Reinforced Autonomous Agents with Attack-Defense Exercises in Realistic Environments

Frédéric Guihery[1], Georges Bossert[2], Damien Crémilleux[1], Olivier Tétard[2], Baptiste Gigodeaux[2] and Édouard Klein[2]

[1]AMOSSYS, Rennes, France
[3]SEKOIA, Rennes, France

Abstract

The current trend is towards automation inside a security operations center (SOC), in particular on the remediation side. However, the implementation of remediation playbooks must be qualified in terms of impact on the protected service, in order to avoid loss of availability. Here we propose an approach aimed at automating the execution of attacker intrusion sets against an IT network, to learn the best countermeasures to apply. This approach is based on an environment that automates attack and defense, with a learning capability. Attack automation relies on simulating the modus operandi of threat actors as well as their attack infrastructures. In defense, automated learning of the most suitable remediation action sequences for the protection of an information system (IS) is carried out. This article details the results obtained by the DALID platform, which allows the behavior of autonomous attackers and defenders to be observed during repeated exercises to automatically create the best remediation strategies for SOC teams. In the future, this platform aims to become an attack-defense gamification framework for evaluating the effectiveness of defensive computer warfare architectures.

Résumé

La tendance actuelle est à l'automatisation des opérations d'un centre opérationnel de la sécurité (SOC, *Security Operations Center*), en particulier sur le volet remédiation. Cependant, la mise en œuvre de playbooks de remédiation doit être qualifiée en termes d'impact sur le service protégé, afin d'éviter des pertes de disponibilité. Nous proposons ici une démarche visant à automatiser l'exécution de modes opératoires d'attaquants vis-à-vis d'un système d'information, afin d'apprendre les meilleures contre-mesures à appliquer. Cette démarche s'appuie sur un environnement d'exercice automatisant l'attaque et la défense, dans une optique d'apprentissage. L'automatisation de l'attaque repose ici sur la simulation de modes opératoires de groupes d'attaquants et d'infrastructures d'attaque. En défense, il est effectué un apprentissage automatisé des séquences d'actions de remédiations les plus adaptées à la protection d'un système d'information (SI). Cet article détaille les résultats obtenus par la plateforme DALID qui permet d'observer le comportement d'agents autonomes d'attaques et de défenses au cours d'exercices répétés afin de constituer automatiquement les meilleures stratégies de remédiations pour des équipes SOC. Cette plateforme a pour ambition de devenir un cadre de gamification de l'attaque-défense permettant d'évaluer l'efficacité d'architectures de lutte informatique défensive.

Keywords

Reinforcement learning, APT simulation, Attack infrastructure, TTP, XDR, Remediation, Course of action

C&ESAR'21: Computer Electronics Security Application Rendezvous, November 16-17, 2021, Rennes, France
frederic.guihery@amossys.fr (F. Guihery); georges.bossert@sekoia.fr (G. Bossert); damien.cremilleux@amossys.fr (D. Crémilleux); olivier.tetard@sekoia.fr (O. Tétard); baptiste.gigodeaux@sekoia.fr (B. Gigodeaux); edouard.klein@sekoia.fr (É. Klein)

1. Introduction

1.1. Contexte

La bonne santé d'une entreprise ou d'une organisation dépend fortement de l'intégrité de son système d'information. Une indisponibilité ou un vol de données peuvent avoir un impact significatif pouvant aller jusqu'à la faillite [1]. Une configuration statique du SI et des équipements de sécurité est insuffisante et rarement possible, le SI devant s'adapter en particulier à des évolutions techniques, organisationnelles, ou encore réglementaires.

Il est donc nécessaire de surveiller le SI et d'adapter sa configuration et celle des équipements de sécurité en cas d'intrusion ou tentative d'intrusion mais également en fonction d'évènements liés à l'entreprise (e.g. lancement d'un produit) ou ses outils de travail (e.g. publication de vulnérabilité). C'est l'une des missions d'un centre opérationnel de la sécurité (SOC, *Security Operations Center*) ou sa version externalisée (MSSP, *Managed Security Service Provider*).

La mise en place et l'exploitation d'un SOC sont complexes et coûteuses. Tout d'abord les compétences sont rares et les attaques évoluent rapidement. Par ailleurs, il y a une asymétrie entre attaque et défense : l'attaquant a en général un objectif restreint, dispose d'outils automatisant une partie ou la totalité de l'attaque, et peut choisir quand déclencher son attaque ; le défenseur doit défendre le SI dans son intégralité, assurer sa disponibilité et réagir à tout moment, même la nuit.

Afin de tenir compte de ces contraintes, nous présentons les contributions suivantes :

— une `plateforme de simulation` afin de tester et entraîner un agent autonome, dans un environnement maîtrisé, face à des scénarios d'attaque représentatifs de modes opératoires de groupes d'attaquants ;

— un `agent autonome` capable de réagir rapidement et compléter voire suppléer les opérateurs humains via une réponse adaptée à une posture de sécurité définie tout en permettant d'interagir avec eux pour une remédiation complète.

Ces travaux sont réalisés au sein du projet DALID, dans le cadre du dispositif RAPID de DGA-MI et de l'Agence de l'Innovation de Défense.

1.2. Problématiques et verrous à lever

Un environnement de simulation d'attaque et d'apprentissage en défense soulève différentes problématiques. Nous distinguons d'une part les verrous liés à la simulation de modes opératoires d'attaquants :

— Afin d'obtenir une plateforme de simulation visant à tester et entraîner une architecture de défense opérationnelle, un premier objectif est d'arriver à produire un haut niveau de réalisme des modes opératoires d'attaquants, c'est à dire qu'un scénario d'attaque doit être en mesure de dérouler les TTP (Tactiques, Techniques, et Procédures) [2] caractéristiques de modes opératoires connus, tout en exposant des IoC (Indicator of Compromise) [3] crédibles.

— Par ailleurs, la plateforme de simulation d'attaque doit être en mesure de jouer automatiquement des scénarios d'attaques complets, permettant de représenter l'ensemble de la kill chain.

— Enfin, cette plateforme d'attaque doit permettre de simuler un profil attaquant externe, c'est-à-dire qu'elle ne doit pas laisser de traces liées à l'exécution « interne » de l'agent sur l'environnement attaqué.

Sur le volet défensif, nous identifions les problématiques suivantes :

— Pour faire face à une attaque, les équipes SOC appliquent des contre-mesures qui, in fine, vont leur permettre de maîtriser les conséquences de l'intrusion sur le système à défendre. Ces contre-mesures sont de natures différentes et doivent être paramétrées en fonction du contexte, par exemple avec l'IP de l'attaquant. Un premier verrou porte sur l'instanciation automatique d'une contre-mesure étant donné un contexte de détection de l'attaque.
— Par nature une contre-mesure peut avoir des impacts positifs et/ou négatifs sur la qualité de service du système à protéger. Ainsi, étant donné une attaque, un environnement à défendre et le contexte courant, un autre verrou adressé par ces travaux réside dans l'évaluation et la qualification des impacts d'une contre-mesure.
— Pour finir sur le volet défensif, il existe de nombreuses contre mesures unitaires et donc de playbooks possibles qui les combinent. Ce volume rend impossible une approche d'évaluation systématique étant donné les fortes contraintes de réalisme que nous imposons à notre environnement d'évaluation.

Enfin, plusieurs objectifs sont directement liés à l'environnement d'apprentissage :

— L'environnement peut apporter un biais d'apprentissage si les systèmes d'informations sont trop statiques. Ceci implique un objectif de génération d'environnements d'expérimentation suffisamment représentatifs et variés pour qu'ils soient intéressants en termes d'apprentissage en défense.
— Enfin, comme dans tout environnement de jeu, il faut pouvoir identifier un vainqueur. Dans notre contexte, l'objectif est d'identifier quelle stratégie d'attaque ou de défense a été la plus efficace.

Nous présentons dans les chapitres suivants, l'état de l'art du domaine, puis la démarche employée pour permettre l'apprentissage par renforcement d'un agent autonome de remédiation. Le chapitre 4 détaille ensuite l'implémentation de notre solution en présentant de quelle manière nous avons traité les problématiques liées à l'environnement d'apprentissage. Pour finir, nous analysons les résultats obtenus lors de nos expérimentations au cours du chapitre 5 et concluons sur les perspectives de nos travaux.

2. État de l'art

Aujourd'hui il n'existe pas de solution complète permettant d'automatiser des attaques avec des scénarios complets (construction d'une kill chain complète) et réalistes (s'appuyant sur des sources de renseignement sur la menace). Il existe cependant des outils permettant l'exécution d'attaques unitaires ou des scénarios d'attaques unitaires. Citons notamment :

— Tear Security [4] : Outil qui se rapproche le plus de la solution souhaitée. Il identifie les différents logiciels et vulnérabilités présents sur le système (via des outils publics tel que Nessus ou Nexpose). Un chemin d'attaque peut être créé à l'aide d'algorithmes de machine learning. Tear Security est une entreprise anglaise. L'outil semble peu connu et récent, pour information une vidéo de présentation a été réalisée il y a 4 mois, mais celle-ci ne

comptabilise que 25 vues.

- Atomic Red Team[5] : Ensemble de tests d'attaque basés sur le framework ATT&CK du MITRE [6], dont l'objectif est d'exécuter ces tests sur les systèmes testés. Cette solution ne permet pas de réaliser de réelles attaques sur le système d'informations, mais de tester certains éléments de sécurité. Il s'agit d'un projet open source, développé par Red Canary.
- Red Team Automation [7] : Ensemble de scripts implémentant des techniques du MITRE, ces scripts peuvent ensuite être utilisés pour évaluer le système. Il s'agit d'un projet open source, développé par Endgame.
- Metta [8] : Machine virtuelle permettant de simuler des attaques, l'outil se base sur les techniques définies dans le framework MITRE ATT&CK. De nombreux scripts d'attaques sont présents, il est possible de créer des scénarios afin de les réaliser lors des tests. L'outil permet donc d'évaluer un système, mais celui-ci ne met pas en œuvre d'intelligence artificielle permettant de s'adapter à l'environnement et aux réactions potentielles de mécanismes de défense. L'utilisateur doit ainsi définir une liste d'actions à réaliser au sein de chaque scénario. Il s'agit d'un projet développé par Uber Security.
- APT Simulator [9] : Ensemble de scripts Windows utilisés afin de rendre le poste équivalent à un poste compromis suite à une attaque APT (Advanced Persistent Threat). Il s'agit d'un projet open source, développé par Nextron Systems.
- CALDERA [10] : Outil qui agit après une compromission d'un poste, il simule les actions de l'attaquant. L'utilisateur définit un objectif et l'outil essaie d'atteindre celui-ci. Caldera présente des problèmes de passage à l'échelle, ainsi que de stabilité d'après les différents tests réalisés. Il s'agit d'un projet open source, développé par le MITRE.
- Deep Exploit [11] : Outil qui dispose de capacités de collecte de renseignements, de modélisation de la menace, d'analyse de vulnérabilités, d'exploitation, de post-exploitation et de reporting en s'appuyant sur des techniques Reinforcement Learning et Metasploit. Il met à jour une base de connaissance mais ne semble pas planifier de stratégie d'attaque. Il s'agit d'un projet open source, développé par un japonais du nom de Isao Takaesu intervenant régulièrement à plusieurs conférences cyber telles que la Black Hat ou la DEFCON.
- GyoiThon [12] : Outil permettant d'identifier automatiquement les logiciels installés sur un serveur Web et de provoquer les exploits relatifs grâce à l'utilisation de Metasploit. Il s'appuie sur des techniques Machine Learning d'apprentissage et de détection de patterns caractéristiques de logiciels. Il s'agit d'un projet open source, développé par un groupe de hackers Japonais intervenant régulièrement à plusieurs conférences cyber telles que la Black Hat ou la DEFCON.

Il existe par ailleurs la catégorie des produits de *Red Teaming* (assistance au pentest / Red Team), tels que Cobalt Strike et Empire, mais ces outils permettent essentiellement d'assister le pentesteur dans sa démarche de test intrusif, et n'intègrent donc pas entièrement les besoins d'automatisation.

Notre analyse de l'état de l'art montre qu'aucune solution ne permet actuellement l'exécution automatisée d'attaques complexes, réalistes, apprenant de l'environnement et de la défense.

Concernant l'apprentissage des meilleures stratégies de remédiation, l'automatisation repose sur les technologies d'apprentissage par renforcement (ou *Reinforcement Learning*) [13]. Ce type d'apprentissage fait figure d'approche prometteuse en ce qui concerne le Machine Learning,

pour supporter le développement d'agents autonomes apprenant à appliquer une séquence de décision optimale, dans un environnement complexe et incertain [14].

Sur les 5 dernières années, plus de 1 500 articles portant sur l'apprentissage par renforcement ont été publiés dans les top conférences Machine Learning. Toutefois, dans le domaine de la cyber sécurité, la plupart des papiers traitent à cette date de la faisabilité, en se posant la question de savoir dans quelle mesure il est possible d'utiliser l'apprentissage par renforcement pour automatiser des tâches complexes, telles que celles de la cyberdéfense [15].

Récemment, la recherche dans ce domaine a néanmoins été considérablement impulsée par la forte disponibilité d'environnements d'apprentissage tels qu'OpenAI Gym (OpenAI) [16] ou AI Safety Gridworlds (DeepMind) [17]. Toutefois, les progrès faits sont bien plus en retard que dans d'autres domaines, en raison de la forte complexité, le manque de données ou d'environnement réalistes d'apprentissage.

Nombreux sont les papiers de recherche discutant les challenges que représentent l'application d'apprentissage par renforcement sur l'attaque ou la défense d'environnement réseau, tels que des réseaux d'entreprise, réseaux IoT (*Internet of Things*), système cyber-physique ou *software-defined networking* (SDN) [18].

Jusqu'à cette année, il n'existait pas, ou peu, d'environnements d'expérimentation d'apprentissage par renforcement de cybersécurité permettant de relever les défis actuels dans ce domaine et d'améliorer l'état de l'art [19]. À ce jour, plusieurs projets d'environnement d'expérimentation et d'entraînement par renforcement ont été publiés. Une collaboration entre la NSA et MITRE a permis le développement de FARLAND (*Framework for Advanced Reinforcement Learning for Autonomous Network Defense*) [20]. Sa modélisation permet aux chercheurs de concevoir des environnements selon des hypothèses qui correspondent plus étroitement aux menaces réelles. Un apprentissage côté défense est aussi réalisé, toutefois la gamme d'actions possibles est très limitée. En avril 2021, Microsoft publie son projet CyberBattleSim, s'appuyant sur l'interface OpenAI Gym sur Python. Il s'agit là de créer une première plateforme de recherche expérimentale basée sur une abstraction de haut niveau des réseaux informatiques et des concepts de cybersécurité pour y entraîner des agents autonomes utilisant l'apprentissage par renforcement [21]. Cependant, l'approche est davantage basée sur la simulation de cyberattaques dans lesquelles l'agent autonome a pour objectif de prendre le contrôle du réseau. L'agent tente de déjouer les stratégies de cyberdéfense simulées, permettant aux chercheurs de mieux comprendre comment un attaquant se déplace latéralement au sein d'un réseau. D'autre part, CyberBattleSim abstrait l'environnement et ne réalise donc pas les attaques de manière concrète.

Il n'est cependant pas question dans ce projet d'optimiser la défense ou la remédiation. cette raison, en août 2021, une équipe de chercheurs s'est donnée la mission de réutiliser l'environnement CyberBattleSim en y incorporant des éléments de cyber déception en vue d'automatiser la défense et d'observer les effets sur la prise de contrôle d'un réseau de ces différents éléments de cyber déception (decoys, honeypots and honey tokens) [22]. Depuis, de nouveaux environnements d'expérimentation sont en cours, dont CybORG [23], offrant une plus grande variété de scénarios et d'opérations, mais dont le travail actuel n'a été mené que sur la red team. Dans le même esprit, l'approche CyGIL [24] offre une implémentation stateless, plus facilement étendable et scalable. Étant donné un environnement émulé, l'agent s'entraîne sur des configurations de réseau réelles pour construire des modèles de décision pertinents et applicables aux opérations cyber.

Ainsi à ce jour, il n'existe pas d'environnement dans lequel l'agent autonome est complètement tourné vers l'optimisation de la défense et l'application d'une séquence optimale de contre mesures.

3. Notre solution

3.1. Agent autonome d'attaque

Cette phase vise à répondre à deux objectifs : d'une part le réalisme des modes opératoires simulés, et d'autre part la capacité à construire automatiquement un scénario d'attaque complet, de type kill chain. Un scénario d'attaque comporte plusieurs aspects nécessitant d'être automatisés pour un déroulement complet sans intervention humaine. L'automatisation de la partie attaque que nous proposons, afin de répondre aux objectifs ci-dessus, comporte les volets successifs suivants :

— `Choix du groupe d'attaquant`. Les attaquants peuvent être de diverses natures, et agir selon des caractéristiques et des préférences distinctes. Un groupe d'attaquant est ainsi défini selon des aspects tels qu'un ou des secteurs cibles, des tactiques, techniques, et procédures (TTP), ou une zone géographique d'action. La première partie d'un scénario d'attaque consiste donc à choisir l'adversaire. Selon le système d'information simulé (emplacement géographique et secteur d'activité) et les attaques disponibles au sein de la simulation, l'attaquant est choisi depuis une base de connaissances MISP regroupant les groupes d'attaquants connus.

— `Construction d'un graphe d'attaque`. Comme indiqué précédemment, un groupe d'attaquant est caractérisé par des tactiques, techniques et procédures définissant sa kill chain. Dans la plateforme DALID, ces données sont utilisées pour générer un graphe d'attaques correspondant aux actions possibles selon l'avancement du scénario d'attaque. Une base de connaissance, s'appuyant sur la taxonomie ATT&CK du MITRE, contient toutes les attaques publiquement disponibles, enrichie des pré-conditions nécessaires à leur réalisation. Certaines attaques peuvent par exemple ne s'exécuter que sur des machines de type Linux, ou bien nécessiter un reverse shell préalable.

— `Parcours du graphe d'attaque`. Lors de la simulation, le graphe d'attaque est parcouru automatiquement afin de simuler un attaquant. Le résultat des attaques, qu'elles soient un succès ou non, est ajouté dans la base de connaissance pour actualiser le graphe d'attaque et en déduire la suite des actions possibles.

— `Construction dynamique de l'infrastructure d'attaque`. Les infrastructures d'attaquants se modernisent. Le but des ces infrastructures est de fournir un canal de communication fiable entre l'attaquant et la victime, de ne pas se faire détecter, et d'exécuter des actions malveillantes (extraction de données, chargement d'une payload, etc). Nous automatisons cette capacité de déploiement d'infrastructures d'attaquant avec différents niveaux de complexité et de furtivité (infrastructure unique d'attaque, infrastructures interne et externe d'attaque, canaux de C&C, anonymisation de la source, etc.) En fonction du groupe d'attaquant choisi, cette infrastructure d'attaquant est construite pour permettre de jouer leurs TTP, et en faisant attention à exposer au maximum les mêmes IoC (noms de domaine, adresses IP, URL, etc.). Chaque infrastructure d'attaque s'appuie

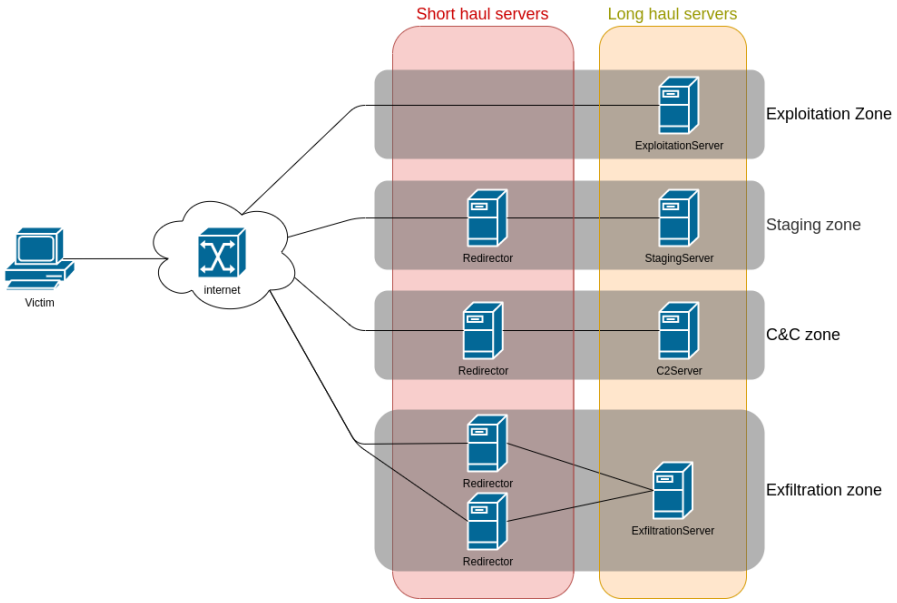

Figure 1 : Infrastructure générique d'attaque, prévoyant des zones dédiées à différentes étapes d'attaque (exploitation, staging, controle et commande, exfiltration).

sur un modèle générique, qui est instancié avec les ressources nécessaires pour dérouler le scénario d'attaque, tel que montré sur la Figure 1.

— `Exécution automatique des techniques d'attaques`. Un moteur d'attaque permet de jouer automatiquement les attaques unitaires définies. Cette base d'attaque se veut modulaire et extensible. Actuellement, une cinquantaine de techniques d'attaques sont supportées, avec notamment la capacité à exploiter plusieurs vecteurs d'attaques (*phishing*, *watering hole*, *man-in-the-middle* et *man-on-the-side* notamment).

Ces différentes étapes sont jouées par un orchestrateur d'attaque, qui s'appuie sur une base de connaissance construite à priori (ensemble des IoC et TTP pour chaque groupe d'attaquant) et en direct lors des séquences d'attaque (découverte d'informations sur l'environnement cible).

3.2. Agent autonome de défense

Une solution de détection et de remédiation est une technologie capable de normaliser les événements produits par les différentes sondes réseaux, systèmes et applicatives déployées sur un système d'information et de rechercher dans ceux-ci des indicateurs d'attaques, d'intrusions et de compromissions en temps réel. En cas de détection, une telle solution propose des contre-mesures aux équipes sécurité et infrastructure en charge des opérations sur le système d'information à protéger.

Ces contre-mesures peuvent être de natures différentes : techniques ou organisationnelles et permettent une réponse différente à la menace identifiée. Parmi les contre-mesures les plus courantes, on retrouve par exemple l'ajout d'une règle de filtrage dans un pare-feu, l'extinction d'une machine virtuelle, le déclenchement d'une analyse antivirale ou encore l'isolation d'un compte utilisateur. Face à une alerte de sécurité, il est de la responsabilité de l'opérateur de

sélectionner puis d'exécuter la ou les combinaisons de contre mesures les plus adaptées à son contexte et sa posture de sécurité. Pour être pleinement efficace, ce mode de fonctionnement fait l'hypothèse que :

— les équipes sécurité peuvent agir aussi rapidement que les attaquants ;
— les équipes sécurité sont sur le pont 24h / 24 7j / 7 ;
— les équipes de sécurité ne sont pas surchargées par d'autres alertes/attaques en cours.

Sur le terrain, ces conditions sont rarement réunies. Il est donc important d'aider les équipes de sécurité en proposant des combinaisons de contre-mesures les plus pertinentes possibles tout en prenant en compte les effets de bords qu'elles peuvent engendrer sur la disponibilité du SI.

Ces travaux proposent une solution pour identifier automatiquement les contre-mesures les plus adaptées à la protection d'un système d'information tout en maîtrisant l'impact sur les usages légitimes du SI. La découverte automatique des séquences de contre-mesures les plus adaptées à une menace est réalisée à l'aide d'un apprentissage actif. Cet apprentissage s'appuie sur :

— un ensemble de règles et d'indicateurs de compromissions contextualisés et actionnables ;
— un catalogue de remédiations ;
— un agent autonome capable de mettre en œuvre une contre-mesure sur le système à défendre ;
— un environnement d'exercice où des agents autonomes attaquent un système d'information défendu par l'agent autonome ;
— un score calculé à l'issue de chaque exercice qui permet de mesurer la qualité de la remédiation.

Au cours de chaque exercice, les attaques exécutées sur le système à défendre engendrent des événements de sécurité produits par les différents équipements qui participent au fonctionnement du SI. Ces évènements, qu'ils soient réseaux, systèmes ou applicatifs sont utilisés pour détecter l'attaque en cours. Cette détection s'appuie sur un catalogue de règles mettant en œuvre les bonnes pratiques de sécurité ainsi que sur des indicateurs de compromissions qui se rapportent aux infrastructures, outils, et TTP couramment employés par les attaquants. Ainsi au cours d'un exercice, il est attendu qu'une ou plusieurs alertes de sécurité soient produites à chaque étape de la killchain opérée par l'attaquant. Toutes ces détections sont réalisées en temps réel et utilisées pour identifier les contre-mesures les plus adaptées pour faire face à la menace en cours.

Les règles et les IoC utilisés pour la détection sont fortement contextualisés c'est-à-dire qu'ils sont associés à des techniques d'attaques, des groupes d'attaquants, leurs campagnes et leurs outils. Cette qualification capitalisée grâce au langage STIX v2.1 [25], permet d'associer instantanément une détection avec par exemple, une technique d'attaque.

En outre, cette connaissance des menaces capitalisées dans une base de renseignement est également rendue fortement actionnable en associant les séquences de contre-mesures recommandées avec les techniques d'attaques et les outils des attaquants. Une telle association permet donc d'obtenir un ensemble cohérent de contre-mesures recommandées pour faire face à une détection.

Parmi les contre-mesures recommandées, certaines doivent être paramétrées pour adapter leurs mise en œuvre à l'environnement et ou l'attaque en cours. La résolution de l'ensemble des paramètres qui participent à la définition de la contre-mesure est nécessaire pour considérer

la contre-mesure comme étant applicable. Ainsi en cas de détection d'une attaque mettant en œuvre un canal de communication, la contre-mesure de filtrage réseau ne pourra être applicable qu'à la condition que la détection réalisée permette l'identification des caractéristiques réseaux nécessaires à la création de la règle de pare-feu. De la même manière, en cas d'infection virale détectée, la contre-mesure qui permet d'arrêter une machine virtuelle ne sera considérée comme applicable qu'à la condition que la détection réalisée permettent l'identification de la machine infectée. Tous ces paramètres et conditions d'instanciations sont définis à l'aide du langage de définition des contre-mesures OpenC2.

Au fur et à mesure de l'exécution d'un exercice, les détections et l'application des contre-mesures s'enchaînent jusqu'à la fin de l'exercice. Tout au long, le composant d'arbitrage de l'exercice calcul une récompense qui permet de mesurer la qualité du comportement défensif par rapport à l'attaquant et aux utilisateurs légitimes. Puis au cours de l'exercice suivant, la connaissance des contre-mesures retenues et exécutées lors des exercices précédents est utilisée pour sélectionner parmi les contre-mesures applicables celles-qui seront réellement appliquées. Cette sélection se faisant avec comme objectif de maximiser la récompense globale.

3.3. Environnement d'exercice

En ce qui concerne l'environnement d'exercice, deux problématiques sont à résoudre : d'une part, il s'agit d'arriver à atteindre une bonne représentativité des SI simulés afin de ne pas biaiser le moteur d'apprentissage, dans l'optique qu'il s'adapte à la variabilité et la dynamicité des SI surveillés. D'autre part, il est important de pouvoir désigner un vainqueur, afin d'estimer quelle stratégie d'attaque ou de défense a été la plus efficace.

La représentativité des SI simulés est ici prise en compte sous deux aspects. Tout d'abord, pour que le moteur d'apprentissage ne reçoive pas en entrée uniquement des événements malveillants, il est important de pouvoir produire de l'activité légitime sur l'ensemble du système d'information. Ceci est géré par un moteur de simulation de vie qui joue des comportements utilisateur sur les postes de travail, afin d'avoir une activité légitime. Ce moteur, qui fonctionne sans agent déployé sur l'OS afin d'être le plus furtif possible, réalise des enchaînements d'actions de vie en manipulant la souris et le clavier et en tenant compte de la sortie écran en termes de déport d'affichage. Les actions de vie jouées couvrent les catégories suivantes : ouverture et fermeture de session, navigation sur internet, manipulation de mails, accès à un partage de fichiers et manipulation de documents. Ces actions ont comme effet de laisser des traces réseau et système sur l'environnement, au même titre que les actions d'attaque. Cette activité de fond permet ainsi de générer des traces réelles au sein desquelles sont mélangées les traces laissées par le mode opératoire d'attaque.

La variabilité des SI simulés est assurée par un générateur de topologie paramétrable avec quelques caractéristiques : type de topologie (topologie à plat, cloisonnement en zones, administration en silo, etc.), plan d'adressage, nombre de machines et nombre de sous-réseaux. Sur ces environnements, des services et ressources sont déployés. L'ensemble de la chaîne de collecte des évènements est automatiquement déployée, afin de pouvoir surveiller différentes catégories d'événements potentiellement malveillants (flux depuis et vers l'extérieur, rebond latéraux, mise en persistence, ...).

Un mécanisme de scoring permet enfin d'évaluer le résultat de chaque exercice d'attaque-

défense, avec comme finalité d'identifier la contre-mesure la plus pertinente.

Ce mécanisme de scoring repose sur un point de vue multidimensionnel, classique en apprentissage statistique : il faut tenir compte à la fois de l'impact des actions volontairement délétères jouées par le moteur d'attaque sur le SI, mais aussi des actions involontairement délétères dues aux mauvais choix de réaction en défense.

Le score global est calculé en fonction des scores des trois entités actives sur le terrain d'exercice :

- Le score de l'agent de défense (blue team) est calculé sur la base du pourcentage d'assets compromis, en tenant compte de la criticité de chaque asset. Plus ce score est bas, plus l'agent de défense est efficace.
- Le score des utilisateurs (yellow team) est calculé en fonction de la capacité des utilisateurs à correctement accéder aux services exposés. Si l'agent de défense prend des décisions trop restrictives, il risque de bloquer l'accès à un service par un utilisateur légitime. Ce comportement, qui peut permettre de bloquer une éventuelle attaque, fera alors perdre des points à la yellow team.
- Le score de l'attaquant (red team) dépend de la tactique de l'attaque réussie. Ainsi le score augmente graduellement selon la difficulté de la tactique, allant de 0 (aucune attaque n'a réussi) à 100 pour une tactique de type impact (telle un rançongiciel) avec les paliers suivant : discovery, execution, initial access, defense evasion, credential access.

Ce score fait miroir aux compromis entre deux mesures statistiques complémentaires utilisées classiquement, par exemple la sensibilité et la spécificité, le pouvoir prédictif positif et le pouvoir prédictif négatif.

Ce score tridimensionnel permet en effet de prévenir le cas d'un système de défense trop prudent qui aurait un effet délétère sur le fonctionnement normal du SI (par exemple : blocage d'un site légitime) afin d'obtenir un plus fort taux de blocage des attaques. Cela apporte une certaine souplesse dans l'arbitrage, grâce au réglage d'un facteur d'équité entre la criticité du SI (par exemple la valeur des secrets qui s'y trouvent) et le coût d'une perturbation de son fonctionnement.

4. Implémentation

4.1. Environnement d'exercice

L'environnement d'exercice s'appuie d'une part sur la plateforme Cyber Range éditée par AMOSSYS, pour mettre automatiquement à disposition un système d'information simulant un environnement IT d'entreprise et, d'autre part, sur la plateforme SEKOIA.IO éditée par SEKOIA pour collecter les logs, détecter les attaques et proposer des contre-mesures vis-à-vis du SI à protéger.

La plateforme Cyber Range est utilisée pour instancier un système d'information fixe pour une campagne de tests donnée, et réinitialisé à chaque exercice. Cette plateforme est également employée pour instancier un environnement simulant internet, depuis lequel il est possible de déployer une infrastructure d'attaque. Cette dernière peut soit s'appuyer sur des sites légitimes compromis, afin de simuler des vecteurs d'attaque tels que le watering hole ou pour permettre

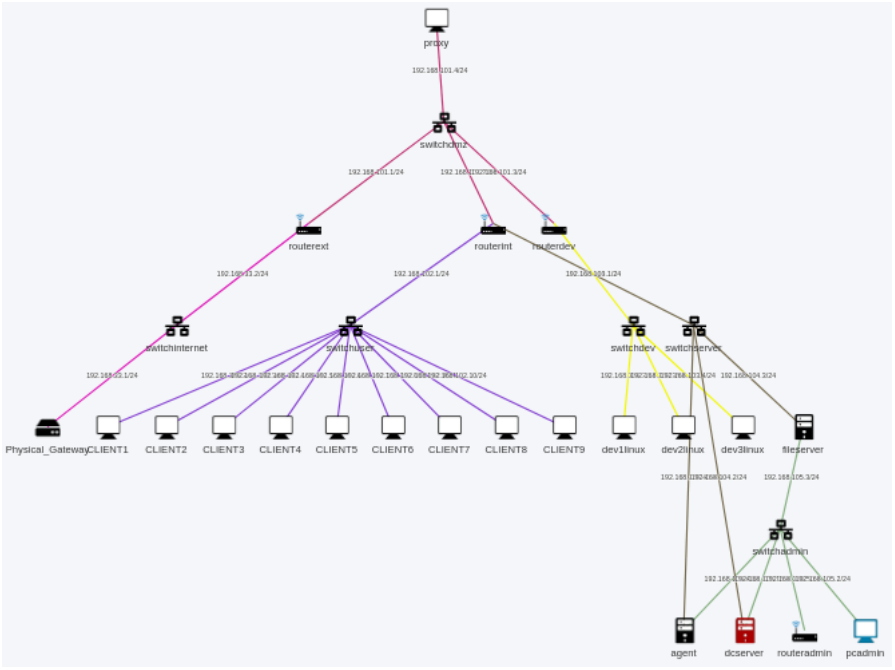

Figure 2 : Exemple de topologie instanciée par la plateforme d'expérimentation, simulant un SI d'entreprise et un SI « Internet » depuis lequel l'infrastructure d'attaque est ensuite automatiquement construite.

l'exfiltration de données sur des plateformes telles que Pastebin, soit instancier des éléments d'infrastructure d'attaque dédiés (Figure 2).

La plateforme SEKOIA.IO est utilisée pour superviser le SI à défendre en détectant des traces d'attaques, d'intrusions et de compromissions en temps réel. Elle s'appuie sur un catalogue de règles de détection et une base de connaissances sur les menaces. Les évènements produits par les équipements réseaux et les systèmes d'exploitations des postes utilisateurs et serveurs sont collectés, normalisés, enrichis puis analysés pour la détection. En cas d'alerte, la plateforme sélectionne la contre-mesure la plus appropriée et ordonne à un agent de remédiation présent sur le SI de la mettre en œuvre.

D'un point de vue temps d'exécution, l'objectif est d'arriver à limiter au maximum la durée d'un exercice afin de pouvoir produire un nombre suffisant d'itération dans un temps raisonnable (quelques jours au maximum) pour permettre l'apprentissage des meilleures contre-mesures. Lors de nos expérimentations, nous avons constaté un temps d'exercice en moyenne de 10 à 15 minutes.

Ce temps inclut les étapes suivantes :

— instanciation du système d'information (démarrage des machines virtuelles du simulateur) ;
— provisioning des systèmes d'exploitation, notamment pour déployer la chaîne de collecte des évènements ;
— exécution de scénarios de vie pour avoir une activité légitime de fond ;
— en parallèle, exécution d'un scénario d'attaque, en parcourant un graphe d'attaque ;

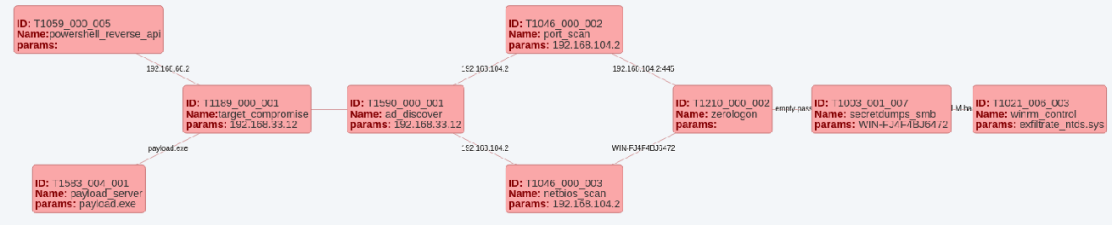

Figure 3 : Déroulement du scénario d'attaque simulant le mode opératoire TA505 (qui inclut notamment le vol de credential et la technique d'exploitation de la faille Zerologon en post exploitation).

— en parallèle, collecte des logs, détection des comportements suspects et mise en oeuvre de mesures de remédiation ;
— arrêt de la simulation.

Ce temps d'exercice peut apparaître particulièrement important, lorsqu'on le compare aux travaux récents sur l'apprentissage dans un environnement « réel » (comme par exemple AlphaStar). Nous devons néanmoins tenir compte, ici, d'actions réelles telles que les joueraient des profils utilisateurs et attaquants. Des optimisations sont néanmoins implémentées afin de limiter notamment le temps d'instanciation des SI (notion de snapshot de SI) et le temps de provisioning de ces SI (utilisation de volume partagés pour déployer des configurations, au lieu de dérouler des playbooks avec outils de provisioning qui peuvent être lents).

4.2. Simulation d'un mode opératoire

Lors de la phase d'expérimentation, un mode opératoire est modélisé : TA 505. TA 505 est un mode opératoire publiquement attribué à un groupe d'attaquant actif depuis 2014, ayant des motivations financières. Il a dernièrement marqué l'actualité pour avoir utilisé la vulnérabilité Zero Logon afin de réaliser des exploitations de privilège. Ce groupe d'attaquant est modélisable avec la kill chain suivante :

— phishing mail et compromission d'un poste de travail ;
— C2 transmettant des commandes à un malware exécuté sur un poste de travail ;
— scan de ports, pour la phase de reconnaissance interne ;
— scan Netbios, également pour la phase de reconnaissance interne ;
— attaque Zero Logon pour l'élévation de privilège ;
— utilisation de SMB pour l'extraction de secrets ;
— tactique d'impact (ransomware).

La simulation du mode opératoire TA505 permet de dérouler le scénario d'attaque illustré sur la Figure 3.

4.3. Agent de remédiation

Une fois que le SI simulé est créé via une expérimentation, la liste des principaux équipements sensibles (*assets*) est envoyée à SEKOIA.IO. Ces informations sont utilisées à la fois pour déterminer le niveau de criticité des alertes (plus les équipements associés à l'alerte sont

Figure 4 : Processus d'apprentissage par renforcement.

sensibles, plus l'alerte sera critique) et pour permettre de déterminer la cible des contre-mesures (« éteindre la machine cible "y" »).

L'application des contre-mesures au sein du SI simulé est réalisée grâce à un agent spécialement développé. Ce dernier s'inscrit comme consommateur du flux d'alertes produites par SEKOIA.IO et permet d'exécuter les contre-mesures qui y sont associées. L'agent se charge alors d'exécuter les commandes nécessaires sur le SI grâce à des playbook Ansible puis d'indiquer à SEKOIA.IO que le résultat de l'exécution de la contre mesure. Les playbooks Ansible fonctionnent différemment selon les équipements ciblés, l'application peut se faire via une connexion SSH pour les postes GNU/Linux ou via des commandes PowerShell distantes pour les postes Windows.

L'agent chargé d'appliquer les contre-mesures est considéré comme partie intégrante du système d'information. Il dispose naturellement des accès administrateurs sur les différents équipements afin de pouvoir agir sur les différents équipements qui composent le système d'information.

4.4. Modélisation de l'environnement d'apprentissage

La modélisation de l'environnement d'apprentissage est basée sur OpenAI Gym, une boîte à outils permettant de développer des environnements interactifs dans lesquels on va pouvoir y entraîner et évaluer un agent autonome selon plusieurs algorithmes d'apprentissage par renforcement. OpenAI Gym fournit donc la base pour créer un environnement dans lequel l'agent va pouvoir interagir et apprendre, via la définition d'actions, d'observations et d'une fonction de récompense (reward), voir Figure 4.

Dans notre cas de figure, les contre-mesures à appliquer sont les actions. Les observations sont un échantillon caractéristique de ce qu'est la nouvelle alerte que l'on considère, à savoir son type et sa catégorie, la règle qui l'a générée ainsi que son urgence. Enfin, la fonction de récompense est calculée à partir des 3 scores (blue team, red team et yellow team), permettant un apprentissage non biaisé de l'agent, de sorte qu'il ne favorise pas un comportement plutôt qu'un autre dans son choix d'actions à mener.

À l'initialisation, nous définissons un point de l'espace correspondant à une situation que l'on pourrait qualifiée de critique, à savoir que l'ensemble des assets sont compromis (blue team), toutes les attaques ont été menées avec succès (red team), et les utilisateurs ne parviennent pas à accéder aux services exposés (yellow team). Dans cette situation, les trois scores sont à 100.

À chaque étape, de nouveaux scores sont calculés en fonction des actions menées par l'agent et des attaques simulées par le scénario. La distance entre le point des scores actuels et le point

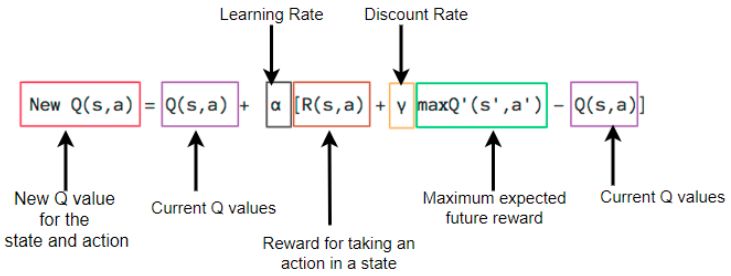

Figure 5 : Fonction de valeur Q-Learning, avec alpha et gamma respectivement facteur d'apprentissage (entre 0 et 1) et facteur d'actualisation.

critique est recalculée et comparée à la distance de l'étape précédente. Ainsi, la fonction de récompense est calculée sur la base du rapprochement ou de l'éloignement à cette situation critique par rapport à la distance précédente. De plus, une composante temporelle (de temps sans alerte) est insérée dans le calcul de la fonction de récompense, pour prendre en compte la situation où le score tridimensionnel n'évolue pas mais les actions précédemment menées empêchent l'apparition de nouvelles alertes. L'agent, au cours de son apprentissage, va donc chercher à appliquer les actions qui maximisent sa fonction de récompense.

4.5. Choix des mesures de remédiation

Concernant le choix des contre-mesures, après définition de l'environnement via OpenAI Gym, nous avons choisi d'entraîner l'agent sur les bases d'un apprentissage par renforcement. De plus, compte tenu de l'aspect continu des observables et de la potentielle forte quantité d'actions à mener, notre choix s'est porté vers un algorithme d'apprentissage profond (Deep Reinforcement Learning) : DQN (Deep Q-Network), fourni par la librairie keras-rl2.

Le fonctionnement de cet algorithme se base en partie sur un apprentissage Q-Learning, dont l'objectif est d'optimiser la politique de sélection des actions en maximisant la récompense (reward). L'agent qui interagit dans un environnement avec un ensemble d'états (ou observations) S et d'actions A, à chaque passage d'un état s à un état s', reçoit une récompense r. Pour chaque état, l'action optimale correspond à celle avec la plus grande récompense sur le long terme. Cette récompense est la somme pondérée de l'espérance mathématique des récompenses de chaque étape future à partir de l'état actuel. Ainsi l'algorithme calcule une fonction de valeur action-état, qui est mise à jour à chaque étape de la façon suivante (Figure 5).

Dans la situation où l'environnement comprend une grande quantité d'états ou observations ou que ceux-ci sont continus, ainsi qu'un large champ d'actions, il devient laborieux de créer et mettre à jour la table des Q-values (table des Q(s,a)). Ainsi on utilise un réseau de neurones pour approximer ces Q-values. En effet, les états ou observations sont donnés en entrée et la fonction de Q-value pour chaque action possible est générée en sortie. Ainsi l'expérience passée est stockée en mémoire et la prochaine action est déterminée par le maximum des sorties du Q-network.

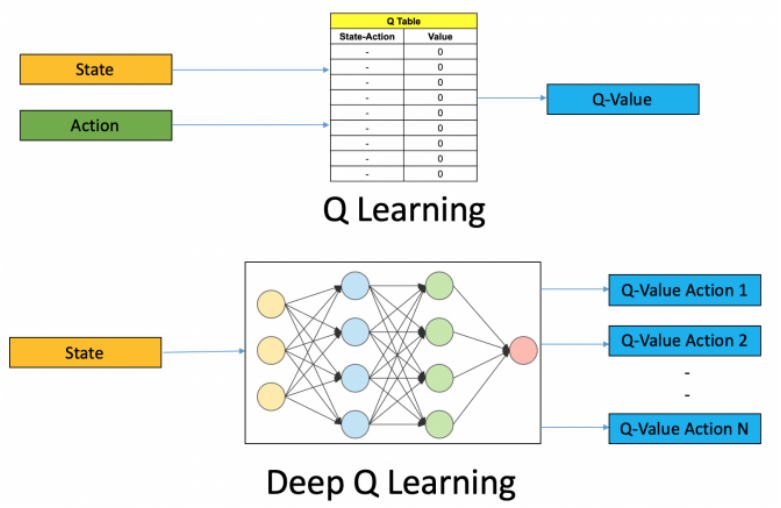

Figure 6 : Comparaison entre les algorithmes Q-Learning et Deep Q-Learning (ou Deep Q Network). Premier cas calcul de la fonction de valeur stockée dans une table, deuxième cas prédiction à partir d'un calcul de poids via un réseau de neurones.

```
[+] Computing attack score
  [+] tactic 'execution' (technique 1059_000_005 - 'powershell_reverse_api') gives 20 points
  [+] tactic 'initialaccess' (technique 1583_004_001 - 'payload_server') gives 30 points
  [+] tactic 'discovery' (technique 1518_001_001 - 'av_scanner') gives 10 points
  [+] tactic 'defenseevasion' (technique 1562_002_001 - 'disable_windows_logging') gives 30 points
  [+] tactic 'defenseevasion' (technique 1562_004_001 - 'disable_firewall') gives 30 points
  [+] tactic 'credentialaccess' (technique 1555_000_001 - 'lazagne') gives 50 points
  [+] tactic 'discovery' (technique 1592_004_001 - 'os_host_info') gives 10 points
  [+] tactic 'discovery' (technique 1518_000_001 - 'list_local_process') gives 10 points
  [+] tactic 'discovery' (technique 1518_000_002 - 'list_local_software') gives 10 points
  [+] tactic 'credentialaccess' (technique 1003_001_006 - 'mimikatz_wdigest') gives 50 points
  [+] tactic 'impact' (technique 1486_000_001 - 'ransomware') gives 100 points
```

Figure 7 : Calcul du score de la red team. Dans cet exercice, l'agent d'attaque a réussi à atteindre la tactique "impact", ce qui lui permet d'obtenir le score maximum, 100.

5. Expérimentations

Le déroulé du scénario d'attaque, modélisant le mode opératoire TA 505, permet d'obtenir l'évolution suivante du score. Chaque tactique donne en effet un score dépendant de son importance et de sa complexité dans une kill chain. Sur la Figure 7, le scénario a pu être exécuté jusqu'à la tactique "impact", permettant ainsi d'obtenir le score maximal (100) pour l'agent autonome d'attaque.

Du point de vue de la défense, nous nous sommes positionnés dans le cas d'observations à treize variables pour sept contre-mesures applicables. À chaque étape de l'entrainement, les alertes courantes sont traitées sous la forme de paquet, dont on vient extraire les caractéritiques, pour définir les variables d'observations. Ces variables d'observations sont les suivantes : type d'alerte, catégorie de l'alerte, règle ayant levé l'alerte, urgence de l'alerte et présence ou non de `source.ip`, `destination.ip`, `source.domain`, `url.domain`, `destination.domain`,

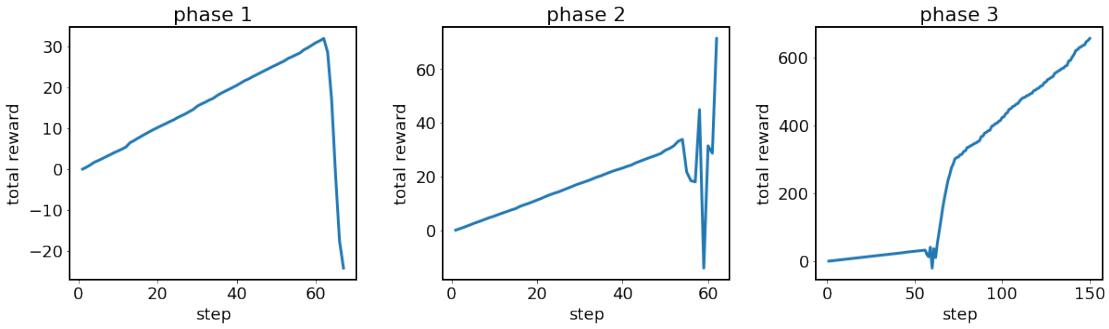

Figure 8 : Cumul de la fonction de récompense lors des différentes phases de l'apprentissage, en fonction des étapes de l'épisode. Phase 1 : exploration et victoire de la red team. Phase 2 : premiers contres coté défense avec des contre-mesures très restrictives. Phase 3 : exploitation des entrainements précédents et blocage des attaques red team. Plus la récompense croît, plus la situation est stable et donc l'attaque potentiellement bloquée

`log.hostname`, `process.pid`, `process.executable`, `process.name` dans les événements associés. L'ensemble des contre-mesures applicables est :

- ne rien faire
- bloquer via le proxy mandataire le nom de domaine source présent dans l'alerte,
- bloquer via le proxy mandataire le nom de domaine destination présent dans l'alerte,
- arrêter, sur tous les postes, tous les processus caractérisés par l'identifiant de processus et/ou le chemin de l'exécutable présent dans l'alerte,
- éteindre la machine qui a produit l'événement à l'origine de l'alerte,
- ajouter l'adresse IP source présente dans l'alerte à tous les pare feux des postes et serveurs,
- ajouter l'adresse IP destination présente dans l'alerte à tous les pare-feux des postes et serveurs.

Le réseau de neurones utilisé dans l'algorithme de Deep Q-Learning est un perceptron constitué de 2 couches cachées denses de 24 neurones. Nous avons entraîné l'agent sur plus de 70 épisodes et observé l'évolution de la fonction de récompense, ainsi que des scores de blue, red et yellow team. Les épisodes pouvant atteindre jusqu'à 150 étapes, le nombre maximal à partir duquel on considère que la situation est stable, soit le SI totalement compromis par l'attaque, soit l'attaque correctement bloquée.

On observe globalement trois phases lors de l'apprentissage de l'agent autonome (Figure 8). La phase 1 est une phase d'exploration, dans laquelle l'agent explore l'ensemble des actions/contre-mesures applicables en fonction des observations qui lui sont présentées, ayant pour conséquence le succès des attaques red team, dans la majorité des épisodes. Les premiers effets de la découverte réseau se font ressentir à partir de l'étape 50, ce qui va lever les premières alertes. Puis au fur et à mesure de la progression de l'attaquant, les scores blue et red baissent. Quelques étapes suffisent pour que l'attaquant réussisse à mettre en oeuvre ses différentes techniques d'exécution et d'évasion telles que la désactivation du pare-feu et de la journalisation. Les étapes finales de l'épisode correspondent aux phases finales de la kill chain de l'attaque avec la mise en oeuvre d'un ransomware d'où la chute importante de la récompense. Lors de la phase 2, l'agent commence à

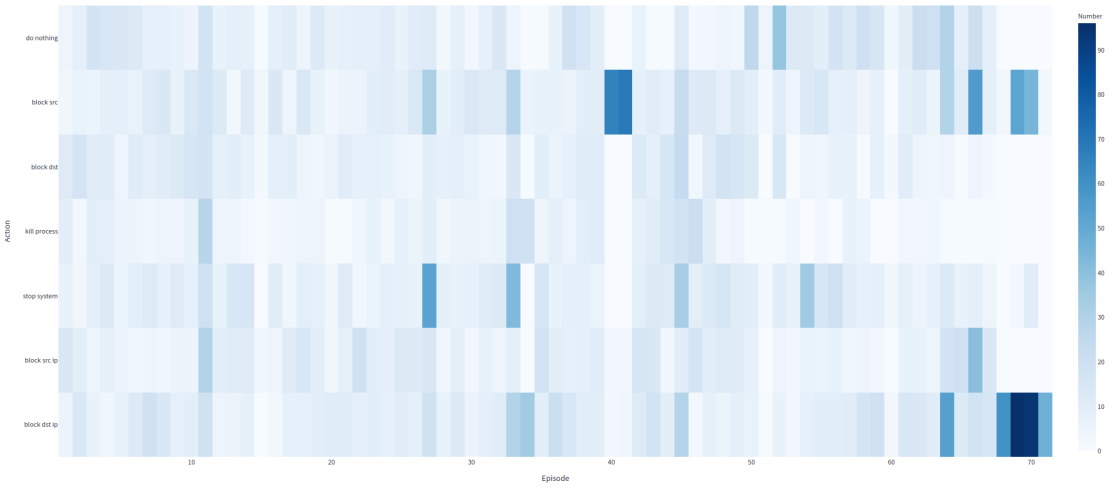

Figure 9 : Représentation heatmap des actions choisies par l'agent lors de l'apprentissage en fonction des épisodes.

comprendre les actions qui permettraient de contrer l'attaquant, on observe quelques épisodes où les attaques échouent. Néanmoins, les contre-mesures mises en place sont très restrictives, de sortes que le score yellow team est très haut. A la phase 3, l'agent explore ce qu'il a appris dans le passé et parvient à contrer l'attaque, tout en stabilisant le SI.

La Figure 9 nous permet de distinguer ces trois phases. Les 20 premiers episodes montrent une répartition assez homogène dans le choix des contre-mesures à appliquer. Dans les épisodes 20 à 60, l'agent commence à explorer ses résultats précédents pour parvenir à freiner l'attaque. À partir de l'épisode 60, on observe que les actions menées sont beaucoup plus ciblées, avec une forte représentation des contre-mesures de blocage du domaine source et de l'adresse ip de destination combinée à une plus faible représentation de la contre-mesure visant à éteindre une machine (`stop system`).

La Figure 10 illustre l'augmentation de la fréquence de blocage des attaques par l'agent de remédiation. Plus on avance dans les épisodes, plus le cumul des récompenses explose fréquemment, correspondant à un blocage efficace de l'attaquant, tout en préservant les usages légitimes du SI. Évidemment, ce scénario est assez simplifié, de sorte que dans un cas plus complexe, il faudrait plus d'épisodes et donc de temps d'entraînement pour parvenir à des résultats similaires.

6. Conclusion et perspectives

La validation automatique de contre-mesure est importante afin d'en évaluer l'efficacité sur le service protégé. La création d'une plateforme d'automatisation de simulations d'attaques et d'apprentissage en défense afin de permettre une évaluation automatique soulève plusieurs problématiques, aussi bien pour le volet offensif, le volet défensif que pour l'environnement d'apprentissage. Après avoir explicité ces verrous, nous avons proposé un mode opératoire

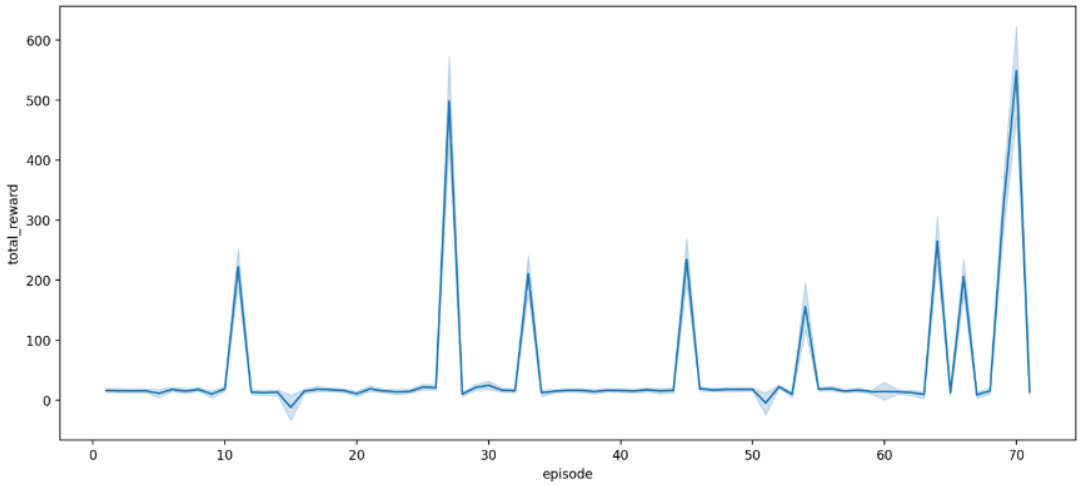

Figure 10 : Evolution du cumul de la récompense à la fin de chaque episode. Fonction de récompense élevée dans le cas d'une attaque bloquée.

permettant de traiter l'automatisation des différentes étapes : choix du groupe d'attaquant, construction d'un graphe d'attaque, exécution d'un scénario d'attaque, ainsi que la remédiation avec un agent autonome.

La plateforme de simulation DALID répond aux problématiques liées à l'environnement d'exercice, avec des SI simulés représentatifs, et un mécanisme de parcours de graphe d'attaque permettant de rejouer et de tester les possibilités de réaction. Cette plateforme a pour ambition de devenir un cadre de gamification de l'attaque-défense permettant d'évaluer l'efficacité d'architectures de lutte informatique défensive. En particulier, il serait pertinent de pouvoir confronter des « IA » de défense face à des « IA » d'attaque.

Références

[1] A. Seydtaghia, Comment un ransomware a coulé ma société, Le Temps (2021). URL : https://www.letemps.ch/economie/un-ransomware-coule-societe.

[2] C. C. Editor, tactics, techniques, and procedures (TTP) - Glossary | CSRC, 2021. URL : https://csrc.nist.gov/glossary/term/Tactics_Techniques_and_Procedures.

[3] C. C. Editor, IoC - Glossary | CSRC, 2021. URL : https://csrc.nist.gov/glossary/term/ioc.

[4] Tear Security - Red Team Automation and Adversary Simulation, 2021. URL : https://tearsecurity.com/.

[5] Meet the Atomic Family, 2021. URL : https://atomicredteam.io/.

[6] MITRE ATT&CK®, 2021. URL : https://attack.mitre.org/.

[7] Red Team Automation (RTA), 2021. URL : https://github.com/endgameinc/RTA, original-date : 2018-03-19T19:59:39Z.

[8] Metta, 2021. URL : https://github.com/uber-common/metta, original-date : 2017-11-01T21:24:47Z.
[9] APT Simulator, 2021. URL : https://github.com/NextronSystems/APTSimulator, original-date : 2018-02-03T14:19:42Z.
[10] CALDERA™, 2021. URL : https://github.com/mitre/caldera, original-date : 2017-11-29T01:25:10Z.
[11] machine_learning_security/DeepExploit at master · 13o-bbr-bbq/machine_learning_security, 2021. URL : https://github.com/13o-bbr-bbq/machine_learning_security.
[12] GyoiThon, GyoiThon : Next generation penetration test tool, 2021. URL : https://github.com/gyoisamurai/GyoiThon, original-date : 2018-03-07T05:14:12Z.
[13] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement Learning : A Survey, Journal of Artificial Intelligence Research 4 (1996) 237–285. URL : https://www.jair.org/index.php/jair/article/view/10166. doi :10.1613/jair.301.
[14] R. S. Sutton, A. G. Barto, Reinforcement learning : An introduction, MIT Press, Cambridge, MA, 2nd Edition (2017).
[15] T. Chen, J. Liu, Y. Xiang, W. Niu, E. Tong, Z. Han, Adversarial attack and defense in reinforcement learning-from ai security view, SpringOpen (2019).
[16] OpenAI, Gym : A toolkit for developing and comparing reinforcement learning algorithms, 2021. URL : https://gym.openai.com.
[17] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, S. Legg, AI Safety Gridworlds, arXiv:1711.09883 [cs] (2017). URL : http://arxiv.org/abs/1711.09883, arXiv : 1711.09883.
[18] T. T. Nguyen, V. J. Reddi, Deep reinforcement learning for cyber security, ArXiv (2020).
[19] A. Molina-Markham, R. Winder, A. Ridley, Network defense is not a game, ArXiv (2021).
[20] A. Molina-Markham, C. Miniter, B. P. (Mitre), A. R. (NSA), Network environment design for autonomous cyberdefense, ArXiv (2021).
[21] M. D. R. Team., Cyberbattlesim, https://github.com/microsoft/cyberbattlesim, 2021. Created by Christian Seifert, Michael Betser, William Blum, James Bono, Kate Farris, Emily Goren, Justin Grana, Kristian Holsheimer, Brandon Marken, Joshua Neil, Nicole Nichols, Jugal Parikh, Haoran Wei.
[22] E. C.Walter, K. J. Ferguson-Walter, A. D.Ridley, Incorporating deception into cyberbattlesim for autonomous defense, ArXiv (2021).
[23] M. Standen, M. Lucas, D. Bowman, T. J. Richer, J. Kim, D. Marriott, Cyborg : A gym for the development of autonomous cyber agents, ArXiv (2021).
[24] L. Li, R. Fayad, A. Taylor, Cygil : A cyber gym for training autonomous agents over emulated network systems, ArXiv (2021).
[25] Stix™ version 2.1, 2021. URL : https://docs.oasis-open.org/cti/stix/v2.1/csprd01/stix-v2.1-csprd01.html.