POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Renforcement de systèmes de détection d'intrusions par des attaques GAN et métaheuristiques

SIMON MSIKA
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Janvier 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Renforcement de systèmes de détection d'intrusions par des attaques GAN et métaheuristiques

présenté par **Simon MSIKA**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Samuel PIERRE, président
Alejandro QUINTERO, membre et directeur de recherche
Foutse KHOMH, membre et codirecteur de recherche
Gabriela NICOLESCU, membre

DÉDICACE

À toute ma famille.

REMERCIEMENTS

Je tiens à remercier mes directeurs de recherche Alejandro Quintero et Foutse Khomh pour leur aide précieuse, leur disponibilité et leur sympathie tout au long de ma maîtrise. Je remercie également l'École polytechnique de Montréal pour m'avoir permis d'étudier au sein d'une prestigieuse institution, ainsi que l'École polytechnique de Paris pour m'avoir donné l'opportunité d'effectuer cet échange.

RÉSUMÉ

Un système de détection d'intrusion est un outil capital en cybersécurité. Il sert aux administrateurs réseau d'identifier le traffic internet malicieux, et les cyberattaques : une fois celle-ci détectée, l'administrateur prend des décisions afin d'arrêter l'attaque en cours. Grâce au développement de l'Intelligence artificielle, de plus en plus de système de détection d'intrusion sont basés sur des algorithmes d'apprentissage automatique. Ces algorithmes ont en effet la particularité d'être capable de traiter des données rapidement une fois entraînés, ce qui est un facteur important dans le cadre de la détection d'attaques.

Ces algorithmes de détection se basent sur des métriques issues du réseau, comme le débit entrant par seconde, le port de destination, ... Des ensembles de données des métriques lors de différents types d'attaques informatiques existent, et c'est sur l'un d'eux que nous nous baserons pour notre étude.

Néanmoins, diverses études ont montré que ces algorithmes sont peu robustes lorsque confrontés à des types d'attaques nouveaux. Considérant le nombre croissant de nouvelles attaques informatiques ainsi que la possibilité de générer des attaques grâce à des algorithmes d'apprentissage, disposer d'un outil de détection capable de déceler ces attaques inhabituelles devient une nécessité.

Ce travail explore la possibilité d'utiliser des modèles génératifs dans le but de rendre des systèmes de détection basés sur des algorithmes d'intelligence artificielle plus robuste. Nous avons développé la méthode SIGMA, utilisant des Generative Adversarial Networks (GANs) et des métaheuristiques afin de confronter des IDS à de nouveaux types d'attaques. L'usage de ces deux méthodes permet en effet d'explorer un espace de solutions trop grand pour effectuer des méthodes de résolution exactes.

SIGMA génère des exemples d'attaques, itérativement, et les utilise pour réentraîner le système de détection jusqu'à convergence, c'est à dire jusqu'à ce qu'il ne soit plus possible d'améliorer l'IDS.

Une itération consiste en une étape générative, au cours de laquelle des attaques sont générées à l'aide de métaheuristiques et de GANs ; une étape d'évaluation au cours de laquelle le nouveau taux de détection d'attaques générées est mesuré ; et une étape d'entraînement, durant laquelle l'IDS apprend de ces nouvelles attaques générées.

Nous avons évalué la méthode SIGMA pour quatre différents algorithmes de classification, jouant le rôle de système de détection d'intrusion. Un algorithme hybride génétique-recherche

locale, et un réseau GAN ont servi à générer de nouveaux ensembles d'attaques.

Les résultats montrent qu'il est possible de générer des attaques non-détectées par les systèmes de détection, et que la méthode SIGMA permet de détecter 100% des nouvelles attaques en seulement deux itérations.

Les résultats sont donc positifs. L'objectif de ce travail était d'améliorer la robustesse des systèmes de détection d'intrusion face à des attaques générées par de l'intelligence artificielle. L'objectif a été atteint, et des perspectives d'amélioration du modèle pour des classifieurs plus complexes peut-être envisagées.

ABSTRACT

An Intrusion Detection System (IDS) is a key cybersecurity tool for network administrators as it identifies malicious traffic and cyberattacks. With the recent successes of machine learning techniques such as deep learning, more and more IDS are now using machine learning algorithms to detect attacks faster. However, these systems lack robustness when facing previously unseen types of attacks. With the increasing number of new attacks, especially against Internet of Things devices, having a robust IDS able to spot unusual and new attacks becomes necessary.

Those algorithms are trained with data from the network, such as the destination port, the incoming data rate, ... Datasets of network data during cyberattack are accessible, and we'll be using a recent dataset for our study.

This work explores the possibility of leveraging generative adversarial models to improve the robustness of machine learning based IDS. More specifically, we propose a new method named SIGMA, that leverages adversarial examples to strengthen IDS against new types of attacks. Using Generative Adversarial Networks (GAN) and metaheuristics, SIGMA generates adversarial examples, iteratively, and uses it to retrain a machine learning-based IDS, until a convergence of the detection rate (i.e. until the detection system is not improving anymore).

Using metaheuristics and GANs allow to efficiently explore the solution space of a problem too complex to use exhaustive search methods.

A round of improvement consists of a generative phase, in which we use GANs and metaheuristics to generate instances ; an evaluation phase in which we calculate the detection rate of those newly generated attacks ; and a training phase, in which we train the IDS with those attacks.

We have evaluated the SIGMA method for four standard machine learning classification algorithms acting as IDS, with a combination of GAN and a hybrid local-search and genetic algorithm, to generate new datasets of attacks.

Our results show that SIGMA can successfully generate adversarial attacks against different machine learning based IDS. Also, using SIGMA, we can improve the performance of an IDS to up to 100% after as little as two rounds of improvement.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

IDS	Intrusion Detection System
NIST	National Institute of Standards and Technology
IoT	Internet of Things
GAN	Generative Adversarial Network
SVM	Support Vector Machine
RF	Random Forest
DOS	Denial of Service
DDOS	Distributed Denial of Service
XML	Extensible Markup Language
XSS	Cross Site Scripting
U2R	User to Root
R2L	Remote to Local

CHAPITRE 1 INTRODUCTION

L'Internet des objets est une infrastructure consistant en un réseau d'objets physiques interconnectés, capables de communiquer les uns avec les autres. Il s'agit d'étendre les applications d'Internet à différents domaines de notre vie quotidienne comme la santé, le transport ou la domotique.

Plus communément appelé Internet of Things (IOT), ce domaine de l'informatique est en pleine expansion, et le nombre d'objets connectés intelligents ne cesse de croître : de la voiture autonome aux appareils électroménagers intelligents, les domaines d'application continuent de se diversifier. D'ici 2025, le nombre d'objets connectés dans le monde devraient atteindre 21.5 milliards, selon le site iot-analytics [2]. Les objets intelligents vont donc rapidement prendre une importance significative dans notre environnement et notre vie de tous les jours.

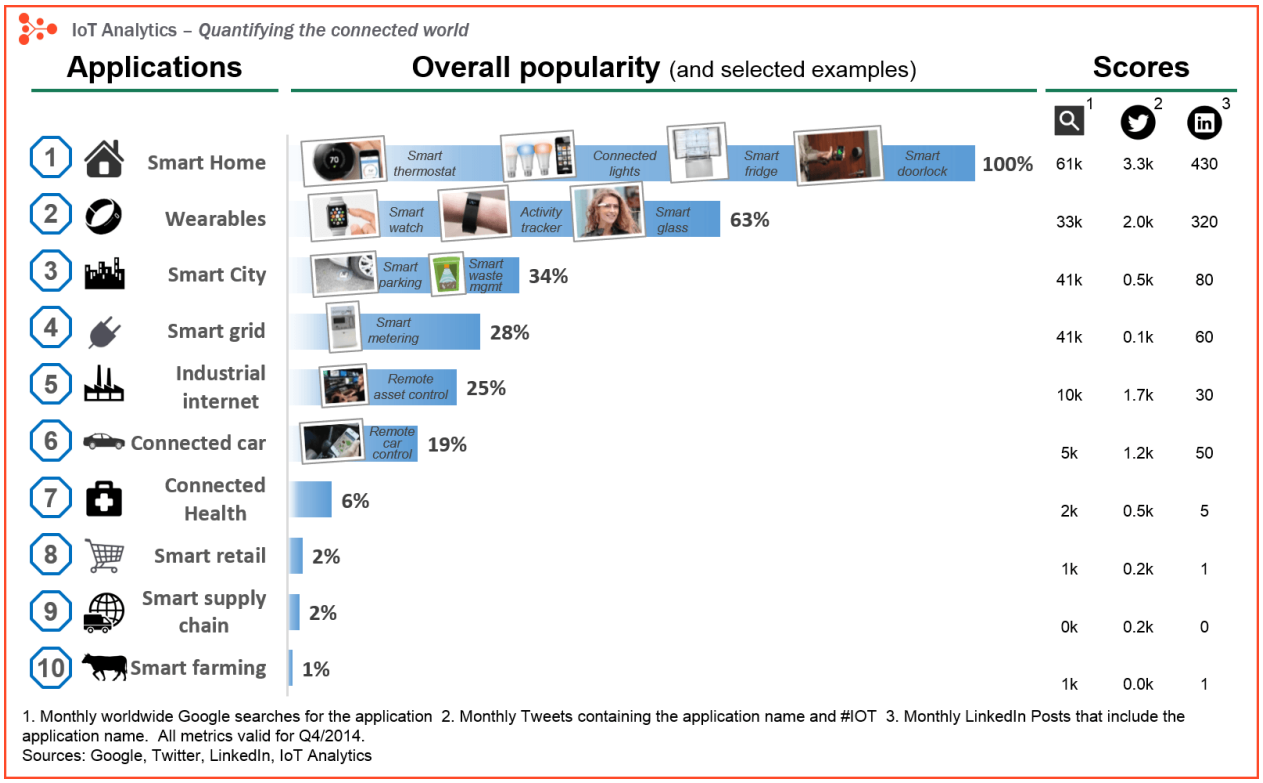

Figure 1.1 Exemples d'applications de l'Internet des objets, classés par popularité sur Internet.

Avec ces nouvelles technologies viennent de nouveaux enjeux : la sécurité de ces objets devient particulièrement critique. De fait, une application pour l'IOT doit tenir compte de la

complexité des systèmes répartis, de la variété des protocoles de communication... Autant de facteurs pouvant constituer des failles exploitables par un attaquant malicieux. Dans les faits, ces objets ne sont pas encore suffisamment sécurisés. On peut en effet penser à la vulnérabilité exploitée par le robot Mirai en 2016, prenant le contrôle de près de 600 000 d'objets connectés afin de mener une attaque de déni de service, causant une période d'indisponibilité conséquentes pour de nombreux sites internet (Twitter, Netflix, CNN, ...). Ce type d'attaques contre les objets connectés vulnérables continue encore, à des échelles moindre, aujourd'hui. [3]

Pire encore, dans le cas d'objets connectés dans le domaine de la domotique, une attaque permettrait non seulement de récupérer des informations sur les utilisateurs en mettant en danger leur vie privée, mais aussi potentiellement de contrôler les différents objets à des fins malicieuses, menant à des conséquences bien plus graves. [4] En conséquence, le risque d'attaque par intrusion dans un réseau d'objets connectés devient problématique de par les interactions possibles avec le monde physique : Par exemple, il deviendrait possible pour un attaquant de contrôler une voiture intelligente à distance et de provoquer des accidents, ou d'espionner quelqu'un chez lui à l'aide de sa caméra connectée.

Il est donc désirable de développer des systèmes permettant la détection de ces attaques.

1.1 Contexte

Détection d'une attaque dans un réseau informatique

Afin de prévenir les intrusions dans leurs réseaux, les entreprises utilisent un dispositif de détection d'intrusions, ou Intrusion Detection System (IDS). Celui vient en complément de pare-feux traditionnels, et est chargé d'analyser le trafic et de repérer les activités sur le réseau qui lui semblent anormales. Une fois une anomalie détectée, un administrateur du réseau est notifié, celui-ci décidant alors les mesures à prendre. Un IDS est donc un composant **essentiel** pour la sécurité d'une entreprise.

Comme exemple de dispositif de détection d'intrusion, on peut citer les logiciels libres Snort ou Bro.

Une première méthode est d'utiliser des algorithmes de recherche de motif afin de repérer un motif d'une attaque déjà connue (une séquence de bytes dans le trafic, ou une séquences d'instructions utilisées par un malware). Ainsi, la mise à jour fréquente de la base de données de signatures d'attaques est capitale afin d'avoir un système robuste.

Une seconde méthode est l'analyse de métriques du réseau. Il est en effet possible de déduire

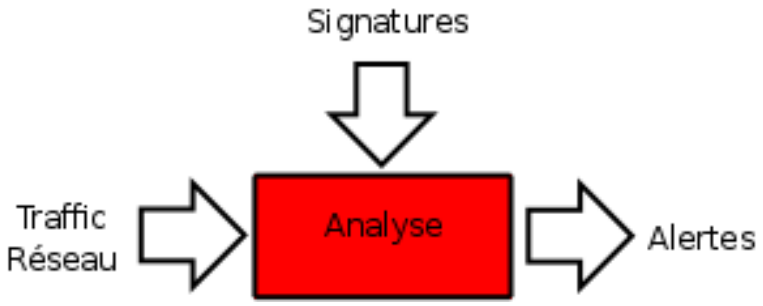

Figure 1.2 Schéma du fonctionnement d'un Système de détection d'intrusion de réseau.

de par certaines données concernant le réseau si l'entreprise subit une attaque ou non. Par exemple, le flux entrant par seconde peut indiquer si une attaque de Déni de Service (ou DOS) est en cours, celui-ci se caractérisant par un afflux massif de requêtes vers le serveur de l'entreprise.

Utilisation de l'intelligence artificielle

Plus récemment, du fait de la progression des algorithmes d'intelligence artificielle, les IDS utilisent de l'apprentissage machine pour faire face à la complexité croissante des cyber-attaques. En effet, l'intelligence artificielle permet de traiter des quantités massives de données efficacement, et s'avère donc être un outil particulièrement utile pour analyser le flux de données internet toujours grandissant que nous émettons.

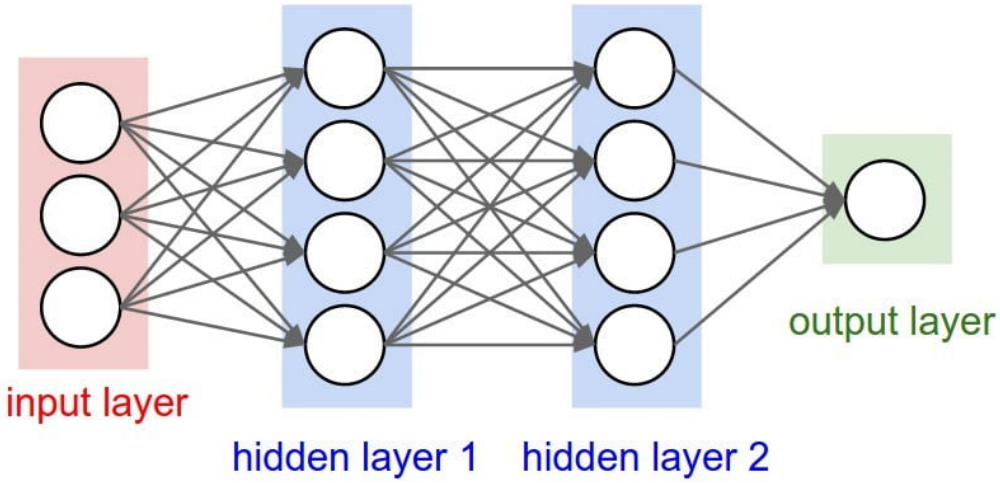

Figure 1.3 Schéma d'un classificateur par réseau de neurones.

De fait, l'apprentissage automatique est très souvent utilisé à des fins de classification. [5] Des algorithmes simples et très efficaces existent, comme les réseaux neuronaux. L'algorithme

"apprend" alors sur un ensemble dit d'apprentissage, afin de pouvoir classifier les futures données qui lui seront soumises. Ces algorithmes, bien que parfois simples à implémenter, peuvent atteindre une précision supérieure à 95%.

L'analyse d'importantes quantités de données bien plus rapidement que peuvent le faire les algorithmes classiques permet donc d'augmenter la vitesse et l'efficacité du système. Ceci est particulièrement important dans le cas d'une analyse "en direct" du flux de données, et s'avère être un enjeu clé dans le cas de l'Internet des objets : l'abondance d'objets connectés et l'importance du trafic qu'ils génèrent signifie qu'il est nécessaire d'avoir des outils rapides et robustes afin de détecter des anomalies dans le trafic internet. Il s'agirait dans notre cas de classifier le trafic internet capturé par le détecteur d'intrusion en trafic "sain" ou "malicieux", voire en différents types d'attaques.

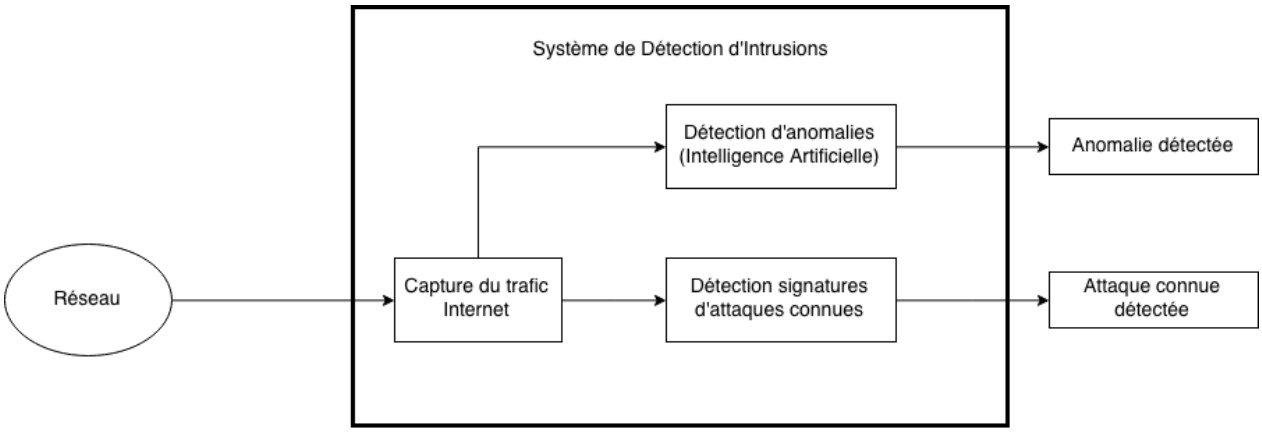

Figure 1.4 Fonctionnement général d'un Système de détection d'intrusions.

1.2 Éléments de la problématique

Néanmoins, la robustesse des systèmes de détection d'intrusion est variable lorsque celui-ci est confronté à un comportement nouveau : une attaque nouvelle peut ne pas être détectée, tout comme du trafic légitime peut être identifié comme une menace. Étant donné l'importance de la protection des données, en particulier dans le cadre de l'Internet des objets, continuer d'améliorer le taux de détection de comportement malicieux de nos systèmes d'intrusions est primordial.

Les nouvelles attaques sont extrêmement fréquentes : selon le National Institute of Standards and Technology (NIST), il y a eu plus de 16 000 failles de sécurité recensées au cours de l'année 2018.

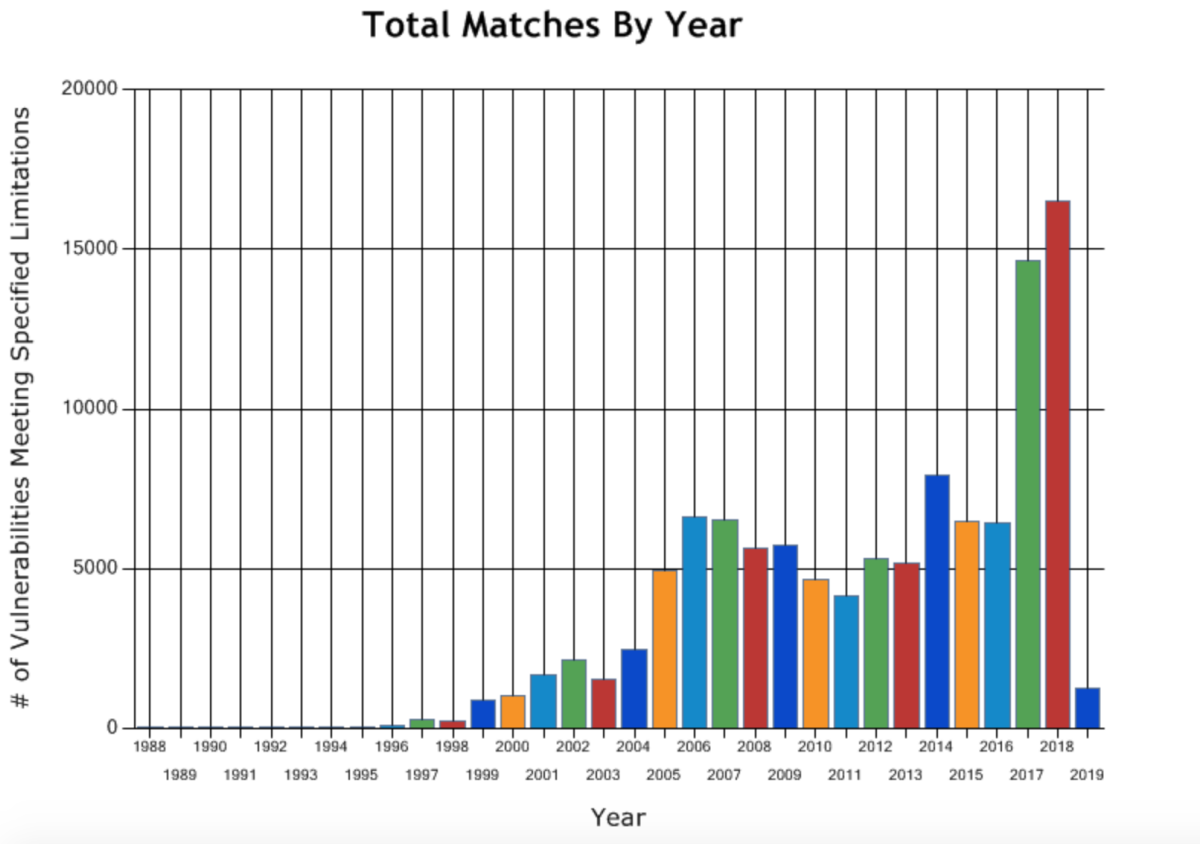

Figure 1.5 Évolution du nombre de failles de sécurité recensées par le NIST au cours du temps.

Il est en outre possible de générer des attaques nouvelles à partir d'attaques pré-existantes.

Des techniques d'Intelligence Artificielle non supervisées peuvent créer des attaques de toute pièce, et celles-ci peuvent donc potentiellement ne pas être détectées par l'IDS. En particulier, les Generative Adversarial Networks (ou GAN) ont la capacité de créer de nouvelles attaques à partir d'attaques pré-existantes. Ce type d'algorithme s'est illustré en 2014 de par sa capacité à générer des images singulièrement réalistes par rapport à d'autres techniques d'intelligence artificielle, avant de s'exporter dans d'autres domaines comme la sécurité et la génération de cyberattaques [6], ou de malwares [7].

1.3 Objectifs de recherche

1.3.1 Objectif global

L'objectif global de ce projet est la conception d'une méthode de renforcement de systèmes de détection d'intrusions face à des attaques génératives.

1.3.2 Objectifs spécifiques

Afin d'atteindre cet objectif, nous avons établi plusieurs étapes :

- La conception d'une méthode générique de renforcement de systèmes de détection d'intrusions.
- La conception d'un système de détection d'intrusions basé sur un algorithme d'intelligence artificielle.
- La sélection de l'ensemble de données d'apprentissage.
- L'implémentation de la méthode de renforcement et de l'IDS.
- L'évaluation de notre méthode de renforcement sur notre IDS pour l'ensemble de données considéré.

1.4 Plan du mémoire

Ce mémoire sera composé de 4 chapitres, le premier étant cette introduction.

Le second chapitre étudiera les actuelles architectures d'Internet des objets au sein desquelles un algorithme de détection d'intrusion pourra être déployé. Il fera aussi un état de l'art des algorithmes d'intelligence artificielle et d'apprentissage machine pouvant être implémenté dans un système de détection d'intrusion. Nous parlerons ainsi des réseaux de neurones, et des Generative adversarial networks dans le contexte de cybersécurité, avant de discuter des différents ensemble d'apprentissage possibles pour notre projet.

La partie suivante consistera en la présentation du modèle de renforcement de système de détection par métaheuristiques et GAN. Il servira par la même à approfondir des concepts

évoqués dans le chapitre précédent. Enfin, le dernier chapitre concluera en récapitulant les résultats principaux des travaux.

CHAPITRE 2 REVUE DE LITTÉRATURE

Dans ce chapitre, nous effectuerons d'abord une revue des travaux relevant de l'Internet des objets, de leur architecture et des faiblesses qui en émerge. Ensuite, nous verrons les différentes stratégies actuelles basées sur l'Intelligence artificielle pour résoudre ces problèmes. Enfin, nous discuterons des différents ensembles d'apprentissage dans le domaine de la détection d'intrusion.

2.1 Internet of Things

Bien qu'il n'y ai pas une unique architecture de déploiement, l'internet des objets implique un paradigme différent de celui de la sécurité informatique habituelle. Nous allons ici étudier les différentes architectures de l'Internet des objets, et discuter des différents choix d'architectures de la détection d'intrusion.

Nous détaillerons deux types d'architectures : l'architecture centralisée et l'architecture distribuée. [8]

2.1.1 Architecture centralisée

Comme son nom l'indique, cette architecture consiste à considérer les objets connectés comme collecteurs de données uniquement : ces derniers ne sont pas capables d'effectuer des calculs ou de combiner des informations afin d'en déduire une action à effectuer. Des serveurs récoltent ces données, les traitent et renvoient des instructions aux objets avec lequel l'utilisateur interagit.

En quoi cette approche est-elle intéressante ?

Cette approche a déjà été utilisée par le passé, notamment pour les objets connectés déjà sur le marché, car elle simplifie la gestion des données pour les objets intelligents (qui ne font que recevoir leurs instructions et envoyer des données). Ce modèle implique aussi l'existence d'API universelle pour une gamme de produit, simplifiant l'interopérabilité.

2.1.2 Architecture distribuée

Ce paradigme implique que tous les objets, ou plus généralement toute entité faisant partie de l'architecture, est capable d'effectuer des calculs ou d'interpréter des informations. Les différents objets peuvent alors se communiquer des informations entre eux. On parlera alors de collaboration.

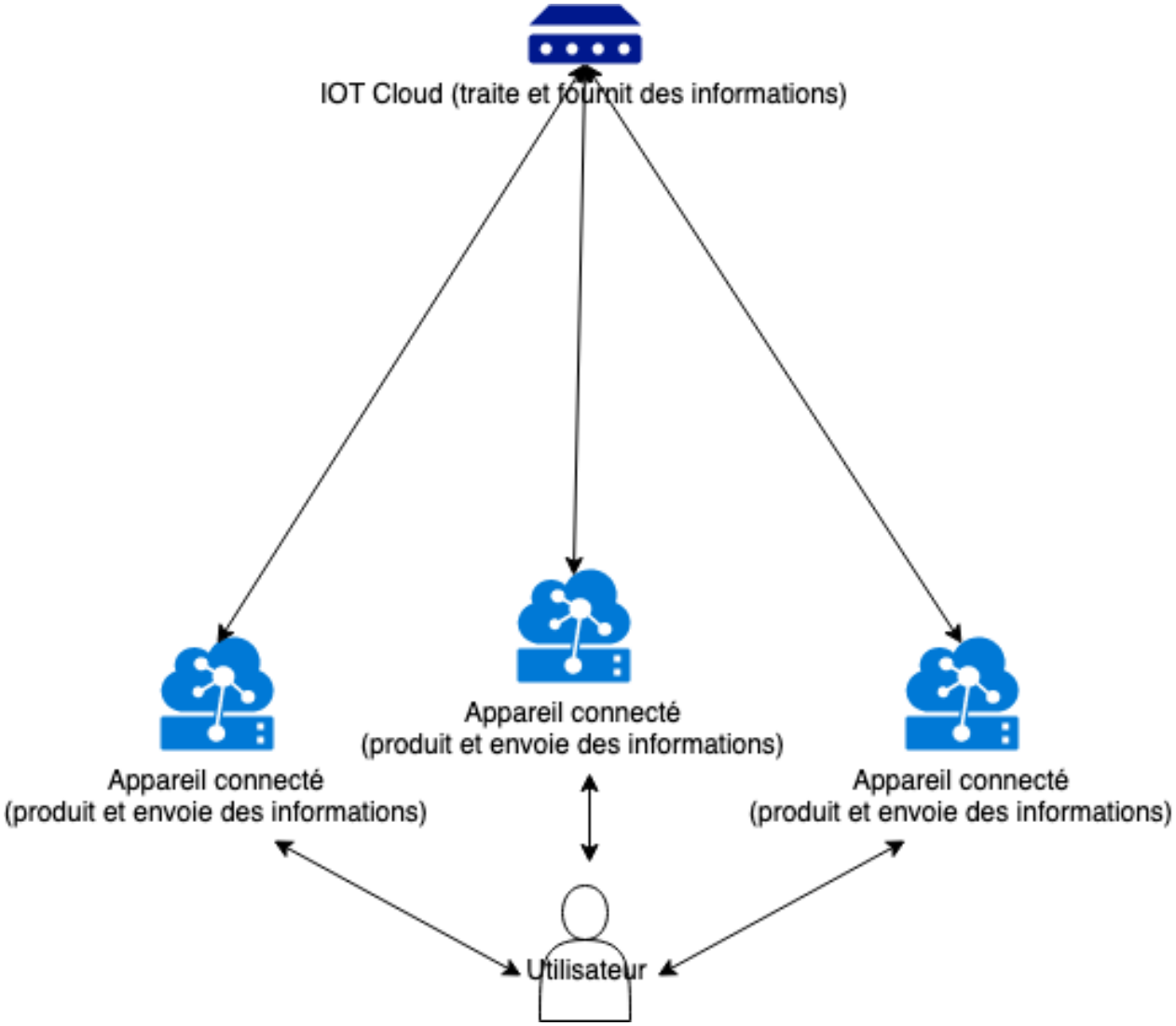

Figure 2.1 Illustration de l'architecture centralisée. Chaque appareil envoie les données récoltées aux nuage, qui les traite et en tire les instructions à effectuer qu'il renvoie aux appareils.

On peut citer comme exemple de collaboration, dans le cas d'une maison connectée, la communication entre le système de détection de cambriolage et les serrures connectées, enfermant alors le voleur jusqu'à l'arrivée des services de police.

En quoi cette approche est-elle intéressante ?

L'idée principale de cette approche est de doter les objets connectés "de bord", c'est à dire les objets avec lequel l'utilisateur interagit directement (réfrigérateur intelligent, lumière connectée, ...), de capacité d'aggrégation de données, voire de possibilités de calculs et d'intelligence. Ainsi, pour effectuer une action, un objet connecté ne se réfèrera pas à un serveur afin de

prendre une action, ne se cantonnant pas au rôle de fournisseur de données au serveur qui analyserait les informations fournies.

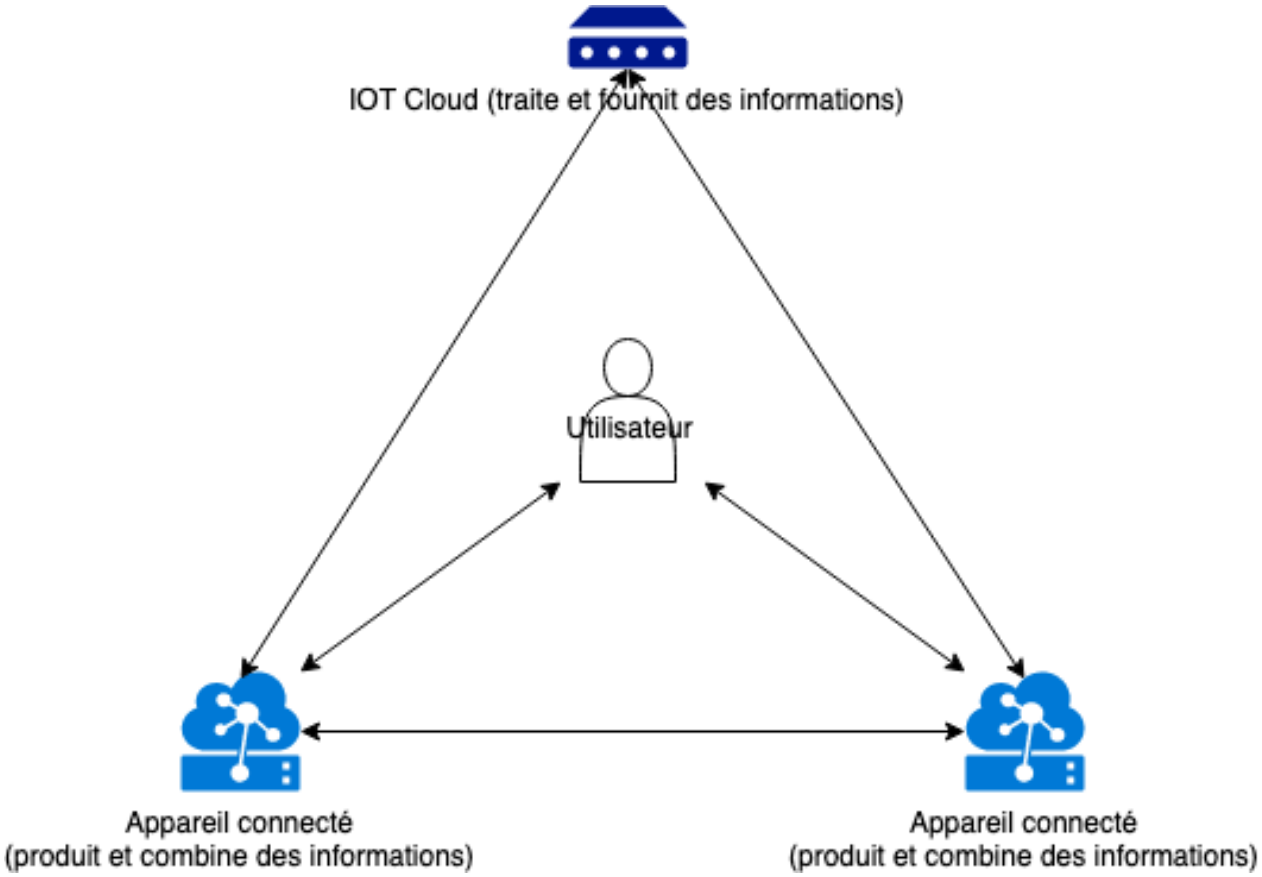

Figure 2.2 Illustration de l'architecture distribuée. Les appareils connectés peuvent être par exemple des verrous intelligents et des lumières connectées.

Elle permet plus de flexibilité du système en ne limitant plus les ressources disponibles à celles du centre de données, et implique aussi une répartition des calculs entre les différents objets, allégeant ainsi à terme significativement la charge de travail des data centers ainsi que la bande passante utilisée.

La sécurité devient alors un enjeu extrêmement important dans cette vision de l'Internet des objets : il s'agit ici de protéger la vie privée des utilisateurs, ainsi que de garantir la robustesse du parc des objets connectés. En outre, des objets connectés contrôlés par un attaquant malicieux peuvent mener à des conséquences dramatiques.

Pour autant, il convient de noter que les vecteurs principaux d'attaques restent les mêmes que pour un réseau standard :

— Déni de service (ou Déni de service distribué), rendant la machine inutilisable pendant un temps donné.

— Écoute du trafic, permettant à l'attaquant d'obtenir des informations sur le réseau, sur l'utilisateur, ...

— Prise de contrôle d'une machine (typiquement via des attaques de type Botnet, ou attaques d'infiltration), donnant le contrôle total de la machine à l'attaquant.

En particulier, détecter les attaques dans un réseau d'objet connecté permettrait aux appareils de protéger leurs services en se prémunissant des attaques de Déni de service et de prise de contrôle d'une machine.

Le déploiement d'un système de détection d'intrusion peut alors prendre deux formes [9], selon l'architecture d'IOT choisie :

— Le système de détection au sein de chaque appareil connecté : on parlera alors de système d'intrusion Onboard. Du fait des capacités de calcul limitées de chaque appareil, celui-ci suivra des règles relativement simples, par exemple issues d'une analyse statistique.

— La détection est reléguée à un serveur du centre de données, évitant ainsi de munir les appareils d'une force de calculs conséquente. Ici, on peut ainsi se permettre plus de liberté et de robustesse quant à l'algorithme de détection. On parlera alors de système Offboard, les processus les plus complexes étant relégués à un serveur de calcul.

2.2 Intelligence artificielle dans la détection d'intrusion

2.2.1 Réseaux de neurones

Principe de fonctionnement

Les réseaux de neurones sont une des technologies d'apprentissage machine les plus utilisées pour classifier des données.

Comme le nom d'apprentissage machine l'indique, un réseau de neurones a besoin d'apprendre sur un ensemble de données, qualifiées d'exemples. Ces exemples sont "étiquetés" au préalable : un expert a au préalable classifié ces données. Par exemple, dans le cas d'un algorithme de détection de chat sur une image, l'ensemble d'apprentissage se composera d'images, ainsi qu'une valeur indiquant si oui ou non il y a un chat sur l'image considérée. On parle ainsi d'apprentissage supervisé.

Pourquoi un réseau de neurones classifieur ne suffit-il pas ?

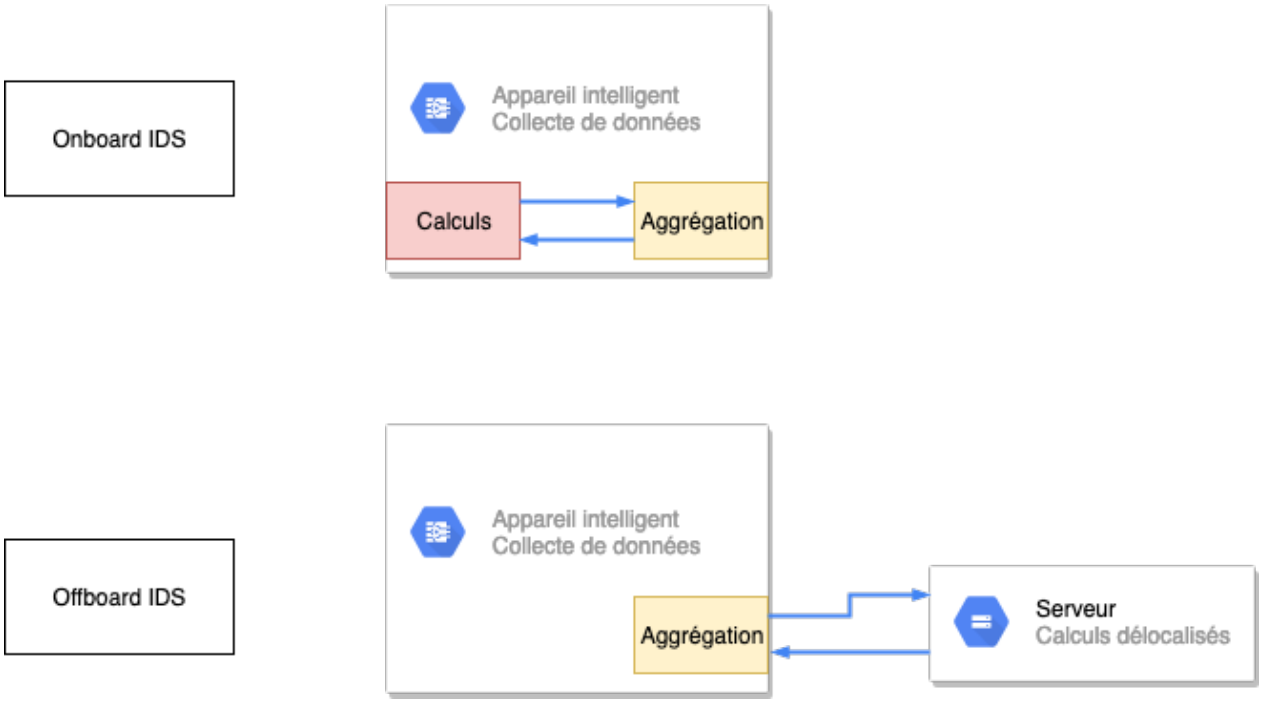

Figure 2.3 Illustration conceptuelle de l'architecture Onboard ou Offboard.

Comme présenté par [10], l'utilisation de l'intelligence artificielle dans un contexte de sécurité est radicalement différent de celle qui peut en être faite dans d'autres contextes :

— L'apprentissage supervisé suppose l'existence d'un ensemble d'apprentissage. Bien que des ensembles de données d'attaques accessibles pour tous existent, la question des performances de l'algorithme face à des attaques ne faisant pas partie de l'ensemble d'apprentissage se pose : si celles-ci sont trop différentes des comportements présentés dans l'ensemble de données, il est fort possible qu'une attaque soit catégorisée comme trafic bénin.

  En outre, du fait de la sensibilité des données concernant les cyberattaques, il n'existe que peu d'ensembles de données relatifs à des attaques. De fait, ces données peuvent révéler des informations sur la topologie du réseau d'une entreprise, des informations confidentielles ou des secrets d'entreprise.

— Un problème intrinsèque du Machine Learning est l'existence de "faux-positifs" et de "faux-négatifs". Dans le cas de la sécurité, il s'agirait respectivement d'une classification d'une activité bénigne en activité malicieuse et la classification d'un comportement anormal en activité normale, non dangereuse. Si la première est problématique, car elle demande une nouvelle analyse du trafic afin de déterminer si oui ou non le trafic est

effectivement dangereux (et implique donc une perte de temps), la seconde peut être dramatique : un comportement anormal peut faire partie d'une cyberattaque et donc causer la perte de données de l'entreprise ou d'autres conséquences tout aussi graves. En effet, contrairement à des utilisations plus classiques de l'intelligence artificielle (système de recommandations, vérificateurs d'orthographe, ou encore détecteur de pourriels), une erreur de classification a des répercussions bien plus coûteuses.

2.2.2 Random Forest

L'algorithme de forêt d'arbres décisionnels, ou Random Forest, est un algorithme de classification supervisé.

Basé sur les arbres de décision, l'algorithme consiste en la construction de plusieurs arbres de décision, sur plusieurs ensembles d'apprentissage légèrement différents.

Un arbre de décision cherche à classifier les données en fonction de leurs attributs. Les attributs se retrouvent sur les embranchements de l'arbre de décision, chaque branche correspondant à une valeur de l'attribut comme illustré sur la figure 2.4.

De fait, les arbres de décision ont été reconnus pour avoir une tendance au surapprentissage : ils sont alors capables de classifier très efficacement des exemples déjà vus, mais fournissent de moins bons résultats quand confrontés à de nouveaux exemples. L'algorithme de Random Forest permet de moyenner les différents arbres, et donc d'effacer cette tendance à l'overfitting.

2.2.3 Support Vector Machine

Une machine à support de vecteurs, ou Support Vector Machine (SVM) est un algorithme de classification, d'apprentissage supervisé. Tout comme le réseau de neurones, il s'agit de classifier des instances en deux classes.

La SVM travaille avec des vecteurs d'entrées de dimension potentiellement élevée. Le but de la SVM est alors de trouver le moyen optimal de diviser l'espace d'entrée en deux pour la classification. La limite entre ces deux zones est un hyperplan : les données se trouvant d'un côté de l'hyperplan appartiennent alors à une classe, et les autres à l'autre classe.

Ceci est illustré dans la figure 2.5.

Une fois l'hyperplan trouvé après la phase d'apprentissage, les nouvelles données sont classifiées selon l'endroit où elles se trouvent dans l'espace d'entrée, *ie.* d'un côté ou de l'autre de l'hyperplan.

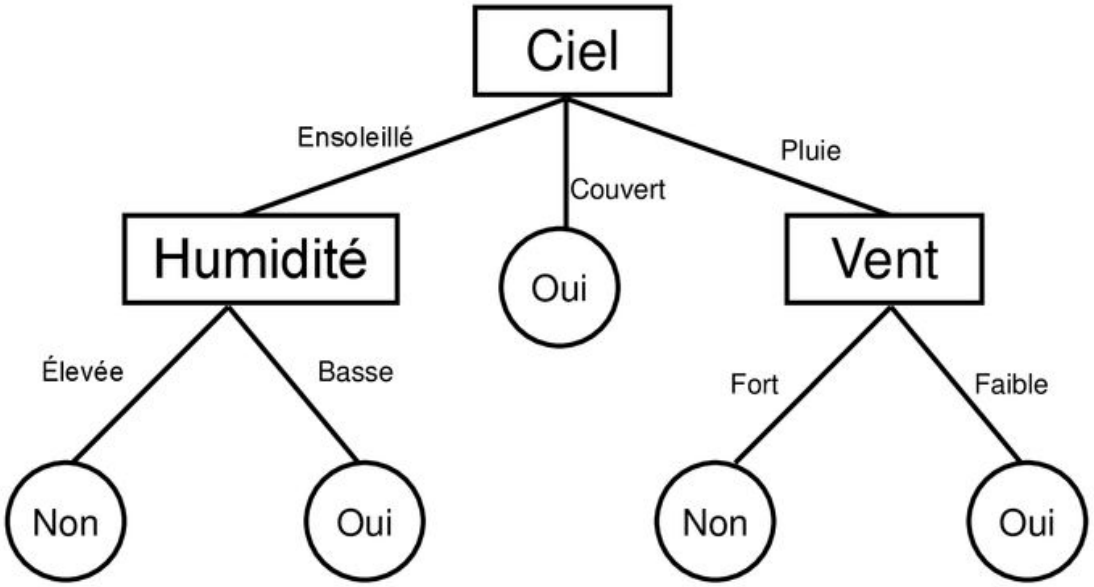

Figure 2.4 Arbre de décision pour la question : Faut-il sortir dehors ?

2.2.4 Naive Bayes

Un classifieur bayésien naïf est un modèle probabiliste, dont les résultats sont basés sur l'hypothèse de l'indépendance deux à deux des attributs des données. Cette hypothèse est très forte, d'où le qualificatif "Naïf" : ce modèle est néanmoins un standard de l'apprentissage supervisé et produit souvent des résultats corrects.

En notant les attributs des données A_1..., A_n, on peut appliquer la formule de Bayes à la probabilité conditionnelle d'appartenir à la classe C :

$p(C|A_1, ..., A_n) = \frac{p(C)p(A_1,...,A_n|C)}{p(A_1,...,A_n)}$

Dès lors, diposant d'une entrée, $A_1, ..., A_n$ sont connus. De plus, le dénominateur étant une constante, il ne nous intéresse pas.

On a alors, pour le numérateur :

$p(C)p(A_1, ..., A_n|C) = p(C)p(A_1|C)p(A_2|A_1, C)...p(A_n|A_1, ...A_{n-1}, C)$

Or, l'hypothèse d'indépendance des attributs nous amène donc à :

$p(C)p(A_1, ..., A_n|C) = p(C) \prod i = 1^n p(A_i|C)$

D'où finalement :

$p(C|A_1, ..., A_n) = \frac{p(C) \prod i=1^n p(A_i|C)}{M}$ Avec M une constante.

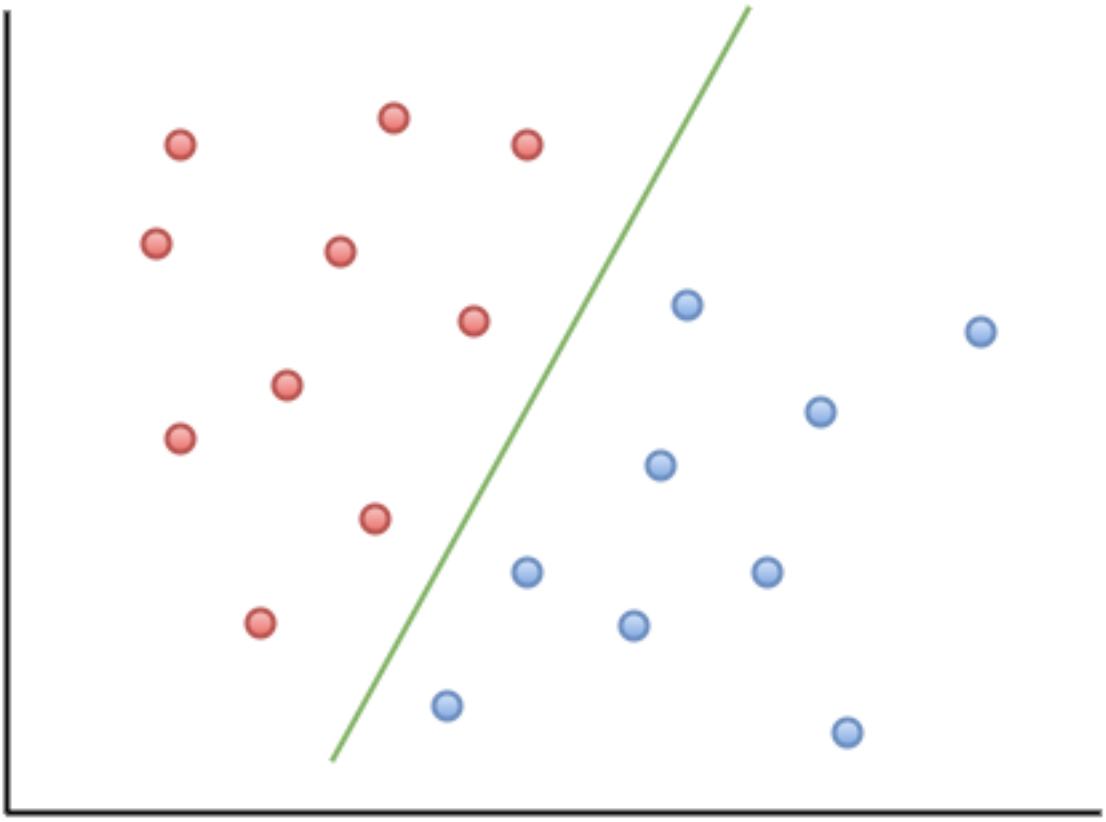

Figure 2.5 Application de la SVM pour séparer deux classes : bleue et rouge.

Il est ainsi possible de classifier une instance du fait de ses attributs. Naive Bayes a déjà été utilisé par le passé dans la classification d'attaques : Messieurs Panda et Patra ont ainsi développé un système de détection d'intrusion basé sur un classifieur Naive Bayes, conduisant à une précision de 90% à 96% selon le type d'attaque étudié. [11]

2.2.5 Generative Adversarial Networks (GAN)

Principe de fonctionnement

Les Generative Adversarial Networks sont une classe d'algorithmes d'Intelligence artificielle, d'apprentissage non supervisé [12].

Ils consistent en deux réseaux de neurones : un réseau générateur, et un réseau discriminateur. Le réseau discriminateur consiste en un classificateur classique, comme les réseaux de neurones vus dans la partie précédente. Le réseau générateur, désigné par G, va quant à lui apprendre les caractéristiques de chaque classe dans le but de les reproduire : son but est donc de générer de nouvelles instances.

Le but du discriminateur D est d'identifier si l'instance qui lui est présentée est réelle (càd provient d'un véritable ensemble de données) ou si elle a été générée par le réseau générateur. Le générateur G, quant à lui, a pour objectif de générer des instances que le discriminateur ne saura pas distinguer du véritable ensemble de données.

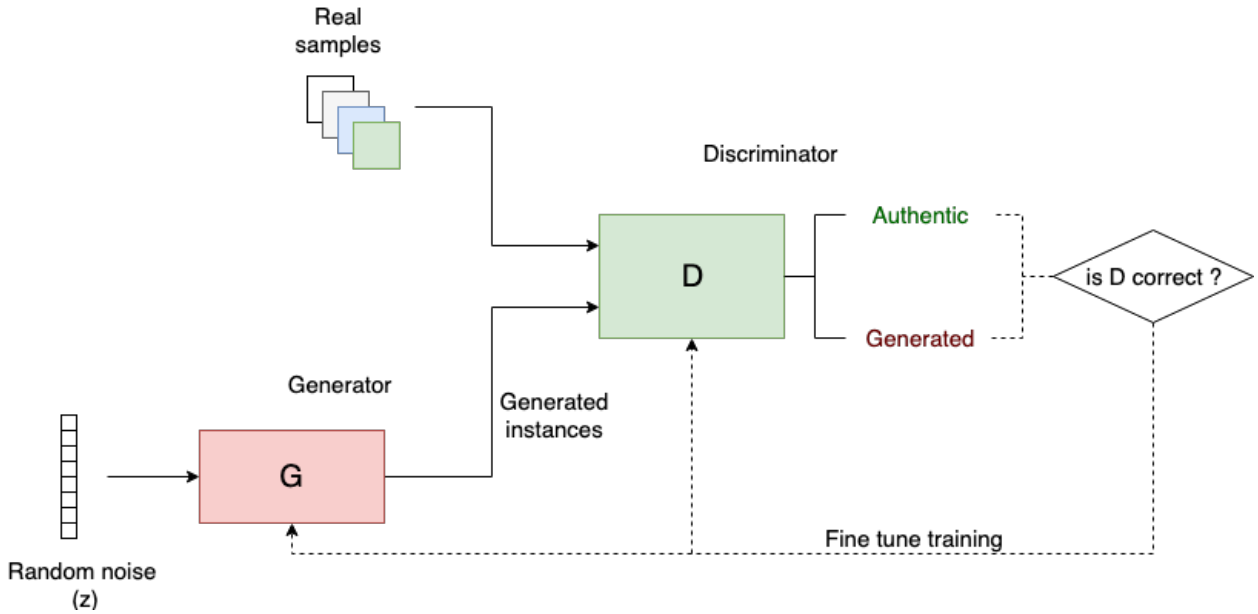

Figure 2.6 Illustration du fonctionnement des GAN. Source : MS&E 238 Blog

En notant x une instance des données (de l'ensemble d'apprentissage, ou générée), on note

$D(x)$ la probabilité estimée par le discriminateur que x soit une instance générée, *ie* que cette instance n'appartienne pas à l'ensemble d'apprentissage et soit créée par le générateur G.

De même, nous notons z le bruit aléatoire, et $G(z)$ l'instance générée par le générateur à partir de ce bruit. Avec ces notations, nous entraînons G de façon à minimiser $D(G(z)))$.

Les Generative Adversarial Networks se sont démarqués en 2014 de par leur capacité à générer des images particulièrement réalistes. Dans [12], les chercheurs ont illustré l'efficacité de leur méthodes avec (entre autre) le jeu de données MNIST. Les instances générées par le réseau générateur sont présentées ci-après.

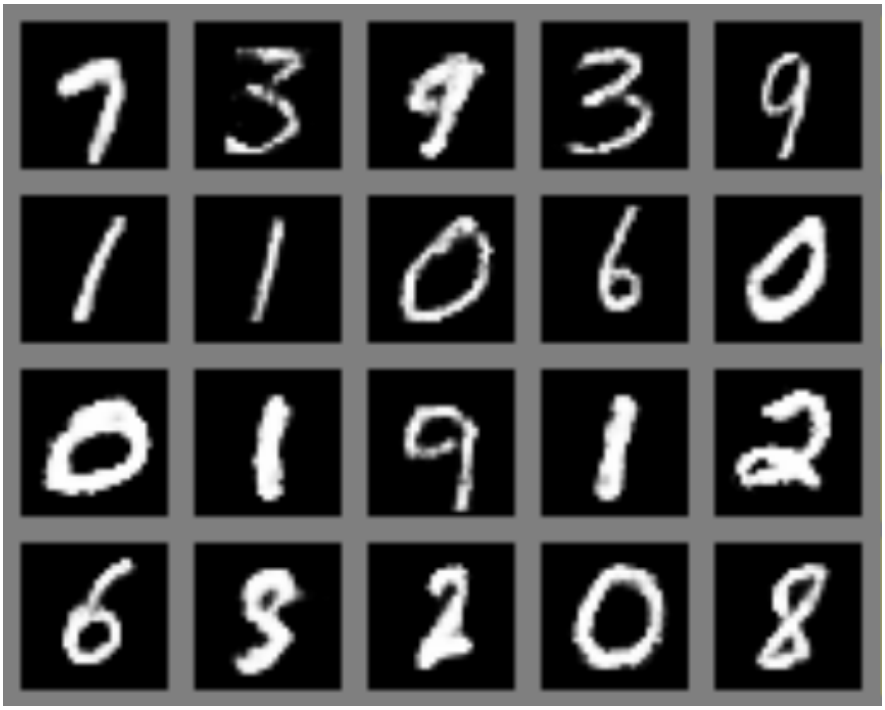

Figure 2.7 Instances du MNIST générées par le réseau générateur.

Les GANs ont aussi été utilisés afin de tromper des classifieurs pré-entraînés. Par exemple, dans le cas de reconnaissance d'image, un réseau adversariel a été entraîné pour effectuer de petites modifications invisibles à l'oeil nues à une image de façon à tromper le détecteur. Ainsi, dans le cas de l'étude de Athalye et. al. [13], une image de tortue modifiée par un réseau adversariel s'est retrouvée classifiée en fusil par l'algorithme de reconnaissance d'images.

En quoi cette approche est-elle intéressante ?

Du fait de ses résultats impressionnants dans le domaine de la photographie [14], de la vidéo,

et du traitement audio, l'application des GAN s'est rapidement vue transposée au milieu de la sécurité informatique. En effet, contrairement aux limitations d'un réseau de neurones évoquées dans la partie précédente, les GAN ont l'avantage de pouvoir générer de nouvelles attaques.

Ainsi, [6] a montré que des exemples d'attaques générés par le réseau de neurones pouvait, dans plus de 98% des cas, échapper à la classification de la part des algorithmes d'intelligence artificielle de classification standards (Support Vector Machine, MultiLayer Perceptron, ...), et [7] a montré que les GAN étaient capables de générer des programmes malveillants non détectés par un algorithme d'intelligence artificielle en "boîte noire" (c'est à dire sur lequel les chercheurs n'avaient pas de contrôle).

L'avantage de l'approche adversarielle est qu'elle permet d'entrainer simultanément le générateur, mais aussi le détecteur. Ainsi, si un algorithme de classification entrainé sur un ensemble de données ne sera que très rarement en capacité de distinguer une véritable instance de l'ensemble de données d'un exemple généré par le générateur du GAN, il est possible d'entraîner le réseau discriminateur D à distinguer ces exemples forgés dans plus de 90% des cas, voire dans 100% des cas [7].

Les chercheurs Aidin Ferdowsi et Walid Saad ont proposé une méthode permettant de détecter de telles attaques générées dans un environnement distribué. [1]

Chaque appareil connecté est dispose d'un discriminateur, entraîné à l'aide d'attaques générées par un Générateur central.

Comme chaque objet connecté dispose de données privées, qui sont des données potentiellement sensibles, à chaque étape de l'entraînement, les poids des discriminateurs sont échangés selon un graphe cyclique. Ainsi, une fois que les poids retournent au discriminateur initial, ils ont été entraîné sur l'ensemble des données disponibles par tous les objets connectés du réseau.

Le fonctionnement est illustré sur la figure 2.8.

Néanmoins, cette méthode n'évoque pas la génération d'attaque de manière itérative.

2.2.6 Métaheuristiques

Principe de fonctionnement

Les métaheuristiques sont des techniques utilisées afin de trouver des solutions à des problèmes souvent très complexes, par exemple des problèmes NP-difficiles, pour lesquels l'optimisation standard serait trop coûteuse (car beaucoup de calculs, et donc un temps nécessaire à trouver une solution très long). Les métaheuristiques ne garantissent pas de trouver la

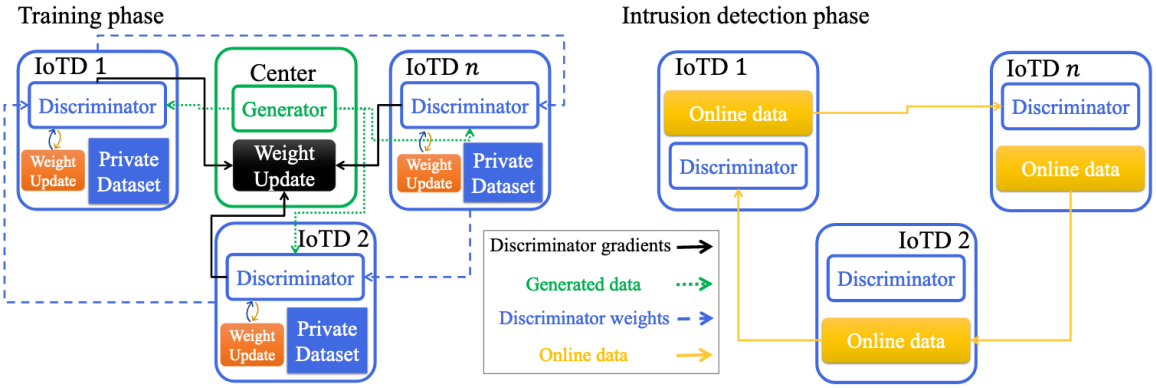

Figure 2.8 Fonctionnement de l'architecture distribuée de Generative Adversarial Networks détaillée dans [1]

solution optimale du problème, mais permettent de trouver "une bonne solution suffisante", sensée s'approcher d'une solution optimale. L'approche de l'algorithme peut être basée sur une seule solution, ou sur une population de solutions.

La première approche peut-être illustrée par l'algorithme d'optimisation locale.

Une solution est initialisée aléatoirement. L'algorithme explore alors le voisinage de cette solution en appliquant des changements à la solution actuelle. L'exploration de l'espace des solutions dépend alors de la notion de voisinage, qui doit être définie par l'expérimentateur. La recherche de la solution continue jusqu'à ce qu'une solution répondant au critère d'arrêt de l'expérimentateur soit trouvée, ou qu'une limite de temps ou d'itération soit dépassée.

La recherche locale est potentiellement très efficace dans le cas d'une fonction disposant un seul optimum. Malheureusement, dans le cas de fonctions plus complexes, il peut converger vers des optima locaux, et donc ne pas apporter la meilleure solution au problème.

L'algorithme génétique est l'algorithme de métaheuristiques basé sur les populations le plus classiquement utilisé de par ses bons résultats.

Cet algorithme est inspiré du processus de sélection naturelle au sein d'une population. L'algorithme est initialisé avec une population de solutions, chaque solution étant générée aléatoirement comme dans le cas précédent. La population évolue de génération en génération : on sélectionne les meilleures solutions de la population (sélection), recombine ces solutions en d'autres (cross-over), avant d'appliquer des mutations aléatoires à cette nouvelle population. Ceci permet d'avoir une population diverse, permettant d'explorer certaines caractéristiques qui n'étaient pas apparus jusqu'alors. On explore alors plus de solutions de

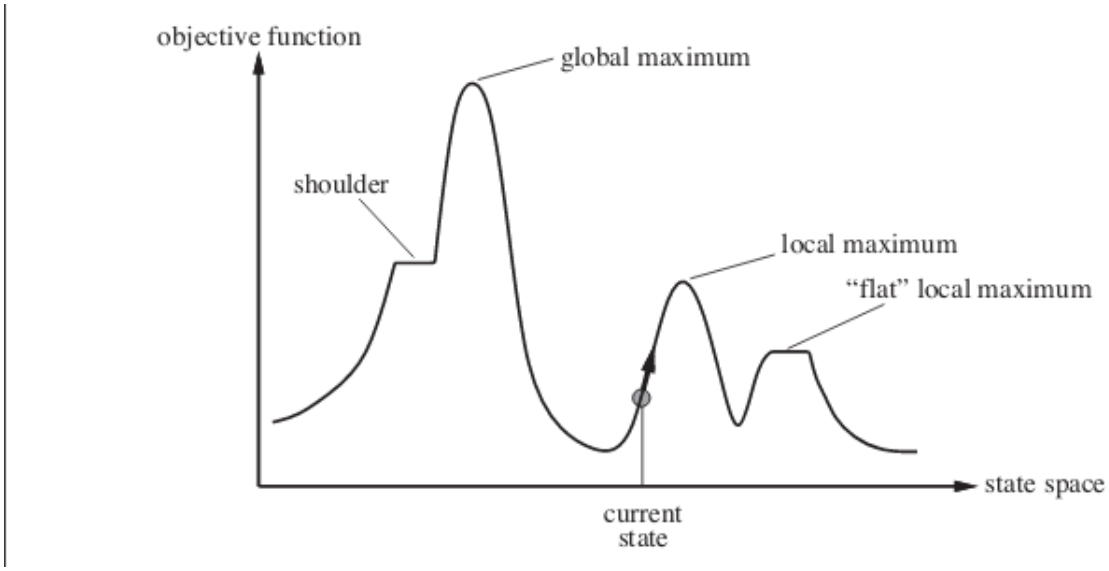

Figure 2.9 Schéma de l'algorithme de Hill Climbing, algorithme de recherche locale. Comme on peut le voir, cette méthode risque de converger vers un optimum local, sans trouver le maximum global de la fonction à optimiser.

l'espace des solutions. L'algorithme s'arrête quand la population répond au critère d'arrêt fixé par l'expérimentateur, ou après un certain nombre de générations.

Cette méthode est utilisée dans le cas où la fonction à maximiser dispose de plusieurs optima : l'approche par population permet d'explorer plus d'une seule zone de l'espace solution à la fois. Néanmoins, cet algorithme peut être assez long à converger.

Il est possible de combiner les deux approches, à l'aide d'un algorithme dit "hybride". Le processus est similaire à celui de l'algorithme génétique, mais chaque membre de la population de solutions est optimisée par l'algorithme de recherche locale avant le processus de sélection.

Ainsi, chaque individu de la population passant par l'algorithme de recherche locale, et donc voyant son score augmenter, cet algorithme dispose plus rapidement de résultats satisfaisants. Cet algorithme est utilisé pour résoudre des problèmes complexes, convergeant plus rapidement que l'algorithme génétique standard, menant à de meilleurs résultats globaux.

En quoi cette approche est-elle intéressante ?

Les métaheuristiques ont été utilisées dans plusieurs domaines différents de la cybersécurité.

De fait, l'approche basée sur des populations de solutions permet d'explorer plus facilement l'espace de solution qu'une approche d'optimisation : ainsi, plusieurs chercheurs ont appliqué

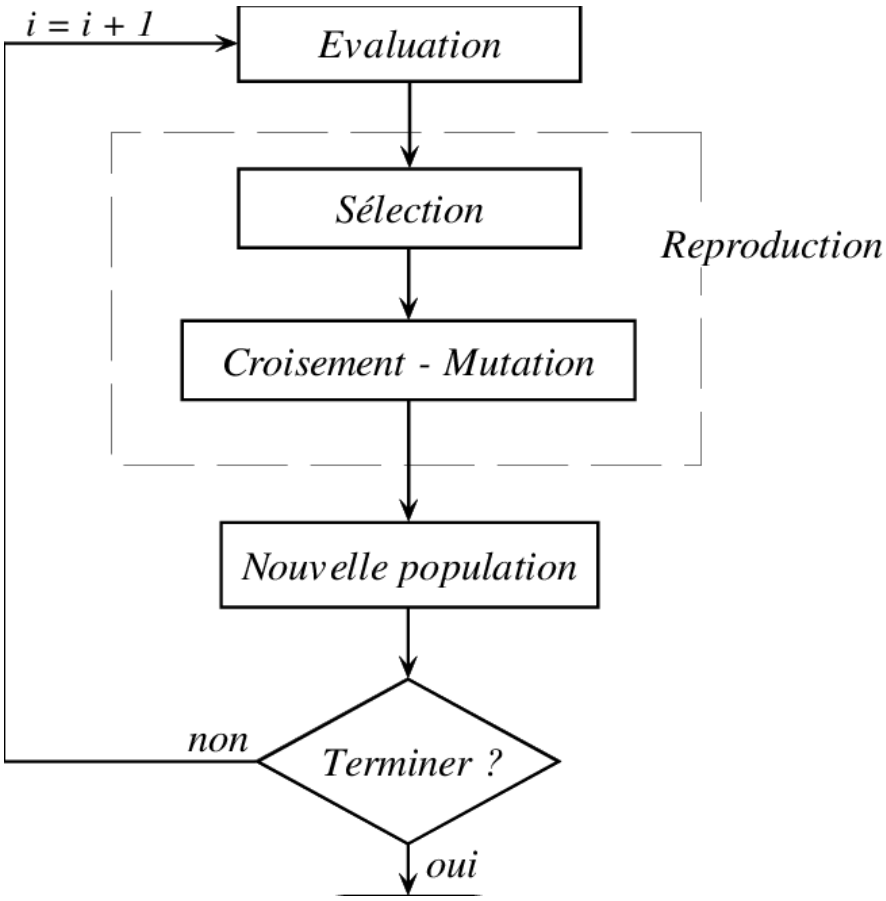

Figure 2.10 Schéma du déroulement de l'algorithme génétique.

des algorithmes génétiques à leur domaines.

Sadeeq Jan et. al ont utilisé les métaheuristiques pour générer des attaques d'injection XML pour des applications Web. [15]

Une attaque XML consiste à injecter du code malicieux dans une application dont les services utilisent XML. Le but de ces attaques consiste en la compromission du système, ou de celui traitant le message XML, pouvant mener à une élévation de privilège, ou au crash du système dans les cas les plus extrêmes.

Ainsi, dans un formulaire XML simple, incluant nom d'utilisateur, mot de passe et courriel, si l'on considère les entrées malicieuses, suivantes :

$username = Tom$

$password = Un6Rkb!e < /password >< --!$

$email = -- >< userid > 0 < /userid >< mail > admin@uni.lu$

On obtient le message XML suivant transmis au serveur :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.
    xmlsoap.org/soap/envelope/">
    <soapenv:Header/>
    <soapenv:Body>
        <user>
            <username>Tom</username>
            <password>Un6Rkb!e</password>
            <!--
                </password>
                <userid>500</userid>
                <mail>
            -->
            <userid>0</userid>
            <mail>admin@uni.lu</mail>
        </user>
    </soapenv:Body>
</soapenv:Envelope>
```

L'utilisateur malicieux a dès lors remplacé son identifiant utilisateur par 0, 0 étant (la plupart du temps) l'userid de l'administrateur. Ceci est une illustration d'une possible attaque XML menant à une élévation de privilèges : dans ce cas précis, un attaquant a réussi à acquérir les privilèges de l'administrateur en injectant du code XML dans le formulaire d'identification.

Les chercheurs ont développé un outil permettant d'automatiser la génération de tels messages. Cet outil est basé sur les métaheuristiques, et explore plusieurs algorithmes afin de générer des attaques (l'algorithme génétique standard, ainsi que l'algorithme de Hill Climbing).

Ainsi l'algorithme génétique a généré des attaques d'injection XML avec succès dans plus de 95% des cas, tandis que l'algorithme de Hill Climbing a réussi à générer des attaques atteignant le serveur dans 79% des cas.

Néanmoins, la perspective du papier n'est pas de développer des outils pour détecter de telles attaques : l'outil de génération d'attaques par métaheuristiques est uniquement un outil de test, afin de vérifier s'il existe des injection XML passant au travers les algorithmes sensés prévenir de telles injections (on parle de *sanitization*, en anglais). Si de telles injections XML existent, c'est aux développeurs de trouver d'autres méthodes de *sanitization* plus efficaces pour éviter des attaques contre leurs sites.

Ces travaux constituent néanmoins la preuve que les métaheuristiques peuvent permettre la

création de nouvelles attaques, ici dans le cas d'XML, et que l'algorithme génétique peut s'avérer très efficace pour cela.

De même, Cordy et. al ont utilisé l'algorithme génétique pour générer des attaques sur le long terme contre des algorithmes de détection d'attaques de Denial of Service basés sur le clustering. [16]

La méthode de détection utilisé par les chercheurs était l'algorithme D-Stream. Les nouvelles données étant considérées plus importantes que les anciennes par l'algorithme, afin de s'adapter à de nouveaux comportements émergents, chaque nouvelle donnée est associée d'un poids qui diminue au cours du temps.

Cette méthode a l'avantage de s'adapter facilement et rapidement à de nouveaux comportements, mais il est possible pour l'attaquant de tirer partie de la spécificité de cet algorithme pour mener de nouvelles attaques.

Les attaques par entraînement sont des attaques très sophistiquées, changeant progressivement le comportement du réseau. Ces changements se faisant petit à petit, ils ne sont pas perçus comme dangereux par l'algorithme de clustering, mais permettent, sur le long terme de faire catégoriser comme bénin des comportements dangereux.

Sur la figure 2.11, on constate ainsi qu'une attaque par entraînement augmente progressivement le flux de données sur le réseau SCADA étudié. Au bout de quelques étapes, une attaque de type Déni de Service (DOS) se caractérisant par un important flux de données ne serait plus détectée.

Les chercheurs ont utilisé un algorithme génétique afin de générer des attaques par entraînement, ainsi que pour améliorer la défense du système de détection d'intrusion. Ainsi, les stratégies d'attaques et de défense co-évoluent : en générant itérativement des stratégies de défense, basée sur des attaques ayant fonctionné précédemment, cette méthode permet d'augmenter la résistance du système de détection.

Ainsi, si le système n'a détecté aucune des 50 attaques à la première itération, le système de détection s'est renforcé de par la confrontation à de nouvelles attaques : au terme de la sixième et septième itération, l'algorithme détecte 49 des 50 attaques générées (soit 98%).

De ces travaux, nous pouvons conclure que la confrontation à de nouvelles attaques permet de renforcer préemptivement le système : générer des attaques itérativement permet d'anticiper le plus de comportements possibles, et de confronter de manière préventive le système à de nouvelles attaques.

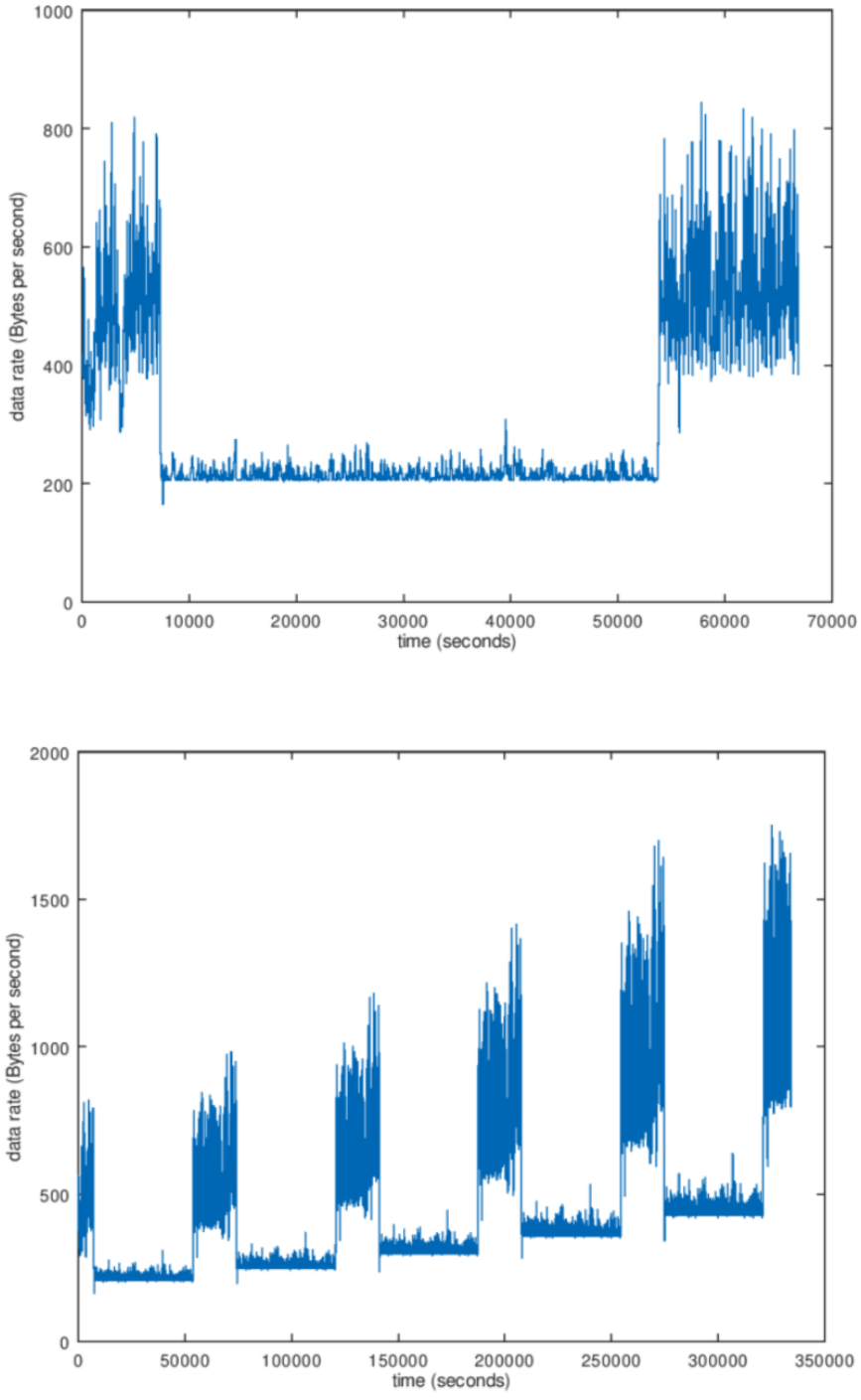

Figure 2.11 À gauche, le trafic normal, en bytes par seconde, du réseau SCADA étudié par Cordy et. al. À droite, le trafic d'une attaque "par entraînement" : le volume du trafic augmente au fur et à mesure, et permettra à l'attaquant de mener une attaque DOS non détectée par le système.

2.3 L'ensemble d'apprentissage

La sélection d'un ensemble d'apprentissage est un facteur clé de notre étude. De fait, l'apprentissage machine suppose l'utilisation d'un tel ensemble de données. C'est de cet ensemble d'apprentissage que l'algorithme va inférer son modèle (ie sa fonction de prédiction) qui, dans notre cas, déterminera à partir de données réseau si le trafic est malicieux ou non. Les données dans le domaine de la détection d'intrusion sont rares, du fait de leur caractère sensible pour les entreprises.

Dans notre cas, l'ensemble d'apprentissage des attaques informatiques n'est pas constitué de l'intégralité du trafic, mais bel et bien de caractéristiques de l'état de celui-ci. Nous étudierons donc ici un dispositif de détection d'intrusions utilisant des métriques du réseau pour caractériser la nature du trafic (bénin ou attaque). On citera par exemple les ports de destination des paquets, la taille des paquets, le protocole, et d'autres.

2.3.1 L'ensemble NSL-KDD

L'ensemble de données NSL-KDD est l'ensemble de données le plus important dans le domaine de la détection d'intrusion. [17] Cet ensemble de données est une amélioration du KDD-99, ensemble de donnée issu de l'initiative de la DARPA en 1998 afin de permettre l'évaluation de systèmes de détection d'intrusion. Les données sont générées à partir d'une simulation d'un réseau, s'abstrayant ainsi du caractère sensible que peuvent représenter les données réseau d'une entreprise.

Trois machines contenant des données sensibles, chacune disposant de systèmes d'exploitation et de services accessibles différents, constitue le réseau à protéger. Trois autres machines sont utilisées en tant qu' "attaquantes", en prenant de fausses adresses IP. L'ensemble de données est l'intégralité du trafic du réseau enregistré durant 7 semaines. Le trafic peut être catégorisé comme bénin, ou comme attaque, catégorie elle même divisée en 4 sous-catégories d'attaques :

— Denial of Service (ou DOS)

  Le but de cette attaque est de rendre un service indisponible, en inondant le réseau afin d'empêcher son fonctionnement. Elle peut ainsi par exemple bloquer les serveurs web, courriel ou les serveurs de fichiers d'une entreprise.

— Probe

  L'attaquant essaye d'acquérir des informations sur sa cible en tentant de demander différents services. Ceci peut par exemple être mené à l'aide de l'outil Nmap.

— User to Root (ou U2R)

L'attaque consiste à une élévation de privilèges : un utilisateur d'une machine locale tente d'acquérir les droits d'un "super-utilisateur".

— Remote to Local (R2L)

Le but est de gagner un accès à une machine victime depuis une machine extérieure.

Pour classifier la nature du trafic, l'ensemble d'apprentissage exhibe 41 caractéristiques représentant l'état actuel du réseau. Parmi ces caractéristiques, on peut citer le type de protocole utilisé, le nombre d'identification ratées, ou encore le nombre de connections vers la même destination.

Tableau 2.1 Quelques une des 41 caractéristiques exhibées par l'ensemble d'apprentissage KDD.

Feature	Description	Type
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
urgent	nb of urgent packets	continuous
num_failed_logins	number of failed login attempts	continuous
num_root	number of “root” accesses	continuous
is_hot_login	1 if the login belongs to the “hot” list ; 0 otherwise	discrete
serror_rate	% of connections that have “SYN” errors	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

L'ensemble initial, KDD-99, souffrait de plusieurs défauts (données doublons, très peu de données les attaques U2R et R2L, ...), et a donc été amélioré plusieurs depuis sa publication en 1999, jusqu'à sa dernière version, NSL-KDD en 2009.

Néanmoins, plusieurs reproches persistent quant à cette amélioration de l'ensemble d'apprentissage. Parmi ceux-ci, on retrouve le manque de données pour les attaques R2L et U2R. Ceci implique un taux de détection de ces attaques particulièrement faible à l'aide des algorithmes d'Intelligence artificielle par rapport aux autres types d'attaques, largement mieux représentées dans le dataset. [18]

Cet ensemble de données date d'il y a désormais 10 ans, et bien qu'il soit considéré comme l'ensemble classique dans le domaine de la détection d'intrusion, sa pertinence aujourd'hui est questionnable. Dans les faits, cet ensemble de données souffre de manque de diversité de trafic et de volume, ne couvrant ainsi pas l'ensemble des cyber-attaques. [19]

2.3.2 D'autres ensembles de données

Bien qu'il existe peu d'ensembles de données standards en détection d'intrusions, il existe d'autres données plus récentes. Plus particulièrement, le Canadian Institute for Cybersecurity a développé en 2018 un nouvel ensemble de données pour la détection d'intrusion [20], incluant diverses attaques, dont :

— Des attaques de bruteforce

  L'attaque consiste en l'utilisation exhaustive de tous les mots de passes (ou URL, ...) jusqu'à trouver le bon.

— Heartbleed

  Il s'agit d'une attaque exploitant une faille d'OpenSSL, utilisé par le protocole TLS.

— DOS et Distributed Denial of Service (DDOS)

— Botnet

  Il s'agit d'un virus se diffusant entre plusieurs objets connectés. Les botnets servent alors à effectuer des tâches diverses, que ce soient mener des attaques DDOS, envoyer des pourriels ou voler des informations des utilisateurs. L'exemple le plus connu de Botnet est Mirai, qui a infecté plusieurs milliers d'objets connectés.

— Des attaques Web

  Des exemples communs de ces attaques sont les attaques de SQL-injection pour les bases de données, ou de Cross Site Scripting (XSS).

— Des attaques d'infiltration

  L'infiltration est souvent effectuée par le biais d'un logiciel disposant d'une faille (comme Adobe Acrobat Reader). Elle permet à un utilisateur distant d'avoir accès à une machine du réseau.

Ces différentes attaques sont représentatives des attaques actuelles, et le fait de disposer de 11 catégories plutôt que 4 pour NSL-KDD permet de classifier de manière beaucoup plus précise le type d'attaque subit par les systèmes informatiques ; et donc de potentiellement mieux s'adapter à la situation en cours.

80 caractéristiques du réseau (port de destination, débit entrant par seconde, écart type du temps de connexion, ...) ont été extraites par les chercheurs du Canadian Institute for Cybersecurity, permettant alors de classifier le trafic dans les différentes catégories d'attaques ou en trafic bénin.

Cet ensemble de données est particulièrement intéressant car il répond aux différents critères permettant de juger d'un ensemble de données fiable pour la détection d'intrusion : tous les protocoles communs (HTTP, HTTPS, FTP, SSH, SMTP ...) sont représentés, et les attaques sont diversifiées, l'intégralité du trafic et des métadonnées générées sont accessibles. [21]

Les caractéristiques exhibées par cet ensemble de données sont des métriques plus précises sur le flux de données entrant et sortant que celles que peut fournir NSL-KDD : on peut trouver par exemple le nombre de paquets dans la direction source vers destinataire par seconde, la déviation standard de la taille du paquets sur la durée du flux de donnée, ou encore le nombre de paquets portant le tag ACK : autant de caractéristiques que l'ensemble de données de 2009 ne comporte pas.

En outre, il n'est pas possible d'extraire ces features du dataset NSL-KDD : de fait, nous ne disposons pas du trafic étudié dans le cas de cet ensemble de données, mais uniquement des flux internet déjà classifiés selon les métriques sélectionnées. Il apparait donc difficile de comparer les résultats obtenus sur ces deux ensembles de données, les métriques n'étant pas les mêmes.

CHAPITRE 3 ARTICLE 1 : SIGMA : STRENGTHENING IDS WITH GAN AND METAHEURISTICS ATTACKS

Simon Msika, Alejandro Quintero, Foutse Khomh (2019) submitted to the IEEE Transactions on Information Forensics and Security (18 Dec. 2019).

An Intrusion Detection System (IDS) is a key cybersecurity tool for network administrators as it identifies malicious traffic and cyberattacks. With the recent successes of machine learning techniques such as deep learning, more and more IDS are now using machine learning algorithms to detect attacks faster. However, these systems lack robustness when facing previously unseen types of attacks. With the increasing number of new attacks, especially against Internet of Things devices, having a robust IDS able to spot unusual and new attacks becomes necessary.

This work explores the possibility of leveraging generative adversarial models to improve the robustness of machine learning based IDS. More specifically, we propose a new method named SIGMA, that leverages adversarial examples to strengthen IDS against new types of attacks. Using Generative Adversarial Networks (GAN) and metaheuristics, SIGMA generates adversarial examples, iteratively, and uses it to retrain a machine learning-based IDS, until a convergence of the detection rate (i.e. until the detection system is not improving anymore). A round of improvement consists of a generative phase, in which we use GANs and metaheuristics to generate instances ; an evaluation phase in which we calculate the detection rate of those newly generated attacks ; and a training phase, in which we train the IDS with those attacks.

We have evaluated the SIGMA method for four standard machine learning classification algorithms acting as IDS, with a combination of GAN and a hybrid local-search and genetic algorithm, to generate new datasets of attacks. Our results show that SIGMA can successfully generate adversarial attacks against different machine learning based IDS. Also, using SIGMA, we can improve the performance of an IDS to up to 100% after as little as two rounds of improvement.

3.1 Introduction

In the last few years, the emergence of the Internet of Things (IoT) has led to new cybersecurity challenges. As connected objects now interact with the real world, privacy and security threats mitigation increasingly become a major issue [8]. With these new entities come the need to protect them from cyberattacks and similar intrusions. For instance, in 2016, the

Mirai botnet [3] infected more than 600,000 Internet of Things devices from which were conducted massive Distributed Denial of Service (DDOS) attacks against several network companies all over the world.

Intrusion Detection Systems (IDS) are an essential tool for IoT system administrators : detecting a cyberattack is the first step to guarantee the privacy and security of users.

But IoT also means a huge increase of internet traffic to analyze, and therefore the need to develop efficient, fast and robust algorithms to detect cyberattacks in this sensitive environment. Recently, machine learning models have shown astonishing performances in retrieving patterns from large volumes of data, in a very short amount of time. This success lead to their wide adoption in IDS [22]. However, as recent works on adversarial models have shown [12], machine learning algorithms, in particular deep learning tend to be fragile to adversarial examples. Making IDS which are based on such models vulnerable to adversarial attacks.

Using Generative Adversarial Networks (GAN) [12] an attacker can generate adversarial requests (i.e., attacks) that share the characteristics of requests that are considered to be genuine by the IDS. Although these GANs represent formidable weapons for attackers, as they can deceive most IDS into classifying attacks as benign traffic, they also provide an opportunity to preemptively strengthen intrusion detection systems against new attacks. By exposing an IDS to generated attacks as a preventive measure, we can prepare for new malicious behaviors.

In this paper, we propose a method to strengthen IDS against generated adversarial attacks, called **SIGMA**, which stands for **S**trengthening **I**DS with **G**AN and **M**etaheuristics **A**ttacks. The method consists in the iterative generation of attack datasets using adversarial learning and metaheuristics algorithms. The generated datasets are then used to retrain IDSs ; i.e., teaching them to cope with patterns of attacks similar to those contained in our generated datasets. We repeat the retraining process until the detection rate of the IDS on generated attacks converges, meaning the detection rate is not improving anymore. We stop the algorithm after 3 runs without a detection rate improvement.

We evaluated SIGMA on IDSs based on four different classification algorithms : Neural Network, Random Forest, Support Vector Machine, and Naive Bayes Classifier. Each IDS was composed of two parts : a discriminator, trained to detect generated attacks, and an attack-classifier trained on the original dataset, to classify benign traffic and attacks. We trained a GAN and ran a local-search and genetic algorithm hybrid [23] to generate attacks against our IDSs. We compare the results of our model trained with both GAN and metaheuristics generated instances, with a model trained only with GAN generated instances over time and

another model trained only with metaheuristics generated instances.

Results show that for IDS consisting of a Neural Network or a Random Forest algorithm, the SIGMA method allowed for a detection rate of 100% of generated attacks two to three times faster than the model strengthened only with the GAN generated attacks. However, models trained only with the instances created using metaheuristics search were almost completely unable to detect GAN generated attacks, suggesting that metaheuristics alone are not sufficient to increase the robustness of the studied IDS.

The remainder of this paper is organized as follows. Section 3.2 provides an overview of the technologies used in our model. We discuss the related literature in Section 3.3. Section 3.4 presents our strengthening method to increase the robustness of IDS against generated adversarial attacks (i.e., SIGMA). Section 3.5 describes the approach followed to evaluate SIGMA, while Section 3.5.5 discusses the obtained results. Section 3.6 discusses threats to the validity of our study and Section 3.7 presents some implications of our work. Finally, Section 3.8 concludes the paper, summarising our findings along with some recommendations for future work.

3.2 Background

This section provides background information about Generative Adversarial Networks and metaheuristics used in this paper.

3.2.1 Generative Adversarial Networks

Generative Adversarial Networks are a class of unsupervised machine learning algorithm. They are composed of two neural networks : a generator G and a discriminator D.

Considering a dataset, the generator generates new data instances similar to the ones in the dataset. The discriminator, on the other hand, evaluates the data authenticity, i.e., whether or not the data it reviewed belongs to the actual dataset. The goal of the generator is to generate data labeled as genuine by the discriminator.

The generator takes random numbers as input (referred to as random noise), and returns a data instance.

With x as input of the discriminator D, we represent the probability that x is an attack generated by G as $\mathrm{D}(x)$. Therefore, $\mathrm{D}(x)$ is equal to zero when x is considered as an authentic data from the dataset, and equal to one when x is labeled as generated data, or fake.

With z as random noise, we represent the instance generated by the generator network G as $\mathrm{G}(z)$.

The generator G is trained to maximize the function $1 - D(G(\mathrm{z}))$.

As shown on Fig. 3.1, the Generator and the Discriminator are trained simultaneously, therefore being on a constant battle throughout the training process.

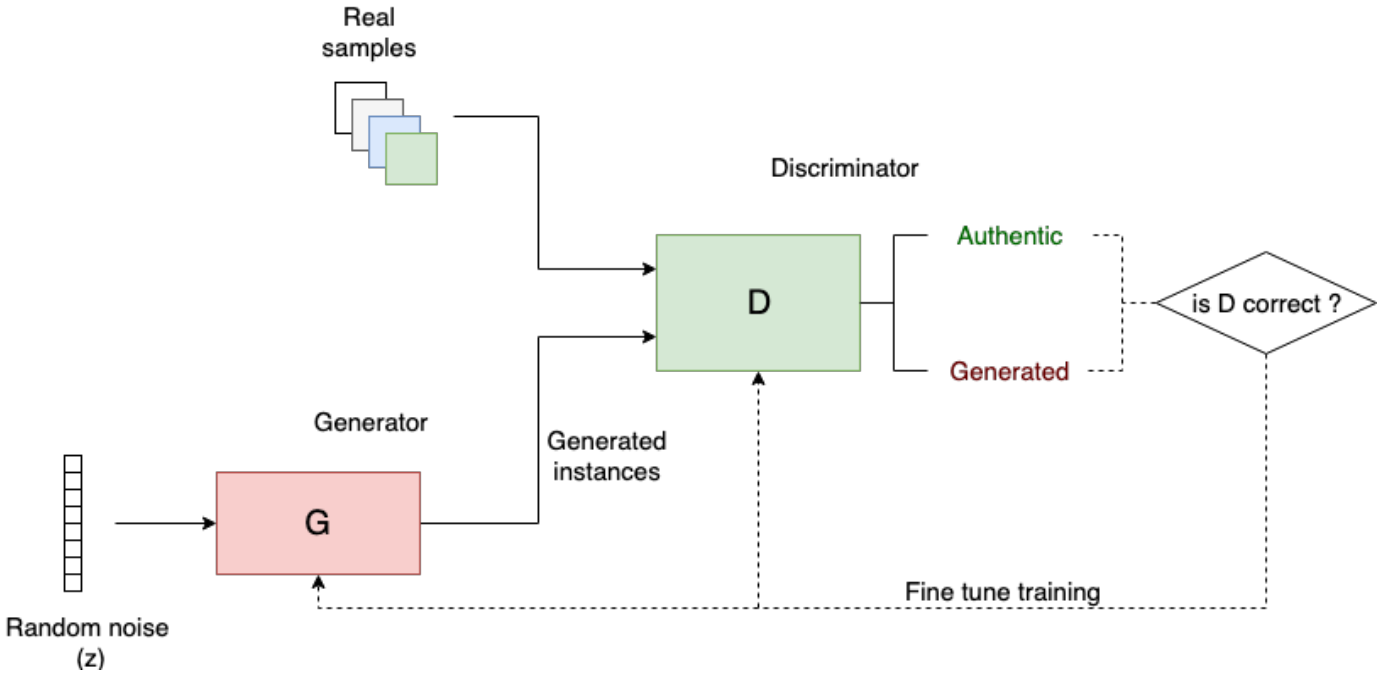

Figure 3.1 Diagram of a Generative Adversarial Network (GAN).

This algorithm has rapidly grown in popularity thanks to its performance in image generation [14]. It can generate realistic examples, and has a better performance than Deep Belief Networks or Boltzmann Machines [12].

GAN are also notably used to disrupt trained classifiers [13] : slight controlled modifications to the original input leads to misclassification. This has been extensively applied to images classification, due to its impressive results in this field : for instance, small visible changes made to “Stop” traffic signs tricked autonomous cars into misclassifying them into speed limit signs [24].

In this paper, we will use this ability of GANs to disrupt trained classifier by training them to generate attacks able to bypass the detection algorithm, *i.e.,* attacks classified as benign traffic by our IDS. Adding subtle modifications to the features of existing attacks could in fact lead to misclassification.

3.2.2 Metaheuristics

A metaheuristic is an algorithm used to find, generate, or select a heuristic (i.e., a partial search algorithm) that can provide a sufficiently good solution to an optimization problem with incomplete information. This type of algorithm is usually employed to solve computationally hard problems for which regular optimization would be too costly. Even if they do not guarantee finding the optimal solution for the problem, they usually provide good results, often close to the optimal solution [25]. A metaheuristic approach could be either single-solution based or population based.

A single-solution based approach could be local optimization : we randomly initialize a solution and explore the *neighbourhood* of the solution by applying local changes to the current solution. The search continues until a solution meeting the initial stopping criteria is found or a time bound is elapsed. Local optimization could be very effective in case the criterion to maximize only has a single optimum. In other cases, the local search algorithm can converge to a local optimum, therefore not giving the best possible solution.

The hill climbing algorithm is an example of local optimization algorithm. This algorithm is an iterative algorithm that tries to improve a solution by making an incremental change to it. If the change produced a better solution, then it becomes the new solution, and another incremental change is made to this new solution. This algorithm runs until there is no further improvement possible.

Genetic algorithm [26] is an example of a population based metaheuristic. It is inspired by the process of natural selection. It starts with a population of solutions, where each solution is randomly generated. The population then evolves until the stopping criteria is met or until a certain number of generations is reached. From this pool of solutions, we select the best solutions (selection) and recombine them into a new population of solutions (crossover). We then apply random mutations to this population, in order to have a diverse population of solutions and possibly exploring other parts of the solution space that were not explored yet. As a global search algorithm, the genetic algorithm metaheuristic is more likely to find global optima for multimodal functions but it is slower at converging [27].

Since the genetic algorithm is rather slow to converge, it is possible to combine those two approaches (local optimization and a population based solution) to have a faster convergence. We then refer to this method as an hybrid algorithm [23]. It consists of a slight modification of the genetic algorithm to incorporate a local optimization element : after the selection process, we optimize each solution of the population with the local search algorithm. This leads to overall better results, since the local search can only improve a solution, and could mean a faster convergence [25].

The overall processing of the hybrid algorithm is exposed in algorithm 1.

Algorithm 1: Pseudocode of the hybrid algorithm

```
Input: Instance (I), size of population (α), selection rate(β), mutation rate (m), number of
       iteration (nb_it), Local search algorithm (local_search)
Output: Population of solutions to I
/* Initialization                                                    */
/* Generate α random solutions to I                                  */
solutions = generate_random(α);
for i = 1 to nb_it do
    /* Local Search                                                  */
    solutions = local_search(solutions);
    /* Selection                                                     */
    n = α * β ;
    pop = select_best(n, solutions) ;
    /* Crossover                                                     */
    p = (α − n);
    for j = 1 to p do
        randomly select Sol_A and Sol_B from solutions;
        generate X_AB by mixing Sol_A and Sol_B;
        save X_AB to offsprings;
    end
    /* Mutation                                                      */
    for child in offsprings do
        mutate child with probability m;
    end
    solutions = pop + offsprings
end
return solutions
```

This hybrid method has often been used to solve complex problems with good results [25].

In the intrusion detection domain, such algorithms can be used to generate attacks that the intrusion detection system is unable to detect. In this case, a solution would be an actual attack, and all operations (cross-over, mutation, ...) would be modifications of the attacks features.

3.3 Related work

Over the last two decades, researchers have built several intrusion detection datasets by extracting different network features from real networks during cyberattacks. [17] [20] Different machine learning algorithms have been explored to build IDS : from a simple feed-forward neural network, to Extreme Machine Learning [28], to complex Recurrent Neural Networks [9]. Studies show that even simple algorithms, such as a Support Vector Machine

or J48 decision trees, could lead to good detection results, with 95% accuracy for the SVM and more than 97% accuracy for the decision tree [29]. Those algorithms could be used in practice by smart objects as intrusion detection systems. In fact, they don't require as much energy as complex Deep Learning models, which is an important factor to consider with such resource-constrained environment [9].

Nonetheless, these machine learning methods suffer from a severe flaw as an IDS : they are totally vulnerable to new types of attacks. Successful attacks can lead to terrible consequences : economic loss, important privacy issues for smart objects users, etc. Moreover, it is now possible to automatically build new attacks against which those systems will be utterly useless, thanks to metaheuristics and Generative Networks, for example.

The use of metaheuristics for attack generation has been explored by Jan et al. [15]. In this work, Hill Climbing and Genetic algorithms are used to generate malicious XML injections. These generated attacks were used for testing purposes but demonstrated the possibility to automatically create attacks using metaheuristics techniques ; in particular, the genetic algorithm managed to create a wide variety of attacks evading the web application sanity check more than 95% of the time.

Hu et al. [7] leveraged generative adversarial networks for malware detection, by considering the detection algorithm as a black-box (as would an attacker). The attacker does not know the internal structure of the detection model, but only knows the detection result of the detection model under attack. Even without having any information on the detection system, this approach led to very impressive results ; GANs deceiving the malware detection algorithm almost every time.

Furthermore, recent work has shown that it is possible to generate adversarial examples with intrusion detection datasets. In particular, Lin et al. [6] used a Wasserstein GAN [30] to generate adversarial attacks against different classifiers considered as black-box algorithms by the attacker, trained with the NSL-KDD [29] dataset. The GAN was able to mislead several classification algorithms into classifying generated attacks as benign traffic. Nonetheless, the NSL-KDD dataset is now 10 years old and its relevance is then questionable. Moreover, researchers have pointed out several problems with NSL-KDD [19], e.g., the lack of Remote to Local and User to Root attacks, as well as the lack of more recent type of attacks [21].

These last few years, some progress has been made on protecting Intrusion Detection Systems against generated adversarial attacks. Generative models are a double-edged sword, as they can be preemptively used to train the detection model as well.

Cordy et al. [16] created increasingly resilient defense strategies to detect training attacks

against a clustering-based IDS. The IDS was improved by simultaneously searching for attacks against the IDS and constantly improving the defense strategy : two genetic algorithms (one for creating attacks, the other to elaborate defense strategies) were used. Their resulting system detected 98% of the generated attacks, whereas the attack generation process systematically found a way to deceive the IDS without defense strategy. This promising result suggests that metaheuristics can be successfully used to preemptively strengthen an IDS against generated attacks. Nonetheless, this work does not provide any insight regarding the vulnerability of this strategy to other types of generated attacks (GAN generated instances for example).

The work of Ferdowsi and Saad [1] presents an approach to deploy a distributed intrusion detection architecture capable of detecting adversarial generated attacks. In this work, GANs were trained to generate adversarial attacks, and were then used to train a discriminator, which determined whether the current internet traffic was benign or an attack. However, this system might be susceptible to iterative generated attacks : once the discriminator is trained, it may still be possible to find ways to generate instances able to bypass the detection system. An IDS resilient to generated iterative attacks has not yet been explored in the intrusion detection domain.

3.4 SIGMA : An approach to improve the robustness of IDSs

In order to increase the robustness of IDS, we propose the following SIGMA method.

We take as input a Machine Learning based Intrusion Detection System, and a dataset consisting of attacks and benign traffic. We iteratively generate attacks with two different methods to train the IDS.
Each training iteration is designated by its number. We note $Score_i$ the detection rate of generated attacks by the IDS at iteration number i, meaning :

$Score_i = \frac{number\ of\ detected\ generated\ attacks}{total\ number\ of\ generated\ attacks}$

We consider that the generated attacks detection rate of our IDS has converged if : For $\epsilon > 0$, there exist an iteration number N such that for all iterations i after N, we have :

$|Score_i - Score_{i+1}| < \epsilon$

The SIGMA method instructions are as follows :

While the generated attacks detection rate of our IDS has not converged :

— Step 1 : We train a GAN to generate adversarial attacks against the IDS, considered as

a black-box. The goal for this algorithm is to generate attacks deceiving the intrusion detection system. Considering the same notation as in Section II.B, the function to maximize for the generator is $1 - D(G(z))$ where $G(z)$ is the generated attack, $D(x)$ is the probability (computed by the IDS) that x is an attack : the IDS plays the role of discriminator.

At each iteration, the generative algorithm generates new attacks to fool the Intrusion Detection System.

— Step 2 : We use the trained GAN to generate attacks against the IDS. We evaluate the score of the detection system for these generated attacks. If the score has not improved for 3 consecutive rounds, we stop the algorithm.

— Step 3 : We run a Search-based method in order to search for other possible attacks deceiving the IDS that the GAN might have missed.

The function to maximize for this generative algorithm is : $1 - D(sol)$ where sol is the solution generated by the Search-based algorithm, and $D(x)$ is the probability (computed by the IDS) that x is an attack.

— Step 4 : We use the Search-based method to generate attacks against the IDS. We then train the IDS with the generated instances from both algorithms (i.e., GAN and metaheuristics) and with data from the original dataset. Exposing its classifier to real data and generated attacks prevents it from overfitting to generated instances and losing accuracy on other type of traffic.

The overall proceedings is illustrated in the algorithm 2.

By combining attacks from both the Machine Learning and the Metaheuristics methods, we expect to explore a larger solution space since the two techniques are significantly different ; we expect the generated attacks to be widely distinct. Being confronted with a large sample of diverse attacks, an IDS is likely to gain in robustness.

3.5 Evaluation of SIGMA

In this section, we evaluate the effectiveness of SIGMA at improving the effectiveness of an IDS. The quality focus is the improvement of the attack detection rate, through iterative reinforcement using GANs and metaheuristics. The perspective is that of researchers interested in developing efficient IDS, and practitioners interested in improving the robustness of their IDS. The context consists of the CICIDS2017 benchmark dataset [20], containing 11 types of networks attacks, and four machine learning-based IDS (i.e., a 3-layers Neural Network, a Random Forest, A Support Vector Machine (SVM), and A Naive Bayes Classifier). In the

Algorithm 2: Pseudocode of the SIGMA process

```
Input: IDS to improve (IDS), training set (train_set),
Output: Improved IDS
converged = False;
counter = 0;
previous_score = 0;
while converged = False do
    /* Step 1 : GAN training                                          */
    generator = GAN.train(IDS, train_set));
    /* Step 2 : Attack generation and evaluation                      */
    GAN_attacks = generator(noise);
    predict = IDS(GAN_attacks);
    score = nb_attacks(predict) / length(GAN_attacks);
    if score ≤ previous_score then
        counter = counter + 1;
    else
        counter = 0;
        previous_score = score;
    end
    /* If the score has not improved after 3 rounds, we stop the algorithm
       */
    if counter = 3 then
        converged = True;
        break;
    end
    /* Step 3 : Search-based method                                   */
    search_based.run(IDS);
    SB_attacks = search_based.generate();
    /* Step 4 : IDS Training                                          */
    IDS.train(GAN_attacks);
    IDS.train(SB_attacks);
    IDS.train(train_set);
end
return IDS
```

following, we provide detailed information about the CICIDS2017 benchmark dataset and the implementation of SIGMA using the four selected machine learning-based IDS.

3.5.1 Dataset

The CICIDS2017 benchmark dataset [20] consists of more than 80 network flow features (flow duration, destination port, ...). Table 3.1 provides a summary of those characteristics. This recent intrusion detection dataset contains 11 types of attacks along with benign traffic. Each entry of the dataset consists of more than 80 columns (namely the extracted network flow features) and is labeled as one of those 11 types of attacks or as benign traffic. We grouped the 11 different attacks into four different groups as shown in Table 3.2, building four different balanced binary datasets (Attack, Benign), to counterbalance the unbalanced number of attacks per type.

Table 3.1 Some Network features used by CICIDS 2017.

Feature name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in forward direction
tot_bw_pk	Total packets in backward direction
fl_pkt_s	Number of packets transferred per second
ack_cnt	Number of packets with ACK
pkt_size_avg	Average size of packet
idl_avg	Mean time a flow was idle

We first deleted the constant columns of the dataset, as they don't provide any useful information for classification. Data now consists of 71 columns, 70 of them being network flow features, and the last one being the label (*i.e.*, 0 if it is benign traffic, 1 if it is an attack).

Then, since the values of each feature throughout the data widely varies, each column was normalized to have values between 0 and 1. Feature scaling allows for much faster convergence for neural networks.

We normalized data by applying the min-max normalization, namely :

$c'_i = \frac{c - c_{min}}{c_{max}}$.

Where :

— c_i is the column from the original dataset.

— c'_i is the normalized column.

Table 3.2 Attacks labels and distribution in the CICIDS2017 dataset.

Attack group	Number of attacks	Types of attack
Denial of Service	252661	DOS Hulk DOS GoldenEye DOS Slowloris DOS Slowhttptest
Distributed DOS	128027	DDOS
Bruteforce	15342	FTP-Patator SSH-Patator Bruteforce Portscan Botnet
Infiltration	720	SQL Injection XSS Heartbleed Infiltration

— c_{min} is the minimum value of the column.

— c_{max} is the maximum value of the column.

Each dataset was split into a training set and a test set, respectively representing 90% and 10% of the overall dataset.

3.5.2 Implementation of SIGMA

Step 1 : GAN training

We chose to implement SIGMA with a 4-layers Wasserstein GAN. The architecture of the GAN is detailed on Fig. 3.3. The dimensions of hidden layers were chosen experimentally, being the ones with the best results.

As mentioned in Section II.B., the Wasserstein GAN takes random numbers (or random noise) as input to generate attacks. We refer to the number of random numbers as the random noise size.

The goal of this generator is to generate attacks able to deceive the IDS. To ensure that the output of the generative algorithm is indeed an attack, we keep the functional features of an original attack.

Since every feature of our data has been normalized, each feature is represented as a number between 0 and 1. As shown on Fig. 3.2, we keep the functional features of real attacks for our generated attacks.

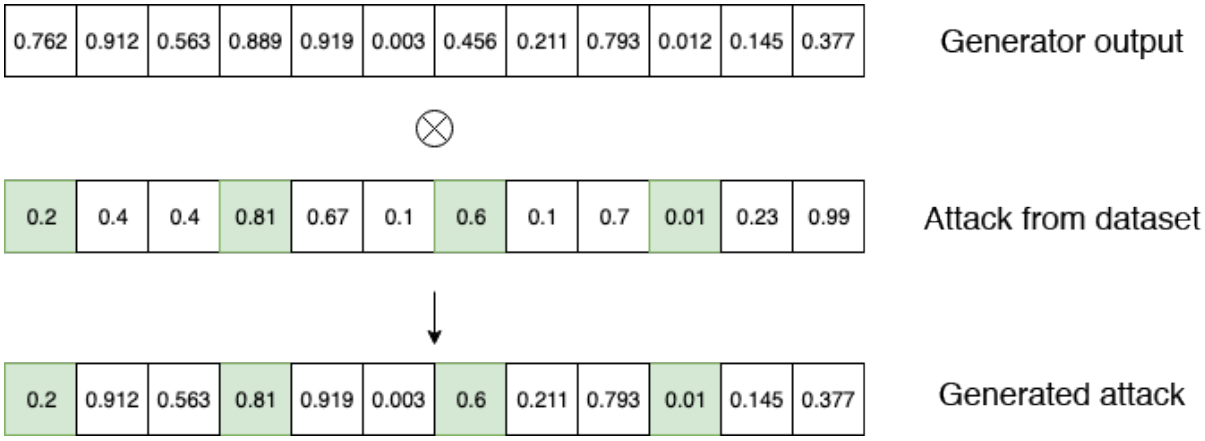

Figure 3.2 Diagram of the generative algorithm's process. In green, the functional features of the attack.

The functional features per attack type were identified by a statistical analysis of the datasets, with the help of the analysis conducted by the creators of the dataset [20]. They are presented in Table 3.3.

With the aim to have a Generative Adversarial Networks with the best performance and therefore explore the largest attack space possible, we trained several Generators with different sizes of noise as input. Furthermore, since there is unpredictability in the training of Generators due to the randomized weights initialization, we trained the Generators several times. We then select the GAN with the best performance among those, *i.e.,* the most able to deceive the IDS.

The process followed to train the GAN is presented in algorithm 3.

Step 2 : Attack generation and evaluation

This step is to evaluate the current score of our IDS. To do so, we need to generate attacks with the GAN and gauge the robustness of our IDS against those attacks.

After the GAN has been trained at step 1, we use it to generate attacks. Generated attacks will use the functional features of attacks from the test set.

We evaluate the score of the IDS with those generated attacks. With previous notations, we consider that an instance x is considered an actual attack by the IDS if $D(x) > 0.5$. The score is therefore the number of generated attacks $G(z)$ with $D(G(z)) > 0.5$, divided by the total number of generated attacks.

Algorithm 3: Pseudocode of the GAN training process

```
Input: IDS (IDS), training_set (train_set), Maximum noise size (max_noise_size),
       Number of training epochs (nb_epoch)
Output: Trained GAN
/* Initialization                                                    */
best_score = 1.0;
noise = 1;
for attempt = 1 to 5 do
    for noise_size = 1 to max_noise_size do
        /* We construct a GAN with the corresponding noise size as input  */
        GAN = Generator(noise_size);
        for epoch = 1 to nb_epoch do
            for (batch, labels) in train_set do
                /* First select the attacks from the training set         */
                is_attack = non_zero(labels);
                attacks = select(batch, is_attack);
                /* Then generate attacks                                  */
                z = random_noise(noise_size);
                generated_attacks = GAN(attacks,z);
                /* Backpropagation                                        */
                loss = mean(IDS(generated_attacks));
                loss.backward();
                optimizer.step();
                if loss ≤ best_score then
                    best_score = loss;
                    noise = noise_size;
                    best_GAN = GAN;
                end
            end
        end
    end
end
return best_GAN
```

Table 3.3 Functional features per attack type. Those features are not going to be modified by the generator.

Attack group	Functional features
DOS	Flow Duration, Active Mean, Average Packet Size, Packet Length Std, Flow IAT Mean, PSH Flag Count, Idle Max
DDOS	Flow Duration, Bwd Packet Length Std, Average Packet Size, Packet Length Std, Flow IAT Std, ACK Flag Count
Bruteforce	PSH Flag Count, Flow Duration, Total Length of Fwd Packets, Init Win bytes forward, Packet Length Std, Subflow Fwd Bytes, Fwd PSH Flags
Infiltration	Subflow Fwd Bytes, Total Length of Fwd Packets, Flow Duration, Idle Mean, Active Mean, Init Win bytes backward, PSH Flag Count

If the score has not improved in three rounds, we stop the algorithm.

Step 3 : Search-based method

In this step, we run a metaheuristic algorithm in order to generate additional attacks to further improve our detection system.

As our Search-based method, we used an hybrid genetic local-search algorithm. Indeed, local search and the Genetic Algorithm both have their pros and cons. The Genetic Algorithm is rather slow to converge whereas the Local search could converge to local optima. We chose to combine the two with an hybrid genetic algorithm [23], as it has been demonstrated to have been more efficient in complex problems such as the Traveling Salesman [31].

The hybrid algorithm that we chose is a modification of the genetic algorithm : before proceeding to the selection process of the algorithm, every solution from the solution pool is improved by the local-search algorithm. As each solution is enhanced before the selection process, this algorithm allows for overall better performances, and usually a faster conver-

gence than the standard Genetic Algorithm.

The goal of this metaheuristic algorithm is also to generate attacks against the IDS. Similarly to the proceedings of the GAN, functional features of our generated attacks will be from real attacks from the original dataset.

We first create a population of random solutions. We chose a population size of 30, as the recommended values in the literature are within the range of 30 to 80 [32]. Having a bigger population affected the performances of our algorithm.

Before the selection process, we optimize each solution of the population with a local search algorithm. The pseudocode for this local search method is given in algorithm 25.

Crossover is made by selecting two parents in the solution pool. We select only members of the population with the highest score (meaning, the attacks the most able to fool the IDS). The offspring will have the first half of its features from its first parent, and the other half from its second parent.

The mutation process is carried out to the entire population of children of this iteration. For each child, a non-functional feature selected at random is modified. The modification follows a uniform distribution, varying from -0.01 to 0.01.

Then, the new generation is equally composed of parents from the previous generation, and of its offspring. The fact of having members of the previous generation prevents the deterioration of the ability of the overall population to deceive the Intrusion Detection System.

We stop the hybrid genetic algorithm after 500 generations, or after 50 generations without improvement. These numbers were found to be experimentally sufficient for successfully training the four different IDS.

This population-based approach makes the solution pool iteratively evolve to better evade the detection system, and therefore generates a wide variety of adversarial attacks.

Step 4 : IDS training

In this final step, we aim to retrain the detection system for it to take the generated attacks into account. We train the IDS with :

— All the attacks generated by the hybrid algorithm during its run at step 3.

— The trained GAN generated attacks from the training set.

— Examples from the original training set.

3.5.3 Execution of SIGMA

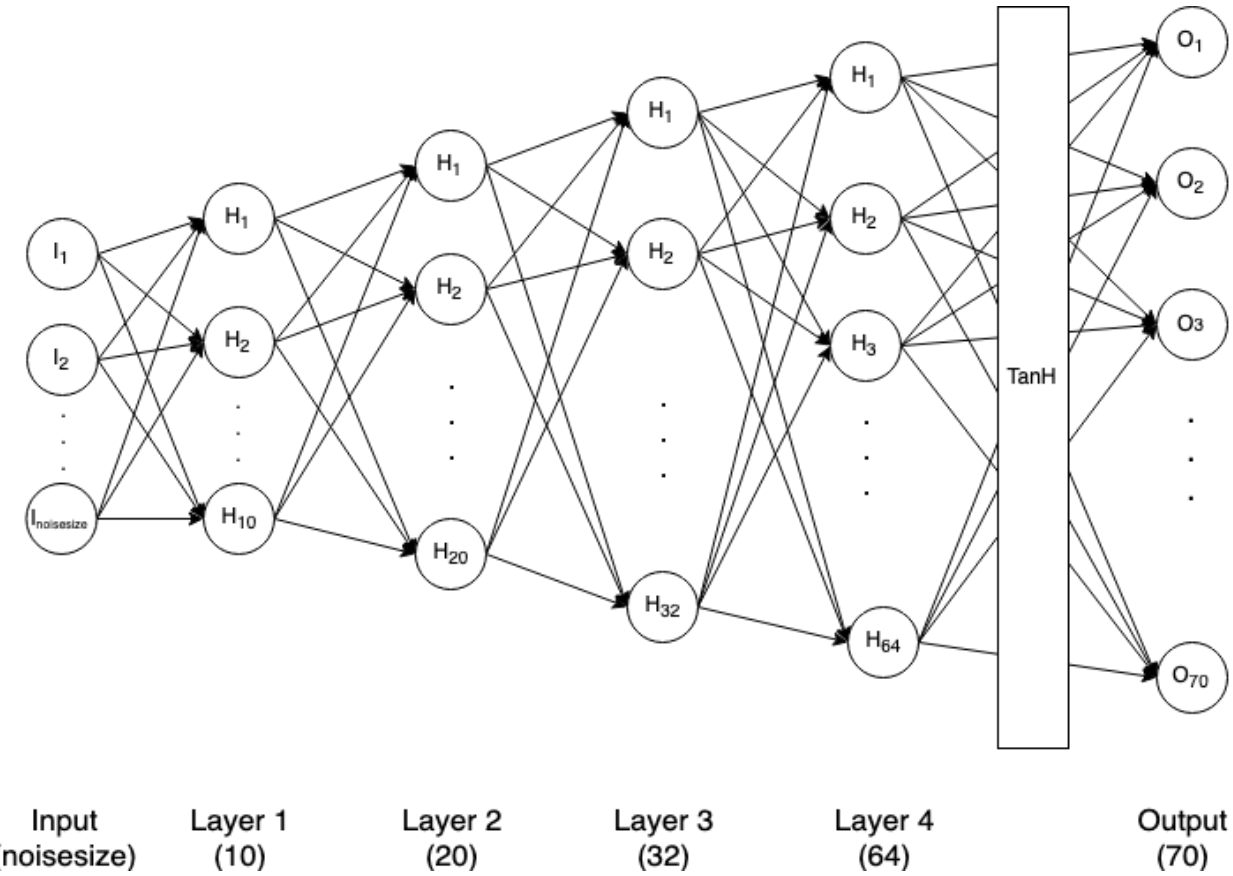

Figure 3.3 Architecture of the Generator. Below each layer is shown its dimension. As our GAN takes noise as input, we can make the dimension of this noise vary to change the architecture of the network. The dimension of the output is 70, the number of network features of a dataset entry.

We executed SIGMA on the CalculQuebec Cloud service with the following computing resource : 15 X Intel Xeon @2,5Ghz, 128Go RAM, 10 core, 8 X Nvidia K20-GK110 GPU.

The Pytorch module was used to implement all the neural networks.

In Table 3.4, we present all the parameters used to train the Neural Networks.

3.5.4 Research questions

To evaluate the effectiveness of SIGMA at improving the effectiveness of IDSs, we formulate the following two research questions :

— (RQ1) *To what extent SIGMA can generate adversarial attacks able to deceive trained classifiers, acting as Intrusion Detection System ?*
This research question aims to assess the effectiveness of SIGMA at generating meaningful adversarial attack queries.

— (RQ2) *To what extent is the effectiveness of IDS improved using SIGMA ?*
This research question aims to examine if through the successive re-training steps of SIGMA, IDSes are successfully improved.

In the following, we describe the approach followed to answer RQ1, RQ2.

For RQ1, we use four different classification algorithms as IDS : Neural Network, Random Forest, Support Vector Machine and a Naive Bayes Classifier. We generate attacks against each of the IDS for all four attacks datasets (DOS, DDOS, Bruteforce, Infiltration) by using a GAN, trained with the methodology described above.

We compute the score of each of the detection systems for the GAN generated attacks, and therefore assess if SIGMA is able to deceive standard classification algorithms acting as IDS.

For RQ2, we use a more complex intrusion detection system. We build an IDS consisting of two classifiers : an attack classifier, and a discriminator. The attack classifier is trained with the entries from the original dataset, whereas the discriminator is trained with both regular attacks from the dataset as well as with generated entries to classify the input as a generated attack or as regular traffic. Traffic is first analyzed by the discriminator to determine whether it is an adversarial instance or real traffic. If the input is labeled as real traffic, it then comes through the attack classifier whose role is to recognize attacks. This architecture prevents from training the classifier with the adversarial examples, which could lead to a loss of performance for previously seen regular attacks because of overfitting to adversarial instances. It consists of a simple adaptation of the GAN discriminator to detect both generated instances and attacks from the dataset. Therefore, since the goal of the discriminator is to identify generated instances, it will be the part of the IDS trained with the SIGMA generated attacks.

The overall process of the Intrusion Detection System studied is detailed on Fig. 3.4

As attack classifier and discriminator, we used the same algorithms as for RQ1 : Neural Network, Random Forest, Support Vector Machine (SVM), and a Naive Bayes Classifier.

We study the largest dataset of the four (the DOS attacks dataset). For each round of improvement of SIGMA, we compute the score of the IDS.

To measure the performance of SIGMA, we compare our strengthened model to a baseline, in which the discriminator is trained only with GAN generated instances. We also verify that metaheuristics alone are not enough to train our system against generated adversarial attacks by comparing the model strengthened by SIGMA with a model trained only with the metaheuristics attacks, and submitting it to GAN generated attacks.

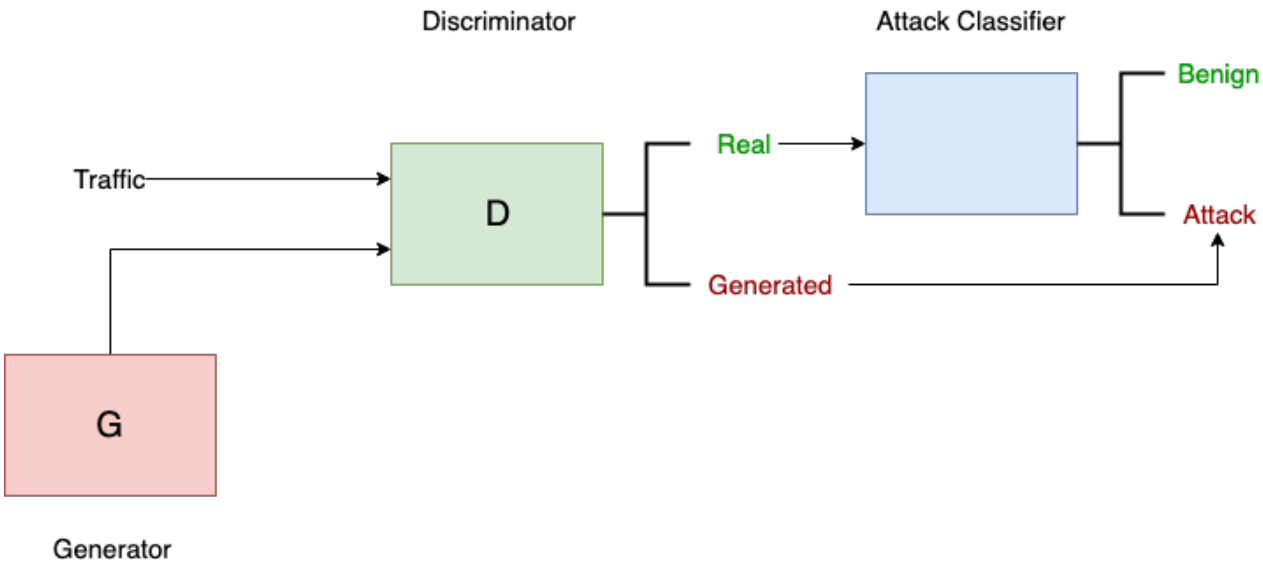

Figure 3.4 Diagram of the Intrusion detection system. Characteristics from the traffic are the inputs of the Discriminator, and goes into the attack Classifier if the Discriminator labels it as a real entry of the dataset.

We will judge the quality of the reinforcement by :

— The speed of convergence of the detection rate.

— The value of the limit of the detection rate.

— The overall performance of the model for all iterations.

Algorithm 4: Pseudocode of the used local search algorithm

```
1  Function score(sol : array) : float is
2      score = 1 - max(discriminator(sol),classifier(sol));
3      return score;
4  end
   Input: Population of solutions (solutions), Discriminator, Classifier, functional features
          (func_feat)
   Output: Optimized population of solutions
5  for sol in solutions do
       /* For each solution in the population, we slightly modify all the non
          functional characteristics to find the best solution in the
          neighborhood                                                    */
6      for characteristic in sol do
7          if characteristic not in func_feat then
8              modif = -0.01;
9              current_value = characteristic;
10             best_value = characteristic;
11             best_score = score(sol);
               /* We test all modifications from -0.01 to 0.01            */
12             while modif < 0.01 do
13                 modif = modif + 0.001;
14                 characteristic = current_value + modif ;
15                 score = score(sol);
16                 if score > best_score then
17                     best_value = characteristic;
18                     best_score = score;
19                 end
20             end
21             characteristic = best_value;
22         end
23     end
24 end
25 return solutions
```

Table 3.4 Training parameters for our Generative Adversarial Network, and for Neural Networks used as IDS.

Number of training epochs	30
Batch size	64
Learning rate	0.01
Loss function	$L1(\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}, \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}) = \sum_i \lvert u_i - v_i \rvert$
Optimizer	Adam

Table 3.5 Detection rates for test attacks from the dataset and GAN generated instances.

Classifier type	Neural Net		Random Forest		SVM		Naive Bayes	
Type of attack	Normal	Adversarial	Normal	Adversarial	Normal	Adversarial	Normal	Adversarial
DOS	94,9%	0%	99,8%	0%	97,6%	0%	95,9%	0%
DDOS	98,6%	29,9%	99,9%	0%	98,3%	47,1%	97,1%	0%
Bruteforce	95,6%	1,9%	99,9%	0%	96,3%	0%	97,8%	0%
Infiltration	95,4%	5,8%	100%	0%	96,2%	4,1%	97,4%	0%

Table 3.6 Evolution of the detection rates of adversarial attacks for our model

Classifier type	Neural Net		Random Forest		SVM		Naive Bayes	
Iteration number	Normal	Reinforced	Normal	Reinforced	Normal	Reinforced	Normal	Reinforced
1	0%	0%	0%	0%	0%	0%	0%	0%
2	6,3%	0%	51%	0%	100%	100%	100%	100%
3	49%	100%	99%	100%	100%	100%	100%	100%
4	100%	100%	18%	100%	100%	100%	100%	100%
5	68%	100%	100%	100%	100%	100%	100%	100%
6	100%	100%	100%	100%	100%	100%	100%	100%
7	100%	100%	100%	100%	100%	100%	100%	100%

It should be noted that for the first iteration of the algorithm, the discriminator has not yet been trained : the generator is thus only trained against the classifier at the first iteration.

3.5.5 Results of the Evaluation of SIGMA

In this section we present the answers to our two research questions that aim to evaluate SIGMA.

RQ1 : To what extent SIGMA can generate adversarial attacks able to deceive trained classifiers, acting as Intrusion Detection System ?

The results of the detection of normal and generated attacks are presented in Table 3.5, and on Fig. 3.5.

All four classifiers in our study (Neural Network, Random Forest, SVM, Naive Bayes) have good results in classifying standard entries of the datasets. Even though our classifiers are standard machine learning algorithms, they are sufficient to obtain high accuracy, with the Random Forest algorithm performing with the best results with an overall 99,9% accuracy, followed by the Support Vector Machine with 97,1%. In fact, those two algorithms have often been used in intrusion detection thanks to their good performances [33].

However, the generated attacks detection rates is significantly low for all classifiers with most type of attacks. Both the Random Forest and the Naive Bayes are utterly unable to detect the GAN generated adversarial attacks. The neural network and the SVM are the most resilient classifiers, but the generator still manages to deceive our IDS with over a 90% evasion rate for the DOS, Bruteforce and Infiltration attacks.

The results show very good performance of the Generative Adversarial Network for all different types of attacks. It is therefore possible to generate attacks able to fool Machine Learning based classifiers for all four types of attacks.

RQ2 : To what extent is the effectiveness of IDS improved using SIGMA ?

We compared the evolution of our model trained with the hybrid local-search-genetic reinforcement and adversarial attacks with a model trained only with adversarial attacks.

The results are presented in Table 3.6 and Fig. 3.6.

First, we notice that both models with the SVM and the Naive Bayes as classifiers only need one step to detect adversarial attacks : those two classifiers are the most able to generalize from the previously seen data. The generated attacks detection rate converges after only one iteration for both the strengthened and the standard model.

The multi-layer Neural Network and the Random Forest standard models both take time to

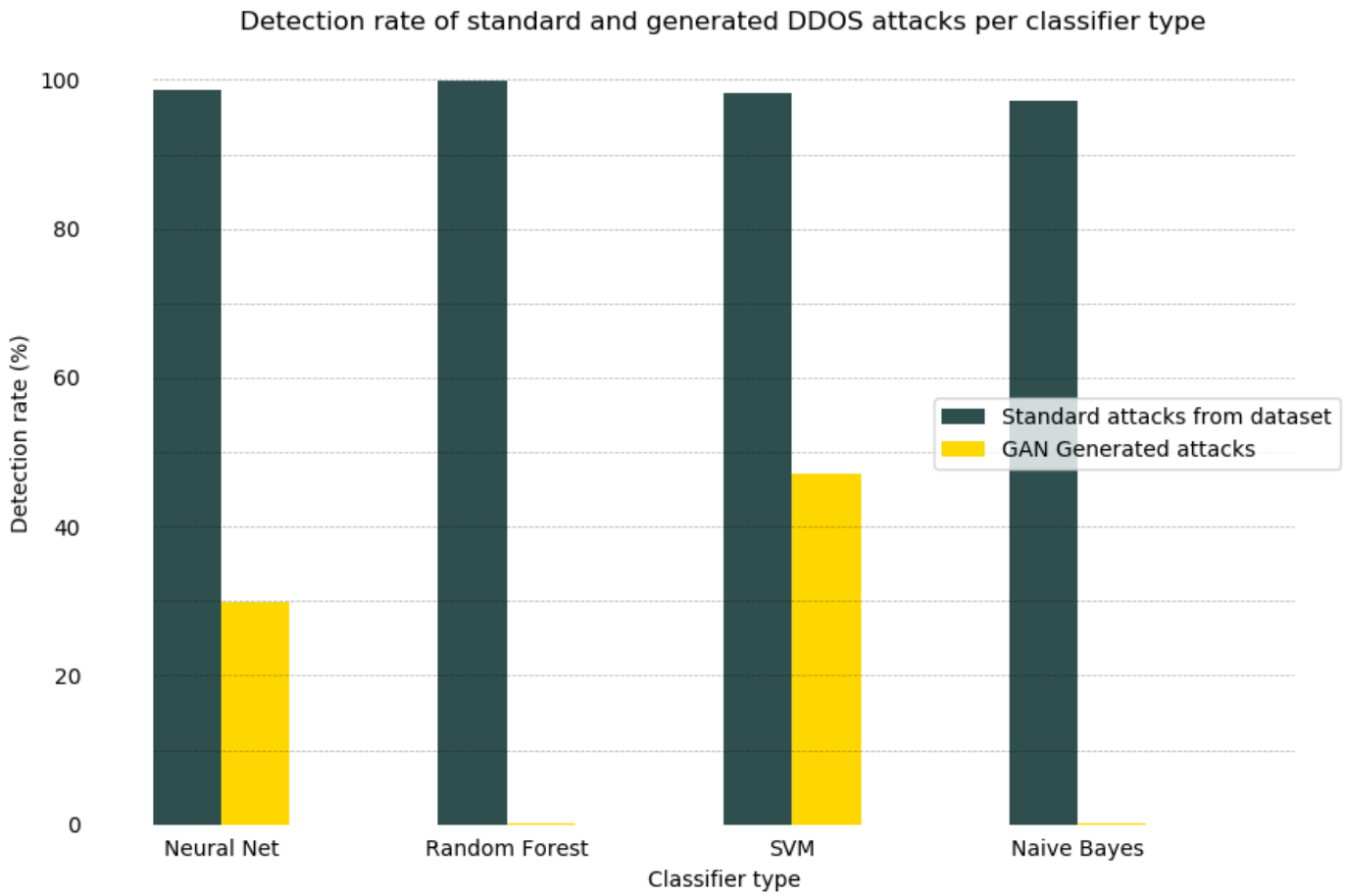

Figure 3.5 Detection scores per classifier with test set attacks and generated attacks for the DDOS dataset.

converge to a 100% generated attacks detection rate : 6 iterations for the model with the Neural Network as classifier, 5 iterations for the Random Forest model. Furthermore, we also note that both models suffered from overfitting : their performance increased (until iteration 4 and 3 respectively) before dropping significantly by 32% and 81%.

The SIGMA method improved the models' results : as we can see, the strengthened model converged faster than the standard model to a 100% detection rate for both the Neural Network and the Random Forest classifiers ; the reinforced versions took only two iterations to detect all adversarial instances, that is to say respectively four and three iterations less. As the other two classifiers, namely the SVM and the Naive Bayes classifier, detected all attacks from iteration 2, the reinforcement method did not affect their performance.

Furthermore, we can observe that the SIGMA method prevented the Neural Network and the Random Forest model from overfitting to generated attacks, therefore preventing a per-

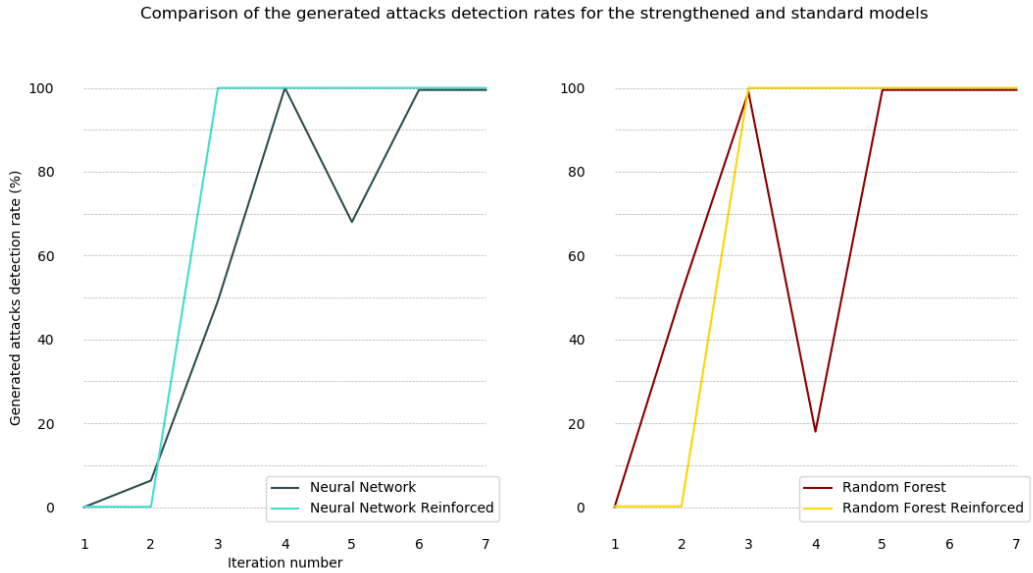

Figure 3.6 Time evolution of our reinforced model with two different classifiers (Neural Network and Random Forest) and their corresponding standard model.

formance drop of the algorithm. The combination of the metaheuristic algorithm and the Generative Adversarial Network permitted to generate a sufficiently wide variety of attacks ; avoiding fitting closely to previously seen attacks.

Table 3.7 Evolution of the detection rate of adversarial attacks for a model trained only with Metaheuristics generated attacks.

Iteration	Neural Net	Random Forest	SVM	Naive Bayes
1	0	0	0	0
2	0	0	0	0
3	0	0	40.1	0
4	0	0	100	0
5	0	0	58.1	0
6	0	0	0.8	0
7	0	0	52.9	0

Table 3.7 presents the results of models trained only from the Metaheuristics generated attacks. From these results, we can also conclude that Metaheuristics alone are not sufficient to train an IDS against generated adversarial attacks : every classifier, except the Support Vector Machine, was utterly unable to detect any instance generated by our Wasserstein GAN. The SVM stands out from the other classifiers thanks to its ability to generalize, but

fails at consistently detecting GAN generated attacks.

We can therefore conclude that the attacks generated by the Metaheuristics algorithm complement the ones generated by the Generative Adversarial Networks, as the Metaheuristics algorithm alone was not enough to successfully train the IDS.

3.6 Threats to validity

This section discusses the threats to validity of our study following common guidelines for empirical studies [34]

Construct validity threats concern the relation between theory and observation. This is mainly due to possible mistakes in the generation of attacks. Even though we kept the functional features of real attack untouched for our generated attacks, we can not guarantee that the generated attacks metrics are indeed plausible attacks.

Internal validity threats concern the selection of tools and analysis methods. We split the dataset into a training and a test set in order to ensure the validity of our results. This prevents having a biased evaluation of our model. As the aim of the method was to try to detect as many generated attacks as possible, we chose to study the generated attacks detection rate as a metric to gauge the quality of the strengthening.

Reliability validity threats concern the possibility to replicate our study. All the tools used in this study are open-source.

Conclusion validity threats concern the relation between treatment and the outcome. We paid attention to not make too broad statements about the performances of our model.

External validity threats concern the possibility to generalize our results. The results of the SIGMA method have to be interpreted carefully, as they may depend on the dataset used to run the experiment and on the used Intrusion Detection System. The iterative strengthening method has only been studied for DOS attacks of the CICIDS 2017 dataset [20]. We used four different classifiers acting as IDS, and were able to significantly improve the results of two of the four IDS. We therefore suggest that our results can be generalized to other detection systems and other datasets.

3.7 Implication for Practitioners and the Industry

Artificial Intelligence is a really powerful tool that could and will be used in future cybersecurity systems : IBM's Watson is one of the illustrations of the application of Machine Learning in this field. Nonetheless, this work illustrated possible vulnerabilities of such systems as Artificial Intelligence can also be leveraged by attackers to disrupt detection systems.

Generative Adversarial Networks can be used to forge almost undetectable adversarial attacks for systems that have not already faced such attacks. Our method confronted our studied Intrusion Detection Systems with attacks generated with both GANs and Metaheuristics in order to improve the systems resilience, as our analysis has shown that the more attacks the system faces, the more it will be able to efficiently generalize to other potential attacks.

Repetitively training an IDS with generated attacks is a way to anticipate for every possible generative scheme that could target the system. By doing so, our method SIGMA is able to detect all the attacks generated by our GAN, thus preventing future intrusion by adversarial generated attacks.

Such methods should be applied to any AI-based cybersecurity system in the industry to preemptively confront them to new types attacks, therefore preventing them from such possible threats.

3.8 Conclusion

The novel ability to use Machine Learning techniques to generate adversarial attacks requires the development of a robust IDS able to detect unusual behaviors. Generative Adversarial Networks are both a terrible weapon for detection systems, and an incredible opportunity to preemptively strengthening IDSs against adversarial attacks.

We have shown experimentally that it is possible to effectively evade intrusion detection classifiers with Generative adversarial networks. We demonstrated the possibility to forge undetected adversarial attacks with GANs against four standard Machine Learning algorithms acting as IDS, with the generated attacks detection rates dropping near 0% for most of them.

To prevent adversarial generated attacks, we presented in this paper a method SIGMA, to improve the robustness of IDS. This method is based on the iterative generation of attacks by a Machine Learning Generative algorithm and Metaheuristics. We have shown that applying this method to Machine Learning based IDS can speed up the convergence of the generated attacks detection rate, and prevent overfitting to previously seen generated attacks.

Our model may help design Intrusion detection systems robust against recurrent generative attacks and improve the security of Machine Learning systems.

Further considerations are the explorations of other more complex detection algorithms, such as Recurrent Neural Networks, the application of the SIGMA method to other datasets and the design of a distributed detection system robust to adversarial attacks.

CHAPITRE 4 DISCUSSION GÉNÉRALE

Un des buts de ce travail de recherche était de démontrer la possibilité de générer des métriques d'attaques capables de tromper des classifieurs pré-entraînés par l'ensemble de données CICIDS2017. Cet ensemble de données récent conduit à de très bons résultats de classification d'attaques, et permet de disposer de types d'attaques plus variés que NSL-KDD.

Nous avons entraîné plusieurs réseaux génératifs dans le but de générer des attaques non détectées par nos classifieurs. Nous avons conclu de cette étude que :

— Il est possible de générer des attaques quasiment non détectées pour les quatre types d'attaques étudiés pour les classifieurs étudiés (Réseau de neurones, Random Forest, SVM, Naive Bayes).

— Le classifieur disposant de la meilleur capacité à généraliser face à des attaques générées adversarielles est la SVM.

L'objectif principal du travail de recherche était le développé d'une méthode de renforcement de systèmes de détection d'attaques. Nous avons développé une méthode, appelée SIGMA, de renforcement de système de détection basé sur la génération d'attaques par des réseaux GANs, ainsi que par des métaheuristiques. Celle-ci répond donc à l'utilisation des deux méthodes (GAN et Métaheuristiques) en sécurité évoqué dans la revue de littérature.

Nous avons alors étudié l'application de cette méthode à un IDS composé d'un discriminateur (dont le rôle est de détecter le trafic généré), et d'un détecteur d'attaque pré-entraîné.

Nous avons conclu des résultats que :

— Pour le réseau de neurones et la Random Forest, la méthode SIGMA permet la détection de 100% des attaques générées près de deux fois plus vite que le détecteur entraîné par des attaques générées uniquement par des GANs.

— La méthode SIGMA n'affecte pas les performances des deux autres classifieurs SVM et Naive Bayes, capables dès l'étape 2 de détecter 100% des attaques générées.

Cette méthode conduit donc à des résultats positifs pour l'ensemble de données CICIDS2017. Il conviendrait de l'étudier dans le cas d'algorithmes de détection plus complexes comme les Recurrent Neural Networks, et d'étudier les résultats de l'application de cette méthode sur d'autres ensembles de données.

CHAPITRE 5 CONCLUSION

La nouvelle capacité d'utiliser des méthodes issues de l'apprentissage machine afin de générer des attaques adversarielles appelle au développement d'un système de détection d'intrusions robuste, capable de détecter de tels comportement.

Les Generative Adversarial Networks constituent à la fois une arme considérable pour les attaquants, ainsi qu'une opportunité de se défendre préhemptivement, en renforçant les systèmes de détection d'intrusion.

5.1 Synthèse des travaux

Nous pouvons tirer deux conclusions de nos travaux.

Tout d'abord, nous avons montré qu'il est expérimentalement possible de générer des attaques à l'aide de réseaux génératifs, et que celles-ci peuvent échapper aux algorithmes de classification classiques, jouant ici le rôle de système de détection d'intrusion.

Ainsi, les attaques générées par les GAN ont pu échapper quatre classifieurs standard : réseau de neurones, Random forest, SVM, et Naive Bayes, avec des taux de détection d'attaques générées avoisinant 0% dans la plupart des cas.

Nous avons alors présenté une méthode afin de renforcer les systèmes de détection d'intrusion contre ces attaques générées. Cette méthode, appelée SIGMA, est basée sur la génération itérative d'attaques par méthode métaheuristiques et par réseaux adversariels.

Des attaques sont générées par les deux méthodes, entraînant alors un discriminateur, dont le rôle est de classifier le traffic entrant en "traffic généré" ou "traffic authentique".

Nous avons montré que cette méthode de renforcement appliquée aux algorithmes de classification standard de l'intelligence artificielle permet d'augmenter la robustesse du système. En effet, le système converge plus rapidement vers un état stationnaire, dans lequel il n'est plus possible de générer d'attaques adversarielles à l'aide de réseaux génératifs. Cette méthode permet aussi d'éviter l'overfitting des modèles aux attaques générées, évitant ainsi des baisses de performances pour des attaques vues auparavant.

Notre méthode est donc un pas fait vers l'amélioration de la robustesse des systèmes de détection d'intrusions face aux attaques générées par les techniques récentes issues du domaine de l'intelligence artificielle.

5.2 Limites de la solution proposée

Il est impossible de se prémunir de l'intégralité des attaques informatiques. Notre méthode permet de se prémunir d'attaques dont les caractéristiques sont générées artificiellement.

Néanmoins, rien n'indique le caractère plausible de ces attaques générées : il est effectivement possible que les métriques générées ne correspondent pas totalement à des situations réelles. Il s'agirait alors d'étudier le spectre possible des métriques des différentes attaques informatiques étudiées dans l'ensemble de données CICIDS2017.

5.3 Améliorations futures

Il conviendrait d'explorer la possibilité de générer des attaques à l'aide de Generative Adversarial Networks contre des systèmes de détection d'attaques plus sophistiqués : on peut par exemple parler de réseaux de neurones récurrents, modèles plus complexes notamment utilisés pour prendre en compte la notion temporelle des attaques.

De plus, le déploiement de tels algorithmes est coûteux en terme de ressources de calculs : il serait par exemple avantageux de tirer partie du paradigme distribué dans le cas de l'Internet des Objets, en répartissant les calculs dans un écosystème d'objets connectés.

RÉFÉRENCES

[1] A. Ferdowsi et W. Saad, "Generative adversarial networks for distributed intrusion detection in the internet of things." *arXiv preprint arXiv :1906.00567*, 2019.

[2] IOT-Analytics. (2018) State of the IoT 2018 : Number of IoT devices. [En ligne]. Disponible : https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b

[3] C. Kolias *et al.*, "Ddos in the iot : Mirai and other botnets," *Computer(50),7*, p. 80–84, 2017.

[4] E. Zeng, S. Mare et F. Roesner, "End user security & privacy concerns with smart homes," dans *Symposium on Usable Privacy and Security (SOUPS)*, 2017.

[5] M. Meneganti, F. S. Saviello et R. Tagliaferri, "Fuzzy neural networks for classification and detection of anomalies," *IEEE transactions on neural networks*, vol. 9, n°. 5, p. 848–861, 1998.

[6] Z. Lin, Y. Shi et Z. Xue, "Idsgan : Generative adversarial networks for attack generation against intrusion detection," *arXiv e-prints*, 2018.

[7] W. Hu et Y. Tan, "Generating adversarial malware examples for black-box attacks based on gan," *arXiv preprint arXiv :1702.05983*, 2017.

[8] R. J. Lopez et R. J. Zhou, "On the features and challenges of security and privacy in distributed internet of things." *Computer Networks*, 2013.

[9] G. Loukas *et al.*, "Cloud-based cyber-physical intrusion detection for vehicles using deep learning." *IEEE Access*, n°. 6, p. 3491–3508, 2017.

[10] R. Sommer et V. Paxson, "Outside the closed world : On using machine learning for network intrusion detection," dans *Security and Privacy (SP), 2010 IEEE Symposium on.* IEEE, 2010, p. 305–316.

[11] P. M. et P. M. R., "Network intrusion detection using naive bayes," *International journal of computer science and network security, 7(12)*, 2007.

[12] I. Goodfellow *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, p. 2672–2680, 2014.

[13] A. Athalye *et al.*, "Synthesizing robust adversarial example," *arXiv preprint arXiv :1707.07397*, 2017.

[14] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network." *arXiv preprint*, 2017.

[15] J. Sadeeq et et al., “Automatic generation of tests to exploit xml injection vulnerabilities in web applications,” *IEEE Transactions on Software Engineering*, p. 335–362, 2017.

[16] C. M. *et al.*, “Search-based test and improvement of machine-learning-based anomaly detection systems,” *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, p. 158–168, 2019.

[17] H. G. Kayacik, A. N. Zincir-Heywood et M. I. Heywood, “Selecting features for intrusion detection : A feature relevance analysis on kdd 99 intrusion detection datasets,” *Proceedings of the third annual conference on privacy, security and trust*, 2005.

[18] M. Sabhnani et G. Serpen, “Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set,” *Intelligent data analysis*, vol. 8, n°. 4, p. 403–415, 2004.

[19] J. McHugh, “Testing intrusion detection systems : a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory.” *ACM Transactions on Information and System Security (TISSEC) 3(4)*, p. 262–294, 2000.

[20] I. Sharafaldin *et al.*, “Toward generating a new intrusion detection dataset and intrusion traffic characterization.” *ICISSP*, p. 108–116, 2018.

[21] ——, “Towards a reliable intrusion detection benchmark dataset.” *Software Networking*, p. 177–200, 2018.

[22] T. C. *et al.*, “Intrusion detection by machine learning : A review.” *Expert Systems with Applications 36(10)*, p. 11 994–12 000, 2009.

[23] I. Hisao et T. Murata, “Multi-objective genetic local search algorithm.” *Proceedings of IEEE international conference on evolutionary computation*, 1996.

[24] E. K. *et al.*, “Robust physical-world attacks on deep learning visual classification.” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, p. 1625–1634, 2018.

[25] O. I. S., L. J. S. et B. R. Moon, “Hybrid genetic algorithms for feature selection.” *IEEE Transactions on pattern analysis and machine intelligence, 26(11*, p. 1424–1437, 2004.

[26] G. Mitsuo et L. Lin, “Genetic algorithms.” *Wiley Encyclopedia of Computer Science and Engineering : 1-15*, 2007.

[27] M. Melanie, J. H. Holland et S. Forrest., “When will a genetic algorithm outperform hill climbing.” *Advances in neural information processing systems*, 1994.

[28] S. Prabavathy, K. Sundarakantham et S. Shalinie, “Design of cognitive fog computing for intrusion detection in internet of things.” *Journal of Communications and Networks 20(3)*, p. 291–298, 2018.

[29] D. L. et S. S. P., "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms." *International Journal of Advanced Research in Computer and Communication Engineering, 4(6)*, p. 446–452, 2015.

[30] A. Martin, S. Chintala et L. Bottou, "Wasserstein gan," *arXiv preprint arXiv :1701.07875*, 2017.

[31] U. N. LJ et et al., "Genetic local search algorithms for the traveling salesman problem." *International Conference on Parallel Problem Solving from Nature*, 1990.

[32] C. H. G. et J. J. Grefenstette, "Genetic algorithms for tracking changing environments," *Naval Research Lab Washington DC*, 1993.

[33] H. M. A. Mehedi et et al., "Support vector machine and random forest modeling for intrusion detection system (ids)." *Journal of Intelligent Learning Systems and Applications 6.01 : 45*, 2014.

[34] R. K. Yin, "Case study research : Design and methods." *Third Edition,3rd ed. SAGE Publications*, 2002.