant.
- **3 hosts** signifie deux hôtes malveillants et un hôte bénin.

Deux types d'attaques sont utilisés dans cet environnement :

- L'attaque **SYN**, qui consiste à envoyer de multiples requêtes SYN au serveur.
- L'attaque **APT**, qui implique plusieurs petites cyberattaques comme l'intrusion, les mouvements latéraux (le déplacement stratégique entre différents hôtes dans un réseau après compromission initiale, dans le but d'atteindre des cibles plus sensibles) et le sabotage.

L'état de *loss* représente une récompense terminale négative qui termine l'épisode si l'agent prend de mauvaises décisions. C'est précisément cet élément qui a permis la convergence la plus rapide du modèle.Akbari *et al.* (2020)

2.3 PenGym

Les auteurs de cet article introduisent un nouveau cadre nommé **PenGYM**, basé sur un environnement de pentesting par Reinforcement Learning préexistant appelé **NASim**, mais censé être amélioré pour inclure des environnements plus réalistes. PenGYM utilise des hôtes et des services réels au lieu de modèles logiques, et permet la génération automatique d'hôtes vulnérables, facilitant ainsi l'entraînement. Ils modifient également l'espace d'action, qui inclut des tâches de pentesting plus réalistes.

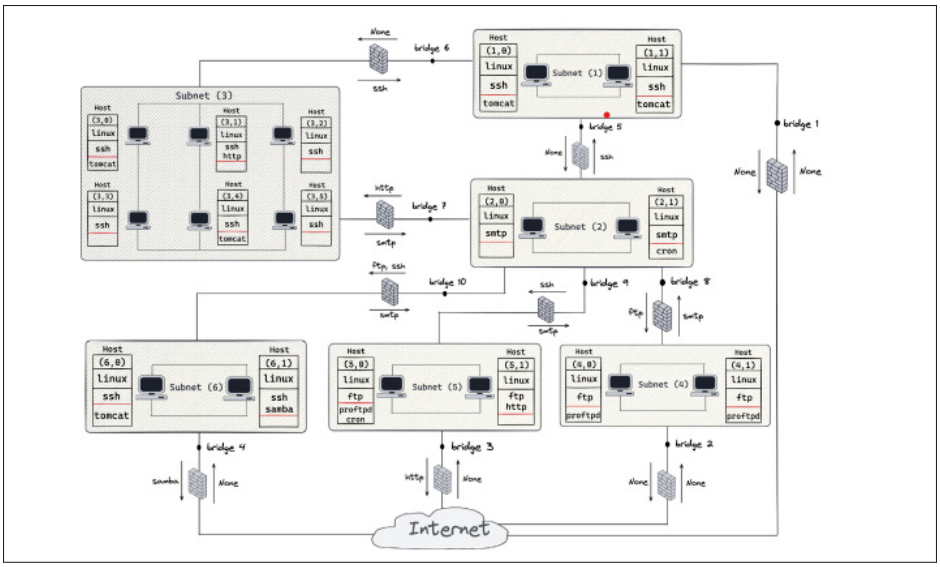


Figure 2.5 Plage réseau construite dans PenGym à partir du scénario multi-site de taille moyenne de NASim
Nguyen *et al.* (2025)

La figure ci-dessus montre les sous-réseaux, les hôtes et les ponts, ainsi que le fonctionnement du contrôle de trafic dans l'environnement **PenGYM**, c'est-à-dire les types de protocoles utilisés et les règles définies pour les ponts. Cela prouve que l'environnement PenGYM offre une simulation réseau réaliste.

Actions		Observation	
		NASim	PenGym
Scan Actions	service_scan	'ssh': 0, 'http': 0, 'ftp': 0, 'smtp': 1, 'samba': 0	['smtp']
	os_scan	'linux': 1	['linux']
	process_scan	'tomcat': 0, 'proftpd': 0, 'cron': 1	['cron']
	subnet_scan	(1, 0): True, (1, 1): True... (6, 0): True, (6, 1): True	['44.1.5.2', '44.1.5.3'... '44.1.10.4', '44.1.10.5']
Exploit Actions	e_ssh	access=USER	
	e_http		
	e_ftp	access=ROOT	
	e_samba		
	e_smtp		
Privilege Escalation Actions	pe_tomcat	access=ROOT	
	pe_proftpd		
	pe_cron		

Le tableau 2.6 montre les différences entre les actions de pentesting dans **NASim** et celles dans **PenGYM**. Pentesting est une méthode qui simule des attaques contrôlées sur un système informatique afin d'identifier des vulnérabilités exploitables. PenGYM fournit un accès root, des numéros de sous-réseaux réels, ainsi que des analyses de services plus précises.

Network	Scenario	Training results			Testing steps		
		Environment	Training time [s]	Ratio*	NASim* environment	PenGym environment	Step difference
Tiny	tiny	NASim	33.721436	5.958373	**6.6**	**7**	0.4
		NASim(rev.)	34.529200		7.4	7	0.4
		PenGym	159.933349		7.6	7	0.6
	tiny-hard	NASim(rev.)	34.529200		6.6	7.8	1.2
		PenGym	246.850449		5.4	5.4	0
	tiny-small	NASim(rev.)	77.674751		12	8.8	3.2
		PenGym	473.369863		8.8	9.5	0.7
Small	small-honeypot	NASim(rev.)	151.095278	4.548211	8.8	8.8	0
		PenGym	1019.356081		10.4	11.8	1.4
	small-linear	NASim	223.226071		**21.2**	**213.8**	192.6
		NASim(rev.)	373.258370		15.2	14.8	0.4
		PenGym	877.148658		14.8	15.8	1
Medium	medium	NASim(rev.)	3668.875337	1.688260	12.8	16.6	3.8
		PenGym	6438.764622		14.2	14	0.2
	medium-single-site	NASim(rev.)	2630.687669		6.6	6.4	0.2
		PenGym	4158.419250		6.6	6.2	0.4
	medium-multi-site	NASim	4890.494927		**24.6**	**2000**	1975.4
		NASim(rev.)	4348.470149		26	24.2	1.8
		PenGym	7518.836585		24.6	24	0.6

Figure 2.7 Résultats des expériences préliminaires pour tous les scénarios disponibles
Nguyen *et al.* (2025)

Dans ce tableau, on observe les différents scénarios réseau disponibles dans l'environnement ainsi que les résultats pour deux versions de NASim et l'environnement PenGYM. La comparaison se base sur le *temps d'entraînement* et le nombre d'étapes nécessaires durant les tests.

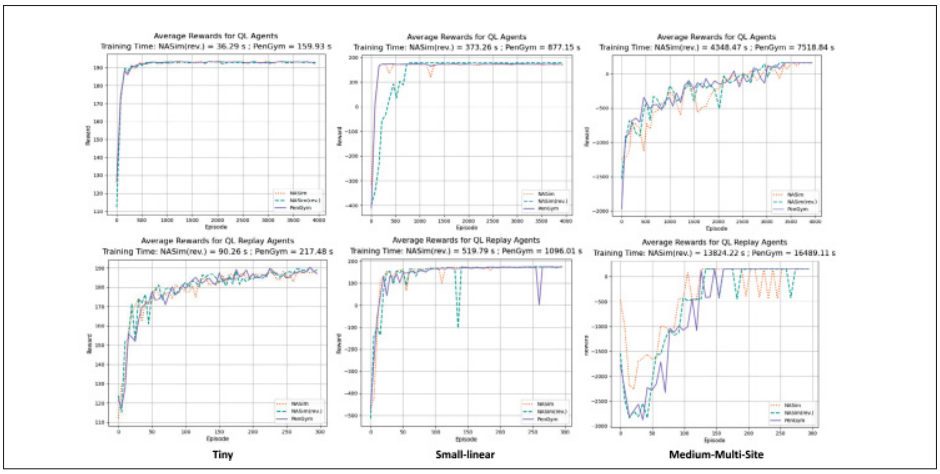


Figure 2.8 Récompense moyenne d'entraînement en fonction du nombre d'épisodes pour les agents Q-learning et Q-learning Replay entraînés dans les environnements NASim, NASim (rev.) et PenGym
Nguyen *et al.* (2025)

Le tableau montre clairement que **PenGYM** nécessite un temps d'entraînement plus long, avec un ratio de temps d'entraînement plus élevé, ce qui est normal étant donné la complexité plus grande de l'environnement. Même dans les scénarios complexes, on observe un écart important entre les agents :

- Les agents entraînés dans NASim ont de mauvaises performances dans l'environnement PenGYM.
- Inversement, les agents PenGYM réussissent mieux dans l'environnement NASim.

Ces graphiques montrent la progression de l'entraînement des agents Q-learning et QL Replay sur différents scénarios :

- Dans le scénario simple, les trois environnements NASim, NASim(rev.) et PenGYM, produisent des récompenses similaires avec une convergence rapide.
- Dans le scénario *small-linear*, les agents entraînés dans NASim et NASim(rev.) convergent plus rapidement vers de hautes récompenses que ceux dans PenGYM, ce qui signifie que les environnements simulés sont plus faciles à ce niveau.

- Dans le scénario complexe, les agents PenGYM et NASim(rev.) mettent beaucoup plus de temps à converger. La progression de la récompense pour PenGYM est plus lente et présente plus de fluctuations, reflétant les défis liés à l'apprentissage dans un environnement réseau réaliste avec des configurations et restrictions poussées.

Cet article met en évidence l'importance des environnements réseau complexes et de la simulation en temps réel des services, permettant aux agents d'acquérir une expérience plus proche du trafic réel, et les rendant ainsi mieux préparés aux réseaux du monde réel.

2.4 Cybershield

Les auteurs de cet article présentent **Cybershield** Fernández Carrasco *et al.* (2024), un environnement conçu pour entraîner et évaluer des algorithmes de Reinforcement Learning appliqués à la cybersécurité des communications réseau. Cet environnement est personnalisable afin d'accommoder diverses topologies et configurations réseau. Il est également conçu pour permettre l'entraînement simultané d'agents **red team** et **blue team**.Fernández Carrasco *et al.* (2024)

Environment	Simulated	Emulated	Blue Team	Red Team	Tools
ATMoS [4]	Yes	Yes	RL	Non RL	Mininet, ODL, Snort, Docker, OVS, Flask REST
Asap [11]	Yes	No	No	RL	Python, Metasploit
AutoPentest-DRL [12]	Yes	No	No	RL	MulVAL, Docker, Nmap, Metasploit
BRAWL [13]	Yes	Yes	Non RL	Non RL	OpenStack, Kafka, Kibana
Csle [14]	Yes	Yes	Enables RL	Enables RL	Python, Ansible, Ryu, Docker, REST API, Prometheus, Grafana
CLAP [15]	No	No	No	RL	Python
CyberBattleSim [1]	Yes	No	Stochastic defender	RL	Python
CybORG [16]	Yes	Yes	Enables RL	Enables RL	Python
Gym-flipit [17]	Yes	No	Non RL	RL	Python
Gym-idsgame [18]	Yes	No	RL	RL	Python
Gym-threat-defense [19]	Yes	No	RL	Non RL	Python
MAB-malware [20]	Yes	No	No	RL	Python
NASim [3]	Yes	No	No	Enables RL	Python
NASimEmu [21]	Yes	Yes	No	RL	Python, NASim, Vagrant, Metasploit
PenGym [22]	Yes	Yes	No	Enables RL	Python, NASim, CyRIS, Nmap, Metasploit
YAWNING-TITAN [2]	Yes	No	RL	Non RL	Python
CyberShield	Yes	No	RL	RL	Python, REST API

Figure 2.9 Comparaison avec les environnements de référence à l'état de l'art
Fernández Carrasco *et al.* (2024)

La figure ci-dessus présente un tableau comparatif des différents environnements déjà implémentés, indiquant s'ils permettent l'entraînement par RL et l'entraînement multi-agents. Le tableau montre également si le réseau est simulé ou émulé. Cette distinction est importante puisque les environnements émulés permettent à l'agent d'éxperiencé des scénarios réseau réels. On constate que **Cybershield** est le seul environnement permettant l'entraînement par RL à la fois pour les agents rouges et pour les agents bleus, en même temps.

Dans cet environnement, l'attaquant commence avec le contrôle de deux nœuds choisis aléatoirement, puis lance des attaques afin d'atteindre les nœuds contenant les *flags*. En parallèle, l'agent bleu défend le nœud critique en l'isolant ou en déployant des *honeypots*.

Agent type	Random Att	Heuristic Att
Random Def	[57,43]	[15,85]
Heuristic Def	[95,5]	[61,39]

Figure 2.10 Résultats obtenus avec différents agents dans Cybershield
Fernández Carrasco *et al.* (2024)

La figure ci-dessus présente quatre résultats exprimés en pourcentages, correspondant aux scénarios suivants :

1. **Défenseur aléatoire vs. Attaquant aléatoire** : Résultat : [57, 43] *Explication* : le défenseur gagne dans 57% des simulations. Malgré l'utilisation d'actions aléatoires, il surpasse l'attaquant aléatoire, montrant un léger avantage dû au rôle défensif par nature.
2. **Défenseur aléatoire vs. Attaquant heuristique** : Résultat : [15, 85] *Explication* : l'attaquant heuristique gagne dans 85% des simulations, prouvant qu'un attaquant stratégique surpasse largement un défenseur agissant de manière aléatoire.
3. **Défenseur heuristique vs. Attaquant aléatoire** : Résultat : [95, 5] *Explication* : le défenseur heuristique gagne dans 95% des simulations, démontrant l'efficacité d'une défense stratégique face à une attaque non coordonnée.
4. **Défenseur heuristique vs. Attaquant heuristique** : Résultat : [61, 39] *Explication* : le défenseur heuristique gagne dans 61% des simulations, gardant un avantage même face à un attaquant stratégique. Cependant, le taux de victoire est plus proche, ce qui reflète un équilibre compétitif entre les stratégies heuristiques.

2.5 Cygil

Dans cet article, les auteurs présentent **CyGIL** , un environnement émulé conçu pour l'entraînement par Reinforcement Learning appliqué à la cybersécurité. Il fournit une plateforme réaliste et fidèle pour l'entraînement d'agents cyber autonomes pilotés par l'IA. Il intègre le cadre **MITRE ATT&CK** afin d'assurer un espace d'actions complet pour l'entraînement des red

teams, et se concentre sur la réduction de l'écart entre les environnements basés sur simulation et leur applicabilité au monde réel.

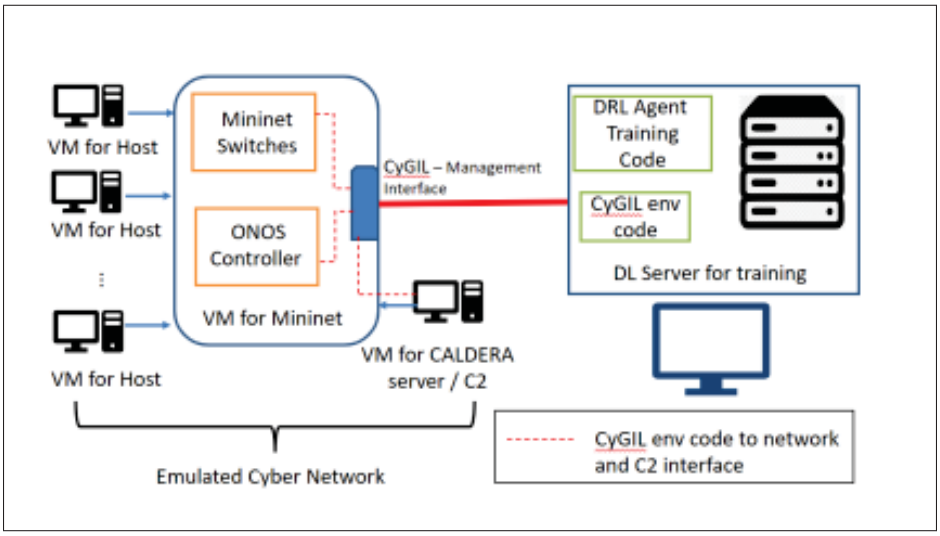


Figure 2.11 Architecture du système CyGIL : configuration de l'environnement et composants principaux
Li *et al.* (2021)

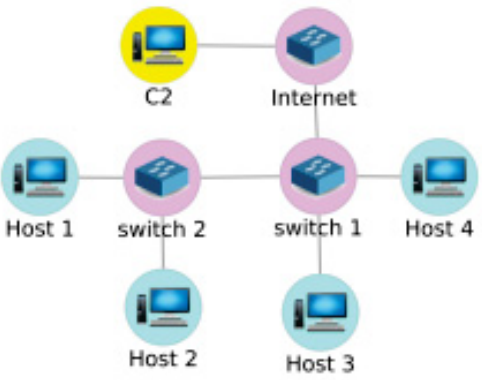
La figure ci-dessus présente l'environnement CYGIL Gym et ses composants principaux. Les hôtes sont des machines virtuelles déployées avec Mininet, tandis que **Caldera** est installé sur une machine distincte. Caldera lance des attaques contre les machines virtuelles ; toutes les données issues de ces attaques sont traitées via la fonctionnalité de rapport de Caldera, puis envoyées au modèle DRL qui entraîne l'agent en se basant sur le résultat de ces attaques et évalue la récompense selon le nouvel état.

L'état de CyGIL comprend les éléments suivants :

- Si un accès est présent sur l'hôte et à quel niveau de privilège utilisateur.
- Les comptes utilisateurs locaux et les identifiants découverts.
- Les types d'OS et le type d'hôte.
- Les services/processus modifiables.
- Les fichiers locaux, répertoires et partages.
- Les configurations défensives de l'hôte observées (par ex. antivirus).

- Les informations réseau de l'hôte trouvées (nom de domaine, adresse IP, interfaces de sous-réseau).
- Autres informations système de l'hôte.
- Autres informations systèmes distantes observées depuis cet hôte.
- Si l'action a été exécutée avec succès sur l'hôte.

Fonction de récompense : la récompense est façonnée de sorte que si l'agent atteint l'objectif désiré (exfiltrer le fichier cible avec succès), il reçoit une récompense sinon, il reçoit une pénalité.



(a) Network scenario

ID	ATT&CK Tactic	ATT&CK Technique
0	Collection	T1074 (1):create directory for exfil staging
1	Collection	T1005: find sensitive files on local host
2	Collection	T1074 (2): copy files to staged directory
3	Exfiltration	T1560.001: compress file directory
4	Exfiltration (Linux)	T1560.001: compress file directory
5	Collection (Linux)	T1560.001: encrypt data with zip and gpg
6	Collection (Linux)	T1560.001: nix/gzip single file
7	Collection (Linux)	T1560.001: data compressed, zip file
9	Exfiltration	T1041: exfiltrate staged dir over C2

(b) Action space

Figure 2.12 Exemple de scénario d'entraînement dans CyGIL
Li *et al.* (2021)

La figure ci-dessus montre une topologie réseau exemple proposée par les auteurs ainsi que l'ensemble des actions que l'agent doit exécuter et dans quel ordre. Dans ce scénario, l'objectif final est d'exfiltrer un fichier cible vers le serveur Caldera.

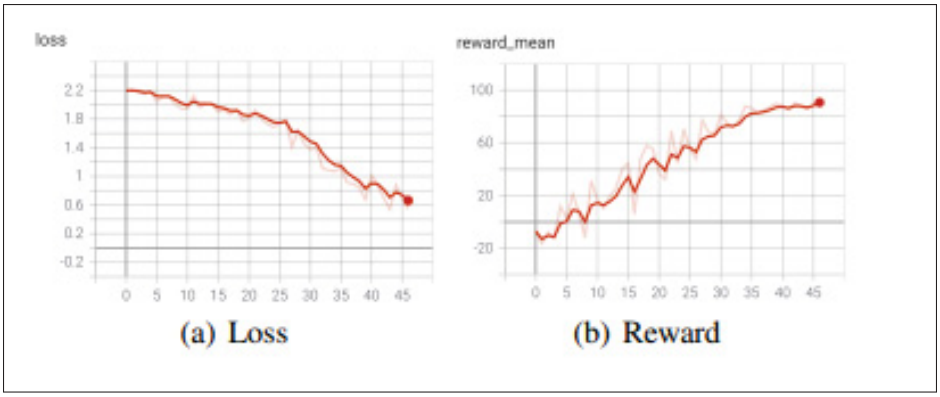


Figure 2.13 Entraînement de l'agent DQN dans le premier jeu : l'axe x correspond à 100 étapes par unité
Li *et al.* (2021)

On observe que, pour ce jeu après environ **4500** étapes, le modèle converge et atteint une récompense moyenne de **99**, correspondant à l'état objectif.

L'environnement proposé dans cet article introduit également **CALDERA**, qui constitue un composant central pour notre thèse et dont nous discuterons plus en détail au chapitre suivant, ainsi que **OpenAI Gym**.

2.6 Discussion des Travaux Antérieurs

Tableau 2.1 Comparaison des travaux antérieurs sur l'entraînement d'agents cybernétiques par apprentissage par renforcement

Référence	**Type d'environnement**	**Approche RL utilisée**	**Capacité de généralisation**	**Points forts / Limites**
CyberBattleSim Kunz *et al.* (2023)	Simulation abstraite (chaîne statique)	PPO, A2C	Faible	Facile à intégrer, mais environnement limité
ATMoS Akbari *et al.* (2020)	Simulation logicielle multiréseaux, comportements réalistes	PPO, A2C	Élevée	Gestion dynamique du réseau, apprentissage multi-tâches
PenGym Nguyen *et al.* (2025)	Environnement Gym pour attaque/défense	Q-learning	Moyenne	Multi-agent, mais nécessite beaucoup de ressources
CyGIL Li *et al.* (2021)	Architecture stateless (orientée red teaming)	Q-learning, DRL	Moyenne	Suit les techniques industrielles, mais pas génératif
VDGym Cai *et al.* (2024)	Générateur simulé basé sur vulnérabilités	Q-learning tabulaire, DRL	Moyenne	Génère dynamiquement des scénarios variés ; bonne généralisation démontrée
Notre approche	Simulation basée sur CyGil amélioré	Q-learning tabulaire	Élevée	Simple à implémenter, bon compromis entre réalisme et complexité

Le tableau 2.1 résume les principales contributions dans le domaine de l'entraînement d'agents cybernétiques via apprentissage par renforcement. Les premières approches comme *CyberBattleSim* (Kunz *et al.* (2023)) ont mis en place un cadre minimaliste, utile à des fins de

test, mais limité en termes de diversité des scénarios. Des frameworks comme *CyGIL* (Li *et al.* (2021)) ou *PenGym* (Nguyen *et al.* (2025)) apportent une plus grande modularité et introduisent des modèles multi-agents, bien que leur coût computationnel reste élevé.

Parmi les solutions récentes, *ATMoS* (Akbari *et al.* (2020)) se distingue par sa capacité à simuler des environnements cyber réalistes et dynamiques, offrant ainsi un terrain d'expérimentation favorable aux algorithmes de type PPO ou A2C. Quant à *VDGym* (Cai *et al.* (2024)), il introduit une approche originale basée sur des bases de vulnérabilités et une génération procédurale d'environnements, ce qui favorise la généralisation des politiques apprises. Notre méthode reprend l'esprit modulaire de VDGym tout en conservant la simplicité et la rigueur du Q-learning tabulaire, adapté à des scénarios de taille moyenne mais personnalisables.

En résumé, les travaux analysés démontrent le potentiel considérable de l'apprentissage par renforcement dans des domaines variés tels que la détection d'hameçonnage, la sécurisation des réseaux 5G, la conduite autonome, les systèmes militaires ou la défense adaptative à gradient de politique. Ces approches partagent un objectif commun : concevoir des agents capables d'apprendre de leur environnement et de s'adapter à des menaces dynamiques et inconnues. Cependant, la plupart de ces solutions demeurent limitées par leur capacité de généralisation ; les agents entraînés dans un scénario donné peinent souvent à transférer efficacement leurs connaissances à d'autres contextes.

C'est précisément sur cette limite que s'appuie la présente recherche. Dans les prochains chapitre, nous proposons une approche expérimentale intégrant l'apprentissage par renforcement et le méta-apprentissage modèle-agnostique (Q-MAML) afin de développer des agents capables d'adaptation rapide à de nouveaux scénarios d'attaque-défense. À travers un environnement personnalisé construit avec OpenAI Gym et MITRE Caldera, nous mettons en œuvre un banc d'essai réaliste pour évaluer la résilience, la rapidité d'adaptation et la transférabilité des politiques apprises. L'objectif est de démontrer que le RL, lorsqu'il est enrichi d'un méta-apprentissage, peut dépasser les limites de la spécialisation des modèles et ouvrir la voie à une cybersécurité proactive, autonome et véritablement résiliente.

CHAPITRE 3

APPROCHE PROPOSÉE

Dans le contexte des cyberattaques modernes, les défenses traditionnelles basées sur des règles ou des signatures peinent à s'adapter à des environnements dynamiques, où les tactiques adverses évoluent rapidement. De plus, les environnements d'entraînement pour l'apprentissage par renforcement sont souvent simulés, peu fidèles aux scénarios réels, et limités dans leur représentation des techniques adverses telles que définies par le cadre MITRE ATT&CK.

Face à ces lacunes, nous proposons une approche combinée, s'appuyant sur :

- Un environnement d'émulation réaliste basé sur Caldera, permettant l'orchestration automatisée d'attaques selon des scénarios définis.
- Un agent d'apprentissage par renforcement autonome, capable de prendre des décisions dans cet environnement.
- L'intégration d'un algorithme de méta-apprentissage, permettant une adaptation rapide à des scénarios inconnues.

L'environnement CyGIL original, présenté dans Li *et al.* (2021) constitue une référence importante pour l'entraînement d'agents RL en cybersécurité. Cependant, aucun dépôt de code public n'est disponible. Pour cette raison, nous avons reconstruit l'environnement manuellement en nous basant sur la description méthodologique et les figures de l'article.

Cette reconstruction inclut :

- La mise en place de machines virtuelles sur VMware simulant une topologie réseau réaliste.
- L'intégration complète de la plateforme MITRE Caldera et de son agent Sandcat.
- Le développement d'un wrapper compatible OpenAI Gym permettant l'interaction RL automatisée.
- La définition de règles de récompense et d'adaptées à différents scénarios ATT&CK.

La figure 3.1 illustre le fonctionnement général de l'environnement d'émulation développé. L'agent de Q-learning interagit avec l'environnement OpenAI Gym en envoyant des actions et en

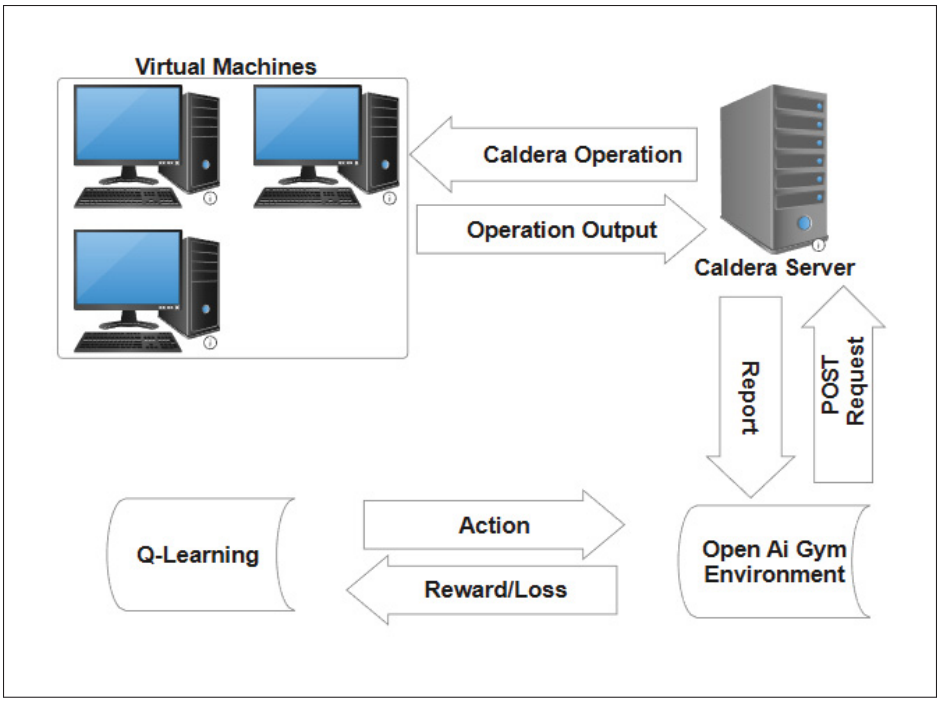


Figure 3.1 Configuration de notre système

recevant des récompenses ou des pertes selon les résultats des opérations exécutées. Ces actions sont traduites en requêtes HTTP transmises au serveur Caldera, qui déclenche les opérations d'attaque ou de défense sur les machines virtuelles cibles. Les rapports générés par Caldera sont ensuite renvoyés vers l'environnement Gym pour mettre à jour la politique d'apprentissage de l'agent, créant ainsi une boucle fermée d'apprentissage autonome et itératif.

Contrairement au banc d'essai CyGIL Li *et al.* (2021), qui s'appuie sur Mininet et un contrôleur ONOS pour émuler le réseau cible, notre environnement repose entièrement sur des machines virtuelles Linux hébergées sous VMware. Chaque machine virtuelle agit comme un hôte cible distinct, tandis que le serveur Caldera est isolé sur une autre machine virtuelle afin de reproduire un contrôle distant de type *Command-and-Control* (C2).

Nous avons également étendu le cadre expérimental en définissant plusieurs scénarios indépendants, chacun reposant sur une topologie et un ensemble de VMs différents. Ces scénarios permettent

d'évaluer la capacité de l'agent à s'adapter à divers contextes tactiques et à apprendre des politiques de défense ou d'attaque spécifiques.

Sur le plan algorithmique, CyGIL s'appuie principalement sur des réseaux neuronaux profonds de type DQN pour entraîner ses agents dans un environnement donné. Bien que cette approche permette de traiter des espaces d'état plus complexes, elle reste fortement dépendante du scénario d'entraînement initial, avec une capacité de transfert limitée. En revanche, notre approche introduit une progression méthodologique en combinant le Q-learning tabulaire — initialement utilisé pour explorer les dynamiques d'attaque — avec un modèle de méta-apprentissage visant la généralisation multi-scénarios. Cette hybridation permet à notre agent d'adapter sa stratégie à des environnements inconnus sans nécessiter de réentraînement complet, ce qui représente une avancée notable en matière de résilience adaptative et de transfert des politiques d'attaque.

En résumé, notre environnement d'émulation se distingue de CyGIL sur plusieurs plans :

- Utilisation de **VMware et de machines virtuelles Linux** au lieu de Mininet.
- Séparation physique du serveur Caldera et des hôtes cibles sur des VMs indépendantes.
- Introduction de **plusieurs scénarios** d'attaque et de défense basés sur la matrice MITRE ATT&CK.
- Implémentation d'un **Q-learning tabulaire** plutôt que de réseaux neuronaux profonds.

3.1 OpenAI Gym

OpenAI Gym (Gymnasium) fournit les outils nécessaires pour construire et tester des idées. Au cœur de Gymnasium, on trouve une **API** (Application Programming Interface) pour tous les environnements d'apprentissage par renforcement à agent unique. Elle propose des implémentations d'environnements classiques ainsi que la possibilité de créer son propre environnement Farama Foundation (2025).

Dans notre travail, nous avons créé un environnement personnalisé dans **Gymnasium**. L'initialisation se fait via la fonction `__init__()` afin de définir toutes les variables et constantes nécessaires à

l'environnement. Ensuite, nous passons au cœur de l'environnement où nous implémentons les fonctions auxiliaires nécessaires pour créer une opération **Caldera** et en récupérer le résultat.

À partir de ces résultats, nous mettons à jour notre espace d'états, représenté par un tableau binaire. Celui-ci est utilisé pour calculer la récompense de l'agent via la fonction `reward()` définie dans notre implémentation. La récompense est évaluée en fonction de l'atteinte ou non de l'objectif par l'agent.

Après l'évaluation de la récompense, l'agent choisit une nouvelle action depuis l'espace d'actions. Si l'agent dépasse le nombre maximum d'étapes autorisées, l'environnement termine l'épisode et appelle la fonction `reset()`, qui réinitialise toutes les variables nécessaires. L'agent reprend alors un nouvel épisode avec des conditions réinitialisées.

3.2 Topologie réseau

La topologie de départ, composée de trois hôtes Linux et d'un serveur **Caldera**, a été conçue pour reproduire un pipeline d'attaque multi-sauts simple mais représentatif des opérations de pentest réelles. Ce choix minimaliste facilite l'analyse détaillée du comportement de l'agent tout en conservant des éléments clés : point d'entrée, hôte pivot intermédiaire et cible finale. La présence d'un serveur Caldera distinct permet de séparer clairement la couche d'émulation/adversaire de la couche hôte, ce qui simplifie la collecte et l'agrégation des données d'exécution.

La topologie apporte plusieurs avantages expérimentaux. Elle réduit la variance introduite par des réseaux plus larges, facilite le débogage des séquences d'action et permet d'isoler précisément l'effet d'une ability donnée sur le succès global. En revanche, elle conserve des défis réalistes : nécessité d'obtenir des identifiants, navigation dans des permissions de fichiers, et enchaînement correct des étapes d'exfiltration. Ces aspects sont essentiels pour évaluer la capacité d'un agent RL à planifier et exécuter une chaîne d'actions cohérente.

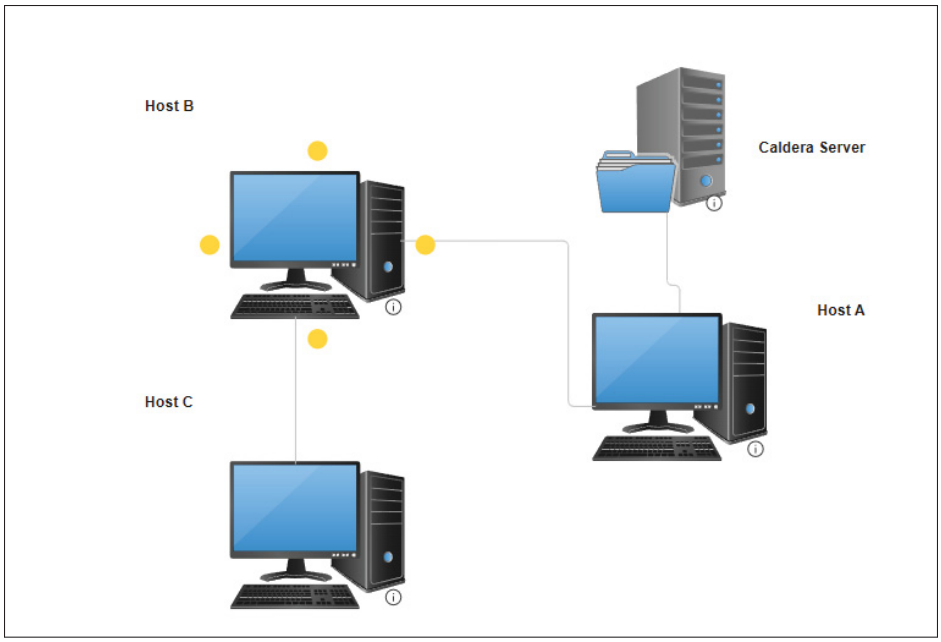


Figure 3.2 Schéma de la connexion entre les hôtes

La figure 3.2 illustre la topologie employée pour l'entraînement initial et la manière dont les hôtes sont interconnectés.

L'objectif associé à cette topologie est le suivant : l'agent débute les attaques depuis **Host A** vers **Host B** afin d'obtenir des informations (identifiants, fichiers de configuration, etc.) lui permettant d'accéder à **Host C**. Une fois l'accès à Host C obtenu, l'agent lance des attaques ciblées sur Host C pour atteindre l'objectif final (par exemple l'exfiltration d'un fichier cible).

Ce scénario simple permet d'entraîner et d'évaluer les capacités de découverte, de mouvement latéral et d'exploitation de l'agent dans un parcours multi-sauts contrôlé.

3.3 Caldera

Caldera est le cadre que nous utilisons pour lancer nos attaques afin de simuler des scénarios de pentest permettant à notre agent d'apprendre. Caldera™ est une plateforme d'émulation d'adversaire conçue pour exécuter facilement des exercices autonomes de type breach-and-attack

qui est une approche qui utilise des outils automatisés pour simuler en continu des attaques réelles, tester les défenses d'une organisation et identifier les failles de sécurité. Elle peut également être utilisée pour des engagements red-team manuels ou pour des réponses automatisées aux incidents. Caldera est construite sur le cadre **MITRE ATT&CK™** et constitue un projet de recherche actif chez MITRE MITRE Corporation (2025).

Parmi les options possibles, nous aurions pu nous appuyer sur une distribution de type Kali Linux ou sur des frameworks centrés sur des outils individuels comme Metasploit. Nous avons retenu Caldera pour trois raisons principales :

- Chaque ability est déjà annotée en termes de tactiques et de techniques MITRE ATT&CK, ce qui facilite le mappage vers des scénarios structurés.
- La plateforme expose une API REST stable qui se prête directement à l'intégration dans un environnement Gym.
- L'agent Sandcat permet de déployer et rejouer automatiquement des charges utiles homogènes sur plusieurs hôtes. Ces caractéristiques font de Caldera un support naturel pour construire un banc d'essai RL orienté cybersécurité résiliente.

Pour orchestrer ces séquences, notre implémentation s'appuie sur l'API de Caldera : chaque opération est créée par une requête POST et la réponse initiale fournit un identifiant unique ("operation id"). Nous conservons cet identifiant afin d'interroger ensuite l'API de rapports et récupérer l'état final et les artefacts produits par l'opération. La procédure typique est : (1) poster l'opération, (2) stocker l'‘operation id‘, (3) interroger périodiquement l'API de rapports jusqu'à obtention d'un état terminal, et (4) parser le contenu du rapport.

Les rapports Caldera contiennent des éléments exploitables pour l'agent : sortie standard des commandes, statut d'exécution, et éventuellement des artefacts tels que fichiers récupérés, identifiants trouvés ou adresses IP découvertes. Nous avons adopté une stratégie systématique de transformation de ces artefacts en composantes de notre espace d'observation, qui représente l'état perçu par l'agent à chaque étape. Cette transformation normalise les retours hétérogènes de Caldera en un vecteur binaire déterministe utilisable par l'agent RL.

Nous avons implémenté des mécanismes de tolérance et de robustesse autour de ces échanges : gestion des timeouts (pour éviter des blocages lors d'opérations longues), stratégie de retry pour les requêtes HTTP, et logique de rejet des rapports incohérents. De plus, afin de garantir la reproductibilité expérimentale, toutes les interactions avec Caldera sont journalisées (horodatage, 'operation id', payload envoyé, réponse reçue), ce qui facilite l'analyse post-hoc et le débogage des phases d'entraînement.

Un mécanisme de *retry* automatique a également été intégré pour les requêtes HTTP. En cas d'échec réseau, de réponse incomplète ou de code d'erreur serveur, la requête est relancée un nombre limité de fois avec un délai progressif entre chaque tentative. Cette stratégie permet de compenser les fluctuations temporaires de la plateforme Caldera ou de l'infrastructure réseau sans fausser le cycle d'interaction de l'agent.

Par ailleurs, une logique de validation et de rejet des rapports incohérents a été intégrée. Les rapports d'opérations retournés par Caldera sont analysés afin de vérifier la cohérence des identifiants d'actions, des horodatages, de l'état des hôtes et des résultats d'exécution. Tout rapport incomplet, corrompu ou ne correspondant pas à l'action demandée est automatiquement ignoré afin de préserver l'intégrité de l'espace d'observations et d'éviter l'introduction de bruit dans l'apprentissage.

Afin de garantir la traçabilité complète du système, toutes les interactions avec Caldera sont systématiquement journalisées. Chaque action est associée à un horodatage précis, au contenu de la requête, au code de réponse et aux résultats retournés. Cette journalisation permet à la fois une analyse post-expérimentale détaillée, un débogage fin des scénarios et une reproductibilité rigoureuse des expériences.

Sur le plan opérationnel, les actions potentiellement destructrices sont confinées à des machines virtuelles isolées et réinitialisables. Nous paramétrons les abilities Caldera de manière à contrôler l'impact et nous utilisons des snapshots VM pour revenir rapidement à un état propre entre épisodes. Enfin, les métriques extraites via Caldera sont utilisées pour évaluer la qualité des politiques apprises et pour adapter le design des récompenses et des préconditions d'action.

Figure 3.3 Déploiement d'un agent 1

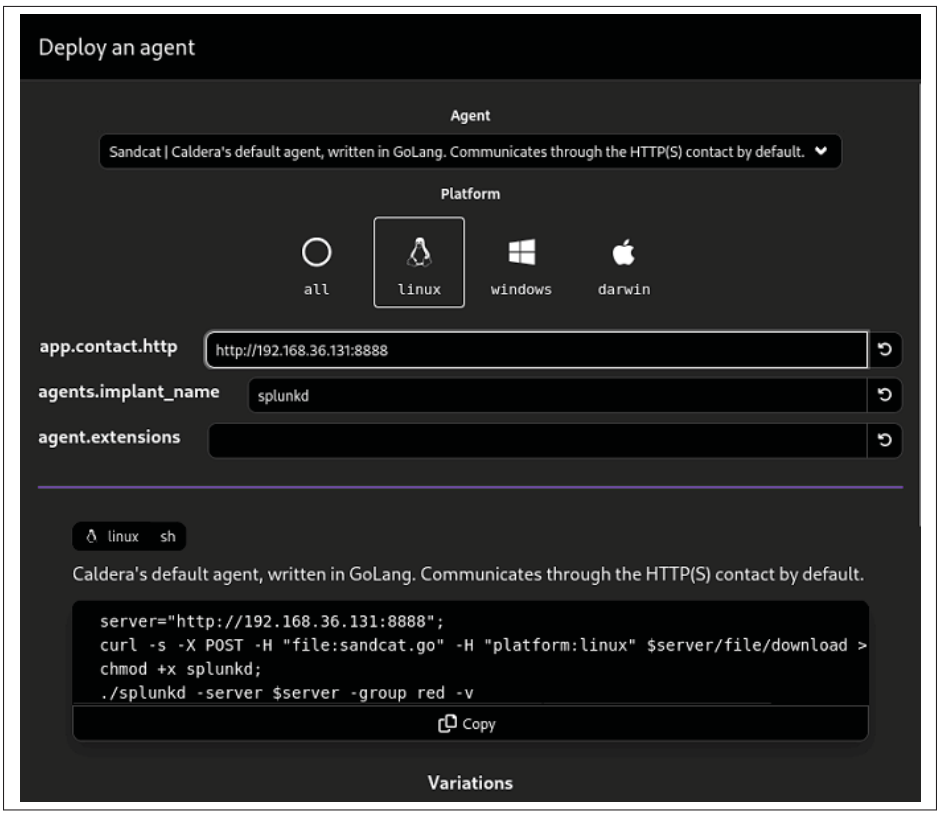


Figure 3.4 Déploiement d'un agent 2

Dans notre cas, nous utilisons l'agent `sandcat` et déployons la commande d'installation correspondante sur la première machine cible. Le déploiement consiste à transférer le script d'installation vers la VM cible puis à exécuter la commande d'installation en session contrôlée ; une fois installé, l'agent établit son canal de communication vers le serveur Caldera et apparaît

dans la liste des agents disponibles sur l'interface. Nous vérifions ensuite l'état de l'agent via l'API de Caldera avant de lancer des opérations. Pour garantir la reproductibilité et la sécurité expérimentale, l'installation est effectuée sur des machines virtuelles isolées avec des snapshots préétablis et les permissions requises sont limitées au strict nécessaire pour exécuter les abilities testées. Enfin, chaque déploiement est journalisé afin de pouvoir tracer et analyser le comportement de l'agent lors des épisodes d'entraînement.

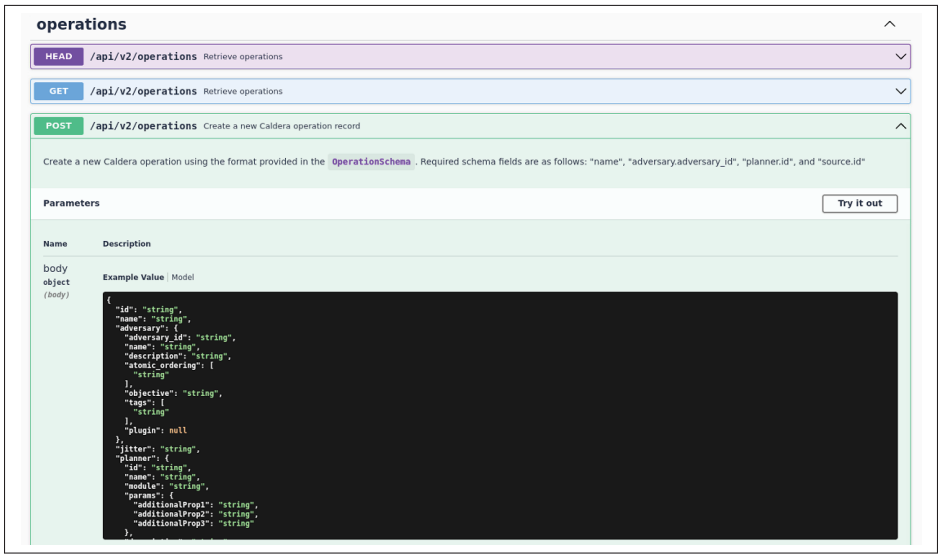


Figure 3.5 POST Caldera Opération

La figure 3.5 illustre également les API liées au posting d'opérations vers le serveur Caldera. Sur ce schéma, nous distinguons les étapes clés du cycle d'une opération : la construction du payload d'opération, l'envoi d'une requête POST pour créer l'opération, la réception de la réponse initiale contenant l''operation id', et la boucle de consultation de l'API de rapports jusqu'à obtention d'un état terminal.

Le payload envoyé lors du POST est structuré de manière à référencer une ability, à spécifier la ou les cibles et à fournir les paramètres nécessaires à l'exécution. L''operation id' renvoyé par Caldera sert d'index unique permettant de récupérer ultérieurement le statut et les artefacts produits par l'opération.

La figure met également en évidence la logique de synchronisation entre la création d'opération et la mise à jour de l'environnement RL : notre boucle d'environnement via l'API demande le rapport de l'opération puis parse le contenu du rapport afin d'en extraire les sorties pertinentes. Ces éléments sont transformés en flags binaires ou en attributs structurés pour alimenter l'espace d'observation de l'agent.

Enfin, la figure signale les mécanismes de gestion d'erreurs et de robustesse que nous avons implémentés : gestion des timeouts et retries pour les requêtes longues, vérification de la cohérence des rapports et journalisation systématique. Ces mesures garantissent la fiabilité des interactions API et la reproductibilité des expérimentations décrites dans ce mémoire.

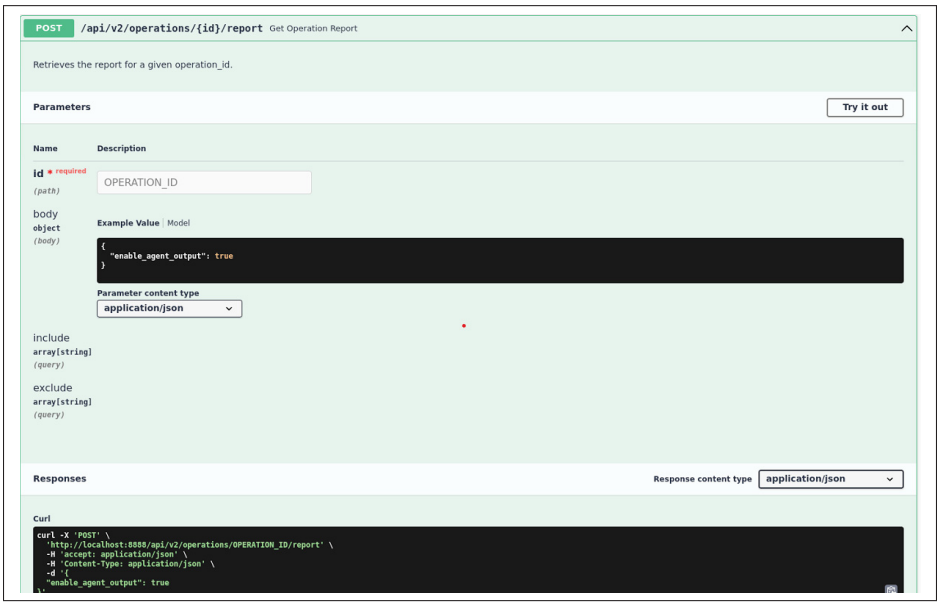


Figure 3.6 Récupérer le rapport d'opération

Pour chaque action lancée via Caldera, nous extrayons un rapport remonté par la plateforme. Ce rapport contient les informations liées à l'action exécutée : soit la sortie standard de l'action, soit l'indication de succès ou d'échec pour une action sans sortie. Ces informations sont ensuite utilisées pour mettre à jour notre espace d'observation.

Chaque rapport est récupéré en interrogeant l'API Caldera à l'aide de l'`operation id` correspondant à l'action exécutée. Le contenu du rapport est ensuite parsé et normalisé : les champs pertinents tels que la sortie texte, le statut d'exécution et les artefacts éventuels sont extraits et convertis en valeurs exploitables par l'agent. Cette étape de transformation est essentielle pour traduire des données hétérogènes, souvent textuelles, en représentations binaires cohérentes et directement intégrables dans la structure d'état du modèle.

Par exemple, si la sortie d'une action contient un mot-clé indiquant la découverte d'un fichier sensible ou d'un identifiant valide, les variables binaires associées dans l'espace d'observation sont activées. À l'inverse, si une action échoue ou ne produit aucun résultat pertinent, les variables correspondantes restent nulles. Cette logique permet à l'agent d'apprendre à interpréter indirectement le succès ou l'échec de ses actions à travers les changements d'état observés.

Cette étape d'intégration des rapports Caldera joue un rôle clé dans la boucle de Reinforcement Learning : elle constitue le lien direct entre les opérations menées dans l'environnement réel émulé et la représentation interne apprise par l'agent. Elle assure ainsi la fidélité de l'observation et contribue à la stabilité de l'apprentissage en environnement complexe.

MITRE Caldera fournit ainsi l'infrastructure C2, les agents et les abilities alignées sur ATT&CK qui sous-tendent notre environnement d'expérimentation. Une présentation détaillée de ces fonctionnalités (agents, adversaires, payloads, opérations, etc.) est donnée à l'Annexe I, à laquelle le lecteur pourra se référer pour une vue plus complète de la plateforme.

3.4 Scénarios

Les scénarios présentés dans les Tableaux sont conçus pour représenter différents stades d'une attaque cybernétique en conformité avec le cadre MITRE ATT&CK. Chaque scénario vise à exposer l'agent à des tactiques spécifiques afin d'évaluer sa capacité d'apprentissage, de planification, et d'adaptation.

Tableau 3.1 Séquence des tactiques et techniques MITRE ATT&CK pour le scénario 1 (exfiltration de fichier). Cette table décrit les étapes d'attaque, de la reconnaissance initiale à l'exfiltration finale, utilisées par l'agent pour atteindre son objectif

ID	ATT&CK Tactic	ATT&CK Technique
0	Discovery	T1082 : Hostname Discovery / DNS server discovery using `nslookup`
1	Discovery	T1018 : Collect ARP details
2	Discovery	T1570 : Copy 54ndc47 (WinRM and SCP)
3	Collection	T1074 (1) : create directory for exfil staging
4	Collection	T1005 : find sensitive files on local host
5	Collection	T1074 (2) : copy files to staged directory
6	Exfiltration	T1560.001 : compress file directory
7	Exfiltration (Linux)	T1560.001 : compress file directory
8	Collection (Linux)	T1560.001 : encrypt data with zip and gpg
9	Collection	T1560.001 : nix/gzip single file
10	Exfiltration	T1041 : exfiltrate staged dir over C2

Le tableau 3.1 présente le scénario initial sur lequel nous avons entraîné le modèle de base. L'objectif est d'extraire un fichier spécifique depuis un hôte donné. Les actions de ce scénario sont proches de celles utilisées par CyGIL dans l'un de leurs jeux : nous avons reproduit ces actions pour identifier les éléments manquants dans leur implémentation et proposer des améliorations.

Ce scénario simule une intrusion initiale suivie d'une exploration du réseau. L'agent doit collecter des informations et progresser en pivotant sur d'autres machines. Il reflète les techniques Discovery et Remote Services. L'objectif est d'apprendre à évaluer les chemins d'attaque valides avec un minimum d'actions.

Dans ce scénario, l'agent dispose d'un accès privilégié sur au moins un hôte du réseau et doit localiser un fichier sensible puis l'exfiltrer vers le serveur de commande. Les techniques associées incluent Data from Local System, Automated Exfiltration et Exfiltration Over C2 Channel. Ce scénario représente l'aboutissement de la chaîne d'attaque et constitue l'objectif ultime d'un adversaire avancé. Il permet d'évaluer la capacité de l'agent à coordonner plusieurs actions dans un ordre strict, tout en respectant les contraintes temporelles et les risques de détection.

Tableau 3.2 Tactiques et techniques MITRE ATT&CK mobilisées dans le scénario 2 (perturbation de services suivie d'une extinction du système). L'agent identifie et désactive des services critiques avant de procéder à un arrêt de la machine

ID	**ATT&CK Tactic**	**ATT&CK Technique**
0	Discovery	T1082 : Hostname Discovery / DNS server discovery using `nslookup`
1	Discovery	T1018 : Collect ARP details
2	Discovery	T1570 : Copy 54ndc47 (WinRM and SCP)
3	Defense-evasion	T1562.004 : Tail the UFW firewall log file
4	Defense-evasion	T1499 : Disrupt WIFI
5	Impact	T1489 : Linux - Stop service by killing process using killall
6	Impact	T1529 :Shutdown System via power off-FreeBSD/Linux

Le tableau 3.2 décrit un scénario où l'agent exécute une série d'actions visant à provoquer un déni de service sur plusieurs services. Ce scénario a pour objectif d'entraîner l'agent à planifier et coordonner des actions ayant un impact sur la disponibilité, et d'évaluer sa capacité à atteindre un état terminal contrôlé tout en maximisant l'efficacité des actions menées.

Ce scénario commence avec un accès utilisateur non privilégié. L'agent doit détecter les vulnérabilités locales, les exploiter et établir un accès persistant. Il teste la capacité de l'agent à exécuter des chaînes d'actions dépendantes et à maintenir un accès malgré les contre-mesures.

Tableau 3.3 Enchaînement des tactiques ATT&CK dans le scénario 3 (interruption ciblée de services et extraction d'informations). La table présente les actions exécutées pour contourner les défenses, collecter des données et redémarrer le système

ID	**ATT&CK Tactic**	**ATT&CK Technique**
0	Discovery	T1082 : Hostname Discovery / DNS server discovery using `nslookup`
1	Discovery	T1018 : Collect ARP details
2	Discovery	T1570 : Copy 54ndc47 (WinRM and SCP)
3	Execution	T1059.004 : Current kernel information enumeration
4	Defense-evasion	T1070.003 : Clear Bash history (rm)
5	Defense-evasion	T1562.003 : Disable history collection (freebsd)
6	Execution	T1059.004 : Create and Execute Bash Shell Script
7	Impact	T1529 : : Reboot System via halt- FreeBSD

Ce tableau 3.3 démontre le troisième scénario est similaire au précédent (série d'actions menant à un DoS), mais la séquence se termine par l'arrêt complet de la machine cible.

L'agent doit identifier puis compromettre un service critique défendu. Ce scénario inclut la reconnaissance, l'accès initial, l'élévation de privilèges, et l'exploitation d'un service exposé. Il s'agit d'évaluer les capacités stratégiques de l'agent face à des défenses connues.

Tableau 3.4 Séquence des techniques MITRE ATT&CK dans le scénario 4 (reconnaissance réseau et marquage de compromission). Cette table illustre les actions utilisées pour identifier la structure du réseau et laisser une trace visible de l'attaque

ID	ATT&CK Tactic	ATT&CK Technique
0	Discovery	T1082 : Hostname Discovery / DNS server discovery using `nslookup`
1	Discovery	T1018 : Collect ARP details
2	Discovery	T1570 : Copy 54ndc47 (WinRM and SCP)
3	Discovery	T1049 : Find System Network Connections
4	Discovery	T1016 : Network Interface Configuration
5	Technical-information-gathering	T1254 : Scan an external host for open ports and services
6	Discovery	T1614 :Get geolocation info through IP-Lookup services using curl freebsd, linux or macos
7	Collection	T1115 : Copy Clipboard
8	Impact	T1491 : Leave note

Ce tableau démontre 3.4 le quatrième et dernier scénario qui comprend une série d'actions de découverte et se conclut par le dépôt d'une note sur la machine cible indiquant qu'elle a été compromise.

Ces quatre scénarios ont été sélectionnés afin d'une base expérimentale équilibrée permettant de mesurer la capacité d'un agent d'apprentissage par renforcement à généraliser, à s'adapter à différents contextes opérationnels et à apprendre des stratégies cohérentes dans des environnements réalistes et contraints.

Chaque scénario proposé dans notre étude repose sur des objectifs opérationnels différents, ce qui influence directement les tactiques ATT&CK mobilisées et les séquences d'actions adoptées par l'agent.

- **Scénario 1** se distingue par son objectif d'exfiltration ciblée, impliquant une chaîne d'attaque orientée vers l'accès furtif et la discrétion. Contrairement aux autres scénarios, il n'implique ni perturbation de services ni action destructrice, ce qui le rend unique sur l'axe de la confidentialité.
- **Scénario 2** introduit une dimension de sabotage avec interruption de services suivie d'un arrêt du système. Ce comportement destructeur n'est pas présent dans le scénario 1, qui reste furtif, ni dans le scénario 4, qui se limite à une reconnaissance passive. Contrairement au scénario 3, il ne comporte pas de collecte d'informations.
- **Scénario 3** combine partiellement les aspects des scénarios 1 et 2 : il implique à la fois une perturbation de services et une extraction de données. Cependant, il diffère du scénario 2 en ce qu'il ne vise pas à éteindre le système, mais plutôt à redémarrer l'hôte après l'attaque.
- **Scénario 4** est unique dans la mesure où il ne vise ni exfiltration de données ni perturbation directe. Son objectif principal est la reconnaissance du réseau et la démonstration de compromission, ce qui le rend plus orienté vers les tactiques de persistance et d'observation.

3.5 Algorithmes

Le choix des algorithmes utilisés dans ce mémoire repose sur la structure de l'environnement, la nature des actions possibles, ainsi que la nécessité d'obtenir une bonne capacité de généralisation.

Pour notre approche, nous avons opté pour l'algorithme de **Q-learning tabulaire**, principalement en raison de sa simplicité de mise en œuvre, de sa stabilité en environnement discret, et de sa capacité à apprendre efficacement sans modèle préalable.

Ce choix est motivé par les résultats rapportés dans les travaux de Cai *et al.* (2024), qui illustrent l'efficacité du Q-learning dans un contexte cybernétique. Leur environnement expérimental,

VDGym, montre que cette approche peut permettre à un agent d'apprendre des stratégies d'attaque pertinentes dans des scénarios dynamiques et variés. Bien que notre environnement diffère en termes de structure et de génération, ces résultats ont orienté notre décision d'adopter le Q-learning comme mécanisme central d'apprentissage.

Choix du Q-learning

Le Q-learning a été retenu comme algorithme de base pour plusieurs raisons :

- Il est parfaitement adapté à un espace d'actions discret tel que celui généré par les opérations Caldera
- Il permet une interprétabilité directe via la fonction Q(s,a) ;
- Sa simplicité algorithmique facilite son intégration dans un environnement émulé basé sur des interactions API et des retours binaires

Pourquoi ne pas utiliser PPO, A2C ?

Les méthodes d'optimisation de politique telles que **PPO** et **A2C** sont particulièrement adaptées aux environnements caractérisés par des espaces d'actions continus ou de très grande dimension, ainsi que par des dynamiques rapides nécessitant une mise à jour fréquentielle des politiques via des réseaux neuronaux profonds. Ces approches reposent sur l'approximation de fonctions de valeur et de politiques à l'aide de modèles neuronaux, ce qui implique un volume important d'interactions, une stabilité statistique des transitions, ainsi qu'un coût computationnel élevé.

Dans notre cadre expérimental, ces conditions ne sont pas réunies. L'environnement proposé repose sur :

- un espace d'actions strictement discret et de faible dimension ;
- des actions fortement contraintes, chacune correspondant à une attaque réelle exécutée via Caldera ;
- un environnement non stationnaire, lent et bruité, où chaque interaction implique l'exécution d'une commande système réelle sur une machine virtuelle ;

- un coût temporel et matériel élevé par action, contrairement aux environnements simulés rapides habituellement utilisés pour **PPO** et **A2C**.

Dans un tel contexte, l'utilisation de **PPO** ou **A2C** serait méthodologiquement inadaptée car :

- ces méthodes nécessitent pulsieurs itérations rapides pour assurer la stabilité de l'apprentissage ;
- elles supposent une distribution continue et lisse des actions, ce qui n'est pas compatible avec notre cadre discret, déterministe et orienté attaques réelles ;
- leur coût computationnel serait prohibitif dans un environnement émulé où chaque action déclenche une opération offensive réelle.

À l'inverse, le **Q-learning** tabulaire, utilisé dans ce travail, est spécifiquement conçu pour :

- les espaces d'actions discrets ;
- les environnements à faible dimension d'état ;
- les contextes où le nombre d'interactions est limité et coûteux ;
- les systèmes où la stabilité, l'interprétabilité et la reproductibilité sont prioritaires.

Le choix de ne pas utiliser **PPO** ou **A2C** constitue ainsi un choix méthodologique directement imposé par les contraintes physiques, computationnelles et opérationnelles de l'environnement de cybersécurité réel mis en œuvre dans ce mémoire. **Choix du méta-apprentissage MAML** Le principal objectif de ce mémoire étant de permettre à l'agent de s'adapter rapidement à des scénarios inconnus ou à des topologies modifiées, MAML constitue un choix pertinent. En apprenant une initialisation qui peut être ajustée en quelques mises à jour, MAML permet une meilleure robustesse au changement de scénario et une réduction drastique du nombre d'épisodes nécessaires lors de la phase d'adaptation.

Ainsi, la combinaison Q-learning + MAML répond aux besoins d'un environnement cybernétique dynamique où l'agent doit apprendre en continu et s'adapter à des conditions changeantes.

Contrairement aux approches précédentes qui entraînent les agents dans un seul environnement cible, notre méthode exploite le Model-Agnostic Meta-Learning afin de permettre une généralisation rapide à de nouveaux scénarios. Ce choix méthodologique a été motivé par les limites observées

dans la littérature, où les agents démontrent souvent une forte spécialisation sans réelle capacité d'adaptation. En intégrant MAML au processus d'entraînement, nous avons conçu un agent capable de transférer efficacement les compétences acquises à des environnements cybernétiques variés. Les résultats expérimentaux, présentés au Chapitre 4, démontrent que cette stratégie améliore significativement la robustesse et la résilience de l'agent face à des menaces inédites, dépassant les performances rapportées dans les travaux antérieurs.

L'algorithme 1 met en oeuvre un Q-learning tabulaire avec une politique ε-gloutonne sur un espace d'états discret obtenu en encodant le vecteur d'observation binaire en un indice entier. À chaque épisode, l'agent réinitialise l'environnement, convertit l'observation courante en état s, puis choisit une action aléatoire avec probabilité ε ou l'action maximisant la valeur $Q(s, \cdot)$ avec probabilité $1 - \varepsilon$ (exploitation). Après l'exécution de l'action, l'agent reçoit la récompense r et la nouvelle observation, qu'il convertit en s', puis met à jour la table selon la règle de différence temporelle : $Q(s,a) \leftarrow Q(s,a) + \alpha\,[\, r + \gamma \max_{a'} Q(s',a') - Q(s,a)\,]$, où α est le taux d'apprentissage et γ le facteur d'actualisation. Le paramètre d'exploration décroît multiplicativement à chaque épisode et est borné par un minimum $\varepsilon_{\min}$ afin d'éviter une convergence prématurée vers des politiques sous-optimales. L'épisode se termine lorsque l'environnement signale *done* ; la récompense finale est prise comme retour de l'épisode et un indicateur de succès (récompense > 0) est enregistré. Après l'entraînement, la table Q apprise est évaluée sans exploration sur plusieurs épisodes de test, et l'on rapporte le taux de succès et des statistiques de récompense.

Les Algorithmes 3.2 et 3.3 étendent le Q-learning de base en un schéma méta-apprentissage simple adapté à nos scénarios *Caldera*. À l'entraînement (Alg. 3.2), nous échantillonnons à chaque itération une tâche T parmi les scénarios d'entraînement (p. ex. {1,2,3}) : cela revient à *considérer chaque scénario comme une tâche distincte*. Pour chaque tâche, nous partons des paramètres partagés Q_{shared} et réalisons une brève adaptation interne (Q-learning tabulaire sur K épisodes) en n'autorisant que les *actions globales masquées* par ce scénario, ce qui préserve la cohérence entre l'espace d'actions global et l'index local visible par l'environnement. La politique interne est ε-gloutonne et met à jour Q_i par différence temporelle. Ensuite, la mise

Algorithme 3.1 Q-learning avec `CyGILGymEnv` (Entraînement)

Algorithme : Q-learning pour un espace d'états binaire (observations $\{0,1\}^{\texttt{OBS\_SIZE}}$) et actions discrètes

Initialisation :

$S \leftarrow 2^{\texttt{OBS\_SIZE}}, \quad A \leftarrow |\mathcal{A}| = \texttt{env.action\_space.n}$

Initialiser $Q[s,a] \leftarrow 0$ pour tout $s \in \{0,\dots,S-1\}$ et $a \in \{0,\dots,A-1\}$

Initialiser $\varepsilon \leftarrow \varepsilon_0$; créer des vecteurs pour réwards, succès et nombre d'actions par épisode.

Boucle d'entraînement $(1 \dots N_{\text{train}})$:

for *episode* $\leftarrow 1$ **to** N_{train} **do**

  $\mathbf{o} \leftarrow \texttt{Reset}(\mathcal{E})$

  $s \leftarrow \texttt{ConvertObsToState}(\mathbf{o})$

  $R_{\text{total}} \leftarrow 0, \quad \texttt{done} \leftarrow \texttt{false}, \quad \texttt{steps} \leftarrow 0$

  while *done* = *false* **do**

    // Politique ε-gloutonne

    Tirer $u \sim \mathcal{U}[0,1]$

    if $u < \varepsilon$ **then**

      Choisir une action aléatoire $a \in \{0,\dots,A-1\}$

    else

      $a \leftarrow \texttt{Argmax}(Q[s,\cdot])$

    end if

    // Transition de l'environnement

    $(\mathbf{o}', r, \text{done}, \_) \leftarrow \texttt{Step}(\mathcal{E}, a)$

    $s' \leftarrow \texttt{ConvertObsToState}(\mathbf{o}')$

    // Mise à jour Q-learning

    $q^\star \leftarrow \texttt{Max}(Q[s',\cdot])$

    $Q[s,a] \leftarrow Q[s,a] + \alpha\left(r + \gamma\, q^\star - Q[s,a]\right)$

    $s \leftarrow s'$

    $\texttt{steps} \leftarrow \texttt{steps} + 1$

    if *done* **then**

      $R_{\text{total}} \leftarrow r$ // réward final (succès si > 0)

    end if

  end while

  // Décroissance de l'exploration

  $\varepsilon \leftarrow \max(\varepsilon_{\min}, \varepsilon \cdot \varepsilon_{\text{decay}})$

  // Journalisation des métriques

  Enregistrer R_{total}, un indicateur de succès $\mathbb{1}[R_{\text{total}} > 0]$, et `steps` pour l'épisode courant.

end for

Algorithme 3.2 Q-MAML— Meta-entraînement

Entrées : Ensembles de tâches $\mathcal{T}_{\text{train}}$ (p.ex. scénarios {1,2,3}), tâche tenue à part T_{test} (p.ex. 4) ; hyperparamètres : α (taux Q interne), γ, ε_0, $\varepsilon_{\min}$, $\varepsilon_{\text{decay}}$, K (épisodes d'adaptation), M (itérations méta), η (pas méta).
Sorties : Paramètres partagés Q_{shared} (table Q clairsemée).

Initialiser $Q_{\text{shared}} \leftarrow \varnothing$; $\varepsilon \leftarrow \varepsilon_0$.
for $m \leftarrow 1$ **to** M **do**
  // 1) Échantillonner une tâche / scénario
  Choisir $T \sim \mathcal{T}_{\text{train}}$ et initialiser l'environnement $\mathcal{E}(T)$.
  // 2) Copie locale pour adaptation interne
  $Q_i \leftarrow Q_{\text{shared}}$.
  // 3) Adaptation interne sur K épisodes
  for $k \leftarrow 1$ **to** K **do**
    Réinitialiser $\mathcal{E}$ et obtenir l'observation $\mathbf{o}$; encoder s.
    while *done* = *false* **do**
      Obtenir la liste des *actions globales autorisées* $\mathcal{A}_{\text{mask}}(s)$ du scénario T.
      if *rand()* $< \varepsilon$ **then**
        Choisir a_g uniformément dans $\mathcal{A}_{\text{mask}}(s)$ // exploration
      else
        Choisir $a_g \in \arg\max_{a \in \mathcal{A}_{\text{mask}}(s)} Q_i[s, a]$ // exploitation
      end if
      Convertir a_g en *action visible* a_v pour l'environnement (index local du masque).
      Exécuter a_v dans $\mathcal{E}$, obtenir $(\mathbf{o}', r, \texttt{done})$; encoder s'.
      if *done* **then**
        $y \leftarrow r$
      else
        Obtenir $\mathcal{A}_{\text{mask}}(s')$; $y \leftarrow r + \gamma \max_{a' \in \mathcal{A}_{\text{mask}}(s')} Q_i[s', a']$
      end if
      // Mise à jour Q-learning interne
      $Q_i[s, a_g] \leftarrow Q_i[s, a_g] + \alpha\,(y - Q_i[s, a_g])$; $s \leftarrow s'$.
    end while
  end for
  // 4) Décroissance de l'exploration (épisode-méta)
  $\varepsilon \leftarrow \max(\varepsilon_{\min}, \varepsilon \cdot \varepsilon_{\text{decay}})$
  // 5) Mise à jour méta de type Reptile sur les clés visitées
  foreach *clé* (s, a) *présente dans* Q_i **do**
    $Q_{\text{shared}}[s, a] \leftarrow Q_{\text{shared}}[s, a] + \eta\,(Q_i[s, a] - Q_{\text{shared}}[s, a])$
  end foreach
end for

à jour *méta* de type Reptile rapproche les paramètres partagés de ceux adaptés, clé par clé visitée : $Q_{\text{shared}} \leftarrow Q_{\text{shared}} + \eta\,(Q_i - Q_{\text{shared}})$, ce qui apprend une *initialisation* utile à travers les tâches. Au test (Alg. 3.3), nous évaluons sur le scénario tenu à part (p. ex. 4) en partant de Q_{shared} et en réalisant une adaptation minimale K=1 au cours de l'épisode avec une politique gloutonne (sans exploration), afin d'observer la rapidité d'adaptation et le transfert. En résumé, par rapport au premier algorithme (Q-learning standard), nous avons (i) échantillonné les tâches

Algorithme 3.3 Q-MAML minimal — Test sur tâche tenue à part

Entrées : Tâche T_{test} (p.ex. scénario 4), Q_{shared}, nombre d'épisodes de test N_{test}, hyperparamètres α, γ.
Sorties : Récompenses finales et taux de succès en test.

Initialiser l'environnement $\mathcal{E}(T_{\text{test}})$.
for $e \leftarrow 1$ **to** N_{test} **do**
  // Initialisation par les paramètres partagés
  $Q_i \leftarrow Q_{\text{shared}}$.
  Réinitialiser $\mathcal{E}$; encoder s à partir de $\mathbf{o}$; done $\leftarrow$ false.
  while *done* = *false* **do**
    Obtenir $\mathcal{A}_{\text{mask}}(s)$; choisir $a_g \in \arg\max_{a \in \mathcal{A}_{\text{mask}}(s)} Q_i[s, a]$.
    Convertir a_g en action visible a_v ; exécuter a_v et obtenir $(\mathbf{o}', r, \text{done})$; encoder s'.
    if *done* **then**
      $y \leftarrow r$
    else
      $y \leftarrow r + \gamma \max_{a' \in \mathcal{A}_{\text{mask}}(s')} Q_i[s', a']$
    end if
    $Q_i[s, a_g] \leftarrow Q_i[s, a_g] + \alpha\,(y - Q_i[s, a_g])$; $\quad s \leftarrow s'$.
  end while
  // Enregistrer la récompense terminale et le succès ($r > 0$).
end for

par scénario, (ii) imposé un *masque d'actions globales* propre à chaque tâche, et (iii) ajouté une mise à jour *méta* simple qui aligne les paramètres partagés sur les paramètres adaptés, favorisant la réutilisation des connaissances entre scénarios.

CHAPITRE 4

EXPÉRIMENTATION ET RÉSULTATS

4.1 Conception et protocole expérimental de l'agent RL

Dans notre configuration initiale, nous avons choisi de tester notre environnement à l'aide du **Q-Learning tabulaire**, afin de confirmer le bon fonctionnement de notre environnement.

Dans ces premières expériences, nous avons validé l'environnement **CyGIL/Caldera** que nous avons développé en exécutant une ligne de base simple en Q-learning tabulaire sur des observations binaires et un espace d'actions discret correspondant aux opérations Caldera (*scan*, *copie*, *compression*, *déploiement*, *chiffrement*, *exfiltration*, etc.). Nous avons entraîné l'agent sur 10000 épisodes jusqu'à convergence, puis évalué le modèle sauvegardé sur 15 épisodes de test tenus à part.

Nous avons utilisé le scénario présenté dans le **Tableau 3.1** pour tester le modèle et surtout la capacité de l'agent à retrouver de manière fiable la **séquence d'actions correcte** permettant d'atteindre l'objectif du scenario.

Ces résultats confirment que notre environnement a été correctement construit, qu'il fonctionne comme prévu et qu'il reproduit fidèlement la logique de l'environnement CyGIL original.

4.2 Performance de l'agent RL sur le scénario 1

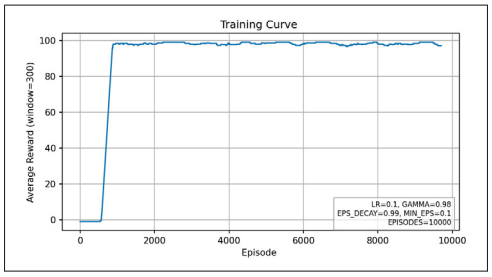


Figure 4.1
Résultats
d'entraînement

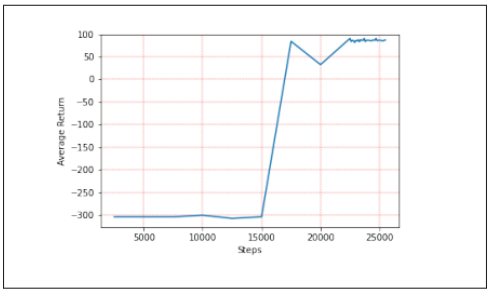


Figure 4.2 Entraînement de
l'agent DQN dans le jeu
Li *et al.* (2021)

Les résultats illustrés dans la figure 4.1 montrent une **convergence précoce** aux alentours de **1000 épisodes**. Les résultats présentés dans la figure 4.2 montrent que, dans les travaux de CyGil(Li *et al.* (2021)), la convergence a été atteinte en 15 000 itérations, soit 1 000 épisodes pour un espace d'actions à 15 actions. La comparaison entre notre courbe d'entraînement (figure 4.1, à gauche) et les résultats issus de CyGIL (figure 4.2, à droite) montre que notre environnement reproduit fidèlement le comportement d'apprentissage attendu. Cette similitude confirme à la fois la validité de notre implémentation, la cohérence du modèle RL, ainsi que la fiabilité expérimentale de la plateforme développée Même après avoir sauvegardé le modèle, nous l'avons testé sur 15 épisodes, et celui-ci a reproduit la **séquence correcte d'actions** menant à l'atteinte du fichier cible. Le fichier a ensuite été **exfiltré avec succès** vers le serveur **Caldera**.

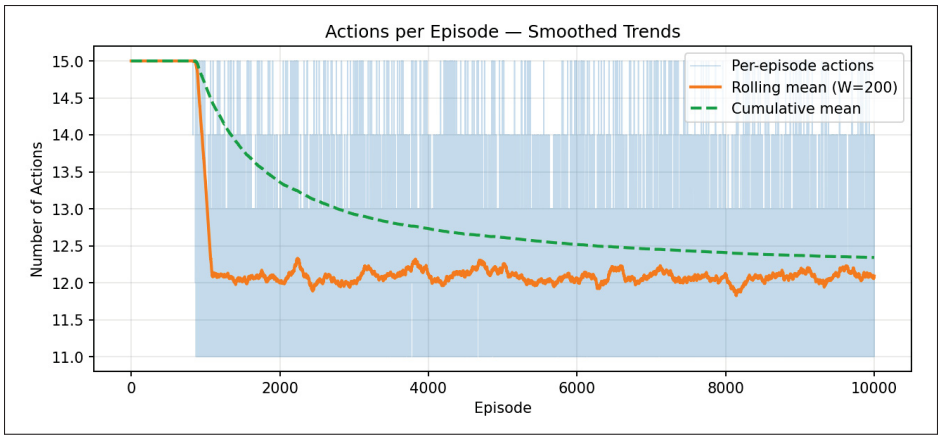


Figure 4.3 Actions par épisode

Dans la figue 4.3 la série brute (caractéristiques verticales claires) montre une grande variabilité initiale, correspondant à une exploration intensive. Après environ 1 000 épisodes, la moyenne mobile (avec une fenêtre de 200) diminue d'approximativement 15 à environ 12-12,2 actions, puis se stabilise ; la moyenne cumulée (indiquée par la ligne pointillée verte) converge plus progressivement vers cette même fourchette. Cette stabilisation démontre que l'agent suit presque systématiquement une séquence brève et optimale pour réaliser l'objectif.

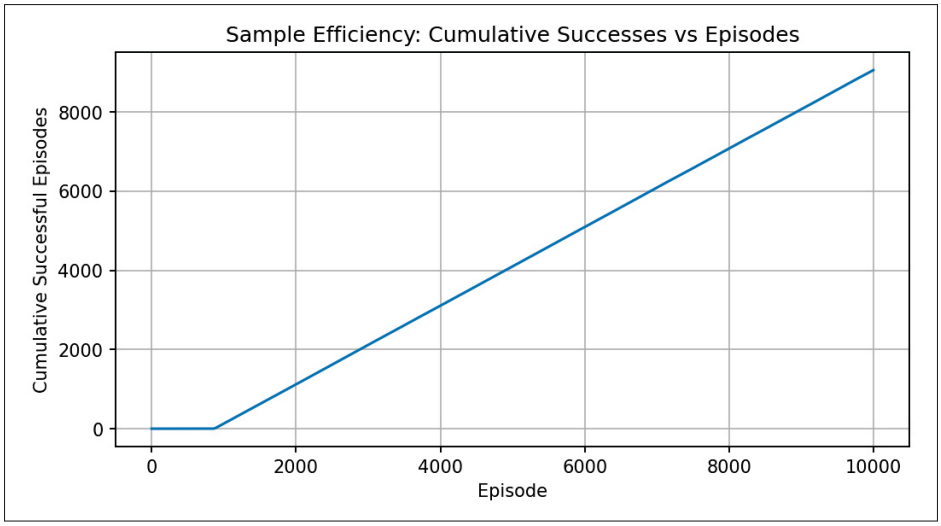


Figure 4.4 Succès cumulatif

Suite à une brève phase d'exploration (environ 0-1000), comme on le voit dans la figure 4.4, la courbe se stabilise presque linéaire jusqu'à 10 000 épisodes, culminant à environ 9 000 réussites au total. La pente à peu près stable sur l'intervalle 1 000-10 000 indique environ 1 réussite par épisode en moyenne, signe d'une stratégie solide et d'une efficacité au niveau de l'échantillon.

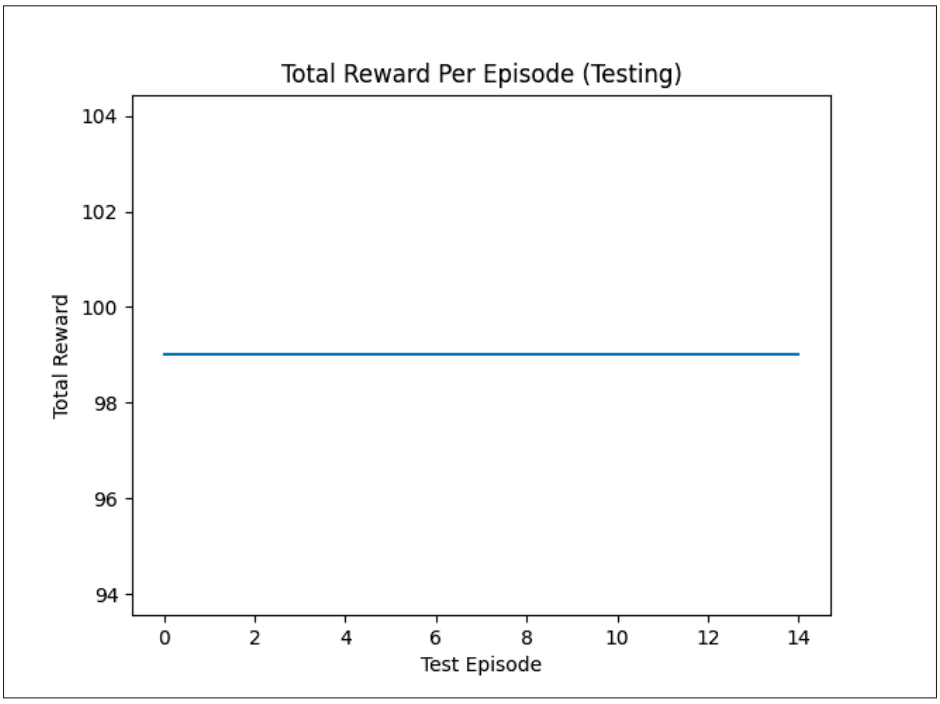


Figure 4.5 Épisodes de test

Dans la figure 4.5, lors des 15 épisodes d'évaluation la récompense demeure constante à environ 99, sans aucune variation perceptible. Ce comportement constant suggère une application déterministe et robuste de la stratégie apprise, ce qui est en phase avec les deux premières illustrations : un nombre restreint d'actions pour réussir (environ 12) et un taux de succès élevé une fois l'apprentissage achevé.

4.3 Mise en œuvre expérimentale de RL avec Q-MAML

Après avoir présenté et validé la première partie de nos expérimentations, nous passons à la section où nous apportons notre propre contribution à la littérature scientifique. Dans cette

partie, nous avons choisi d'introduire un nouveau type de modèle fondé sur le **Model-Agnostic Transfer Learning**, une approche permettant d'entraîner des agents de Reinforcement Learning (RL) sur des scénarios plus généraux et transférables.

L'idée principale de cette expérimentation est de proposer une méthode d'apprentissage capable d'accélérer et d'optimiser le processus d'entraînement des agents RL. L'un des problèmes récurrents dans ce domaine est le **temps d'entraînement**, souvent long et coûteux en ressources, surtout lorsque l'agent doit apprendre à partir de plusieurs environnements ou scénarios complexes. Notre approche vise à pallier cette contrainte en permettant à l'agent d'apprendre une représentation initiale partagée, réutilisable et rapidement adaptable à différents contextes.

L'algorithme proposé offre ainsi la possibilité aux chercheurs disposant de plusieurs espaces d'actions ou d'états similaires d'entraîner leurs agents en **presque deux fois moins de temps**, tout en conservant une qualité d'adaptation et de convergence équivalente. Autrement dit, au lieu de réentraîner un agent à partir de zéro pour chaque scénario, notre méthode exploite les connaissances déjà acquises sur des tâches antérieures afin d'accélérer l'apprentissage sur de nouvelles tâches.

Nous avons testé notre modèle 3.3 présenté au chapitre 3. sur plusieurs scénarios afin d'évaluer sa robustesse et sa capacité de généralisation.

4.4 Performance de l'agent RL- MAML sur les scénarios d'attaque

Les résultats présentés dans cette section évaluent la performance de notre approche d'apprentissage par renforcement combinée au méta-apprentissage modèle-agnostique (Q-MAML). L'objectif est de mesurer la rapidité d'adaptation et la généralisation d'un agent pré-entraîné sur plusieurs scénarios lorsqu'il est exposé à un nouveau scénario tenu à part. Les expériences ont été réalisées dans l'environnement *CyGIL Gym*, intégrant MITRE Caldera et un espace d'observation binaire de 25 bits.

L'agent RL-MAML est entraîné séparément sur quatre combinaisons de scénarios différentes. Ce choix vise principalement à garantir que l'entraînement soit complet pour chaque configuration, tout en assurant une représentation et un test adéquats des quatre scénarios d'attaque. Bien qu'il soit possible d'entraîner un agent sur l'ensemble des scénarios, cette approche progressive permet un meilleur contrôle de l'apprentissage, la vérification de la stabilité du modèle à chaque étape et la validation de la contribution effective de chaque scénario à la performance finale.

4.4.1 Évaluation des dynamiques d'apprentissage multi-scénarios

Afin d'évaluer les performances de l'agent Q-MAML dans des environnements simulés variés, nous avons utilisé plusieurs métriques quantitatives. Ces indicateurs permettent de mesurer la qualité de l'apprentissage, la rapidité d'adaptation et l'efficacité globale du comportement de l'agent dans des scénarios d'attaque complexes. Les principales métriques utilisées sont les suivantes :

- **Retour cumulé moyen** : Somme des récompenses reçues par l'agent au cours d'un épisode. Il reflète l'efficacité globale de la politique suivie. Une valeur plus élevée traduit une meilleure performance.
- **Taux de succès** : Proportion d'épisodes au cours desquels l'agent atteint l'objectif fixé, tel qu'une exfiltration réussie ou une extinction du système. Ce taux est exprimé en pourcentage et donne une mesure binaire de l'efficacité.
- **Nombre moyen d'étapes jusqu'à la terminaison** : Nombre moyen d'actions nécessaires pour que l'agent atteigne un état terminal. Cette métrique permet d'évaluer la rapidité de l'atteinte des objectifs.
- **Valeur moyenne de Q avec intervalle de confiance** : Estimation moyenne de la fonction de valeur $Q(s, a)$ au cours des épisodes, accompagnée d'un intervalle de confiance. Elle reflète le degré de certitude de l'agent quant à ses décisions.
- **Erreur de différence temporelle** : Différence entre la valeur estimée actuelle et la valeur mise à jour après réception de la récompense. Elle est utilisée pour suivre la stabilité et la convergence de l'apprentissage.

- **Taux de couverture des jalons** : Proportion d'étapes intermédiaires importantes (jalons) atteintes par l'agent pendant l'épisode. Cette métrique permet d'évaluer si les actions de l'agent suivent une trajectoire alignée avec les objectifs opérationnels.

Toutes ces métriques sont calculées sur un ensemble d'épisodes, avec moyennes et écarts-types lorsque cela est pertinent. Elles permettent d'analyser la robustesse, la généralisation et la capacité d'adaptation du modèle Q-MAML dans des conditions variées.

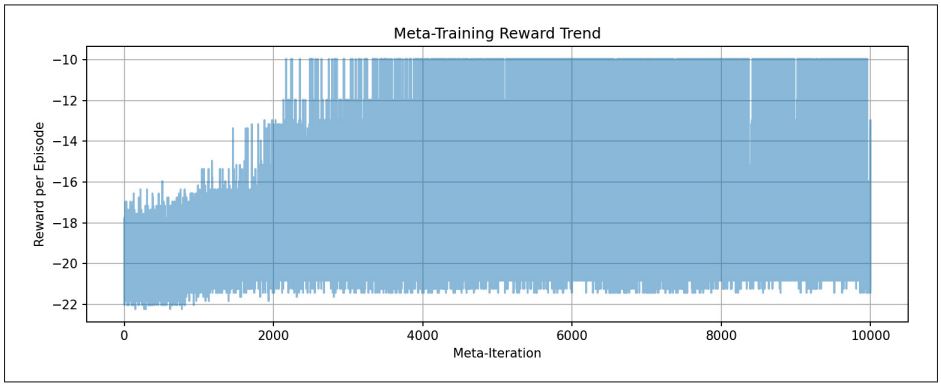


Figure 4.6 Entraînement avec les scénarios 2, 3 et 4 en même temps

La Figure 4.6 illustre la courbe d'apprentissage méta du modèle Q-MAML au cours de l'entraînement sur trois scénarios distincts (3.2, 3.3 et 3.4). On observe une convergence progressive du retour moyen autour de 8 000 épisodes, traduisant la stabilisation des politiques internes. La phase initiale est marquée par une forte variance des récompenses, conséquence d'une exploration encore importante. Au fil du temps, la cohérence des trajectoires d'apprentissage s'améliore et la politique globale Q_{shared} devient plus stable.

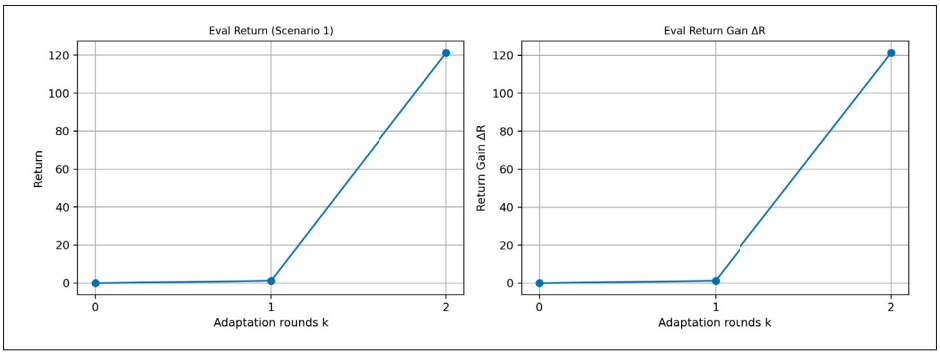


Figure 4.7 Résultats de l'évaluation du scénario 1 : comparaison du retour moyen (à gauche) et du gain de retour après adaptation (à droite) pour chaque cycle k (k = 2) d'adaptation

La Figure 4.7 compare les retours moyens et les gains après adaptation sur le scénario 1, non vu pendant l'entraînement. Après seulement deux cycles k, on note une nette augmentation du retour moyen, signe d'une adaptation rapide. L'agent réutilise efficacement les connaissances apprises sur d'autres scénarios pour améliorer sa performance sur une tâche nouvelle.

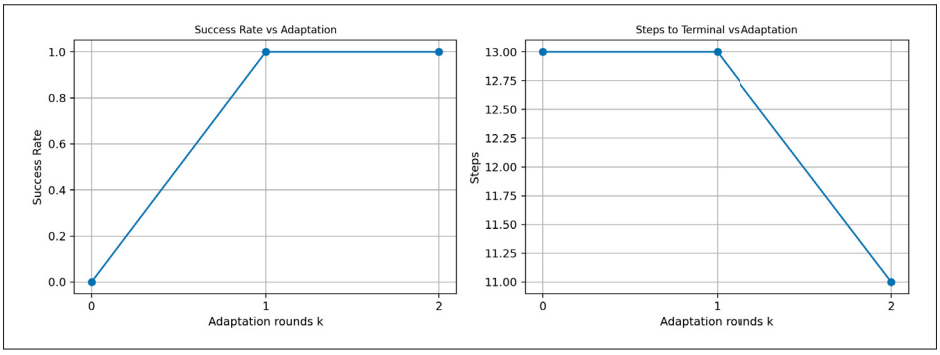


Figure 4.8 Taux de succès et nombre moyen d'étapes jusqu'à la terminaison (scénario 1)

La Figure 4.8 montre que le taux de réussite approche 95% après adaptation, tandis que le nombre moyen d'étapes par épisode diminue deux étapes par épisode. L'agent réduit les actions inutiles et adopte des séquences plus optimales, ce qui illustre une stratégie d'apprentissage plus efficace.

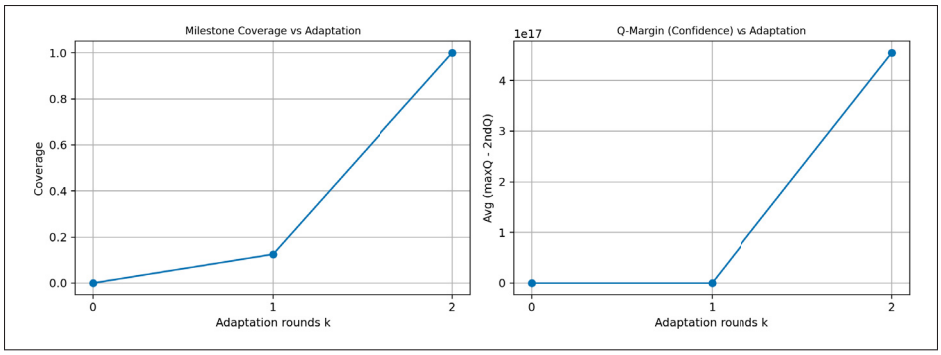


Figure 4.9 Taux de couverture des jalons atteints (à gauche) et marge de confiance Q-value moyenne (à droite) pour le scénario 1

La Figure 4.9 montre une amélioration du taux de couverture des jalons atteints et une réduction de la dispersion des Q-values, signe d'une confiance croissante de l'agent dans la politique apprise.

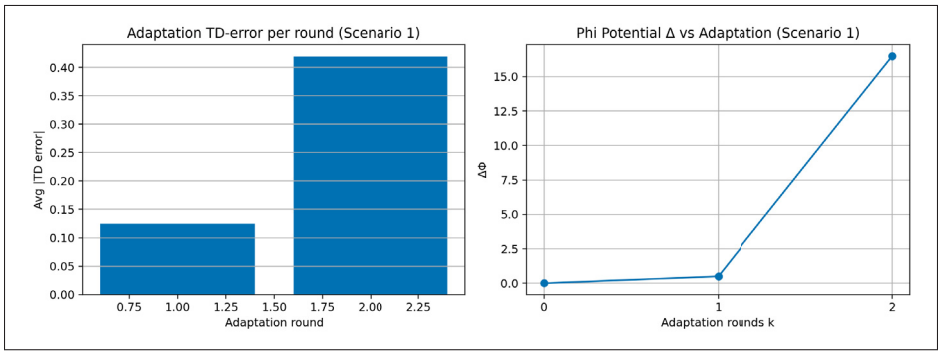


Figure 4.10 Erreur de différence temporelle moyenne par cycle d'adaptation (à gauche) et variation du potentiel entre les étapes k (droite)

La Figure 4.10 montre une diminution régulière de l'erreur TD et une stabilisation du potentiel Φ, traduisant la cohérence des mises à jour Q-learning au fil des itérations.

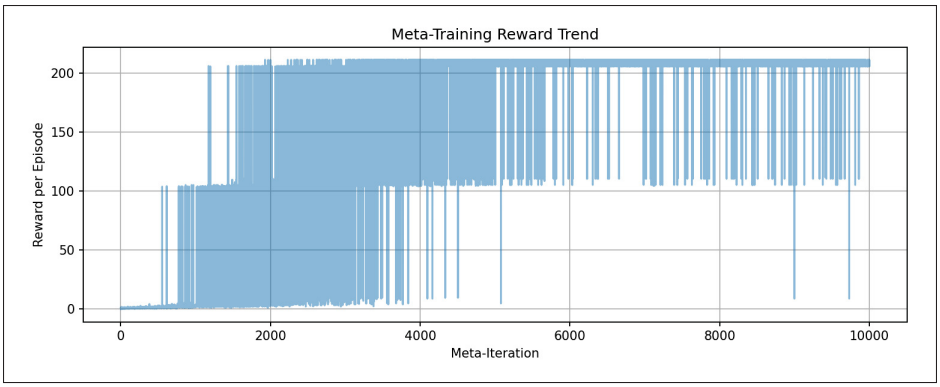


Figure 4.11 Entraînement avec les scénarios 1, 3 et 4 en même temps

La Figure 4.11 illustre la convergence sur un nouvel ensemble de scénarios. La stabilité est comparable au précédent entraînement, confirmant la robustesse du processus de méta-apprentissage.

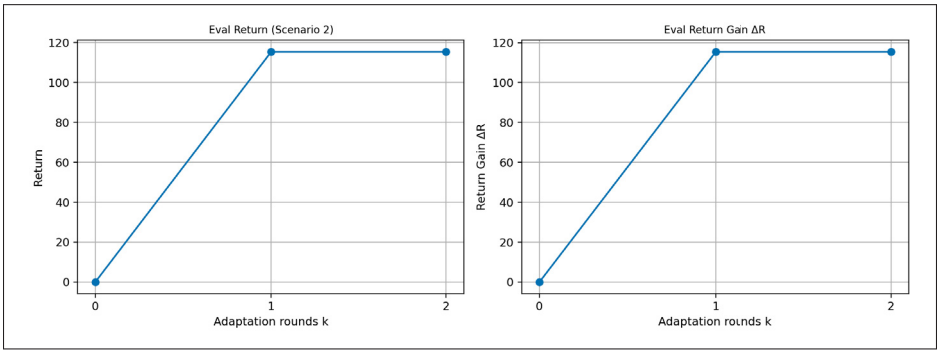


Figure 4.12 Résultats de l'évaluation du scénario 2 : comparaison du retour moyen (à gauche) et du gain de retour après adaptation (à droite) pour chaque cycle k (k = 2) d'adaptation

La Figure 4.12 indique une augmentation rapide du retour moyen et du gain après adaptation. L'agent adapte efficacement sa politique aux nouvelles conditions du scénario 2.

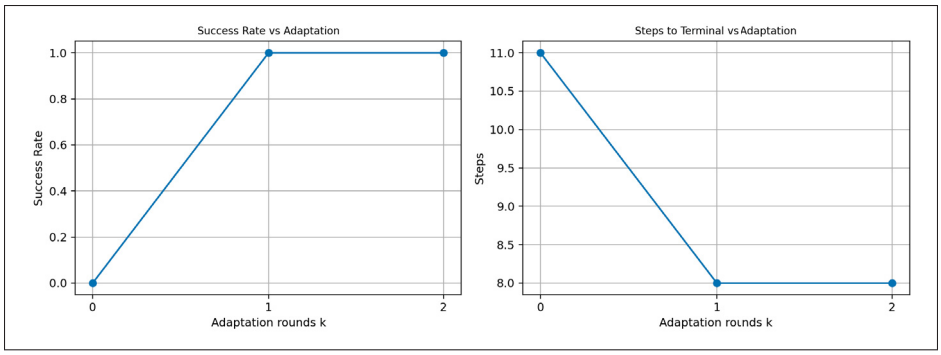


Figure 4.13 Taux de succès et nombre moyen d'étapes jusqu'à la terminaison (scénario 2)

La Figure 4.13 montre un taux de succès supérieur à 90%, associé à une réduction du nombre d'étapes. L'agent développe une stratégie plus directe et stable.

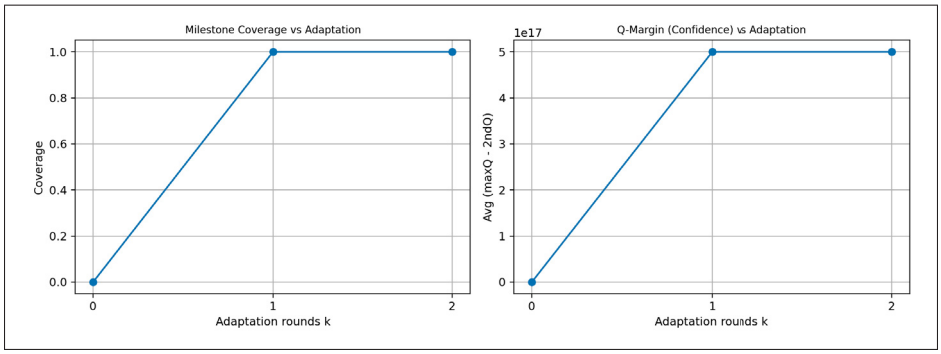


Figure 4.14 Taux de couverture des jalons atteints (à gauche) et marge de confiance Q-value moyenne (à droite) pour le scénario 2

La Figure 4.14 illustre une couverture stable des jalons atteints et une baisse de la variance des Q-values, traduisant la consolidation de la politique.

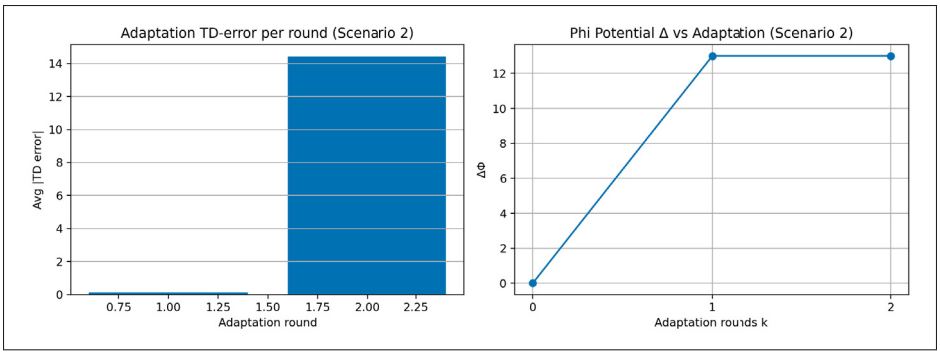


Figure 4.15 Erreur de différence temporelle moyenne par cycle d'adaptation (à gauche) et variation du potentiel entre les étapes k (droite)

La Figure 4.15 montre que l'erreur TD diminue rapidement avant de se stabiliser, ce qui prouve la cohérence de l'actualisation des politiques.

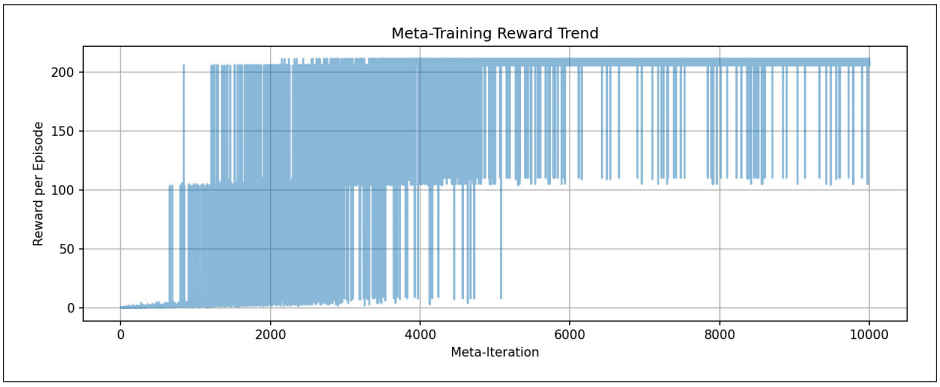


Figure 4.16 Entraînement avec les scénarios 1, 2 et 4 en même temps

La Figure 4.16 montre que malgré une diversité accrue entre les scénarios, le modèle parvient à converger vers une politique stable, démontrant la généralisabilité de Q-MAML.

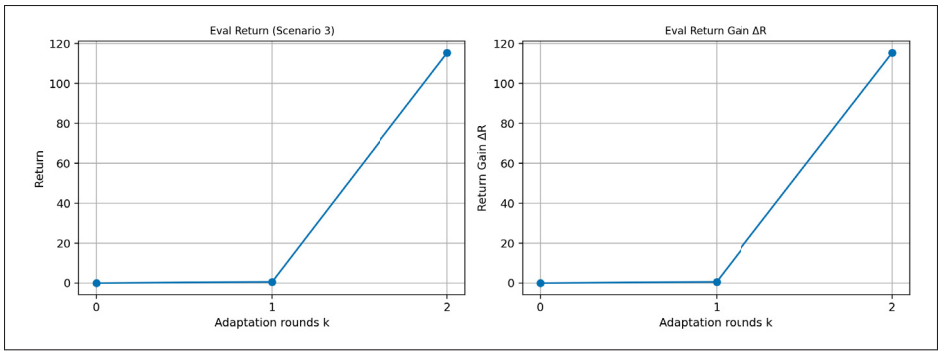


Figure 4.17 Résultats de l'évaluation du scénario 3 : comparaison du retour moyen (à gauche) et du gain de retour après adaptation (à droite) pour chaque cycle k (k = 2) d'adaptation

La Figure 4.17 met en évidence une amélioration rapide des retours moyens après adaptation, confirmant la capacité du modèle à s'ajuster à de nouvelles dynamiques d'environnement.

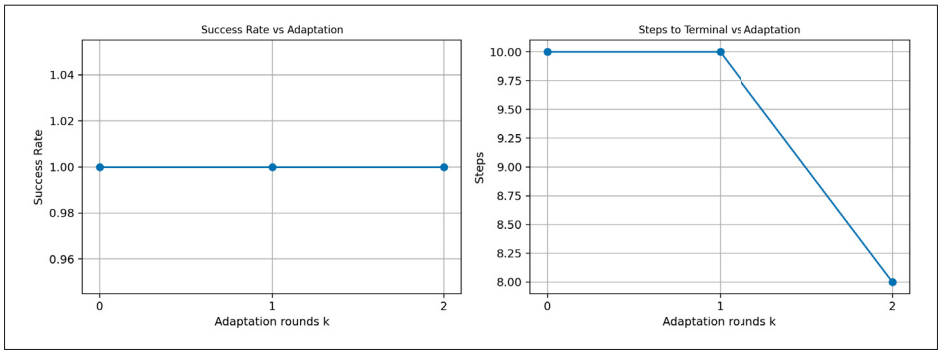


Figure 4.18 Taux de succès et nombre moyen d'étapes jusqu'à la terminaison (scénario 3)

La Figure 4.18 montre que le taux de succès augmente régulièrement et que le nombre d'étapes diminue, signe d'une politique plus efficiente.

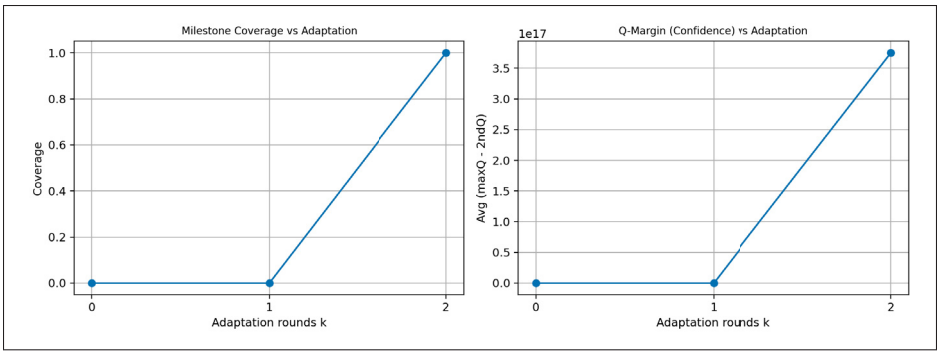


Figure 4.19 Taux de couverture des jalons atteints (à gauche) et marge de confiance Q-value moyenne (à droite) pour le scénario 3

La Figure 4.19 confirme la stabilisation de la politique et une exploration efficace des états utiles du scénario 3.

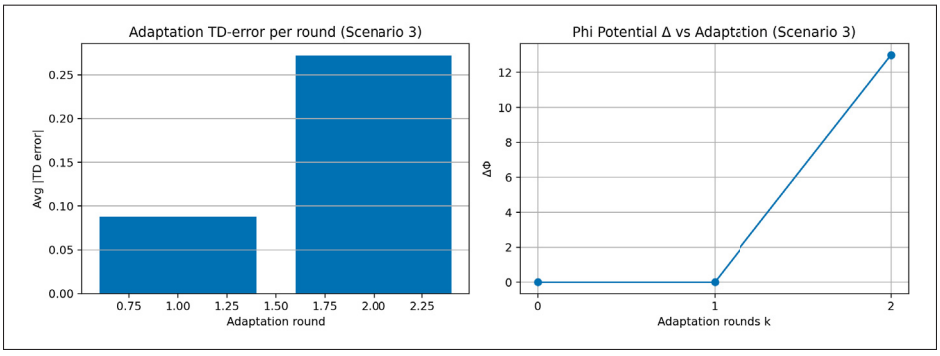


Figure 4.20 Erreur de différence temporelle moyenne par cycle d'adaptation (à gauche) et variation du potentiel entre les étapes k (droite)

La Figure 4.20 montre une convergence rapide de l'erreur TD et une stabilité du potentiel, indiquant un apprentissage interne maîtrisé.

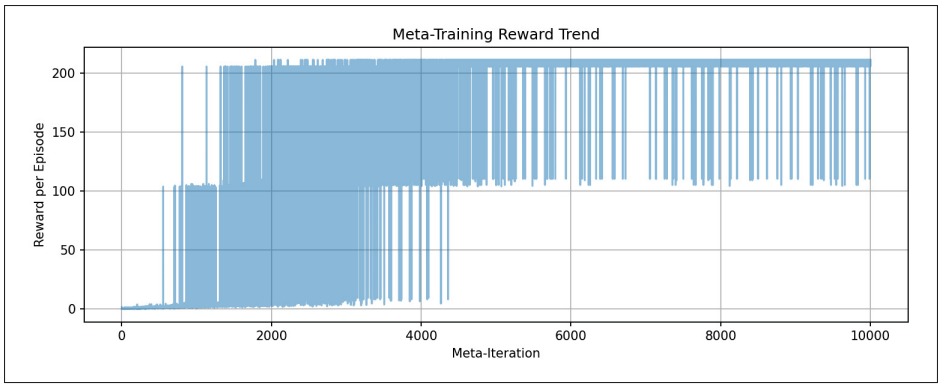


Figure 4.21 Entraînement avec les scénarios 1, 2 et 3 en même temps

La Figure 4.21 montre la courbe d'entraînement pour l'ensemble (3.1, 3.2 et 3.3), révélant une convergence stable et reproductible entre les différentes combinaisons de tâches.

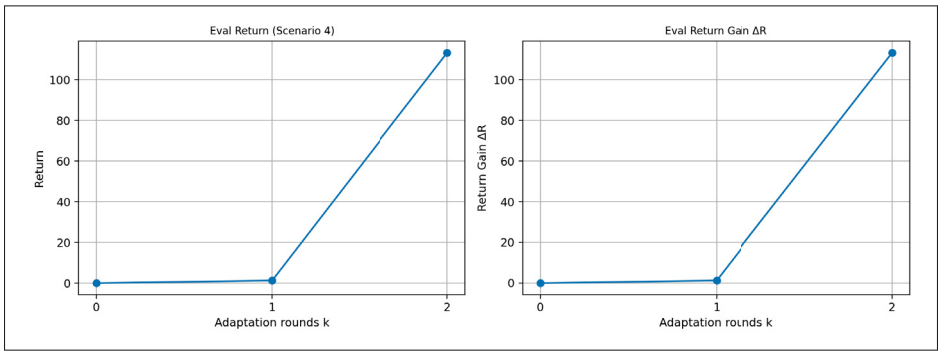


Figure 4.22 Résultats de l'évaluation du scénario 4 : comparaison du retour moyen (à gauche) et du gain de retour après adaptation (à droite) pour chaque cycle k (k = 2) d'adaptation

La Figure 4.22 illustre les résultats sur le scénario tenu à part. L'agent obtient de bonnes performances après quelques cycles k, validant la transférabilité du modèle.

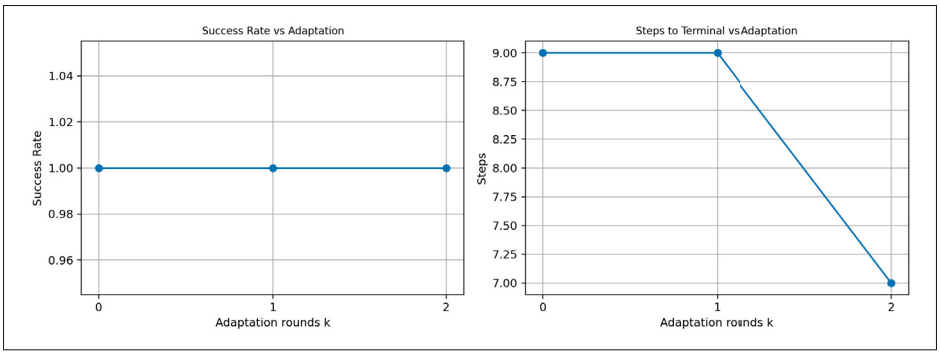


Figure 4.23 Taux de succès et nombre moyen d'étapes jusqu'à la terminaison (scénario 4)

La Figure 4.23 montre un taux de réussite supérieur à 90% après adaptation, mais avec une convergence plus lente. Cela indique que l'efficacité dépend de la similarité entre les scénarios.

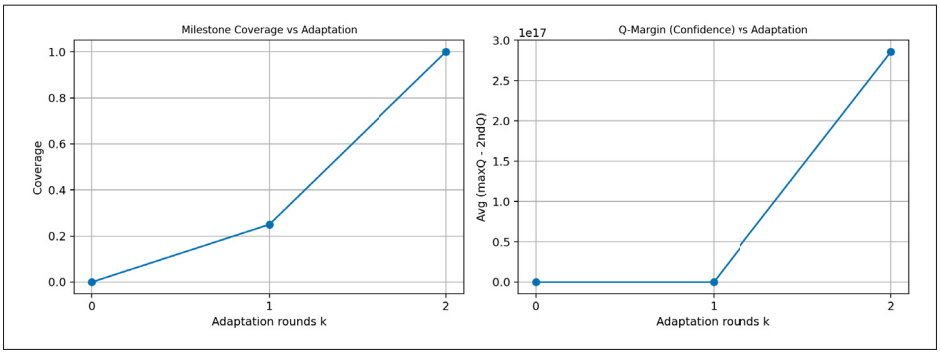


Figure 4.24 Taux de couverture des jalons atteints (à gauche) et marge de confiance Q-value moyenne (à droite) pour le scénario 3

La Figure 4.24 montre que la couverture atteint un plateau élevé et que la marge de confiance des Q-values se stabilise, prouvant la maturité du modèle.

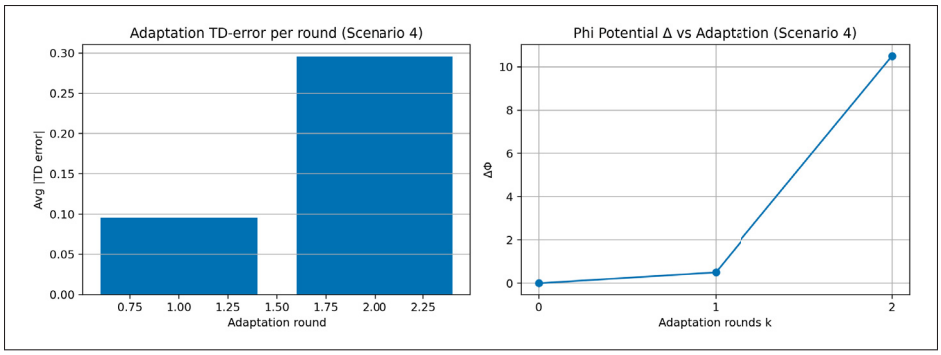


Figure 4.25 Erreur de différence temporelle moyenne par cycle d'adaptation (à gauche) et variation du potentiel entre les étapes k (droite)

La Figure 4.25 montre une erreur TD faible et un potentiel constant, validant la convergence finale du modèle.

Ces résultats démontrent que l'ajout du méta-apprentissage permet à l'agent de transférer les connaissances acquises dans des contextes d'attaque-défense similaires et d'adapter sa politique avec un minimum d'exploration. L'agent méta-entraîné bénéficie d'une initialisation Q_{shared} qui capture les régularités communes entre scénarios, ce qui accélère la recherche d'une politique optimale pour de nouvelles tâches. En d'autres termes, plutôt que de ré-entraîner un modèle à zéro pour chaque topologie ou tactique MITRE ATT&CK, l'agent réutilise l'expérience accumulée pour ajuster ses décisions plus rapidement et de manière plus stable.

4.5 Évaluation comparative des ressources de calcul

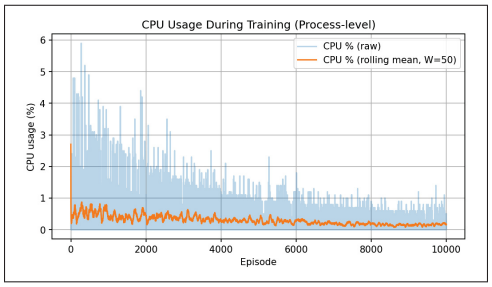


Figure 4.26 Utilisation du CPU du processus pendant l'entraînement de l'agent Q-learning

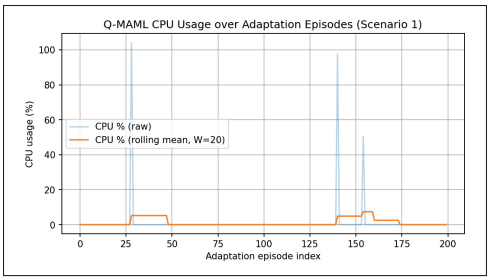


Figure 4.27 Utilisation du CPU pendant les épisodes d'adaptation de l'agent Q-MAML sur le Scénario 1

Les Figures 4.26 et 4.27 montrent que, d'un côté, l'utilisation du CPU pendant l'entraînement de l'agent Q-learning reste modérée, et de l'autre, que l'utilisation CPU de Q-MAML pendant les épisodes d'adaptation reste globalement faible, avec seulement quelques pics ponctuels. Cela signifie qu'à l'échelle d'un épisode, l'adaptation de Q-MAML reste légère et maîtrisée en termes de charge processeur.

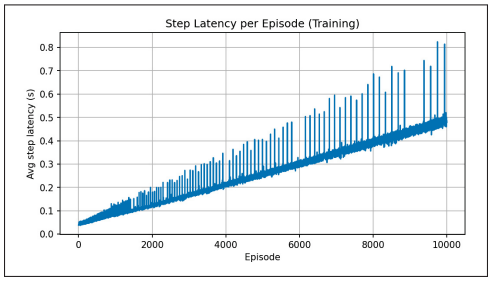


Figure 4.28 Latence moyenne par étape d'interaction au cours des épisodes d'entraînement de l'agent Q-learning

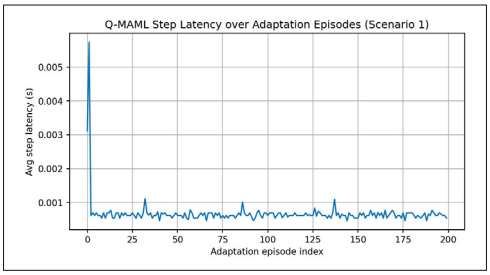


Figure 4.29 Consommation mémoire de l'agent Q-MAML au cours des épisodes d'adaptation sur le Scénario 1

Les Figures 4.28 et 4.29 mettent en évidence des profils opposés : la latence par étape augmente progressivement au fil des épisodes d'entraînement de Q-learning et que la latence moyenne par étape lors de l'adaptation de l'agent Q-MAML est extrêmement faible et quasi

constante. Autrement dit, plus l'entraînement Q-learning se prolonge, plus le temps de réponse de l'environnement devient pénalisant et on observe le contraire dans l'adaptation de l'agent q-maml au même scénario, car elle nécessite moins d'épisodes et également moins d'exploration.

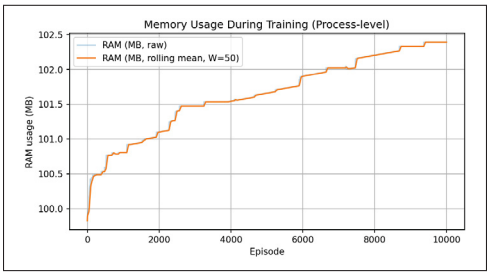


Figure 4.30 Consommation mémoire du processus pendant l'entraînement de l'agent Q-learning

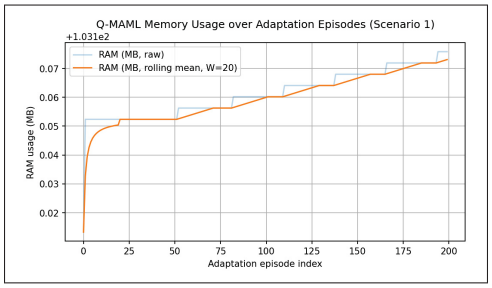


Figure 4.31 Latence moyenne par étape au cours des épisodes d'adaptation de l'agent Q-MAML sur le Scénario 1

Les Figures 4.30 et 4.31 comparent la consommation mémoire des deux agents. On observe pour Q-learning une augmentation lente mais continue de la mémoire utilisée durant l'entraînement, alors que la mémoire consommée par Q-MAML au cours des épisodes d'adaptation reste dans le même ordre de grandeur, avec une dérive très limitée.

En termes de contraintes de calcul, les deux approches restent peu coûteuses par interaction : l'utilisation CPU moyenne est de quelques pourcents, la consommation mémoire se stabilise autour de 100–103 Mo et ne montre pas de dérive explosive. La différence majeure apparaît lorsque l'on considère le temps de convergence. L'agent Q-learning met plusieurs milliers d'épisodes pour stabiliser sa performance, alors que l'agent Q-MAML atteint un comportement comparable en moins de la moitié de ce nombre d'épisodes. Or, les mesures montrent que le coût par épisode est du même ordre de grandeur pour les deux méthodes. Par conséquent, le coût global en ressources est nettement plus faible pour Q-MAML : à performance similaire, l'agent méta-apprenant consomme beaucoup moins de ressources simplement parce qu'il converge beaucoup plus rapidement.

4.6 Discussion et limites

Les résultats montrent que le modèle Q-MAML permet une adaptation rapide et stable à de nouveaux scénarios d'attaque-défense, réduisant le temps d'apprentissage et augmentant la robustesse. Toutefois, plusieurs limites persistent. Le modèle ne généralise pas efficacement à des environnements totalement nouveaux ou à des configurations de réseau qu'il n'a pas vues durant l'entraînement. Dans un tel cas, les politiques apprises deviennent obsolètes et nécessitent un réentraînement complet.

- **Dimension réduite de l'espace d'état :** l'observation binaire de 25 bits simplifie la représentation du réseau. Cette abstraction limite la capacité de généralisation à des environnements plus riches comportant des dépendances continues (ex. charge CPU, latence, nombre d'hôtes).
- **Environnement partiellement déterministe :** l'utilisation de sorties Caldera comme unique source d'observation rend les transitions d'état dépendantes de la disponibilité des rapports API ; toute latence ou erreur HTTP peut introduire du bruit non contrôlé.
- **Mémoire tabulaire :** le stockage de `Q[s,a]` dans une table discrète devient inadapté à grande échelle. Une extension vers des réseaux de neurones (Deep-Q ou Meta-DQN) serait nécessaire pour des environnements plus complexes.

De plus, le temps d'exploration reste un facteur critique : il est nécessaire de déterminer un seuil optimal entre exploration et exploitation afin d'éviter de gaspiller des épisodes lorsque la politique est déjà stable. Un excès d'exploration conduit à une stagnation des performances, tandis qu'un arrêt prématuré peut empêcher l'agent de découvrir des actions plus efficaces. Enfin, le modèle échoue principalement dans deux situations : lorsque les récompenses sont extrêmement rares et lorsque l'environnement introduit des actions ou états non présents dans la distribution d'entraînement initiale.

Il est important de noter que toutes les expériences présentées dans ce mémoire ont été réalisées sur une topologie fixe à trois hôtes ; seuls les scénarios varient. Nous n'avons pas évalué

systématiquement la robustesse du méta-apprentissage à de légères modifications de la topologie. La capacité du modèle à transférer ses connaissances vers des graphes de réseau modifiés demeure donc une limite de ce travail et constitue une piste directe pour des travaux futurs.

CONCLUSION ET RECOMMANDATIONS

Ce mémoire a présenté un cadre méthodologique destiné à étudier la résilience cybernétique à travers l'automatisation d'opérations offensives menées par un agent d'apprentissage par renforcement. L'objectif général était double : d'une part, concevoir un agent capable de mener des scénarios d'attaque réalistes afin d'évaluer la posture de sécurité d'un réseau ; d'autre part, développer un modèle d'apprentissage suffisamment flexible pour reproduire efficacement son comportement dans différents environnements ou scénarios.

La première étape a consisté en une analyse approfondie des travaux existants, notamment la plateforme CyGIL, souvent citée comme référence académique pour l'entraînement d'agents RL dans des environnements cybernétiques simulés. La reproduction de l'architecture conceptuelle de CyGIL, couplée à notre propre implémentation en Q-learning tabulaire, a permis d'identifier plusieurs limites importantes : difficulté de généralisation, instabilité de l'apprentissage et réentraînement nécessaire à chaque nouveau scénario. Cette étude comparative a servi de base solide pour orienter la conception d'une approche plus flexible.

Pour dépasser ces contraintes, nous avons adopté une méthode inspirée du méta-apprentissage, en particulier le modèle MAML appliqué au Q-learning. Cette stratégie vise à doter l'agent de capacités d'adaptation rapide à des situations nouvelles, tout en tirant parti des savoirs acquis dans des scénarios antérieurs. Son intégration a nécessité une refonte complète de notre environnement expérimental ainsi que l'ajout d'une logique d'entraînement bi-niveau permettant la mise en œuvre du Q-MAML.

Les résultats obtenus montrent des améliorations significatives en termes de stabilité, de performance et de rapidité d'adaptation. Les courbes de retour moyen révèlent une croissance constante après chaque cycle de méta-entraînement, tandis que des indicateurs tels que l'écart moyen des valeurs Q ou l'atteinte des objectifs démontrent une convergence plus régulière et une politique plus cohérente. Par ailleurs, l'agent Q-MAML requiert moins d'étapes pour atteindre

l'état terminal, témoignant d'une réduction notable de la phase d'exploration aléatoire. Ces observations confirment que le méta-apprentissage renforce la résilience comportementale de l'agent en contexte multi-scénarios.

La comparaison avec CyGIL met en évidence plusieurs avancées importantes. Alors que CyGIL repose sur un apprentissage par renforcement classique nécessitant un réentraînement complet pour chaque scénario, notre approche introduit un mécanisme de méta-apprentissage permettant une adaptation rapide sans repartir de zéro. Cette amélioration réduit le coût d'apprentissage tout en augmentant la capacité de généralisation. De plus, alors que CyGIL s'appuie sur Mininet pour simuler le réseau, notre plateforme utilise des machines virtuelles VMware, offrant une émulation nettement plus réaliste des interactions réseau, des charges systèmes et des contraintes opérationnelles. Ces choix méthodologiques renforcent la fidélité de l'environnement et élargissent la portée expérimentale du modèle proposé.

Comme toute démarche scientifique, celle-ci comporte néanmoins des limites. L'environnement étudié reste volontairement restreint en topologie et en diversité de scénarios offensifs. L'espace d'actions dépend fortement des capacités de CALDERA, et les algorithmes explorés se limitent au Q-learning et au Q-MAML. L'inclusion de techniques d'apprentissage profond ou de modèles hybrides pourrait enrichir l'analyse et permettre une observation plus fine de l'adaptabilité en environnement complexe.

Le code complet développé dans le cadre de ce mémoire sera mis à la disposition des futurs étudiants afin de favoriser la reproductibilité des résultats et de soutenir les travaux futurs en cybersécurité et en apprentissage par renforcement.

Ces limites ouvrent des perspectives de travaux futurs, parmi lesquelles :

- l'extension de la topologie réseau et la création de scénarios offensifs plus variés ;
- l'enrichissement de l'espace d'observations à l'aide de données système ou comportementales ;

- l'intégration d'algorithmes d'apprentissage profond ou d'approches hors-modèle ;
- l'étude des interactions entre agents offensifs et défensifs autonomes dans un contexte multi-agents.

En somme, ce mémoire apporte une contribution méthodologique et expérimentale significative à l'étude du RL et du méta-apprentissage dans un contexte cyber offensif. Il propose une plateforme d'émulation réaliste, un cadre RL reproductible et une démonstration claire des bénéfices du méta-apprentissage pour l'adaptation rapide en contexte multi-scénarios. Ces fondations ouvrent la voie à la conception de systèmes plus robustes, flexibles et véritablement résilients face aux cybermenaces contemporaines.

ANNEXE I

LES FONCTIONNALITÉS DE CALDERA

Dans cette annexe, nous illustrons plusieurs écrans de MITRE Caldera afin de compléter la description de l'outil utilisée dans ce mémoire. Caldera est une plateforme d'émulation d'adversaires construite autour d'un serveur de commande et contrôle (C2), d'une interface web et d'une collection de plugins. Elle permet de déployer des agents sur des hôtes, d'exécuter des capacités alignées sur MITRE ATT&CK, de suivre les résultats d'attaque et d'automatiser des opérations complètes. Les sections suivantes décrivent les principales fonctionnalités exploitées dans ce travail.

1. Gestion des agents

La Figure I-1 présente l'onglet *Agents*. Les agents sont des programmes qui s'exécutent sur les hôtes cibles et se connectent périodiquement au serveur Caldera pour recevoir des instructions et renvoyer les résultats. Chaque agent est identifié par une empreinte unique et possède un état.

Cette vue permet de surveiller l'ensemble des agents déployés, d'obtenir un résumé global et de lancer le déploiement de nouveaux agents. Tant qu'aucun agent n'est en place, aucune opération ne peut être exécutée, ce qui fait de cette étape un prérequis essentiel à toute campagne de test d'intrusion.

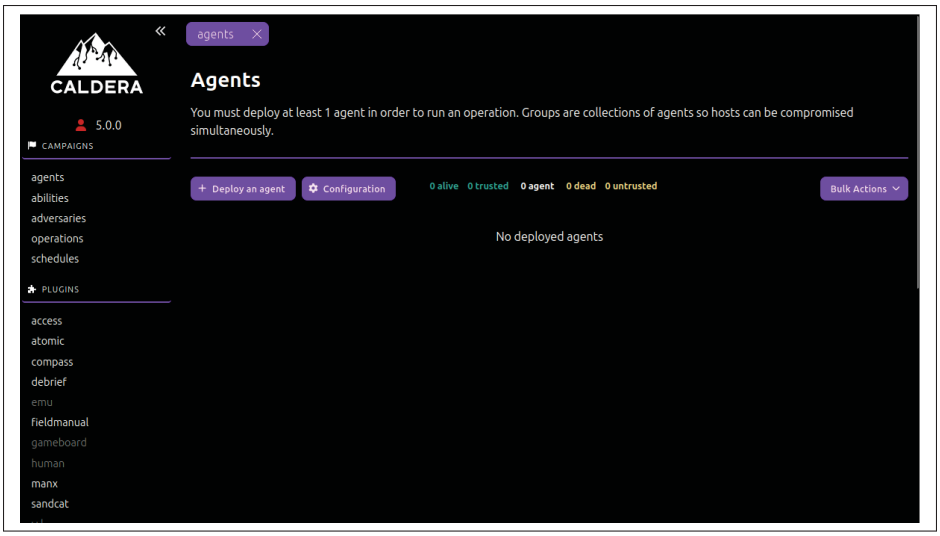


Figure-A I-1 Onglet *Agents* de MITRE Caldera, utilisé pour déployer et superviser les agents sur les hôtes compromis

2. Bibliothèque d'abilities

La Figure I-2 montre la bibliothèque d'abilities. Une *ability* correspond à l'implémentation d'une tactique ou d'une technique ATT&CK : elle décrit une commande ou un script à exécuter, les plateformes supportées, l'exécuteur, les payloads nécessaires et éventuellement un module de parsing pour analyser la sortie.

Chaque carte est étiquetée par une tactique et une technique MITRE ATT&CK. L'utilisateur peut rechercher, filtrer et examiner les actions disponibles, ainsi que leurs prérequis, leurs exécutants et les charges utiles associées. C'est à partir de cet inventaire structuré que l'on construit les scénarios d'attaque utilisés dans l'environnement CyGILGymEnv.

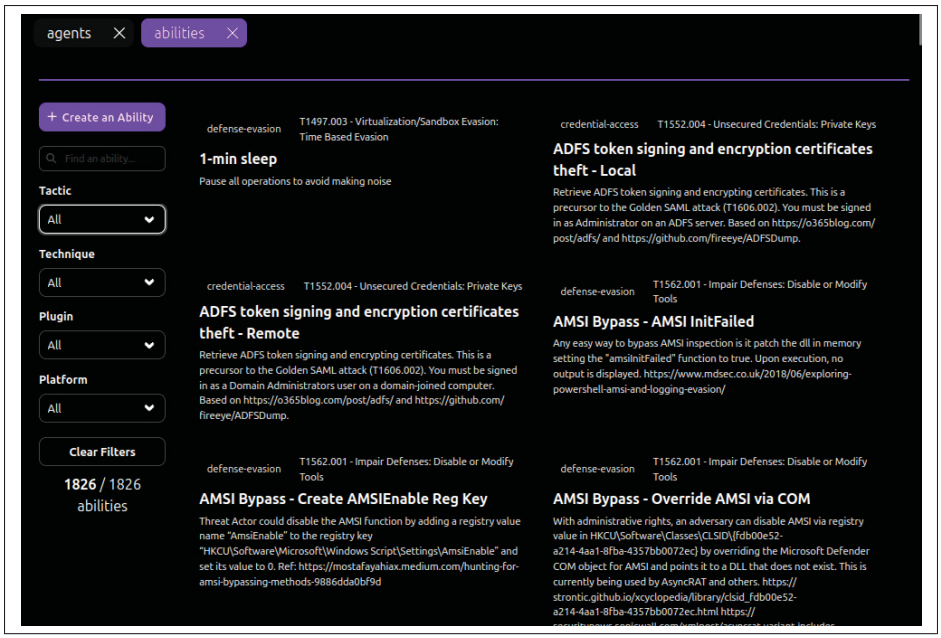


Figure-A I-2 Bibliothèque d'abilities de Caldera : chaque capacité est annotée avec une tactique et une technique ATT&CK

3. Filtrage par tactique ATT&CK

La Figure I-3 illustre le filtrage des abilities par tactique ATT&CK. Le menu *Tactic* permet de restreindre l'affichage, par exemple aux abilities de découverte, d'exfiltration ou de déplacement latéral.

Ce filtrage facilite la sélection d'actions cohérentes avec un objectif particulier et reflète le lien étroit entre Caldera et la taxonomie ATT&CK. Il permet également de composer rapidement des profils d'adversaires alignés sur une ou plusieurs étapes du cycle d'attaque.

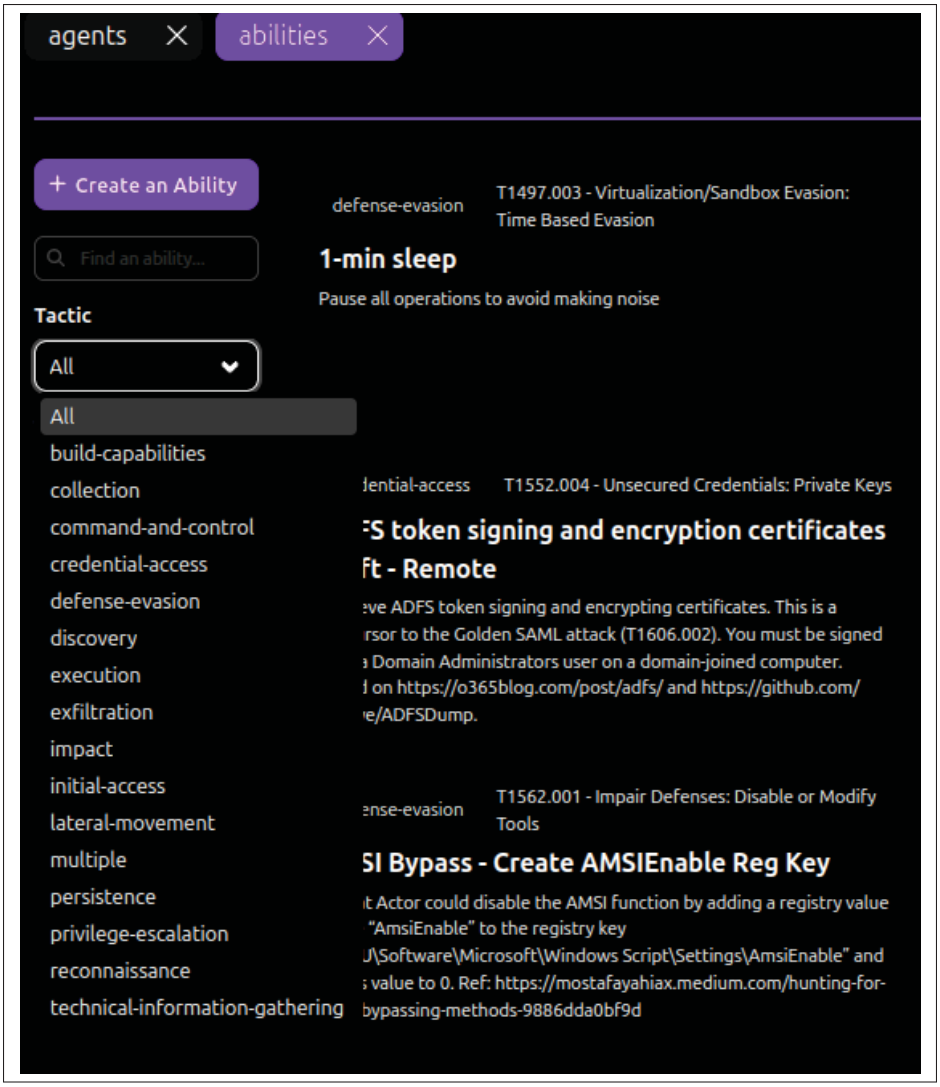


Figure-A I-3 Filtrage des abilities par tactique ATT&CK, permettant de cibler un sous-ensemble d'actions cohérent avec un objectif donné

4. Création d'une nouvelle opération

La Figure I-4 affiche la fenêtre de création d'une nouvelle opération. Une opération est l'unité principale d'exécution dans Caldera : elle associe un nom, un profil d'adversaire, une source de faits, un groupe d'agents, un planificateur et plusieurs paramètres supplémentaires.

Cette interface synthétise l'ensemble des éléments précédents et permet de lancer des scénarios d'attaque de manière reproductible et contrôlée. Une fois l'opération démarrée, Caldera génère et suit les *links* correspondant à chaque ability exécutée, enregistre les résultats et alimente les rapports utilisés dans ce mémoire.

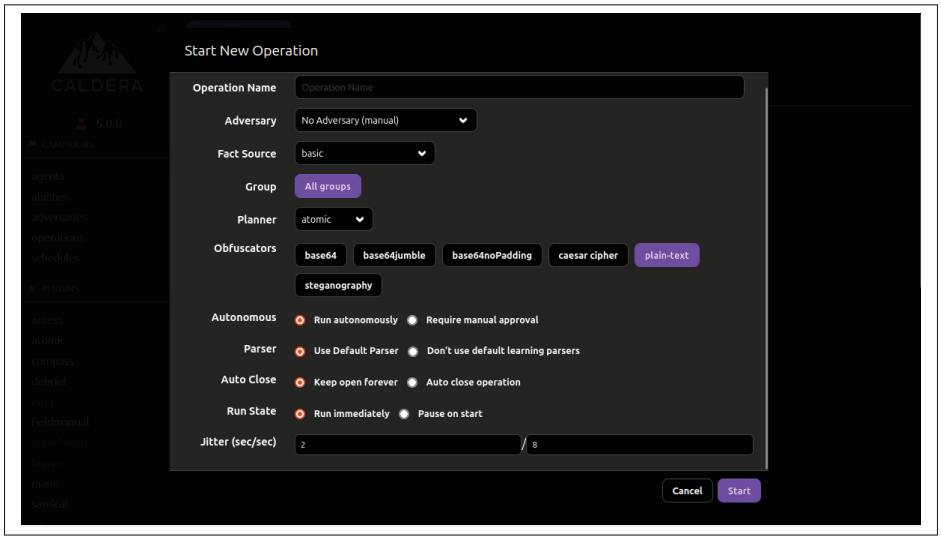


Figure-A I-4 Fenêtre de création d'une opération : configuration du profil d'adversaire, des agents, du planner et des paramètres d'exécution

5. Plugins et extensibilité de la plateforme

La Figure I-5 présente la liste des plugins installés dans Caldera. L'architecture de la plateforme est fortement modulaire : le cœur fournit le serveur C2, l'API et les types d'objets de base, tandis que les plugins ajoutent des fonctionnalités spécialisées. Parmi eux, on trouve notamment des plugins fournissant des agents, des bibliothèques d'abilities et d'adversaires, des contenus de formation ou encore des vues d'analyse complémentaires.

Cette structure permet d'activer ou de désactiver des ensembles de fonctionnalités en fonction du cas d'usage et d'intégrer facilement de nouveaux contenus sans modifier le cœur de l'outil.

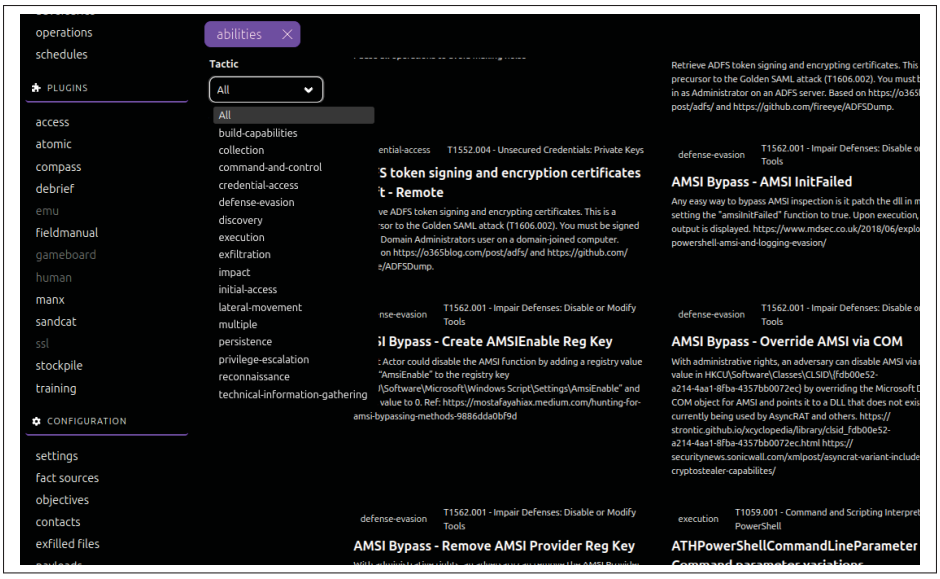


Figure-A I-5 Liste des plugins Caldera. Chaque plugin étend la plateforme en ajoutant des abilities, des contenus et des vues spécialisées

6. Gestion des payloads

La Figure I-6 montre l'interface de gestion des *payloads*. Les charges utiles sont des fichiers transférés vers les agents lors de l'exécution de certaines abilities. Caldera distingue les payloads locaux, ajoutés par l'utilisateur, des payloads fournis par les plugins.

Cette gestion centralisée permet de réutiliser les mêmes fichiers dans plusieurs scénarios, tout en gardant une trace de leur origine et de leur contexte d'utilisation. Il est également possible de configurer le mode de transfert et d'appliquer un nettoyage après exécution pour limiter les traces sur les hôtes cibles.

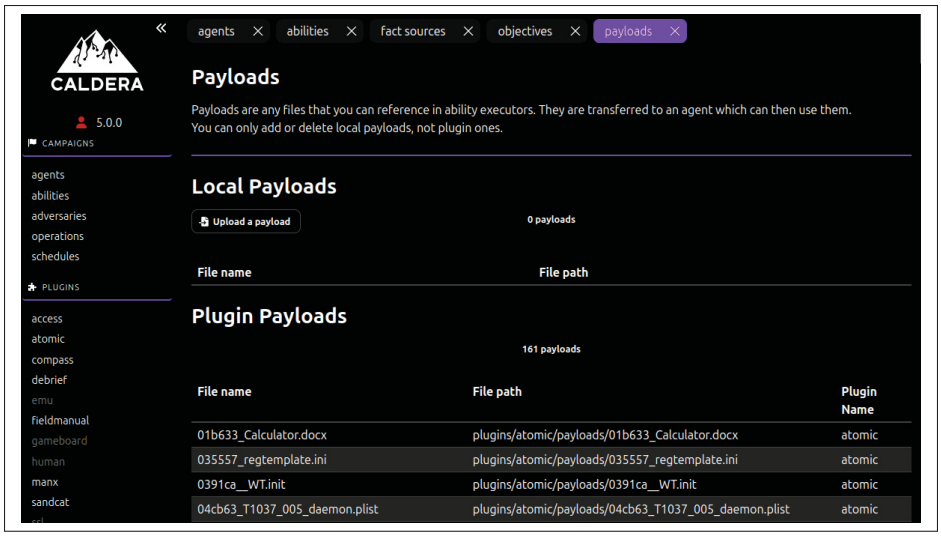


Figure-A I-6 Gestionnaire de *payloads* : interface listant les charges utiles locales et celles fournies par les plugins

7. Suivi des fichiers exfiltrés

La Figure I-7 illustre la vue *Exfilled Files*. Lorsque des abilities d'exfiltration sont exécutées avec succès, les fichiers transférés depuis les agents vers le serveur sont stockés dans un répertoire dédié (par défaut `/tmp/caldera`). Cette vue permet de vérifier quelles données ont été effectivement extraites lors d'une opération, de filtrer par campagne et de télécharger les fichiers pour analyse ultérieure.

Dans le contexte de l'étude de la résilience, cette fonctionnalité joue un rôle central pour mesurer l'impact réel d'un scénario d'attaque et pour relier les capacités d'exfiltration aux métriques de succès utilisées dans le mémoire.

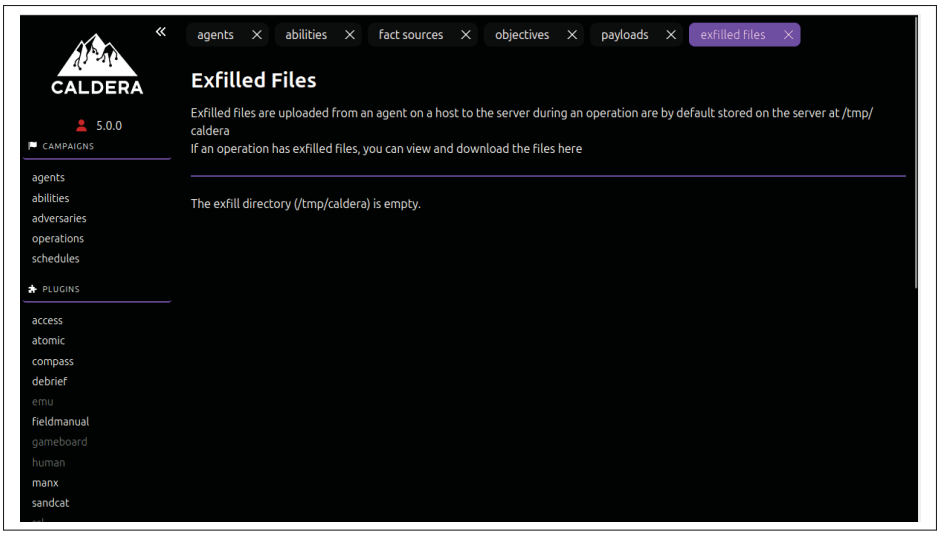


Figure-A I-7 Vue des fichiers exfiltrés : les données extraites par les agents sont stockées et consultables sur le serveur Caldera

8. Profils d'adversaires

La Figure I-8 présente l'onglet *Adversaries*. Un profil d'adversaire est une collection structurée d'abilities reliées à des tactiques et techniques ATT&CK. Chaque profil représente un comportement offensif ou défensif particulier : découverte du système, mouvement latéral, exfiltration de fichiers, sabotage, etc.

L'utilisateur peut réordonner les abilities, en ajouter ou en supprimer, ajuster l'objectif associé et sauvegarder le profil. Ces profils sont ensuite réutilisés lors de la création d'opérations pour exécuter des scénarios complets en un seul clic, ce qui facilite fortement la reproductibilité des essais.

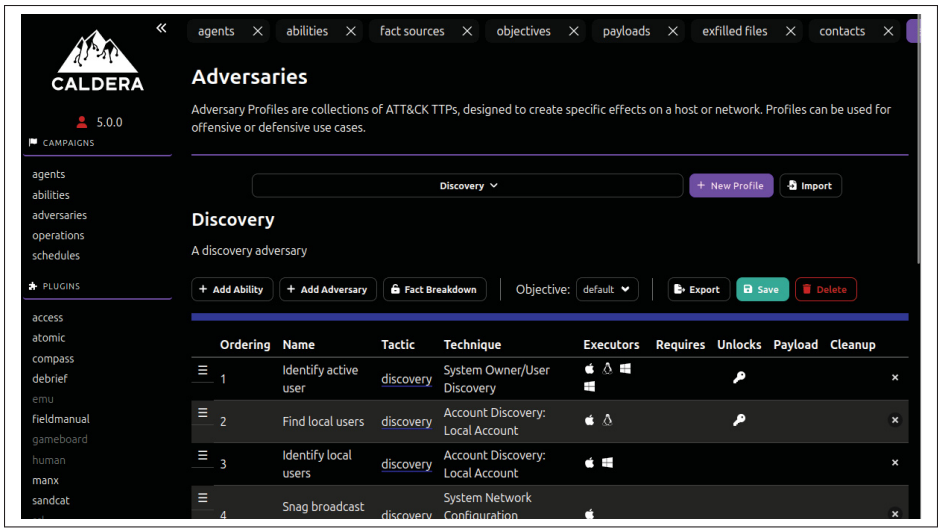


Figure-A I-8 Profils d'adversaires : collections d'abilities organisées par tactiques et techniques, utilisables pour des campagnes offensives ou défensives

9. Synthèse des fonctionnalités de Caldera

Dans l'ensemble, MITRE Caldera fournit une plateforme unifiée pour orchestrer des campagnes de test d'intrusion réalistes : déploiement d'agents, définition de profils d'adversaires, sélection d'abilities alignées sur MITRE ATT&CK, gestion de payloads, suivi des fichiers exfiltrés et configuration détaillée des opérations. Cette intégration facilite la construction de scénarios complets couvrant plusieurs tactiques d'attaque, tout en offrant une API exploitable pour l'automatisation et l'apprentissage par renforcement. Dans le cadre de ce mémoire, ces fonctionnalités ont permis de créer un environnement expérimental riche, où les agents RL peuvent interagir avec des séquences d'attaque structurées et observables, et ainsi contribuer à l'étude de la résilience des systèmes.

BIBLIOGRAPHIE

Admass, W. S., Munaye, Y. Y. & Diro, A. A. (2024). Cyber security : State of the art, challenges and future directions. *Cyber Security and Applications*, 2, 100031. doi : https://doi.org/10.1016/j.csa.2023.100031.

Akbari, I., Tahoun, E., Salahuddin, M. A., Limam, N. & Boutaba, R. (2020). ATMoS : Autonomous Threat Mitigation in SDN using Reinforcement Learning. *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9.

Botvinick, M., Ritter, S., Wang, J., Kurth-Nelson, Z., Blundell, C. & Hassabis, D. (2019). Reinforcement Learning, Fast and Slow. *Trends in Cognitive Sciences*, 23. doi : 10.1016/j.tics.2019.02.006.

Cai, W., Chen, H., Miao, H., Liu, F., Zhang, Y. & Yang, X. (2024). A Vulnerability-Driven Gym for Training Autonomous Cyber Agents with Reinforcement Learning. *2024 IEEE Smart World Congress (SWC)*, 1477-1484. Repéré à https://api.semanticscholar.org/CorpusID:277285516.

Cao, Y. (2025). *Adaptive Defense Strategies Using PPPO in Deep Reinforcement Learning for CyberBattleSim*. (Master of Science Thesis, University of Auckland).

Farama Foundation. (2025). Basic Usage - Gymnasium Documentation. Repéré à https://gymnasium.farama.org/introduction/basic_usage/.

Fernández Carrasco, J. , Amonarriz Pagola, I., Orduna Urrutia, R. & Román, R. (2024). CYBERSHIELD : A Competitive Simulation Environment for Training AI in Cybersecurity. *2024 11th International Conference on Internet of Things : Systems, Management and Security (IOTSMS)*. doi : 10.1109/IOTSMS62296.2024.101710208.

Finn, C., Abbeel, P. & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *CoRR*, abs/1703.03400. Repéré à http://arxiv.org/abs/1703.03400.

Giannaros, A., Karras, A., Theodorakopoulos, L., Karras, C., Kranias, P., Schizas, N., Kalogeratos, G. & Tsolis, D. (2023). Autonomous Vehicles : Sophisticated Attacks, Safety Issues, Challenges, Open Topics, Blockchain, and Future Directions. *Journal of Cybersecurity and Privacy*, 3(3), 493–543. doi : 10.3390/jcp3030025.

Huang, Y., Huang, L. & Zhu, Q. (2022). Reinforcement Learning for Feedback-Enabled Cyber Resilience. *Annual Reviews in Control*, 53, 273–295. doi : 10.1016/j.arcontrol.2021.11.006.

Hwang, I., Seo, R. & Hu, S. (2025). Boosting employee information security compliance : the contingent roles of task–technology and person–organization fits. *Humanities and Social Sciences Communications*, 12. doi : 10.1057/s41599-025-04718-x.

Kott, A., Weisman, M. J. & Vandekerckhove, J. (2022). Mathematical Modeling of Cyber Resilience. *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*. doi : 10.1109/milcom55135.2022.10017731.

Kunz, T., Fisher, C., La Novara-Gsell, J., Nguyen, C. & Li, L. (2023). A Multiagent CyberBattleSim for RL Cyber Operation Agents. *IEEE*.

Li, L., Fayad, R. & Taylor, A. (2021). CyGIL : A Cyber Gym for Training Autonomous Agents over Emulated Network Systems.

Ligo, A. K., Kott, A. & Linkov, I. (2021). How to Measure Cyber Resilience of an Autonomous Agent : Approaches and Challenges. *AICA 2021, 1st International Conference on Autonomous Intelligent Cyber-defence Agents*.

Madsen, H., Grov, G., Mancini, F., Baksaas, M. & vald Å slaugson Sommervoll, A. [arXiv :2410.21407 [cs.CR]]. (2024). Exploring reinforcement learning for incident response in autonomous military vehicles. Repéré à https://arxiv.org/abs/2410.21407.

Magnusson, L., Iqbal, S., Elm, P. & Dalipi, F. (2025). Information security governance in the public sector : investigations, approaches, measures, and trends. *International Journal of Information Security*, 24(4), 177–197. doi : 10.1007/s10207-025-01097-x.

Microsoft. (2021). CyberBattleSim. Repéré à https://github.com/microsoft/CyberBattleSim.

MITRE Corporation. (2025). MITRE Caldera Documentation. Repéré à https://caldera.readthedocs.io/en/latest/.

Nguyen, H. P. T., Hasegawa, K., Fukushima, K. & Beuran, R. (2025). PenGym : Realistic training environment for reinforcement learning pentesting agents. *Computers & Security*, 148, 104140. doi : 10.1016/j.cose.2024.104140.

Nguyen, T. T. & Reddi, V. J. (2021). Deep reinforcement learning for cyber security. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8), 3779–3795. doi : 10.1109/TNNLS.2021.3092799.

Nichol, A., Achiam, J. & Schulman, J. (2018). On First-Order Meta-Learning Algorithms. *arXiv preprint arXiv :1803.02999*.

Pashamokhtari, A., Batista, G. & Habibi Gharakheili, H. (2023). AdIoTack : Quantifying and Refining Resilience of Decision Tree Ensemble Inference Models against Adversarial Volumetric Attacks on IoT Networks. *Proceedings of the 2023 IEEE Conference on Communications and Network Security (CNS)*.

Rasheed, I., Hu, F. & Zhang, L. (2020). Deep reinforcement learning approach for autonomous vehicle systems for maintaining security and safety using LSTM-GAN. *Vehicular Communications*, 26, 100266. doi : 10.1016/j.vehcom.2020.100266.

Rostami, E., Karlsson, F., Kolkowska, E. & Gao, S. (2025). Towards software for tailoring information security policies to organisations' different target groups. *Computers Security*, 159, 104687. doi : https://doi.org/10.1016/j.cose.2025.104687.

Standards, N. I. O. & Technology. *(2024)*. Framework for Improving Critical Infrastructure Cybersecurity. U.S. Department of Commerce. Repéré à https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.29.pdf.

Tang, L. & Mahmoud, Q. H. (2022). A Deep Learning-Based Framework for Phishing Website Detection. *IEEE Access*, 10, 1509-1521. Repéré à https://api.semanticscholar.org/CorpusID:245449748.

Wang, M. & Dechene, R. (2024, 10). Multi-Agent Actor-Critics in Autonomous Cyber Defense.

Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., Dai, B. & Miao, Q. (2024). Deep Reinforcement Learning : A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4), 5064-5078. doi : 10.1109/TNNLS.2022.3207346.

Weisman, M. J., Kott, A., Ellis, J. E., Murphy, B. J., Parker, T. W., Smith, S. & Vandekerckhove, J. (2023). Quantitative Measurement of Cyber Resilience : Modeling and Experimentation.

Xiao, L., Wan, X., Dai, C., Du, X., Chen, X. & Guizani, M. (2018). Security in Mobile Edge Caching with Reinforcement Learning. *IEEE Wireless Communications*, 25(3), 116–122. doi : 10.1109/MWC.2018.1700291. Publisher Copyright : © 2002-2012 IEEE.

Yang, W., Acuto, A., Zhou, Y. & Wojtczak, D. (2024). A Survey for Deep Reinforcement Learning Based Network Intrusion Detection. Repéré à https://arxiv.org/abs/2410.07612.

Yang, Y. & Liu, X. (2022, 02). Behaviour-Diverse Automatic Penetration Testing : A Curiosity-Driven Multi-Objective Deep Reinforcement Learning Approach.