République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme
de master Recherche en Informatique

Thème

Implémentation d'une technique formelle pour la génération de scénarios d'attaques sous Java.

Soutenu devant le jury composé de :

Présidente	Mme	S. BOULFKHAR
Promotrice	Mlle	L. HAMZA
Examinateurs	Mme	S. AIT HACENE
	Mlle	A. TIAB

Présenté par :

Mlle *Aida BOUKEMOUCHE*
M. *Samir IDIR*

Juin 2016

Remerciements

Tout acte de recherche n'est que concrétisation d'un travail collectif.

Avant de présenter ce mémoire nous tenont à remercier notre Promotrice Mlle L.Hamza pour l'intérêt porté à notre travail, ses conseils, la qualité de son encadrement, ses encouragements, sa disponibilité et ses compétences scientifiques qui nous ont permis d'élargir nos connaissances.

Notre reconnaissance va également aux membres du Jury pour nous avoir fait l'honneur d'y participer.

Un grand merci à tous nos enseignants et plus particulièrement à ceux de Master 2 pour leurs disponibilité et précieux conseils.

Notre reconnaissance est également adressée à M. OMAR Mawloud, pour sa présence, son écoute et sa disponibilité tout au long de notre cursus.

Nos sincères remerciements vont à Mme G. Lahdiri , la secrétaire de notre département, pour sa remarquable gentillesse et M. K. Amroun pour sa présence, son écoute et sa disponibilité.

A tous nos professeurs.

Nous ne pourrions clore ces remerciements sans citer Nos parents sans qui notre travail n'aurai jamais pu aboutir.

Dédicaces

Louange à Dieu, sans Lui rien de tout cela n'aurait pu être.

A vous, très chers parents, auxquels on doit ce que nous sommes, puisse notre travail présent et futur être digne de votre fierté.

A nos très chers frères, soeurs, beaux frères et belles soeurs pour leurs compréhensions et encouragements.

A toutes nos familles grand parents, oncles, tantes, cousins et cousines que Dieu vous protège et vous garde en Bonne Santé.

A tous nos amis et amies.

A toute la promotion 2016.

A toutes les personnes qui nous ont soutenu et participé de prés ou de loin à l'élaboration de ce mémoire.

Nous vous dédions ce modeste travail.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

API	**A**pplication **P**rogramming **I**nterface.
CTL	**C**omputation **T**ree **L**ogic.
CVSS	**C**ommon **V**ulnerability **S**coring **S**ystem.
FTP	**F**ile **T**ransfert **P**rotocol.
HTTP	**H**ypertext **T**ransfert **P**rotocol.
IDS	**I**ntrusion **D**etection **S**ystem.
IIS	**I**nternet **I**nformation **S**ervices.
IP	**I**nternet **P**rotocol.
JAR	**J**ava **A**rchive.
JDOM	**J**ava **D**ocument **O**bject **M**odel.
JESS	**J**ava **E**xpert **S**ystem **S**hell.
JGRAPH	**J**ava **G**raph.
LTL	**L**inear **T**ree **L**ogic.
NetSPA	**N**etwork **S**ecurity **P**lanning **A**rchitecture.
NVD	**N**ational **V**ulnerability **D**atabase.
OSVDB	**O**pen **S**ource **V**ulnerabilty **D**ata Base.
RCP	**R**eality **C**o-**P**rocessor .
SMTP	**S**imple **M**ail **T**ransfert **P**rotocol.
SQL	**S**tructured **Q**uerry **L**anguage.
SSH	**S**ecure **S**hell.
XML	**E**xensible **M**arkup **L**anguage.

Introduction générale

Aujourd'hui la plupart des systèmes informatiques deviennent de plus en plus complexes et l'utilisation des protocoles de communication, engendrent des environnements incontrôlables et imprévisibles. Face à l'émergence massive des programmes malveillants et à la multiplication des attaques et leurs variantes sur Internet, les systèmes informatiques sont devenus plus vulnérables et plus exposés aux différentes sortes d'attaques. Ainsi, des outils de détection d'intrusions (IDS), souvent développés empiriquement, ont vu le jour.

Malgré, le rôle important qu'ils jouent face aux nouvelles menaces sur les systèmes informatiques, les IDS actuellement en opération produisent de nombreuses alertes. Les informations qu'elles contiennent manquent de précisions et sont, de plus parcellaire et de très bas niveau. Il est donc nécessaire de proposer de nouvelles techniques de modélisation des vulnérabilités et d'analyse des systèmes à protéger, dans le but de construire des scénarios d'attaques cherchant à exploiter ces vulnérabilités.

A l'heure où les technologies informatiques prennent de plus en plus de place dans notre vie quotidienne, il est devenu primordial de sécuriser ces données ou encore son réseau d'informations. C'est pour cela que nous devons songer à une méthode pour la génération de graphes d'attaques afin de prévenir les actions de l'intrus.

Notre projet de fin d'étude consiste, en la réalisation d'une application pour la génération de graphes d'attaques, pour différents réseaux informatiques, dont le but est d'améliorer la sécurité informatique dans différents réseaux d'entreprises.

Afin d'atteindre notre objectif, nous avons utilisé un algorithme pour la génération de scénarios d'attaques, avec en sortie un graphe d'attaques basé sur une logique et une stratégie de l'intrus, tout cela à l'aide d'une implémentation sous Java.

Ce mémoire comprend une introduction générale, trois chapitres, une conclusion et une bibliographie. Le premier chapitre comporte notre problématique, quelques définitions, par la suite nous introduisons quelques travaux relatifs. Nous terminons le chapitre avec une classification de ces travaux.

Le deuxième chapitre est consacré à quelques notions sur le type de graphes adopté, une description des différentes techniques suivies pour la génération de scénarios d'attaques,

une étude détaillée de la démarche étudiée, une définition des règles suivies. Pour finir avec la construction du graphe d'attaques.

Le troisième chapitre est consacré à la description des étapes suivies durant le développement de l'application. Pour cela nous avons identifié les outils de développement utilisés, une présentation du projet et des différentes fonctions de l'application, pour finir avec deux exemples explicatifs.

Notre travail s'achève par une conclusion et des perspectives.

1

Etat de l'art sur la génération de scénarios d'attaques

1.1 Introduction

Internet est devenu en très peu de temps un phénomène de société, voire de civilisation, abolissant les frontières entre les nations, permettant une libre circulation des informations mais aussi favorisant une libre action d'intrusion pour les malins du piratage informatique, exploitant les vulnérabilités des réseaux informatiques. Mais cette avancée fulgurante a certainement un prix, celui de la sécurité logicielle et matérielle. C'est d'ailleurs pour cela que nous nous sommes intéressés aux vulnérabilités existantes dans ces réseaux informatiques, et ce pour la génération de graphes d'attaques, dans le but de permettre une meilleure sécurité de ces réseaux.

Ce présent chapitre va faire l'objet d'une introduction aux notions d'attaques. Nous commençons par une problématique. Nous présentons ensuite des notions sur les graphes d'attaques et scénarios d'attaques. Enfin, nous terminons avec des travaux relatifs.

1.2 Problématique

Avec l'émergence massive à laquelle assiste la société moderne et ce dans différents domaines notamment celui de l'informatique. Avec ses vastes secteurs d'activités, il est devenu indispensable au monde moderne d'avoir un système d'informations propre à lui. C'est d'ailleurs pour cela qu'il est important de savoir sécuriser son réseau.

Avec l'apparition du piratage informatique et toutes les informations procurées sur Internet, sachant que les systèmes d'informations et les réseaux de télécommunications sont devenus indispensables au fonctionnement et à l'évolution de toutes les activités des entreprises. Il est devenu plus que primordial de trouver un moyen pour sécuriser ces réseaux.
Pour se faire les experts en sécurité informatique analysent les menaces causées par les pirates et ce dans le but de mettre en place des mécanismes rigoureux déductifs ou encore préventifs comme l'application de nouvelles techniques d'ingénieries lors de la conception et la réalisation des logiciels, et ce dans le but d'assurer une meilleure sécurité et de faciliter la correction des failles qui pourrais être découvertes.

D'autres mécanismes ont été conçus pour réagir aux risques d'attaques réseaux comme les pare-feux et les systèmes de détection ou de prévention d'intrusions. Mais ces derniers produisent de trop nombreuses fausses alertes.

Les approches déjà existantes pour la génération de scénarios d'attaques s'avèrent peu fiables, souvent basées sur des données qui reflètent la connaissance que l'on possède du système et/ou des attaques potentielles, mais n'expriment pas forcément une politique de sécurité.

Une alerte levée par l'outil de détection ne correspond, donc, pas systématiquement à une violation effective de la politique de sécurité mais seulement à la présence d'un symptôme, associé empiriquement au risque de violation de la politique, l'alerte peut ainsi correspondre à l'exploitation d'une vulnérabilité qui n'est pas présente sur le système attaqué.
Ainsi les alertes produites par le détecteur ne correspondent pas à des violations effectives de la politique (fausses alertes), et ne sont donc pas générées à partir d'une description d'un système informatique.

C'est pour cela qu'il est devenu indispensable de trouver un moyen de générer les différents scénarios d'attaques qu'un intrus puisse exploiter, et ce à l'aide des vulnérabilités présentes dans le système et d'une stratégie de l'intrus, afin de mieux sécuriser son réseau.

1.3 Définitions

1.3.1 Graphe d'attaques

Un graphe d'attaques est une abstraction mathématique des détails d'attaques possibles contre un réseau particulier, dont la cause de la faille n'est pas interne, en d'autres termes c'est une collection de scénarios d'attaques montrant comment un utilisateur malveillant peut compromettre l'intégrité d'un système cible. Où, chaque chemin dans le graphe d'attaque conduit à un état indésirable, comme dire qu'un intrus gagne l'accès d'un administrateur à un serveur de fichiers [1].

Il y a fondamentalement deux types de graphes d'attaques. Dans le premier type, chaque sommet représente l'état entier du réseau, et les arcs représentent des transitions d'états provoquées par les actions d'un attaquant. Ce type de graphes d'attaques s'appelle parfois un graphe d'attaques d'énumération d'états.
Dans le deuxième type de graphes d'attaques, un sommet ne représente pas l'état entier d'un système mais plutôt une condition du système sous forme d'une certaine expression logique. Les arcs dans ces graphes représentent les relations de causalité entre les conditions du système. Ce type de graphes d'attaques est appelé un graphe d'attaques de dépendances [2].

1.3.2 Scénario d'attaques

Les nœuds et les arcs du graphe d'attaques représentent des scénarios que l'attaquant prend et change selon l'état des réseaux. Un scénario d'attaque est un ensemble de privilèges acquis par l'exploitation des vulnérabilités dont le dernier privilège est un objectif de l'intrus. Les scénarios d'attaques impliquent généralement un exploit ou des étapes d'exploitation qui tirent profit des vulnérabilités des logiciels et des protocoles. L'objectif de ces scénarios est pour l'attaquant d'obtenir des privilèges sur un ou plusieurs hôtes cibles, où la cible pourrait être l'ordinateur d'un utilisateur, un routeur, un pare feu, ou un autre composant réseaux [3].

1.4 Exemple intuitif

Nous avons machine 0, machine 1, machine 2, qui représentent respectivement, l'utilisateur, le serveur web et le serveur de base de données. Le pare feu permet des requêtes http et ssh de la machine 0 vers la machine 1. Au cours du fonctionnement normal, l'utilisateur effectue une requête http au serveur 1, qui passe par le pare feu. Le serveur 1 accède au serveur de base de données pour les données de correspondance,

et puis, communiquent avec la machine 0 avec http [1]. Si l'utilisateur tente d'accéder à la machine 2 directement, le pare-feu bloque la communication. Une demande de ssh [2] de la machine 0 à la machine 2 est considérée comme un comportement anormal qui est bloqué par le pare-feu. En outre, la base de données sur le serveur 2 aurait des données privées des utilisateurs autres que celui de machine 0. Sauf si une attaque par injection de commande [3] est lancée avec succès sur le serveur 1 pour le compromettre. Ensuite, à l'aide d'un invité de commande compromit, une attaque par injection SQL [4] est lancée sur la base de données à la machine 2. La donnée est siphonnée au serveur 1 puis la machine 0. Ce scénario est représenté avec les deux figures 1.1 et 1.2 [4].

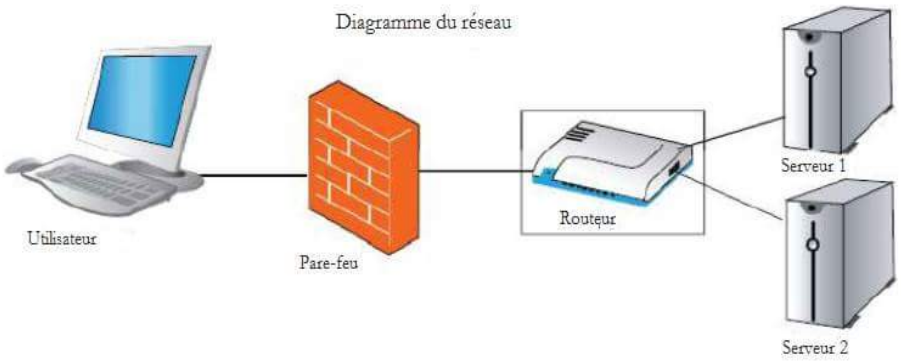

FIGURE 1.1 – Exemple d'un réseau [4].

1. *Mode d'adressage qui utilise le protocole TCP /IP.*
2. *Protocole permettant à un client d'envoyer des commandes ou fichiers de manière sécurisée.*
3. *Groupe de méthodes d'exploitation de faille de sécurité d'une application interagissant avec une base de données.*
4. *Attaque utilisant les méthodes d'exploitation d'une injection SQL(langage de base de données).*

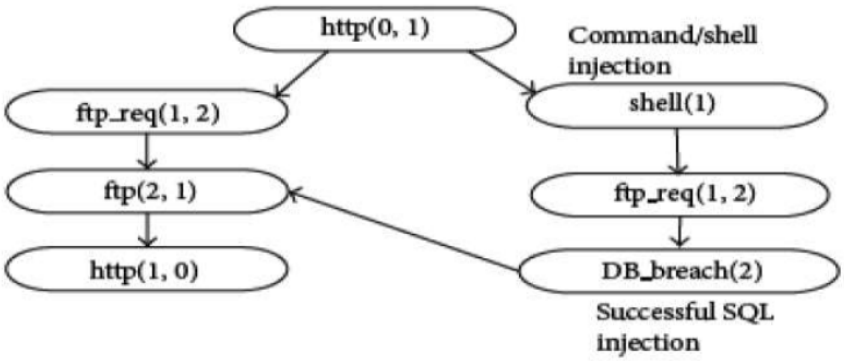

FIGURE 1.2 – Exemple de graphe d'attaques [4].

1.5 Travaux relatifs

Plusieurs travaux ont été réaliser dans ce domaine. Dans ce qui suit nous présentons quelques travaux dédiés à la génération de scénarios d'attaques.

- **Fondement théorique**

  Idika [6] a proposé la caractérisation des paramètres de sécurité basés sur des graphes d'attaques pour améliorer le durcissement des contraintes budgétaires fixées. En conséquence, les ressources limitées attribuées pour la sécurité doivent être déployées de façon appropriée pour localiser et corriger les vulnérabilités et éviter la violation de hautes propriétés de priorité.

  La modélisation a besoin de faciliter la représentation naturelle d'une exécution infinie de la plupart des systèmes tels que les systèmes d'exploitation et les serveurs opérant sur Internet s'exécutant continûment. Dans de tels systèmes, une exploitation réussie des vulnérabilités résultantes d'une violation de propriétés peut mener à un état indésirable. Ainsi, une attaque réussie conduira le système à une succession d'états indésirables. Ceci est exprimé logiquement en utilisant la logique temporelle linéaire (LTL).

- **Motivation pratique**

  Il y a principalement deux objectifs pratiques pour la modélisation. Le premier, est la gestion efficace des entrées et sorties du graphe d'attaque. Les entrés étant les paramètres du système et doivent être présentés dans le modèle résultant du graphe. Puis, l'analyse du modèle pour les différentes propriétés de sécurité et la

détection de violations, devraient trouver des réponses efficaces en sortie. Le second, est la capacité de générer ces graphes d'une manière autonome, efficace et avec comptabilité pour les systèmes complexes. Il y a eu de nombreuses tentatives pour atteindre ces objectifs aux divers degrés de succès.

1.5.1 Plan de la classification

Cette section présente une classification détaillée de quelques travaux mis en œuvre pendant la décennie passée. Diverses recherches sont centrées sur le développement de nouvelles technologies tandis que d'autres mettent l'accent sur de nouvelles méthodes.

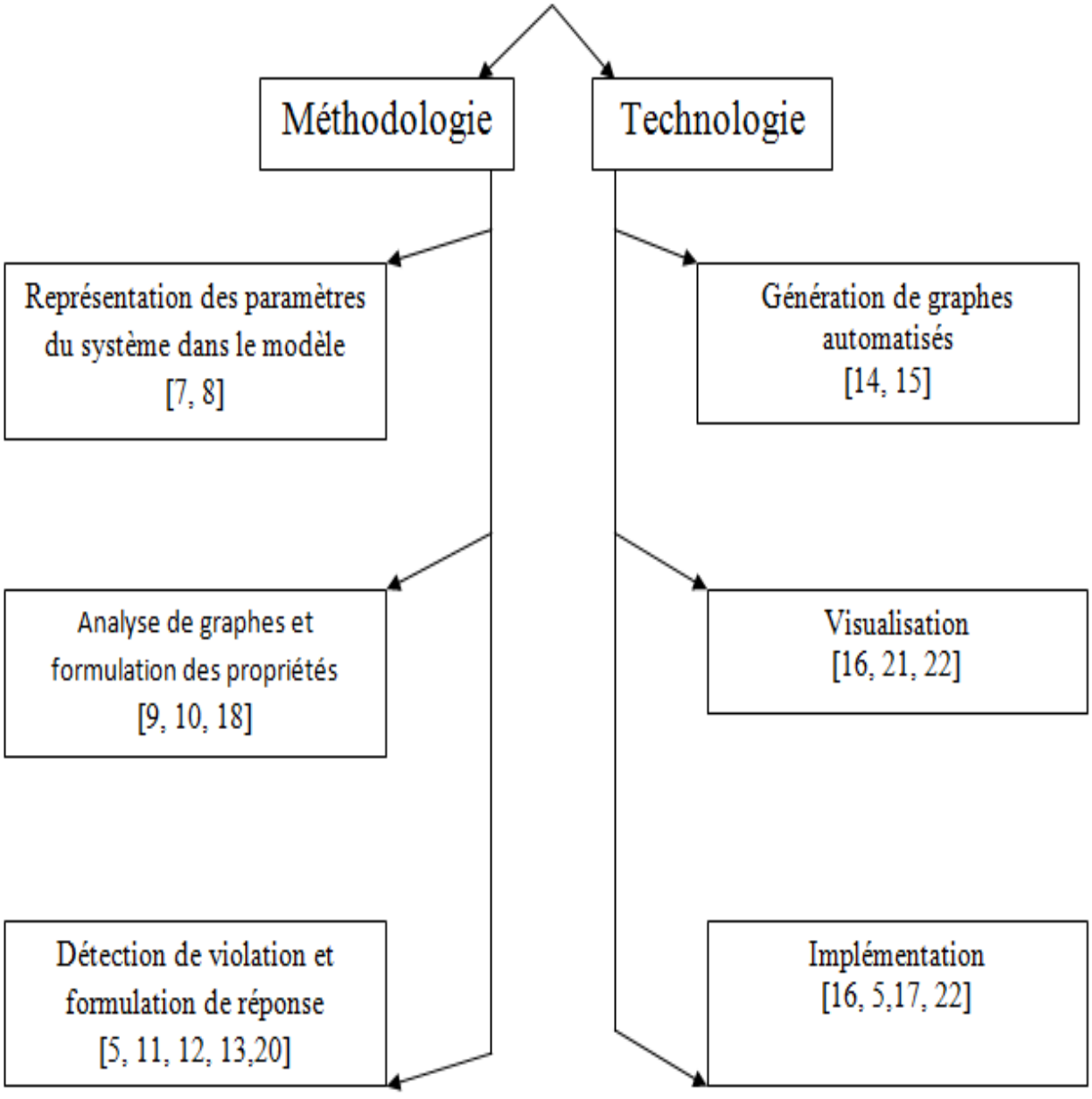

FIGURE 1.3 – Classification des travaux étudiés.

1.5.2 Méthodologie

Dans ce qui suit nous présentons les travaux basés sur la méthodologie.

1.5.2.1 Représentation des paramètres du système dans le modèle

Hong et Kim [5] proposent un modèle de graphe à deux couches pour capter les informations de vulnérabilités du réseau, et fournir une étude comparative de la complexité de l'analyse de la sécurité de leur modèle, avec celle des graphes d'attaques classiques. Ils prétendent que leur modèle est plus favorable pour la génération distribuée et l'analyse et est donc plus efficace. Ils construisent des graphes à deux couches (deux sous-graphes).

La couche inférieure contenant des informations sur la vulnérabilité de l'hôte en recueillant tous les différents processus (potentiels) d'intérêt de sécurité, elle dispose d'une structure d'arbre dirigé conduisant à l'actif le plus valorisé avec la vulnérabilité du premier objectif de l'attaquant. Cette couche aura un nœud racine dans la couche supérieure qui représente un hôte unique. La couche supérieure contenant l'information topologique du réseau, cette dernière peut être non acyclique en fonction de la structure du réseau. Les arêtes dans le graphe de cette couche représentent un réseau d'arêtes entre les hôtes.
Individuellement les deux graphes obtenus des deux couches constituent le graphe complet.

Ce travail théorique est sans mise en œuvre. La principale contribution est de recueillir des paramètres topologiques du réseau dans une couche du graphe et des informations individuelles de vulnérabilité de l'hôte dans une autre, aidant ainsi l'analyse pratique.

1.5.2.2 Analyse graphique et sécurité des propriétés de formulation

Kotenko et Stephashkin [17] ont utilisés les graphes d'attaques et des mesures de sécurité afin d'évaluer le facteur de sécurité des actions de malfaiteurs. Tous les objets de graphes d'attaques sont généralement divisés en deux groupes, l'un de base (élémentaire) et l'autre des objets combinés. Le premier étant l'hôte et actions d'attaque et le second des objets de type itinéraire, menace et graphe. L'évaluation du niveau de sécurité est mesurée sous deux formes, méthodologies qualitatives d'évaluation des risques et le calcul quantitatif de niveau de sécurité du réseau (sur la base de réseaux bayésiens, la théorie des possibilités, et ensembles flous).

1.5.2.3 Détection de violation et formulation de réponse

Dewri et al. [18] présentent une méthodologie pour le renforcement de la sécurité des systèmes de réseau basé sur des arbres d'attaques. Ils formulent le problème comme étant un problème d'optimisation à objectifs multiples qui consiste à associer les efforts

de renforcement avec leurs coûts et en optimisant le choix des vulnérabilités à fixer pour la sécurité optimale dans un budget donné. Ensuite utiliser un algorithme génétique de tri 1-1 non dominée pour le résoudre. Ils présentent les attributs comme une instance propositionnelle d'un attribut-modèle. Ils construisent l'arbre d'attaque à partir des attributs et les conditions formulées sur la base de l'objectif final de l'attaquant. La contribution principale est de proposer la méthodologie pour sélectionner les mesures de sécurité et minimiser les dommages résiduels dans un budget donné.
Les paramètres du système, tels que les propriétés génériques de la configuration matérielle, les logicielles du réseau, les vulnérabilités du système, la configuration du réseau et du système, les privilèges d'accès, ainsi que la connectivité sont utilisées pour définir et créer une instance propositionnelle afin d'obtenir l'attribut - templates [5].
En utilisant les attributs, l'objectif final de l'attaquant, et les conditions d'attributs détenues pour satisfaire un lancement réussi d'attaques sont utilisées pour fournir un simple exemple avec 4 - nœuds pour illustrer l'efficacité de leur méthode avec un serveur FTP, un serveur SMTP, un Terminal et un serveur de données.

Dans ce travail, l'idée est qu'une fois une violation se produit, ses effets restent pour toujours, afin d'éviter l'explosion combinatoire tout en construisant l'arbre d'attaques. Cependant, ils n'ont proposé aucune technologie (outil spécifique) pour générer l'arbre d'attaques. Les auteurs de ce travail ne précisent pas de méthodes de visualisation particulières (technologies des arbres d'attaques). Néanmoins, ils sont en mesure d'obtenir l'ensemble rentable, des mesures de sécurité visant à réduire au minimum les dommages résiduels.

Wang et al. [19] présentent une métrique de sécurité basée sur les graphes d'attaques. Ils présentent la motivation et l'interprétation de la métrique via l'utilisation de trois facteurs. Le premier facteur est la vulnérabilité individuelle à chaque nœud déterminé par les conditions d'ordinateurs hôtes qui constituent le réseau. Le deuxième facteur est l'état d'activité de l'attaquant. Le troisième facteur est le lien de causalité entre l'exploitation des vulnérabilités individuelles. Ils considèrent à la fois les graphes d'attaques cycliques et acycliques. La principale contribution est un algorithme de recherche en $largeurd'abord$. L'algorithme est recommandé pour le calcul de la métrique de sécurité en utilisant les probabilités des vulnérabilités et des attaques.

1.5.3 Technologie

Dans ce qui suit nous présentons les travaux basés sur les nouvelles technologies.

5. *Attribut modèle désignant un exemple sur lequel on se base pour concevoir un logiciel, un désigne déjà fait, etc.*

1.5.3.1 Génération de graphes automatisés

Ou et al. [20] présentent un outil de génération de graphes d'attaques automatisés testé sur les réseaux avec des milliers de nœuds. Pour ce faire, ils adaptent l'outil MulVal, un analyseur de sécurité de réseau basée sur la programmation logique en utilisant un système Prolog [6] qui évalue les règles d'interactions Datalog [7] sur le système donné. Leur méthode utilise les dépendances logiques entre le but de l'attaquant et les informations de configuration. La principale contribution est d'éviter le problème de l'explosion combinatoire du procédé utilisant le formalisme de norme et fournir un graphe d'attaques à évolution automatisée (graphe d'attaques logique) outil de génération.
Les détails de la configuration du réseau et du but attendue de l'attaquant avec le coût/bénéfice connexes sont utilisés pour construire les graphes d'attaques logiques. Cependant, la spécification de pré et post conditions pour l'attaque formule les attaques.

Les graphes générés sont appelés graphes d'attaques logiques avec deux types de nœuds appelés dérivation et fait des nœuds.
Les propriétés ne sont pas explicitement analysées depuis le graphe. Mais peuvent être exprimées par la logique du graphe d'attaques en termes de configurations système et des objectifs d'attaques connus.

1.5.3.2 Visualisation

Xie et al. [21] ont proposés un graphe d'attaques à deux couches pour contrecarrer les attaquants malveillants, à partir de l'attaque réseau de l'intérieur, la couche supérieure est l'accès au graphe de l'hôte et la couche inférieure est le graphe d'attaques ou graphe de la paire d'hôtes.

Williams et al. [30] présentent un moteur de calcul efficace qui génère des graphes d'attaques étape par étape et fournit une occasion interactive pour tracer le chemin de l'attaquant. Le module de calcul est écrit en C ++ pour des raisons de vitesse tandis que le module de visualisation est mis en œuvre en Java.

1.5.3.3 Implémentation

Huang et al [23] présentent une procédure pour distiller à partir d'un graphe d'attaques complet du réseau un petit graphe d'attaques critique. Cette approche est proposée de sorte que l'utilisateur peut contrôler la quantité d'informations présentées par filtrage sur la partie la plus critique du graphe d'attaques complet. Le travail ce fait en transformant un graphe d'attaques dans une formule booléenne et attribuant des métriques de couts

6. *Langage de programmation logique.*
7. *Règles de programmation logique*

pour attaquer les variables dans la formule sur la base des indicateurs de gravitée.

1.6 Conslusion

Avec l'apparition de plus en plus de nouvelles technologies informatiques dans notre vie quotidienne, il est devenu primordial de sécuriser ces données ou encore son réseau d'informations des différentes attaques. Pour ce faire, nous avons exploité les vulnérabilités des composantes de réseaux et ce dans le but de générer les graphes d'attaques correspondant à ces réseaux.

Dans ce chapitre, nous avons donné un aperçu sur les graphes d'attaques, ainsi que les différents travaux ayant permis leurs évolutions dans le monde de la sécurité informatique. Dans le prochain chapitre nous allons aborder la démarche étudiée pour la génération du graphe d'attaque associé à chaque réseau.

2
Démarche étudiée

2.1 Introduction

Malgré, le rôle important que les outils de détection d'intrusions jouent face aux nouvelles menaces et aux vulnérabilités actuelles et potentielles des systèmes informatiques. Avec l'avancement des technologies de l'information et face à l'émergence massive des programmes malveillants les IDS actuellement en opération produisent de nombreuses alertes. Les informations qu'ils contiennent sont parcellaires, de très bas niveau, et manquent de précision. Il est devenu nécessaire de proposer de nouvelles techniques de modélisation des vulnérabilités et d'analyse des systèmes à protéger dans le but de construire des scénarios d'attaques cherchant à exploiter ces vulnérabilités.

Le but de ce chapitre est de présenter une approche proposée par L.Hamza et al. [25]. Sachant qu'une étude à déjà été faite par les étudiants S. Ould amara et M. Lounis [24], n'ayant pas aboutis au résultat voulu, vu qu'il n'ont pas exploiter la base de données des vulnérabilités, nous avons réitérer ce projet.

Ce présent chapitre comprend trois parties. La première partie est consacrée aux techniques formelles de génération de scénarios d'attaques, à savoir, la vérification formelle, le model checking, la preuve formelle et les logiques temporelles. La seconde partie quand à elle est consacrée à une description de la démarche étudiée, du model d'attaque et spécification des règles pour finir avec une troisième partie consacrée à la construction du graphe d'attaques avec une stratégie.

2.2 Graphes des privilèges

La notion de graphes des privilèges a été proposée par Dacier [32] en 1994. Dans ce graphe, les sommets représentent des ensembles de droits (privilèges) qu'un sujet peut posséder sur un objet, et les arcs des transferts de privilèges : Il existe un arc (étiqueté M) allant de X à Y s'il est possible, ayant les privilèges représentés par le sommet X, d'acquérir les privilèges du sommet Y, en utilisant la méthode M. Cette méthode peut être un transfert licite de privilèges (l'utilisateur Y fait confiance à X), ou un transfert implicite (Y est un sous ensemble de X), ou encore une attaque élémentaire. Un poids peut être affecté aux différentes méthodes, selon la difficulté pour un possible attaquant d'exploiter la méthode ou selon le temps nécessaire pour réaliser l'attaque. Les graphes de privilèges peuvent être exploités pour construire des graphes d'attaques.

Avec :
Pour un arc $X \rightarrow Y$

1 X peut devenir le mot de passe de Y.

2 X peut être un cheval de troie pour Y.

3 X peut exploiter une faille du mailler de Y.

4 Y est un sous ensemble de X.

5 Y utilise un programme que X peut modifier.

6 X peut modofier un programme "s-uid" de Y.

7 X est le ".rhost" de Y.

Dans cet exemple, le nœud *Min* représente les privilèges minimaux de n'importe quel utilisateur autorisé du système. Les sommets B, C et F représentent les privilèges de trois utilisateurs particuliers, alors que *Adm* représente les privilèges du groupe des administrateurs du système (dont F est membre) et $P1$ est un utilisateur virtuel correspondant à un projet. Dans cet exemple, les méthodes 1, 2 et 3 correspondent à des négligences des utilisateurs $P1$, F et C ; alors que les méthodes 4 à 7 correspondent à des transferts de privilèges volontaires (justifiés par la confiance des utilisateurs entre eux).
Un graphe d'attaques est une abstraction mathématique des détails d'attaques possibles contre un réseau particulier, en d'autres termes c'est une collection de scénarios d'attaques montrant comment un utilisateur malveillant peut compromettre l'intégrité d'un système cible. Il y a fondamentalement deux types de graphes d'attaques. Dans le premier type, chaque sommet représente l'état entier du réseau et les arcs représentent des transitions d'états provoquées par les actions d'un attaquant. Ce type de graphe d'attaques s'appelle parfois *un graphe d'attaques d'énumération d'états.*

Dans le deuxième type de graphe d'attaques, un sommet ne représente pas l'état entier d'un système mais plutôt une condition du système sous la forme d'une certaine expression logique. Les arcs de ces graphes représentent les relations de causalité entre les conditions du système. Ce type de graphe d'attaques est appelé *graphe d'attaques de dépendances* [33].

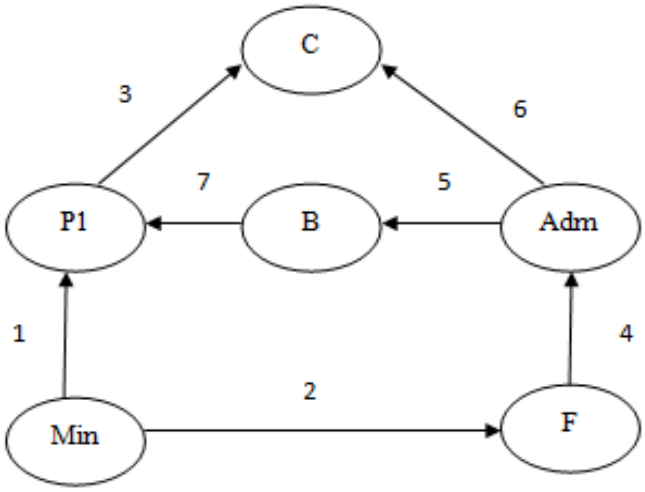

FIGURE 2.1 – Exemple de graphe de privilèges [32].

2.3 Techniques formelles de génération de scénarios d'attaques

Les méthodes formelles permettent d'obtenir une très forte assurance de l'absence de bug dans les logiciels, c'est-à-dire d'acquérir des niveaux d'évaluations d'assurances élevés, elles sont basées sur des descriptions mathématiques formelles. Cependant, elles sont généralement coûteuses en ressources (humaines et matérielles) et actuellement réservées aux logiciels les plus critiques. Leurs améliorations et l'élargissement de leurs champs d'applications sont la motivation de nombreuses recherches.

2.3.1 Vérification formelle

La vérification formelle, consiste à s'assurer que certaines propriétés souhaitées sont bien respectées sur un système donné. Cette vérification se fait par la confrontation d'une représentation concise et non ambigüe des propriétés et d'un modèle mathématique, basé sur un langage formel, représentant le système [34].

2.3.2 Approches de la vérification formelle

Il existe deux principales approches de la vérification formelle : la preuve formelle et le Model Checking.

a. Model Checking

Le Model Checking est une technique automatique pour vérifier des systèmes à états finis. Les spécifications du système sont exprimées en propositions de logique temporelle et le système est modélisé en un graphe d'états transition. Une procédure de recherche permet alors de déterminer si les spécifications sont satisfaites par le graphe d'états transition. Les dernières évolutions du Model Checking permettent de traiter un grand nombre d'états grâce à une représentation binaire efficace des transitions d'états [35].

b. Preuve formelle

Cette approche est introduite par Hoare [36], elle est basée sur une preuve mathématique. Elle consiste à décrire le système et ses propriétés dans un modèle sémantique axiomatique et à démontrer que les propriétés peuvent être obtenues à partir du système en utilisant des règles de déduction. Ces propriétés sont exprimées en logique temporelle, cette dernière utilise les propositions et les connecteurs de la logique propositionnelle auxquels elle intègre de nouveaux opérateurs exprimant le temps. Plusieurs classifications ont été proposées pour les logiques temporelles, nous avons choisi de présenter les plus répandues à savoir :

- La logique temporelle linéaire (LTL).
- La logique temporelle arborescente (CTL).

2.3.2.1 Logique temporelle linéaire :

La logique temporelle linéaire LTL permet de représenter le comportement des systèmes réactifs au moyen des propriétés qui décrivent le système pour lesquels le temps se déroule linéairement. En clair, on spécifie le comportement attendu du système, en spécifiant l'unique futur possible.

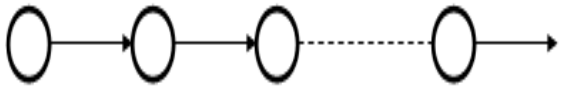

FIGURE 2.2 – Déroulement temporel d'un système dans une logique linéaire.

Syntaxe :

Les formules de la logique temporelle linéaire sont définies par la grammaire suivante : Soient Φ, $\Psi \in LTL$, les formules :

- Les propositions atomiques :
  $\Phi, \Psi ::= p \mid q \mid ... \mid true \mid false$
- Les connecteurs booléens :
  $\Phi, \Psi ::= \Phi \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid \Phi \Rightarrow \Psi \mid \Phi \Leftrightarrow \Psi$
- Les connecteurs temporels : ils permettent de parler du séquencement des états/évènements observés le long d'une exécution. LTL utilise les connecteurs suivants : $X(next)$, $F(Eventuelly)$, $G(Always)$, $U(until)$.
  $\Phi, \Psi ::= G\Phi \mid F\Phi \mid \Phi U \Psi \mid X\Phi$

Sémantique :

Les opérateurs temporels sont définis dans le tableau suivant [37] :

Opérateurs temporels	**Signification**
X $(Next)$	$X\Phi$ veut dire que Φ est vérifiée dans le prochain état.
F $(Eventually)$	$F\Phi$ veut dire qu'il existe un état pour lequel Φ est vraie.
$G(Always)$	$G\Phi$ veut dire que Φ est toujours vérifié dans le futur.
$U(Until)$	$\Phi U \Psi$ veut dire que Φ est vérifiée jusqu'à ce que Ψ le soit.

TABLE 2.1 – Sémantique LTL

2.3.2.2 Logique temporelle arborescente :

La logique du temps arborescent ou CTL permet de considérer plusieurs futurs possibles à partir d'un état du système plutôt que d'avoir une vue linéaire du système considéré. Elles permettent d'exprimer des propriétés portant sur les arbres d'exécution (issus de l'état initial) du programme.

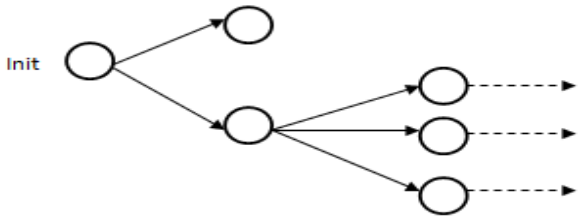

FIGURE 2.3 – Déroulement temporel d'un système dans une logique arborescente.

Syntaxe :

Les formules de la logique temporelle arborescente sont définies par la grammaire suivante :
Soient Φ, $\Psi \in CTL$, les formules :

- Les propositions atomiques :
  $\Phi, \Psi ::= p \mid q \mid ... \mid true \mid false$
- Les connecteurs booléens :
  $\Phi, \Psi ::= \Phi \mid \Phi \wedge \Psi \mid \Phi \vee \Psi \mid \Phi \Rightarrow \Psi \mid \Phi \Leftrightarrow \Psi$
- Les connecteurs temporels :
  $\Phi, \Psi ::= EF\Phi \mid FG\Phi \mid E\Phi U\Psi \mid EX\Phi \mid AF\Phi \mid AG\Phi \mid AX\Phi \mid A\Phi U\Psi$

Sémantique :

Les opérateurs temporels sont définis dans le tableau suivant [37] :

Les opérateurs temporels	**Signification**
$EF\Phi$	Il est possible d'atteindre un état ou il existe une exécution (opérateur E) conduisant à un état ou Φ est vérifiée (opérateur F).
$AF\Phi$	Φ est vérifiée dans le futur ou pour toute exécution (opérateur A), il existe un état ou Φ est vérifiée (opérateur F).
$AG\Phi$	Φ est vérifiée pour tout état atteignable ou pour toute exécution (opérateur A), Φ est toujours vérifiée (opérateur G).
$EG\Phi$	Il existe une exécution (opérateur E) ou Φ est toujours vérifiée (opérateur G).
$E\Phi U\Phi$	Il existe une exécution (opérateur E) ou Φ est vérifiée jusqu'à ce que Ψ le soit.
$A\Phi U\Psi$	Φ est vérifiée pour toute exécution (opérateur A) jusqu'à ce que Ψ le soit.
$AX\Phi$	Tous les états immédiatement successeurs satisferont Φ.
$EX\Phi$	Il existe une exécution (opérateur E) dont le prochain état satisfait Φ.

TABLE 2.2 – Sémantique CTL

2.4 Détaille de l'approche étdudiée

L'approche étudiée [25, 26, 27] se base sur le graphe de privilèges et une politique de sécurité pour générer le graphe d'attaques. Elle permet de construire des scénarios d'attaques en utilisant une preuve formelle.

Dans cette approche, un nœud du graphe d'attaques représente un privilège acquis par l'exploitation d'une vulnérabilité et les arcs représentent la relation de dépendance existante entre ces différents privilèges. Ainsi, une attaque est définie comme étant l'acquisition d'un privilège, en exploitant une vulnérabilité, et un scénario d'attaques est l'acquisition d'un ensemble de privilèges, dont le dernier est l'objectif de l'intrus, sachant qu'un scénario d'attaques ne se termine que si cet objectif est atteint.

Les principales étapes 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.

2.4.1 Modèle d'attaques

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.

1 Hôtes (H)

D'après Sheyner [29], un ordinateur h ∈ H est décrit par le tuple (*id, svcs, sw, vuls*), où ;

- ***id***, est l'identificateur unique de l'ordinateur (nom de la machine, adresse réseau),
- ***svcs***, la liste de services (nom du service, port d'écoute),
- ***sw***, la liste des logiciels actifs (en opération),
- ***vuls***, la liste des composantes vulnérables [1].

2 Relation de connectivité (C)

D'après Ritchy et Amman [28], la connexion est exprimée par la relation $C \subseteq H \times H \times P$ où P est l'ensemble des numéros de port. La relation C (h_1, h_2, p) signifie que l'ordinateur h_2 est accessible à partir de h_1 sur le port p, sachant que la relation de connectivité incorpore les règles de filtrage des firewalls qui restreignent l'activité d'un ordinateur sur un autre.

3 Action de l'intrus (A)

Les actions d'un intrus sont basées sur l'exploitation d'une vulnérabilité existante d'un élément de l'ensemble d'hôtes H. Après chaque exploitation d'une vulnérabilité existante, l'intrus acquière quelques privilèges.

1. *Il existe plusieurs analyseurs de vulnérabilités (COPS, Renaud deraison's, ect).*

Tel que :

Priv(v_x(h)) : est le privilège de l'intrus sur chaque hôte h en exploitant la vulnérabilité v_x [25].

2.4.2 Spécification des règles de déductions

Le schéma ci-après représente le modèle du processus d'attaques défini par Gad et al. [31].

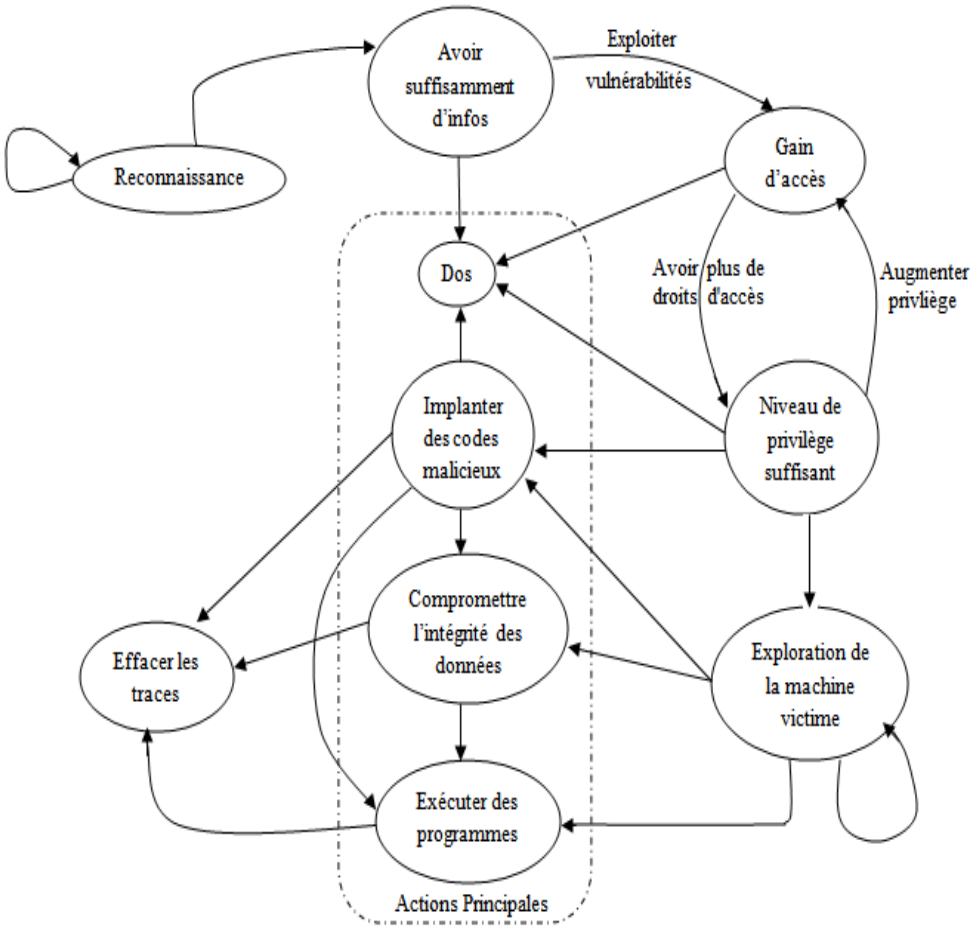

FIGURE 2.4 – Modèle de processus d'attaque [31].

Afin de générer des scénarios d'attaques variés correspondants aux différents profils d'attaques, l'article [25] propose trois règles d'intrusions qui définissent les conditions initiales, l'élévation de privilèges et la règle d'oracle.

$priv_{init}$:	Privilège initial
$priv$:	Séquence d'attaques
Φ :	Statégie de l'intrus
Règle 1 :	$\frac{\square}{priv{=}priv_{init}}$
Règle 2 :	$\frac{priv^{+} \supseteq cond(v_x(h)) \wedge priv.priv(v_x(h)) \vDash \Phi}{priv{=}priv.priv(v_x(h))}$
Règle 3 :	$\frac{(p_1.priv(v_x(h)).p_2)^{+} \supseteq cond(v_x(h)) \wedge p_1.priv(v_x(h)) \vDash \Phi}{priv{=}p_1.priv(v_x(h)).p_2}$

TABLE 2.3 – Règles de l'intrus

Avec :

- $priv.priv(v_x(h))$: Exprime la séquence.
- $v_x(h)$: Exprime la vulnérabilité x de la machine h.
- $cond(v_x(h))$: Condition pour l'exploitation de la vulnérabilité x de la machine h.

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

Règle 2 (Elévation des privilèges) : Signifie que l'intrus ne peut passer d'un état vers un autre que si l'ensemble des privilèges acquis par ce dernier dans un premier lieu le mène à acquérir un autre ensemble de privilèges 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 ne lui convient pas et que l'autre chemin choisi est plus court et meilleur d'après sa stratégie.

2.5 Construction du graphe d'attaques

Avant de créer un graphe d'attaques pour un système d'ordinateurs, la politique de sécurité du système doit être définie dans un langage formel. C'est pourquoi, l'application d'une logique adéquate suffisamment expressive est nécessaire pour générer des scénarios d'attaques.
Dans notre cas, cette propriété est exprimée au moyen d'une logique temporelle linéaire LTL. Afin de générer des scénarios d'attaques variés qui correspondent aux différents profils d'attaques, dans l'article [25] trois règles d'intrus sont proposés pour définir les stratégies de reconnaissance, d'acquisition d'accès et d'escalade de privilèges.
Durant la phase de reconnaissance, l'intrus tente d'avoir les informations nécessaires sur la machine cible ainsi que sur le réseau pour trouver le moyen d'atteindre son objectif.

Dans la phase d'acquisition d'accès, l'intrus agit et gagne un ou plusieurs privilèges lui permettant d'avoir plus d'informations afin de pouvoir progresser dans son processus d'attaque ; c'est ce qui est définit par la dernière règle qui est l'escalade de privilèges.

L'exécution de la stratégie de l'intrus, organisée dans un scénario d'intrusion permet de changer l'état du système d'un état initial saint, où la politique de sécurité est respectée vers un état final où l'objectif de l'intrusion est accomplie et la politique de sécurité est violée.
Il est possible d'admettre que les conditions d'exploitation de la première vulnérabilité soient satisfaites tandis qu'une condition peut être satisfaite par n'importe quelle attaque, ainsi la construction d'attaques complexes est effectuée graduellement par l'application de règles de déduction d'intrus [25].

2.5.1 Algorithme de génération

Contrairement à l'algorithme proposé par S. Ould amara et M. Lounis dans [24], cet algorithme inclut la logique LTL.

Algorithm 1 Algorithme de construction du graphe d'attaques.

Variables :
  $GrapheAttaque$: liste d'adjacences initialisée à un seul nœud.
  $ListVul$, $ListNœudCourant$, $ListNouveauNœud$: listes.
Début /* Règle 1 */
  $ListNœudCourant \leftarrow GrapheAttaque$;
  Tant que($ListVul \neq vide$) faire
    Etape 1 : /* Règle 2 */
    Pour(i allant de 1 à $ListNœudCourant.taille()$)faire
      Pour(j allant de 1 à $ListVul.taille()$)faire
        Si(($ListVul.Elt(j).Cond \subset ListNœudCourant.Elt(i).Priv$) et
            $LTL(ListNœudCourant.Elt(i), logic)$)alors
          Créer un nouveau nœud k ayant le nœud i comme père
          Ajouter ce dernier à la liste $ListNœudCourant$ et Supprimer la vulnérabilité j
          Si$LTL(k, logic)$)alors
            Ajouter ce nœud à la liste $ListNouveauNœud$
          Finsi ;
        Finsi ;
      Finpour ;
    Finpour ;
    Etape 2 :
    Pour (i allant de 1 à $ListNouveauNœud.taille()$)faire
      Pour (j allant de 1 à $ListNouveauNœud.taille()$)faire
        Si(($ListNouveauNœud.Elt(i) = ListNouveauNœud.Elt(j))et(i \neq j)$)alors
          Ajouter le nœud i aux successeurs du père du nœud j
          Supprimer le nœud j de la liste $ListNouveauNœud$
        Finsi ;
      Finpour ;
    Finpour ;
    /* Règle 3 */
    Pour (i allant de 1 à $ListNouveauNœud.taille()$)faire
      Pour (j allant de 1 à $GrapheAttaque.taille()$)faire
        Si($ListNouveauNœud.Elt(i) = GrapheAttaque.Elt(j)$)alors
          Ajouter le nœud j du graphe au successeus du père du nouveau nœud i
          Supprimer le nœud j de la liste $ListNouveauNœud$
        Finsi ;
      Finpour ;
    Finpour ;
    Etape 3 :
    Pour (i allant de 1 à $ListNouveauNœud.taille()$)faire
      Ajouter le nœud i aux successeurs de son père
    Finpour ;
    Ajouter $ListNouveauNœud$ à $GrapheAttaque$
    $ListNœudCourant \leftarrow ListNouveauNœud$
    $ListNouveauNœud \leftarrow null$
  FinTantque ;
Fin ;

Avec LTL, une fonction récursive permettant d'injecter une logique temporelle linéaire à l'implémentation pour la génération du graphe d'attaque.

Description de l'algorithme

L'algorithme proposé permet la construction du graphe d'attaques, les nœuds de ce dernier sont des attaques et chaque attaque est définie comme étant l'exploitation d'une vulnérabilité, fournie à partir de propriétés, pour acquérir un ensemble de privilèges. Le graphe d'attaques est représenté sous forme d'une liste d'adjacences, [2] contenant initialement un seul nœud.
La première étape consiste à parcourir la liste des vulnérabilités $ListVul$, contenant les vulnérabilités du système, et de chercher un nœud parmi ceux de la liste $ListNœudCourant$, contenant les nœuds en cours de traitement, qui inclut l'ensemble des pré-conditions, $Cond$, de la vulnérabilité dans son ensemble de privilèges, si un tel nœud est trouvé, et suit la stratégie adoptée, un nouveau nœud est créé et ajouté à la liste $ListNouveauNœud$, qui contient les nœuds construits à chaque étape, et la vulnérabilité en question est supprimée.
Dans la deuxième étape, nous supprimons les doubles dans la liste $ListNouveauNœud$ et dans le graphe. Enfin dans la troisième étape permet d'ajouter les arcs entre les nœuds et ainsi, de construire le graphe.

Calcul de la complexité de l'algorithme

Soit n : le nombre de vulnérabilités du réseau,
m : la taille de la liste $ListNœudCourant$,
l : la taille de la liste $ListNouveauNœud$,
tel que $m < n$, $l < n$.

La complexité au pire des cas de l'étape 1 est : $m \times n \simeq n^3$.
La complexité au pire des cas de l'étape 2 est : $l^2 + n \times l \simeq 2n^2$.
La complexité au pire des cas de l'étape 3 est : $l \simeq n$.
La complexité des trois étapes est donc : $n^3 + 2n^2$+n.
La complexité de l'algorithme au pire des cas est : $n^3 + 2n^2 + n \simeq O(n^3)$, donc c'est une complexité polynomiale.

2. *C'est une structure de données utilisées pour représenter un graphe en mémoire.*

2.6 Conclusion

Dans ce chapitre nous avons fait une analyse des techniques formelles pour la génération de scénarios d'attaques suivie de logiques temporelles pour finir avec une étude détaillée du model d'attaques suivie et de l'algorithme implémenté utilisant une technique existante, avec une analyse des vulnérabilités qui sont des étapes très importantes pour la réalisation de notre projet. Ce qui nous a bien préparés à entamer la bonne réalisation de notre application, qui va être éclaircie dans le chapitre suivant.

3
Implémentation

3.1 Introduction

Ce chapitre est composé de deux grandes parties. La première partie théorique et consacrée à la présentation du projet, et une description des outils de développement suivis, où on décrit les objectifs de l'application ainsi que les langages utilisés durant le développement, et la seconde, pratique et consacrée à la partie graphique de l'application suivie de deux exemples explicatifs pour finir avec une évaluation des performances.

3.2 Présentation du projet

3.2.1 Contexte

Une application graphique sera réalisée afin de solutionner notre problématique posée antérieurement. Permettant de générer des graphes d'attaques, cette application met en œuvre tous les scénarios possibles qu'un intrus peut suivre dans le but d'atteindre une cible fixée au début de la génération.

La spécification principale est de déterminer le niveau de fiabilité dans un réseau, de telle façon qu'on pourrait déterminer les vulnérabilités existantes dans ce dernier. Cette spécification peut servir à confirmer la cohérence d'une politique de sécurité réseau, ou même de trouver comment un intrus pourrait exploiter ou contourner l'implémentation défectueuse de certaines politiques.

Le projet est réparti en trois étapes :

- La création du réseau,
- La vérification de la cohérence de la topologie étant donné une politique de contrôle d'accès donnée,
- La génération automatique du graphe d'attaques concernant le réseau décrit dans la première étape.

Le logiciel décrit dans ce document, servira tout d'abord à créer la représentation graphique d'une topologie de réseau. L'utilisateur pourra affecter des propriétés aux divers ordinateurs présents dans cette topologie, et pourra créer des liens entre les éléments de ce réseau et les vulnérabilités existantes. Finalement, l'application devra être en mesure de générer le graphe d'attaques adopté à ce réseau en fonctions des vulnérabilités présentes dans ce dernier.

3.2.2 Objectifs

À la fin de ce projet, l'utilisateur devra disposer d'un outil graphique qui permet de :

✓ Modéliser une topologie de réseau en insérant les éléments du réseau dans une fenêtre à l'aide de boutons,

✓ Définir les liens entre les nœuds du réseau.

✓ Visualiser et modifier les propriétés de toutes les machines .. les composants du réseau (nom, identifiant, processus, système d'exploitation, utilitaires etc.).

✓ Sauvegarder la spécification (représentation graphique) de la topologie créée dans un fichier XML.

✓ Récupérer la représentation graphique déjà sauvegardée en fichier XML dans un dossier.

✓ Extraire les vulnérabilités existantes dans le réseau en fonctions des propriétés des composantes de ce dernier.

✓ Produire une bonne génération du graphe d'attaques associé au réseau produit que ce soit avec ou sans stratégie de l'intrus et ce en suivant une logique LTL cohérente.

3.3 Outils de développement

Afin d'assurer le bon développement de notre application, nous avons eu recours aux outils suivants :

- **Le langage Java :** qui est un langage de programmation informatique orienté objet, pratique pour l'implémentation des applications notamment les applications graphiques.

- **Eclipse :** qui est un environnement de production de logiciels libre, extensible, universel et polyvalent. Utilisé pour produire et fournir des outils pour la réalisation de logiciels, englobant les activités de programmation en s'appuyant principalement sur Java ; Vu qu'il est destiné au langage Java, même si grâce à un système de plugins il peut également être utilisé avec d'autres langages de programmation.

- **Jar :** Java Archive, ou JAR, est un fichier compressé utilisé par le Java Runtime Environment pour distribuer des programmes et bibliothèques écrites en Java. Les fichiers JAR contiennent des fichiers de classe et les ressources d'applications.

- **API Jdom.jar :** JDOM est une API open source Java, dont le but est de représenter et manipuler un document XML de manière intuitive pour un développeur Java, sans requérir une connaissance pointue de XML. Par exemple, JDOM utilise des classes plutôt que des interfaces. Ainsi pour créer un nouvel élément, il faut simplement instancier une classe.

- **API Jgraphx.jar :** est une API open source java, dont le but est de représenter et manipuler un graphe.

- **API Com.mysql.jdbc-5.1.5.jar :** une API permettant de faire le lien avec la base de données utilisée.

- **Wamp server :** est un serveur de bases de données Mysql qui permet de gérer la base de données des vulnérabilités.

- **Base de données de vulnérabilités (OSVDB) :** crée en 2002, c'etait une base de données indépendante et open-source, fournissant des informations techniques précises, détaillées, actuelles et impartiales sur les failles de sécurité, depuis 2008 elle ne devient plus disponble, c'est d'ailleurs pour cela qu'on a récupérer celle-ci à laquelle on a ajouter de nouvelles vulnérablités.

3.4 Architecture logicielle

3.4.1 Modèle de données

Le modèle de données de cette application sera principalement composé de trois objets : la topologie du réseau et ses composants, la spécification des propriétés, et le générateur de scénarios d'attaques. Le premier rôle de l'objet de type Topologie sera de gérer l'ajout et la suppression des nœuds et liens entre eux. Pour éviter la confusion, il ne permettra pas à deux nœuds différents de porter le même nom. Il devra aussi prendre en charge la lecture et la sauvegarde dans un fichier XML. La topologie sera aussi en charge du suivi du nœud et du lien sélectionnés, et ce pour la génération. Finalement, le contrôleur devra parfois savoir si la topologie a été modifiée depuis la dernière sauvegarde.

Chaque nœud aura comme attributs, un nom visible à l'écran, et des propriétés internes attribuées qui seront utilisées plus tard pour la génération, mais aussi des informations notamment le type (routeur, ordinateur, pare-feu, etc.) et l'identifiant réseau.

Le générateur de scénarios d'attaques gère les vulnérabilités existantes dans les nœuds par rapport aux propriétés, attribue une logique de génération, et éventuellement une stratégie de l'intrus afin d'assurer la bonne génération du graphe d'attaque correspondant au réseau, préalablement créé.

3.4.2 Contrôleur

Dans cette application, le contrôleur contiendra une barre de menus et une barre d'outils. Ils donneront accès aux actions bien connus dans les menus Fichier, Édition et Aide. Ces actions sont décrites plus en profondeur dans les tableaux qui suivent. La pré condition indique quand l'action doit être accessible, et la post condition indique quelles conditions doivent être respectées avant d'exécuter l'action.

Détails des actions du menu Fichier			
Action	**Pré-condition**	**Post-condition**	**Résultat**
Ouvrir	– –	L'utilisateur pourra enregistrer un projet déjà ouvert.	La topologie courante est remplacée par celle dans le fichier.
Fermer	Fichier ouvert	L'utilisateur pourra enregistrer un fichier déjà ouvert.	Une topologie vide remplace la précédente.
Enregistrer	Fichier modifié	– –	La topologie est enregistrée
Enregistrer sous	Fichier modifié	Le nom du fichier change.	La topologie est enregistrée.
Quitter	– –	L'utilisateur pourra enregistrer un fichier déjà ouvert.	Arrêt du programme.

TABLE 3.1 – Détails des actions du menu Fichier.

Détails des actions du menu Édition			
Action	**Pré-condition**	**Post-condition**	**Résultat**
Annuler	Fichier non-vide	– –	Ignorer la dernière action faite sur la topologie.
Suivant	retour arrière (Annuler) enclenché	– –	Annuler le dernier retour arrière effectué.
Copier	Nœud sélectionné	– –	Faire une copie du Nœud en sauvegarde.
Couper	Nœud/lien sélectionné	– –	L'objet sélectionné est supprimé.
Coller	Nœud précédemment copié ou coupé	– –	Le dernier objet copié ou coupé est rajouté à la topologie.

TABLE 3.2 – Détails des actions du menu Édition.

Détails des actions du menu Aide			
Action	**Pré-condition**	**Post-condition**	**Résultat**
À propos	– –	– –	Affichage du dialogue « À propos »

TABLE 3.3 – Détails des actions du menu Aide.

Le contrôleur devra aussi intercepter les actions provenant du clavier et de la souris. Par exemple, le déplacement d'un Nœud s'effectuera avec la souris, et l'utilisateur pourra supprimer un lien avec les touches Ctrl-X. Les tableaux qui suivent décriront les actions liées à ceux-ci.

Détails des actions du clavier			
Touche(s)	**Pré-condition**	**Post-condition**	**Résultat**
Ctrl-O	– –	L'utilisateur aura déjà enregistré un projet.	La topologie courante est remplacée par celle dans le fichier.
Ctrl-F	– –	L'utilisateur aura déjà créé un projet.	La topologie courante est remplacée par un nouveau projet vide.
Ctrl-S	– –	L'utilisateur aura déjà créé un projet.	La topologie courante est sauvegardée dans un fichier XML.
Ctrl-Q	– –	– –	Fermeture de la Fenêtre de l'application.
Ctrl-Z	– –	Au moins un Nœud qui figure dans la topologie.	Ignorer la dernière action faite sur la topologie.
Ctrl-Y	retour arrière (Annuler) enclenché	– –	Annuler le dernier retour arrière effectué.
Ctrl-X	Nœud/lien sélectionné	– –	L'objet sélectionné est sauvegardé et supprimé.
Ctrl-C	Nœud sélectionné	– –	L'objet sélectionné est sauvegardé.
Ctrl-V	Nœud précédemment copié ou coupé	– –	Le dernier objet copié ou coupé est rajouté à la topologie.

TABLE 3.4 – Détails des actions des touches clavier.

Détails des actions de la souris			
Touche(s)	**Pré-condition**	**Post-condition**	**Résultat**
Clic bouton gauche	Le clic se fait sur un objet	– –	L'objet est sélectionné.
	Clic dans une zone vide	– –	Il n'y a plus d'objet sélectionné
Clic bouton gauche + déplacement	Le clic se fait sur un Nœud	– –	Le noeud et ses liens sont déplacés. Le Nœud devient l'objet sélectionné.
Clic bouton droit	Le clic se fait sur un Nœud (Machine)	– –	Le Nœud devient sélectionné et le menu contextuel apparait avec les choix Copier/Couper/Supprimer/Propriétés.
	Le clic se fait sur un Nœud (Dispositif)	– –	Le Nœud devient sélectionné et le menu contextuel apparait avec les choix Copier/Couper.
	Le clic se fait dans une zone vide	– –	Le menu contextuel apparait avec le choix Copier.

TABLE 3.5 – Détails des actions de la souris.

3.4.3 Internationalisation

L'application sera disponible en français (par défaut) et en anglais aussi, de façon que toutes les fonctionnalités disponibles affichées en français pourrons êtres traduites en anglais. Ainsi on a une possibilité d'intégrer toutes les langues voulues.

Cette fonctionnalité permet à l'utilisateur de choisir la langue désirée pour les interactions de l'application lors de son lancement.

3.5 Application graphique

En ce qui concerne l'interface homme-machine, elle devra être conviviale et facile d'utilisation. On devra y trouver toutes les qualités d'une application graphique réussie. Il faudra pouvoir manipuler la topologie avec la souris et des menus contextuels. La barre de menus devra être jumelée à une barre d'outils pour avoir un accès rapide aux actions. Les divers contrôles devront être activés ou désactivés selon le contexte actuel. Afin de faciliter son utilisation, le programme devra implémenter des standards d'utilisation bien connus (par exemple : $Ctrl + C$ pour copier un nœud, $Ctrl + V$ pour coller une copie du nœud, etc.). La représentation graphique de la topologie devra aussi suivre certaines normes de qualité.

Premièrement, elle devra être suffisamment grande pour permettre à l'utilisateur d'avoir une bonne vue d'ensemble de la topologie.

Ensuite, elle sera simple et propre, dépourvue d'objets qui encombrent le champ de vision de l'utilisateur. Aussi, il faudra laisser ce dernier étaler la topologie comme il le souhaite : il pourra déplacer les objets après leur création.

À la fin de ce projet, il existera une version française et anglaise du programme. Le choix du langage se fera au moment du lancement de l'application. Si le langage n'est pas spécifié, le programme sera lancé en français par défaut. Cependant, il est possible d'y intégrer d'autres langues.

Le menu principal de l'application est divisé en 2 régions distinctes : la première contiendra la zone de création de la topologie, et le bas affichera les propriétés affectées au nœud sélectionné. La première devra occuper la majorité de la fenêtre pour donner à l'utilisateur un maximum d'espace pour étaler la topologie. La barre de menus contiendra au minimum les actions classiques qu'on s'attend à trouver dans la majorité des applications. Le menu *Fichier* contiendra les actions *Ouvrir*, *Fermer*, *Enregistrer*, *Enregistrersous* et *Quitter*. Le menu *Edition* aura les actions *Annuler*, *Suivant*, *Copier*, *Couper* et *Coller*. Finalement, le menu *Aide* aura l'option *Apropos*. Les actions dans les menus *Fichier* et *Edition* se trouveront également sur la barre d'outils.

Dans la zone de création de la topologie, l'utilisateur pourra manipuler deux types d'objets : les nœuds, et les liens entre eux. Sept différents types de nœuds devront être pris en charge : *ordinateur*, *serveur*, *parefeu*, *routeur*, *hub*, *switch* et *IDS*. L'interface devra permettre la création de ces objets, leurs suppressions, leurs déplacements, ainsi que les actions copier, couper et coller.

En plus, l'utilisateur devra pouvoir modifier les propriétés des objets existants. L'application devra donc fournir une vue contenant toutes les informations du nœud, laissant l'utilisateur faire ses changements, que ce soit par rapport à la topologie, ou encore à la logiques suivie pour la génération stratégique. Deux boutons permettront soit de confirmer les changements effectués, ou de revenir aux données originales. L'utilisateur sera en mesure de choisir l'intrus et la machine cible.

Le programme devra aussi permettre de générer un graphe d'attaques selon, les propriétés présentes par rapport aux nœuds existants dans la topologie, l'intrus et la cible, ainsi que la stratégie de l'intrus adoptée pour le cas du graphe avec stratégie.

3.5.1 Fenêtre de choix de langue

A l'exécution de l'application, une boite de dialogue permettant de choisir la langue souhaitée est affichée pour conduire à une interface principale avec la langue choisie. La boite de dialogue est affichée comme suit :

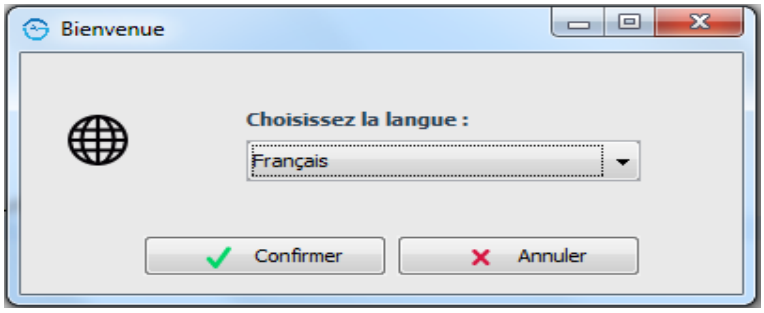

FIGURE 3.1 – Boite de dialogue pour le choix de la langue.

3.5.2 Menu principal

La partie la plus visible de notre application sera la zone de création de topologie. La première fonction qu'elle devra assurer est la création de nouveaux nœuds. Cette action

pourra être faite à tout moment en cliquant sur le bouton approprié sur la barre d'outils puis à l'endroit voulu dans la zone de création. Un nouveau nœud pourra aussi être inséré dans la topologie par l'entremise de *Copier*.

Dans les deux cas, l'application génère automatiquement un nom unique pour le nouvel objet.

Voici une topologie nous permettant de visualiser tout cela :

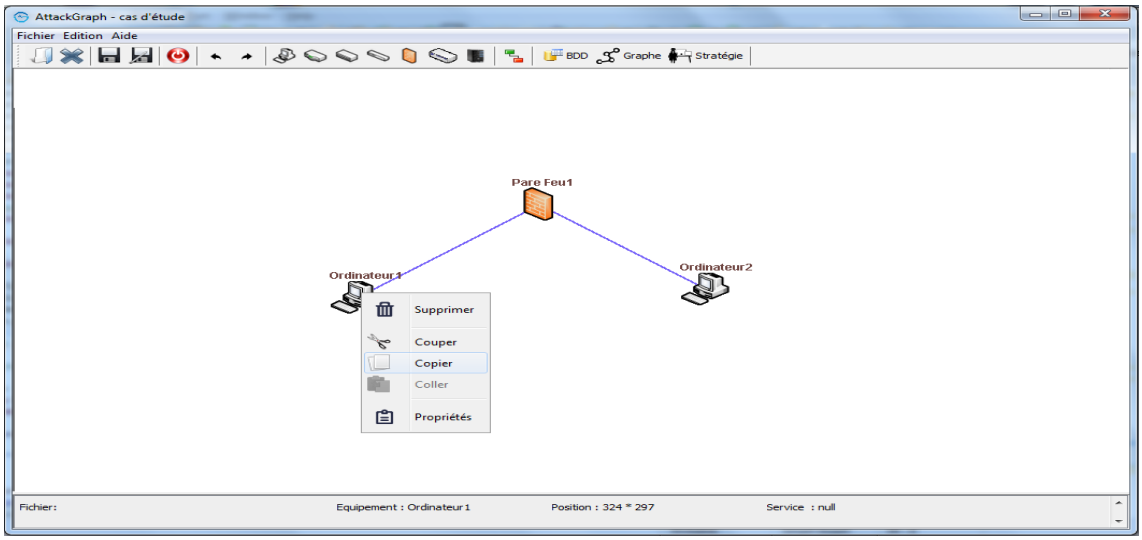

FIGURE 3.2 – Interface de menu avec une topologie réseau.

L'utilisateur a aussi la possibilité d'ouvrir une topologie déjà enregistrée auparavant, comme illustré :

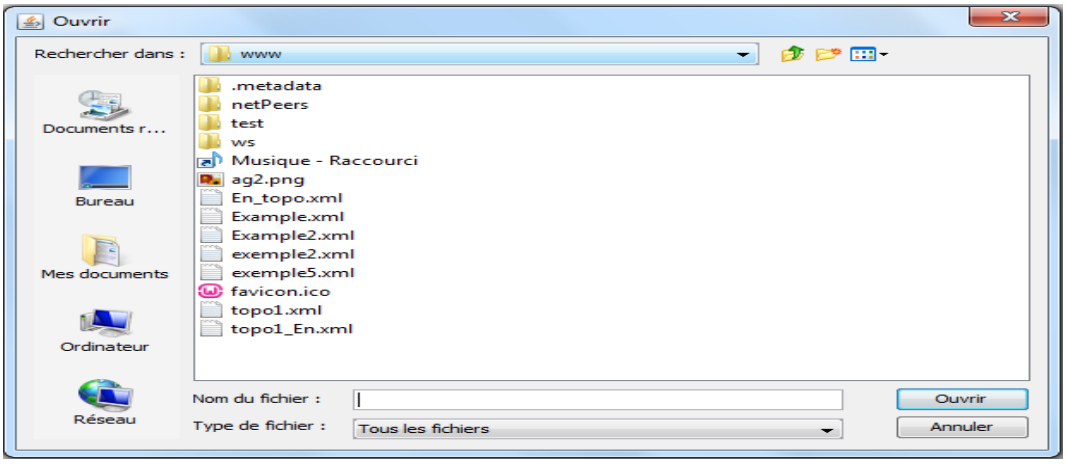

FIGURE 3.3 – Ouverture d'une topologie existante.

Le déplacement d'un objet se fera uniquement avec la souris, en cliquant dessus et en bougeant le curseur à l'endroit voulu. L'objet déplacé sera automatiquement sélectionné s'il ne l'était pas auparavant.

Après la création de la topologie, l'utilisateur pourra modifier les propriétés des nœuds existants dans la topologie. L'application devra donc fournir une vue contenant toutes les informations du nœud, laissant l'utilisateur faire ses changements, comme illustré :

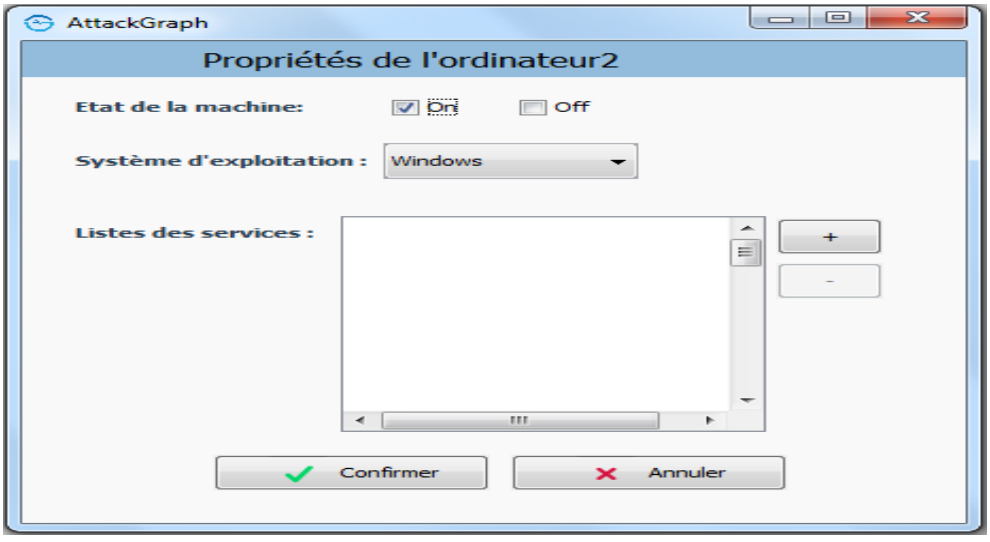

FIGURE 3.4 – Interface permettant l'ajout de propriétés.

Cette interface permet de définir si la machine en question est en marche ou non, et

de determiner le système exploité par cette dernière, permettant de donner accès à une autre interface, afin de choisir les services appropriés parmis une liste propre au système d'exploitation déterminé, comme illustré ci-après :

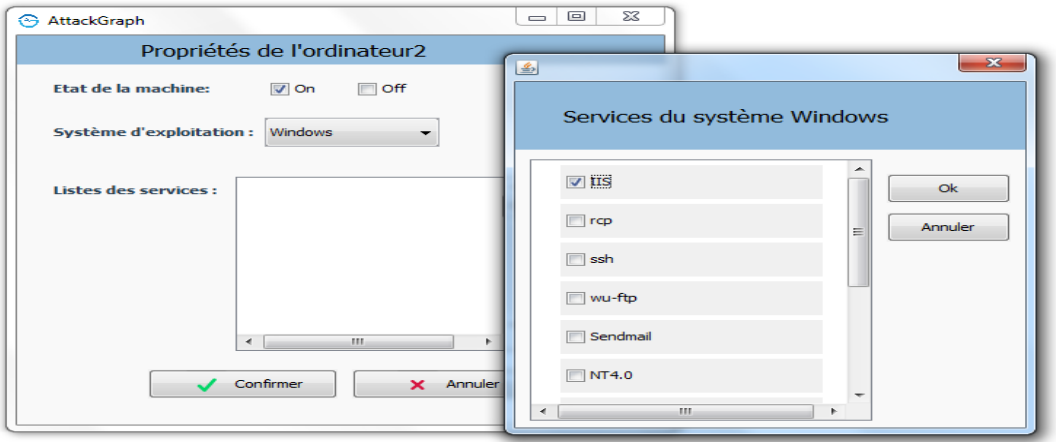

FIGURE 3.5 – Interface permettant la selection de services.

A noté que l'utilisateur a aussi la possibilité d'enregistrer sa topologie à n'importe quel moment, et ce en appuyant sur le bouton enregistrer ou enregistrer-sous, la figue ci-après illustre cela :

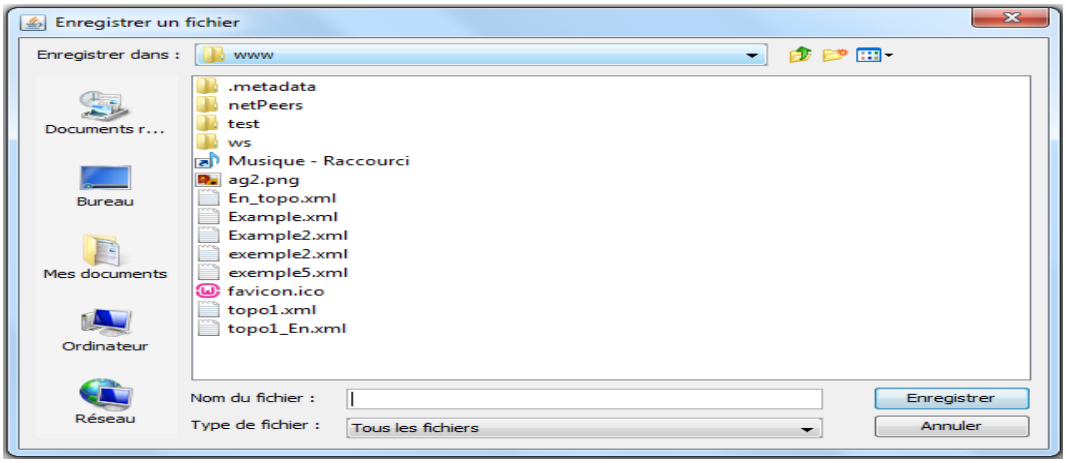

FIGURE 3.6 – Interface permettant l'enregistrement d'une topologie.

3.5.3 Choix de l'intrus et de la cible

Une fois la topologie complète et les propriétés attribuées aux machines (ordinateurs et serveurs), l'utilisateur a la possibilité de choisir la machine faisant office d'intrus ainsi que la machine cible.
Un exemple est illustré dans la figure ci-après :

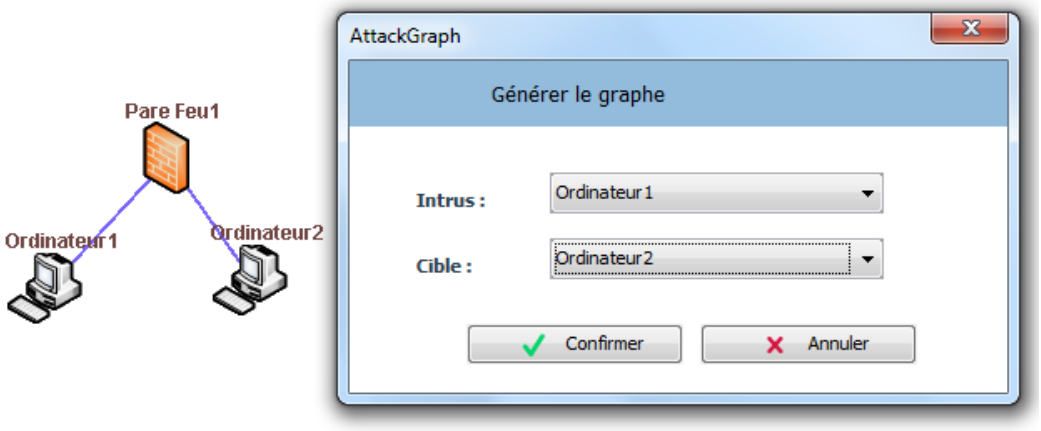

FIGURE 3.7 – Interface de choix de l'intrus et de la cible.

Après avoir effectué le choix de l'intrus et de la cible, l'utilisateur pourra générer le graphe d'attaque associé à la topologie antérieurement crée, et ce à l'aide des deux boutons pour créer soit avec une stratégie de l'intrus ou sans.

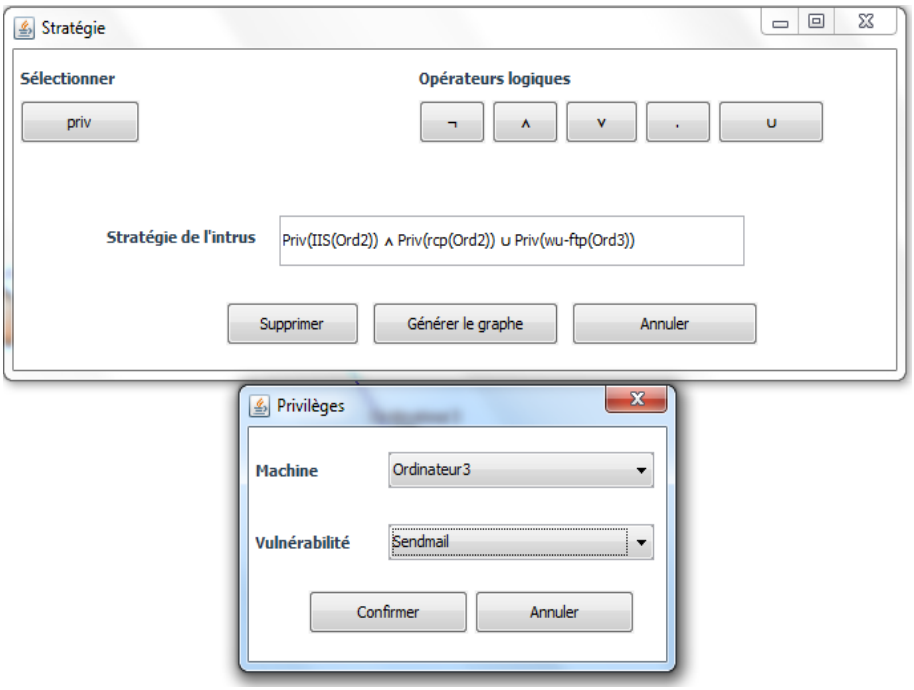

FIGURE 3.8 – Interface de choix pour la stratégie de l'intrus.

3.6 Exemples explicatifs

Dans cette section, nous allons étudier deux exemples de réseaux pour, à la fin, générer leurs graphes d'attaques respectifs.

3.6.1 Exemple 1

Dans ce premier exemple, nous avons un réseau composé de trois hôtes qui sont : $Ordinateur1$, $Ordinateur2$, $Ordinateur3$, un Hub $Hub1$ et d'un pare-feu $Parefeu1$. La machine $Ordinateur1$ 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 pour atteindre la machine $Ordinateur3$.

Le pare feu sépare le monde interne du monde externe, le réseau interne contient les deux ordinateurs $Ordinateur2$ et $Ordinateur3$, avec un hub entre les deux.

L'intrus lance son attaque en commençant par un seul ordinateur $Ordinateur1$ qui se trouve dans le réseau externe.

Le pare-feu met donc en œuvre la politique suivante :

✓ Le trafic ssh est autorisé à la fois à *Ordinateur*2 et *Ordinateur*3 ;

✓ Le trafic web est autorisé uniquement à *Ordinateur*2 qui exécute IIS ;

✓ Email est autorisé à *Ordinateur*3 ;

✓ Le trafic FTP est bloqué parce que *Ordinateur*3 exécute le serveur wu-ftpd ;

✓ Tout trafic sortant est autorisé.

La figure ci-après schématise le modèle du réseau de ce premier exemple :

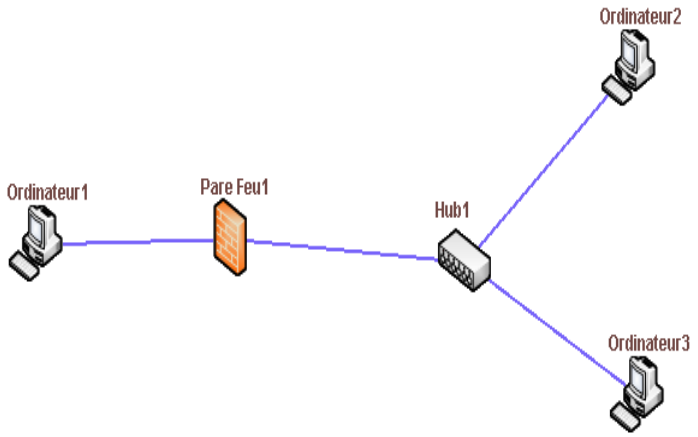

FIGURE 3.9 – Exemple de réseau.

3.6.2 Services et vulnérabiltés du réseau

Le tableau suivant présente les services et les vulnérabilités de chaque machine du réseau :

Code	Service/SE	Vulnérabilité	Machine
IIS	Windows/ Unix	IIS Remote Data Services	$Ordinateur2$
RCP	Windows	RCPDOWNLOAD	$Ordinateur2$
SSH	Windows	PORTFORWARD	$Ordinateur3$
FTP	Windows/ Unix	wuftpd	$Ordinateur3$
$Send-mail$	Windows	– –	$Ordinateur3$

TABLE 3.6 – Services et vulnérabiltés du réseau

3.6.3 Description des vulnérabilités

Le but d'un intrus est d'acquérir des privilèges pour ateindre sa cible, pour cela il doit exploiter les vulnérabilités, chaque vulnérabilité correspond à une attaque, dans ce qui suit, nous allons expliquer chaque vulnérabilité et son impact sur la machine en question :

- IIS Remote Data Services (attaque 0) : cette attaque exploite la vulnérabilité IIS, qui utilise Internet, permettant d'exécuter des programmes sur la machine cible.

- RCPDOWNLOAD (attaque 1) : cette attaque exploite la vulnérabilité *rcp*, permettant le transfert de programmes et services depuis la machine A vers I.

- PORTFORWARD (attaque 2) : cette attaque exploite la vulnérabilité *ssh*, permettant de contourner le pare-feu pour avoir un libre accès à la machine cible.

- WUFTPD (attaque 3) : cette attaque exploite la vulnérabilité ftp, permettant de créer une relation de confiance entre la machine I et la machine B, tel que I pourrait envoyer un mail à B (exécutant le service $send-mail$) en tant que A.

3.6.4 Génération du graphe d'attaques

L'attaquant étant doué de mémoire, certains scénarios ne diffèrent que par l'ordre des attaques. ceci se traduit sur le graphe d'attaques. La figure ci-après présente le graphe paramétré par son temps d'exécution :

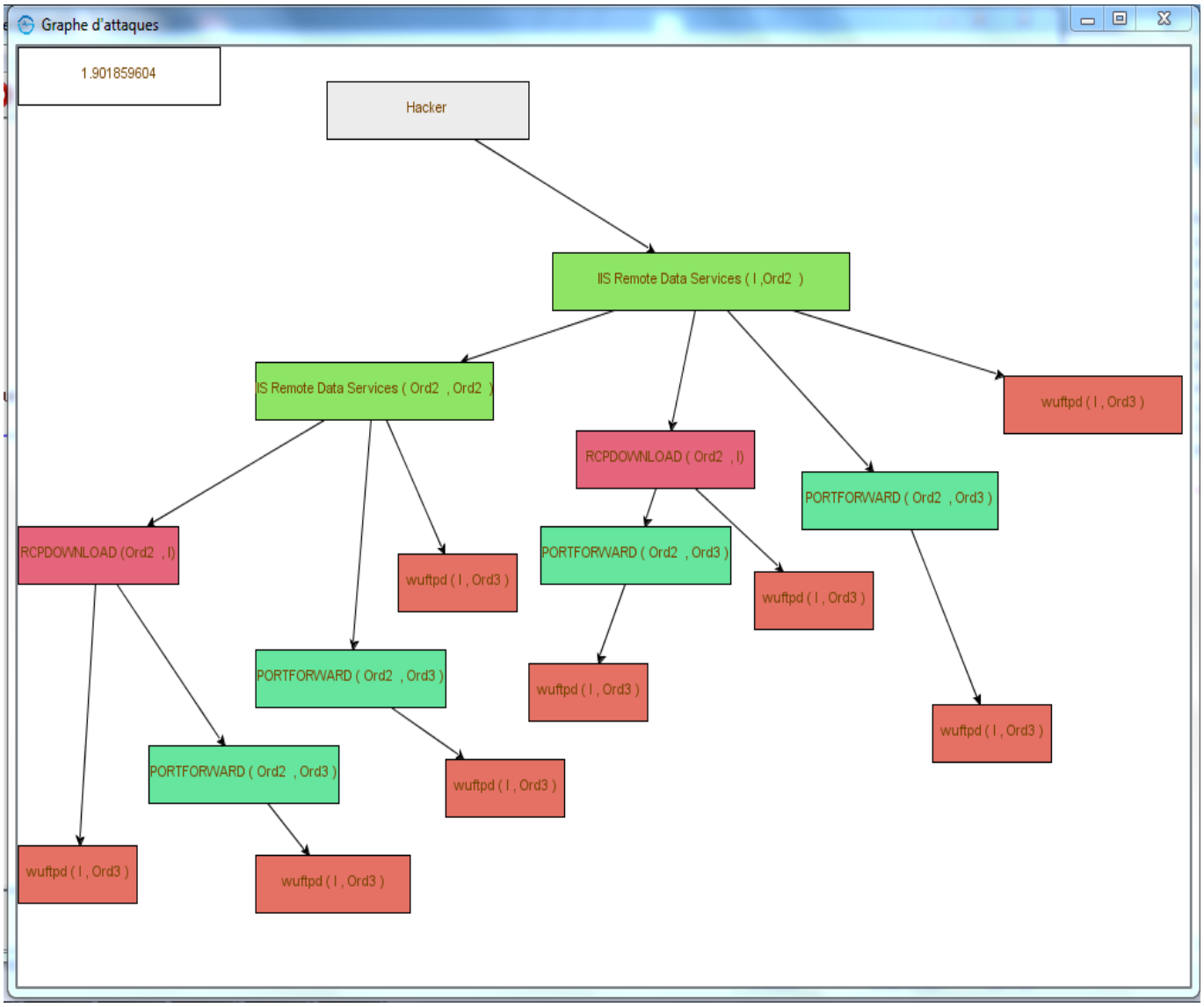

FIGURE 3.10 – Interface de graphe d'attaques.

3.6.5 Exemple 2

Dans ce deuxième exemple, nous avons un réseau composé de trois machines qui sont : $Ordinateur1$, $Ordinateur2$, un serveur $Serveur1$, un Routeur $Routeur1$ et d'un pare-feu $Parefeu1$. La machine $Ordinateur1$ 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 pour atteindre la machine $Serveur1$.

Le pare feu sépare le monde interne du monde externe, le réseau interne contient les deux ordinateurs $Ordinateur2$ et $Serveur1$, avec un routeur entre les deux.

L'intrus lance son attaque en commençant par un seul ordinateur $Ordinateur1$ qui se trouve dans le réseau externe. Pour être plus concret, nous supposons que l'objectif d'intrusion est de perturber le fonctionnement normal de la base de données du serveur $Serveur1$. Pour atteindre cet objectif, l'intrus a besoin d'un accès *root* à la base de données du serveur. Les états du modèle comprennent des services en cours d'exécution sur chaque ordinateur, la relation de connectivité entre les ordinateurs, Le pare-feu met donc en œuvre la politique suivante :

- ✓ Le trafic sshd est autorisé uniquement à $Ordinateur2$;
- ✓ Le trafic http est autorisé uniquement à $Ordinateur2$;
- ✓ Le serveur est un serveur de base de données ;

La figure ci-après montre la topologie du réseau :

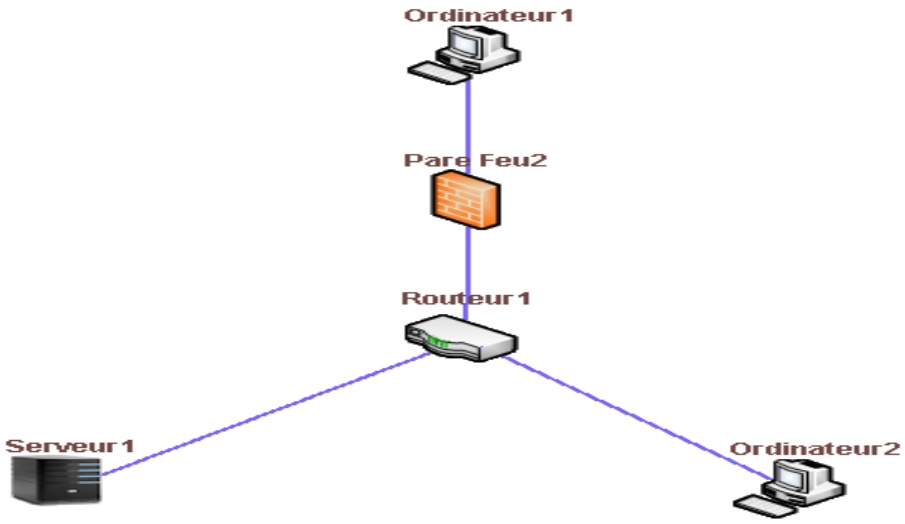

FIGURE 3.11 – Exemple de réseau.

Services et vulnérabiltés du réseau

Le tableau suivant présente les services et les vulnérabilités de chaque machine du réseau :

Code	Service/SE	Vulnérabilité	Machine
$SSHd$	Windows	buffer overflow sshd	$Ordinateur2$
FTP	Windows/ Unix	ftp.rhost	$Ordinateur2$

TABLE 3.7 – Services et vulnérabiltés du réseau

3.6.6 Description des vulnérabilités

Le but d'un intrus est d'acquérir des privilèges pour atteindre sa cible, pour cela il doit exploiter les vulnérabilités, chaque vulnérabilité correspond à une attaque, dans ce qui suit, nous allons expliquer chaque vulnérabilité et son impact sur la machine en question :

- BUFFER OVERFLOW SSHD (attaque 0) : Cette attaque exploite une sshd de vulnérabilité qui donne immédiatement "root shell" dans la machine cible.

- FTP.RHOST (attaque 1) : Cette attaque utilise la vulnérabilité de ftp ; l'intrus crée un fichier "Rhost" dans le répertoire source de ftp ; ceci crée une relation de

confiance entre sa machine et la cible.

– Connexion à distance (attaque 2) : Cette attaque utilise la vulnérabilité de ftp ; L'intrus exploite la relation de confiance entre les deux machines et atteint un terminal de l'utilisateur de la machine cible sans mot de passe.

– BUFFER OVERFLOW LOCAL (attaque 3) : Une fois que l'intrus a accès à l'utilisateur shell sur la machine cible, la prochaine étape est l'exploitation de la vulnérabilité "buffer overflow" sur le fichier "setuid root" pour obtenir un chemin d'accès root.

3.6.7 Génération du graphe d'attaques

Le graphe d'attaques correspond à ce deuxième exemple en exploitant toutes les vulnérabilités du réseau, la figure ci-après montre le graphe obtenu paramétré par son temps d'exécution :

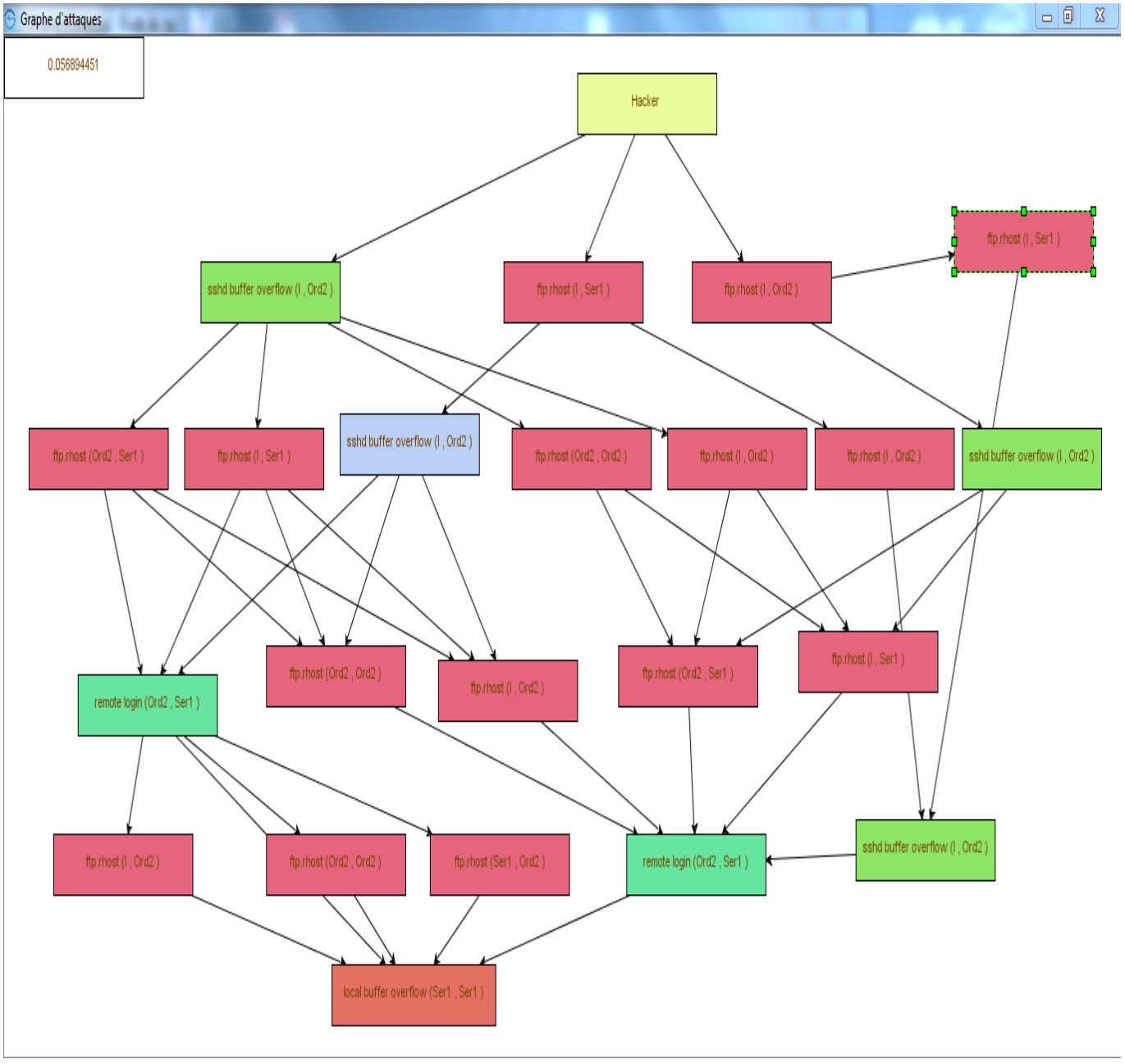

FIGURE 3.12 – Interface de graphe d'attaques.

3.7 Analyse de performances et résultats

Dans cette section nous allons analyser les performances de l'algorithme décrit dans le chapitre précédent. Le temps que prend cet algorithme pour générer un graphe dépend essentiellement du nombre de machines et de vulnérabilités du réseau en question.

3.7.1 Temps d'exécution en fonction du nombre de machines

Le tableau suivant présente le temps d'exécution pour des réseaux qui différent selon le nombre de machines.

Nombre de machines (unité)	27	375	982	1359
Temps d'exécution (secondes)	0.17825793	13.952353	83.45197	178.53055

TABLE 3.8 – Services et vulnérabiltés du réseau

Les temps d'exécution nécessaires à la génération des graphes d'attaques correspondants à chacun des réseaux spécifiés dans le tableau précédent sont illustrés par la figure ci-dessous. Ces résultats sont obtenus en utilisant une machine avec deux processeurs 2.20 Ghz CPU et 3Go de RAM.

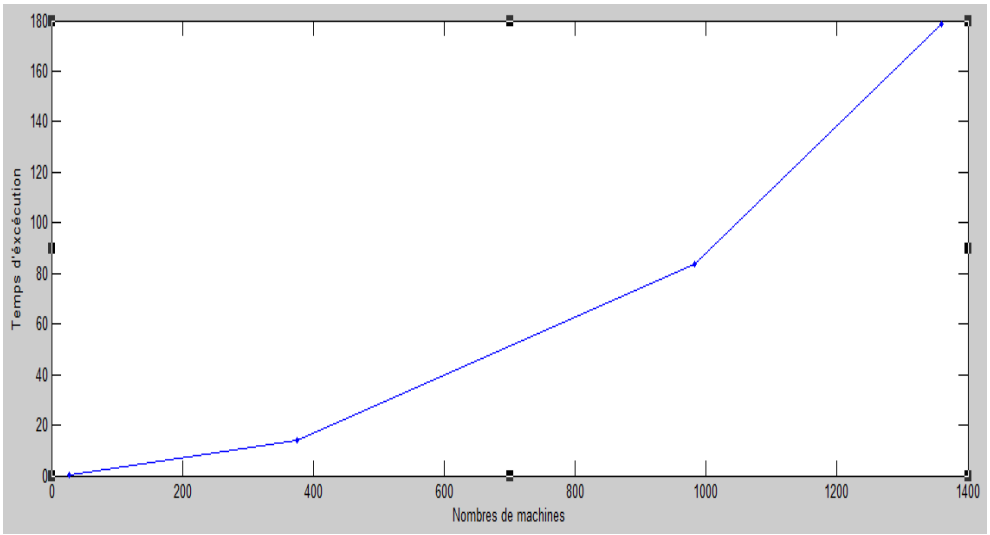

FIGURE 3.13 – Temps d'exécution des graphes générés en fonction du nombre de machines.

Dans cette représentation graphique, nous constatons que les temps de génération des graphes d'attaques sont raisonnables. En effet, pour les réseaux dont le nombre de

machines est inférieur à 100 machines, le temps que prend l'algorithme pour générer le graphe d'attaques est inférieur à 1 seconde, même pour un réseau de plus de 100 équipements le temps reste quand même raisonnable.

3.7.2 Temps d'exécution en fonction du nombre de machines et de vulnérabilités

Le tableau suivant présente le temps d'exécution pour des réseaux avec un nombre de machines et de vulnérabilités différent.

30 machines				
Nombre de vulnérablités (unité)	50	400	1000	4000
Temps d'exécution (secondes)	0.079691775	0.38168165	0.98985577	3.7308333
100 machines				
Nombre de vulnérablités (unité)	50	400	1000	4000
Temps d'exécution (secondes)	0.27892122	1.6441672	3.2673628	14.4724455
150 machines				
Nombre de vulnérablités (unité)	50	400	1000	4000
Temps d'exécution (secondes)	0.33764148	1.9650314	5.2344913	17.91567

Table 3.9 –

Dans cette section nous avons étudié trois exemples de réseaux de 30, 100 et 150 machines. On varie pour chacun de ces réseaux le nombre de vulnérabilités. On calcule le temps d'exécution nécessaire à l'algorithme pour générer le graphe d'attaques correspondant à chacun des trois réseaux et cela pour un nombre de vulnérabilités de 50, 400, 1000 et 4000 vulnérabilités.
Les résultats obtenus sont illustrés par la représentation graphique suivante :

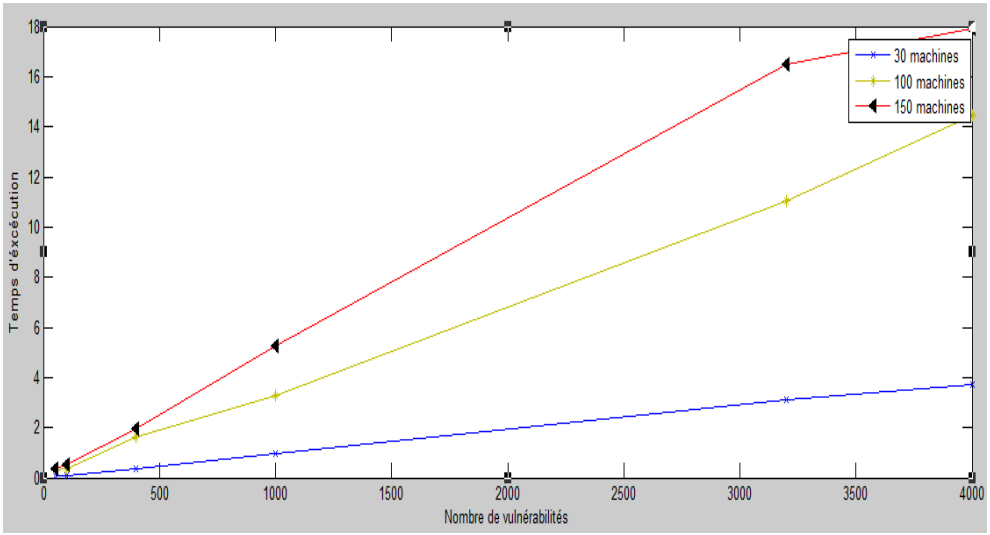

FIGURE 3.14 – Temps d'exécution des graphes générés en fonction du nombre de machines et de vulnérabilités.

3.8 Conclusion

La partie implémentation est l'étape la plus importante dans notre projet. Ce chapitre a décrit trois phases principales. La phase de description où nous avons définis les outils mis en œuvre pour réaliser notre application, ainsi que les différentes fonctionnalités de cette dernière. La phase réalisation où nous avons décris les différentes fonctions de l'application mises en pratique avec quelques interfaces pour bien illustrer le projet. Et la phase de simulation, où nous avons permis une extension de notre application à un certain nombres de machines.

Nous clôturons ce chapitre avec deux exemples explicatifs bien exposés, suivant la logique de l'application, depuis la topologie jusqu'au graphe d'attaques généré.

Conclusion et perspectives

Etant au cœur de la dynamique économique des années à venir. Le développement des échanges électroniques entraîne des changements profonds dans l'organisation et le fonctionnement des entreprises, dans leurs rapports avec les clients, dans leurs comportements sur le marché mondial. L'efficacité et la pertinence du recours aux technologies de l'information et de la communication deviennent des éléments discriminants dans la concurrence. C'est pour cela que les entreprises et les administrations en font une priorité stratégique et en oublie les problèmes posés par ces derniers.

L'étude réalisée dans ce présent mémoire qui consistait en la réalisation d'une application graphique pour la génération de graphes d'attaques pour des réseaux d'entreprises diverses où l'objectif majeur était de mettre en œuvre une stratégie de l'intrus avec une implémentation d'un algorithme déjà pensé pour la génération des scénarios d'attaques. Cette application permet à l'utilisateur de créer une topologie de réseau comme il la souhaite et de générer le graphe d'attaques approprié avec les différents scénarios d'attaques le composant que ça soit avec une stratégie de l'intrus ou sans.

Le travail accompli au cours de ce projet nous a permis de mieux comprendre le milieu de la sécurité informatique et nous a aidé à avoir une vision réelle de la mise en place d'une politique de sécurité pour chaque entreprise.

La phase d'implémentation de l'algorithme générant le graphe d'attaques a été le cœur du développement. Au cours de cette étape, nous avons essayé de structurer et de définir les besoins attendus du futur système, et de les mettre en pratique.

Ce projet se conclut par une réalisation d'une première version opérationnelle, qui nous a permis d'atteindre les objectifs initialement fixés. Cependant, c'est une application à perfectionner. Il pourrait constituer le point de départ d'un projet plus complexe et plus riche en termes de nouvelles vulnérabilités dans le but de développer des environnements pour le renforcement automatique de politiques de sécurité dans les systèmes informatiques.

En effet cette petite expérience nous a été bénéfique et très utile, car elle nous a permis de nous familiarisés avec de nouvelles notions d'une part, et d'enrichir nos connaissances avec les outils de programmations d'autre part. Nous espérons enfin que ce travail sera au niveau de la tâche qui nous a été confiée et qu'il sera bénéfique pour les étudiants qui

feront référence à ce mémoire.

Cependant, le projet que nous avons réalisé demeure un chantier ouvert à d'éventuels perfectionnements, à savoir :

- Analyse du graphe selon un algorithme plus optimal,
- Assurer la portabilité de la base de données,
- Générer un fichier executable afin d'avoir une application portable, etc.

Bibliographie

[1] V. Shandilya, Chris B. Simmons, S. Shiva, *Use of attacks graphs in security systems,* Journal of Computer Networks and Communications Volume 2014, Article ID 818957, 13 pages Hindawi Publishing Corporation Tzonelih Hwang Memphis, USA, 2014.

[2] X. Ou, R. Swallia, *Identifying critical attak assets in dependency attack graphs,* Third European Symposium on Research in Computer Security (ESORICS), September, 2008.

[3] J. M, R. Wing, P. Lippmann, J. Haines, O. Sheyner, S. Jha, *Automated generation and analysis of attack graphs,* IEEE Symposium on Security and Privacy, Oakland, California, 2002.

[4] S. Vivek, S. Chris B, S. Sajjan, *Use of attacks graphs in security systems,* Journal of computer networks and communication volume, Article ID 818957, Hindawi Publishing Corporation, 2014.

[5] J. Hong and D.-S. Kim, *Harms : Hierarchical attack representation models for network security analysis,* SRI Security Research Institute, Edith Cowan University, Perth, Australia, 2012.

[6] N.C. Idika, *Characterizing and aggregating attack graph based security metric,* Purdue University, Center for Education and Research, Information Assurance and Security, thèse de doctorat en Informatique, 2010.

[7] K. Ingols, M. Chu, R. Lippmann, S. Webster, S. Boyer, *Modelingmodern network attacks and countermeasures using attack graphs,* the 25th Annual Computer Conference Security Applications (ACSAC '09), pages 117-126, 2009.

[8] V. Viduto, W. Huang, and C. Maple, *Toward optimal multiobjective models of network security : survey,* the 17th International Conference onAutomation and Computing (ICAC '11), pages 6-11, 2011.

[9] P. Xie, J. H. Li, X. Ou, P. Liu, R. Levy, *Using bayesian networks for cyber security analysis,* the 2010,IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '10), pages 211-220, Chicago, Ill, USA, 2010.

[10] M.Chu,K. Ingols, R. Lippmann,S.Webster,S. Boyer, *Visualizing attack graphs, reachability, and trust relationships withnavigator,* the 7th International Symposium on Visualization for Cyber Security (VizSec '10), pages 22-33, ACM, New York, USA, 2010.

[11] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing, *Ranking attack graphs,* the 9th International Conference on Recent Advances in Intrusion Detection (RAID '06) pages 127-144, Springer, Berlin, Germany, 2006.

[12] J. Homer, S. Zhang, X. Ou, *Aggregating vulnerability metrics in enterprise networks using attack graphs,* Journal of Computer Security, volume 21, no. 4, pages 561-597, 2013.

[13] P. Cheng, L. Wang, T. Long, *Compressing attack graphs through reference encoding,* the 10th IEEE International Conference on Computer and Information Technology, (CIT '10), pages 1026-1031, 2010.

[14] P. Kijsanayothin, R. Hewett, *Analytical approach to attack graph analysis for network security,* the fifth International Conference on Availability, Reliability, and Security (ARES '10), pages 25-32, 2010.

[15] S.Noel, S. Jajodia, L.Wang, A. Singhal, *Measuring security risk of networks using attack graphs,* International Journal of Next-Generation Computing, volume 1, no. 1, 2010.

[16] R. Lipmann, K. Ingols, *An annotated review of past papers on attack graphs,* Tech. Rep., Lincoln Laboratory, 2005.

[17] I. Kotenko, M. Stephas-hkin, dans *Attack graph based evaluation of network security,* Communications and Multimedia Security, pages 216-227, Springer, Berlin, Germany, 2006.

[18] R. Dewri, N. Poolsappasit, I. Ray, D. Whitley, *Optimal security hardening usingmulti-objective optimization on attack tree models of networks,* the 14th ACM Conference on Computer and Communications Security (CCS'07), pages 204-213, ACM, 2007

[19] L. Wang, T. Islam, T. Long, A. Singhal, S. Jajodia, *An attack graph-based probabilistic securitymetric,* the 22nd Annual IFIP WG 11.3 Working Conference on Data and

Applications Security, pages 283-296, Springer, Berlin, Germany, 2008.

[20] X. Ou, WF. Boyer,A.McQueen, *A scalable approach to attack graph generation,* the 13th ACM Conference on Computer and Communications Security (CCS '06), pages 336-345, ACM, New York, USA, 2006.

[21] A. Xie, Z. Cai, C. Tang, J. Hu, Z. Chen, *Evaluating network security with two-layer attack graphs,* the 25th Annual Computer Conference Security Applications (ACSAC '09), pages 127–136, 2009.

[22] E. S. Abramov, A. V. Andreev, D. V. Mordvin, and O. B. Makarevich, *Corporate networks security evaluation based on attack graphs,* the 4th International Conference on Security of Information andNetworks (SIN '11), volume 11,pages 29-36, ACM, NewYork, NY,USA, 2011.

[23] H. Huang, S. Zhang, X. Ou, A. Prakash, K. Sakallah, *Distilling critical attack graph surface iteratively through minimum-cost sat Solving,* the 27th Annual Computer Security Applications Conference (ACSAC '11), pages 31- 40, New York, USA, 2011.

[24] L. Massinissa,S. OULD AMARA, *Génération automatique des attaques complèxes* Université de béjaia, département d'informatique, option Réseaux et Systèmes Distribués thèse de Master 2, 2011.

[25] L. Hamza, K. Adi, *Formal technique for discovering complex attacks in computer systems,* International Conference on New Software Methodologies Tools and Techniques, Italie, 2007.

[26] L. Hamza, K. Adi, K. El Ghemhioui, *Automatic generation of attack scenarios for intrusion detection systems,* International Conference of Internet and Web applications and services, IEEE Computer Society press, 2006.

[27] L. Hamza, *Génération des scénarios d'attaques pour les systèmes de détection d'intrusions,* 3ème forum de béjaia, Algérie, 2009.

[28] R. Ritchey, P. Amman, *Using model checking to analyze network vulnerabilities,* the IEEE Symposium on Security and Privacy, pages 156-165, Oakland, California, 2000.

[29] O. Sheyner, *Scenario graphs and attack graphs,* School of Computer Science Computer Science Department Carnegie Mellon University Pittsburgh, Avril, 2004.

[30] L. Williams, R. Lippmann, K. Ingols, *An interactive attack graph cascade and reachability display,* the Workshop on Visualization for ComputerSecurity, Mathematics and Visualization, pages 221-236, Springer, Berlin, Germany, 2008.

[31] M. Gadelrab, A. Abou El Kalam, Y. Deswarte, *Execution patterns in automatic malware and human-centric attacks NCA 2008,* the Seventh IEEE International Symposium, 2008.

[32] M. Dacier, Y. Deswarte, *Privilege graph : an extension to the typed access matrix model,* the Third European Symposium on Research in Computer Security (ESORICS'94), pages 319-334, London, UK, 1994.

[33] E. Sawilla, X. Ou, *Identifying critical attack assets in dependency attack graphs,* the Third European Symposium on Research in Computer Security (ESORICS), volume 5283 of Lecture Notes in Computer Science, pages 18-34, Springer, Berlin, Germany, 2008.

[34] L. Hamza, *Génération automatique de scénarios d'attaques pour les systèmes de détection d'intrusions,* Université de Bejaia, département d'Informatique, Thèse de magister en Informatique, 2005.

[35] C. Llorrens, *Mesure de la sécurité logique d'un réseau d'un opérateur de télécommunications,* Ecole Nationale Supérieure des Télécommunications, thèse de Doctorat en Informatique, France, 2005.

[36] R. Hoare, *An axiomatic basis for computer programming,* Communications of the ACM, pages 576-580, 1969.

[37] L. Mohand Oussaid, *Vérification formelle des propriétés de sécurité des logiciels,* Institut National d'Informatique,Thèse de magister en Informatique, département d'informatique, 2006.

Résumé

Les systèmes informatiques actuels deviennent de plus en plus vulnérables, avec l'utilisation fréquente de plus en plus de protocoles de communication réseau, et le besoin grandissant d'ouverture de ces systèmes à Internet, de nouvelles vulnérabilités voient le jour ainsi, de nouvelles attaques exploitant ces vulnérabilités apparaissent régulièrement et de façon continue. C'est pour cela qu'il est devenu primordial non de définir des mécanismes pour empêcher les intrus d'attaquer mais de pouvoir les détecter afin d'avoir une vue sur les failles de ces systèmes vulnérables et ainsi d'y remédier. C'est ce qui fait que la sécurité de ces systèmes est devenue un souci majeur. Dans ce domaine, la génération automatique de graphes d'attaques est bien utile pour l'analyse et la configuration de la sécurité réseau ainsi que pour la détection des attaques informatiques.

Le principal but de ce projet est de réaliser une application graphique permettant d'avoir une topologie de réseau en entrée pour générer un graphe d'attaques associé à cette topologie en sortie, selon une stratégie spécifiée.

Mots-clés :
Sécurité informatique, attaques, graphes, scénarios d'attaques, logique, topologie.

Abstract

Current computer systems become increasingly vulnerable, with the frequent use of more network communication protocols, and the growing need for openness of these systems to the Internet, new vulnerabilities are emerging as well, new attacks exploiting these vulnerabilities appear regularly and continuously. This is why it has become essential not to define mechanisms to prevent intruders to attack but to detect them in order to get a view of the shortcomings of these vulnerable systems and thus to remedy. This is what makes the security of these systems has become a major concern. In this area, the automatic generation of attack graphs is useful for analysis and network security configuration and for the detection of computer attacks.

The main goal of this project is to achieve a graphical application to have an entry in network topology to generate attack graph associated with this output topology, according to a specified strategy.

Keywords :
Computer security, attacks, graphs, attacks scenario, logic, topology.