République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master en informatique
Option : Réseaux et Systèmes Distribués

Thème

Génération automatique des scénarios d'attaques dans les systèmes informatiques

---

Présenté par

MESSOUAF Sonia

Devant le jury composé de :

M^{lle} BATTAT Nadia	Présidente	Université de Béjaia
Mr AISSANI Soufiane	Examinateur	Université de Béjaia
M^{lle} HOUARI Rima	Examinatrice	Université de Béjaia
M^{lle} HAMZA Lamia	Promotrice	Université de Béjaia

Promotion 2012-2013

Remerciements

A l'issu de ce travail je tiens à remercier en premier lieu le bon Dieu tout puissant de m'avoir donné la force pour réaliser ce modeste travail.

Je remercie grandement ma promotrice madame HAMZA Lamia, sans lequel mon travail et l'aboutissement de ce mémoire, n'auraient pas vu le jour. Merci pour l'honneur que vous m'avais fait en acceptant de m'encadrer. votre disponibilité, vos orientations et vos encouragements.

Veuillez trouver dans ce travail, l'expression de ma profonde gratitude pour l'immense aide que vous m'avez toujours apportée. Avec toute mon estime, respect et l'admiration que je vous porte, mes souhaits les plus sincères de joie et plus de réussite.

Je remercie également les membres de jury d'avoir acceptés de juger mon travail et consacré leurs temps à la lecture de ce mémoire.

Je remercie tous ceux qui m'ont apporté leur aide.

Sans oublier de remercier les enseignants qui ont contribué à notre formation et appuyé notre cursus universitaire, et le personnel administratif de département d'informatique.

Je conclurai, en remerciant vivement ma famille respective qui m'a toujours supporté moralement et financièrement pendant toutes mes années d'études.

Dédicaces

Je dédie ce modeste travail à :

- Mon cher père qui a veillé à ma réussite en déployant tous les efforts nécessaires et qui a été toujours un exemple pour moi.
- Ma chère mère pour tout son amour et son soutien. Quoique je puisse dire, je ne peux exprimer mes sentiments d'amour et de respect à ton égard et ma gratitude.
- Mes cher frères abdelhalim,nassim et boubaker , à qui je souhaite une très grande réussite dans leurs vie.
- Mon adorable fiancé, ton soutien et amour ont été pour moi une source de courage et de confiance.
- à toute ma famille de prés et de loin .
- à mes copines de la chambre B35 de la résidence Aamriw.
- je ne peux oublier l'ensemble de mes amis qui m'ont apporté leur soutien et je pense tout particuliérement à tounes, katia, nassima.

Sonia

Table des matières

Liste des figures

Introduction générale

Les systèmes d'information et les réseaux jouent un rôle grandissant dans la vie quotidienne et dans les entreprises. Ainsi la notion du risque lie à ces derniers devient une source d'inquietude et une donnée importante à prendre en compte. Les attaques réalisées par des utilisateurs malveillants et visant à exploiter les vulnérabilités de système d'information sont plus en plus fréquentes. Les mécanismes de détection d'intrusion sont pas efficace pour protéger les réseaux.
Dans ce cas, il est donc essentiel de générer les graphes d'attaques pour l'analyse et la configuration de la sécurité réseau ainsi que pour la détection des attaques complexes.

Notre objectif est de réaliser une application permettant de générer automatiquement des graphes d'attaques à partir des réseaux et une logique de l'intrus introduits par l'utilisateur.

Le mémoire est organisé de la manière suivante :

Dans le premier chapitre, nous donnons des généralités sur la sécurité informa-

tique. Dans le deuxième chapitre, nous présentons l'état de l'art que nous avons fait dans le cadre de la génération des graphes d'attaques. Dans le troisième chapitre, nous présentons la démarche étudiée et l'algorithme de construction du graphe d'attaque. Dans le quatrième chapitre, nous trouvons la modélisation et la conception de l'application. Dans le dernier chapitre consiste à la réalisation de l'application.

1

Généralités sur la sécurité informatique

1.1 Introduction

La sécurité des systèmes informatiques consiste à protéger l'accès et la manipulation des données et des ressources d'un système par des mécanismes d'authentification, d'autorisation, de contrôle daccès, etc. Cependant, avec l'ouverture des entreprises et des personnes à Internet, l'assurance de la sécurité des systèmes devient très difficile, du fait que, les attaques et les intrusions augmentent de plus en plus et deviennent de jour à l'autre plus complexes et difficiles à éviter.

Ce chapitre introductif présente les notions de base de la sécurité des systèmes informatiques et les différentes attaques possibles de se produire ainsi que les méca-

nismes et outils pouvant être mis en en place pour assurer la sécurité.

1.2 Sécurité informatique

La sécurité informatique est l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaire et mis en place pour réduire la vulnérabilité d'un système contre les menaces accidentelles et intentionnelles [**1**]. La sécurité consiste à assurer :

-La confidentialité : assurer que l'information ne sera lue que par les personnes autorisées.

-L'intégrité : assurer que les informations ne peuvent être modifier ou altérer que par les personnes autorisées.

-La disponibilité : assurer que l'information est disponible pour les personnes autorisées.

-L'authentification : vérifier l'identité d'un utisateur pour lui associer des droits d'accès.

-La non-répudiation : garantir qu'aucun des correspondants ne pourra nier la transaction (l'envoi ou la réception des données).

1.3 Attaques informatiques

Une *attaque* est l'exploitation d'une faille d'un système informatique à des fins non connus par l'exploitant du système et est généralement préjudiciable.

Les objectifs d'un attaquant sont diverses :[2]

-Vols d'informations.
-Terrorisme, espionnage, chantage.
-Attirer l'attention.
-Avantages concurrentiels et bénéfices financiers.
-Vérification de la sécurité d'un système.

1.3.1 Etapes d'une attaque

Dans ce qui suit nous allons présenté les étapes d'une attaque [3] :

• **Reconnaissance :** il est logique pour un attaquant de chercher les informations nécessaires sur les victimes potentielles avant de les cibler avec les outils d'attaques les plus appropriés (codes d'exploits, toolkits, etc.).

• **Gain d'accès (Gain Access) :** afin d'atteindre leurs objectifs, les attaquants ont généralement besoin d'avoir un accès aux ressources des victimes ; le niveau d'accès requis dépend évidemment de l'attaque. Notons toutefois que certains types d'attaques, comme les attaques en déni de service, n'ont pas besoin d'accès sur la machine victime.

• **Augmentation de privilèges (Privilege Escalation) :** après avoir acquis suffisamment de privilèges, l'attaquant essaie généralement d'explorer la machine ou

le réseau cible (par exemple, en fouillant les fichiers et les répertoires), pour rechercher un compte particulier (comme un compte invité ou un compte ftp anonyme), pour identifier les composants matériels, pour identifier les programmes installés, pour rechercher les hôtes de confiance (typiquement, ceux ayant des certificats installés sur la machine de la victime), etc..

- **Actions principales (Principal Actions) :** cette étape peut prendre différentes formes ; par exemple, l'attaquant peut exécuter une attaque en déni de service, installer un code malveillant, compromettre l'intégrité des données ou exécuter un programme.

- **Cacher les traces (Hiding Traces) :** les attaquants les plus expérimentés utilisent généralement cette dernière étape pour effacer leurs traces et rendre ainsi la détection plus difficile.

1.3.2 Les types d'attaques

Il existe plusieurs types d'attaques,ces attaques peuvent être regroupées en trois grandes classes qui sont :[2]

1.3.2.1 Les attaques directes

C'est les plus simples attaques, l'intrus attaque directement la victime à partir de son ordinateur.

L'avantage de ces attaques est qu'on peut remonter facilement à l'origine.

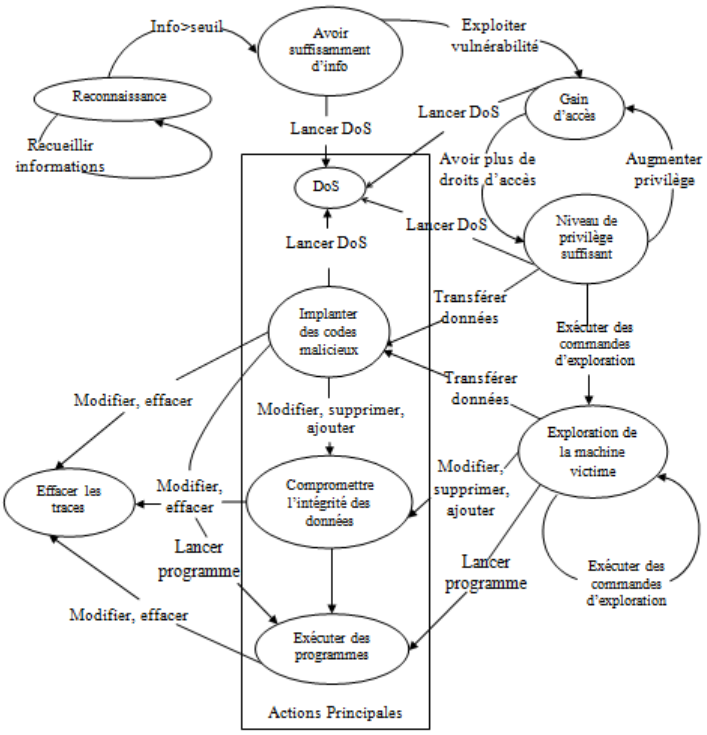

FIGURE 1.1 – Exemple pour les étapes d'une attaque [3]

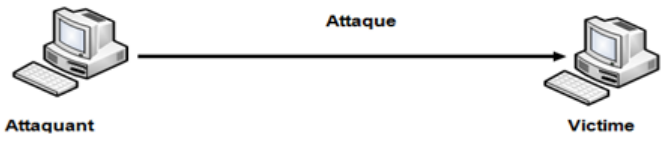

FIGURE 1.2 – Attaque directe [2]

1.3.2.2 Les attaques indirectes par rebond

Dans ce type d'attaques, l'intrus attaque la machine cible par une machine intermédiare.

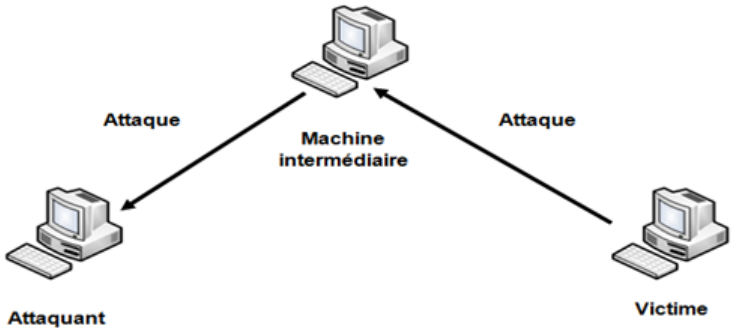

FIGURE 1.3 – Attaque indirecte par rebond [2]

L'avantage du rebond est de masquer l'identité de l'intrus et d'utiliser les ressources de la machine intermédiare car elle est puissante (CPU,bande passante,...) pour réaliser son attaque.

1.3.2.3 Les attaques indirectes par réponse

Ces attaques sont des variantes des attaques par rebond. Elles offrent les mêmes avantages, du point de vue de l'intrus. Mais au lieu qu'il envoie une attaque à la machine, il lui envoie une requête et la réponse à cette requête sera envoyée à la victime.

1.3.3 Exemples d'attaques informatiques

1.3.3.1 Les attaques sur les mots de passe

L'obtention du mot de passe d'un utilisateur permet à un attaquant d'accéder à un système ou à un réseau avec les mêmes droits d'accès que cet utlisateur. Cela peut être trés grave si l'utilisateur en question a un accès privilégié du type root ou

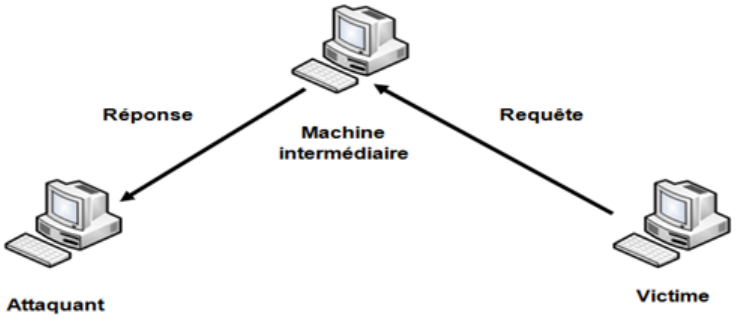

FIGURE 1.4 – Attaque indirecte par reponse [2]

administrateur de domaine. Pour obtenir un mot de passe, on a souvent recours au craquage de celui-ci.[**2**]

Les attaques sur les mots de passe consistent à calculer des empreintes, c'est-à-dire les valeurs de sortie d'une fonction de hachage, et à les comparer à celles contenues dans les fichiers de mots de passe. On peut distinguer :[**4**]

-Les craquages par force brute : "attaque bourrin", consiste à cracker un mot de passe en testant exhaustivement toutes les combinaisons possibles.

Connaissant la taille maximale d'un mot de passe et l'espace des caractères qui peuvent le composer, il s'agit de calculer l'empreinte cryptographique de toutes les combinaisons possibles et de les comparer à celles enregistrées dans le système de contrôle des mots de passe.

-Les craquages par dictionnaire : consiste à tester une série de mots issus d'un dictionnaire.

Il existe toutes sortes de dictionnaires disponibles sur Internet pouvant être utilisés pour cette attaque (dictionnaire des prénoms, dictionnaire des noms d'auteurs, dictionnaire des marques commerciales,etc).

-Les craquages par compromis temps/mémoire : Il est possible d'utiliser des dictionnaires pré-calculés contenant une liste de mots de passe et leur empreinte associée. Même si cette possibilité accélère le temps nécessaire pour retrouver un mot de passe, elle nécessite une place plus importante en mémoire.

Cependant sur les fonctions de hachage faibles (par exemple la fonction de hachage LM sur les systèmes Microsoft Windows), il est possible d'utiliser des compromis temps/mémoire pour calculer une fois pour toutes les dictionnaires pré-calculés et rendre ensuite rapide l'attaque proprement dite.

1.3.3.2 Le déni de service (DOS, Denial of Service)

Dans cette partie d'attaques, on peut citer :

a- Une technique de déni de service :"Le smurf"

La technique du "smurf" est basée sur l'utilisation de serveurs broadcast pour paralyser un réseau. Un serveur broadcast est un serveur capable de dupliquer un message et de l'envoyer à toutes les machines présentes sur le même réseau [2].

Le scénario d'une attaque de ce type est le suivant :

1. La machine attaquante envoie un ping (le ping est un outil exploitant le protocole ICMP, permettant de tester les connexions sur un réseau en envoyant un paquet et en attendant la réponse) à un ou plusieurs serveurs broadcast en falsifiant l'adresse IP source (adresse à laquelle le serveur devrait théoriquement répondre) et en fournissant l'adresse IP d'une machine cible.
2. Le serveur broadcast répercute la requête sur l'ensemble du réseau.
3. Toutes les machines du réseau envoient une réponse au serveur broadcast.

4. Le serveur broadcast redirige les réponses vers la machine cible.

Ainsi, lorsque la machine attaquante adresse une requête à plusieurs serveurs broadcast situés sur des réseaux différents, l'ensemble des réponses de tous les ordinateurs des différents réseaux vont être routées sur la machine cible.

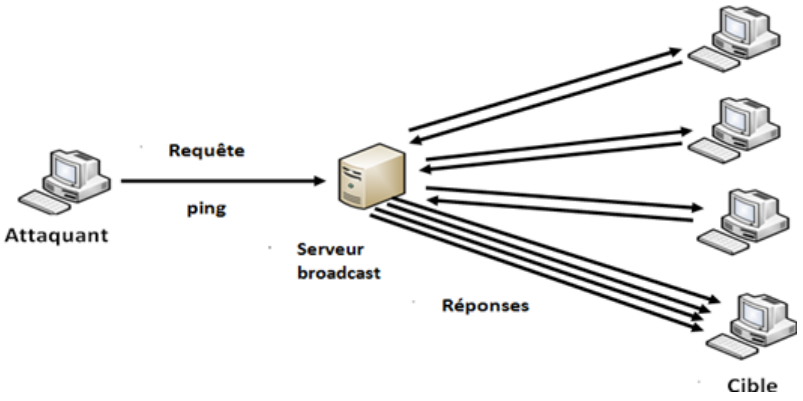

FIGURE 1.5 – Attaque smurf

De cette façon, l'éssentiel du travail de l'attaquant consiste à trouver une liste de tous les serveurs broadcast et d'arriver à falsifier l'adresse de réponse afin de les diriger vers la machine cible.

b- Le TCP-SYN/Flooding

Quand un système client essaie d'établir une connexion TCP à un système fournissant un service (le serveur), le client et le serveur échangent une séquence de messages suivant ce schéma [2].

Les abus viennent au moment où le serveur a envoyé un accusé de réception du SYN (ACK-SYN) au client mais n'a pas reçu le "ACK" du client. C'est alors une connexion à demi-ouverte. Le serveur construit dans sa mémoire système une structure de données décrivant toutes les connexions courantes. Cette structure de don-

nées est de taille finie, ce qui veut dire qu'il peut se créer un dépassement de capacité en créant intentionnellement trop de connexions partiellement ouvertes.

L'ordinateur de l'agresseur envoie des messages SYN à la machine victime ; ceux-ci paraissent provenir d'un ordinateur bien défini mais qui en fait, fait référence à un système client qui n'est pas capable de répondre au message SYN-ACK. Ce qui veut dire que le message ACK de confirmation finale ne sera jamais renvoyé au serveur victime.

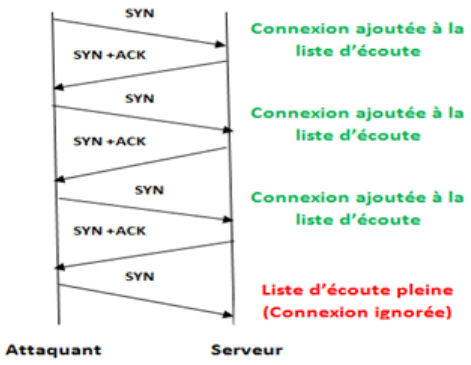

FIGURE 1.6 – Attaque SYN-Flooding [2]

1.3.3.3 L'attaque Man In The Middle

L'attaque man-in-the-middle (Homme au milieu) est une technique de piratage informatique consistant à intercepter des échanges cryptés entre deux personnes ou deux ordinateurs A et B pour décoder les messages. L'attaquant doit donc être capable de recevoir les messages des deux parties et d'envoyer des réponses à une partie en se faisant passer pour l'autre.[**5**]

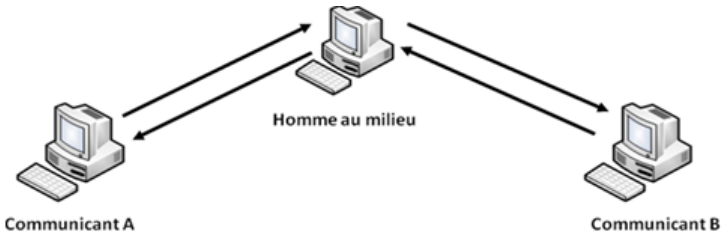

FIGURE 1.7 – Attaque Man In The Middle [5]

1.3.4 Les logiciels malveillants

Parmi les multiples procédés d'attaque contre le système d'information, il existe une famille de logiciels malveillants qui se répandent en général par le réseau, soit par accès direct à l'ordinateur attaqué, soit cachés dans un courriel ou sur un site Web attrayant, mais aussi éventuellement par l'intermédiaire d'une disquette, d'une clé USB ou d'un CD-Rom. Parmi ces logiciels, on peut cité :

1.3.4.1 Virus

Un virus informatique est un programme, généralement de petite ou très petite taille, doté des propriétés suivantes : infection , multiplication , possession d'une fonction nocive (payload).

La fonction d'infection permet au virus de s'introduire dans des fichiers de programme, dans des fichiers de données utilisant un langage de script, ou dans une partie de la disquette ou du disque dur contenant un petit programme (secteur de démarrage). Lors de l'accès à ces programmes ou secteur, le code du virus s'exécutera de façon d'abord silencieuse (phase de multiplication pendant laquelle il infectera d'autres fichiers) puis visible (activation de la fonction nocive).

La fonction nocive pourra être déclenchée par des facteurs très variables selon le virus (au bout de n réplications, à une date fixe, lors de l'exécution de certaines

tâches précises...). Elle peut se limiter à l'affichage d'un message agaçant ou, plus généralement, conduire à des perturbations graves de l'ordinateur (ralentissement du fonctionnement, effacement ou corruption de fichiers, formatage du disque dur...). Les virus sont donc des programmes parasites qui doivent être hébergés dans d'autres fichiers (ou secteur exécutable du disque).

On emploie souvent de façon impropre le mot virus pour désigner d'autres programmes nocifs (malwares), en particulier les vers [**5**].

1.3.4.2 Ver

Un ver (worm) est une variété de virus qui se propage par le réseau (en utilisant ses mécanismes) sans avoir besoin d'un support physique ou logique (disque dur, programme hôte, fichier,...).

1.3.4.3 Cheval de Troie

Un cheval de Troie (Trojan Horse en anglais) est un logiciel d'apparence légitime, conçu pour exécuter des actions à l'insu de l'utilisateur. En général, il utilise les droits appartenant à son environnement pour détourner, diffuser ou détruire des informations, ou encore pour ouvrir une porte dérobée (fonctionnalité inconnue de l'utilisateur légitime, qui donne un accès secret au logiciel qui permet à un pirate informatique de prendre, à distance, le contrôle de l'ordinateur). Les trojans sont programmés pour être installés de manière invisible, notamment pour corrompre l'ordinateur hôte [**6**].

1.3.4.4 Porte dérobée

Une porte dérobée (backdoor) est un logiciel de communication caché, installé par exemple par un virus ou par un cheval de Troie, qui donne à un agresseur extérieur accès à l'ordinateur victime, par le réseau [**7**].

1.3.4.1 Logiciel espion

Un logiciel espion, comme son nom l'indique, collecte à l'insu de l'utilisateur légitime des informations au sein du système où il est installé, et les communique à un agent extérieur, par exemple au moyen d'une porte dérobée [**7**].

1.4 Mécanismes et outils de protection

Les mécanismes et les outils de détection d'intrusion permettent d'assurer la sécurité des systèmes informatiques et les rendre plus efficaces, parmi ces mécanismes et outils nous citons :

1.4.1 Les antivirus

Un antivirus est un programme capable de détecter la présence de virus sur un ordinateur et, dans la mesure du possible de désinfecter ce dernier. On parle ainsi d'éradication de virus pour désigner la procédure du nettoyage de l'ordinateur. Les antivirus s'appuient sur la signature virale propre à chaque virus pour le détecter. Il sagit de la méthode de recherche de signature (scanning), la plus ancienne méthode utilisée par l'antivirus.

Un antivirus utilise plusieurs méthodes pour l'éradication des virus, nous avons :[**6**]

- La suppression du code correspondant au virus dans le fichier infecté.
- La suppression totale du fichier infecté.
- La mise en quarantaine du fichier infecté, c'est-à-dire le déplacer dans un emplacement où il ne pourra pas être exécuté.

subsectionPare-feu (firewall)

Un pare-feu est système permettant de protéger un ordinateur, ou un réseau d'ordinateurs des intrusions provenant d'un réseau tiers(notamment Internet). Le pare-feu est un système permettant de filtrer les paquets de données échangés avec le réseau, il sagit ainsi d'une passerelle filtrante compotant au minimum les interfaces réseau suivante :

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

Un système pare-feu contient un ensemble de règles prédéfinies permettant :

- D'autoriser la connexion (allow) ;
- De bloquer la connexion (deny) ;
- De rejeter la demande de connexion sans avertir l'émetteur (drop).

L'ensemble de ces règles permet de mettre en œuvre une méthode de filtrage dépendant de la politique de sécurité adoptée par l'entité. On distingue habituellement deux types de politique de sécurité permettant :

- Soit d'autoriser uniquement les communications ayant été explicitement autorisée :« tout ce qui n'est pas autorisé est interdits ».
- Soit d'empêcher les échanges qui ont été explicitement interdits.

1.4.2 Les systèmes de détection d'intrusion-IDS

Un système de détection d'intrusions (IDS :Intusion Detection System) est un système qui surveille le trafic réseau ou les journaux d'audit « logs »sur les machines hôtes pour déterminer s'il y a eu une violation d'une certaine politique de sécurité

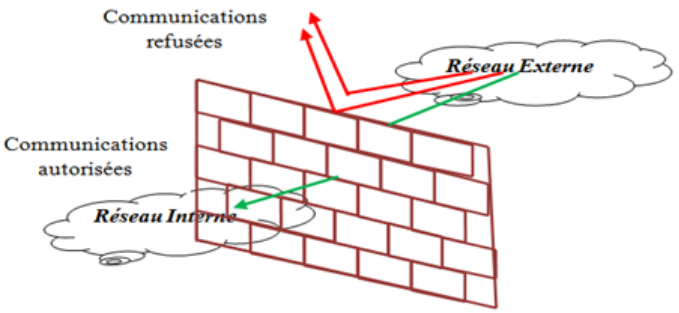

FIGURE 1.8 – Pare-feu

dune organisation. Un IDS peut détecter des intrusions qui ont circulées ou ont passées à travers un pare-feu, ou celles qui se passent à l'intérieur du réseau local « LAN » derrière le pare-feu.

1.4.3.1 Les types des IDS

Les systèmes de détection d'intrusions qui opérants sur une machine hôte spécifique et détecte les activités malicieuses au sien de cette même machine sont dit « H-IDS : Host-Besed IDS ». Les systèmes de détection d'intrusion qui opèrent sur des segments de réseau et analyse le trafic de ces même segments sont dits « N-IDS : Network-Besed IDS ».

Les techniques de détection d'intrusion se répartissent en deux grandes classes : détection d'anomalies, aussi appelée approche comportementale, et détection d'attaques dites également approche par scénarios.

a- Approche comportementale

L'approche comportemental(«anomaly detection» en englais) consiste à comparer les comportements observés à une référence de comportement normale. Toute déviation entre les deux comportements déclenche une alerte. Différentes méthodes ont été proposées pour définir ce qui est normal (des outils statistiques, des systèmes

experts, etc.)

b- Approche par signature (par scénarios)

L'approche par ségnature est basée sur la comparaison du comportement observé avec une référence correspondante à des scénarios d'attaque est reconnu comme intrusif, le reste est considéré comme normale. Différents mécanismes ont été utilisés :l'analyse de signatures(pattern matching), les algorithmes génétique,etc.

1.5 Conclusion

Dans ce premier chapitre, nous avons donné les principales connaissances dans le domaine de la sécurité informatique, en précisant les différentes étapes d'une attaque ainsi que les différentes techniques utilisées par les attaquants et les outils et mécanismes de protection pouvant être employés contre ces attaques. Ces informations sont très nécessaires à connaître pour la génération de scénarios d'attaque. En effet, c'est ces informations qu'un intrus peut exploiter pour réaliser son attaque. La génération des scénarios d'attaques est le domaine de recherche de plusieurs travaux.

2

Etat de l'art

2.1 Introduction

Le graphe d'attaque joue un rôle important dans la sécurisation d'un réseau informatique. En effet, il indique l'état de sécurité global du réseau après chaque action d'un attaquant, ce qui aide les administrateurs du réseau à prendre les mesures de sécurité appropriées. Pour cela, la génération des scénarios d'attaques est le domaine de recherche de plusieurs travaux.

2.2 Langages de description de scénarios d'attaque

Pour les spécifications des scénarios d'attaque, Cuppens et Ortalo [8] définissent un langage, appelé Lambda. Dans l'approche de Cuppens [8], le travail est confiné

à la description, Lambda, des attaques avec leurs pré et post-conditions. Les scénarios possibles sont établis automatiquement, à partir d'une description d'une base d'attaques, par un mappage des prés et des post-conditions.

Mé et Michel [10] proposent un langage de description des scénarios d'attaques appelé ADeLe. Une attaque en ADeLe est décrite avec ses pré et post-conditions. Les auteurs d'ADeLe n'ont pas poussé les détails jusqu'au niveau atteint dans le langage Lambda où chaque information utile pour l'attaque est codifiée sous une forme similaire aux faits de Prolog [11].

La description d'un scénario d'attaque nécessite un langage multi-événements pour la reconnaissance de signature. Dans [12], Templeton et levitt proposent un langage pour la description des attaques appelé JIGSAW. Ce langage est basé sur l'expression des pré-conditions d'attaque et de leurs effets en ce qui concerne l'attaquant. Le langage JIGSAW est destiné pour la description des attaques élémentaires et des scénarios d'attaque non complexes.

Les langages cités ci-dessus impliquent que nous connaissons, à priori, tous les scénarios que nous voulons détecter, mais sans aucune garantie que nous serions capables de générer ces scénarios [11].

2.3 Approches formelles de génération de scénarios d'attaque

Les auteurs [13, 14,15] proposent une approche de génération automatique de graphes d'attaques basée sur le Model Checking. Cependant, ces graphes doivent faire face aux problèmes d'explosion combinatoire lorsque le réseau comporte un nombre important d'ordinateurs qui interagissent entre eux.

Pour analyser les exploits dans un grand réseau, Amman et Al [16] proposent une méthode évolutive d'analyse des vulnérabilités des réseaux basée sur la supposition de la monotonicité (l'attaquant ne perd jamais sa capacité de favoriser une attaque), ce qui réduit la complexité d'analyse de l'exponentielle au polynomiale. Jajodia et Al[17] ont utilisé ces approches pour développer un outil pour l'analyse topologique des vulnérabilités (TVA :Topological Vulnerability Analysis), TVA est l'un des outils les plus compréhensibles développés jusqu'à présent pour la construction et l'analyse des graphes d'attaques, les auteurs[12,18,19] ont étendu cette approche par des détails complémentaires qui expliquent comment les recommandations d'endurcissement du réseau sont faites dans le système TVA et ils décrivent les différentes approches pour réduire les parties de la génération des graphes d'attaques au système TVA. L'outil TVA génère un graphe de dépendances entre les exploits qui représentent tous les chemins d'attaques possibles, mais ils ne spécifient pas explicitement la politique de sécurité à respecter et il ne peut pas fournir des informations, comme par exemple : si une séquence potentielle d'attaques est vraiment nocive [20,21,22].

Dans [23], Dacier introduit le concept de graphe de privilèges, cette approche analyse seulement un seul hôte et utilise des outils de construction de graphes et des mesures qui ne peuvent pas être utilisées pour un grand réseau.

Dans [24], un modèle de génération de scénarios d'attaque est proposé (les arbres d'attaque).Les arbres d'attaque peuvent intercepter les étapes d'une attaque et de leurs interdépendances [24].

Les principaux éléments constitutifs des arbres sont des nœuds. Dans les travaux [15] et [24] les nœuds sont utilisés pour les étapes du modèle d'attaque ou bien pour

les actions d'un attaquant.

Chaque arbre a un seul nœud supérieur qui représente la réalisation de l'objectif final d'attaque. Le modèle développé dans [25] est également basé sur le modèle de l'arbre d'attaque. Son objectif principal est de fournir un moyen de documenter les intrusions, chaque chemin à travers un arbre d'attaque représente une attaque unique. Par conséquent, il tend à être une approche descriptive pure.

Un autre modèle (Demandes/offres) a été proposé dans [12].Ce modèle a été dérivé pour décrire des scénarios complexes et de les généraliser à des attaques inconnues plutôt que de considérer les attaques simples. Au lieu de décrire les attaques par une séquence spécifique des actions qu'un attaquant emploie pour atteindre un but spécifique, cette approche décrit les attaques en utilisant des composants abstraits. Les auteurs considèrent les attaques comme un ensemble de capacités qui offrent un soutien pour des concepts abstraits qui attaquent à leur tour de nouvelles capacités pour soutenir d'autres concepts. L'avantage de ce modèle est qu'il ne nécessite aucune connaissance préalable d'un scénario particulier et donc de nombreuses attaques inconnues peuvent être décrites de manière implicite. Mais, ce modèle n'a pas été appliqué à un grand réseau [11].

Le modèle (Réseaux de pétri d'Attaque) introduit d'abord dans [26] est principalement utilisé pour la modélisation du processus de tests de pénétration. Dans ce modèle, les étapes d'attaque sont représentées par endroits similaires à des nœuds dans les arbres d'attaque. Les transitions sont utilisées pour la modélisation explicite des actions attaquant qui s'étend sensiblement l'expressivité de ce modèle par rapport aux arbres d'attaque. Les réseaux de Pétri ne sont pas limités à une structure arborescente, ce qui permet la réutilisation de sous-graphe dans des contextes différents. Il fournit également un niveau suffisant de détails pour la simulation et

l'exécution d'attaques. Mais, ce modèle peut être compliqué pour un grand réseau.

Dans [20,21,22] les auteurs HAMZA et ADI ont proposé une nouvelle approche formelle basée sur la notion de privilèges pour produire un graphe d'attaque contenant des scénarios d'attaque du système étudié. Le graphe représente les privilèges acquis par l'exploitation des vulnérabilités du système. Les arêtes du graphe indiquent les relations de causalité entre les privilèges. Leur contribution consiste à représenter formellement ces relations de dépendance et de générer un graphe d'attaque à travers un système de déduction. De plus, leur approche spécifie explicitement la politique de sécurité violée. Celui-ci est codé par une formule logique temporelle. La méthodologie adoptée pour résoudre leur problème est expliqué dans les trois étapes suivantes : - Le réseau est représenté par un automate fini. - La politique de sécurité est définie par une formule logique. -Les scénarios d'attaque sont générés à partir d'un ensemble de règles représentant la stratégie de l'intrus.

2.3.1 Systèmes de détection d'intrusions et corrélation d'alertes

Afin de détecter des attaques complexes et augmenter le niveau de la granularité des alertes, les auteurs de [27] proposent une technique de corrélation d'alertes basée sur les graphes afin de raisonner au sujet des attaques probablement manquées par les IDS (Intusion Detection System). Ceci est accompli par la construction du graphe d'attaques d'un réseau protégé par l'IDS. Si les alertes d'IDS peuvent être associées aux actions du graphe d'attaques, alors les alertes qui arrivent dans un ordre prévu, en procédant le long d'un chemin simple du graphe d'attaques, peuvent indiquer qu'un attaquant exécute avec succès les étapes de ce chemin. Cette approche est l'une des méthodes de corrélation d'alertes qui dépend des alertes fournies par les IDS.

Les auteurs de [28] ont proposé une approche pour la visualisation, la corrélation

et la prédiction des graphes d'attaques complexes à travers les réseaux. Cette approche considère les vulnérabilités logicielles sur tous les hôtes du réseau, la connectivité, les effets du pare feu et les exploits potentiels de l'attaquant. Cette approche augmente les représentations traditionnelles d'un graphe avec les matrices d'adjacences, elle permet de générer des graphes d'attaques complexes tout en incluant tous les chemins d'attaques connus.

L'objectif des auteurs M.Saber, T.Bouchentouf et A. Benazzi [25] est d'améliorer les systèmes de détection d'intrusion (IDS). Cette amélioration est basée sur trois secteurs de recherche : classification des attaques, génération de scénarios d'attaque est finalement les méthodes d'évaluation.

Les auteurs se focalisent sur la génération des scénarios d'attaques afin de réduire au minimum le nombre de fausses alertes déclenchées par les IDS. Ils ont adopté le modèle de processus d'attaque de Gad et d'Al [29] qui est basé initialement sur l'analyse des attaques de malware comme les virus les plus répondus et les vers.

Ils ont ainsi proposé deux algorithmes de génération de scénarios d'attaques. Le premier algorithme permet la conversion du problème en un problème de programmation de contraintes (CSP en anglais :constraint programming problem). Le second est basé sur la recherche du plus court chemin.

Le CSP « Problème de Satisfaction de Contraintes » est définit comme étant un ensemble de contraintes C devront être satisfaites par l'ensemble des variables V, chacune de ces variables prend ses valeurs dans un domaine D (initial, final, ou initial et final).

Les algorithmes de génération des scénarios des scénarios d'attaque peuvent être

implémentés par plusieurs langages tels que Mozart, Jacop, ILOG ou en utilisant la bibliothèque de java Choco. C'est cette dernière qui est utilisée pour l'implémentation de l'algorithme CSP.

Le MASP « Modified Algorithm Shortest Path » produit en sortie une liste des scénarios valides à partir d'un nœud de départ (Reconnaissance, Gain d'accès et Dénis de service). Le scénario d'attaque généré est le plus court, et cela est assuré par utilisation d'une liste contenant les nœuds visités à partir du nœud de départ pour éviter la boucle infinie.

Après la mise en application des deux algorithmes CSP et MASP en développant un détecteur automatique d'intrusions. Le CSP a été implémenté en utilisant la bibliothèque de CHOCO et le langage java et l'algorithme MASP en java.

L'analyse des résultats obtenue par les deux algorithmes montre que le MASP est le meilleur en termes de temps d'exécution, cependant il exige plus de ressources car les données sont stockées dans la RAM, alors que l'algorithme CSP fait toutes les combinaisons selon la taille du scénario d'attaque désiré, et il ne génère donc que les scénarios valides satisfaisant cette contrainte.

2.4 Algorithmes de génération de scénarios d'attaque

Tito Waluyo et Kuspriyanto [30] ont décrit plusieurs algorithmes pouvant être utilisé pour générer les graphes d'attaques.

Le premier algorithme est proposé par Zhong et al [31] qui adoptent un algorithme positif et en largeur d'abord. Cet algorithme se base sur quelques principes :

- Sauvegarder le nœud représentant l'hôte d'un attaquant dans une liste appelée Node-queue initialisé à vide. Tant que cette dernière n'est pas vide, alors pointer sur l'hôte directement relié à l'attaquant considérée comme attaquant d'accueil. A partir de cette hôte, on sauvegarde dans Node-queue toutes les hôtes- non enregistrées auparavant- auxquelles elle est reliée (ces hôtes peuvent être attaquées par un pirate).
- L'algorithme se termine lorsque Node-queue devient vide.

Le second algorithme de Zhong et al [26] divise les machines à l'intérieur d'un réseau en trois types d'hôtes, le noeud cible qui est l'objectif de l'attaquant, les noeuds intermédiaires qui sont les noeuds du réseau exploités par l'attaquant afin d'y arriver à la cible, et les noeuds de marionnette qui sont les noeuds du réseau externe qui ont le privilège d'accéder au réseau interne. Les noeuds que nous devons protéger sont considérés comme étant la cible. Cet algorithme se base sur la notion de récursivité, ce qui facilite son implémentation. Il définit deux fonctions récursives :

- attack_condition : vérifie si les conditions de la règle à appliquer sur la vulnérabilité sont satisfaites par le noeud courant. La condition d'arrêt est lorsqu'on retourne au noeud racine. Cette fonction a comme paramètre d'entrée la précondition de la règle.
- attack_chain : construit les chaines possibles d'attaques à partir de la racine. Cette fonction prend comme paramètre d'entrée l'hôte courante.

Le graphe général d'attaque est créé en combinant l'ensemble des chaines d'attaque.

Le troisième algorithme est proposé par Bhattacharya et al [32]. L'approche pro-

posé aborde le problème de scalabilité de la génération de graphe d'attaque par un algorithme générique de détection de chemin d'attaque, ce qui permet de réduire le problème de la redondance dans le graphe d'attaque par élimination des cycles. Cet algorithme utilise deux structures de données de base les piles et les files.

Les algorithmes cités ci-dessus ne sont pas généralement appropriés au grand réseau (problème de scalabilité), pour leur complexité élevée de temps, et la consommation élevée de l'espace.

2.5 Génération de scénarios d'attaque pour les réseaux sans fil

Plusieurs techniques de recherche numérique théorique sont présentées dans la littérature mais sont peu adapté aux attaques des réseaux sans fil, et spécialement aux réseaux ad-hoc.

En effet, ces réseaux sont particulièrement caractérisés par la large fonte (ce genre de réseaux peut rassembler un nombre important de machines), la nature incertaine des liens et l'absence de l'infrastructure, ce qui les force à montrer de nouvelles vulnérabilités aux attaques de sécurité en plus de ceux qui menacent les réseaux filaires et rendent plus dur leur employement de l'ensemble des techniques d'analyse des scénarios d'attaques proposées dans la littérature.

S. Rekhis & N. Boudriga [33] proposent une approche formelle pour la recherche numérique des attaques dans les réseaux sans fil. Ils présentent un modèle décrivant les scénarios d'attaques dans un environnement sans fil et l'état du réseau en conséquence.

L'intérêt de ces approches formelles est d'éviter la génération des résultats empêchant l'exactitude de l'analyse et l'intégrité de la recherche. La recherche numérique est définie comme étant l'utilisation des méthodes scientifiques dérivées et démontrées vers la préservation, la collection, la validation, l'identification, l'analyse, l'interprétation et la présentation de l'évidence numérique dérivée des sources numériques. Les objectifs de cette recherche numérique sont :

- La reconstruction d'un scénario d'attaque potentiel.
- L'identification du location (s) duquel l'attaquant a exécuté à distance une partie d'actions du scénario.
- L'interprétation de ce qui s'est produit pour empêcher les futurs incidents semblables.
- Augmentation des résultats sans réfutables preuves (des résultats certains) concernant les scénarios d'attaques reconstruits.

Les deux auteurs ont développé un système d'inférence qui intègre deux types d'évidence, la manipulation de l'imperfection et la duplication de l'information. Pour illustrer leur proposition, ils ont considéré une étude traitant la recherche d'attaques à distance de débordement d'amortisseur.

2.6 Analyse de graphe d'attaques

L'objectif de Nilima R. Patil [34] est d'étudier les différentes manières d'analyser le graphe d'attaque. Ils ont étudié plusieurs articles de recherche sur la génération et l'analyse de graphe d'attaque, ils ont inclus quelques articles basés sur l'analyse de graphe d'attaque. La visualisation est une méthode primaire employée pour analyser le graphe d'attaque. Laura Swiler et Al [35] représente un outil pour l'évaluation des attributs et des vulnérabilités de sécurité dans des réseaux informatiques. Le

graphe d'attaque exige trois entrées : type d'attaque, dossier de configuration, et un profil d'attaquant. Tel que le type d'attaque contient les attributs nécessaires et l'état acquis, ainsi le dossier de configuration contient des informations architecturales initiales sur le système spécifique à analyser, et le profil d'attaquant contient les informations de l'attaquant qui sont utilisées pour assortir des types d'attaque.

Dans [24], l'objectif de M. Albanese, S. Jajodia, et S. Noel est de remédier aux problèmes dont souffrent les approches précédentes de durcissement de réseau. En effet, elles recherchent les solutions exactes et non mesurables. De plus, les éléments durcissant ont été traité indépendamment, ce qui est inadéquat pour les environnements réels. Les auteurs ont examiné d'abord plus en détail les limites de l'approche proposée dans [36], puis ils ont présenté leur algorithme d'approximation, et ils ont montré que ses résultats sont raisonnables et bons en termes de temps du calcul et donnent un coût optimal. L'algorithme BackwardSearch qui a été réécrit par les auteurs M. Albanese, S. Jajodia, et S. Noel [24] est fonctionnellement équivalent à celui décrit dans [36].

Cet algorithme a plusieurs avantages par rapport à sa version originale. Tout d'abord, il est plus général, car il ne suppose pas que les conditions initiales peuvent être désactivées individuellement. Et intègre les notions d'action valable et la stratégie de durcissement. Cette stratégie définie comme étant un ensemble d'actions qui permet de séparer tous les chemins d'attaque. Ils ont aussi défini un modèle de coût, ce qui permet de préciser l'analyse des actions de durcissement disponible.

L'algorithme proposé [24] calcule directement un ensemble de stratégies de durcissement possibles, plutôt que d'une logique de proposition qui nécessite un traitement supplémentaire afin de fournir des renseignements utilisables. Enfin, l'algorithme se termine dès qu'une solution soit trouvée.

Cette approche a les mêmes limites. Il n'est pas toujours possible de retirer les exploits arbitraires (ou chemins d'attaque) sans enlever leurs causes.

Dans [37], S. Roschke, F. Cheng, R. Schuppenies Ch. Meinel ont donné une extraction automatique de l'information significative de diverses bases de vulnérabilités existantes. Après avoir comparé les bases de données de vulnérabilités existantes, une nouvelle méthode est proposée pour une extraction automatique d'information de vulnérabilité pour des descriptions textuelles. Ils ont après implémenté un prototype (met en application) pour prouver l'applicabilité de la méthode proposée pour la construction de graphe d'attaque.

2.7 Conclusion

Dans ce chapitre, nous avons étudié quelques travaux existant traitant le problème de la génération de scénarios d'attaque dans les systèmes informatiques en citant quelques solutions proposées à savoir les modèles formels, les algorithmes, les langages de description de scénarios d'attaque proposés dans la littérature ainsi que le problème de la génération de scénarios d'attaque dans les réseaux sans fil.

3

Démarche étudiée

3.1 Introduction

L'approche étudiée [20,21,22] se base sur le graphe de privilège et une politique de sécurité pour générer le graphe d'attaques. Elle construit les scénarios d'attaques en utilisant une preuve formelle. Le choix de la preuve formelle est justifié par le fait qu'elle a la capacité de traiter des systèmes avec un nombre infini d'états et aussi elle est capable d'en déduire des conclusions directement à partir d'une description abstraite du système et d'une politique de sécurité[11]. La figure ci-dessous illustre les différentes étapes de cette approche : Dans cette approche, un noeud dans le graphe d'attaque représente un privilège acquis par l'exploitation d'une vulnérabilité et les arcs représentent la relation de dépendances existantes entre ces différents privilèges. Ainsi, une attaque est définit comme étant l'acquisition d'un privilège en

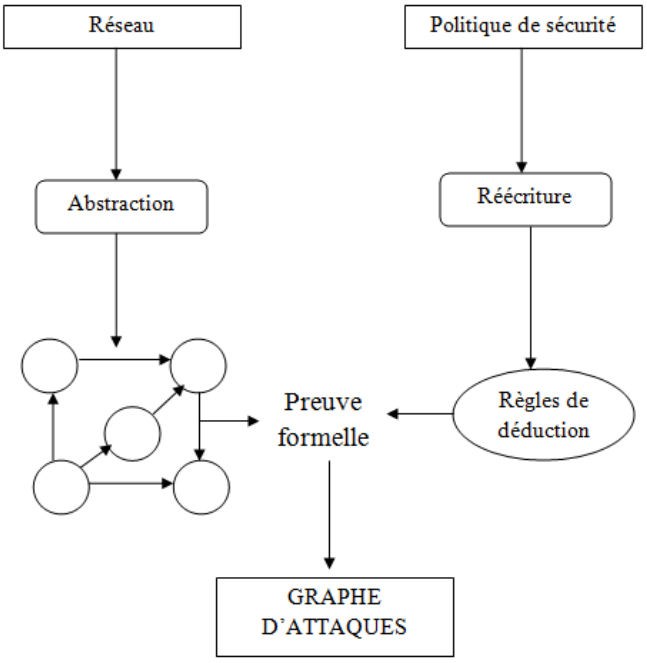

FIGURE 3.1 – Démarche étudiée [20,21,22]

exploitant une vulnérabilité et un scénario d'attaque est l'acquisition d'un ensemble de privilèges dont le dernier est l'objectif de l'intrus et un scénario d'attaque ne se termine que si cet objectif est atteint.

Les étapes principales de cette approche sont :

1. La définition du modèle d'attaque.
2. La spécification des règles de déduction.
3. La construction du graphe d'attaques.

3.2 Le modèle d'attaque

Le modèle d'attaque est représenté dans [20,21,22] par une machine à états finis. Le choix de ce modèle est justifié par le fait qu'il inclut toutes les familles de modèles de politique de sécurité, chacun de ces différents modèles est caractérisé par la manière de définir un état, la fonction de transition ou l'ensemble des états qui

violent la sécurité. Ce modèle décrit le système par les composants suivants :

-H : L'ensemble des ordinateurs reliés au réseau ;

-C : La relation de connectivité ;

-A : L'ensemble des différentes actions qu'un intrus peut appliquer pour effectuer un scénario d'attaques ;

-O : L'ensemble des objectifs d'intrusion.

- **Hôtes (H) :** L'intérêt principal de modéliser des hôtes est de capturer le maximum d'information sur ces composants vulnérables. Selon Sheyner [38], un ordinateur $h \in H$ est décrit par le tuple (id, svcs, sw, vuls) où :

  – id : est l'identifiant simple de l'ordinateur (nom de machine, adresses de réseau) ;
  – svcs : est la liste des services (le nom du service, le port) ;
  – sw : est la liste des logiciels actifs (en fonction) ;
  – vuls : est la liste des composants vulnérables.

- **Relation de connectivité (C) :** La connectivité est exprimée par la relation $C \subseteq H \times H \times P$ où P est l'ensemble de numéros de port. La relation $C(h_1 , h_2 , p)$ signifie que l'ordinateur h_2 est accessible pour l'ordinateur h_1 sur le port . Il faut noter que la relation de connectivité comprend les règles de filtrage et du firewall qui limitent l'activité d'un ordinateur sur un autre.

- **Actions de l'intrus :**Les actions de l'intrus sont basées sur l'exploitation des vulnérabilités existantes d'un élément de l'ensemble d'hôtes H. Ainsi, il est nécessaire de modéliser les attaques associées à chaque vulnérabilité.

3.3 La politique de sécurité du réseau :

La gestion de la sécurité des réseaux est très difficile, et ça est dû à leur hétérogénéité et leur complexité. L'abstraction de ces contraintes nécessite la spécification de la politique de sécurité en termes de niveau élevé. Considérons action l'ensemble des évènements qui peuvent être autorisés ou interdits par une politique de sécurité. Dans cette approche [2,20,21,22], ces événements représentent l'accès d'un utilisateur malveillant à une ressource et une politique de sécurité est définie comme suit :

- Une politique de sécurité définie sur un ensemble de séquence d'action est la propriété indiquant pour une séquence d'actions si elle est autorisé ou pas.
- Un système informatique est dit qu'il est conforme avec une politique de sécurité si aucune action (ou séquence d'actions) ne peut violer cette politique.

L'intrus cherche à violer la politique de sécurité et sa stratégie consiste à exploiter les vulnérabilités du système afin d'acquérir de plus en plus de privilèges tant que son objectif d'intrusion n'est pas encore atteint. Dans la génération de scénarios d'attaques suivant cette approche, l'objectif de l'intrus (la motivation qui justifie ses actions) est considéré comme étant l'objectif de l'intrusion et non l'objectif détecteur qui s'agit de toute violation de la politique de sécurité.

3.3.1 Spécification de la politique de sécurité

La génération de scénarios d'attaques pour un système informatique nécessite la définition de la politique de sécurité à appliquer sur ce système dans un langage formel. L'approche [20,21,22] présente quelques définitions importantes :

- Scénario d'attaques : un scénario $\tau =< \ priv_0, \ priv_1, ..., \ priv_n >$ est une

séquence de privilèges où le $i^{ème}$ privilège est noté par τ_i, le préfixe de la trace τ jusqu'à la $i^{ème}$ action est noté par $\tau_i^{<}$ et le suffixe de la trace partant de l'évènement i est noté $\tau_i^{>=}$. Soit τ une trace, on dénote par τ^+ l'ensemble des actions qui composent la trace. Formellement :

a.$\tau^+ = \{a\} \bigcup \tau^+$ et $\epsilon^+ = \Phi$

- Attaque : un évènement dans un scénario est une exploitation d'une vulnérabilité définie comme étant l'acquisition d'un privilège.
- Objectif d'intrusion : est une action interdite définie par la politique de sécurité.

Pour construire des scénarios d'attaques complexes, les auteurs de l'article définissent une logique simple pour spécifier la politique de sécurité. La syntaxe de cette logique est donnée dans la table suivante :

***Syntaxe* :**

$$\Phi, \Psi ::= p \mid \neg \Phi \mid \Phi \vee \Psi \mid \Phi . \Psi \mid \Phi \cup \Psi$$

Syntaxe Dans la table précédente, p indique une proposition atomique employée pour vérifier l'exploitation d'une vulnérabilité, nous dénotons cette proposition atomique par $v_x(h)$. Ceci indique une acquisition du privilège sur l'hôte par l'exploitation de la vulnérabilité Ce privilège a pu inclure la découverte d'information de réseau, acquisition des relations de confidence avec d'autres hôtes, aussi bien que d'autres effets. Avant d'exploiter la vulnérabilité , certaines conditions doivent être rencontrées, nous dénotent ces conditions par l'ensemble *cond*(). Ces conditions peuvent inclure l'existence de la vulnérabilité sur l'hôte et de la connectivité entre deux hôtes. La formule logique est établie en employant un ensemble de proposi-

tions atomiques, d'opérateurs logiques $\neg$,$\bigcup$, $\bigvee$ et l'opérateur de concaténation de séquence$\ll \cdot \gg$.

La sémantique de la logique proposée est donnée dans la table suivante :

Sémantique **:**

$\tau \models priv(v_x(h))$	Si	$\tau_0 = priv(v_x(h))$
$\tau \models \neg \Phi$	Si	$\neg (\tau \models \Phi)$
$\tau \models \Phi \vee \Psi$	Si	$(\tau \models \Phi) \vee (\tau \models \Psi)$
$\tau \models p . \Phi$	Si	$\tau_0 = \models p$ et $\tau_i^+ = \Phi$
$\tau \models \Phi \cup \Psi$	Si	$(\exists\, i > 0,\ \tau_i^{\geq} \models \Psi)$ et $(\forall\, 0 < k < i,\ \tau_k^{<} \models \Phi)$

La table ci-dessous définit la relation de satisfaction entre les traces et les formules logiques. La relation dénotée par $\tau \models \Phi$ signifie que la trace τ satisfait la formule logique Φ. Les trois premières clauses du tableau ci-dessus coïncident avec la sémantique de la logique de propositionnelle, une trace satisfait p .Φ si sa première action satisfait le prédicat p et son reste satisfait Φ. La formule $\Phi \bigvee \Psi$ est satisfaite par une trace τ s'il existe un suffixe de τ, appelé τ_i^+ satisfaisant Ψ et que tous les suffixes précédents (sans exception) satisfont Φ.

3.4 Spécification des règles de déduction de l'intrus

Pour établir des scénarios d'attaque qui sont défini comme étant un ensemble de privilèges obtenu par l'exploitation des vulnérabilités, les auteurs de [21] ont présenté quatre règles d'intrus qui sont définies dans le tableau suivant :

Avec :

- $priv.priv(v_x(h)$: est la séquence.
- $v_x(h)$: exprime la vulnérabilité x de la machine h.
- $cond(v_x(h))$: La condition pour l'exploitation de la vulnérabilité x de la machine h.

$priv_{init}$: Privilège initial
$priv$: Séquence d'attaque
Φ : Stratégie de l'intrus

Règle 1 : $\frac{\square}{priv=priv_{init}}$

Règle 2 : $\frac{priv^{+} \supseteq cond(v_x(h)) \wedge priv.priv(v_x(h)) \models \Phi}{priv=priv.priv(v_x(h))}$

Règle 3 : $\frac{(p_1.priv(v_x(h)).p_2)^{+} \supseteq cond(v_y(h)) \wedge p_1.priv(v_y(h)) \models \Phi}{priv=p_1.priv(v_y(h)).p_2}$

Règle 4 : $\frac{supp(priv) \wedge p_1.priv(v_x(h)).p_2 \models \Phi}{supp(p_1.priv(v_x(h)).p_2)}$

TABLE 3.1 – Règles de l'intrus [21]

- $supp(priv)$: Une fonction permettant de supprimer les privilèges et dans ce cas les traces une fois le but est atteint.

Règle 1 (Conditions initiales) : Signifie que l'intrus doit avoir au moins un privilège initial concernant sa victime pour commencer l'intrusion.

Règle 2 (Elévation de privilèges) : Signifie que l'intrus ne peut passer d'un état vers un autre et augmentant ses privilèges que si l'ensemble des privilèges acquis dans un premier lieu le mène à acquérir un autre ensemble de privilège dans un second temps.

Règle 3 (Oracle) : Signifie que l'intrus peut changer de direction c'est-à-dire quitter un chemin si ce dernier lui convient pas et que l'autre chemin qui va procéder est plus court et meilleur d'après sa stratégie.

Règle 4 (Effacement de traces) : une fois le but atteint, l'intrus se sert de cette étape pour effacer ses traces ce qui rend la détection de l'intrusion beaucoup plus difficile.

3.5 Construction du graphe d'attaques

L'exécution de la stratégie de l'intrus organisée dans un scénario d'intrusion qui permet de changer le système d'un état initial sûr, où la politique de sécurité est respectée, à un état final où l'objectif d'intrusion est atteint et la politique de sécurité est violée.

Dans [21] les auteurs ont supposé que les conditions de l'exploitation de la première vulnérabilité sont satisfaites, en d'autres termes l'intrus devrait avoir un ou plusieurs privilèges initiaux. La construction du graphe de privilèges est basé sur le fait qu'une attaque ne peut pas être réalisée jusqu'à ce que toutes ses conditions soient satisfaites tandis qu'une condition peut être satisfaite par n'importe quelle attaque, ainsi la construction du graphe d'attaques est faite graduellement par l'application des règles de déduction d'intrus données dans le tableau 2.

Le graphe d'attaque spécifie le point de départ de l'attaquant, les informations sur l'hôte ainsi que la topologie du réseau. Pour analyser la sécurité du réseau, basé sur l'analyse des incidents de la sécurité informatique et les actions de l'attaquant, il est à supposer que :

1. L'intrus ne possède pas de connaissances, à priori, dans le graphe complet. Il regarde seulement les attaques qu'il peut immédiatement effectuer.
2. L'intrus est doté d'une mémoire pour enregistrer l'ensemble des privilèges cumulés.
3. L'intrus est doté d'une raison, il ne met pas en application une attaque qui va diminuer les privilèges qu'il possède déjà.

Pour créer des scénarios d'attaque cherchant à exploiter les vulnérabilités du réseau, les auteurs de l'article [21] ont présenté un graphe d'attaque AG. La construction de ce graphe est basée sur le fait que le processus d'attaque change l'état quand un nouveau privilège a été acquis en appliquant les règles d'intrus décrites dans le tableau 2, les nœuds de graphe peuvent être marqués par les attaques liées au privilège acquis. Quand le processus d'attaque passe d'un état à l'autre, l'ensemble total de privilèges qu'il tient après la transition soit toujours supérieur à celui qu'il a eu avant la transition. La définition formelle de graphe d'attaques est comme :

Etant donné une séquence de privilèges d'un intrus $priv_1$ et un ensemble d'arcs $R \subseteq priv_1 \times priv_1$ qui correspond aux règle de l'intrus, un graphe d'attaques AG est le graphe orienté AG ($priv_1$,R)

3.6 Algorithme de construction du graphe d'attaque

L'algorithme suivant est une amélioration de l'algorithme présenté dans [2], il permet de construire le graphe d'attaque selon la stratégie de l'intrus (l'algorithme détaillé est dans l'annexe) :

Variables

GrapheAttaque : Graphe résultant en appliquant l'algorithme, c'est une liste d'adjacence qui contient initialement un seul nœud(Règle 1).

ListVul :C'est une liste qui contient les vulnérabilités du système.

ListNoeudCourant :c'est une liste qui contient les nœuds en cours de traitement.

trace :C'est une liste qui contient les nœuds construits à chaque étape.

ListNouveauNoeud :C'est une liste qui contient les nœuds construits à chaque étape quand la liste trace est satisfait la logique de l'intrus.

Algorithm 1 Construction du graphe d'attaque

```
Var logique,phi,psi,trace :Liste ;
Listevul :ListVul ;
GrapheAttaque,ListNœudCourant,ListNouveauNœud : ListeAdjacente ;
Début
    Tant que(ListVul <> vide) faire
Etape1 :
/*********************************Règle2*************************************/
     pour(i allant de 1 à taille(ListNœudCourant)) faire
      pour(j allant de 1 à taille(ListVul)) faire
       si inclusion(ListVul,ListNœudCourant) alors
          Créer un nouveau nœud ;
          Ajouter (nouveau nœud *.machine) à la liste trace ;
          Ajouter (nouveau nœud *.vulnérabilité) à la liste trace ;
        si satisfait(trace,logique) alors
          Mettre le nœud i comme étant le père de ce nouveau nœud
          Ajouter ce dernier à la liste ListNouveauNœud
          Supprimer la vulnérabilité numéro j
        finsi ;
       finsi ;
      finpour ;
          trace ← Nil ;
     finpour ;
Etape2 :
/*************Supprimer les doubles dans ListNouveauNœud *************/
     pour(i allant de 1 à taille(ListNouveauNœud)) faire
          LNN ← ListNouveauNœud;
    pour(j allant de 1 à taille(LNN)) faire
     si ((ListNouveauNœud*.suivant = LNN*.suivant) et(i<>j)) alors
        Ajouter le nœud i aux successeurs du père du nœud j
        Supprimer le nœud j de la liste ListNouveauNœud
     finsi ;
        LNN ← LNN ∗ .père ;
    finpour ;
   finpour ;
/*********************************Règle3*************************************/
pour(i allant de 1 à taille(ListNouveauNœud)) faire
     pour(j allant de 1 à taille(GrapheAttaque)) faire
      si (ListNouveauNœud*.père = GrapheAttaque*.père) alors
        Ajouter le nœud j du graphe au successeurs du père du nouveau nœud i
        Supprimer le nœud j de la liste ListNouveauNœud
      finsi ;
     finpour ;
   finpour ;
Etape3 :
          Ajouter ListNouveauNœud à GrapheAttaque
/*********************************Règle4*************************************/
          ListNœudCourant ← ListNouveauNœud;
          ListNouveauNœud ← null;
          ListVul ← ListVul.suivant;     finTantque ;
Fin.
```

3.6.1 Description de l'algorithme

L'algorithme proposé permet de produire un graphe d'attaques selon la stratégie de l'intrus, les nœuds de graphe représentent les attaques sachant que chaque attaque est un ensemble de privilèges acquis par l'exploitation des vulnérabilités.
Le principe de l'algorithme est défini en trois étapes :

Etape1 : consiste à parcourir la liste des vulnérabilités du réseau et de chercher un nœud de la liste nœud courant qui inclut l'ensemble des pré-conditions de la vulnérabilité dans son ensemble de privilèges, si la vulnérabilité est inclut dans un nœud trouvé dans la liste nœud courant alors un nouveau nœud est créé et va l'ajouter à la liste trace, si cette dernière satisfait la logique de l'intrus, le nouveau nœud crée est ajouté à la liste nouveau nœud et la vulnérabilité courante est supprimé.
Etape2 : consiste à supprimer les doubles dans la liste nouveau nœud et dans le graphe.
Etape3 : c'est l'étape qui permet d'ajouter les arcs entre les nœuds de graphe.

3.6.2 Calcul de la complexité de l'algorithme de construction de graphes d'attaques

Soit n : la taille de la liste ListVul, m : la taille de la liste ListNœudCourant, k : la taille de la liste ListNouveauNœud, tel que m < n, k < n.

Instruction	Complexité
Etape 1	$135\text{mn}^2 + 450mn + 90n^2 + 243m + 300n + 162 \simeq 135n^3 + 540n^2 + 543n + 162$
Etape 2	$81\text{k}^2 + 111k + 38 \simeq 81n^2 + 111n + 38$
Etape 3	4
Total	$135\text{n}^3 + 621n^2 + 654n + 200$

La complexité de l'algorithme est :$135n^4 + 621n^3 + 654n^2 + 200n$, elle est donc de $O(n^4)$, qui est une complexité polyno*miale*.

3.7 Conclusion

Dans ce chapitre, nous avons présenté un modèle du processus d'attaque qui représente les étapes qui peut suivre l'intrus, et nous avons présenté les quatre règles d'intrus ,et par la suite nous avons adopté l'algorithme qui permet de générer un graphe d'attaque selon une logique préalablement définie par l'intrus.Pour cela nous allons besoin de modéliser notre systeme pour implémenter notre algorithme

4

Conception

4.1 Introduction

La plupart des nouveaux langages sont orientés objet. Le passage de la programmation fonctionnelle à l'orienté objet n'était pas facile. L'un des soucis était d'avoir une idée globale en avance de ce qu'on doit programmer. En occurrence UML qui nous a permis de faire la conception de notre application. Pour ce faire nous allons commencé par le diagramme de cas d'utilisation qui permet de donner une vue globale de l'application. En deuxième lieu nous allons présenter le diagramme de classe . Et en fin nous allons donner la chronologie des opérations par le diagramme de séquences.

4.2 Pourquoi modéliser ?

Un modèle est une simplification de la réalité qui permet de mieux comprendre le système à développé. Il permet de [39] :

- visualiser le système comme il est ou comme il devrait l'être.
- valider le modèle vis à vis des clients.
- spécifier les structures de données et le comportement du système.
- fournir un guide pour la construction du système.
- documenter le système et les décisions prises.

4.3 Modélisation de notre système

Nous avons modélisé notre système en utilisant le modèle UML (Unified Modeling Langage) et une méthode UP (Unified Process) qui s'agit d'une démarche s'appuyant sur la modélisation UML.

A la fin de ce projet, nous devrons disposer d'un outil graphique qui permet de :

- Modéliser une topologie de réseau en glissant les éléments de réseau dans une fenêtre,
- Définir les liens entre les nœuds du réseau,
- Visualiser et modifier les propriétés de tous les composants du réseau,
- Enregistrer une topologie de réseau,
- Importer une topologie de réseau,
- Générer le graphe ou le chemin (scénario) d'attaques une fois le réseau est spécifié et la logique de l'intrus est satisfaite,
- Visualiser le graphe d'attaques.

4.3.1 Spécification des besoins

Le but ultime de cette application est de générer un graphe d'attaques à partir d'un réseau créé par l'utilisateur. Cette génération doit être faite en utlisant les règles et la stratégie d'intrus. Le diagramme de cas d'utilisation d'UML suivant permet de bien spéciér les besoins et objectifs de l'application :

Avant de générer un scénario d'attaques, il faut d'abord définir le modèle réseau

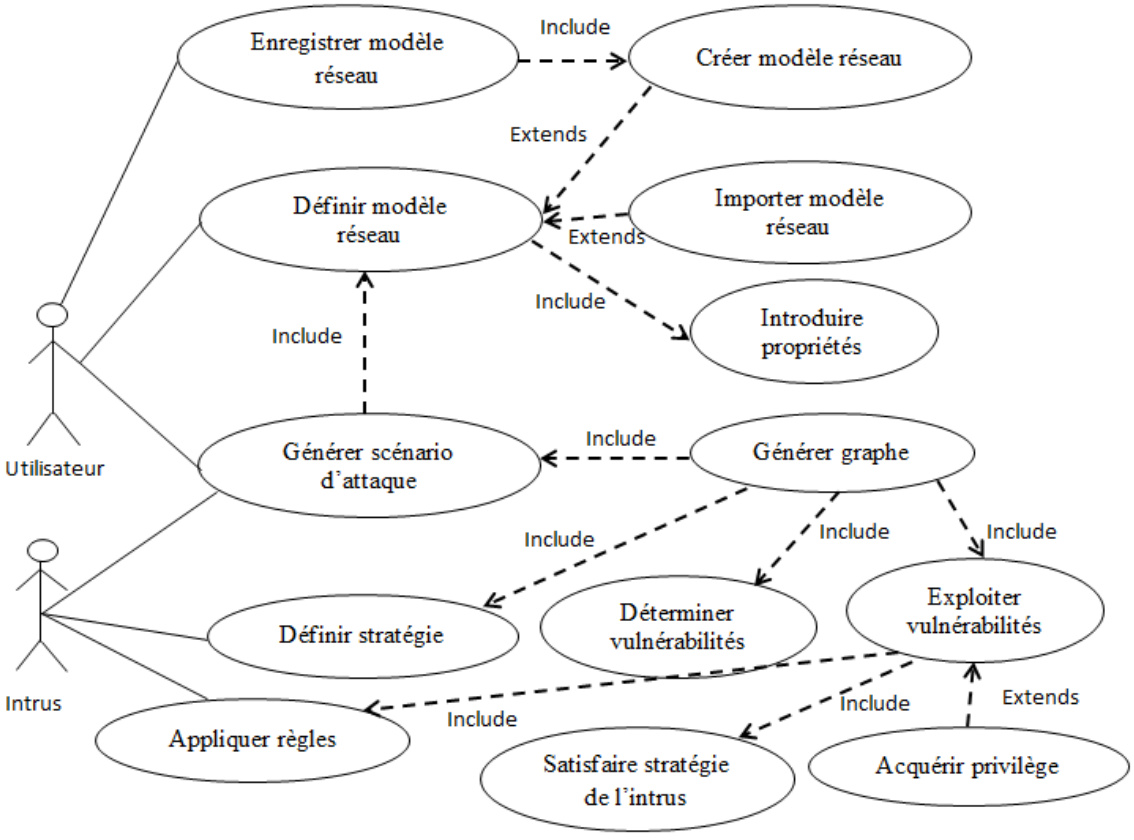

FIGURE 4.1 – Diagramme de cas d'utilisation

et cela en le créant directement ou en important un modèle déja existant (déja enregistré), pour chaque nouveau modèle il faut définir pour chacun des ses équipement ses propriétés et ses vulnérabilités.

Le graphe d'attaques qui sera proposé dépendra du comportement de l'intrus et de sa façon d'interagir pour atteindre son objectif. Son rôle et ses cas d'utilisation sont définis comme suit :

- **Choisir victime :** Après avoir scanner les machine du réseau, l'intrus choisi la cible qui l'interesse.
- **Appliquer les règles d'intrusions :** L'intrus utilise ces règles afin d'exploiter les vulnérabilités dans le but d'acquérir de nouveaux privilèges.
- **Acquérir un privilège :** l'intrus acqui le privilège si la stratégie de l'intrus est aquise.
- **Générer un graphe d'attaque :** Donne tous les chemins possibles que l'intrus peut générer.

4.3.2 Analyse et conception

Dans la conception de l'application, nous avons utilisé deux diagrammes d'UML : le diagramme de classes qui permet de présenter les classes et les interfaces de notre système ainsi que les différentes relations entre celles-ci. À partir de ce diagramme nous retrouvons les corps les différentes classes de notre application, et le diagramme de séquence qui permet de de décrire comment se déroulent les actions entre les objets.

Diagramme de classes

Le diagramme de classe représente une vue statique de l'application. Dans le cadre de notre projet, ce dernier se compose principalement de 21 classes , les relations entre eux sont illustrées dans la figure suivante :

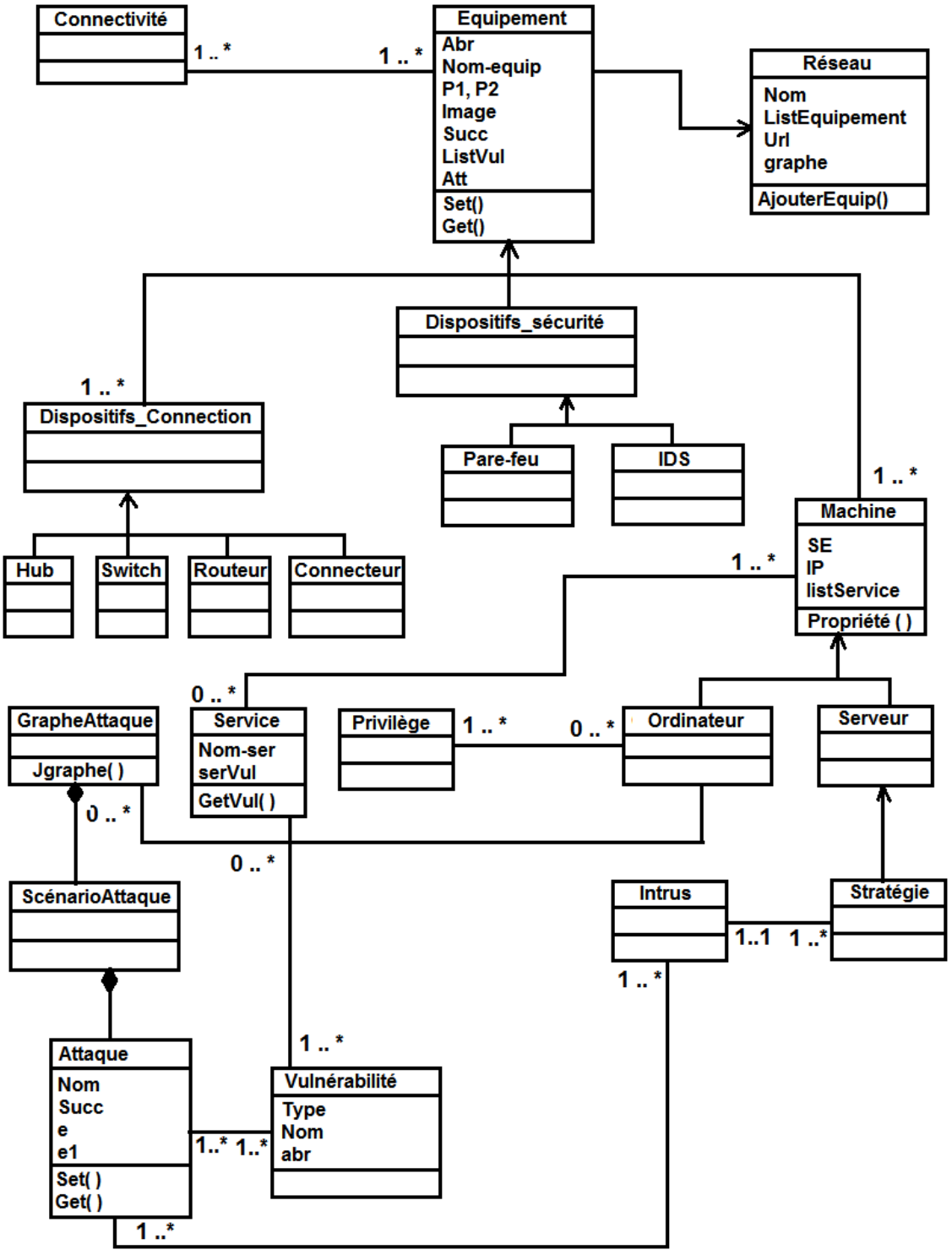

FIGURE 4.2 – Diagramme de classe

Diagramme de séquence

Le diagramme de séquence représente la succession chronologique des opérations réalisées par l'utilisateur, il peut être déduit à partir du diagramme de cas d'utilisation. Notre diagramme de séquence est présenté dans la figure suivante :

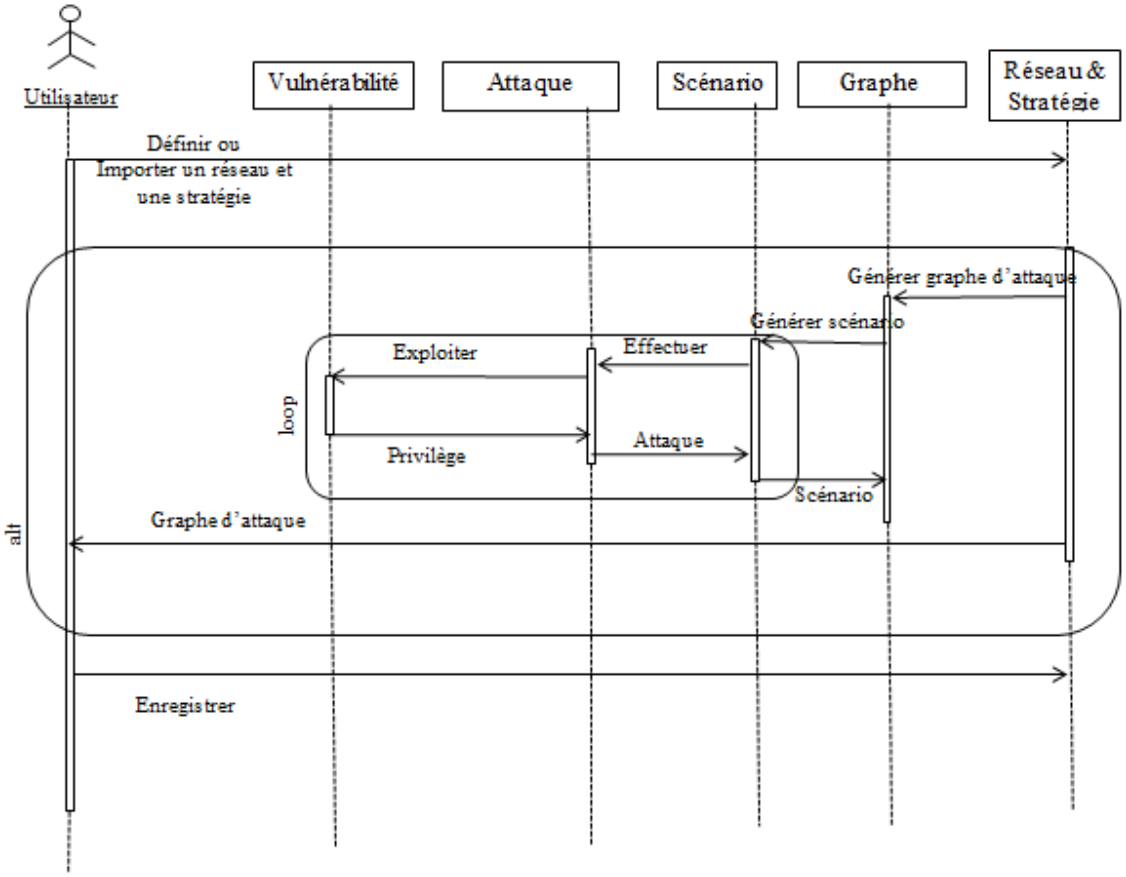

FIGURE 4.3 – Diagramme de séquence

4.4 Conclusion

Dans le cadre de ce chapitre nous avons expliqué le choix du modèle UML (Unified Modeling Langage) et que la démarche choisi est UP (Unified process), puis nous avons présenté le diagramme de cas d'utilisation, le diagramme de classe ainsi que le diagramme de séquence associés à notre application.

5
Réalisation

5.1 Introduction

Nous allons étudier dans ce chapitre les deux dernière étapes de la démarche UP à savoir, la réalisation et tests. Dans la première partie nous allons préciser l'environnement et les outils nécessaires que nous allons utiliser pour le développement de l'application. Nous expliquerons également, le déroulement des étapes pour générer un graphe d'attaques, par la suite on passera à la deuxième étape qui nous permettra de tester l'application.

5.2 Environnement et outils de développement

- La JDK (Java Development Kit) version 1.6.0 20 : l'environnement dans lequel le code java est compilé pour être transformé en bytecode (code intermédiaire) afin que la JVM (Java Virtuel Machine) de java puisse l'interpréter.
- L'environnement Eclipse : Eclipse est un environnement de développement intégré (IDE, Integrated Development Environment) Java qui a été créé dont le but de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.
- API externe JGraph : une librairie externe Swing utilisée pour le dessin du graphe d'attaques.
- Les API externes Swingx et PowerSwing : deux librairies externes Swing utilisées pour les interfaces graphiques.
- les API externes Xstream et Xpp3 : deux librairies pour l'enregistrement des fichier sous format XML.
- Xampp : ensemble de serveurs dont nous avons utilisé le serveur de bases de données MySql pour gérer la base de données des vulnérabilités.
- La base de données des vulnérabilités OSVDB(Open Source Vulnerabilty Data Base) : C'est base de données qui contient les vulnérabilités, elle est remplie manuellement.

5.3 Description de l'application

La fenêtre principale se compose d'une barre de menu, une barre d'outils, deux panneaux et un espace de travail.

Sur la barre de menu on trouve le menu Fichier dans lequel on peut débuter avec un nouveau réseau, ouvrir et sauvegarder des fichiers réseau. Le menu Action, quant à lui, présente l'ensemble des solutions algorithmiques traitées par l'application(graphes et scénarios d'attaques). La barre d'outils contient des boutons per-

mettant les manipulations de base relatives à l'ajout des composants du réseau, l'affichage des vulnérabilités du réseau, un bouton "define strategy" qui nous a permet d'accéder á une autre fenêtre pour définir la stratégie de l'intrus puis on génère le graphe d'attaques selon la stratégie de l'intrus, ainsi qu'à la consultation de la base de données des vulnérabiltés OSVDB. Sur le coté gauche se trouve un panneau, ce dernier permet de visualiser les noms des réseaux ouverts dans l'application . Quant au panneau ce trouvant sur le coté droit, il permet de visualiser et de spécifier les propriétés de chacun des composants du réseau.

Enfin, le reste de la fenêtre est occupé par un espace de travail dans lequel seront dessinés touts les équipements concernant le réseau. La figure ci-après permet de concrêtiser cette description :

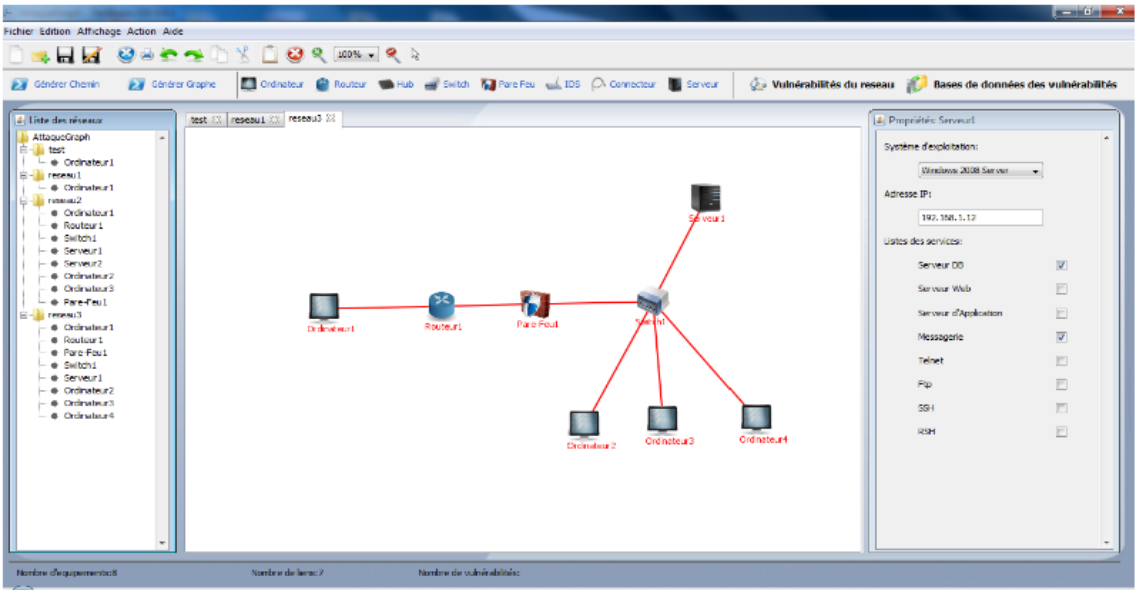

FIGURE 5.1 – Fenêtre principale

5.3.1 Création d'un nouveau réseau

La création d'un réseau se fait de deux manières, la premiére consiste á importer un réseau á partir d'un fichier XML déja enregistré, dans ce cas, les propriétés des équipements du réseau sont déja définies, nous pouvons les modifier en utilisant le panneau propriétés se trouvant dans le coté droit de la fenêtre.

Quant à la deuxième faÃ§on, elle consiste à insérer interactivement les équipements

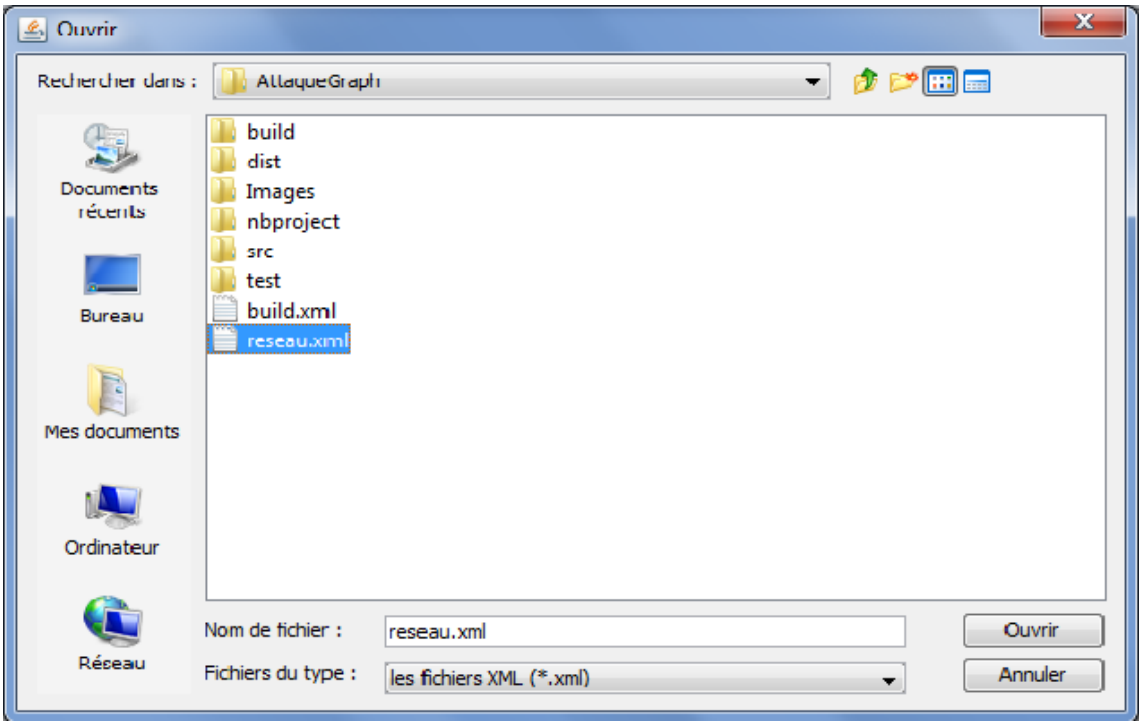

FIGURE 5.2 – Importer un réseau

du réseau et cela en les faisant glisser à partir de la barre d'outils et à spécifier au fur et à mesure leurs propriétés.

Nous pouvons aussi renommer, supprimer, afficher les propriétés et les vulnérabilités de chaque équipement et cela avec un clic droit sur chaque équipement.

En fin nous avons la possibilité de bouger les équipements avec la souris, ce qui nous permettra de mieux organiser notre réseau.

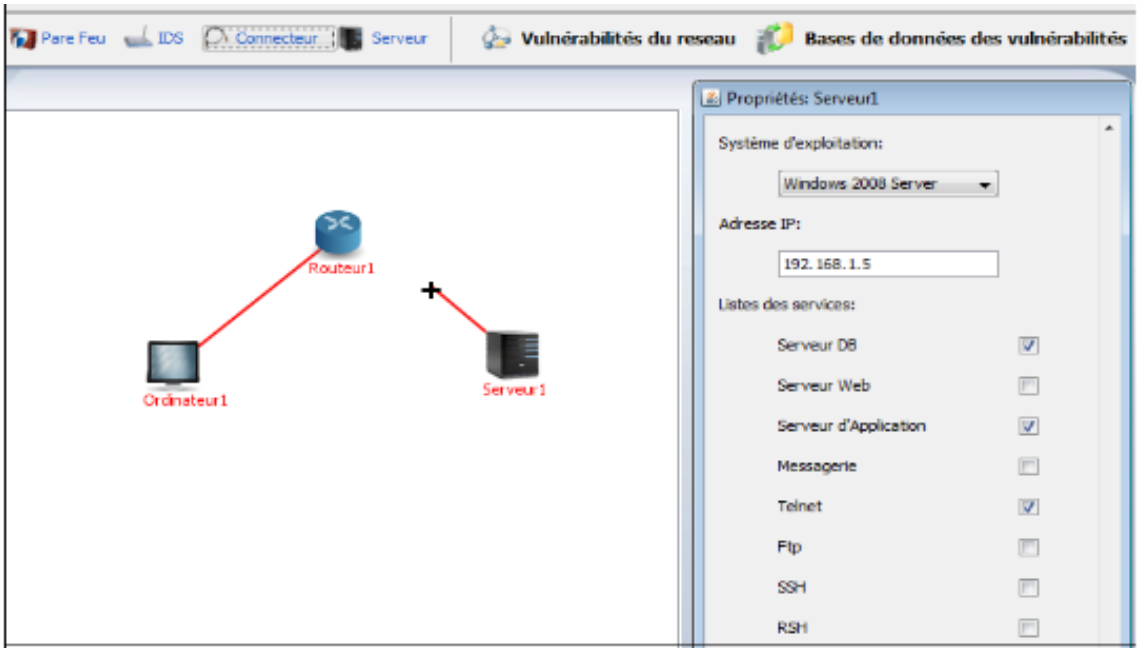

FIGURE 5.3 – Dessiner un réseau et spésifier ses propriétés

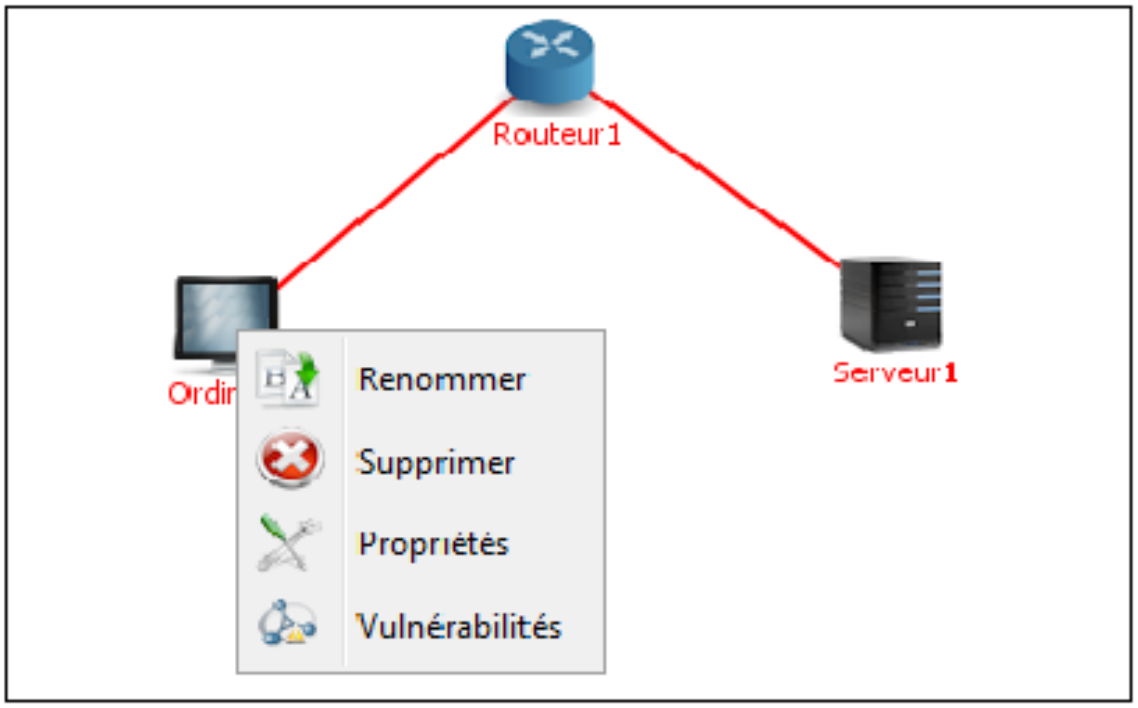

FIGURE 5.4 – Opération sur un équipement

5.3.2 Génération d'un graphe d'attaques

prés avoir importé ou créé interactivement le réseau, en spécifiant un équipement qui sera considéré comme étant l'attaquant aprés avoir cliqué sur le bouton "Define strategy" puis on défine la logique de l'intrus et on clique sur le bouton "Générer Graphe".

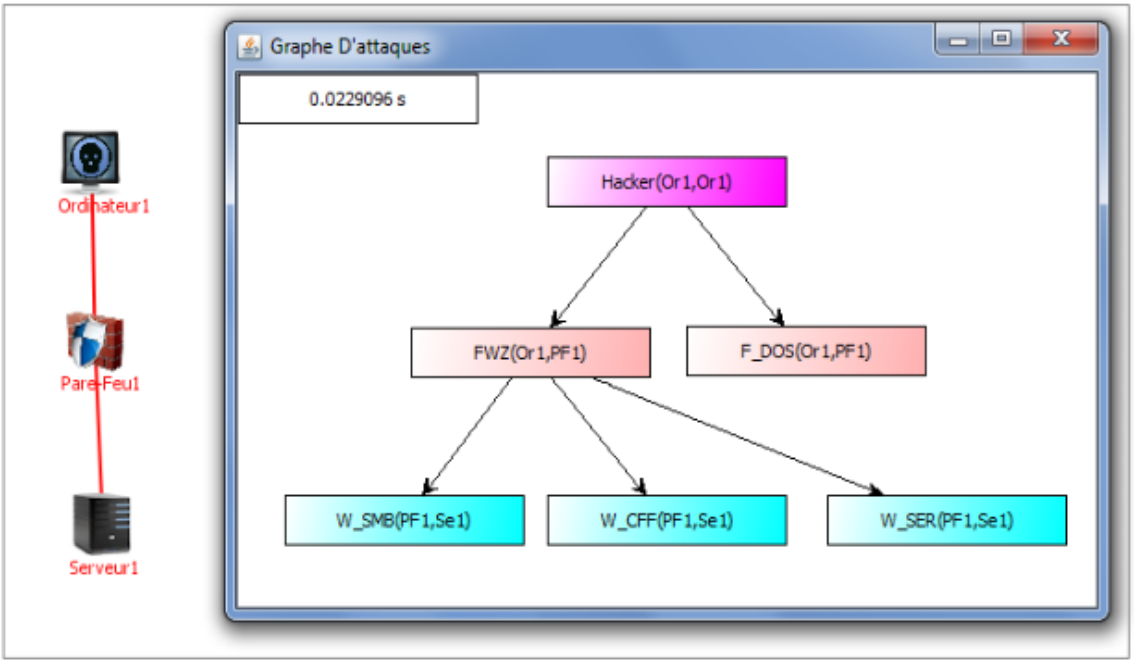

FIGURE 5.5 – Graphe d'attaques

la fenêtre de de stratégie de l'intus est la suivante :

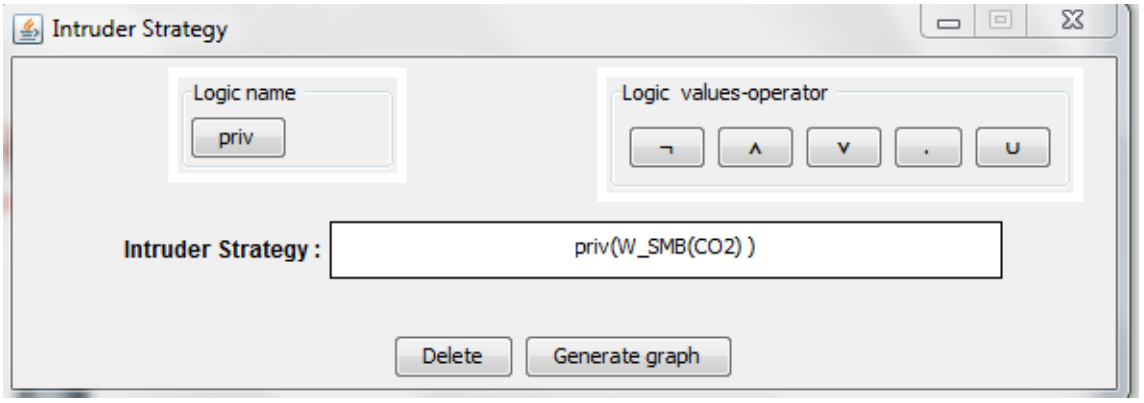

FIGURE 5.6 – Fenêtre de logique

5.4 Exemple explicatif

Dans cet exemple, nous avons un réseau composé de quatre hôtes qui sont : ordinateur1 (Or1), Ordinateur2 (Or2), Ordinateur3 (Or3), Ordinateur4 (Or4) et Serveur1 (Se1) et d'un pare-feu. La machine Or1 est considérée comme étant l'intrus, le graphe d'attaques qui sera généré va représenter tous les scénarios d'attaques que peut effectuer l'intrus sur toutes les machines du réseau. La figure ci-aprés shématise le modèle du réseau de cet exemple :

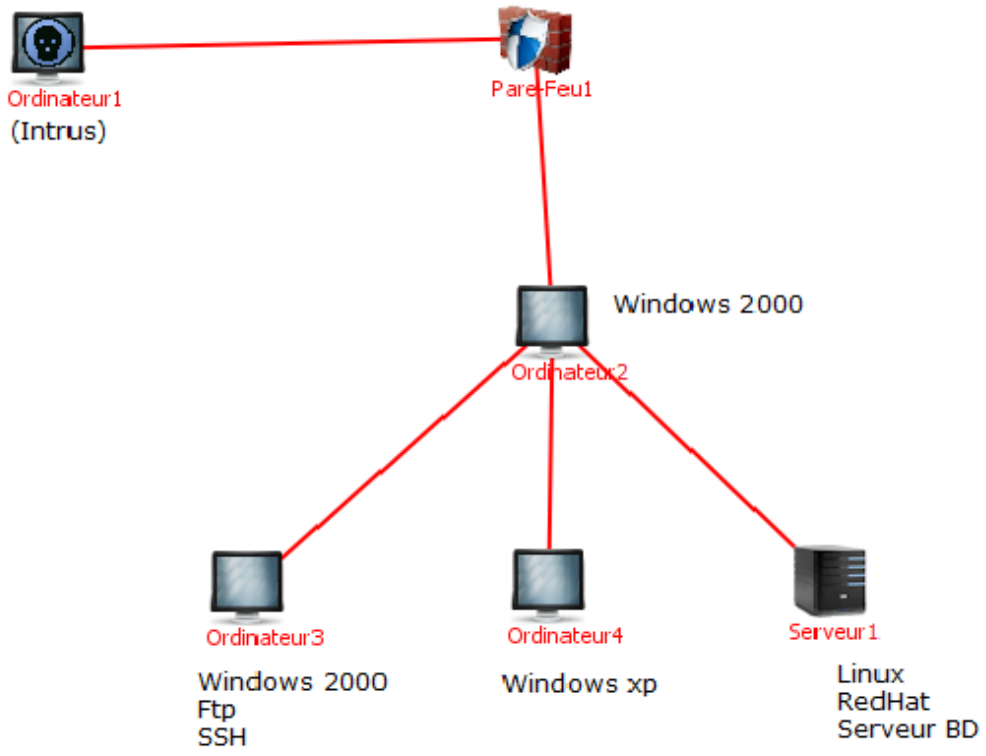

FIGURE 5.7 – Réseau

Le graphe d'attaques correspondant à cet exemple se génère en exploitant toutes les vulnérabilités du réseau et cela en utilisant les différentes règles expliquées dans le chapitre trois et la stratégie de l'intrus.

5.5 Conclusion

Dans ce chapitre, nous avons expliqué briévement le fonctionnement de l'application à travers un exemple. Cette application permet de générer automatiquement un graphe d'attaques pour un réseau et une logique de l'intrus donné par l'utilisateur.

Conclusion générale

Dans le cadre de ce mémoire, l'objectif était la génération automatique des graphes d'attaques dans les systèmes informatique à partir d'un réseau donné par l'utilisateur et selon la stratégie de l'intrus.

Dans le premier chapitre, nous avons présenté quelques notion de base sur les attaque et la sécurité informatique où nous avons donné un aperçu des différentes techniques et mécanisme de sécurité.

Dans le deuxième chapite, nous avons présenté une synthèse des travaux déja réalisés dans ce contexte.

Dans troixième chapitre, nous avons donné la démache étudiée puis l'algorithme de construction du graphe d'attaque amélioré.

Dans le quatrième chapitre, nous avons présenté la modélisation en utilisant le modèle UML et la conception de notre application.

Dans le dernier chapitre, nous avons présenté brièvement l'application.

Enfin, nous avons réaliser une application permettant de générer des scénarios d'attaques pour n'importe quel réseau, cependant, ce travail serait encore plus intéressant si nous peuvons générer des scénarios d'attaques pour les réseaux sans fil

Bibliographie

[**24**] M. Albanese, S. Jajodia S. Noel, "Time-Efficient and Cost-Effective Network Hardening Using Attack Graphs", 2012.

[**14**] P. Ammann R. Ritchey. Using Model Checking to Analyze Network Vulnerabilities. In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, pp. 156-165, 2000.

[**32**] S. Bhattacharya, S. Malhotra, S.K. Ghsoht, "A Scalable Representation towards Attack Graph Generation", Proceedings of the 2008 1st International Conference on Information Technology, Gdansk, Poland, 19-21 Mai 2008.

[**7**] Laurent Bloch, Christophe Wolfhugel,"Sécurité informatique : Principes et méthode à l'usage des DSI, RSSI et administrateurs", deuxième édition, Eyrolles, 2009.

[**23**] M. Dacier, "Vers une évaluation quantitative de la sécurité informatique", Thèse de doctorat, Université Catholique de Louvain, Décembre 1994.

[**28**] W.H.Debany, T.J.Parisi, "Advanced Cyber Attack Modeling, Analysis, and Visualization", Air Force Research Labratory, George Mason University, Mars 2010.

[**11**] lamia hamza, « Génération automatique de scénario d'attaques pour les systèmes de détection d'intrusions, mémoire de magitère , université de Bejaia , 2005

[**22**] L. Hamza, generation des scenarios dattaques pour les systems de

detection dintrusions, poster, 3ème forum de bejaia, 2009. Algérie.

[**21**] L. Hamza, K. Adi, Formal Technique for Discovering Complex Attacks in Computer Systems, International Conference on New methodologies Tools and Techniques, Italie, 2007.

[**20**] L. Hamza, K. Adi et K. El Ghemhioui. Automatic generation of attack scenarios for intrusion detection systems. International Conference on Internet and Web Applications and Services. Published by the IEEE Computer Society Press. 2006.

[**27**] C. Healey R, St. Amant, P. Ning, D. Xu, "Building Attack Scenarios through Integration of Complementary Alert Correlation Methods", Proceedings of the 11th Annual Network and Distributed System Security Symposium, 2004.

[**16**] S. Kaushik P.Ammann, D. Wijesekera, "Scalable, Graph-Based Network Vulnerability Analysis", Proceedings of 9th ACM Conference on Computer and Communications Security (CCS), Washington, DC, Novembre 2002.

[**2**] M.Lounis, S.Ould Amara :"Génération automatique des attaques complèxes", mémoire de Master, Université de Béjaia, 2011.

[**10**] L. Mé C. Michel. Adele "an Attack Description Language for Knowledge based Intrusion Detection" Proceedings of the 16th International Conference on Information Security (IFIP/SEC2001), Paris, France, Kluwer, pp. 353-365, Juin 2001.

[**29**] Mohammed S. Gadelrab, Anas Abou El Kalam, Yves Deswarte, "Defining categories to select representative attack test-cases", Proceedings of the 2007 ACM workshop on Quality of protection (QoP07), Alexandria, Virginia, USA, 2007.

[**3**] Mohammed S. Gadelrab, Anas Abou El Kalam, and Yves Deswarte, Execution Patterns in Automatic Malware and Human-centric Attacks, NCA 2008 : Proceedings of the 2008 Seventh IEEE International Symposium on Network Computing and Applications vol. 29-36.

[**1**] S.Natkin :"Les protocoles de sécurité d'Internet", Dunod science sup, 2002.

[**34**] Nilima R. Patil, Nitin N. Patil, "A comparative study of network vulnerability analysis using attack graph", World Journal of Science and Technology, Avril 2012.

[**19**] S. Noel and S. Jajodia, "Managing Attack Graph Complexity Through Visual Hierarchical Aggregation", Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, New York : ACM Press, 2004.

[**17**] B. O'Berry V. Kumar J. Srivastava A. Lazarevic (eds.) S. Jajodia, S. Noel, "Opological Analysis of Network Attack Vulnerability in Managing Threats : Issues, Approaches and Challenges", Kluwer Academic Publisher, 2004.

[**18**] B. OBerry S. Noel, S. Jajodia and M. Jacobs, "Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs", Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas, Nevada, 2003.

[**39**] R. Ogor, "Modélisation avec UML, mai 2003

[**8**] R. Ortalo F.Cuppens. LAMBDA "A Language to Model a Database for Detection" Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID2000), LNCS 1907, pp. 197-216, October 2000.

[**12**] K. Piwowarski K. Ingols, R. Lippmann, "Practical Attack Graph Generation for Network Defense", In : Proceedings of Computer Security Applications Conference, ACSAC'06, pp.1-12, Miami Beach, FL, USA Décembre, 2006.

[**33**] S. Rekhis, N. Boudriga, "Formal reconstruction of attack scenarios in mobile ad hoc and sensor networks", EURASIP Journal on Wireless Communications and Networking, 2011.

[**37**] S. Roschke, F. Cheng, R. Schuppenies Ch. Meinel, "Towards Unifying Vulnerability Information for Attack Graph Construction", Hasso Plattner

Institute (HPI), University of Potsdam, P.O. Box 900460, 14440, Potsdam, Germany, 2009.

[**36**] L. Wang, S. Noel S. Jajodia, "Minimum-cost Network Hardening Using Attack Graphs", Computer Communications, vol. 29, no. 18, pp. 3812-3824, Novembre2006.

[**25**] M. Saber, T. Bouchentouf, A. Benazzi, "Generation of Attack Scenarios for Evaluating IDS", juin 2012.

[**15**] O.M. Sheyner. Scenario Graphs and Attack Graphs. Thesis, School of Computer Science Computer Science Department Carnegie Mellon University Pittsburgh, Avril, 2004.

[**38**] O. Sheyner, J. Haines, S. Jha, R. Lippmann, J. Wing, Automated Generation and Analysis of Attack Graphs, In : Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, 2002.pp.1-12.

[**35**] Swiler, al, "Computer-attack graph generation tool", DARPA Information Survivability Conference and Exposition, 2001.

[**30**] Tito Waluyo Purboyo, Kuspriyanto, "Some Algorithms for Generating Attack Graph", School of Electrical Engineering and Informatics Institute Teknologi Bandung, Indonesia, Aout 2012.

[**13**] J. M. Wing, R. P. Lippmann, J. Haines, O. Sheyner, S. Jha, "Automated Generation and Analysis of Attack Graphs", in 2002 IEEE Symposium on Security and Privacy, Oakland, California, 2002.

[**26**] B. Zhong, K. Lu, X. Pan, Z. Wu, "Reverse Search Based Network Attack Graph Generation", Proceedings of International Conference on Computational Intelligence and Software Engineering (CiSE) 2009, 11-13 Décembre 2009.

[**31**] S. Zhong, D. Yan, C. Liu, "Automatic Generation of Host-based Network Attack Graph", World Congress on Computer Science and Information Engineering, 2009.

[**4**] http ://www.securite-informatique.gouv.fr

[**5**] http ://www.futura-sciences.com

[**6**] http ://www.wikipedia.org

1- Algorithme de construction du graphe d'attaque

********** taille **********

Fonction taille(A :ListAdjacente) :entier ;

Debut

si (A=Nil) ***alors***

taille $\leftarrow 0$;

sinon

$taille \leftarrow taille(A*.suivant) + 1$;

finsi;

fin.

**********$egale$**********

Fonction egale$(trace, logique : Liste) : booleen$;

Debut

si$(trace*.machine = logique*.machine) et (trace*.vul = logique*.vul)$***alors***

$egale \leftarrow vrai$;

sinon

$egale \leftarrow faux$;

finsi

**********$inclusion$**********

Fonction inclusion$(V : vul; NC : Adjacence) : booleen$;

Debut

Tantque$(NC <> Nil) et (V.machine <> NC.priv*.machine) ou (V.nom <> NC.priv*$
$.vulnérabilité)))$***faire***

$NC.priv \leftarrow NC.priv*.suivant$;

Fin Tantque

si$(V.machine = NC.priv*.machine) et (V.nom = NC.priv*.vulnérabilité))$***alors***

$inclusion \leftarrow vrai$;

sinon

$inclusion \leftarrow faux$;

finsi;

fin.

$**********satisfaction**********$

Fonction satisfait$(trace, logique : Liste) : booleen$;

var$opérateur = and, or, not, sequence, until$;

Debut

$logique = phiopérateurpsi$

$Logique*.machine \leftarrow list*.machine$;

$Logique*.vul \leftarrow list*.vul$;

si$((taille(logique) = 1)etegale(trace, logique))$***alors***

$satisfait \leftarrow vrai$;

sinon

selon que$(opérateur)$***faire***

$and : satisfait \leftarrow satisfait(trace, phi)etsatisfait(trace, psi)$;

$or : satisfait \leftarrow satisfait(trace, phi)ousatisfait(trace, psi)$;

$not : satisfait \leftarrow nonsatisfait(trace, phi)$;

$until :$

Tantque$((trace <> Nil)et(satisfait(trace, phi))et(non(satisfait(trace, psi)))$***faire***

$trace \leftarrow trace*.suivant$;

FinTantque

si$satisfait(trace, psi)$***alors***

$satisfait \leftarrow vrai$;

sinon

$satisfait \leftarrow faux$;

finsi;

$sequence :$

Tantque$((trace <> Nil)et(satisfait(trace, phi))$***faire***

$trace \leftarrow trace * .suivant$;

FinTantque

si$(trace = Nil)$***alors***

$satisfait \leftarrow vrai$;

sinon

$satisfait \leftarrow faux$;

$autres : satisfait \leftarrow faux$;

finsi

fin.

1.1 Algorithme construisant le graphe d'attaques

Type Liste=*Priv

Priv=*enregistrement*

machine :chaîne de caractères ;

vulnérabilité :chaîne de caractères ;

suivant :Liste ;

finpriv ;

Type ListVul=*Vul

Vul=*enregistrement*

machine :chaîne de caractères ;

nom :chaîne de caractères ;

suivant :ListVul ;

finVul ;

Type ListeAdjacente=*adjacence

adjacence=*enregistrement*

priv :ListeAdjacente ;

suivant :ListeAdjacente ;

finadjacence ;

Var logique,phi,psi,trace :Liste ;

Listevul :ListVul ;

GrapheAttaque,ListNœudCourant,ListNouveauNœud : ListeAdjacente ;

Début

/**********Règle1**********/

nouveau(trace) ;

lire(GrapheAttaque*.priv*.machine) ;

lire(GrapheAttaque*.priv*.vulnérabilité) ;

$trace * .machine \leftarrow GrapheAttaque * .priv * .machine$;

$trace * .vulnérabilité \leftarrow GrapheAttaque * .priv * .vulnérabilité$;

Tant que(ListVul <> vide) ***faire***

Etape1 :

/*******************************Règle2***********************************/

pour(i allant de 1 à taille(ListNœudCourant)) ***faire***

pour(j allant de 1 à taille(ListVul)) ***faire***

si inclusion(ListVul,ListNœudCourant) ***alors***

Créer un nouveau nœud ;

Ajouter (nouveau nœud *.machine) à la liste trace ;

Ajouter (nouveau nœud *.vulnérabilité) à la liste trace ;

si satisfait(trace,logique) ***alors***

Mettre le nœud i comme étant le père de ce nouveau nœud

Ajouter ce dernier à la liste ListNouveauNœud

Supprimer la vulnérabilité numéro j

finsi ;

finsi ;

finpour ;

$trace \leftarrow Nil$;

finpour ;

Etape2 :

/*************Supprimer les doubles dans ListNouveauNœud *************/

pour(i allant de 1 à taille(ListNouveauNœud)) ***faire***

$LNN \leftarrow ListNouveauN \not\in ud$;

pour(j allant de 1 à taille(LNN)) ***faire***

si ((ListNouveauNœud*.suivant = LNN*.suivant) et(i<>j)) ***alors***

Ajouter le nœud i aux successeurs du père du nœud j

Supprimer le nœud j de la liste ListNouveauNœud

finsi ;

$LNN \leftarrow LNN * .père$;

finpour ;

finpour ;

/**********Règle3**********/

pour(i allant de 1 à taille(ListNouveauNœud)) ***faire***

pour(j allant de 1 à taille(GrapheAttaque)) ***faire***

si (ListNouveauNœud*.père = GrapheAttaque*.père) ***alors***

Ajouter le nœud j du graphe au successeurs du père du nouveau nœud i

Supprimer le nœud j de la liste ListNouveauNœud

finsi ;

finpour ;

finpour ;

Etape3 :

Ajouter ListNouveauNœud à GrapheAttaque

/**********Règle4**********/

ListNœudCourant $\leftarrow$ $ListNouveauN \lhd ud$;

ListNouveauNœud $\leftarrow$ $null$;

finTantque ;

ListVul $\leftarrow$ $ListVul.suivant$;

Fin.

1.2 La complexité des fonctions utilisées dans l'algorithme

Fonction	Complexité
Taille	3n+2
Egale	4
Inclusion	6n+9
Satisfait	9n+23

Instruction	Complexité
Etape 1	$135\text{mn}^2 + 450mn + 90n^2 + 243m + 300n + 162 \simeq 135n^3 + 540n^2 + 543n + 162$
Etape 2	$81\text{k}^2 + 111k + 38 \simeq 81n^2 + 111n + 38$
Etape 3	4
Total	$135\text{n}^3 + 621n^2 + 654n + 200$

La complexité de l'algorithme est :$135n^4 + 621n^3 + 654n^2 + 200n + 5$, elle est donc de $O(n^4)$, qui est une complexité polyno*miale*.

Résumé

La sécurité informatique est de nos jours devenue un problème majeur dans la gestion des réseaux d'entreprise ainsi que pour les particuliers toujours plus nombreux à se connecter, à Internet. Les attaques réalisées par des utilisateurs malveillants et visant à exploiter les vulnérabilités de système d'information sont plus en plus fréquentes. Dans ce contexte, la sécurité des ces systèmes est devenue un souci majeur. Dans ce domaine, la génération automatique des graphes d'attaques est bien utile pour l'analyse et la configuration de la sécurité réseau ainsi que pour la détection des attaques informatique.

Le but de ce projet de recherche est de réaliser une application graphique en langage JAVA qui permet de générer des graphes d'attaques à partir d'un modèle de réseau importé d'un fichier XML ou crée manuellement selon la stratégie d'intrus spécifiée.

Mots clés : Sécurité informatique, scénarios d'attaque, graphes d'attaque, JAVA, Stratégie de l'intrus, pare-feu, IDS, Antivirus.