N° d'ordre : 07/2007-M/IN

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 diplôme de Magister

En : INFORMATIQUE

Spécialité : Base de données et intelligence artificielle

Par : SAHMADI Brahim

Sujet

Une Approche Mimétique pour la Détection d'Intrusions

Soutenu le 09/07/2007, devant le jury composé de :

Azzoune Hamid, Maître de Conférences (USTHB) ..... Président
Drias Habiba, Professeur (INI/ USTHB) ..... Directrice de mémoire
Baba Ali Riadh, Maître de Conférences (USTHB) ..... Examinateur
Souami Feryal, Maître de Conférences (USTHB) ..... Examinatrice
Boughaci Dalila, Chargée de Cours (USTHB) ..... Invitée

Résumé :

La détection d'intrusions est la capacité d'un système informatique de déterminer automatiquement, à partir d'événements relevant de la sécurité, qu'une violation de sécurité se produit ou s'est produite dans le passé. Pour ce faire, la détection d'intrusions nécessite qu'un grand nombre d'événements de sécurité soient collectés et enregistrés afin d'être analysés.

Il existent deux approches pour la détection d'intrusion : 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. L'approche comportementale permet de détecter des attaques inconnues. L'approche par scénarios consiste à définir des comportements anormaux et ce, afin d'analyser les données susceptibles d'être des attaques. L'approche par scénarios utilise souvent une base de scénarios d'attaques.

Dans ce travail, on s'intéresse à l'approche par scénarios. Notre but est de déterminer les attaques potentiellement présentes dans le fichier d'audit qui contient une masse très importante d'événements.

En effet, le problème d'analyse du fichier d'audit de sécurité est un problème NP-complet, c'est pourquoi nous proposons une approche méta-heuristique basée sur les algorithmes génétiques et le recuit simulé.

L'approche méta-heuristique pour la détection d'intrusions, que nous proposons, permettra de rechercher les scénarios d'attaques prédéfinies dans les traces d'audit. Le but de cette approche est de déterminer la présence d'une ou plusieurs signatures d'attaques dans les données d'audit.

Mots-clés : Détection d'intrusion, approche comportementale, approche par scénarios, algorithme mimétique, sécurité, attaques.

Abstract :

The intrusions detection is the capacity of a computer system to determine automatically, from an audit file that a violation of security occurs or has been occurred in the past. With this intention, the intrusions detection requires that a great number of data, events or users activities are collected and saved in order to be analyzed.

There are two main models of intrusion detection:

- An anomaly detection approach: where the detection is performed by detecting changes in the patterns of utilization or behavior of system and users.
- A misuse detection approach: where detection is performed by exploiting known attacks called signature.

In this work, we are interested in the misuse approach. Our goal is to determine the potentially attacks presented in the audit file which contains a very important mass of events. Indeed, the problem of the audit file analysis is an Np-complete, this is why we propose a metaheuristic approach based on the genetic algorithms and the simulated annealing.

The metaheuristic approach for the intrusions detection, which we propose, will make it possible to seek the attacks scenarios predefined in the audit traces. The approach aim is to determine the presence of an attacks signature in the audit data.

Keywords : Intrusion detection, Misuse Detection, Anomaly Detection, Mimetic algorithm, security, attacks.

ملخص :

كشف التعدي على النظام المعلوماتي هو قدرة هذا النظام على تحديد وبطريقة آلية عن طريق تحليل الأحداث، حصول إنتهاك لأمن النظام المعلوماتي. ولهذا فكشف أي تعدي يتطلب جمع قدر كبير من المعطيات و الأحداث و نشاطات المستعملين، تخزينها و من ثم تحليلها.

هناك طريقتين لكشف التعدي على النظام المعلوماتي :

- طريقة تعتمد على نمذجة السلوك العادي للمستعمل، وأي عمل يخرج عن نطاق هذا السلوك يعتبر نشاط غير عادي.
- الطريقة الثانية تعتمد نمذجة السلوكات غير العادية، و تحليل المعطيات المخزنة لتحديد هل هناك تعدي أم لا.

في هذا العمل نهتم بتطوير طريقة لتحديد الهجومات المحتملة على النظام المعلوماتي من خلال تحليل سجل الأحداث الخاص بالنظام، و الذي يحتوي على كمية كبيرة من المعطيات بحيث يصعب إستعمال خوارزميات البحث العادية.

الطريقة المقترحة تعتمد على البحث عن سيناريوهات الهجومات المعروفة في سجلات أحداث النظام بإستعمال خوارزم يعتمد على تقنيتين للذكاء الإصطناعي، و هما الخوارزميات الجينية و التبريد المقلد.

الكلمات المفتاحية : كشف التعدي، الكشف بالسلوك، الكشف بالسيناريو، الخوارزميات المقلدة، الأمن المعلوماتي، الهجومات.

Remerciements

Mes remerciements sont d'abord au dieu tout puissant de m'avoir donner la force et la patience pour terminer ce travail.

J'exprime ma reconnaissance à Mme Drias. H professeur à l'institut national d'informatique (INI) pour ses conseils et ses orientations durant la réalisation de ce mémoire.

J'exprime ma sincère gratitude à Mlle Boughaci. D pour son aide précieuse, leurs conseils et leur soutien permanant.

Je tiens à remercier Mr Azzoune. H Maître de conférences à l'USTHB, qui me fait l'honneur d'accepter de présider le jury de mémoire.

Je remercie Mr Baba Ali. R Maître de conférences à l'USTHB, ainsi que Mme Souami. F Maître de conférences à l'USTHB, qui ont eu l'amabilité d'accepter de faire partie du jury et d'évaluer ce travail.

Enfin, je remercie tous les membres de ma famille, ainsi mes chers amis pour leurs encouragements et leurs soutien moral.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Les systèmes d'information sont aujourd'hui de plus en plus ouverts sur Internet. Cette ouverture, a priori bénéfique, pose néanmoins un problème majeur : il en découle un nombre croissant d'attaques. Afin de contrer ces attaques, et garantir un niveau élevé de protection du réseau et du système d'informations, on peut utiliser des services, des mécanisme et des outils que l'on nomme, des solutions ou des mesures de sécurité, comme le politique de sécurité, les Firewalls, le contrôle d'accès et les protocoles sécurisés.

Néanmoins, malgré la mise en œuvre des mécanismes préventifs, il est parfois possible pour un attaquant externe, ou même un utilisateur interne mal intentionné, de contourner les mécanismes de contrôle d'accès aux ressources, afin de porter atteinte à la confidentialité, à l'intégrité ou à la disponibilité des services et des données. En effet toutes les statistiques prouvent que le nombre d'intrusions (internes et externes) ne cesse de s'accroître, le tableau Tab.1 montre la croissance du nombre d'incidents de sécurité rapportés annuellement au CERT/CC qui n'incite pas à l'optimisme.

Année	**Incidents**	**Année**	**Incidents**	**Année**	**Incidents**
1988	6	**1994**	2340	**2000**	21756
1989	132	**1995**	2412	**2001**	52658
1990	252	**1996**	2573	**2002**	82094
1991	406	**1997**	2134	**2003**	137529
1992	773	**1998**	3734		
1993	1334	**1999**	9859		

Tab.1 - Nombre d'incidents rapportés au CERT/CC chaque année depuis sa création [Cert]. Un incident peut impliquer un ou de plusieurs sites.

Enfin, pour des raisons à la fois techniques, humaines, et fonctionnelles, il n'est pas possible de concevoir ou de mettre en œuvre un système en garantissant qu'il est exempt de failles. Ces pourquoi il est nécessaire de compléter la politique préventive de sécurité par

des outils de détection d'intrusions visant à surveiller les activités d'un système ou d'un réseau de systèmes, à détecter les usages anormaux des ressources, à journaliser ces événements, à analyser cette information à la recherche de violation ou d'abus, à générer des alertes et parfois, à amorcer certaines actions.

Les systèmes de détection d'intrusion (IDS) sont divisés en deux catégories, suivant l'approche utilisée :

1- 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. L'approche comportementale permet de détecter des attaques inconnues auparavant ainsi que les abus de privilèges des utilisateurs légitimes du système.

Parmi les méthodes proposées pour construire les profils, nous citons :

- **Méthodes statistiques** : le profil est calculé à partir de variables prises aléatoirement et échantillonnées à intervalles réguliers. Ces variables peuvent être, par exemple, la durée et l'heure des connexions, le temps machine utilisé, etc [JVL+93].
- **Systèmes experts** : leur base de règles 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 [VaL 89].
- **Réseaux de neurones** : son principe consiste à apprendre à un réseau de neurones le comportement normal d'un utilisateur. Par la suite, lorsqu'on lui fournira les actions courantes, il devra décider de leur normalité [DBS 92].

2- Approche par scénarios : consiste à définir des comportements anormaux et ce, afin d'analyser les données susceptibles d'être des attaques. L'approche utilise souvent une base de scénarios d'attaques.

Parmi les méthodes proposées analyser le fichier de l'audit de sécurité, nous citons:

- **Systèmes experts** : leur base de règles décrit les attaques. Les événements d'audit sont traduits en des faits [LuJ 88].
- **Algorithmes génétiques** : Chaque individu de la population code un sous-ensemble particulier d'attaques qui sont présentes dans les traces d'audit. Cette

approche permet d'optimiser de temps de recherche dans le journal d'audit [Mé 98].

- **Pattern matching** : Des signatures d'attaques sont fournies. Divers algorithmes sont utilisés pour localiser ces signatures dans les traces d'audit. Pour plus d'information, veuillez consulter la référence [DDW 98].

Dans notre travail, nous utilisons l'approche par scénario pour développer un système de détection d'intrusion, puisque on s'intéresse à des failles qui sont déjà déterminées. Cette méthode a été choisie pour remédier aux inconvénient dus à l'approche comportementale qui n'est pas prouvée exacte car on obtient beaucoup de fausses alertes, c'est-à-dire que le système croit être attaqué alors qu'il ne l'est pas.
Contrairement à l'approche comportementale, l'approche par scénarios peut prendre en compte les comportements exacts des attaques potentielles. Les inconvénients sont dans la base des attaques qui doit être bien construite et les performances des signatures d'attaques qui sont limitées par l'esprit humain qui les a conçues.

Notre but est de déterminer les attaques potentiellement présentes dans le fichier d'audit qui contient une masse très importante d'événements.

Le problème d'analyse du fichier d'audit de sécurité est un problème NP-complet, et les temps de traitement nécessaires à sa résolution ne permettent pas d'envisager une solution algorithmique non polynomiale dans les cas réels [Mé98]. C'est pourquoi nous utilisons une approche méta-heuristique.

L'approche méta-heuristique pour la détection d'intrusions permettra de rechercher les scénarios d'attaques pré-définies dans les traces d'audit. Le but de cette approche est de déterminer la présence d'une ou plusieurs signatures d'attaques dans les données d'audit, et ce même si le nombre de signatures d'attaques est important et que les données d'audit sont très volumineuses.

Les scénarios d'attaques utilisées par la méthode de recherche proposée sont définis dans une matrice décrivant les événements que génère chaque attaque sans tenir compte de l'ordre d'apparition des événements.

La méthode de recherche proposée se base sur un algorithme évolutionnaire hybride, constituer d'un algorithme génétique étendu par une recherche locale basée sur le principe de recuit simulé.

Notre document sera organisé comme suit :

Le premier chapitre est consacré aux généralités et aux principaux concepts du domaine de la sécurité informatique.

Dans le deuxième chapitre, nous présentons les systèmes de détection d'intrusions.

Nous décrivons au troisième chapitre les algorithmes génétiques, le recuit simulé et les algorithmes mimétiques que nous allons utiliser pour l'analyse du fichier d'audit de sécurité.

Le quatrième chapitre propose une solution pour le problème de l'analyse du fichier d'audit de sécurité. Une approche mimétique a été conçue et mise en œuvre.

Au cinquième chapitre, nous exposons les différents tests numériques réalisés et les résultats obtenus.

Enfin, la conclusion générale porte sur les résultats obtenus et dresse quelques perspectives sur la suite de notre travail.

Une annexe est ajoutée à la fin pour élucider et détailler les attaques mentionnées dans le chapitre 5.

Chapitre I

La sécurité informatique

I.1. Introduction :

Faire de la sécurité sur un système informatique consiste à s'assurer que celui qui modifie ou consulte des données du système en a l'autorisation et qu'il peut le faire correctement car le service est disponible.

Dans le but de renforcer la sécurité du système informatique, plusieurs outils ont été développés. Parmi ces outils, les firewalls et les systèmes de détection d'intrusions.

Dans ce chapitre nous donnons, quelques définitions utiles sur la sécurité informatique, les risques informatiques, les mesures de sécurité etc. Ensuite nous abordons un type de mécanismes de sécurité informatique qui sont les systèmes de détection d'intrusions.

I.2. Définition de la sécurité informatique :

La sécurité d'un système informatique est la capacité du système informatique à résister à des agressions externes physiques (incendie, inondation, etc.) ou logiques (erreurs de saisie, intrusions, piratages, logique malicieuse, etc.).

La sécurité d'un système informatique consiste à préserver l'intégrité, la disponibilité et la confidentialité de ce système. Dans ce sens, des mesures et moyens doivent être appliqués pour minimiser les risques volontaires ou non volontaires [Maiw01].

Parmi les actions menaçant la sécurité des informations, nous avons :

- Accès aux informations confidentielles.
- Destruction de données.
- Détérioration, fraude et perte des informations.

Du point de vue organisationnel, la sécurité informatique peut toucher les domaines suivants :

- La sécurité logicielle : gère la sécurité au niveau logiciel du système d'information, et qui intègre des protections logicielles comme les antivirus.
- La sécurité du personnel : comprend la formation et la sensibilisation des personnes utilisant ou travaillant avec le système d'information.
- La sécurité physique : consiste à prévoir une sécurisation des locaux, une politique d'accès aux matériels informatiques, et des règles de sécurité pour la protection des équipements réseaux.
- La sécurité procédurale : consiste à définir des procédures et des règles d'utilisation du système informatique.
- La sécurité réseau : s'occupe de l'architecture physique et logique de réseau, la politique d'accès aux différents services, la gestion des flux d'informations sur les réseaux, et surtout les points de contrôle et de surveillance du réseau.
- La veille technologique : qui permet d'évaluer la sécurité au cours du temps afin de maintenir un niveau suffisant de protection du système informatique.

Les objectifs majeurs à prendre en compte en matière de sécurité des données sont les suivants : "la disponibilité", "l'intégrité" et "la confidentialité" de l'information (voir la figure fig.I.1) [Fesc05] .Toutefois, cela reste un problème complexe, car aucun moyen n'est encore capable de garantir une sécurité absolue (à 100%).

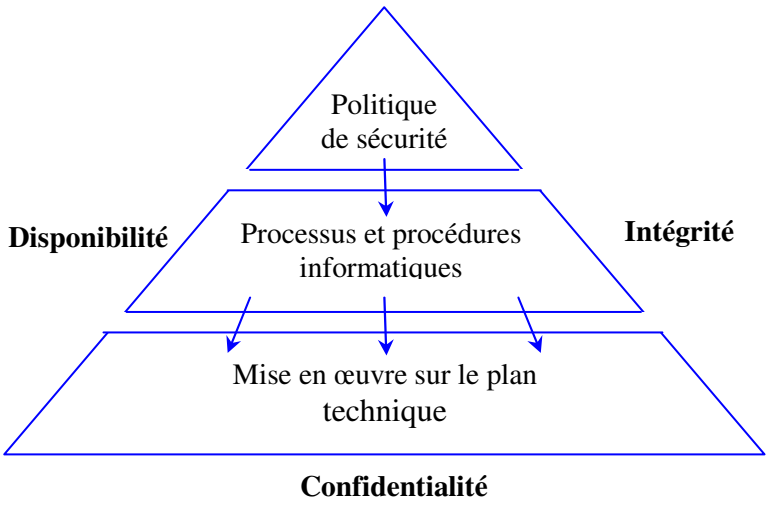

Fig.I.1 - Les trois objectifs majeurs de la sécurité informatique

I.3. Terminologie de la sécurité informatique :

La sécurité informatique utilise un vocabulaire bien défini, qu'on va définir quelques termes:

- *Les vulnérabilités* : ce sont des failles de sécurité dans un système, ou des fautes accidentelles ou intentionnelles (avec ou sans volonté de nuire), introduites dans la spécification, la conception ou la configuration du système. Tout système vu dans sa globalité présente des vulnérabilités, qui peuvent être exploitables ou non.
- *Une attaque (exploit)* : Une attaque peut être définie comme toute action ou ensemble d'actions qui peut porter atteinte à la sécurité des informations d'un système ou réseau informatique. Elle représente le moyen d'exploiter une vulnérabilité. Il peut y avoir plusieurs attaques pour une même vulnérabilité.
- *Une intrusion* : est définie comme étant une faute opérationnelle, externe, intentionnellement nuisible, résultant de l'exploitation d'une vulnérabilité dans le système [Post03]. L'usage courant du mot intrusion couvre le fait de pénétrer illégalement ou sans y être convié dans un lieu.

  Il existe toujours deux causes sous-jacentes à une intrusion :

  - Une action malveillante ou attaque qui tente d'exploiter une faiblesse dans le système et de violer un ou plusieurs besoins de sécurité ;
  - Au moins une faiblesse, faille ou vulnérabilité, qui est une faute accidentelle ou intentionnelle (avec ou sans volonté de nuire), introduite dans la spécification, la conception ou la configuration du système.
- *Les contre mesures* : ce sont les procédures ou techniques permettant de résoudre une vulnérabilité ou de contrer une attaque spécifique.
- *Les menaces* : ce sont des adversaires déterminés capables de monter une attaque exploitant une vulnérabilité.

I.4. Les services de sécurité :

Une des principales vocations d'un système informatique est de contrôler si la fonction de sécurité est prise en compte. Cette fonction doit prendre en considération les préoccupations qui suivent: l'intégrité, la confidentialité, la disponibilité, l'authentification et la non répudiation. Ces derniers sont définis comme suit:

I.4.1. *Confidentialité* :

La confidentialité est la propriété d'une information de ne pas être révélée à des utilisateurs non autorisés à la connaître. Ceci signifie que le système informatique doit :

- empêcher les utilisateurs de lire une information confidentielle (sauf s'ils y sont autorisés), et
- empêcher les utilisateurs autorisés à lire une information et de la divulguer à d'autres utilisateurs (sauf autorisation) [Abka03].

I.4.2. *Intégrité :*

Assurer que les données ne seront pas altérées (intentionnellement ou non) pendant leur transmission ou leur stockage [Chva95]. 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, et
- faire en sorte qu'aucun utilisateur ne puisse empêcher la modification légitime de l'information.

I.4.3. *Disponibilité* :

La disponibilité est la propriété d'une information d'être accessible lorsqu'un utilisateur autorisé en a besoin. Cela signifie que le système informatique doit :

- fournir l'accès à l'information pour que les utilisateurs autorisés puissent la lire ou la modifier, et
- faire en sorte qu'aucun utilisateur ne puisse empêcher les utilisateurs autorisés d'accéder à l'information [Abka03].

I.4.4. *Authentification / Identification* :

Avant même de vérifier qu'une entité est autorisée à accéder à un objet, le système doit être sûr de l'identité de l'entité. On peut dire que l'authentification est la première étape de sécurité afin de protéger un système informatique [Chva95].

I.4.5. *Non répudiation* :

Par définition la répudiation est la possibilité, pour une des entités impliquées dans la communication, de nier avoir participe aux échanges, totalement ou en partie. Donc La non répudiation est alors l'impossibilité de nier la participation à une communication. Par suite, ce service permettant 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.

I.5. Les attaques informatiques :

Les attaques peuvent porter sur les communications, les machines, les traitements, les personnels et l'environnement. Elles sont à classer en deux catégories attaques physiques et logiques.

I.5.1. Attaques physiques :

Dans les attaques physiques nous citons :

a. *Interception* **:** Récupération d'un signal électromagnétique. Après interprétation, il est possible d'en déduire des informations.

b. *Brouillage* **:** Utilisé en télécommunication, il rend inopérant le système d'information.

c. *Ecoute* **:** Technique traditionnelle qui consiste à se placer sur un réseau, à analyser et sauvegarder les informations qui y transitent.

d. *Balayage* **:** Envoi, au système d'information, d'un panel d'informations pour déterminer celles qui suscitent une réponse.

e. *Piégeage* **:** À la conception, à la maintenance, seule une évaluation permettra la détection.

I.5.2. Attaques logiques :

Voici quelques exemples d'attaques logiques :

a. *Déguisement* **:** Le déguisement est le procède par lequel une entité se fait passer pour une autre. Le déguisement est en général utilise avec d'autres formes d'attaques actives. Par exemple, une entité autorisée ayant peu de privilèges peut utiliser un déguisement pour obtenir des privilèges supplémentaires en usurpant l'identité d'une entité qui a ces privilèges.

b. *Logiques malignes* **:** Les logiques malignes recouvrent des attaques dues à des fautes de développement humaines comme les chevaux de Troie, les portes dérobées, et les bombes logiques, ou dues à des fautes opérationnelles humaines comme les virus et les vers.

I.6. Les mécanismes de sécurité :

Afin de protéger le système informatique, des mécanismes de sécurité ont été proposés. Parmi ces mécanismes nous citons :

I.6.1. 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.

La notion de contrôle d'accès englobe les fonctions suivantes [Lalet89]

- Les fonctions permettant de contrôler les flux d'informations,
- Les fonctions d'administration, c'est à dire l'octroi et le retrait, des droits d'accès et leur vérification,
- Les fonctions servant à établir et entretenir toutes les listes ou règles qui régissent les droits d'effectuer différents types d'accès.

Parmi les moyens de contrôle d'accès on trouve *Identification et Authentification :*

L'authentification a pour but de vérifier l'identité dont une entité se réclame. Généralement l'authentification est précédée d'une identification qui permet à cette entité de se faire reconnaître du système par un élément dont on l'a doté.

L'Identification et l'Authentification répondent à des exigences de détermination et de contrôle des utilisateurs qui sont autorisés à avoir accès aux ressources.

La notion d'identification et d'authentification rassemble les fonctions suivantes :

- Le contrôle des utilisateurs qui accèdent à une ressource,
- L'établissement de l'identité annoncée par un utilisateur et la vérification que cet utilisateur est bien la personne qu'il prétend être,
- L'ajout de nouvelles identités et l'élimination ou l'invalidation d'anciennes identités,
- La création et la modification des informations, permission données aux utilisateurs autorisés de contrôler les informations d'authentification nécessaires pour vérifier l'identité d'utilisateurs particuliers,
- L'intégrité des informations d'authentification et la prévention d'un usage non autorisé,
- La limitation des possibilités d'essais répétés d'établissement d'une fausse identité.

I.6.2. La cryptographie :

La cryptographie est à la base de nombreux service et mécanismes de sécurité. Le chiffrement, utilisé pour assurer la confidentialité, transforme les données sensibles (les données à protégées) en données moins sensibles. Pour l'intégrité ou l'authentification, les techniques cryptographiques sont utilisées pour traiter des fonctions non falsifiables.

I.7. Quelques outils de sécurité :

Pour renforcer la sécurité des systèmes informatiques, des outils ont été développés. Parmi ces outils on trouve :

I.7.1. L'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.

I.7.2. 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.

I.7.3. Le Firewall :

Un pare-feu (Firewall en anglais), est un système physique (matériel) ou logique (logiciel) servant d'interface entre un ou plusieurs réseaux afin de contrôler et éventuellement bloquer la circulation des paquets de données, en analysant les informations contenues dans les couches 3, 4 et 7 du modèle OSI. Il s'agit donc d'une machine (machine spécifique dans le cas d'un Firewall matériel ou d'un ordinateur sécurisé hébergeant une application particulière de pare-feu comportant au minimum deux interfaces réseau [Firew]:

- Une interface pour le réseau à protéger (réseau interne)
- Une interface pour le réseau externe

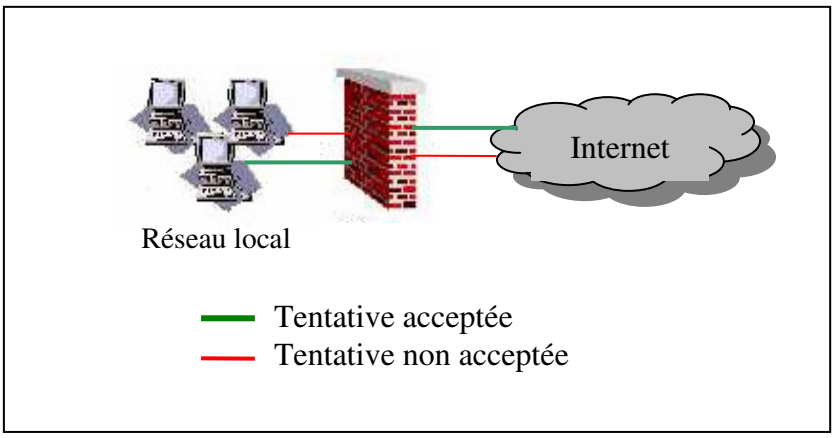

Fig.I.2 - Le FireWall

Le pare-feu représente ainsi généralement dans les entreprises un dispositif à l'entrée du réseau qui permet de protéger le réseau interne d'éventuelles intrusions en provenance des réseaux externes (souvent l'Internet).

I.7.4. Le système de détection d'intrusions (IDS : Intrusion Detection System) :

Système logiciel ou matériel, qui permet de détecter en temps réel ou différé les tentatives d'intrusion sur un réseau interne ou sur un seul ordinateur hote, de neutraliser ces attaques réseaux ou systèmes et d'assurer ainsi la sécurité du réseau. Deux méthodes sont principalement utilisées par les systèmes de détection d'intrusion : la reconnaissance de signatures et la détection d'anomalies. La reconnaissance de signatures est une approche consistant à rechercher dans l'activité de l'élément surveille les signatures (ou empreintes) d'attaques connues. Le système de détection d'intrusion fait appel à une bibliothèque de signatures et ne petit alors détecter que les attaques dont il possède la signature. De sont coté, la détection d'anomalies utilise l'analyse de statistiques du système : changement de mémoire, utilisation excessive du CPU,... etc. le système de détection signalera les divergences par rapport au fonctionnement normal (ou de référence) des éléments surveilles. Contrairement au pare-feu, qui traite des requêtes et les interdit, un système de

détection d'intrusion les analyse de façon continue et ne réagit qu'en cas d'anomalies [Guill].

I.8. Conclusion :

Le domaine de la sécurité informatique est très vaste, et très difficile à cerner par quelques définitions. En général la sécurité informatique peut être mesurée par le niveau de confiance donné à la confidentialité, l'intégrité, et la disponibilité de l'information.

Dans ce chapitre on a vu brièvement les principaux services de sécurité : la confidentialité, l'intégrité, la disponibilité, l'identification et la non répudiation, et pour les mécanismes et les outils permettant d'assurer ces services on a cité quelques outils tel que les antivirus, les scanners de vulnérabilité, les firewalls et les systèmes de détection d'intrusions.

Dans notre travail, on s'intéresse aux systèmes de détection d'intrusions qui seront détaillé dans le chapitre suivant.

Chapitre II

Les systèmes de détection d'intrusions

II.1. Introduction :

Le but de la sécurité informatique est de créer un système complètement sécurisé. Mais il est très rarement possible de rendre un système complètement inattaquable pour plusieurs raisons :

° La plupart des systèmes informatiques ont des failles de sécurité qui les rendent vulnérables aux intrusions. Les trouver et les réparer toutes n'est pas possible pour des raisons techniques et économiques.

° Les systèmes existants ayant des failles connues ne sont pas facilement remplacés par des systèmes plus surs, principalement parce qu'ils ont des fonctionnalités intéressantes que n'ont pas les systèmes plus surs, ou parce qu'ils ne peuvent pas être remplacés pour des raisons économiques.

° Déployer des systèmes sans failles est très dur voire impossible car des failles sont inconnues ou inévitables

° Même les systèmes les plus sûrs sont vulnérables aux abus de la part d'utilisateurs légitimes qui profitent de leurs privilèges, ou souffrent de la négligence des règles de sécurité par ceux-ci.

En réponse à ces difficultés pour développer des systèmes sécurisés, et pour être plus réaliste, la prévention n'est qu'une partie de la gestion de la sécurité d'un système

informatique ajouter à elle une partie de détection. Ce qui font les systèmes de détection d'intrusions (IDS : intrusion detection system).

Dans ce qui suit, nous présentons quelques définitions des termes utilisés dans le domaine de la détection d'intrusions, un modèle générique d'un processus de détection, une classification des IDS, le système d'audit et les activités liées à l'audit de sécurité, et nous donnons les limites des systèmes de détection d'intrusions.

II.2. Définitions :

a) *Intrusion* : c'est toute utilisation d'un système informatique à des fins autres que celles prévues, généralement dues à l'acquisition de privilèges de façon illégitime. L'intrus est généralement vu comme une personne étrangère au système informatique qui a réussi à en prendre le contrôle, mais les statistiques montrent que les utilisations abusives proviennent le plus fréquemment de personnes internes ayant déjà un accès au système. En générale une intrusion est une action (ou tentative d'action) qui a pour conséquence de compromettre l'intégrité, la confidentialité ou la disponibilité d'une ressource.

b) *Mécanisme d'audit* **:** nous appellerons mécanisme d'audit toute partie de code du système informatique dont le but est de reporter des informations sur les opérations qu'il lui est demandé d'accomplir.

c) Événement : nous appellerons événement toute opération élémentaire de système reportée par un mécanisme d'audit.

d) *Détection d'intrusion* : la détection d'intrusions consiste à analyser les informations collectées par les mécanismes d'audit de sécurité, à la recherche d'éventuelles attaques. Les méthodes de détection d'intrusion diffèrent sur la manière d'analyser le journal d'audits.

e) *IDS* : (Inrusion Detection System), ou système de détection d'intrusions est un système logiciel ou matériel conçu afin de pouvoir automatiser la surveillance d'événements survenant dans un système informatique, et de pouvoir signaler à l'administrateur système, toute trace d'activité anormale sur ce dernier.

f) *Faux positif* : est le cas où une intrusion est signalée (détectée) mais qu'il n'y a pas attaque; c'est typiquement une fausse alerte (l'IDS fait mal son travail).

g) *Faux négatif* : est un cas d'attaque réelle non détectée; dans ce cas là on considère que l'IDS ne fait pas son travail. La figure Fig.II.1 montre où se situent les faux-positifs et fauxs-négatifs par rapport aux activités système et aux capacités de détection de l'IDS [Frme02].

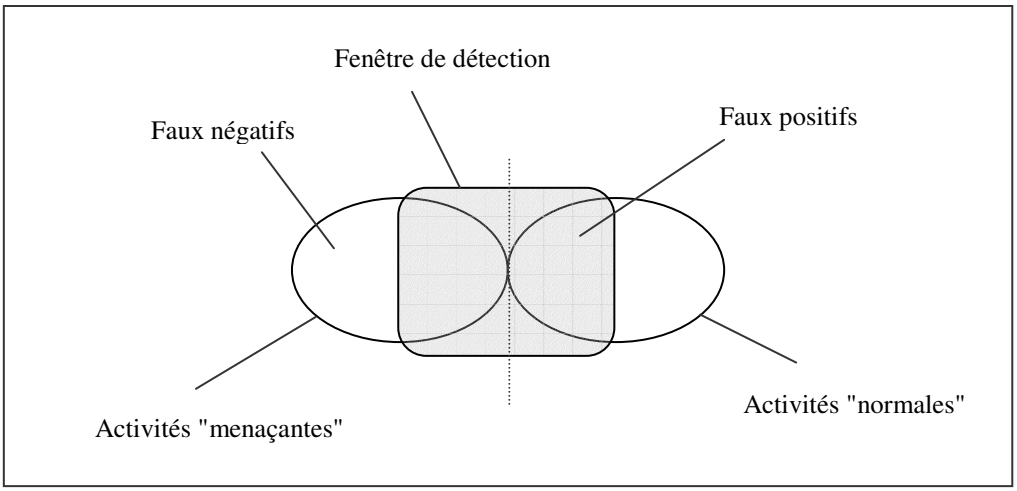

Fig.II.1 - Faux-positifs et Faux-négatifs

II.3. Modèle de processus de détection d'intrusion :

La majorité des systèmes de détection d'intrusions peuvent être décrits par le modèle proposé par l'IDWG (Intrusion Detection Working Group de l'IETF), qui représente un modèle générique de processus de détection d'intrusion (voir figure Fig.II.2) [Woerl02].

Les différents éléments de ce modèle sont les suivants :

° ***Source de données*** : 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.

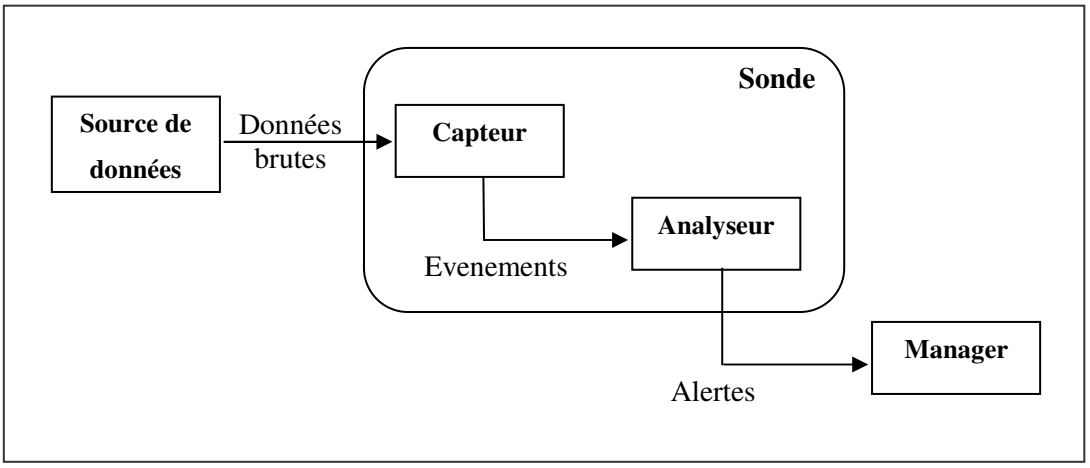

Fig.II.2 - Le modèle générique de la détection d'intrusions proposé par le groupe IDWG (Intrusion Detection Working Group)

° ***Capteur*** : dispositif générant des événements en filtrant et en formatant les données brutes provenant d'une source de données.

° ***Evénement*** : 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*** : dispositif analysant les événements à la recherche de traces d'intrusions.

° ***Alerte*** : message formaté émis par un analyseur s'il trouve des traces d'intrusions.

° ***Sonde*** : ensemble constitué d'un capteur et d'un analyseur.

° ***Manager*** : composant permettant à l'administrateur du système de configurer les différents éléments (capteur, analyseur) et de gérer les alertes reçues.

Il faut noter que le modèle de la figure Fig.II.2 est par nature récursif. En effet, un manager peut être considéré par un capteur comme une source de données [Mé03].

II.4. Une classification des systèmes de détection d'intrusions :

Le nombre de systèmes de détection d'intrusions existants ou ayant existé étant important, nous ne visons pas ici à l'exhaustivité. Nous exposons seulement leurs caractéristiques communes.

Nous présentons une classification selon différents critères qui ne sont pas forcément mutuellement exclusifs : par exemple, un système de détection d'intrusions (IDS) peut mettre en œuvre deux analyseurs utilisant des méthodes différentes.

La classification adoptée n'est pas hiérarchique : elle présente tour à tour et au même niveau les catégories caractérisant chaque IDS. Elle utilise les critères suivants (voir la figure Fig.II.3) :

- la source des données à analyser,
- la méthode de détection utilisée,
- le lieu de l'analyse des données,
- la fréquence de l'analyse,
- le comportement en cas d'attaque détectée.

II.4.1. Source des données à analyser :

Les sources possibles de données à analyser sont une caractéristique essentielle des systèmes de détection d'intrusions puisque ces données constituent la matière première du processus de détection. Les données proviennent soit de logs générés par le système d'exploitation, soit de logs applicatifs, soit d'informations provenant du réseau, soit encore d'alertes générées par d'autres IDS.

II.4.1.1 *Source d'information système :*

Un système d'exploitation fournit généralement plusieurs sources d'information :

− Commandes systèmes : presque tous les systèmes d'exploitation fournissent des commandes pour avoir un "instantané" de ce qui se passe. Ainsi, sous UNIX, des commandes telles que "ps" ou "vmstat" fournissent des informations précises sur les activités courantes du système.

− Accounting : l'accounting fournit de 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 [DBS92] utilisent cette source d'information.

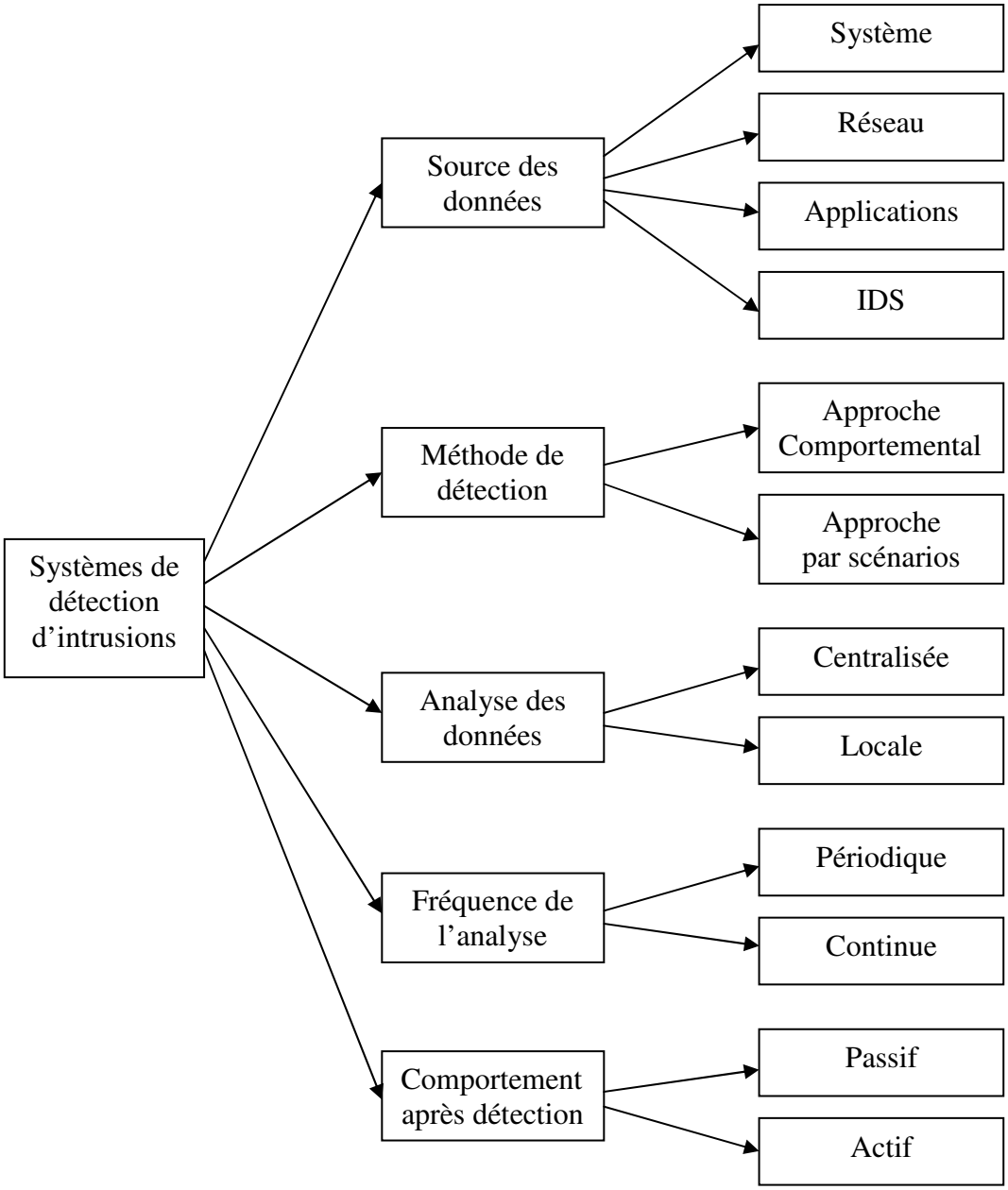

Fig.II.3 - Caractéristiques des systèmes de détection d'intrusions

– Audit de sécurité : tous les systèmes d'exploitation modernes proposent ce service pour fournir des événements système, les associer à des utilisateurs et assurer leurs collecte dans un fichier d'audit. On peut donc potentiellement disposer d'informations sur tout ce que font (ou ont fait) les utilisateurs : accès en lecture à un fichier, exécution d'une application, etc.

Par exemple, l'audit BSM [Sun] représente les appels systèmes produits par les programmes qui s'exécutent sur un système Solaris.
Les outils utilisant cette source de données sont appelés Host Based Intrusion Detection Systems (HIDS).

II.4.1.2 *Source d'information réseau :*

Des dispositifs matériels ou logiciels (snifiers) permettent de capturer le trafic réseau. Cette source d'information est particulièrement adaptée lorsqu'il s'agit de rechercher les attaques en déni de service qui se passent au niveau réseau ou les tentatives de pénétration à distance. Le processus d'interception des paquets peut être rendu quasiment invisible pour l'attaquant car on peut utiliser une machine dédiée juste reliée à un brin du réseau, configurée pour ne répondre à aucune sollicitation extérieure et dont personne ne soupçonnera l'existence. Néanmoins, il est difficile de garantir l'origine réelle de l'attaque que l'on a détectée car il est facile de masquer son identité en modifiant les paquets réseau. De plus, l'utilisation du chiffrement peut rendre impossible la détection si elle base sur le contenu utile des paquets (les données brutes sans les entêtes).
Presque tous les outils commerciaux récents utilisent les informations issues du réseau. Les outils utilisant cette source de données sont appelés Network-based Intrusion Detection systems (NIDS).

II.4.1.3 *Source d'information applicative :*

Les applications peuvent également constituer une source d'information pour les IDS [Sie99]. Les capteurs applicatifs sont de deux natures [Cedr03]:

- Capteur interne : le filtrage sur les activités de l'application est alors exécuté par le code de l'application.
- Capteur externe : le filtrage se fait à l'extérieur de l'application. Plusieurs méthodes sont utilisées : un processus externe peut filtrer les logs produits par l'application ou bien l'exécution de l'application peut être interceptée (au niveau de ses appels de librairies ou d'un Proxy applicatif (ex : Netsecure Web [Cal03])).

II.4.1.4 *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 [Cedr03].

II.4.2. Méthodes de détection :

Les deux approches (permettant de classer les méthodes de détection) proposées à ce jour sont *l'approche comportementale* (anomaly detection) et *l'approche par scénarios* (misuse detection ou knowledge-based detection). La première se base sur l'hypothèse que l'on peut définir un comportement "normal" de l'entité à surveiller (utilisateur, service, application ...) 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 occurrence.

II.4.2.1 *L'approche comportementale :*

La détection d'anomalies consiste à définir, dans une première phase, un certain comportement du système, des utilisateurs, des applications, etc. considéré comme «normal» (appelé profil); dans une seconde phase, le système de détection d'intrusions compare l'activité courante à ce profil. Tout comportement déviant est alors considéré intrusif. Cela permet potentiellement de détecter des attaques inconnues auparavant [Zimé02].

On peut distinguer deux catégories de profils :

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 (exemple : 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 [JVL+93] 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 [DBS92] 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 [Cedr03].

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

Pour 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 [ZMB02], 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.

II.4.2.2 *L'approche par scénarios :*

On construit des scénarios d'attaque en spécifiant ce qui est caractéristique de l'attaque et qui doit être observé dans les traces d'audit. L'analyse des traces d'audit se fait à la recherche de ces scénarios. Les méthodes proposées à ce jour sont les suivantes :

- **Système expert** : le système expert comporte une base de règles qui décrit les attaques. Les événements d'audit sont traduits en des faits qui sont interprétables par le système expert. Son moteur d'inférence décide alors si une attaque répertoriée s'est ou non produite.
- **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. Plusieurs IDS commerciaux (tels que Realsecure [Sys98] ou NetRanger [Cis98]) utilisent cette méthode.
- **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 [VK99] utilise des

diagrammes de transitions d'états (State Transition Diagrams) pour représenter les scénarios d'attaques.

L'approche par scénarios permet de savoir précisément quelle attaque a été détectée. Néanmoins, elle ne permet de détecter que des attaques connues ce qui oblige à maintenir à jour la base de scénarios d'attaque [Mécéd99].

II.4.3. 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 :

- ***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.
- **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.

II.4.4. Fréquence de l'analyse :

Une autre caractéristique des systèmes de détection d'intrusions est leur fréquence d'utilisation :

- ***Périodique*** : certains systèmes de détection d'intrusions analysent périodiquement les fichiers d'audit à la recherche d'une éventuelle intrusion ou anomalie passée. Cela peut être suffisant dans des contextes peu sensibles (on fera alors une analyse journalière, par exemple).
- ***Continue*** : la plupart des systèmes de détection d'intrusions récents effectuent leur analyse des fichiers d'audit ou des paquets réseau de manière continue afin de proposer une détection en quasi temps réel. Cela est nécessaire dans des contextes sensibles

(confidentialité) et/ou commerciaux (confidentialité, disponibilité). C'est toutefois un processus coûteux en temps de calcul car il faut analyser à la volée tout ce qui se passe sur le système.

II.4.5. Comportement après détection

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

- ***Passive*** : la plupart des systèmes de détection d'intrusions n'apportent qu'une réponse passive à l'intrusion. Lorsqu'une attaque est détectée, ils génèrent une alarme et notifient l'administrateur système par e-mail, message dans une console, voire même par beeper. C'est alors lui qui devra prendre les mesures qui s'imposent.
- ***Active*** : d'autres systèmes de détection d'intrusions peuvent, en plus de la notification à l'opérateur, prendre automatiquement des mesures pour stopper l'attaque en cours. Par exemple, ils peuvent couper les connexions suspectes ou même, pour une attaque externe, reconfigurer le pare-feu pour qu'il refuse tout ce qui vient du site incriminé. Des outils tel que RealSecure [Sys98] proposent ce type de réaction. Toutefois, il apparaît que ce type de fonctionnalité automatique est potentiellement dangereux car il peut mener à des dénis de service provoqués par l'IDS. Un attaquant déterminé peut, par exemple, tromper l'IDS en usurpant des adresses du réseau local qui seront alors considérées comme la source de l'attaque par l'IDS. Il est préférable de proposer une réaction facultative à un opérateur humain (qui prend la décision finale).

II.5. L'audit de sécurité :

L'audit de sécurité permet d'enregistrer tout ou partie des actions effectuées sur le système. L'analyse de ses informations permet de détecter d'éventuelles intrusions. Les systèmes d'exploitation, et même certaines applications disposent généralement de systèmes d'audit intégrés. Les différents évènements du systèmes sont enregistrés dans un journal d'audit qui devra être analysé fréquemment, voire en permanence.

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'audi*t [Méal96].

II.5.1. Activités liées à l'audit de sécurité :

Le journal d'audit doit permettre d'enregistrer toutes les opérations liées à la sécurité, que ces opérations soient réussies (parce qu'autorisées) ou qu'elles aient échouées (empêchées par les mécanismes de contrôle d'accès). Les principales opérations à surveiller sont [Boud00]:

- 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.

Ils 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.).

II.5.1.1. *Spécification des activités système à auditer* :

Selon le niveau de sécurité qu'il souhaite atteindre, l'officier de sécurité définira des évènements auditables correspondant à certaines informations. [Méal96] 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 :

C'est bien sûr un point très sensible puisque les fichiers contiennent l'information. 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. 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).

II.5.1.2. *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 c'est les sondes réseaux, qui permettent de récupérer

les données du trafic réseau (les paquets circulant sur le réseau) comme l'outil "TCP Dump".

II.5.1.3. *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'officier 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 les accès aux données.

II.6. Les critères de bon fonctionnement d'un IDS :

Les systèmes de détection d'intrusions sont réellement devenus indispensables lors de la mise en place d'une infrastructure de sécurité opérationnelle. Un bon système de détection d'intrusions, quelque soit son architecture, doit :

- tourner en permanence sans superviseur humain.
- être tolérant aux fautes et résister aux attaques.
- utiliser un minimum de ressources du système surveillé
- être fiable : les alertes qu'il génère doivent être justifiées et aucune intrusion ne doit pouvoir lui échapper. Un IDS générant trop de fausses alertes sera a coup sur désactivé par l'administrateur et un IDS ne détectant rien sera rapidement considère comme inutile.

- être capable de détecter les nouveaux types d'attaque le plus rapidement possible; pour cela il doit rester constamment à jour. Des capacités de mise à jour automatique sont pour ainsi dire indispensables.
- être facile à mettre en oeuvre et doit pouvoir surtout s'adapter au contexte dans lequel il doit opérer.
- donner un maximum d'information sur l'attaque détectée afin de préparer la réaction.

II.7. Les limites des systèmes de détection d'intrusion :

La plupart des systèmes de détection d'intrusions actuels on un comportement passif et se contentent d'une alarme à l'administrateur du réseau et ne peuvent pas appliquer un scénario de défense.

Les traces d'une attaque simple sont souvent identifiables à partir d'un seul fichier log ou d'une seule carte réseau. Cependant des attaques plus élaborées ne sont pas concentrées sur une machine unique, mais tentent au contraire de dissimuler leurs actions en les distribuant sur plusieurs hôtes. Ainsi chaque action apparaît comme bénigne, alors que leur ensemble constitue une attaque. Il est donc nécessaire de collecter et de corréler les données d'audit provenant de différentes sources et de dialoguer si nécessaire avec chacune d'entre elles.

La plupart des IDS effectuent le traitement des données de manière centralisée, bien que la collecte des données soit distribuée. Ceci introduit des limitations en termes de scalabilité, configurabilité et de tolérance aux pannes. En effet, une défection de l'unité centrale désactive tout l'IDS. De plus, sa capacité de traitement limite le nombre d'événements qu'il peut prendre en compte sur un intervalle de temps donné. Ainsi, un trop grand flux de messages peut entraîner une augmentation du temps de réaction du système ou une perte de données.

La scalabilité est donc limitée, car le traitement de la totalité des données d'audit par une seule unité impose une limite sur la taille du réseau. Au delà de cette limite, l'analyseur central n'est plus capable de prendre en charge le flux de données. Notons aussi que la collecte des données peut générer une surcharge du réseau. Par ailleurs, la centralisation rend difficile la reconfiguration ou l'accroissement de la capacité de l'IDS.

En pratique, l'IDS en seul bloc pose des problèmes de mise à jour, et il constitue un point d'attaque unique. Il va donc s'agir de distribuer les fonctionnalités de l'IDS sur plusieurs machines.

On peut résumer les défauts les plus courants des IDS dans :

- Manque d'efficacité;
- Taux élevé de fausses détections
- Maintenance pénible
- Flexibilité limitée
- Vulnérabilité aux attaques directes
- Capacité de réponse limitée
- Pas de méthodologie générique pour leur conception.

II.8. Conclusion :

Le système de détection d'intrusions représente une composante indispensable de la sécurité des systèmes informatiques, car on ne peut pas assurer la sécurité totale d'un système avec seulement des outils préventifs comme les firewalls.

Les outils de détection d'intrusions actuels se basent en général sur deux approches : l'approche comportementale et l'approche par scénarios, et l'implémentation de la majorité de ces outils est basée sur des techniques d'intelligence artificielle.

Notre travail consiste à proposer une méthode de recherche de scénarios d'attaques dans les traces d'audit on utilisant des algorithmes heuristiques dits "les algorithmes mimétiques". La présentation des notions associées à ces algorithmes sera l'objet du chapitre suivant.

Chapitre III

Les algorithmes mimétiques

III.1. Introduction :

Dans ce chapitre, nous présentons un état de l'art des algorithmes mimétiques, qui sont une famille des méta-heuristiques évolutives utilisées pour la résolution des problèmes d'optimisation combinatoire. On débutera par un aperçu sur les algorithmes génétiques et les principes reliés à ce type d'algorithme tel que la population, le codage des individus, la fonction objective, et les opérateurs génétiques : sélection proportionnelle, crossover et mutation. Ensuite on va voir la recherche locale et précisément la technique de recuit simulé. Et enfin nous présentons l'approche mimétique (hybride) et nous définissons un algorithme mimétique basé sur la combinaison d'un algorithme génétique avec une technique de recherche locale qui est le recuit simulé, cet algorithme sera utilisé dans la suite de notre travail.

III.2. Les algorithmes génétiques :

III.2.1. Les origines :

Les algorithmes génétiques représentent une famille très intéressante d'algorithmes d'optimisation, ils sont inspirés de la génétique et de la théorie de la sélection naturelle citée par (Charles Darwin). Leur développement a pour but de modéliser des systèmes adaptatifs naturels et de construire des systèmes artificiels dotés des mêmes propriétés.

Principalement les algorithmes génétiques ont été développés pour la première fois par «John Holland » et ses collègues du l'université de Michigan qui ont, dès 1960, travaillé sur ce sujet. La nouveauté introduite par ce groupe de chercheurs a été la prise en compte de l'opérateur de "Crossing over" en complément des mutations. Et c'est cet

opérateur qui permet le plus souvent de se rapprocher de l'optimum d'une fonction en combinant les gènes contenus dans les différents individus de la population. Le premier résultat de ces recherches a été la publication en 1975 de "Adaptation in Natural and Artificial System" [Wiki].

III.2.2. Analogie avec la biologie :

Les algorithmes génétiques étant basés sur des phénomènes biologiques, il convient de rappeler au préalable quelques termes de l'évolution génétique. Les organismes vivants sont tout d'abord constitués de chromosomes qui correspondent en fait à des chaînes d'ADN. L'élément de base de ces chromosomes (le caractère de la chaîne d'ADN) est un allèle. Sur chacun de ces chromosomes, une suite d'allèles constitue une chaîne qui code les fonctionnalités de l'organisme (la couleur des yeux, ...). La position d'un allèle sur le chromosome est son locus. L'ensemble des chromosomes correspond au génome de l'individu.

On utilise aussi, dans les algorithmes génétiques, une analogie avec la biologie, qui concerne l'évolution, hypothèse émise par Darwin et qui propose qu'au fil du temps, les gènes conservés au sein d'une population donnée sont ceux qui sont le plus adaptés aux besoins de l'espèce et à son environnement [Wiki].

La génétique a mis en évidence l'existence de plusieurs opérations au sein d'un organisme donnant lieu au brassage génétique. Ces opérations interviennent lors de la phase de reproduction lorsque les chromosomes de deux organismes fusionnent. Ces opérations ont été reprises par les algorithmes génétiques afin de faire évoluer les populations de solutions de manière progressive.

a) Les "crossing over" ou recombinaisons:

Lors de cette opération, deux chromosomes s'échangent des parties de leurs chaînes, cela résulte de nouveaux chromosomes. Ces "crossing-over" peuvent être simples ou multiples. Dans le premier cas, les deux chromosomes se croisent et s'échangent des portions d'ADN en un seul point contre plusieurs pour les crossing-over multiples. Pour les algorithmes génétiques, c'est cet opérateur (le plus souvent sous sa forme simple) qui est le dominant, sa probabilité d'apparition lors d'un croisement entre deux chromosomes est un

paramètre de l'algorithme génétique. En règle générale, la proportion d'apparition est de l'ordre de 0,7 [Wiki].

b) Les mutations :

De façon aléatoire, un gène peut, au sein d'un chromosome être substitué à un autre. De la même manière que pour les crossing-over, on définit ici un taux de mutation lors des changements de population qui est généralement compris entre 0,001 et 0,01. Il est nécessaire de choisir pour ce taux une valeur relativement faible de manière à ne pas tomber dans une recherche aléatoire et conserver le principe de sélection et d'évolution.

III.2.3. Présentation des algorithmes génétiques :

Un algorithme génétique fait évoluer une population d'individus (chromosomes) interagissant très simplement grâce aux mécanismes de la sélection naturelle : les individus les plus forts (au sens d'un critère qui est directement relié à la fonction d'évaluation ou d'adéquation à optimiser) auront plus de descendants que les autres.

L'algorithme génétique est un algorithme itératif de recherche globale dont le but est d'optimiser une fonction définie par l'utilisateur appelée « fonction d'évaluation » ou d'adéquation, pour atteindre cet objectif l'algorithme travaille en parallèle sur une population de points candidats appelés individus ou chromosomes. Chaque individu est constitué d'un ensemble d'éléments appelés caractéristiques ou gènes pouvant prendre plusieurs valeurs (allèles) appartenant à un alphabet non nécessairement numérique.

Le but est donc de chercher la combinaison optimale de ces éléments (allèles) qui donnent lieu au maximum d'adéquation, à chaque itération appelée génération, est créée une nouvelle population d'individus, cette nouvelle génération comporte généralement des individus mieux adaptés à l'environnement tel qu'il est représenté par la fonction d'adéquation, ces deniers sont crées en utilisant des parties des meilleurs éléments de génération précédente ainsi que des parties innovatrices. Au fur et à mesure des générations, les individus vont tendre en général vers l'optimum de la fonction d'adéquation (voir la figure Fig.III.1).

Différents critères d'arrêt de l'algorithme peuvent être choisis :

- Le nombre de générations que l'on souhaite exécuter peut être fixé à priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.
- L'algorithme peut être arrêté lorsque la population n'évolue plus ou plus suffisamment rapidement.

Pour utiliser un algorithme génétique on doit disposer des cinq éléments suivants :

1. Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'états une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne le succès des algorithmes génétiques. Le codage binaire a été très utilisé à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles.

2. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.

3. Une fonction à optimiser. Celle-ci retourne une valeur de $\Re^{+}$ appelée *fitness* ou fonction d'évaluation de l'individu.

4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'états. L'opérateur de croisement "crossover" recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états.

5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation [Aldu05].

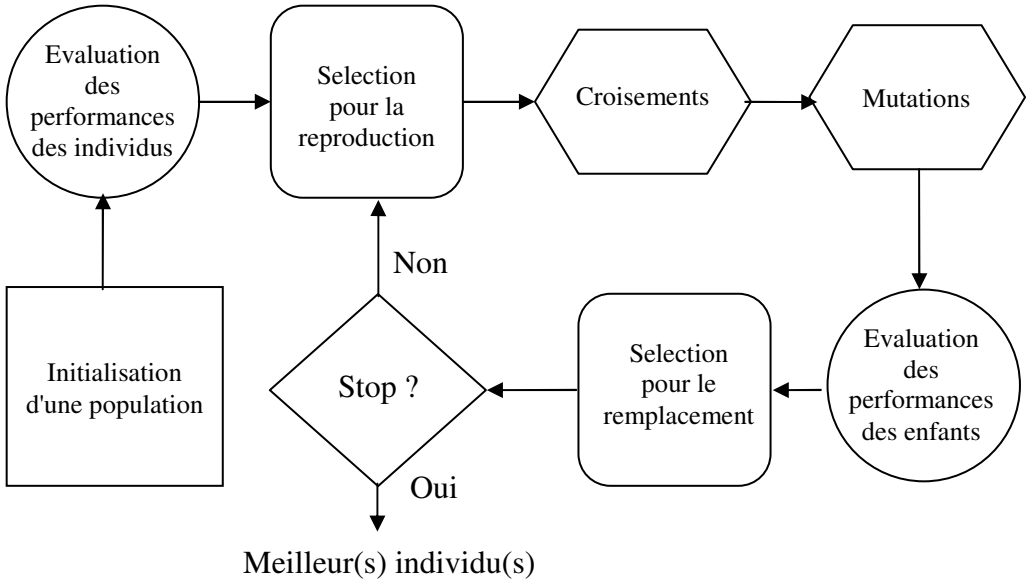

Fig.III.1 - Schéma général d'un algorithme génétique [Gonz04]

III.2.4. Description détaillée des algorithmes génétiques :

III.2.4.1. Le Codage :

Les Algorithmes Génétiques nécessitent le codage des paramètres à optimiser sous forme d'un ou plusieurs individus et opèrent sur le code plutôt que sur les paramètres eux-mêmes.

Chaque individu est composé d'un ensemble d'éléments appelés gènes pouvant prendre plusieurs valeurs (allèles) appartenant à un alphabet non nécessairement numérique.

Le choix du codage a un effet sur les performances d'un Algorithme Génétique, d'une part on doit choisir un codage simple dans le sens ou le temps de décodage est minimal, d'autre part, il doit être valide dans le sens ou les résultats obtenus doivent correspondre aux contraintes du problème.

D'un point de vue informatique, on utilise dans les algorithmes génétiques un codage binaire, un chromosome est un tableau de gènes, un individu est un tableau de chromosomes, et la population est un tableau d'individus. Notons qu'on pourrait aussi

utiliser d'autres formes de codage (réel, codage de Gray…) où La structure du problème est conservée dans le codage [Wiki].

III.2.4.2. La Population initiale :

La population initiale représente un ensemble d'individus de taille N, chaque individu est une solution potentielle du problème.

La population initiale est générée aléatoirement, Cependant certaines méthodes garantissent le démarrage avec des solutions assez bonnes, obtenues à l'aide des algorithmes approchés ou expérimentalement comme c'est le cas des applications industrielles.

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace d'états est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'états en veillant à ce que les individus produits respectent les contraintes. Si par contre, des informations a priori sur le problème sont disponibles, il parait bien évidemment naturel de générer les individus dans un sous-domaine particulier afin d'accélérer la convergence [AGs].

III.2.4.3. La Fonction d'évaluation :

C'est l'une des concepts les plus importants et les plus compliqués dans les Algorithmes Génétiques. Elle est appelée aussi fonction d'adéquation (fitness). Elle évalue chaque solution en utilisant certains critères qui vont définir son aptitude à donner la bonne solution au problème. Ainsi, on associe à chaque individu un poids qui va définir l'existence ou l'élimination de cet individu dans la prochaine génération.

III.2.4.4. Les opérateurs génétiques :

III.2.4.4.1. L'opérateur de sélection :

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. Ainsi, pour que 1'algorithme puisse converger, un élément se reproduira d'autant mieux que sa fonction d'évaluation est bonne, pour appliquer une sélection, il suffit donc de connaître pour chaque individu la valeur de critère ou

"fitness". Il existe de nombreuses méthodes de sélection, La plus simple et plus utilisée est : la sélection suivant la roulette.

L'opérateur de sélection choisit un individu i avec une probabilité : $\frac{f_i}{\sum_{j=1}^{N} f_j}$

f_i : représente la qualité d'adéquation de l'individu i (*fitness*),

N : est la taille de la population.

L'opérateur de sélection est appliqué N fois (i-e. autant de fois qu'il y a d'individus dans la population car la taille des générations est constante) [Rasue04].

L'espérance mathématique du nombre d'enfants de 1'individu i sera : $N * \frac{f_i}{\sum_{j=1}^{N} f_j}$

Individus	***Fitness***	***Fitness relative***
1	15	0.1
2	30	0.2
3	45	0.3
4	22.5	0.15
5	7.5	0.05
6	30	0.2

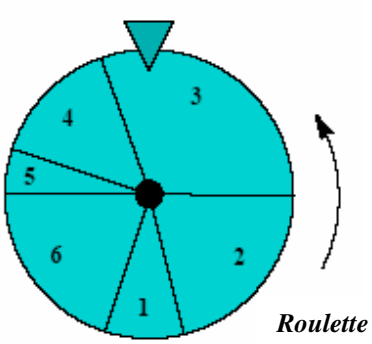

Fig III.2 - Exemple d'application de la sélection suivant la roulette [Lass00].

Pour réaliser la sélection, il suffit de faire tourner une roue un nombre de fois égale à la taille de la population, puis la valeur de la roue est attribuée à l'individu i.

Les individus les mieux adaptés ont plus chance d'être sélectionnés, ils peuvent même être sélectionnés plusieurs fois.

III.2.4.4.2. Opérateur de croisement :

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes existants. En général, les croisements sont envisagés avec deux parents et génèrent deux enfants mais il est possible d'imaginer des croisements avec N parents et K enfants.

Pour des problèmes discrets on utilise le croisement à découpage de chromosomes (slicing crossover). On effectue ce type de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position interne dans chacun des parents (P1et P2) et on échange ensuite les deux sous-chaînes terminales (par exemple) des deux chromosomes, ce qui produit deux enfants (C1et C2) [AGs]. (Voir figure Fig III.3)

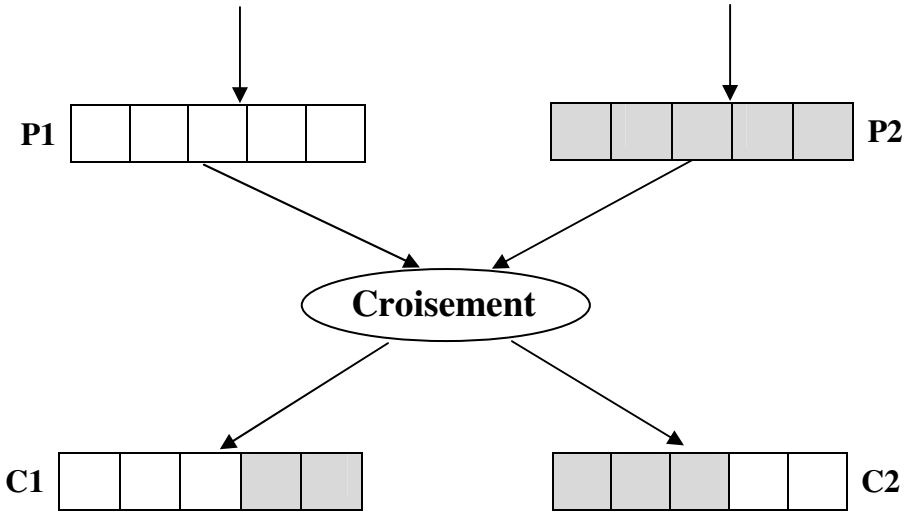

Fig.III.3 - Croisement à découpage de chromosome

Les Types de croisement :

Il y a trois types de croisements standard :

• Croisement uni-point :

On choisit au hasard un site de croisement (un nombre entre 1 et L, L étant la longueur du chromosome).

Soit K l'indice de site, le 1er enfant sera formé à partir des K premiers gènes du 1er parent et (L-K) derniers gènes du second parent. Le 2ème enfant sera formé a partir des K premiers gènes du 2ème parent avec les (L-K) derniers gènes du 1er parent. (Comme exemple, voir la figure Fig III.3).

• Croisement bipoint **:**

Le croisement bipoint consiste a choisir aléatoirement deux points sur la longueur de chacun des deux parents (les deux points sont les mêmes pour les deux parents). On obtient

donc trois parties pour chaque parent (un segment tête, un segment centre et un segment queue).

Le 1er enfant contient: les (1ère et dernière) parties du 1er parent, la partie du milieu de l'enfant contiendra la partie centre du 2ème parent.

La construction du 2ème enfant suit le même principe en inversant les parents. Exemple:

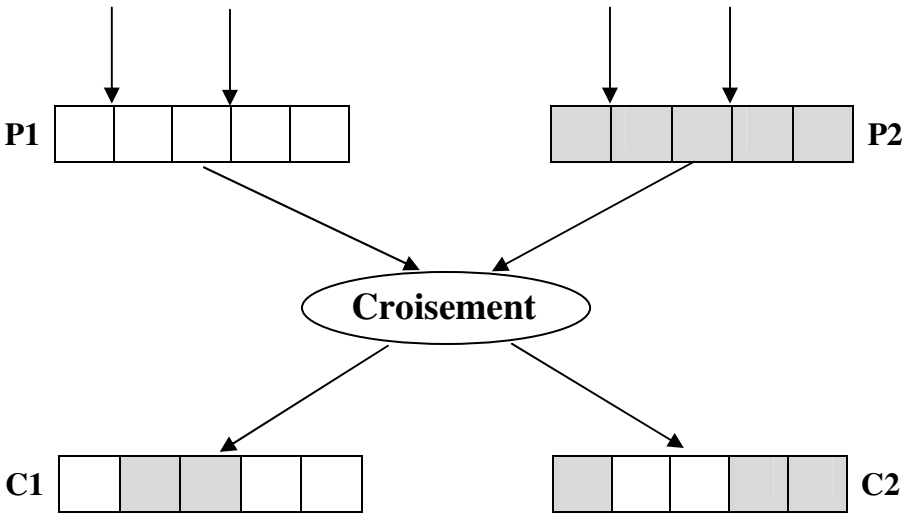

Fig.III.4 - Représentation schématique du croisement bipoint.

• Croisement uniforme :

Il existe deux variantes de croisement uniforme :

1ère variante :

Le 1er enfant sera produit de la manière suivante :

On prend le 1er gène du 1er parent concaténé au second gène du 2ème parent concaténé au 3ème gène du 1er parent et ainsi de suite Jusqu'a ce que la longueur de 1'enfant sera égale a celle des parents.

Le 2ème enfant sera construit de la même manière en partant du 2ème parent. (Comme l'exemple de la figure Fig.III.5)

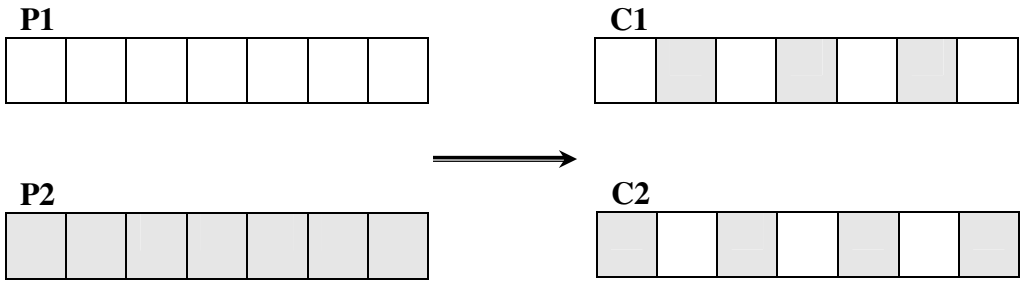

Fig.III.5 - Représentation schématique du croisement uniforme 1ère variante

2ème variante :

Chaque gène est crée en copiant le gène du 1er parent ou du 2ème parent, dans la même position selon un masque de croisement généré aléatoirement

Le 1er enfant est conçu de la manière suivante :

Un '1' dans le masque signifie que le gène du 1er parent est copie dans le 1er enfant et un '0' signifie que le gène du 2ème parent est copie dans le 1er enfant.

Le 2ème enfant est conçu de la manière suivante :

Un '1' dans le masque signifie que le gène du 2ème parent est copie dans le 2ème enfant et un '0' signifie que le gène du 1er parent est copie dans le 2ème enfant. Comme l'exemple suivant :

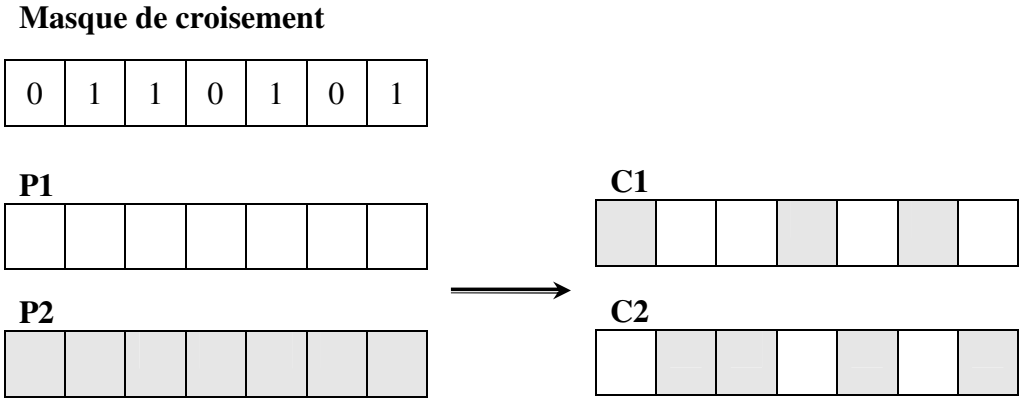

Fig.III.6 - Représentation schématique du croisement uniforme 2ème variante

On peut imaginer et tester des opérateurs de croisement plus ou moins complexes sur un problème donné mais l'efficacité de ce dernier est souvent liée intrinsèquement au problème.

III.2.4.4.3. Opérateur de Mutation :

La mutation est appliquée après le croisement, elle réalise la modification aléatoire avec une probabilité *Pm* (généralement faible) de la valeur d'un ou plusieurs gènes d'un individu.

Lorsqu'un gène est muté, sa valeur est remplacée par une autre appartenant au même alphabet.

On estime le nombre de gènes qui vont subir 1'opération de mutation par:
Pm x *Taille_Chromosome* x *Taille_pop*, où *taille-chromosome* est le nombre de bits du chromosome.

L'opérateur de croisement devient moins efficace avec le temps, car les individus deviennent similaires. C'est à ce moment là que 1'operateur de mutation prend toute son importance. En effet, il est utilisé pour éviter une perte irréparable de la diversité dans la population donc son rôle est la restauration de 1'information génétique perdue.

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'états, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implantations fonctionnent de cette manière. Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur sur le plan théorique.

Pour les problèmes discrets, l'opérateur de mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire (voir la figure Fig.III.7). Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale [AGs].

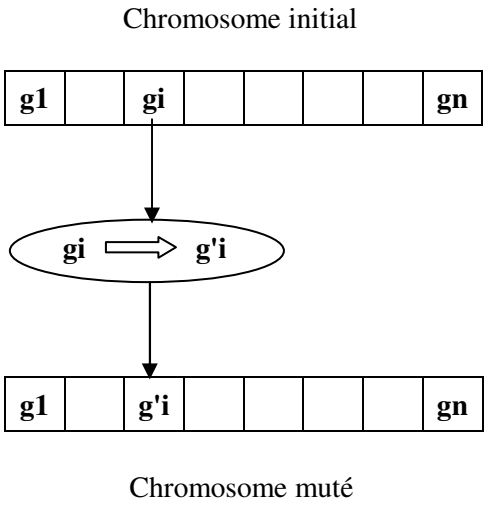

Fig.III.7 - Représentation schématique d'une mutation dans un Chromosome

III.2.4.5. L'arrêt :

Stopper le processus au bon moment est essentiel du point de vue pratique. Si l'on a peu ou pas d'informations sur la valeur cible de l'optimum recherché (ce qui autorise un arrêt dès que cette valeur est atteinte par le meilleur individu de la population courante), il est délicat de savoir quand arrêter l'évolution. En l'absence de toute information, une stratégie couramment employée consiste à stopper l'algorithme dès qu'un nombre maximal d'itérations est atteint, ou qu'un stade de "stagnation" est identifié. Il est aussi possible de tester la dispersion de la population. Il est évident qu'une bonne gestion de l'arrêt de l'évolution contribue de façon importante à l'efficacité de la méthode, et intervient au même titre que le réglage des principaux paramètres de l'algorithme (taille de population, probabilités de croisement et de mutation, pression sélective, proportion de remplacement) [Lut04].

III.2.5. Domaine d'application :

Les algorithmes génétiques sont des méthodes souples et applicables à une grande variété de problèmes dans divers domaines notamment les domaines suivants :

• La recherche : Il existe une multitude de problème discret qui sont un champ d'application pour les algorithmes génétiques, comme exemple : le problème du voyageur de commerce.

• Economie (ordonnancement des tâches, gestion de la production, suivi de projet, la prévision boursière…).

• Médecine (traitement d'images…).

• Robotique (planification automatique de trajectoire d'un robot, application Industrielle (voiture, missile...)).

III.2.6. Les limites des algorithmes génétiques :

- Le temps de calcul : bien que ces algorithmes soient plus performants que des algorithmes qui parcourent toutes les solutions, ils nécessitent tout de même de nombreux calculs, en particulier au niveau de la fonction d'évaluation.

- Ils sont aussi souvent difficiles à programmer. Certains paramètres comme la taille de la population n'étant pas facilement évaluables. Un autre point souvent difficile à programmer concerne la fonction d'évaluation. Celle-ci doit souvent prendre en compte plusieurs paramètres du problème qu'il faut considérer avec attention.

- Il faut aussi noter l'impossibilité d'être assuré, même après un nombre important de générations, que la solution que l'on a trouvé est la meilleure. On peut seulement être sûr que l'on s'est approché de la solution optimale, sans la certitude de l'avoir atteinte.

- Un autre problème important est celui des optimums locaux, ou le problème de convergence Prématurée, du fait de la sélection, la population finit par n'être constituée que d'individus tous similaires. L'une des préoccupations majeures dans les algorithmes génétiques consiste d'ailleurs a préserver le plus longtemps possible une diversité suffisante dans la population. Face à ce type de difficulté, la solution consiste à diriger la poursuite de la recherche vers de nouvelles zones, *i.e.*, à recourir à l'exploration.
  En effet, lorsqu'une population évolue, il se peut que certains individus qui à un instant occupent une place importante au sein de cette population deviennent majoritaires. À ce moment, il se peut que la population converge vers cet individu et s'écarte ainsi d'individus plus intéressants mais trop éloignés de l'individu vers lequel on converge. Pour vaincre ce problème, il existe différentes méthodes comme l'ajout de quelques

individus générés aléatoirement à chaque génération, des méthodes de sélection différentes de la méthode classique [Wiki]

III.3. L'approche de recherche locale :

III.3.1. Principe :

Les méthodes de recherche locale sont des algorithmes itératifs qui explorent l'espace de solutions X en se déplaçant pas à pas d'une solution à une autre. Une méthode de ce type débute à partir d'une solution $s_0 \in X$ choisie arbitrairement ou alors obtenue par le biais d'une méthode constructive.

Le passage d'une solution admissible à une autre se fait sur la base d'un ensemble de modifications élémentaires qu'il s'agit de définir selon le problème étudié. Une solution s'obtenue à partir de s en appliquant une modification élémentaire. Le voisinage N(s) d'une solution $s \in X$ est défini comme l'ensemble des solutions admissibles atteignables depuis s en effectuant une modification élémentaire.

Un tel processus d'exploration est interrompu lorsqu'un ou plusieurs critères d'arrêt sont satisfaits. Le fonctionnement d'une méthode de recherche locale est illustré de manière générale dans la figure Fig.III.8. Les passages successifs d'une solution à une solution voisine définissent un chemin au travers de l'espace des solutions admissibles. La modélisation d'un problème d'optimisation et le choix du voisinage doivent être effectués de telle sorte qu'il existe au moins un " chemin " entre chaque solution $s \in X$ et une solution optimale s*. En effet, l'existence de tels chemins permet à la méthode de recherche locale d'atteindre une solution optimale à partir de n'importe quelle solution admissible [Wid01].

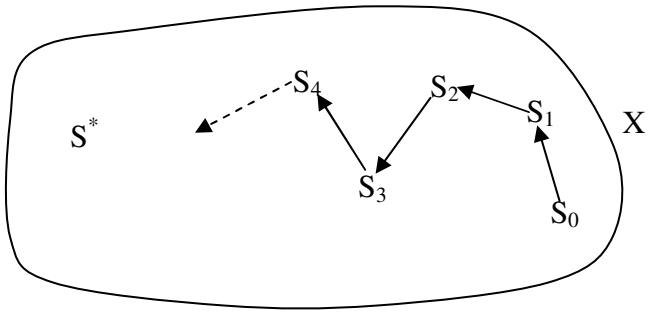

Fig.III.8 - Exploration de l'espace de solutions X par une approche de recherche locale

Plusieurs méthodes de recherche locale ont été développées au cours de ces dernières années. Parmi elles on peut citer la recherche taboue et le recuit simulé. Dans le paragraphe suivant on va présenter la méthode de recuit simulé qui sera utilisée dans l'algorithme mimétique final.

III.3.2. Le recuit simulé :

Le recuit est un processus physique de refroidissement. Ainsi, lorsqu'on chauffe un métal solide, il devient liquide à une certaine température, dans ce cas les atomes qui le composent leur degré de liberté augmente. Inversement lorsque l'on baisse la température le degré de liberté diminue jusqu'à obtenir un solide. Maintenant suivant la façon dont on diminue la température on obtient différents solides :

- Baisse brutale de la température (La trempe), cela produit une structure amorphe, un verre. On a alors un minimum local d'énergie.
- Baisse progressive de la température de façon à atteindre le minimum global d'énergie. On obtient dans ce cas un cristal.

Lorsque la baisse progressive est trop rapide on a alors des défauts au niveau du cristal. Le recuit permet de redonner de la liberté aux atomes pour tenter d'atteindre un nouvel état dynamique [Dam02].

Le recuit simulé est une méthode de recherche locale dont le mécanisme de recherche est calqué sur le principe du recuit thermodynamique. Le refroidissement progressif d'un système de particules est simulé en faisant une analogie entre l'énergie du système et la fonction objectif du problème (P) d'une part, et entre les états du système et les solutions admissibles de (P) d'autre part (voir l'organigramme de la Figure Fig.III.9). Pour atteindre des états avec une énergie aussi faible que possible, on porte initialement le système à très haute température puis on le refroidit petit à petit. Tout nouvel état est obtenu en faisant subir un déplacement infinitésimal aléatoire à un atome quelconque. Soit ΔE la différence d'énergie occasionnée par une telle perturbation. Le nouvel état est accepté si l'énergie du système diminue ($\Delta E < 0$). Dans le cas contraire ($\Delta E = 0$), il est accepté avec une certaine probabilité: $prob(\Delta E, t) = \exp(\frac{-\Delta E}{K_B t})$ où t est la température du système et k_B une constante physique connue sous le nom de constante de Boltzmann. A chaque étape, l'acceptation d'un nouvel état dont l'énergie n'est pas inférieure à celle de l'état courant est

décidée en générant de manière aléatoire un nombre q∈ [0,1[. Si q est inférieur ou égal à prob(ΔE, t), alors le nouvel état est accepté. Autrement l'état courant est maintenu. L'utilisation répétée d'une telle règle fait évoluer le système vers un état d'équilibre thermique.

En règle générale, la température est diminuée par paliers à chaque fois qu'un certain nombre d'itérations est effectué. L'algorithme est interrompu lorsque aucune solution voisine n'a été acceptée pendant un cycle complet d'itérations à température constante.

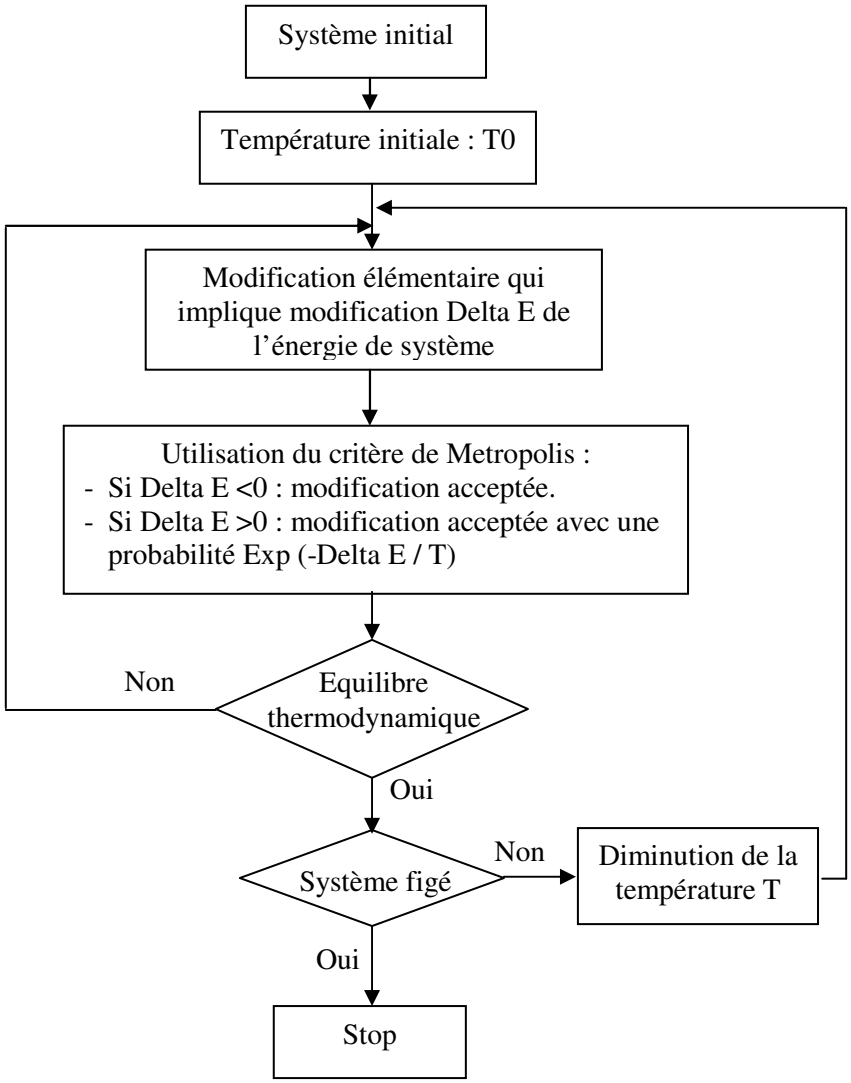

Fig.III.9 - L'organigramme de l'algorithme du recuit simulé

III.4. Les algorithmes mimétiques :

Dans les dernières années de nombreuses adaptations et améliorations ont été proposées pour les algorithmes génétiques dans le cadre de combler les déficiences principales de ces algorithmes. La plupart des innovations introduites dans les algorithmes génétiques font appel à des concepts qui ne sont plus liés à des principes d'évolution génétique. L'une des approches proposées est les algorithmes "Mimétiques" qui insèrent une méthode de recherche locale dans la phase d'adaptation individuelle d'un algorithme génétique**.**

III.4.1. Concept de mime :

Un algorithme mimétique prend son nom de mot "Mime" (Meme) qui à été introduit par R.Dawkins dans son livre "The Selfish Gene" (1976), qui dénote l'idée d'une unité d'information dans la transmission culturelle. L'idée de mime est introduite comme une alternative au concept de gène. Non seulement le matériel génétique est transmis d'une génération à la suivante, mais aussi des idées et des concepts. Le concept de mime est assimilé comme une unité d'information passée entre des individus. Quand un gène est passé aux descendants, il n'est pas altéré : les individus n'ont aucun contrôle sur leur composition génétique. Par contre, l'individu peut adapter le mime à son propre environnement. [Chan02].

P.Moscato [Mosca] donne un bon exemple de "Mime" qui est les arts martiaux. Une séquence de mouvement appelée "Form" est composée de plusieurs sous-unités de séquences défensives et agressives, qui peuvent également être divisé. Les mouvements indivisibles dans une "Form" sont considérés comme des mimes, et un mouvement défensif est généralement composé des actions ordonnées de plusieurs de ces mimes. Ce modèle ressemble à la structure d'un chromosome, les gènes et les allèles.

III.4.2. Schéma général d'un algorithme mimétique :

Tandis que les algorithmes génétiques ont été inspirés en essayant d'émuler l'évolution biologique. Les algorithmes mimétiques essayeraient d'imiter l'évolution culturelle. L'approche mimétique est basée sur le concept de l'évolution, comme il est utilisé dans les algorithmes génétiques. Mais contrairement à un algorithme génétique qui

prend son modèle de l'évolution biologique, l'algorithme mimétique s'attache au modèle de l'évolution culturelle ou l'évolution des idées ("Mimes"). La différence principale entre ces deux modèles est que l'idée ("Mime") peut être modifié et amélioré chez l'individu qu'il possède, par contre dans l'évolution biologique un individu n'a aucun contrôle sur sa composition génétique (c-à-d le gène ne peut être modifié durant la vie d'un individu). L'amélioration ou l'adaptation de "mime" dans un algorithme mimétique est obtenue par l'introduction de la recherche locale dans l'algorithme génétique [Digma03].

Les algorithmes mimétiques sont des algorithmes évolutionnaires qui appliquent le processus de la recherche locale pour raffiner les individus de la population (améliorer leurs valeurs de la fonction d'adaptation). Un algorithme mimétique est un mariage entre une recherche globale basée sur la population et la recherche locale heuristique faite sur chacun des individus. [Mosca]. Pour cela, ils sont connus aussi par les algorithmes évolutionnaires hybrides ou la recherche locale génétique. Le schéma suivant résume le fonctionnement d'un algorithme mimétique. [Chan02].

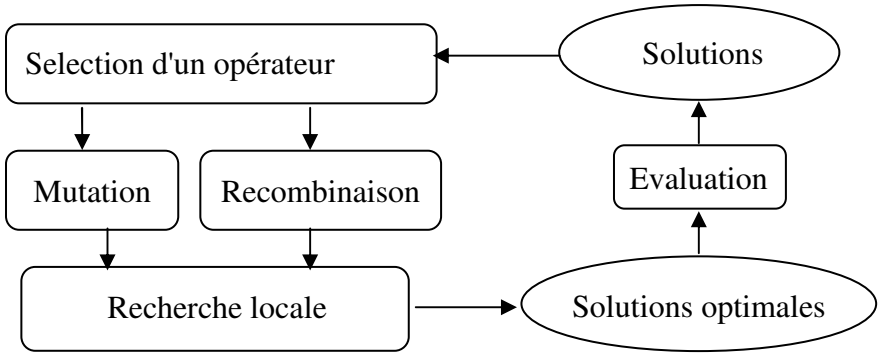

Fig.III.10 - Le schéma global d'un algorithme mimétique

Et voici une description générale d'un algorithme mimétique : premièrement on va donner une représentation du problème d'optimisation, ensuite créer un certain nombre d'individus. L'état de ces individus peut être aléatoirement choisi ou selon une certaine procédure d'initialisation. Une heuristique peut être choisie pour initialiser la population. Ensuite, pour chaque individu on applique la recherche locale. Le mécanisme de la recherche locale peut avoir comme but d'atteindre des optimums locaux ou d'améliorer les individus (concernant la fonction objective) jusqu'à un niveau prédéterminé. Ensuite, et

après que l'individu a atteint un certain développement, il interagir avec les autres membres de la population. L'interaction peut être concurrentielle ou coopérative. La concurrence peut être semblable aux processus de sélection dans les algorithmes génétiques. Le comportement coopératif peut être considéré en tant que les mécanismes du croisement dans les algorithmes génétiques ou d'autres types de reproduction qui a comme résultat la création d'un nouvel individu. Plus généralement, nous devons comprendre la coopération comme échange d'information.

La recherche locale et la coopération (échange d'information) ou la concurrence (sélection des bons individus) sont répétées jusqu'à ce qu'un critère d'arrêt soit satisfait [Mosca].

III.4.3. Exemple d'un algorithme mimétique :

On propose un algorithme hybride basé sur le schéma de combinaison décrit dans la figure Fig.III.11 ci-dessous. L'algorithme qui en découle est un algorithme évolutif combinant un algorithme génétique avec une méthode de recherche locale basée sur la technique de recuit simulé.

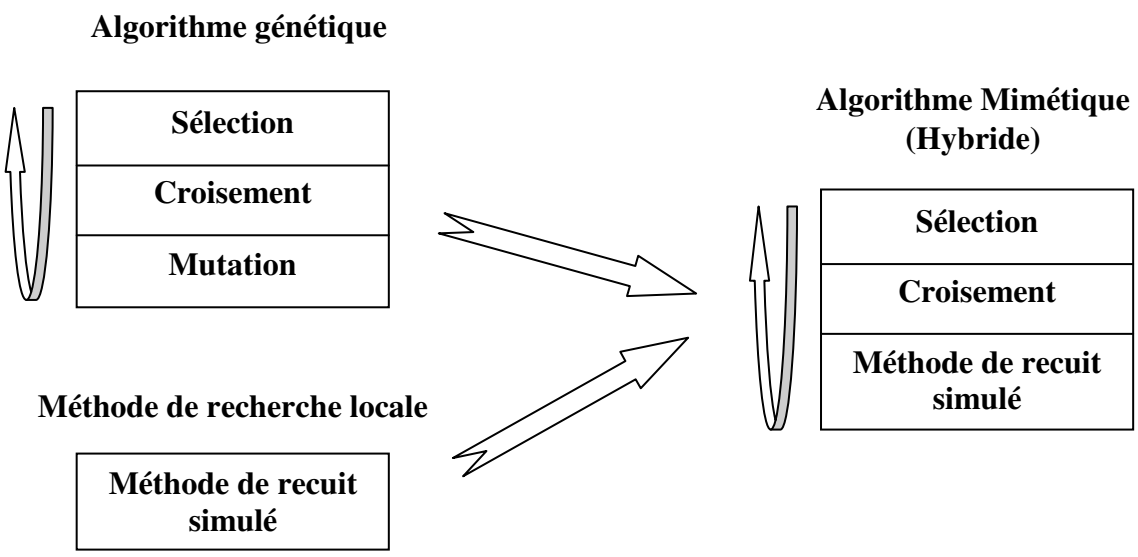

Fig.III.11 - Schéma de combinaison d'un algorithme mimétique.

Dans notre algorithme la méthode de recherche locale remplace la phase de mutation dans l'algorithme génétique. Alors l'algorithme proposé combine la capacité de l'exploration dans le algorithme génétique et la capacité de l'exploitation dans la recherche locale. Le fonctionnement de cet algorithme peut être défini par le pseudo code de la figure Fig.III.12.

1. *Initialisation des paramètres de l'algorithme mimétique*
2. *Générer aléatoirement la population initiale d'individus*
3. *Pour chaque individu appliquer la recherché locale pour améliorer sa fonction d'adaptation*
4. *Reproduction en utilisant le Croisement et la recherche locale*
5. *Remplacement des individus*
6. *Si le critère d'arrêt non satisfait aller à* ***4.*** *Sinon fin*

Fig. III.12 - Le pseudo-code de l'algorithme mimétique

III.5. Conclusion :

La nature peut nous inspirer des méthodes d'optimisation très générales, comme les algorithmiques génétiques et la méthode de recuit simulé. Ces méthodes ont prouvé leurs performances, et ils ont été utilisés dans plusieurs domaines de la recherche et de l'industrie. Les algorithmes mimétiques (hybride) viennent comme une amélioration de ces méthodes, parce que quand une technique de recherche locale est introduite dans l'algorithme génétique, l'espace des solutions est bien exploré. Cependant le choix des opérateurs, des caractéristiques et des paramètres de l'algorithme en lui-même reste une étape très complexe.

Dans la suite de notre travail on essaye d'appliquer l'algorithme mimétique proposé dans ce chapitre dans le domaine de détection d'intrusions.

Chapitre IV

Une approche mimétique pour l'audit de sécurité

IV.1. Introduction :

Ce chapitre est consacré à la formalisation du problème d'analyse de fichier d'audit de sécurité en utilisant une approche basée sur les algorithmes mimétiques. Nous donnons, tout d'abord, la méthode choisie pour développer notre système de détection d'intrusion. Ensuite, nous présentons notre approche mimétique.

Après avoir formalisé le problème d'analyse de fichier de l'audit au moyen d'une approche mimétique, nous présentons l'architecture globale de notre système de détection d'intrusion incluant le module mimétique.

IV.2. Choix de la méthode de détection d'intrusions :

Notre objectif 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, afin de détecter les éventuelles intrusions.

Notre choix est justifié par le fait que l'approche par scénarios peut remédier aux inconvénients de l'approche comportementale qui génère beaucoup de fausses alertes, c'est-à-dire que le système croit être attaqué alors qu'il ne l'est pas.

Contrairement à l'approche comportementale, l'approche par scénarios peut prendre en compte les comportements exacts des attaques potentielles. Les inconvénients sont dans la base des attaques qui doit être bien construite et les performances des signatures d'attaques qui sont limitées par l'esprit humain qui les a conçues.

IV.2.1. Expression des scénarios d'attaques :

Un scénario d'une attaque est défini par la suite des activités que l'intrus réalisant cette attaque entreprend sur le système. Ces activités sont ensuite traduites en termes de données d'audit.

L'utilisation de tels scénarios d'attaque présente divers avantages :

- l'officier de sécurité peut lui-même construire les scénarios d'attaques en fonction de ce qui est réellement à craindre pour le système,
- la modification d'un scénario déjà existant est simple. De même, il est facile de prendre en compte un nouveau scénario,
- les évènements que l'on audite peuvent être restreints à ceux qui apparaissent dans les différents scénarios (cela ne signifie pas que seuls des évènements correspondant à une attaque sont enregistrés car un évènement présent dans un scénario particulier peut très bien apparaître en dehors de tout comportement intrusif).

Les scénarios d'attaques 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'attaques peuvent être considérés comme des sous-suites qu'il s'agit de localiser dans cette chaîne principale.

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

L'analyse de fichier d'audit de sécurité consiste à rechercher les sous suites d'évènements qui représentent les scénarios d'attaques dans une chaîne qui représente le fichier d'audit.

IV.3. Un algorithme mimétique pour l'analyse de fichier d'audit :

L'audit de sécurité, à l'image du diagnostic médical, a pour objectif de déterminer l'ensemble des conditions qui peuvent expliquer la présence de symptômes observés. Pour cela, un expert humain utilise des connaissances spécifiques de type cause à effet. Dans notre cas, les symptômes sont les évènements enregistrés dans le fichier d'audit et les connaissances de l'expert sont les scénarios d'attaques. Dans ce qui suit nous présentons notre approche mimétique pour l'analyse de fichier d'audit de sécurité.

IV.3.1. Formalisation du problème de l'audit de sécurité :

Notre approche est inspirée de GASSATA [Mé98] où les scénarios d'attaque sont considérés 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 est décrit comme suit :

- L'expression d'un scénario d'attaque :

Soient N_e le nombre de types d'événement auditables et N_a le nombre de scénarios d'attaques possibles. Chaque scénario d'attaque est exprimé sous 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* $(AE_{ij} \ge 0)$.

- L'expression du fichier d'audit :

Le fichier d'audit peut être exprimé sous forme d'une matrice d'entiers de dimension N_e, où N_e est le nombre de types d'événement qui sont auditables. La matrice

colonne résultante est appelée matrice d'audit notée *O* (pour Observé). Le i^{ème} élément O_i de cette matrice 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* (pour Risque) 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).

- L'objectif :

Le but consiste à 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.

En d'autres termes, 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 $(AE \cdot H)_i \leq O_i$, *i* variant de 1 à N_e (voir la figure IV.1).

Notons que le produit matriciel $AE \cdot H$ résulte en une matrice colonne de dimension N_e. L'élément $(AE \cdot H)_i$ $(1 \leq i \leq N_e)$ de cette matrice correspond à la somme des éléments $AE_{ij} (1 \leq j \leq N_a)$ de la matrice Attaque-Evénement, *j* prenant les valeurs successives des indices de la matrice *H* pour lesquels $H_j = 1$. Il s'agit 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. 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 $(AE \cdot H)_i$.

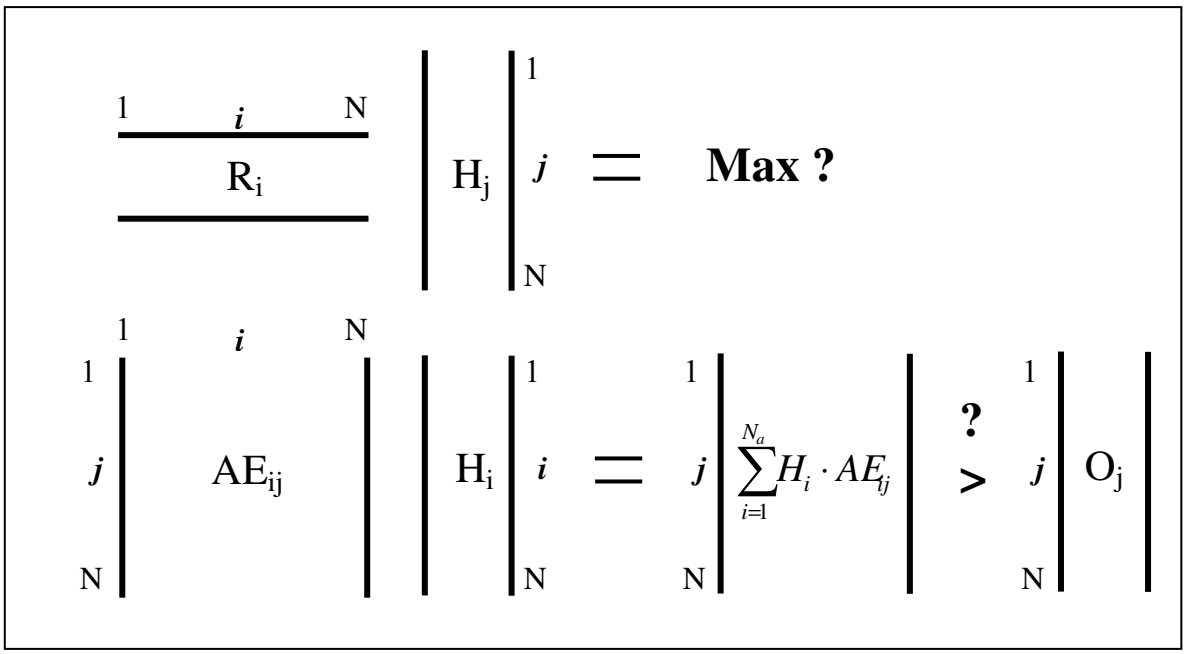

Fig.IV.1 - Le problème de l'analyse simplifiée de fichiers d'audit de sécurité

Le problème d'analyse du fichier d'audit de sécurité décrit précédemment est un problème NP-complet, et le temps de traitement nécessaire à sa résolution ne permet pas d'envisager une solution algorithmique polynomiale dans les cas réels [Mé98]. C'est pourquoi en a envisagé d'utiliser une méthode heuristique basée sur un algorithme mimétique qui est l'hybridation d'un algorithme génétique et d'une recherche locale de recuit simulé.

Dans ce qui suit, nous présentons les différentes composantes de notre approche mimétique constituée de :

- Une population constituée de plusieurs individus représentant des solutions potentielles du problème,
- Un mécanisme d'évaluation de l'adaptation (fitness) de chaque individu de la population,

- Un mécanisme d'évolution composé d'opérateurs permettant d'éliminer certains individus et de produire de nouveaux individus à partir des individus sélectionnés,
- Un mécanisme d'amélioration de l'adaptation des individus basé sur le principe de recuit simulé.

IV.3.2. Codage des individus :

Un individu est une chaîne de longueur *l* codant une solution potentielle au problème. Comme notre but est de déterminer la matrice colonne *H* maximisant le produit matriciel $R \cdot H$, tout en respectant la contrainte $(AE \cdot H)_i \ (1 \le i \le N_e)$, et comme la matrice *H* est une matrice binaire, de dimension N_e 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. Alors nous sommes donc dans une situation où le codage des individus est immédiat puisque la solution du problème à résoudre s'exprime précisément sous forme d'une suite binaire.

Un individu est donc une suite binaire de N_a gènes. Chaque individu correspond à une occurrence particulière de la matrice *H*. En d'autres termes, le gène de locus *i* d'un individu vaudra 1 si l'individu correspond à une solution dans laquelle l'attaque est déclarée présente. Dans le cas contraire, ce gène vaudra 0.

Il faut noter que ce codage ne permet pas la détection de la réalisation multiple de la même attaque du fait de l'utilisation d'un alphabet chromosomique binaire.

IV.3.3. Fonction sélective :

Notre problème d'analyse de fichier d'audit de sécurité est un problème de maximisation consistant à déterminer la matrice *H* maximisant le produit matriciel $R \cdot H$. Alors il parait donc facile de proposer la fonction sélective *F* permettant d'évaluer la valeur sélective d'un individu *I* comme suit :

$$F = \sum_{i=1}^{N_a} R_i \cdot I_i$$

Cependant, cette fonction sélective ne tient pas compte du fait qu'un individu doit respecter les contraintes spécifiées par les inégalités suivantes : $(AE \cdot H)_i \leq O_i$, avec i variant de 1 à N_e. La prise en compte de ces contraintes par notre algorithme consiste à éliminer les individus codant une solution ne respectant pas les contraintes. Pour ce faire, une fonction de pénalité est introduite. Elle permet de pénaliser les individus ne respectant pas les contraintes en diminuant leur valeur sélective. On aura donc la fonction sélective similaire à celle utilisée dans [Mé98] et décrite comme suit :

$$F(I) = \alpha + \left(\sum_{i=1}^{N_a} R_i \cdot I_i - \beta \cdot T_e^P \right) \text{ où :}$$

- I_i est le gène de locus i de l'individu I,
- T_e est le nombre de types d'événement auditables $(T_e \leq N_e)$ vérifiant l'inégalité $(AE \cdot I)_i > O_i$ pour un individu I. on va pénaliser un individu proportionnellement au nombre de types d'événement pour lesquels la contrainte n'est pas respectée.
- Le paramètre β permet de pondérer l'importance relative de la pénalité par rapport au terme $\sum_{i=1}^{N_a} R_i \cdot I_i . \beta$ doit vérifier l'inégalité suivante : $\beta > \max(R_i)$. Pour une matrice risque unitaire, tous les R_i sont à 1, alors $\beta = 2$.

Une valeur sélective n'ayant de sens que si elle est positive, toute valeur négative est en fait ramenée à 0. Pour cela on utilise le paramètre α pour assurer de ne pas avoir de valeur négative pour la fonction d'adaptation.

IV.3.4. Opérateurs évolutionnaires :

Notre algorithme utilise les opérateurs de sélection, de croisement, et une stratégie de remplacement des individus. La sélection a pour objectif de choisir les individus qui vont pouvoir se reproduire pour transmettre leurs caractéristiques à la génération suivante, tandis que le croisement ou recombinaison cherche à combiner les caractéristiques des individus

parents pour créer des individus enfants avec de nouvelles potentialités dans la génération future. La phase de mutation est remplacée par la une recherche locale de recuit simulé.

IV.3.4.1. La sélection :

Le principe de sélection utilisé se base sur la méthode de sélection par la roulette "*Roulette wheel selection*", qui consiste à donner une plus grande chance aux individus de bonne fitness qu'aux individus moins adaptés. La probabilité de sélection de chaque individu est égale à sa fitness divisée par la fitness de toute la population.

L'opération se déroule en deux phases : premièrement, on calcule la probabilité de sélection (pourcentage) pour chaque individu en fonction de sa fitness et de la somme des fitness de chaque individu. Deuxièmement, on tire au hasard des individus (un individu peut être choisi plusieurs fois).

On associe à chaque individu un segment dont la longueur est proportionnelle à sa fitness. On reproduit ici le principe de tirage aléatoire utilisé dans la roulette de salle de jeux avec une structure linéaire. Ces segments sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1 (voir l'exemple de la figure Fig.IV.2). On tire alors un nombre aléatoire, de distribution uniforme entre 0 et 1, puis on regarde quel est le segment sélectionné. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent adressés que les petits, on privilégie ainsi les individus ayant une forte fitness au détriment des individus moins forts, tout en gardant une structure aléatoire.

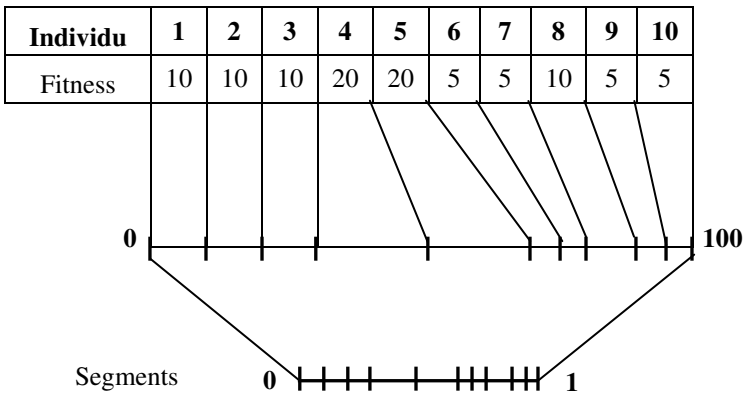

Fig.IV.2 - Principe de roulette

IV.3.4.2. Le croisement :

Un individu est une suite binaire de N_a gènes. Le gène de locus i d'un individu vaudra 1 si l'individu correspond à une solution dans laquelle l'attaque i est déclarée présente. Dans le cas contraire, ce gène vaudra 0.

Dans le but d'enrichir la diversité de la population et pour assurer une bonne recombinaison des individus nous avons opté pour le croisement bipoint qui fonctionne comme suit :

Soient deux parents P1 et P2 de longueur N_a, on tire au sort deux positions p et q avec $p \leq q$. On obtient donc trois parties pour chaque parent (un segment tête, un segment centre et un segment queue).

Le premier enfant E1 contient les (première et dernière) parties du premier parent, la partie du milieu de l'enfant contiendra la partie centre du deuxième parent. La construction du deuxième enfant E2 suit le même principe en inversant les parents. Ce principe est bien présenté dans l'exemple de la figure Fig.III.4 du chapitre III. L'algorithme mimétique proposé offre un aspect stochastique d'où les individus d'une population sont reproduits avec une probabilité Pc qui représente le taux de croisement qui un paramètre de l'algorithme à fixer d'une manière empirique.

IV.3.4.3. La stratégie de remplacement :

Pour diminuer la convergence prématurée qui est une caractéristique inhérente dans les algorithmes génétiques classiques, seule une petite partie des meilleurs individus survivra d'une génération à une autre. Nous avons choisi de garder uniquement les deux meilleurs individus de chaque génération pour la génération suivante. Le reste des individus de la génération courante seront remplacés par les nouveaux individus.

IV.3.5. Amélioration des individus par une recherche locale :

La phase d'amélioration est basée sur le recuit simulé décrit précédemment dans le chapitre III. Pour chaque individu de la population P_i, le recuit simulé est appliqué pour améliorer sa fonction d'adaptation. Nous obtenons des individus qui seront au moins aussi bons que les précédents.

L'algorithme de recherche locale démarre par une configuration initiale qui représente un individu de la population courante de l'algorithme mimétique. La température est diminuée par paliers à chaque fois qu'un certain nombre d'itérations est effectué. À chaque itération on génère une configuration voisine par modification d'un bit choisi aléatoirement de la configuration courante. La nouvelle configuration est acceptée si elle est meilleure ou elle respecte une certaine probabilité d'acceptation. La figure Fg.IV.3 donne les grandes lignes de l'algorithme de recherche locale proposée.

Algorithme du recuit simulé (T°, N)*;*

T° *: température initiale*

N *: nombre d'itérations par palier*

```
Début
S := Solution initiale; //  tirée de la population courante de coût E_S.
T := T°;
Tant que T >0  faire
  début
  J := 0;
  Tant que  J < N  faire // Une étape
    début
      S' := Générer une transition à partir de S; // par changement d'un bit choisi aléatoirement de S.
      ΔE := E_S – E_S' ;
      Si  ΔE ≤ 0  Alors  S := S' // Accepter la transition
      Sinon  début
              Proba := exp (– ΔE/T)
              Si aléatoire (0,1) <= Proba  Alors  S := S' // Accepter la solution
              fin;
      J := J+1;
    fin;
  T := T-1;
  fin;
Solution finale := S;
Fin.
```

Fig.IV.3 - les grandes lignes de l'algorithme de la technique de recuit simulé.

IV.3.6. L'organigramme de l'algorithme mimétique :

L'organigramme de la Figure Fig.IV.4 donne les différentes étapes constituant notre approche qui n'utilise pas l'opérateur de mutation, mais inclut deux phases nécessaires qui sont :

- La recherche locale et le croisement.
- La compétition (sélection des meilleurs individus).

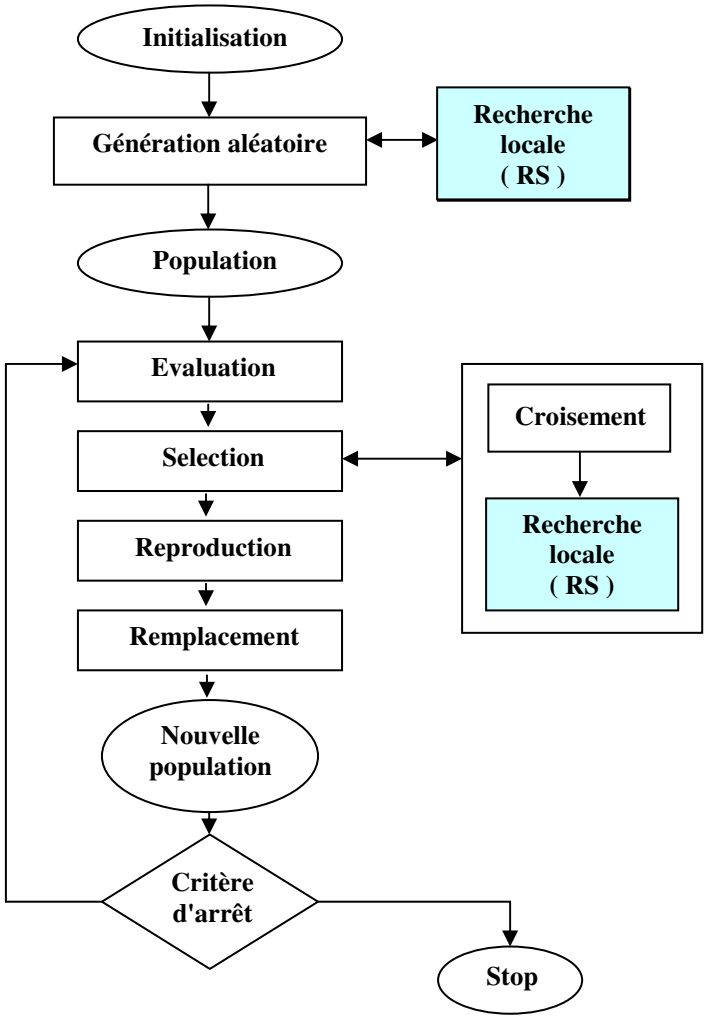

Fig.IV.4. – Les composantes principales de l'algorithme mimétique proposé

IV.3.7. L'algorithme mimétique :

Le pseudo-code de la figure Fig.IV.5 donne les grandes lignes de l'algorithme mimétique utilisé pour la résolution du problème d'analyse de fichier d'audit de sécurité.

```
Procedure Memetic Algorithm;
Entrée : matrice O + matrice AE
Sortie : Vecteur H binaire qui indique les attaques détectées

Début
Initialisation de la  population;
Pour chaque  individu  faire recherche_locale(individu);
Répéter
  Pour individu =1 à #crossovers  faire
    début
     Sélectionner deux parents P1, P2 de la population aléatoirement par roulette;
    enfant := crossover (P1, P2);
    enfant := recherche_locale (enfant);
    ajouter enfant à la population;
     fin;
  population :=  sélection (population);
 Jusqu'à terminer = true;
Fin.
```

Fig.IV.5. – Les grandes lignes d'un algorithme mimétique en pseudo-code

IV.4. Architecture globale de système :

La figure Fig.IV.6 décrit l'architecture globale du système de détection proposé qui se base sur le modèle général proposé par le groupe IDWG (Intrusion Detection Working Group) décrit dans la figure Fig.II.2 du chapitre II.

Ce système adopte l'approche par scénarios qui consiste à rechercher les signatures d'attaques dans le fichier d'audit de sécurité, et 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 filtrées par un capteur qui va les traduit 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 classés par type d'événement.

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

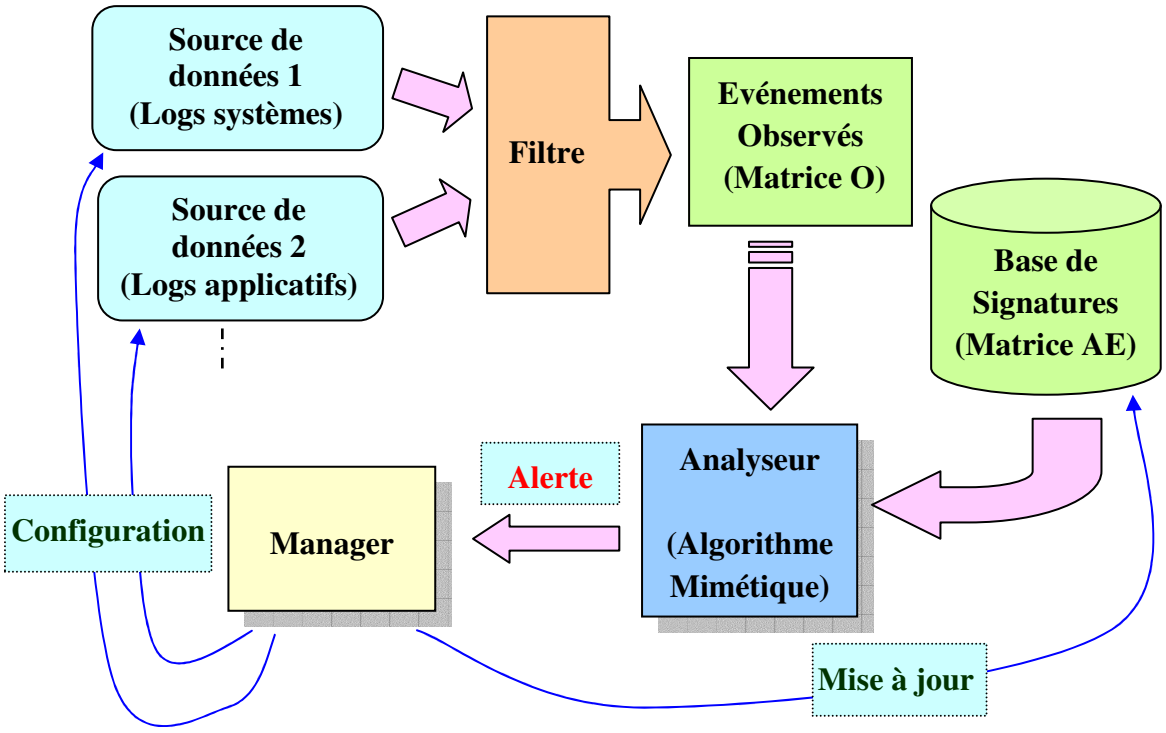

Fig.IV.6 – L'architecture globale du système de détection d'intrusions proposé

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

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 notre algorithme mimétique 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.

IV.5. Conclusion :

Dans ce chapitre, nous avons formalisé le problème de recherche des traces d'attaques dans le fichier d'audit de sécurité, sous forme d'un problème d'optimisation contraint. Le problème en question est un problème très ardu traitant, dans la pratique, un volume de données très important ce qui conduit souvent à des temps de traitements prohibitifs. Pour cela, nous avons proposé un algorithme mimétique évolutionnaire qui utilise la sélection, le croisement comme opérateurs évolutionnaires et une recherche locale de recuit simulé pour améliorer la qualité des solutions. L'architecture globale de notre système incluant le module mimétique est donnée aussi dans ce chapitre.

Le prochain chapitre est consacré à l'implémentation et les résultats numériques.

Chapitre V

Implémentation et résultats

V.1. Introduction :

Dans ce chapitre nous donnons une description de l'implémentation de l'algorithme mimétique décrit dans le chapitre précèdent, et les résultats expérimentaux obtenus après simulations de comportements intrusifs.

Nous présentons dans un premier temps un diagramme de classes pour l'implémentation de cet algorithme mimétique, et un diagramme d'états-transitions. Ensuite nous donnons une description de 24 scénarios d'attaques sous le système AIX (type de système UNIX) et ces traductions en termes d'événements auditables, ce qui a permet de construire une matrice Attaques/Evénements (de taille 28 X 24) pour les expérimentations.

Après la définition de la matrice Attaques/Evénements, nous présentons la matrice d'audit observée qui représente le fichier d'audit obtenu par simulation du comportement d'un utilisateur expérimenté sur une période de 30 minutes. A partir de la matrice d'audit observée nous ajoutons en nombre variable, certains comportements intrusifs de la matrice d'attaques, et nous appliquons l'algorithme mimétique ensuite nous évaluons la qualité d'analyse obtenue.

V.2. Implémentation :

V.2.1. Diagramme de classes :

- Algorithme Mimétique :

Permet de définir l'ensemble des paramètres de l'algorithme mimétique, et permet de démarrer la résolution du problème d'analyse de fichier d'audit de sécurité par le déroulement de l'algorithme mimétique durant plusieurs itérations.

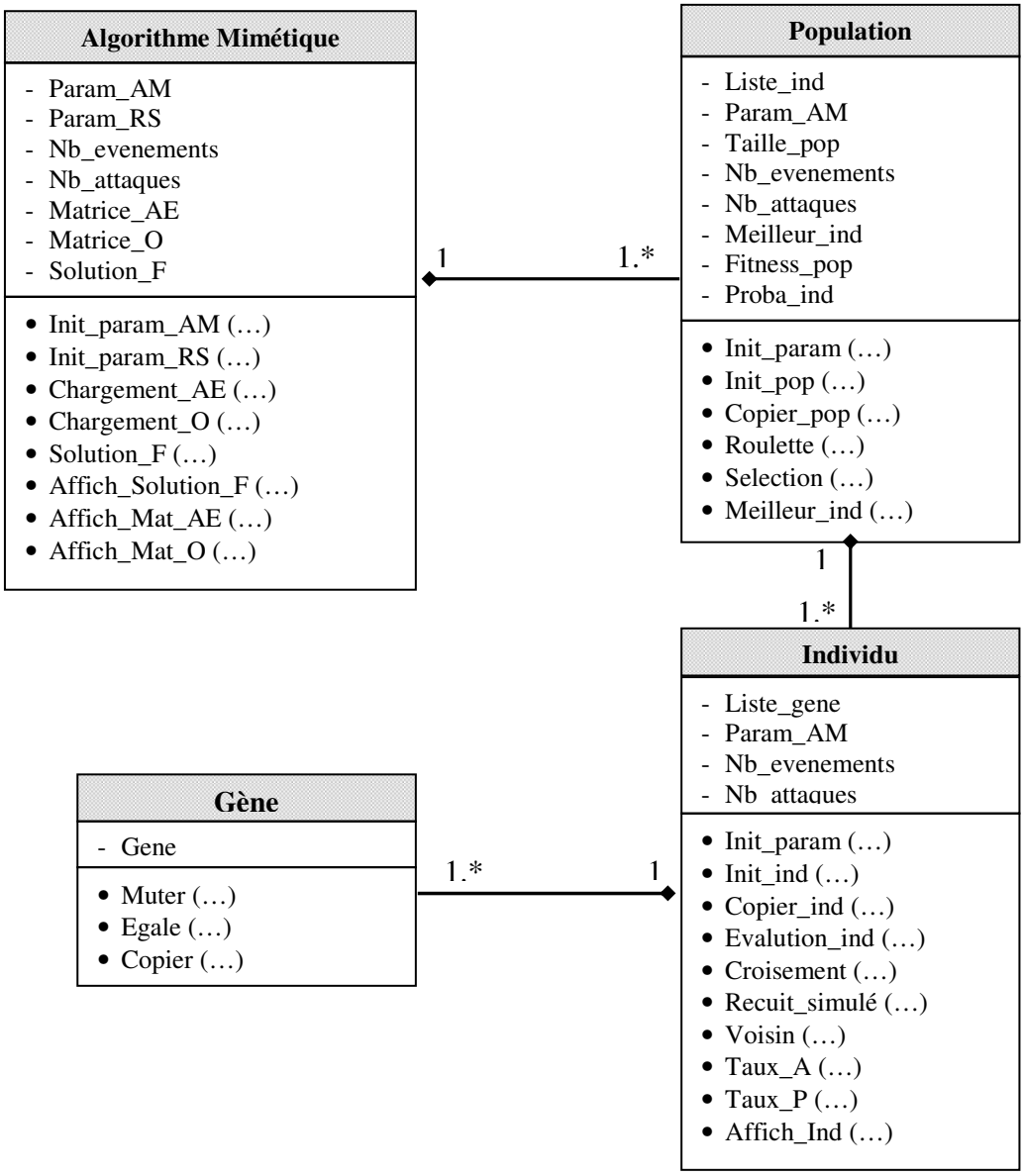

Fig.V.1 – Le diagramme de classes pour l'implémentation de
L'algorithme mimétique proposé.

- 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 et l'application de recuit simulé.

- Gène :

Un gène est associé à un entier de vecteur H (hypothèse) qui peut prendre une valeur égale à 0 ou 1 (Codage binaire).

V.2.2. Diagramme d'états-transitions de l'algorithme mimétique :

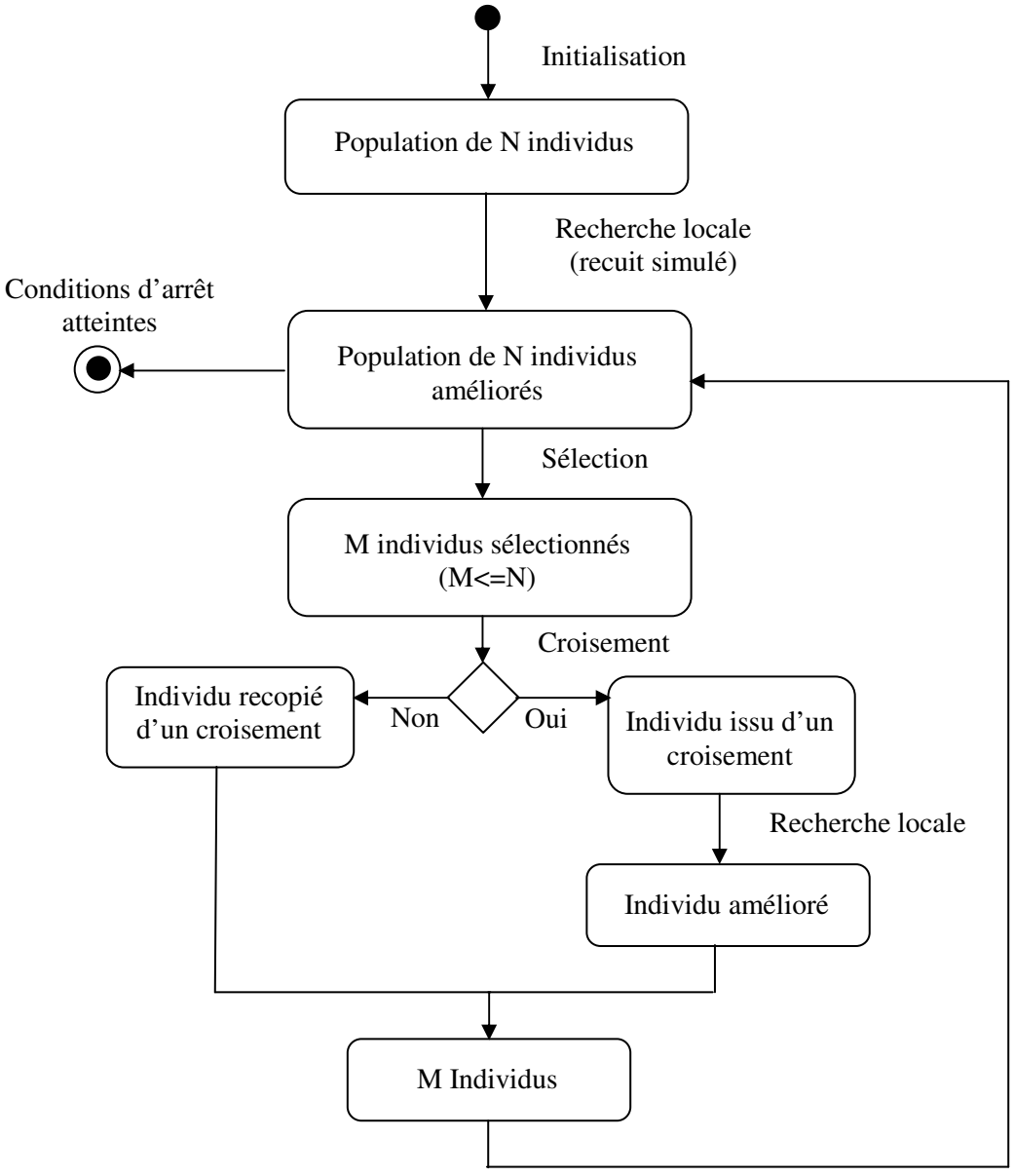

Fig.V.2 – Diagramme d'états de l'algorithme mimétique

V.3. Expérimentations :

Les expérimentations faite sur l'efficacité de l'algorithme proposé utilisent la matrice Attaques/Événements et la matrice d'audit observée qui ont été utilisées pour les tests de simulations décrits dans [Mé94].

A partir de la matrice d'audit observée on va ajouter en nombre variable, certains comportements intrusifs de la matrice d'attaques, et nous appliquons l'algorithme mimétique décrit précédemment et nous évaluons les résultats.

V.3.1. La base de signatures :

Le fichier d'audit de sécurité qui fera l'objet de nos expérimentations c'est le fichier d'audit généré par le système UNIX. Un enregistrement d'audit peut contient les informations suivantes :

- Le type de l'événement.
- Le nom de l'utilisateur pour le compte duquel s'exécute le processus qui a généré l'événement.
- Le résultat de l'événement (échec ou succès).
- 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 4 classes :

- 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.

La base de signatures utilisée dans les expérimentations est sous forme d'une matrice Attaques/Evénements (de taille 28 X 24) qui a été construite à partir de 24 scénarios d'attaques ou d'actions litigieuses réalistes que l'on peut redouter lorsque on travaille sous le système UNIX. 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 et 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}.

V.3.2. Source de données :

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

La matrice d'audit observé que nous venons de l'utiliser est décrite dans le tableau Tab.V.2, elle 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 son ceux définis dans l'ensemble des attaques de la matrice d'attaques/événements donnée en tableau Tab.V.1.

Événement		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.V.1. – Matrice Attaques/Evénements utilisée dans les expérimentations.

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

Tab.V.2 – Matrice d'audit observée représente le comportement d'un utilisateur expérimenté sur une période de 30 minutes

V.3.3. Résultats :

Après que nous avons défini la matrice attaques/événements et la matrice d'audit observée, on va analyser cette dernière en utilisant l'algorithme défini dans la figure Fig.IV.4 de chapitre IV. A chaque expérimentation on va introduit des comportements

intrusifs dans la matrice d'audit observée avec un nombre variable.

Pour l'implémentation de l'algorithme mimétique 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 avons travaillé avec une matrice R unité (c-à-d $R_i = 1$ pour i = 1 à N_a), pour ne pas privilégier une attaque par rapport les autres, et aussi pour facilité l'interprétation des résultats.

Les paramètres de l'algorithme mimétique ont été fixés comme suit :

- La taille de population = 100,
- La probabilité de croisement = 0,7,
- le nombre maximum de générations = 20.

Pour la technique de recherche locale les paramètres sont :

- Le nombre d'itérations par palier N = 10,
- La température initiale $T^0 = 20$.

Toutes les expériences ont été exécutées sur un ordinateur de CPU Intel Celeron 1.70 Ghz avec 128 MO de RAM.

Pour caractériser les performances de l'algorithme proposé, nous définissons les taux suivants :

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

On s'intéresse au T_{Pf} et T_{Af} : les taux T_P et T_A à la génération finale. Les taux T_{P0} et T_{A0} de la génération initiale sont approximativement : $T_{P0} \approx 0,5$ 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 gènes égaux à 1 au locus correspondant à une attaque présente, et des gènes égaux à 0 au autres locus, on a donc $T_{Pf} = 1$ et $T_{Af} = 0$.

Nous observons l'évolution de T_P et T_A en fonction du nombre de génération et en fonction du nombre d'attaques présentes dans la matrice d'audit. Dans nos expériences on va varier le nombre d'attaque de 0, 2, 4, 8, 10, 15, et 20.

Dans ci-après nous donnons les courbes qui présentent les évolutions moyennes (sur 10 exécutions) des valeurs sélectives 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.3 à Fig.V.9).

La figure Fig.V.10 donne des exemples d'évolution des valeurs sélectives minimale, maximale et moyenne pour une seule exécution dans le cas où Nb_attaques = 2 en fonction de générations. Les figures Fig.V.11 et Fig.V.12 montrent les évolutions des taux T_A et T_P en fonction de générations pour une seule exécution (pour Nb_attaques = 2).

À travers des courbes des évolutions des taux T_A et T_P on peut conclure que dans le cas d'une matrice AE contenant 24 attaques notre algorithme permet une excellente discrimination entre attaques présentes dans la matrice d'audit analysée et attaques absentes de cette matrice, et ce quelque soit le nombre d'attaques réellement présentes dans la matrice d'audit analysée O comme le montre les figures Fig.V.13 et Fig.V.14 (le taux T_A entre 0.003 et 0.007, et le taux T_P entre 0.971 et 0.979).

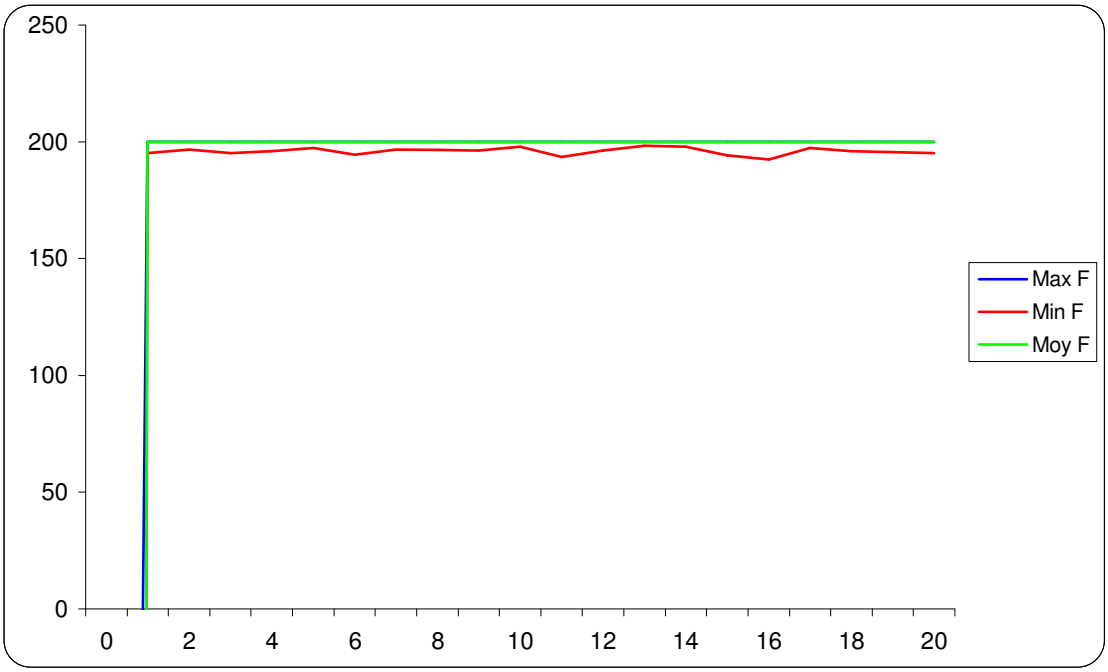

Fig.V.3 - Exemple d'évolution moyenne (sur 10 exécutions) des valeurs sélectives minimale, maximales et moyennes en fonction de la génération pour Nb_attaques = 0.

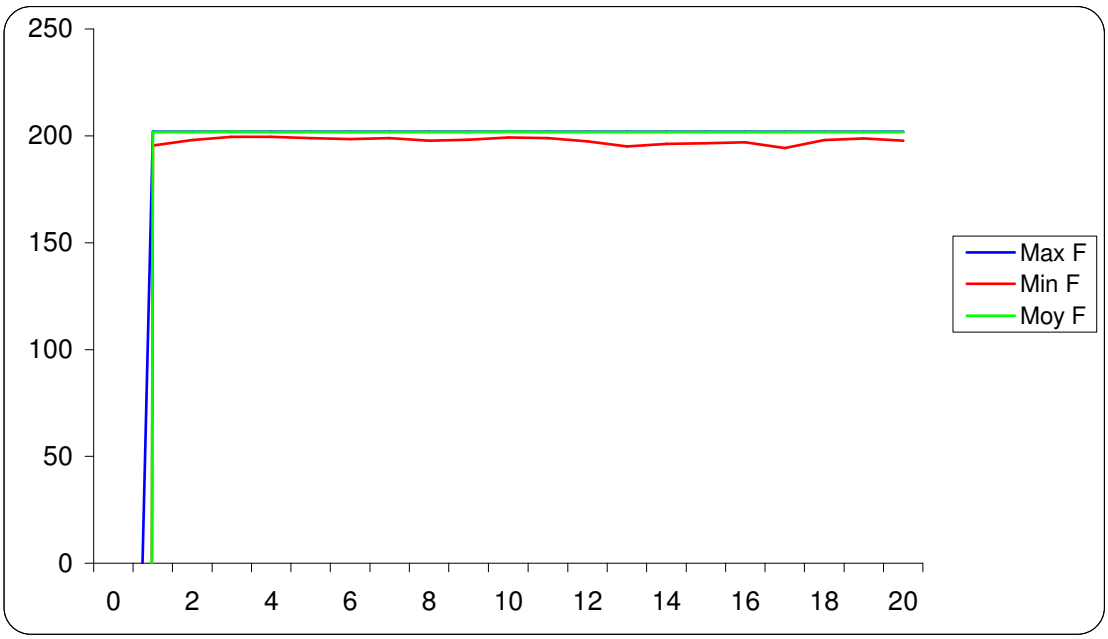

Fig.V.4 - Exemple d'évolution moyenne (sur 10 exécutions) des valeurs sélectives minimale, maximales et moyennes en fonction de la génération pour Nb_attaques = 2.

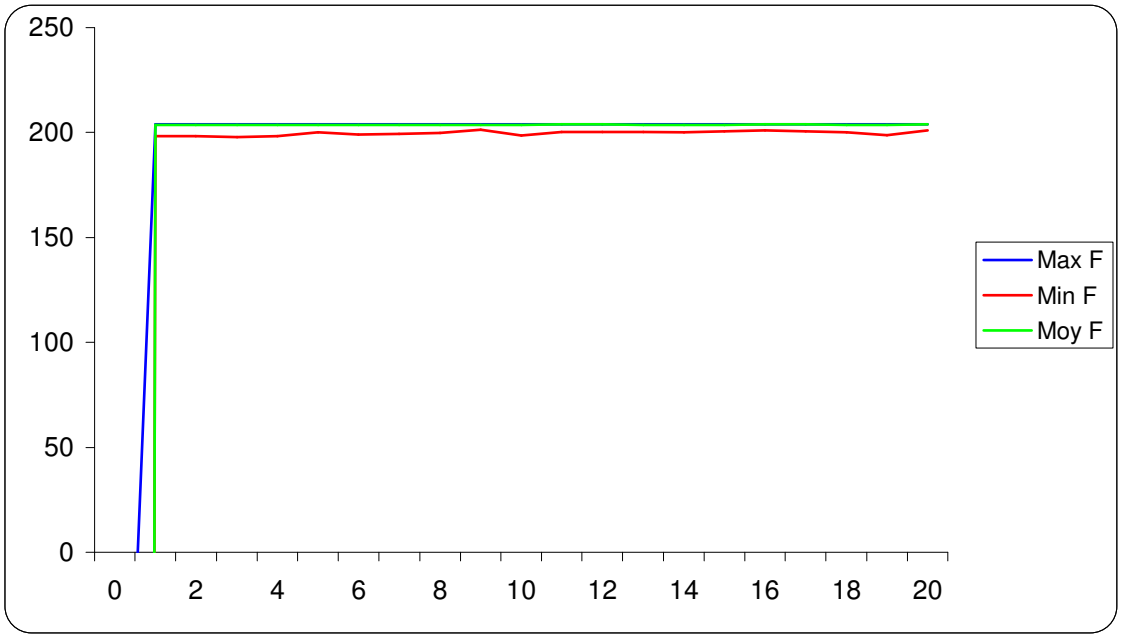

Fig.V.5 - Exemple d'évolution moyenne (sur 10 exécutions) des valeurs sélectives minimale, maximales et moyennes en fonction de la génération pour Nb_attaques = 4.

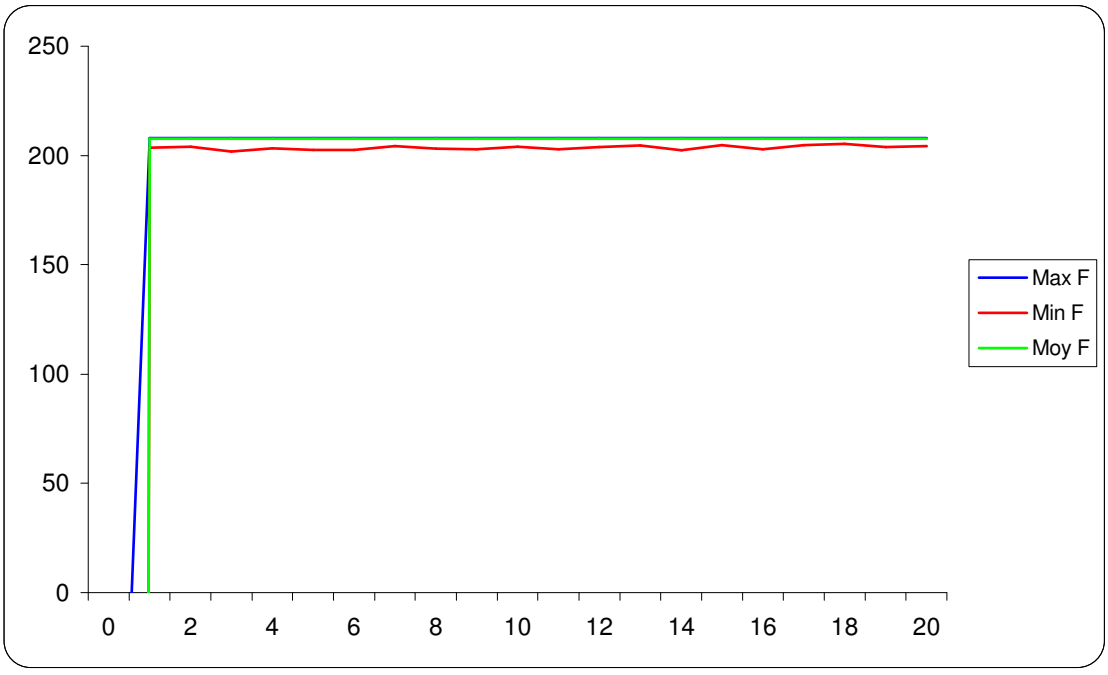

Fig.V.6 - Exemple d'évolution moyenne (sur 10 exécutions) des valeurs sélectives minimale, maximales et moyennes en fonction de la génération pour Nb_attaques = 8.

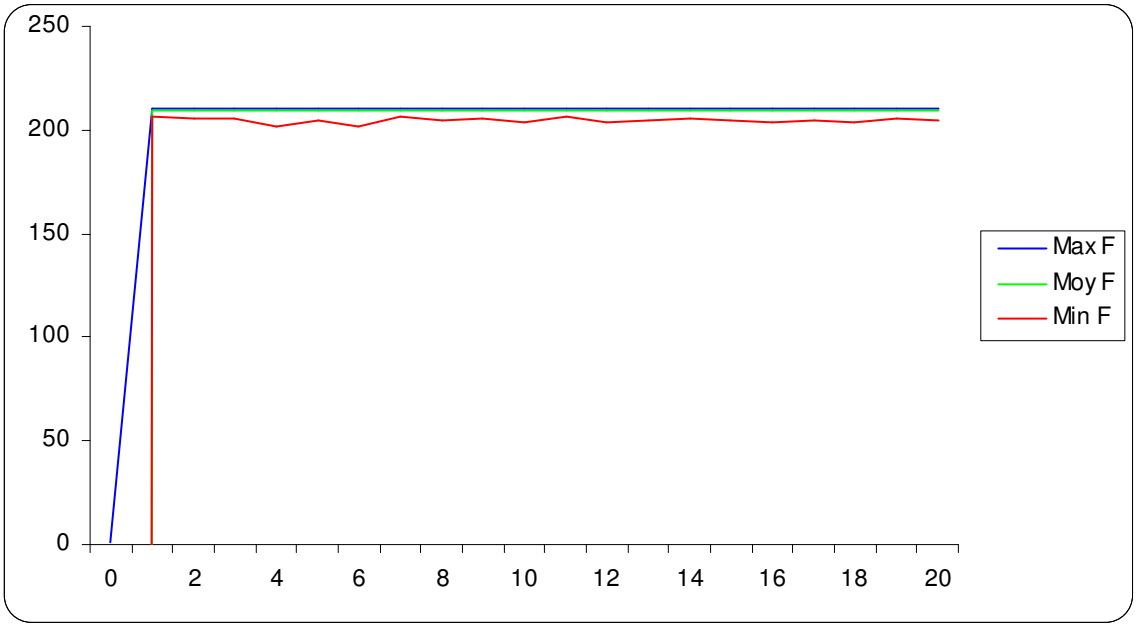

Fig.V.7 - Exemple d'évolution moyenne (sur 10 exécutions) des valeurs sélectives minimale, maximales et moyennes en fonction de la génération pour Nb_attaques = 10.

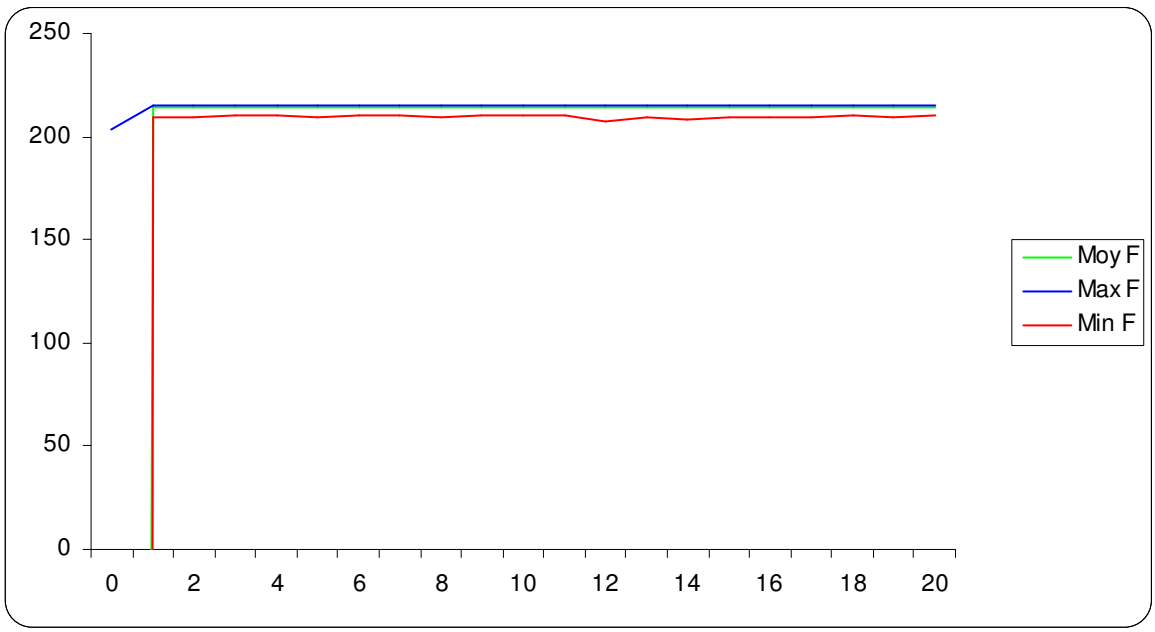

Fig.V.8 - Exemple d'évolution moyenne (sur 10 exécutions) des valeurs sélectives minimale, maximales et moyennes en fonction de la génération pour Nb_attaques = 15.

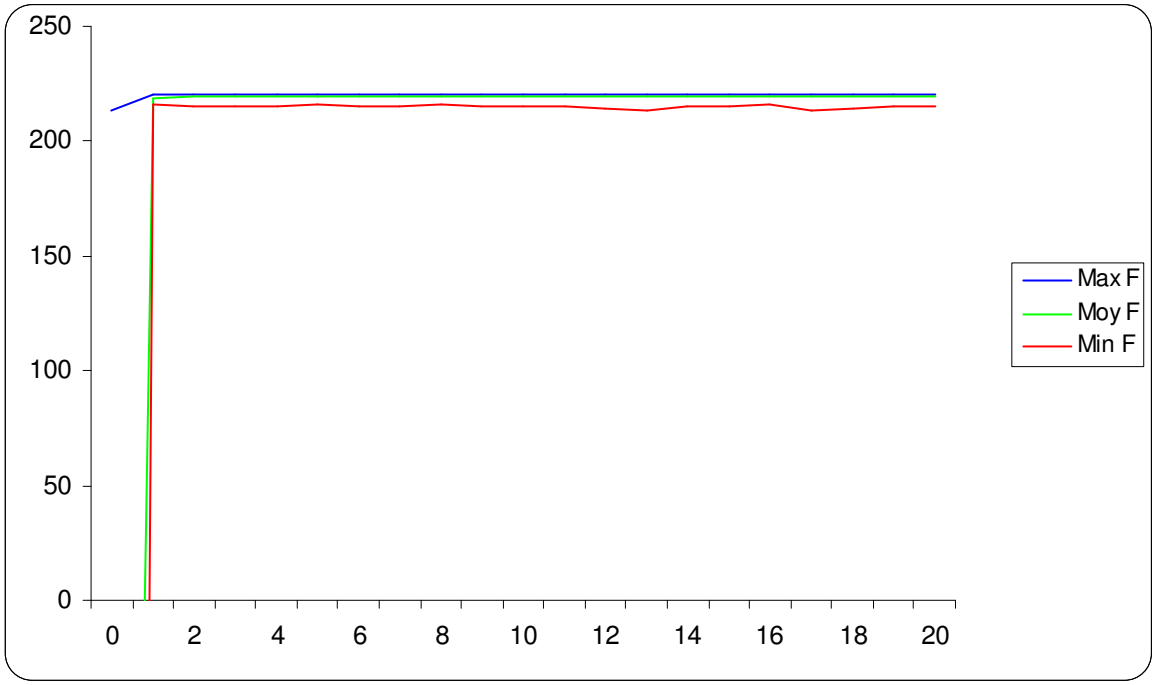

Fig.V.9 - Exemple d'évolution moyenne (sur 10 exécutions) des valeurs sélectives minimales, maximales et moyennes en fonction de la génération pour Nb_attaques = 20.

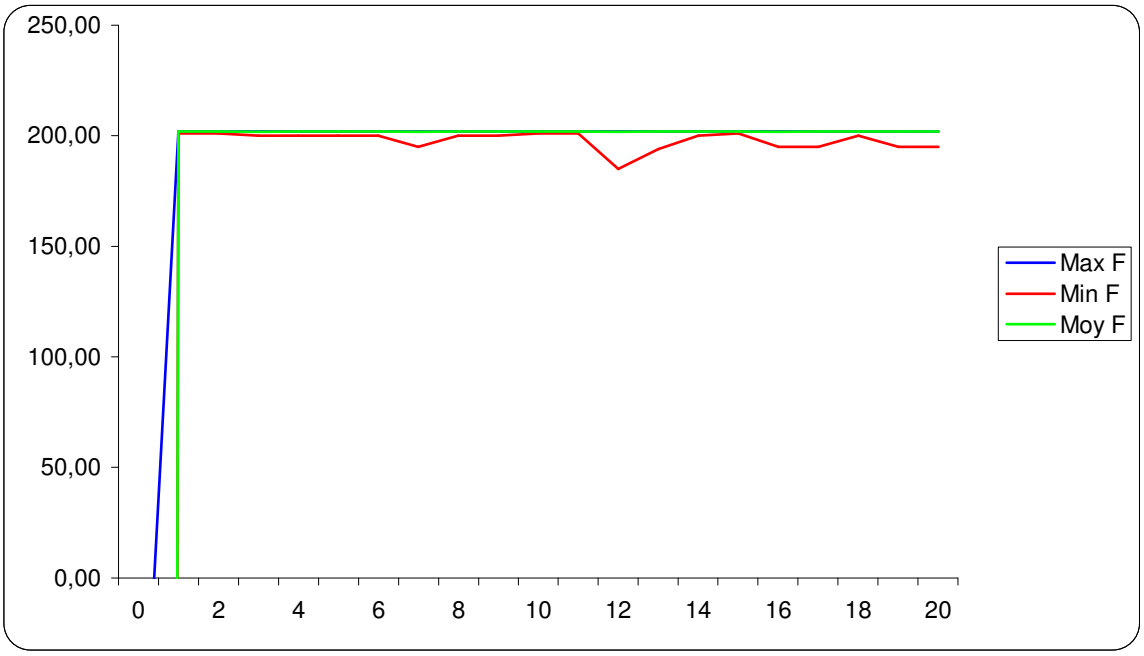

Fig.V.10 - Exemple d'évolution des valeurs sélectives minimale, maximale et moyenne en fonction de la génération pour une seule exécution et pour Nb_attaques =2

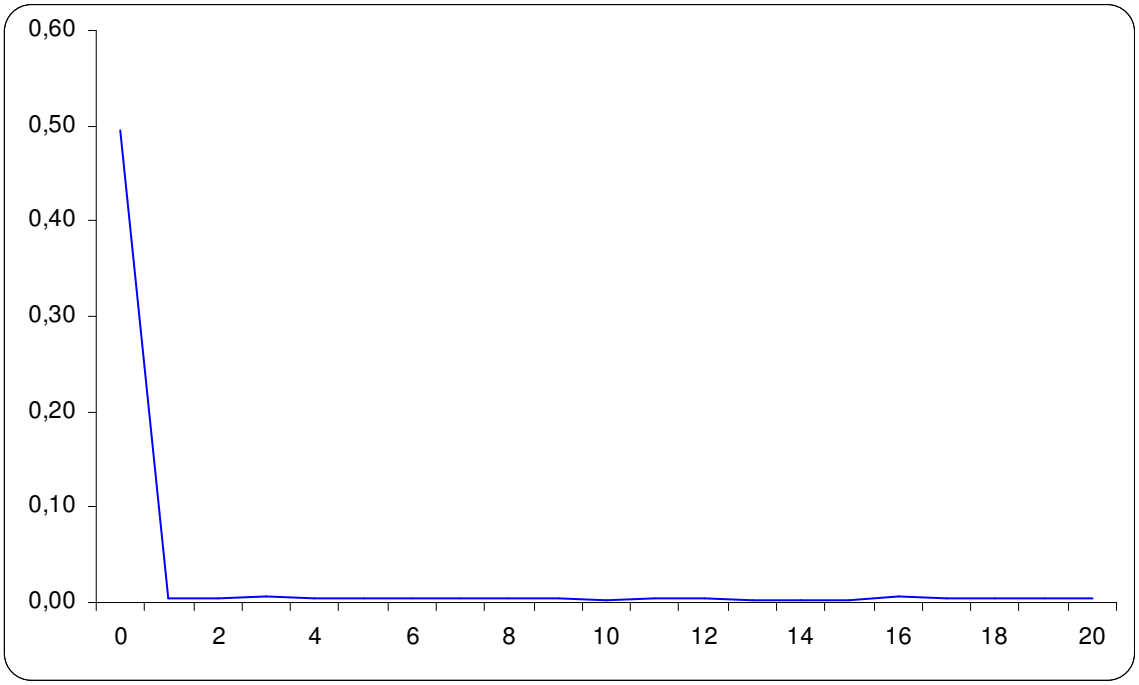

Fig.V.11 - Evolution de taux Ta en fonction de la génération pour une seule exécution, pour le cas où Nb_attaques = 2.

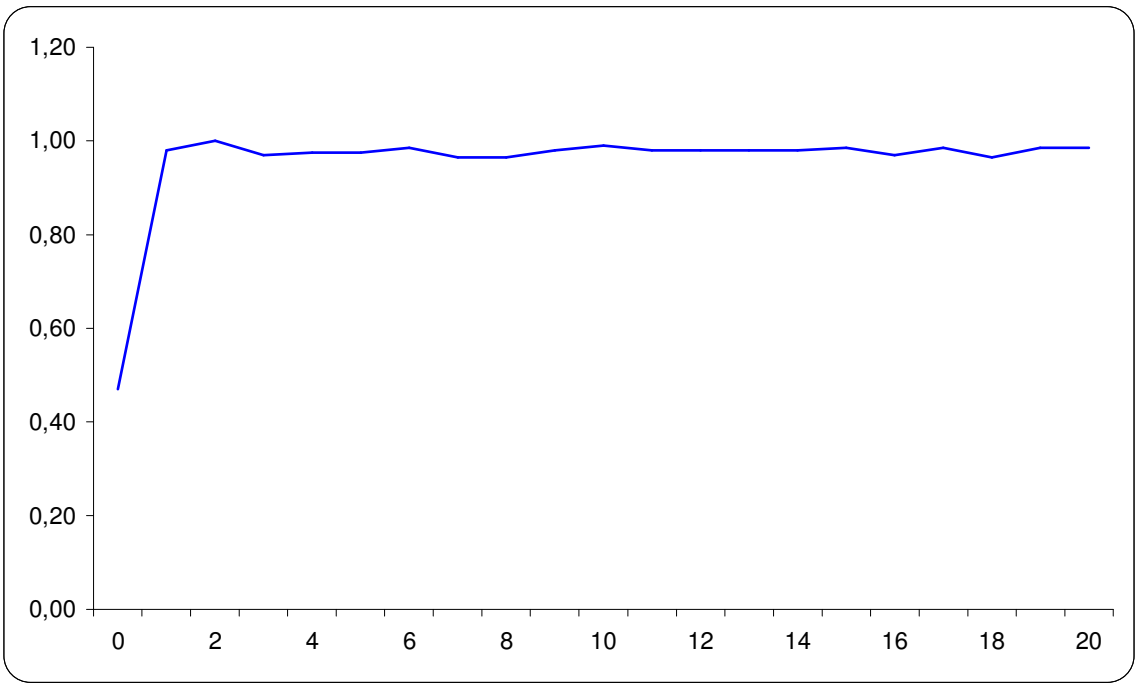

Fig.V.12 - Evolution de taux Tp en fonction de la génération pour une seule exécution, pour le cas où Nb_attaques = 2.

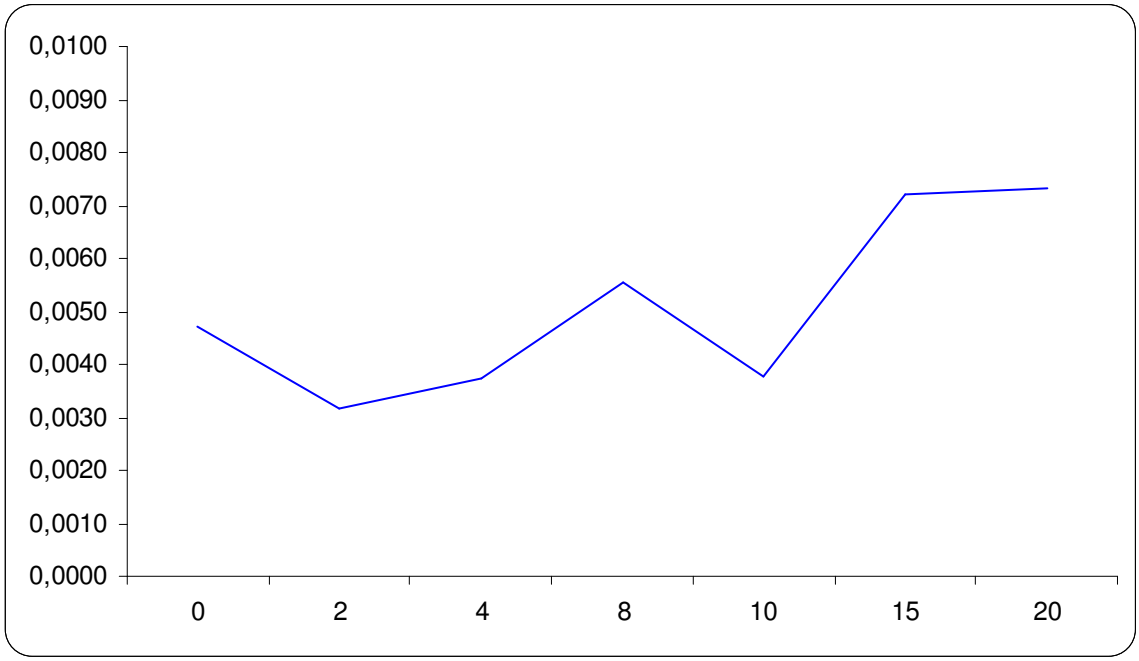

Fig.V.13 - Evolution moyenne (sur 10 exécutions) de taux Ta en fonction du nombre d'attaques présentes dans le fichier d'audit.

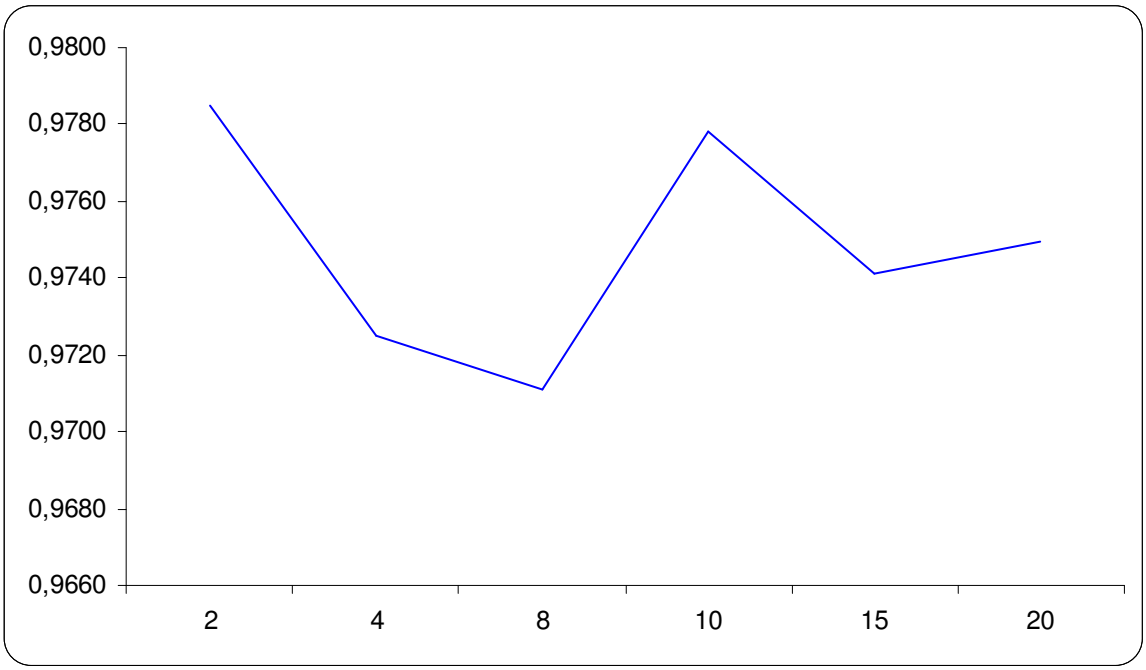

Fig.V.14 - Evolution moyenne (sur 10 exécutions) de taux Tp en fonction du nombre d'attaques présentes dans le fichier d'audit.

V.4. Conclusion :

Nous avons présenté dans ce chapitre les grandes caractéristiques de l'algorithme mimétique utilisé pour la recherche des attaques dans le fichier d'audit de sécurité, par un diagramme de classes qui comporte les différentes classes utilisées pour l'implémentation cet algorithme, et par un diagramme d'états-transitions.

Pour le test de l'implémentation nous avons utilisé 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 demi-heure, 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 toutes les attaques introduites sont détectées pour tous les cas de nombre d'attaques variant de 0 à 20, et la performance de l'algorithme proposé est presque indépendante de la matrice d'audit observée.

Conclusion générale

La détection d'intrusions est une phase importante et complémentaire à la phase de prévention dans la sécurité des systèmes informatiques. Dans ce mémoire, nous avons proposé une approche par scénarios pour la détection d'intrusions. Elle consiste à rechercher les scénarios d'attaques prédéfinis dans les traces d'audit de sécurité, en utilisant un algorithme mimétique pour l'analyse de fichier d'audit. Les scénarios d'attaques sont définis sous forme de matrice attaques/événements (AE), où chaque élément AE_{ij} de cette matrice indique le nombre d'occurrences de événement i dans l'attaque j. Ceci permet 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 est basée 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 initial, et une analyse ultérieure permettra de localiser précisément les attaques détectées. Le problème d'analyse de fichier d'audit est exprimé sous forme d'une optimisation contrainte, et un algorithme mimétique est appliqué pour sa résolution. L'algorithme utilise des opérateurs génétiques à chaque génération : la sélection, le croisement et une phase d'amélioration des individus basée sur la méthode de recuit simulé. Le codage des individus est binaire où chaque bit code la présence ou l'absence d'une attaque indépendamment de celle des autres. Pour l'évaluation des individus, une fonction sélective, qui prend en compte les contraintes du problème, est utilisée.

L'algorithme proposé a été implémenté et des expérimentations ont été réalisées en 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 a pu 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.

Il faut noter que la mise en œuvre de cette approche nécessite un bon choix de paramètres de l'algorithme mimétique qui n'est pas facile, tel que la taille de la population, le nombre maximum de génération, la probabilité de croisement,…etc. L'ajustement des paramètres est effectué d'une manière empirique.

La réalisation de ce travail a été très fructueuse vu son apport considérable en matière de connaissances acquises sur le domaine de la sécurité informatique, et précisément les systèmes de détection d'intrusions et sur l'implémentation des algorithmes mimétiques qui englobe les opérateurs génétiques et la recherche locale.

Comme perspectives de ce travail, nous prévoyons :

- Le test de cet algorithme sur des matrices d'Attaques/Evénements plus grande (contenant un nombre important d'attaques), et si possible sur un environnement réel.

- La standardisation des alertes résultantes de l'analyse (sous format IDMEF - Intrusion Detection Message Exchange Format -) pour permettre une coopération avec d'autres IDS et une possibilité de corrélation d'alertes.

- La comparaison des résultats d'utilisation de l'algorithme mimétique proposé, un algorithme génétique et la méthode de recuit simulé, de point de vue de la qualité de diagnostic et du temps de calcul.

Annexe

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

Bibliographie

[Abka03] 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

[Alliot] Jean-Marc Alliot "Les algorithmes génétiques". www.recherche.enac.fr/opti/papers/thesis/HABIT/index.html.

[Aldu05] Jean-Marc Alliot, Nicolas Durand. Algorithmes génétiques. Mars 2005.

[BoDr04] Dalila BOUGHACI, Habiba Drias. A Performance Comparison of Evolutionary Meta-heuristics and Solving MAX-SAT Problems ENFORMATIKA V1 2004. ISSN : 1305-5313

Boud00] Détection d'intrusions : Une nouvelle approche par systèmes multi-agents, Karima Boudaoud, Ecole Poly echnique Fédérale de Lausanne, 2000.

[Cal03] Calyx Netsecure. Netsecure web. http://www.calyxnetsecure.com/download/NSWeb-400/doc-fr/White_Paper.pdf, 2003.

[Cedr03] 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.

[Cert] Site Web de CERT : www.cert.org/stats/.

[Chan02] CHAN Yew Cheong, Peter. "La planification du personnel : acteurs, actions et termes multiples pour une planification opérationnelle des personnes". *Thèse de Doctorat. L'université JOSEPH FOURIER – Grenoble 1.* Octobre 2002.

[Chva95] Christophe Bidan et Valérie Issarny, Un aperçu des problèmes de sécurité dans les systèmes informatiques, Octobre 1995

[Cis98] Cisco Systems. Netranger intrusion detection system technical overview. http ://www.cisco.com/warp/public/778/security/netranger/ntran_tc.pdf, December 1998.

[Dam02] Olivier Damien. Recherche stochastique : Méthode du recuit simulé. 2002. www.lih.univ-lehavre.fr/~olivier/Enseignement/Maitrise/TD/recuit/TDrecuit.html.

[DBS92] 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.

[DDW98] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. Internal RZ 3030, IBM Zurich Research Laboratory, Saumerstrasse 4, CH-8803 Ruschlikon, Switzerland, Juin 1998.

[Digma03] J. Digalakis , K. Margaritis. "Performance comparison of memetic algorithms". Parallel and Distributed Processing Laboratory, University of Macedonia, Greece. 2003.

[Dist05] Kate Distin. The Selfish Meme, A Critical Reassessment. Cambridge University Press. New York. 2005.

[Fesc05] Ferdinand Kobelt, Tom Schmidt. "La sécurité informatique - une affaire des dirigeants".https://www2.eycom.ch/publications/items/praxis/20050542/fr.pdf.

[Firew] site : http://webenic.enic.fr/~vanoudendycke/firewall.html.

[Frme02] Frédéric Meunier Détection d'intrusions : Notions avancées de NIDS axées sur le logiciel ManHunt (recourse technologies), Août 2002. www.recourse.com

[Guill] site Guill : http://www.guill.net

[Gonz04] Gonzalez Daniel "Les algorithmes génétiques". Février 2004.

[JVL+93] Javitz. H.S, Valdes. A, Lunt. T.F, Tamaru. A, Tyson. M et Lowrance. J. Next Generation Intrusion Detection Expert System (NIDES). Rapport technique, A016-Rationales, SRI, 1993.

[KraSm] Natalio Krasnogor et Jim Smith. Competent Memetic Algorithms: Model, Taxonomy and Dessing Issues

[Lalet89] J.M.Lamère, Y.Leroux, J.Tourly : la sécurité des réseaux méthodes et techniques, Dunod Paris 1989.

[Lass00] Lassoued Yassine. Etude du paramétrage d'un algorithme d'optimisation hybride. Septembre 2000.

[LuJ88] T.F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. In Proceedings of the IEEE Symposium on Security and Privacy, pages 59-66, 1988

[Lut04] Evelyne LUTTON. Darwinisme artificiel : une vue d'ensemble. INRIA - Rocquencourt - Equipe COMPLEX - Projet FRACTALES, France. février 2004

[Maiw01] Eric Maiwald. "Sécurité des réseaux", Compus Press 2001.

[Mé03] Ludovic Mé, Détection des intrusions dans les systèmes d'information: la nécessaire prise en compte des caractéristiques du système surveillé. Université de Rennes1, juin 2003.

[Mé94] Ludovic Mé. Audit de sécurité par algorithmes génétiques. Thèse de doctorat. Université de Rennes 1. juillet 1994.

[Mé98] Ludovic Mé. GASSATA, A Genetic Algorihm as an Alternative Tool for Security Audit Trails Analysis. In First international workshop on the Recent Advances in Intrusion Detection. http://www.zurich.ibm.com/~dac/Prog_RAID98/Table_of_content.html.1998.

[Méal96] Ludovic Mé - Véronique Alanou. Détection d'intrusion dans un système informatique : méthodes et outils. 1996.

[Mécéd99] Ludovic Mé et Cédric Michel. La détection d'intrusions : bref aperçu et derniers développements. mars 1999.

[Mosca] Pablo Moscato. "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms".Caltech Concurrent Computation Program 158-79. California Institute of Technology.U.S.A.

[Post03] D. Powell, R. Stroud (Eds.), Malicious and Accidental Fault Tolerance for Internet Applications : Conceptual Model and Architrecture, Final Version, Rapport LAAS, janvier 2003.

[Price98] K. Price. *Intrusion Detection Page*s. Purdue University, 1998. http://www.cs.purdue.edu/coast/intrusion-detection/ids.html.

[RaSue04] Randy L. Haupt et Sue Ellen Haupt. PRACTICAL GENETIC ALGORITHMS. Deuxième édition. WILEY-INTERSCIENCE. A JOHN WILEY & SONS, INC., PUBLICATION. 2004.

[Sie99] Sielken Robert S. Application Intrusion Detection. Rapport technique. Dept. of Computer Science, University of Virginia, juin 1999.

[Sqa04] Samuel Pierre, Alejandro Quintero et Agnès de Montgolfier. Approches heuristiques pour l'affectation de cellules aux commutateurs dans les réseaux mobiles. IEEE Canadian Review, N°: 46, 2004.

[Sys98] Systems (Internet Security). Network- vs. host-based intrusion detection : A guide to intrusion detection technology. http://www.iss.net/prod/nvh_ids/nvh_ids.pdf, Octobre 1998.

[Val89] H.S. Vaccaro and G.E. Liepins. Detection of anomalous computer session activity. In Proceedings of the IEEE Symposium on Security and Privacy, Mai 1989.

[VK99] Vigna (Giovanni) et Kemmerer (Richard A.). Netstat : A network-based intrusion detection system. Journal of Computer Security, Fevrier 1999.

[Wid01] Marino WIDMER. "Les métaheuristiques : des outils performants pour les problèmes industriels". *3ème Conférence Francophone de Modélisation et Simulation*. avril 2001.

[Wiki] http://fr.wikipedia.org/wiki/Algorithme génétique

[Woerl02] Mark Wood and Mije Erlinger. Intrusion detection message exchange requirements. Intrusion Detection Working Group, Internet-Draft. draft-ietf-idwg-requirements-10, Octobre 2002.

[Zimé02] Jacob Zimmermann et Ludovic Mé. Les systèmes de détection d'intrusions : principes algorithmiques. Supélec, équipe SSIR. 2002.

[ZMB02] 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).