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

MEMOIRE DE MASTER RECHERCHE

En
Informatique

Option
Intelligence Artificielle

Thème

Application des alliances pour l'analyse des graphes d'attaques

Présenté par : M. YAHIAOUI Rayan

Présenté le devant le jury composé de :

Président Pr H. SLIMANI Professeur U. A/Mira Béjaia.
Examinateur Dr M. MOHAMMEDI M.C.A U. A/Mira Béjaia.
Encadreur Dr K. OUAZINE M.C.B ESTIN Amizour Béjaia.
Co-encadreur Dr L. HAMZA M.C.A U. A/Mira Béjaia.

Béjaia, Septembre 2022.

⁂ *Remerciements* ⁂

A l'issue de ce travail, je tiens à remercier mon encadrante Dr OUAZINE Kahina et ma co-encadrante Dr HAMZA Lamia pour l'honneur qu'elles m'ont fait en acceptant de m'encadrer, pour la qualité de leur encadrement, et pour leur patience inépuisable. Qu'elles trouvent ici l'expression de ma profonde reconnaissance.
Je tiens également à remercier les membres du jury pour avoir accepté de juger ce travail :
Pr. SLIMANI Hachem pour l'honneur qu'il m'a fait en acceptant de présider le jury de ce mémoire.
Dr MOHAMMEDI Mohamed qui m'a fait l'honneur de faire partie du jury en tant qu'examinateur.
Je tiens à remercier tous les enseignants qui m'ont accompagné tout au long de ma formation, pour leur soutien et pour leur confiance.

⁕ *Dédicaces* ⁕

A mes très chers parents.
A mes deux soeurs.
A tous mes amis.

M. ***YAHIAOUI Rayan***

Table des matières

Table des figures

Liste des tableaux

Notations et symboles

	∂S	Frontière du sous-ensemble S.
	μ	Connectivité algébrique
	μ_*	Rayon spectral Laplacian
	a^d	Cardinal minimum d'une alliance défensive.
	$\hat{a}^d$	Cardinal minimum d'une alliance défensive forte.
	γ_a^d	Cardinal minimum d'une alliance défensive globale.
	$\gamma_{\hat{a}}^d$	Cardinal minimum d'une alliance défensive globale forte.
	a_k^d	Cardinalité minimale d'une k-alliance défensive.
	γ_k^d	Cardinalité minimale d'une k-alliance défensive globale.
	a^o	Cardinal minimum d'une alliance offensive.
	$\hat{a}^o$	Cardinal minimum d'une alliance offensive forte.
	γ_a^o	Cardinal minimum d'une alliance offensive globale.
	$\gamma_{\hat{a}}^o$	Cardinal minimum d'une alliance offensive globale forte.
	a_k^o	Cardinalité minimale d'une k-alliance offensive.
	γ_k^o	Cardinalité minimale d'une k-alliance offensive globale.
	a^p	Cardinal minimum d'une alliance puissante.
	$\hat{a}^p$	Cardinal minimum d'une alliance puissante forte.
	γ_a^p	Cardinal minimum d'une alliance puissante globale.
	$\gamma_{\hat{a}}^p$	Cardinal minimum d'une alliance puissante globale forte.
	a_k^p	Cardinalité minimale d'une k-alliance puissante.
	γ_k^p	Cardinalité minimale d'une k-alliance puissante globale.
C	C_n	Graphe cycle
D	$d_S(v)$	Degré d'un sommet v dans S.
E	E	Ensemble d'arêtes d'un graphe G.
G	$G = (V, E)$	Graphe sans boucles ou arêtes multiples.
K	K_n	Graphe complet
	$K_{r,s}$	Graphe biparti complet
N	$N(v)$	Voisinage ouvert du sommet v.
	$N_S(v)$	Voisinage ouvert du sommet v appartenant au sous ensemble S.
	$N[v]$	Voisinage fermé du sommet v.
	$N_S[v]$	Voisinage fermé du sommet v appartenant au sous ensemble S.
P	P_n	Chaine
S	$< S >$	Sous-graphe du graphe G induit par le sous-ensemble $S \subseteq V$.
	$\overline{S} = V - S$	Complémentaire de S dans V.
V	V	Ensemble de sommets d'un graphe G.

Introduction générale

Suite au besoin grandissant d'ouverture des systèmes informatiques à Internet, ses systèmes sont devenus de plus en plus inter-connectés les uns aux autres. De nouvelles vulnérabilités voient donc le jour, et de nouvelles attaques exploitant ces vulnérabilités apparaissent régulièrement et de façon continue[7]. C'est pour cela qu'il ne suffit plus de se défendre contre telle ou telle vulnérabilité, mais de prendre en compte l'interdépendance des systèmes et d'avoir une vue générale sur leurs failles afin de pouvoir 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.
D'un autre coté, depuis leur introduction en 2002 [22], le concept d'alliance dans les graphes a connu un essor croissant, allant d'un concept théorique dans les graphes à leur utilisation dans bien d'autres domaines pratiques comme le clustering ou la bioinformatique.
Dans ce projet de fin de cycle nous abordons le problème d'analyse des graphes d'attaques, et essayant d'intégrer le concept des alliances dans les graphes d'attaques.
Notre mémoire est subdivisé en trois chapitres principaux :

Dans le premier chapitre intitulé "Introduction aux alliances dans les graphes", nous présentons quelques concepts de base des alliances dans les graphes, leurs différents types et leurs généralisations. On termine le chapitre en abordant les différents domaines d'applications des alliances.

Le deuxième chapitre présente un état de l'art sur l'analyse des graphes d'attaques et sur les alliances. Nous commençons par quelques définitions préliminaires, ensuite nous présentons différents travaux antérieurs. Enfin on termine par une synthèse d'analyse de ces travaux.

Le troisième chapitre est consacré à la présentation de notre proposition. Ce chapitre commence d'abord par définir la proposition, puis nous présentons un algorithme qui renvoie les alliances défensives dans un graphe d'attaques, par la suite nous comparons les différentes alliances défensives et ressortons les vulnérabilités communes. Pour finir, nous présentons un exemple de

motivation.

Enfin, le mémoire se termine par une conclusion et quelques perspectives en ce qui concerne l'avenir des alliances dans les graphes d'attaques.

Chapitre 1

Introduction aux alliances dans les graphes

1.1 Introduction

Dans la vie de tous les jours, une alliance est une collection d'entités telles que l'union est plus forte que l'individu. Une alliance peut être faite pour se protéger contre une attaque, ou pour affirmer la volonté collective. Dans ce chapitre, nous présentons les différents types d'alliances avec les paramètres principaux associés.

1.2 Alliances et graphes

En théorie des graphes, une alliance est un sous-ensemble non vide d'entités tel qu'il assure, pour chacune d'entre elles, certaines propriétés sur le nombre d'entités voisines qui sont dans l'alliance et celles qui n'y sont pas. Le concept a été introduit par Kristiansen et al.[22], où ils ont défini trois types d'alliances qu'ils ont appelées alliances défensives, alliances offensives et alliances puissantes. Nous commençons par énoncer la terminologie utilisée :
Soit $G = (V, E)$ un graphe simple (où V représente l'ensemble des sommets et E l'ensemble des arêtes).
Pour tout sommet v de G, le voisinage ouvert de v est noté $N_G(v) = \{u \in V(G) : uv \in E(G)\}$, et le voisinage fermé de v est défini par $N_G[v] = N_G(v) \cup \{v\}$. Pour un sous-ensemble $S \subseteq V(G)$, le voisinage ouvert de v dans S est noté par $N_S(v) = \bigcup_{v \in S} N_G(v)$ et le voisinage fermé de v dans S est $N_S[v] = \bigcup_{v \in S} N_G[v]$. Parfois pour alléger les notations et lorsqu'il n'y a aucune confusion sur le graphe G, les voisinages ouverts et fermés d'un sommet v seront notés simplement par $N(v)$ et $N[v]$ à la place de $N_G(v)$ et $N_G[v]$; respectivement. De même pour $N(S)$ et $N[S]$ au lieu de $N_S(v)$ et $N_S[v]$ respectivement. L'ensemble $\partial S = N[S] - S$ est la frontière de S. Le sous-graphe induit par S, c'est à dire le sous-graphe $(S, E \cap (S \times S))$, est noté $< S >$, et le complément de S dans V est noté $\overline{S}$. Nous rappelons qu'un ensemble $S \subset V$est un ensemble dominant dans G si, pour

chaque sommet $x \in \overline{S}$, $d_S(x) > 0$ (chaque sommet de $\overline{S}$ est adjacent à au moins un sommet de S).

Définition 1.2.1. (Matrice d'adjacence)[16] : Étant donné un graphe simple $G = (V, E)$ avec $|V| = n$, sa matrice d'adjacence $A(G)$ est une matrice binaire $n \times n$ où l'entrée $a_{i,j}$ est égale à 1 si $i, j \in E$ et 0 sinon.

Définition 1.2.2. (Matrice des degrés)[16] : Étant donné un graphe simple $G = (V, E)$ avec $|V| = n$, sa matrice de degré $D(G)$ est une matrice diagonale $n \times n$ où l'entrée d_{ii} est égale au degré du sommet i.

Définition 1.2.3. Étant donné un graphe simple $G = (V, E)$ avec $|V| = n$, sa matrice Laplacienne $L(G)$ est une matrice $n \times n$ définie comme suit :

$$L(G) = D(G) - A(G) \tag{1.1}$$

Définition 1.2.4. La connectivité algébrique μ [16] d'un graphe G est la deuxième plus petite valeur propre (en comptant plusieurs valeurs propres séparément) de la matrice Laplacienne de G. Cette valeur propre est supérieure à 0 si et seulement si G est un graphe connexe. Ceci est un corollaire du fait que le nombre de fois 0 apparaît comme valeur propre dans le Laplacien est le nombre de composantes connexes dans le graphe. L'amplitude de cette valeur reflète à quel point le graphe global est bien connecté.

1.2.1 Alliances défensives

Nous commençons par définir les alliances défensives et leurs paramètres associés [22] :

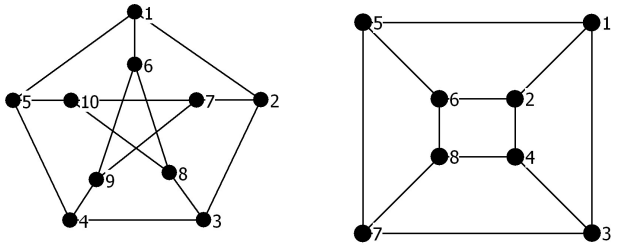
Définition 1.2.5. Une alliance est dite défensive si, pour chaque membre de l'alliance, il y a au moins autant de voisins alliés que de voisins non alliés.

Définition 1.2.6. Une alliance défensive est un sous-ensemble S de V tel que $x \in$ S implique $|N[x] \cap S| \geq |N[x] - S|$. On peut considérer les sommets de $N[x] - S$ comme des attaquants de x et ceux de $N[x] \cap S$ comme des défenseurs de x. Ainsi, pour tout x d'une alliance défensive, il y a au moins autant de défenseurs que d'attaquants, et toute attaque sur un seul sommet peut être contrecarrée. Autrement dit : Un ensemble non vide de sommets $S \subseteq V$ est une alliance défensive, si : $\forall v \in S, |N_S[v]| \geq |N_{\overline{S}}(v)|$.Si l'inégalité est stricte, nous appelerons ça une alliance défensive forte.

Définition 1.2.7. Le nombre d'alliance défensive $a^d(G)$ (respectivement le nombre d'alliance défensive forte $\hat{a}^d(G)$) est le cardinal minimum d'une alliance défensive (respectivement d'une alliance défensive forte). Une alliance défensive S est dite globale si S est un ensemble dominant.

Définition 1.2.8. Le nombre d'alliance défensive globale γ_a^d(respectivement, le nombre d'alliance défensive globale forte $\gamma_{\hat{a}}^d$) est le cardinal minimum d'une alliance défensive globale (respectivement, d'une alliance défensive globale forte).

Exemple 1.2.1. Dans le graphe G de la Figure 1.1. L'ensemble $S = \{a, b\}$ est une alliance défensive globale du graphe G, d'où $\gamma_a^d(K_{1,3})= 2$ et $S = \{a, b, c\}$ est une alliance défensive globale forte de G, d'où $\gamma_{\hat{a}}^d(G) = 3$:



(a) $S = \{3, 6, 8, 10\}$ est une alliance défensive globale forte.

(b) $S = \{2, 3, 4, 5, 6\}$ est une alliance défensive forte

FIGURE 1.1 – Exemples d'alliances défensives dans les graphes[18].

1.2.2 Alliances Offensives

Un autre type d'alliances appelés alliances offensives ont aussi été introduite par Kristiansen et al.[22] :

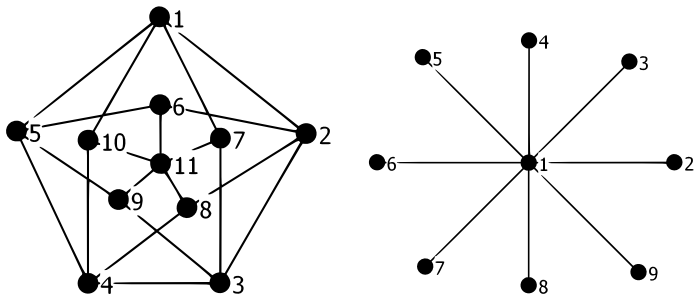
Définition 1.2.9. Un ensemble S est une alliance offensive si, pour chaque sommet v dans sa frontière $\partial S = N(S) - S$, il est certain que la majorité des sommets du voisinage fermé de v sont dans S. Un ensemble non vide de sommets $S \subseteq V$ est dit une alliance offensive si et seulement si pour chaque $v \in \partial S; |N(v) \cap S| \geq |N[v] \cap (\overline{S})|$ Dans ce cas, nous disons que chaque sommet de ∂S est vulnérable à une attaque possible par des sommets de S (par la force des nombres). Aussi si l'inégalité est stricte, alors S est une alliance offensive forte.

Définition 1.2.10. Un ensemble $S \subset V$ est une alliance offensive globale [12] si pour chaque $v \in \overline{S}, |N[v] \cap S| \geq |N[v] - S|$ Si l'inégalité est stricte, nous appelerons ça une alliance offensive globale forte.

Définition 1.2.11. Le nombre d'alliance offensive $a^o(G)$(respectivement le nombre d'alliance offensive forte $\hat{a}^o(G)$) est le cardinal minimum d'une alliance offensive(respectivement d'une alliance offensive forte) de G.

Définition 1.2.12. Le nombre d'alliance offensive globale $\gamma_a^o(G)$ (respectivement, le nombre d'alliance offensive globale forte $\gamma_{\hat{a}}^o(G)$) est le cardinal minimum d'une alliance offensive globale (respectivement, d'une alliance offensive globale forte) de G.

Exemple 1.2.2. Dans le graphe G de la Figure 1.2, L'ensemble $S = \{2, 3, 4, 5, 6\}$ est une alliance offensive forte :



(a) $S = \{1, 2, 3, 4, 5, 11\}$ est une alliance offensive globale forte.

(b) $S = \{2, 3, 4, 5, 6\}$ est une alliance offensive forte

FIGURE 1.2 – Exemples d'alliances offensives dans les graphes[18].

1.2.3 Alliances puissantes

Nous définissons à présent le dernier type d'alliance dit puissante [22] :

Définition 1.2.13. Une alliance est dite puissante si elle est à la fois défensive et offensive. Autrement dit, un ensemble non vide de sommets $S \subset V$ est une alliance puissante, si :
$\forall v \in N[S], |N[v] \cap S| \geq |N[v] - S|$.
Si de plus l'inégalité est stricte, alors S est dite alliance puissante forte.

Définition 1.2.14. Une alliance puissante globale de G est une alliance puissante avec S ensemble dominant dans le graphe G.

Définition 1.2.15. Le nombre d'alliance puissante a^p(respectivement, d'alliance puissante globale γ^p) est le cardinal minimum d'une alliance puissante (respectivement, d'une alliance puissante globale).

Voici quelques valeurs exactes de $a^p(G)$ et γ^p dans des familles de graphes spécifiques [4] :

1. Pour toute chaine P_n avec $n \geq 2$, $a_p(P_n) = \gamma_p(P_n) = [2n/3]$, et
2. Pour tout cycle C_n, $a_p(C_n) = \gamma_p(C_n) = [2n/3]$
3. Pour le graphe complet K_n, $a_p(K_n) = \gamma_p(K_n) = [n/2]$
4. Pour un graphe biparti complet $K_{r,s}$; $1 \leq r \leq s$, $a_p(K_{r,s}) = \gamma_p(C_{r,s}) = \min(r + \frac{s}{2}, [\frac{r+1}{2}] + [\frac{s+1}{2}])$;

1.2.4 k-Alliances

Une généralisation des alliances présentées ci-dessus appelée k-alliances (ou r-alliances)ont été introduite par Shafique et Dutton [35]

Définition 1.2.16. Considérons un graphe $G = (V, E)$.Un sommet v dans l'ensemble $A \subseteq V$ est dit être $k-$satisfait par rapport à A si $d_A(v) \geq d_{V-A}(v) + k$, où $d_A(v) = |N(v) \cap A| = |N_A(v)| = d(v) - d_{V-A}(v)$.

- Un ensemble non vide $S \subseteq V$est une k-alliance défensive [10] dans $G = (V, E)$,$k \in$ {-d,..., d }, si pour tout $v \in S$

$$d_S(v) \geq d_{\overline{S}}(v) + k \tag{1.2}$$

  Ou bien :

$$d(v) \geq 2d_S(v) + k. \tag{1.3}$$

  Une k-alliance défensive S est dite globale si elle est un ensemble dominant.
  . Pour la Figure 1.3 $\{2, 5, 6\}$ est une $0-$alliance défensive et $\{3, 4, 5\}$ est une $(-1)-$alliance défensive globale.
- Un ensemble non vide $S \subseteq V$ est une k-alliance offensive [10] dans $G = (V, E)$,$k \in$ {2-d,..., d }, si pour tout $v \in \partial$(S)

$$d_S(v) \geq d_{\overline{S}}(v) + k \tag{1.4}$$

  Ou bien

$$d(v) \geq 2d_S(v) + k. \tag{1.5}$$

- Une k-alliance offensive S est dite globale si elle est un ensemble dominant. 1.3 montre des exemples de k-alliances offensives(globales).

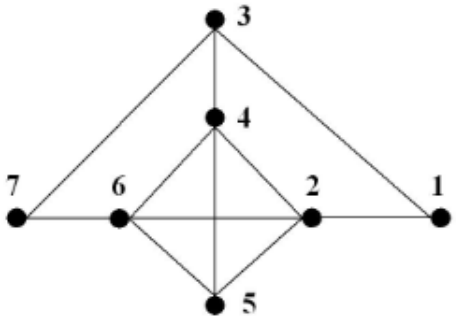


FIGURE 1.3 – {2,6} est une (0)-alliance offensive et {2, 4, 6} est une (-1)-alliance offensive globale [40]

Il est clair que si $k > d$, aucun ensemble S ne satisfait (1.1) et, si $k < 2 - d$, tous les sous-ensembles de V le satisfont.

Définition 1.2.17. Le nombre de k-alliances offensives de G, noté par $a_k^o(G)$, est défini comme la cardinalité minimale d'une k-alliance offensive dans G et le nombre global de k-alliances offensives de G, noté $\gamma_k^o(G)$, est défini comme la cardinalité minimale d'une k-alliance offensive globale dans G.La complexité du calcul de la cardinalité minimale des k-alliances offensives(globales) dans les graphes a été étudiée dans [8] et il a été prouvé que c'est un problème NP-complet.

Remarque 1.2.1. Si $k > 1$, le graphe étoile $K_{1,t}$ n'a pas de k-alliances défensives et chaque ensemble composé de deux sommets adjacents dans un graphe cubique est une (-1)-alliance défensive

Définition 1.2.18. Pour les graphes ayant des alliances défensives,le nombre de k-alliances défensive de G, noté par $a_k^d(G)$, est défini comme la cardinalité minimale d'une k-alliance défensive dans G et le nombre global de k-alliances défensive de G, noté $\gamma_k^d(G)$, est défini comme la cardinalité minimale d'une k-alliance défensive globale dans G.La complexité du calcul de la cardinalité minimale des k-alliances défensives(globales) dans les graphes est aussi NP-complet[8].

- Un ensemble non vide $S \subseteq V$ est une k-alliance puissantes dans $G = (V, E)$,$k \in \{$-d,..., $d-2$ $\}$, si S est une k-alliance défensive et une $(k+2)-$alliance offensive. Une k-alliance puissante est dite globale si elle est un ensemble dominant. En reprenant l'exemple de la Figure 1.3,l'ensemble $\{2, 6\}$ est une (-2)-alliance puissante et l'ensemble $\{2, 3, 4, 6\}$ représente une (-1)-alliance puissante globale.

1.2.4.1 k-alliances limites dans les graphes

Un cas particulier de k-alliances a été introduit par Yero et al.[39] :

Définition 1.2.19. Nous définissons une k-alliance défensive limite dans un graphe comme un ensemble S de sommets avec la propriété que chaque sommet dans S a exactement k voisins de plus dans S qu'il n'en a en dehors de S.

Définition 1.2.20. L'ensemble $S \subset V$ est une k-alliance défensive limite dans G, avec $k \in \{-d_1,....,d_1\}$, si :

$$d_s(v) = d_{\overline{S}}(v) + k, \forall v \in S \tag{1.6}$$

Une k-alliance défensive limite dans G est globale si elle forme un ensemble dominant dans G.

Remarque 1.2.2.

- Si G est un graphe simple et $k \in \{-d_1,....,d_1\}$.Si pour tout $v \in V$, $d(v) = k$est un nombre impair, alors G ne contient pas de k-alliance défensive limite. Par conséquent, pour un graphe G d-régulier et soit $k \in \{-d_1,....,d_1\}$:.Si $d-k$ est impair, alors G ne contient pas de k-alliance défensive limite.
- Si S est une k-alliance défensive dans G et $\overline{S}$ est une (-k)-alliance offensive globale dans G, alors S est une k-alliance défensive limite dans G.

Exemple 1.2.3. Dans le graphe G de la Figure 1.4, $S = \{1, 2, 3\}$ est une $1-$alliance défensive limite.

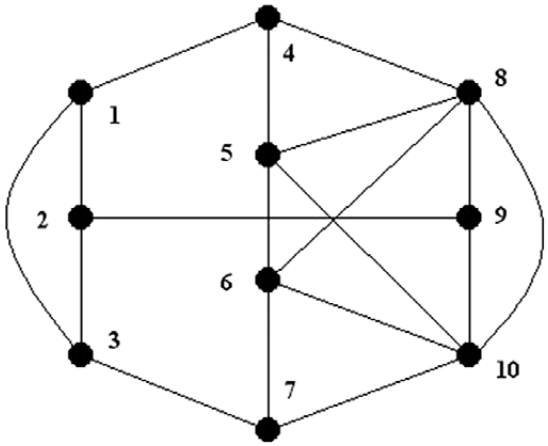


FIGURE 1.4 – 1-alliance défensive limite[39].

I.G.Yero et J.A. Rodríguez-Velázquez[32] ont étudié les propriétés mathématiques des k-alliances défensives limites qui consistent en :

Propriétè 1.2.1. *Pour tout graphe $G = (V, E)$ d'ordre n, si S est une k-alliance défensive limite dans un graphe G, alors :*

1. $\frac{d_n+k+2}{2} \leq |S| \leq \frac{2n-d_n+k}{2}$
   Ce qui implique que la cardinalité de chaque k-alliance défensive limite S dans un graphe complet d'ordre n est $|S| = \frac{n+k+1}{2}$
2. *Soit $m < S >$ la taille de S et soit c le nombre d'arêtes de G ayant un point d'extrémité dans S et un autre point d'extrémité en dehors de S :*
   (a) $m < S >= \frac{c+|S|k}{2}$
   (b) *Si le graphe G est d-régulier, alors $m < S > \frac{|S|(d+k)}{4}$ et $c = \frac{|S|(d-k)}{2}$*
3. *Si G est un graphe connexe, alors : $\frac{n(\mu-[\frac{d_1-k}{2}])}{\mu} \leq |S| \leq \frac{\mu_*-[\frac{d_n-k}{2}]}{\mu_*}$ et $\frac{n(\mu+k+2-\mu)}{2n} \leq |S| \leq n - \frac{n(\mu-k)-\mu}{2n}$*

Définition 1.2.21. Une k-alliance offensive limite globale dans un graphe G est un ensemble S de sommets de G avec la propriété que chaque sommet dans $\overline{S}$ a exactement k voisins de plus dans S qu'il n'en a en dehors de S.

Définition 1.2.22. L'ensemble $S \subset V$ est une k-alliance offensive limite dans G,si $k \in \{2 - d_1,....,d_1\}$:

$$d_s(v) = d_{\overline{S}}(v) + k, \forall v \in \bigcup_{u \in S} N_{\overline{S}}(u) \tag{1.7}$$

Définition 1.2.23. Une k-alliance puissante limite globale est un ensemble S de sommets de G, qui est à la fois une k-alliance défensive limite globale et une $(k+2)$-alliance offensive limite globale.

Exemple 1.2.4. Dans le graphe G de la Figure 1.5, $S = \{1, 2, 3, 4\}$ est une $2-$alliance puissante limite et $S = \{5, 6, 7, 8\}$ est une $0-$alliance puissante limite

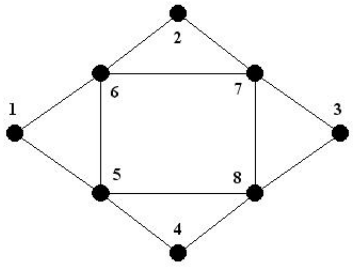


FIGURE 1.5 – Exemple de deux alliances puissantes limites[40].

Remarque 1.2.3. Si S est une k-alliance puissante limite dans un graphe, alors

1. $\frac{d_n+k+2}{2} \leq |S| \leq \frac{2n-d_n+k}{2}$
2. De (1), si G est un graphe complet K_n alors :$|S| = [\frac{n+k+1}{2}]$

Propriétè 1.2.2. *Pour tout graphe $G = (V, E)$ d'ordre n, si S est une k-alliance puissante limite dans G, alors[40] :*

1. $\frac{n(2d_n+k+2)-2m}{2d_n+2} \leq \frac{n(2d_1+k+2)-2m}{2d_1+2}$
2. $\frac{2m+n(k+2)}{2d_1+2} \leq |S| \leq \frac{2m+n(k+2)}{2d_n+2}$

3. Soit c le nombre d'arêtes de G ayant un point d'extrémité dans S et un autre point d'extrémité en dehors de S avec $k \neq -1$: $|S| = \frac{2(m+n-2c)+nk}{2(k+1)}$
4. Si S est une $(-1)-$alliance puissante limite globale alors $\frac{n(d_n+1)}{d_1+d_n+2} \leq |S| \leq \frac{n(d_1+1)}{d_1+d_n+2}$
   Et si $M \subset E$ est une coupe ayant un point d'extrémité dans S et un autre point d'extrémité en dehors de S : $\frac{2m+n}{2d_1+2} \leq |S| \leq \frac{2m+n}{2d_n+2}$ et $|M| = \frac{2m+n}{4}$

1.3 Domaines d'application des alliances

Bien que les alliances soient appliquées dans des domaines divers et variés, nous nous intéresserons dans cette partie à trois d'entre eux.

1.3.0.1 Clustering

Le Clustering est un processus de partitionnement d'un ensemble de données en clusters, où un cluster est une collection de points de données qui sont à la fois similaires et différents les uns aux autres des autres points de données. Ce problème et ses nombreuses variantes ont été largement étudiés en mathématiques ainsi qu'en sciences appliquées. Ces dernières années, la disponibilité de vastes quantités de données (en raison de l'émergence du web, de l'augmentation considérable de

la puissance de calcul, du stockage des données et de la vitesse de communication), ont revitalisé la recherche sur ce problème. Différents algorithmes de clustering utilisent différents concepts de cluster. Shafique et al. [34] se base sur le fait que chaque cluster est une alliance défensive forte. En effet, une alliance défensive puissante est un ensemble de sommets qui ont au moins autant de voisins à l'intérieur de l'ensemble qu'à l'extérieur. En se basant sur ce principe, ils proposent un algorithme qui génère une hiérarchie de clusters en dédoublant chaque cluster (en commençant par le cluster qui contient toutes les données) en deux plus petits clusters (0−alliances défensives) jusqu'à ce qu'aucun cluster ne puisse être divisé.

1.3.0.2 Bioinformatique

L'acide ribonucléique (ARN) est une molécule biologique présente chez pratiquement tous les êtres vivants, et aussi chez certains virus. Chimiquement, l'ARN est un polymère linéaire constitué d'un enchaînement de nucléotides. Chaque nucléotide contient un groupe phosphate, un sucre (le ribose) et une base nucléique. L'analyse et la prédiction de la structure des ARNs, et en particulier de leur structure secondaire, est un champ de recherche en plein essor à la fois dans le domaine de la biologie moléculaire et de la bio-informatique. Haynes et al. [13] ont appliqué le concept des graphes et des alliances défensives à l'analyse de la structure secondaire de l'ARN. Ils représentent les tiges (branches) en tant qu'arêtes et les ruptures dans les tiges qui se traduisent par des renflements et des boucles en tant que sommets. Un bombement nucléotidique, une boucle en épingle à cheveux ou une boucle interne sont représentées par un sommet lorsqu'il y a plus d'un nucléotide non apparié ou une paire de bases non complémentaires.

1.3.0.3 Réseaux VANETs

Un VANET(Vehicular Ad hoc NETworks "VANETs") est un type de réseau ad hoc mobile ayant des caractéristiques spécifiques telles que la forte mobilité,la connectivité intermittente, les changements de topologie, et la forte densité du réseau.Un VANET est composé principalement d'unités mobiles (véhicules) et d'unités de bords de routes(Road Side Units "RSUs"). K.Ouazine et al. [29] ont étudie en détails le problème de saturation des RSUs dans les VANETs, ils ont aussi proposé une nouvelle approche de coopération entre les RSUs d'un VANET (dans le but de réduire sa congestion et d'éviter autant que possible la saturation de ces RSUs). Cette approche, appelée "D2A2RS" (Defensive Alliance based Approach for Reducing RSUs Saturation), est basée sur le concept de k-alliances défensives dans les graphes qui assure une collaboration efficace entre les RSUs.

1.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'étude des alliances dans les graphes, leurs différents types et propriétés, et certains de leurs domaines d'application dans différentes études. Le prochain chapitre portera sur les principaux travaux antérieurs des alliances et des graphes d'attaques.

Chapitre 2

Travaux antérieurs sur les alliances et les graphes d'attaques

2.1 Introduction

Dans ce chapitre, nous présenterons les concepts de base des graphes d'attaques, par la suite nous nous intéresserons aux travaux antérieurs reliés à leur analyse. Nous nous intéresserons ensuite aux travaux antérieurs reliés à l'utilisation des alliances dans les graphes.

2.2 Concepts de base des graphes d'attaques

Nous commençons par définir quelques concepts de base concernant les graphes d'attaques[15] :

2.2.1 Vulnérabilité

Une vulnérabilité est une faiblesse exploitable dans la conception, la mise en œuvre ou la gestion d'un système. Elle comprend une combinaison d'un ou de plusieurs états du système appelés pré-conditions.

2.2.2 Précondition

Les préconditions sont un ensemble de propriétés du système qui doivent exister pour qu'un exploit soit réussi. Une précondition initiale est une propriété du système qui existe de manière intégrée et qui n'est pas apparue à la suite d'une exploitation. La résolution de ces vulnérabilités pourrait rendre toutes les étapes ultérieures d'une attaque nulles. Il existe au moins trois types de pré-conditions :

1. Statuts/services : La cible détient ou annonce des versions particulières d'exploitation, de logiciels systèmes, ou se trouve dans un état matériel/logiciel particulier.
2. Accessibilité : La cible est accessible.

3. Les capacités du pirate : Le pirate a des capacités particulières, comme la capacité d'exécuter un processus sur une cible, l'accès à des outils ou des niveaux de privilèges [143] et/ou est en possession d'outils pour mener une attaque et possède le niveau de compétence nécessaire.

Parmi les types de préconditions évoqués ci-dessus, les statuts/services sont couramment représentés dans les graphes d'attaques et les arbres d'attaques. Cependant l'accessibilité et la capacité du pirate sont moins souvent représentées.[15]

2.2.3 Postcondition

La réussite dans la violation d'un exploit donne lieu à une ou plusieurs postconditions. Ces conditions peuvent également constituer les pré-conditions d'autres exploits, c'est pourquoi la plupart des chercheurs se réfèrent donc aux postconditions comme à des préconditions.

2.2.4 Exploits

Un exploit est un ensemble d'étapes qui tirent profit d'une ou plusieurs vulnérabilités dans un système cible et fournissent des capacités spécifiques au pirate. Les exploits peuvent être définis comme un prédicat de la forme : v(h_s, h_d) où [15]

- v est la vulnérabilité exploitée. Comme indiqué précédemment, la vulnérabilité peut être décomposée en plusieurs pré-conditions préalables, ou une combinaison de celles-ci.
- h_s : est l'hôte source, c'est-à-dire l'hôte qui commet l'exploit.
- h_d : est la cible de l'exploit.

2.2.5 Conditions de sécurité

Une condition de sécurité est un prédicat c(h_s, h_d) qui indique une condition de sécurité satisfaite c impliquant l'hôte source h_s et l'hôte de destination h_d (quand une condition implique un seul hôte, nous écrivons simplement c(h)). Des exemples de conditions de sécurité comprennent l'existence d'une vulnérabilité sur un hôte donné ou la connectivité entre deux hôtes.

2.2.6 Graphe d'attaques

Un graphe d'attaques est défini comme suit, selon Wang et al.[38] :

Définition 2.2.1. Un graphe d'attaques est un graphe orienté ayant deux types de sommets, les exploits et les conditions de sécurité. Les interdépendances entre exploits et conditions de sécurité forment les arêtes d'un graphe d'attaques. Une arête allant d'une condition de sécurité à un exploit indique que l'exploit ne peut être exécuté tant que la condition de sécurité n'est pas satisfaite, une arête allant d'un exploit à une condition de sécurité indique que l'exécution de l'exploit satisfait la condition de sécurité.

Définition 2.2.2. Plus formellement : Étant donné un ensemble d'exploits E, un ensemble de conditions de sécurité C, une relation d'exigence $R_T \subseteq C \times E$ (qui indique qu'un exploit E exige un ensemble de conditions C pour qu'il soit réussi), et une relation d'implication $R_i \subseteq E \times C$ (qui indique que la violation d'un exploit E implique et donne lieu à un ensemble de postconditions C). Un graphe d'attaques G est le graphe orienté G = (E ∪ C,$R_T \cup R_i$), où E ∪ C est l'ensemble des sommets et $R_T \cup R_i$ l'ensemble des arêtes.

2.3 Travaux antérieurs sur les graphes d'attaques

Dans cette section, nous nous intéresserons aux travaux antérieurs sur l'analyse des graphes d'attaques, nous citerons trois travaux sur l'analyse des graphes.

2.3.1 Application de la théorie des jeux pour l'analyse des graphes d'attaques[3]

La sous-section suivante résume un travail [3] qui se base sur la théorie des jeux pour la résolution du problème d'analyse des graphes d'attaques :

2.3.1.1 Modélisation

- **Les joueurs :** L'ensemble des joueurs est N = {Attaquant,Administrateur}.
- **Les stratégies :** Bouafia et al.[3] ont défini les stratégies par un ensemble S = $S_1 \cup S_2$ où S_1 sont les stratégies d'attaques S1 = $\{s_1^1, s_2^1, ..., s_n^1\}$, $|S_1|$ = n. et S_2 sont les stratégies de défenses avec S2 = $\{s_1^2, s_2^2, ..., s_n^2\}$, $|S_2| = m$.
- **Les récompenses :** Un intrus essaie de pénétrer dans le réseau en menant une série d'attaques et en essayant de causer un maximum de dégâts.
  L'administrateur essaie de minimiser ses pertes et maximiser ses gains en réagissant contre les attaques menées par l'intrus.
- **Utilité :** Pour chaque joueur i une fonction d'utilité(ou gains) U avec :$U_i : S_1 \times S_2 \rightarrow R$ qui associe à chaque profil de stratégies une valeur numérique représentant un gain du joueur i.
- **Solution** :
  1. **Représenter le jeu sous forme normale :** Cette étape consiste à représenter les deux joueurs, leurs stratégies et les issues correspondants aux profils de stratégies, à l'aide d'un tableau à deux dimensions.
  2. **Analyser le graphe d'attaques :** Après avoir représenter le jeu sous forme normale, les graphes d'attaques sont analysés dans le but de réduire les vulnérabilités existantes. Pour se faire, le processus d'élimination des stratégies dominées est appliqué.

2.3.2 Une technique d'analyse des graphes d'attaques[11]

L. Hamza et al. [11] ont proposé une approche qui consiste à analyser un graphe d'attaques généré par n'importe quelle approche. Le réseau est analysé pour aider l'administrateur à détecter les failles de son système, en lui montrant d'où elles proviennent et lui proposer une solution pour mieux sécuriser son réseau qui est la fermeture des ports représentant un risque dans le réseau. L'approche peut se résumer en quatre étapes :

1. Parcourir un graphe d'attaques : Le parcours est en profondeur et deux fonctions sont utilisées dans cette étape
   — Voisins(S) : renvoie la liste des sommets adjacents à S ;
   — Marquer(S) : marque un sommet S comme exploré, pour éviter la redondance dans la recherche.
2. Mesurer les risques de sécurité : Après avoir calculé le nombre de chemin d'attaques dans la premiere étape, le risque provoqué par chaque chemin est calculée avec la formule suivante :

$$R(i) = \frac{S_i}{S} \tag{2.1}$$

   S_i :représente le nombre des services de chemin i,
   S : représente le nombre de services activés dans tout le réseau.
   $\frac{S_i}{S}$: la probabilité que le chemin i sera vulnérable au S_i.

3. Quantifier les risques de sécurité : Cette étape se focalise sur les types des vulnérabilités des services activés sur le réseau.
   — Chaque service a un numéro de port (deux types de ports réservés et non réservés).
   — La stratégie de durcissement se repose sur la fermeture de quelques ports qui ont le risque le plus élevé (obtenu dans la première étape).
   — Pour calculer le nombre de ports N les auteurs ont utilisé un compteur initialisé à zéro.
4. Calculer le coût de durcissement :
   Coût(S)=nombre de ports fermés / nombre de ports max des R(i).

2.3.3 Durcissement des réseaux efficace en termes de temps et de coût grâce aux graphes d'attaques [1]

Noel et al. [1]ont formalisé la notion de stratégie de durcissement en termes d'actions atomiques autorisées, et ont défini un modèle de coût qui prend en compte l'impact des actions de durcissement interdépendantes. Ils ont également introduit un algorithme d'approximation quasi-optimal qui évolue linéairement avec la taille des graphes. Les approches précédentes pour le durcissement des réseaux cherchent des solutions exactes et ne sont donc pas évolutives, les éléments de durcissement ont été traités indépendamment, ce qui est inapproprié pour les environnements réels. Le travail

présenté par Noel et al. [1] diffère considérablement des travaux précédents, car ils (i) ont abandonné l'hypothèse selon laquelle les conditions initiales peuvent être désactivées indépendamment ; (ii) ont introduit un modèle de coût formel ; et (iii) ont présenté un algorithme d'approximation qui génère efficacement des solutions sous-optimales.

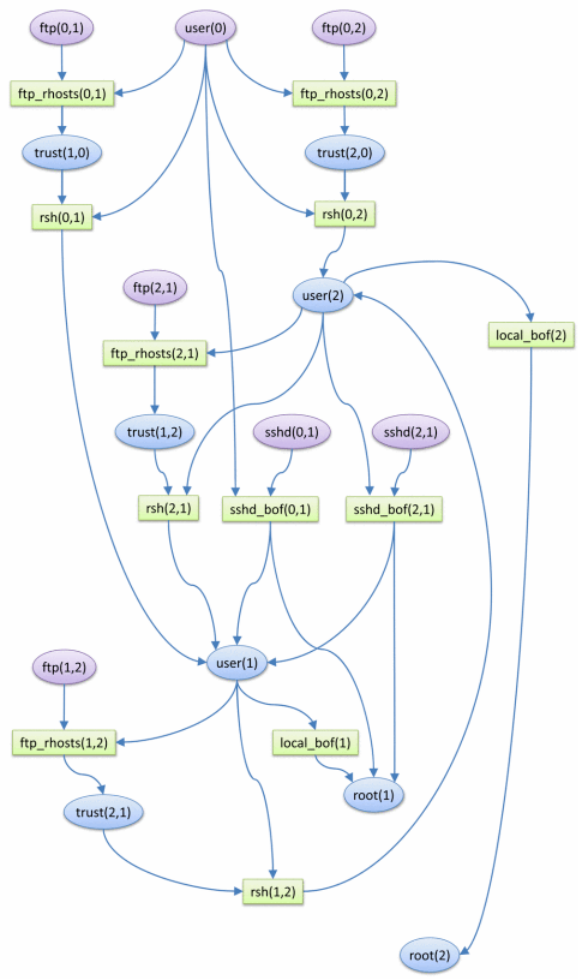


FIGURE 2.1 – Exemple de graphe d'attaques, comprenant les conditions initiales (ovales violets), les exploits (rectangles verts) et les conditions intermédiaires (ovales bleus).[1]

La Figure 2.1 présente un exemple de graphe d'attaques. Dans cet exemple, l'objectif est de durcir le réseau en ce qui concerne la condition cible root(2), le but étant d'empêcher l'attaquant d'obtenir des privilèges root sur l'hôte 2. Le scénario représenté sur la Figure 1 est relativement simple :

— Trois hôtes - respectivement hôtes 0, 1 et 2 -

— Quatre types de vulnérabilités - ftp rhosts, rsh, sshd bof et local bof-.

Étant donné que plusieurs chemins d'attaques entrelacés peuvent conduire à la condition recherchée, une solution optimale pour renforcer le réseau n'est toujours pas apparente à partir du graphe d'attaques lui-même.

Intuitivement, pour empêcher la condition de l'objectif d'être satisfaite, une solution de

durcissement du réseau doit briser tous les chemins d'attaques menant à l'objectif. Cette intuition a été capturée par le concept d'**ensemble critique**, c'est-à-dire un ensemble d'exploits (et de conditions correspondantes) dont la suppression du graphe d'attaques invalidera tous les chemins d'attaques.

Il a également été démontré que la recherche d'ensembles critiques avec la cardinalité minimale est NP-hard, alors que la recherche d'un ensemble critique minimal (Un ensemble critique dont aucun sous-ensemble approprié n'est un ensemble critique) est polynomiale.

Il existe de nombreux ensembles critiques minimaux dans la Figure 2.1 comme $\{rsh(0,2), rsh(1,2)\}, \{ftp-rhosts(0,2), rsh(1,2)\}, \{ftp-rhosts(1,2), rsh(0,2)\}$.

La solution ci-dessus ne prend pas en compte le fait que tous les exploits ne sont pas sous le contrôle direct des administrateurs.

En effet, un exploit ne peut être supprimé qu'en désactivant ses conditions requises, mais toutes les conditions ne peuvent pas être désactivées à volonté. Autrement dit, on ne peut pas supprimer une conséquence sans supprimer ses causes.

Seules les conditions initiales qui ne sont pas impliquées par un exploit peuvent être désactivées indépendamment des autres exploits. Du point de vue d'une condition cible, le graphe d'attaques peut être vu comme un arbre dont la racine est la condition cible et dont les conditions initiales sont les noeuds feuilles.

Une condition peut être impliquée par un ou plusieurs exploits. À leur tour, ces exploits ont besoin d'une ou plusieurs conditions préalables pour être exécutés.

Ainsi le graphe d'attaques de la Figure 2.1 possède les mêmes solutions de durcissement que le graphe d'attaques simplifié de la Figure 2.2.

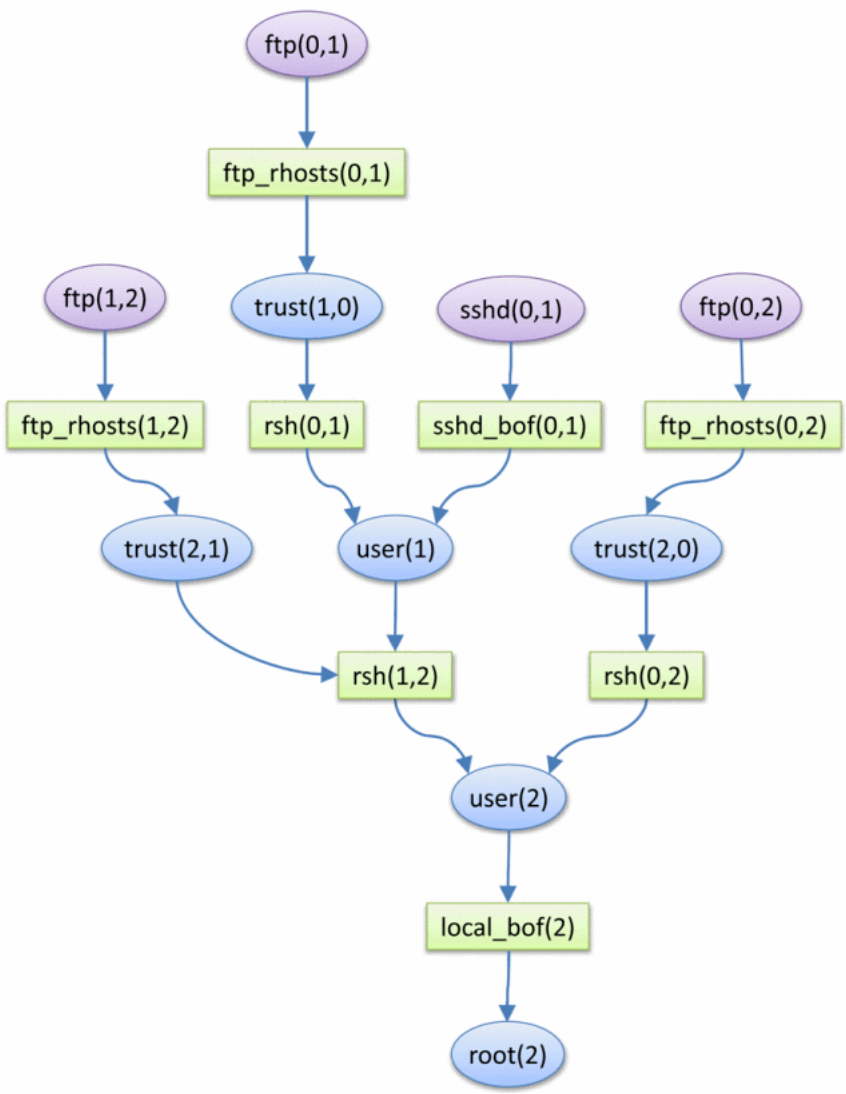


FIGURE 2.2 – Un graphe d'attaques de type arbre équivalent au graphe de la Figure 2.1[1]

La désactivation d'un ensemble de conditions initiales afin d'empêcher les attaques sur des cibles données peut entraîner des effets indésirables tels que le déni de service aux utilisateurs légitimes. Ces effets sont considérablement amplifiés lorsque les conditions initiales ne peuvent pas être désactivées individuellement, mais nécessitent plutôt des actions qui désactivent un plus grand nombre de conditions. Par exemple, une action de durcissement autorisée peut consister à arrêter le service ftp sur un hôte donné. Ainsi, chaque action peut avoir des effets supplémentaires en plus de la désactivation d'une condition souhaitée. De tels effets doivent être pris en compte lors du calcul des solutions à coût minimum.
Dans le graphe d'attaques de la Figure 2.1, la désactivation de ftp(1,2) pourrait ne pas être possible sans désactiver également ftp(0, 2).

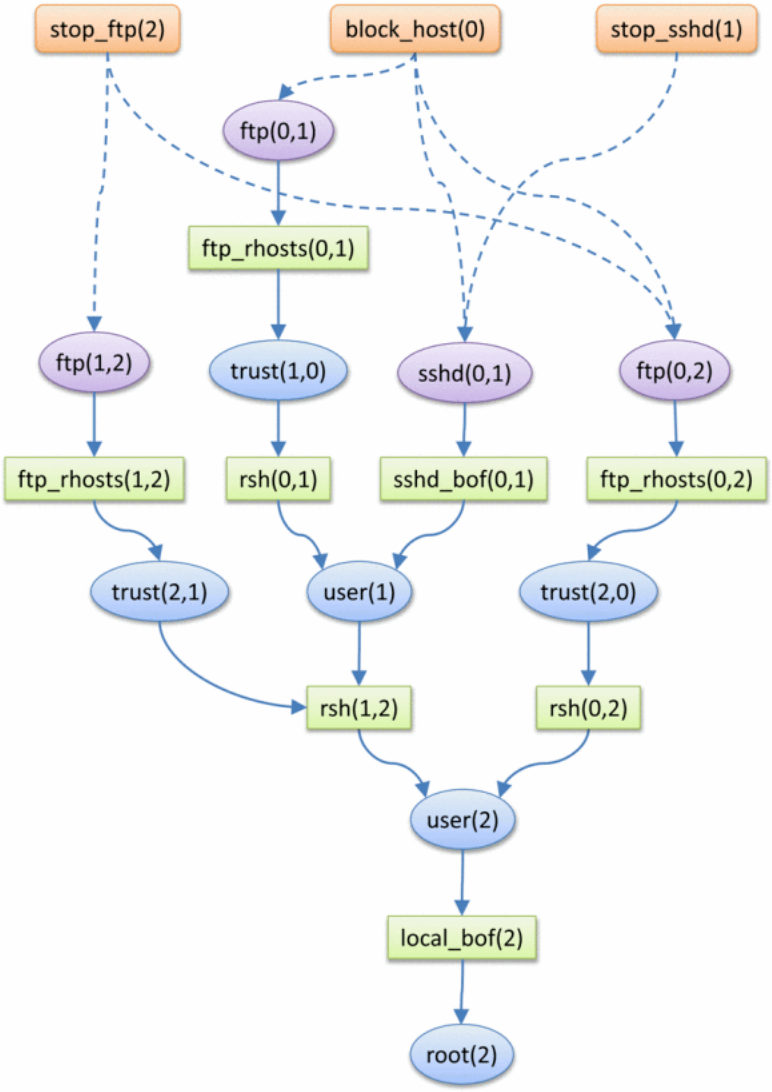


FIGURE 2.3 – Actions de durcissement possibles (rectangles orange) pour le graphe d'attaques de la Figure 2.2[1]

La Figure 2.3 représente le même graphe d'attaques que la Figure 2.2, mais elle montre explicitement les actions de renforcement autorisées, représentées par des rectangles oranges. Quand une action A est prise, toutes et seulement les conditions de A sont supprimées. Dans l'exemple de la Figure 2.3, A = $\{stopftp(2), blockhost(0), stopsshd(1)\}$, $stopftp(2) = \{ftp(0,2), ftp(1,2)\}$, $blockhost(0) = \{ftp(0,1), sshd(0,1), ftp(0,2)\}$, et $stopsshd(1) = \{sshd(0,1)\}$.

Dans cet exemple, la condition ftp(1,2) ne peut pas être désactivée qu'à travers l'action stop-ftp(2), qui désactive également ftp(0,2). Par conséquent, lors du choix d'un ensemble de conditions initiales à supprimer afin d'empêcher des attaques sur des cibles données, toutes les implications de la suppression de ces conditions doivent être prises en compte. Pour résoudre ce problème, les auteurs ont formalisé une notion de stratégie de durcisse-

ment en termes d'actions autorisées, et ont défini un modèle de coût qui prend en compte l'impact des actions de durcissement.

Modèle de coût formel : Les auteurs ont introduit un modèle de coûts pour permettre une analyse plus précise des options de durcissement. Pour cela, ils ont introduit une fonction de coût de durcissement $A : S \longrightarrow R^+$ avec S un ensemble d'actions de durcissement du réseau $\{A_1, .., A_m\}$ que les conditions $c_1, ..., c_n$ ne puissent pas être atteintes après que toutes les actions de S ont été prises. S désigne l'ensemble de toutes les stratégies possibles, et $C(S)$ l'ensemble des conditions désactivées par la stratégie. La fonction satisfait les axiomes suivants :

1. $Cost(\varnothing) = 0$.....(1)
2. $\forall(S_1, S_2 \in S)(C(S_1) \subseteq C(S_2) \Rightarrow Cost(S_1) \leq Cost(S_2))$.....(2)
3. $\forall(S_1, S_2 \in S)(Cost(S_1 \cup S_2) \leq Cost(S_1) + Cost(S_2))$.....(3)

En combinant les 3 axiomes précédents, on peut conclure :
$\forall(S_1, S_2 \in S)(0 \leq max(Cost(S_1), Cost(S_2)) \leq (Cost(S_1 \cup S_2) \leq Cost(S_1) + Cost(S_2))$.....(3)

Algorithme d'approximation : Pour remédier aux limitations des algorithmes de durcissement du réseau, Noel et al.[1] ont proposé un algorithme d'approximation ForwardSearch. L'algorithme consiste à parcourir le graphe d'attaques vers l'avant, en partant des conditions initiales. Le principal avantage de cet algorithme est que les solutions intermédiaires sont des stratégies de renforcement du réseau par rapport aux conditions intermédiaires. En un seul passage, l'algorithme ForwardSearch peut calculer des stratégies de durcissement par rapport à n'importe quelle condition dans C.

Les auteurs ont pris le graphe de la Figure 2.2 comme exemple. Les trois seules actions permises sur le réseau correspondant sont : $stop - ftp(2) = ftp(1, 2), ftp(0, 2), block - host(0) = ftp(0, 1), sshd(0, 1), ftp(0, 2)$ et $stop\_shd(1) = sshd(0, 1)$ En supposant que Cost(Stop ftp(2)) = 20, Cost(block-host(0)) = 10, Cost(stop-sshd(1)) = 15. Il est clair que la stratégie optimal par rapport à $root(2)$ est $S = block - host(0)$,

2.4 Etude comparative des méthodes d'analyse des graphes d'attaques

Dans cette section, nous comparons les différents types de travaux concernant les graphes d'attaques que nous avons présenté précédemment :

Critères / Travaux	Type de graphe d'attaques	Méthode d'analyse	Approche	Inconvénients
L.Hamza et al.[11]	Généré par n'importe quelle approche.	Algorithme qui parcourt le graphe.	Identifie le chemin le plus probable et le port à plus haut risque.	Un chemin avec un seul port sensible présente un risque de sécurité faible par rapport aux autres possibilités avec plus d'un.
K.Bouafia et Hamza.[3]	Problème modélisée sous forme de jeu.	Théorie des jeux.	Durcissement du réseau, prédiction du comportement des attaques.	L'attaquant peut diriger un gigantesque flux de données malicieuses à l'IDS qui se retrouvera bloqué et non opérationnel
S.Noel et al. [27]	Graphe d'attaques où les conditions et les exploits sont représentés par des sommets de type différents.	Problème des conditions de coût minimum (Minimum-Cost Conditions Problem (MCCP))	Durcissement du réseau, prédiction du comportement des attaques.	Au fur et à mesure que l'algorithme traverse le graphe vers les conditions cibles, il peut y avoir un effet multiplicatif dans l'erreur d'approximation.

Tableau 2.1 – Comparaison des méthodes d'analyse des graphes d'attaques

Les objectifs des trois méthodes sont les mêmes, c'est-à-dire trouver le bon durcissement du réseau en limitant les coûts. L'algorithme de minimisation des coûts prend en compte l'impact des actions de durcissement.La méthode basée sur la théorie des jeux quant à elle considère l'interaction entre les parties offensives et défensives. Par conséquent, elle peut obtenir des résultats plus utiles pour le durcissement. Les trois méthodes font face aux mêmes problèmes, comme la fixation du coût et la complexité du calcul. Les méthodes ci-dessus ne peuvent fournir des stratégies de renforcement de la sécurité que pour une seule cible. De plus, afin d'assurer le fonctionnement normal des réseaux, les administrateurs ne peuvent pas supprimer certains éléments de sécurité. Cela diminue la disponibilité des méthodes ci-dessus dans une certaine mesure.

2.5 Quelques travaux antérieurs sur les alliances dans les graphes

2.5.1 Introduction

L'étude des alliances dans les graphes a été introduite pour la première fois par Hedetniemi et Kristiansen [22]. Ils ont introduit les concepts d'alliances défensives et offensives (respectivement défensive globale et offensive globale) et les ont étudiés sur différents types de graphes tels que les cycles, les roues, les grilles et les graphes complets. Le domaine s'est ensuite étendu, Haynes et al. [14] ont étudié les nombres d'alliances défensives globales de différentes classes de graphes. Ils ont donné des limites inférieures pour les graphes généraux, les graphes bipartites et les arbres, et des limites supérieures pour les graphes généraux et les arbres. Rodriquez-Velazquez et Sigarreta [32] ont étudié le nombre d'alliances défensives et le nombre global d'alliances défensives des graphes linéaires. Les études sur les alliances se sont donc multipliés et ont fini par inclure la généralisation appelé k-alliance [35], leur complexité [8] et différentes autres propriétés. Les alliances dans les graphes servent de modèle mathématique pour plusieurs problèmes pratiques et théoriques qui sont apparus dans la littérature de différents domaines, comme comme la structure des données [34], les communautés web [9], la bio-informatique (étude du protéome et du génome) [13],la réduction de la saturation et de la congestion dans les réseaux VANETs [28] ainsi que les systèmes de défense [22].

2.5.2 Ensembles sécurisés et alliances défensives dans les graphes[2]

Amano et al. [2] ont d'abord présenté un algorithme à paramètre fixe pour trouver un petit ensemble sûr, dont le temps d'exécution est beaucoup plus rapide que ceux présentés précédemment. Ils ont ensuite présenté des limites améliorées sur les plus petites tailles des alliances défensives et des ensembles sécurisés pour les hypercubes.

— **Préliminaires :**Avant d'aborder l'article d'Amano et al. nous définissons d'abord certains termes utilisés :

Le produit cartésien des graphes G et H, noté G □ H, est le graphe avec l'ensemble des sommets V(G) × V(H) et l'ensemble des arêtes $\{\{(g,h),(g,h')\}.|g \in V(G), \{h,h'\} \in E(H)\} \cup \{\{(g,h),(g',h)\}|h \in V(H), \{g,g'\} \in E(G)\}$.

La d-ième puissance cartésienne d'un graphe G, notée G^d, est définie par $G^1 = G$ et $G^d = G\square G^{d-1}$ pour $d \geq 2$. Le graphe de Hamming K_k^d est la d-ième puissance cartésienne du graphe complet K_k pour un certain d et k.

L'hypercube à d dimensions Q_d est la d-ième puissance cartésienne de K_2 ; autrement dit, $Q_d = K_2^d$. Le nombre d'alliance défensive $a^d(G)$ de G est la taille d'une plus petite alliance défensive de G. Amano et al. ont défini les ensembles sécurisés à base de la définition des alliances défensives comme suit :

Un ensemble non vide $S \subseteq V(G)$ est un ensemble sécurisé si pour chaque $X \subseteq S, |N[X] \cap S| \geq |N[X] \backslash S|$. Le nombre de sécurité $s(G)$ de G est la taille du plus petit ensemble sécurisé de G. Clairement, $a^d(G) \leq s(G)$ pour tout graphe G.

— **Un algorithme plus rapide à paramètre fixe pour les ensembles sécurisés :**
Enciso et Dutton [2] ont présenté précédemment un algorithme à paramètres fixes pour résoudre le problème de décision si un graphe donné à $n$sommets possède un ensemble sécurisé de taille au plus égale à k en temps $O(2^{klog2k}n)$. Les auteurs ont proposé un algorithme amélioré en temps $O(2^{3k}k^2n)$. L'algorithme des auteurs est basé tout d'abord sur deux revendications :

— **Revendication 1 :** Si $|N[S]| \geq 2k$, alors il n'existe aucun ensemble sécurisé S' qui satisfait $S \subseteq S'$ et $|S'| \leq k$.

— **Revendication 2 :** Si $S' \supsetneq S$ est un ensemble minimal sûr, alors il existe un sommet $u \in S' \backslash S$ tel que $u \in N[S] \backslash S$.

L'algorithme est composé de deux fonctions :

— $Secure(S)$ qui vérifie si un sous ensemble$S$ est un ensemble sécurisé.

— $Find(S, F)$:Avec $S \subseteq V(G)$ et $F \subseteq N[S] \backslash S$, la procédure $Find(S, F)$ trouve un ensemble sécurisé S' de taille égale au plus k tel que $S \subseteq S'$ et $F \cap S' = \emptyset$. Si S lui-même est sécurisé, alors $Find(S, F)$ sort S en résultat et s'arrête.

Si S n'est pas sécurisé, et qu'il existe un ensemble minimal sécurisé $S' \supsetneq S$ avec $|S'| \leq k$ et $S' \cap F = \emptyset$.Clairement, $|S| \leq k$. Puisque $F \subseteq N[S'] \backslash S'$, on a $|F| \leq |S'| \leq k$.Par la revendication 1, $|N[S]| \leq 2k$.

Si toutes ses conditions ne sont pas satisfaites, alors S' n'existe pas. Sinon, la procédure trouve récursivement S'. Si S' existe, alors il existe un sommet $u \in S' \backslash S$ tel que $u \in N[S] \backslash (S \cup F)$ par la revendication 2.

De plus, si $u \in N[S] \backslash (S \cup F)$, alors soit $u \in S' \backslash S$, soit $u \in N[S'] \backslash S'$. Dans le premier cas u est ajouté dans S. Dans le second cas, il est ajouté dans F. Pour effectuer la vérification des deux cas, $Find(S \cup \{u\}, F)$ et $Find(S, F \cup \{u\})$ sont appelés.

Le reste de l'algorithme consiste a appelé la procédure $Find(\{v\}, \emptyset)$ pour chaque $v \in V(G)$, le tout pour trouver un ensemble sécurisé contenant v.

Il ne reste plus qu'à prouver la complexité en temps, tout d'abord nous avons n branches correspondant à $Find(\{v\}, \emptyset)$ (la procédure a été appelé pour chaque $v \in V(G)$ donc n fois pour n sommets).

Chaque appel de $Find$ a soit 0 branche ou 2 branches correspondant aux deux procédures récursives appelés, par conséquent l'arbre de recherche a une profondeur d'au plus $2k$ dans la racine a n enfants, et les autres noeuds intérieurs ont 2 enfants chacun. Il a donc au plus $1 + n \sum_{i=0}^{2k-1} 2^i 2^{2k} n$ noeuds.

Vu que l'algorithme a une complexité en temps de $O(2^k k^2)$ pour vérifier si S est sécurisé pour chaque noeud de l'arbre de recherche, la complexité totale est de $O(2^{3k}k^2n)$.

2.5.3 Schéma de Clustering basé sur les alliances pour la gestion des clés de groupe dans les réseaux mobiles ad hoc[33]

La gestion des clés de groupe dans les MANET(mobile ad hoc networks / réseaux mobiles ad hoc) est une tâche difficile. La topologie changeant fréquemment en raison du déplacement des noeuds, il est très important de savoir comment créer et maintenir un groupe.
Pour résoudre ce problème, les solutions existantes organisent les membres du groupe en cluster. L'avantage des cluster est que le renouvellement des clés peut se faire rapidement. Cependant, la mise en cluster pose de nouveaux défis aux MANET, comme la synchronisation et la maintenance. La stabilité des clusters est une propriété appréciée de l'algorithme de clustering. Dans leur article, Seba et al.[33] ont proposé d'utiliser le concept d'alliance dans les graphes comme critère de mise en cluster pour la gestion des clés de groupe dans les MANET.
Les auteurs ont également proposé un algorithme de clustering qui organise le réseau en clusters de telle sorte que chaque cluster est une alliance défensive dans le graphe qui représente le réseau. Les auteurs ont représenté le réseau mobile ad hoc par un graphe $G(V, E)$, où V désigne l'ensemble des noeuds ou sommets et E l'ensemble des arêtes non orientées. Tous les liens du réseau sont bidirectionnels et le réseau est connecté. Pour tout noeud i, ils définissent les variables locales suivantes :

- **Une variable d'ensemble** N_i **:** qui stocke le voisinage du noeud i. Le noeud i peut calculer N_i au moyen des messages d'accueil diffusés périodiquement par les noeuds mobiles. N_i varie dans le temps en raison de la mobilité des noeuds.
- **Une variable entière** deg_i **:** qui stocke le nombre de voisins du noeud i, c'est-à-dire $deg_i = |N_i|$
- **Une variable entière** a_i **:** qui stocke le nombre de voisins du noeud i qui sont dans le même cluster que i.
- **Une variable entière** C_i **:** qui indique le cluster auquel appartient le noeud i. Si le noeud i ne fait pas encore partie d'une alliance, C_i est définie comme nulle. Chaque cluster a une tête de cluster. Lorsque C_i=i, le noeud i est un clusterhead.
- **Une variable booléenne** P_i **:**qui indique que le noeud i doit se déplacer vers un autre cluster pour satisfaire le critère d'alliance.
- **Une variable entière** nC_i **:**qui indique le prochain cluster du noeud i. Si le noeud i n'est pas destiné à se déplacer vers un autre cluster, nC_i est défini comme null.
- **Une variable pointeur** S_i **:**qui pointe soit vers null, soit vers un voisin du noeud i. Dans ce dernier cas, cela signifie que le noeud i est dans un cluster mais qu'il n'a pas assez de voisins dans celui-ci. Le noeud i demande donc à l'un de ses voisins de rejoindre le cluster. S_i pointe vers le voisin sélectionné.
- **Une variable pointeur** L_i **:** qui agit comme un verrou. Lorsque L_i pointe vers un noeud j, cela signifie que le noeud i a pris en compte la demande de j de rejoindre son cluster. Il

est donc en mesure de répondre à des demandes similaires. Dans le cas contraire, L_i prend la valeur nulle.

Ils ont ensuite défini les fonctions suivantes :

- La fonction $mind(S)$ retourne le noeud minimum de S ayant le degré minimum, $mind(S) = k \in S, \forall j \in S, deg_k < deg_j$ ou ($deg_k = deg_j$ et $k < j$). Si S est vide, $mind(S)$ retourne null.
- La fonction $maxd(S)$ retourne le noeud minimum de S ayant le degré maximal, $maxd(S) = k \in S, \forall j \in S, deg_k > deg_j$ ou ($deg_k = deg_j$ et $k < j$). Si S est vide, $maxd(S)$ renvoie null.
- La fonction entière $h(i)$ est l'alliance du noeud i si elle existe, c'est-à-dire que $h(i)$ est le cluster minimum c tel que $c \neq$ null et $|\{k \in N_i, C_k = c\}| \leq |\{k \in N_i, C_k \neq c\}|$. $h(i) =$ null si c n'existe pas.
- La fonction booléenne $\psi(i)$ est défini de la façon suivante : $\psi(i) = (h(i) = C_i$ ou $h(i) \neq null)$. L'état légitime global du système est un état du système dans lequel $\psi(i) =$ vrai pour tous les noeuds du graphe, chaque noeud fait donc partie d'une alliance.
- La fonction $ch(i)$ détermine les têtes de cluster. Lorsqu'un noeud i est une tête de cluster $C_i = i$. Au départ, les têtes de cluster servent à lancer le processus de clustering. Comme chaque cluster doit être une alliance de graphes, ce sont les noeuds les plus isolés qui se déclarent têtes de cluster. La fonction $ch(i)$ renvoie le clusterhead initial approprié du noeud i si un tel clusterhead existe et null sinon.

  Il existe deux situations où un noeud i se déclare clusterhead :

  - **Situation 1 :** il n'y a pas d'alliance dans le voisinage du noeud i et c'est le noeud qui a le plus petit degré dans son voisinage, c'est-à-dire $h(i)$= null C_i = null mind($N_i \cup \{i\}$) = i.
  - **Situation 2 :** le noeud i ne peut rejoindre aucune des alliances qui se trouvent dans son voisinage, principalement parce qu'il n'a pas assez de voisins dans aucune d'entre elles, $h(i) = null\ \forall j \in N_i, C_j \neq null$.

  L'algorithme est ensuite décomposé en six règles :

- La règle R1 est exécutée par le noeud i lorsqu'il se trouve dans un cluster C_i qui n'a pas assez de voisins pour satisfaire le critère d'alliance. Le noeud i scanne alors ses voisins pour voir si certains d'entre eux peuvent rejoindre son cluster. Un voisin j contenu dans $C_j \neq C_i$ peut rejoindre le cluster de i si son verrou L_j est faux. Si plusieurs voisins répondent à ce critère, le noeud i sélectionne celui qui a le plus faible degré et place son pointeur S_i sur celui ci. Si le noeud i n'a aucun voisin capable de rejoindre son cluster, il doit se déplacer vers un autre cluster. Il fixe donc nC_i dans l'un des clusters environnants.
- La règle R2 teste si un noeud i doit se déplacer vers un autre cluster et fixe nC_i à la valeur appropriée qui est :
  - le cluster d'un voisin j qui pointe vers i car il a besoin de plus de voisins dans son cluster pour constituer une alliance.
  - la tête de cluster de i si $h(i)$ retourne null. Selon la fonction $ch(i)$, la tête de cluster de i est soit i, soit une tête de cluster voisine si elle existe.

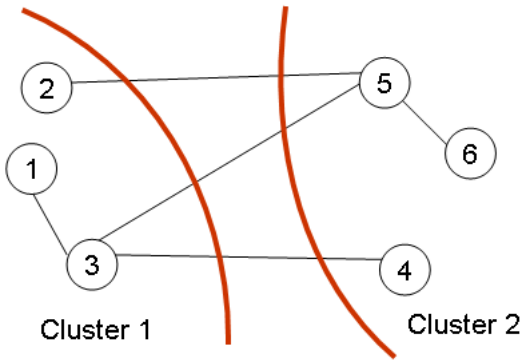


FIGURE 2.4 – Exemple d'exécution de la règle R4[33]

— $h(i)$ sinon.

— Le noeud i exécute la règle R3 lorsque la valeur de son prochain cluster nC_i n'est pas nulle et est différente de la valeur de son cluster actuel C_i. Dans ce cas, le noeud i indique son intention de passer à un autre cluster en fixant P_i à vrai.

— Lorsque le noeud i est le candidat de degré le plus bas, de son voisinage, pour passer à un autre cluster, il exécute la règle R4. Le noeud i change sa variable C_i en nC_i. La condition sur le degré agit comme un verrou et évite que deux voisins dans différents clusters se déplacent indéfiniment entre deux clusters. La Figure 2.4 donne un exemple d'une telle situation. Dans cette Figure, les noeuds 3 et 5 doivent changer de cluster. Le noeud 3 doit se déplacer vers le cluster 2 tandis que le noeud 5 doit se déplacer vers le cluster 1. S'ils se déplacent simultanément, ils se retrouvent dans la même situation. Ainsi, la règle R4 impose un ordre de mouvements entre deux candidats voisins. Le premier qui exécute la règle R4 changera sa variable C_i, le second réévaluera nC_i et P_i en exécutant les autres règles.

— La règle R5 met la valeur de C_i à zéro si elle pointe vers un cluster inexistant.

— La règle R6 sert à réaliser le verrouillage L_i du noeud i lorsque le noeud vers lequel elle pointe quitte le cluster du noeud i. Les auteurs ont ensuite prouvvé les théorèmes suivant regardant leur algorithme :

  — Lorsque le protocole se termine, ψ(i) est vrai à chaque noeud. Autrement dit, le système est dans un état légitime.

  — Lorsque le protocole se termine, chaque cluster est une alliance défensive.

  — A partir de n'importe quel état illégitime, la règle R4 sera exécutée au plus $n(\Delta + 1)$ fois.

  — L'algorithme converge vers une complexité temporelle de $O(\Delta^{2n2})$.

  A titre d'exemple, la Figure 2.5 montre le déroulement de l'algorithme :

  (a) -Tout d'abord, les noeuds 1, 8 et 9 se déclarent clusterheads selon la fonction $ch()$ lors de l'exécution de la règle R2.

  (b) -Les noeuds 2 et 3 qui sont adjacents au clusterhead 1 rejoignent ce cluster. De même,

le noeud 6 rejoint le cluster de 8. Les noeuds 10 et 7 sont adjacents à deux clusterheads, chacun d'entre eux sélectionne celui qui a le plus grand degré. Ainsi, ils rejoignent tous les deux le cluster 8.

(c) -Le noeud 4 a plus de voisins dans le cluster 1 que dans le cluster 8, il rejoint donc le cluster 1. Le noeud 5 a autant de voisins dans le cluster 1 que dans le cluster 8, il rejoint donc celui qui a l'id minimum selon la définition de la fonction $h(i)$, c'est-à-dire le cluster 1.

(d) -Enfin, Le noeud 9 a plus de voisins dans le cluster de 8 que dans son propre cluster, il quitte donc son statut de tête de cluster et rejoint le cluster de 8.

Finalement, deux clusters sont formés : l'un a le noeud 1 comme tête de cluster et l'autre a le noeud 8 comme tête de cluster

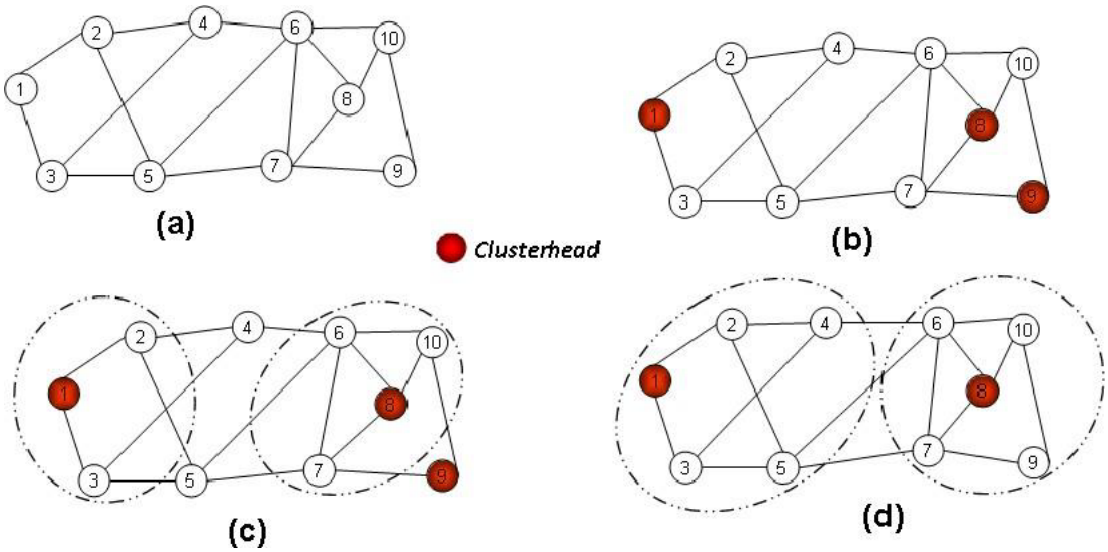


FIGURE 2.5 – Un exemple de clustering [33]

2.5.4 Partitionnement d'un graphe en alliances et son application au Data Clustering[34]

Shafique et al. [34] ont présenté un algorithme pour le partitionnement d'un graphe en alliances où chaque alliance est traitée comme un cluster et ils ont fourni des résultats expérimentaux sur son application aux problèmes de regroupement de données.

L'algorithme de Shafique et al. repose sur la prémisse que pour former un cluster optimal, l'algorithme doit choisir la meilleure partition parmi toutes les partitions satisfaisantes du graphe à chaque division. Il faut donc définir une certaine mesure de qualité de chaque partition.

Pour tout p, $0 \leq p \leq 1$, une p-alliance défensive, S, est un ensemble de sommets pour lequel le rapport entre le nombre de voisins à l'intérieur de l'ensemble S et la taille du voisinage est au moins p, c'est-à-dire que pour tous les sommets $v \in S, deg_S(v) \geq pdeg_{\overline{S}}(v)$. Notez que toute alliance défensive forte est une p-alliance, pour un certain $p \geq 0,5$. La mesure de la qualité d'une partition

en alliances défensives fortes, < A,B >, est définie par la valeur maximale $p \leq 1$, pour laquelle à la fois A et B sont des $p-$alliances. L'algorithme proposé génère une hiérarchie de clusters en divisant chaque cluster (en commençant par le cluster constitué de l'ensemble des données) en deux clusters plus petits (alliances défensives fortes) jusqu'à ce qu'aucun cluster ne puisse plus être partitionné. Pour expliquer leurs algorithmes, les auteurs ont adopté la notation suivante :

- I_n : matrice unitaire d'ordre nn.
- e_n : vecteur de dimension $n$de tous les uns, c'est-à-dire $e_n = [11..1]^T$.
- A(i, j) : élément de la i-ème ligne et de la j-ième colonne de la matrice A,, A(i, j) = A_{ij}
- tr(A) : trace de la matrice A, c'est-à-dire la somme des éléments diagonaux de la matrice A
- AB : produit interne de la matrice, $AB = tr(A^T B)$
- S_n : ensemble de toutes les matrices symétrique $n \times n$.
- M_n : ensemble de toutes les matrices nn.
- S_n^+ : ensemble de toutes les matrices semi-définies positives $A \in S_n^+, \Leftrightarrow A \geq 0$
- $Diag(x)$: matrice diagonale dont la diagonale est le vecteur x.
- $diag(A)$: vecteur diagonal de la matrice A.
- W : matrice de poids d'un graphe pondéré G.W_{ij} est le poids de l'arête entre les sommets v_i et v_j.

Dans un graphe G = (V,E), le but est de chercher une bipartition de l'ensemble des sommets V en ensembles A,B tel que A et B sont des p-alliances défensives et que p est maximal parmi toutes ces bipartitions. Dans le cas où il existe plus d'une telle partition, celle qui a le nombre minimum d'arêtes entre les ensembles A et B est prise. Une telle coupe est désignée comme étant une coupe maximale satisfaisante minimale du graphe G. Le problème est défini comme suit :
Coupe Maximale Satisfaisante Minimale (CMSM)
Comme entrée Un graphe G(V,E) et une fonction de poids $w : E \to R$. Le but est de trouver une partition A,B de V et un nombre réel $p, 0 \leq p \leq 1$, tel que :

1. $\forall v \in A, \sum_{u \in N(v) \cap A} w(u, v) \geq p \sum_{u \in N(v)} w(u, v)$,
2. $\forall v \in B, \sum_{u \in N(v) \cap B} w(u, v) \geq p \sum_{u \in N(v)} w(u, v)$,
3. p est le maximum parmi toutes les partitions, A,B, satisfaisant 1 et 2.
4. La coupure entre les ensembles A et B est minimale parmi toutes les partitions satisfaisant 1,2 et 3.

Afin de trouver une solution approximative du problème ci-dessus, Shafique et al. ont présenté une formulation de programmation quadratique de CMSM :
Soit $V = v_1, v_2, ..., v_n$. Pour chaque sommet $v_i \in V$, la variable x_i est défini. Le but étant de trouver une partition, A,B, de V en $p-$alliances défensives, $A = \{v_i | x_i = 0\}$ et $B = \{v_i | x_i = 1\}$ telle que p soit maximal parmi toutes ces partitions. Une partition A,B de V est une p-partition satisfaisante (une partition en p-alliances défensives, si et seulement pour chaque sommet v_i :

$$\sum_{v_j \in N(v_i)} w(v_i, v_j) \leq p(2x_i - 1) \sum_{v_j \in N(v_i)} \leq \sum_{v_j \in N(v_i)} x_j w(v_i, v_j) \leq 0$$

Il faut aussi que les deux ensembles A et B soient non vides. En fait, pour que les ensembles A

et B soient des alliances défensives fortes, chacun d'entre eux doit avoir au moins 2 sommets. Et donc $\sum_{i=1}^{n} x_i \geq 2$ et $\sum_{i=1}^{n} x_i \leq n$- 2

La fonction et les contraintes du programme proposée ont la forme d'une fonction quadratique générale, c'est-à-dire $x^T Qx + 2b^T x$. Les auteurs ont donc homogénéisé leur programme en introduisant une nouvelle variable x_0 et en la fixant à 1. Chaque terme linéaire ax_i est remplacé par le terme quadratique $ax_i x_0$. Ainsi, la version homogène du programme quadratique s'écrit comme suit :

Maximiser : $Kpx_0 - \sum_{1 \leq i < j \leq n} w(v_i, v_j)(x_i - x_j)^2$

Soumis aux contraintes suivantes :

- $p(2x_i - x_0) \sum_{v_j \in N(v_i)} w(v_i, v_j) - \sum_{v_j \in N(v_i)} x_j x_0 w(v_i, v_j) \leq 0, 1 \leq i \leq n$
- $p(2x_i - x_0) \sum_{v_j \in N(v_i)} w(v_i, v_j) - \sum_{v_j \in N(v_i)} x_j x_0 w(v_i, v_j) \geq - \sum_{v_j \in N(v_i)} w(v_i, v_j), 1 \leq i \leq n$
- $x_i^2 = x_i x_0, 1 \leq i \leq n$
- $\sum_{i=1}^{n} x_i x_0 \geq 2$
- $\sum_{i=1}^{n} x_i x_0 \leq n - 2$
- $px_0 \geq 0$
- $px_0 \leq 1$
- $x_0^2 = 1$

Le second terme,$Kpx_0 - \sum_{1 \leq i < j \leq n} w(v_i, v_j)(x_i - x_j)^2$ est la valeur de la coupe et sa minimisation correspond à la maximisation de la fonction objectif.

D'autre part l'augmentation de la valeur de p augmente le terme Kp et donc, la fonction objectif. K est une constante qui contrôle la précision du calcul de p et doit être choisie de telle sorte que l'augmentation de la valeur de p par la quantité de précision requise ait plus d'effet sur la valeur de la fonction objective que la valeur de l'une des coupures. En général,$K \gg \sum_{1 \leq i < j \leq n} w(v_i, v_j)$

2.5.5 Synthèse d'analyse

Dans cette section, nous présenterons une comparaison entre les différents travaux que nous avons étudiés.

Approches	Principe	Avantages	Inconvénients
H.Seba et al. [33]	Utilisation des alliances dans les graphes comme critère de mise en clusters pour la gestion des clés de groupe dans les MANET -L'algorithme organise le réseau en clusters de telle sorte que chaque cluster est une alliance dans le graphe qui représente le réseau.	-Approche très performante comparé à LID et SGCP. -Très bonne stabilité des clusters obtenus pour une mobilité faible moyenne et élevée	-L'algorithme proposé est très sensible aux attaques de type : Sybil, inondation par Hello, trou de ver.
Shafique et al. [34]	Présentation d'un algorithme pour le partitionnement d'un graphe en alliances où chaque alliance est traitée comme un cluster et ils ont aussi fourni des résultats expérimentaux sur son application aux problèmes de regroupement de données.	-L'algorithme est très performant.	-L'algorithme proposée présente de mauvais résultats en ce qui concerne le dataset des plantes d'iris de Firsher. L'algorithme de coupe normalisée a donné de meilleurs résultats pour cet ensemble de données.

Tableau 2.2 – Synthèse des travaux antérieurs sur les alliances dans les graphes

Du tableau, nous remarquons que les alliances sont adaptés à differents types de graphes, et qu'elles sont idéales en ce qui concerne le partitionnement d'un graphe, le regroupement de données et le clustering.

2.6 Conclusion

Ce chapitre a été consacré au regard de la littérature existante sur l'analyse des graphes d'attaques. Les modélisations présentées nous ont permises d'avoir un nouveau regard sur le problème d'analyse des graphes d'attaques et de mieux comprendre l'importance de ses derniers. Le prochain chapitre sera dédié à la présentation d'un modèle qui utilisera les alliances dans les graphes d'attaques.

Chapitre 3

Alliances défensives pour l'analyse des graphes d'attaques

3.1 Introduction

Une propriété fondamentale des graphes d'attaques est le degré de connexion entre les différents sommets du graphe. Les graphes d'attaques qui ont peu de connexions ou des connexions faibles sont plus faciles à défendre, tandis que ceux qui ont des connexions plus nombreuses et plus fortes sont plus difficiles à défendre. L'analyse des graphes d'attaques est une tâche ardue et complexe. La désactivation d'un ensemble de conditions initiales afin d'empêcher les attaques sur des cibles données peut entraîner des effets indésirables tels que le déni de service aux utilisateurs légitimes. Et étant donné que plusieurs chemins d'attaques peuvent conduire à la condition recherchée, une solution optimale pour renforcer le réseau n'est pas toujours apparente à partir du graphe d'attaques lui-même. Ainsi, chaque action peut avoir des effets supplémentaires en plus de l'effet recherché.
Dans ce chapitre, nous présenterons notre proposition "Application des alliances pour l'analyse des graphes d'attaques" dans le but de trouver des alliances défensives, et de résoudre les problèmes cités précédemment. Nous commençons par décrire notre approche ensuite nous allons présenter un algorithme qui résume ces étapes et nous concluons par un exemple de motivation.

3.2 Approche proposée

Notre approche consiste à analyser un graphe d'attaque afin de trouver des alliances défensives permettant de réduire les pertes de l'administrateur et l'aider à prendre une décision optimale afin de mieux sécuriser son réseau.

3.2.1 Algorithme

L'algorithme consiste à parcourir un graphe d'attaques et renvoyer les différentes alliances défensives qui atteignent le noeud but.

Pour chaque noeud sans prédécesseur, une liste de noeud est formé dans le but étant d'atteindre une condition cible. Pour chaque noeud sans prédécesseur nous ajoutons les noeuds dans le voisinage au degré le plus faible. L'algorithme s'arrête quand le noeud cible est atteint et renvoie les alliances défensives qui atteignent le noeud but.

Nous définissons tout d'abord les variables suivantes :

- F est un ensemble qui stocke les noeuds ajoutés.
- Une variable entière $Degre^{-}$, qui représente le nombre d'arcs dirigés vers le sommet S.

Nous définissons ensuite les fonctions suivantes :

- La fonction $tete(G)$ détermine les premiers noeuds à ajouter dans une liste dans le but de former des alliances. Les noeuds sans prédécesseur sont déclarés têtes de liste. La fonction $tete(G)$ renvoie les noeuds initiaux approprié si de tels noeuds existent et null sinon.
- *Voisinmin(S)* : retourne le noeud voisin non exploré ayant le degré minimum de S, $Voisinmin(S) = v \in S, \forall j \in S, deg_v < deg_j$ ou $(deg_v = deg_j)$. En cas d'égalité, renvoie un noeud au hasard. Si S est vide, $Voisinmin(S)$ retourne null.
- La fonction AllDef(F) qui vérifie si une liste passée en paramètres est une alliance défensive $|\forall i \in F, |N_F(i)|| \geq ||N_{\overline{F}}(i)||$.

Algorithm 1 Alliances défensives

Entrée : Un graphe d'attaques G(V,E) où V est un ensemble de conditions, et E est un ensemble d'exploits, une condition but C_t.
Sortie : Les alliances défensives menant à la condition but.

$Tete(G) = \emptyset$
Pour chaque v dans G **faire**
Si $Degre^{-}(v) = 0$ **alors**
$Tete(G) = Tete(G) \cup v$
Fin Si
Fin Pour
Retourner($Tete(G)$)

$F = \emptyset$
Pour chaque v dans $Tete(G)$ **faire**
Tant que $v \neq null$ **faire**
Si $C_t \in N(v)$ **alors**
$F = F \cup v$
$F = F \cup C_t$
Si AllDef(F) = Vrai **alors**
Retourner(F)
Fin Si
Sinon
$F = F \cup v$ // Ajouter le noeud
$v = VoisinMin(v)$ //Avancer vers le voisin au degré le plus bas.
Fin Si
Fin Tant Que
Fin Pour

3.2.2 Exemple d'application

Pour mieux comprendre notre proposition, nous appliquons notre démarche sur l'exemple de la Figure 3.1 qui présente un réseau composé d'un firewall qui sépare le monde interne du monde externe, un IDS qui voit le trafic entre le réseau interne et le monde externe, le réseau interne contient deux ordinateurs sous Unix, l'ordinateur A exécute les services ftp et sshd, l'ordinateur B exécute les services ftp et une base de données. L'intrus lance son attaque commençant par un seul ordinateur qui se trouve dans le réseau externe. Pour être plus concret, on suppose que le but de l'intrus est de perturber le fonctionnement de la base de données de l'ordinateur B, pour atteindre ce but, l'intrus a besoin d'un accès root sur la base de données de B.

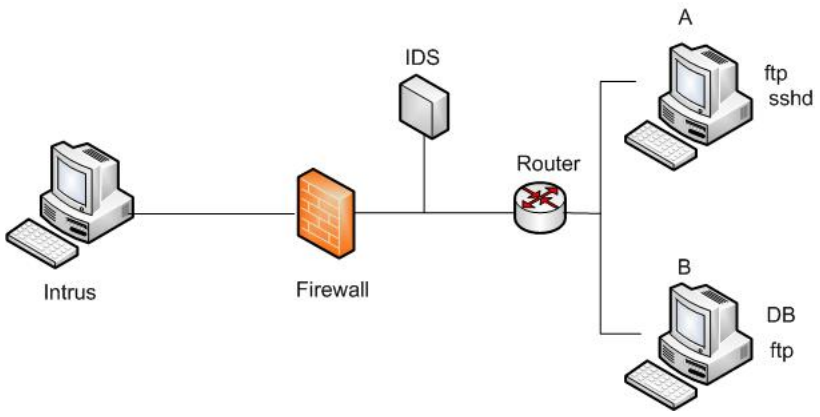


FIGURE 3.1 – Exemple de réseau [3]

Les actions de l'intrus sont :

1. **sshd buffer overflow :** Cette attaque exploite la vulnérabilité sshd qui donne immédiatement root shell dans la machine cible.
2. **ftp rhost :** Cette attaque utilise la vulnérabilité ftp. L'intrus crée un fichier rhost dans le répertoire home ftp, il crée ainsi une relation de confiance entre sa machine et la machine cible.
3. **remote login :** L'intrus utilise la relation de confiance entre deux machines et accède à user shell de la machine cible sans mot de passe.
4. **local buffer overflow :** Une fois que l'intrus a l'accès à user shell de la machine cible, la prochaine étape est l'exploitation de la vulnérabilité buffer overflow dans le fichier setuid root pour avoir un accès root.

Nous représentons cet exemple sous la forme d'un graphe d'attaques G(V,E) où V est l'ensemble de sommets représentant les conditions et E est l'ensemble d'arêtes représentant les exploits, nous appliquons donc l'algorithme précèdent :

- **Étape 1 :** Les noeuds sans prédécesseur(conditions initiales) sont marqués.

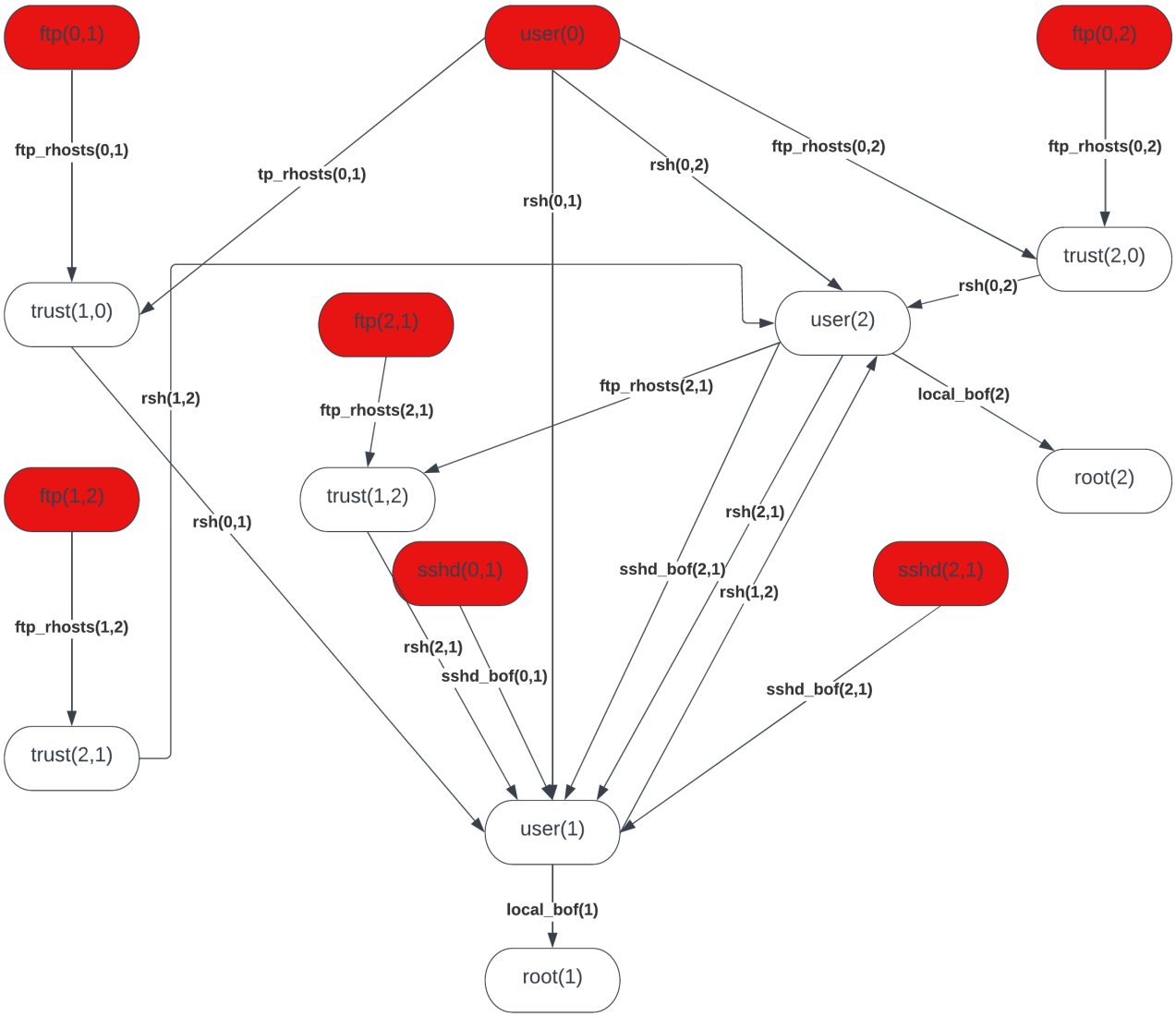


FIGURE 3.2 – Etape 1

- **Etape 2 :** Pour chaque noeud, nous parcourons le graphe d'attaques à la recherche d'une alliance qui atteint la condition cible.

FIGURE 3.3 – Etape 2

- **Etape 3 :** L'algorithme renvoie uniquement les alliances défensives qui atteignent le noeud but. La Figure (a) et (b) représentent deux chemins différents pour atteindre la condition cible root(2). La Figure(a) est un chemin allant de la condition ftp(0,2), les noeuds trust(2,0) et user(2) sont ajoutés avant d'atteindre la condition cible. La Figure(b) représente le chemin allant du protocole ftp(1,2), les noeuds trust(2,1) et user(2) sont ensuite ajoutés avant d'atteindre la cible.

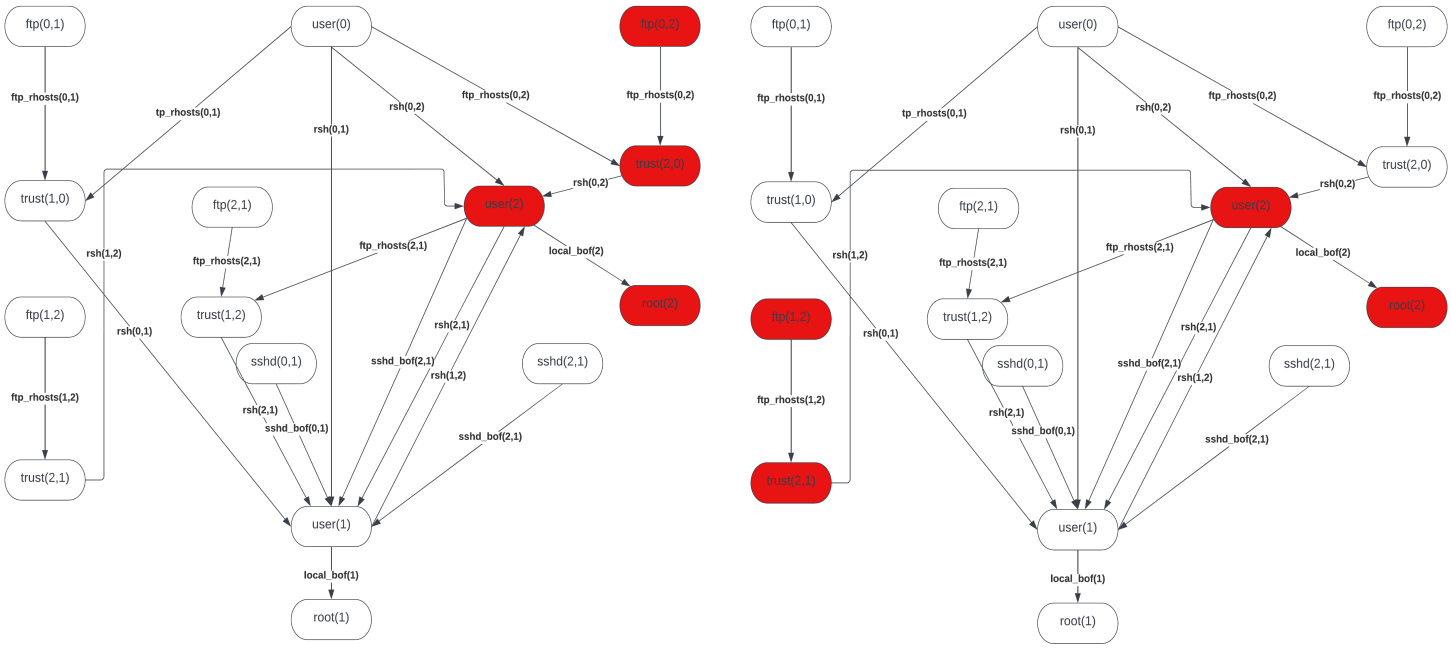


FIGURE 3.4 – Différentes alliances défensives qui atteignent le noeud but

Après l'application des alliances sur le graphe d'attaques nous constatons que {User(2),root(2)} est un chemin indispensable pour attaquer la machine 2. Nous concluons que la faille de ce réseau est l'existence du protocole FTP et les relations de confiance entre les machines 0 et 2 ainsi que la confiance entre les machines 2 et 1. Ainsi pour mieux sécurisé ce réseau, l'une des solutions est de supprimer les relations de confiance.

3.3 Conclusion

Au cours de ce chapitre, nous avons proposé un algorithme permettant de ressortir les alliances défensives dans un graphe d'attaques. Nous avons ensuite analysé le graphe en comparant les différentes alliances entre elles, et nous avons ressorti les vulnérabilités partagées.

Conclusion générale et perspectives

Avec la propagation des technologies informatiques, la protection des réseaux a acquis une place importante dans notre quotidien, nos activités professionnelles et personnelles en dépendent.

De ce fait, de nombreuses recherches ont été développées sur la sécurité informatique afin de pouvoir garantir la protection des ressources, en assurant les objectifs de la sécurité informatique. Beaucoup de recherches ont été effectuées en ce qui concerne l'analyse des graphes d'attaques et l'impact des alliances sur les graphes mais à notre connaissance, jamais l'application des alliances sur les graphes d'attaques.

Dans le cadre de ce travail, nous avons abordé un grand axe de recherche dans le domaine de la sécurité informatique qui est les graphes d'attaques, plus précisément nous avons abordé le problème d'analyse de ces derniers, nous avons aussi abordé le concept d'alliances dans les graphes le tout afin d'essayer d'unir ces deux concepts. Nous avons présenté une étude comparative sur les différentes méthodes d'analyse des graphes d'attaques et une synthèse concernant les alliances dans les graphes. Dans le but de concevoir une méthode permettant l'analyse des graphes d'attaques pour réduire les vulnérabilités du réseau, nous avons présenté un algorithme permettant de ressortir les différentes alliances défensives dans un graphe d'attaques, et nous avons essayé d'analyser les différentes alliances afin de déduire des vulnérabilités communes. Nous avons ensuite déroulé l'algorithme sur un exemple de graphe d'attaques, où nous avons détaillé chaque étape : Marquage des noeuds sans prédécesseur, parcours du graphe à la recherche d'une alliance qui atteint la condition cible, renvoie des alliances défensives atteignant le but.

Comme perspectives, nous proposons une analyse mathématique de l'impact des alliances sur les graphes d'attaques et l'implémentation de notre proposition.

Bibliographie

[1] M. Albanese, S. Jajodia, and S. Noel, *Time-efficient and cost-effective network hardening using attack graphs*, Institute of Electrical and Electronics Engineers/International Federation for Information Processing International Conference on Dependable Systems and Networks, (2012), pp. 1–12.

[2] K. Amano, K. M. Oo, Y. Otachi, and R. Uehara, *Secure sets and defensive alliances in graphs : A faster algorithm and improved bounds*, Institute of Electronics, Information and Communication Engineers Transactions on Information and Systems, 98 (2015), pp. 486–489.

[3] K. BOUAFIA and L. HAMZA, *Game theory approach for analyzing attack graphs*, Master's thesis, University of Bejaia, (2020).

[4] R. C. Brigham, R. D. Dutton, T. W. Haynes, and S. T. Hedetniemi, *Powerful alliances in graphs*, Discrete Mathematics, 309 (2009), pp. 2140–2147.

[5] R. C. Brigham, R. D. Dutton, and S. T. Hedetniemi, *Security in graphs*, Discrete applied mathematics, 155 (2007), pp. 1708–1714.

[6] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, *Nusmv 2 : An opensource tool for symbolic model checking*, International conference on computer aided verification, (2002), pp. 359–364.

[7] W. Evan, *Security risk management : Building an information security risk management program from the ground up*, Elsevier, (2011).

[8] H. Fernau and D. Binkele-Raible, *Alliances in graphs : a complexity-theoretic study*, SOFSEM(SOFtware SEMinar) (2), (2007), pp. 61–70.

[9] G. W. Flake, S. Lawrence, and C. L. Giles, *Efficient identification of web communities*, Proceedings of the sixth ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) International conference on Knowledge discovery and data mining, (2000), pp. 150–160.

[10] I. Gonzalez Yero, *Contribution to the study of alliances in graphs*, PhD thesis, Universitat Rovira i Virgili, Tarragona, Spain, (2010).

[11] L. Hamza, A. Rili, and K. Taraki, *A technique for analyzing attack graphs*, in In proceeding The Third International Conference on Multimedia Information Processing (CITIM), Mascara, (2018).

[12] A. Harutyunyan, *Global offensive alliances in graphs and random graphs*, Discrete Applied Mathematics, 164 (2014), pp. 522–526.

[13] T. Haynes, D. Knisley, E. Seier, and Y. Zou, *A quantitative analysis of secondary rna structure using domination based parameters on trees*, BMC bioinformatics, 7 (2006), pp. 1–11.

[14] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning, *Global defensive alliances in graphs*, the electronic journal of combinatorics, (2003), pp. R47–R47.

[15] T. Heberlein, M. Bishop, E. Ceesay, M. Danforth, C. Senthilkumar, and T. Stallard, *A taxonomy for comparing attack-graph approaches*, https ://core.ac.uk/display/101725694, (Consulté le 08 Octobre 2022), (2012).

[16] M. Hussein, *Energy efficiency in LEO satellite and terrestrial wired environments*, PhD thesis, (2016).

[17] S. Jajodia and S. Noel, *Topological vulnerability analysis a powerful new approach for network attack prevention, detection, and response. in algorithms, architectures and information systems security*, Cyber Situational Awareness, Advances in Information Security, Springer-Verlag US, 46 (2009), pp. 285–305.

[18] J.Almira and M.Sigarreta, *Alianzas en grafos*, PhD thesis, Universidad Carlos III de Madrid, (2007).

[19] S. Jha, O. Sheyner, and J. Wing, *Two formal analyses of attack graphs*, Proceedings 15th Institute of Electrical and Electronics Engineers Computer Security Foundations Workshop. CSFW-15, (2002), pp. 49–63.

[20] W. Jiang, H.-L. Zhang, Z.-H. Tian, and X.-f. Song, *A game theoretic method for decision and analysis of the optimal active defense strategy*, International Conference on Computational Intelligence and Security (CIS), (2007), pp. 819–823.

[21] M. Jun-chun, W. Yong-jun, S. Ji-yin, and C. Shan, *A minimum cost of network hardening model based on attack graphs*, Procedia Engineering, 15 (2011), pp. 3227–3233.

[22] P. Kristiansen, S. Hedetniemi, and S. Hedetniemi, *Alliances in graphs*, Journal of Combinatorial Mathematics and Combinatorial Computing, 48 (2004), pp. 157–178.

[23] K.Tariki and A.Rili, *Analyse des graphes d'attaques*, Master's thesis, University of Bejaia, (2015).

[24] K.-w. Lye and J. M. Wing, *Game strategies in network security*, International Journal of Information Security, 4 (2005), pp. 71–86.

[25] M.Powel, *Alliance in graph*, In Proceedings of the 255th of the USA Military Academy, Army Research Laboratory, Aberdeen Proving Ground, Harford County, MD, USA, (2004), p. 1350–1415.

[26] L. Muñoz-González, D. Sgandurra, M. Barrère, and E. C. Lupu, *Exact inference techniques for the analysis of bayesian attack graphs*, Institute of Electrical and Electronics Engineers Transactions on Dependable and Secure Computing, 16 (2019), pp. 231–244.

[27] S. Noel and S. Jajodia, *Understanding complex network attack graphs through clustered adjacency matrices*, 21st Annual Computer Security Applications Conference (ACSAC'05), (2005), p. 10.

[28] K. Ouazine, *Alliances dans les graphes : propriétés et application pour la réduction de la saturation et de la congestion dans les réseaux VANETs*, PhD thesis, (2018).

[29] K. Ouazine, H. Slimani, H. Nacer, N. Bermad, and S. Zemmoudj, *Reducing saturation and congestion in vanet networks : Alliance-based approach and comparisons*, International Journal of Communication Systems, 33 (2020), p. e4245.

[30] K. Ouazine, H. Slimani, and A. Tari, *Alliances in graphs : Parameters, properties and applications—a survey*, AKCE International Journal of Graphs and Combinatorics, 15 (2018), pp. 115–154.

[31] N. R. Patil and N. N. Patil, *A comparative study of network vulnerability analysis using attack graph*, Proceedings of National Conference on Emerging Trends in Computer Technology (NCETCT), (2012).

[32] J. A. Rodríguez-Velázquez, I. G. Yero, and J. M. Sigarreta, *Defensive k-alliances in graphs*, Applied Mathematics Letters, 22 (2009), pp. 96–100.

[33] H. Seba, S. Lagraa, and H. Kheddouci, *Alliance-based clustering scheme for group key management in mobile ad hoc networks*, The Journal of Supercomputing, 61 (2012), pp. 481–501.

[34] K. H. Shafique, *Partitioning a graph in alliances and its application to data clustering*, PhD thesis, University of Central Florida, (2004).

[35] K. H. Shafique and R. D. Dutton, *Maximum alliance-free and minimum alliance-cover sets*, Congressus Numerantium, (2003), pp. 139–146.

[36] O. Sheyner, *Scenario Graphs and Attack Graphs*, PhD thesis, Carnegie Mellon University, (2004).

[37] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian, *Computer-attack graph generation tool*, Proceedings Defense Advanced Research Projects Agency (DARPA) Information Survivability Conference and Exposition II. DISCEX'01, 2 (2001), pp. 307–321.

[38] L. Wang, A. Liu, and S. Jajodia, *Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts*, Computer communications, 29 (2006), pp. 2917–2933.

[39] I. G. Yero and J. A. Rodríguez-Velázquez, *Boundary defensive k-alliances in graphs*, Discrete Applied Mathematics, 158 (2010), pp. 1205–1211.

[40] I. G. Yero and J. A. Rodriguez-Velazquez, *Boundary powerful k-alliances in graphs*, Ars Combinatoria, 111 (2013), pp. 495–504.

Résumé

Avec l'explosion récente en terme de quantités de données, la sécurité des réseaux est plus importante que jamais. Les réseaux nécessitent donc un système de défense respectant la complexité des dépendances et les compétences des attaquants.
Depuis l'introduction des alliances en 2002, bien des recherches ont été publiées sur leurs différentes propriétés mathématiques, leurs effets sur les graphes et notamment sur le clustering. Cependant, à notre connaissance, aucune recherche n'a été effectuée en ce qui concerne leur application sur les graphes d'attaques.
Dans ce mémoire, nous proposons une nouvelle méthode pour l'analyse des graphes d'attaques, basé sur l'application des alliances dans les graphes. L'approche proposée permet de renvoyer les différentes alliances défensives dans les graphes d'attaques. Nous comparons ensuite les différentes alliances et nous ressortons les vulnérabilités communes.
Mots Clés : Graphes D'attaques, Vulnérabilité, Alliance Défensive, Sécurité Informatique.

Abstract

With the recent explosion in the volume of data, network security is now more important than ever. Therefore networks require a defense system that respects the complexity of dependencies and the skills of attackers. Since the introduction of alliances in 2002, a lot of research has been published on their different mathematical properties, their effects on graphs and on clustering. However, to our knowledge, no research has been done on their application to attack graphs.
In this thesis, we propose a new method for the analysis of attack graphs, based on the application of alliances in graphs. The proposed approach return the different defensive alliances in the attack graphs. We then compare the different alliances and highlight common vulnerabilities.
Key words : Attack Graphs, Vulnerability, Defensive Alliance, Computer Security.