Détection d'Attaques Persistantes Avancées par Hachage et Apprentissage sur les graphes en Flux

Walid Megherbi, Abd Errahmane Kiouche, Mohammed Haddad, Hamida Seba

Détection d'Attaques Persistantes Avancées par Hachage et Apprentissage sur les graphes en Flux

Walid MEGHERBI*, Abd Errahmane KIOUCHE *
Mohammed HADDAD*, Hamida SEBA*

*Université de Lyon, Université Lyon 1, LIRIS UMR 5205 F-69622 France
{walid.megherbi,abd-errahmane.kiouche,mohammed.haddad, hamida.seba}@univ-lyon1.fr

Résumé. De nombreuses activités, notamment dans le domaine de la cybersécurité, peuvent être modélisées par des graphes dynamiques en flux, tels que les graphes d'appels. Dans ce travail, nous proposons une approche visant à détecter les attaques persistantes avancées (APT) dès leur commencement. Notre méthode se distingue par sa capacité à capturer à la fois l'information structurelle et temporelle, éléments clés pour différencier les activités normales des activités malveillantes. Pour répondre aux défis posés par le traitement en flux, nous nous appuyons sur des techniques de hachage, permettant d'obtenir une représentation compacte des données. Cette stratégie, combinée à une approche d'apprentissage automatique dynamique, offre une détection rapide et incrémentale tout en garantissant une faible consommation de mémoire. Les tests conduits démontrent l'efficacité de notre méthode, permettant une réponse proactive face aux menaces en identifiant les APT dès leurs premiers signes d'activité.

1 Introduction

Avec l'explosion du volume de données générées par les systèmes informatiques modernes, la capacité à extraire des connaissances pertinentes de ces données est devenue cruciale. Dans le domaine de la cybersécurité, cette extraction est particulièrement essentielle pour la détection des menaces persistantes avancées (APTs). Ces attaques, sophistiquées et ciblées, exploitent justement la complexité et la richesse des données pour se dissimuler, rendant leur détection en temps réel d'autant plus impérative pour protéger les informations sensibles. Dans ce contexte, la détection des APTs nécessite une analyse approfondie et continue des données, allant au-delà des approches traditionnelles de cybersécurité. Il s'agit non seulement de surveiller les signaux d'alerte, mais aussi d'interpréter les schémas subtils et les anomalies dans les flux de données, afin d'identifier et de contrer ces menaces avant qu'elles ne causent des dommages irréparables.
La modélisation par graphes offre une perspective unique pour capturer à la fois les aspects structurels et temporels des systèmes informatiques. Dans le contexte des APTs, cette dualité est essentielle. Les graphes permettent de cartographier les interactions et les comportements au sein des systèmes, offrant une représentation structurelle claire des activités. Les anomalies structurelles, telles que des connexions inhabituelles ou des motifs d'interaction atypiques,

peuvent ainsi être mises en évidence. Parallèlement, l'aspect temporel est tout aussi crucial. Les graphes en flux, en particulier, capturent la dynamique des interactions au fil du temps, mettant en lumière les anomalies temporelles, comme des séquences d'actions suspectes ou des changements brusques dans le comportement du système.
Les graphes d'appels, construits à partir des traces d'exécution des programmes, fusionnent ces deux dimensions. Ils tracent les appels de fonctions et leurs relations, tout en conservant l'ordre temporel de ces interactions, offrant ainsi une vue complète des activités potentiellement malveillantes. Cette capacité à détecter simultanément les anomalies structurelles et temporelles est fondamentale pour identifier les APTs, des menaces qui exploitent souvent des failles subtiles dans les deux domaines pour se dissimuler et progresser.
Cependant traiter les graphes en flux présente des défis majeurs, notamment en raison de la taille massive des flux de données et des contraintes de mémoire souvent limitées. Dans un environnement où les menaces évoluent rapidement, il est impératif de fournir des réponses en temps réel, voire instantanément. Cela nécessite des méthodes capables de traiter et d'analyser les données à la volée, sans compromettre la précision ou la pertinence de la détection. De plus, capturer simultanément les informations structurelles et temporelles est une tâche complexe. Chaque dimension apporte son lot d'informations, mais les intégrer de manière cohérente pour obtenir une vue complète des activités est un défi en soi. L'incrémentalité s'ajoute à cette complexité. Les systèmes doivent être capables d'assimiler de nouvelles données, de s'adapter aux changements et d'évoluer avec le temps, tout en conservant une mémoire des événements passés pour garantir une détection robuste et continue des APTs.
Les méthodes existantes pour la détection des APTs présentent plusieurs limites. Premièrement, les approches statiques, bien qu'efficaces dans certains contextes, s'avèrent inadaptées face à la nature dynamique des APTs.
De plus, la majorité des techniques actuelles se concentrent sur la détection d'anomalies au niveau des arêtes et des nœuds. Or, les APTs, de par leur sophistication, nécessitent une analyse au niveau du graphe en entier pour être correctement identifiées. En se limitant à une vision microscopique, ces méthodes passent souvent à côté des schémas d'attaque complexes. Enfin, bon nombre des solutions disponibles sont lourdes et coûteuses en termes de mémoire. Par exemple, les méthodes basées sur des modèles d'apprentissage profonds.Face aux limitations des méthodes existantes, nous proposons une nouvelle approche pour la détection des APTs qui capitalise sur la richesse des informations temporelles et structurelles contenues dans les données. Notre démarche s'appuie sur une modélisation fine des activités sous forme de graphes dynamiques en flux, permettant de capturer simultanément les nuances structurelles et les évolutions temporelles des activités. Cette représentation, combinée à des techniques d'analyse avancées, offre une perspective holistique pour détecter efficacement les APTs dès leurs premiers stades. Nos principales contributions et avantages sont :

— **Modélisation en flux :** Contrairement aux approches traditionnelles, notre méthode traite les données comme un flux continu, offrant une détection en temps réel tout en minimisant la consommation de mémoire.
— **Capture d'informations temporelles et structurelles :** Notre approche est conçue pour capturer à la fois les motifs structurels et temporels des anomalies, offrant une détection plus robuste des APTs sophistiquées.
— **Efficacité et scalabilité :** Grâce à l'utilisation de techniques de hashage, notre approche est à la fois compacte et rapide, permettant son déploiement même dans des environ-

nements avec des flux de données massifs.
— **Détection au niveau du graphe :** Plutôt que de se concentrer uniquement sur les anomalies au niveau des nœuds ou des arêtes, notre méthode analyse le graphe dans son ensemble, offrant une meilleure compréhension des schémas d'attaque complexes.

2 Détection anomalies dans les flux de graphes

La détection d'anomalies dans les flux de graphes est un domaine de recherche en plein essor, confronté à des défis uniques en raison de la nature dynamique et évolutive des graphes. Les données structurées sous forme de graphes présentent des complexités inhérentes. Les changements structurels, l'apparition de nouveaux nœuds et arêtes, et l'évolution des motifs au fil du temps rendent la détection d'anomalies particulièrement ardue. De plus, la nécessité d'opérer en temps réel, face à un afflux constant de nouvelles données, impose des contraintes de performance, d'efficacité et de mémoire. Dans cette section, nous passons en revue les principales approches et techniques utilisées pour détecter les anomalies dans les flux de graphes, tout en soulignant les défis associés.

Approches statiques : Ces méthodes, bien qu'efficaces dans des scénarios spécifiques, se concentrent principalement sur l'analyse d'un instantané du graphe à un moment précis. Elles tendent à négliger l'évolution dynamique et temporelle des relations au sein du graphe. L'analyse statistique des nœuds, par exemple, vise à établir un modèle de comportement normal et à identifier les observations qui s'écartent significativement de ce modèle comme étant anormales Yamanishi et Takeuchi (2002). D'autre part, les techniques basées sur le clustering, telles que celles présentées dans Pu et al. (2020); Ahmad et al. (2019), supposent que les anomalies ont tendance à se former dans des clusters moins denses ou à se situer à une distance significative des centroïdes de clusters. Les techniques de plongement de nœuds Grover et Leskovec (2016); Wu et al. (2020) (node embedding) ont gagné en popularité. Ces méthodes visent à représenter les nœuds sous forme de vecteurs dans un espace de faible dimension tout en préservant leurs propriétés structurelles et relationnelles. Ces représentations vectorielles peuvent ensuite être utilisées pour diverses tâches, y compris la détection d'anomalies. Cependant, ces approches peuvent manquer d'adaptabilité face à des graphes en constante évolution et ne parviennent pas toujours à détecter les motifs d'attaque sophistiqués caractéristiques des APTs.

Les approches de détection des arêtes anormales : Ces techniques sont utiles pour identifier des connexions inhabituelles comme des relations anormalement fortes ou faibles entre certains nœuds, ou des motifs de connexion qui se démarquent du comportement habituel du réseau. Néanmoins, elles manquent souvent de contexte temporel. Elles se concentrent principalement sur des motifs instantanés, tels que la détection de sous-graphes Eswaran et al. (2018); Yu et al. (2018) ou l'analyse de la fréquence des arêtes Yu et al. (2018); Chang et al. (2021). Cependant, leur sensibilité aux changements rapides peut les rendre moins efficaces pour détecter des menaces évoluant sur une période prolongée, d'où elles sont inefficaces pour détecter les APTs en temps réel.

Approches basées sur l'apprentissage profonds : Ces approches tirent parti de la capacité des réseaux de neurones profonds à apprendre des représentations hiérarchiques des graphes, capturant ainsi des motifs structurels et temporels complexes. Des architectures comme les Graph Neural Networks (GNN) ont été spécifiquement conçues pour traiter les données struc-

turées sous forme de graphes Liu et al. (2021); Wu et al. (2020). Cependant, malgré leur potentiel, ces méthodes basées sur l'apprentissage profond présentent des défis majeurs pour une utilisation en temps réel. La complexité computationnelle de l'entraînement et de l'inférence dans de tels modèles, en particulier dans des environnements dynamiques, peut rendre la détection en temps réel difficile. De plus, la capacité du modèle à s'adapter rapidement à de nouvelles données sans un ré-entraînement complet est souvent coûteuse en termes de temps et de ressources, limitant ainsi leur efficacité dans des scénarios où la rapidité de détection est cruciale.

Les approches de détection dynamiques incrémentales : Bien que certaines méthodes se concentrent sur l'analyse de flux de données, tandis que d'autres se basent sur des signatures ou des modèles de comportement. La majorité de ces méthodes sont statiques, ce qui signifie qu'elles s'appuient sur des règles préétablies. Cette approche peut s'avérer inefficace face à des attaques sophistiquées qui évoluent constamment, notamment dans les flux de graphes où le dynamisme des relations entre entités est crucial. C'est pourquoi l'exploration de méthodes dynamiques est impérative. Deux exemples notables sont les méthodes LEADS Lagraa et al. (2021) basé sur la distance d'édition incrémentale et StreamSpot Manzoor et al. (2016) basé sur stream text mining, des méthodes incrémentales qui s'adaptent continuellement aux nouvelles menaces, offrant ainsi une détection proactive des APTs.

3 Approche

La détection des APTs nécessite une surveillance rigoureuse des activités du système. Ces activités, souvent capturées à travers les journaux systèmes, sont illustrées dans la Figure 1. Pour une analyse plus approfondie, nous avons adopté la modélisation utilisée dans Manzoor et al. (2016). Dans cette modélisation, les graphes sont hétérogènes, avec des types d'arêtes correspondant à des appels système tels que "$read$", "$fork$", "$sock_wr$", etc., et des types de nœuds incluant "$socket$","$file$","$memory$", etc. Chaque arête est représentée sous la forme : <id-source, type-source, id-destination, type-destination, timestamp, type-arête, tag-flux>. Ces arêtes forment des graphes en évolution dynamique, où les arêtes ayant le même tag-flux appartiennent au même graphe. Plusieurs graphes peuvent évoluer simultanément. Nous avons opté pour une méthode basée sur la forêt d'isolation pour détecter les graphes anormaux, en maintenant une représentation incrémentale des graphes en mémoire et en introduisant une nouvelle mesure de similarité pour les comparer. Cette méthode, adaptée à notre contexte, permet une détection efficace et en temps réel des APTs en se basant sur les écarts par rapport au modèle des graphes bénins.

Les principaux composants de notre approche de Détection d'Anomalies par Hachage pour APTs, nommée « DAHAPT » [1], sont les suivants, et seront détaillés plus loin dans les sous-sections mentionnées :

1. **Plongement incrémental, rapide et compact de graphes en flux :** Dans cette étape, nous utilisons une technique d'incorporation de graphes pour transformer nos flux de graphes en une représentation vectorielle. À chaque arrivée d'une nouvelle arête, cette représentation est mise à jour de manière incrémentale, garantissant ainsi une actualisation rapide et efficace de nos données. Pour ce faire, nous nous appuyons sur une

1. https://anonymous.4open.science/r/EGC2024-C5DB/DAHAPT.py

Instant	Origine	Evénement	Cible	Tag
19/10/2023 08:23	IP: 192.168.1.10	Request Start	PID: 3075	AUTH-001
19/10/2023 08:25	PID: 3075	Token Generation	Memory: 0x3FFB23	AUTH-001
19/10/2023 08:27	Memory: 0x3FFB23	Token Validation	File: /usr/data	AUTH-001
19/10/2023 08:30	IP: 192.168.1.20	Connection Start	PID: 4156	NET-105
19/10/2023 08:32	IP: 192.168.1.20	Data Package Create	Memory: 0x3FFB30	NET-105
19/10/2023 08:35	Memory: 0x3FFB30	Data Transmit	File: /var/logs	NET-105
19/10/2023 08:37	File: /usr/data	Request End	IP: 192.168.1.10	AUTH-001
19/10/2023 08:43	File: /var/logs	Connection End	IP: 192.168.1.20	NET-105

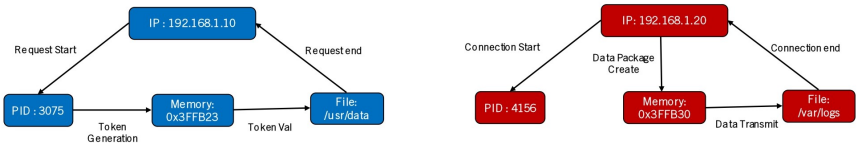


FIG. 1 – *Illustration de graphes d'appel.*

fonction de hachage universel. Cette fonction nous permet de calculer rapidement la similarité entre les flux de graphes, sans avoir à stocker l'intégralité des arêtes des graphes, ce qui rend notre approche à la fois rapide et peu gourmande en mémoire.

2. **Détection d'anomalie en temps réel :** Une fois notre représentation vectorielle mise à jour, chaque graphe est immédiatement évalué pour déterminer s'il est anormal ou non. Pour cette évaluation, nous utilisons la méthode de la forêt d'isolation (Isolation Forest). Ce modèle est particulièrement adapté à notre contexte car il est initialisé avec l'évolution des vecteurs de graphes bénins à tous les stades. Cela lui permet de détecter efficacement les anomalies dès leurs premiers signes, offrant ainsi une détection en temps réel des APTs.

3.1 Plongement incrémental de graphes en flux

Pour une détection en temps réel des APTs, il est impératif d'avoir une représentation qui non seulement capture la nature hétérogène des multi-graphes, mais aussi l'ordre temporel des arêtes. Cette représentation doit être conçue de manière à permettre des mises à jour en temps réel tout en utilisant un espace mémoire constant de complexité $O(k)$ où k est égale à la taille de la représentation vectorielle, même face à un flux continu et infini de nouvelles arêtes. Afin de répondre à cette exigence, nous introduisons une fonction de similarité pour les graphes qui prend en compte à la fois la structure locale des graphes et l'ordre temporel des arêtes. Cette fonction s'inspire des techniques de text mining, en particulier de la méthode des k-grammes, largement utilisée pour construire des représentations vectorielles des textes Yang et al. (2010). Cette dernière présente plusieurs avantages qui justifient notre choix. Elle est naturellement adaptée à la modélisation de données en flux, car elle permet de traiter des séquences d'événements de manière incrémentale. De plus, elle offre une flexibilité dans la prise en compte de l'ordre temporel des événements, ce qui est crucial pour capturer les motifs d'attaque dans les APTs.

La représentation des arêtes dans notre modèle est conçue pour capturer de manière efficace la structure et la dynamique des graphes d'activités. Initialement, chaque arête est définie par

une concaténation des types de nœuds source et destination, ainsi que du type d'arête elle-même. Pour enrichir cette représentation et offrir une meilleure compréhension de la structure sous-jacente des graphes, nous y intégrons un encodage spécifique. Cet encodage s'inspire d'études antérieures Bolton et Anderson-Cook (2017); Milajerdi et al. (2019) qui ont souligné la présence marquée de sous-structures denses, en particulier autour de nœuds à degré élevé, agissant souvent comme des points focaux d'activités coordonnées. Ces motifs d'activités, lorsqu'ils sont transcrits en séquences d'appels systèmes, dévoilent des structures récurrentes et interconnectées, ressemblant à des multi-étoiles.

L'encodage que nous adoptons se compose de trois symboles distincts : "0", "1" et "2". Le symbole "0" est utilisé pour dénoter la répétition de la même arête, évoquant la récurrence d'un appel système spécifique entre les mêmes entités, souvent indicatif d'une activité suspecte. Le symbole "1" signale un changement d'arête (une des entités d'appel) mais reste au sein de la même étoile, reflétant la variété des appels systèmes associés à une activité spécifique. Le symbole "2", quant à lui, indique une transition hors de l'étoile actuelle, suggérant un déplacement vers un autre nœud central. Par exemple, une série continue de "0" pourrait être indicative d'une répétition anormale d'un événement, comme des tentatives répétées d'authentification, évoquant un brut forcing. À l'inverse, une succession de "2" pourrait indiquer des mouvements fréquents entre différentes étoiles, suggérant des activités potentiellement malveillantes, complexes et coordonnées. Dans le contexte de la détection des APTs, cet encodage, basé sur un vocabulaire restreint, se révèle particulièrement efficace pour identifier des motifs d'activités malveillantes coordonnées et répétées, offrant ainsi une base solide pour la détection précise des APTs comme indiqué dans la Figure 2.

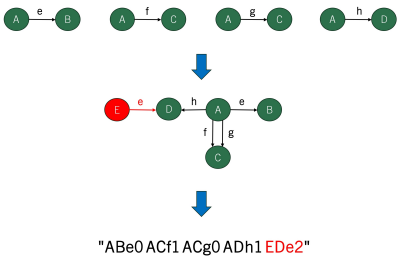


FIG. 2 – *Illustration de l'encodage utilisé*

La représentation des graphes sous forme de chaînes de caractères offre une opportunité unique de les modéliser comme des vecteurs d'histogrammes. Ces vecteurs, notés z_G, capturent la fréquence d'apparition de chaque sous-chaîne (ou mot) de longueur k dans la chaîne de caractères associée au graphe. La similarité entre deux graphes G et G' est alors mesurée par la similarité cosinus entre leurs vecteurs respectifs z_G et $z_{G'}$.

Un défi majeur de cette modélisation est que l'espace des sous-chaînes possibles est inconnu à l'avance et peut être extrêmement vaste. Cela rend difficile la construction des vecteurs dans un modèle de flux de données.Pour surmonter ce défi, nous employons la technique du *Locality-Sensitive Hashing (LSH)* Indyk et Motwani (1998); Dasgupta et al. (2011). LSH est une méthode pour hacher des données de manière à ce que la probabilité de collision soit maximale pour les données qui sont similaires selon une certaine mesure de distance. Dans le

contexte de nos graphes représentés par des chaînes de caractères, chaque sous-chaîne de longueur k peut être considérée comme un point dans un espace métrique. Nous utilisons LSH pour hacher ces sous-chaînes de manière à préserver leur similarité cosinus.

Considérons l'espace des sous-chaînes comme $\mathbb{U}$. Une famille $\mathcal{H}$ de fonctions de hachage est utilisée pour mapper $\mathbb{U}$ dans un ensemble fini de compartiments de hachage. Cette famille est dite universelle si, pour chaque paire distincte de sous-chaînes, la probabilité qu'elles soient mappées dans le même compartiment est proportionnelle à leur similarité cosinus.

Lorsqu'une nouvelle arête arrive dans le graphe, la sous-chaîne associée w_e est hachée en utilisant une fonction de hachage h tirée de $\mathcal{H}$. Cette fonction est choisie de manière à ce que les sous-chaînes similaires soient mappées dans le même compartiment avec une probabilité élevée, facilitant ainsi l'estimation de la similarité cosinus dans un contexte de flux de données. Le vecteur d'histogramme z_G est mis à jour incrémentalement à chaque arrivée d'une nouvelle arête, en utilisant l'indice de compartiment obtenu par le hachage de la sous-chaîne associée :

$$z_G[h(w_e)] = z_G[h(w_e)] + 1$$

Choisir h : Une famille satisfaisant certaines propriétés est dite fortement universelle Wegman et Carter (1981). Nous adoptons la famille multilinéaire fortement universelle pour les chaînes Lemire et Kaser (2014). Dans cette famille, la chaîne d'entrée w est divisée en k composants (c'est-à-dire "caractères") comme $s = c_1c_2...c_{|k|}$. Dans le contexte du hachage, s est envisagée comme une séquence de coefficients c_i d'un polynôme modulo un grand nombre premier p. Afin de hacher cette chaîne, un nombre aléatoire a est sélectionné uniformément à partir de l'ensemble [p]. La méthode de hachage pour la chaîne se base alors sur la formule suivante :

$$h(s) = \sum_{i=0}^{k} c_i \cdot a^{k-i} \mod p$$

Cette formule permet de transformer la chaîne s en une valeur de hachage en utilisant ses coefficients c_i et le nombre aléatoire a. La mise à jour incrémentale permet de maintenir une représentation compacte et efficace des graphes en flux, tout en capturant leurs caractéristiques essentielles pour des applications telles que la détection des APTs. La propriété clé du LSH est que la similarité estimée en utilisant les hachages est une approximation proche de la similarité réelle, avec des garanties probabilistes.

3.2 Détection d'anomalie en temps réel

La détection en temps réel des APTs nécessite une évaluation immédiate de chaque graphe dès son arrivée. Grâce à notre représentation vectorielle, chaque graphe est transformé en un vecteur, comme décrit dans l'algorithme 1. Une fois cette transformation effectuée, l'étape suivante consiste à évaluer si le graphe est anormal ou non.
Pour cette évaluation, nous avons choisi d'utiliser la méthode de la forêt d'isolation (Isolation Forest). Cette méthode a été préférée pour plusieurs raisons. Tout d'abord, elle est intrinsèquement conçue pour détecter les anomalies dans les données. Elle fonctionne en isolant les observations anormales dans des arbres de décision. Les observations anormales nécessitent généralement moins d'étapes pour être isolées, ce qui les rend rapidement identifiables.

Algorithm 1 Détection des APTs en utilisant la représentation des graphes

```
1:  procedure DÉTECTERAPT(G, k)                ▷ G est le graphe d'activité, k est la taille des k-grammes
2:      z_G ← InitVecteurHist()
3:      chaine ← ""
4:      for chaque arête e dans G do
5:          w_e ← Encoder(e)                   ▷ Représentation de l'arête sous forme de chaîne
6:          encodage ← ObtenirEncodage(e, G)   ▷ Obtenir l'encodage
7:          chaine ← chaine + w_e + encodage
8:          if longueur(chaine) ≥ k then
9:              index ← Hacher(chaine)         ▷ Utiliser LSH pour obtenir un indice
10:             z_G[index] ← z_G[index] + 1
11:             chaine ← ""
12:         end if
13:         anomalie ← IsolationForest(z_G)
14:         if anomalie then
15:             return "Activité malveillante détectée"
16:         else
17:             return "Activité normale"
18:         end if
19:     end for
20: end procedure
```

De plus, la forêt d'isolation est adaptée à notre contexte car elle peut être initialisée avec des vecteurs de graphes considérés comme bénins à différents stades de leur évolution. Cela signifie que le modèle est constamment mis à jour avec les représentations les plus récentes des graphes normaux, lui permettant de s'adapter aux évolutions normales du système tout en restant sensible aux anomalies.
En combinant notre représentation vectorielle avec la forêt d'isolation, nous sommes en mesure de détecter efficacement les APTs en temps réel. Dès qu'un graphe est considéré comme anormal par la forêt d'isolation, une alerte est générée, permettant une intervention rapide pour contrer la menace potentielle.

4 Évaluation

Jeux de données. Nous avons utilisé les jeux de données Subtreamspot Manzoor et al. (2016). Ces derniers sont constitués de flux de graphes d'appel issus de 2 attaques et 5 activités bénignes. Les scénarios bénins concernent une activité de navigation normale, notamment le visionnage de vidéos sur YouTube, le téléchargement de fichiers, la navigation sur cnn.com, la consultation de Gmail et d'un jeu vidéo. Les attaques impliquent un téléchargement non sollicité déclenché par la visite d'un URL malveillante qui exploite une vulnérabilité Flash, ainsi que l'exécution d'un fichier malveillant JAR. Les graphes de flux ont été compilés en 3 jeux de données : ALL, YDC et GFC, contenant toutes les activités. Nous avons veillé à ce que les graphes bénins et anormaux soient de même ordre de taille. Cette démarche a été adoptée pour éliminer toute différence évidente entre les graphes et rendre la détection plus difficile, reflétant ainsi un scénario plus réaliste. De plus, le taux d'anomalies dans chaque jeu de données est très faible, ne dépassant pas 5 %. Cette proportion a été choisie pour s'approcher de la réalité, où les activités malveillantes sont souvent noyées dans un volume massif d'activités normales.

Références de base (Baseline). Nous avons confronté notre méthode à deux approches existantes : LEADS et Streampot. Ces deux méthodes ont été choisies car elles sont parmi les

rares qui peuvent être directement appliquées à notre problématique. Il est important de souligner que de nombreuses méthodes basées sur l'apprentissage automatique profond conçues spécifiquement pour les graphes dynamiques ne sont pas adaptées à notre contexte. La raison principale est leur coût computationnel élevé. En effet, ces méthodes, bien qu'efficaces dans de nombreux scénarios, s'avèrent très coûteuses en termes de temps de calcul, ce qui les rend inadaptées à une détection en temps réel, surtout lorsque cette détection doit être effectuée à chaque arrivée d'une nouvelle arête. Dans notre cas, la rapidité et la réactivité sont essentielles, et toute méthode qui ne peut pas fournir de résultats en temps réel serait inadéquate.

Paramètres expérimentaux : Nous avons testé notre méthode dans deux configurations en calculant les métriques d'évaluation : AUC, BACC, F1, PR, FPR, FNR et la vitesse de traitement.

— **Statique :** Utilisant 25% des graphes bénins pour l'entraînement et le reste pour les tests. Les graphes sont représentés par leurs représentations vectorielles.

— **En flux :** 25% des graphes bénins servent à un entraînement initial. Les graphes de test sont ensuite traités en temps réel, une arête à la fois.

Résultats : Nous commençons par l'évaluation dans le cadre statique, où nous évaluons les activités à la fin du flux, une fois que tous les événements ont été observés. Cela nous permet d'évaluer la performance des méthodes dans des conditions idéales. Le tableau ci-dessous 1 présente les résultats obtenus. Les résultats montrent que DAHAPT offre une légère amélioration par rapport aux méthodes LEADS et Streamspot. Cependant, il est important de noter que les performances des trois approches sont assez similaires, indiquant que chacune d'elles est capable de détecter efficacement des activités malveillantes dans des scénarios variés. Ces résultats sont très satisfaisants, reflétant la robustesse et l'efficacité de notre méthode.

TAB. 1 – *Étude comparative dans le cas statique*

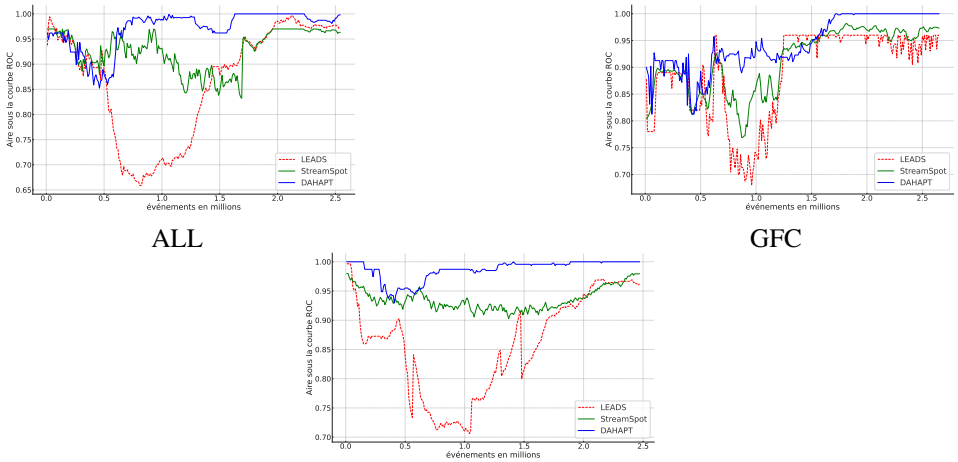
		AUC	BACC	PR	F1	FPR	FNR
ALL	DAHAPT	**98,73%**	**97,00%**	**95,00%**	**97,44%**	**6,00%**	**0%**
	LEADS	97,30%	96,00%	94,70%	97,28%	8,00%	**0%**
	Streamspot	96,3%	95,48%	94,60%	97,24%	9,00%	**0%**
YDC	DAHAPT	**100,00%**	**98,20%**	**97,30%**	**98,64%**	**3,60%**	**0%**
	LEADS	96,00%	95,10%	94,50%	97,22%	9,80%	**0%**
	Streamspot	98,00%	96,70%	95,40%	97,66%	6,60%	**0%**
GFC	DAHAPT	**100,00%**	**98,10%**	**97,20%**	**98,58%**	**3,80%**	**0%**
	LEADS	96,00%	95,10%	94,80%	97,38%	9,80%	**0%**
	Streamspot	97,00%	95,40%	95,10%	97,44%	9,20%	**0%**

Évaluation dans le cadre dynamique :

Après avoir examiné les performances dans un contexte statique, nous nous tournons maintenant vers l'évaluation de ces méthodes dans un environnement dynamique, où les graphes sont analysés et évalués en temps réel à mesure que de nouvelles arêtes arrivent. En contexte dynamique, la vitesse de traitement est cruciale. StreamSpot traite environ 25 000 événements par seconde, tandis que LEADS et DAHAPT sont plus rapides, traitant en moyenne 60 000 événements par seconde, ce qui les rend adaptées pour le traitement en temps réel.

Les trois courbes 3 présentées illustrent l'évolution de l'AUC en fonction du nombre d'événements observés. Sur l'axe des abscisses (x-axis), nous avons le nombre d'événements observés, exprimé en millions, tandis que sur l'axe des ordonnées (y-axis), nous avons l'AUC. Cette visualisation permet de comprendre comment la performance de détection évolue à me-

FIG. 3 – *Étude comparative dans le cas dynamique*



ALL GFC

YDC

sure que davantage d'événements sont observés et traités.L'analyse des courbes d'évaluation révèle plusieurs observations clés. D'emblée, il est manifeste que notre approche surpasse les autres, cette supériorité devenant particulièrement évidente après le premier tiers des événements observés. Au commencement du flux, toutes les méthodes affichent des performances assez similaires, avec un AUC supérieur à 80%. Cette convergence initiale peut s'expliquer par le fait qu'au début du flux, les activités anormales semblent se mêler aux activités bénignes, rendant la distinction entre elles délicate. Cela pourrait également suggérer qu'à ces stades précoces, les anomalies n'ont pas encore pleinement émergé, ce qui rend leur détection plus ardue.

Cependant, à mesure que le flux progresse, une divergence se manifeste. Notre méthode commence à identifier les anomalies à des stades précoces, atteignant un AUC impressionnant de plus de 90% dans les trois jeux de données. Une amélioration notable des performances est observée pour StreamSpot également après le premier tiers du flux. Notre méthode, en particulier, montre une avance nette sur les trois jeux de données. En revanche, la méthode LEADS ne parvient à améliorer ses performances et à détecter efficacement les anomalies qu'à partir du dernier tiers du flux. Ces observations soulignent la capacité de notre approche à détecter efficacement les anomalies à des stades bien plus précoces que les méthodes concurrentes.

Cette performance témoigne de l'efficacité de l'encodage que nous avons proposé, qui parvient à capturer à la fois les patterns temporels et structurels des anomalies des APTs. Cette observation souligne la capacité de notre approche à détecter efficacement les anomalies à des stades bien plus précoces que les méthodes concurrentes. En conclusion, ces résultats confirment la robustesse et la pertinence de notre approche pour la détection en temps réel des APTs.

5 Conclusion

Au terme de cette étude, nous avons mis en avant une méthode novatrice pour la détection en temps réel des APTs, offrant des performances supérieures par rapport aux méthodes existantes. L'efficacité de notre approche repose sur une représentation riche des graphes d'activités, capturant à la fois les motifs structurels et temporels. Cependant, l'évolution rapide du paysage des menaces nous pousse à envisager des améliorations futures. L'intégration des techniques d'apprentissage profond, notamment les RNN ou les Transformers, pourrait offrir une capacité prédictive, permettant d'anticiper les événements malveillants avant qu'ils ne surviennent. En fusionnant notre approche actuelle avec ces méthodes avancées, nous aspirons à développer un système de détection des APTs encore plus robuste et prédictif.

Remerciements

Ces travaux sont soutenus par l'Agence Nationale de la Recherche (ANR) dans le cadre du projet GLADIS ANR-20-CE39-0008.

Références

Ahmad, B., W. Jian, Z. A. Ali, S. Tanvir, et M. S. A. Khan (2019). Hybrid anomaly detection by using clustering for wireless sensor network. *Wireless Personal Communications 106*, 1841–1853.

Bolton, A. D. et C. M. Anderson-Cook (2017). Apt malware static trace analysis through bigrams and graph edit distance. *Statistical analysis and data mining : the ASA data science journal 10*(3), 182–193.

Chang, Y.-Y., P. Li, R. Sosic, M. Afifi, M. Schweighauser, et J. Leskovec (2021). F-fade : Frequency factorization for anomaly detection in edge streams. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 589–597.

Dasgupta, A., R. Kumar, et T. Sarlós (2011). Fast locality-sensitive hashing. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1073–1081.

Eswaran, D., C. Faloutsos, S. Guha, et N. Mishra (2018). Spotlight : Detecting anomalies in streaming graphs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1378–1386.

Grover, A. et J. Leskovec (2016). node2vec : Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864.

Indyk, P. et R. Motwani (1998). Approximate nearest neighbors : towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613.

Lagraa, S., K. Amrouche, H. Seba, et al. (2021). A simple graph embedding for anomaly detection in a stream of heterogeneous labeled graphs. *Pattern Recognition 112*, 107746.

Lemire, D. et O. Kaser (2014). Strongly universal string hashing is fast. *The Computer Journal 57*(11), 1624–1638.

Liu, Y., S. Pan, Y. G. Wang, F. Xiong, L. Wang, Q. Chen, et V. C. Lee (2021). Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*.

Manzoor, E., S. M. Milajerdi, et L. Akoglu (2016). Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1035–1044.

Milajerdi, S. M., R. Gjomemo, B. Eshete, R. Sekar, et V. Venkatakrishnan (2019). Holmes : real-time apt detection through correlation of suspicious information flows. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1137–1152. IEEE.

Pu, G., L. Wang, J. Shen, et F. Dong (2020). A hybrid unsupervised clustering-based anomaly detection method. *Tsinghua Science and Technology 26*(2), 146–153.

Wegman, M. N. et J. L. Carter (1981). New hash functions and their use in authentication and set equality. *Journal of computer and system sciences 22*(3), 265–279.

Wu, Z., S. Pan, F. Chen, G. Long, C. Zhang, et S. Y. Philip (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems 32*(1), 4–24.

Yamanishi, K. et J.-i. Takeuchi (2002). A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 676–681.

Yang, Z., J. Yu, et M. Kitsuregawa (2010). Fast algorithms for top-k approximate string matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 24, pp. 1467–1473.

Yu, W., W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, et W. Wang (2018). Netwalk : A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2672–2681.

Summary

Many activities, especially in the field of cybersecurity, can be modeled using dynamic stream graphs, such as call graphs. In this work, we propose an approach aimed at detecting Advanced Persistent Threats (APTs) from their inception. Our method stands out for its ability to capture both structural and temporal information, which are key elements in distinguishing normal activities from malicious ones. To address the challenges posed by streaming processing, we rely on hashing techniques to obtain a compact representation of the data. This strategy, combined with a dynamic machine learning approach, provides rapid and incremental detection while ensuring low memory consumption. The conducted tests demonstrate the effectiveness of our method, enabling a proactive response to threats by identifying APTs at the earliest signs of activity.