Expérimentation et évaluation d'algorithmes de détection d'anomalies appliqués à des logs de proxy pour l'aide au *hunting*

Erwan Godefroy[1], Philippe Caparroy[1], Frédéric Majorczyk[1]
Loïc Cloâtre[2], Quentin Payet[2], and Cédric Hien[2]

[1] DGA-MI †, Bruz, France prenom.nom@intradef.gouv.fr
[2] CALID ‡, Rennes, France prenom.nom@intradef.gouv.fr

Résumé L'activité de *hunting* prend de plus en plus d'importance dans les SOCs (*Security Operation Center*). Cette activité se concentre sur l'analyse des logs émis par les systèmes d'information pour repérer les traces d'attaques, notamment celles d'attaques ciblées. En effet, ces attaques sont sophistiquées et les attaquants peuvent prendre le contrôle d'une partie du système d'information pendant de très longues périodes sans être détectés par les outils classiques de détection. Pour aider l'analyste dans cette activité, des chercheurs ont proposé l'utilisation d'algorithmes de détection d'anomalies permettant de trouver des éléments anormaux dans les logs. Cet article présente des travaux d'expérimentation et d'évaluation d'algorithmes de détection d'anomalies sur des logs de proxy. L'objectif est d'étudier la viabilité de ces techniques à la fois en termes de performances de calcul et de détection.

Keywords: détection d'intrusion, détection d'anomalies, analyse de logs

1 Introduction

Dans les systèmes d'information actuels, il est possible de stocker de plus en plus de types de logs différents et de les conserver sur des périodes plus longues. Dans le domaine de la sécurité, ces logs servent principalement deux objectifs : d'une part, après un incident de sécurité, leur analyse permet de déterminer le périmètre réel de l'attaque et de définir des indicateurs de compromission (IoC) ; d'autre part, ils sont également utilisés lors des activités de *hunting*, c'est-à-dire lors de la recherche des traces d'activités malveillantes qui ont échappé aux outils classiques de protection et détection.

De nombreux algorithmes d'analyse de données ont déjà été développés et appliqués dans le domaine de la sécurité informatique [1]. Dans les travaux décrits dans cet article, nous nous sommes intéressé à une catégorie spécifique d'algorithmes : les algorithmes de détection d'anomalies non-supervisés. Ce type d'algorithmes permet de s'affranchir d'une phase d'apprentissage nécessitant un

†. Direction Générale de l'Armement - Maîtrise de l'Information
‡. Centre d'Analyse et de Lutte Informatique Défensive

jeu de données labélisées. La production d'un tel jeu de données est un objectif difficile à atteindre de manière générale. D'une part, la fréquence des anomalies labélisées est trop faible pour permettre un apprentissage correct. D'autre part, chaque système est spécifique par son architecture et le choix des logs considérés.

Dans cet article, nous présentons une application de différents algorithmes de détection d'anomalies non-supervisés à des logs réels provenant du Ministère des Armées. Cette étude a pour but d'évaluer la viabilité de ce type de solutions pour un usage réel dans le cadre de l'activité de *hunting* du CALID. Nous nous sommes intéressés à la fois aux performances de calcul et aux performances de détection.

La suite de l'article est constituée de la manière suivante. La section 2 présente la méthodologie globale de conception, du choix des entités à classifier aux algorithmes choisis. La section 3 présente deux algorithmes de détection d'anomalies évalués dans cette étude. La section 4 présente l'expérimentation en elle-même : le jeu de données utilisé et l'architecture logicielle développée. La section 5 présente l'évaluation et les résultats de ces deux algorithmes de détection au cas d'usage. La section 6 discute des limitations de l'approche et présente les travaux futurs envisagés. La section 7 conlut l'article.

2 Méthodologie de conception

Avant d'appliquer des algorithmes de détection d'anomalies, il est nécessaire de définir l'objectif de détection et les données d'entrée des algorithmes. Dans notre cas d'usage, il n'est pas possible de modifier la configuration de la remontée de logs du système surveillé. L'objectif de détection et les données d'entrées vont donc être définis en fonction des logs actuellement générés et stockés.

La première étape consiste donc à référencer et analyser les types de logs disponibles pour déterminer une stratégie de détection en fonction des menaces recherchées. Dans notre exemple, nous disposons de journaux applicatifs issus d'équipements réseau ou de sécurité comme des serveurs mandataires, des serveurs DNS ou des pare-feux. L'objectif est de rechercher des indices de communications suspectes pouvant être causées par des malwares, comme, par exemple, un canal de communication avec un serveur de contrôle ou une exfiltration de données.

La deuxième étape consiste en la détermination des entités à classifier. Suivant le type de logs considérés, il peut s'agir d'un utilisateur, d'une machine ou d'un flux réseau par exemple. Dans notre cas d'usage, l'objectif est de découvrir des machines susceptibles d'être compromises, identifiées par leur adresse IP dans les logs.

Il est nécessaire de définir une période temporelle adaptée au rythme opérationnel des analystes et qui inclut la dynamique de fonctionnement du système. Il est ainsi possible de fixer la période temporelle dans une plage de valeurs allant d'une heure à une semaine dans notre cas d'usage.

L'étape suivante consiste à définir un ensemble de métriques (ou caractéristiques) décrivant au mieux une entité. Les caractéristiques sont regroupées en

différents types. Il peut s'agir de compteurs simples (par exemple, le nombre de lignes de logs ayant une valeur de champ spécifique), de compteurs dépendants de valeurs seuils (par exemple, la détection du nombre de pics de connexions), de ratio de valeurs entre deux grandeurs ou encore de métriques spécifiques (par exemple, l'entropie des valeurs d'un champ donné). Une entité est ainsi décrite par un vecteur de métriques. Dans notre exemple, il est nécessaire de calculer autant de vecteurs que d'adresses IP sources pour la période temporelle fixée.

L'étape suivante consiste à choisir et appliquer des algorithmes de détection d'anomalies dans le but de détecter les vecteurs les plus anormaux pour la période temporelle définie. Ces algorithmes prennent en entrée un ensemble de vecteurs (ou une matrice) et produisent un score qualifiant le degré d'anormalité de chaque entité (vecteur). Ces algorithmes ayant des propriétés de détection éventuellement différentes (un algorithme peut être mieux adapté à détecter certaines anomalies qu'un autre), il est possible de soumettre les vecteurs à différents algorithmes de détection d'anomalies. Chaque algorithme calcule un score d'anomalie pour chaque vecteur. Pour classer les vecteurs, il est donc nécessaire de déterminer une méthode pour combiner ces scores.

Enfin, il est nécessaire de définir une méthodologie d'évaluation des résultats. Étant donné que le jeu de données d'entrée n'est pas labellisé, il n'est pas possible de calculer les valeurs classiques de la matrice de confusion. Une méthode pour revenir à un cas plus classique est d'injecter des comportements correspondant à des intrusions connues dans les logs et d'analyser les résultats pour voir si l'entitée affectée obtient un score d'anomalie élevé.

3 Algorithmes de détection d'anomalies

Dans notre approche, plusieurs algorithmes de détection d'anomalies affectent des scores d'anomalie aux vecteurs de caractéristiques. Cette section détaille les principes de fonctionnement de deux des algorithmes utilisés pour affecter ces scores : la forêt d'isolation et l'auto-encodeur.

3.1 Forêt d'isolation

Parmi les différents algorithmes de détection d'anomalies, celui de la Forêt d'isolation [2] possède des caractéristiques remarquables qui ont motivé son choix pour notre étude. Il est simple, repose sur très peu d'hypothèses, peut très facilement être adapté aux paradigmes de programmation distribuée, et a prouvé son efficacité dans la détection d'anomalies [3]. L'algorithme de la forêt d'isolation débute par une phase d'apprentissage non supervisé, qui aboutit à la construction d'arbres binaires. Considérant un lot de données à traiter constitué de vecteurs de caractéristiques possédant N dimensions, un sous échantillon aléatoire de taille constante est sélectionné pour la construction de chaque arbre binaire. La génération des branches s'effectue en sélectionnant au hasard (loi uniforme) : une des dimensions $Xi \in \{X_1, X_2, ..., X_N\}$ de la matrice de données, puis une valeur de seuil v, choisie également au hasard entre le minimum et le

maximum observés pour cette caractéristique. Si un vecteur de caractéristiques possède une valeur inférieure ou égale à v, il est passé à la branche droite, sinon il est passé à la branche gauche. De cette manière, les données au niveau de chaque nœud de l'arbre en construction, sont réparties en deux partitions. Ce processus de génération des branches est effectué de manière récursive jusqu'à ce qu'un vecteur de caractéristiques se retrouve seul dans un nœud (feuille), ou qu'une profondeur limite soit atteinte. Ce processus est répété, en sous échantillonnant, sans remise, la matrice de données, afin de générer d'autres arbres constituant la forêt d'isolation. Pendant la phase de calcul des scores d'anomalies, on calcule pour chaque vecteur de caractéristiques la profondeur de sa trajectoire dans chaque arbre de la forêt d'isolation. A partir de la racine de l'arbre, et à chaque nœud rencontré sur son parcours, le vecteur est dirigé vers l'une ou l'autre des branches en comparant sa valeur de la caractéristique associée au noeud, au seuil de partitionnement précédemment décrit.

Nous avons implémenté une version distribuée de la forêt d'isolation, sous Apache Spark, les résultats présentés sont obtenus à l'aide d'une forêt de 100 arbres, construits chacun à l'aide d'un échantillon de 256 vecteurs de caractéristiques.

3.2 Auto-encodeur

Cette méthode, précédemment utilisée par différents auteurs [4,1] repose sur un processus de compression/décompression de l'information. Un réseau de neurones à plusieurs couches, comprime en réduisant le nombre de dimensions, puis reconstitue les vecteurs de caractéristiques. Cet apprentissage est non supervisé au sens où le réseau n'a pas à apprendre à différencier les vecteurs normaux des vecteurs anormaux, mais plus simplement à minimiser l'erreur de reconstruction. L'erreur de reconstruction pour le vecteur de caractéristiques i est donnée par :

$$e_i = \sum_{j=1}^{N} (x_j - r_j)^2 \tag{1}$$

La fonction de coût global minimisée lors de la phase d'apprentissage est la somme des e_i (définis dans l'équation 1) pour l'ensemble des vecteurs. De cette manière la majorité des vecteurs de caractéristiques, représentant statistiquement les comportements normaux, sont reconstruits correctement, alors que les vecteurs anormaux ne le sont pas. Le score d'anormalité est donc simplement l'erreur de reconstruction des vecteurs de caractéristiques (équation 1). Ce type de réseau neuronal, est un réseau multi-couches , à propagation antérieure, suivie d'une rétro propagation.

Comme décrit dans [1] , nous avons utilisé un réseau composé de trois couches cachées. Les couches d'entrée et de sortie ont le même nombre (N = 30) de dimensions, alors que les couches cachées ont un nombre réduit de dimensions. Les troisièmes et premières couches cachées ont N/2 (15) neurones, alors que la couche centrale est composée de N/4 (7) neurones. La fonction d'activation utilisée est de type rectification Linéaire (Relu), et le calcul de gradient est réalisé

par la méthode de Nesterov. Le taux d'apprentissage est dans un premier temps estimé par le biais d'une recherche en quadrillage à l'aide d'un nombre réduit d'itérations, puis une fois la valeur minimisant la fonction de coût identifiée, un nombre plus important d'itérations jusqu'à stabilisation de la valeur de la fonction de coût.

3.3 Normalisation

Au préalable à l'injection des vecteurs de caractéristiques dans les algorithmes précédemment décrit, on procède à leur normalisation de manière à équilibrer la contribution des différentes caractéristiques. Pour chaque caractéristique ($Xi \in \{X_1, X_2, ..., X_n\}$) , une valeur normalisée est calculée en considérant son maximum et son minimum observés :

$$x_n = \frac{x_i - min(x_i)}{max(x_i) - min(x_i)} \tag{2}$$

Les scores d'anormalité fournis comme résultats par les algorithmes doivent également subir un processus de régularisation / normalisation. Cet aspect est critique dans le but de bénéficier de l'effet cumulatif des différents algorithmes. Cette problématique, centrale à la détection d'anomalies, a été discutée en profondeur par différents auteurs (voir [5] pour une revue), et nous retiendrons la normalisation linéaire la plus simple, déformant au minimum les distributions de scores obtenus par chaque algorithme. De manière identique à l'équation 2, pour chaque algorithme de détection d'anomalie appliqué, le score d'anormalité d'un vecteur de caractéristiques est normalisé en prenant en compte les minimums et maximums du score considéré, sur l'ensemble de la population de vecteurs.

3.4 Combinaison des scores d'anomalies

Chaque algorithme produit ainsi un score d'anormalité avec une amplitude et une signification propre. La combinaison de ces scores d'anomalies en un indicateur unifié est réalisée en calculant le score d'anomalie moyen.

4 Expérimentation

Cette section décrit l'expérimentation menée en suivant la méthodologie présentée dans la section 2.

4.1 Description du jeu de données

Le jeu de données est un sous-ensemble des données collectées par le CALID, concrètement un extrait de logs de Splunk contenant les champs pertinents pour l'étude. Les logs considérés proviennent d'un proxy du Ministère et sont représentatifs d'une diversité de population utilisateur.

Cela représente environ 70 jours de logs au moment de l'écriture, avec en moyenne 10 Go de données journalières compressées en gzip. Pour cette expérimentation, cela correspond à environ 4To de données brutes avec plus de 10 000 adresses IP différentes et plus de 270 000 noms de domaines différents.

Pour être traités, ces logs doivent être regroupés en instances représentatives de l'activité d'une machine sur la période temporelle fixée, comme décrit dans la section 2.

4.2 Description des métriques

À partir de ces extraits de logs, des vecteurs caractéristiques sont calculés pour chaque IP source par période temporelle fixe de 6h. Les métriques que nous avons définies pour chaque IP peuvent être soit des métriques simples telles que des compteurs, soit des métriques plus complexes prenant en compte à la fois le passé et l'ensemble des IPs sources. Ce dernier cas est celui des métriques relatives à des pics (pics de connexions par exemple). Les métriques relatives à des pics restent des compteurs mais nécessitent de définir une sous-unité temporelle et de définir un seuil à partir duquel on considère qu'il y a un pic. Ce seuil ne peut pas être défini dans l'absolu : il est en effet dépendant du système et peut également changer au cours sa vie. Nous avons donc choisi de le calculer en fonction du passé.

Dans notre cas d'usage, la sous-unité temporelle choisie est 10 minutes. Le seuil est calculé pour chaque période temporelle de 6h à partir des logs des trois jours précédents de la manière suivante :

1. pour chaque sous-unité temporelle (10 minutes dans notre cas), on compte le nombre d'événements concernés (par exemple, le nombre de connexions) pour chaque IP des trois jours précédents ;
2. le seuil est défini comme le 9ème décile de l'ensemble des valeurs calculées.

La métrique (pour une IP pour 6h) correspond ainsi au nombre de sous-unités temporelles pour laquelle le nombre d'événements est supérieur au seuil.

Dans notre cas d'usage, plus d'une trentaine de métriques ont été définies dont voici quelques exemples :

— nombre de connexions HTTP utilisant la méthode POST ;
— nombre de pics de connexions HTTP utilisant la méthode POST ;
— nombre d'octets échangés ;
— nombre de de pics d'octets échangés ;
— nombre de connexions dont le code de retour est 2xx (resp. 3xx/4xx/5xx) ;
— nombre de connexions pour laquelle l'URL a une longueur relativement grande ou basse ;
— nombre de requêtes vers des domaines sur liste noire ;
— nombre de chaînes user-agents différentes.

4.3 Description de l'architecture

D'un point de vue matériel, le processus est déployé sur un serveur avec 256Go de RAM, 24 CPU et 9To de stockage sur une debian 9 kernel 4.9.0-6amd64(2018-03-02).

Les problématiques de normalisation, prétraitements et parsing des logs sont absorbées par des parseurs de la production. Il en résulte des extraits de journaux applicatifs journaliers qui sont transférés vers une instance de serveur Splunk installée sur le serveur. Ces journaux sont stockés dans un modèle de données accéléré pour permettre des calculs statistiques performants. Lors de cette phase, des champs additionnels sont précalculés pour des traitements ultérieurs.

Les vecteurs de caractéristiques sont ensuite créés via des requêtes adéquates vers l'instance du serveur Splunk. L'ensemble des requêtes est envoyé au serveur Splunk en utilisant l'API REST proposée et en parallèle ; le service Splunk se charge de l'ordonnancement des tâches. À la fin de cette opération, une matrice de vecteurs de caractéristiques est obtenue.

Cette matrice est ensuite traitée par une instance de Spark qui va appliquer en parallèle les deux algorithmes de détection d'anomalies aux vecteurs de caractéristiques. Les scores d'anomalies sont écrits dans des fichiers csv qui sont intégrés dans le serveur Splunk pour analyse des résultats.

5 Évaluation

L'évaluation des performances englobe quatre dimensions. La première est dédiée au temps d'exécution. La seconde partie consiste à évaluer les performances de détection à l'aide d'une injection contrôlée de traces malveillantes dans des données existantes. Une analyse de certaines anomalies présentes dans les données est ensuite réalisée. La dernière partie est dédiée aux problématiques de présentation des résultats à l'analyste.

5.1 Évaluation des performances d'exécution

Pour un ensemble de plus d'une trentaine de métriques, le temps de construction des vecteurs est compris entre onze et quinze minutes, et le temps d'exécution des algorithmes de détection d'anomalie est en moyenne de 10 minutes. En tout, pour des instances dont la résolution temporelle est 6h, le temps total de calcul est inférieur à une heure.

5.2 Évaluation des performances de détection avec des comportements malveillants synthétiques

L'objectif de nos travaux est de fournir une aide à l'analyse et à la décision. Le mécanisme mis en place est ainsi différent d'un IDS renvoyant des alertes lorsqu'un comportement malveillant est détecté. Dans nos travaux, nous voulons identifier les entités du réseau les plus intéressantes pour que les analystes puissent prioriser leurs investigations.

Toutes les adresses IP de la fenêtre temporelle de 6h se voient affecter un score d'anormalité, ce qui permet de les classer par degré d'anormalité décroissant. Ce classement est ensuite utilisé par les analystes pour prioriser leurs investigations en ne traitant qu'un nombre restreint d'activités d'adresses IP. La détection d'un comportement malveillant est ainsi considérée comme réussie lorque l'IP incriminée est dans l'ensemble des N éléments les plus anormaux sur au moins une fenêtre temporelle, N étant le nombre d'IP que les analystes peuvent traiter pour chaque fenêtre temporelle (ici 6h).

Il est également à noter que nous ne disposons pas de jeu de données labellisé pour évaluer l'approche. Cependant, la nature des données rend possible la création artificielle de comportements malveillants dont le degré de réalisme est contrôlable.

Injection de comportements malveillants synthétiques Un outil développé en interne permet de générer des lignes de logs correspondants à des comportements malveillants. Il permet de contrôler la structure des lignes de logs (valeur des champs et relation entre les champs notamment) ainsi que leur temporalité.

Chaque comportement malveillant est ensuite fusionné avec l'activité d'une adresse IP choisie aléatoirement dans la journée pendant laquelle l'activité malveillante débute. Ceci permet d'éviter de créer artificiellement une activité pour une IP qui est inactive. Les comportements malveillants sont donc fusionnés à l'activité quotidienne du système.

L'injection de comportements malveillants synthétiques dans le jeu de données a l'avantage de laisser un contrôle fin sur la nature et la temporalité de l'anomalie. Par contre, le risque est d'introduire des incohérences dans le jeu de données.

Par exemple, des lignes de logs dont la valeur du champ *user-agent* correspond à une valeur non utilisée ou incohérente par rapports aux données initiales. Un autre exemple est l'introduction de logs HTTP dont le code de retour est 200 alors que les URL associées seraient normalement inaccessibles ou bloquées par la politique de sécurité. Les spécifications des comportements malveillants ont été manuellement vérifiées pour éviter ces types d'incohérences.

Comportements malveillants considérés Nous avons choisi de générer un certain nombre de motifs malveillants présentés dans le tableau 1. Les durées, nombres d'événements ainsi que la quantité d'octets échangée sont présentés dans le tableau 2. Le tableau 5 présente également le rapport signal sur bruit du comportement malveillant. Ce rapport R est calculé avec la formule proposée par l'équation (3). A correspond au vecteur de caractéristiques d'une instance pour laquelle un comportement malveillant a été ajouté au comportement de base. B correspond au vecteur de caractéristiques de la même instance sans le comportement anormal.

$$R = \frac{\|A - B\|_2}{\|B\|_2} \tag{3}$$

Comportement malveillant	Description
post-flood	Attaque HTTP POST Flood, 1000 événements toutes les cinq secondes.
post-flood-2	Attaque HTTP POST Flood, 200 événements toutes les minutes.
malw-cc	Activité d'un malware (téléchargement du binaire et communications avec des CC)
url-exfilt	Exfiltration de données via une URL suspecte en une dizaine de minutes.
url-exfilt-2	Exfiltration de données via une URL suspecte en une demi-journée.
coms-dga	Contact avec de nombreux domaines DGA (moyenne d'un événement par seconde).
dga-exfilt	Exfiltration de données via une cinquantaine de domaines DGA.
fb-exfilt	Exfiltration régulière via la messagerie de Facebook (1 message par seconde).
pdf-downl	Téléchargement répété d'une ressource volumineuse pour saturer un site.

Table 1. Comportements malveillants générés

Ces comportements sont sélectionnés pour leur complémentarité en termes de dimensions impactées. Ainsi, le motif *post-flood* met en jeu l'utilisation intensive d'une unique methode HTTP vers un domaine populaire, alors que *url-exfilt* met en jeu une URL inconnue dans le système. Le motif *fb-exfilt* correspond à une exfiltration furtive utilisant un service connu et populaire alors que *dga-exfilt* met en oeuvre un ensemble de domaines inconnus pour réaliser une tâche similaire. La durée des comportements varie également de quelques minutes pour *url-exfilt* à plus d'une journée pour *coms-dga*.

Résultats Lorsqu'un comportement malveillant est réparti sur plusieurs périodes de 6h, les résultats sont alors présentés uniquement pour la période dans laquelle le comportement ressort le mieux. Ce choix se justifie car il est possible d'identifier l'acteur à l'origine du comportement malveillant à partir du moment où ce dernier est repéré sur au moins une tranche de 6h.

Comme les algorithmes décrits dans la section 3 possèdent une dimension aléatoire, nous avons répété l'expérimentation cinq fois en changeant la graine du générateur de nombres aléatoires. Le tableau 3 présente la moyenne des rangs globaux des IP pour lesquelles les comportements malveillants ont été ajoutés.

Comportement malveillant	Durée	Événements	Volumétrie
post-flood	53min	641k	19 Go
post-flood-2	8h	50k	1.5 Go
malw-cc	10h	30k	3.2 Go
url-exfilt	10min	34k	260 Go
url-exfilt-2	12h	45k	182 Mo
coms-dga	27h	100k	152 Mo
dga-exfilt	112min	118k	100 Go
fb-exfilt	12h	45k	151 Mo
pdf-downl	1h	244k	2 To

Table 2. Statistiques structurelles des comportements malveillants générés

Il s'agit du rang issu de la moyenne des probabilités d'anomalies, comme évoqué à la fin de la section 3.4. La troisième colonne indique le rang de cette même IP avant l'injection du comportement malveillant ; ceci permet de mesurer l'effet de l'injection sur le classement de l'IP. La quatrième colonne indique le nombre de périodes temporelles de 6h pour lesquelles l'IP concernée est classée dans les cinq premières IP au comportement le plus anormal par rapport au nombre total de périodes pendant lesquelles le comportement malveillant est présent.

Comportement malveillant	Moyenne du meilleur rang (± écart-type)	Moyenne du rang de référence (± écart-type)	Périodes dans le top 5
post-flood	2,67 (± 1,52)	1612 (± 42)	1/1
post-flood-2	3,45 (± 2,10)	22194 (± 761)	1/2
malw-cc	28,60 (± 8,90)	10214 (± 1237)	0/2
url-exfilt	2,33 (± 1,65)	14933 (± 243)	1/1
url-exfilt-2	13,78 (± 5,80)	6712 (± 1373)	0/3
coms-dga	1,56 (± 0,76)	7004 (± 901)	4/5
dga-exfilt	2,83 (± 2,31)	10288 (± 1009)	1/1
fb-exfilt	406,67 (± 7,23)	15990 (± 985)	0/3
pdf-downl	3,58 (± 2,73)	11588 (± 405)	1/1

Table 3. Classement des IPs affectées des différents comportements malveillants générés. Le rang de référence correspond au rang obtenu sans l'injection du comportement malveillant.

On peut noter, en observant l'écart-type de la moyenne du meilleur rang et celui de la moyenne du rang de référence, que la partie non-déterministe des algorithmes n'influence pas de manière significative le classement des IPs (ce qui est une propriété attendue des algorithmes). Le tableau 5 présente également deux mesures de la corrélation entre les classements de toutes les IPs sans injection du comportement et avec injection des comportements malveillants. Lorsque ces

valeurs sont supérieures à 0,7, la corrélation entre les deux classements est forte : les comportements malveillants que nous injectons pour certaines IPs spécifiques ne pertubent pas le classement des autres IPs.

Dans cette configuration et en supposant que seules les cinq premières IPs anormales peuvent êtres qualifiées par les analystes sur chaque période temporelle, six attaques sur les neuf sont détectées.

Comportement malveillant	Contributions relatives (%)	Métriques impactées
post-flood	19 19 19	octets sortants, requêtes POST, type text/*
post-flood-2	55 32 07	octets sortants, requêtes POST, type text/*
malw-cc	42 19 19	connexions à des sites catégorisées comme suspectes, octets totaux, octets entrants
url-exfilt	39 39 14	octets totaux, octets sortants, requêtes POST
url-exfilt-2	42 33 15	accès à des catégories suspectes, requêtes POST, type text/*
coms-dga	65 23 07	accès à des domaines non populaires, pics d'accès à des domaines non populaires, accès à des catégories suspectes
dga-exfilt	24 23 20	octets sortants, octets totaux, accès à des catégories suspectes
fb-exfilt	16 13 09	pics vers des URL longues, type application/*, requêtes POST
pdf-downl	28 28 20	octets totaux, octets entrants, type application/*

Table 4. Valeurs des trois contributions les plus importantes pour chaque comportement malveillant, avec les métriques associées à ces contributions (dans l'ordre de contribution). Ces contributions sont calculées à partir de l'autoencodeur.

Contribution des métriques au score d'anomalie Il est important de pouvoir remonter aux causes d'une anomalie afin d'identifier et de qualifier l'activité réelle d'une entité anormale. Une manière d'aider l'analyste dans la qualification d'une anomalie est de lui présenter les métriques qui ont le plus contribué au score d'anomalie d'une IP. Le tableau 4 référence les pourcentages de contribution des trois métriques qui contribuent le plus à chaque score d'anomalie. Pour deux des comportements malveillants, une dimension contribue à plus de la moitié du score d'anomalie (*post-flood-2* et *coms-dga*). Dans deux autres cas (*malw-cc* et *url-exfilt-2*), une métrique contribue à plus de 40% au score d'anomalie. À chaque fois, les métriques impactées sont cohérentes avec le type d'anomalie.

Lien entre le score d'anomalie et le rapport signal sur bruit Les résultats présentés dans le tableau 3 et 5 montrent, de manière assez naturelle, que la méthode est capable de faire ressortir des comportements malveillants dont le rapport signal sur bruit est élevé. Cependant, ce rapport n'est pas suffisant pour prédire la

Comportement malveillant	Rapport signal sur bruit	Moyenne du taux de Kendall	Moyenne du rho de Spearman
post-flood	33,9	0.81	0.89
post-flood-2	17,9	0.92	0.89
malw-cc	32	0.91	0.85
url-exfilt	142	0.95	0.85
url-exfilt-2	39,1	0.89	0.87
coms-dga	180	0.89	0.79
dga-exfilt	93,8	0.80	0.79
fb-exfilt	20	0.81	0.90
pdf-downl	162,8	0.89	0.79

Table 5. Rapport signal sur bruit et métriques de corrélation pour mesurer l'influence des différents comportements injectés sur les données.

détection ou non, comme le montre la comparaison des classements et rapports signaux sur bruit des IPs ayant les comportements malveillants *post-flood-2* et *fb-exfilt*. Ces deux comportements sont caractérisés par un rapport signal sur bruit relativement proche mais des rangs très différents. Pour un même rapport signal sur bruit, les métriques utilisées ne parviennent à capturer avec la même efficacité ce qui différencie le comportement malveillant (particulièrement pour le cas *fb-exfilt*) du bruit de fond. Ce résultat permet ainsi d'identifier une des raisons possibles de son mauvais classement.

5.3 Processus d'analyse

Les résultats des algorithmes de machine learning sont injectés dans Splunk pour permettre aux analystes du CALID de mener les investigations. L'objectif premier des investigations est opérationnel, i.e. déterminer si les IP les mieux classées présentent un comportement malveillant ou non. Un objectif secondaire est d'apporter des informations pouvant être prises en compte pour l'amélioration des méthodes de détection (voir la sous-section 6.2).

Une adresse IP pouvant être liée à des centaines de milliers d'événements sur une période de 6h, il est nécessaire d'apporter des informations de contexte à l'analyste pour l'aider à comprendre le score d'anomalie affecté à une IP. Le dashboard Splunk présenté en Figure 1 a donc été conçu pour répondre à ce besoin. Il est composé de 3 panneaux. Le premier panneau présente le classement des adresses IP. On voit les scores des 2 algorithmes utilisés. Le deuxième et le troisième panneau sont obtenus une fois une adresse IP sélectionnée. Ils permettent de connaître le classement de l'adresse sur une période large via un histogramme et de déterminer la part des différentes caractéristiques dans le score d'anomalie de l'IP via un graphique en secteur. Ces deux graphiques permettent à l'analyste de comprendre les raisons du classement de l'IP et voir l'évolution de ce classement au cours des jours/mois qui précèdent la qualification commencée. Ces trois panneaux permettent donc à un analyste d'avoir rapidement des informations nécessaires à l'étude du comportement de l'adresse IP.

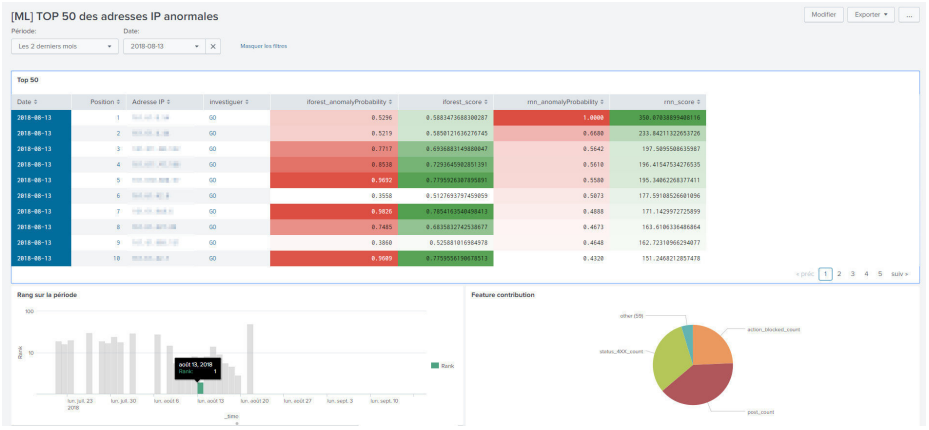

Figure 1. Dashboard pour la qualification des anomalies

5.4 Analyse de faux positifs

Une liste d'anomalies observées au CALID est présentée ci-dessous. Elle permet d'illustrer le fait que les algorithmes de machine learning mettent en avant tout type de comportement anormal. Cela implique forcément la génération de nombreux faux-positifs lors de la recherche de comportements malveillants, les ensembles des comportements anormaux et malveillants étant différents.

Analyse d'une machine classée cinquième : la caractéristique ayant le plus contribuée, à hauteur de 75% au score d'anomalie, est liée au visionnage de vidéo. Après investigation, il est apparu que l'utilisateur s'est rendu sur un site de vente de matériel sportif et qu'une page contenant une vidéo est restée ouverte toute la nuit. La vidéo a tourné en boucle et le trafic de la machine est apparu anormal pour les algorithmes.

Analyse d'une machine classée première : la caractéristique contribuant le plus est un taux d'exceptions important. Après investigation, il s'agit d'un utilisateur ayant probablement configuré une synchronisation de boîte universitaire avec son compte Gmail. C'est un problème d'authentification qui est à l'origine des nombreuses exceptions.

Analyse d'une machine observée pendant trois jours consécutifs en première position : la caractéristique contribuant le plus au score d'anomalie est lié à un nombre élevé de requêtes POST de type application. Après investigation, il s'agit d'un utilisateur qui était en conférence sur un exercice de défense.

6 Discussion

Même si les résultats présentés précédemment sont prometteurs, les évaluations ont tout de même montré que l'approche choisie présente des limitations pour détecter certains comportements. Cette section discute tout d'abord de certaines limitations de l'approche puis des travaux futurs que nous envisageons.

6.1 Limitations

Les résultats précédents montrent que les algorithmes permettent de détecter de manière relativement fiable des comportements malveillants ayant un rapport signal sur bruit élevé. Pour certains comportements malveillants où ce rapport est faible, l'IP affectée n'apparaît pas dans les cinq premières IPs et ne sera donc pas qualifiée par un analyste. Il est probable qu'avec les paramétrages actuels des comportements malveillants plus fins ne vont pas remonter suffisamment dans le classement. Pour vérifier ce fait, il est nécessaire de spécifier des comportements plus fins (en se basant sur les comportements de malware de Stratosphere IPS par exemple) et étudier les résultats en termes de rang. La sous-section suivante présente des techniques qui, à notre avis, vont permettre d'améliorer la détection de ces comportements.

Pour compléter notre étude, il est nécessaire de changer les méta-paramètres que nous avons choisis : période de temps, période historique pour le calcul des seuils de pics, ensemble de métriques. Les combinaisons entre ces différents méta-paramètres sont extrêmement nombreuses : il serait tout de même intéressant d'étudier les méthodes permettant de trouver les méta-paramètres optimaux (ou proches de l'optimal).

6.2 Travaux futurs

Cette étude confirme l'intérêt d'appliquer des algorithmes d'apprentissage non-supervisés à l'analyse des logs web, afin de détecter des comportements potentiellement malveillants. Elle amène cependant à mieux appréhender les limites des modèles de détection d'anomalies et il est nécessaire d'intégrer l'expertise des analystes au fil du temps dans ces modèles. Cet objectif principal étant fixé, nous envisageons différentes voies pour tenter de l'atteindre.

Active learning Les algorithmes non supervisés de détection d'anomalies, une fois déployés, sont principalement critiquables pour les taux élevés de faux positifs et faux négatifs qu'ils génèrent. Cette première limite résulte du fait que toute mesure déviante extrême (outlier) n'est pas nécessairement une anomalie, et que toute anomalie n'est pas nécessairement associée à un comportement malveillant. La principale manière de remédier à cette limite, est d'incorporer dans le système d'apprentissage, le retour d'expérience de l'analyste humain, afin de créer un cercle vertueux entre lui et les algorithmes d'apprentissage. Cette approche, globalement appelée Active Learning, utilise le retour d'expérience de l'analyste pour modifier les hyper paramètres du modèle d'apprentissage et augmenter la précision de détection. Ce couplage peut être effectué en associant un algorithme supervisé au système [1], celui-ci prenant en compte les labels affectés aux anomalies par l'analyste, puis corrigeant/pondérant les résultats des algorithmes non supervisés. Cependant, la très faible proportion de vrais positifs dans les données (1/10000 pour les logs de serveurs web) se traduit par un fort déséquilibre entre les effectifs des classes et pénalise l'apprentissage supervisé. Une approche plus récente, l'Active Anomaly Discovery (AAD) [6], permet

d'utiliser directement le feedback de l'analyste pour ajuster les paramètres des algorithmes non supervisés, et permet d'augmenter la proportion de vrais positifs au fil du temps, tout en respectant un effort limité et constant de validation par l'analyste. L'application de l'AAD, permet également de sélectionner des dimensions (caractéristiques) permettant de discriminer les différents types de comportements malveillants.

Dimensionnalité et sélection de caractéristiques Dans notre étude, le nombre de dimensions (30) est relativement réduit et les performances d'exécution des algorithmes sont bonnes. Il n'est donc pas nécessaire de chercher à réduire la dimensionnalité du problème. On pourra à l'inverse chercher à élargir le nombre de dimensions afin d'augmenter le spectre de détection du système. L'hyper-espace défini par ces dimensions/caractéristiques, est générique et des anomalies de types très distincts peuvent y être détectées. Donc, sélectionner les caractéristiques associées à un type d'anomalies permet nécessairement d'augmenter l'acuité de la détection. A cette fin, une méthode [7] nous semble particulièrement adaptée à la sélection de caractéristiques dans le contexte de notre approche. En effet, cette méthode est déterministe, indépendante du type d'algorithme, et aboutit de surcroît à la génération de règles simples, susceptibles d'aider l'analyste à mieux comprendre les anomalies et les labelliser.

Réduction des faux positifs Un autre aspect indispensable à l'amélioration de notre système, est l'incorporation d'un mécanisme permettant la réduction du nombre de faux positifs au fil du temps. Une méthode [8] adaptée à notre système a été récemment proposée et pourrait être expérimentée.

Dynamiques temporelles La détection d'anomalies, réalisée aux échelles de temps que nous avons proposées, pourrait rester insensible à des aspects de la dynamique temporelle autres que des dépassements de seuils de compteurs utilisés. En complément de l'examen statique de plages temporelles, on pourra chercher à appréhender plus finement la dynamique des séries temporelles de chaque caractéristique, et ce pour chaque entité du réseau surveillé. Des méthodes récentes [9] ont formalisé et validé la description des paramètres de séries temporelles à prendre en compte pour cela.

Autres algorithmes Enfin, la liste des algorithmes de détection d'anomalies choisis dans notre étude pourrait être enrichie d'autres candidats au profil complémentaire, tel Loda [10], ou la méthode des K plus proches voisins (KNN) [11]. Ces algorithmes sont très performants, peuvent être distribués, et sont capables de participer à la sélection de caractéristiques.

7 Conclusion

Dans cet article, nous avons présenté une expérimentation d'algorithmes de détection d'anomalies appliqués à des logs de proxy réels. Nous avons décrit la

méthodologie de conception, les algorithmes de détection choisis et l'implémentation pour notre cas d'usage. Les résultats en termes de performance d'exécution sont prometteurs pour un passage à l'échelle. L'évaluation des résultats de détection des algorithmes permet déjà de faire ressortir des comportements malveillants, même si ces derniers sont relativement bruyants. De nombreuses pistes telles que l'active learning, la sélection de caractéristiques ou la prise en compte des dynamiques temporelles sont envisagées pour améliorer les résultats de détection en termes de faux négatifs et de faux positifs.

Références

1. Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., Li, K. : Ai2 : Training a big data machine to defend. In : 2016 IEEE 2nd Int. Conf. on Big Data Security on Cloud (BigDataSecurity), IEEE Int. Conf. on High Performance and Smart Computing (HPSC), and IEEE Int. Conf. on Intelligent Data and Security (IDS). (April 2016) 49–54
2. Liu, F.T., Ting, K.M., Zhou, Z.H. : Isolation forest. In : 2008 Eighth IEEE Int. Conf. on Data Mining, IEEE (2008) 413–422
3. Susto, G.A., Beghi, A., McLoone, S. : Anomaly detection through on-line isolation forest : An application to plasma etching. In : Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th Int. Convention on, IEEE (2017) 89–94
4. Hawkins, S., He, H., Williams, G., Baxter, R. : Outlier detection using replicator neural networks. In : Int. Conf. on Data Warehousing and Knowledge Discovery, Springer (2002) 170–180
5. Schubert, E., Wojdanowski, R., Zimek, A., Kriegel, H.P. : On evaluation of outlier rankings and outlier scores. In : Proc. of the 2012 SIAM Int. Conf. on Data Mining, SIAM (2012) 1047–1058
6. Das, S., Wong, W.K., Dietterich, T., Fern, A., Emmott, A. : Incorporating expert feedback into active anomaly discovery. In : Data Mining (ICDM), 2016 IEEE 16th International Conference on, IEEE (2016) 853–858
7. Pevnỳ, T., Kopp, M. : Explaining anomalies with sapling random forests. In : Information Technologies–Applications and Theory Workshops, Posters, and Tutorials (ITAT 2014). (2014)
8. Grill, M., Pevnỳ, T., Rehak, M. : Reducing false positives of network anomaly detection by local adaptive multivariate smoothing. Journal of Computer and System Sciences **83**(1) (2017) 43–57
9. Hyndman, R.J., Wang, E., Laptev, N. : Large-scale unusual time series detection. In : Data Mining Workshop (ICDMW), 2015 IEEE Int. Conf. on, IEEE (2015) 1616–1619
10. Pevnỳ, T. : Loda : Lightweight on-line detector of anomalies. Machine Learning **102**(2) (2016) 275–304
11. Maillo, J., Ramírez, S., Triguero, I., Herrera, F. : knn-is : An iterative spark-based design of the k-nearest neighbors classifier for big data. Knowledge-Based Systems **117** (2017) 3–15