N° 04/2011-M/INF

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie HOUARI BOUMEDIENE

Faculté d'électronique et d'informatique

Mémoire présenté pour l'obtention du grade de Magister en Informatique

Option : Système intelligents et ingénierie de logiciels

Par AZOUZ Yacine

Thème :

UNE APPROCHE DE RECHERCHE LOCALE STOCHASTIQUE POUR LA DETECTION D'INTRUSIONS

Soutenu publiquement le 17/02/2011 devant le jury :

Mme. MOKHTARI Aicha, Professeur, USTHB, Algérie. **Présidente**
Mme. DRIAS Habiba, Professeur, USTHB, Algérie. **Directrice de mémoire**
Mlle. BOUGHACI Dalila, Maitre de conférences, USTHB, Algérie. **Examinatrice**
Mr. M. ISLI Amar, Maitre de conférences, USTHB, Algérie. **Invité**

2011

Université des Sciences et de la Technologie

HOUARI BOUMEDIENE (USTHB)

Date :

Nom : AZOUZ Yacine

Titre : une approche de recherche locale stochastique pour la détection d'intrusions

Département : informatique

Grade : Magistère en informatique

A mes chers parents

A mes frères et sœurs

A tous mes proches

A tous mes amis

Remerciements :

Si ce mémoire a pu voir le jour, c'est certainement grâce au soutien et l'aide de plusieurs personnes qui m'ont permit d'accomplir ce travail. Je profite de cet espace pour les remercier tous.

Je voudrais tout d'abord exprimer mes profonds remerciements à Madame DRIAS HABIBA, Professeur à l'université USTHB, pour être directeur de ce mémoire, pour son encadrement, sa disponibilité, sa patience et pour le soutien qu'il a eu su m'accorder durant toute cette période.

Je remercie chaleureusement Madame MOKHTARI AICHA, Professeur à l'USTHB, de me faire l'honneur de présider le jury de ce mémoire.

Je désire sincèrement remercier Madame BOUGHACI DALILA, Maitre de conférences à l'USTHB, pour son expérience, ses qualités scientifiques et humaines ont toujours été source d'enrichissement me permettant de mener à bien ce travail de recherche.

Je tiens remercier Monsieur ISLI AMAR, Maitre de conférences à l'USTHB, pour avoir accepté de faire partie de jury de ce mémoire.

En fin, de manière plus personnelle, je souhaite remercier ma famille pour leur soutien inconditionnel, merci pour m'avoir encouragé, supporté et pour avoir accepté tant de sacrifices durant cette période.

RESUME :

Les systèmes de détection d'intrusion visent à détecter des attaques contre les systèmes informatiques et les réseaux ou généralement contre des systèmes d'information. En effet, c'est difficile de fournir des systèmes d'information prouvable sécurisé et pour les maintenir dans un état sécurisé pendant leur vie et utilisation.

La détection d'intrusions nécessite qu'un grand nombre d'événements de sécurité soient collectés et enregistrés afin d'être analysés.

Deux approches pour la détection d'intrusion ont été proposées : l'approche comportementale et l'approche par scénarios. L'approche comportementale consiste à décrire le comportement (profil) usuel d'un utilisateur et ce, afin de détecter toute action anormale ou inhabituelle de cet utilisateur, elle permet de détecter des attaques inconnues. L'approche par scénarios consiste à définir des événements anormaux et ce, afin d'analyser les données susceptibles d'être des attaques, elle utilise souvent une base de scénarios d'attaques.

Dans ce mémoire nous proposons un algorithme de recherche locale stochastique pour la recherche des scénarios d'attaques prédéfinis dans le fichier d'audit de sécurité. Au début nous présentons un état de l'art sur la sécurité informatique, et les systèmes de détection d'intrusions. En suite nous présentons les algorithmes de recherche locale stochastique et leurs fonctionnements. Par la suite, nous présentons une formalisation du problème d'analyse de fichier d'audit sous forme d'un problème contraint FASPAS, et nous proposons une approche de recherche locale stochastique pour ce problème. Enfin, et pour prouver l'efficacité de cette approche, nous présentons une étude comparative entre notre approche et celle de M. SAHMADI présenté dans [59].

Mots clés :

Détection d'intrusion, approche comportementale, approche par scénario, algorithme de recherche locale stochastique (SLS), sécurité, attaques.

Table des matières

Liste des figures

Liste de Tableaux

Introduction générale

Avec le développement de l'utilisation d'internet, de plus en plus d'entreprises s'orientent vers l'ouverture de leur système d'information à leurs partenaires, leurs clients et leurs fournisseurs. Par ailleurs, avec le nomadisme, consistant à permettre aux personnels de se connecter au système d'information à partir de n'importe quel endroit, les personnels sont amenés à « transporter » une partie du système d'information hors de l'infrastructure sécurisée de l'entreprise.

Actuellement, l'enjeu principal des réseaux, et en particulier les réseaux connectés à Internet, du point de vue de la sécurité, est de se préserver des attaques externes et internes. Ces attaques peuvent nuire au maintien de la confidentialité, de l'intégrité et de la disponibilité du réseau et/ou des données qui y cheminent. Il est donc essentiel pour une entreprise de se doter d'une politique de sécurité renforcée avec des mécanismes de contrôle d'accès et de gestion des autorisations, permettant d'assurer la sécurité de toutes ses ressources, en particulier celles mises en partage sur internet ou au travers de réseaux virtuels. Il y va de même lorsque les utilisateurs appartenant à cette entreprise accèdent aux ressources de ces partenaires. On parle alors d'interopérabilité bidirectionnelle de politiques de sécurité inter-domaines dans le cadre de consortiums.

Pour faire face à ces menaces, il est essentiel de définir des politiques de sécurité et leurs mécanismes de mise en œuvre.

La sécurité, et en particulier la sécurité informatique, repose sur une analyse du risque, dans laquelle les facteurs humains et d'organisation revêtent une importance majeure. Un degré de confiance suffisant dans la mise en œuvre des mécanismes d'une politique de sécurité doit être assuré. Ainsi une pacification, formelle ou informelle, du mécanisme d'une politique de sécurité, ou d'ailleurs d'un système est essentielle.

Une autre caractéristique inhérente à la sécurité informatique est qu'elle repose sur des développements théoriques des mathématiques et de l'informatique assez importants.

1. Objectif :

Le système d'information est généralement défini par l'ensemble des données et des ressources matérielles et logicielles de l'entreprise permettant de les stocker ou de les faire circuler. Le système d'information représente un patrimoine essentiel de l'entreprise, qu'il convient de protéger.

Dans ce mémoire nous proposons une approche méta-heuristique pour la détection d'intrusions, qui permet de rechercher les scénarios d'attaques prédéfinies dans les traces d'audit. Le but de cette approche est de déterminer le plus rapidement possible la présence d'une ou plusieurs signatures d'attaques dans les données d'audit, et ce même quand le nombre de signatures d'attaques est important et que les données d'audit sont très volumineuses.

La méthode de recherche proposée se base sur un algorithme de recherche locale stochastique.

2. Organisation :

La première partie de ce mémoire présente un état de l'art, Au chapitre I nous donnons une introduction au domaine de la sécurité informatique d'une façon générale. Le deuxième chapitre donne une description des systèmes de détection d'intrusions : les termes utilisés dans le domaine de la détection d'intrusions, un modèle générique d'un processus de détection, le système d'audit de sécurité et les activités liées à l'audit, une classification des systèmes de détection d'intrusions.

Dans le troisième chapitre, nous abordons les algorithmes de recherche locale stochastique, et les principes reliés à ce type d'algorithme. Nous commençons par la citation de quelques définitions, puis les structure de ces méthodes, et les principes types de méthodes SLS et nous donnons des exemples pour chaque type.

Le chapitre quatre donne une formalisation de problème d'analyse de fichier d'audit de sécurité, une définition des scénarios d'attaques, et la méthode de recherche locale stochastique proposée pour résoudre ce problème.

Dans le cinquième chapitre nous présentons une architecture globale de système de détection d'intrusions à base de méthode SLS. Nous détaillons l'approche SLS proposée pour l'analyse de fichier d'audit de sécurité, et à la fin de chapitre nous présentons les expérimentations effectuées sur l'algorithme implanté, et les résultats obtenus.

Enfin, nous terminons par une conclusion générale, et quelques perspectives qui peuvent améliorer notre travail.

Chapitre I : *La sécurité informatique*

1. Introduction :

Les évolutions récentes et rapides de l'informatique ont contribué à l'accélération des échanges d'informations. Les entreprises se trouvent désormais confrontées au contrôle efficace de la confidentialité, de l'intégrité et de la disponibilité de ces informations.
Véritable point névralgique, le système d'information est souvent la proie de multiples attaques qui menacent l'activité économique des entreprises et requièrent la mise en place d'une politique interne de sécurité.

2. Terminologie de la sécurité informatique :

La sécurité informatique utilise un vocabulaire bien défini. Nous allons définir certains de ces termes :

2.1 La menace :

En anglais « Theart », représente le type d'action susceptible de nuire dans l'absolu.

2.2 Les vulnérabilités :

En anglais "vulnerability", appelée parfois *faille* ou *brèche*, ce sont les failles de sécurité dans un ou plusieurs systèmes. Tout système vu dans sa globalité présente des vulnérabilités, qui peuvent être exploitables ou non.

2.3 Les contre-mesures:

Ce sont les procédures ou techniques permettant de résoudre une vulnérabilité ou de contrer une attaque spécifique (auquel cas il peut exister d'autres attaques sur la même vulnérabilité).

2.4 Risque :

Le risque en termes de sécurité est généralement caractérisé par l'équation suivante :
Risque = menace * vulnérabilité / contre-mesure
Les contre-mesures à mettre en œuvre ne sont pas uniquement des solutions techniques mais également des mesures de formation et de sensibilisation à l'intention des utilisateurs, ainsi qu'un ensemble de règles clairement définies.

Afin de pouvoir sécuriser un système, il est nécessaire d'identifier les menaces potentielles, et donc de connaître et de prévoir la façon de procéder de différentes attaques.

3. Objectifs de la sécurité informatique :

La sécurité informatique, d'une manière générale, consiste à assurer que les ressources matérielles ou logicielles d'une organisation sont uniquement utilisées dans le cadre prévu.

La sécurité informatique vise généralement cinq principaux objectifs :

- **L'intégrité**, c'est-à-dire garantir que les données sont bien celles que l'on croit être ;
- **La confidentialité**, consistant à assurer que seules les personnes autorisées aient accès aux ressources échangées ;
- **La disponibilité**, permettant de maintenir le bon fonctionnement du système d'information ;
- **La non répudiation**, permettant de garantir qu'une transaction ne peut être niée ;
- **L'authentification**, consistant à assurer que seules les personnes autorisées aient accès aux ressources.

3.1 La confidentialité :

La confidentialité consiste à rendre l'information inintelligible à d'autres personnes à l'exception des acteurs de la transaction, Ceci signifie que le système informatique doit :
- empêcher les utilisateurs de lire une information confidentielle (sauf s'ils y sont autorisés).
- empêcher les utilisateurs autorisés à lire une information et de la divulguer à d'autres utilisateurs (sauf autorisation) [01].

3.2 L'intégrité :

Vérifier l'intégrité des données consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle), Cela signifie que le système informatique doit:

- empêcher une modification indue de l'information, c'est-à-dire une modification par des utilisateurs non autorisés ou une modification incorrecte par des utilisateurs autorisés.
- faire en sorte qu'aucun utilisateur ne puisse empêcher la modification légitime de l'information.

3.3 La disponibilité :

L'objectif de la disponibilité est de garantir l'accès à un service ou à des ressources pour un utilisateur autorisé, c.-à-d. le système informatique permet de :

- Fournir l'accès à l'information pour que les utilisateurs autorisés puissent la lire ou la modifier.
- Faire en sorte qu'aucun utilisateur ne puisse empêcher les utilisateurs autorisés d'accéder à l'information [01].

3.4 La non-répudiation :

La non-répudiation de l'information est la garantie qu'aucun des correspondants ne pourra nier la transaction (communication), et donc par la suite, ce service permet de garantir qu'un message a bien été envoyé par un émetteur et bien reçu par un destinataire. Ce service est particulièrement important dans les courriers électroniques et dans les applications commerciales électroniques.

3.5 L'authentification :

L'authentification consiste à assurer l'identité d'un utilisateur, c'est-à-dire de garantir à chacun des correspondants que son partenaire est bien celui qu'il croit être. Un contrôle d'accès peut permettre (par exemple par le moyen d'un mot de passe qui devra être crypté) l'accès à des ressources uniquement aux personnes autorisées.

4. Sécurité des systèmes informatiques :

D'une manière générale [54] le système informatique concerne l'ensemble des moyens (organisation, acteurs, procédures, outils de sécurité …) nécessaires à l'élaboration, au traitement, au stockage, à l'acheminement et à l'exploitation des informations.
Dans les faits, de nos jours, l'essentiel du système d'information est porté par le système informatique et la notion de sécurité informatique recouvre pour l'essentiel la notion de sécurité des systèmes d'information (SSI).

Le concept de SSI recouvre donc un ensemble de méthodes, techniques et outils chargés de protéger les ressources d'un système informatique afin d'assurer la disponibilité des services, la confidentialité et l'intégrité des informations.

Les échanges au travers notamment d'Internet ont rendu également nécessaire le développement de propriétés nouvelles comme l'authentification, la paternité et la traçabilité de l'information.
La sécurité fait donc appel à différentes techniques complémentaires dont :

- le chiffrement de l'information (cryptologie).
- la protection contre les signaux parasites compromettants (sécurité électronique).
- la protection contre les accidents naturels et les actes malveillants (sécurité physique).
- la protection contre les intrusions dans les logiciels, mémoires ou banques de données.

Les éléments font l'objet du sujet de notre travail, et qui sera bien détaillés dans le chapitre suivant.

4.1 Les menaces du système informatiques :

Bien que souvent invisibles, du moins tant qu'elles n'ont pas eu de conséquences directes, les menaces sont cependant bien réelles :

4.1.1 Les menaces physiques :

Actes de délinquance (vols, détérioration) et accidents naturels sont des menaces physiques qui visent directement le matériel. Souvent ignorés, les évènements naturels, accidentels ou malveillants, représentent pourtant jusqu'à 8% des sinistres déclarés. [66]
Quelle entreprise peut se prétendre totalement à l'abri d'une inondation, d'une tempête ou d'un incendie d'autant que, si le matériel peut être, le plus souvent, aisément remplacé il n'en va pas de même des données qu'il contenait ?

4.1.2 Les menaces informatiques :

Bien plus connues et envisagées, dès lors qu'on parle de sécurité des systèmes d'informations, les menaces informatiques (virus, chevaux de Troie, spams...) n'en sont pas moins de réels dangers. En 2003 environ 18% des entreprises ayant répondu à l'enquête du CLUSIF (étude sur la sinistralité) ont déclaré avoir été infectées par un ou plusieurs virus et l'impact financier en a été jugé élevé dans 11% des cas.
Heureusement la plupart des entreprises ont désormais saisi l'importance et l'intérêt des outils destinés à se prémunir de ces attaques (antivirus, firewall) et ces dernières voient leur efficacité reculer d'années en années. [66]

4.1.3 Les menaces internes :

Bien moins identifiées, les menaces internes sont plus souvent liées à la négligence et à l'ignorance du personnel de l'entreprise ; il s'agit plus d'inconscience que d'une réelle volonté de nuire.

En général ces menaces correspondent à un usage personnel du matériel informatique de l'entreprise avec d'une part le risque d'infection par virus (surtout dans le cas des ordinateurs portables confiés aux employés) et d'autre part un risque pénal par le téléchargement de programmes ou de fichiers pirates (films, musiques) qui sont incompatibles avec les applications professionnelles ou utilisés en dehors du cadre légal (licences d'exploitation).

Les cas d'espionnage industriel sont plus rares mais restent néanmoins un danger bien réel qui pèse sur les entreprises œuvrant sur des marchés stratégiques. La sécurité des systèmes d'information passe aussi par la mise en place d'une politique efficace de protection visant à empêcher toute possibilité d'action malveillante par du personnel temporairement affecté à l'entreprise (stagiaires...). [66]

4.2 Les différents types d'attaques :

Les informations ou les systèmes d'informations d'une entreprise peuvent subir des dommages de plusieurs façons : certains intentionnels (malveillants), d'autres par accident. Ces événements seront appelés des « attaques ».

Il existe plusieurs catégories principales d'attaque : [5]

- L'accès.
- La modification.
- Le déni de service.
- La répudiation.

4.2.1 Les attaques d'accès :

Une attaque d'accès est une tentative d'accès à l'information par une personne non autorisée. Ce type d'attaque concerne la confidentialité de l'information.

Parmi les attaques d'accès les plus connus :

a- Le sniffing :

Cette attaque est utilisée par les pirates informatiques pour obtenir des mots de passe. Grâce à un logiciel appelé renifleur de paquets (sniffer), on peut intercepter toutes les paquets qui circulent sur un réseau même ceux qui ne nous sont pas destinés. [5]

b- Les chevaux de Troie :

Les chevaux de Troie sont des programmes informatiques cachés dans d'autres programmes. Ce nom vient de la légende grecque de la prise de Troie à l'aide d'un cheval en bois rempli de soldats qui attaquèrent la ville une fois à l'intérieur. [5]

c- Le craquage de mots de passe :

Le craquage consiste à faire de nombreux essais jusqu'à trouver le bon mot de passe. Il existe deux grandes méthodes : [5]

· *L'utilisation de dictionnaires* : le mot testé est pris dans une liste prédéfinie contenant les mots de passe les plus courants et aussi des variantes de ceux-ci (à l'envers, avec un chiffre à la fin…). Les dictionnaires actuels contiennent dans les 50 000 mots et sont capables de faire une grande partie des variantes.

· *La méthode brute* : toutes les possibilités sont faites dans l'ordre jusqu'à trouver la bonne solution.

d- L'ingénierie sociale :

L'ingénierie sociale (social engineering en anglais) n'est pas vraiment une attaque informatique, c'est plutôt une méthode pour obtenir des informations sur un système ou des mots de passe. Elle consiste surtout à se faire passer pour quelqu'un que l'on n'est pas (en général un des administrateurs du serveur que l'on veut pirater) et de demander des informations personnelles (login, mots de passe, accès, numéros, données…) en inventant un quelconque motif (plantage du réseau, modification de celui-ci…). Elle se fait soit au moyen d'une simple communication téléphonique ou par courriel. [5]

e- Porte dérobée :

Lorsqu'un pirate informatique arrive à accéder à un serveur à l'aide d'une des techniques présentées dans cette section, il souhaiterait y retourner sans avoir à tout recommencer. Pour cela, il laisse donc des portes dérobées (backdoor) qui lui permettrons de reprendre facilement le contrôle du système informatique. [5]

Il existe différents types de portes dérobées :

· Création d'un nouveau compte administrateur avec un mot de passe choisi par le pirate.

· Création de compte ftp

· Modification des règles du pare-feu pour qu'il accepte des connections externes.

Dans tous les cas, l'administrateur perd le contrôle total du système informatique. Le pirate peut alors récupérer les données qu'il souhaite, voler des mots de passe ou même détruire des données. [5]

4.2.2 Les attaques de modification :

Une attaque de type « modification » consiste, pour un attaquant à tenter de modifier des informations. Ce type d'attaque est dirigé contre l'intégrité de l'information.

Les vers, virus, Les chevaux de Troie : Il existe une grande variété de virus qui peuvent causer ce type d'attaque. On peut cependant définir un virus comme un programme caché dans un autre qui peut s'exécuter et se reproduire en infectant d'autres programmes ou d'autres ordinateurs. [5]

Les dégâts causés vont du simple programme qui affiche un message à l'écran au programme qui formate le disque dur après s'être multiplié. On ne classe cependant pas les virus d'après leurs dégâts mais selon leur mode de propagation et de multiplication :

· Les vers capables de se propager dans le réseau;

· Les « chevaux de Troie » créant des failles dans un système;

· Les bombes logiques se lançant suite à un événement du système;

· Les canulars envoyés par mail.

4.2.3 Les attaques par saturation (déni de service) :

Les attaques par saturation sont des attaques informatiques qui consiste à envoyer des milliers de messages depuis des dizaines d'ordinateurs, dans le but de submerger les serveurs d'une société, de paralyser pendant plusieurs heures son site Web et d'en bloquer ainsi l'accès aux internautes. [5]

Cette technique de piratage assez simple à réaliser est jugée comme de la pure malveillance. Elle ne fait que bloquer l'accès aux sites, sans en altérer le contenu.

Parmi les attaques par saturation les plus connus, on trouve :

· Le flooding

· Le TCP-SYN flooding

· Le smurf

· Le débordement de tampon

a- Le flooding :

Cette attaque consiste à envoyer à une machine de nombreux paquets IP de grosse taille. La machine cible ne pourra donc pas traiter tous les paquets et finira par se déconnecter du réseau. [5]

b- Le TCP-SYN flooding :

Le TCP-SYN flooding est une variante du flooding qui s'appuie sur une faille du protocole TCP. En effet, on envoie un grand nombre de demande de connexions au serveur (SYN) à partir de plusieurs machines. Le serveur va envoyer un grand nombre de paquet SYN-ACK et attendre en réponse un paquet ACK qui ne viendra jamais. Si on envoie les paquets plus vite que le timeout des « demi-connexion » (connexions autorisées mais non terminé), le serveur sature et finit par se déconnecter. [5]

c- Le smurf :

C'est un "ping" flooding un peu particulier. C'est une attaque axée réseaux, faisant partie de la grande famille des Refus De Service (DOS : Denial Of Service). [5]

Ce procédé est décomposé en deux étapes:

- La première est de récupérer l'adresse IP de la cible par spoofing.
- La seconde est d'envoyer un flux maximal de packets ICMP ECHO (ping) aux adresses de Broadcast. Chaque ping comportant l'adresse spoofée de l'ordinateur cible.

Si le routeur permet cela, il va transmettre le broadcast à tous les ordinateurs du réseau, qui vont répondre à l'ordinateur cible. La cible recevra donc un maximum de réponses au ping, saturant totalement sa bande passante...

d- Le débordement de tampon :

Cette attaque se base sur une faille du protocole IP. On envoie à la machine cible des données d'une taille supérieure à la capacité d'un paquet. Celui-ci sera alors fractionné pour l'envoi et rassemblé par la machine cible. A ce moment, il y aura débordement des variables internes. Suite à ce débordement, plusieurs cas se présentent : la machine se bloque, redémarre ou ce qui est plus grave, écrit à l'extérieur de l'espace alloué au tampon, écrasant ainsi des informations nécessaires au processus. [5]

4.2.4 Les attaques de répudiation :

La répudiation est une attaque contre la responsabilité. Autrement dit, la répudiation consiste à tenter de donner de fausses informations ou de nier qu'un événement ou une transaction se soient réellement passé.

Parmi les attaques de répudiation, on trouve Le "IP spoofing", cette attaque consiste à se faire passer pour une autre machine en falsifiant son adresse IP. Elle est en fait assez complexe. Il existe des variantes car on peut "spoofer" aussi des adresses e-mail, des serveurs DNS ou NFS. [5]

5. Les mécanismes de sécurité :

5.1 Cryptographie :

C'est l'étude des principes, méthodes et techniques mathématiques reliés aux aspects de la sécurité de l'information tels que la confidentialité, l'intégralité des données, l'authentification d'entités et l'authentification de l'originalité des données. C'est un ensemble de techniques qui fournit la sécurité de l'information. La cryptographie permet de stocker des informations sensibles ou de les transmettre à travers des réseaux non sûrs (comme Internet) de telle sorte qu'elles ne peuvent être lues par personne à l'exception du destinataire convenu.

5.2 Le contrôle d'accès :

Le contrôle d'accès doit garantir que les utilisateurs (et les processus qui agissent pour le compte de ceux-ci) se voient interdire l'accès aux informations et ressources auxquelles ils ne sont pas autorisés à accéder.

Les ressources peuvent être physiques : par exemple un bâtiment, un local, un pays, ou logiques : par exemple un système d'exploitation ou une application informatique spécifique.

Le contrôle peut être réalisé à l'aide de l'utilisation d'éléments permettant l'authentification de l'entité (par exemple un mot de passe, une carte, une clé, un élément biométrique, …).

6. Les différents outils de sécurité :

6.1 Les Firewalls :

En informatique, un Firewall est un système physique (matériel) ou logique (logiciel) qui protège la partie privée d'un réseau de la partie publique. C'est en réalité l'élément qui permet de distinguer la partie privée du réseau de celle que l'on nommera publique (Internet), lui seul peut donc en atteindre les deux extrémités. Il permet donc de protéger le réseau prive d'éventuelles attaques provenant d'Internet et peut également contrôler certaines actions effectuées de l'intérieur du réseau prive. [59]

De manière simplifiée, Les Firewalls examinent chaque paquet de données entrant ou sortant et en bloquent certains selon des règles de filtrage prédéfinies.

Voici un schéma descriptif :

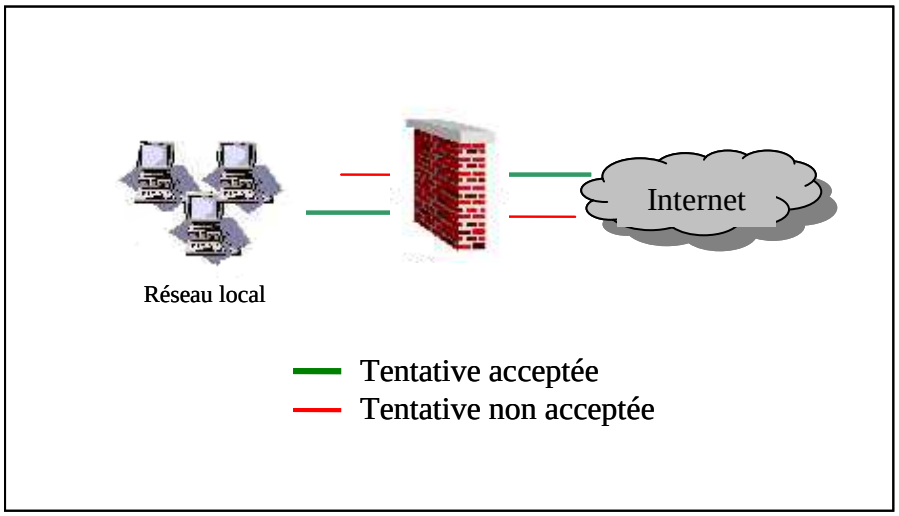


Fig. I.4 - Le Firewall. [59]

6.2 Les IDS :

On appelle IDS (Intrusion Detection System) un mécanisme écoutant le trafic réseau de manière furtive afin de repérer des activités anormales ou suspectes et permettant ainsi d'avoir une action de prévention sur les risques d'intrusion.

6.3 Les antivirus :

Les premiers antivirus sont apparus en 1988 sous la forme de graticiels et de partagiciels (freeware et shareware) suite au besoin pressant de la part des entreprises de se débarrasser des premières infections (on comptait alors une dizaine de souches différentes dans le monde..).

Le nombre croissant de ces dernières a conduit les concepteurs de ces « produits miracles » à fonder leurs sociétés commerciales en 1991. Le nombre d'infections recensées est passe de 18 en 1989 pour avoisiner les 32000 en l'an 2000 !

Ces logiciels ont fait des progrès considérables mais aujourd'hui encore il n'existe pas de mesures anti-infections idéales. Toutefois, l'utilisation judicieuse de mesures simples et de campagnes de sensibilisation peut réduire les risques à un niveau acceptable.

6.4 Scanners de vulnérabilités :

Les scanners de vulnérabilités automatisent la découverte des failles de sécurité. Ils sont utilisés par les administrateurs pour localiser les faiblesses du réseau et corriger les vulnérabilités de leurs systèmes informatiques.

Cependant les scanners présentent quelques limites qui peuvent être résumées en trois points : l'exhaustivité, la mise à jour et l'exactitude. En effet, malgré le grand nombre de vulnérabilités détectées, les scanners d'aujourd'hui sont inaptes à déterminer toutes les faiblesses possibles. De plus, la mise à jour de ces produits ne suit pas le rythme de la découverte des nouvelles vulnérabilités.

7. Conclusion :

Ce chapitre nous a permet d’aborder plusieurs points au niveau de la sécurité des informatique et de mieux comprendre les différents mécanismes mis en œuvre régulièrement dans l'utilisation des réseaux informatique ouverts sur Internet.

Dans le chapitre suivant nous parlerons sur les systèmes de détection d’intrusion.

Chapitre II : *Les Systèmes de détection d'intrusion*

1. Introduction :

La croissance de l'internet et l'ouverture des systèmes ont fait que les attaques dans les réseaux informatiques soient de plus en plus nombreuses. D'une part les vulnérabilités en matière de sécurité s'intensifient, au niveau de la conception des protocoles de communication ainsi que au niveau de leur implantation et d'autre part les connaissances, les outils et les scriptes pour lancer les attaques sont facilement disponibles et exploitables. D'où la nécessité d'un système de détection d'intrusions (IDS). Cette technologie consiste à rechercher une suite de mots et/ou de paramètres caractérisant une attaque dans un flux de paquets. Les systèmes de détection d'intrusion ont devenus un composant essentiel et critique dans une architecture de sécurité informatique.

Un IDS doit être conçu dans une politique globale de sécurité, dont l'objectif est de détecter toute violation liée à la politique de sécurité, et de signaler les attaques.

2. Définitions:

2.1. Une intrusion :

Une intrusion peut être considérée comme l'ensemble d'actions qui ont pour but de compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource. Ces actions de franchissement d'un accès non-autorisé ou de manipulation interdite d'une ressource, peuvent être menées par un individu externe n'ayant aucun privilège sur les ressources d'un système, ou par un individu interne qui outrepasse ses privilèges.

2.2. La Détection d'intrusion :

La détection d'intrusion est un ensemble de techniques et de méthodes employées dans l'analyse des informations collectées par les mécanismes d'audit de sécurité pour détecter toute soupçonneuses activités au niveau de réseau et ses hosts.

2.3. Une attaque :

Une attaque peut être définie comme toute action ou ensemble d'actions qui peut porter atteinte à la sécurité d'informations d'un système ou d'un réseau informatique. Elle représente le moyen d'exploiter une vulnérabilité. Il peut y avoir plusieurs attaques pour une même vulnérabilité.

2.4. Événement :

Nous appellerons événement toute opération élémentaire du système reportée par un mécanisme d'audit.

2.5. Mécanisme d'audit :

Un mécanisme d'audit est un ensemble de code du système informatique qui a la responsabilité de garder les traces de toute opération ou transaction exécutées dans ce système.

2.6. Journal d'audit :

Nous appellerons journal d'audit l'ensemble des informations générées par les mécanismes d'audit. Les enregistrements du journal d'audit sont appelés traces d'audit.

2.7. Faux positif : (Détection en absence d'attaque)

Une alerte provenant d'un IDS mais qui ne correspond pas à une attaque réelle, Autrement dit : Signifie qu'un système IDS détecte une intrusion là où aucune intrusion réelle n'a été perpétrée.

2.8. Faux négatif : (absence de détection en présence d'attaque)

C'est la non-génération d'alarme par un IDS pour un événement illégal. A l'inverse d'un faux positif, le faux négatif signifie que le système IDS n'a pas détecté une intrusion ayant réussi.

Log : Une ligne d'un fichier d'un logiciel qui enregistre les données transitant sur un système pour le surveiller ou faire des statistiques.

Le fichier log est un fichier qui contient les événements s'étant produits sur un système.

3. Systèmes de détection d'intrusion (IDS):

On appelle IDS (Intrusion Detection System) un mécanisme écoutant le trafic réseau de manière furtive afin de repérer des activités anormales ou suspectes et permettant ainsi d'avoir une action de prévention sur les risques d'intrusion.

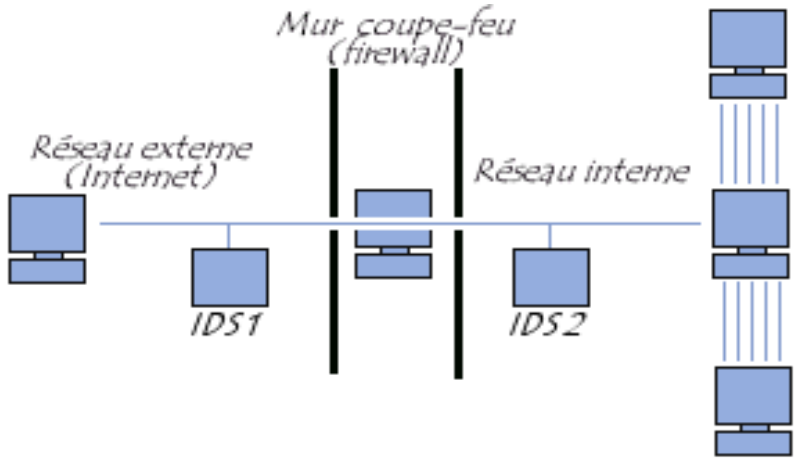


Fig. II.1 : Les systèmes IDS

3.1. Modèle Général D'IDS :

L'IDWG (Intrusion Detection Working Group de l'IETF) a proposé un modèle générique pour les systèmes de détection d'intrusion (voir Fig. II.02)

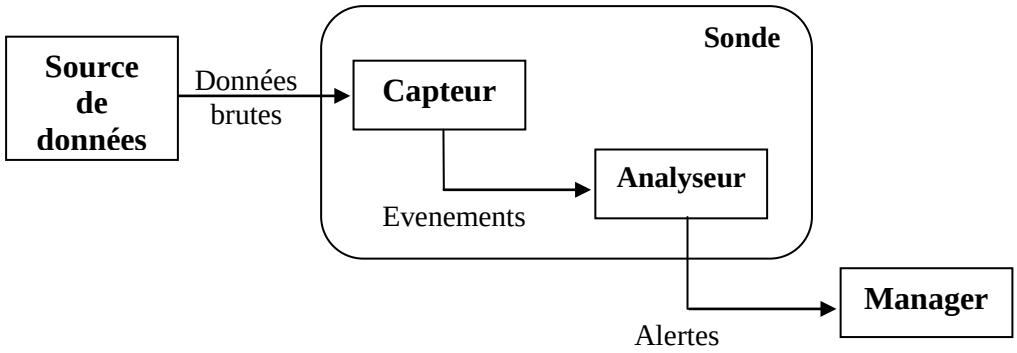


Fig. II.2: Modèle général de système de détection d'intrusions. [59]

- ***Source de données*** : est un dispositif générant de l'information sur les activités des entités du système d'information. Exemple : *sniffers* réseau, système d'audit d'un système d'exploitation, etc.
- ***Capteur*** : est un dispositif générant des événements en filtrant et formatant les données brutes provenant d'une source de données.

- ***Evénement*** : est un message formaté émis par un capteur. C'est l'unité élémentaire utilisée pour représenter une étape d'un scénario d'attaque connu. Ces événements sont parfois appelés événements d'audit, ou données d'audit.
- ***Analyseur*** : est un dispositif analysant les événements à la recherche de traces d'intrusions.
- ***Alerte*** : est message formaté émis par un analyseur s'il y a des traces d'intrusions.
- ***Sonde*** : est un ensemble constitué d'un capteur et d'un analyseur.
- ***Manager*** : est un composant permettant à l'administrateur du système de configurer les différents éléments (capteur, analyseur) et de gérer les alertes reçues.

3.2. Classification Des IDS :

Ils existent plusieurs critères pour classifier les IDS, parmi eux on a :

- Source des données ;
- Méthode de détection ;
- Analyse des données ;
- Fréquence de l'analyse ;
- Comportement après détection.

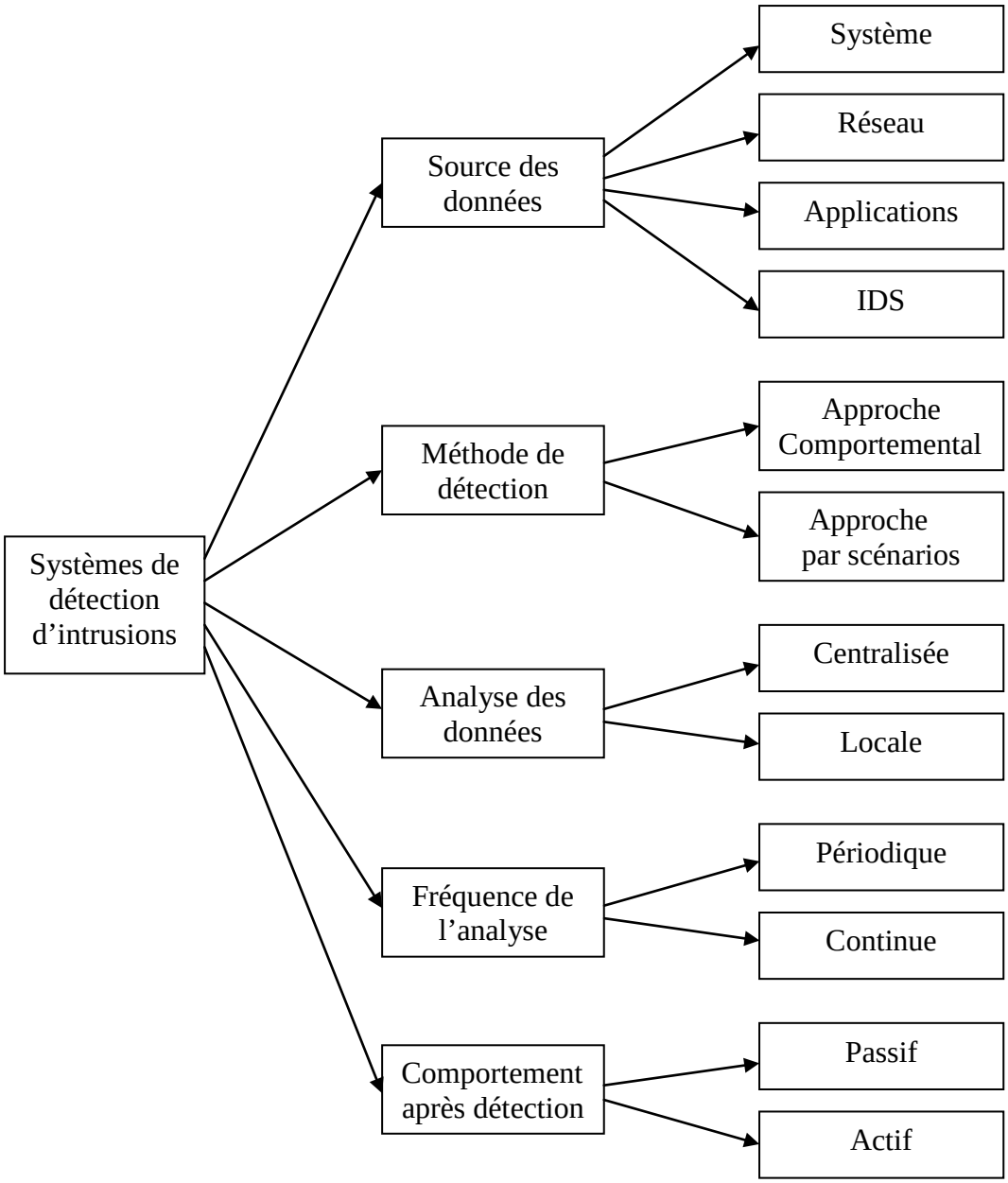


Fig. II.3 - Caractéristiques des systèmes de détection d'intrusions. [59]

3.2.1. L'emplacement des sources de données :

L'emplacement des sources d'audits est le plus connus pour classifier les IDS en plusieurs familles :

3.2.1.1. Sources de données à base de réseaux :

Les IDSs peuvent se classer selon trois catégories majeures selon qu'ils s'attachent à surveiller :

- **Les NIDS** (*Network Based Intrusion Detection System*), qui assurent la sécurité au

niveau du réseau.

- Les **HIDS** (*Host Based Intrusion Detection System*), qui assurent la sécurité au niveau des hotes.
- Les **IDS hybrides**, qui utilisent les NIDS et HIDS pour avoir des alertes plus Pertinentes.

a- Les NIDS :

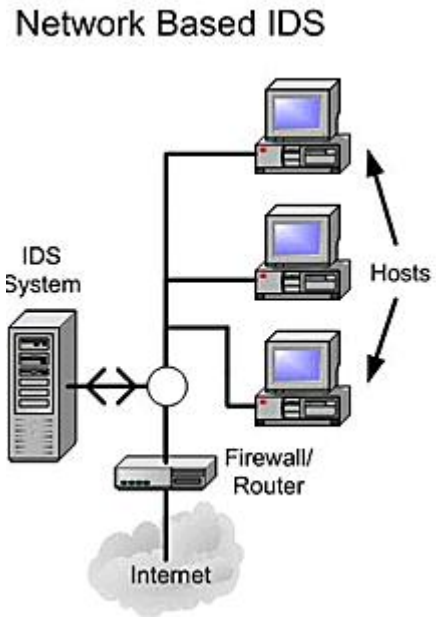


Fig. II.4 - les N-IDS

Un NIDS se découpe en trois grandes parties : La capture, les signatures et les alertes (Voir Fig. II.5), il écoute donc tout le trafic réseau, puis l'analyse les flux en transit sur le réseau et génère des alertes si des paquets semblent dangereux.

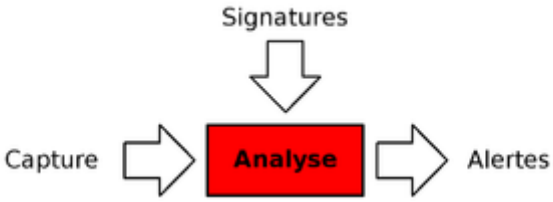


Fig. II.5 – les composants d'un NIDS

Le rôle essentiel d'un NIDS est l'analyse et l'interprétation des paquets circulant sur ce réseau.

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, pour les analyser et les traiter éventuellement. [52]

Les capteurs du 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. [52]

a- *Les avantages des NIDSs* :

- Ils peuvent être complètement cachés sur le réseau, donc un attaquant ne saura pas qu'il est contrôlé.
- Un système NIDS unique peut être employé pour contrôler le trafic d'un grand nombre de systèmes cibles potentiels.
- Ils peuvent capturer le contenu de tous les paquets envoyés à un système cible.

b- *Les inconvénients des NIDSs :*

- Ils ne peuvent donner d'alarmes que si le trafic correspond aux règles ou aux signatures préconfigurées.
- Ils peuvent manquer le trafic intéressant si le trafic est important sur la bande passante ou si des routes altérées sont utilisées.
- Il ne peut pas déterminer si une attaque a réussie.
- Il ne peut pas examiner le trafic chiffré.
- Il faut des configurations spéciales sur les réseaux commutés pour que le NIDS puisse voir tout le trafic.

b- Les HIDS :

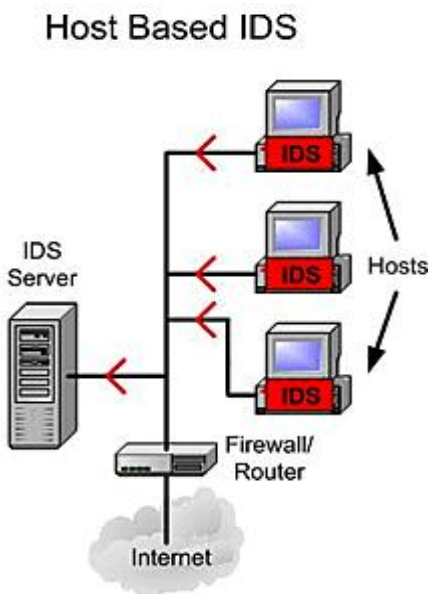


Fig. II.6 - les HIDS

Le HIDS réside sur un hôte particulier, il analyse exclusivement l'information concernant cet hôte. Le HIDS se comporte comme un démon ou un service standard sur un hôte serveur/système.

De plus, l'impact sur la machine concernée est sensible immédiatement, par exemple dans le cas d'une attaque réussie par un utilisateur. Ces systèmes de détection d'intrusions utilisent deux types de sources pour fournir une information sur l'activité de la machine : les logs et les traces d'audit du système d'exploitation : les traces d'audit sont plus précises et détaillées et fournissent une meilleure information alors que les logs qui ne fournissent que l'information essentielle sont plus petits. [52]

c- Les IDS hybrides :

Les IDS hybrides rassemblent les caractéristiques de plusieurs IDS différents. En pratique, on ne retrouve que la combinaison de NIDS et HIDS. Ils permettent, en un seul outil, de surveiller les réseaux et les terminaux. Les sondes sont placées en des points stratégiques, et agissent comme NIDS et/ou HIDS suivant leurs emplacements. Toutes ces sondes remontent alors les alertes à une machine qui va centraliser le tout (voir Fig. II.6), et agréger/lier les informations d'origines multiples. [60]

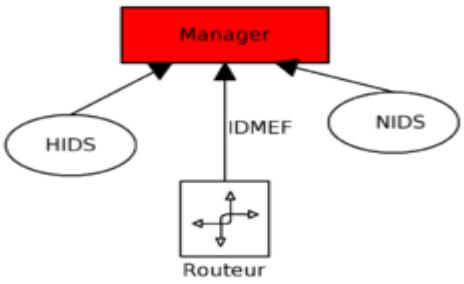


Fig. II.7 : IDS hybride.

Les IDS hybrides sont donc basés sur une architecture distribuée, où chaque composant unifie son format d'envoi d'alerte (Exemple typique : IDMEF : Intrusion Detection Message Exchange Format) permettant à des composants divers de communiquer et d'extraire des alertes plus pertinentes.

Les avantages des IDS hybrides sont multiples :

- Moins de faux positifs.
- Meilleure corrélation.
- Possibilité de réaction sur les analyseurs

3.2.1.2. Source d'information à base de système :

Parmi les sources d'information à analyser qui peuvent être fournies par un système d'exploitation sont : les commandes systèmes, l'audit de sécurité et l'accounting. [59]

a- Les commandes systèmes :

La plupart des systèmes d'exploitation fournissent des commandes permettant de savoir les différentes fonctionnalités qui s'exécutent sur le système. Par exemple la commande "ps" et "vmstat" fournissent des informations précises sur les activités courantes du système.

b- Audit de sécurité :

Un audit est défini comme un examen d'un groupe, compte individuel, ou activité. Ainsi, le sous-système d'audit fournit un moyen de dépistage et d'enregistrer ce qui se passe sur votre système.

Ce service qui est proposé dans les systèmes d'exploitation moderne permet d'avoir dans fichier tous les événements système associés aux utilisateurs, dans ce fichier on trouve bien pour chaque utilisateur, quels sont les différentes actions qui ont été exécutées, par exemple accès en écriture à un fichier, exécution d'un script…etc.

Parmi les systèmes d'audit avancés est l'audit BSM (Solaris Basic Security Mode Auditing), Lorsque ce dernier est activé, il fournit une trace sur tous les applications et processus qui s'exécutent sur Solaris.

c- Accounting :

L'accounting fournit l'information sur l'usage des ressources partagées par les utilisateurs (temps processeur, mémoire, espace disque, débit réseau, applications lancées, ...). Les modules statistiques et neuronaux d'Hyperview [11] ont utilisé cette source d'information.

Parmi les systèmes d'accounting est l'accounting d'AIX qui nous donne des informations sur l'utilisation des ressources système AIX, ces données peuvent être utilisées pour [58]:

- Contrôler l'utilisation de certaines ressources (comme l'espace disque par exemple).
- Justifier la nécessité d'une expansion du système.
- Effectuer le suivi des performances.
- Maintenir une piste d'audit à des fins de sécurité.

3.2.1.3. Source d'information applicative :

Les systèmes de détection d'intrusions basés sur les applications sont un sous-groupe des HIDSs. 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 on opère entre un utilisateur et un programme, il est facile de filtrer tout comportement notable. Ils se situent au niveau de la communication entre un utilisateur et l'application surveillée. [8]

L'avantage de ce système de détection d'intrusions 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. [7]

Par contre, du fait que ce système de détection d'intrusions 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 de système de détection d'intrusions sont des cibles faciles pour les attaquants et ne sont pas aussi sûrs. [7]

Ce type des systèmes de détection d'intrusions 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 systèmes de détection d'intrusions afin de ne pas compromettre les performances de la machine. [7]

3.2.1.4. Source d'information basée IDS :

Une autre source d'information, souvent de plus haut niveau que les précédentes, peut être exploitée. Il s'agit des alertes remontées par des analyseurs provenant d'un IDS. Chaque alerte synthétise déjà un ou plusieurs événements intéressants du point de vue de la sécurité. Elles peuvent être utilisées par un IDS pour déclencher une analyse plus fine à la suite d'une indication d'attaque potentielle. De surcroît, en corrélant plusieurs alertes, on peut parfois détecter une intrusion complexe de plus haut niveau. Il y aura alors génération d'une nouvelle alerte plus synthétique que l'on qualifie de méta alerte.

La frontière entre un événement et une alerte est parfois floue car tout dépend à quel niveau d'abstraction on se place. C'est le cas notamment chez les IDS où le capteur et l'analyseur sont indissociables [9].

3.2.2. Méthodes de détection :

Deux approches de détection ont été proposées à ce jour, l'approche comportementale (anomaly detection) et l'approche par scénario (misuse detection ou knowledge based detection).

La première se base sur l'hypothèse que l'on peut définir un comportement «normal» de l'utilisateur et que toute déviation par rapport à celui-ci est potentiellement suspecte. La seconde s'appuie sur la connaissance des techniques employées par les attaquants : on en tire des scénarios d'attaque et on recherche dans les traces d'audit leur éventuelle survenue. [2][3]

3.2.2.1. L'approche comportementale :

Dans cette approche, l'IDS construit un profil du comportement de l'utilisateur ou de l'application. Ainsi toute activité ne respectant le profil créé engendrera une alerte. Autrement dit qu'une intrusion implique un usage anormal du système et donc un comportement inhabituel d'un utilisateur.

Cette méthode peut se dérouler en deux étapes :

- Définir de modèles de comportements considérés comme « normaux » (profil).
- Recherche de déviation par rapport à cette norme.

L'objectif à atteindre par cette méthode c'est de répondre à la question : **le comportement actuel de l'utilisateur est-il cohérent avec son comportement passé ?**

Notion de profil : un profil est une vue synthétique du comportement d'un utilisateur ou d'un groupe d'utilisateurs.

On peut distinguer deux catégories de profils : les profils construits par apprentissage (empiriques) et les profils spécifiant une politique de sécurité (policy-based) :

a- Profils construits par apprentissage (empiriques) :

Parmi les méthodes proposées pour construire les profils par apprentissage, les plus marquantes sont les suivantes :

- Méthode statistique : le profil est calculé à partir de variables considérées comme aléatoires et échantillonnées à intervalles réguliers. Ces variables peuvent être le temps processeur utilisé, la durée et l'heure des connexions, etc. Un modèle statistique (ex : covariance) est alors utilisé pour construire la distribution de chaque variable et pour mesurer, au travers d'une grandeur synthétique, le taux de déviation entre un

comportement courant et le comportement passé. L'outil NIDES [12] utilise cette méthode.

- Système expert : ici, c'est une base de règles qui décrit statistiquement le profil de l'utilisateur au vu de ses précédentes activités. Son comportement courant est comparé aux règles, à la recherche d'une anomalie. La base de règles est rafraîchie régulièrement.
- Réseaux de neurones : la technique consiste à apprendre à un réseau de neurones le comportement de l'entité à surveiller. Par la suite, lorsqu'on lui fournira en entrée les actions courantes effectuées par l'entité, il devra décider de leur normalité. L'outil Hyperview5 [11] comporte un module de ce type.
- Analyse de signatures : l'approche s'inspire de l'immunologie biologique et met en œuvre des algorithmes de pattern matching. Il s'agit de construire un modèle de comportement normal des services réseaux. Le modèle consiste en un ensemble de courtes séquences d'appels système représentatifs de l'exécution normale du service considéré. Des séquences d'appels étrangères à cet ensemble sont alors considérées comme l'exploitation potentielle d'une faille du service [10].

b- Profils spécifiant une politique de sécurité (policy-based) :

Les IDS dits policy-based, il n'y a pas de phase d'apprentissage. Leur comportement de référence est spécifié par une politique de sécurité : la détection d'une intrusion intervient chaque fois que la politique est violée.

Dans [10], Zimmermann, Mé et Bidan proposent un système IDS où le profil est une politique de sécurité représentée par un graphe qui définit les flux d'informations autorisés entre objets du système. Dans ces deux cas, toute séquence d'appels systèmes, pour être conforme à la politique de sécurité (profil), doit vérifier un certain prédicat (théorème prouvé). Par opposition, une suite d'appels systèmes qui ne vérifie pas ce prédicat contient une violation de la politique de sécurité.

Pour toutes ces méthodes, le comportement de référence utilisé pour l'apprentissage étant rarement exhaustif, on s'expose à des risques de fausses alarmes (faux positifs). De plus, si des attaques ont été commises durant cette phase, elles seront considérées comme normales (risque de faux négatifs). L'avantage de cette approche c'est la possibilité de détecter les intrusions inconnues.

Voici un schéma qui résume le fonctionnement d'un IDS à approche comportementale :

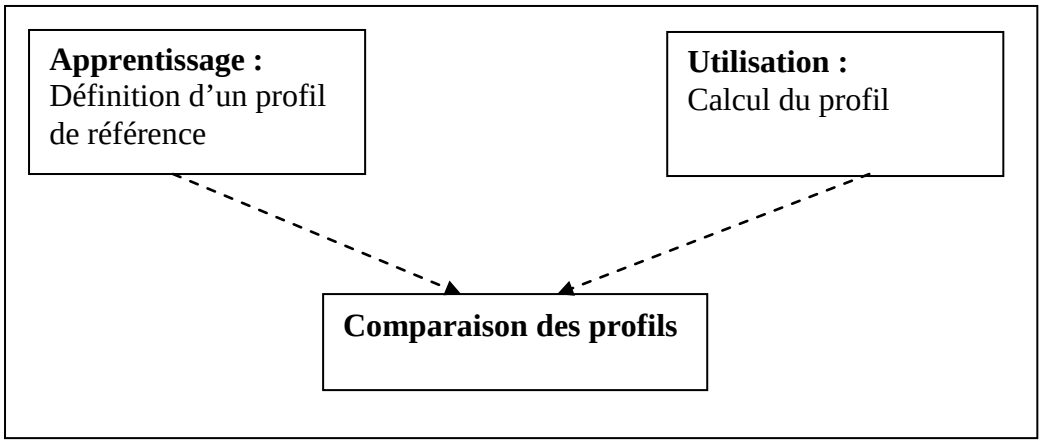


Fig. II.8 Fonctionnement d'un IDS à l'approche comportementale.

***Avantages* :**

- Pas besoin d'une base d'attaque
- Détection d'intrusions inconnues possible

***Inconvénients* :**

- Pour un utilisateur au comportement erratique, toute activité est normale.
- En cas de profonde modification de l'environnement du système cible, déclenchement d'un flot ininterrompu d'alarmes de gros risque de faux positifs.
- Utilisateur pouvant changer lentement de comportement dans le but d'habituer le système à un comportement intrusif risque de faux négatifs

3.2.2.2. L'approche par scénarios :

Dans cette approche, on compare les actions des utilisateurs aux scénarios d'attaques que l'on connaît. On se base dans cette méthode sur une base de connaissances répertoriant tous les scénarios d'attaques.

Les méthodes proposées à ce jour sont les suivantes :

- Analyse de signatures :

Il s'agit là de la méthode la plus en vue actuellement. Des signatures d'attaques sont fournies à des niveaux sémantiques divers selon les outils (de la suite d'appels système aux commandes passées par l'utilisateur en passant par les paquets réseau). Divers algorithmes sont utilisés pour localiser ces signatures connues dans les traces d'audit. Ces signatures sont toujours exprimées sous une forme proche des traces d'audit. La plupart des IDS commerciaux utilisent cette méthode, tels que : Realsecure [13] ou NetRanger [14].

- Automates à états finis :

Plusieurs IDS utilisent des automates à états finis pour coder le scénario de reconnaissance de l'attaque. Cela permet d'exprimer des signatures complexes et comportant plusieurs étapes. On passe d'un état initial sûr à un état final attaqué via des états intermédiaires. Chaque transition entre états est déclenchée par des conditions sur les événements remontés par les capteurs. Par exemple l'outil NetSTAT [15] utilise des diagrammes de transitions d'états (STD :State Transition Diagrams) pour représenter les scénarios d'attaques.

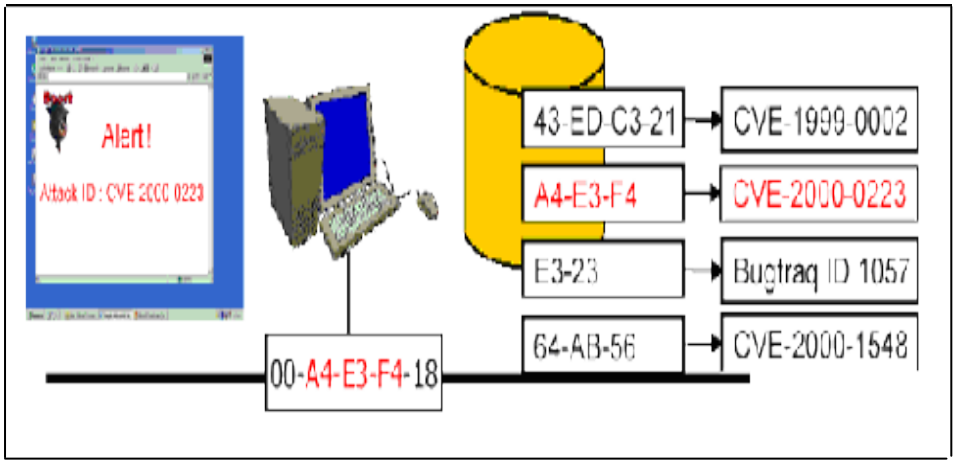


Fig. II.9 Fonctionnement d'un IDS à l'approche par scénarios.

Avantage :

- Prise en compte des comportements exacts des attaquants potentiels, donc peu de faux positifs.

Inconvénient :

- Base de scénarios difficile à construire et, surtout à maintenir, donc il y a un risque d'avoir des faux négatifs.
- Pas de détection d'attaques non connues, et dans ce cas il y aura un risque des faux négatifs.
- Détection de scénarios complexe difficile.

3.2.3. Localisation de l'analyse des données :

On peut également faire une distinction entre les IDS en se basant sur la localisation réelle de l'analyse des données :

3.2.3.1. Analyse centralisée :

Certains IDS ont une architecture multi-capteurs (ou multi-sondes). Ils centralisent les événements (ou alertes) pour analyse au sein d'une seule machine. L'intérêt principal de cette architecture est de faciliter la corrélation entre événements puisqu'on dispose alors d'une vision globale. Par contre, la charge des calculs (effectués sur le système central) ainsi que la charge réseau (due à la collecte des événements ou des alertes) peuvent être lourdes et risquent de constituer un goulet d'étranglement.[59]

3.2.3.2. Analyse locale :

Si l'analyse du flot d'événements est effectuée au plus près de la source de données (généralement en local sur chaque machine disposant d'un capteur), on minimise le trafic réseau et chaque analyseur séparé dispose de la même puissance de calcul. En contrepartie, il est impossible de croiser des événements qui sont traités séparément et l'on risque de passer à côté de certaines attaques distribuées.

3.2.4. Fréquence de l'analyse :

Suivant la fréquence d'analyse, on trouve deux modes de fonctionnement des IDS :

- IDS à Analyse continue.
- IDS à Analyse périodique.

3.2.4.1. Analyse continue (en temps réel) :

Dans l'analyse continue, il y a une surveillance dynamique du système, on peut donc détecter une intrusion en temps réel. Aujourd'hui la majorité des IDS réseau utilise ce type d'analyse, cette technique prédominante de synchronisation pour les NIDSs, qui récoltent l'information du trafic réseau. Par conséquent les systèmes de détection d'intrusions peuvent prendre des actions pour affecter la progression d'une attaque détectée.

Cependant les informations fournies par l'IDS ne sont pas tout le temps pertinentes, dans certains cas, il y a tellement de volume de données à traiter qu'il est impossible de le faire en temps réel. De plus certaines actions ne peuvent être repérées que trop tard, pour un système temps réel.

3.2.4.2. Analyse périodique(En temps différé) :

Dans l'analyse périodique, on ne détecte pas réellement les intrusions, on détecte plutôt leurs effets, ou leurs prémices. On effectue en fait une analyse périodique des fichiers "journal" du réseau. Cette approche est employée aussi dans les HIDSs qui scrutent les logs du système d'exploitation dans des intervalles de temps réguliers.

3.2.5. Comportement après détection :

Le comportement après la détection décrit la réponse du système de détection d'intrusion à une attaque, elle est qualifiée d'active, si le détecteur réagit activement par des actions correctives, ou proactives (changer les règles de filtrage de Firewall des connexions TCP, encore attaquer l'attaquant, etc...). Si le système de détection d'intrusion génère simplement des alarmes (afficher un message sur l'écran, générer un son spécifique, envoi d'un email, archivage dans un fichier ou dans une base de données, etc...), la réponse est qualifiée de passive. [53][3]

Donc une autre façon de classer les systèmes de détection d'intrusions qui consiste à les classer par type de réaction lorsqu'une attaque est détectée :

3.2.5.1. Réponse active :

Les réponses actives des systèmes de détection d'intrusions sont des actions automatisées prises quand certains types d'intrusions sont détectés.

Il y a trois catégories de réponses actives : [53]

- *Rassembler des informations additionnelles :*

Il est très important de rassembler des informations additionnelles sur une attaque afin de l'identifier avec précision. Chacun de nous a fait probablement l'équivalent de cela une fois réveillée par un bruit étrange pendant la nuit. La première chose à faire dans une telle situation est d'écouter d'avantage, recherchant l'information additionnelle qui nous permet de décider si on doit agir ou non. Dans le cas des systèmes de détection d'intrusions, cela se traduira par l'exigence d'analyse des informations additionnelles, faire de corrélations, ou bien communiquer avec d'autres types de systèmes de détection d'intrusions installés sur le réseau. [52]

- *Changer l'environnement :*

Une autre réponse active doit stopper une attaque en progression ensuite bloquer l'accès de l'attaquant. Typiquement les systèmes de détection d'intrusions n'ont pas les capacités de bloquer l'accès d'une personne spécifique, mais ils peuvent uniquement rompre des connexions ou bloquer certains paquets spécifiques en s'appuyant sur les mécanismes des protocoles Internet. Parmi ces actions on trouve : [52]

- L'envoi des paquets TCP de type Reset ou des paquets ICMP au système de l'attaquant pour arrêter la connexion.

- La configuration des routeurs et des Firewalls pour bloquer les paquets provenant des adresses IP de l'attaquant.

- La configuration des routeurs et des Firewalls pour bloquer les paquets selon le numéro de port, le protocole, ou le service utilisé par l'attaquant.

- Agir contre l'intrus :

La première option dans la réponse active est d'agir contre l'intrus. En effet, la forme la plus agressive de cette réponse implique le lancement des contres attaques ou d'essayer d'obtenir activement les informations sur l'hôte ou l'emplacement de l'attaquant.

La première question concernant le choix de cette option même avec beaucoup d'attention est : « est ce que notre action peut être illégale ?». Beaucoup d'attaquants emploient de fausses adresses de réseau quand ils attaquent des sites Internet ou de torts causés aux utilisateurs innocents. Donc, il faut prendre les actions avec plus de prudence. [52]

3.2.5.2. Réponse passive :

Les réponses passives des systèmes de détection d'intrusions fournissent l'information nécessaire aux administrateurs réseau et aux responsables de la sécurité pour les aider à prendre des mesures basées sur cette information. Beaucoup de systèmes de détection d'intrusions se fondent seulement sur des réponses passives dont les principales sont : [53]

- L'alarme :

Les alarmes sont produites par les systèmes de détection d'intrusions pour informer les administrateurs réseau quand des attaques sont détectées. La forme la plus connue est d'afficher un message d'alerte concernant des informations détaillées de l'intrusion détectée

sur la console du responsable de la sécurité réseau. Une autre option très utile consiste à envoyer ces alertes au téléphone du responsable, on peut aussi envoyer des emails, ou générer des alertes sonores. [52]

- SNMP Trap :

Certains systèmes de détection d'intrusions sont conçus pour produire des alertes et envoyer les rapports aux systèmes de gestion du réseau (Network Management System). Ils utilisent le protocole SNMP (Sample Network Management Protocol), qui est un protocole dédié à la gestion du réseau. [52]

- L'archivage :

L'archivage permet aux analystes de faire des analyses approfondies, et de faire des corrélations avec l'historique dont ils disposent concernant les événements qui se sont produits auparavant. [52]

4. L'audit de sécurité :

Le mécanisme d'audit de sécurité consiste à enregistrer tout ou une partie d'actions effectuées sur le système pour en faire une analyse a posteriori, permet de détecter d'intrusions.

L'objectif de l'audit est ainsi de vérifier que chaque règle de la politique de sécurité est correctement appliquée et que l'ensemble des dispositions prises forme un tout cohérent. Un audit de sécurité permet ainsi de s'assurer que l'ensemble des dispositions prises par l'entreprise sont réputées sûres.

Toute opération entreprise sur un système informatique se traduit par une séquence d'actions effectuées par le système. Ces actions sont appelées "*activités système*". Une activité système intervenant à un certain moment est appelée "*évènement*". Un journal d'audit de sécurité est un fichier dans lequel est enregistré chronologiquement tout ou partie des évènements. Les enregistrements du journal d'audit sont aussi appelés *''traces d'audit''*. [55]

4.1. Activités liées à l'audit de sécurité :

Le journal d'audit doit permettre d'enregistrer toutes les opérations liées à la sécurité, de telle sorte que ces opérations soient réussies (parce qu'elle sont autorisées) ou qu'elles aient échouées (empêchées par les mécanismes de contrôle d'accès). Les principales opérations à surveiller sont: [56]

- la connexion et la déconnexion des utilisateurs ;
- la création, modification, destruction des informations de sécurité (droits d'accès, mots de passe, etc.) ;
- les changements de privilèges.

Les journaux d'audit doivent porter sur tous les utilisateurs (y compris les administrateurs et les responsables de la sécurité) et contenir un maximum d'informations utiles (date et heure, identité de l'utilisateur, type d'opération, référence de l'information...etc.).

4.1.1. Spécification des activités système à auditer :

L'administrateur de sécurité définira, selon le niveau de sécurité qu'il souhaite atteindre, des évènements auditables correspondant à certaines informations. [55] fournit une liste non exhaustive des informations à auditer classée comme suivent :

a) Informations sur les accès au système :

Il s'agit d'obtenir des informations sur ce qui constituera le point de départ de toute investigation sur une violation éventuelle de la sécurité :

- qui a accédé au système (identifiant de l'utilisateur ou du processus),
- quand (horodatage de l'accès au système),
- où (identifiant du terminal, adresse),
- comment (mode d'entrée : interactif, batch local, connexion distante).

b) Informations sur l'usage fait du système :

Il s'agit de montrer quelles ressources du système ont été utilisées et comment :

- commandes systèmes utilisées et leur résultat (échec ou succès),
- accès à une unité d'entrée/sortie,
- utilisation de la CPU,
- taux d'occupation mémoire par utilisateur.

c) Informations sur l'usage fait des fichiers :

Puisque les fichiers contiennent l'information, c'est un point très sensible, alors pour chaque accès à un fichier, on s'intéressera aux points suivants :

- horodatage de l'accès,
- type de l'accès (ouverture, fermeture, lecture, ajout d'enregistrement(s), modification d'enregistrement(s), purge du fichier, ...),
- source de l'accès (triplet (utilisateur, terminal, application)),

- volume d'informations échangées lors de l'accès.

d) Informations relatives à chaque application :

Chaque application conduit à des évènements qui peuvent influer sur la sécurité du système. Dans ce cadre on pourra enregistrer les évènements suivants :

- les lancements et arrêts d'application,
- les modules réellement exécutés,
- les commandes exécutées et leurs résultats (échec ou succès),
- les données entrées,
- les sorties produites.

e) Informations sur les violations éventuelles de la sécurité :

Il s'agit d'enregistrer tous les évènements pour lesquels il y a eu une tentative d'accès à une ressource du système sans que cet accès soit autorisé par les règles de la politique de sécurité. Parmi ces évènements, on peut citer :

- la tentative d'exécution d'une application dans un mode privilégié (par exemple la modification des droits d'accès sous Unix),
- la tentative d'accès à un fichier non autorisé ou la fourniture d'un mot de passe erroné pour cet accès,
- la tentative d'utilisation de certaines commandes du système réservées à des utilisateurs privilégiés,
- le changement des droits d'accès à des fichiers sensibles,
- l'accès au système à des moments ou depuis des lieux inhabituels. Il faut définir ce qui est habituel pour chaque utilisateur, un groupe d'utilisateurs, ou tous les utilisateurs. Les informations de ce type doivent être exploitées rapidement de manière à limiter les effets éventuels d'une atteinte à la politique de sécurité. Même si on a une exploitation en quasi-temps réel, on gardera une trace de ces informations pour en permettre une étude ultérieure.

f) Informations statistiques sur le système :

Il est possible de repérer des activités anormales dans un système en observant attentivement quelques facteurs clé. Par exemple, on pourra tirer des conclusions des informations suivantes :

- niveau anormalement élevé des refus d'accès au système,
- niveau anormalement élevé ou bas de l'usage de certaines commandes du système (en particulier les commandes demandant des privilèges particuliers).

4.1.2. Collecte des événements :

La collecte des événements peut être assurée grâce à des sous-systèmes d'audit fournis par la plupart des systèmes d'exploitation. Ces systèmes d'audit permettent de générer certains types d'événements, dont la collecte est assurée par le noyau du système. Par exemple nous retrouvons au niveau du système UNIX, un certain nombre de journaux d'audit, qui gardent une trace de tout ce qui se passe sur le système. D'autres outils permettant la collecte des événements sont les sondes réseaux, qui permettent de récupérer les données du trafic réseau (les paquets circulant sur le réseau) comme l'outil "TCP Dump".

4.1.3. Analyse du journal d'audit :

L'analyse des journaux d'audit est une activité essentielle dans un environnement où des violations de sécurité sont produites, afin de détecter toute violation des règles de sécurité. Elle permet ainsi à l'administrateur de sécurité de :

- Déterminer les auteurs de ces violations,
- Déterminer les vulnérabilités du système,
- Estimer les dégâts éventuels et d'assurer leur réparation,
- Reconstruire l'historique de ce qui s'est passé dans le système,
- Faire une comptabilité.

Afin de faciliter la tâche et d'aider l'officier de sécurité dans l'analyse des journaux d'audit, il serait judicieux d'utiliser des outils qui analyseraient ces journaux de manière automatique. C'est la tache des systèmes de détection d'intrusions qui permet surtout d'analyser les journaux d'audits concernant les accès au système et aux données.

5. Les critères de bon fonctionnement d'un IDS :

5.1.1. L'efficacité d'un système IDS :

Un IDS n'est efficace que :

1. S'il est constamment à jour : le fonctionnement de l'IDS repose sur une base de signatures qui doit être enrichie constamment pour prendre en compte les nouvelles attaques.
2. S'il est correctement configuré pour son environnement : un IDS mal configuré génère un nombre important d'alertes non pertinentes et inexploitables.
3. Si les alertes qu'il génère sont traitées et analysées en permanence.

4. Si les informations sur ces alertes et leur traitement sont facilement accessibles aux exploitants.
5. S'il contribue à l'application rigoureuse de procédures de réponse.

5.1.2. Les critères de choix d'un IDS :

Les systèmes de détection d'intrusion sont devenus indispensables lors de la mise en place d'une infrastructure de sécurité opérationnelle. Ils s'intègrent donc toujours dans un contexte et dans une architecture imposant des contraintes très diverses. Certains critères (toujours en accord avec le contexte de l'étude) peuvent être dégagés:

5.1.2.1. Fiabilité :

Pour qu'un IDS soit fiable, il faut que toutes les alertes générées doivent être justifiées et aucune intrusion ne doit pouvoir lui échapper.

5.1.2.2. *Réactivité* :

Un IDS doit être capable de détecter les nouveaux types d'attaques le plus rapidement possibles; pour cela il doit rester constamment à jour. Des capacités de mise à jour automatique sont pour ainsi dire indispensables.

5.1.2.3. Facilité de mise en œuvre et adaptabilité :

Un IDS doit être facile à mettre en œuvre et surtout s'adapter au contexte dans lequel il doit opérer ; il est inutile d'avoir un IDS émettant des alertes en moins de 10 secondes si les ressources nécessaires à une réaction ne sont pas disponibles pour agir dans les mêmes contraintes de temps.

5.1.2.4. *Performance :*

La mise en place d'un IDS ne doit en aucun cas affecter les performances des systèmes surveillés. De plus, il faut toujours avoir la certitude que l'IDS a la capacité de traiter toute l'information qui est à sa disposition, par exemple un IDS réseau doit être capable de traiter l'ensemble du flux pouvant se présenter à un instant donné dans le réseau sans jamais supprimer de paquets, car dans le cas contraire il devient trivial de masquer les attaques en augmentant la quantité d'information.

5.1.3. Les imperfections des IDS :

Avec la croissance rapide de l'Internet, les incidents de la sécurité ont été augmentés. En outre, la technologie s'est développée en une approche complexe comme les attaques coordonnées et les attaques coopératives. Sous ces circonstances, il y a un grand besoin pour des outils logiciels qui peuvent automatiquement détecter une variété d'intrusions. En tant que des portiers importants d'un réseau, les systèmes de détection d'intrusions doivent avoir la capacité de détecter et de défendre des intrusions plus proactivement dans un bref délai. Cependant, les systèmes de détections d'intrusions de nos jours présentent quelques imperfections. [4]

Beaucoup de systèmes de détection d'intrusions existants (NIDS et HIDS) sont faits d'un seul bloc ou module qui se charge de toute l'analyse. Ils réalisent leur collecte de données ainsi que leur analyse en utilisant une architecture centralisée. Ces systèmes monolithiques exigent beaucoup de données d'audit, ce qui consomme beaucoup de ressources de la machine à protéger, posent des problèmes de mise à jour et font de ce point central d'analyse une cible d'attaque propice (Si un intrus parvient à le faire compromettre alors la totalité du réseau se retrouve sans protection).

6. Conclusion :

Les IDS sont actuellement des produits mûrs et aboutis. Ils continuent d'évoluer pour répondre aux exigences technologiques du moment mais offrent d'ores et déjà un éventail de fonctionnalités capables de satisfaire les besoins de tous les types d'utilisateurs. Néanmoins comme tous les outils techniques ils ont des limites que seule une analyse humaine peut compenser.

Les détecteurs d'intrusion deviennent aussi de plus en plus sensibles aux erreurs liées à l'analyse des données d'audit. Par conséquent, il est plus que fondamental d'utiliser d'autres techniques d'analyse plus performantes et qui permettent aux IDS de nous donner des résultats plus fiables.

Dans le chapitre suivant, nous exposerons une approche intelligente de résolution de problèmes et qui est la recherche locale stochastique. Cette méthode nous permettre par la suite d'aborder le problème de détection d'intrusions.

Chapitre III : *Les algorithmes de recherche locale stochastique*

1. Introduction :

Les algorithmes SLS (Stochastic local search) sont parmi les techniques les plus efficaces pour résoudre les problèmes difficiles dans de nombreux domaines de l'informatique, la recherche opérationnelle et d'ingénierie. De manière générale, les techniques SLS vont de l'amélioration des algorithmes assez simple constructifs et itératifs aux méthodes à usage général, largement connues sous le nom de méta-heuristiques, comme l'optimisation des fourmis, le calcul évolutif, la recherche locale itérative, les algorithmes mimétique, le recuit simulé, la recherche tabou et la recherche de voisinage variable.

En raison de leur polyvalence et de l'efficacité et simplicité de leur implémentation, les méthodes de recherche locales stochastiques jouissent d'une popularité sans cesse croissante chez les chercheurs et praticiens.

2. Définitions:

2.1 Espace de recherche :

L'ensemble des solutions candidates pour une instance d'un problème donné constitue certain nombre de composants de la solution discrète.

2.2 Ensemble de solutions :

Il spécifie toutes les positions de recherche qui sont considérées comme des solutions (réalisables) de l'instance du problème donné.

2.3 Relation de voisinage :

Elle spécifie les voisins directs de chaque solution candidate nommé s, soit, les positions de recherche qui peuvent être atteintes à partir de s en une seule étape de recherche.

2.4 Etats Mémoire:

Il permet de conserver des informations sur le mécanisme de recherche pour chaque étape de recherche (par exemple, Tabu tenure des composants de la solution à la recherche tabou, ou la température dans le recuit simulé). Il peut être composé d'un seul état constant simple comme dans le cas des algorithmes qui n'utilisent pas de mémoire tels que l'amélioration itérative simple.

2.5 Fonction objectif:

Elle détermine le calcul des étapes de recherche par le mapping de chaque position de recherche et de l'état de la mémoire à partir d'une distribution de probabilité sur ses positions de recherche voisines et états de la mémoire.

2.6 Prédicat de terminaison (arrêt):

Il est utilisé pour décider l'arrêt de la recherche fondé sur la position de la recherche actuelle et l'état de la mémoire.

3. Structure des méthodes SLS :

Sur la base des éléments définis précédemment, un algorithme SLS fonctionne comme l'illustre la figure III.1. Lorsqu'il est appliqué à des problèmes d'optimisation, dont la définition comprend une fonction objectif qui spécifie la qualité des solutions, un algorithme SLS nécessité de garder une trace de la meilleure solution rencontrée pendant le processus de recherche. Ce dernier est généralement arrête dès qu'une solution pour l'instance du problème donnée est trouvée.

Algorithme de recherche locale stochastique :

Début

Détermine l'état initial de la recherche

Tant que le critère de terminaison n'est pas satisfait :

Faire :

Effectuer une étape de recherche.

Si nécessaire, mettre à jour la solution actuelle.

Fait

Retourner la dernière solution courante.

Fin

Fig. III.1: Schéma général d'un algorithme SLS**.** [31]

L'étape de recherche consiste à exécuter les opérations de voisinage en basant sur la solution courante et un paramètre probabiliste dont l'objectif est de trouver une solution condidate qui sera prise comme nouvelle solution courante. Pour mettre à jour la solution courante, il existe

deux moyens possibles, soit nous choisissons aléatoirement une solution voisine, soit nous sélectionnons la voisine qui donne la meilleure valeur de la fonction objectif.

3.1. Les Relations de voisinage :

Parmi [17] les composantes de base de n'importe quel algorithme SLS, la relation de voisinage et la fonction objectif sont essentielles. En règle générale, les relations de voisinage doivent être définies dans un problème de manière spécifique, et il est souvent difficile de prédire lequel des différents choix qui peuvent être faits dans ce contexte et qui se traduira par la meilleure performance. Cependant, les types standards de relations de voisinage existent, surtout largement utilisés sont les voisinages appelés : k-échange, dans lesquels deux solutions candidates sont des voisins directs, si et seulement si, elles diffèrent au plus de k composantes de solutions.

À titre d'exemple, nous considérons le voisinage 2-échange du problème (symétrique) du voyageur de commerce (TSP), selon lequel deux solutions candidates sont des voisins directs si l'une d'elles peut être obtenue à partir de l'autre suite au remplacement de deux arêtes par une paire de deux autres arêtes (voir Figure III.2).

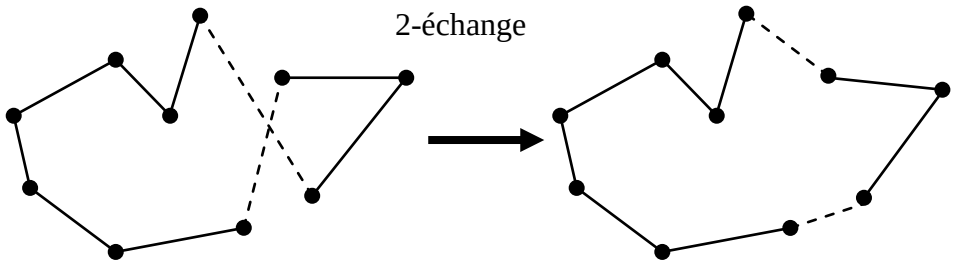


Fig. III.2 Exemple de 2-exchange pour le TST symétrique.

Toute relation de voisinage induit un graphe de voisinage, dont l'ensemble de sommets correspond à l'espace de recherche donné, et dans lequel chaque couple de voisins des positions de recherche est relié par une arête (voir Figure 3.2). De nombreuses propriétés importantes de la relation de voisinage sont consignées dans le graphe de voisinage, par exemple, le voisinage k-échange symétriques induit k-graphes de voisinage ordinaire.

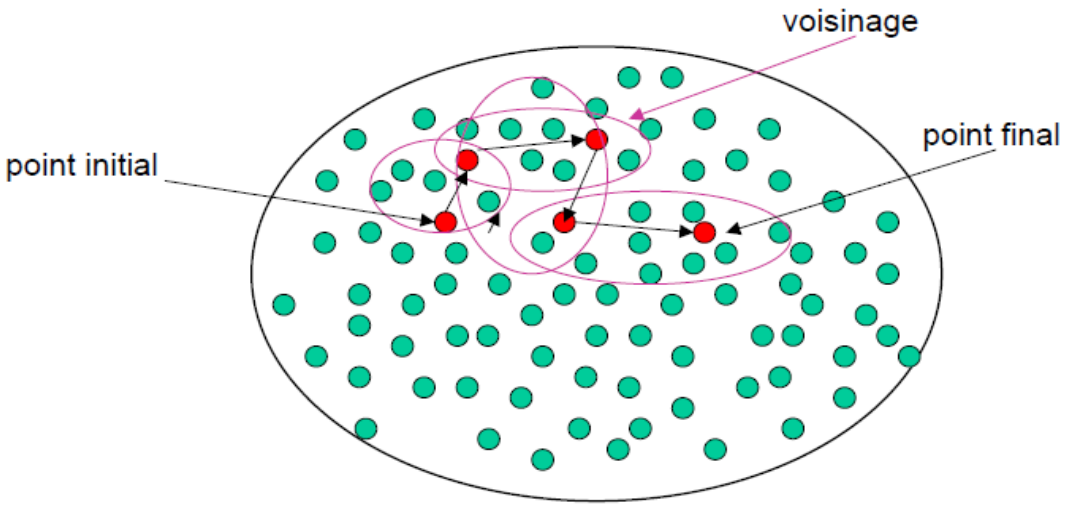


Fig. III.2: Exemple d'un graphe de voisinage.

3.2. La fonction objectif :

La fonction objectif définit la manière dont le processus de recherche se déplace d'un état de recherche à un autre, où un état de recherche est une combinaison d'une position de recherche et d'un état de mémoire. Les étapes de recherche sont généralement déterminées au moyen d'une procédure qui dessine un échantillon de la distribution de probabilité déterminée par la fonction objectif sous-jacente. Les spécifications procédurales similaires sont utilisées pour les fonctions d'initialisation et celles de terminaison. Le processus de recherche est habituellement guidé par une fonction d'évaluation qui est utilisé pour mesurer l'heuristique ou le rang des solutions candidates.

Pour des problèmes d'optimisation combinatoire, la fonction objectif qui fait partie de la définition du problème est également souvent utilisée comme une fonction d'évaluation. Dans le cas de problèmes de décision combinatoire, il peut y avoir une grande liberté dans le choix d'une fonction d'évaluation appropriée. En outre, certaines méthodes SLS utilisent les fonctions d'évaluation multiples ou modifient la fonction d'évaluation lors de la recherche. (Pour cette raison, nous ne précisons pas explicitement une fonction d'évaluation unique en tant que composante de notre définition formelle d'un algorithme de SLS.)

4. Concepts et Discussion :

Le concept de la recherche locale stochastique, tel que défini ci-dessus, fournit un cadre unificateur pour de nombreux types d'algorithmes. En particulier, il capture des algorithmes constructifs, dont l'espace de recherche contiennent des solutions partielles d'un problème

donné, c'est à dire, des solutions candidates qui peuvent être étendues en ajoutant des composants de la solution (comme les arêtes dans le cas du TSP). Par conséquent, les algorithmes qui sont fortement basés sur les procédures de recherche de manière constructive, comme le GRASP ou d'optimisation de colonie de fourmis, entrent dans le cadre SLS général. De même, les algorithmes basés sur la population, qui opèrent sur des ensembles de solutions candidates, se conforment par la définition générale de SLS en tenant compte des positions de recherche se composent d'ensembles de solutions candidates individuelles. [17]

De nombreux algorithmes SLS sont basés sur des méthodes génériques pour le contrôle et la direction d'un simple processus de recherche subsidiaire (par exemple une procédure d'amélioration itérative, voir la section 5), et les efforts de recherche considérables ont été axées sur le développement et l'étude de ces méthodes SLS générales, qui sont aussi communément appelés méta-heuristiques [33]. Bien que les stratégies de recherche de haut niveau et les mécanismes de fournir une base importante pour le développement d'algorithmes SLS pour un large éventail de problèmes, il est important de garder à l'esprit que d'autres aspects, y compris le choix de l'espace de recherche et de la relation de voisinage, ainsi que les techniques pour la mise en œuvre efficace des mesures de recherche locale, sont également des éléments essentiels éligible pour une bonne performance d'algorithmes SLS.

Une recherche dans le domaine de la recherche locale stochastique comprend tous les aspects de la conception, la mise en œuvre et l'analyse d'algorithmes SLS.

Le reste de ce chapitre donne un aperçu des méthodes SLS largement utilisées pour les problèmes combinatoires discrètes, notamment: les techniques d'amélioration itérative (section 5); les méthodes SLS "simples" (section 6), y compris le recuit simulé et recherche tabou, ainsi que la méthode moins connue de la recherche locale dynamique, les méthodes SLS hybrides (section 7), telles que la recherche locale itérative et GRASP, et des méthodes SLS basées sur la population (Section 8), en particulier, Optimisation de colonie de fourmi et algorithmes évolutionnaires. Le chapitre se termine par une brève discussion de certaines questions d'ordre général et des directions de recherche intéressantes dans le domaine de la recherche locale stochastique.

5. L'amélioration itérative :

L'amélioration itérative est la méthode SLS la plus élémentaires. Elle est basée sur l'idée d'améliorer itérativement une solution candidate du problème posé relativement à une fonction d'évaluation. Plus précisément, la recherche est démarré à partir d'une position (solution) initiale, et à chaque étape de recherche, la solution actuellement candidate notée s

est remplacée par une solution candidate voisine s' avec une valeur de la fonction d'évaluation. [17]

Pour un problème de minimisation, la recherche est terminée lorsque le minimum local est atteint, c'est à dire une solution candidate s avec g(s) <= g(s') pour tous les voisins directs s' de s, où g est la fonction d'évaluation de minimisation. Cette méthode SLS est appelée SLS d'amélioration itérative, Elle est aussi connu sous le nom de descente itératif ou *hill-climbing* (ce dernier est motivé par une formulation équivalente de maximisation de la fonction d'évaluation).

Les algorithmes d'amélioration itérative trouvent un optimum local d'une fonction d'évaluation donnée. Depuis les minimum locaux sont définis par rapport à une relation de voisinage N donnée. Il est bien évident que le choix de N a une importance cruciale pour la performance de tout processus d'amélioration itérative. Alors que l'exploitation d'un voisinage plus large engendre un optimum local de qualité, la complexité temporelle de la détermination de l'amélioration des mesures de recherche augmente avec la taille de voisinage.

5.1. Facteurs de performance :

En pratique, les étapes de recherche avec la complexité quadratique ou cubique de temps conduisent souvent à des temps de calcul prohibitif lorsqu'il s'agit de résoudre des problèmes importants. [17]

La complexité temporelle des étapes de recherche locale peut être considérablement réduite en utilisant deux techniques génériques. Tout d'abord les techniques de mise en cache et de mise à jour incrémentale peuvent être utilisées pour réduire significativement les coûts de calcul, à partir des valeurs de la fonction d'évaluation de tous les voisins de la position de recherche en cours dans chaque étape de recherche. Au lieu de cela, ces valeurs sont stockées et mises à jour à base d'effets réels de chaque étape de recherche (voir également le chapitre 1 de [19]).

Deuxièmement, la taille des voisinages peuvent être considérablement réduite en excluant de l'examen des voisins de la solution actuellement candidate qui sont ou non susceptibles d'amélioration. Il existe des techniques d'élagage de voisinage qui ont un long historique. Elles se confinent dans un rayon fixe de recherche. Ces techniques sont essentielles pour la conception des algorithmes SLS de bonne performance basés sur le voisinage large, mais ils peuvent aussi conduire à une amélioration significative des performances lorsqu'ils sont utilisés en combinaison avec des voisinages plus petits.

Un autre facteur qui a une influence significative sur la vitesse et les performances d'un algorithme d'amélioration itérative est le mécanisme utilisé pour déterminer les étapes de recherche, c'est ce qu'on appelle les règles de pivotement [57]. Les deux règles de pivotement plus utilisées sont la meilleure-amélioration et la première-amélioration itérative.

5.2. La meilleure amélioration itérative :

Cette méthode choisit à chaque étape l'une des positions de recherche voisines qui résulte en une amélioration maximale possible de la valeur de la fonction d'évaluation. Les liens peuvent être répartis de manière aléatoire, fondée sur l'ordre dans lequel le voisinage est examiné ou en utilisant des critères secondaires.

5.3. La première amélioration itérative :

Cette méthode porte sur le voisinage selon certains ordres prédéfinis et effectue la première étape de l'amélioration de recherche rencontrée au cours de cette inspection. De toute évidence, les optimums locaux déterminés par cette méthode dépendent de l'ordre dans lequel le voisinage est balayé. Pour assurer la diversification de la recherche, il peut être utile d'utiliser un ordre aléatoire plutôt que des ordres fixes prédéfinis.
Au lieu d'utiliser les ordres prédéfinis, un ordre aléatoire pour le balayage du voisinage peut être utile, et les exécutions répétées d'algorithmes de première amélioration avec ordre aléatoire sont souvent capables d'identifier de nombreux optimums locaux différents même lorsqu'elles démarrent à partir de la même position initiale [17].

La première amélioration itérative exige souvent plus étapes que la meilleure amélioration itérative afin d'atteindre des optimums locaux de qualité comparable. Mais les étapes d'amélioration individuelles sont généralement rencontrés beaucoup plus rapidement, puisque, dans ces cas, elles n'ont pas besoin d'inspecter tout le voisinage. Toutefois, les algorithmes de meilleure amélioration bénéficient souvent de façon plus significative des stratégies de mise à jour supplémentaires. Par conséquent, elles ne sont pas toujours plus lentes que les algorithmes de première amélioration. Il est également à noter que dans le but d'identifier un candidat en tant que solution optimale locale, les algorithmes de première et de meilleure amélioration nécessitent d'inspecter le voisinage local en entier. [21, 22]

Un moyen intéressant de réaliser un bon compromis entre la taille de voisinage et le temps de la complexité d'exécution des étapes de recherche locale est d'utiliser de multiples relations de voisinage.

- *La méthode VND (Descente à voisinage variable) :*

Bien qu'il ne s'agisse pas d'unique facteur, il apparait aisément que la probabilité de rencontrer rapidement un optimum local est fortement liée à la taille du voisinage. Un algorithme de descente utilisant un voisinage trop étroit risque de retourner rapidement un optimum local de mauvaise qualité, tandis qu'un voisinage trop large apparenterait la recherche à un parcours aléatoire de l'espace.

Lorsque la descente retourne un optimum local, il existe plusieurs manières de s'en extraire. La plupart des techniques perturbent la configuration courante, ce qui a pour effet de détériorer son cout. L'unique moyen d'améliorer la configuration courante est de changer le voisinage. Ce concept fut formalisé en un cadre assez large nommé VNS (*Variable Neighborhood Search*) [Mladenovic and Hansen, 1997 ; Hansen and Mladenovic, 1999]. La descente à voisinage variable (*Variable Neighborhood Descent*) s'inscrit dans ce schéma. [64]

Le concept VNS [23, 24] est une méthode d'amélioration itérative qui bascule systématiquement entre plusieurs relations de voisinage N_1, N_2….N_K, qui sont généralement classées selon leur taille incrémentale.

A partir d'une solution candidate initiale, VND effectue une amélioration itérative avec N1. Une fois l'optimum locale qui concerne N1 est trouvé, la recherche se poursuit avec N2. En général, quand un optimum local de Ni est trouvée, VND bascule vers Ni+1. Toutefois, dès qu'une amélioration est réalisée dans n'importe quel voisinage Ni (avec $i > 1$), la recherche est poursuivie en utilisant Ni+1. L'idée sous-jacente de ce mécanisme est d'utiliser des petits voisinages.

Concernant la méthode VND, les itérations se terminent quand un optimum local de Nk a été trouvé. Les résultats empiriques ont montré que les algorithmes VND sont souvent beaucoup plus efficaces dans la recherche d'un optimum local de qualité que les algorithmes d'amélioration itérative simple avec les grands voisinages. D'autres variantes de l'approche plus générale de recherche à voisinage variable, telles que la VNS basique ou la VNS biaisée, qui sont conceptuellement plus étroitement liée à la recherche locale itérative (voir section 4).

Enfin, un certain nombre d'algorithmes SLS utilisent les voisinages à très grande échelle [17], dont la taille est souvent exponentielle en fonction de la taille de l'instance d'un problème donné. Ce type de voisinage généralement nécessite une heuristique de recherche de profondeur variable (variable-depth search) [25, 26] et les algorithmes de la chaîne d'éjection (Ejection Chain Algorithms) [27]. Toutefois, il existe des voisinages spécialement structurés

qui peuvent être explorés exactement et efficacement en utilisant des techniques de networkflow ou la programmation dynamique [28, 29, 30, 31, 32]. Pour un aperçu de ces techniques, lire le chapitre R-XX de [32].

Les algorithmes d'amélioration itérative s'arrête quand ils atteignent un optimum local. Dans ce qui suit, nous discuterons de plusieurs approches qui permettent aux algorithmes SLS d'échapper à l'optimum local actuel en acceptant parfois une aggravation des mesures de recherche.

6. Méthodes SLS ``simple`` :

Les méthodes décrites dans ce chapitre sont dites "simples" dans le sens où elles sont généralement fondées sur une seule relation de voisinage fixe.

6.1. Algorithme d'amélioration itérative RII :

Les algorithmes RII (Randomised Iterative Improvement) constituent une version les plus simples pour permettre à l'aggravation des mesures de recherche et d'aller parfois à un choix au hasard vers une position voisine de recherche. En RII, cette idée est mise en œuvre par commutation probabiliste entre deux types d'étapes de recherche dans une même structure de voisinage : d'étapes d'amélioration itérative et d'étapes de recherches aléatoires non-informées de mesures, dans lequel un voisin est choisi uniformément au hasard. Plus précisément, à chaque étape de recherche, une étape de marche aléatoire uniforme est effectuée avec une probabilité **wp**, et une démarche d'amélioration itérative est réalisée autrement (c'est à dire avec une probabilité 1 - wp). Le paramètre wp est appelé probabilité de recherche ou un paramètre de bruit. [17]
Grâce à ce mécanisme de longues séquences d'étapes de recherche aléatoire peuvent être effectuées arbitrairement où la probabilité de r étapes consécutives de recherche aléatoire est p_r.

Une extension d'un algorithme d'amélioration itérative dans un algorithme RII exige habituellement que quelques lignes de code et ne possède qu'un seul paramètre. En dépit de leur simplicité conceptuelle, les algorithmes RII peuvent être efficaces. Par exemple, ils ont fourni la base pour GSAT avec la marche aléatoire, un algorithme bien connu pour le problème de satisfiabilité propositionnelle (SAT) [33]. Il existe cependant peu d'algorithmes

RII, peut être parce que d'autres algorithmes SLS plus complexes atteignent souvent de meilleures performances.

6.2. Algorithme probabiliste d'amélioration itérative (PII) :

Les étapes de recherche aléatoire, telle que pratiquée dans RII, permettent de conduire à une détérioration (cas de minimisation) significative de la valeur de la fonction d'évaluation. Une alternative intéressante consiste à fonder l'acceptation d'une aggravation de l'étape de recherche sur le changement de fonction d'évaluation qu'elle a causé, ce qui constitue l'idée clé derrière l'amélioration itérative probabiliste (PII). A chaque étape, l'algorithme PII sélectionne une solution candidate voisine selon une fonction P(g, S), qui détermine une distribution de probabilité sur le voisinage de S en tenant compte de la fonction d'évaluation g. [17]

Dans la pratique, des échantillons de cette distribution de probabilités peuvent être prises comme suit: d'abord, un voisin nommé s0 de la position de recherche actuelle nommé s est sélectionné au hasard, puis une décision est prise si s0 est acceptée comme la nouvelle position de recherche. En minimisant g, cette décision est dans de nombreux cas sur la base des faits suivants, bien connus par : la condition Metropolis:

$$\begin{cases} 1 & \textbf{si } g(s') < g(s) \\ \text{Exp}((g(s) - g(s'))/T) & \textbf{sinon} \end{cases}$$

Où T est un paramètre qui détermine le degré d'aggravation des mesures de recherche qui sont susceptibles d'être acceptées. La condition de Metropolis est fréquemment utilisée dans les algorithmes de recuit simulé (voir ci-dessous) et dans ce contexte le paramètre T est appelé température. [34, 35]

6.3. Recuit simulé (SA) :

Une généralisation naturelle de PII pour la condition de Metropolis est obtenue en modifiant le paramètre T lors de la perquisition, par exemple, en diminuant progressivement T, le processus de recherche devient de plus en plus gourmand. C'est l'idée clé qui sous-tend le recuit simulé (SA), une méthode SLS qui a été initialement inspirée du recuit de processus solides et de manière indépendante proposée dans [36] et [37].

6.4. La recherche Tabou(TS) :

La recherche Tabou est une méta-heuristique d'optimisation présentée par Fred Glover en 1986. On trouve souvent l'appellation recherche avec tabous en français. Cette méthode est une méta-heuristique itérative qualifiée de recherche locale au sens large. [23]

L'idée de la recherche Tabou consiste, à partir d'une position donnée, à explorer le voisinage et à choisir la position dans ce voisinage qui minimise la fonction objectif.

Le risque cependant est qu'à l'étape suivante, on retombe dans le minimum local auquel on vient d'échapper. C'est pourquoi il faut que l'heuristique ait de la mémoire : le mécanisme consiste à interdire (d'où le nom de *tabou*) de revenir sur les dernières positions explorées.
C’est dans ce sens que la méthode Tabou diffère des autres méthodes SLS discutées précédemment. La recherche Tabou rend l'utilisation directe et systématique de la mémoire à diriger le processus de recherche. [23]
Lors de l'exécution des mesures d'amélioration itérative, seuls les voisins admissibles de la solution sont considérés comme des candidats actuels, c'est-à-dire, des positions voisines de recherche qui ne sont pas déclarées comme tabous. La mémoire taboue est mise à jour après chaque étape de recherche. [23]

6.5. Recherche locale dynamique (DLS) :

Une approche différente pour trouver un optimum local constitue la base d'algorithmes de recherche locale dynamique: Plutôt que d'utiliser explicitement l'aggravation des mesures de recherche, DLS modifie la fonction d'évaluation donnée chaque fois qu’une filiale d'algorithme de recherche locale rencontre un optimum local. Plus précisément, DLS fonctionne comme suit :
Initialement, une filiale d'algorithme de recherche locale, typiquement une démarche d'amélioration itérative, est exécutée jusqu'à ce qu’un optimum local s soit rencontré. À ce point, la fonction d'évaluation est modifiée telle que s ne soit plus un optimum local. Puis, la filiale de recherche locale est exécutée à nouveau jusqu'à ce qu'elle atteigne un optimum local de la fonction d'évaluation modifiée. Mais à quel point, les modifications supplémentaires de la fonction d'évaluation sont effectuées. ? Ces phases de la recherche locale suivie par des modifications de la fonction d'évaluation sont répétées jusqu'à ce qu'un critère de terminaison soit satisfait.

Les modifications apportées à la fonction d'évaluation sont généralement obtenues au moyen de poids de pénalité qui sont associés à des composants de la solution individuelle. La forme d'évaluation utilisée par la filiale d'algorithme de recherche locale est donc comme suit [31]:

$$\mathbf{gp(s) := g(s) + \sum_{i \in C(s)} pénalité(i)}$$

Où g(s) est la fonction d'évaluation initiale, C (s) est l'ensemble des composants de la solution de la solution de candidate s, et pénalité(i) est la pénalité du composant i de s. Les pénalités sont initialisées à zéro.

Les algorithmes DLS ont démontré une bonne performance pour résoudre des problèmes combinatoires divers. [65]

7. Les méthodes SLS Hybrides :

7.1. Recherche locale itérative (ILS) :

L'idée clé derrière cette méthode SLS est d'échapper de l'optimum local actuel de la fonction d'évaluation donnée par le moyen d'un mécanisme de perturbation. Essentiellement, trois composants sont au cœur de tout algorithme de recherche locale itératif (ILS) [17].

- Une procédure filiale de recherche locale est utilisée de façon efficace pour trouver l'optimum local, ce qui est généralement basée sur un algorithme d'amélioration itérative ou d'une méthode SLS simple.
- La procédure de perturbation qui introduit une modification à une solution candidate donnée afin de permettre au processus de recherche de s'échapper de l'optimum local.
- Un critère d'acceptation est utilisé pour décider si la recherche doit être nouvellement poursuivie à partir d'un optimum local trouvé.

Sur la base de ces composants, L'algorithme ILS fonctionne comme décrit dans la Figure. III.4.

Recherche locale réitérée (IlS) :

Détermine une solution candidate initiale s

Effectue la recherche filiale locale sur s

Tant que le critère d'arrêt n'est pas satisfait faire:

r := s
Effectue la perturbation sur s
Exécute la filiale de recherche locale sur s
En se basant sur le critère d'acceptation :
Maintenir s ou revenir à s := r

Fig. III.4 Aperçu de la recherche locale itérative. [17]

Lors de l'utilisation d'un algorithme d'amélioration itérative comme une procédure filiale de recherche locale, les ILS peuvent être considérés comme une démarche aléatoire biaisée dans l'espace des optimums locaux atteint par cette procédure. La procédure de perturbation devrait introduire une modification qui ne peut être immédiatement annulée par la procédure de recherche locale, afin de parvenir à une diversification effective de la recherche. Le critère d'acceptation détermine le degré d'intensification de recherche. Par exemple, si seulement l'amélioration des solutions candidates qui sont acceptées, l'ILS exécute la recherche de la première-amélioration aléatoire dans l'espace de l'optimum local. En résumé, la procédure de perturbation et le critère d'acceptation détermine l'équilibre entre l'intensification et de diversification de recherche.
Typiquement, l'attraction principale de l'ILS est le fait qu'elle est en général très facile pour faire des extensions des réalisations existantes d'un algorithme de recherche local simple dans un algorithme ILS [17].

7.2. Algorithmes Itératif Greedy (IG) :

Les méthodes de construction Greedy sont au cœur de nombreux algorithmes d'approximation bien connus. Ils fournissent également la base pour les algorithmes itératifs Greedy, qui peuvent être considérés comme une variante de l'ILS dans lequel la recherche locale et les phases de perturbation sont remplacées par les phases de construction et de destruction. A partir d'une solution candidate complète, IG alterne entre des phases de destruction, durant lesquelles certains composants de la solution sont retirés de la solution candidate actuels s (par exemple, au hasard ou par l'utilisation d'une heuristique, en fonction de leur impact sur la fonction d'évaluation), et les phases de construction, au cours desquelles les composants des solutions sont ajoutés heuristiquement, jusqu'à ce qu'une nouvelle solution complète **s'** candidat ait été obtenue. Comme dans ILS, un critère d'acceptation est utilisé pour décider s'il convient de poursuivre la recherche de **s'** ou **s**.

On note que la solution candidate initiale en IG peut être générée par une méthode de construction qui peut être différente de celle utilisée dans les phases de construction ultérieures. En outre, une procédure de recherche locale perturbatrice peut être utilisée pour améliorer encore les solutions candidates complètes considérées durant la recherche. Dans ce cas, IG peut être considéré comme une variante de l'ILS dans laquelle on combine une phase destruction et une phase de construction des formes de la procédure de perturbation.
Les méthodes IG constituent une approche prometteuse pour résoudre les problèmes pour lesquels les bons algorithmes constructifs existent. [17]

7.3. Procédures de recherche randomisée adaptative Greedy (GRASP):

Une combinaison de la recherche locale constructive et perturbatrice constitue la base pour le GRASP [38, 39], qui fonctionne comme suit: En utilisant une procédure randomisée de construction, une solution candidate complète est générée et qui est ensuite améliorée en utilisant une procédure perturbatrice de recherche locale. Ce processus à deux phases est répété jusqu'à ce qu'un critère de terminaison soit satisfait.
La procédure de construction dans le GRASP ajoute itérativement des composants de la solution qui sont choisis au hasard à partir d'une liste restreinte de candidats. Les éléments de cette liste sont déterminés en utilisant une fonction heuristique, dont la valeur d'une composante donnée peut dépendre de composants de la solution déjà présente dans la solution candidate courante partielle (c'est l'aspect adaptatif du processus de recherche). GRASP a été appliquée aux plusieurs problèmes combinatoires. Pour une description détaillée des différentes extensions de GRASP, voir [39].

8. Les méthodes SLS à base de population :

Diverses méthodes de SLS maintiennent une population de solutions candidates qui sont manipulées simultanément à chaque étape de recherche. Comme indiqué précédemment, ces méthodes à base de population peuvent être formulées dans le cadre de SLS unifié décrit dans la section 1 en considérant les positions de recherche qui consistent en des ensembles de solutions candidates et convenablement défini les relations de voisinage ainsi que l'initialisation et les étapes (valeurs) de la fonction objectif. L'appel aux méthodes SLS basées-population découle en partie du fait que l'utilisation d'une population est un moyen simple pour parvenir à la diversification de recherche. Mais par rapport aux autres méthodes SLS, ceci présente souvent des couts importants et de grands efforts d'implémentation et plus des paramètres qui doivent être réglés afin d'atteindre une performance compétitive.

Les deux méthodes basées sur la population, largement connues, décrites dans l'article suivant sont inspirés par des mécanismes biologiques. Toutefois, il convient de noter qu'il existe de nombreux autres méthodes SLS basées-population, telles que les méthodes fondées sur l'ILS, qui sont inspirées des mécanismes biologiques. [17]

8.1. Modèle d'optimisation de colonie (ACO) :

Dans ACO (Ant colony optimisation), les fourmis (artificielles) sont les méthodes de construction aléatoires qui prennent en compte les traces (artificielles) de phéromone et de l'information de l'heuristique lors de la construction itérative des solutions candidates complètes.

Essentiellement, les traces de phéromone sont modélisées par des valeurs numériques qui sont associées à des composants de la solution. [17]

À chaque étape de la construction, les composants de la solution sont choisis avec une probabilité proportionnelle à leur valeur de phéromone et de l'information heuristique. Par exemple, dans le TSP, le premier problème abordé par des algorithmes ACO [26], une valeur de phéromone est associée à chaque arête du graphe donné, et l'information heuristique est généralement définie comme l'inverse de la longueur du chemin (les arêtes). Dans ce cas, la procédure de construction d'excursion appliquée par chaque fourmi ressemble à une variante probabiliste du plus proche voisin de la construction heuristique.

De nombreux algorithmes ACO appliquent une procédure perturbatrice de recherche locale (comme l'amélioration itérative) à chaque solution candidate complète. Il a été montré que dans le contexte de la résolution de problèmes NP-difficiles, la performance globale d'un algorithme ACO dépend souvent de façon cruciale de cette phase de recherche locale [40, 42].

Ensuite, les valeurs de phéromone sont mises à jour en deux étapes :

- Dans la première étape, où l'évaporation des modèles phéromone, toutes les valeurs à phéromones sont réduites, généralement en les multipliant par un facteur constant.
- Dans la deuxième étape (qui dépose des modèles de phéromones), les valeurs de phéromone de composants de la solution contenue dans une ou plusieurs solutions candidates actuelles sont augmentées. Le montant de la majoration dépend souvent de la qualité des solutions respectives. Les cycles de la construction (et sa filiale de recherche locale) et les phases suivies de mises à jour à phéromones sont répétées jusqu'à ce qu'un critère de terminaison soit satisfait.

Plusieurs variantes ACO ont notamment été proposées, elles partagent toutes les mêmes idées sous-jacentes (voir [40] Pour un aperçu). L'ACO méta-heuristique [43] donne un cadre général pour ces variantes.

8.2. Algorithmes évolutionnaires (AE) :

Les AEs (Evolutionary Algorithms) est l'une des classes les plus éminentes de méthodes SLS à base-population ont été inspirés par les concepts biologiques. Ces algorithmes évolutionnaires commencent par une première série de solutions candidates (qui peuvent être générées de façon aléatoire ou moyen des méthodes de recherche heuristique de construction). Cette population est modifiée itérativement au moyen de trois types d'opérations: la mutation, la recombinaison et de sélection.

Les opérateurs de mutation typique : introduisent de petites perturbations dans les solutions candidates de façon aléatoires. La valeur de perturbation appliquée est généralement contrôlée par un paramètre appelé taux de mutation. [17]

Les opérateurs de recombinaison : génèrent une ou plusieurs nouvelles solutions candidates, souvent appelées : progéniture «offspring en Anglais», en combinant des informations de deux ou plusieurs solutions candidates parents. L'un des types les plus communs de la recombinaison est connu comme le croisement. Il permet d'engendrer la descendance par l'assemblage de solutions candidates partielles des deux parents. [17]

Les opérateurs de sélection : sont utilisés pour déterminer quelles solutions candidates de la population actuelle et de l'ensemble des nouvelles solutions candidates obtenues grâce à la mutation et la recombinaison formeront la population utilisée dans la prochaine itération du processus de recherche. [17]

Ce choix est généralement fondé sur la valeur des solutions candidates en vertu de la fonction donnée d'évaluation (qui dans le cadre des évaluations environnementales, est communément appelé la fonction objectif), telles que de meilleures solutions candidates ont une probabilité plus élevée à survivre le processus de sélection. Les opérateurs de sélection sont également utilisés pour le choix des solutions candidates de la population actuelle de subir de mutation ou de recombinaison.

La performance des AEs dépend de façon cruciale sur le choix des opérateurs de l'évolution. Quant à l'ACO, des améliorations de performances substantielles peuvent souvent être obtenues par une optimisation des solutions en utilisant une méthode perturbatrice de recherche locale, telle que l'amélioration itérative. Une large classe d'algorithmes hybrides de ce genre sont aussi appelés algorithmes mimétiques (MA) [44]. Pour un exposé détaillé sur les

algorithmes mimétiques nous vous renvoyons au chapitre R-XX de l'ouvrage [17]. La recherche d'éparpillement est une méthode SLS qui a reçu une attention considérable ces dernières années, elle peut être considérée comme un cas particulier de l'algorithme mimétique, dans laquelle un type particulier de recombinaison et d'opérations de sélection sont utilisés. [45, 46].

9. Discussion et orientations de recherche:

L'étude des méthodes de recherche locales stochastiques se situe à l'intersection de l'informatique, la recherche opérationnelle, des statistiques et divers domaines d'application. Leur développement réussi exige une connaissance des diverses techniques de SLS et de leurs propriétés. En dépit de la nature générique de beaucoup de méthodes SLS, les problèmes à résoudre sont souvent difficile à mettre en œuvre.

Dans l'ensemble, le comportement de la plupart des algorithmes SLS performants n'est pas bien compris, et bien que certains résultats théoriques existent (en particulier, pour le recuit simulé et algorithmes évolutionnaires). Ils sont souvent obtenus sous des hypothèses spécifiques qui limitent leur intérêt pratique. Pourtant, certaines propriétés théoriques, comme l'exhaustivité approximative empirique sont révélées être utiles pour orienter le développement des algorithmes SLS de bonne performance.

10. Conclusion :

De nombreux défis restent à relever dans le cadre des méthodes SLS pour des problèmes combinatoires complexes, y compris les problèmes multi-objectifs, dynamiques et stochastiques. En outre, il semble exister un potentiel considérable dans le contexte des méthodes SLS pour résoudre les problèmes d'optimisation continue et les problèmes hybrides avec des composants discrets et continus. Globalement, il ne fait aucun doute que, comme l'une des approches les plus polyvalentes et efficaces pour résoudre les problèmes combinatoires difficiles, les méthodes SLS continuent d'attirer l'attention des chercheurs et des praticiens de nombreux établissements universitaires et domaines d'application.

Dans le chapitre suivant, nous aborderons l'adaptation d'une méthode SLS pour le problème d'analyse des fichiers d'audit pour les systèmes de détections d'intrusion.

Chapitre IV : *Un algorithme SLS pour l'analyse d'audit*

1. Introduction :

Dans ce chapitre nous proposons une méthode de détection d'intrusion basée sur l'analyse du fichier d'audit de sécurité. Nous proposons une formalisation du problème d'analyse de fichier d'audit de sécurité qui décrit la définition des scénarios d'attaques et la méthode de recherche de ces scénarios dans le fichier d'audit par une formalisation.

2. Détection d'intrusion par analyse de fichier d'audit de sécurité :

L'objectif de notre travail est de proposer une méthode d'analyse de fichier d'audit basée sur la recherche des scénarios d'attaques (approche par scénarios) dans la trace d'audit, pour détecter les éventuelles intrusions.

2.1 Scénarios d'attaques :

Un scénario d'une attaque est défini par la suite des activités réalisées par un intrus. Chaque activité est utilisée comme une donnée d'audit.

Parmi les avantages de l'utilisation scénarios d'attaque, on peut citer :

- la possibilité de construire les scénarios d'attaque en fonction de ce qui est réellement à craindre pour le système,
- la facilité de modifier des scénarios déjà existants, ou prendre en compte un nouveau scénario,
- les évènements que l'on audite peuvent être restreints à ceux qui apparaissent dans les différents scénarios d'attaques, c.-à-d. les évènements correspondant à une attaque peuvent très bien apparaître en dehors de tout comportement intrusif.

Les scénarios d'attaque peuvent être exprimés par un langage de description qui permet l'expression la plus compacte possible d'un scénario en évitant que la seule manière de faire soit d'énumérer la suite des évènements du scénario. Chaque évènement peut être considéré comme une lettre prise dans un alphabet constitué par l'ensemble des évènements auditables. Le fichier d'audit peut être considéré comme une chaîne de caractères principale et les scénarios d'attaque peuvent être considérés comme des sous-suites qu'il s'agit de localiser dans cette chaîne principale.

2.2 L'analyse de fichiers d'audit de sécurité :

L'importance de l'analyse de l'audit de sécurité [47] est de détecter les éventuelles violations de la politique de sécurité d'un système.

2.3 Temps de traitement nécessaire à la résolution de PASFAS :

Le problème d'analyse de fichier d'audit (PASFAS) est un problème NP-complet [51]. Le tableau IV.1 donne le temps de traitement nécessaire à la résolution PASFAS sur IBM RS6000 3200, comme le montre Tab. IV.1, le temps de traitement nommé t_{PASFAS} augmente d'une manière exponentielle quand le nombre d'attaque N_a augmente, c'est pourquoi il convient d'envisager l'utilisation d'une méthode heuristique telle que les algorithmes de recherches locales stochastiques.

N_a	t_{PASFAS}
1	33.6 μsec.
5	3.12 msec.
10	0.19 sec.
15	8.8 sec.
20	6.16 minutes
24	1 heure 57 minutes
30	155 heures 20 minutes
40	24 années
50	305 siècles et 8 ans

Tab. IV.1 - Temps de traitement nécessaire à l'analyse simplifiée d'un fichier d'audit de sécurité (avec Ne = 50) sur un IBM RS6000 320.

3. Un algorithme SLS pour le problème d'analyse de fichier d'audit:

3.1 Formalisation du problème de l'audit de sécurité :

Dans notre approche qui est inspirée de celle qui est utilisée dans l'implémentation du prototype GASSATA [48], les scénarios d'attaque ne sont plus considérés comme des séquences d'évènements, mais comme des ensembles de couples (E_i, N_i) où E_i représente un type d'évènement et N_i le nombre d'occurrences de ce type d'évènement dans le scénario.

Cette approche est appelée "analyse simplifiée d'un fichier d'audit de sécurité" puisque elle ne prend pas en considération l'ordre des événements.

Le problème de l'analyse simplifiée de fichiers d'audit de sécurité s'exprime comme suit :
Soit N_e le nombre de types d'événement auditables et N_a le nombre de scénarios d'attaque possibles. Chaque scénario d'attaque est exprimé sous la forme d'un ensemble de N_e couples (E_i, N_j) où E_i est le type d'événement $(1 \le i \le N_e)$ et N_j son nombre d'occurrences dans le scénario $(N_j \ge 0)$. On peut regrouper les scénarios d'attaque dans une matrice notée AE de dimension $N_e \times N_a$ appelée matrice Attaque-Evénement, dans laquelle AE_{ij} représente le nombre d'occurrences de l'événement *i* dans le scénario *j* $(AE_{ij} \ge 0)$.

Le fichier d'audit peut être exprimé sous la forme d'une matrice d'entiers de dimension N_e, si N_e types d'événement sont auditables. Nous appelons matrice d'audit cette matrice et nous la notons *O* (pour Observé). Le i^{ème} O_i élément de *O* donne le nombre d'occurrences de l'événement *i* observées durant la session auditée. On associe à chaque attaque un poids R_i proportionnel au risque encouru par le système si cette attaque est réalisée. Ces poids sont regroupés dans une matrice ligne notée *R* de dimension N_a dans laquelle R_i donne le poids de l'attaque *i* $(R_i > 0)$.
Soit *H* (pour Hypothèse) une matrice colonne binaire, de dimension N_a, dans laquelle H_i vaut 1 si on émet l'hypothèse que l'attaque *i* s'est produite tandis qu'il vaut 0 dans le cas contraire (la matrice décrit un sous-ensemble d'attaques).

Notre but est de déterminer le sous-ensemble d'attaques potentiellement présentes dans le fichier d'audit (si N_a attaques sont définies, on peut construire 2^{Na} sous-ensembles).
L'analyse du fichier d'audit de sécurité consiste à déterminer la matrice *H* maximisant le produit matriciel $R \cdot H$, tout en respectant les contraintes $(AE \cdot H)_i \le O_i$, *i* variant de 1 à N_e (voir la figure IV.1).

Le produit matriciel $AE \cdot H$ résulte en une matrice colonne de dimension N_e. L'élément $(AE \cdot H)_i$ $(1 \le i \le N_e)$ de cette matrice correspond à la somme des éléments $AE_{ij}(1 \le j \le N_a)$ de la matrice Attaque-Evénement, *j* prenant les valeurs successives des indices de la matrice *H* pour lesquels $H_j = 1$. Il s'agit par conséquent du minimum des nombres d'occurrences de l'événement *i*, que l'on devrait trouver dans le fichier d'audit, si la

combinaison d'attaques codée dans la matrice *H* avait effectivement eu lieu. $(AE \cdot H)_i$ est un minimum car des événements de type *i* peuvent être générés en dehors de tout scénario d'attaque. L'hypothèse codée dans *H* n'est donc réaliste que si le nombre d'événements de type *i* enregistrés dans la matrice d'audit est supérieur ou égal au nombre de ces événements donné par $(AE \cdot H)_i$.

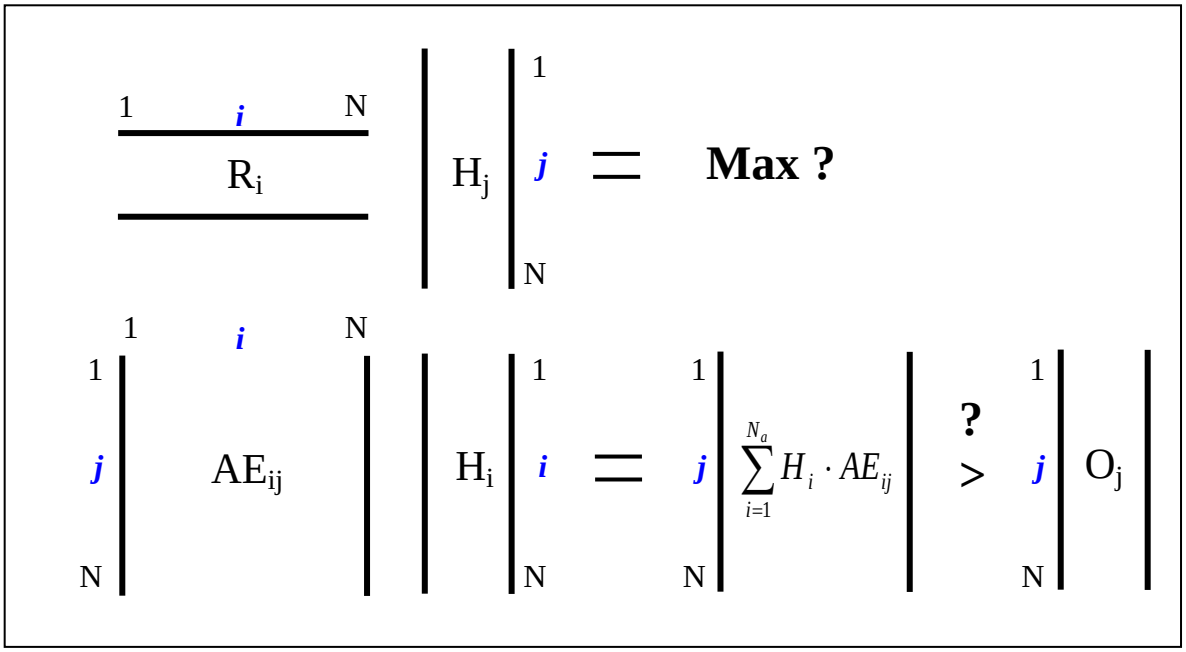


Fig. IV.1 - PASFAS : problème de l'analyse simplifiée de fichiers d'audit de sécurité.

Exemple : on a pris le fichier d'audit généré par le système d'exploitation Unix (type : AIX)

La confidentialité consiste à rendre l'information inintelligible à d'autres personnes que les seuls acteurs de la transaction, Ceci signifie que le système informatique doit :

- empêcher les utilisateurs de lire une information confidentielle (sauf s'ils y sont autorisés).
- empêcher les utilisateurs autorisés à lire une information et de la divulguer à d'autres utilisateurs (sauf autorisation) [01].

3.2 Le codage d'une solution:

Une solution est une chaîne de longueur codant une solution potentielle au problème à résoudre. Or, résoudre PASFAS consiste a déterminé la matrice colonne H maximisant le produit matriciel R.H, tout en respectant la contrainte $(AE.H)_i \leq Oi$, variant de 1 à Ne. La matrice H est une matrice binaire, de dimension Na, dans laquelle H_i vaut 1 si H traduit une hypothèse où l'attaque i s'est produite tandis qu'il vaut 0 dans le cas contraire.

Nous sommes donc dans une situation où le codage des solutions est immédiat puisque la solution du problème à résoudre s'exprime précisément sous la forme d'une suite binaire. Une solution est donc est une suite de Na bits pouvant prendre les valeurs 0 ou 1. Chaque solution correspond à une occurrence particulière de la matrice H. En d'autres termes, le bit de locus i d'une solution vaudra 1 si la solution correspond à une solution dans laquelle l'attaque i est déclarée présente. Dans le cas contraire, ce bit vaudra 0.

Dans ce codage chaque schéma de la forme (. . . 1 . . .), le bit fixé étant de locus i a un sens vis-à-vis du problème à résoudre puisqu'il code la présence de l'attaque i, sans préjuger de celle des autres attaques.

3.3 Fonction objectif:

La fonction objectif consiste à déterminer le sous ensemble d'attaques potentiellement présents dans le fichier d'audit.

Nous avons affaire à un problème de maximisation : trouver le maximum du produit matriciel $R \times H$. Or un algorithme SLS est un algorithme de recherche d'optimum, la fonction à optimiser étant la fonction sélective. Il parait donc facile de conclure que la fonction objectif permettant d'évaluer la valeur sélective d'une solution, est donnée par :

$$F = \sum_{i=1}^{Na} Ri \times Ii$$

Cependant, cette fonction sélective ne tient pas compte du fait que PASFAS est un problème contraint. Pour représenter une solution possible au problème, une solution doit en effet respecter les inégalités :

$$(AE \,.\, I)_i \leq O_i, \text{ avec } 1 \leq i \leq Ne.$$

4. Représentation de la solution de notre approche:

Une solution est une collection d'offres satisfaisant les contraintes du problème et les cas échéants optimisant la fonction objectif, Nous utilisons un vecteur **I** de taille variable (ne

dépassant pas le nombre d'attaques) pour représenter une telle solution, où chaque composante **Ii** reçoit un numéro pour l'attaque détectée.

La solution initiale de la recherche locale que nous proposons est générée aléatoirement qui fonctionne comme suit :

Nous générons une suite **r** de n nombres binaires aléatoirement où **n** est le nombre d'attaques soumises. Pour chaque attaque **j** une valeur de clef $\mathbf{r_j}$ lui est associée et qui représente sa détection par notre système.

Pour produire une solution admissible :

Produire l'ensemble des solutions voisines (vecteurs binaires de longueur **Na**), en appliquant les modifications élémentaires sur la solution courante, puis on génère aléatoirement une valeur **v** entre 0 et 1.

On a deux cas pour sélectionner la solution admissible :

- le premier cas c'est la condition suivante est vérifiée : **v < wp,** nous choisissons de façon au hasard un de vecteurs voisins.
- le deuxième cas c'est la condition suivante, n'est pas vérifiée : **v < wp,** nous choisissons un de vecteurs voisins en appliquant la fonction objectif (le meilleur voisin).

À la fin de ce processus, la solution finale (la solution optimum) est la dernière meilleure solution trouvée pendant les itérations.

4.1 L'Algorithme SLS pour IDS :

Entrée : Matrice : O, AE, max_itr, wp

Sortie : Solution optimale S*

Début :

Générer une solution initiale **S0** selon les contraintes.

Pour : i=1 à **max_itr**

Faire :
r <---- nombre aléatoire entre 0 et 1

Si (r<wp)

Alors

S' =Une solution choisie aléatoirement.
Sinon

S'= Une solution meilleur est choisi.
Fsi.
S*=S'.
Fait.

Retourner la meilleure solution trouvée S*

Fin.

4.2 Organigramme de l'algorithme proposé :

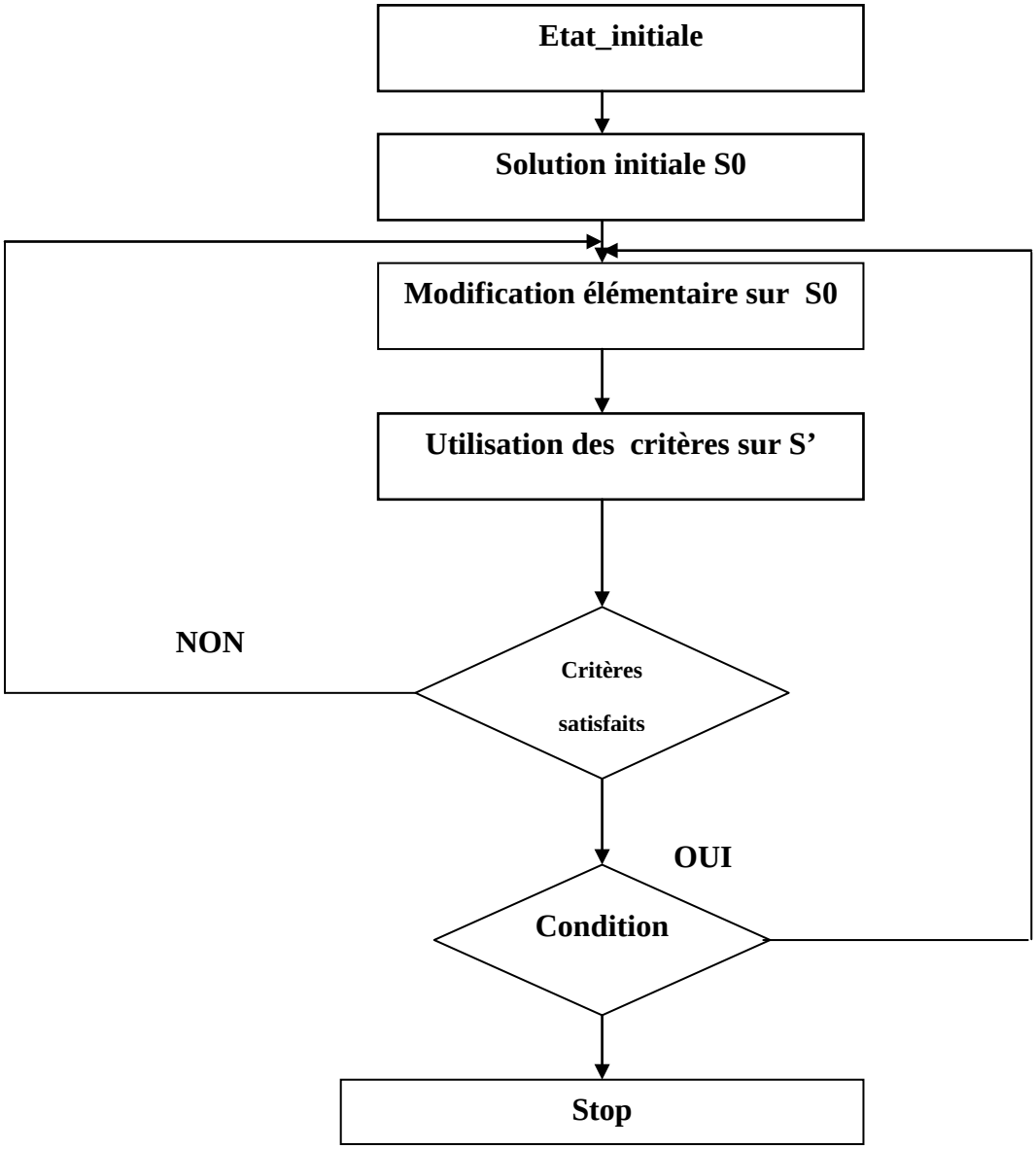


Fig. IV.2 Organigramme de l'algorithme proposé.

Critères satisfaits : système figé ou aucun changement sur la solution **S'** ou le nombre d'itérations est atteint.

Condition: Nombre itération est atteint.

5. Conclusion :

Nous avons abordé dans ce chapitre les différents points essentiels de notre approche proposée, on a commencé par la détection d'intrusion par analyse de fichier d'audit de sécurité, puis nous avons proposé un algorithme SLS pour le problème d'analyse du fichier d'audit. Nous avons détaillé la méthode SLS proposée pour résoudre le problème d'analyse de fichiers d'audit pour les IDS, en formalisant le problème de l'audit de sécurité, le codage des solutions, et la fonction objectif...etc.

Dans le chapitre suivant nous allons voir l'implémentation de la méthode SLS proposée.

Chapitre V : *Implémentation et résultats*

1. Introduction :

Dans ce chapitre nous allons donner une description de l'implémentation de l'algorithme de recherche locale stochastique décrit dans le chapitre précèdent, et les résultats expérimentaux obtenus après simulations de comportements intrusifs.

Dans un premier temps nous présentons l'architecture globale du système de détection d'intrusions envisagé, et les grandes lignes de l'algorithme de recherche locale stochastique utilisé pour l'analyse de fichier d'audit. Ensuite nous donnons une description de 24 scénarios d'attaques sous le système AIX (V5.4, type de système UNIX) et ces traductions en termes d'événements auditables, et nous nous basons sur ces derniers pour construire la matrice Attaques/Evénements pour les expérimentations.

Pour représenter le fichier d'audit obtenu par la simulation du comportement d'un utilisateur expérimenté, nous définissons une matrice d'événements observés O qui contient tous les événements exécutés par cet utilisateur pendant une période de 30 minutes, et à partir de cette dernière on choisit des comportements intrusifs de la matrice d'attaque. Puis nous appliquons l'algorithme de recherche locale stochastique, et pour chaque exécution nous prenons les valeurs d'évaluation de la qualité d'analyse (Taux de détection, nombre de faux négatif, nombre de faux positif...).

2. Architecture globale de système :

2.1 Scénarios d'attaques :

L'approche de détection adoptée est l'approche par scénarios qui consiste à rechercher les signatures d'attaques dans le fichier d'audit de sécurité. Le choix de cette approche est justifié par le fait que l'approche par scénarios théoriquement ne génère pas de faux positifs, contrairement à l'approche comportementale, et qu'il est difficile de définir un comportement normal d'un utilisateur, alors définir ce qui est anormal est plus facile que de définir ce qui est normal.

La méthode d'analyse formalisée dans le chapitre précèdent peut utiliser comme source de données brutes les fichiers d'audit de système, les logs applicatifs, ou même les paquets de réseau. Ces données brutes sont ensuite filtrer par un capteur qui va les traduire à des événements, et à partir de ces événements sera construit un vecteur d'entiers qui

contient le compte des événements observés pendant une période classé par type d'événement.

Le processus d'analyse se fait périodiquement, et à chaque fois qu'une ou plusieurs attaques est détectée, une alerte est envoyée à l'administrateur pour l'informer sur l'attaque et lui donner les informations nécessaires pour qu'il prenne une contre mesure.

2.2 Architecture et description :

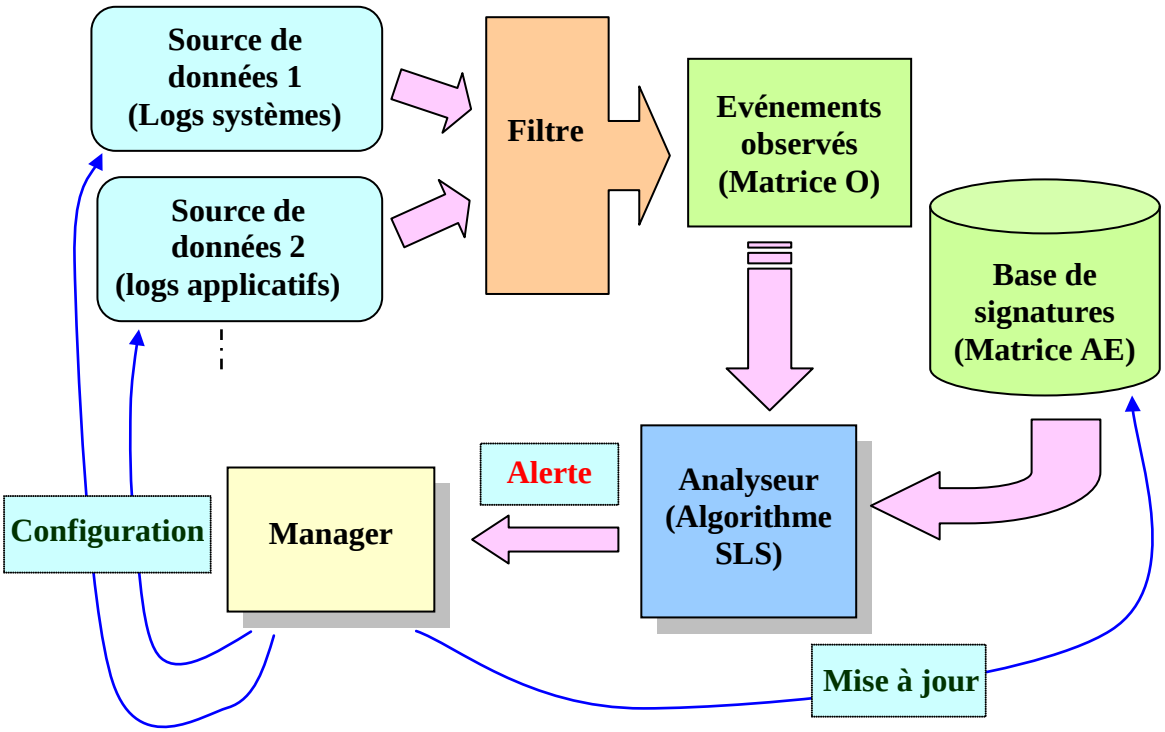


Fig. V.1: L'architecture globale du système de détection d'intrusions proposée.

La source de données : dispositif générant de l'information de traces sur les activités des entités du système (par exemple : système d'audit d'un système d'exploitation, sniffer réseau, application . . . etc).

Filtre de données brutes : la partie qui génère les événements en filtrant et formatant les données brutes intéressantes provenant d'une source d'information (logs du système, logs applicatifs, ou paquets du réseau). Le résultat de ce filtre est une matrice d'entiers qui contient le nombre d'occurrence de chaque type d'événement dans le fichier d'audit.

La base des signatures : contient les signatures des différents scénarios d'attaques. C'est une matrice appelée matrice attaques/évènement AE qui donne pour chaque attaque les événements qu'elle génère. $AE_{ij} >= 0$ est le nombre d'événements de type i générés par l'attaque j. La mise à jour de la base de signatures est assurée par l'administrateur de système.

L'analyseur : L'outil qui assure la détection, c'est un algorithme de recherche locale stochastique qui a comme entrée la matrice des événements observés O et la matrice des scénarios d'attaques AE, et génère des alertes lorsqu'il détecte une intrusion.

Manager : composant de système de détection permettant à l'opérateur de gérer les autres composants. Ses fonctions comportent généralement la configuration des capteurs et analyseurs, la notification des alertes à l'opérateur et éventuellement la réaction.

2.3 L'algorithme de recherche locale stochastique :

L'approche qu'on a proposée pour le problème d'analyse de fichier d'audit est basée sur la méthode de recherche locale stochastique.
L'algorithme SLS proposé, inclut trois phases nécessaires qui sont :

- La génération de la solution initiale.
- La recherche de voisinage et évaluation.
- Sélection de la solution optimale.

Entrée : Matrice : O, AE, max_itr, wp
Sortie : Solution optimale S*

Début :

// *La génération de la solution initiale.*

Générer une solution initiale I0 selon les contraintes.
Initialiser wp; teq : 0<= wp <=1

Solution courante IC :=I0
S*:=IC

// La recherche de voisinage et évaluation.

Pour : i=1 à **max_itr**

Faire :

r <---- nombre aléatoire entre 0 et 1

Si (r<wp)

Alors

IC =Une solution choisie aléatoirement.
Sinon
Générer le voisinage
IV= Sélection meilleure solution
Si F(IV) >F(IC)

Alors

IC := IV

Fsi

Fsi

S*=IC.
Fait.

// *Sélection de la solution optimale.*

Retourner la meilleure solution trouvée S*

Fin.

Fig. V.3 L'algorithme SLS proposé

2.4 Diagramme de classes :

Algorithme SLS :

Permet de définir l'ensemble des paramètres de l'algorithme SLS, et permet de démarrer la résolution du problème d'analyse de fichier d'audit de sécurité par le déroulement de l'algorithme SLS durant plusieurs itérations (nombre précisé comme entré).

Population :

La population contient un ensemble d'individus représentés par leurs génomes, à ce niveau on calcule le passage d'une génération à une autre et l'initialisation de la population.

Individu (ou chromosome) :

Un individu est un ensemble de gènes, il est associé à un vecteur H (hypothèse) qui représente une solution de problème d'analyse de fichier d'audit. C'est au niveau du l'individu que s'effectuent le croisement, le calcul d'une solution voisin.

Gène :

Un gène est associé à un entier de vecteur H (hypothèse) qui peut prendre une valeur égale à 0 ou 1 (Codage binaire), son longueur est égale au nombre d'attaques défini dans la matrice AE.

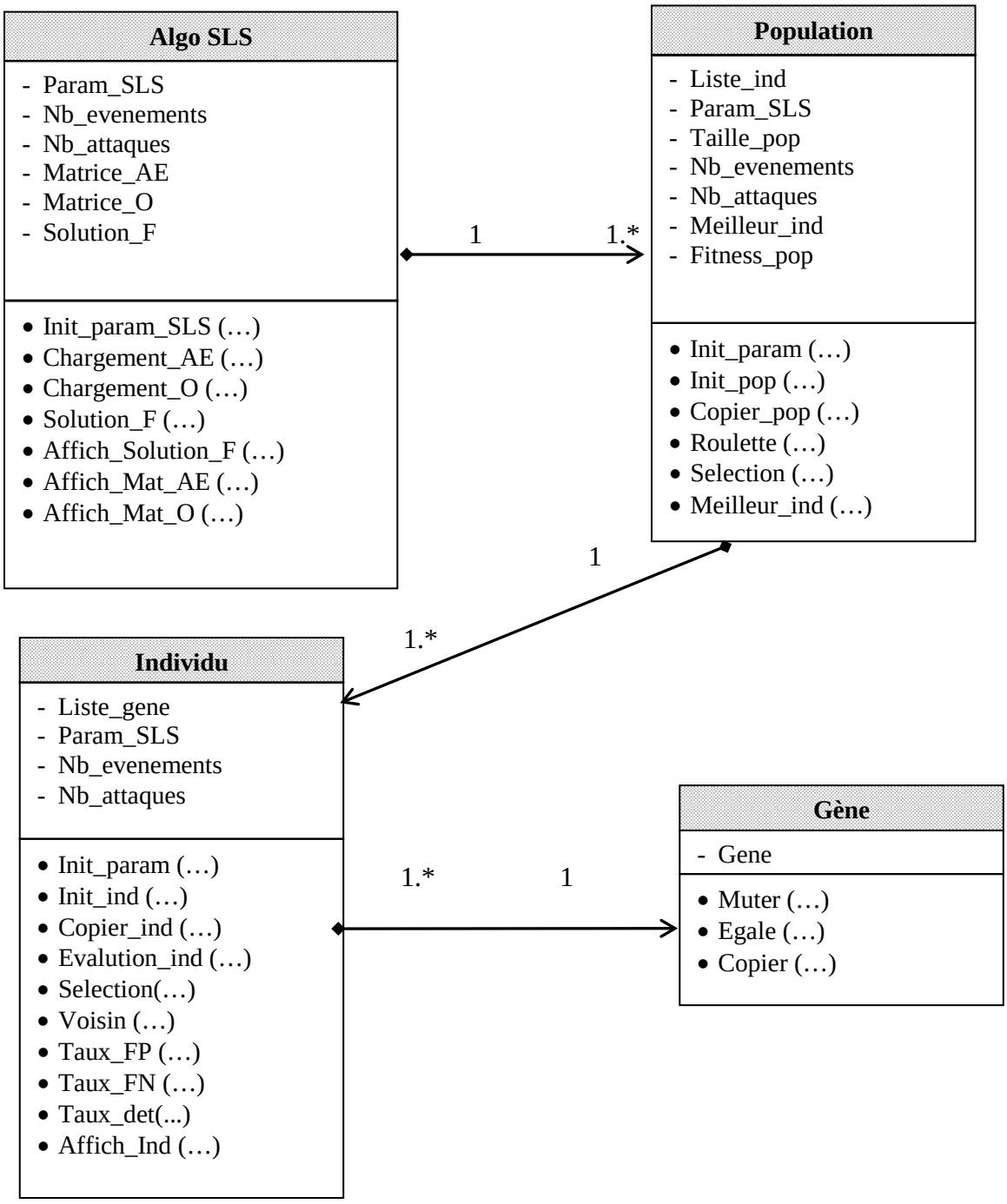


Fig. V.4 – Le diagramme de classes pour l'implémentation de L'algorithme SLS proposé.

3. Expérimentations et évaluation :

Dans cette section nous allons présenter les expérimentations trouvées par l'application de l'algorithme SLS proposé. En utilisant la matrice Attaques/Evénements et la matrice d'audit observée qui ont été utilisées pour les tests de simulations décrits dans [49].

3.1 La base de signature:

L'objet de nos expérimentations est le fichier d'audit généré par le système UNIX, un enregistrement d'audit peut contenir les informations suivantes :

- Le type de l'événement.
- Le résultat de l'événement (échec ou succès).
- Le nom de l'utilisateur (le compte duquel s'exécute le processus qui a généré l'événement).
- L'horodatage de l'événement.
- La commande qui a généré l'événement.
- Le numéro de processus qui a généré l'événement.
- L'information spécifique au type d'événement (par exemple pour une ouverture de fichier, le nom de ce fichier).

Les événements à auditer sont ceux qui apparaissent dans les différentes attaques auxquelles on s'intéresse. Ces événements peuvent être regroupés en 5 classes :

Classes d'événements :	**Exemple d'événements :**
Classe connexion	user_login, short_session.
Classe super-utilisateur	user_su.
Classe commandes	who, w, finger, ps, f, last, groups,…
Classe processus	proc_execute, proc_setpri (modification de niveau de priorité).
Classe fichiers	file_read, file_write, file_unlink, file_mode

Tab. V.1 - Classement d'événements à auditer

La base de signatures utilisée dans les expérimentations est sous forme d'une matrice attaques/événements (de taille 28 X 24) qui a été construite à partir de 24 scénarios d'attaques ou d'actions malveillantes réalistes que l'on peut redouter lorsque l'on travaille

sous le système AIX.

Chaque attaque de a_1 jusqu'à a_{24} est exprimée sous forme d'ensemble de couple (E_i, N_i) où :

- E_i représente un type d'événement.
- N_i le nombre d'occurrence de ce type d'événement (voir le tableau Tab.V.1).

Les attaques de la matrice attaques/événements sont décrites dans l'annexe, où pour chaque attaque il y a une brève description de son scénario, et les trames d'audit générées. Ces attaques peuvent être classées comme suit :

- Intrusion : a_1, a_2, a_3.
- Sessions inhabituelles : a_4, a_5.
- Furetage : a_6, a_7, a_8.
- Recherche d'information sur le système : a_9, a_{10}, a_{11}, a_{12}.
- Vol de donnés : a_{13}, a_{14}, a_{15}, a_{16}.
- Sabotage logique : a_{17}, a_{18}, a_{19}.
- Exploitation du mécanisme de confiance : a_{20}.
- Divers : a_{21}, a_{22}, a_{23}, a_{24}.

3.2 Source de données :

Pour obtenir les matrices d'audit observées utilisées dans les expérimentations qui ont été construites à partir des fichiers d'audit, nous simulons le comportement d'un utilisateur sur une période de 30 minutes. Avec cette simulation nous savons exactement quelles activités ont été réalisées et nous incluons nous même les attaques, ce qui nous facilite d'interpréter les résultats obtenus.

Le tableau Tab.V.2 décrit la matrice d'audit observé que nous venons d'utiliser, cette dernière représente le comportement d'un utilisateur expérimenté du système UNIX sur une période de l'ordre de 30 minutes.

Les événements auditables de la matrice d'audit observé sont ceux définis dans l'ensemble des attaques de la matrice d'attaques/événements donnée en tableau Tab.V.1.

Evénement		Attaque																							
		a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	A_{14}	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}	a_{21}	a_{22}	a_{23}	a_{24}
e_1	User_Login fail	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_2	User_Login (23h à 6h)	..	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_3	Short_Session	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_4	User_SU OK	.	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_5	User_SU fail	.	.	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_6	Who, w, finger, f, ps...	.	3	.	.	.	.	.	.	8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_7	more, pg, cat, vi, ...	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	5	.	.
e_8	ls OK	.	.	.	.	.	30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_9	ls fail	.	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_{10}	df, hostname, uname	.	.	.	.	.	.	.	.	.	3	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_{11}	arp, netstat, ping	.	.	.	.	.	.	.	.	.	.	.2	.	.	.	.	.	.	.	.	.	.	.	.	.
e_{12}	ypcat	.	.	.	.	.	.	.	.	.	.	.	3	.	.	.	.	.	.	.	.	.	.	.	.
e_{13}	lpr	.	.	.	.	.	.	.	.	.	.	.	.	.	.	10	1	.	.	.	.	.	.	.	.
e_{14}	rm,mv	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.
e_{15}	ln	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.1	.	.	.	.	.	.	.	.
e_{16}	whoami, id	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	4
e_{17}	rexec, rlogin, rsh, ...	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	
e_{18}	Proc_Execute	.	3	.	.	.	35	5	.	8	3	2	3	.	.	10	3	.	300	.	2	.	5	.	4
e_{19}	Proc_SetPri	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	100	.	.	.	.	.
e_{20}	File_Open fail	.	.	.	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
e_{21}	File_Open fail cp	.	.	.	.	.	.	.	.	.	.	.	.	.	.10	.	.	.	.	.	.	.	.	.	.
e_{22}	File_Open .netrc	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.
e_{23}	File_Read lpr	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.10	.	.	.	.	.	.	.	.	.
e_{24}	File_Read passwd, ...	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	5	.	.
e_{25}	File_Write passwd, ... fail	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.1	.
e_{26}	File_write cp OK	.	.	.	.	.	.	.	.	.	.	.	.	30	.	.	.	.	.	.	.	.	.	.	.
e_{27}	File_Unlink rm	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.50	.	.	.	.	.	.	.
e_{28}	File_Mode	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	3	.	..	.

Tab.2. – Matrice Attaques/Evénements utilisée dans les expérimentations.

Evénement		Nombre d'événements
$\mathbf{e_1}$	User_Login fail	.
$\mathbf{e_2}$	User_Login (23h à 6h)	.
$\mathbf{e_3}$	Short_Session	.
$\mathbf{e_4}$	User_SU OK	.
$\mathbf{e_5}$	User_SU fail	.
$\mathbf{e_6}$	Who, w, finger, f, ps...	**4**
$\mathbf{e_7}$	more, pg, cat, vi, ...	**3**
$\mathbf{e_8}$	ls OK	.
$\mathbf{e_9}$	ls fail	.
$\mathbf{e_{10}}$	df, hostname, uname	.
$\mathbf{e_{11}}$	arp, netstat, ping	.
$\mathbf{e_{12}}$	Ypcat	.
$\mathbf{e_{13}}$	Lpr	**1**
$\mathbf{e_{14}}$	rm, mv	.
$\mathbf{e_{15}}$	Ln	.
$\mathbf{e_{16}}$	whoami, id	.
$\mathbf{e_{17}}$	rexec, rlogin, rsh, ...	**2**
$\mathbf{e_{18}}$	Proc_Execute	**17**
$\mathbf{e_{19}}$	Proc_SetPri	.
$\mathbf{e_{20}}$	File_Open fail	**4**
$\mathbf{e_{21}}$	File_Open fail cp	.
$\mathbf{e_{22}}$	File_Open .netrc	.
$\mathbf{e_{23}}$	File_Read lpr	.
$\mathbf{e_{24}}$	File_Read passwd, ...	.
$\mathbf{e_{25}}$	File_Write passwd, ... fail	.
$\mathbf{e_{26}}$	File_write cp OK	.
$\mathbf{e_{27}}$	File_Unlink rm	.
$\mathbf{e_{28}}$	File_Mode	**1**

Tab. V.3 – Matrice d'audit observée représente le comportement d'un utilisateur expérimenté sur une période de 30 minutes

3.3 Résultats :

Après que nous ayons défini la matrice attaques/événements et la matrice d'audit observée, nous allons analyser cette dernière en utilisant l'algorithme défini dans la figure Fig.V.3. A chaque expérimentation nous allons introduire des comportements intrusifs dans la matrice d'audit observée avec un nombre variable.

Pour l'implémentation de l'algorithme SLS nous avons utilisé la fonction sélective

suivante : $F(I) = 200 + \left(\sum_{i=1}^{N_a} R_i \cdot I_i - 2 \cdot T_e^2 \right)$. Et nous utilisé une matrice R unité (c-à-d $R_i = 1$ pour i = 1 à N_a), pour ne pas privilégier une attaque par rapport aux autres, et aussi pour faciliter l'interprétation des résultats.

Toutes les expériences ont été exécutées sur un ordinateur de CPU Intel Genuine 1.86 Ghz avec 1.87 GO de RAM.

Pour caractériser les performances de l'algorithme proposé, nous définissons les métriques suivantes :

- Taux de présence T_P : est la proportion de bits égaux à 1 pour les locus correspondant à une attaque présente dans la matrice d'audit observée,
- Taux d'absence T_A : est la proportion de bits égaux à 1 pour les locus correspondant à une attaque absente de la matrice d'audit observée.

En effet, on s'intéresse au T_{Pf} et T_{Af} ou les taux T_P et T_A à la génération finale. Les taux $T_{P0} \approx 0{,}5$ et $T_{A0} \approx 0{,}5$, puisqu'il y a un tirage aléatoire de la première génération. En cas de convergence parfaite vers la bonne solution, le meilleur individu de la population finale possède des bits égaux à 1 au locus correspondant à une attaque présente, et des bits égaux à 0 au autres locus, on a donc $T_{Pf} = 1$ et $T_{Af} = 0$.

- Nombre de faux positifs : Nombre de gènes égaux à 1 pour les locus correspondant à une attaque absente dans la matrice d'audit observée.
- Nombre de faux négatifs : Nombre de gènes égaux à 0 pour les locus correspondant à une attaque présente de la matrice d'audit observée.

Pour mesurer la performance de notre algorithme, nous observons l'évolution de la fonction sélective, T_P et T_A en fonction du nombre de génération et en fonction du nombre d'attaques présentes dans la matrice d'audit. Pour ce faire, nous avons fait varier le nombre d'attaques de 2, 4, 8, 10, 12, 16, 18, et 20. Nous avons tracé les courbes qui présentent les évolutions moyennes de la fonction objectif : minimales, maximales et moyennes, et ce pour un nombre (Nb_attaques) variant d'attaques présentes dans la matrice d'audit O (voir les figures Fig. V.5 à Fig. V.15).

Les figures Fig. V.6, V.7 donnent des exemples d'évolution des valeurs minimales, maximales et moyennes du Faux et négatif en fonction de nombre d'attaques présentes.

A travers des courbes des évolutions de la fonction sélective, des faux positif et négatif, on peut conclure que dans le cas d'une matrice AE contenant 24 attaques notre algorithme permet une excellente discrimination de nombre de fausse attaques comme le montre

Fig.V.06, Fig. V.07 et aussi une discrimination de nombre de non détection d'attaques présentes réellement dans la matrice d'audit analysée O comme le montre la figurre V.3.

Dans ci-après les histogrammes qui présentent les évolutions moyennes (sur 10 exécutions) de la fonction objectif : minimale, maximale et moyenne en fonction du nombre d'itérations.

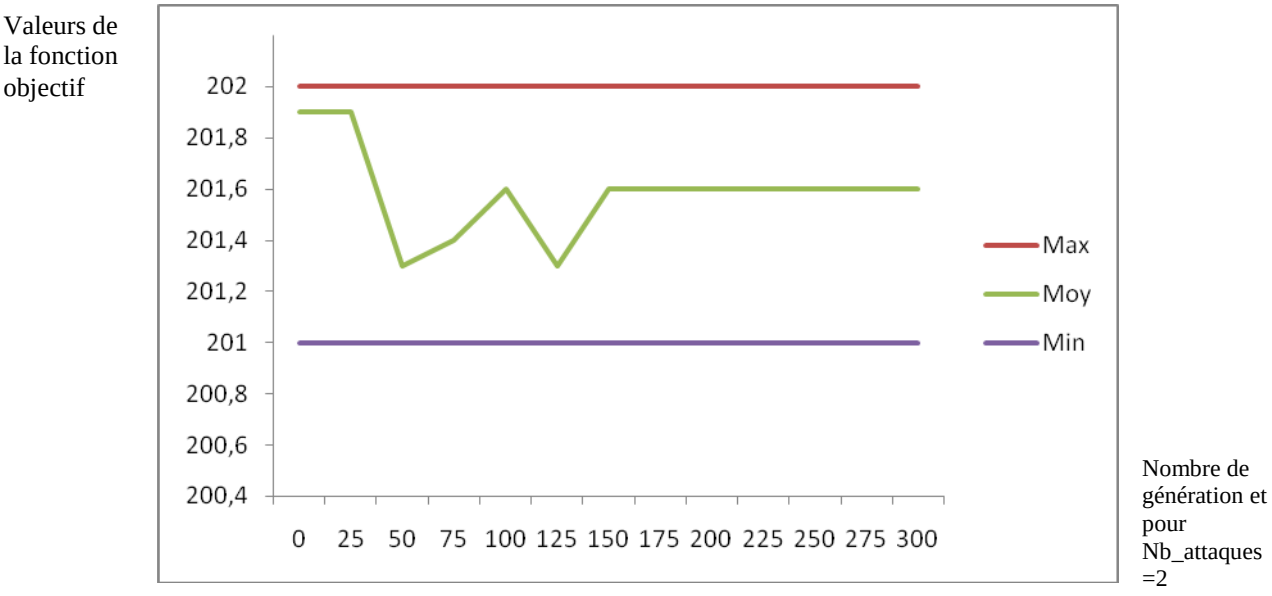


Fig. V.5 - Exemple d'évolution de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =2

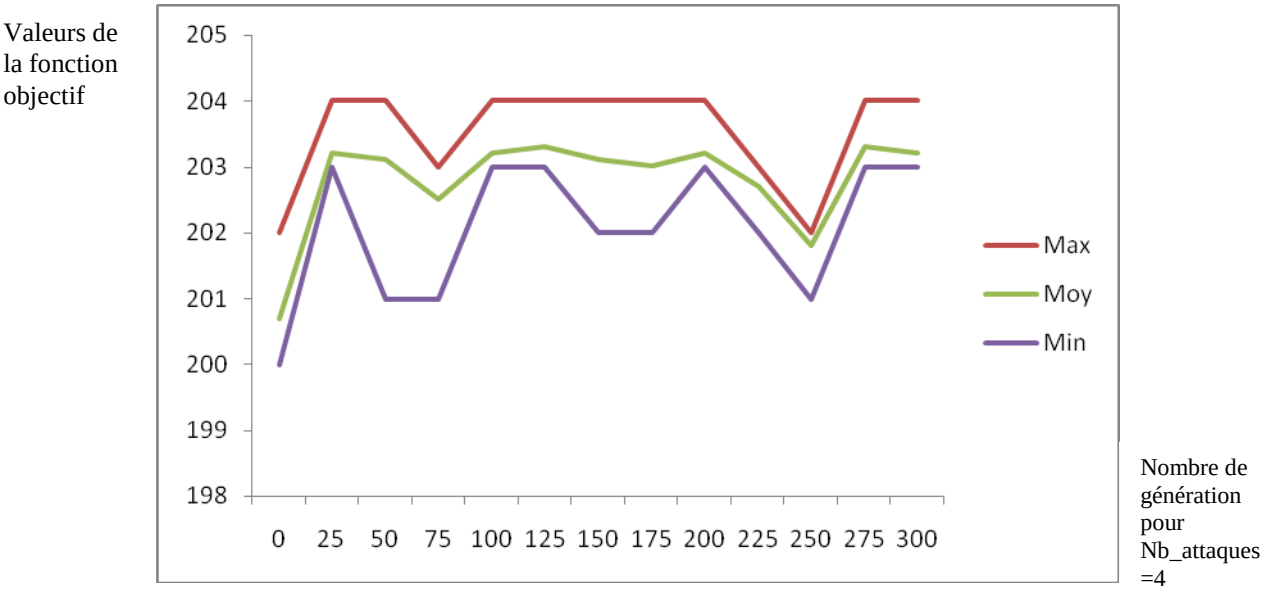


Fig. V.6 - Exemple d'évolution de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =4

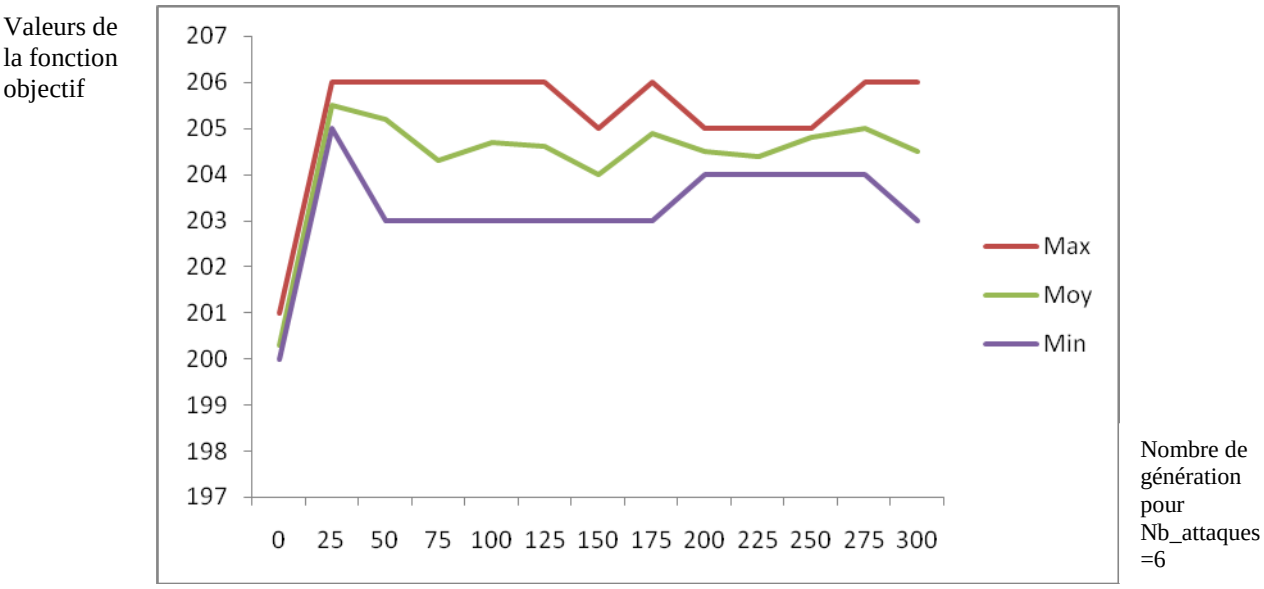


Fig. V.7 - Exemple d'évolution de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =6

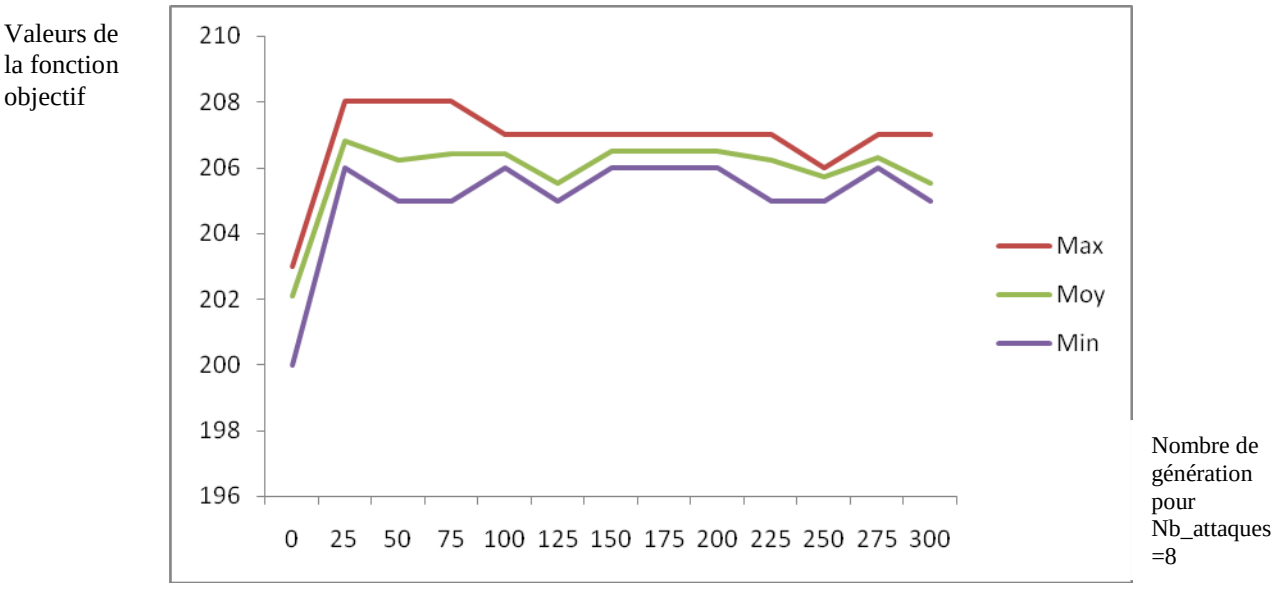


Fig. V.8 - Exemple d'évolution de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =8

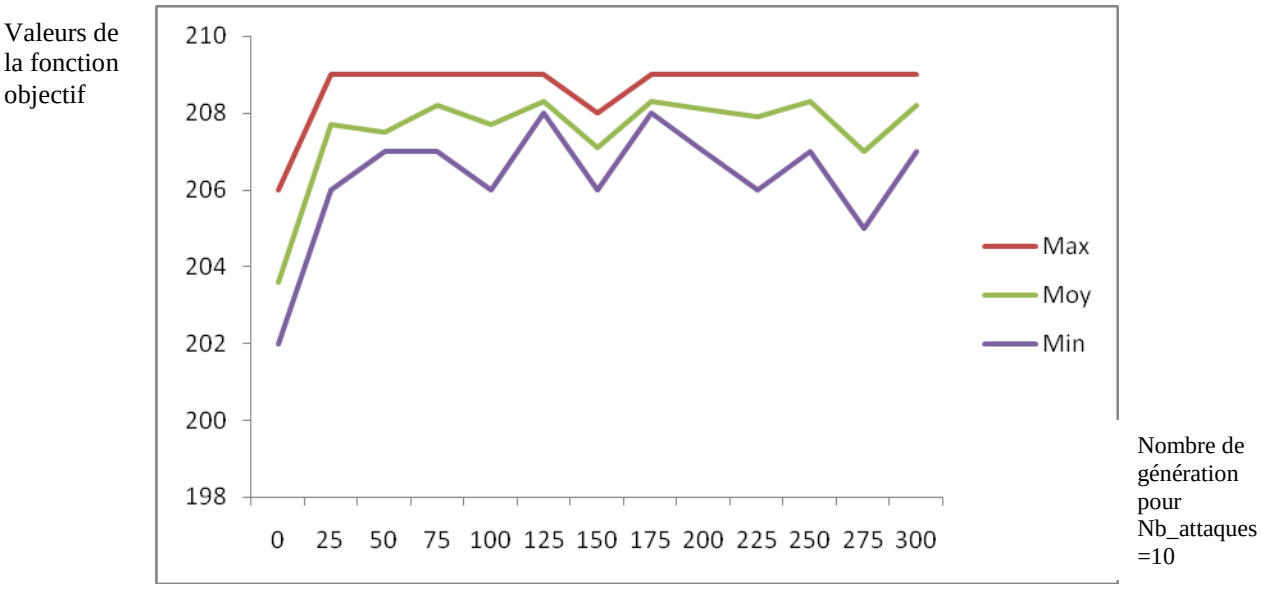


Fig. V.9 - Exemple d'évolution de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =10.

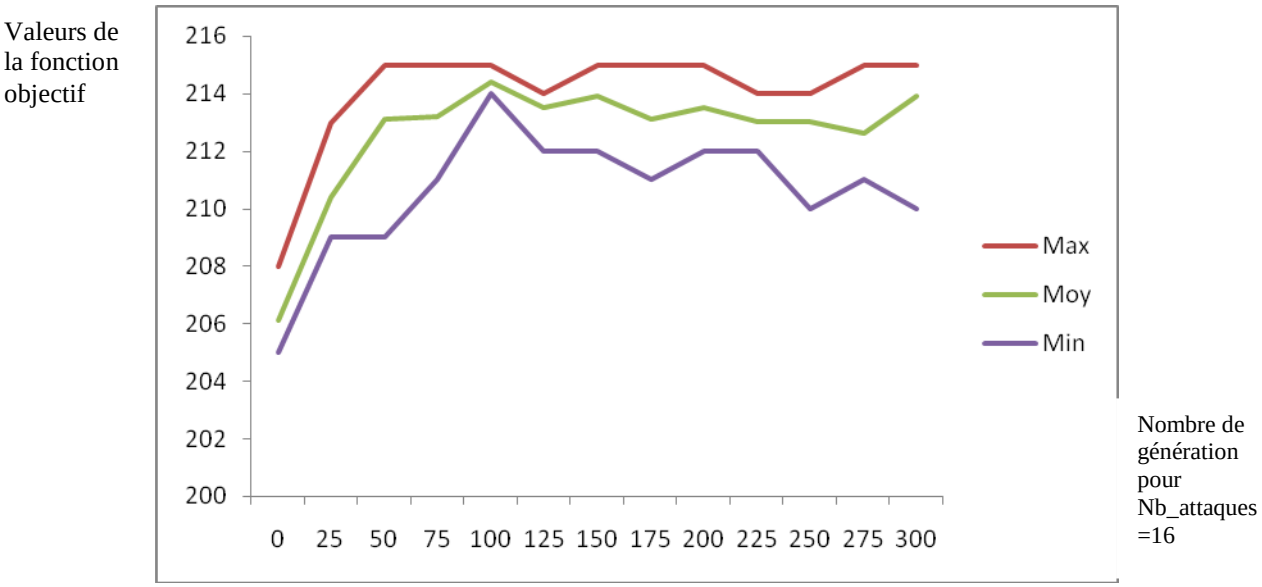


Fig. V.10 - Exemple d'évolution de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =16

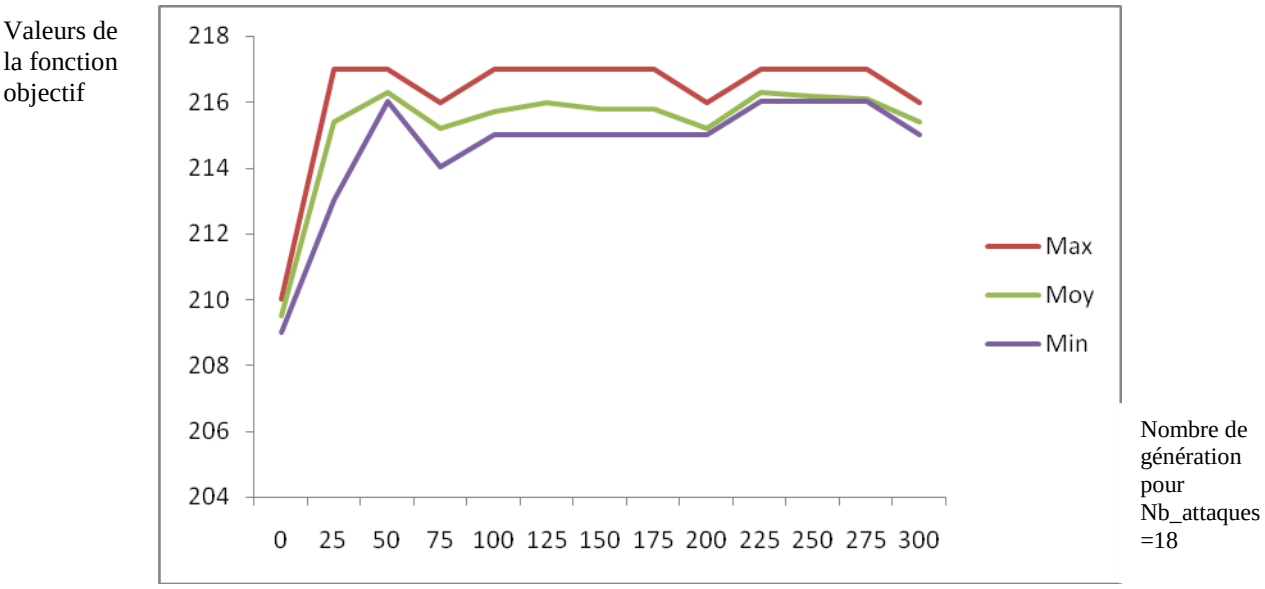


Fig. V.11 - Exemple d'évolution de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =18

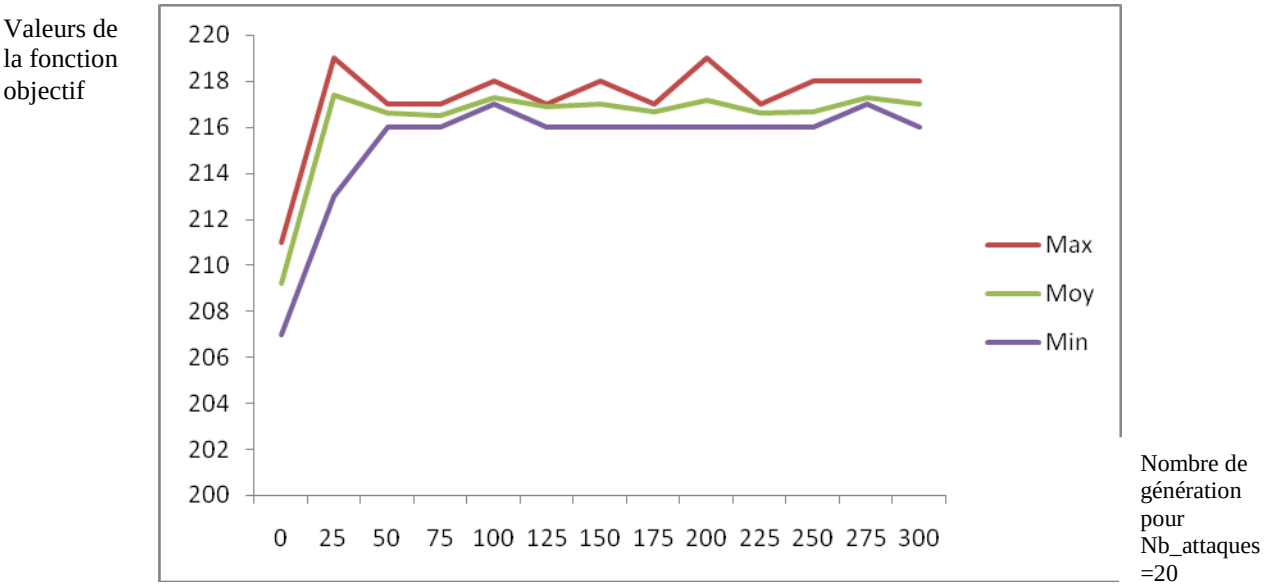


Fig. V.12 - Exemple d'évolution de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =20

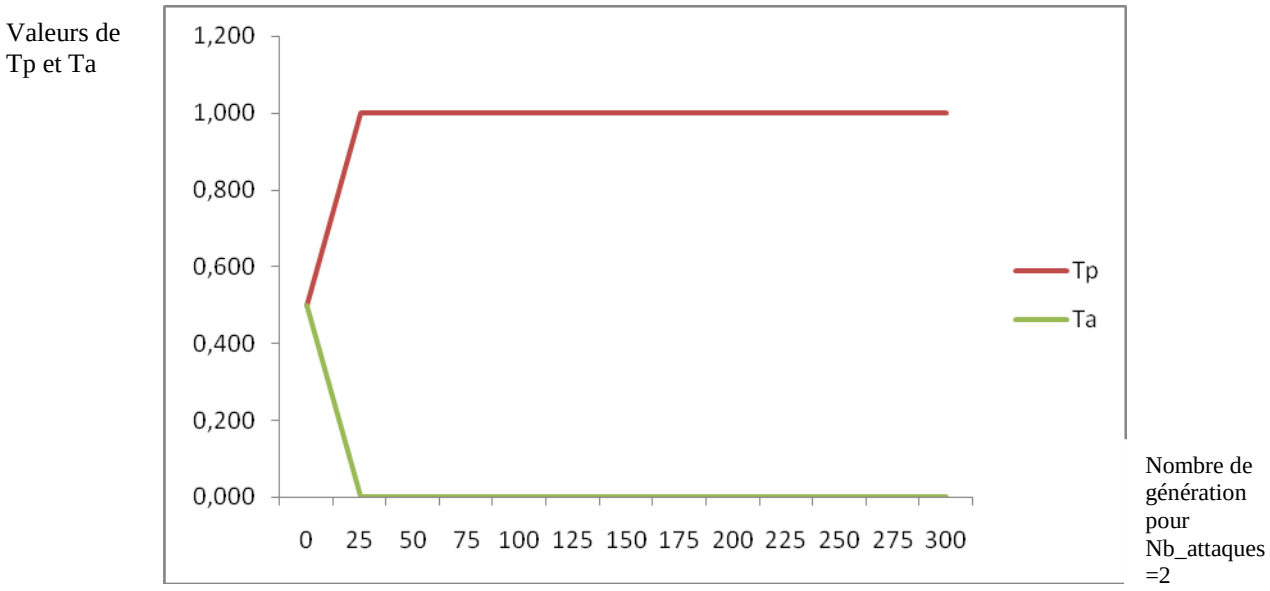


Fig. V.13 - Evolution de taux Tp et Ta en fonction de la génération pour une seule exécution, pour le cas où Nb_attaques = 2.

Tp : taux de présence.

Ta : taux d'bsence.

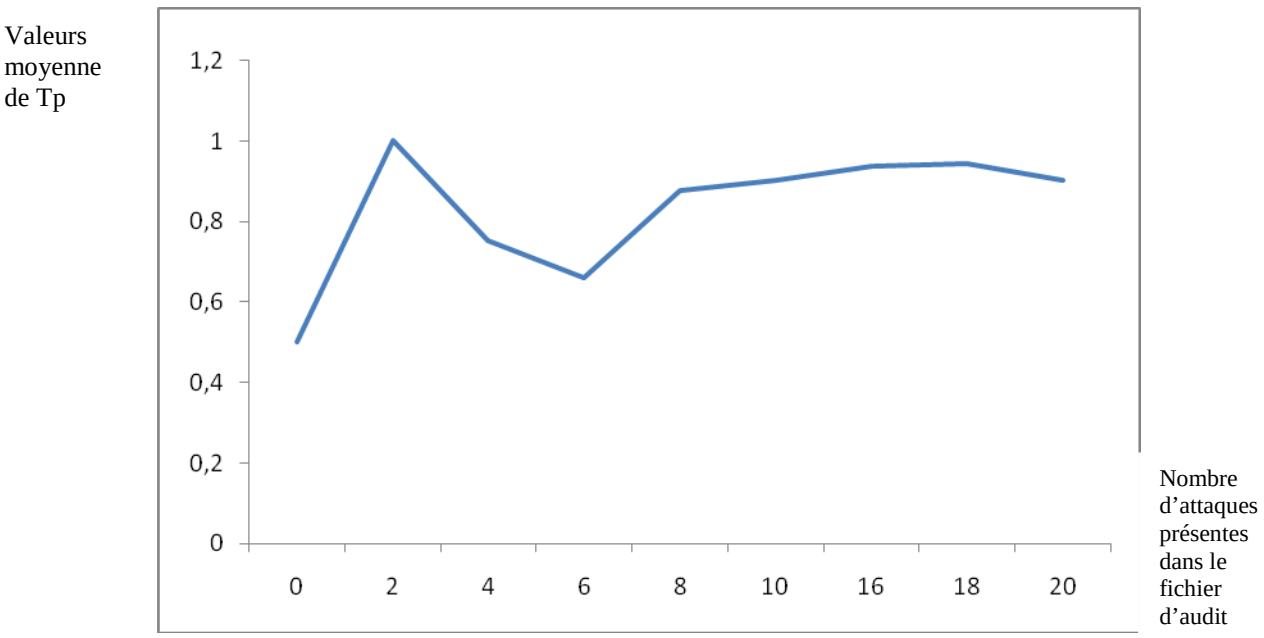


Fig. V.14 - Evolution moyenne (sur 10 exécutions) de taux Tp en fonction du nombre d'attaques présentes dans le fichier d'audit.

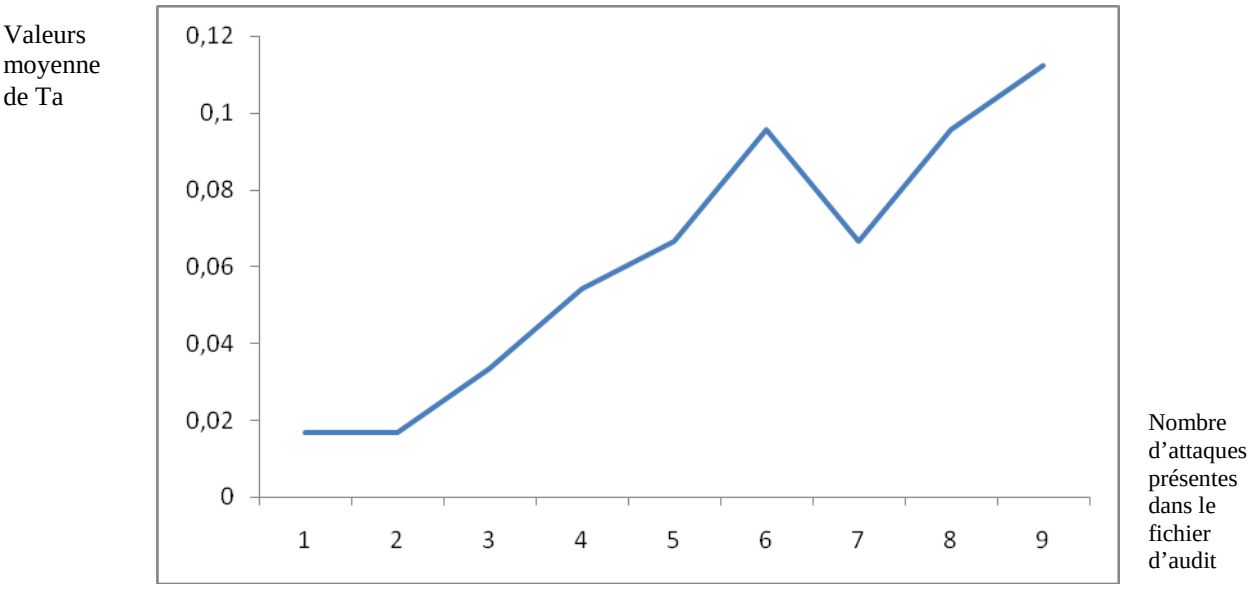


Fig. V.15 - Evolution moyenne (sur 10 exécutions) de taux Ta en fonction du nombre d'attaques présentes dans le fichier d'audit.

4. Etude comparative :

Dans cette section nous présentons l'étude comparative entre les résultats trouvés par notre approche et ceux donnés par l'algorithme mimétique proposé par SAHMADI dans [59], et ce dans le but de prouver la performance de notre méthode dans la résolution du problème d'analyse de fichier d'audit de sécurité.

Nos critères de comparaison sont donnés comme suit :

- Comparaison selon la fonction sélective
- Comparaison selon les taux de présence et d'absence

- *Pour l'évolution de la fonction objectif :*

Dans ci-après, nous citons les histogrammes trouvés dans l'approche mimétique qui représentent l'évolution moyenne (sur 10 exécutions) de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour Nb_attaques= 2, 4, 6, 8, 10, 16, 18, 20.

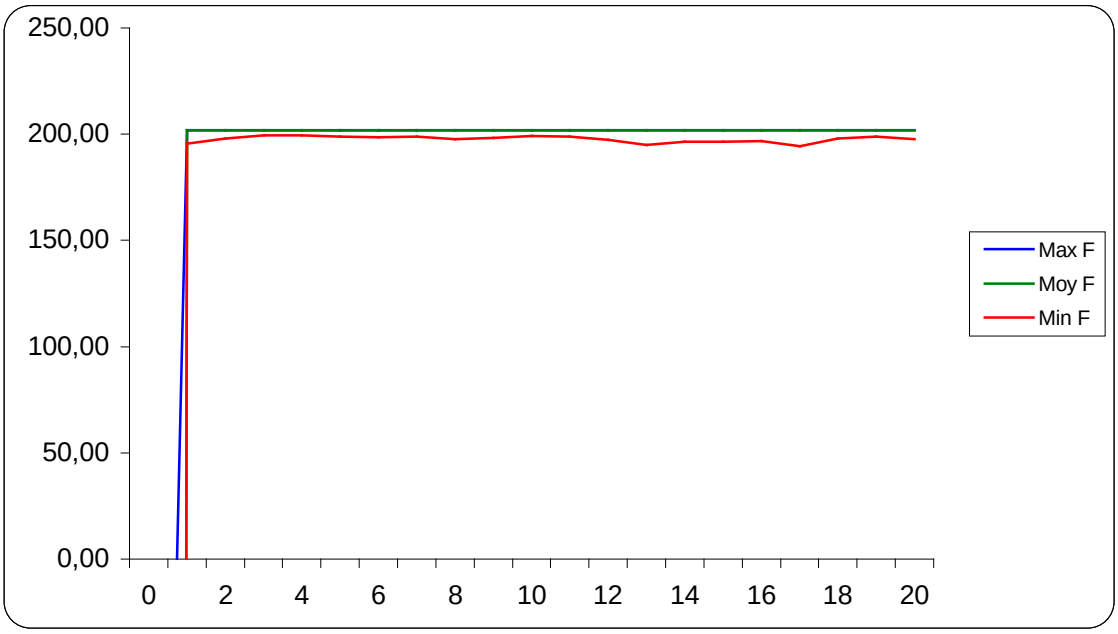


Fig. V.16 - Exemple d'évolution moyenne (sur 10 exécutions) de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour Nb_attaques = 2. [59]

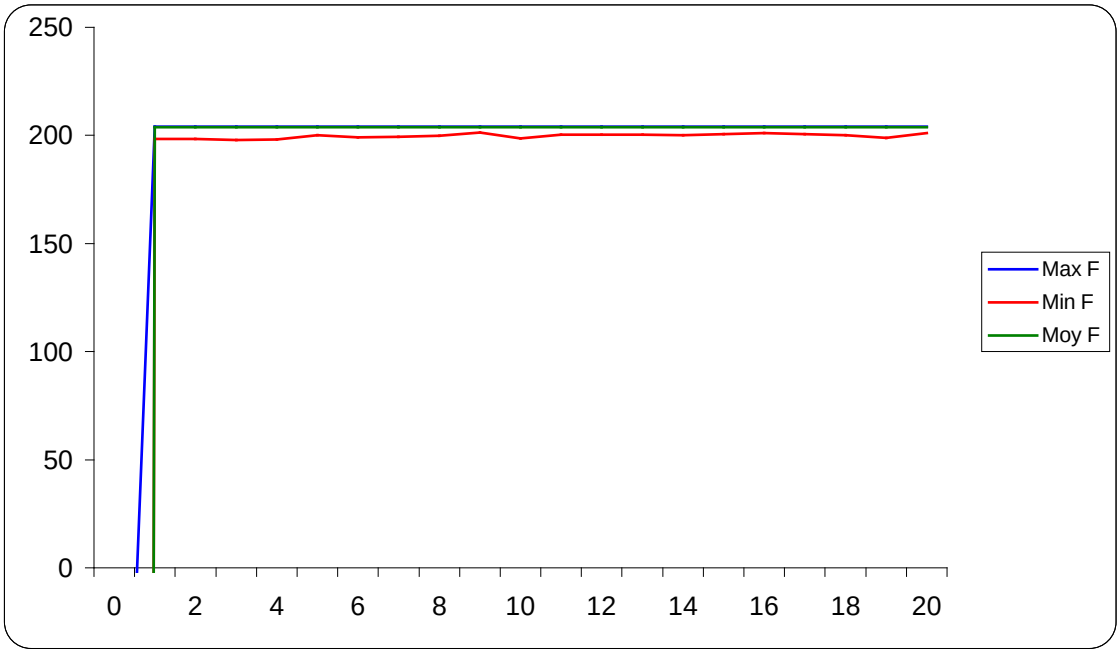


Fig. V.17 - Exemple d'évolution moyenne (sur 10 exécutions) de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour Nb_attaques = 4. [59]

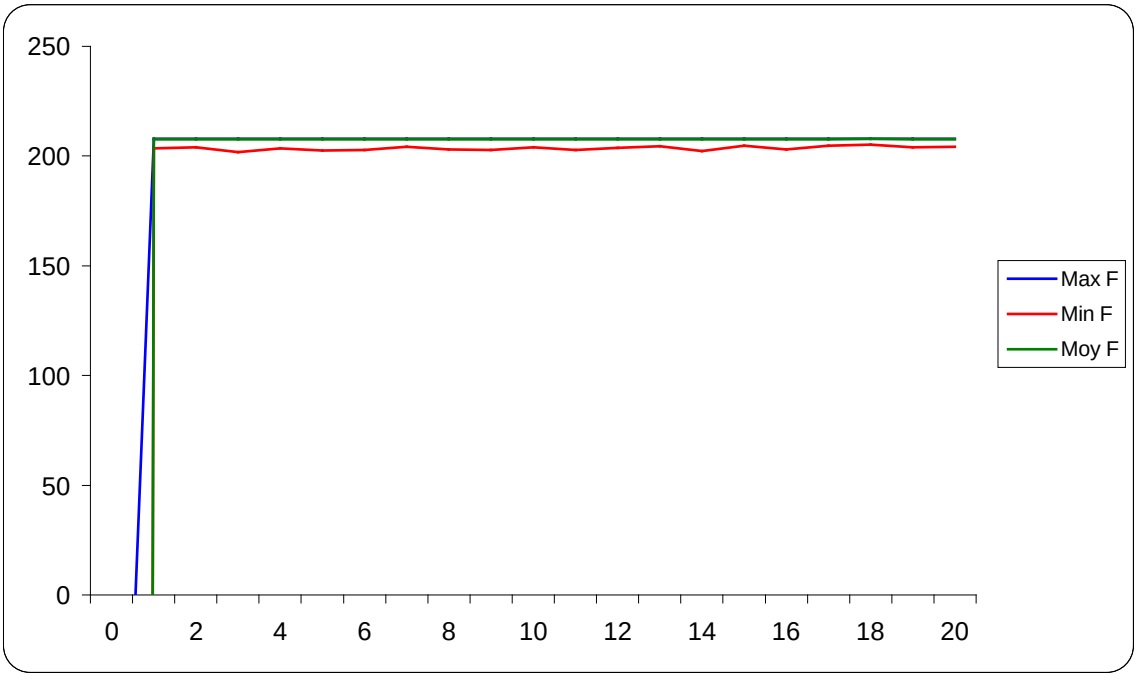


Fig. V.18 - Exemple d'évolution moyenne (sur 10 exécutions) des valeurs sélectives minimale, maximale et moyenne en fonction de la génération pour Nb_attaques = 8. [59]

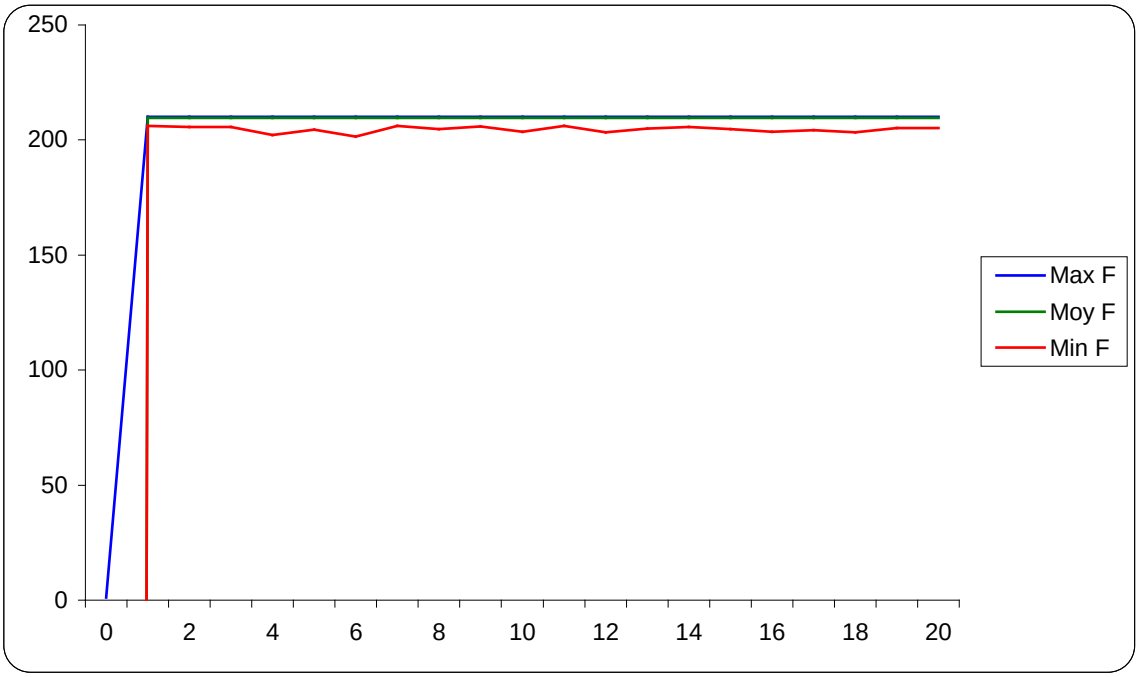


Fig. V.19 - Exemple d'évolution moyenne (sur 10 exécutions) de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour Nb_attaques = 10. [59]

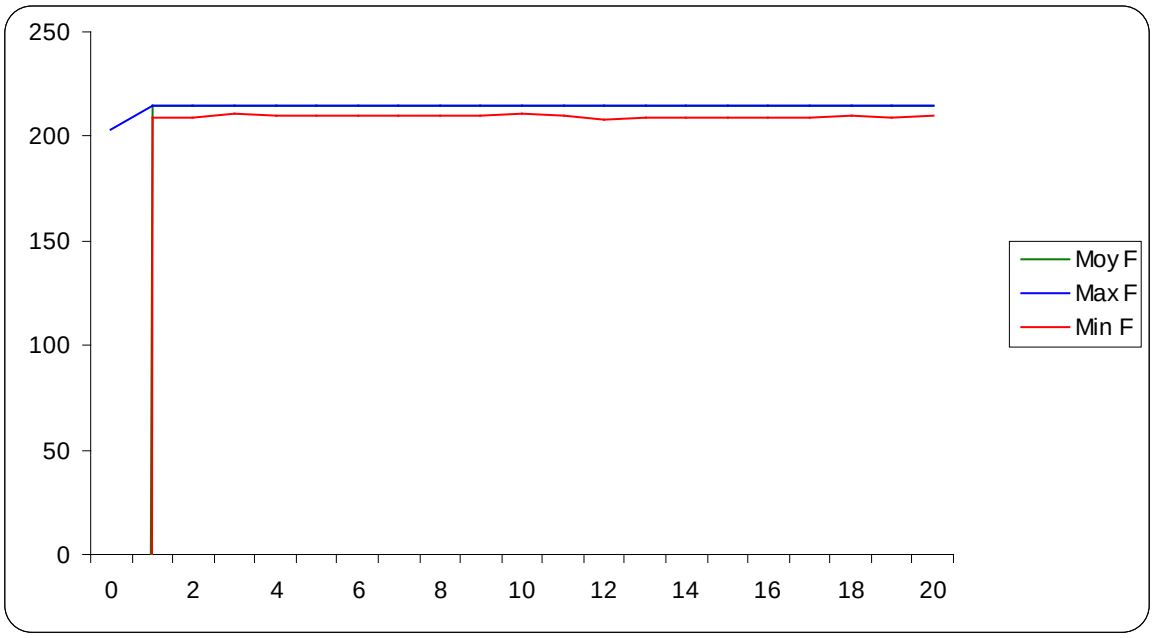


Fig. V.20 - Exemple d'évolution moyenne (sur 10 exécutions) de la fonction objectif : minimale, maximale et moyenne en fonction de la génération pour Nb_attaques = 15. [59]

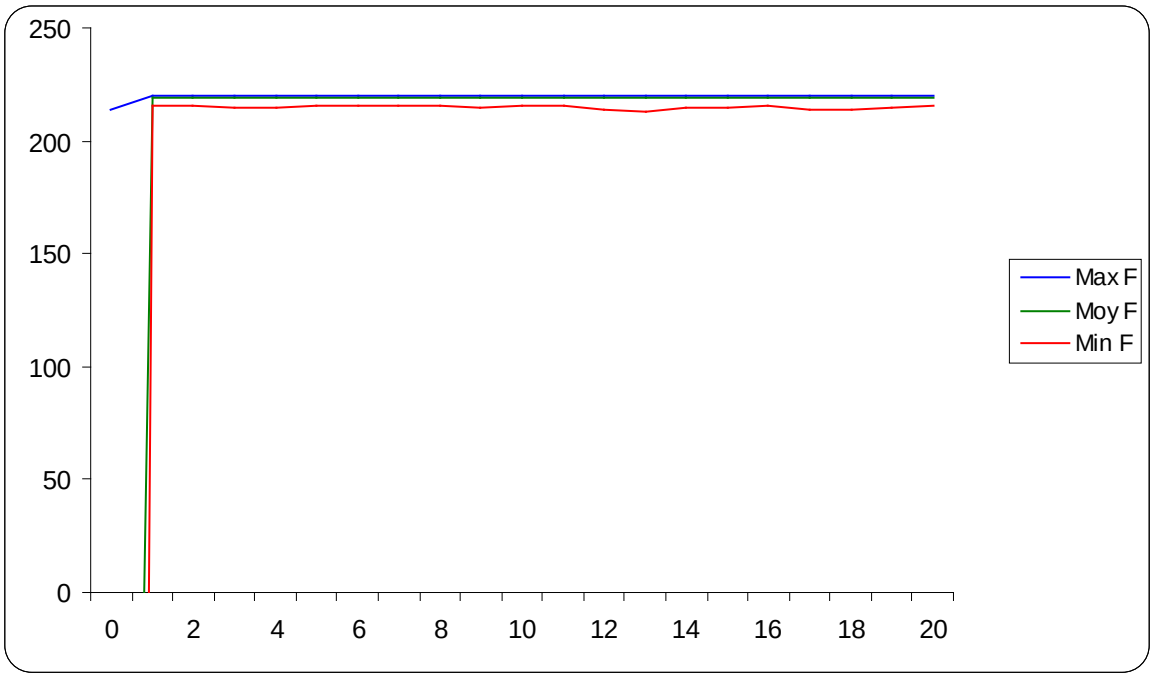


Fig. V.21 - Exemple d'évolution moyenne (sur 10 exécutions) de la fonction objectif : minimales, maximale et moyenne en fonction de la génération pour Nb_attaques = 20. [59]

Discussion 01 :

Comme le montre les figures allant de Fig V.5 à Fig V. 12, nous remarquons que les résultats trouvés par l'approche SLS sont nettement meilleures à ceux donnés par l'approche mémitique proposé dans [59].

- *Pour le taux de détection :*

Dans cette section nous citons les histogrammes trouvés dans l'approche mimétique qui représentent l'évolution de taux de présence et d'absence.

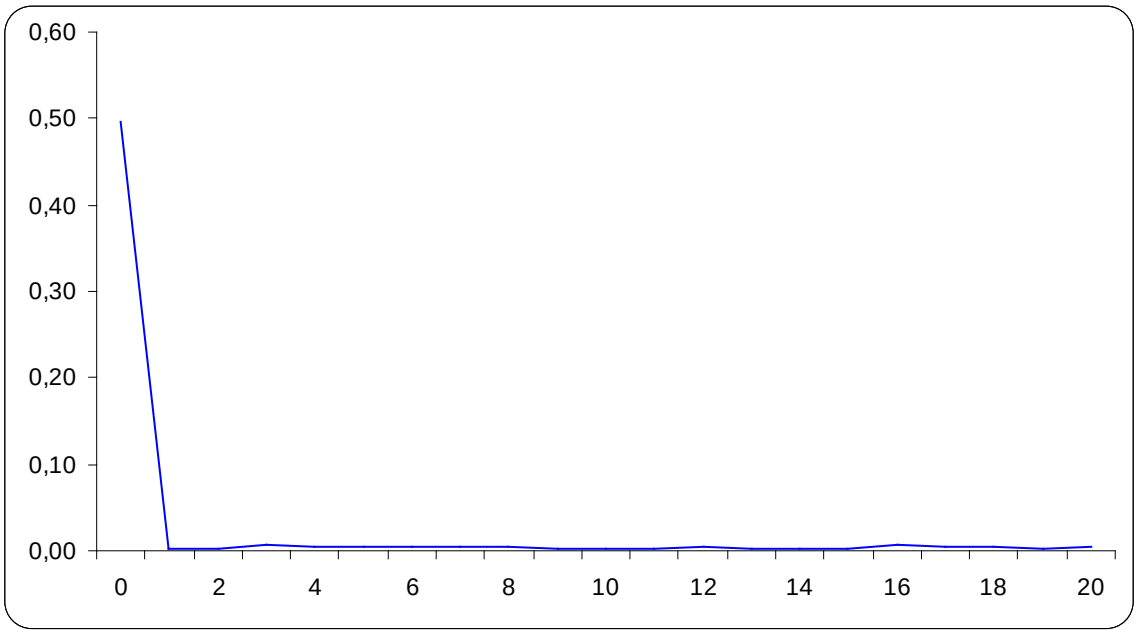


Fig. V.22 - Evolution de taux Ta en fonction de la génération pour une seule exécution, pour le cas où Nb_attaques = 2. [59]

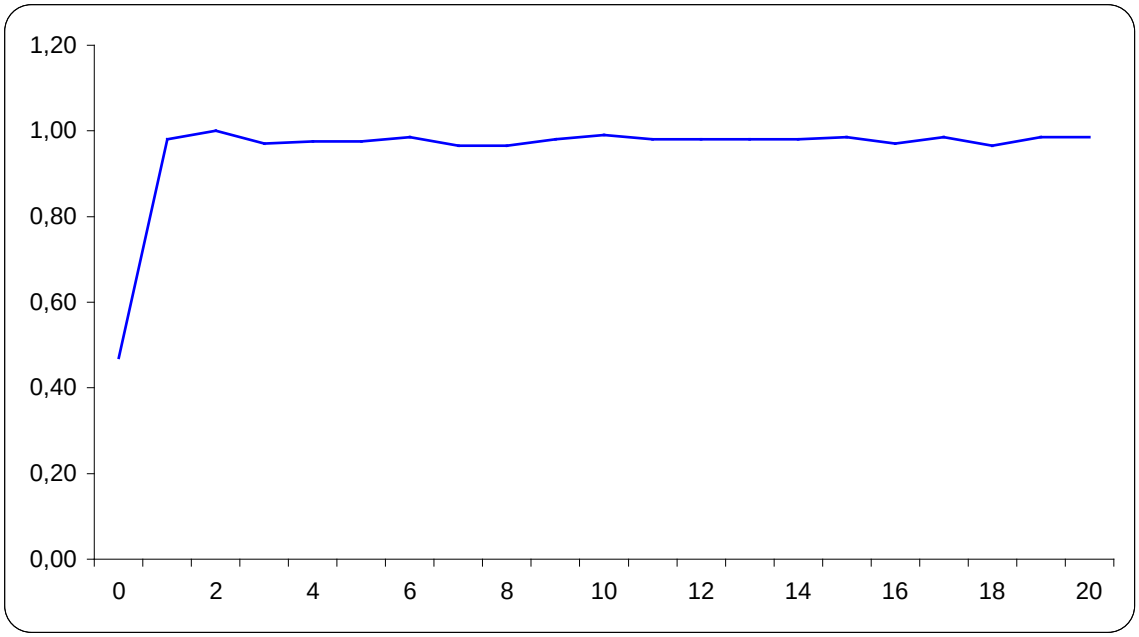


Fig. V.23 - Evolution de taux Tp en fonction de la génération pour une seule exécution, pour le cas où Nb_attaques = 2. [59]

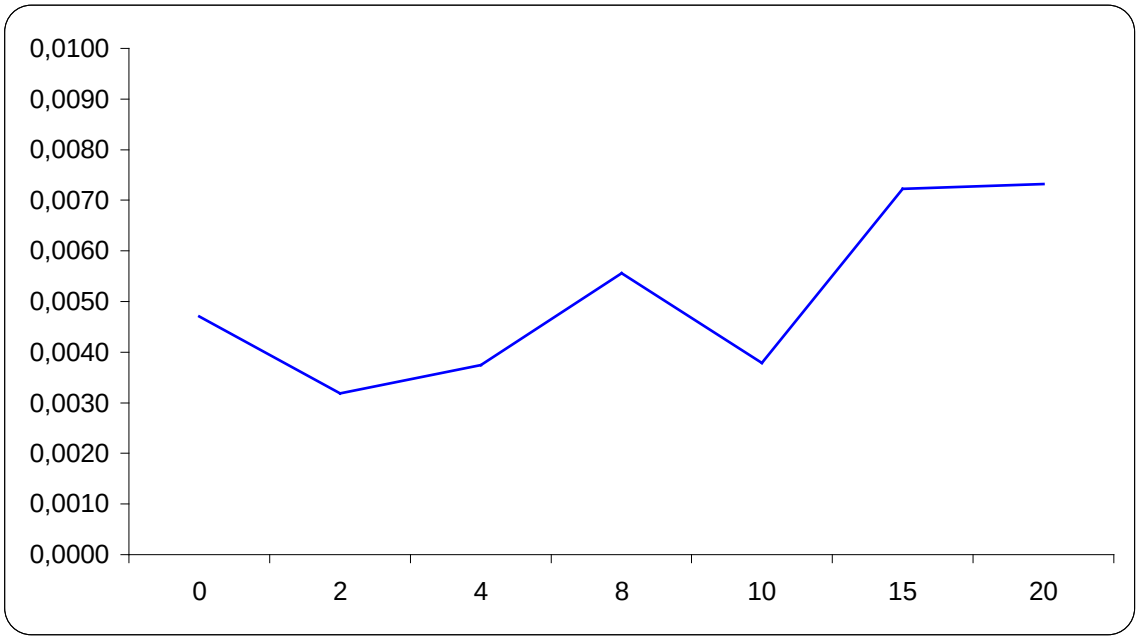


Fig. V.24 - Evolution moyenne (sur 10 exécutions) de taux Ta en fonction du nombre d'attaques présentes dans le fichier d'audit. [59]

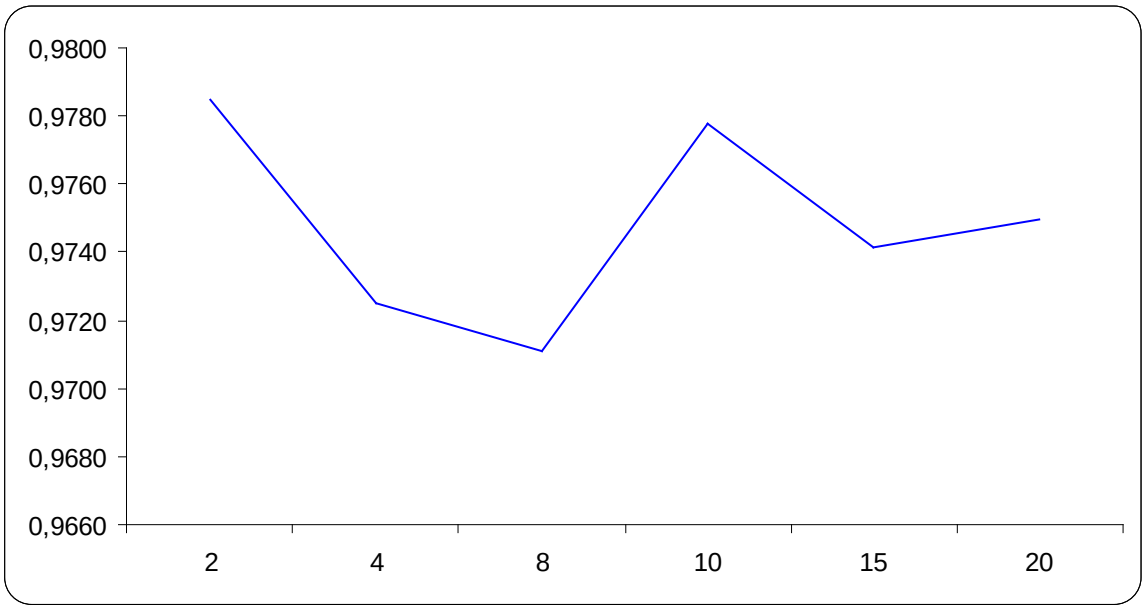


Fig. V.25 - Evolution moyenne (sur 10 exécutions) de taux Tp en fonction du nombre d'attaques présentes dans le fichier d'audit. [59]

Discussion02 :

Concernant l'évolution du taux de présence Tp et d'absence Ta en fonction de la génération pour une seule exécution pour le cas où Nb_attaques = 2 (voir Fig. V.13, Fig. V.23 et Fig. V.24), les deux méthodes (MA et SLS) sont comparables.

Pour l'évolution moyenne (sur 10 exécutions) de taux Tp en fonction du nombre d'attaques présentes dans le fichier d'audit (voir Fig. V.14 et Fig. V.26), d'une manière générale la variation dans le taux de présence se fait d'une manière incrémentale c'est-à dire lorsque on augmente le nombre d'attaque, la valeur de Tp s'augmente aussi. Cependant lorsque le nombre d'attaques est entre [3, 7] on remarque que le taux Tp diminue.

A la comparaison entre le SLS et le MA en terme de taux d'absence, on remarque que le taux d'absence généré par le SLS et celui généré par le MA ont pris des valeurs très faibles (voir Fig. V.15 et Fig. V.25).

En général les résultats trouvés par le SLS sont très encourageants, ce qui montre la performance de notre approche.

5. Conclusion :

Nous avons présenté dans ce chapitre les grandes caractéristiques de système de détection d'intrusion proposé, puis nous avons détaillé l'algorithme SLS utilisé pour la recherche des attaques dans le fichier d'audit de sécurité.

Dans le but d'évaluer et tester l'application que nous avons implémentée, nous avons défini une matrice Attaques/Evénements (AE) regroupant 24 attaques réalistes possibles sous un système UNIX, et le fichier d'audit utilisé représente une simulation du comportement d'un utilisateur sur une période d'une 30 minutes, ce fichier est représenté sous forme d'un vecteur d'audit observé O. A chaque expérimentation on introduit dans la trace d'audit des attaques de la matrice Attaques/Evénements en nombre variant.

Les résultats expérimentaux obtenus sont très bons de point de vue de la qualité d'analyse, où presque toutes les attaques introduites dans la matrice d'audit sont détectées pour tous les cas de nombre d'attaques variant de 2 à 20, par exemple, le nombre de faux négatif se diminue lorsque le nombre d'attaques présentes s'augmente.

Conclusion générale

Les systèmes de détection d'intrusions sont des moyens très importants et complémentaires pour garantir une prévention dans la sécurité des systèmes informatiques. Dans ce mémoire nous avons proposé une approche par scénarios pour la détection d'intrusions, qui consiste à rechercher les scénarios d'attaques pré-définis dans les traces d'audit de sécurité, en utilisant un algorithme de recherche locale stochastique pour l'analyse de fichier d'audit.

Dans cette approche les scénarios d'attaques sont définis sous forme de matrice, appelée matrice attaques/événements (AE), où chaque élément AE_{ij} de cette matrice indique le nombre d'occurrences de l'événement i dans l'attaque j. Ceci fournit à l'officier de sécurité plus de liberté dans la construction et la mise à jour des signatures d'attaques. En plus, la collecte des événements d'audit se limite à celle des événements qui apparaissent dans la matrice attaques/événements.

L'approche proposée se base sur une formalisation simplifiée du problème d'analyse de fichier d'audit, qui fait l'hypothèse de l'indépendance des attaques, et ne prend pas en compte l'ordre temporel des événements, mais l'approche est très utile comme un filtre pour détecter les attaques du système. Le problème d'analyse de fichier d'audit est exprimé sous forme d'une optimisation contrainte, et un algorithme SLS est appliqué pour sa résolution. L'algorithme utilise l'opérateur de sélection. Le codage des solutions est binaire où chaque bit code la présence ou l'absence d'une attaque indépendamment de celle des autres.

Pour évaluer chaque solution, une fonction objectif est utilisée qui est constituée d'un terme permettant de rechercher l'ensemble des attaques maximisant le risque total encouru par le système et d'un terme pénalisant les solutions qui ne respectent pas la contrainte.

Une implémentation a été faite pour l'algorithme proposé, et des expérimentations ont été réalisées on utilisant une matrice d'Attaques/Evénement de 24 attaques et des matrices observées (O) obtenues de la simulation de comportements intrusifs d'un utilisateur. Dans toutes les expérimentations faites sur l'algorithme proposé, on peut

discriminer très nettement les attaques potentiellement présentes dans les traces d'audit et les performances observées sont très satisfaisantes de point de vue de la qualité du diagnostic proposé, puisqu'il y a une discrimination totale entre les attaques présentes dans les traces analysées et les attaques absentes de ces traces, et ces résultats sont indépendants du nombre d'attaques contenues dans les traces.

Après la réalisation de notre travail, nous avons acquis beaucoup de connaissances sur le domaine de la sécurité informatique, et précisément les systèmes de détection d'intrusions et sur l'implémentation des algorithmes SLS.

Du point de vu de la mise en œuvre de notre approche, on note l'importance de bien choisir les paramètres tel que le nombre maximum de génération, la valeur de la probabilité wp,…etc.

Comme perspectives à ce travail, nous prévoyons :

- D'augmenter la taille de matrices d'Attaques/Evénements utilisée par notre algorithme SLS, et si possible le tester sur un environnement réel.
- De proposer et d'appliquer une autre approche que les algorithmes de recherche locale stochastique et comparer les résultats obtenus avec ceux de la SLS.

Annexe A : Description des attaques utilisées dans les expérimentations :

Attaque 1 :

Description : Une intrusion par un attaquant extérieur est souvent précédée par un grand nombre de tentatives infructueuses de connexion (l'attaquant essaye des noms de login tels que test ou essai).

Activités : Le fichier /bin/login est exécuté à plusieurs reprises et la connexion est refusée.

Audit :

Évènement	Nombre	statut
USER_Login	3	fail

Attaque 2 :

Description : Nombre de passages en super-utilisateur abusif de la part d'un utilisateur qui connaît illégalement le mot de passe et qui se retire dés qu'il constate qu'un administrateur légal est connecté.

Activités : Usage des commandes su et who, w, ps ou f.

Audit :

Évènement	Nombre	Commandes	statut
USER_SU	3		OK
Proc_Execute	3	who, w, ps, finger, f	-

Attaque 3 :

Description : L'attaquant tente de passer super-utilisateur. Il essaye de deviner le mot de passe en fonction de ce qu'il connaît de l'organisation de la société et/ou de l'administrateur.
Activités : Usage infructueux de la commande su.

Audit :

Évènement	Nombre	statut
USER_SU	3	fail

Attaque 4 :

Description : A fin de ne pas se faire repérer par l'administrateur, les attaquants pratiquent parfois par de courtes sessions de l'ordre de 5 minutes.

Activités : Connexion et déconnexion en moins de 5 minutes.
Audit :

Évènement	Nombre
Short_Session	1

Attaque 5 :

Description : Connections à des heures particulièrement tardives ou matinales.

Activités : Usage de la commande login.

Audit :

Évènement	Nombre	Heure
USER_Login	3	Entre 23h 00 et 6h 00

Attaque 6 :

Description : L'attaquant visualise les noms des fichiers dans les différents répertoires du disque, à la recherche d'informations qui pourraient lui être utiles. Il utilise donc les commandes cd et ls (rappelons que l'utilisation de cd n'est pas auditable car c'est une commande interne du shell). Lorsqu'un fichier semble intéressant, l'attaquant visualise son contenu ou son type.

Activités : Utilisation de la commande ls et des commandes more, pg, cat, head, vi, view ou file.

Audit :

Évènement	Nombre	Commandes	statut
Proc_Execute	30	ls	OK
Proc_Execute	5	more, pg, cat, head, vi, view,file	OK

Attaque 7 :

Description : En cas de furetage par commandes cd et ls, cette dernière commande échoue lorsque le répertoire est protégé (par exemple avec les droits drwx--x--x).

Activités : Echec de l'utilisation de la commande ls.

Audit :

Évènement	Nombre	Commandes	statut
Proc_Execute	5	ls	fail

Attaque 8 :

Description : En cas de furetage l'attaquant peut parvenir à lister le contenu de répertoire non protégé puis tenter d'accéder à des fichiers protégés (par exemple avec les droits rw-------).

Activités : Echec de l'ouverture de fichiers.

Audit :

Évènement	Nombre	statut
File_Open	5	fail

Attaque 9 :

Description : Pour utiliser le compte d'un autre utilisateur, il faut connaître son nom de login. Pour cela, un attaquant possédant lui-même un compte sur la machine (de manière légale ou non) utilise fréquemment les commandes permettant d'identifier les utilisateurs.

Activités : Exécution des commandes permettant d'obtenir des informations sur les utilisateurs légaux du système : who, w, finger, f, last, groups, ps, id ou lsuser.

Audit :

Évènement	Nombre	Commande
Proc_Execute	8	who, w, finger,f, last, groups, ps, id, lsuser

Attaque 10 :

Description : Utilisation abusive de commandes permettant d'obtenir des informations sur la configuration du système.

Activités : Exécution des commandes df, hostname ou uname.

Audit :

Évènement	Nombre	Commande
Proc_Execute	3	df, hostname, uname

Attaque 11 :
Description : Utilisation abusive de commandes permettant d'obtenir des informations sur le réseau.

Activités : Exécution des commandes arp, netstat ou ping

Audit :

Évènement	Nombre	Commande
Proc_Execute	2	arp, netstat, ping

Attaque 12 :

Description : Lorsque les pages jaunes sont utilisées, il est possible d'obtenir de nombreuses informations sur le système comme la liste des noms de login légaux (incluant le mot de passe chiffré, ce qui permet une attaque par dictionnaire) ou la liste des machines du réseau avec leur adresse IP. Ce sont des informations précieuses pour un attaquant.

Activités : Usage de la commande ypcat
Audit :

Évènement	Nombre	Commande
Proc_Execute	3	ypcat

Attaque 13 :

Description : Par exemple à la suite d'un furetage concluant, l'attaquant copie des fichiers des comptes visités vers le sien.

Activités : Utilisation de la commande cp.

Audit :

Évènement	Nombre	Commandes	statut
File_Write	30	cp	OK

Attaque 14 :

Description : L'attaquant tente de copier un fichier vers son compte. S'il n'a pas les droits de lecture sur le fichier, cette copie échoue.

Activités : L'ouverture du fichier échoue.

Audit :

Évènement	Nombre	Commandes	statut
File_Open	10	cp	Fail access

Attaque 15 :

Description : Par exemple à la suite d'un furetage concluant, l'attaquant imprime des fichiers qui l'intéressent.

Activités : Accès en lecture à des fichiers par un processus résultant de l'utilisation de la commande lpr.

Audit :

Évènement	Nombre	Commandes
File_Read	10	lpr

Attaque 16 :

Description : Décrite par M.Bishop en 1986, l'attaque suivante est toujours possible sous AIX 3.2.3. Elle permet à un utilisateur quelconque d'imprimer un fichier sur lequel il n'a aucun droit. Le principe de l'attaque consiste à créer un lien symbolique indirect entre le fichier convoité et un fichier se trouvant dans le spool d'impression.

Activités : L'attaquant pratique en cinq étapes :

1. création d'un fichier temporaire quelconque (appelons le f),
2. attente jusqu'à ce que la file d'impression ne soit plus vide (usage de la commande lpstat) ou blocage de l'imprimante (selon les possibilités par impression d'un fichier de taille importante, par usage de la commande disable ou par mise hors tension physique),
3. impression du fichier f en créant un lien symbolique entre la zone de spool et le fichier f (lpr -s f),
4. destruction du fichier f (rm f ou mv f f'),
5. création d'un lien symbolique de nom f vers le fichier convoité, par exemple le fichier des mots de passe (ln -s /etc/security/passwd f).

Audit :

Évènement	Nombre	Commandes
Proc_Execute	1	lpr
Proc_Execute	1	rm ou mv
Proc_Execute	1	ln

Attaque 17 :

Description : Un utilisateur détruit de nombreux fichiers.

Activités : Usage de la commande rm.

Audit :

Évènement	Nombre	Commandes
File_Unlink	50	rm

Attaque 18 :

Description : L'utilisateur lance de nombreuses commandes ou exécute de nombreux programmes afin de diminuer les performances de la machine ou de saturer la table de processus.

Activités : La primitive système exec est appelée.

Audit :

Évènement	Nombre
PROC Execute	300

Attaque 19 :

Description : L'utilisateur lance de nombreux processus en arrière plan.

Activités : La primitive système fork est appelée et le niveau de priorité du processus est modifie.

Audit :

Évènement	Nombre
PROC_SetPri	100

Attaque 20 :

Description : L'attaquant ayant réussi à se connecter à la place d'un utilisateur légal consulte le fichier ~/.netrc afin d'obtenir les mots de passe permettant de se connecter sur une machine distante.

Activités : Accès en lecture au fichier ~/.netrc suivi de l'exécution d'une commande à distance (rexec, rlogin, rsh, rcp) ou de l'utilisation de ftp.

Audit :

Évènement	Nombre	Commandes	Fichier
Proc_Execute	1	more, pg, cat, head, vi, view	~/.netrc
Proc_Execute	1	rexec, rlogin, rsh, rcp, ftp	

Attaque 21 :

Description : Un utilisateur ayant réussi à se connecter à la place d'un autre modifie les droits d'accès aux fichiers qui l'intéressent, afin de pouvoir les consulter ultérieurement de son propre compte.

Activités : Usage de la commande chmod.

Audit :

Évènement	Nombre	statut
File_Mode	3	OK

Attaque 22 :

Description : L'attaquant consulte le contenu de plusieurs fichiers sensibles, tels que celui des mots de passe ou celui des groupes.

Activités : Accès en lecture (par more, pg, cat ou vi) à \etc\passwd, \etc\group, \etc\opasswd, \etc\ogroup, \etc\hosts, \etc\hosts.equiv, \etc\inetd.conf ou \etc\services.

Audit :

Évènement	Nombre	Fichier	Commandes
FILE_Read	5	\etc\passwd, \etc\group, \etc\opasswd, \etc\ogroup, \etc\hosts, \etc\hosts.equiv, \etc\inetd.conf ou \etc\services	more, pg, cat, vi

Attaque 23 :

Description : L'utilisateur tente de modifier le contenu d'un fichier sensible et cette tentative échoue.

Activités : Accès en écriture à \etc\passwd, \etc\group, \etc\opasswd, \etc\ogroup ou \etc\hosts.

Audit :

Évènement	Nombre	Paramètre	Statut
FILE_Write	1	\etc\passwd, \etc\group, \etc\opasswd, \etc\ogroup, \etc\hosts	fail

Attaque 24 :

Description : L'utilisateur modifie régulièrement son identité, si bien qu'il lui arrive de ne plus savoir où il en est. Il utilise alors la commande whoami ou la commande id.

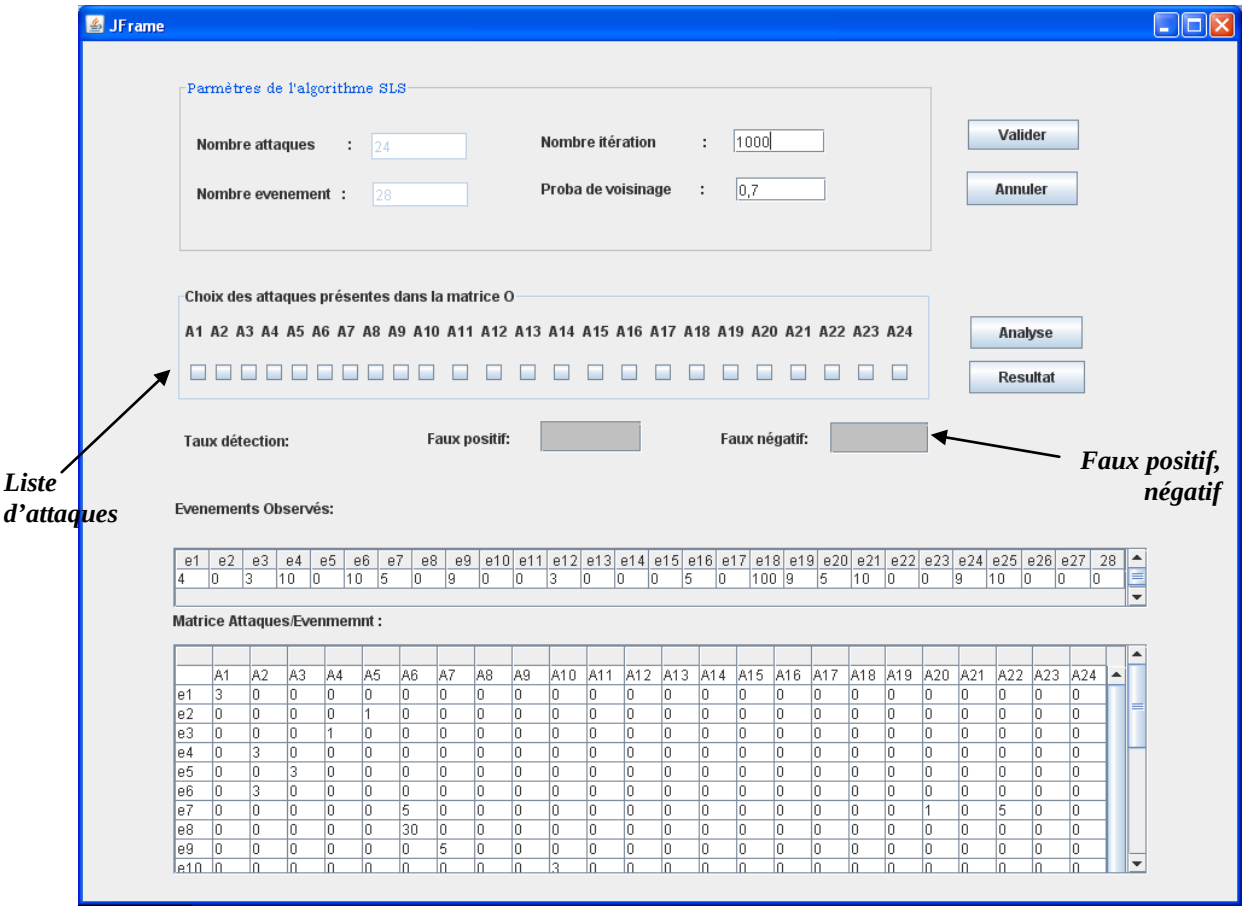
Activités : Exécution de la commande \bin\whoami.

Audit :

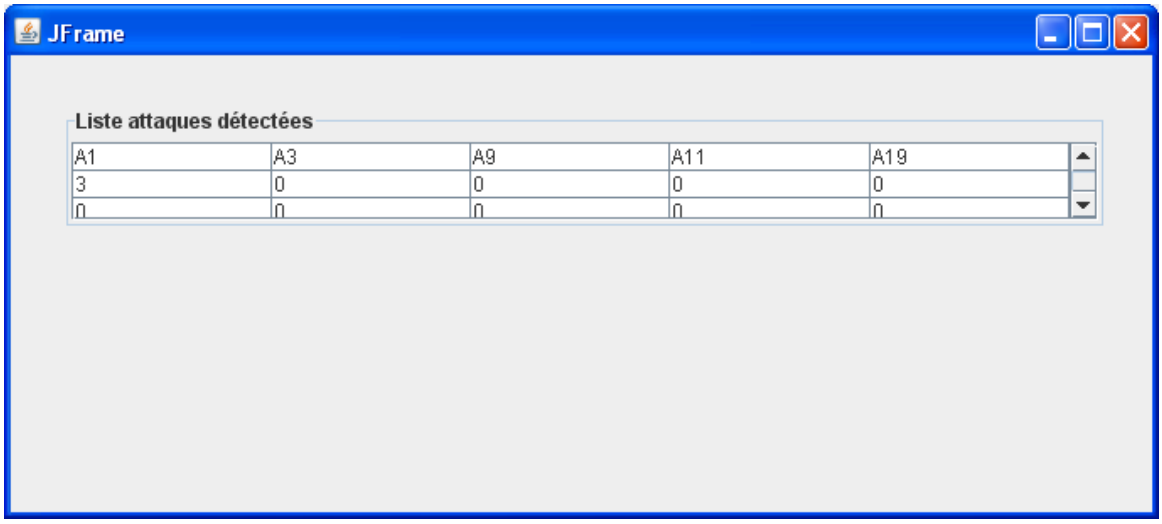
Évènement	Nombre	Commandes
ROC_Execute	4	whoami, id

Annexe B :

Interface principale de l'application (IDS_SLS) :



Interface de résultats d'analyse avec la liste d'attaques détectées :



Bibliographie :

[1]: Anas ABOU EL KALAM. Modèles et politiques de sécurité pour les domaines de la santé et des affaires sociales, Thèse de Doctorat. L'Institut National Polytechnique de Toulouse. Décembre 2003.

[2] : Jacob Zimmermann et Ludovic Mé. Les systèmes de détection d'intrusions : principes algorithmiques. Supélec, équipe SSIR.
Url: http://www.supelecrennes.fr/rennes/si/equipe/lme/.

[3] : Ludovic Mé et Cédric Michel. La détection d'intrusion : bref aperçu et derniers développements. Mars 1999.

[4]: Chunsheng Li, Qingfeng Song, et Chengqi Zhang. MA-IDS Architecture for Distributed Intrusion Detection using Mobile Agents. Proceedings of the 2nd International Conference on Information Technology for Application (ICITA). 2004.

[5]: Noria Foukia. IDReAM- Intrusion Detection and Response executed with Agent Mobility- A distributed and Self- Organizing Approach. Thèse de doctorat de l'Université de Genève. 2003.

[6] : Farah Abdel Majid Barika. Vers un IDS Intelligent à base d'Agents Mobiles. Mémoire de DEA de l'Université de Tunis, Institut Supérieur de Gestion. 2003.

[7] : Etienne Duris, Nicolas Baudoin et Marion Karle. NT Réseaux : IDS et IPS. l'UFR d'Ingénieurs de l'Université Paris-Est Marne-la-Vallée Ingénieurs. 2000.

[8] : Klauss Muller. IDS : Système de détection d'intrusion, partie I. LinuxFocus article number 292. Url : http://linuxfocus.org.

[9] : Cédric Michel. «Langage de description d'attaques pour la détection d'intrusions par corrélation d'événements ou d'alertes en environnement réseau hétérogène », thèse de doctorat. Université de Rennes 1, Décembre 2003.

[10]: Zimmermann Jacob, Mé Ludovic et Bidan Christophe. Introducing reference flow control for intrusion detection at the os level. In Proceedings of the 5th International Symposium on the Recent Advances in Intrusion Detection (RAID'2002).

[11]: Debar. H, Becker. M et Siboni D. A neural network component for an intrusion detection system. In Proceedings of the IEEE Symposium of Research in Computer security and Privacy. Mai 1992.

[12]: Javitz. H.S, Valdes. A, Lunt. T.F, Tamaru. A, Tyson. M et Lowrance. J. Next Generation Intrusion Detection Expert System (NIDES). Rapport technique, 1993.

[13]: Internet Security Systems ISS, Network- vs. host-based intrusion detection: A guide to intrusion detection technology, Atlanta, Octobre 1998.

[14]: Cisco Systems. Netranger intrusion detection system technical overview. December 1998.

[15] : Vigna (Giovanni) et Kemmerer (Richard A.). Netstat : A network-based intrusion detection system. Journal of Computer Security, Fevrier 1999.

[16]: Holger Hoos and Thomas Stutzle. Stochastic Local Search. In Teofilo F. Gonzalez, editor, Approximation Algorithms and Metaheuristics, Chapman and Hall / CRC Press, 2007.

[17]: Holger H. Hoos, Thomas Stützle. Stochastic Local Search Foundations andApplications (book), Departments of Computer Science of University of British Columbia (Canada) and Darmstadt University of Technology (Germany), Mars 2006.

[18]: S. Voss, S. Martello, I. H. Osman, and C. Roucairol, editors. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers. 1999.

[19]: H. Hoos and T. Stutzle. Stochastic Local Search|Foundations and Applications. Morgan Kaufmann Publishers, 2004.

[20]: M. Yannakakis. The analysis of local search problems and their heuristics. In C. Choffrut and T. Lengauer, editors, STACS 90, 7th Annual Symposium on Theoretical Aspects of Computer Science, volume 415 of LNCS, pages 298-310. Springer Verlag, 1990.

[21]: O. C. Martin, S. W. Otto, and E. W. Felten. Large-step Markov chains for the traveling salesman problem. Complex Systems, 5(3):299-326, 1991.

[22]: Jon Jouis Bentley. Fast algorithms for geometric traveling salesman problems. ORSA Journal on Computing, 1992.

[23]: P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. European Journal of Operational Research, 130(3):449-467, 2001.

[24]: P. Hansen and N. Mladenovic. Variable neighborhood search. In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, pages 145-184. Kluwer Academic Publishers, 2002.

[25]: B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. Bell Systems Technology Journal, 49:213-219, 1970.

[26]: S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. Operations Research, 21(2):498-516, 1973.

[27]: F. Glover. Ejection chain, reference structures and alternating path methods for traveling salesman problems. Discrete Applied Mathematics, 65(1-3):223-253, 1996.

[28]: P. M. Thompson and J. B. Orlin. The theory of cycle transfers. Working Paper OR 200-89, Operations Research Center, MIT, Cambridge, MA, 1989.

[29]: P. M. Thompson and H. N. Psaraftis. Cyclic transfer algorithm for multivehicle routing and scheduling problems. Operations Research, 41:935-946, 1993.

[30]: C. N. Potts and S. van de Velde. Dynasearch: Iterative local improvement by dynamic

programming; part I, the traveling salesman problem. Technical Report LPOM-9511, Faculty of Mechanical Engineering, University of Twente, Enschede, The Netherlands, 1995.

[31]: R. K. Congram, C. N. Potts, and S. van de Velde. An iterated dynasearch algorithm for the singlemachine total weighted tardiness scheduling problem. INFORMS Journal on Computing, 14(1):52-67, 2002.

[32]: R. K. Ahuja, J. B. Orlin, and D. Sharma. Multi-exchange neighbourhood structures for the capacitated minimum spaning tree problem. Working Paper, 2000.

[33]: B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In Proceedings of the 12th National Conference on Artificial Intelligence, pages 337-343. AAAI Press / The MIT Press, 1994.

[34]: D. T. Connolly. An improved annealing scheme for the QAP. European Journal of Operational Research, 46(1):93-100, 1990.

[35]: M. Fielding. Simulated annealing with an optimal fixed temperature. SIAM Journal on Optimization, 11(2):289-307, 2000.

[36]: S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. Science, 220:671-680, 1983.

[37]: V. Cerny. A thermodynamical approach to the traveling salesman problem. Journal of Optimization Theory and Applications, 45(1):41-51, 1985.

[38]: Thomas. A. Feo and Mauricio. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters, 8(2):67-71, 1989.

[39]: M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, pages 219-249. Kluwer Academic Publishers, 2002.

[40]: M. Dorigo and T. Stutzle. Ant Colony Optimization. MIT Press, 2004.

[41]: M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

[42]: T. Stutzle and H. H. Hoos. MAX-MIN Ant System. Future Generation Computer Systems, 16(8):889-914, 2000.

[43]: M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 11-32. McGraw Hill, 1999.

[44]: P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 219-234. McGraw Hill, 1999.

[45]: F. Glover, M. Laguna, and R. Marti. Scatter search and path relinking: Advances and applications. In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, pages 1-35. Kluwer Academic Publishers, 2002.

[46]: M. Laguna and R. Marti. Scatter Search: Methodology and Implementations in C, volume 24. Kluwer Academic Publishers, 2003.

[47]: Anderson (J.P.). Computer Security Threat Monitoring and Surveillance. James Anderson Company, Fort Washington, Pennsylvania, 1980.

[48]: Ludovic Mé. GASSATA, A Genetic Algorihm as an Alternative Tool for Security Audit Trails Analysis. 1998.

[49]: Ludovic Mé. Audit de sécurité par algorithmes génétiques. Thèse de doctorat de l'université de Rennes 1 - Numéro d'ordre 1069, 1994.

[50]: Baker, J. E. Reducing Bias and Inefficiency in the Selection Algorithm. In [51], pp. 14-21, 1987

[51]: Grefenstette, J. J. (ed.). Proceedings of the Second International Conference on Genetic Algorithms and their Application. Hillsdale, New Jersey, USA: Lawrence Erlbaum

Associates, 1987

[52]: Ghenima BOURKACHE. Un IDS réparti basé sur une société d'agents intelligents, mémoire de magister informatique, université de Boumerdes. Algérie 2007.

[53]: Kenaza Tayeb. Détection d'intrusion coopérative basée sur la fusion de données, mémoire de Magister de l'INA. 2006.

[54] : Sécurité des systèmes d'information, www.e-picardie.net, 2002.

[55] : Ludovic Mé - Véronique Alanou, Détection d'intrusion dans un système informatique : méthodes et outils. 1996.

[56] : Karima Boudaoud, Détection d'intrusions : Une nouvelle approche par systèmes multi-agents, Ecole Poly echnique Fédérale de Lausanne, 2000.

[57]: A. S. Jain, B. Rangaswamy, and S. Meeran. New and "stronger" job-shop neighbourhoods: A focus on the method of Nowicki and Smutnicki. Journal of Heuristics, 6(4):457-480, 2000.

[58]: Laurent vanel, Rosabelle Zapata-Balingit, Gonzalo R. Archondo-Callao, International Technical Support Organization: Auditing and accounting on AIX, ibm.com/redbooks, 2000.

[59]: Brahim Sahmadi, Thèse de magister : Un algorithme mimétique pour l'analyse d'audit dans les systèmes IDS, département d'informatique USTHB, 2007.

[60]: Lehmann Guillaum, cours de sécurité informatique 2003-04-13.

[61]: T. Bäck. Evolutionary Algorithms in Theory and Practice, Oxford University Press, 1996, 1997.

[62]: Mark Wood and Mije Erlinger. «Intrusion detection massage exchange requirements». Intrusion Detection Working Group, Internet-Draft. draft-ietf-idwg-requirements-10, Octobre 2002.

[63]: G. Bendall and F. Margot, Greedy Type Resistance of Combinatorial Problems, Discrete Optimization 3 (2006), 288–298.

[64] : Adrien GOÄEFFON, «Nouvelles heuristiques de voisinage et mimétiques pour le problème Maximum de parcimonie», thèse de doctorat : Angers 2006.

[65]: F. Hutter, D. A. D. Tompkins, and H. H. Hoos. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In P. Van Hentenryck, editor, Principles and Practice of Constraint Programming - CP 2002, volume 2470 of LNCS, pages 233-248. Springer Verlag, 2002.

[66]: Etude 2003 du CLUSIF « Club de la sécurité des systèmes d'information français » sur la sinistralité.