N° d'ordre : .....

UNIVERSITE DE M'SILA

FACULTE DE MATHEMATIQUES ET D'INFORMATIQUE

Département d'Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de Master

Domaine : Mathématiques et Informatique

Filière : Informatique

Option : Système d'Information Avancée

Par :

CHIKOUCHE Soumia

SUJET

Système de détection d'intrusion basé sur la classification comportementale des processus

Soutenu publiquement le : 28/06/2012 devant le jury composé de :

Mr. MEHANI Taher	**Université de M'sila**	**Président**
Mr. BOUHOUITA GUERMECH Salah Eddine	**Université de M'sila**	**Rapporteur**
Mr. BELHAJ Foudil	**Université de M'sila**	**Rapporteur**
Mr. ASSAS Ouarda	**Université de M'sila**	**Examinateur**

Promotion : 2011/2012

Remerciements

Je remercie Allah le tout puissant de m'avoir donné le courage jusqu'à l'achèvement de ce mémoire.

Au terme de ce travail, J'adresse ma profonde gratitude à Monsieur BOUHOUITA GUERMECH Salah Eddine. Je le remercie pour l'aide inestimable, la disponibilité, la compréhension, la gentillesse, les conseils et les encouragements avec lesquelles il a bien voulu diriger ce travail, il était vraiment responsable, compétant et patient. J'ai eu le grand plaisir de travailler sous sa direction.

Je remercie également Mr Belhadj FOUDIL, pour avoir accepté de suivre notre projet. J'espère être digne de la confiance qu'ils ont placée en moi.

Un grand merci à tous nos enseignants pour toutes les connaissances qu'ils nous ont inculquées tout au long des cinq années.

Mes remerciements s'adressent également à tous les membres du jury pour l'immense honneur qu'ils nous font en acceptant d'évaluer ce modeste travail.

Mes remerciements vont aussi à tous ceux qui ont contribué de près ou de loin à la concrétisation de ce travail pour leurs conseils, leurs encouragements et leurs soutiens. Qu'ils trouvent tous ici l'expression de ma gratitude et ma parfaite considération spécialement MR. OUALI Ahmed.

Dédicaces

Que nulle dédicace ne puisse exprimer ce que je dois, pour votre bienveillance, votre affection et votre soutien …en ce jour solennel, je voudrais renouveler mon indéfectible attachement à mes très chers parents, dont l'amour, la générosité et l'attention ne m'ont jamais fait défaut. Votre volonté de me voir réussi a été pour moi un stimulant tout au long de mes études.

A mes très chères sœurs Aicha, Meriem, Ouafa, Bouchera, Iman.

A mes grands-parents, je vous souhaite une longue vie INCHALLAH. A mes tantes et oncles et toute ma famille de proche grande ou de loin.

A toutes mes amies avec qui j'ai partagé les plus beaux et inoubliables moments de ma vie : Siham, Fadhila, Asma, Khaoula et Meriem.

Je n'oublierai pas pour autant, tous ceux qui sont absents sur cette feuille mais toujours présents dans mon cœur.

Toute personne qui m'a aidé un jour à réussir jusqu'ici

Soumia

Résumé :

L’intrusion est toute violation de la sécurité d’un système informatique. Le mécanisme d’audit de sécurité, consistant en l’enregistrement de tout ou partie des actions effectuées sur le système pour fournit des données permettant de déterminer la manière dont la sécurité du système a été compromise en posteriori. Pour cela, deux approches sont actuellement utilisées : l’approche comportementale et l’approche par scénarios. Nous utiliserons ces deux approches avec les algorithmes de fouilles de données appliqués sur les données fournies par le projet DARPA 98. La donnée de base utilisée est l’appel système où l’apport d’information de cette dernière a été mesuré suivant la théorie de l’information. Nous procédons, ensuite, à une analyse des résultats qui nous permet de situer les méthodes l’une par rapport à l’autre. Nous nous sommes principalement investis dans l’étude du KNN ainsi que la méthode bayésienne et leurs variantes.

Mots-clés : HIDS, DARPA 98, théorie de l’information, KNN, IF-IDF, réseau bayésien naïf, TAN, FAN, détection comportement, détection par scénario et python.

Abstract:

The intrusion is any breach of security of a computer system. The security audit mechanism, consisting of the recording of all or part of the actions performed on the system to provide data to define how the security system was compromised in posteriori. For this, two approaches are currently used: misuse intrusion detection and the anomaly intrusion detection. We will use these two approaches with data mining algorithms applied to data set from the project of DARPA 98. The basic data is the system call where the gain of information was measured according to the theory of information. We proceed then to analyze the results which allow us to compare the methods. We mainly invested in the study of KNN and the Bayesian method and their variants.

Keywords: HIDS, DARPA 98, theory of information, KNN, IF-IDF, naive Bayesian network, TAN, FAN, misuse detection, anomaly detection and python.

Sommaire

Chapitre 2 : Données d'apprentissage

Chapitre 3 : L'algorithme KNN

Table des matières :

Les figures :

Les exemples :

Les algoritmes :

Les tableaux :

Les courbes:

Introduction générale :

Au commencement, c'était de simples réseaux conçus spécifiquement pour le partage de ressources particulières telle l'imprimante, parmi un certain nombre d'ordinateurs, qui à leur tour peuvent inter-échanger des informations. Les réseaux de l'époque, avaient a assuré ces services de base, au sein d'une organisation unique, telle que les universités, les compagnies et les bases militaires…etc. Les réseaux connectaient des utilisateurs de la même organisation, d'où la nécessité de sécuriser les transmissions n'était pas sensible.

La venue d'Internet, et son utilisation universellement émergeante, et sa politique démocratique permettant à toute machine d'être connectée au réseau, et à toute personne d'en bénéficier ainsi que de proposer ses propres services. Cette vision d'échange libre de l'information impliquant un accès aussi libre, nous laisse à réfléchir sur la nature de l'information elle-même, sur plusieurs aspects, l'un des plus importants et surement crucial est le degré de confidentialité de cette information. Une confidentialité qui devrait être garantie avec un contrôle rigoureux des accès à cette information, et les opérations engagées. Afin d'empêcher les accès non-autorisés ou les opérations non-définies, une politique de sécurité est recommandée, pour cette fin, plusieurs stratégies et protocoles sécuritaires ont été élaborés. Parmi les solutions proposées, la mise en place des systèmes de détection d'intrusions est devenue nécessaire. Cependant, la complexité croissante des attaques nécessite une amélioration de telles techniques de détections.

Les activités d'intrusion peuvent s'exercer au niveau du réseau comme au niveau des machines hôtes, et donc les méthodes et outils de défense devront être engagés à ces mêmes deux niveaux précédents, d'où on parle de systèmes de détection d'intrusion basés réseaux et ceux basés hôtes. Notre projet traite les systèmes de détection basés hôte, qui représente en fait, une couche de défense.

L'IDS est de deux natures, celui qui reconnait l'attaque par un modèle d'action et, celui qui reconnait le comportement de l'activité intrusive. Un comportement est assimilé progressivement, est donc demande des techniques possédant la capacité d'apprentissage, d'où l'implication des algorithmes d'apprentissage dans le processus du design de l'IDS.

Nous étudions dans notre projet, deux approches distinctes, la méthode du plus proches voisins basée sur une similarité définie sur un espace euclidien représentant les processus d'apprentissage ainsi que les processus de tests, qu'ils soient normaux ou anormaux.

Des améliorations exigées de la performance de la méthode, nous amène à proposer une représentation plus fine des processus à l'aide de la technique IF-IDF.

La deuxième méthode, introduit la notion de probabilité, et plus précisément, la causalité probabiliste entre les évènements, qui sont dans notre cas les appels système du processus. La méthode en question est celle du bayes naïve, avec ses deux variantes augmentées par un arbre et une forêt.

La discussion des performances des méthodes utilisées, seront discutées suivant deux paramètres : le taux de détection et le faux positif. Des courbes sont dessinées afin de bien exprimer les résultats des expérimentations réalisées. Les données d'apprentissage et de test ont été extraites du projet DARPA 98.

La théorie de l'information est utilisée pour mesurer l'importance des appels systèmes intervenants dans les processus d'apprentissage et du test. Les appels systèmes les plus pertinents seront sollicités.

Objectif :

La réalisation d'un système de détection d'intrusion basé sur la classification comportementale des processus en utilisant les appels systèmes qui sont extraits à partir de Basic Security Monitoring (BSM) du système d'exploitation Solaris. Un bon système offrira un grand taux de détection (le pourcentage des processus anormaux détectés comme anormaux) et un faible

taux de faux positifs (pourcentage de fausses alarmes, un processus normal est détecté comme anormal).

Travaux antérieurs :

1. Utilisation de k plus proches voisins

[Nguyen, 10] a utilisé l'algorithme des plus proches voisins pour classer le comportement des processus. Il a utilisé les données d'apprentissage du DARPA 98 pour la construction et le test du système. Dans chaque session, il existe plusieurs processus. Pour chaque processus, il extrait une liste ordonnée des appels système à partir du fichier d'audit BSM. Pour caractériser le comportement d'un processus, les fréquences des appels système sont considérées. Il a utilisé une formule de cosinus pour calculer la similarité entre deux processus au cours de l'application de l'algorithme. Ses données d'entraînement est une collection de processus normaux. Dans l'algorithme, il y a deux paramètres qui peuvent être utilisés pour contrôler le résultat de la classification: k le nombre de voisins et le seuil de la similarité avec laquelle il peut définir si un processus est normal ou non. Par exemple, si le seuil est de 0,72, ceci signifie que le processus de test doit avoir une similarité plus de 72% par rapport à k-plus proche voisins pour qu'ils puissent être classés comme un processus normal, sinon, il s'agit d'un processus anormal. Cela signifie que, si le seuil est élevé, la chance d'être un processus normal est faible, la chance d'être un processus anormal est élevée, et le taux de détection sera élevé et aussi le taux de faux positifs. Son ensemble de données d'apprentissage contient 920 processus normal et pour le test 39 attaques et 377 processus normaux. Avec k = 10, seuil de= 0.96, ils ont eu 100% de détection et de 7,69% taux de faux positifs. Avec k = 15, seuil = 0.96, ils avaient un taux de détection de 100% et 8,22% de faux positifs taux.

[Liao & Vemuri, 02] ils ont utilisé l'algorithme de KNN. Leurs travails est semblables au précédent mais ils considèrent chaque processus comme un document avec les appels système comme des mots. La classification des

processus est convertie à un problème de classification de texte. Une matrice de documents terme est construite pour coder les données. Dans une matrice des termes de documents, les lignes correspondent aux documents de la collection et les colonnes correspondent à des termes. Dans ce problème, les documents sont des processus, les termes sont les appels système. Deux techniques de pondération du texte sont utilisées pour convertir les appels système des processus aux vecteurs de la matrice de documents terme. La première technique est *term frequency inverted document frequency* (TF-IDF) et le second est *term frequency*. La performance du système de Liao et Vemuri est mentionné ici : Leur ensemble de données contient 35 attaques et 5285 processus normaux. Avec k = 10, seuil de= 0,72, ils ont eu 100% de détection et de 0,44% taux de faux positifs. Avec k = 15, seuil = 0,99, ils avaient un taux de détection de 100% et 0,87% de faux positifs taux.

2. Utilisation de réseau bayésien naïve :

[Amor & Benferhat & Elouedi, 06] ont présenté des résultats expérimentaux montrant que les réseaux bayésiens naïfs, avec leur structure simple et leur hypothèse forte sur l'indépendance, peuvent être compétitifs. L'étude expérimentale est effectuée sur la base de données KDD'99. Les données KDD'99 sont au fait des données formatées fournies par DARPA à MIT Lincoln Lab. Ces données sont très appropriées pour évaluer un système de détection d'intrusion de type comportementale puisque la base des tests contient des attaques qui n'apparaissent pas dans la base d'apprentissage. Leurs expérimentations considèrent trois niveaux de granularités des attaques: cas où ils traitent toutes les attaques, cas où ils les regroupent en quatre classes principales et le cas où ils se focalisent uniquement sur le comportement normal ou anormal des connexions. Après expérimentation, le résultat générale est : taux de détection = 89.75% et le faux positive = 1.52 %.

[BOUDJELIDA, 08] il a appliqué l'apprentissage des réseaux Bayésiens augmentés de type TAN pour la détection d'intrusions. Il a montré que l'hypothèse d'indépendance entre les attributs utilisés dans le classifier de

Bayes naïf est généralement fausse (hypothèse naïve). Il a adopté la structure de l'arbre pour relier les attributs, pour sa construction il a utilisé l'algorithme de MWST (Maximal Weight Spanning Tree). Ce dernier, s'applique à la recherche de structure d'un réseau bayésien en fixant un poids à chaque arête potentielle de l'arbre. Ce poids est l'information mutuelle entre les variables. Après obtention de la matrice, il a choisis algorithmes de Kruskal pour la détermination des arcs pertinents. Ensuite, il a appliqué l'algorithme de TAN en établissant les relations défini par l'arbre. Le résultat est comme suit : Taux de détection = 46.70 % (moyenne de tous les classes), Faux positif = 1.9 %

La structure du mémoire :

Afin de répondre à cet objectif, ce mémoire est structuré de la façon suivante :

Le premier chapitre est une présentation de la notion des systèmes de détection d'intrusions, avant de donner les définitions relatives à cette notion, nous allons illustrer des aspects théoriques comparatifs .Ensuite une classification des IDS sera abordée. Pour entamer ensuite les mesures partielles qui peuvent être utilisées pour tester l'efficacité des IDS.

Le deuxième chapitre, sera consacré aux notions sur les attaques informatiques, leurs stratégies ainsi que leur classification. Nous allons invoquer aussi une analyse sur l'audit de sécurité. Nous détaillerons l'ensemble de données que nous allons utiliser pour construire et évaluer le système de détection d'intrusion objet de ce travail, à savoir, la base de données DARPA. Ensuite nous expliquerons les étapes appliqués à ces données brutes pour transformer à une présentation vectorielle qui sera utilisé prochainement. Nous exposerons les notions relatives à la théorie de l'information.

Le troisième chapitre est dédié à la présentation des concepts relatifs à la classification avec la méthode des plus proches voisins . Pour cela nous allons d'abord exposer quelques notions sur les algorithmes de classification

en générale .Nous présenterons aussi l'utilisation du technique de IF-IDF pour l'amélioration des résultats obtenus.

Dans le quatrième chapitre nous allons nous focaliser sur la notion de la classification bayésienne avec ses variantes. Nous présenterons aussi l'utilisation d'une nouvelle technique, à savoir l'utilisation d'informations fournit par les appels systèmes pour la construction du modèle probabiliste.

Nous conclurons ce mémoire par une conclusion générale et des perspectives.

Chapitre 1 :

GENERALITE SUR LES SYSTEMES DE DETECTION D'INTRUSIONS

1. Introduction :

Au cours des dernières années, les réseaux ont évolués, plus que jamais. L'Internet a changé l'informatique. Les possibilités et opportunités deviennent illimitées, mais d'autre part, les risques et les chances d'intrusions malveillantes ont augmenté. La sécurité est devenue un problème majeur dans la gestion des réseaux d'entreprise ainsi que pour les particuliers toujours plus nombreux à se connecter à Internet. La conception des mécanismes de sécurité de manière à empêcher l'accès non autorisé à des ressources système et de données est très importante.

A l'heure actuelle, l'empêchement complet des violations est irréel. On peut essayer de détecter ces tentatives d'intrusion afin que des mesures puissent être prises pour réparer les dégâts plus tard. Ce domaine de recherche est appelé détection d'intrusion.

Ce manuscrit a pour but de présenter les notions de bases sur qui s'articulent les systèmes de détections d'intrusions ainsi que les différentes méthodes, techniques et outils pour développer des systèmes plus performants.

Pour bien présenter les concepts du système de détection d'intrusion, nous allons donner tout d'abord une vision globale sur le pare-feu et l'IDS. Nous allons expliquer pourquoi les systèmes de détection d'intrusion sont nécessaires. Ensuite, nous présenterons les différentes sortes d'IDS. Nous étudierons les critères importants pour sa construction et enfin, nous citerons quelques exemples avec des brèves descriptions des systèmes déjà existant dans le marcher.

2. Concepts et outils de la sécurité :

Depuis l'article d'Anderson en 1980 [Cisco, 03] [Stef, 99], plusieurs techniques pour la détection des intrusions ont été étudiées. Plusieurs nouvelles

méthodes de détection ont été mises en place. Plusieurs efforts de recherche ont été lancés et des résultats efficaces ont été obtenus.

Les intrusions sont causées par des attaquants qui accèdent aux systèmes par Internet, qui sont des utilisateurs autorisés par le système et qui tentent de gagner plus de privilèges pour lesquels ils ne sont pas autorisés, ou des utilisateurs autorisés qui abusent des privilèges qui leurs sont attribués. La technologie de la détection d'intrusion renforce la sécurité du réseau en ajoutant une couche de protection.

La détection d'intrusion permet aux organisations de protéger leurs systèmes contre les menaces qui s'introduisent avec la connectivité croissante des réseaux. Les IDS ont obtenu l'acceptation comme un complément nécessaire à l'infrastructure de sécurité de chaque organisation. Même si, la technologie de détection d'intrusion ne puisse pas offrir une protection complète contre les attaques, ils améliorent l'approche de défense en profondeur, qui est la tendance à la mode de la sécurité réseau. En fin, On va discuter comment on peut différencier entre les produits de détection d'intrusion et pourquoi on a besoin de les utiliser et comment ils renforcent la sécurité du réseau.

2.1. Pare-feu :

C'est un dispositif de sécurité réseau, qui fonctionne selon une politique de sécurité qui a été défini et configuré. Il est l'un des produits de sécurité qui implémente la politique de sécurité. Il est également impératif qu'il ne peut pas offrir une protection complète contre le trafic malveillant. La différence fondamentale entre un pare-feu et un IDS, c'est que le pare-feu offre une protection active contre les attaques, alors que les produits IDS peuvent déclencher une alerte et détecter les attaques.

Comme les pare-feu, les produits de détection d'intrusion aussi ne peuvent pas offrir une protection complète et ne peuvent pas remplacer tous les autres produits.

2.2. Fonctionnalités:

- le pare-feu fournit le contrôle d'accès du trafic Internet à partir de l'intérieur et de l'extérieur.
- Il fonctionne activement par la politique de sécurité configurée et en autorisant uniquement le trafic légitime défini par celle-ci. Par exemple, un pare-feu peut être configuré pour autoriser le trafic du port 80 du serveur, et un certain trafic uniquement sur le port 25 du serveur e-mail. Dans cet exemple, il peut être clairement observé que le pare-feu n'examine pas le contenu du trafic légitime.

2.3. Limitations :

C'est l'une des questions les plus fréquentes pour les personnes novices dans la détection d'intrusion. Comme la plupart des gens pensent que leur pare-feu peut tout seul protéger leur réseau, chose qui n'est pas vrai, en considérant les constats suivants :

- tout accès à Internet ne passe par le pare-feu.
- Certaines menaces ne proviennent pas de l'extérieur du pare-feu.
- Le pare-feu est sujet à lui-même peut être attaqué.
- il n'examine pas le contenu du trafic légitime.
- Le pare-feu n'offre aucune protection si le réseau est déjà affranchi.
- Le pare-feu ne peut pas empêcher toutes sortes d'attaques et surtout certaines variantes de ces attaques.
- ils n'offrent pas toute sorte d'analyse médico-légale.

3. Définition du Système de Détection d'Intrusion :

Tout d'abord, IDS signifie Intrusion Détection System. Il s'agit d'un équipement permettant de surveiller l'activité d'un réseau ou d'un hôte donné, afin de détecter toute tentative d'intrusion et éventuellement de réagir à cette tentative [Baud & Mar, 04].

La détection d'intrusion est le processus de la surveillance des événements qui se produisent dans un ordinateur ou dans un réseau et de les analyser pour des signes d'intrusions, qui sont définis comme des tentatives qui compromettent la confidentialité, l'intégrité ou la disponibilité ou bien qui dépassent les conditions de la sécurité d'un ordinateur ou un réseau. [Mart & Mark, 03]

3.1. Plusieurs raisons intéressantes pour l'utilisation des IDS:

- Pour fournir les informations possibles sur les intrusions et les tentatives qui ont eu place, permettant l'amélioration du diagnostic, la récupération, et la correction des facteurs causals.
- Pour agir et contrôler la qualité de la sécurité et l'administration, en particulier dans les grandes entreprises.
- Pour déceler les objectifs des attaques.
- Pour éviter les problèmes, on augmente la protection des risques qui sont découverts, et infliger une punition pour ceux qui voudraient attaquer ou autrement abuser du système.
- Pour détecter les attaques et les violations de sécurité qui ne sont pas prévus par d'autres mesures de sécurité.

3.2. L'aspect réel:

Dans le monde réel, IDS ressemble à l'alarme antivol dans une maison. Disons que le Dispositif de commande d'authentification, qui est situé à l'entrée de la porte ressemble au Pare-feu. Il contrôle l'accès à la maison par carte d'identité et

mot de passe. Il y a plusieurs façons de contourner ce dispositif. Soit un voleur peut casser la fenêtre et pénètre dans la maison avec succès sans détecter, ou un voleur peut usurper la carte d'identité et le mot de passe. Dans les deux cas ci-dessus. Le voleur n'est pas détecté et il n'y a aucune information concernant les dégâts, comment il s'est pris, qui est le voleur et quand a-t-il tenté le cambriolage.

Voyons comment il est différent avec les alarmes antivol. Alarmes contre le vol sont activés dans la maison et configuré pour donner l'alarme si quelqu'un pénètre par infraction. Si le voleur entre dans la maison, le système d'alarme se déclenche, en faisant alerter le gardien de la sécurité ou le propriétaire de la demeure, il peut informer les personnes associées, il peut enregistrer l'heure du cambriolage et de plus amples informations en fonction de la configuration politique de la sécurité et la capacité de traitement de l'alarme anti-intrusion. Ainsi, les alarmes anti-intrusion réduisent les dommages d'un cambriolage, en fournissant les informations complètes et les indices pour l'enquête médico-légale et de prendre une action en justice contre les voleurs.

La détection d'intrusion comporte également des mécanismes fonctionnels similaires, qui réduisent le risque, et aide à l'enquête médico-légale, et contribue à rapporter les événements au management. Dans ce qui suit nous allons discuter quelques-uns des mythes associés à la technologie de détection des intrusions.

3.3. Fonctionnalités :

- il donne l'image claire sur ce qui se passe dans le réseau et le système.
- IDS peut détecter, reconnait les attaques et alerter le système.
- il offre une plus grande flexibilité et intégrité à l'infrastructure de sécurité existante.
- il peut enregistrer les sessions en format spécifié.
- il fournit et améliore le processus d'enquête médico-légale à l'aide de journaux de sessions, la corrélation des événements et des interfaces graphiques.

- il surveille le réseau ou les systèmes en temps réel et de réaliser l'analyse en temps réel.
- il peut alerter les personnes de sécurité avec des modèles spécifiés.
- il peut prendre des réponses actives, comme le blocage des adresses IP, l'arrêt des connexions.
- il permet de manière efficace de donner des rapports pour la gestion.
- Plus important encore, IDS fournit des lignes directrices qui aident dans le développement de la politique de sécurité de l'organisation.

3.4. Limitations:

- il est tout simplement active, pas proactive (optimiste).
- Il ne peut pas empêcher l'attaque.
- il n'est pas automatisé, il a besoin d'importantes ressources humaines pour leur gestion.
- il ne peut pas offrir une protection complète pour les ressources. C'est juste une couche supplémentaire. Ce n'est pas une panacée.
- il ne peut pas compenser les lacunes des protocoles réseau.
- il ne peut pas protéger tous les types d'attaques. Il a des limites.
- il ne peut pas résister à des volumes élevés et des vitesses élevées de trafic Internet.

4. Classification des IDS:

Nous allons présenter ici la technologie de détection d'intrusion d'une manière taxonomique. Il y a plusieurs types d'IDS disponibles aujourd'hui, caractérisé par différente approche de la surveillance et de l'analyse. Chaque approche a ses avantages et des inconvénients distincts. Toutes les approches peuvent être décrites en termes d'un modèle d'IDS.

- L'emplacement d'IDS
- Les méthodes de détection

- Les types de réponse
- Fréquence d'utilisation

Nous allons tout d'abord étudier la détection d'intrusion basée sur l'hôte, puis basée sur une application, avant de nous intéresser aux IDS réseaux (Network IDS : NIDS).

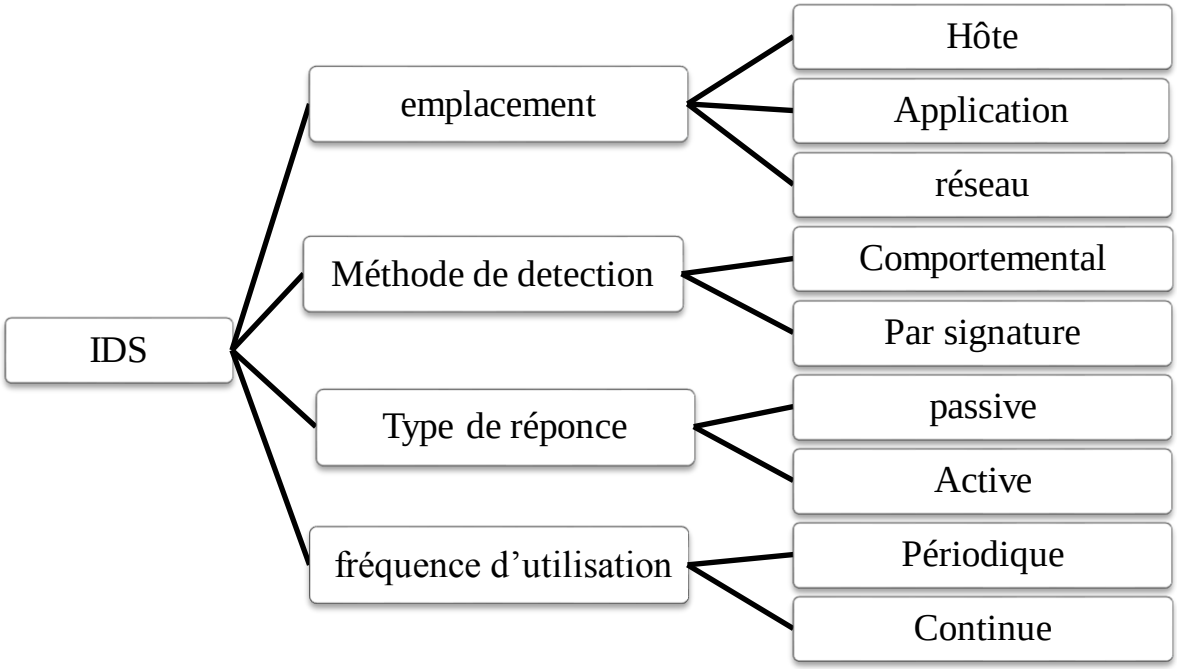


Fig. 1. 1. Les différentes sortes des IDS

4.1. L'emplacement d'IDS :

La façon la plus commune pour classifier les IDS est de les regrouper par emplacement de l'information source où ils opèrent. Les sources des informations primaires sont : les paquets réseau capturés à partir des réseaux ou des segments de réseau local ou les systèmes d'exploitation et les fichiers critiques.

4.1.1. La détection d'intrusion basée sur l'hôte :

Les systèmes de détection d'intrusion basés sur l'hôte ou HIDS (Host IDS) analysent exclusivement l'information concernant cet hôte. Comme ils n'ont pas à contrôler le trafic du réseau mais "seulement" les activités d'un hôte.

Avec une grande fiabilité et précision, l'IDS peut déterminer exactement quels utilisateurs ou quels processus sont impliqués dans une attaque. Avec ces types de systèmes, le résultat peut être déterminée à la différence du NIDS, le HIDS peut accéder directement et de contrôler les fichiers de données et les processus habituellement visés par les attaques.

Ces IDS utilisent généralement les traces d'audit du système d'exploitation afin de fournir une information sur l'activité de la machine.il possède un certain nombre d'avantages : il est possible de constater immédiatement l'impact d'une attaque et donc de mieux réagir. Grâce à la quantité d'information étudiée, il est possible d'observer les activités se déroulant sur l'hôte avec précision et d'optimiser le système en fonction des activités observées.

De plus, les HIDS sont extrêmement complémentaires des NIDS. En effet, ils permettent de détecter plus facilement les attaques de type "Cheval de Troie", alors que ce type d'attaque est difficilement détectable par un NIDS. Les HIDS permettent également de détecter des attaques impossibles à détecter avec un NIDS, car elles font partie du trafic crypté.

Néanmoins, ce type d'IDS possède également ses faiblesses, qui proviennent de ses qualités : du fait de la grande quantité de données générées, ce type d'IDS est très sensible aux attaques de type DoS, qui peuvent faire exploser la taille des fichiers de logs.

Un autre inconvénient tient justement à la taille des fichiers de rapport d'alertes à examiner, qui est très contraignante pour le responsable de sécurité. La taille des fichiers peut en effet atteindre plusieurs Mégaoctets.

Du fait de cette quantité de données à traiter, ils sont assez gourmands en CPU et peuvent parfois altérer les performances de la machine hôte.

Les HIDS sont en général placés sur des machines sensibles, susceptibles de subir des attaques et possédantes des données sensibles pour l'entreprise. Les serveurs web ou applicatifs, peuvent notamment être protégés par un HIDS.

4.1.2. Détection d'Intrusion basée sur une application :

Les IDS basés sur les applications (AIDS) sont un sous-groupe des IDS hôtes. Ils contrôlent l'interaction entre un utilisateur et un programme en ajoutant des fichiers de log afin de fournir de plus amples informations sur les activités d'une application particulière. Puisque vous opérez entre un utilisateur et un programme, il est facile de filtrer tout comportement notable. Un ABIDS se situe au niveau de la communication entre un utilisateur et l'application surveillée.

L'avantage de cet IDS est qu'il lui est possible de détecter et d'empêcher des commandes particulières dont l'utilisateur pourrait se servir avec le programme et de surveiller chaque transaction entre l'utilisateur et l'application. De plus, les données sont décodées dans un contexte connu, leur analyse est donc plus fine et précise.

Par contre, du fait que cet IDS n'agit pas au niveau du noyau, la sécurité assurée est plus faible, notamment en ce qui concerne les attaques de type "Cheval de Troie".

De plus, les fichiers de log générés par ce type d'IDS sont des cibles faciles pour les attaquants et ne sont pas aussi sûrs, par exemple, que les traces d'audit du système. Ce type d'IDS est utile pour surveiller l'activité d'une application très sensible, mais son utilisation s'effectue en général en association avec un HIDS. Il faudra dans ce cas contrôler le taux d'utilisation CPU des IDS afin de ne pas compromettre les performances de la machine.

4.1.3. La Détection d'Intrusion Réseau (NIDS)

La forme la plus commune de systèmes commerciaux de détection d'intrusion est basée sur le réseau. Ces systèmes détectent les attaques en capturant et en analysant les paquets réseau en écoutant sur le segment de réseau ou d'un commutateur. Ils le font en faisant correspondre entre un ou plusieurs paquets contre une base de données de signatures d'attaques connues, ou d'effectuer le décodage du protocole pour détecter les anomalies.

Le NIDS est capable à la fois de déclencher des alertes et de clôturer les connexions instantanément chaque fois qu'il remarque des activités suspectes. Le « *mode Promiscuous* » est la forme la plus commune du fonctionnement .ils surveillent chaque paquet qui est en transmission du segment local.

L'implantation d'un NIDS sur un réseau se fait de la façon suivante : des capteurs sont placés aux endroits stratégiques du réseau et génèrent des alertes s'ils détectent une attaque. Ces alertes sont envoyées à une console sécurisée, qui les analyse et les traites éventuellement. Cette console est généralement située sur un réseau isolé, qui relie uniquement les capteurs et la console.

L'emplacement des capteurs :

Les capteurs placés sur le réseau sont placés en mode furtif (ou stealth mode), de façon à être invisibles aux autres machines. Pour cela, leur carte réseau est configurée en mode "promiscuous", c'est à dire le mode dans lequel la carte réseau lit l'ensemble du trafic, de plus aucune adresse IP n'est configurée.

Un capteur possède en général deux cartes réseaux, une placée en mode furtif sur le réseau, l'autre permettant de le connecter à la console de sécurité.

Du fait de leur invisibilité sur le réseau, il est beaucoup plus difficile de les attaquer et de savoir qu'un IDS est utilisé sur ce réseau.

L'emplacement des capteurs :

Il est possible de placer les capteurs à différents endroits, en fonction de ce que l'on souhaite observer. Les capteurs peuvent être placés avant ou après le pare-feu, ou encore dans une zone sensible que l'on veut protéger spécialement.

Si les capteurs se trouvent après un pare-feu, il leur est plus facile de dire si le pare-feu a été mal configuré ou de savoir si une attaque est venue par ce pare-feu. Ils ont pour mission de détecter les intrusions qui n'ont pas été arrêtées par ce dernier. Il s'agit d'une utilisation courante d'un NIDS.

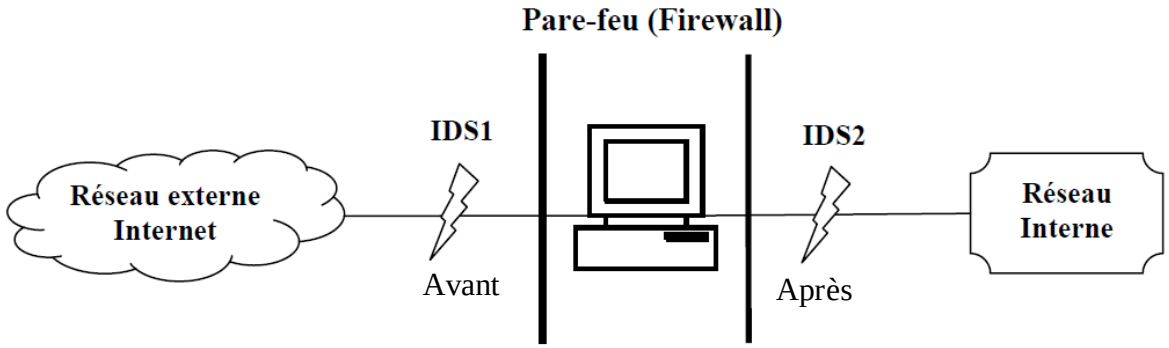


Fig. 1. 2. Installation des N-IDS

Il est également possible de placer un capteur à l'extérieur du pare-feu (avant le firewall). L'intérêt de cette position est que le capteur peut ainsi recevoir et analyser l'ensemble du trafic d'Internet. Si nous plaçons le capteur ici, il n'est pas certain que toutes les attaques soient filtrées et détectées. Pourtant, cet emplacement est le préféré de nombreux experts parce qu'il offre l'avantage d'écrire dans les logs et d'analyser les attaques (vers le pare-feu...), ainsi l'administrateur voit ce qu'il doit modifier dans la configuration du pare-feu.

Les capteurs placés à l'extérieur du pare-feu servent à détecter toutes les attaques en direction du réseau, leur tâche ici est donc plus de contrôler le fonctionnement et la configuration du firewall que d'assurer une protection contre toutes les intrusions détectées (certaines étant traitées par le firewall).

Il est également possible de placer un capteur et un autre après le firewall. En fait, cette variante réunit les deux cas mentionnés ci-dessus. Mais elle est très dangereuse si on configure mal les capteurs et/ou le pare-feu, en effet on ne peut simplement ajouter les avantages des deux cas précédents à cette variante.

Les capteurs IDS sont parfois situés à l'entrée de zones du réseau particulièrement sensibles (parcs de serveurs, données confidentielles…), de façon à surveiller tout trafic en direction de cette zone.

Les avantages des NIDS sont les suivants : les capteurs peuvent être bien sécurisés puisqu'ils se contentent d'observer le trafic et permettent donc une surveillance discrète du réseau, les attaques de type scans sont facilement détectées, et il est possible de filtrer le trafic. Son déploiement à peu d'impact sur le réseau, les NIDS sont généralement des dispositifs passifs qui écoutent sur le fil réseau sans interférer avec le fonctionnement normal d'un réseau. Un large réseau peut être contrôlé par quelques NIDS bien placés.

Les NIDS sont très utilisés et remplissent un rôle indispensable, mais ils présentent néanmoins de nombreuses faiblesses. En effet, la probabilité de faux négatifs (attaques non détectées) est élevée. Certains NIDS ont des difficultés à traiter des fragments de paquets, ce qui peut provoquer le dysfonctionnement d'un IDS. Ils ont des problèmes dans le traitement des vitesses élevées et des volumes élevés de trafic. Ils ne peuvent pas analyser les informations cryptées.

Pour finir, à l'opposé des IDS basés sur l'hôte, ils ne perçoivent pas les impacts d'une attaque .Même si nous distinguons HIDS et NIDS, la différence devient de plus en plus réduite puisque les HIDS possèdent maintenant les fonctionnalités de base des NIDS.

4.2. Modes de détection :

Il existe deux modes de détection, la détection d'anomalies et la reconnaissance de signatures.

Il faut noter que la reconnaissance de signature est le mode de fonctionnement le plus implémenté par les IDS du marché. Cependant, les nouveaux produits tendent à combiner les deux méthodes pour affiner la détection d'intrusion.

4.2.1. La détection d'anomalies :

Elle consiste à détecter des anomalies par rapport à un profil "de trafic habituel". La mise en œuvre comprend toujours une phase d'apprentissage au cours de laquelle les IDS vont "découvrir" le fonctionnement "normal" des éléments surveillés. Ils sont ainsi en mesure de signaler les divergences par rapport au fonctionnement de référence.

Dans le cas du HIDS, ce type de détection peut être basé sur des informations telles que le taux d'utilisation CPU, l'activité sur le disque, les horaires de connexion ou d'utilisation de certains fichiers (horaires de bureau…). Citons, à titre d'exemple, un projet a été sponsorisé par la DARPA (en 98 et 99), en collaboration avec le laboratoire Lincoln du MIT et qui était l'un des projets les plus ambitieux [Lippmann & al, 00]. Le but était de fournir un ensemble significatif de données d'apprentissage, comprenant trafic de fond et activités intrusives (c'est-à-dire du trafic intrusif ou des événements systèmes causés par des attaques). Le trafic de fond était déduit des données statistiques collectées sur le réseau des bases de l'Air Force alors que les attaques étaient générées par des scripts créés spécialement, mais aussi par des scripts collectés à travers des sites spécialisés et des listes de diffusion. Les données collectées concernaient à la fois des HIDS (par exemple données d'audit de stations Solaris, disk dump de machines UNIX et BSM «Basic Security Monitoring ») et des NIDS.

Le jeu de test DARPA'98 utilisait environ 300 attaques (classées en 38 types d'attaques), tandis que DARPA'99 utilisait environ 50 types d'attaques.

Les avantages:

- Les systèmes de détection d'intrusion basés sur la détection d'anomalie détectent le comportement peu commun, et ils ont ainsi la capacité de détecter des symptômes des attaques connues et inconnues sans la connaissance spécifique des détails.
- Cette approche permet de produire l'information utile pour la définition des signatures pour les systèmes de détection d'intrusion à base de signatures.

Les inconvénients:

- Le point noir de cette approche est le grand nombre de fausses alarmes dues aux comportements imprévisibles des utilisateurs du réseau.
- Elle exige souvent l'historique à long terme des évènements enregistrés afin de caractériser les modèles normaux de comportement. Les systèmes basés sur cette approche doivent être dotés d'une certaine intelligence pour raison d'apprentissage automatique.

4.2.2. La reconnaissance de signature :

Cette approche consiste à rechercher dans l'activité de l'élément surveillé les empreintes (ou signatures) d'attaques connues. Ce type d'IDS est purement réactif ; il ne peut détecter que les attaques dont il possède la signature. De ce fait, il nécessite des mises à jour fréquentes.

De plus, l'efficacité de ce système de détection dépend fortement de la précision de sa base de signature. C'est pourquoi ces systèmes sont contournés par les pirates qui utilisent des techniques dites "d'évasion" qui consistent à maquiller les attaques utilisées. Ces techniques tendent à faire varier les signatures des attaques qui ainsi ne sont plus reconnues par l'IDS.

Il est possible d'élaborer des signatures plus génériques, qui permettent de détecter les variantes d'une même attaque, mais cela demande une bonne

connaissance des attaques et du réseau, de façon à stopper les variantes d'une attaque et à ne pas gêner le trafic normal du réseau Une signature permet de définir les caractéristiques d'une attaque, au niveau des paquets (jusqu'à TCP ou UDP) ou au niveau protocole (HTTP, FTP…).

Au niveau paquet, l'IDS va analyser les différents paramètres de tous les paquets transitant et les comparer avec les signatures d'attaques connues. Au niveau protocole, l'IDS va vérifier au niveau du protocole si les commandes envoyées sont correctes ou ne contiennent pas d'attaque. Cette fonctionnalité a surtout été développée pour HTTP actuellement.

Cet exemple nous a semblé très formateur sur le rôle des signatures et leur élaboration. Il faut savoir que les sites des vendeurs d'IDS proposent des mises à jour des signatures en fonction des nouvelles attaques identifiées.

Néanmoins, plus il y a de signatures différentes à tester, plus le temps de traitement sera long, l'utilisation de signatures plus élaborées peut donc procurer un gain de temps appréciable.

Cependant, une signature mal élaborée peut ignorer des attaques réelles ou identifier du trafic normal comme étant une attaque. Il convient donc de manier l'élaboration de signatures avec précaution, et en ayant de bonnes connaissances sur le réseau surveillé et les attaques existantes.

Les avantages:

- Très efficace pour détecter des attaques sans produire un grand nombre de fausses alarmes.
- Peut rapidement et sûrement diagnostiquer l'utilisation d'un outil spécifique ou une technique d'attaque.
- Ceci peut aider les responsables de sécurité à donner la priorité aux mesures correctives.

Les inconvénients:

- Peut seulement détecter les attaques connues, dont les signatures sont introduites dans le système, donc le système de détection doit être constamment mis à jour avec les signatures des nouvelles attaques.
- Beaucoup de systèmes adoptant cette approche sont conçus pour employer un nombre limité de signatures qui peuvent être définis, ce qui les empêchent de détecter des variantes de ces attaques.

Une fois une attaque détectée, un IDS a le choix entre plusieurs types de réponses, que nous allons maintenant détailler.

4.3. Les types de réponse:

Il existe deux types de réponses, suivant les IDS utilisés. La réponse passive est disponible pour tous les IDS, la réponse active est plus ou moins implémentée.

4.3.1. Réponse active :

La réponse active au contraire a pour but de stopper une attaque au moment de sa détection. Pour cela on dispose de deux techniques : la reconfiguration du firewall et l'interruption d'une connexion TCP.

La reconfiguration du firewall permet de bloquer le trafic malveillant au niveau du firewall, en fermant le port utilisé ou en interdisant l'adresse de l'attaquant. Cette fonctionnalité dépend du modèle de firewall utilisé, tous les modèles ne permettant pas la reconfiguration par un IDS. De plus, cette reconfiguration ne peut se faire qu'en fonction des capacités du firewall.

L'IDS peut également interrompre une session établie entre un attaquant et sa machine cible, de façon à empêcher le transfert de données ou la modification du système attaqué.

Pour cela l'IDS envoie un paquet TCP reset aux deux extrémités de la connexion (cible et attaquant). Un paquet TCP reset a le flag RST positionné, ce qui indique une déconnexion de la part de l'autre extrémité de la connexion. Chaque extrémité en étant destinataire, la cible et l'attaquant pensent que l'autre extrémité s'est déconnectée et l'attaque est interrompue.

Dans le cas d'une réponse active, il faut être sûr que le trafic détecté comme malveillant l'est réellement, sous peine de déconnecter des utilisateurs normaux. En général, les IDS ne réagissent pas activement à toutes les alertes. Ils ne répondent à des alertes que quand celles-ci sont positivement certifiées comme étant des attaques. L'analyse des fichiers d'alertes générés est donc une obligation pour analyser l'ensemble des attaques détectées.

4.3.2. Réponse passive:

La réponse passive d'un IDS consiste à enregistrer les intrusions détectées dans un fichier de log qui sera analysé par le responsable de sécurité.

Certains IDS permettent de d'enregistrer l'ensemble d'une connexion identifiée comme malveillante. Ceci permet de remédier aux failles de sécurité pour empêcher les attaques enregistrées de se reproduire, mais elle n'empêche pas directement une attaque de se produire.

4.4. Fréquence d'utilisation :

Enfin, une dernière différenciation est la caractéristique d'utilisation qui peut se faire d'une façon : continue (online) ou périodique (offline).

4.4.1. Utilisation continue :

La détection d'attaque se fait au moment où elle se produit, Dans la plupart des cas, avant d'attaquer un réseau, l'attaquant doit scanner l'environnement pour

récolter des informations. Par conséquent, en voyant cela, on peut détecter une attaque avant même qu'elle ne se produise et ainsi y répondre le plus tôt possible.

4.4.2. Utilisation périodique :

Cette méthode s'exécute périodiquement et on ne voit que le résultat d'attaque, elle est préférable pour avoir une défense plus fiable du point de vue du temps de calcul que pour la première utilisation. Contrairement à l'IDS online, l'attaque ne peut pas être détectée le plus tôt possible pour l'éviter. Plus une attaque est détectée tardivement, les dommages sont importants.

5. Critères d'évaluation :

Malgré le fait que les systèmes de détection d'intrusions sont devenus les outils de défense omniprésents dans les systèmes informatiques d'aujourd'hui, jusqu'à présent on n'a aucune méthodologie complète et scientifique rigoureuse pour examiner l'efficacité de ces systèmes [Mell & al, 03]. Dans ce qui suit, nous allons donner un ensemble de mesures partielles qui peuvent être utilisées pour tester le rendement des IDS.

5.1. Erreurs de classification :

Il existe plusieurs types d'erreurs venant d'un détecteur, influençant plus ou moins sa puissance. Les vrais positifs sont les cas où une alarme se déclenche quand il y a une violation des politiques de sécurité. Les vrais négatifs sont les cas où aucune alarme ne se déclenche et rien d'anormal ne se produit. Les faux positifs sont les cas où une alarme se déclenche alors qu'il ne se produit rien d'anormal. Les faux négatifs sont les cas où une alarme ne se déclenche pas alors qu'il se produit une chose anormale. A première vue, on pourrait supposer qu'un faux positif est moins dangereux qu'un faux négatif. [LERM, 06]

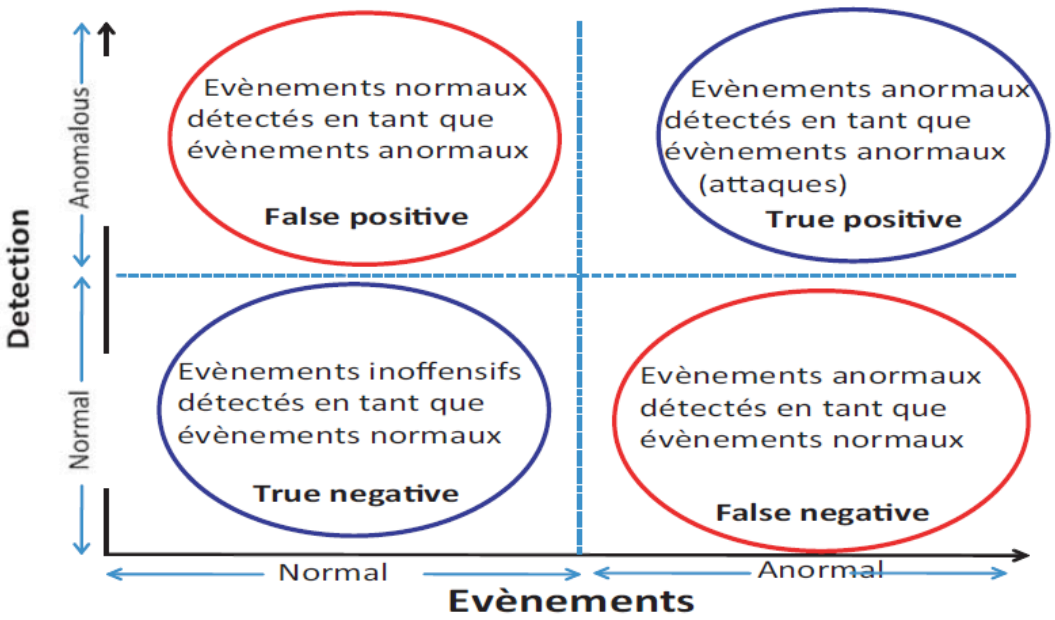


Fig. 1. 3. Les différentes sortes des IDS

6. Logiciels existants :

Le marché des IDS est très vaste. Certains produits sont gratuits et d'autres payants. Voici une liste non-exhaustive :

- Ossec : HIDS gratuit et facilement configurable qui fonctionne sur plusieurs OS tels que Windows, Linux et MacOs.
- Bro : NIDS gratuit construit par des groupes de recherche fonctionnant sur Unix. Il mélange des signatures d'attaques connues et des comportements normaux pour détecter une intrusion.
- IDSNet : NIDS basé sur la machine learning, créé par le département of the Informatics and Mathematical Modelling de l'université technique du Danemark (DTU) qui a permis à d'implémenter son propre modèle de machine learning pour créer un NIDS.
- Snort : NIDS gratuit mélangeant les signatures d'attaques et des comportements normaux des utilisateurs.
- Prelude : IDS hybride utilisant le format IDMEF 24 pour faire communiquer ses différentes parties.

7. Conclusion :

Dans ce chapitre, nous avons présenté un état de l'art en matière de détection d'intrusions. Pour bien positionner le problème nous avons d'abord donné une définition à cette notion, ensuite nous avons présenté une classification des systèmes de détection d'intrusions (IDS) en se basant sur la source de données de ces derniers ainsi que sur leurs emplacements dans le système informatique. Cette classification comporte les IDS basés hôte (H-IDS) , les IDS basés sur l'application (AB-IDS) les IDS basés réseau (N-IDS).

Nous avons abordé aussi les deux grandes approches utilisées par les Modes de détection IDS à savoir : La détection d'anomalies et La reconnaissance de signature , chacun d'entre eux présent des méthodes avec leurs avantages et leurs inconvénients. Ensuite nous avons présenté les deux réponses apportées par les systèmes après détection d'une intrusion, que ce soit la réponse active qui englobe plusieurs techniques comme la reconfiguration du firewall et l'interruption d'une connexion TCP, ou bien la réponse passive dont le principe repose sur l'émission des alertes, la chose qui est assurée par la plupart des IDS actuels. Pour donner par la suite un ensemble de procédures à respecter lors de l'implémentation des IDS.

Pour conclure on peut dire que les IDS constituent une seconde ligne de défense indispensable pour assurer la sécurité opérationnelle, surtout dans un contexte d 'interconnexion et d 'ouverture croissant des systèmes informatiques.

Le chapitre suivant sera consacré à la base principale du développement des IDS à savoir les attaques informatiques ainsi que des notion sur les données d'apprentissage.

Chapitre 2 :

Données d’apprentissage

1. Introduction :

Les systèmes d'information sont aujourd'hui de plus en plus ouverts sur Internet. Cette ouverture, a priori bénéfique, pose néanmoins un problème majeur : il en découle un nombre croissant d'attaques. La mise en place des outils de surveillance pour auditer ces système et détecter d 'éventuelles intrusions, afin de renforcer la politique de sécurité, nécessite l'utilisation d'un ensemble de données représentant ces attaques. Un exemple d'un tel ensemble est la base de données de l'ensemble DARPA 98 [MIT, 00], qui vise à fournir des données pour les chercheurs travaillant sur la détection d'intrusions.

Le corpus DARPA 98 contient des données réseau pour configurer et évaluer les IDS. Ces données contiennent des connexions normales et des connexions qui représentent des attaques. Avant de présenter cet ensemble de données avec un peu plus de détails , nous allons d'abord donner quelques notions sur les attaques informatiques . Ensuite vu l'importance des données tirées par les mécanismes d'audit afin de construire les signatures de ces attaques, nous allons présenter une analyse des activités relatives à l'audit de sécurité. En fin, nous expliquerons les détails concernant l'extraction et la transformation des données d'apprentissage et de test, de plus l'utilisation de la théorie d'information pour la sélection des attributs pertinents pour notre projet.

2. Les attaques :

2.1. Les différentes étapes d'une attaque :

La plupart des attaques, de la plus simple à la plus complexe fonctionnent suivant le même schéma :

- **Identification de la cible :** cette étape est indispensable à toutes attaques organisées, elle permet de récolter un maximum de

renseignements sur la cible en utilisant des informations publiques. On peut citer par exemple l'interrogation des serveurs DNS,….

- **Le scanning :** l'objectif est de compléter les informations réunies sur une cible visées. Il est ainsi possible d'obtenir les adresses IP utilisées, les services accessibles de même qu'un grand nombre d'informations de topologie détaillée (OS, versions des services, subnet, règles de firewall….).
- **L'exploitation :** Cette étape permet à partir des informations recueillies d'exploiter les failles identifiées sur les éléments de la cible, que ce soit au niveau protocolaire, services et applications ou systèmes d'exploitation présents sur le réseau.
- **La progression** : Il est temps pour l'attaquant de réaliser ce pourquoi il a franchi les précédentes étapes. Le but ultime étant d'élever ses droits vers le root sur un système afin de pouvoir y faire tout ce qu'il souhaite (inspection de la machine, récupération d'informations, installation de backdoors, nettoyage des traces,…).

2.2. Classification des Attaques

Plusieurs classifications des attaques ont été proposées selon plusieurs critères tels la gravité de l'attaque, les vulnérabilités exploitées par cette dernière, et la manière de procéder.

La classification la plus utilisée dans la littérature est celle adoptée par le Massachusetts Institute of Technology (MIT) [MIT, 99], qui comporte cinq classe d'attaques que nous allons présenter dans ce qui va suivre.

2.2.1. Dénis de Service (Denial Of Service DOS)

Un déni de service est une attaque dans laquelle le pirate rend certaines ressources du traitement ou du stockage indisponibles ou bien trop occupées pour pouvoir répondre aux demandes des utilisateurs légitimes

d'une machine. Il y a plusieurs types d'attaques par déni de service (DOS) [Kendall, 99]. Certaine attaque comme « smurf » abuse parfaitement des dispositifs légitimes. D'autre « ping of death » créent des paquets mal formés qui confondent la pile TCP/IP de la machine cible, cette dernière va essayer de reconstruire ces paquets par la suite. Encore d'autres « apache2, dos, syslogd » tirent profit des bugs dans le réseau.

2.2.2. Attaques Utilisateur vers Administrateur (User To Root U2R)

Sont une classe d'exploit dans laquelle l'attaquant commence par un accès à un compte utilisateur normal sur le système (obtenu en sniffant les mots de passe, en utilisant par exemple l'ingénierie sociale), ensuite il essaie d'exploiter la vulnérabilité sur ce système pour obtenir un accès administrateur.

Il y a plusieurs types d'attaques U2R [Kendall, 99], La plus commune est l'attaque «buffer overflow» qui se produit quand un programme copie beaucoup de données dans une zone mémoire tampon statique sans vérifier si la taille de cette dernière est suffisante, ce qui provoquera un débordement. Les données débordées seront stockées dans la pile du système, recouvrant ainsi les prochaines instructions qui devaient être exécutées. En manipulant soigneusement les données qui débordent sur la pile, un pirate peut causer l'exécution de quelques commandes arbitraires par le système d'exploitation qui vont l'aider à obtenir ce qu'il cherche. Une autre classe des attaques U2R exploite les programmes qui donnent des idées sur l'environnement dans lequel ils s'exécutent, un bon exemple d'une telle attaque est l'attaque «load module». Une autre classe d'attaques U2R exploite des programmes qui ont une mauvaise gestion des fichiers temporaires. Certaines attaques U2R utilisent des vulnérabilités dues à des conditions concurrentielles exploitables pondant le déroulement d 'un programme unique, de deux programmes ou plus s'exécutant simultanément

[Garfinkel & al, 96]. Bien qu'une programmation contrôlée pouvait éliminer toutes ces vulnérabilités, de tels bugs sont présents dans chaque version d'UNIX et de Microsoft Windows disponibles aujourd'hui.

2.2.3. Attaques Distant vers Local (Remote To Local R2L)

Une attaque R2L se produit quand un pirate qui peut envoyer des paquets vers une machine à travers un réseau mais qui n'a pas de compte sur cette machine exploite une vulnérabilité afin d'obtenir un accès locale comme utilisateur de cette machine. Il y a plusieurs manières avec lesquelles un attaquant peut aboutir à son objectif [Kendall, 99].

Certaines attaques exploitent le débordement des tampons causé par les logiciels du serveur du réseau « imap, named, sendmail ». Les attaques « dictionary, ftp-write, guest et xsnoop », essaient d'exploiter la faiblesse ou la mauvaise configuration des politiques de sécurité du système. L'attaque « xlock » utilise l'ingénierie sociale, pour réussir, l'attaquant doit parodier les opérateurs humains à fournir leurs mots de passe aux économiseurs d'écran qui sont en réalité des chevaux de Troie.

2.2.4. Attaques par Sondes (Probe)

Ces dernières années, on distribue un nombre de plus en plus important des programmes qui peuvent automatiquement balayer un réseau d'ordinateurs pour recueillir des informations utiles ou pour trouver des vulnérabilités connues [Garfinkel & al, 96]. Ces sondes réseau sont tout à fait utiles pour un pirate qui prépare une future attaque. Un attaquant avec un mappage des machines et des services disponibles sur un réseau peut employer ces informations pour rechercher tous les points faibles de ce dernier. Certaines de ces outils de balayage « satan, saint, mscan » permettent, même à un pirate débutant, d'examiner très rapidement des centaines ou des milliers de machines sur un réseau.

2.2.5. Attaques de Données (Data)

Les attaques de données impliquent quelqu'un (utilisateur ou administrateur) qui effectue certaines actions, dont il a la possibilité de les exécuter sur un poste donné, mais selon la politique de sécurité du site où il opère, il n'a pas les permissions suffisantes pour le faire. Souvent, ces attaques impliquent le transfert des fichiers de données secrètes de ou vers des sources où elles n'appartiennent pas.

3. l'audit de sécurité :

L'audit est un concept fondamental de la sécurité des systèmes et de la maintenance [Admin, 12]. L'audit est le processus d'analyse de l'historique des actions et des événements sur un système afin de déterminer ce qui s'est passé. L'historique est enregistré dans un journal répertoriant ce qui a été effectué, quand, par qui, et ce qui a été affecté.Les journaux appelés fichiers d'audit stockent des enregistrements d'audit au format binaire.

L'audit consiste à collecter des données sur l'utilisation des ressources système. Les données d'audit fournissent un enregistrement des événements système ayant trait à la sécurité. Ces données peuvent ensuite être utilisées pour déterminer la responsabilité quant aux actions survenant sur un hôte. Un audit réussi commence par deux fonctions de sécurité : identification et authentification. A chaque connexion, une fois qu'un utilisateur fournit un nom d'utilisateur et que l'authentification réussit, un ID utilisateur d'audit immuable est généré et associé à l'utilisateur et un ID de session d'audit unique est généré et associé au processus de l'utilisateur.

L'ID de session d'audit est hérité par tous les processus démarrés au cours de la session de connexion. Par défaut, certaines actions, telles que l'initialisation et la fermeture du système, sont toujours soumises à un audit.

Le service d'audit effectue les opérations suivantes :

- Surveillance des événements liés à la sécurité survenant sur l'hôte.
- Enregistrement des événements dans une piste d'audit.
- Détection des utilisations inappropriées et des activités non autorisées
- Examen des modèles d'accès et des historiques d'accès des individus et des objets
- Identification des tentatives de contournement des mécanismes de protection
- Détection de l'utilisation étendue d'un privilège survenant lorsqu'un utilisateur change d'identité

3.1. L'audit et la sécurité :

L'audit permet de détecter des violations de sécurité potentielles en révélant des modèles suspects ou anormaux d'utilisation du système. L'audit offre également un moyen de suivre desactions suspectes, permettant ainsi de remonter à un utilisateur particulier, ce qui a un effet dissuasif. Lorsque les utilisateurs savent que leurs activités sont auditées, ils sont moinssusceptibles de tenter des activités malveillantes.

La protection d'un système informatique, en particulier d'un système sur réseau, requiert des mécanismes permettant de contrôler des activités avant que les processus système ou les processus utilisateur ne commencent. La sécurité nécessite des outils permettant de surveillerles activités lorsque celles-ci se produisent. La sécurité requiert également des rapports d'activités après que les activités ont eu lieu.

La manière dont l'audit surveille les événements et génère des rapports ne peut pas empêcher les pirates d'entrer dans le système de manière non autorisée.

Cependant, le service d'audit permet par exemple de générer des rapports indiquant qu'un utilisateur spécifique a effectué certaines actions à une heure et une date données. Le rapport d'audit peut identifier l'utilisateur.

3.2. Fonctionnement de l'audit :

L'audit génère des enregistrements d'audit lorsque des événements donnés se produisent. Le plus souvent, les événements générant des enregistrements d'audit sont les suivants :

- Démarrage et arrêt du système
- Connexion et déconnexion
- Création ou destruction de processus et création ou destruction de threads
- Ouverture, fermeture, création, destruction, ou modification du nom d'objets
- Actions d'identification et d'authentification
- Modification d'autorisations par un processus ou un utilisateur
- Actions d'administration, telles que l'installation d'un package
- Applications spécifiques à un site

Les enregistrements d'audit sont générés à partir de trois sources :

- Par une application
- A la suite d'un événement d'audit
- A la suite d'un appel système de processus

Une fois recueillies, les informations pertinentes d'événement prennent la forme d'un enregistrement d'audit. Chaque enregistrement d'audit contient des informations qui identifient l'événement, la cause, l'heure et d'autres informations pertinentes.

3.3. Caractéristiques de l'audit :

Lors de la connexion initiale, l'audit est défini selon des caractéristiques, celles importants pour notre projet sont les suivantes :

- ID de session d'audit : l'ID de session d'audit est assigné lors de la connexion. Tous les processus enfants héritent de cet ID.
- ID utilisateur d'audit : un processus acquiert un ID utilisateur d'audit immuable à la connexion de l'utilisateur. Cet ID est hérité par tous les processus enfants qui ont été lancés par le processus initial de l'utilisateur. L'ID utilisateur d'audit permet d'appliquer la responsabilité. Même après qu'un utilisateur endosse un rôle, l'ID utilisateur d'audit reste le même. L'ID utilisateur d'audit enregistré dans chaque enregistrement d'audit permet de toujours retracer les actions jusqu'à l'utilisateur de connexion.

3.4. Analyse du Journal d'Audit :

L'objectif d'analyse du Journal d'Audit est la surveillance du système en quasi-temps réel (ce doit être un élément de la politique de sécurité). Cette contrainte a des conséquences fortes sur les méthodes d'analyse utilisées, qui doivent permettre de traiter très rapidement des volumes de données considérables. En plus le fichier d'audit doit donc être protégé contre tout utilisateur malicieux voulant effacer ou modifier les traces de ses opérations. L'analyse doit donc permettre de détecter toutes sortes de transgressions, depuis l'intrusion dans le système par un utilisateur non autorisé jusqu'aux abus de toutes formes imputables aux utilisateurs connus du système.

Dans ce qui va suivre nous allons analyser un ensemble de données de développement et d'évaluation des systèmes de détection d'intrusions qui rassemble la plupart des notions concernant les attaques informatiques et la plupart des fonctionnalités relatives à l'audit de sécurité qu'on vient de citer.

4. DARPA 1998 pour la détection d'intrusion:

Il y avait deux parties à l'évaluation des systèmes de détection d'Intrusion du DARPA1998: une évaluation hors ligne et une évaluation en temps réel. Dans l'évaluation hors ligne les systèmes de détection d'intrusion ont été testés en utilisant l'audit collectés et le trafic du réseau sur un réseau simulé. Les systèmes traitent ces données en mode batch et tentent d'identifier les sessions d'attaque dans le milieu des activités normales.

Les premières données d'apprentissage ont d'abord été disponibles en Février 1998. Ils contiennent le trafic réseau et les journaux d'audit. Après, un autre sous-ensemble de quatre heures, mis à la disposition en mai 1998. Le but de ces quatre heures est de fournir des données initiales pour les chercheurs du DARPA pour s'assurer que les données peuvent être lues correctement, et que les informations sont suffisantes à l'évaluation.

Ces quatre heures sont similaire aux données qui seront incluses dans la première semaine des données d'apprentissage, mais non identique. Durant 7 semaines d'entrainement, les données du réseau contiennent des sessions d'attaques dans le milieu des sessions normales. Et la même chose pour les données de test qui dure deux semaines.

4.1. Ensemble de Données DARPA 98 :

L'ensemble des données DARPA 98 étaient le premier corpus standard pour l'évaluation et la détection d'intrusion. L'évaluateur des systèmes de détection d'intrusions off -line DARPA travaille sous la direction du Laboratoire des Recherches de l'Armée de l'Air Américain et de la DARPA-IPTO (Defense Advanced Research Projects Agency – Information Processing Technology Office). Sept semaines de rassemblement de données d'entraînement avec des attaques étiquetées ont eu lieu, utilisées par la suite pour le développement des systèmes. Suivi de deux semaines de

rassemblement des attaques de teste, non étiquetées, utilisées pour une évaluation aveugle des systèmes développés.

Un ensemble d'ordinateurs de test produisaient les données en émulant des centaines d'utilisateurs agissant l'un sur l'autre dans des milliers de sites. Avec le trafic de masse, il y avait plus de 300 instances de 38 attaques différentes contre trois machines victimes opérant sous le noyau UNIX (les systèmes d'exploitation SunOS, Solaris, et Linux). Les données de test incluent des attaques créées spécifiquement pour l'évaluation, des attaques récentes, et les attaques des données d'entraînement. Les détails de l'évaluation DARPA 1998 peuvent être trouvés dans [Kendall, 99], [Lippmann & al, 00] et [MIT, 00]. La figure 2.1 montre le diagramme du réseau qui a été utilisé pour faire des expériences.

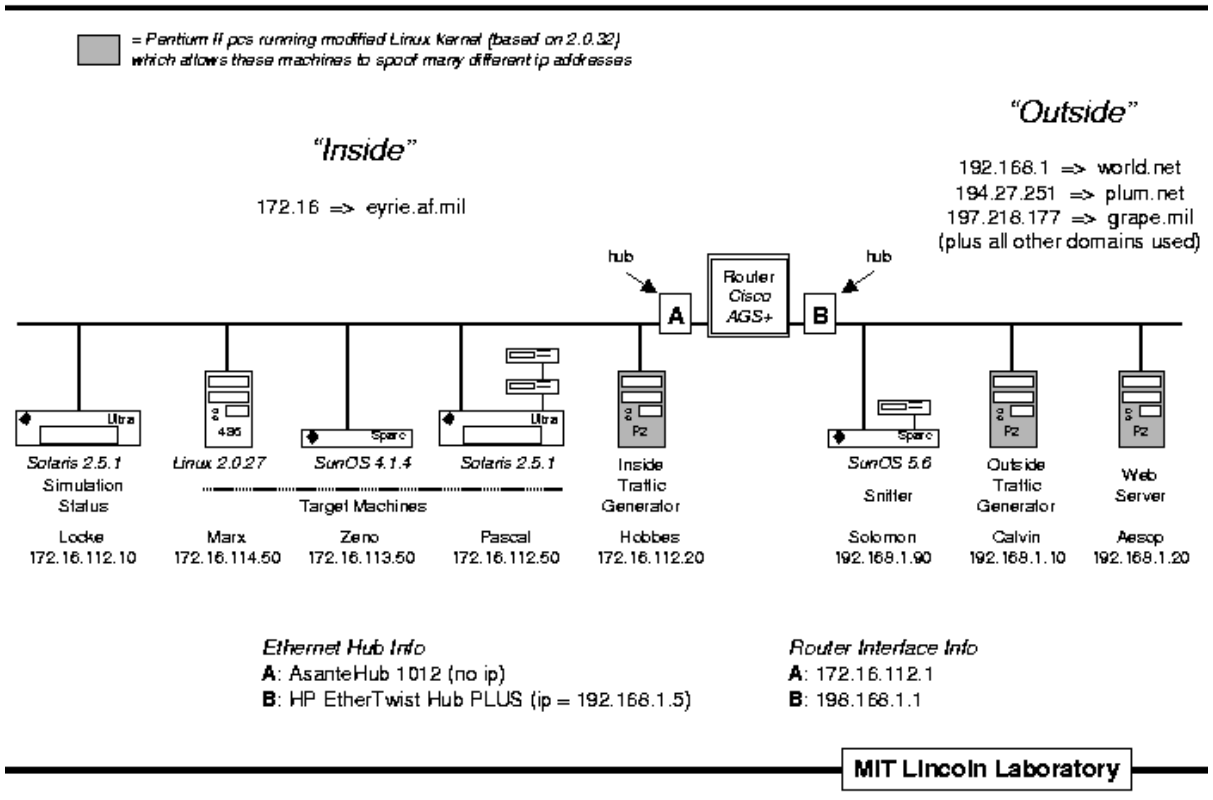


Fig. 2. 1. Le réseau simulé pour l'évaluation hors ligne [MIT, 00]

Les résultats de l'évaluation ont été analysés en traçant des taux de détection d'attaques contre les taux des fausses alarmes en utilisant des courbes caractéristiques du fonctionnement des récepteurs. Beaucoup de systèmes de détection d'intrusions pouvaient détecter les attaques utilisées dans les données d'entraînement avec une exactitude élevée (63% à 93%) et peu de fausses alarmes (10 par jour). Cependant, ces systèmes donnent des mauvaises performances avec les nouvelles attaques, particulièrement quand le mécanisme de ces attaques diffère de celui des attaques utilisées dans l'ensemble d'entraînement [Lippmann & al, 00].

Des logiciels dédiés émulent des centaines d'utilisateurs, utilisant des applications UNIX et des services réseau, ont été utilisés. Le trafic réseau produit par ces utilisateurs inclut l'envoie et la réception des emails , des fichiers en utilisant le FTP, l'accès à d'autres ordinateurs par l'intermédiaire de sessions TELNET, et la consultation des pagesWeb.

4.2. Rassemblement des Données

Il existe un tas d'informations qu'un système de détection d'intrusions peut utiliser pour la détection d'une attaque. Les N-IDS (systèmes de détection d'intrusion basés réseau), utilisent les informations rassemblées en sniffant le trafic réseau. Les H-IDS (systèmes de détection d'intrusion basés hôtes), utilisent les données rassemblées à partir des log d'un ordinateur personnel. Il y a quelques systèmes qui utilisent les deux sources d'informations.

Les expériences menées dans DARPA 98 permettent de collecter les informations nécessaires afin de satisfaire les entrées pour tous les systèmes de détection d'intrusions participants à l'évaluation. Un programme appelé «Tcpdump » [Lawrence, 99], fonctionnant sur le sniffer de Solaris, a été utilisé pour enregistrer le trafic dans le réseau de test. En plus, différents types de données d'hôtes tels les données de l'audit du

module de sécurité basic de Sun ***(BSM)*** ont été rassemblées à partir des machines victimes.

Après que toutes les données ont été rassemblées à partir du réseau de test, elles ont été enregistrées pour être publiées par la suite.

4.2.1. Les Listes BSM :

Dans les données DARPA, il y a des listes de fichiers. Ces dernières spécifient les sessions qui doivent être marqués par le système de détection d'intrusion. La liste de fichiers distingués est fournie par les données recueillies à l'aide d'audit de surveillance du logiciel *BSM* « Basic Security Monitoring »de Sun et par les données brutes recueillies par paquets à l'aide de *tcpdump*.

- colonne 1 : Index unique de session.
- colonne 2 : Date de début de la session avec la formejr/ms/an
- colonne 3 : Temps de début de la session avec la forme heur/min/sec
- colonne 4 : Durée de connexion
- colonne 5 : Type de service tel le FTP, le HTTP, le TELNET
- colonne 6 : Numéro de port source
- colonne 7 : Numéro de port destination
- colonne 8 : l'adresse de l'hôte source
- colonne 9 : l'adressede l'hôte destination
- colonne 10 :L'indication attribué à la session pour indiquer si une attaque ou non
- colonne 11 :Le nom d'attaque (optionnelle).

Une session séparée correspond à une ligne dans un fichier de liste de BSM. Chaque session correspond à une connexion TCP / IP entre deux ordinateurs. Une session peut être courte pour service tels que *Finger*, ou long comme le service *Telnet*. Les sessions peuvent également correspondre à des services utilisant des protocoles TCP ou UDP. Une session est unique et

spécifiée par l'heure du début, la durée,l'adresse IP dela source et de la destination et par les numéros des ports de la source et de la destination. Les neuf premières colonnes de la liste des fichiers fournissent des informations qui identifie la connexion TCP / IP. La performance des systèmes de détection d'intrusion seront évalués en utilisant les scores attribués à la dixième colonne. Chaque session est étiquetée avec le nombre 0 ou bien 1, qui représente une session normale ou une session d'attaque. La dernière colonne est facultative et peut être rempli d'un nom d'attaque. L'exemple 2.1 fournit un échantillon de fichier de la liste BSM des ensembles de données DARPA.

```
1773 06/01/1998 08:05:22 00:04:05 telnet 1941 23 135.008.060.182
172.016.112.050 1 format_clear
1782 06/01/1998 08:07:28 00:05:54 telnet 2064 23 135.008.060.182
172.016.112.050 1 ffb_clear
1800  06/01/1998 08:07:54 00:01:00 smtp 33028 25 172.016.112.050
135.008.060.182 0 –
1857  06/01/1998 08:09:33 00:01:05 telnet 2592 23 172.016.114.207
172.016.112.050 0 –
```

Exemple. 2. 1. Un échantillon du BSM Liste

5. La construction des ensembles de données :

En sept semaines de données d'apprentissage. Pendant, les 15 jours il n y a pas eu d'attaque. Durant les 15 jours, la septième semaine est libre d'attaque pendant les 5 jours successifs. Les quatre premiers jours sont utilisés pour construire les processus normaux pour l'ensemble de l'entraînement, et le cinquième jour est utilisé pour l'ensemble du test.

L'ensemble des données DARPA contiennent des fichiers BSM en format ASCII (généré à l'aide de la commande Praudit dans le système SOLARIS). Un exemple d'enregistrement dans un fichier BSM est montré dans l'exemple ci-dessus. Un enregistrement d'audit est une séquence de jetons d'audit. Un jeton

d'audit est un champ d'un enregistrement et décrit un attribut d'un événement d'audit. Chaque champ est séparé par des virgules.

Le type d'informations enregistrées pour chaque événement d'audit est défini par un ensemble de jetons d'audit. Chaque fois qu'un enregistrement d'audit est créé pour un événement, l'enregistrement contient certains ou tous les jetons définis pour l'événement. La nature de l'événement détermine quels jetons sont enregistrés. Dans l'exemple ci-dessus, chaque ligne commence par le nom du jeton d'audit. Le contenu du jeton d'audit suit le nom du jeton. L'ensemble des jetons d'audit *header, subject, path, trailer* et *return* constituent l'enregistrement d'audit *open*.

- ***Le Jeton header :*** Un enregistrement d'audit commence toujours par un jeton header. Ce jeton l'endroit où l'enregistrement d'audit commence dans le fichier d'audit. La quatrième position est l'appel système et la sixième position est l'horodatage de cet appel système, Ces deux champs sont importants dans l'extraction des appels système à partir de la liste de fichier BSM. Dans l'exemple, l'appel système est *« open»* et l'horodatage est « Mon Jun 01 08:07:50 1998».

Après le jeton *header*. Il y a beaucoup d'autres jetons comme *path*, *attribut, exec_args*, ..., mais ces jetons ne sont pas importants pour ce projet. Ce qui nous nous intéressons sont le jeton *subject* et *return*.

- ***Le Jeton subject:*** Il décrit un utilisateur qui exécute ou tente d'effectuer une opération. Le jeton subject est toujours retourné dans le cadre des enregistrements d'audit générés pour les appels systèmes, il contient aussi plusieurs champs, nous nous intéressons au ID du processus et l'ID de la session. Dans l'exemple, ID du processus est 440 et ID de la session est 439.
- ***Jeton return:*** Il contient l'état de retour de l'appel système et la valeur de retour du processus. Le jeton return est toujours retourné dans le

cadre des enregistrements d'audits générés pour les appels systèmes. Dans l'audit de l'application, ce jeton indique l'état de sortie et une valeur retournée par celui-ci, qui peut-être (*-1*) quand il y a une erreur, (*0*) quand l'opération se fait avec succès ou autre valeur, nous nous intéressons au nombre des erreurs qui se produisent. Un jeton *trailer* peut éventuellement conclure l'enregistrement. L'exemple suivant montre un enregistrement d'audit open:

```
header,86,2,open(2) - read,,Mon Jun 01 08:07:50 1998, + 247757034 msec
path,/opt/X11R6.1
subject,2056,2056,100,2056,100,440,439,24 2 finch.eyrie.af.mil
return,failure: No such file or directory,-1
trailer,86
```

Exemple. 2. 2. Un enregistrement d'audit (fichier bsm.praudit)

5.1. Extraction des processus normaux :

5.1.1. L'extraction des appels systèmes :

Les processus normaux sont choisis parmi les 5 jours continus de la septième semaine. Le script python n° 1 est utilisé pour s'exécuter par le biais du fichier *bsm.praudit* pour chaque jour de la septième semaine. Le script trouvera toutes les sessions dans une journée avec leur ID de session et l'heure du début. Pour chaque session il y a plusieurs processus. Chaque processus sera collecté avec son ID et ses appels systèmes.

Les résultats sont les fichiers d'appels systèmes pour chaque jour (*fichier. call).* Les fichiers d'appels systèmes contiennent tous les processus de la journée avec une séquence d'appels systèmes pour chaque processus qui correspond à une ligne. Les appels systèmes sont séparés par un caractère '|'. Les deux premiers nombres sont étiquette du processus (attaque 1 ou 0 normale) et l'ID du processus. Mais l'ID du processus n'est pas utile non plus donc un numéro zéro est mis à sa place. L'exemple ci-dessous présente un fichier d'appel système.

0|0|setaudit|open|ioctl|close|setgroups|setgroups|olds etgid|open|ioctl|close|open |setgroups|oldsetuid |fcntl|close|close|creat|fcntl|close|fcntl|close|close|chdir|old

0|0|open|fcntl|close|close|close|execve|open|mmap|open|mmap|mmap|munmap| mmap|close|open|mmap|mmap|munmap|mmap|close|open|mmap|mmap|munmap |mmap|mmap|close|open|mmap|close|open|mmap|mmap

Exemple. 2. 3. Un échantillon d'un fichier d'appel système

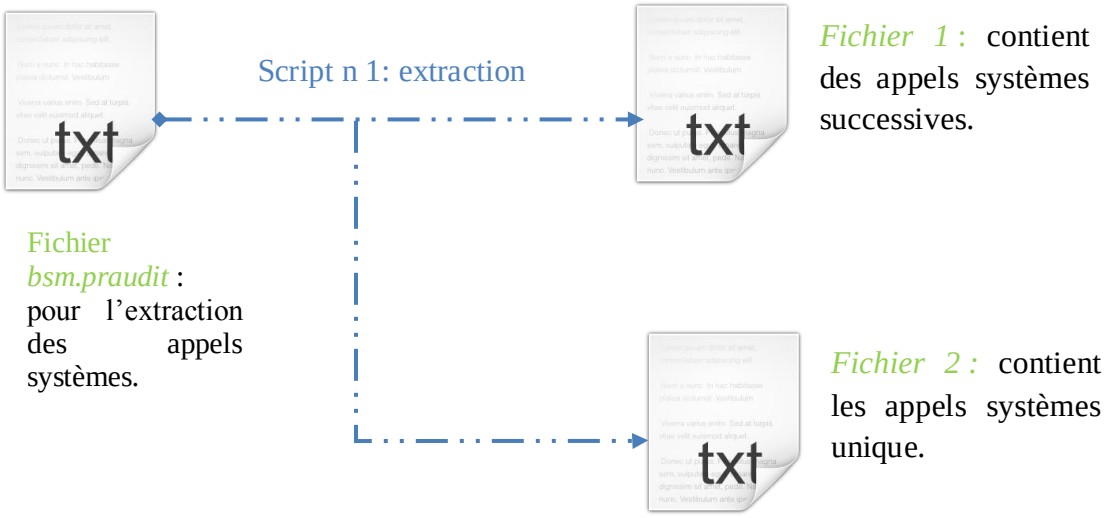


Fig. 2. 2. Etaps1 : l'extraction des appels systèmes

5.1.2. la transformation des appels systèmes :

L'étape suivante est la conversion de chaque processus en vecteur en utilisant l'approche basée sur la fréquence d'occurrence. Pour chaque processus, une liste des mots distingués est extraite. Pour chaque mot distingué dans un processus, le nombre d'occurrences de ce mot est compté. Le nombre d'éléments dans un vecteur est le nombre d'appel système.

Tous les processus sont extraits et convertis en matrice (*fichier. matr*). Deux processus sont différents quand ils ont des séquences différentes des appels systèmes ou quand ils ont des déférents nombres d'occurrences. Le processus de conversion génère également un fichier qui contient la liste des appels systèmes uniques avec leurs occurrences (*fichier. uniq*). Ce fichier d'appel système unique sera utilisé plus tard pour définir les caractéristiques du

vecteur quand nous construirons la matrice des données de teste. Les caractéristiques du vecteur des données d'entrainements ou les données de test sont identiques.

Après l'extraction d'occurrence, nous produisons les matrices, chaque élément dans un vecteur est l'appel système avec son nombre d'occurrences, nous remplaçons ces mots avec leur nombre d'occurrence qui correspondent à la position de l'élément dans le fichier d'appels systèmes uniques. À la fin de chaque vecteur, un nombre 0 pour un processus normal, 1 pour un processus anormal qui est l'étiquette du processus. L'opération de conversion du processus dans un vecteur puis la génération de la matrice est mise en œuvre dans le script n° 2.

rename:126, fork:2020, old setgid:214, fchdir:20, creat:185, getaudit:18, putmsg-connect:1338, old nice:3, chmod:42, getmsg:5291, kill:28225, close:63529, open:40490, setegid:3, logout:5, mmap:14798, mkdir:46, chown:41, access:1083, fchmod:434, exit:920, rmdir:32, login - local:3, old utime:53, munmap:5158, pathdonf:446, fcntl:12171, stat:9986, old setuid:224, lstat:1356, symlink:4, memcntl:8, ioctl:311536, readlink:97, rsh access:2, seteuid:581, setgroups:205, fchown:6, unlink:870, sysinfo:481, execve:799, audit:19, chdir:253, auditon:9, statvfs:24, login - telnet:6, setaudit:19, su:9, pipe:169, setpgrp:5116, putmsg:20146, link:83, vfork:56, return:100.

Exemple.2.4. Liste des appels systèmes uniques

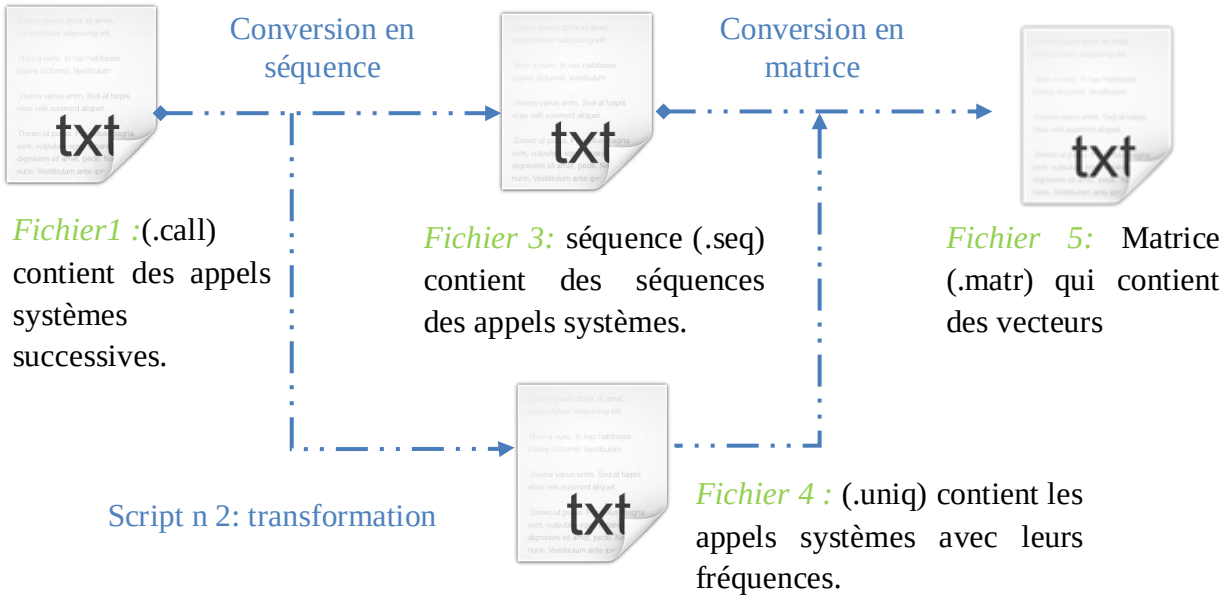


Fig. 2. 3. Étapes 2 : la transformation des appels systèmes

5.2. Extraction des attaques :

Pour chaque jour des sept semaines des données d'entraînements, il y a un fichier de liste BSM. Ce fichier contient une liste des sessions et de leurs étiquettes (attaque 1, normal 0). Un script python n°3 est utilisé pour passer à travers tous ces fichiers des listes de l'ensemble des sept semaines et de trouver une liste des sessions distincts d'attaque. Ces dernières sont différentes avec les différentes instances d'attaques. Ce qui diffèrent les sessions des uns et des autres, c'est quand, ils ont des noms de service de sessions différentes ou des noms de leurs attaques sont différentes.

Le nom de l'attaque comprend le nom attaque initiale et ses variantes. La liste ci-dessous (2,5) contient deux sessions d'attaques différentes. Ces deux sessions ont le même nom de service "Telnet", mais ils ont des noms d'attaques différents. La première session est «format_clear » la seconde est «ffb_clear».

```
1773  06/01/1998 08:05:22 00:04:05 telnet 1941 23 135.008.060.182
172.016.112.050 1 format_clear
1782  06/01/1998 08:07:28 00:05:54 telnet 2064 23 135.008.060.182
172.016.112.050 1 ffb_clear
```

Exemple.2.5. Deux sessions d'attaques différentes.

Trente-neuf sessions d'attaques ont été choisies pour l'extraction. Pour chaque session, il y a le champ du début de la session et aussi la durée. Ces champs seront utilisés dans le script python n °4 pour trouver les enregistrements BSM qui sont à l'intérieur de la durée de la session à partir des fichiers d'audit BSM. Les résultats sont la liste des versions courtes de fichier BSM, une version BSM abrégée pour chaque session d'attaque.

```
header,124,2,close(2),,Mon Jun 01 08:07:50 1998, + 277756930 msec
subject,2056,2056,100,2056,100,440,439,24 2 finch.eyrie.af.mil
return,success,0
header,80,2,sysinfo(2),,Mon Jun 01 08:07:50 1998, + 287758652 msec
subject,2056,2056,100,2056,100,440,439,24 2 finch.eyrie.af.mil
return,success,1
```

Exemple.2.6. Une version BSM abrégée.

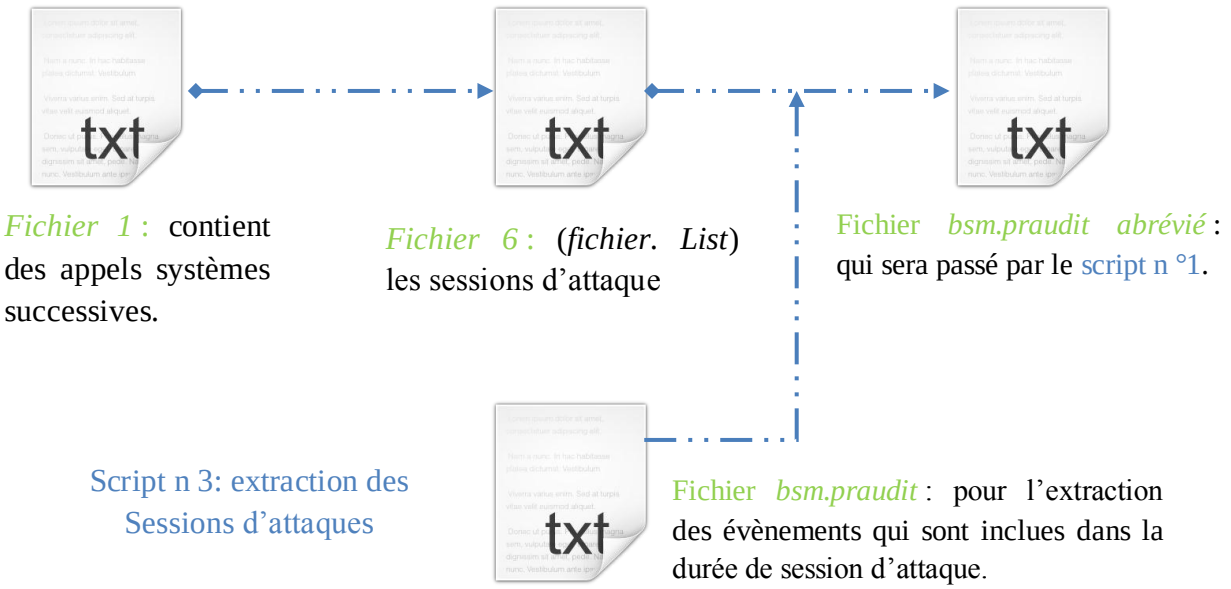


Fig. 2. 4. Étapes 3 : l'extraction des sessions d'attaque

5.3. Sélection des appels systèmes :

Un choix pertinent des attributs influant sur la qualité des résultats obtenus est suggéré. Les méthodes de fouille de données sont plus efficaces s'il existe des connaissances sur les attributs de domaine, sur la priorité de ces attributs, sur les attributs moins importants et les relations éventuelles existant entre eux. [Liao & Vemuri, 02] ont remarqué qu'un choix d'attributs dont les valeurs changent significativement dans les cas d'anomalies par rapport aux cas généraux est essentiel pour optimiser la détection d'intrusions. Nous avons utilisé des mesures issues de la théorie de l'information pour examiner l'utilité de chaque attribut, nous allons tout d'abord expliquer les concepts généraux de la théorie d'information, avant d'expliciter son application utilité au sein notre projet.

5.3.1. La théorie d'information :

La théorie de l'information est due à Shannon (vers 1948), avec bien sûr l'influence des grands théoriciens de l'informatique (Turing, Von Neumann, Wiener). À noter des convergences avec les travaux de Fisher. Le problème est celui de la communication entre une source et un récepteur : la source émet un message que le récepteur lit. On voudrait quantifier «l'information » que contient chaque message émis. Par exemple, il est clair que si l'émetteur dit toujours la même chose, la quantité d'information apportée par une répétition supplémentaire est nulle.

Le cas le plus simple est le suivant : le récepteur attend une information de type oui/non, le oui et le non étant a priori aussi vraisemblables l'un que l'autre. Lorsque la source transmet soit un oui soit un non, on considère que le récepteur reçoit une unité d'information (un bit). Autrement dit : une unité d'information, c'est quand on a apriori un ensemble de deux possibilités, et que l'une d'elles se réalise.

5.3.2. Cadre probabiliste :

Supposons que toutes les possibilités ne sont pas équiprobables mais qu'on sait que certaines, a priori, apparaîtront plus souvent que d'autres. L'idée est que les événements plus rares contiennent plus d'information.

Cette définition a bien la propriété qu'on attendait, à savoir que la survenue d'un événement rare contient plus d'information. Inversement, la survenue d'un événement certain (de probabilité) n'apporte aucune information.

La quantité d'information dépend plus de la distribution de probabilité que d'un événement x particulier.

5.3.3. La divergence de Kullback-Leibler :

En se basant sur les hypothèses de Wiener (1948), le nombre $h_i = -\ln(P(x))$ représente le nombre moyen de *nats* (ou de bits si le log est en base 2) nécessaire pour encoder la valeur x_i (Théorème de Shannon). L'entropie est alors définie comme étant l'espérance de la variable aléatoire $-\log(P(X))$ noté

$$H(X) = -\sum_{x}(P(X = x)\log(P(X = x)) \quad (2.1)$$

Elle représente le nombre moyen de *nats* nécessaire pour encoder des données provenant de la variable aléatoire X et évalue donc l'uniformité d'une variable. A présent, nous allons définir le gain d'information, qui est une mesure d'information particulière. Par exemple, en présence d'un second évènement y, nous nous interrogeons sur le gain d'information que représente y par rapport à x .Pour cela, nous construisons la partition $y_i = \{y \cap x_i\}$ $i \in I$ et les probabilités associées ($P(y_i) = P(x_i, y)$).

Le gain d'information de Kullback-Leibler apporté par l'évènement y sur l'évènement x est alors défini par la formule de l'équation 2.2 (Kullback (1952)).

$$KL\ (P(x)||P(x,y)) \ = \sum_{x} P(x_i) \log \frac{\mathrm{P}(x_i)}{P(x_i,y)} \qquad (2.2)$$

La divergence de Kullback-Leibler est une mesure asymétrique de l'éloignement de deux distributions. Elle évalue le nombre de *nats* supplémentaires pour encoder des données provenant de P en utilisant un code optimisé pour encoder des données provenant de Q (dans le cas précédent, elle évalue comment la connaissance de y permet de compresser les données xi).

$$KL\ (x||\mathrm{Q}) \ = \sum_{x} P(x_i) \log \frac{\mathrm{P}(x_i)}{Q(x_i)} \qquad (2.3)$$

Remarquons que cette formule a une forme très proche de celle de l'équation 2.2 et qui sera adaptée aux réseaux bayésiens (chapitre 4).

5.3.4. L'information mutuelle :

Chow & Liu (1968) ont introduit le critère d'information mutuelle IM, qui peut simplement être exprimé à l'aide de la distance de Kullback-Leibler par

$$IM(X;\ Y) \ = KL(P(X,Y)||P(X)P(Y)) = \ \sum_{x}\sum_{y} P(\mathrm{x},\mathrm{y}) \log \frac{\mathrm{P}(\mathrm{x},\mathrm{y})}{\mathrm{P}(\mathrm{x})\mathrm{P}(\mathrm{y})} \qquad (2.4)$$

$$IM(X;\ Y) \ = \ \sum_{x}^{s1}\sum_{y}^{s2} \frac{O_{ij}}{\mathrm{N}} \log \frac{O_{ij} \times\ N}{E_{ij}} \qquad (2.5)$$

Où O_{ij} représente le nombre de fois où la configuration x = i et y = j apparaît dans la base d'exemples, et N est le nombre de cas dans la base.

$$E_{i\ j} = \ \mathrm{N_i} \ \times \ \mathrm{N_j} \sum_{j=1}^{S2} O_{ij} \sum_{i=1}^{S1} O_{ij} \qquad (2.6)$$

L'information mutuelle représente alors le nombre de nats que deux variables ont en commun. Cette mesure permet donc de décider, s'il est

avantageux de connecter deux variables plutôt que de les conserver indépendantes.

5.3.5. L'information mutuelle moyenne conditionnelle :

Il est possible de définir une Information mutuelle conditionnelle simplement en utilisant la définition de l'information mutuelle et en remplaçant les probabilités par des probabilités conditionnelles. L'information mutuelle (moyenne) conditionnelle est alors définie par l'équation (2.7).

$$IM(X;\ Y|Z) = H(X|Z) - H(X|Y,Z)$$

$$IM(X;\ Y\,|\,Z) = \sum_{z} KL(P(X,Y,Z = z)||\ (P(X,Z = z)\ (P(\ Y,Z = z)\ (Z = z)$$

$$= \sum_{z}\sum_{x}\sum_{y} P(\mathrm{x},\mathrm{y},\mathrm{z}) \log \frac{P(\mathrm{x},\mathrm{y},\mathrm{z})}{\mathrm{P}(\mathrm{x}|\mathrm{z})\mathrm{P}(\mathrm{y}|\mathrm{z})\mathrm{P}(\mathrm{z})} \qquad (2.7)$$

$$= \sum_{z}\sum_{x}\sum_{y} P(\mathrm{x},\mathrm{y},\mathrm{z}) \log \frac{P(\mathrm{x},\mathrm{y}|\mathrm{z})}{\mathrm{P}(\mathrm{x}|\mathrm{z})\mathrm{P}(\mathrm{y}|\mathrm{z})}$$

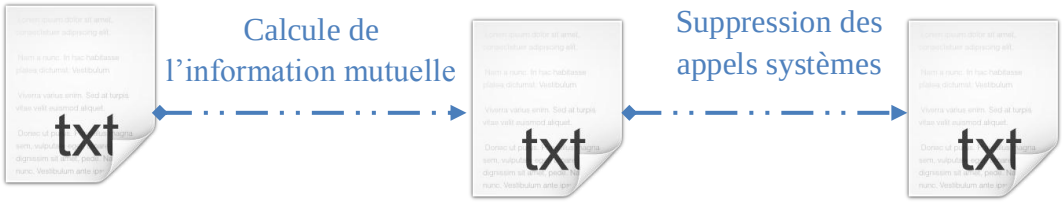
$$= \sum_{z}\sum_{x}\sum_{y} P(\mathrm{x},\mathrm{y},\mathrm{z}) \log \frac{P(\mathrm{x},\mathrm{y},\mathrm{z})\mathrm{P}(\mathrm{z})}{\mathrm{P}(\mathrm{x},\mathrm{z})\mathrm{P}(\mathrm{y},\mathrm{z})}$$

Une évaluation empirique de l'information mutuelle conditionnelle peut alors être donnée par l'équation suivante. Où O_{ijk} représente le nombre de fois où la configuration x = i et y = j et z = k apparaît dans la base d'exemples.

$$= \sum_{k=1}^{S3}\left[\sum_{i=1}^{S1}\sum_{j=1}^{S2} O_{ijk}\ \log \frac{\sum_{i=1}^{S1} O_{ijk}\ \sum_{j=1}^{S2} O_{ijk}}{(\sum_{j=1}^{S2} O_{ijk}\ )(\sum_{i=1}^{S1} O_{ijk})}\right]$$

Cette définition sera alors utilisée pour trois expérimentations la première est la sélection des attributs qui a plus grande information mutuelle. *(Code python : annexe A.1)* Avant, la sélection nous avons eu 53 appels systèmes plus le champ de *return* et après l'application de l'équation 2.7 sur tous les attributs

sachant que la classe et normale ou attaque nous avons choisis 44 appels systèmes. A savoir que nous avons fixé un seuil égale à 0.80 et si l'appel système égale ou inferieure a celui-ci, il sera écarté. La deuxième expérimentation est dans l'algorithme TAN (pour Tree Augmented Naive bayes classifier), La troisième expérimentation est dans l'algorithme FAN (pour Forest Augmented Naive bayes classifier) qui sont introduit dans le chapitre 4.



Fichier 5: Matrice (.matr) qui contient les vecteurs

Script n 4: sélection des appels système

Fichier 6 : Matrice (.matr) contient les appels systèmes utilitaires, après l'élimination de qui comporte moins d'information

Fig. 2. 5. Étapes 4 : la sélection des attribues pertinents

fcntl : 0.977948917829 mkdir : 0.976863148769 old utime : 0.976863148769 lstat : 0.976237261788 access : 0.974557163887 seteuid : 0.974239375012
kill : 0.970692322318 open : 0.968827030905 unlink : 0.968510298471 getaudit : 0.967776348361 fchdir :0.966072943851 fchmod : 0.965656157444 close:0.964277702782 setgroups :0.962845243575 munmap: 0.959213891071 old nice : 0.948311431312 fchown : 0.944151225831 audit : 0.926968432372 logout : 0.918830168324 chdir : 0.9185004373 chmod : 0.918485124333 putmsg-connect: 0.912838135039 sysinfo: 0.907204562321 creat:0.90121266389 rename : 0.897329930212 rsh access : 0.874578877846 memcntl : 0.869977886467 old setuid : 0.856717935941 stat: 0.847152941495 old setgid : 0.846588496436 login – local : 0.835218691696 readlink : 0.82874160368 symlink : 0.81791877945 fork : 0.806732022595 getmsg : 0.804818531219 mmap : 0.791630596869 execve : 0.756499100618 rmdir : 0.725298767747 ioctl : 0.584778816106 setegid : 0.564048733161 exit : 0.563099263806 chown : 0.551183419381 auditon : 0.538692826863 pathdonf : 0.533822796823

Exemple.2.6. les appels systèmes avec leur information mutuelle

5.4. Choix du langage de programmation :

Python est un langage interprété largement utilisé, Il a des fonctionnalités puissantes comme les listes, les tuples, les dictionnaires qui permet de traduire les idées en lignes de code directement. Python est un langage dynamiquement typé. La gestion de la mémoire est automatique. Le langage est multi-paradigmes, car il permet la programmation impérative classique, mais aussi la programmation fonctionnelle et la programmation orientée-objets. L'indentation est utilisée pour reconnaitre un bloc, cela est propre à Python. Ce qui est important, c'est que Le python est très efficace dans le cas de traitement des caractères et dans la gestion des fichiers, Alors que pour notre projet les données sont toujours enregistrées et extraites à partir les fichiers.

6. Conclusion

Dans ce chapitre nous avons donné un aperçu sur la variété des attaques qui ciblent les systèmes informatiques d'aujourd'hui. Quelques attaques se produisent en session unique dans laquelle toutes les actions se produisent proprement, alors que d 'autres sont constituées de plusieurs sessions étendues sur une longue période, pendant laquelle l'attaquant prend des mesures délibérées pour réduire au minimum les chances d'être détecté par un administrateur humain ou par un système de détection d'intrusions.

Dans quelques situations, le pirate informatique mène une attaque contre un système juste pour le plaisir personnel, tandis que dans d'autres cas, il est intéressé par la collecte d'informations confidentielles ou par l'endommagement du système cible. A ces fins, le développement d'un corpus de haute qualité pour la construction et l'évaluation des systèmes de détection d'intrusions requiert non seulement une variété dans les types des attaques, mais également une variété réaliste des méthodes utilisées par les attaquants. Les attaques incluses dans l'ensemble des données DARPA 98 ont

été développées pour fournir toutes ces informations sur les procédures d'attaques.

Ainsi cet ensemble de données sera utilisé par la suite pour construire et évaluer le système de détection d'intrusions l'objet de ce travail basé sur la classification qui sera traité dans le chapitre suivant, dans lequel nous allons présenter aussi la notion de la classification par l'algorithme de Plus Proches Voisins.

Chapitre 3 :

L’algorithme KNN

1. Introduction :

Pour améliorer la sécurité des réseaux, les administrateurs disposent de nombreux outils, dont les systèmes de détection d'intrusions. Ces outils ont connu un essor particulier au cours des dernières années, notamment en raison du nombre grandissant d'attaques.

Dans le contexte de la détection d'intrusion, le terme "classification" est souvent utilisé pour exprimer une "distinction" ou "identification" des attaques. Dans le but de faciliter l'analyse et l'identification, certains ont proposé de répertorier et classifier les attaques pour aider à gérer les incidents et pour traiter les informations d'audit [Lough, 01] Néanmoins, ce même argument qui a permis un déploiement massif des IDS, pose de sérieux problèmes pour les évaluateurs de tels systèmes. En effet, comment tester efficacement et avoir la certitude (prouver) que l'IDS se comporte correctement (par ex. génération d'alarme lors d'une tentative d'intrusion, pas de fausses alertes, etc.)

Malheureusement, malgré leur utilité, en pratique la plupart des IDS souffrent plus ou moins de deux problèmes : le nombre important de faux positifs et de faux négatifs. Les faux positifs (c'est-à-dire les fausses alertes) sont générés lorsque l'IDS identifie des activités normales comme des intrusions, alors que les faux négatifs correspondent aux attaques ou intrusions qui ne sont pas détectées (aucune alerte n'est générée).

Pour bien traiter ce problème, il nous semble nécessaire de commencer par une analyse approfondie de l'algorithme de classification choisis, et de mettre ainsi en évidence les détails de son application au domaine de détection de l'intrusion.

2. Les algorithmes de classification :

Le data mining offre une très grande variété de techniques et d'algorithmes de fouille de données. Ces algorithmes ont des origines diverses ; Certains sont

issus des statistiques (régression…), d'autres de l'intelligence artificielle (réseaux de neurones, arbres de décision…), certains encore s'inspirent de la théorie de l'évolution (algorithmes génétiques…). Cette combinaison de technologies facilite la résolution, la compréhension, la modélisation et l'anticipation des problèmes.

Le data mining est un ensemble de techniques complémentaires dédiées à différentes tâches. Ces technique sont partagées, principalement, entre la classification automatique (supervisée et non supervisée) et la recherche d'associations.

La classification supervisée est une tâche largement appliquée dans la vie courante. En effet, il existe une multitude de problèmes qui entrent dans ce cadre, parmi lesquels on trouve la reconnaissance des caractères manuscrits, la reconnaissance des paroles, la catégorisation des textes, la détection des spams, l'aide au diagnostic médical, la Bioinformatique…etc.

A la différence de la classification non supervisée où les groupes d'objets sont découverts à posteriori, les techniques de classification supervisée s'appliquent lorsqu'on veut rattacher un nouvel objet (observation) à une classe qui est choisie parmi un ensemble de classes connues. Cette étape suit, généralement, l'étape de clustering.

La plupart des algorithmes de classification tentent de trouver un modèle (une fonction mathématique) qui explique le lien entre les données d'entrée et les classes de sortie. Un ensemble d'apprentissage (ensemble de classes) est donc utilisé par l'algorithme. Cette méthode de raisonnement est appelée inductive car on induit la connaissance (le modèle) à partir des données d'entrée (objets à classifier) et des sorties (leurs classes). Grâce à ce modèle, on peut alors prédire les classes de nouvelles données. Le modèle est bon s'il permet de bien prédire. Une autre approche qui n'est pas moins intéressante, c'est le raisonnement à partir des cas. Ces algorithmes ne cherchent pas à calculer le modèle mais à

trouver, pour l'objet à classifier, un ou plusieurs cas similaires déjà résolus pour en déduire la classe, l'algorithme kNN adopte ce principe. Dans ce qui suit, nous allons décrire et détailler le principe d'algorithme de classification KNN.

3. L'algorithme KNN :

K Nearest **N**eighbor (**PPV P**lus **P**roches **V**oisins) est une méthode dédiée à la classification qui peut être étendue à des tâches d'estimation. La méthode PPV est une méthode de raisonnement à partir de cas. Elle part de l'idée de prendre des décisions en recherchant un ou des cas similaires déjà résolus en mémoire.

Contrairement aux autres méthodes de classification (arbres de décision, réseaux de neurones, algorithmes génétiques), il n'y a pas d'étape d'apprentissage consistant en la construction d'un modèle à partir d'un échantillon d'apprentissage. C'est l'échantillon d'apprentissage, associé à une fonction de distance et d'une fonction de choix de la classe en fonction des classes des voisins les plus proches, qui constitue le modèle. L'algorithme générique de classification d'un nouvel exemple par la méthode PPV est :

- Paramètre : le nombre k de voisins
- Donnée : un échantillon de m enregistrements classés ($\vec{x}$, c($\vec{x}$))
    1. Entrée : un enregistrement $\vec{y}$déterminer les k plus proches enregistrements de $\vec{y}$
    2. combiner les classes de ces k exemples en une classe c
- Sortie : la classe de $\vec{y}$ est c ($\vec{y}$)=c

3.1. L'algorithme K Plus proches voisins:

3.1. Définition de la distance :

Le choix de la distance est primordial au bon fonctionnement de la méthode. Nous avons choisis la mesure de cosinus : qui définir la similarité entre les vecteurs.

$$\cos(\mathrm{V1},\mathrm{V2}) = \frac{V_{1J} \,.\, V_{2J}}{||\mathrm{V1}||.||\,\mathrm{V2}||}$$

On considère une séquence d'appel système composée de n items comme un vecteur de n-dimensions. Le cosinus fait l'angle entre des vecteurs normés par leur longueur.

3.2. Sélection de la classe :

L'idée de la méthode est la recherche de cas similaires au cas à résoudre et d'utiliser les décisions des cas proches déjà résolus pour choisir une décision. La méthode la plus simple est de rechercher le cas le plus proche et de prendre la même décision. C'est la méthode 1-PPV (1-NN) du plus proche voisin. Si cette méthode peut fournir de bons résultats sur des problèmes simples pour lesquels les points (les enregistrements) sont bien répartis en groupes denses d'enregistrements de même classe, en règle générale, il faut considérer un nombre de voisins plus important pour obtenir de bons résultats.

Nous supposons avoir déterminé les *k* voisins ($x_1^{\rightarrow}$, c ($x_1^{\rightarrow}$)), ..., ($x^{\rightarrow}c(x_k^{\rightarrow})$) d'un enregistrement $y^{\rightarrow}$ auquel on souhaite attribuer une classe c ($y^{\rightarrow}$). Une première façon de combiner les *k* classes des *k* voisins les plus proches est le *vote majoritaire*. Elle consiste simplement à prendre la classe majoritaire. Dans le cas de deux classes, on choisit une valeur de *k* impaire.

Une seconde façon est le *vote majoritaire pondéré*. Chaque vote, c'est-à-dire chaque classe d'un des *k* voisins sélectionnés, est pondéré. Soit $x_i^{\rightarrow}$ le voisin considéré, Le poids de c ($x_i^{\rightarrow}$) est inversement proportionnel à la distance entre l'enregistrement $y^{\rightarrow}$ à classer et $x_i^{\rightarrow}$.

3.3. Critiques de la méthode

- **pas d'apprentissage :** c'est l'échantillon qui constitue le modèle. L'introduction de nouvelles données permet d'améliorer la qualité de la méthode sans nécessiter la reconstruction d'un modèle. C'est une différence

majeure avec des méthodes telles que les arbres de décision et les réseaux de neurones.

- **clarté des résultats :** bien que la méthode ne produit pas de règle explicite, la classe attribuée à un exemple peut être expliquée en exhibant les plus proches voisins qui ont amené à ce choix.
- **tout type de données :** la méthode peut s'appliquer dès qu'il est possible de définir une distance sur les champs. Or, il est possible de définir des distances sur des champs complexes tels que des informations géographiques, des textes, des images, du son. C'est parfois un critère de choix de la méthode PPV car les autres méthodes traitent difficilement les données complexes. On peut noter, également, que la méthode est robuste au bruit.
- **nombre d'attributs :** la méthode permet de traiter des problèmes avec un grand nombre d'attributs. Mais, plus le nombre d'attributs est important, plus le nombre d'exemples doit être grand. En effet, pour que la notion de proximité soit pertinente, il faut que les exemples couvrent bien l'espace et soient suffisamment proches les uns des autres. Si le nombre d'attributs pertinents est faible relativement au nombre total d'attributs, la méthode donnera de mauvais résultats car la proximité sur les attributs pertinents sera noyée par les distances sur les attributs non pertinents. Il est donc parfois utile de d'abord sélectionner les attributs pertinents.
- **temps de classification :** si la méthode ne nécessite pas d'apprentissage, tous les calculs doivent être effectués lors de la classification. Ceci est la contrepartie à payer par rapport aux méthodes qui nécessite un apprentissage (éventuellement long) mais qui sont rapides en classification (le modèle est créé, il suffit de l'appliquer à l'exemple à classifier). Certaines méthodes permettent de diminuer la taille de l'échantillon en ne conservant que les exemples pertinents pour la méthode PPV, mais il faut, de toute façon, un nombre d'exemples suffisamment grand relativement au nombre d'attributs.

- **stocker le modèle :** le modèle est l'échantillon, il faut donc un espace mémoire important pour le stocker ainsi que des méthodes d'accès rapides pour accélérer les calculs.
- **distance et nombre de voisins** : les performances de la méthode dépendent du choix de la distance, du nombre de voisins et du mode de combinaison des réponses des voisins. En règle générale, les distances simples fonctionnent bien. Si les distances simples ne fonctionnent pour aucune valeur de *k*, il faut envisager le changement de distance, ou le changement de méthode !

notre travail consiste à l'implémentation d'un système de détection d'intrusion qui applique ces algorithmes à une quantité suffisante de données d'audit normales ou anormales pour générer un classificateur capable d'étiqueter comme appartenant à la catégorie normale ou anormale de nouvelles données d'audit.

3.4. Mise en œuvre de la méthode pour la détection d'intrusion:

La méthode ne nécessite pas de phase d'apprentissage. Le modèle sera constitué de l'échantillon d'apprentissage, de la distance et de la méthode de combinaison des voisins.

Il faut choisir l'échantillon, c'est-à-dire les attributs pertinents pour la tâche de classification considérée et l'ensemble des enregistrements. Il faut veiller à disposer d'un nombre assez grand d'enregistrements par rapport au nombre d'attributs et à ce que chacune des classes soit bien représentée dans l'échantillon choisi.

Parfois, nous devons classer les données en deux groupes, par exemple groupe normal et groupe anormal. Parmi les mécanismes d'apprentissage, on a l'approche comportementale qui propose de décrire le comportement «normal»

ou «usuel » d'un utilisateur. Toute déviation par rapport à ce comportement normal est considérée comme une intrusion. Dans ce projet, les données sont représentées par des vecteurs des appels systèmes.

4. Détection d'intrusion par signature :

L'approche par détection de signatures d'attaque repose sur la constitution d'une base de connaissances contenant une description des attaques connues. Le module de détection d'intrusions analyse ensuite les traces d'activités journalisées à la recherche des différents événements correspondant à la description des attaques de la base de connaissances. Ces événements représentent la signature de l'attaque. (Code python de kNN dans Annexe A.2)

Construire l'ensemble d'entrainement
Pour chaque x dans l'ensemble se test **faire**
    Pour chaque processus Dj dans l'ensemble d'entrainement faire
        Calculer Sim (D_j, x)
        Si Sim (D_j, x) égale à 1.0
            X est anormal and **exit**
        Trouver k les plus grandes similarités
        Pour chaque voisin **faire**
            Si voisin j >= seuil **alors**
                Anormale <- anormale + vote
            Sinon
                Normale <- normale + vote
        Si anormal > = normale **alors**
            X est anormale
        Sinon
            X est normale

Algorithme 3.2. KNN pour la détection d'intrusion par signature

4.1. Les données d'apprentissages :

Les signatures d'attaques pour la constitution de la base de données de l'IDS par signature ont été sélectionnées comme suite :

L'expérience menée a considérée deux variantes, les attaques du test sont des mêmes signatures sauvegardées dans la base, la deuxième variante, propose de nouvelles attaques ; les données des tests inclues aussi les processus normaux. Le tableau suivant résume les données engagées dans cette expérience :

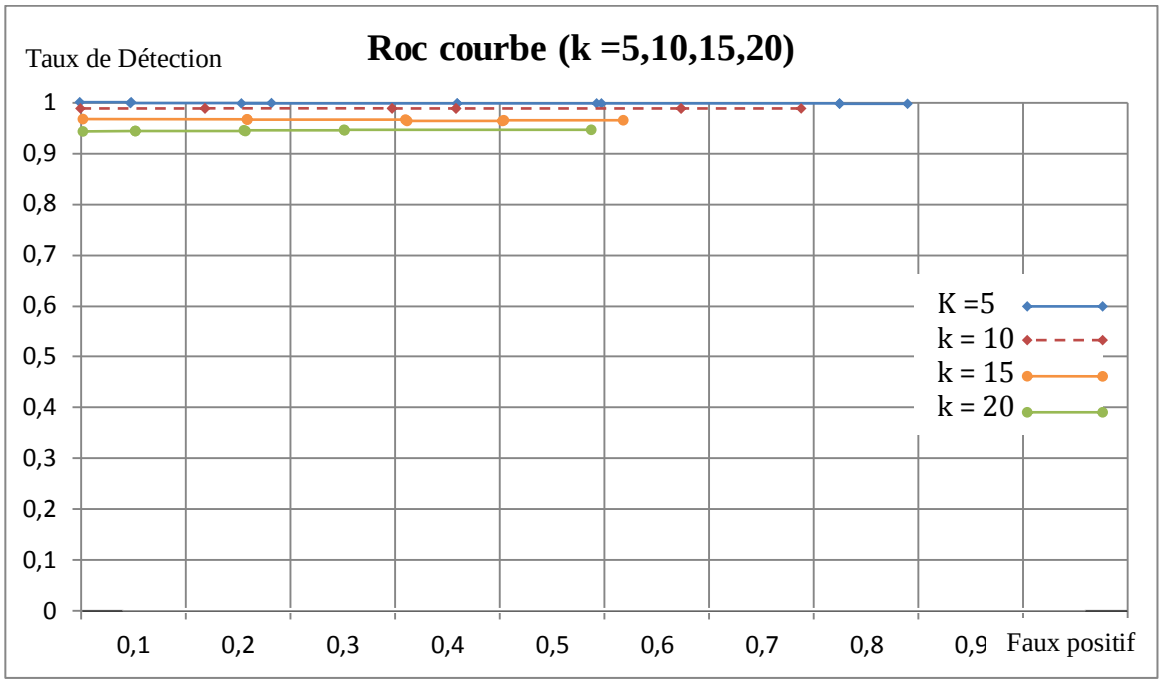
	Les signatures	Tests	
		attaques	**Normale**
Expérimentation 1	51 instances	51	393
Expérimentation 2	31 instances	51	393

Tableau 3.1. Les données sélectionnées pour l'IDS par signature

Deux paramètres peuvent être manipulés pour ajuster la classification : le nombre des voisins k, et le seuil de décision si le processus est normal ou non.

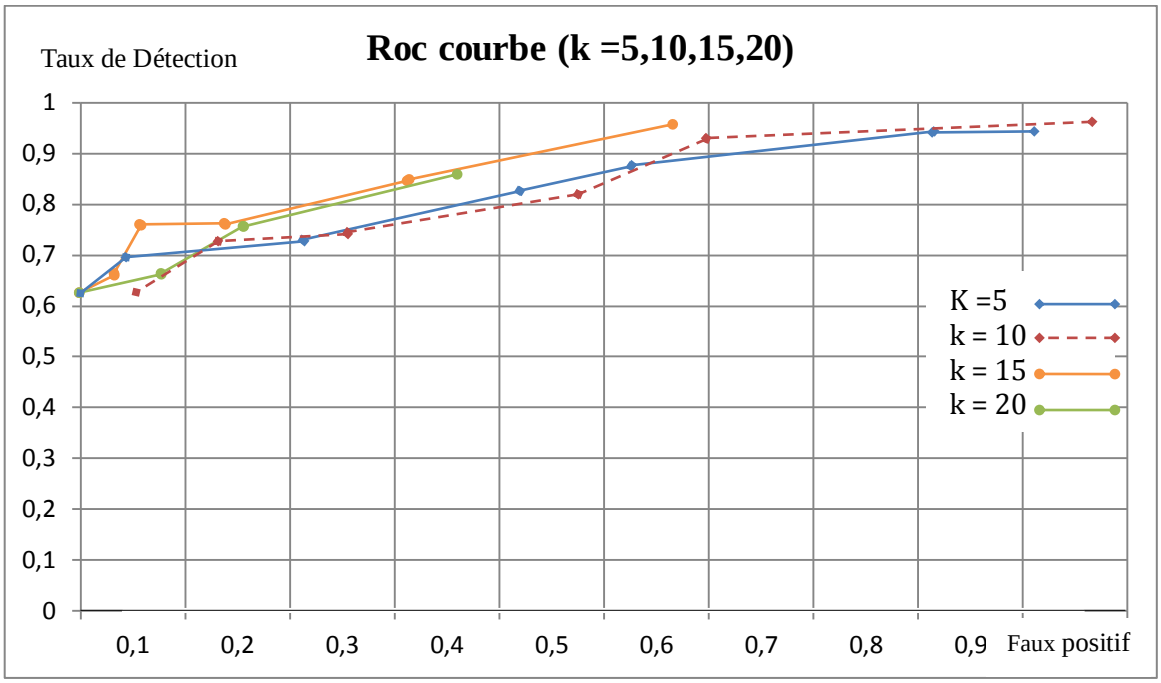
La similarité est comparée au seuil, si elle est plus grande donc l'IDS décide que le processus est anormal, et vice-versa.

En variant ces paramètres, nous pouvons dessiner la courbe ROC (Receiver Operating Characteristic) afin de comparer les performances entre différents k. Quatre valeur du k (k=5, 10, 15, 20) ont été choisi. Pour chacune des k, on a utilisé 7 seuils (0.80, 0.85, 0.90, 0.93, 0.95, 0.97, 0.99) pour décider si le processus est normal ou anormal. (Tableau détaillé des résultats dans Annexe B.2)



ROC courbes 3.1. KNN par signature sans les nouvelles instances d'attaques

D'après la courbes ROC1 de la première expérimentation, un constat immédiat est que, le taux de détection est à 100% (à 0.99 avec k = 10, taux de détection est 100% avec un faux positif = 0.21 %) chose qui peut s'expliquée par le faite que l'IDS contient toutes les signatures de toutes les instances d'attaques utilisées dans cette expérimentation. Le faux positif diminue sensiblement avec l'augmentation du nombre des voisins k ainsi qu'avec le seuil (à 0.95 avec k = 20, taux de détection est 100% avec un faux positif = 0.21 %). Sa parait que cette façon de faire est très efficace, c'est ce qui motive l'adoption de cette technique de détection d'intrusion par signature par la plus part des IDS commerciaux.



ROC courbes 3.2. KNN par signature avec les nouvelles instances d'attaques

Dans la courbe ROC2, nous constatons comme si l'efficacité cette technique est remise en question. Donc le faite d'introduire de nouvelles instances d'attaques dont les signatures ne sont pas incluses dans la base de l'IDS, affaiblie le taux de détection. Nous remarquons que le taux de détection ainsi que le faux positif diminuent avec l'augmentation du seuil et du nombre des voisins (à 0.80 avec k = 5, taux de détection est 94.11% avec un faux positif = 83.05%) et pour (à 0.99 avec k = 20, taux de détection est 62.74% avec un faux positif = 0.21 %).

La spécificité de cette approche est qu'elle ne peut pas détecter les nouvelles attaques, ce qui est attendue car elle se base sur ce qui est connu pour enfin bien le détecté. Par contre ce qui n'est pas connu ne peut pas normalement être détecté (31 de 51 c'est 60%). Toutefois, si la signature n'est pas suffisamment précise, des faux positifs peuvent apparaître. Par ailleurs, si la signature est trop contrainte, des variantes même très proches de l'attaque risquent de ne pas être détectées. (Tableau détaillé des résultats dans Annexe B.2)

Face aux difficultés de l'approche par détection de signature, l'approche comportementale présente certains avantages. En effet, elle repose sur l'apprentissage du comportement normal en considérant que toute déviation de ce comportement est suspecte. Donc, on n'a pas besoin de posséder une connaissance de l'attaque qui peut être menée sur le système. Au lieu d'une base de données préétablie des signatures, nous allons plus en avant, et équiper l'IDS avec la faculté d'apprentissage, ainsi il saura traiter les nouveautés que ce soit de nouvelles attaques ou bien des processus normaux.

5. Détection d'intrusion comportementale :

L'ensemble de données DARPA contient la plupart du temps les données normales, et rarement les données anormales. Cependant, les données anormales sont à la fois différentes et rares, ils apparaissent comme des valeurs bruits dans les données qui peuvent être détectées.

La vraie question est de savoir comment capturer le comportement d'un processus, jusqu'à arriver à décider si un nouveau processus est normal ou non. Cette notion de comportement demande une compréhension approfondie, pour KNN le, comportement normal s'apprend du comportement d'un ensemble assez large de processus connu à être normaux (environ 1700 processus normaux).

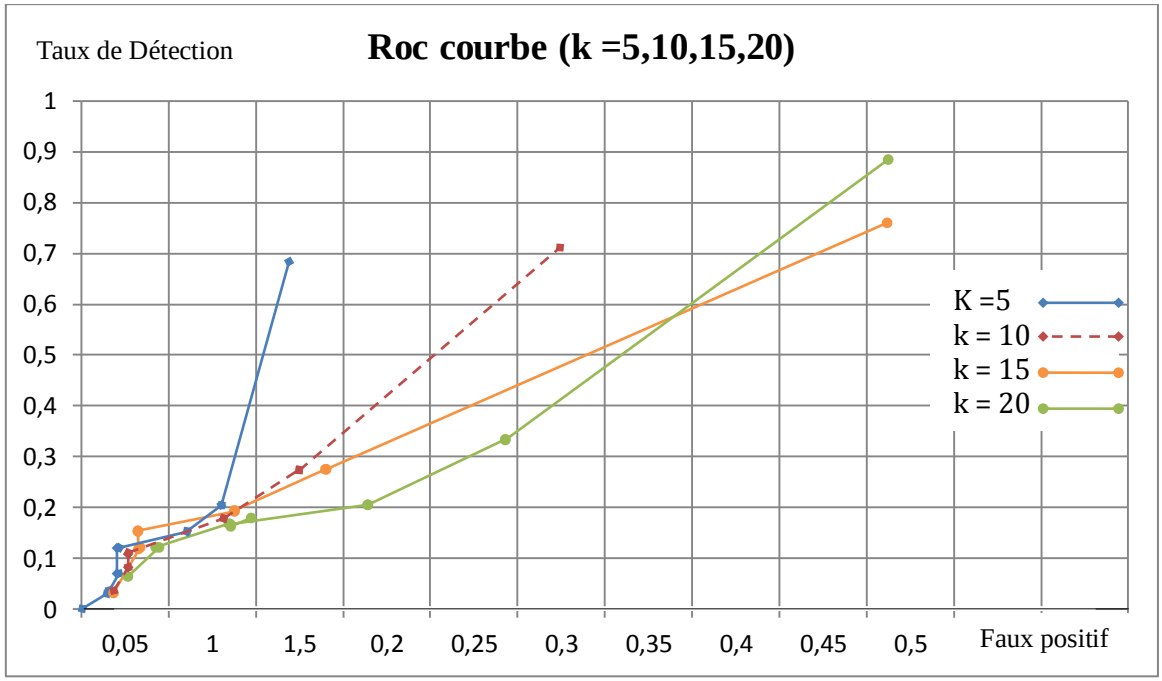
Cette phase, souvent appelée phase d'apprentissage quoiqu'elle est n'est pas si distinguée de la phase de décision, puisqu'à chaque décision le knn consulte tout l'ensemble d'entrainement. Les données sélectionnées sont exposées dans le tableau suivant :

	L'Apprentissage (Normal)	**Tests**	
		Attaques	Normale
KNN	1700 instances	51	393

Tableau 3.2. Les données sélectionnées pour l'IDS comportemental

Construire l'ensemble d'entrainement
Pour chaque x dans l'ensemble se test **faire**
    Pour chaque processus Dj dans l'ensemble d'entrainement faire
        Calculer Sim (D_j, x)
        Si Sim (D_j, x) égale à 1.0
            X est normal and **exit**
        Trouver k les plus grandes similarités
        Pour chaque voisin **faire**
            Si voisin j >= seuil **alors**
                Normale <- Normale + vote
            Sinon
                Anormale <- Anormale + vote
        Si Anormal > = Normale **alors**
            X est anormale
        Sinon
            X est normale

Algorithme.3.3 kNN pour la détection d'intrusion comportementale



ROC courbes 3.3. KNN comportementale

Les deux paramètres qui sont le k et le seuil prennent les mêmes valeurs précédentes, après avoir tracé les courbes ROC correspondantes.

De la courbe ROC3 on peut constater l'influence du k et du seuil sur le taux de détection et le faux positif, de (à 0.80 avec k = 5, taux de détection est 0.00% avec un faux positif = 0.00 %) ceci s'explique simplement par le fait que tous les processus qu'ils soient normaux ou anormaux sont classés normal. Puis qu'avec un seuil si faible nous avons laissé une large marge d'intervalle pour la similarité. Pour bien distingués les processus entre eux, nous augmentons le nombre du k et le seuil, l'impact est visible, le taux de détection augmente avec l'augmentation de k et le seuil. Jusqu'à arriver (à 0.99 avec k = 20, taux de détection est 80.37% avec un faux positif = 6.87 %). Le taux de détection n'est pas si haut, seulement comparant avec celui de la méthode précédente, celle de la détection par signature, nous remarquons qu'il y a une augmentation de presque 20% (80%-62%) ce qui veut dire que nous détectons plus d'attaques. Malgré que nous n'avons aucune connaissance à priori des attaques possibles. Ce qui nous pousse à bien explorer cette approche.

Les 20% de processus anormaux qui ne sont pas détectés, est dû au fait que leur comportement est similaire à ceux normaux avec un seuil plus que 0.99. Pour mieux discerner le comportement normal de celui anormal, nous puiserons du côté du raffinement de la discrimination. L'exploration fine de l'ensemble d'apprentissage, pourrait nous fournir certaines idées pour une représentation plus élaborer et plus fine. Cette idée est bien exploitée par la technique if-idf que nous exposerons ci-après. (Tableau détaillé des résultats dans Annexe B.1)

6. La technique IF-IDF :

Cette approche coexiste deux approches, celle basée sur la fréquence d'occurrences et l'autre sur la valeur de discrimination.

6.1. Approche basée sur la fréquence d'occurrence :

Les appels systèmes possèdent chacun une fréquence d'apparition dans les processus. Ces fréquences permettent de choisir les représentants. Le principe est simple, plus l'appel système est fréquent mieux il est classé.

Processus = [(sys-call1, freq1), (sys-call2, freq2),…, (sys-calln, freqn)]

6.2. Approche basée sur la valeur de discrimination :

Un appel système discrimine un processus s'il le distingue bien des autres processus, c'est-à-dire qu'il est nettement plus fréquent dans ce processus que dans les autres.

Le calcul de la valeur discriminante consiste à comparer la similarité entre les processus avec et sans l'appel système considéré. Cela nous permet de mettre en exergue l'influence de l'appel système sur la collection.

La pondération des appels systèmes permet de mesurer l'importance d'un appel système dans un processus. Le choix de la formule de pondération est crucial, quand à la pertinence des résultats de l'interrogation.

Prendre en compte la fréquence d'occurrence des termes et la valeur de discrimination revient à combiner les avantages des deux approches ci-dessus mentionnées. IF_IDF signifie « term frequency * inverted document frequency».

IF: fréquence d'occurrence du terme dans un processus. Avec la normalisation d'IF :

$$IF = f(c, p) / max [f(c, p)] \text{ : term frequency}$$

Où / max [f(c, p)] est la fréquence maximale des termes dans le document.

IDF: inverted document frequency:

IDF = log ((N / n) +1): inverted document frequency

Où : N : est le nombre des processus dans la collection.

n: est le nombre de processus contenant l'appel système.

Et donc, nous obtenons la valeur de l'IF-IDF :

IF_IDF = IF* IDF est la multiplication l'IF par l'IDF .Ce type de formule met en évidence le pouvoir de discrimination des processus et l'importance de l'appel système dans un processus.

6.3. La représentation du processus avec IF-IDF :

Le KNN s'applique sur un espace euclidien multidimensionnel. Un vecteur représente un processus. Au-dessus les coordonnées du vecteur étaient les fréquences d'occurrence des appels systèmes. Avec l'utilisation de la technique IF-IDF, le processus sera représenté avec un vecteur dont les coordonnées sont calculées avec la formule du IF-IDF appliquée à chacun des appels systèmes. (Code python d'IF-IDF dans Annexe A.3)

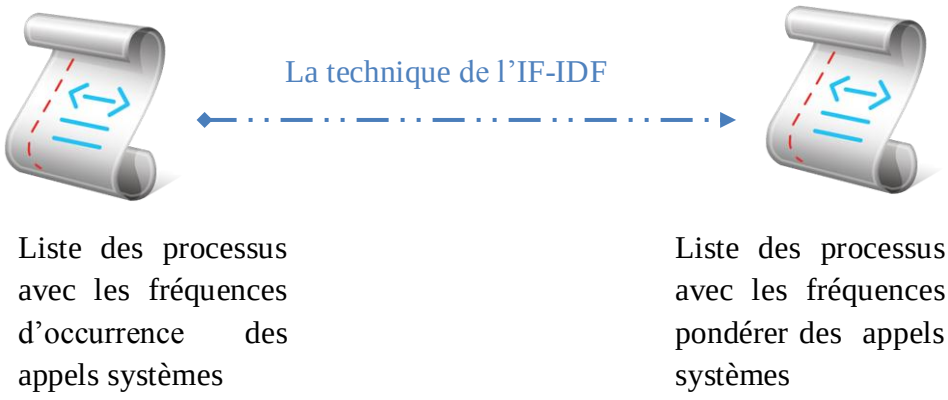
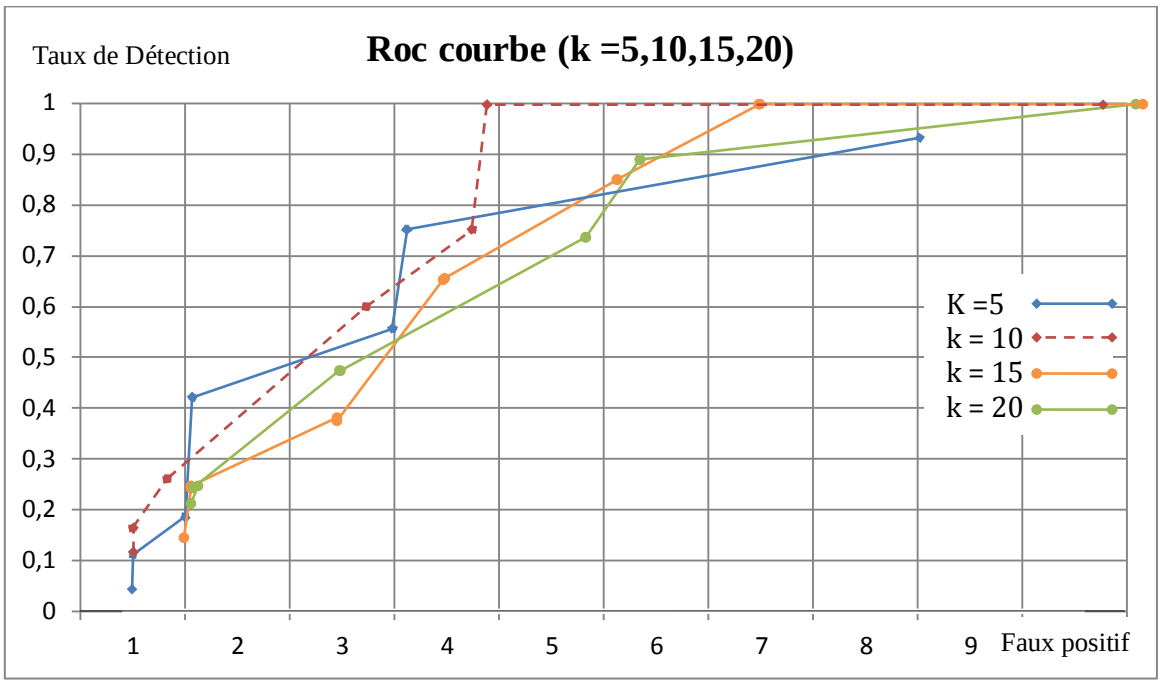


Fig. 3. 1. Étapes 5 : la technique de IF IDF

6.4. L'apport de l'IF-IDF pour le kNN



ROC courbes 3.4. KNN comportementale + IF-IDF

Le KNN est appliquée sur le nouvel espace des vecteurs calculé avec l' IF-IDF qui représentent les processus. Un constat important qui attire l'attention dans la courbe ROC4, est le faite que le taux de détection est largement amélioré allant de (à 0.99 avec k = 5, taux de détection est 94.11% avec un faux positif = 9.80 %) jusqu'à atteindre 100% pour un seuil de 0.97 et 0.99 avec k >= 10 ; les faux positifs sont beaucoup plus influencés par le k, et donc nous pouvons extraire la meilleur solution pour le bon compromis entre k et le seuil : (à 0.97 avec k = 10, taux de détection est 100% avec un faux positif = 5.08 %). (Tableau détaillé des résultats dans Annexe B.1)

7. Conclusion

Les IDS suivent deux approchent distinctes, la détection par signature et celle basée sur le comportement. Nous avons utilisé la méthode KNN pour expérimenter les deux approches. Malgré que les IDS commerciaux adoptent l'approche par signature, nous remarquons que le recours à la détection comportementale est plus qu'un choix, elle peut nous apporter de meilleures

performances surtout vis-à-vis les nouvelles attaques non répertoriés. Dans un milieu dynamique les choses changent, la seule issue est l'apprentissage. La représentation des données est crucial chose qui a été confirmée avec l'utilisation de l'IF-IDF vis-à-vis la représentation par la fréquence d'occurrence tout simple.

Certes, les résultats fournis par le kNN en cas de détection comportementale sont nettement meilleurs que ceux du kNN dans le cas de détection par signature, néant-moins ils ne sont pas totalement satisfaisants. De plus la complexité algorithmique est de l'ordre de O(N), avec le N est le nombre de processus. Ce qui favorise le KNN dans le cas de détection par signature au fait que le nombre de signature n'est pas aussi grand que le nombre de processus utile à l'apprentissage pour le cas de la détection comportementale.

En investissant plus loin dans l'amélioration de la détection, nous avons expérimenté l'IF-IDF qui propose une nouvelle représentation pour le vecteur processus, qui prend en compte l'importance de chaque appel système dans le processus lui-même ainsi que dans les autres processus. Le IF-IDF propose une mesure de cette importance combinant la fréquence d'occurrence de l'appel système considéré dans le processus en question, avec son influence discriminable vis-à-vis les autres processus.

Les résultats confirment l'hypothèse, puisque avec la technique IF-IDF, le taux de détection atteint le chiffre maximal avec un faux positif réduit à environ 5%. Ce que nous pouvons en déduire, c'est que la représentation des données avec le bon choix des paramètres du kNN influence nettement sur la performance de cette méthode pour capturer le comportement normal dans un ensemble de processus d'apprentissage et pouvoir détecter le comportement anormal produit par un quiconque processus.

Chapitre 4 :

Les réseaux bayésiens

1. Introduction :

Les réseaux bayésiens constituent aujourd'hui l'un des formalismes les plus complets et les plus cohérents pour l'acquisition, la représentation et l'utilisation des connaissances par les ordinateurs [BOUDJELIDA, 08].

Les réseaux bayésiens, qui doivent leur nom aux travaux de Thomas Bayes au seizième siècle sur la théorie des probabilités, sont le résultat de recherches qui visaient à intégrer la notion d'incertitudes dans les systèmes experts. Ils offrent ainsi un outil naturel pour résoudre deux problèmes qui apparaissent tout le temps dans les domaines des mathématiques appliquées et de l'ingénierie, l'incertitude et la complexité, et jouent un rôle de plus en plus important dans la conception et l'analyse d'algorithmes de l'apprentissage automatique [Jordan, 98].

Dans le contexte de la détection d'intrusions, domaine de notre travail, la méthode d'inférence utilisée est le réseaux bayésien naïf qui représente une approche de classification très efficace. Cette méthode sera présentée dans ce chapitre structuré comme suite : Dans un premier temps nous allons donner quelques notions sur le calcul probabiliste. nous présenterons les réseaux bayésiens avec ses variantes ainsi que les différentes notions relatives à ces concepts. Nous nous focaliserons sur la notion de la classification en particulier les classificateurs bayésiens. Ensuite, nous montrerons respectivement les mesures et les procédures de l'apprentissage bayésien, avec leurs embauches dans le contexte de la détection d'intrusions.

2. La classification bayésienne :

Au cours des dernières années, [Valdes ,00] a proposé une nouvelle approche hybride pour la détection d'intrusions en se basant sur les réseaux bayésiens [Pearl, 88] [Jensen, 96] qui sont des outils de raisonnement avec des

informations incertaines dans le cadre de la théorie des probabilités. Les réseaux bayésiens utilisent des graphes acycliques dirigés pour la représentation des relations causales et des probabilités conditionnelles (de chaque nœud dans le contexte de ses parents) pour exprimer l'incertitude sur ces relations. Valdes utilise une forme simplifiée des réseaux bayésiens, appelée réseaux bayésiens naïfs, composée de deux niveaux: une racine qui représente la nature de la session (normal et les différents types d'attaques), et plusieurs nœuds enfants, chacun d'entre eux correspond à un attribut de la connexion.

Les réseaux bayésiens naïfs ont plusieurs avantages dus, en particulier, à leur construction qui est très simple. Par ailleurs, l'inférence (classification) est assurée de façon linéaire. En plus, la construction des réseaux bayésiens naïfs est incrémentale, dans le sens qu'elle peut être facilement mise à jour (notamment, il est toujours possible de prendre en considération de nouvelles classes). Cependant, les réseaux bayésiens naïfs travaillent sous une hypothèse d'indépendance très forte entre les attributs dans le contexte de la nature de la session.

3. Calcul Probabiliste

La base du calcul probabiliste est les variables aléatoires (discrètes ou continues). Une variable aléatoire est une variable qui peut prendre un ensemble de valeurs (son domaine) selon des probabilités prédéfinies (la distribution des probabilités conditionnelles).

Dans ce qui va suivre nous allons considérer :

- *P (A) :* La probabilité d'observer un événement A
- *P (A, B) :* La probabilité jointe d'observer deux événements A et B ensemble
- *P (A|B) :* La probabilité conditionnelle d'observer l'événement A, sachant la valeur de l'événement B

- Deux variables sont indépendantes si et seulement si P (A|B) = P (A). P (B).Ceci exprime le fait que l'observation de A ne dépend pas de B : quel que soit la valeur de B, nous avons toujours les mêmes probabilités d'observer A. L'indépendance est une propriété symétrique : Si A est indépendant de B, alors, automatiquement, B est indépendant de A.
- La base de tout calcul probabiliste est le fameux théorème de Bayes qui exprime une probabilité jointe comme le produit d'une probabilité conditionnelle et une autre probabilité.

$$P\ (A,B)\ =\ P\ (A|B).P\ (B)\ =\ P\ (B|A).P\ (A)$$

- Dans le cas où les deux variables aléatoires sont indépendantes le théorème de Bayes devient :

$$P\ (A,B)\ =\ P\ (A).P\ (B)$$

- La règle de chaîne (Chain Rule) de Bayes permet d'appliquer le théorème de Bayes récursivement dans le cas où l'on a plus que 2 variables. Par exemple :

  $P\ (A,B,C)\ =\ P\ (C|\ A,B).P\ (B|A).P\ (A)$

Un réseau bayésien est définie par :

- Un graphe oriente sans circuit (D, A, G) G= (V, E), où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G.
- Un espace probabilisé fini (Ω, Z, p);
- Un ensemble de variables aléatoires associées aux nœuds du graphe et définies sur tel que :

$$P(V_1,V_2,V_3,\dots,V_n) = \sum_{i=0}^{n} P\ (V_i, C(V_i)\ )$$

Où est l'ensemble des causes (parents) de dans le graphe G.

Un réseau bayésien est donc un graphe causal auquel on a associé une représentation probabiliste sous-jacente. Cette représentation permet de rendre quantitatifs les raisonnements sur les causalités que l'on peut faire à l'intérieur du graphe [Naim, 07].

L'utilisation essentielle des réseaux bayésiens est de calculer des probabilités conditionnelles d'événements reliés les uns aux autres par des relations de cause à effet. Cette utilisation s'appelle inférence qu'on va le détailler dans le paragraphe suivante.

Une difficulté essentielle des réseaux bayésiens se situe précisément dans l'opération de transposition du graphe causal à une représentation probabiliste.

4. Les réseaux bayésiens naïfs :

Les réseaux bayésiens sont des outils de représentation de connaissances en présence d'incertitude. Le succès de ces modèles est fortement lié à leur capacité de représenter et de manipuler des relations de (in)dépendance qui sont importantes pour une gestion efficace des informations incertaines.

Les réseaux bayésiens utilisent une représentation basée sur le conditionnement, où les connaissances sont structurées sous la forme d'un graphe acyclique orienté. Les nœuds représentent des variables et les arcs qui codent le lien causal (ou l'influence) entre ces variables. L'incertitude est représentée au niveau de chaque nœud en explicitant toutes les probabilités conditionnelles attachées aux valeurs associées à ce nœud sachant celles de ses parents. Cette incertitude exprime la force de la relation de causalité entre les variables.

Une simple variante des réseaux bayésiens est appelée réseaux bayésiens naïfs. Ces réseaux ont une structure unique qui se compose de deux niveaux seulement. Le premier contient un seul nœud parent qui n'est pas observé et le second plusieurs enfants de ce nœud correspondant aux nœuds observés. Les

réseaux bayésiens naïfs travaillent sous la forte hypothèse d'indépendance entre les nœuds enfants dans le contexte de leur parent. L'utilisation des réseaux bayésiens naïfs est assurée en considérant le nœud parent comme un nœud caché précisant à quelle classe appartient chaque objet de la base de données et les nœuds enfants représentent les différents attributs spécifiant cet objet.

En présence d'un ensemble d'apprentissage on doit juste calculer les probabilités conditionnelles puisque la structure du graphe est unique. Ce calcul peut être résumé comme suit:

- Les probabilités conditionnelles pour les attributs discrets sont calculées à partir des fréquences en comptant combien de fois chaque valeur d'attribut apparaît avec chaque valeur possible du nœud parent.

$$P(c_i \,/\, A) = \frac{\mathrm{f}\,(\mathrm{A}/c_i\,)}{\mathrm{f}(\,c_i)} \qquad (4.1)$$

Une fois le réseau quantifié, il peut être utilisé pour classer de nouveaux objets étant donné leurs valeurs d'attributs en utilisant la règle de Bayes exprimée par:

$$P\,(c_i/A) = \frac{P\,(A_1\,/c_i)\ *\ P\,(c_i)}{P(A)} \qquad (4.2)$$

Où ci est une valeur possible de la classe C et A est l'évidence totale sur les attributs. L'évidence A peut être vue comme un vecteur d'instances a_1, a_2, …, a_n relatives aux attributs a_1, a_2, …, a_n respectivement. Puisque les réseaux bayésiens naïfs travaillent sous l'hypothèse que ces attributs sont indépendants (sachant le nœud parent C), leur probabilité jointe peut être calculée comme suit:

$$P\,(c_i/A) = \frac{P\,(a_1\,/c_i)\ P\,(a_2/\,c_i)\ \ldots.P\,(a_n/c_i) *\ P\,(c_i)}{P(A)} \qquad (4.3)$$

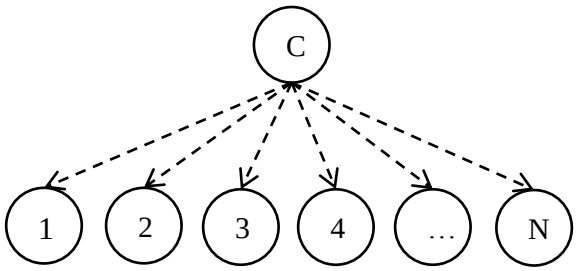


Fig. 4. 1. Schéma de la dépendance des attribues

- **La probabilité nulle :**

Le calcul des probabilités conditionnelles et a priori basé sur les fréquences, ce qui peut s'avérer entaché d'erreur si la valeur d'un attribut n'apparaît pas avec toutes les classes dans l'ensemble d'apprentissage. En effet, ceci peut entraîner des probabilités conditionnelles nulles qui vont réduire à zéro les probabilités de certaines classes.

La technique standard pour éviter ce problème s'appelle estimateur de Laplace [Mahjoub & Khlifia, 08] et elle consiste à ajouter 1 à tous les numérateurs et de compenser ces ajouts dans les dominateurs.

- **Inputs :**
  - les appels systèmes
  - L'ensemble d'apprentissage.
- **Outputs :**
  - déterminer la classe du processus (normal / anormal)

Algorithme :

Apprentissage :

- Déterminer les fréquences d'apparition de chaque classe à partir de l'ensemble d'apprentissage. Ces fréquences sont calculées avec la formule suivante:

$$P(c_i) = \frac{\mathrm{f}\,(c_i\,)}{\mathrm{N}} \qquad (4.4)$$

- Sachant que : f (c_i) est le nombre de fréquence de la classe. N est le nombre total du processus dans les données d'apprentissage.

- Déterminer les fréquences d'apparition de chaque terme sachant la classe en utilisant la formule 4.1. Sachant que : f (A/c_i) est le nombre d'appel système appartenant à la classe normal ou anormal.

- **Inférence :**

- Déterminer la classe du processus I= (selon la formule: 4.3)

Afin de classer n'importe quel nouveau processus caractérisé par ses valeurs d'attributs (appels systèmes): Les classes choisies seront celles dont la probabilité est la plus grande.

Algorithme 4.1. Réseau bayésienne naïve

- **Les ensembles de données**

L'ensemble des processus d'apprentissage est constitué d'environ 1700 processus normaux et un nombre qui atteint les 400 processus anormaux c'est mêmes processus sont extrait de l'ensemble des processus utilisé pour l'identification des appels systèmes influents dans les processus.

Les ensemble de test normal ou attaque sont les mêmes que ceux utilisés dans la méthode KNN en chapitre

- **Expérimentations :**

	détection	**faux positive**
Taux	96.0784313725	12.213740458

Tableau. 4. 1. Résultats du réseau bayésien nave

Nous remarquons que le taux de détection est élevé, il a atteint 96.07% avec un faux positif de 12.21% un résultat assez intéressant. Malgré que la méthode de bayes naïve émis une contrainte forte sur l'ensemble des données, celui de l'indépendance des appels systèmes l'un de l'autre.

Même si nous constatons que le taux de détection est très satisfaisant, le faux positif a besoin d'être réduit, pour augmenter la performance du système. Pour cela, nous avons procédé à assouplir la contrainte imposée par la contrainte du bayes naïve. (Code python annexe A.4)

5. Réseaux bayésiens naïfs augmentés par un arbre : Tree Augmented Naive Bayes (TAN) :

L'hypothèse d'indépendance entre les attributs utilisés dans le classifier de Bayes naïf est généralement fausse (hypothèse naïve). Il existe différentes techniques pour assouplir cette hypothèse. Elles consistent à identifier les dépendances conditionnelles entre les attributs. Nous obtenons alors une sous-structure optimale sur les observations en adaptant la méthode de recherche de l'arbre de recouvrement de poids maximal (Maximal Weight Spanning Tree ou MWST)

Cette méthode s'applique à la recherche de structure d'un réseau bayésien en fixant un poids à chaque arête potentielle de l'arbre. Ce poids peut être par exemple l'information mutuelle entre les variables. Une fois cette matrice de poids définie, il suffit d'utiliser un des algorithmes standards de résolution du problème de l'arbre de poids maximal comme l'algorithme de Kruskal ou celui de Prim. L'arbre non dirigé retourné par cet algorithme doit ensuite être dirigé en choisissant une racine puis en parcourant l'arbre par une recherche en profondeur. La racine peut être choisie soit aléatoirement, soit à l'aide de connaissance a priori [Li, 03].

Un réseau bayésien possède deux niveaux de paramètres : des paramètres quantitatifs qui sont les probabilités conditionnelles associées à chaque nœud, $p(X_{ij}$, p $(X_i))$, et des paramètres qualitatifs qui sont les arcs entre les différents nœuds. L'ensemble de ces arcs forme la structure du réseau.

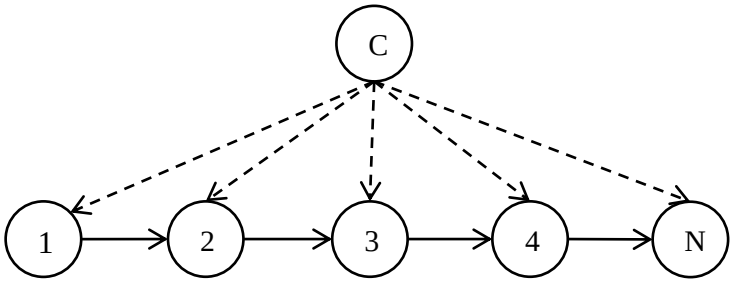


Fig. 4. 2. Réseau bayésien augmenté par un arbre

5.1. Algorithme TAN :

La construction de l'arbre bayésien naïf comporte deux phases :

- L'apprentissage de la structure de l'arbre représentant le problème traité.
- La quantification de l'arbre par les valeurs modélisant le problème.

5.1.1 L'Apprentissage de la structure du réseau.

- **Inputs :**
  - ✓ L'ensemble d'apprentissage.
  - ✓ RB Naïf
- **Outputs :**
  - ✓ - RB Naïf augmenté par un arbre
- **Algorithme :**
  1. Calculer l'information mutuelle MI (O_i, O_j/C) pour chaque couple d'attributs avec $i \neq$ j

$$\mathrm{MI}\left(O_i, O_j\right) = \sum_{\substack{O_i \in x \\ O_j \in y \\ z \in C}} P(x, y, z) \log \frac{P(x, y/z)}{P(x/z)P(y/z)}$$

2- Construire un graphe complet et non orienté entre les attributs ;

3- Transformer le graphe en un arbre dont la somme des arcs est maximum en utilisant l'algorithme MWST. (Code python annexe A.5)

4- Choisir un nœud comme racine de l'arbre et le transformer en un arbre orienté

5 - Ajouter le nœud C et un arc de C vers chacun des nœuds Oi.

Algorithme 4.2. Structure réseau bayésienne naïve augmenté par un arbre

5.1.2 La quantification du réseau

Comme pour le réseau naïf, l'apprentissage des paramètres dans le cas du TAN est basé sur un calcul de fréquences comme illustré dans l'algorithme

- **Inputs :**
  - ✓ L'ensemble d'apprentissage.
  - ✓ Structure des réseaux bayésiens naïf augmenté par un arbre.
- **Outputs :**
  - ✓ Les paramètres des réseaux bayésiens naïf augmenté par un arbre.
- **Algorithme :**

- Déterminer les fréquences d'apparition de chaque classe à partir de l'ensemble d'apprentissage. Ces fréquences sont calculées avec la formule suivant:

$$P(c_i) = \frac{\mathrm{f}\,(c_i\,)}{\mathrm{N}} \qquad (4.5)$$

- Sachant que : f (c_i) est le nombre de fréquence de la classe. N est le nombre total du processus dans les données d'apprentissage.

- Déterminer les fréquences d'apparition de chaque terme sachant la classe en utilisant la formule 4.5. Sachant que : f (A/c_i) est le nombre d'appel système appartenant à la classe normal ou anormal.

- Déterminer les fréquences d'apparition de chaque terme sachant les parents en utilisant la formule suivante :

$$\begin{cases} P(\,x_i, C_i, P_i) = \dfrac{\mathrm{f}\,(C_i, P_i, x_i)}{\mathrm{f}(\,C_i, P_i)} & si\ P_i\ existe \\ P(\,x_i, C_i, P_i) = \dfrac{\mathrm{f}\,(C_i, x_i)}{\mathrm{f}(\,C_i)} & si\ P_i\ n'existe\ pas \end{cases}$$

Sachant que :

- est le nombre d'images qui contient le terme et qui a comme parents.
- est le nombre d'images de la classe qui contient le terme

- **Inférence :**

Une fois le réseau quantifié, il peut être utilisé pour classer de nouveaux images étant données leurs valeurs d'attributs en utilisant la règle suivante:

$$P(C|I) = P(C) \prod_{j} P(I_j | I_{jp}, C)$$

Sachant que I_{jp} est le parent de Ij et :

$$\begin{cases} P(I_j | I_{jp}, C) = P(I_j | I_{jp}, C) \; si \; I_{jp} \neq \emptyset \\ P(I_j | I_{jp}, C) = P(I_j | C) \quad si \; I_{jp} = \emptyset \end{cases}$$

Algorithme 4.3. Quantification réseau bayésienne naïve augmenté par un arbre

- **Expérimentations :**

L'idée est d'assouplir l'hypothèse du bayez naïve par une autre plus souple qui permet l'existence de dépendance d'un appel système d'au plus d'un autre appel système. Ces dépendance sont établie par apprentissage sur l'ensemble le même ensemble de processus utilisé pour les la mesure de l'importance des appels systèmes.

Avec l'algorithme de Kruskal, l'arbre définissant les dépendances voulues sont établie, avec l'hypothèse de l'arbre acyclique maximisant l'information mutuelle entre les appels systèmes. Cet arbre est rendu un arbre dirigé avec un choix du nœud racine convenable.

	détection	**faux positive**
Taux	100 %	45.2926208651 %

Tableau. 4. 2. Résultat du réseau bayésien naïve augmenté par l'arbre

Le taux de détection résultant a atteint le maximum 100% seulement c'était au détriment du faux positif qui s'est amplifié sensiblement jusqu'à 45.29%. C'est une situation dont on devrait y remédier.

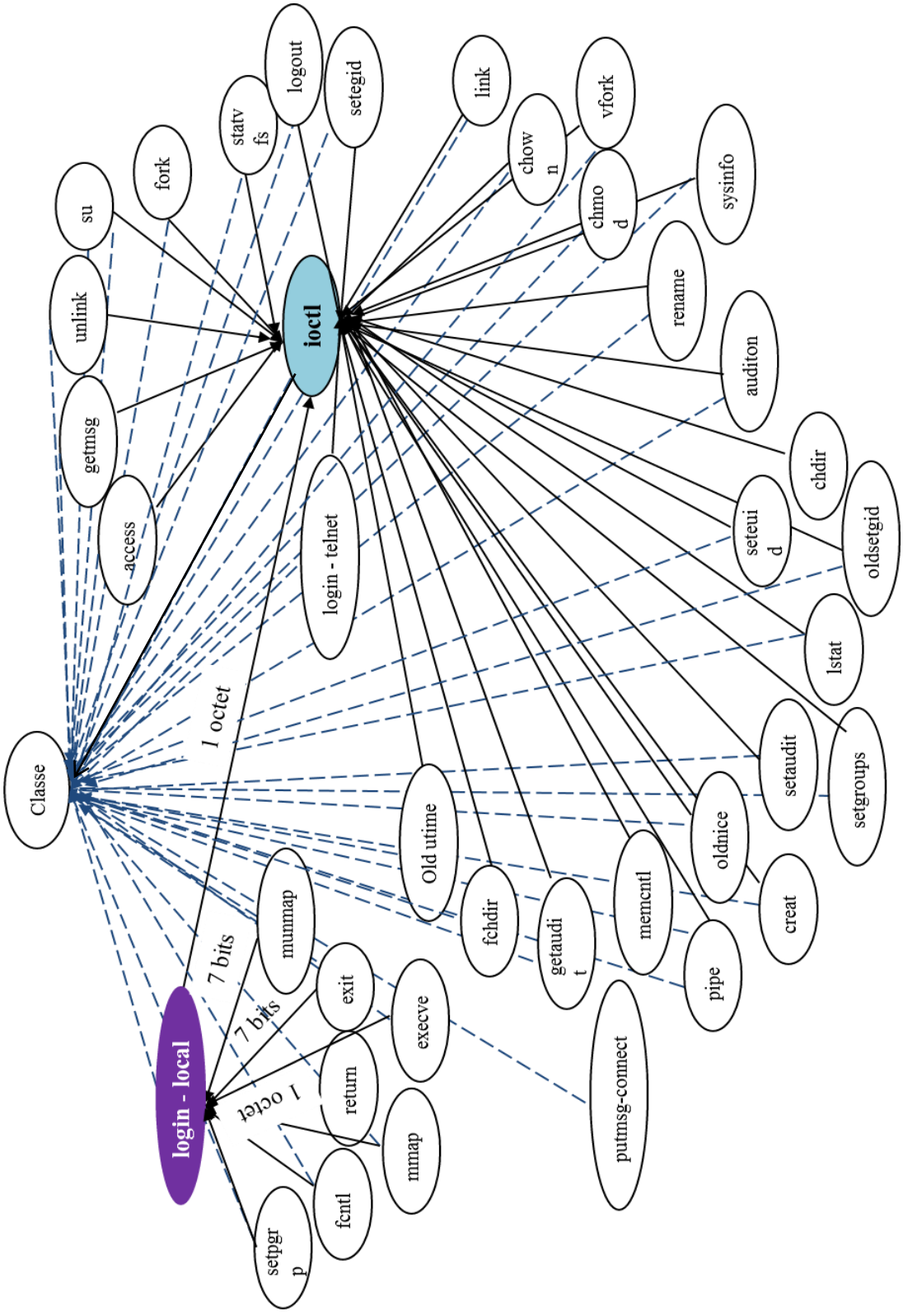


Fig. 4. 3. La structure TAN

6. Réseaux bayésiens naïfs augmentés par une forêt

D'après nos expérimentations, on observe que le réseau bayésien naïf a donné un bon résultat que TAN. Deux facteurs peuvent être à l'origine :

- Les directions de liens dans un TAN sont cruciales. Dans l'étape 5 du l'algorithme TAN, un attribut est choisi aléatoirement comme racine de l'arbre et les directions de tous les liens sont mises après. On remarque que la sélection de l'attribut racine détermine en réalité la structure du TAN résultant, puisqu'un TAN est un graphique dirigé. Ainsi la sélection de l'attribut racine est importante pour construire un TAN.

Des liens non nécessaires peuvent exister dans un TAN. Dans l'étape 4 du TAN, un arbre de recouvrement de poids maximal est construit. Ainsi, le nombre des liens est fixé à n-1. Parfois, il pourrait être sur adapter avec les données, puisque quelques liens ne peuvent pas être nécessaires d'exister dans le TAN.

En se basant sur les observations précédentes, nous modifions l'algorithme TAN également comme suit :

1- Nous choisissons l'attribut A_{racine} qui a l'information mutuelle maximale avec la classe, définie par l'équation ci-dessous, comme racine.

$$A_{racine} = argmax_{AI} IM_i \ (A_i \,|C)$$

Où i = 1 … n. Il est naturel d'utiliser cette stratégie, c'est à dire l'attribut qui a la plus grande influence sur la classification devrait être la racine de l'arbre.

2- Nous filtrons les liens qui ont des informations mutuelles conditionnelles moins qu'un seuil. À notre compréhension, ces liens ont un risque élevé pour sur adapter les données et ensuite l'évaluation de probabilité. Plus précisément, nous utilisons l'information mutuelle conditionnelle moyenne définie dans l'équation ci-dessous comme un seuil. Tous les liens avec les informations mutuelles conditionnelles moins que sont enlevés.

$$I_{avg} = \frac{\sum_i \sum_{j,j \neq i} (A_i, A_j | C)}{n(n-1)}$$

Où n est le nombre d'attributs

Puisque la structure du modèle résultant n'est pas un arbre strict, le nom de l'algorithme sera l'algorithme naïve bayes augmenté par une forêt FAN (Forest Augmented Naive bayes).

6.1. Algorithme d'apprentissage de la structure

- **Inputs :**
  - ✓ L'ensemble d'apprentissage.
  - ✓ Structure des réseaux bayésiens naïf augmenté par un arbre.
- **Outputs :**
  - ✓ Les paramètres des réseaux bayésiens naïf augmenté par une forêt.
- **Algorithme :**

1-Calculer l'information mutuelle MI (O_i, O_j/C) pour chaque couple d'attributs avec $i \neq$ j

$$\text{MI}\left(O_i, O_j\right) = \sum_{\substack{O_i \in x \\ O_j \in y \\ z \in C}} P(x, y, z) \log \frac{P(x, y/z)}{P(x/z)P(y/z)}$$

Et calculer l'information mutuelle conditionnelle moyenne I_{avg} définie dans l'équation ci-dessous :

$$I_{avg} = \frac{\sum_i \sum_{j,j \neq i} (A_i, A_j | C)}{n(n-1)}$$

2- Construire un graphe complet et non orienté entre les attributs ;

3- Transformer le graphe en un arbre dont la somme des arcs est maximum en utilisant l'algorithme MWST.

4- Calculer l'information mutuelle entre chaque attribut et la classe MI (O_i, O_j |C), i = 1 … n, et choisir l'attribut A_{racine} qui a la plus grande information mutuel avec la classe.

$$A_{racine} = argmax_{AI} IM_i \ (A_i \,|C)$$

5 - Transformer l'arbre non orienté résultante en un arbre orienté en mettant A_{racine} comme racine.

6- Supprimer les liens dirigés qui ont le poids des informations mutuelles conditionnelles au-dessous des informations mutuelles conditionnelles I_{avg}.

7 - Ajouter le nœud C et un arc de C vers chacun des nœuds O_i.

Algorithme 4.4. Structure réseau bayésienne naïve augmenté par un Foret

En outre, on rappelle que l'algorithme d'apprentissage de paramètres est une extension de celui du TAN.

- **Expérimentation :**

Regrouper toutes les dépendances en un seul arbre a défavorisé le paramètre de faux positif, pour surmonter cette limitation des TAN, les forets ont été proposés.

La forêt bayésienne est générée par apprentissage sur le même ensemble de processus considéré pour le TAN.

L'expérimentation est faite sur les mêmes ensembles test que le TAN. Notre attention était attirée par l'affaiblissement si léger du taux de détection au profit du faux positif qui s'est réduit vers une valeur de 6.01%. Le résultat est un compromis optimal pour l'IDS comportementale.

	détection	**faux positive**
Taux	93.893129771 %	6.10687022901 %

Tableau. 4. 3. Résultats du réseau bayésien naïve augmenté par Foret

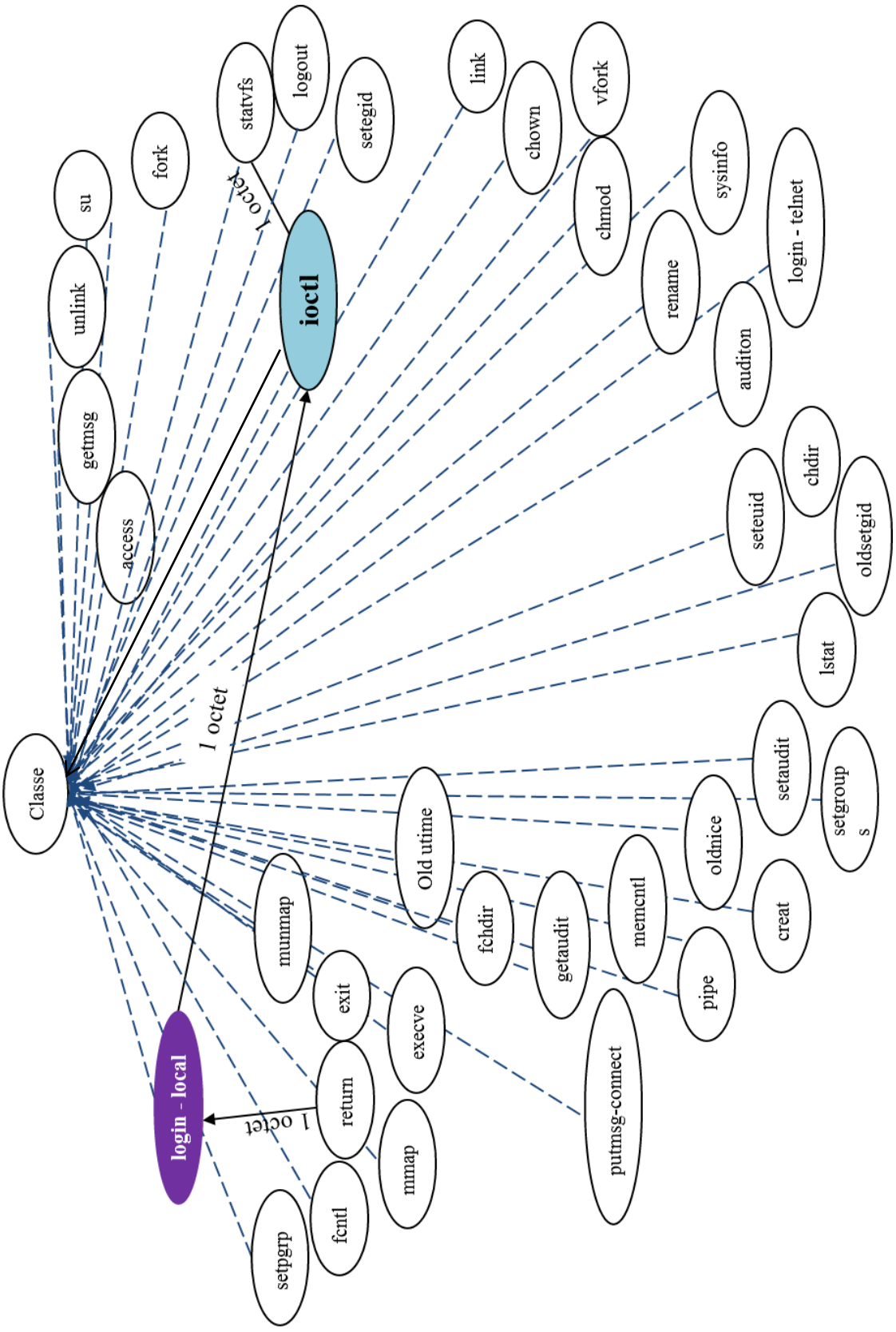


Fig. 4. 4. La structure FAN

7. Conclusion :

L'idée d'introduire la notion de probabilité pour cerner une grande quantité d'information et de données, dans notre cas les processus, à bien réussit en multiple domaine, surtout que l'information elle-même est définie autour d'un concept central qui est la probabilité.

La théorie bayésienne exprime la causalité entre les données, toute une méthode était conçue suivant cette vision. La méthode bayes naïve classifie un processus par le principe de causalité probabiliste. Elle s'est avéré une méthode efficace aussi bien pour le taux de détection élevé qu'elle produit, ainsi que pour le faux positif assez faible qu'elle engendre. Dans l'espoir de diminuer ce faux positif, nous avons introduit l'arbre bayésien augmenté (TAN), seulement, même si le taux de détection était maximal, le faux positif est devenu trop important, mettant en défaut la performance de l'IDS. Les forêts sont utilisés pour corriger cette situation, les résultats étaient plus que satisfaisants, étant donné qu'un compromis optimal entre le taux de détection et le faux positif était enregistré.

Conclusion Générale

L'information et la sécurité sont deux concepts centraux, fatals et fondamentaux dans la vie d'une nation et de toute organisation. Procurer l'information s'est se doter d'une puissance, la sécuriser s'est la garantir.

Le partage et l'inter-échange de l'information entre les individus dans un même organisme devrait être assuré et protégé. L'Internet se vaut être l'éther véhiculant l'information. L'utiliser est un choix élégant, raisonnable et soutenu. Si la communication de l'information est assurée par les protocoles incorporant l'Internet, la sécurité cherche à renforcer son rôle de garantir de cette communication.

Des politiques et des outils sont proposés continuellement pour fournir des mécanismes de défense de plus en plus efficaces. Les solutions de défense statiques, qui sont analogues aux barrières autour des propriétés, peuvent fournir un niveau de sécurité raisonnable. Ils sont prévus pour empêcher les attaques de se produire. Parmi les exemples de ces solutions; le maintien à jour des logiciels tels les systèmes d'exploitation , et le déploiement des pare-feu aux points d'accès.

Les failles de sécurité sont dévoilées passionnément par les hackers. Cette passion persistante est défiée par une surveillance persévérante du réseau et des systèmes connectés. Les traces de toutes les activités sont gardées pour une étude postérieure ou immédiate. Des fichiers logs générés par les audits tel le BSM (Basic Security Module) de Sun Solaris qui s'occupe d'enregistrer tout évènement survenant au sein du système de l'hôte.

Les systèmes de détection d'intrusion (IDS : Intrusion Detection System) ont pour mission d'analyser ses fichier log et déceler les traces d'intrus et les

discerner de celles des processus légaux. Les IDS suivent deux approches dans leur analyse, ceux qui identifient la trace d'une attaque par sa signature, et ceux qui reconnaissent le comportement des intrus.

L'approche par signature est très efficace dans le cas où l'emprunte de l'attaque est déjà répertoriée dans la base de données de l'IDS, seulement dès qu'une nouvelle attaque avec une signature largement différente arrive, le système est complètement compromit. Pour remédier à cette situation, l'approche comportementale tente à assimiler et reconnaitre le comportement des intrus afin de les distingué parmi les activités légales.

L'idée de la classification des comportements est introduite naturellement, cependant il faut noter qu'un IDS devrait faire face au fait qu'il est amené à analyser une quantité extensive de données ce qui peut remettre en cause la vitesse de traitement, et réduire la performance du système. La performance d'un IDS, est mesurée avec le taux de détection et celui des faux positifs, qui sont utilisés pour récapituler les différentes caractéristiques de l'exactitude de la détection. Un IDS est plus précis s'il détecte plus d'attaques et donne peu de fausses alarmes.

L'adaptation est un souci majeur, l'IDS devrait apprendre à traiter les nouveaux évènements. Les méthodes d'apprentissage automatique fournissent une solution potentielle pour les problèmes d'adaptation et d'exactitude dans le domaine de la détection d'intrusions. Dans ce domaine, l'apprentissage signifie l'extraction des modèles du comportement normal ou d'attaque. Selon la méthode d'apprentissage utilisée les algorithmes d'apprentissage sont supervisés ou non. Dans l'apprentissage supervisé, l'algorithme d'apprentissage est appliqué sur un ensemble d'instances classifiées (ou marquées) et utilisé pour traiter les nouvelles instances non classifiées. D'un autre côté, l'apprentissage non supervisé implique la

recherche des associations entre les attributs sans se servir des classes ou des labels.

La classification est une tâche de l'apprentissage automatique qui a une phase d'entraînement où il faut apprendre mathématiquement les modèles à partir de l'ensemble de données introduit. Cet ensemble de données s'appelle également l'ensemble d'entraînement, qui devrait contenir des instances suffisantes et représentatives des modèles recherchés sachant qu'une instance se compose d'un ensemble d'attributs. Les modèles appris sont utilisés pour faire des prédictions à partir d'un nouvel ensemble d'instances de données.

Dans notre travail de recherche, nous avons essayé d'explorer une variante des méthodes de classification. L'objectif principal était d'étudier les systèmes de détection d'intrusion HIDS basés sur la classification comportementale des processus.

Nous nous sommes intéressé à la méthode des plus proches voisins knn, où les expérimentations ont confirmé sont succès déjà prouvé dans ce domaine. Au début, nous avons essayé d'expérimenter la détection par signature utilisant cette méthode, où nous avons mise en évidence le besoin d'IDS basés comportement.

Des améliorations étaient à réaliser pour augmenter l'exactitude de la méthode knn vis-à-vis le taux de faux positifs, pour ce faire nous avons essayé de raffiner la représentation des processus avec la technique IF-IDF, pour mieux discerner les processus normaux de ceux anormaux, de meilleurs résultats sont obtenus.

Une autre méthode utilisant une approche différente pour la classification est investie. La méthode du bayes naïve, qui exploite la causalité pour réaliser l'inférence à la classe adéquate. Les probabilités sont mises en pratique, dans le but de couvrir la quantité énorme de données à étudier.

Des résultats satisfaisants sont produits, néant-moins le paramètre du faux positif est assez élevé, dans l'espoir de le diminuer, nous avons recouru au service des TAN les arbres bayésiennes, qui ont eu un impact sensible sur le taux de détection qui a atteint le maximum, bien que le faux positifs a aussi subit une augmentation indésirable, mettant ainsi en défaut la performance de l'IDS. Les FAN étaient une alternative prometteuse, ils ont fournis un compromis optimal entre le taux de détection et le faux positif.

Certes, les deux approches de classement sont intéressantes pour le développement des HIDS comportementales, cependant certaines différences les distingues l'une par rapport à l'autre.

La décision de classification est plus rapide pour le bayes naïf et ses variantes, étant données qu'elle effectue une phase d'apprentissage et génère un modèle qu'elle utilise par la suite pour faire la classification. Cependant, le KNN consulte tout l'ensemble d'apprentissage à chaque nouveau processus à classifier.

Le KNN se montre robuste aux légères changements de l'ensemble des processus d'apprentissage, à l'égard de l'approche bayésienne qui est sensible au point que la structure, surtout des TANs et FANs, peut être affectée par des modifications essentielles.

L'exactitude des deux approches est très satisfaisante surtout en ce qui concerne le taux de détection, même si nous constatons une performance visible du KNN qui a atteint un taux de détection maximal avec le plus petit faux positifs.

Les données de détection d'intrusions DARPA 98 restent toujours le meilleur corpus de données libellées, qui permet aux concepteurs des IDS de construire, d'évaluer et de comparer les performances de leurs systèmes avec d'autres concepteurs qui ont utilisé le même ensemble de données. Depuis 1999, beaucoup de nouvelles technologies et de nouveaux protocoles ont été développés et beaucoup de nouveaux types d'attaques ont été détectés,

d’où la nécessité d'enrichir cet ensemble de données. D'autre part, les concepteurs des IDS doivent démontrer que leurs systèmes fonctionneront pour des ensembles de données autres que les données de DARPA 98 . Malheureusement, très peu de détails ont été publiés au sujet des techniques et des procédures sur les environnements de détection d'intrusions DARPA, la chose qui rend très difficile le fait de suivre les même procédures pour refaire la collecte des données.

Références :

[Admin, 12] : Administration d'Oracle® Solaris : services de sécurité 2012

[Amor & Benferhat & Elouedi, 06] N. Ben Amor I S. Benferhat2 Z. Elouedi ,Réseaux bayésiens naïfs et arbres de décision dans les systèmes de détection d'intrusions 2006

[Baud & Mar, 04]Nicolas Baudoin ,Marion Karle : NT Réseaux IDS et IPS 2003/2004

[BOUDJELIDA, 08] BOUDJELIDA Abdelhamid , Réseaux Bayésiens Naïfs Augmentés TAN pour les Systèmes de Détection d'Intrusions 2008

[Cisco, 03] Cisco: the Science of Intrusion Detection System Attack Identification, January 2003

[Garfinkel & al, 96] S. Garfinkel, & G. Spafford, "Practical Unix & Internet Secu rity", O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol CA, 95472, 2nd edition, April 1996

[Jensen, 96] Jensen, F. V.: Introduction to Bayesien networks. UCL Press, 1996.

[Jordan, 98] M. I. Jordan, "Learning in Graphical Models", the Netherlands: Kluwer Academic Publishers, 1998.

[Kendall, 99] K. Kendall, "A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems," M . Eng. Thesis, MIT Department of Electrical Engineering and Computer Science, June 1999.

[Lawrence, 99] Lawrence Berkeley National Laboratory, "Netwo rk Research Group Homepage", http://www.nrg.ee.lbl.gov/, May 1999

[LERM, 06] Liran LERMAN : Les systèmes de détection d'intrusion basés sur du machine learning

[Li, 03] Li X : Augmented naive bayesian classifiers for mixed-mode data, December, 2003.

[Liao & Vemuri, 02] Y. Liao and V.R. Vemuri. Using k-nearest neighbor classifier for intrusion detection. Computers Security, 2002

[Lippmann & al, 00] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. Computer Networks, October 2000.

[Lough, 01] Daniel Lowry Lough, "A Taxonomy of Computer Attacks with Applications to Wireless Networks", *PhD thesis*, Virginia Polytechnic Institute and State University, 2001.

[Mahjoub & Khlifia, 08]Mohamed Ali Mahjoub et Khlifia Jayech :Indexation de structures de documents par réseaux bayésiens 2008

[Mart & Mark, 03]Martin Arvidson Markus Carlbark :Intrusion Detection Systems – Technologies, Weaknesses and Trends 2003

[MIT, 99] Lincoln Laboratory DARPA 99 Intrusion Detection Data Set Attack Documentation, http://www.ll.mit.edu/IST/ideval/docs/1999/attackDB.html.

[MIT, 00] Lincoln Laboratory ID Evaluation Website, MIT, http://www.ll.mit.edu/IST/ideval/index.html/, 2000, contains information on the 1998 and 1999 evaluations

[Naim, 07] Naim P., P.H.Wuillemin, P.Leray, O.Pourret, A.Becker. Réseaux bayésiens, Eyrolles, Paris, 2007

[Nguyen, 10] Nguyen Quang Trung, Intrusion Detection System for Classifying Process Behavior. Master Thesis 2010

[Pearl, 88] Pearl J.: Probabilistic Reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmman , Los Altos, CA, 1988.

[Stef, 99] Stefan Axelsson ,The Base Rate Fallacy and its Implications for the Difficulty of Intrusion Detection, 1999

[Valdes , 00] Valdes, A., Skinner K.: Adaptive Model-based Monitoring for Cyber Attack Detection. In proceedings of Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, 80-92, 2000.

Annexe A: code source python

```
def prob_mat (path,infile,outfile,taille):
    f = open (path+infile,'r')
    liste_nor , liste_atk,prob_nor ,prob_atk ,total= [],[],[],[],[] # declaration
    px_nor,px_atk,total_nor,total_atk , prob_total_nor ,prob_total_atk= 0,0,0,0,0,0
    for i in range (taille):   #initialisation avec o pour tous les liste
        liste_nor.append(0)
        liste_atk.append(0)
        prob_nor.append(0)
        prob_atk.append(0)
        total.append(0)
    for l in f :
        l = l.split()
        for i in range (len(l)-1 ):
            l[i]=int(l[i])
            if l[len(l)-1] == '0':
                if l[i] > 0 :
                    liste_nor [i] += 1
                    total [i] += 1
            if l[len(l)-1] == '1':
                if l[i] > 0 :
                    liste_atk [i] += 1
                    total [i] += 1
    f.close ()
    # calculer la  probabilité d'appel système
    for i in range (len(total)) :
        if total [i] != 0 :
            prob_nor [i] = liste_nor[i] /total [i]
            prob_atk [i] =  liste_atk[i]/total [i]
    for i in range (len(liste_nor)) :
        total_nor += liste_nor[i]
        total_atk += liste_atk [i]
    prob_total_nor = total_nor/(total_nor + total_atk)
    prob_total_atk = total_atk/(total_nor + total_atk)
    h_x = (-1)*prob_total_nor * log (prob_total_nor,2)+ (-1)*prob_total_atk *
log(prob_total_atk,2)
    total_par = 0
    total_par_y =[]
    for i in range (len(total) ) :
        total_par += total[i]
    for i in range (len(total)) :
        total_par_y.append(total[i]/total_par)
    h_xy = []
    for i in range (len(total_par_y)) :
        if prob_nor[i] == 0 and prob_atk[i] == 0 :
            h_xy.append( 0 )
        elif prob_nor[i] == 0 :
            h_xy.append( total_par_y[i] *  (-1) *prob_atk[i]*log (prob_atk[i],2)
)
        elif prob_atk[i] == 0 :
            h_xy.append( total_par_y[i] * (-1)*prob_nor[i]*log (prob_nor[i],2)  )
        else :
            h_xy.append( total_par_y[i] * (    (-1)*prob_nor[i]*log
(prob_nor[i],2)+ (-1)*prob_atk[i]*log (prob_atk[i],2)   )       )

    total_hxy = 0
    for i in range (len(h_xy) ) :
        total_hxy += h_xy [i]

    nor_prob_total, atk_prob_total = [],[]
    total_nor_atk =0
    for i in range (len(liste_nor)) :
        total_nor_atk += liste_nor[i] + liste_atk[i]
    for i in range (len(liste_nor)) :
```

```
        nor_prob_total.append(liste_nor[i]/total_nor_atk)
        atk_prob_total.append(liste_atk [i]/total_nor_atk)
    prob_nor_log,prob_atk_log  =[], []
    for i in range (len(nor_prob_total)) :
        if nor_prob_total[i] == 0 :
            prob_nor_log.append(0)
        else :
            prob_nor_log.append( nor_prob_total[i] * (-1)* log
(nor_prob_total[i],2)  )
        if atk_prob_total [i] == 0 :
            prob_atk_log.append( 0 )
        else :
            prob_atk_log.append( atk_prob_total[i] * (-1)* log
(atk_prob_total[i],2)  )

    liste_item_nor , liste_item_atk= [],[]
    for i in range (len(prob_nor_log)) :
        item_nor = (i,prob_nor_log [i])
        item_atk = (i,prob_atk_log [i])
        liste_item_nor.append (item_nor)
        liste_item_atk.append (item_atk)
    liste_item_nor.sort(key=itemgetter(1),reverse=True)
    liste_item_atk.sort(key=itemgetter(1),reverse=True)
    item_total = ()
    liste_item_total = []
    for i in range (len(prob_nor_log)) :
        item_total = (i, h_x - prob_nor_log[i] - prob_atk_log[i] )
        liste_item_total.append (item_total)
    liste_item_total.sort(key=itemgetter(1),reverse=True)
```

A.1. Code python pour l'extraction de l'information mutuelle

```
def similarity(x1, x2):
    '''calculer la similarité entre deux vecteur '''
    norm1= 0
    norm2= 0
    sim = 0.0
    if len(x1) != len(x2):
        raise (ValueError,"vectors must be same length")
    a = dot(x1,x2)
    for i in range(len (x1)):
        norm1 += x1[i]**2
    for i in range(len (x2)):
        norm2 += x2[i]**2
    if (norm1 != 0 and norm2 !=0):
        sim = a  / sqrt(norm1*norm2)
    return  sim
def calculer (liste,x):
    ```Calculer la similarité entre une liste données et un autre vecteur  '''
    liste_s=[] #liste de similarity
    for i in range (len(liste)):
        liste_s.append(similarity(liste[i][:-1],x))
    return liste_s
def choisisV(k,listesim):
    '''le choix des voisins selon le nombre k données'''
    listesim.sort()#trier la liste de similarité
    return listesim[-k:] # le choix des k les plus grands
def classifier (listevoisin,seuil):
    """Classifier la liste des voisins en classe 1 : normale classe 2 attaque,
selon un seuil défini et après on va calculer le nombre des voisins dans chaque
classe et le vote majoritaire se gagne et la réponse c'est la classe """
    classe1,classe2 ,resultat = 0,0,0
    sur = False
    for i in range (len(listevoisin)):
        if (listevoisin [i]== 1) :
            resultat = 0
            sur = True
            break
        else :
            if (listevoisin [i]>= seuil):
```
```

```
                classe1 += 1
            else :
                classe2 += 1
    if sur == False :
        if (classe1 >= classe2 ) :
            resultat = 0
        else :
            resultat = 1
    return resultat
def knn(liste,x,k,seuil):
    listesim = calculer(liste, x[:-1])
    listevoisin = choisisV(k,listesim)
    cl = classifier (listevoisin,seuil)
    return cl
def knnproj (seuil,k):
test = open (path+'test','r')
nbtotal_nor,nbtotal_atk,nbtotal_faux_negatif,nbtotal_faux_positif,nbtotal_vrai_nega
tif,nbtotal_vrai_positif = 0,0,0,0,0,0
    liste  = []
    f = open (path + 'apprentissage.matr','r')
    for l in f :
        l = l.split()
        for i in range (len(l) ):
            l[i] = int(l[i])
        liste.append(l)
    for l in test :
        l = l.split()
        for i in range (len(l)-1 ):
            l[i]=int(l[i])
        if l[len(l)-1] == '0':
            nbtotal_nor +=1
        elif l [len(l)-1] == '1':
            nbtotal_atk +=1
        s = knn(liste ,l,k,seuil)
        if (s == 0) and (l [len(l)- 1] == '0') :
            nbtotal_vrai_negatif += 1
        elif (s == 0) and (l [len(l)- 1] == '1') :
            nbtotal_faux_negatif += 1
        elif (s == 1) and (l [len(l)-1] == '0') :
            nbtotal_faux_positif += 1
        elif (s == 1) and (l [len(l)-1] == '1') :
            nbtotal_vrai_positif += 1
    nbtotal_vrai_negatif = nbtotal_vrai_negatif * 100 / nbtotal_nor
    nbtotal_faux_negatif = nbtotal_faux_negatif * 100 / nbtotal_atk
    nbtotal_vrai_positif = nbtotal_vrai_positif * 100 / nbtotal_atk
    nbtotal_faux_positif = nbtotal_faux_positif * 100 / nbtotal_nor
```

A.2. Code python l'algorithme KNN

```
def idf (path,infile,outfile,taille):
    """Implémentation du technique de term frequency inverse document frequency '
pour donner des poids aux appels systèmes*** party idf """
    inf = open (path+infile,'r')# fichier source
    out = open (path + outfile, 'w') # fichier idf
    liste_idf = [] # liste idf (nombre de tout le documents / nombre d'occurrence
d'appel système dans chaque document)
    nbprocess, fi = 0 ,0
    liste_if = []
    for i in range (taille):
        liste_idf.append (0)# initialisation de la liste avec 0 et la taille est
est le nombre des appels uniques
    for l in inf :
        nbprocess += 1 # compter le nombre total des processus
        l= l.split()
        for i in range (len(l)-1 ):
            l[i]=int( l[i] )
            fi = fi + l[i]
        liste_if.append (fi)
        fi = 0
        for i in range (taille ):
            if l [i] > 0 : # si l'appel système existe dans ce processus ajouter 1
```

```
                liste_idf [i]= liste_idf [i] + 1
    for i in range (taille):
        if (liste_idf [i] != 0 ) : # si total d'occurrence différent de 0
            liste_idf [i]= log (nbprocess / liste_idf [i] + 1 , 10 )#la formule de
idf
        else :
            liste_idf [i] = 0 # sinon idf  = 0
    for i in range (len(liste_idf)):
        liste_idf[i]=str ( liste_idf[i] ) # transformation en chaine de caractère
    for i in range (len(liste_if)):
        liste_if[i]=str ( liste_if [i] ) # transformation en chaine de caractère
    out.write (' '.join (liste_idf)+ ' \n')    # l'écriture dans le fichier
    out.write (' '.join (liste_if) + ' \n')    # l'écriture dans le fichier
    inf.close()
    out.close()
    return liste_if # retourner la liste if (la liste du nombre d'occurrence de
tout appel système dans un processus
def fi (path,infile,idffile,outfile,taille):
    inf = open (path+ infile,'r') # l'ouverture de fichier d'apprentissage
    out = open (path + outfile, 'w') # l'écriture de fichier de sortir ( avec
transformation if idf )
    idf_fi = open (path + idffile, 'r') # l'ouverture de fichier idf
    liste_idf= [] # transformation en liste
    liste_if = []
    ligne = 0
    for k in idf_fi :
        if ligne == 0 :
            k = k.split (' ')
            for i in range (len(k)-1 ):
                liste_idf.append ( float( k[i] )) # transformation en reel
        if ligne == 1 :
            k = k.split (' ')
            for i in range (len(k)-1 ):
                liste_if.append ( float( k[i] )) # transformation en reel
        ligne+=1
    idf_fi.close()
    ind = 0
    for l in inf :
        l= l.split()
        for i in range (len(l)- 1  ):
            l[i]=int( l[i] ) # lecture de fréquence d'un call system dans un
processus
            if (liste_if [ind] != 0 ):
                l [i ] = l [i ]/ liste_if [ind] * liste_idf [i]
        ind += 1
        for i in range (len(l)):
            l[i]=str ( l[i] ) #transformation en chaine de caractère pour
l'écriture
       out.write (' '.join (l)+ ' \n')
      inf.close()
```

A.3. Code python Technique IF-IDF

```
def byasian_apprentissage (path,file,outfile,taille):
    """la construction du modele de la probabilite"""
    f = open (path+file,'r')   # l'ouverture de fichier d'apprentissage
(pascal.praudit)
    w = open (path+outfile,'w')#l'ecriture de fichier (modèle ) d'apprentissage
    liste0_0,liste1_0,liste0_1,liste1_1,liste2_0,liste2_1,liste3_1,liste3_0=
[],[],[],[],[],[],[],[]#déclaration des liste de probabilité
    liste4_10_0,liste4_10_1,liste10_20_0,liste10_20_1= [],[],[],[]#deux liste pour
chaque enchantions (avec 0 , 1 classe)

liste20_100_0,liste20_100_1,liste100_200_0,liste100_200_1,liste200_0,liste200_1=
[],[],[],[],[],[]
    for i in range (taille):   #initialisation avec o pour tous les liste
        liste0_0.append(0)
        liste0_1.append(0)
        liste1_0.append(0)
        liste1_1.append(0)
        liste2_0.append(0)
```

```
        liste2_1.append(0)
        liste3_0.append(0)
        liste3_1.append(0)
        liste4_10_0.append(0)
        liste4_10_1.append(0)
        liste10_20_0.append(0)
        liste10_20_1.append(0)
        liste20_100_0.append(0)
        liste20_100_1.append(0)
        liste100_200_0.append(0)
        liste100_200_1.append(0)
        liste200_0.append(0)
        liste200_1.append(0)
    nb_0,nb_1 =0,0
    for l in f : # ajouter les probabilités
        l= l.split()
        for i in range (len(l)-1 ):
            l[i]= int (l[i])
        for i in range (len(l)-1 ):
            if l [len(l)-1 ] == '0':
                if   l[i]== 0 :
                    liste0_0[i] +=1
                elif l[i]== 1 :
                    liste1_0[i] +=1
                elif l[i]== 2 :
                    liste2_0[i] +=1
                elif l[i]== 3 :
                    liste3_0[i] +=1
                elif l[i]>=4 and l[i]< 10  :
                    liste4_10_0[i] +=1
                elif l[i]>=10 and l[i]< 20  :
                    liste10_20_0[i] +=1
                elif l[i]>=20 and l[i]< 100:
                    liste20_100_0[i] +=1
                elif l[i]>=100 and l[i]< 200:
                    liste100_200_0[i] +=1
                elif l[i]>=200 :
                    liste200_0[i] +=1
            if l [len(l)-1 ] == '1':
                if   l[i]== 0 :
                    liste0_1[i] +=1
                elif l[i]== 1 :
                    liste1_1[i] +=1
                elif l[i]== 2 :
                    liste2_1[i] +=1
                elif l[i]== 3 :
                    liste3_1[i] +=1
                elif l[i]>=4 and l[i]< 10  :
                    liste4_10_1[i] +=1
                elif l[i]>=10 and l[i]< 20  :
                    liste10_20_1[i] +=1
                elif l[i]>=20 and l[i]< 100:
                    liste20_100_1[i] +=1
                elif l[i]>=100 and l[i]< 200:
                    liste100_200_1[i] +=1
                elif l[i]>=200 :
                    liste200_1[i] +=1
        if l [len(l)-1] == '0' :
           nb_0 += 1
        if l [len(l)-1] == '1':
           nb_1 += 1
    liste_0 = []
    liste_1 = []
    for i in range (taille):
        liste_0.append (0)
        liste_1.append (0)
    for i in range (taille):
        if  liste0_0[i] == 0 or liste0_1[i]==0 or liste1_0 [i] ==0 or liste1_1==0
or  liste2_0[i]==0 or liste2_1[i]==0 or liste3_0[i]==0 or liste3_1[i]==0 or
liste4_10_0[i]==0 or liste4_10_1[i]==0 or liste10_20_0[i]==0 or liste10_20_1[i]==0
```

```
or liste20_100_0[i]==0 or liste20_100_1[i]==0 or liste100_200_0[i]==0 or
liste100_200_1[i]==0 or liste200_0[i]==0 or liste200_1[i]==0:
            liste0_0[i] += 1
            liste1_0[i] += 1
            liste2_0[i] +=1
            liste3_0[i] +=1
            liste4_10_0[i] +=1
            liste10_20_0[i] +=1
            liste20_100_0[i] +=1
            liste100_200_0[i] +=1
            liste200_0[i] +=1
            liste0_1[i] += 1
            liste1_1[i] += 1
            liste2_1[i] +=1
            liste3_1[i] +=1
            liste4_10_1[i] +=1
            liste10_20_1[i] +=1
            liste20_100_1[i] +=1
            liste100_200_1[i] +=1
            liste200_1 [i] +=1
            liste_0 [i] += 9
            liste_1 [i] += 9
    f.close()
    for i in range (taille):
        liste0_0 [i]     = str (liste0_0      [i]/(liste_0 [i] + nb_0 ))
        liste0_1 [i]     = str (liste0_1      [i]/(liste_1 [i] + nb_1 ))
        liste1_0 [i]     = str (liste1_0      [i]/(liste_0 [i] + nb_0 ))
        liste1_1 [i]     = str (liste1_1      [i]/(liste_1 [i] + nb_1 ))
        liste2_0 [i]     = str (liste2_0      [i]/(liste_0 [i] + nb_0 ))
        liste2_1 [i]     = str (liste2_1      [i]/(liste_1 [i] + nb_1 ))
        liste3_0 [i]     = str (liste3_0      [i]/(liste_0 [i] + nb_0 ))
        liste3_1 [i]     = str (liste3_1      [i]/(liste_1 [i] + nb_1 ))
        liste4_10_0 [i]  = str (liste4_10_0   [i]/(liste_0 [i] + nb_0 ))
        liste4_10_1 [i]  = str (liste4_10_1   [i]/(liste_1 [i] + nb_1 ))
        liste10_20_0[i]  = str (liste10_20_0  [i]/(liste_0 [i] + nb_0 ))
        liste10_20_1[i]  = str (liste10_20_1  [i]/(liste_1 [i] + nb_1 ))
        liste20_100_0[i] = str (liste20_100_0 [i]/(liste_0 [i] + nb_0 ))
        liste20_100_1[i] = str (liste20_100_1 [i]/(liste_1 [i] + nb_1 ))
        liste100_200_0[i]= str (liste100_200_0[i]/(liste_0 [i] + nb_0 ))
        liste100_200_1[i]= str (liste100_200_1[i]/(liste_1 [i] + nb_1 ))
        liste200_0[i]    = str (liste200_0    [i]/(liste_0 [i] + nb_0 ))
        liste200_1[i]    = str (liste200_1    [i]/(liste_1 [i] + nb_1 ))

    prob_0 = nb_0 / (nb_0 + nb_1)
    prob_1 = nb_1 / (nb_0 + nb_1)
    w.write (str (prob_0 )+'|'+str (prob_1 )+ '\n')
    w.write ('0_0'+'|'+'|'.join(liste0_0)      +'\n')
    w.write ('0_1'+'|'+'|'.join(liste0_1)      +'\n')
    w.write ('1_0'+'|'+'|'.join(liste1_0)      +'\n')
    w.write ('1_1'+'|'+'|'.join(liste1_1)      +'\n')
    w.write ('2_0'+'|'+'|'.join(liste2_0)      +'\n')
    w.write ('2_1'+'|'+'|'.join(liste2_1)      +'\n')
    w.write ('3_0'+'|'+'|'.join(liste3_0)      +'\n')
    w.write ('3_1'+'|'+'|'.join(liste3_1)      +'\n')
    w.write ('4_10_0'+'|'+'|'.join(liste4_10_0)   +'\n')
    w.write ('4_10_1'+'|'+'|'.join(liste4_10_1)   +'\n')
    w.write ('10_20_0'+'|'+'|'.join(liste10_20_0)  +'\n')
    w.write ('10_20_1'+'|'+'|'.join(liste10_20_1)  +'\n')
    w.write ('20_100_0'+'|'+'|'.join(liste20_100_0) +'\n')
    w.write ('20_100_1'+'|'+'|'.join(liste20_100_1) +'\n')
    w.write ('100_200_0'+'|'+'|'.join(liste100_200_0)+'\n')
    w.write ('100_200_1'+'|'+'|'.join(liste100_200_1)+'\n')
    w.write ('200_0'+'|'+'|'.join(liste200_0)    +'\n')
    w.write ('200_1'+'|'+'|'.join(liste200_1)    +'\n')
    w.close()


def byasian_classifier (path,fileapp , vec,taille) :
    """Ce module fait la classification d'un vecteur """
    f = open (path + fileapp,'r')
```

```
    liste0_0,liste1_0,liste0_1,liste1_1,liste2_0,liste2_1,liste3_1,liste3_0=
[],[],[],[],[],[],[],[]#déclaration des liste des probabilités
    liste4_10_0,liste4_10_1,liste10_20_0,liste10_20_1= [],[],[],[]#deux liste pour
chaque enchantions (avec 0 , 1 classe)

liste20_100_0,liste20_100_1,liste100_200_0,liste100_200_1,liste200_0,liste200_1=
[],[],[],[],[],[]
    des1,des2 = [],[]
    for i in range (taille):# initialisation avec o pour tous les liste
        liste0_0.append(0)
        liste0_1.append(0)
        liste1_0.append(0)
        liste1_1.append(0)
        liste2_0.append(0)
        liste2_1.append(0)
        liste3_0.append(0)
        liste3_1.append(0)
        liste4_10_0.append(0)
        liste4_10_1.append(0)
        liste10_20_0.append(0)
        liste10_20_1.append(0)
        liste20_100_0.append(0)
        liste20_100_1.append(0)
        liste100_200_0.append(0)
        liste100_200_1.append(0)
        liste200_0.append(0)
        liste200_1.append(0)
        des1.append(0)
        des2.append(0)
    prob_0,prob_1 = 0,0
    decision1 = 1.0
    decision2 = 1.0
    for l in f :
        l= l.split('|')
        if len (l ) <= 2 :
            prob_0 = float (l[0])
            prob_1 = float (l[1])
        else:
            for i in range (len(l)- 2 ):
                l[i+1]= float (l[i+1])
            if l[0]  =='0_0':
                liste0_0 = l[1:-1]
            if l[0]=='0_1':
                liste0_1= l[1:-1]
            if l[0]=='1_0':
                liste1_0=  l[1:-1]
            if l[0]=='1_1':
                liste1_1 = l[1:-1]
            if l[0]=='2_0':
                liste2_0  = l[1:-1]
            if l[0]=='2_1':
                liste2_1= l[1:-1]
            if l[0]=='3_0':
                liste3_0 = l[1:-1]
            if l[0]=='3_1':
                liste3_1= l[1:-1]
            if l[0]=='4_10_0':
                liste4_10_0 = l[1:-1]
            if l[0]=='4_10_1':
                liste4_10_1 = l[1:-1]
            if l[0]=='10_20_0':
                liste10_20_0 = l[1:-1]
            if l[0]=='10_20_1':
                liste10_20_1 = l[1:-1]
            if l[0]=='20_100_0':
                liste20_100_0 = l[1:-1]
            if l[0]=='20_100_1':
                liste20_100_1 = l[1:-1]
            if l[0]=='100_200_0':
                liste100_200_0 = l[1:-1]
            if l[0]=='100_200_1':
```

```
                liste100_200_1 = l[1:-1]
            if l[0]=='200_0':
                liste200_0 = l[1:-1]
            if l[0]=='200_1':
                liste200_1 = l[1:-1]
    f.close()
    for i in range (len(vec)- 1 ):
        if vec[i] == 0 :
            des1[i] =  liste0_0[i]
            des2[i] =  liste0_1[i]
        elif vec[i] == 1 :
            des1[i] =  liste1_0[i]
            des2[i] =  liste1_1[i]
        elif vec[i]== 2 :
            des1[i] =  liste2_0[i]
            des2[i] =  liste2_1[i]
        elif vec[i] == 3 :
            des1[i] =  liste3_0[i]
            des2[i] =  liste3_1[i]
        elif vec[i] >=4 and vec[i] < 10  :
            des1[i] =  liste4_10_0[i]
            des2[i] =  liste4_10_1[i]
        elif vec[i] >=10 and vec[i]< 20  :
            des1[i] =  liste10_20_0[i]
            des2[i] =  liste10_20_1[i]
        elif vec[i] >=20 and vec[i]< 100:
            des1[i] =  liste20_100_0[i]
            des2[i] =  liste20_100_1[i]
        elif vec[i] >=100 and vec[i] < 200:
            des1[i] =  liste100_200_0[i]
            des2[i] =  liste100_200_1[i]
        elif vec[i] >=200 :
            des1[i] =  liste200_0[i]
            des2[i] =  liste200_1[i]
    for j in range (len(des1)-1):
        decision1 = decision1 * des1 [j]
        decision2 = decision2 * des2 [j]
    des = '5 '
    decision1 = decision1 * prob_0
    decision2 = decision2 * prob_1
    if decision1 >= decision2 :
        des = 0
    else :
        des = 1
    return des
def bayesian_proj (path ,test,taille):
    test = open (path+test,'r')
nbtotal_nor,nbtotal_atk,nbtotal_faux_negatif,nbtotal_faux_positif,nbtotal_vrai_nega
tif,nbtotal_vrai_positif = 0,0,0,0,0,0
    for l in test :
        l = l.split()
        for i in range (len(l)-1 ):
            l[i]=int(l[i])
        if l [len(l)-1] == '0':
            nbtotal_nor +=1
        elif l [len(l)-1] == '1':
            nbtotal_atk +=1

        s = byasian_classifier (‘apprentissage_file.matr’,'fileout.bys',l[:-
1],taille)
        if (s == 0) and (l [len(l)- 1] == '0') :
            nbtotal_vrai_negatif += 1
        elif (s == 0) and (l [len(l)- 1] == '1') :
            nbtotal_faux_negatif += 1
        elif (s == 1) and (l [len(l)-1] == '0') :
            nbtotal_faux_positif += 1
        elif (s == 1) and (l [len(l)-1] == '1') :
            nbtotal_vrai_positif += 1
    nbtotal_vrai_negatif = nbtotal_vrai_negatif * 100 / nbtotal_nor
    nbtotal_faux_negatif = nbtotal_faux_negatif * 100 / nbtotal_atk
    nbtotal_vrai_positif = nbtotal_vrai_positif * 100 / nbtotal_atk
```

```
    nbtotal_faux_positif = nbtotal_faux_positif * 100 / nbtotal_nor
    print ('nombre totale normale :', nbtotal_nor)
    print ('nombre totale attack :', nbtotal_atk )
    print ('nombre totale vrai negatif :', nbtotal_vrai_negatif )
    print ('nombre totale vrai positif :', nbtotal_vrai_positif )
    print ('nombre totale faux positif :', nbtotal_faux_positif )
    print ('nombre totale faux negatif :', nbtotal_faux_negatif )
    test.close()
```

A.4. Code python du réseau bayésien naïve

```
def prob (path,infile,outfile):
    f = open ( path + infile , 'r' )
    out = open (path  + outfile , 'w')
    liste_pross = [] #la liste des processus
    total,prob_n,prob_a = [],[],[] # liste de probabilité pour un processus pour
être normale ou attaque sur le total normaux et attaques
    nb_nor,nb_atk ,nb_pross = 0,0,0 # nombre des processus normale et processus
atak et le total
    liste_pross_n,liste_pross_a  =[],[]
    # transformation en matrice
    for l in f :
        l =  l.split()
        for i in range (len(l) - 1 ):
            l[i]= int (l[i])
        liste_pross.append (l)
        if l [ len (l )- 1 ] == '0' :
            nb_nor += 1 # nombre normale
        if l [ len (l )- 1 ] == '1' :
            nb_atk += 1# nombre attack
    f.close ()
    out.close ()
    nb_pross = nb_nor + nb_atk # calcule du totale
    # pour le calcul de nombre d'occurrence sachant classe normale ou attaque
    for i in range (len (liste_pross[0]) -1 ):
        pros_n,pros_a = 0,0
        for j in range (len (liste_pross ) ):
            if liste_pross [j][ len(liste_pross [0] )-1 ] == '0' and liste_pross
[j][i ] > 0 :
                pros_n += 1 # calculer l'occurence normale d'un system call
            if liste_pross [j][ len(liste_pross [0] )-1 ] == '1' and liste_pross
[j][i ] > 0 :
                pros_a += 1  # calculer l'occurrence d'attaque  d'un system call
        liste_pross_n.append(pros_n) # ajouter liste des occurrence
        liste_pross_a.append(pros_a)
    for i in range (len (liste_pross_a) ):
        prob_n.append( liste_pross_n [i]/ nb_nor) # calcule du probabilité
        prob_a.append( liste_pross_a [i]/ nb_atk)  # calcule du probabilité
    p_n,p_a  = [],[]# probabilite pour les arc
    for i in range (len (prob_n) ):
        ligne_n,ligne_a = [],[]
        for j in range (len (prob_n) )  :
            if ( j > i ):
                if (prob_n [i] != 0 )and (prob_n [j] != 0 ):
                    ligne_n.append((liste_pross_n [i] + liste_pross_n [j ] ) /
nb_pross )
                else :
                    ligne_n.append(0)
            else :
                ligne_n.append(0)
            if ( j > i ):
                if (prob_a [i] != 0 )and (prob_a [j] != 0 ):
                    ligne_a.append((liste_pross_a [i] + liste_pross_a [j ] ) /
nb_pross )
                else :
                    ligne_a.append(0)
            else :
                ligne_a.append(0)

        p_n.append (ligne_n)
        p_a.append (ligne_a)
```

```
   p_n_x, p_a_x = [],[]# probabilité pour les arc  sachant x
    for i in range (len (prob_n) ):
        ligne_n,ligne_a = [],[]
        for j in range (len (prob_n) )  :
            if ( j > i ):
                ligne_n.append((liste_pross_n [i] + liste_pross_n [j ] ) / nb_nor )
                ligne_a.append((liste_pross_a [i] + liste_pross_a [j ] ) / nb_atk )
            else :
                ligne_n.append(0)
                ligne_a.append(0)
        p_n_x.append (ligne_n)
        p_a_x.append (ligne_a)
    matrice = []
    for i in range (len (p_a_x) ):
        ligne = []
        for j in range (len (p_a_x[i]) )  :
            if prob_a [i] == 0 or prob_a [j] == 0 or prob_n [i] == 0 or prob_n [j]
== 0  or p_n_x [i][j]== 0 or p_a_x [i][j] == 0 :
                ligne.append (0 )
            else :
                ligne.append (p_n [i][j] * log ( p_n_x [i][j] / (prob_n[i] *
prob_n[j]) ,2)
                              + p_a [i][j] * log ( p_a_x [i][j] /( prob_a[i] *
prob_a[j]) ,2) )
        matrice.append ( ligne )

    return matrice
matri= []
listeitem = []
for i in range (len (matri) ):
    for j in range (len (matri[i]) )  :
        if matri[i][j] != 0 :
            item = (i+1,j+1,matri[i][j])
            listeitem.append (item)
def kruskall( arc ):
    poid = 0
    chemin,ensemble = [],[] # liste de chemins et l'ensemble des sommet
    resultat = ()
    arc.sort(key = itemgetter(2),reverse=True)# trie les arcs
    for i in range (len (arc ) ) :
        resultat = cycle (ensemble, {arc[i][0]},{arc[i][1]}  )
        if  resultat[1] :
            poid +=  arc [i][2]
            chemin.append (  (arc[i][0],arc[i][1],arc [i][2]) )
    return poid , chemin

def cycle (ensemble , pred , succ):
    """ ce module test les cycle dans un graphe a l'aide de l'algorithme de Kruskal
"""
    test_pred , test_succ = False, False # pour tester l'existence de prédécesseur
et le successeur
    ajouter,sup = False,False # pour tester si un arc constitue un cycle ou non
    ensemble_pred,ensemble_succ= [],[] # pour garder l'ensemble qui  contient le
predessesseur et le successeur
    for i in range (len (ensemble)):
        if ensemble[i].issuperset(pred) : #tester existence du predecessor
            test_pred = True
            ensemble_pred = ensemble[i]
        if ensemble [i].issuperset(succ): #tester l'existence du successeur
            ensemble_succ = ensemble[i]
            test_succ = True
    if not test_pred  and not test_succ :
        ensemble.append ( pred | succ  )
        ajouter = True
    else :
        for i in range (len ( ensemble)) :
            if ensemble [i].issuperset(pred) and not test_succ :
                ensemble [i] = ensemble [i] | succ
                ajouter = True
                break
            if ensemble [i].issuperset(succ) and not test_pred :
```

```
                ensemble [i] = ensemble [i] |  pred
                ajouter = True
                break
            if test_succ and test_pred :
                if ensemble_succ != ensemble_pred:
                    ensemble.append (ensemble_pred | ensemble_succ )
                    ajouter,sup = True,True
                break
    if sup :
        for i in range (len (ensemble )) :
            if ensemble [i] == ensemble_pred :
                ensemble.remove(ensemble_pred)
                break
        for i in range (len (ensemble )):
            if ensemble [i] == ensemble_succ:
                ensemble.remove(ensemble_succ)
                break
    return ensemble, ajouter
def avg (matrice ):
    moy = 0
    for i in range( len (matrice )):
        for j in range (len (matrice[i]) )  :
            if i != j :
                moy+= matrice[i][j]
    moy /= (44 * 43)
    return moy
```

A.5. Code python de l'algorithme MWST

Annexe B : Résultats détaillés

		KNN comportementale		KNN comportementale + IF-IDF	
		Détection	Faux positive	Détection	Faux positive
K = 5	0.80	0.0	0.0	5.88235294118	0.50890585241
	0.85	3.92156862745	0.25445292620	11.7647058824	0.50890585241
	0.90	7.8431372549	0.25445292620	19.6078431373	1.01781170483
	0.93	13.7254901961	0.25445292620	41.1764705882	1.78117048346
	0.95	15.6862745098	0.76335877862	56.862745098	3.0534351145
	0.97	21.568627451	1.52671755725	70.5882352941	3.56234096692
	0.99	70.5882352941	2.7989821883	94.1176470588	8.9058524173
K = 10	0.80	3.92156862745	0.25445292620	11.7647058824	0.76335877862
	0.85	9.80392156863	0.25445292620	15.6862745098	0.76335877862
	0.90	11.7647058824	0.25445292620	25.4901960784	1.52671755725
	0.93	19.6078431373	1.01781170483	60.7843137255	3.0534351145
	0.95	19.6078431373	1.27226463104	76.4705882353	4.32569974555
	0.97	29.4117647059	1.52671755725	100.0	5.08905852417
	0.99	74.5098039216	4.83460559796	100.0	12.4681933842
K = 15	0.8	3.92156862745	0.25445292620	15.6862745098	1.01781170483
	0.85	13.7254901961	0.25445292620	25.4901960784	1.27226463104
	0.90	17.6470588235	0.25445292620	39.2156862745	2.54452926209
	0.93	19.6078431373	1.01781170483	66.6666666667	4.58015267176
	0.95	21.568627451	1.27226463104	84.3137254902	5.34351145038
	0.97	29.4117647059	2.54452926209	100.0	6.61577608142
	0.99	78.431372549	6.36132315522	100.0	19.0839694656
K = 20	0.8	7.8431372549	0.25445292620	23.5294117647	1.01781170483
	0.85	13.7254901961	0.25445292620	25.4901960784	1.78117048346
	0.90	17.6470588235	1.01781170483	47.0588235294	3.56234096692
	0.93	19.6078431373	1.78117048346	72.5490196078	4.83460559796
	0.95	23.5294117647	2.29007633588	88.2352941176	5.34351145038
	0.97	35.2941176471	3.81679389313	100.0	9.41475826972
	0.99	80.3921568627	6.87022900763	100.0	21.1195928753

Tableau B.1. Résultats détaillées pour kNN comportementale

		KNN Sans nouvelle attaque		KNN Avec des nouvelles attaque	
		Détection	Faux positive	Détection	Faux positive
K = 5	0.80	100.0	81.3559322034	94.1176470588	83.0508474576
	0.85	100.0	75.2118644068	94.1176470588	75.6355932203
	0.90	100.0	53.1779661017	88.2352941176	53.6016949153
	0.93	100.0	39.8305084746	80.3921568627	43.0084745763
	0.95	100.0	18.6440677966	76.4705882353	21.186440678
	0.97	100.0	5.29661016949	70.5882352941	6.14406779661
	0.99	100.0	0.847457627119	62.7450980392	0.21186440678
K = 10	0.80	100.0	79.4491525424	92.1568627451	77.1186440678
	0.85	100.0	68.2203389831	92.1568627451	57.8389830508
	0.90	100.0	46.8220338983	82.3529411765	42.1610169492
	0.93	100.0	30.2966101695	76.4705882353	20.3389830508
	0.95	100.0	12.7118644068	74.5098039216	8.68644067797
	0.97	100.0	3.38983050847	62.7450980392	0.635593220339
	0.99	100.0	0.21186440678	62.7450980392	0.21186440678
K = 15	0.8	100.0	63.5593220339	90.1960784314	57.2033898305
	0.85	100.0	50.0	86.2745098039	33.8983050847
	0.90	100.0	34.9576271186	76.4705882353	14.6186440678
	0.93	100.0	16.5254237288	70.5882352941	8.68644067797
	0.95	100.0	2.75423728814	62.7450980392	0.635593220339
	0.97	100.0	0.21186440678	62.7450980392	0.21186440678
	0.99	100.0	0.21186440678	62.7450980392	0.21186440678
K = 20	0.8	100.0	48.3050847458	86.2745098039	36.0169491525
	0.85	100.0	26.6949152542	78.431372549	16.313559322
	0.90	100.0	15.2542372881	68.6274509804	7.41525423729
	0.93	100.0	5.29661016949	62.7450980392	0.21186440678
	0.95	100.0	0.21186440678	62.7450980392	0.21186440678
	0.97	100.0	0.21186440678	62.7450980392	0.21186440678
	0.99	100.0	0.21186440678	62.7450980392	0.21186440678

Tableau B.2. Résultats détaillées pour kNN par signature