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

Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

MÉMOIRE DE MASTER RECHERCHE

En
Informatique

Option
Réseaux et Systèmes Distribués

Thème

Analyse des graphes d'attaques dans les systèmes informatiques

Présenté par : M^lle Ouali Lillia
M^lle Bair Narimene

Encadré par : M^lle HAMZA Lamia

Soutenu le 26 Juin 2016 devant le jury composé de :

Présidente	**Dr YAICI Malika**
Examinatrice	**M^me OUAZINE Kahina**

⁕ *Remerciements* ⁕

A l'issue de ce travail, nous tenons à remercier en premier lieu le bon Dieu de nous avoir donné la force, le courage et la santé d'entamer et de terminer ce travail.

Nous tenons à remercier notre promotrice M^{lle} HAMZA Lamia pour avoir accepté de nous encadrer, pour la qualité de son encadrement, sa disponibilité, ses conseils, sa gentillesse, sa confiance et ses compétences scientifiques qui nous ont permis d'élargir nos connaissances. Qu'elle trouve ici l'expression de notre gratitude et notre respect.

Nos remerciements vont également aux membres du jury d'avoir accepté de juger ce travail : M^{me} YAICI Malika pour l'honneur qu'il nous a fait d'avoir accepté de présider le jury d'examinassions et M^{me} OUAZINE Kahina pour l'honneur qu'elles nous a accordé en examinant ce travail.

⁂ *Dédicaces* ⁂

Je dédie ce travail à :

Mes très chers parents qui ont toujours été là pour moi et qui m'ont donné un magnifique model de labeur et de persévérance tout au long de mes études, qu'ils trouvent ici mes profonds remerciements, et que Dieu vous garde pour moi.

Toute ma famille sans exception oncles, tantes, cousins et cousines.

Toute la promotion ReSyD 2015-2016.

Tous mes amis.

Mon aimable binôme lillia et sa famille.

M^{lle} Bair Narimene

⁂ *Dédicaces* ⁂

Je dédie ce travail à :

Mes parents merci pour vos sacrifices, votre soutien dans les moments difficiles, je vous prie de bien vouloir trouver en ce travail le couronnement de vos multiples efforts !

Mes frères et ma sœur, merci pour votre soutien et vos constants encouragements.

Je remercie tout particulièrement mon fiancé pour son aide et son soutien à toute épreuve.

Ma grand-mère à qui je souhaite une longue vie, mes oncles, tentes cousins et cousines.

Mon défunt grand-père, j'aurais tant voulu que tu sois présent avec moi mais Dieu en a voulu autrement.

M^{lle} Ouali Lillia

Table des matières

Table des figures

Liste des algorithmes

Notations et symboles

API : **A**pplication **P**rogramming **I**nterface

C_f : Condition crucial

CP-Net : **C**onditional **P**reference **N**etwork

CSP : **C**onstraint **S**atisfaction **P**roblem

JDom : **J**ava **D**om **o**bject **m**odel

JGraph : **J**ava **G**raph

$\sim$: Négation d'une formule logique

***** : Multiplication

$\cup$: Union

Introduction générale

Le système informatique est généralement défini par l'ensemble des données et des ressources matérielles et logicielles de l'entreprise, permettant de les stocker ou de les faire circuler. Il représente un patrimoine essentiel de l'entreprise, qu'il convient de protéger. Afin de pouvoir sécuriser un système, il est nécessaire d'identifier les menaces potentielles, il est important aussi de connaître et de prévoir la façon de procéder de l'ennemi (la stratégie de l'attaquant).

La majorité des chercheurs se sont concentrés sur la génération des graphes d'attaques car ils sont grands et complexes, mais peu a été faites pour leurs analyse sachant qu'il n'est pas du tout suffisant de générer le graphe d'attaques d'un réseau sans déterminer quelles sont les vulnérabilités suffisantes à enlever afin de mettre un terme aux attaques, ce qui veut dire que l'analyse complète la génération des graphes d'attaques.

Dans ce projet de fin de cycle nous abordons le problème d'analyse des graphes d'attaques, tout en proposant une solution qui facilite aux administrateurs la gestion de la sécurité de leurs réseaux. Notre mémoire est subdivisé en trois chapitres principaux :

Un état de l'art sur l'analyse des graphes d'attaques est présenté dans le premier chapitre. Ce dernier comporte des définitions préliminaires ainsi que

quelques travaux antérieurs. Il comporte également une discussion qui présente une critique de ses derniers et enfin, une conclusion du chapitre.

Le deuxième chapitre intitulé, une technique d'analyse des graphes d'attaques se compose de deux parties. La première partie présente notre proposition qui se compose de trois étapes, et son application à travers un exemple illustratif, ce dernier est généré à partir d'une application de génération des graphes d'attaques faite en parallèle avec notre travail. La deuxième partie est une perspective sur le problème de satisfaction de contraintes (CSP) qui présente la possibilité de l'appliquer pour une éventuelle amélioration.

Le troisième chapitre inclut quelques définitions sur l'environnement utilisé pour le développement de notre application, tel que le langage Java, Eclipse, API extern jGraph et API extern jDom et enfin un exemple qui démontre le bon fonctionnement de notre application.

En termine notre mémoire par une conclusion.

Chapitre 1

État de l'art sur l'analyse des graphes d'attaques

1.1 Introduction

L'analyse des graphes d'attaques a un apport très important dans la sécurité informatique, cela se fait par une analyse qui divulgue des vulnérabilités menaçant le système sachant que les pirates peuvent les exploiter afin de satisfaire leurs exploits.

Ce chapitre est organisé de la manière suivante : on commençent par l'exposition du problème, ensuite des définitions préliminaires et à la fin nous citons quelques travaux antérieurs pour l'analyse des graphes d'attaques.

1.2 Problématique

Les systèmes informatiques sont exposés à de nombreuses attaques. Cependant, le fait de sécuriser un réseau est devenu primordial pour la bonne gestion de ce dernier. Pour se faire, la plupart des chercheurs se sont concentrés sur la génération des graphes d'attaques, afin de détecter les différentes vulnérabilités concernant le réseau, négligeant leurs analyses malgré l'importance de cette étape dans la protection des réseaux informatiques.
La question qui se pose : comment parvenir à sécuriser un réseau en utilisant un graphe d'attaques, contre certaines menaces existantes ?

1.3 Définitions préliminaires

1.3.1 Graphes

Un graphe G est défini par un couple (N , A) avec [7] :

- N : un ensemble fini d'objets, appelés nœuds ou sommets du graphe,
- A $\subseteq$ N $\times$ N : un ensemble de couples de sommets appelés arcs.

Il existe deux types de graphes :

- **Graphe orienté** : Un graphe G= (N , A) est dit orienté lorsque A n'est pas une relation symétrique. C'est à dire, lorsque les arcs (i,j) et (j,i) ne sont pas identiques.
- **Graphe non-orienté** : Un graphe G= (N , A) est dit non-orienté lorsque A est une relation symétrique. C'est à dire, $\forall\ i \in N, \forall\ j \in N : (i\ ,\ j) \in A \Rightarrow (j\ ,\ i) \in A$

✓ L'existence d'un arc implique l'existence de l'arc opposé.

✓ Les arcs d'un graphe non-orienté sont appelés arêtes.

1.3.2 Graphes d'attaques

Un graphe d'attaques représente le comportement des attaquants nuisant à un réseau. Chaque nœud dans le graphe représente un état, généralement spécifié par les attributs du réseau concerné tels que la connectivité entre la source, la victime et le privilège d'accès d'un attaquant dans l'état. Chaque lien représente une action qu'un attaquant prend pour gagner son contrôle d'accès dans le réseau . A partir d'un ensemble des nœuds initiaux, un attaquant peut prendre une action qui exploite la vulnérabilité des réseaux pour atteindre un ensemble d'états satisfaisant son but [4].

1.3.3 Scénarios d'attaques

Un scénario d'attaques est un ensemble de privilèges acquis par l'exploitation des vulnérabilités dont le dernier privilège est un objectif de l'intrus [3].

1.3.4 Réseaux de préférences conditionnelles

Un réseau de préférences conditionnelles (CP-Net) est un modèle graphique qui représente les relations quantitatives de préférences conditionnelles entre les variables de décisions. Chaque nœud du CP-Net représente une variable de décision et chaque lien d'une variable de décision X à une variable de décision Y représente une décision de préférences sur Y conditionnée à une décision sur X. Dom (X) est un ensemble de valeurs de décision pour une variable de décision X.

La préférence sur Y conditionné sur X est annotée dans le graphe et qualitativement exprimée sous la forme $\ll x_1 : y_1 > y_2 \gg$ où, $x_1 \in$ Dom(X) ; y_1,$y_2 \in$ Dom(Y).

Cette expression signifie que nous préférons y_1 que y_2 de la décision sur Y , la décision sur X est x_1. Notez que si chacune des valeurs de Y dépend de la valeur de X correspondante (par exemple, la prime de chaque assureur correspond à l'âge d'un assureur), alors il n'est pas nécessaire d'exprimer une relation de préférence conditionnelle entre X et Y .

La Figure 1.1 montre un CP-Net qui représente une décision de préférence sur un achat de voiture. Il y a deux variables de décision : la couleur et le type de voitures, où Dom (couleur) = {noir, jaune, bleu} et Dom (type) = {camion, fourgon, berline}.

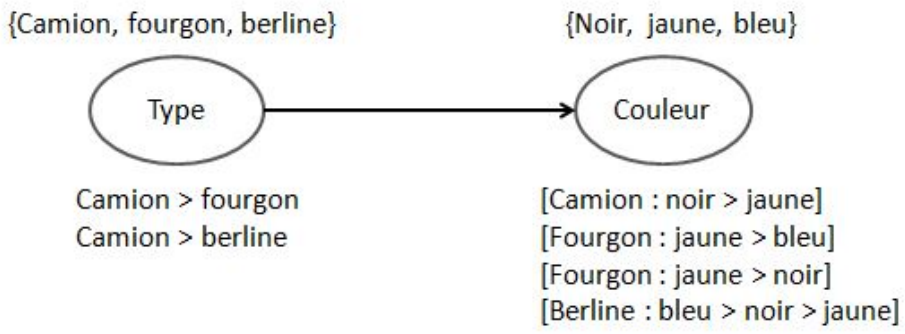

FIGURE 1.1 – CP-Net pour des décisions sur un achat de voiture [4].

Les décisions relatives à la couleur d'une voiture dépendent de la décision du type de voitures (tel que représenté par un lien à partir du nœud type au nœud couleur sur la Figure 1.1). La préférence sur le type de voitures n'est pas conditionnée par d'autres variables de décision. L'ensemble de chaque nœud de valeur du domaine apparaît au dessus du nœud. Il n'y a aucune comparaison de préférence entre une voiture et un fourgon. D'autre part, le choix de la couleur dépend du type de la voiture. Par exemple, le premier choix conditionnel indique que pour un camion, le noir est la couleur préférée que le jaune [4].

1.4 Travaux antérieurs

Dans cette partie nous présentons une synthèse des travaux antérieurs portant sur l'analyse des graphes d'attaques :

Steven Noel et al. [6] présentent la technique de matrice de contigüité [1], cette technique aide les graphes complexes à devenir plus simple. Les lignes et les colonnes d'une matrice d'adjacence peuvent être placées dans un ordre quelconque, sans affecter la structure du graphe d'attaques représenté. Un algorithme de clustring matriciel est appliqué, ce dernier est conçu pour combiner des bloques rectangulaires homogènes à des éléments d'une matrice simultanément. Cette technique utilise un algorithme de clustring automatique, sans paramètres, et varie linéairement avec la taille du problème mais nécessite un calcul de haut niveau de la matrice d'adjacence pour l'accessibilité en plusieurs étapes.

Steven Noel et al. [5] ont décrit un Framework [2] pour gérer la complexité du graphe d'attaques grâce à la visualisation interactive qui comprend l'agrégation hiérarchique des éléments du graphe. Les Framework sont appliqués à des réseaux réels utilisant des logiciels systèmes. Leurs systèmes construisent des modèles vulnérables de réseau automatique utilisant le scanner de vulnérabilité Nessus [12] et inclut quelques exploits. Ce Framework, non seulement utile pour les calculs évolutifs

1. Une matrice de contiguïté est un tableau carré dont les cases indiquent sont par un ou un zéro.
2. Structure logicielle.

mais aussi à l'évolution cognitive. Néanmoins, des détails du niveau inférieur des graphes d'attaques sont cachés jusqu'à ce qu'ils soient désagrégés, et le processus de désagrégation interactive est potentiellement fastidieux.

Vaibhav Mehta et al. [8] proposent une approche basée sur le schéma de classement pour les états de graphe d'attaques ; Dans cette approche on attribue un rang à chaque état, ce rang est la probabilité d'un intrus d'atteindre cet état. Ils donnent deux algorithmes de classement pour classer les états basés sur la probabilité d'un intrus d'atteindre ces états. Le premier algorithme est similaire à Google algorithme de pagerank [11]. Le deuxième algorithme basé sur les rangs des états. Le classement des graphes d'attaques sont utiles pour les systèmes administrateurs car ils lui permettent d'estimer le niveau de sécurité des systèmes et fournir un guide des principales caractéristiques de cette approche :

- facilité et flexibilité de la mobilisation,
- évolutivité et utilité des analyses,
- applicable à des situations variées.

Cette approche présente une difficulté de prendre une décision sur l'action pour protéger le réseau.

Kong wei Lye et al. [2] présentent une méthode de la théorie des jeux pour l'analyse de la sécurité des réseaux informatiques. Ils ont expliqué l'interaction entre un attaquant et un administrateur en deux joueurs de jeu stochastique afin de construire un modèle pour le jeu. Ils calculent l'équilibre de Nash pour les joueurs et l'administrateur. L'équilibre de Nash donne à l'administrateur une idée de la stratégie de l'attaquant pour savoir quoi faire dans chaque état dans le cas d'une attaque. Cette méthode occupe un espace extrêmement grand, et la modélisation de l'action des acteurs particulièrement l'attaquant est difficile.

Dans ce mémoire nous nous sommes basées sur l'analyse statistique automatique des graphes d'attaques à l'aide d'expressions logiques et Cp-Net [4] et l'analyse en temps polynomial des graphes d'attaques [1] .

1.4.1 Analyse statistique automatique des graphes d'attaques à l'aide d'expressions logiques et les réseaux de préférences conditionnelles (Cp-Net)

Phongphun Kijsanyothin et al. [4] présentent une approche de l'analyse statistique automatique des graphes d'attaques à l'aide d'expressions logiques et les réseaux de préférences conditionnelles. L'approche donne des contre-mesures pour les chemins d'attaques sur la base des critères de préférence choisis par l'administrateur.

Etant donné un graphe d'attaques G, où I et F sont des ensembles d'états initiaux et ensemble d'états cible d'un attaquant, respectivement. Chaque action dans G est un exploit d'une vulnérabilité connue d'un réseau appliqué dans un certain état de réseau. Un ensemble de contre-mesures pour la vulnérabilité du réseau est indiqué avec CP-Net qui reflète la préférence de la variable de décision sur la gestion de sécurité. Cette analyse comporte quatre étapes principales :

1. Une extraction d'ensemble d'attaques qui identifie un ensemble minimal d'éléments de base dont chaque retrait garantit une protection de réseau basé sur un graphe donné,
2. Une identification de dépendances détermine la dépendance entre les attaques,
3. Une génération de contre-mesures qui permettent d'obtenir un ensemble de contre-mesures possibles qui sont minimalement requises pour protéger le réseau,
4. Les décisions de préférence identifient des décisions optimales en contre-mesures qui répondent le mieux à une préférence conditionnelle de données sur divers facteurs de décision.

A. Extraction d'ensemble d'attaques

Soit A-exp une expression booléenne qui représente tous les chemins d'attaques simples possibles (c'est-à-dire que chaque nœud sur le chemin est visité une fois) de n'importe quel état initial à n'importe quel état cible. En général, le produit des actions (exploits) dans l'expression booléenne représente une chaine d'attaques le long de la même trajectoire tandis que la somme dans l'expression représente les

branches des alternatives des chemins d'attaques. Donc A-exp est une somme des produits des actions. De ce point de vue, lorsque cela est approprié, les chemins d'attaques peuvent être considèrés comme des expressions booléennes auxquels des opérateurs booléens peuvent être appliqués avec interprétation conventionnelle. La première étape est de trouver un ensemble d'attaques (Remove), qui doit être enlevé afin d'empêcher les attaquants d'atteindre leurs objectifs. Chaque élément d'attaque à supprimer est représenté par son expression booléenne correspondante. Cela veut dire que pour sécuriser le réseau, il faut enlever les termes obtenus dans ∼A-exp.
La Figure 1.2 montre un échantillon d'un graphe d'attaques tel que, S_0 est l'état initial qui représente l'attaquant et S_3 est l'état final qui est la cible.

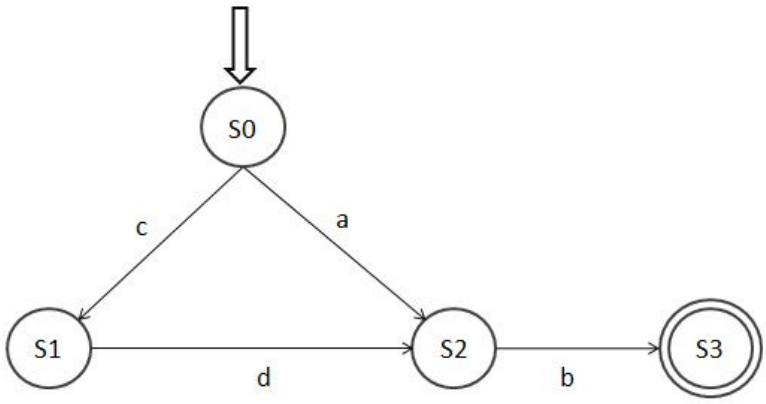

FIGURE 1.2 – Échantillon d'un graphe d'attaques [4].

Considérez un exemple d'un graphe d'attaques S_0 et S_3 sont des états, initial et final respectivement :
A-exp = ab + cdb ;
∼A-exp = ∼(ab + cdb) = (∼ab)(∼cdb) = (∼a + ∼b)(∼c + ∼d + ∼b) = (∼a∼c) + (∼a∼d) + (∼a∼b) + (∼b∼c) + (∼b∼d) + (∼b∼b) ;
Remove = (ac + ad + ab + bc + bd + b).

Cependant, les auteurs ont amélioré Remove en le réduisant à un ensemble

d'éléments de base appellé B-Remove. Chaque élément d'ensemble d'attaques Remove devra être enlevé pour sécuriser le réseau. Ainsi un élément est une base si aucun sous ensemble de cet élément n'appartient à aucun autre élément c'est-à-dire pour A,B $\in$ Remove et A $\subseteq$ B.
Remove = ac, ad, ab, bc, bd, b ; B-Remove = ac, ad, b.

B. Identification de dépendances

Cette section donne un mécanisme pour déterminer si deux attaques sont indépendantes en ce qui concerne un graphe d'attaques donné G. Cela peut être utile pour déterminer d'une façon rentable afin de sélectionner les contre-mesures pour sécuriser le réseau par exemple si l'attaque "a" dépend de l'attaque "b", les contre-mesures pour "b" doivent être capable d'empêcher toutes les deux, plutôt qu'acheter des contre-mesures pour "a" qui pourrais être chère que celui de "b", nous pouvons simplement utiliser une contre-mesures pour "b".
Une attaque "a" dépend d'une attaque "b" si un attaquant "a" ne peut pas effectuer "a" sans "b", autrement, "a" et "b" sont indépendants. Pour déterminer si "a" dépend de "b", les auteurs ont définis Reach(a) comme une expression booléenne qui représente tous les chemins, de tout état initial I dans G tel que "a" dépend de "b" si "b" est faux implique Reach(a) est fausse.

C. Génération des contre-mesures

Soit counter(a) un ensemble de contre-mesures pour l'attaque a et C = counter(a) tel que a est une attaque dans le graphe d'attaques G. Basé sur C et la dépendance des attaques obtenues dans la partie B, cette section fournit une approche à la production de A-counter (contre-attaques), un ensemble de contre-mesures qui sont imperceptiblement exigées pour sécuriser le réseau.
Exemple :
B-Remove= ac, ad, b. Définit des éléments de base pour sécuriser le réseau. Soit une expression représentant un ensemble des contre-mesures $C_1 + C_2$, C_3, $C_1 + C_4$ et $C_2 + C_4$ pour les attaques a, b, c et d respectivement.
La prévention de l'attaque ac exige un ensemble des contre-mesures qui peuvent

être exprimées par counter(a)counter(c)= $(C_1 + C_2)(C_1 + C_4)$.
counter(a)counter(c)= $(C_1 + C_2)(C_1 + C_4)$ = ($C_1\ C_1 + C_1\ C_4 + C_2\ C_1 + C_2\ C_4$) = ($C_1 + C_1\ C_4 + C_1\ C_2 + C_2\ C_4$).
Ceci donne un ensemble de contre-mesures pour ac comme $\{C_1,\ C_1\ C_4,\ C_1\ C_2,\ C_2\ C_4\}$, puisque la contre-mesure C_1 est aussi exigée pour $C_1\ C_2$ et $C_2\ C_4$ donc un ensemble principal des contre-mesures qui sont imperceptiblement exigées pour empêcher ac est $\{C_1,\ C_2\ C_4\}$, de même pour calculer les contre-mesures de ad et b qui résulte : counter(a)counter(d)= $\{C_1,\ C_2\}$, Counter(b)= $\{C_3\}$, et enfin la contre-mesure finale qui est A-counter $\{C_1, C_2\ C_4, C_2\ , C_3\}$= $\{\ C_1, C_2\ , C_3\}$.

D. Génération de préférences

Le choix de la contre-mesure la plus adéquate pour empêcher les attaquants d'atteindre leurs objectifs peut être subjective en particulier lorsque des décisions de compromis sont impliquées. CP-Net permet une représentation explicite de préférences de contre-mesures sur la base de valeurs de variables de décisions pertinentes.

L'analyse de Phongphun Kijsanyothin et al. [4] pose un problème de scalabilité de la génération automatique des graphes d'attaques d'où la restriction de son domaine d'application. De plus la suppression des chemins minimaux ce fait avec une génération de contre-mesures, le choix de cette dernière est faite avec CP-Net ors que l'administrateur devra savoir la conséquence de la suppression du chemin dans le fonctionnement du réseau.

1.4.2 Analyse en temps polynomial des graphes d'attaques

Feng chen et al. [1] ont proposé une méthode qui permet de calculer les chemins d'attaques sans boucle avec une distance donnée par les graphes d'attaques compacts. Leur approche permet d'analyser les graphes d'attaques pour défendre la sécurité du réseau en temps polynomial. Elle donne de bonne évolutivité pour les réseaux

d'entreprise avec des milliers d'hôtes, qui ont un graphe d'attaques complexes. Cette analyse se résume en trois étapes principales :

1. Calculer les chemins d'attaques n-valides,
2. Mesurer les risques de sécurité,
3. Chercher le coût minimum de la solution de durcissement de réseaux.

A. Calculer les chemins d'attaques n-valides

Dans cette section, nous discutons comment les auteurs obtiennent tous les chemins d'attaques n-valides employés par l'attaquant pour compromettre l'ensemble donné des conditions cruciales (C_f).

Propriété

Pour chaque nœud d'exploit (τ) $\in$ T , soit Pré (τ) l'ensemble de pré-conditions de τ et Post (τ) l'ensemble de post-conditions de τ, ($\wedge$L(C_i) $\rightarrow$ L(C_j)), où $C_i \in$ Pré (τ), $C_j \in$ Post (τ), qui montre que quand toutes les pré-conditions des exploits τ sont vraies, les post-conditions des exploits τ seront vraies et L est une fonction de l'étiquetage des conditions avec une proposition vraie [1].

B. Mesurer les risques de sécurité

Soit chemin (C_f) = {chemin(i), 1 $\leq$ i $\leq$ m} tous les chemins d'attaques vers Cf, le risque de sécurité de Cf dépend de trois facteurs. Le premier est le nombre de chemins d'attaques Cf , dénoté m, et plusieurs chemins d'attaques implique qu'il y a plus de chances pour que l'attaquant compromette des conditions cruciales au moyen de chemins d'attaques. Le second est la distance du chemin d'attaques chemin(i), dénoté l_i ($l_i \leq$ n), et Le troisième est le nombre de types d'exploits dans chemin (C_f) noté K, et plus d'exploits indique que les besoins de l'attaquant ont plus de connaissance sur différentes technologies d'exploits. Le risque de sécurité est défini par la formule suivante :

$$Risk = 1/K \times W + (1\text{-W}) \sum_{i=1}^{m} 1/l_i$$

W : est la probabilité de la résistance de la connaissance de l'attaquant.

C. Chercher le coût minimum de la solution de durcissement de réseaux

Le coût minimum de la solution de durcicement de réseau est un sous-ensemble de conditions initiales tel que, la suppression de ce dernier garantit qu'il n'y a pas de chemins d'attaques à la cible dans le graphe d'attaques.
Etant donné un graphe d'attaques AG = ($C_0 \cup C_r$, T, E,L) tel que, C_0 et C_r est un ensemble de conditions initiales et accessibles, respectivement, T est un un ensemble d'exploits, E est est un ensemble de côtés entre les nœuds (conditions ou exploits) et L est une fonction de l'étiquetage des conditions avec une proposition vraie. C_f c'est l'ensemble de conditions cruciales et Cost : $C_0 \rightarrow R^+$ [3]une fonction de coût de l'élimination des conditions initiales.
Pour chaque chemin d'attaques n-valides $chemin_k = \tau_1 \rightarrow \tau_2.....\tau_l$ soit le sous-ensemble de conditions initiales $C_0^k = (\cup \text{ pre}(\tau_i)) \cap \tau_0$ ou $1 \leq \text{i} \leq \text{l}$, alors la première condition initial dans le sous-ensemble C_0^k sont la cause fondamentale qui mène des attaquants afin d'obtenir l'objectif de l'attaque C_f par le chemin d'attaques $chemin_k$ et en supprimant toute condition initiale dans le sous-ensemble C_0^k peut désactiver cette voie d'attaques.

L'analyse de Feng chen et al. [1] se base sur les réseaux de petite taille qui pose un problème majeur pour la génération d'un graphe d'attaques. De plus cette technique est longue car le calcule de coût se fait en calculant un risque pour chaque vulnérabilité en utilisant des conditions crutiales initiales pour chaque chemin.

1.5 Conclusion

Dans ce chapitre nous avons présenté quelques travaux antérieurs qui nous ont permis de mieux comprendre l'utilisation des graphes d'attaques dans la sécurisation des réseaux.

3. intervalle de réel positif.

Chapitre 2

Une technique d'analyse des graphes d'attaques

Première partie

Proposition et son application

2.1 Introduction

N'importe quel réseau est vulnérable à des attaques afin de nuire au fonctionnement du système ou atteindre un autre objectif. Ces attaques se font en combinant les connaissances de l'attaquant avec les interdépendances complexes de la configuration d'un réseau. Notre but est de proposer une technique d'analyse des graphes d'attaques pour sécuriser le réseau informatique.

Ce chapitre est décomposé en deux parties, une exposant les étapes de notre proposition avec un exemple introductif, et l'autre présente une perspective sur le problème de satisfaction de contraintes.

2.2 Démarche proposée

Apres l'étude des travaux [1,4] nous avons proposé une technique qui tire profit des avantages des deux travaux sans en subir la majorité de leurs incovénients.
Cette proposition permet à l'administrateur de connaitre les différentes failles menaçant son système et lui donner le choix de la correction selon un coût qui lui convient.
La technique proposée utilise un graphe d'attaques déjà générée ayant tous les chemins possibles d'un attaquant vers une cible [3].
Elle se résume en trois étapes :

- ✓ Extraction d'ensemble d'attaques ;
- ✓ Identification de dépendances ;
- ✓ Chercher le coût des chemins.

2.2.1 Extraction d'ensemble d'attaques

Nous nous sommes inspiré de l'analyse de Phongphun Kijsanyothin et al. [4] pour aboutir à l'ensemble Remove qui est un ensembles minimal de vulnérabilités à enlever. Contrairement à l'analyse [4] nous avons défini deux étapes pour réduire B-Remove présenté ci-dessous.

1. Calculer la fermeture transitive [4] de Remove(i) ;
2. Si un élément de la fermeture transitive calculé appartient à l'ensemble Remove alors on supprime Remove(i) dans l'ensemble Remove.

L'algorithme 1 permet de concatener les chemins récupérés aprés la gérnération de graphe d'attaques puis d'appliquer une négation et une mutiplication afin d'obtenir un tableau de vulnérabilités minimales Remove. Aprés l'obtention de ce dernier une procédure sera appliquée pour calculer la fermeture transitive de chaque élément, puis faire un test pour vérifier si un élément appartient à l'ensemble Remove alors il sera supprimé, sinon il sera sauvegardé dans le B_Remove.

4. Etant donné un ensemble F de DF, d'autres DF peuvent en être la conséquence logique. Comme les DF sont en nombre fini (si $\cup$ est fini), l'ensemble des conséquences existe et s'appelle la fermeture transitive de F, notée F^+

Algorithm 1 Algorithme_Extraction_d'attaque

Entrée : Tab_chemin[] : Tableau[1..n] de chaîne de caractères ;
n : constante ;
chemin : chaine de caractere ;
Sortie : B_Remove [] : Tableau[1..n] de chaîne de caractères ;
Fonction Extraction_Ensemble_Attaque(Tab_chemin[]) :B_Remove []

1: **Pour** i **allant de** 1 **a** n **Faire**
2: chemin ← chemin + "+" + Tab_chemin[i] ;
3: **FinPour**
4: négation(chemin, Tab_Remove[]) ;
// négation est une procédure prenons les chemins, lui appliquant la négation d'une formule logique puis une décomposition et mettre le résultat dans Tab_Remove[].
5: **Pour** i **allant de** 1 **a** n **Faire**
6: **Si** (Tab_Remove[i] = '∼') et (Tab_Remove[i+1] = '+') **Alors**
7: Supprimer_remplacer(Tab_Remove[i], Tab_Remove[i+1], '*') ;
8: **FinSi**
9: **Si** (Tab_Remove[i] = '∼') et (Tab_Remove[i+1] = '*') **Alors**
10: Supprimer_remplacer(Tab_Remove[i], Tab_Remove[i+1], '+') ;
11: **FinSi**
12: **FinPour**
// Supprimer_remplacer est une procédure supprimons les deux cases et la remplaçant par une autre comportant le troisième élément en paramètre.
13: Multiplication(Tab_Remove[]) ;
// Multiplication est une procédure de distribution des vulnérabilités par rapport à la multiplications.
14: **Pour** i **allant de** 1 **a** n **Faire**
15: **Si** (Tab_Remove[i] = '∼') et (Tab_Remove[i+1] = '∼') **Alors**
16: Supprimer_remplacer(Tab_Remove[i], Tab_Remove[i+1], NULL) ;
17: **FinSi**
18: **FinPour**
19: **Pour** i **allant de** 1 **a** k **Faire**
// k est la taille du Tab_Remove[].
20: Fermeture_transitive(Tab_Remove[i], B_Remove[]) ;
// Fermeture_transitive est une procédure qui calcule la fermeture transitive du $i^{ème}$ élément, s'il n'y a pas d'inclusion alors il sera placé dans B_Remove[].
21: **FinPour**
22: **Retourner** (B_Remove[]) ;
=0

2.2.2 Identification de dépendances

Dans cette étape nous avons proposé d'utiliser les prés-conditions et les post-conditions vue que l'attaquant utilise des connaisances acquises dans le réseau pour se propager, contrairemant à l'étape d'identification de dépendances représentée dans l'analyse de Phongphun Kijsanyothin et al. [4] qui compare deux vulnerabilités et leurs généres des contre-mesures puis enlevent la plus petite, du coup la nécessité du temps pour son application. Dans notre proposition on supprime la plus antérieurs ce qui permet de minimiser l'ensemble des chemins.

L'algorithme 2 présenté ci-dessous permet de calculer toute les paires possible pour chaque composant de B_Remove, le test d'inclusion entre les vulnérabilités engendra la suppression de la vulnérabilité si l'inclusion est vérifiée.

Algorithm 2 Algorithme_Identification_dépendances

Entrée : B_Remove[] : Tableau[1..n] de chaîne de caractères ;
n : constante ;
ch : chaîne de caractères ;
Procedure Dependance(B_Remove[])

1: **Pour** i **allant de** 1 **a** n **Faire**
2: ch ← B_Remove[i] ;
3: Vul(B_Remove[i], Paire_vul[]) ;
4: **FinPour**
// Vul est une procédure retournant le tableau Paire_vul[] dont les possibilités de combinaison des vulnérabilités.
5: **Pour** j **allant de** 1 **a** k **Faire** pas 2
6: **Si** pré(Paire_vul[j]) ⊆ post(Paire_vul[j+1]) **Alors**
7: Supprimer(ch, B_Remove[]) ;
// Supprimer est une procédure qui supprime le ch dans B_Remove[].
8: **FinSi**
9: **FinPour**
=0

2.2.3 Chercher le coût des chemins

Cette étape nous permet de définir les coûts associés à la suppression d'un chemin composé d'un ensemble de vulnérabilités parmis l'ensemble des chemins obtenus dans le B-Remove de l'étape 3 (étape d'identification de dépendances) et cela se fait ainsi :
Tout d'abord on doit identifier le risque de sécurité noté R.
Nous avons trois facteurs à définir :

1. m : c'est le nombre de chemins d'attaques,
2. l_i : la distance de chemin d'attaques(i),
3. K : les différentes vulnérabilités composant les chemins d'attaques.

R est calculé par la formule définie par Feng chen et al. [1] :

$$R = 1/K \times W + (1\text{-W}) \sum_{i=1}^{m} 1/l_i$$

W : est la probabilité de la résistance de la connaissance de l'attaquant.

Algorithm 3 Algorithme_calculer_risque

Entrée : Tab_chemin[] : Tableau[1..n] de chaîne de caractères ;
h, m, w = 0.5, S=0.0 : réel ;
// m est la taille du Tab_chemin[] ;
Sortie : R : réel ;
Fonction Risque(Tab_chemin[]) : réel

1: K ← Differente_vulnerabilite(Tab_chemin[]) ;
   // Differente_vulnerabilite est une fonction retournont le nombre de vulnerabilité distinctes dans Tab_chemin[].
2: **Pour** j **allant de** 1 **a** m **Faire**
3: l_i ← Taille(Tab_chemin[i]) ;
   // Taille est une fonction retournant le nombre de vulnerabilités appartenant à un chemin donner.
4: S = S + $1/l_i$;
5: **FinPour**
6: R ← 1/K × W + (1 - W) × S ;
7: **Retourner** (R) ;
   =0

Ensuite afin de calculer le coût de chaque chemin nous avons défini la formule suivante :

$$\text{Coût}(chemin_i) = \sum_{j=1}^{n} D \times R \ / \ \text{Nb}$$

Tel que,

n : le nombre de vulnérabilités composant un chemin donner,

D : somme des degré des vulnérabilités [1] composant le chemin,

R : risque de sécurité,

Nb : le nombre des différentes vulnérabilités figurant dans les chemins.

1. l'ensemle des arcs liés à la vulnerabilités

Algorithm 4 Algorithme_Calculer_coût

Entrée : B_Remove[] : Tableau[1..n] de chaîne de caractères ; R, S=0.0, nb : réel ;

Sortie : C[] : tableau[1..n] de réel ;
Fonction cout(B_Remove[i], R) : C[]

1: nb ← Differente_vulnerabilite(B_Remove[i])
2: **Pour** j **allant de** 1 **a** n **Faire**
3: S ← Somme_Degres(B_Remove[i]) ;
// Somme_Degres est une fonction retournant le nombre des degrés de chaque vulnérabilité composant l'élément appartenant à B_Rmove[].
4: C[i] ← S × R / nb ;
5: **FinPour**
6: **Retourner** (C[]) ;
=0

2.3 Exemple illustratif

Nous nous somme inspiré de l'exemple utilisé par Feng chen et al. [1] dans lequel on a défini user(0) comme attaquant et user(2) étant une cible.

La figure 2.1 représente un exemple d'un graphe d'attaques avec user(0) et user(2) sont l'attaquant et la cible respectivement.

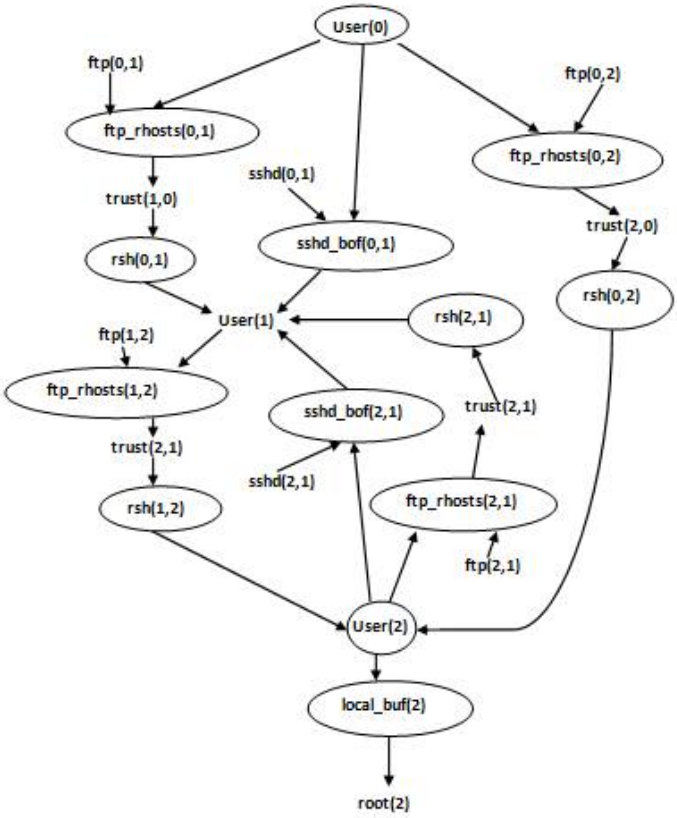

FIGURE 2.1 – Exemple d'un graphe d'attaques [1].

les chemins issues dans la génération de graphe d'attaques sont :
$chemin_1$ = ftp_rhost(0,2) * rsh(0,2).
$chemin_2$ = ftp_rhost(0,1) * rsh(0,1) * ftp_rhost(1,2) * rsh(1,2).
$chemin_3$ = sshd_bof(0,1) * ftp_rhost(1,2) * rsh(1,2).

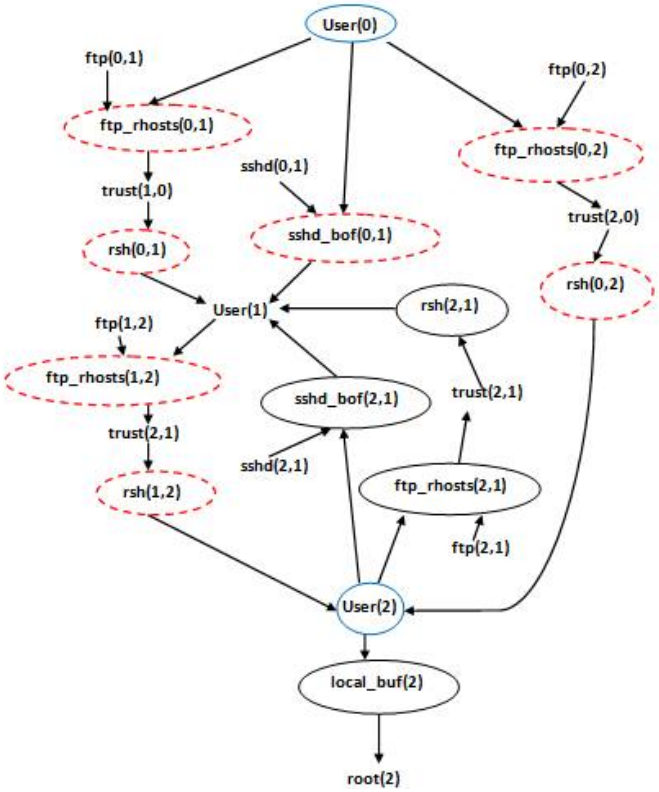

FIGURE 2.2 – Graphe d'attaques avec les chemins d'un attaquant vers une cible [1].

La figure 2.2 montre tous les chemins de l'attaquant user(0) vers la cible user(2). Cette dernière nous permet d'appliquer les étapes de notre proposition.

En appliquant l'étape d'éxtraction d'ensemble d'attaques on obtient :

A-exp = $chemin_1$ + $chemin_2$ + $chemin_3$.

A-exp = [ftp_rhost(0,2) * rsh(0,2)] + [ftp_rhost(0,1) * rsh(0,1) * ftp_rhost(1,2) * rsh(1,2)] + [sshd_bof(0,1) * ftp_rhost(1,2) * rsh(1,2)] .

Ensuite on applique un NON logique sur A-exp et on obtient :

$\sim$A-exp = $\sim$[[ftp_rhost(0,2) * rsh(0,2)] + [ftp_rhost(0,1) * rsh(0,1) *

ftp_rhost(1,2) * rsh(1,2)] + [sshd_bof(0,1) * ftp_rhost(1,2) * rsh(1,2)]] .

∼A-exp = ∼[ftp_rhost(0,2) * rsh(0,2)] * ∼[ftp_rhost(0,1) * rsh(0,1) *
ftp_rhost(1,2) * rsh(1,2)] * ∼[sshd_bof(0,1) * ftp_rhost(1,2) * rsh(1,2)] .

∼A-exp = [∼ftp_rhost(0,2) * ∼ftp_rhost(0,1) + ∼ftp_rhost(0,2) * ∼rsh(0,1) +
∼ftp_rhost(0,2) * ∼ftp_rhost(1,2) + ∼ftp_rhost(0,2) * ∼rsh(1,2) + ∼rsh(0,2) *
∼ftp_rhost(0,1) + ∼rsh(0,2) * ∼rsh(0,1) + ∼rsh(0,2) * ∼ftp_rhost(1,2) +
∼rsh(0,2) * ∼rsh(1,2)] * [∼sshd_bof(0,1) + ∼ftp_rhost(1,2) + ∼rsh(1,2)] .

∼A-exp = ∼ftp_rhost(0,2) * ∼ftp_rhost(0,1) * ∼sshd_bof(0,1) + ∼ftp_rhost(0,2) *
∼ftp_rhost(0,1) * ∼ftp_rhost(1,2) + ∼ftp_rhost(0,2) * ∼ftp_rhost(0,1) * ∼rsh(1,2)
+ ∼ftp_rhost(0,2) * ∼rsh(0,1) * ∼sshd_bof(0,1) + ∼ftp_rhost(0,2) * ∼rsh(0,1) *
∼ftp_rhost(1,2) + ∼ftp_rhost(0,2) * ∼rsh(0,1) * ∼rsh(1,2) + ∼ftp_rhost(0,2) *
∼ftp_rhost(1,2) * ∼sshd_bof(0,1) + ∼ftp_rhost(0,2) * ∼ftp_rhost(1,2) *
∼ftp_rhost(1,2) + ∼ftp_rhost(0,2) * ∼ftp_rhost(1,2) * ∼rsh(1,2) + ∼ftp_rhost(0,2)
* ∼rsh(1,2) * ∼sshd_bof(0,1) + ∼ftp_rhost(0,2) * ∼rsh(1,2) * ∼ftp_rhost(1,2) +
∼ftp_rhost(0,2) * ∼rsh(1,2) * ∼rsh(1,2) + ∼rsh(0,2) * ∼ftp_rhost(0,1) *
∼sshd_bof(0,1) + ∼rsh(0,2) * ∼ftp_rhost(0,1) * ∼ftp_rhost(1,2) + ∼rsh(0,2) *
∼ftp_rhost(0,1) * ∼rsh(1,2) + ∼rsh(0,2) * ∼rsh(0,1) * ∼sshd_bof(0,1) + ∼rsh(0,2)
* ∼rsh(0,1) * ∼ftp_rhost(1,2) + ∼rsh(0,2) * ∼rsh(0,1) * ∼rsh(1,2) + ∼rsh(0,2) *
∼ftp_rhost(1,2) * ∼sshd_bof(0,1) + ∼rsh(0,2) * ∼ftp_rhost(1,2) * ∼ftp_rhost(1,2) +
∼rsh(0,2) * ∼ftp_rhost(1,2) * ∼rsh(1,2) + ∼rsh(0,2) * ∼rsh(1,2) * ∼sshd_bof(0,1)
+ ∼rsh(0,2) * ∼rsh(1,2) * ∼ftp_rhost(1,2) + ∼rsh(0,2) * ∼rsh(1,2) * ∼rsh(1,2).

Les termes obtenus dans ∼A-exp sont les termes qui seront enlevés afin d'empêcher l'attaquant user(0) d'atteindre la cible user(2). On note :

Remove = { ftp_rhost(0,2) ftp_rhost(0,1) sshd_bof(0,1) , ftp_rhost(0,2) ftp_rhost(0,1) ftp_rhost(1,2) , ftp_rhost(0,2) ftp_rhost(0,1) rsh(1,2) , ftp_rhost(0,2) rsh(0,1) sshd_bof(0,1) , ftp_rhost(0,2) rsh(0,1) ftp_rhost(1,2) , ftp_rhost(0,2) rsh(0,1) rsh(1,2) , ftp_rhost(0,2) ftp_rhost(1,2) sshd_bof(0,1) , ftp_rhost(0,2) ftp_rhost(1,2) , ftp_rhost(0,2) ftp_rhost(1,2) rsh(1,2) , ftp_rhost(0,2) rsh(1,2) sshd-bof(0,1) , ftp-rhost(0,2) rsh(1,2) ftp-rhost(1,2) , ftp-rhost(0,2) rsh(1,2) , rsh(0,2) ftp-rhost(0,1) sshd-bof(0,1) , rsh(0,2) ftp_rhost(0,1) ftp_rhost(1,2) , rsh(0,2) ftp_rhost(0,1) rsh(1,2) , rsh(0,2) rsh(0,1) sshd_bof(0,1) , rsh(0,2) rsh(0,1) ftp_rhost(1,2) , rsh(0,2) rsh(0,1) rsh(1,2) , rsh(0,2) ftp_rhost(1,2) sshd_bof(0,1) , rsh(0,2) ftp_rhost(1,2) , rsh(0,2) ftp_rhost(1,2) rsh(1,2) , rsh(0,2) rsh(1,2) sshd_bof(0,1) , rsh(0,2) rsh(1,2) ftp_rhost(1,2) , rsh (0,2) rsh(1,2) rsh(1,2) }

L'ensemble Remove sera réduit en B-Remove en appliquant l'ensemble des étapes définies dans 2.3.2.
B-Remove={ ftp_rhost(0,2) ftp_rhost(0,1) sshd_bof(0,1) , ftp_rhost(0,2) rsh(0,1) sshd_bof(0,1) , rsh(0,2) ftp_rhost(0,1) sshd_bof(0,1) , rsh(0,2) rsh(0,1) sshd_bof(0,1) , rsh(0,2) ftp_rhost(0,1) rsh(1,2), ftp_rhost(0,2) ftp_rhost(1,2) , ftp_rhost(0,2) rsh(1,2) , rsh(0,2) ftp_rhost(1,2) , rsh(0,2) rsh(1,2) }

L'étape d'identification de dépendances consiste à identifier les dépendances entre l'ensemble des vulnérabilités obtenues dans l'étape 2 (étape d'extraction d'ensemble d'attaques).
Prenant par exemple rsh(0,2) ftp_rhost(0,1) rsh(1,2) dans le B-Remove et en appliquant l'algorithme défini dans 2.3.3 on obtient :
ftp-rhost(0,1) $\subseteq$ trust(1,0) $\subseteq$ ftp(1,2) $\subseteq$ trust(2,1) $\subseteq$ rsh(1,2) et dépendance = vrai d'où la suppression de ftp_rhost(0,1), ce qui nous mène à rsh(0,2) rsh(1,2), par suite vérifier si ses vulnérabilités existent déjà dans notre ensemble B-Remove ce qui est le cas, d'où sa suppression et la mise à jour de B-Remove.

B-Remove={ ftp_rhost(0,2) ftp_rhost(0,1) sshd_bof(0,1) , ftp_rhost(0,2) rsh(0,1) sshd_bof(0,1) , rsh(0,2) ftp_rhost(0,1) sshd_bof(0,1) , rsh(0,2) rsh(0,1) sshd_bof(0,1) , ftp_rhost(0,2) ftp_rhost(1,2) , ftp_rhost(0,2) rsh(1,2) , rsh(0,2) ftp_rhost(1,2) , rsh(0,2) rsh(1,2) }

Après l'étape d'identification de dépendances il faut chercher le coût associé à chaque chemins afin de le supprimer en choisissant la suppression du chemin qui a le coût minimum.Prenant l'ensemble B-Remove de l'étape 3 (étape d'identification des dépendances) :
$chemin_1$ = ftp_rhost(0,2) ftp_rhost(0,1) sshd_bof(0,1).
$chemin_2$ = ftp_rhost(0,2) rsh(0,1) sshd_bof(0,1).
$chemin_3$ = rsh(0,2) ftp_rhost(0,1) sshd_bof(0,1).
$chemin_4$ = rsh(0,2) rsh(0,1) sshd_bof(0,1).
$chemin_5$ = ftp_rhost(0,2) ftp_rhost(1,2).
$chemin_6$ = ftp_rhost(0,2) rsh(1,2).
$chemin_7$ = rsh(0,2) ftp_rhost(1,2).
$chemin_8$ = rsh(0,2) rsh(1,2).

Le risque de sécurité R on le calcule à partir de la formule définie dans 2.3.4

$l_1 = 2$; $l_2 = 4$; $l_3 = 3$; m = 3 ; K = 3 ; W = 0.5.

$$R = 1/3 \times 0.5 + (1\text{-}0.5) \times (1/2 + 1/4 + 1/3) = 0.71$$

Le degré de chaque vulnérabilité est mentionné dans le tableau suivant :

Les différentes vulnérabilités	Nombre d'arcs liées
ftp_rhost(0,2)	2
ftp_rhost(0,1)	2
ftp_rhost(1,2)	2
rsh(0,1)	2
rsh(1,2)	3
rsh(0,2)	2
sshd_boff(0,1)	2

Nombre de vulnérabilités est : 7.
En appliquant la formule du calcul des coûts définie dans 2.3.4 on obtient :
$chemin_1$ = (2+2+2) * 0.71/7 = 0.608 .

$chemin_2$ = (2+2+2) * 0.71/7 = 0.608 .
$chemin_3$ = (2+2+2) * 0.71/7 = 0.608 .
$chemin_4$ = (2+2+2) * 0.71/7 = 0.608 .
$chemin_5$ = (2+2) * 0.71/7 = 0.405 .
$chemin_6$ = (2+3) * 0.71 /7 = 0.507 .
$chemin_7$ = (2+2) * 0.71/7 = 0.405 .
$chemin_8$ = (2+3) * 0.71/7 = 0.507 .
Dans cette exemple les meilleurs chemins à traiter sont les $chemin_5$ et $chemin_7$.

Deuxième partie

Perspective sur le Problème de satisfaction de contraintes (CSP)

2.4 Définitions

2.4.1 CSP

Les problèmes de satisfaction de contraintes sont des problèmes mathématiques où l'on recherche des états ou des objets satisfaisant un certains nombre de contraintes.

2.4.2 Contraintes

Une contrainte est une relation entre différentes variables. Cette relation peut être défini en extension ou en intension [10].

- *Extension :* énumérant les tuples de valeur appartenant à la relation.
- *Intension :* utilisant des propriétés mathématiques connues.

2.4.3 Définition formelle

Un CSP est défini par un triplet <X, D, C> tel que [10] :

- X = { X_1, X_2, ..., X_n} qui est un ensemble des variables (les inconnues) du problème ;
- D est la fonction qui associe à chaque variable X_i son domaine D(X_i), c'est-à-dire l'ensemble des valeurs que peut prendre X_i
- C = {C_1, C_2, ..., C_k} est l'ensemble des contraintes. Chaque contrainte C_j est une relation entre certaines variables de X, restreignant les valeurs que peuvent prendre simultanément ces variables [10].

2.5 Fonctionnement du CSP

Consiste à affecter des valeurs aux variables à condition de satisfaire toutes les contraintes défini dans C. Cette solution se base sur la consistance des nœuds ou des arcs.

Consistance des nœuds : Elle consiste à ne considérer qu'une variable. On retire alors

toute les valeurs pour lesquelles au moins une contrainte est impossible à satisfaire. Ce filtrage est très rapide.

Consistance d'arcs : La propagation de contraintes consiste à modifier le domaine des variable impliquées dans une contrainte dans le but d'établir la consistance.

- consistance : c'est une propriété liée à la compatibilité entre les valeurs de domaine et les contraintes.
- filtrage : Elimination d'éléments dont on est sûr qu'ils ne peuvent figurer dans une solution. Le filtrage a deux propriétés :
  - simplification du problème par réduction de l'espace de recherche.
  - dans certains cas détection de l'incohérence.

Inconsistance d'arcs : Pour résumer, on cherche à supprimer des valeurs qui ne mènent à aucune solution.

La figure 2.3 représente la différence entre les arcs consistants et les arcs inconsistants Exemple :

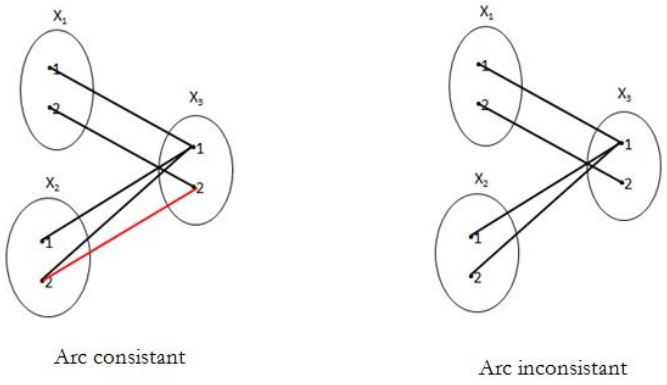

FIGURE 2.3 – Arcs consistants et inconsistants.

2.6 Applicabilité du CSP

Notre analyse est améliorable de telle sorte à avoir une solution plus rapide et offrant à l'administrateur de meilleurs choix afin de contrôler et de sécuriser son réseau prenant en compte le coût.
Nous avons pensé à remplacer les deux étapes extraction d'ensemble d'attaques et identification de dépendances par CSP qui offre à son tour deux choix, soit une solution par rapport au nœuds qui est plus simple et rapide avec un coût abordable et un autre choix qui fait l'étude par rapport aux liens, reliant l'attaquant et la cible ou bien d'un autre terme développer un outil d'intelligence artificiel, afin d'analyser et de sécuriser le réseau.
L'ensemble de ces idée peuvent être utilisées pour une éventuelle amélioration et reste un point ouvert pour un nouveau sujet de recherche.

2.7 Conclusion

Dans ce chapitre nous avons expliqué les trois étapes de notre proposition munie d'un exemple illustratif afin de mieux comprendre ces étapes, ainsi qu'une perspective sur l'applicabilité du CSP, sur l'analyse du graphe d'attaques, enfin il nous reste qu'à développer notre proposition pour servir et protéger les réseaux informatiques.

Chapitre 3

Réalisation

3.1 Introduction

Ce chapitre est la dernière étape de notre projet, qui est consacré à démontrer le bon fonctionnement de notre proposition à travers une application de cette dernière en utilisant des graphes d'attaques générés.

3.2 Environnement du développement de l'application

Pour développer notre application, nous avons eu recours aux outils suivants :

3.2.1 Le langage Java

Java est un langage de programmation et une plate-forme informatique, Il a la particularité principale que les logiciels écrits avec ce dernier sont très facilement portables sur plusieurs systèmes d'exploitation tels qu'Unix, Microsoft Windows, Mac OS ou Linux. Java est rapide, sécurisé et fiable, il permet de développer, de déployer et d'utiliser efficacement des applications et des services fascinants.

3.2.2 Eclipse

Eclipse est un environnement de développement (IDE) historiquement destiné au langage Java, même si grâce à un système de plugins il peut également être utilisé avec d'autres langages de programmation, dont le C/C++ et le PHP. Il nécessite une machine virtuelle Java (JRE) pour fonctionner. Mais pour compiler du code Java, un kit de développement (JDK) est indispensable.

3.2.3 API extern JGraph

Une librairie externe Swing utilisée pour le dessin du graphe d'attaques.

3.2.4 API extern JDom

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.

3.3 Description des interfaces

Notre application contient deux interfaces dont la première a un espace de travail pour afficher le graphe d'attaques et trois boutons, le premier boutons « Importer » qui permet d'importer un graphe d'attaques déjà généré à partir d'un fichier XML et de l'afficher dans le panneau centre, le bouton « Analyse » permettant d'analyser le graphe d'attaques importé, ainsi ouvrant une autre fenêtre affichant tous les chemins possibles à supprimer et leurs coûts associés. Pour chaque chemin, on trouve un bouton « supprimer », afin de supprimer le chemin sur le graphe d'attaques qui à son tours s'affichera dans la première interface dans le panneau centre et le troisieme bouton « Reinitialiser » permet de réinitialiser l'execution à nouveau.
Les figures ci-après permettent de comprendre cette description :
La figure 3.1 présente la fenêtre pricipale de travail

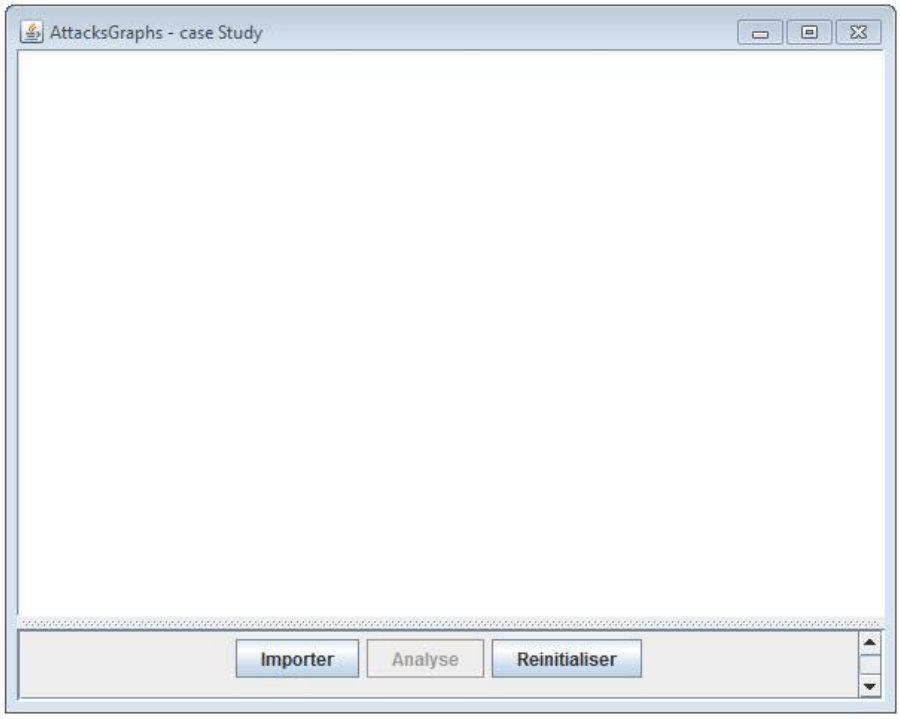

FIGURE 3.1 – Fenêtre principale.

Une boite de dialogue s'affichera après avoir cliquer sur le bouton « Importer » afin d'importer le fichier XML tel que illustré dans la Figure 3.2.

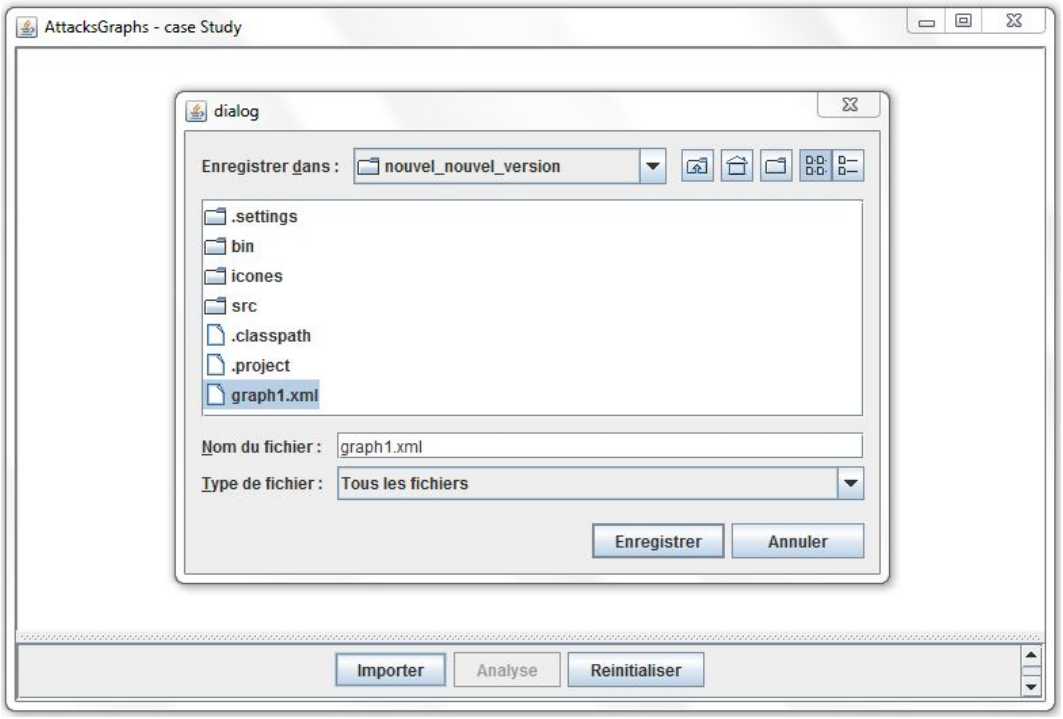

FIGURE 3.2 – Importer le graphe d'attaques.

Après l'importation du graphe on pourra l'analyser en cliquant sur le bouton « Analyse », une autre interface va apparaître dont les vulnérabilités minimales et les coûts associés seront affichés

FIGURE 3.3 – Fenêtre d'analyse.

En choisissant l'un des boutons supprimer ça engendrera la suppression des vulnérabilités équivalentes et le nouveau graphe sera afficher dans la fenêtre princale.

3.4 Exemple Applicatif

L'application utilise un graphe d'attaques importé à partir d'un fichier XML. La figure suivante montre le graphe d'attaques après l'importation, tel que user(0) est l'attaquant et user(2) est la cible.

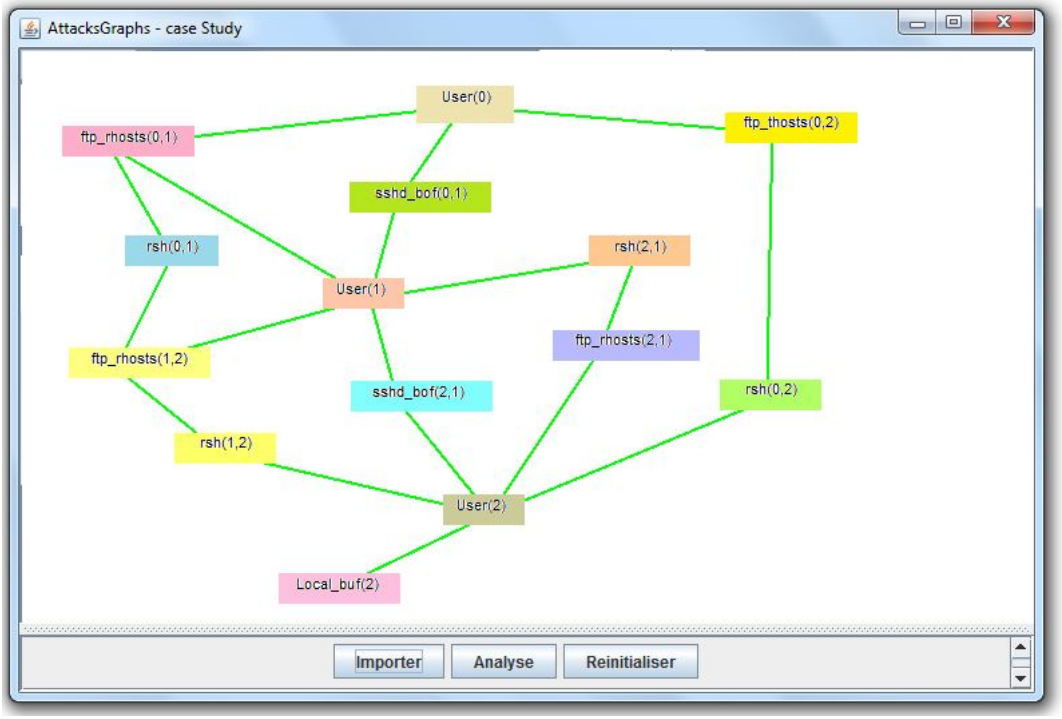

FIGURE 3.4 – Graphe d'attaques importer.

Après l'analyse de graphe les chemins et les coûts seront affichés dans la fenêtre illustrer par la Figure 3.5.

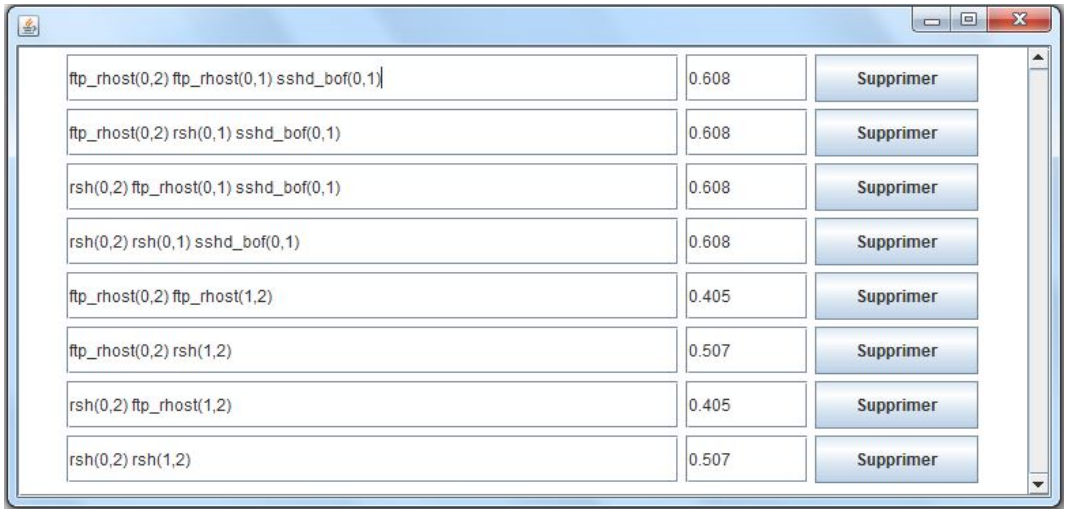

FIGURE 3.5 – Les chemins et les coûts associés.

Par exemple si nous choissisons de supprimer le cinquième chemin, les vulnérabilités ftp_rhost(0,2) ftp_rhost(1,2) seront supprimées et le nouveau graphe est présenté par la Figure 3.6.

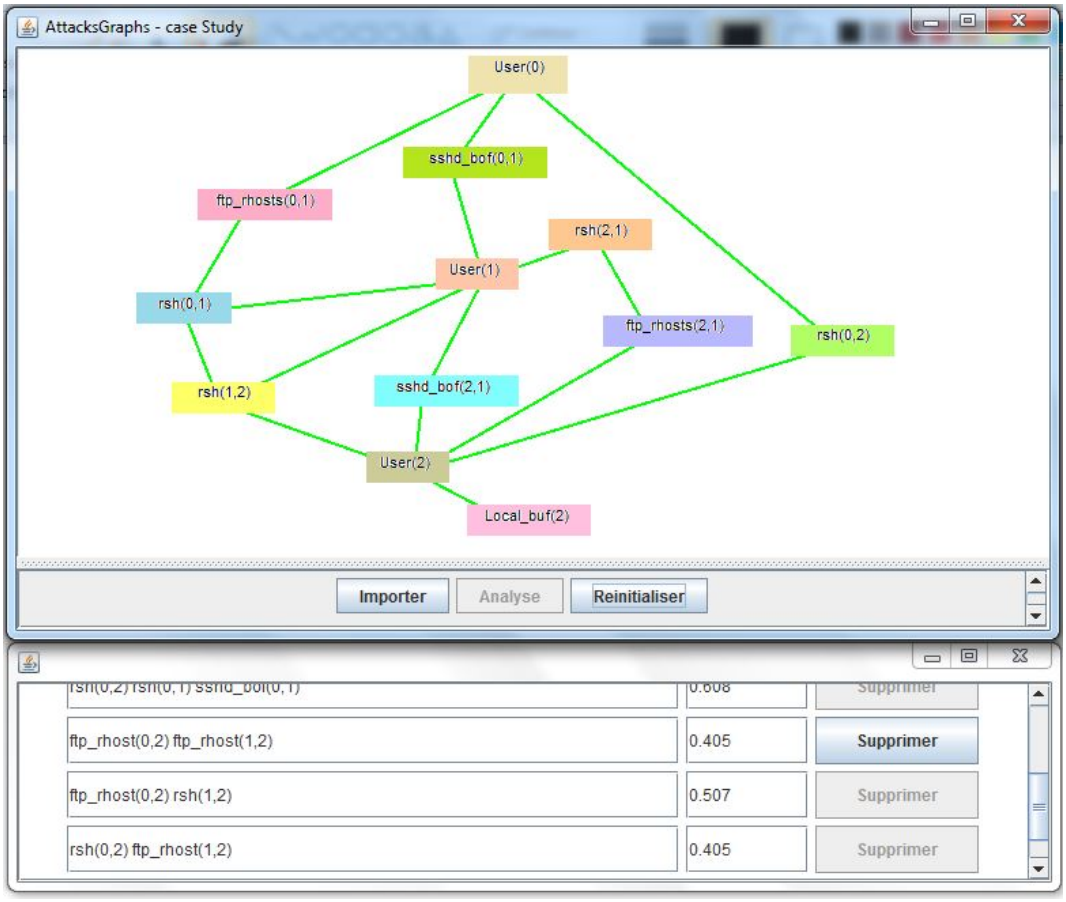

FIGURE 3.6 – Graphe d'attaques aprés la suppression du cinquième chemin.

Prenant un autre exemple : supprimer le septième chemin, les vulnérabilités rsh(0,2) ftp_rhost(1,2) seront supprimées et un nouveau graphe est présenté par la Figure 4.6.

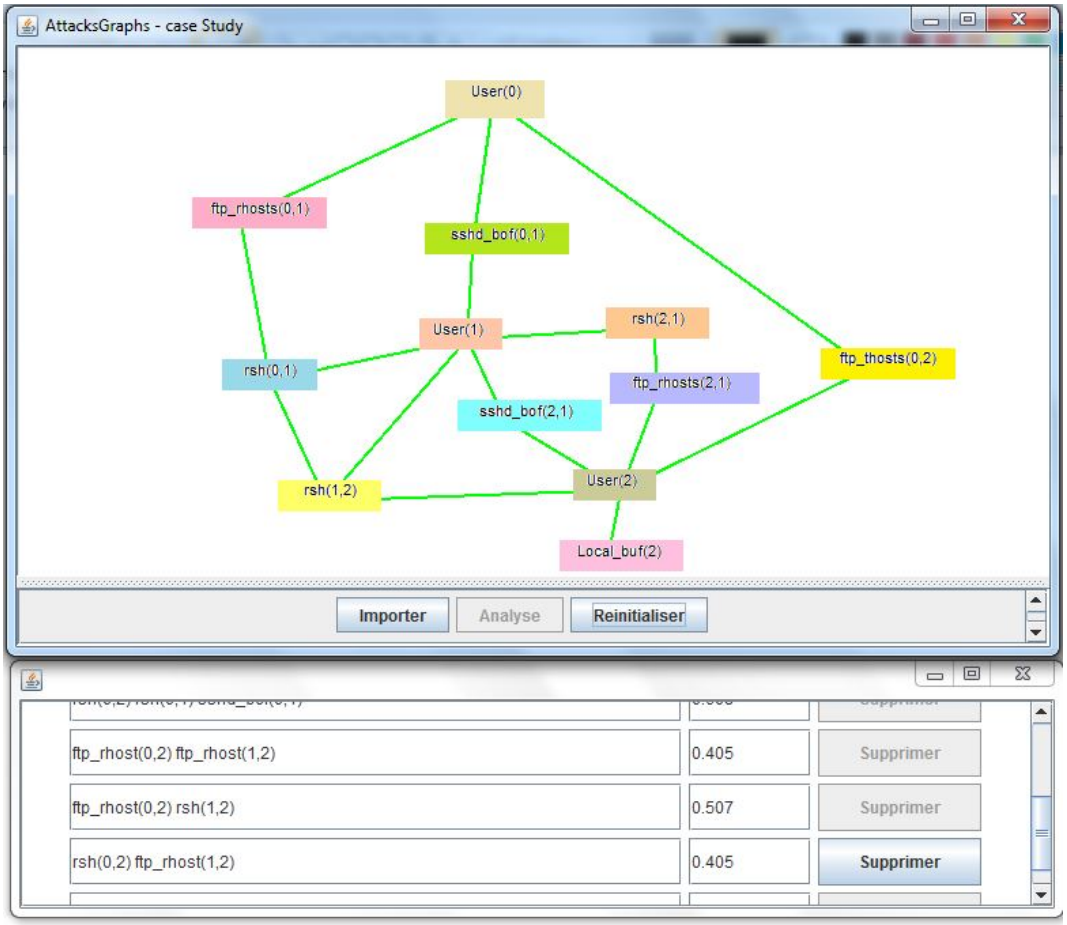

FIGURE 3.7 – Graphe d'attaques aprés la suppression du septième chemin.

3.5 Conclusion

Dans ce chapitre, nous avons expliqué le fonctionnement de notre application, en utilisant un exemple d'un graphe d'attaques généré pour montrer l'utilisation de notre proposition dans l'analyse des graphes d'attaques.

Conclusion générale

L'informatique pénètre peu à peu tous les domaines de notre vie quotidienne. Que ce soit les magasins, les distributeurs, les écoles, les bibliothèques, etc. De ce fait la protection des réseaux informatiques est primordiale, pour assurer les objectifs de la sécurité tel que, l'intégrité et la confidentialité des données informatiques.

Beaucoup de recherches sont effectuées afin de sécuriser et protéger les réseaux informatiques tel que l'utilisation de graphes d'attaques pour détecter les vulnérabilités de ces réseaux, ces derniers sont d'abord générés et ensuite analysés.

L'analyse des graphes d'attaques a moins d'attention comparée à la génération des graphes d'attaques, une synthèse des travaux déjà réalisés a été faite afin d'introduire et mieux comprendre le domaine d'analyse des graphes d'attaques, engendrant le renforcement de la sécurité des réseaux. Cependant, une technique d'analyse est proposée permettant d'analyser n'importe quel graphe d'attaques déjà généré.

Enfin nous avons réalisé une application renforçant notre proposition à l'aide des outils de développement tel que le langage Java, Eclipse et API extern JGraph, etc.

Bibliographie

Bibliographie

[1] F. Chen, R. Tu, Y. Zhang and J. Su, A Scalable Approach to Analyzing Network Security using Compact Attack Graphs, IEEC '09. International Symposium on Information Engineering and Electronic Commerce, pp. 90 - 94, 2009.

[2] K.Wei Lye and J.Wing, Game Strategies in network security, International Journal of Information Security, pp. 71-86, 2005.

[3] L.Hamza and K.Adi, Formal Technique for Discovering Complex Attacks in Computer Systems, In Proceedings of the 2007 conference on New Trends in Software Methodologies, pp. 185-199, 2007.

[4] P.Kijsanyothin and R.Hewett, Analytical Approach to Attack Graph Analysis for Network Security, ARES 10 International Conference on Availability, Reliability and Security, pp. 25-32, 2010.

[5] S.Noel and S.Jajodia, Managing attack graph complexity through visual hierarchical aggragation, In VizSEC/DMSEC'04 :Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pp. 108-118, 2004.

[6] S.Noel and S.Jajodia, Understanding complex network attack graph -through clustered adjacency matrices, In ACSAC'05 : Proceeding of the 21st annual computer security applications conference, pp. 160-169, 2005.

[7] K.Tariki and A.Rili, Analyse des graphes d'attaques, Mémoire de Master en informatique, Université de Béjaia, 2015.

[8] V.Mehta, C.Bartzis, H.Zhu,E.Clarke and J.Wing, Ranking Attack Graphs, in Recent Advances in Intrusion Detection, pp. 127-144, 2006.

Webographie

[9] http ://liris.cnrs.fr/csolnon/Site-PPC/e-miage-ppc-som.htm, visiter le 21-04-2016.

[10] http ://pr.efactory.de/e-pagerank-algorithm.shtml, visiter le 21-04-2016.

[11] http ://www-igm.univ-mlv.fr/ dr/XPOSE2009/Nessus/, visiter le 21-04-2016.

RÉSUMÉ

La notion de risque et de la menace, liée aux réseaux ainsi qu'aux systèmes informatiques, devient une source d'inquiétude et une donnée importante à prendre en compte afin de mieux sécuriser les systèmes.
Dans ce mémoire, nous avons défini une technique d'analyse des graphes d'attaques dans le but de protéger les réseaux et les systèmes informatiques contre tous les risques et les menaces qui peuvent toucher ces derniers.

Mots clés : Graphe d'attaque, Scénario d'attaque, CP-Net, CSP.

ABSTRACT

The notion of the risk and the threat, bound to networks as well as to computer systems, becomes a cause for concern and a datum important to take into account to secure better the systems.
In this report, we defined a technique of analysis of the graphs of attacks with the aim of protecting networks and computer system against all the risks and the threats which can touch the latter.

Key words : Graph attacks, Attack scenario, CP-Net, CSP.