MARFALL N'DIAGA FALL

Sécurisation formelle et optimisée de réseaux informatiques

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de Maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

Octobre 2010

Résumé

Les pare-feu sont des éléments cruciaux pour le renforcement de la politique de sécurité d'un réseau. Ils ont été largement déployés pour sécuriser les réseaux privés, mais leur configuration reste complexe et sujette de plusieurs anomalies. Au cours des dernières années, plusieurs techniques et outils ont été proposés pour configurer correctement un pare-feu. Cependant, la plupart de ces techniques restent informelles et ne prennent pas en compte la performance globale du réseau ou d'autres paramètres de qualités de services. Dans ce mémoire, nous introduisons une approche formelle permettant de configurer optimalement et formellement un réseau de sorte qu'il respecte une politique de sécurité donnée tout en tenant compte de ses qualités de services.

Abstract

Firewalls are crucial elements in enforcing network security policies. They have been widely deployed for securing private networks but, their configuration remains complex and error prone. During the last years, many techniques and tools have been proposed to correctly configure firewalls. However, most of existing works are informal and do not take into account the global performance of the network or other qualities of its services (QoS). In this thesis we introduce a formal approach allowing to formally and optimally configure a network so that a given security policy is respected and by taking into account the QoS.

Avant-propos

Je tiens à remercier tous ceux ou celles qui, de près ou de loin, ont contribué à la réussite de ce travail. Je désire remercier tout particulièrement mon directeur de recherche, le professeur Mohamed Mejri sans qui cette aventure n'aurait vu son épilogue si tôt. J'ai beaucoup appris avec toi et tu m'as fait aimer le travail propre. Ta modestie, ton support, tes conseils et tes encouragements m'ont permis d'arriver à ce stade. J'ai adoré ta façon de me guider tout en me laissant la liberté de décider. Merci pour tout.
J'exprime également toutes mes reconnaissances et ma gratitude aux professeurs Ronald Beabrun et Kamel Adi pour avoir accepté d'évaluer ce mémoire.
Merci à Mahjoub Langar pour ses conseils inestimables et toutes les discussions si riches que nous avons eues à plusieurs reprises.
Mention spéciale à Khadidja Sylla, qui a pris le temps de relire et de corriger tout ce mémoire, chapitre par chapitre afin de le dépouiller de toutes sortes d'erreurs.

Mes remerciements vont également à l'encontre de tous ceux et celles qui ont rendu agréable ce séjour à Québec. Marième, Ishagh, Kadi, Maleye, Rodrigue, Khadidja, Fatima, Abdou Diop, Pierre Marchand et tous les collègues du LSFM.

Merci à Louma et Fatim, je n'oublierai jamais les vacances passées avec vous.

Au-delà de cette harmonie, ce rêve ne serait une réalité sans assistance financière . Je tiens donc à remercier le PCBF (Programme Canadien de Bourses de la Francophonie) qui a couvert en intégralité tous les frais reliés à mes études et mon séjour . Je remercie particulièrement sa gestionnaire Madame Jeanne Gallagher qui est comme une maman pour moi. Je remercie également la direction de l'enseignement supérieur de la Mauritanie de m'avoir choisi pour cette bourse.

Enfin, mon dernier et plus gros mot de remerciement va à celle qui ma donnée naissance ainsi qu'à tous les membres de ma famille à commencer par mon oncle Moustapha et mon frère Dame qui n'ont jamais cessé de m'encourager. Grâce à vous, je n'ai jamais senti l'absence de papa.

Je dédie ce mémoire à très chère maman "AWA FALL".

« Slow and methodical testing like this will eventually get you to a point where you will either be able to spot the bug, or go quietly insane. Maybe both. »

[*Documentation du logiciel Persistence of Vision* 2.1]

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

Contexte

Les technologies informatiques ont acquis une place importante dans notre quotidien, nos activités professionnelles ainsi que nos temps libres. Les ordinateurs nous permettent de stocker une grande quantité de données et d'y effectuer une variété de tâches. De nouveaux appareils apparaissent sur le marché et élargissent l'utilisation des informations numériques. Ce progrès nous facilite grandement les activités menées par d'autres moyens, et fournit de nouvelles façons de communiquer et de travailler. En retour, plusieurs risques liés à la sécurité ont augmenté. Chaque nouvelle pratique vient avec de nouvelles menaces d'où l'importance de former et de donner de bons outils à l'ensemble des intervenants concernés. Ainsi, la sécurité informatique est en pleine et constante évolution.

En effet, depuis l'avènement d'Internet, la liste d'intrusions dans les systèmes informatiques et les vols d'informations via ce réseau ne cessent de s'allonger. Internet a donné une parfaite vitrine aux commerçants du monde entier qui y trouvent l'occasion de profiter d'un marché virtuel mondial. Il a également suscité de nombreuses idées chez tous ceux qui cherchent de l'argent facile et ceux qui trouvent un grand plaisir à saccager les sites informatiques des autres. De ce fait, la protection des réseaux informatiques est devenue aujourd'hui un enjeu crucial, autant pour la confidentialité et l'intégrité des informations que pour simplement maintenir le bon fonctionnement des systèmes. La sécurité est indispensable et même vitale pour assurer le bon fonctionnement des systèmes d'information.

Durant ces dernières décennies, de nombreux travaux ont été réalisés et d'importants outils (matériels ou logiciels) ont été développés dans le but de rendre les réseaux plus sécuritaires. Les pare-feu (*firewalls*), les systèmes de détection et de prévention d'intrusions (IDS/IPS) par exemple, ont des mérites substantielles dans la sécurisation des réseaux informatiques. Cependant, se procurer les outils les plus récents et les plus performants liés à la sécurisation des réseaux informatiques est loin d'être suffisant pour réduire les risques d'intrusions. En effet, le maillon le plus faible dans la chaîne de la sécurité informatique demeure l'intervention humaine qui est généralement nécessaire pour installer et configurer ces outils. Limiter cette intervention permettra sans doute de réduire à la fois les risques et les coûts engendrés par la sécurité.

Problématique

« Même si la sécurité informatique ne se limite pas à celle du réseau, il est indéniable que la plupart des incidents de sécurité surviennent par les réseaux, et vise les réseaux »[1]. Il est donc important de développer des méthodes sûres permettant de configurer automatiquement un réseau informatique de sorte que son comportement soit conforme à une politique de sécurité donnée. C'est dans cet axe de recherche que se situe ce travail. En effet, nous proposons une approche formelle permettant de générer à partir d'une politique de sécurité (spécifiée par une formule logique) et d'un réseau informatique (spécifié par un graphe étiqueté) une configuration sécuritaire et optimale du réseau initial.

Objectifs et méthodologie

La recherche sur le renforcement de politique de sécurité a été très active ces dernières années et de nombreuses approches ont été proposées. Néanmoins, nous prétendons qu'aucune d'elles n'est pleinement satisfaisante au regard des exigences actuelles.
L'objectif de ce travail est double. Dans un premier temps, nous allons faire une synthèse de l'état de l'art sur les différents outils et techniques de sécurisation d'un réseau informatique. Par la suite, nous allons définir une approche algébrique pour générer automatiquement, à partir d'une configuration initiale et d'une politique de sécurité, une nouvelle configuration qui respecte la politique donnée.
Plusieurs configurations peuvent répondre aux besoins sécuritaires et dans ce cas la configuration choisie devrait être celle qui offre les meilleures qualités de services. Ainsi, nous adoptons la méthodologie suivante :

- Étudier les différentes techniques et outils visant à sécuriser les réseaux informatiques et en faire une synthèse.
- Modéliser le réseau à sécuriser en utilisant un graphe orienté et multi-étiqueté ;
- Définir un langage pour spécifier une politique de sécurité et un langage pour spécifier un pare-feu mettant en oeuvre cette politique ;
- Définir une notion de coût (relatif au débit d'un réseau) d'une solution de sécurité afin de pouvoir calculer la configuration à coût minimal ;
- Définir un opérateur de renforcement générant une configuration sécuritaire en tenant compte de la performance et de l'aspect dynamique d'un réseau.

Organisation

La suite de ce mémoire est est organisée comme suit :

- Le chapitre 2 est une introduction générale à la sécurité des réseaux informatiques, ses enjeux, les outils et leurs modes de fonctionnement.
- Le chapitre 3 est consacré aux différents modes de spécification des réseaux. Nous y abordons principalement les structures de graphes et les algèbres de processus. L'étude comparative de ces deux approches nous a permis de définir un modèle de graphe étiqueté pour modéliser un réseau.
- Le chapitre 4 dresse l'état de l'art sur le renforcement de politique de sécurité d'un réseau. Nous y exposons particulièrement les politiques de contrôle d'accès ainsi que les différents types d'anomalies qui compromettent la configuration des pare-feu. Notre but n'est pas de détailler chaque technique individuellement, mais plutôt de nous concentrer sur l'idée principale et de décrire l'approche afin de pouvoir la comparer à la notre.
- Le chapitre 5 est le chapitre principal de ce travail. Nous y définissons une logique pour la spécification d'une politique de sécurité et un graphe étiqueté par des pare-feu (spécifiés formellement) pour modéliser un réseau. Nous avons ensuite introduit une notion de coût relatif aux critères de performance d'un réseau. Cette fonction est ensuite utilisée pour définir un opérateur de renforcement générant une configuration sécuritaire et optimale du réseau. Un exemple complet illustre toute l'approche.
- Les conclusions et perspectives liées à cette recherche font l'objet du chapitre 6.

Première partie

État de l'art

Chapitre 2

Sécurité des réseaux informatiques

Introduction

Les réseaux informatiques deviennent de plus en plus complexes, dynamiques et hétérogènes. Cette situation a d'énormes conséquences sur leur sécurité. En effet, comme tout système informatique, un réseau doit assurer la confidentialité, l'intégrité et la disponibilité de ses données. Cet objectif justifie à lui seul la nécessité d'accorder une attention particulière à la sécurisation des réseaux informatiques. Il faut notamment protéger l'accès et la manipulation des données et autres ressources du réseau par des mécanismes d'authentification, d'autorisation et de contrôle d'accès. Toutefois, il est estimé que la plupart des malveillances informatiques ont une origine ou complicité interne aux organismes. Devant une telle spécificité il est donc essentiel d'élaborer une politique de sécurité et la mettre en oeuvre dès la conception même des infrastructures.

La sécurisation d'un réseau nécessite un long processus qui remonte jusqu'à la conception de son architecture. Comme tout système complexe, la conception d'un réseau d'ordinateurs nécessite une attention particulière afin de mettre en oeuvre une politique globale de sécurité. Il faudra notamment choisir les dispositifs appropriés et les placer aux bons endroits. Parmi ces dispositifs, on peut citer les pare-feu et les systèmes de détection et de prévention d'intrusions (IDS/IPS).

Dans ce chapitre, nous étudions quelques principes de conception sécuritaire. Nous commençons par décrire les différents aspects de la sécurité d'un réseau et définir la notion de politique de sécurité avant d'explorer plusieurs stratégies permettant de la mettre en oeuvre. Nous verrons notamment les dispositifs cités ci-haut ainsi que différents mécanismes de segmentation.

2.1 Aspects de la sécurité

2.1.1 Enjeux de la sécurité réseau

Durant les dernières décennies, les systèmes informatiques ont enregistré une évolution considérable et offrent d'énormes possibilités de traitement automatique de l'information. Mais l'accroissement de ces possibilités a entrainé celle des risques de divulgation d'informations. En outre, cette évolution a affecté plusieurs aspects des réseaux :

– Les architectures sont de plus en plus distribuées ;

– Les tailles des dispositifs de stockage d'information sont devenues immenses ;

– La nécessité de partager des ressources est de plus en plus croissante ;

– etc.

Devant de telles exigences, plusieurs dispositifs ne sont plus adéquats pour assurer une bonne protection. Les réseaux sont ainsi exposés à des attaques si nombreuses qu'il serait illusoire de prétendre les décrire toutes. Il est cependant possible de les classer en fonction des objectifs des pirates qui reposent sur les faiblesses de sécurité [1].

– **Attaques permettant de dévoiler le réseau :** Ces attaques visent souvent à établir la cartographie du réseau, identifier ses systèmes par des balayages ICMP ou TCP, identifier ses routeurs, traverser ses équipements de filtrage, etc.

– **Attaques permettant d'écouter le trafic réseau :** Ces attaques sont utilisées pour capturer des informations telles que les mots de passe. Parmi elles, le *sniffing*.

– **Attaques permettant d'interférer avec une session réseau :** Ces attaques exploitent les faiblesses des protocoles réseau. On peut en noter les usurpations d'adresses (*ARP spoofing et IP spoofing*) et les attaques dites *man-in-the middle* (l'homme au milieu).

– D'autres types d'attaques telles que celles permettant de modifier le routage d'un réseau sont décrites dans [1].

Pour faire face à ces multitudes d'attaques, il faut mettre en place un ensemble de mécanismes de protection pour assurer les propriétés suivantes :

- **Confidentialité :** La confidentialité est la protection des données contre toute forme de divulgation. En d'autres termes, c'est l'assurance que l'information est inaccessible à une entité ou un processus non autorisé.
- **Intégrité :** L'intégrité est la capacité de protéger les données contre toute modification non autorisée.

- **Disponibilité :** La disponibilité assure la protection contre la rétention non autorisée d'une information ou une ressource. C'est donc la garantie que les données sont disponibles dans les conditions normales de temps, de délai et de performance définies.

Ces propriétés sont fondamentales à la sécurité de tout système informatique. On pourrait également en ajouter d'autres telles que :

– **L'authentification** : S'assurer qu'une entité ou un processus est bien celui qu'il prétend être.

– **La vérifiabilité** : Fournir suffisamment d'informations relatives aux activités d'un système.

– **La non-répudiation** ou **l'irrévocabilité** : S'assurer qu'une entité ou un processus engagé dans une communication ne pourra nier avoir reçu ou émis un message.

Toute action, volontaire ou accidentelle, pouvant porter atteinte à une de ces propriétés doit être interceptée. D'où la nécessité de définir une politique globale de sécurité.

2.1.2 Notion de politique de sécurité

Une politique de sécurité d'un réseau est un ensemble de règles visant à protéger ses ressources d'éventuels incidents de sécurité dommageables pour son activité. Elle détermine pour chaque action entreprise si elle doit être autorisée ou non.
Dans [2], Stouls *et al.* définissent trois types de politique :

1. **Politique restrictive :** Elle consiste à spécifier les actions qui sont autorisées. Tout ce qui n'est pas explicitement autorisé est alors interdit.
2. **Politique permissive :** À l'opposé de la restrictive, elle décrit les actions interdites. Tout ce qui n'est pas explicitement interdit est alors autorisé.
3. **Politique combinatoire :** Contrairement aux deux premières, une politique combinatoire consiste à définir aussi bien les actions autorisées que celles interdites.

Une politique doit être complète et cohérente. La complétude est l'assurance qu'il n'y a aucune action qui n'est ni interdite ni autorisée. La cohérence garantit qu'il n'existe pas d'action autorisée et interdite par la même politique. Pour assurer de telles propriétés, il faut analyser tous les éléments critiques du réseau et identifier les besoins de sécurité. Ainsi, il faudra associer à toute entité vitale, ses menaces, sa vulnérabilité et ses conséquences. Après avoir défini les objectifs et le contenu d'une politique de sécurité réseau, nous détaillons dans les sections qui suivent quelques techniques pour sa mise en oeuvre.

2.2 Pare-feu (*firewall*)

Un pare-feu est un dispositif utilisé pour empêcher les accès non autorisés à un réseau. Sa fonction est double : renforcer une politique de sécurité et journaliser un trafic réseau. Le renforcement d'une politique de sécurité consiste à décider s'il faut accepter ou rejeter une connexion selon des règles spécifiques de filtrage permettant de forcer un réseau à se conformer à une politique donnée. La journalisation quant à elle, consiste à enregistrer tous les aspects du traffic afin de pouvoir mieux l'analyser. Un pare-feu est donc un composant clé pour la conception d'un réseau sécurisé.
Cependant, étant un point de passage pour tout le trafic réseau, un pare-feu peut aussi être un unique point de défaillance. Par conséquent, son choix ainsi que son emplacement sont d'importantes tâches pour la sécurité des infrastructures réseau.

2.2.1 Classification des pare-feu

Il existe différents types de pare-feu que nous allons décrire en les classant selon la méthode de renforcement utilisée, la stratégie de gestion du trafic réseau ou la configuration physique des dispositifs [3].

1. **Classement selon la méthode de renforcement utilisée** :
   Lorsqu'ils sont classés selon la méthode de renforcement utilisée, la plupart des pare-feu entrent dans une des trois catégories suivantes :

   – **Filtrage de paquets** (*packet-filtering firewall*) : Un pare-feu de filtrage de paquets opère au niveau de la couche réseau. Il examine le contenu des paquets IP et filtre le trafic en fonction des adresses, ports et autres options des paquets. Le fait d'opérer au niveau réseau lui procure une performance assez élevée car le trafic réseau passe sans délai notable. Ce type de pare-feu est alors une excellente solution lorsque la performance est une exigence importante. Par exemple, la conception d'un réseau qui doit accueillir une application web telle qu'un site de *e-commerce*.

   – **Circuit de passerelles** (*circuit gateway firewall*) : Un pare-feu à circuit de passerelle opère au niveau de la couche transport. Il filtre également le trafic en fonction des adresses. Son principal objectif est de créer un circuit virtuel entre les hôtes source et destination afin d'avoir une connexion plus transparente. Cependant, sa mise en oeuvre requiert des "*sockets*" pour garder une trace des connexions séparées. Ce qui nécessite un "*socket-client*" compatible sur le système de l'hôte source.

- **Application proxy** (*application-proxy firewall*) : Un pare-feu d'application de proxy oeuvre au niveau application et contrôle toutes les connexions entrantes et sortantes du réseau. Si une connexion est autorisée, l'application-proxy l'initie vers l'hôte destination au nom de l'hôte source. Ce type de pare-feu est capable de s'assurer que le trafic qui le traverse est conforme à la politique de sécurité et que les fonctions au sein d'un protocole ou d'une application sont conformes aux politiques spécifiées. Il est donc plus sécuritaire que le filtrage de paquets et masques les vraies adresses du réseau, mais malheureusement il est presque impossible d'attribuer un proxy à toutes les applications existantes.

2. **Classement selon la stratégie de gestion du trafic réseau** :
   On distingue principalement deux types de pare-feu classés selon leur manière de gérer le trafic réseau :
   - **Pare-feu de routage** (*routing firewall*) : Ce type de pare-feu est généralement localisé entre différents réseaux et est responsable du routage des paquets. Un tel pare-feu est un point unique par lequel passe tout le trafic réseau. Ces principaux avantages sont surtout la translation d'adresses et la facilité de sa gestion.

   - **Pare-feu de pontage** (*bridging firewall*) : Un pare-feu de pontage est une configuration relativement nouvelle qui construit un pont sur le trafic au niveau de la couche liaison de données. Il peut s'insérer dans n'importe quel environnement réseau sans d'importantes reconfigurations des passerelles par défaut. Un autre avantage est que le pare-feu n'a pas besoin d'avoir une adresse IP et le dispositif est totalement transparent. Cependant cette configuration s'avère difficile à gérer.

3. **Classement selon la configuration physique des dispositifs :**
   Lorsqu'on classe les pare-feu selon leurs configurations physiques, on peut distinguer deux types de dispositifs :
   - **Pare-feu basé sur un serveur** (*Server-based firewall*) : Un pare-feu basé sur un serveur opère au niveau d'un système d'exploitation sécurisé ou spécialement modifié tel que UNIX, Windows, Solaris, etc. Les avantages de ce type de pare-feu sont multiples : il peut être personnalisé facilement. Il a un haut degré de complexité car s'exécutant sur le matériel de base au sommet d'un système d'exploitation. Sa forte mémoire interne et sa facilité de mise à jour. Cependant, il est vulnérable à toute faiblesse découverte au niveau de la plate-forme du système d'exploitation.

   - **Applications de pare-feu** (*Firewall Appliance*) : Les applications de pare-feu sont des dispositifs matériels spécialement conçus pour exécuter des systèmes d'exploitation propriétaires. Ce type de pare-feu devient de plus en plus répandu et

a pour principale force le fait qu'une bonne partie de la logique du réseau et des fonctions du pare-feu se produisent sur le matériel spécialement conçus et non sur la pile réseau du système d'exploitation. Ce qui permet à ces pare-feu de gérer le trafic à des vitesses élevées et en quantités plus importantes que peut un pare-feu basé sur le serveur. Malheureusement, la faiblesse de ces pare-feu vient également de cette plate-forme qui le rend moins extensible et évolutif qu'un pare-feu basée sur un serveur.

2.2.2 Emplacement d'un pare-feu

Après le choix du pare-feu approprié, son emplacement nécessite également une attention particulière. Un pare-feu n'est pas une solution magique qui résoudra tous les problèmes de sécurité. Son emplacement doit être choisi selon les fonctions et objectifs envisagés. Il peut être localisé à l'extérieur (sur la périphérie) comme à l'intérieur du réseau qu'il protège.

– **Emplacement périphérique**
Étant un dispositif qui doit empêcher tout accès non autorisé, un pare-feu est le plus souvent placé comme intermédiaire entre le réseau qu'il protège et l'extérieur. Cette disposition simple lui permet d'inspecter tout le trafic en provenance ou vers le réseau protégé. La Figure 2.1 illustre une telle disposition.

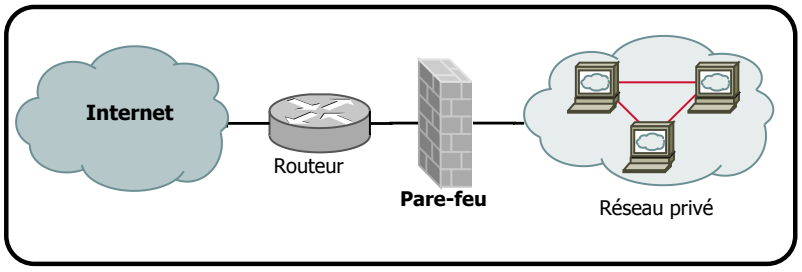


Figure 2.1 – Emplacement d'un pare-feu

– **Emplacement interne**
Un pare-feu peut aussi être placé à l'intérieur d'un réseau pour le diviser en plusieurs sous-réseaux et contrôler les différents accès entre eux. Ainsi le pare-feu fournit une protection interne entre les différents segments du réseau. Il peut être configuré pour octroyer des accès privilégiés ou restreintes. Une telle disposition (Figure 2.2) nécessite

une puissance de traitement élevée ainsi qu'un dispositif avec plusieurs interfaces. Ce qui est plus coûteux qu'un commutateur normal.

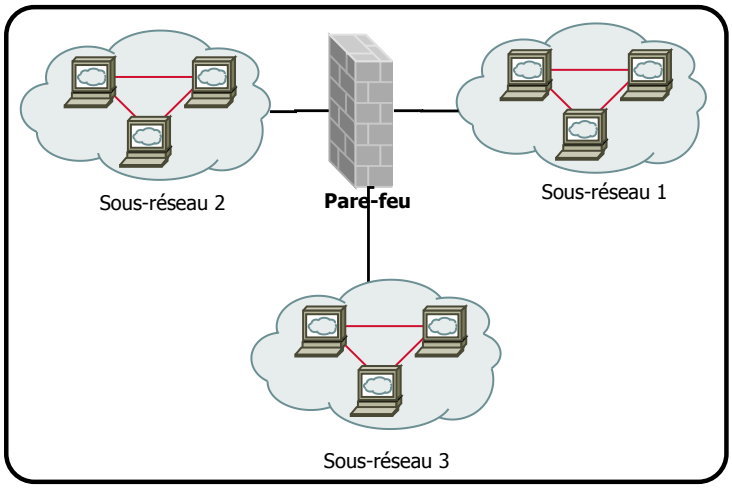


Figure 2.2 – Un pare-feu au centre d'un réseau

2.3 Systèmes de détection et de prévention d'intrusions

Malgré la mise en place des pare-feu, les utilisateurs malveillants tentent de contourner les politiques de sécurité. Il est donc du ressort de l'administrateur d'analyser régulièrement l'état du système et de vérifier qu'il n'a pas été compromis. Cette tâche d'audit suppose un mécanisme d'enregistrement des événements du système au sein de « journaux » et une phase d'analyse de ces journaux afin d'identifier d'éventuelles violations. La détection d'intrusion est née de la nécessité d'automatiser cette tâche d'audit des systèmes informatiques. Ainsi, les premiers systèmes de détection d'intrusions (IDS pour *Intrusion Detection Systems*) avaient pour objectif d'analyser le trafic réseau contre des modèles d'attaques connues et de déclencher une alarme lorsqu'un modèle est rencontré. L'administrateur peut ensuite élaborer un plan de défense contre l'éventuelle attaque. Mais la relative naïveté des algorithmes de détection conduit à un nombre élevé d'alertes, dont une bonne partie n'est constituée que de fausses alertes (faux positifs) alors que certaines intrusions ne sont pas détectées (faux négatifs).
D'énormes améliorations ont été introduites et les nouveaux dispositifs produisent moins de fausses alertes. Dans certains cas, ils ont même la possibilité de stopper les intrusions. D'autres IDS vont plus loin dans la prévention en reconfigurant les listes de contrôle

d'access (ACL) des routeurs et en réécrivant dynamiquement la politique du pare-feu afin d'exclure les paquets suspects. Cette nouvelle génération d'équipements combine les fonctionnalités des IDS et des pare-feu, ce qui leur procure le nom de « Système de prévention d'intrusions »(IPS pour *Intrusion Prevention Systems*).

2.3.1 Approches de détection et de prévention d'intrusions

Les IDS/IPS utilisent principalement deux approches pour détecter une intrusion :

1. **Approche basée sur la connaissance :** Cette approche se base sur la connaissance des techniques employées par les attaquants. On en tire des scénarios d'attaque et on cherche leur éventuelles causes dans les traces d'audits. Cette technique fournit des informations précises sur la signature des intrusions. Cependant, la base de connaissance du système est à mettre à jour continuellement selon les dernières signatures. Elle est également source de faux négatifs car une attaque n'est détectée que lorsque sa signature est dans la base de connaissance.

2. **Approche basée sur le comportement :** Cette approche est fondée sur l'hypothèse qui consiste à définir le comportement normal de l'utilisateur et du système. Toute déviation ou action inhabituelle par rapport à ce comportement sera considérée comme suspecte. Ainsi, une alerte est déclenchée lors qu'un nouveau comportement est détecté. Une telle approche peut détecter beaucoup d'intrusions mais n'est pas aussi précise que la base de connaissance et peut engendrer beaucoup de fausses alertes car elle ne supporte aucune déviation par rapport au comportement préalablement décrit. Ce qui peut résulter d'une simple évolution du système.

Chacune de ces approches présente aussi bien des avantages que des limites. La coopération entre les différents outils existants s'avère donc essentielle pour réduire ces limites. Pour y parvenir, L. Mé *et al.* définissent les préalables d'une telle coopération dans [4].

2.3.2 Emplacement des IDS/IPS

La position optimale pour un IDS/IPS dépend de ses fonctions et caractéristiques. Un IDS de surveillance passive a besoin d'une position lui permettant d'analyser le trafic réseau et de déclencher une alerte à travers un canal prédéfini tel qu'une console, un courriel, etc. La meilleure position d'un tel IDS est derrière le pare-feu et près des don-

nées protégées. Par contre, un IDS/IPS capable de stopper les attaques devrait être déployé entre le routeur et le pare-feu.
Les IDS sont classés selon le type de protection qu'ils fournissent. Ainsi, on peut distinguer un HIDS (*Host Intrusion Detection System*) qui surveille un seul poste d'un NIDS (*Network Intrusion Detection System*) qui contrôle le trafic de tout un réseau.

1. **HIDS**
   Un HIDS est généralement un programme installé sur un hôte nécessitant une protection particulière. Il examine toutes les activités de l'hôte qu'il surveille et relève tout ce qui compromet sa sécurité pour le stopper ou pour déclencher une alerte. Un HIDS s'exécute généralement sur un serveur. En effet, les données d'un serveur change constamment mais sa configuration et ses fonctions restent statiques tandis qu'un poste de travail peut changer à tout moment. Un tel poste n'est pas moins exposé qu'un serveur situé derrière un pare-feu, mais y installer un HIDS demeure un véritable challenge.
   La Figure 2.3 illustre l'emplacement des HIDS. On peut noter la présence d'un HIDS sur tous les serveurs. Nous avons également placé la console loin de ces serveurs « pour ne pas garder une clé avec une serrure qu'elle ouvre ». La console est aussi munie d'un HIDS pour la fortifier.

2. **NIDS**
   Un NIDS est habituellement un équipement connecté à un réseau pour surveiller le trafic. Il a besoin de voir ce trafic afin de pouvoir l'analyser. On peut donc le placer avant ou après le pare-feu périphérique.

   – **Avant le pare-feu :** Un NIDS placé avant le pare-feu peut voir toutes les attaques possibles survenant contre le réseau à partir du moment où elles commencent. Ce qui donne à l'administrateur un temps maximal pour réagir et empêcher tout dommage. En effet, un NIDS n'a pas besoin de console. L'administrateur réseau le configure directement à travers une interface de lignes-de-commande ou un navigateur web pour recevoir les alertes. Cependant, un NIDS placé avant le pare-feu (Figure 2.4) signale des attaques que ce dernier peut facilement stopper. Ce qui entraîne beaucoup de fausses alertes.

   – **Après le pare-feu :** Pour signaler uniquement les attaques entrants dans le réseau protégé, un NIDS doit être placer après le pare-feu. Un tel emplacement permet également de détecter des attaques que le précédent (Figure 2.4) manquerait. En effet, de nombreux réseaux ont des VPNs (*Virtual Private Networks*, réseaux privés virtuels) qui s'étendent jusque derrière leurs pare-feu. Ainsi, toute attaque acheminée par un client VPN compromis pourra contourner l'ensemble des pare-feu ainsi

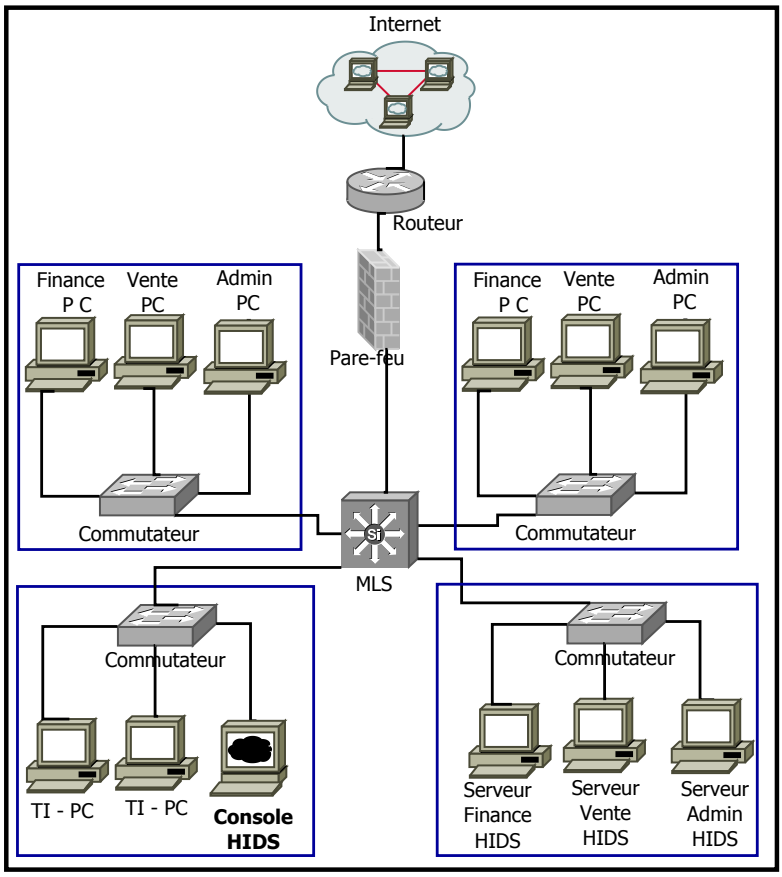


Figure 2.3 – Emplacement d'un HIDS

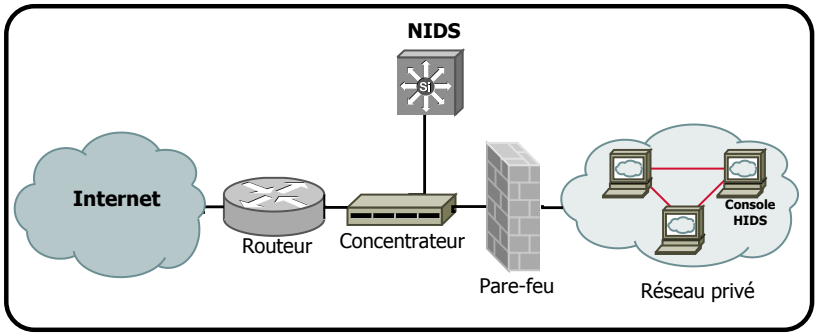


Figure 2.4 – Emplacement d'un NIDS

que le NIDS qui les devance. Limiter le VPN avant le pare-feu peut atténuer ce problème, mais cela ouvre d'énormes trous sur la connexion VPN et le pare-feu aura de la difficulté à stopper les attaques. La meilleure solution serait alors de placer le NIDS entre le pare-feu et le réseau à protéger.

Notons enfin qu'on n'a pas besoin de faire le choix entre un HIDS et un NIDS pour sécuriser un réseau. On devrait plutôt acquérir les deux à la fois.

2.4 Segmentation

La segmentation est une technique couramment utilisée pour atténuer la congestion d'un réseau. Cette technique consiste à diviser l'architecture du réseau en sections afin de réduire la taille des domaines de diffusion (*broadcast*) et ainsi augmenter l'efficacité du réseau. Cette technique peut aussi être utilisée pour augmenter la sécurité d'un réseau. En effet, elle permet d'implémenter des dispositifs de sécurité entre les frontières des différents segments. Ce qui permet de mieux contrôler le trafic en destination des ressources critiques.

La segmentation peut s'effectuer de plusieurs façons :[3]

1. **Segmentation par séparation physique :** La séparation physique des sous-réseaux est probablement la méthode de segmentation la plus sécurisée, mais elle est également la plus coûteuse en termes de cartes réseau, d'infrastructures de commutations additionnelles et intensifie l'administration.

2. **Segmentation avec des VLANs :** Un VLAN (*Virtual Local Area Network* ou Réseau Local Virtuel) est un réseau local regroupant un ensemble de machines de façon logique et non physique. Il s'agit d'un dispositif de la couche 2 (liaison de données) qui fait ce que les VPNs font au niveau de la couche 3 (réseau). Le trafic au sein d'un VLAN ne peut traverser un autre sans passer par un routeur. Ce qui permet de segmenter un réseau sans infrastructures additionnelles. Cependant, la configuration des commutateurs reliant les différents segments est d'une importance capitale car la sécurité d'un VLAN dépend en partie de l'assignation des ports de ces commutateurs.

3. **Segmentation utilisant un DMZ :** Les administrateurs réseau se considèrent en guèrre contre les attaquants et les utilisateurs même de leurs systèmes. Il n'est pas surprenant qu'ils empruntent alors des termes militaires comme DMZ (*demilitarized zone*). Un DMZ désigne une zone tampon d'un réseau, située entre ce dernier et l'Internet (l'extérieur en général). Il s'agit d'un réseau intermédiaire protégé aussi bien contre l'extérieur que le réseau interne. Le but est de pouvoir y regrouper les ressources du réseau offrant des services accessibles de l'interne comme à l'externe afin d'éviter toute connexion directe au réseau. Ces ressources sont souvent des serveurs

publics tels que serveur HTTP, serveur DHCP, etc.
Tel que décrit ci-haut, un DMZ divise le réseau en deux. Ce principe est utilisé pour segmenter un réseau en mettant en place un ou plusieurs DMZ.

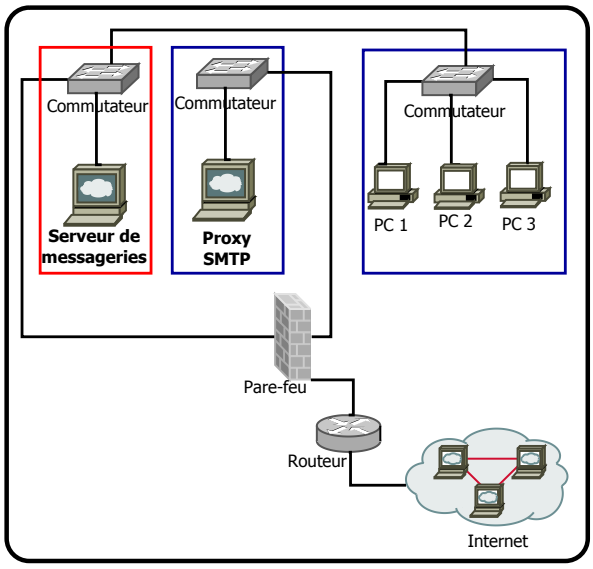


Figure 2.5 – Exemple de réseau segmenté avec un DMZ

La Figure 2.5 montre un exemple de réseau segmenté avec un DMZ. Toutes les connexions passent par cette zone. Le pare-feu envoie tous les courriels entrants au proxy du serveur SMTP qui se fait passer pour le serveur réel de messageries pour des fins de filtrage ou d'autres fonctions. Après l'inspection, le proxy réenvoie le message au pare-feu pour l'acheminer vers le serveur SMTP. Si le proxy du serveur SMTP n'est plus fonctionnel, le réseau interne reste sous la protection du pare-feu.

4. **Segmentation en fonction des services :** Une autre technique de segmentation est de considérer les services fournis par les différentes ressources et segmenter le réseau en conséquence. Chaque segment est alors définie selon le service fourni par ses ressources. Cette approche permet un contrôle très rigoureux entre les différents segments du réseau, mais elle peut mener à un grand nombre de segments dépendamment des services disponibles. Ce qui nécessite également des dispositifs de sécurités additionnels.

Conclusion

Dans ce chapitre, nous avons très rapidement évoqué les enjeux de la sécurité des réseaux afin de mettre en évidence la nécessité d'élaborer des politiques de sécurité complètes et cohérentes. La mise en place de telles politiques nécessite de penser à prévoir une architecture sécurisée dès la conception avec des redondances aux points stratégiques. Nous avons ainsi exploré plusieurs techniques de conception sécuritaire d'un réseau telles que les pare-feu, les systèmes de détection et de prévention d'intrusions et la segmentation. Le pare-feu est l'outil le plus utilisé pour sécuriser un réseau grâce à ses capacités de renforcer une politique de sécurité et d'enregistrer tous les aspects du trafic réseau. Il existe plusieurs types de pare-feu offrant des fonctionnalités variées et pouvant être placés à l'intérieur comme à la périphérique d'un réseau.
Bien qu'étant un puissant outil pour sécuriser un réseau, un pare-feu tout seul ne peut fournir la protection complète souhaitée. La sécurité d'un réseau ne doit pas se reposer sur un seul mécanisme. Une imbrication de mécanismes offre une garantie de sécurité bien supérieur, d'où la nécessité d'inclure des systèmes de détection et de prévention d'intrusion (IDS/IPS) afin de s'assurer d'une meilleur aptitude à détecter toute éventuelle intrusion. Ces IDS/IPS peuvent fonctionner selon une approche basée sur la connaissance ou une approche comportementale. On distingue également les HIDS des NIDS. Les premiers surveillent des postes particuliers tandis que les seconds contrôlent tout le trafic d'un réseau. Le déploiement des IDS/IPS devient de plus en plus vaste grâce à leur capacité d'examiner les trafics réseau.
Enfin, pour éviter de concentrer la sécurité en un seul point et optimiser les performances du réseau en réduisant la taille des zones de diffusion (*broadcast*) et augmenter son efficacité, un réseau peut être segmenté en plusieurs sections en effectuant une séparation physique ou logique de ses composantes. Ainsi, on obtient un équilibre entre la sécurité, la vivacité, le coût et la commodité du réseau.
Dans le prochain chapitre, nous allons voir comment spécifier formellement un réseau afin de mieux le sécuriser.

Chapitre 3

Spécification formelle d'un réseau

Introduction

Avec la complexité croissante des réseaux informatiques, il est nécessaire voire même obligatoire de se doter de méthodes non ambigües pour les décrire, les analyser et optimiser leurs comportements en fonction des ressources disponibles. Une spécification incohérente peut être désastreuse.
Les deux approches les plus utilisées pour spécifier un réseau sont les graphes et les algèbres de processus. Un graphe est une structure mathématique simple constitué d'un ensemble de sommets et d'arcs permettant de modéliser un système complexe avec une abstraction de plusieurs détails. Une algèbre de processus est une technique permettant de spécifier un système de manière non ambigüe et de raisonner par le biais de preuves ou de manipulations mathématiques.

Dans ce chapitre, nous allons présenter ces deux approches ainsi que leurs différentes propriétés avant de les comparer afin d'opter pour celle qui répond le plus à nos objectifs.

3.1 Graphes

3.1.1 Théorie des graphes

Les graphes mettre en pratique l'idée qu'un bon dessin vaut mieux qu'un bon discours. Cette théorie est née de préoccupations qui n'étaient pas directement mathématiques

telles que le problème des sept ponts de Königsberg. Les nombreux travaux du mathématicien L. Euler et la naissance de la recherche opérationnelle ont donné un tournant décisif à la théorie des graphes qui est devenue une branche des mathématiques discrète. Un graphe est un ensemble constitué de noeuds et d'arcs. Il permet de représenter la structure ainsi que les connexions d'un ensemble complexe en exprimant les relations entre ses éléments : réseaux de communication, réseaux routiers, circuits électriques, etc. Dans [5], R. Diestel présente un ouvrage complet sur la théorie des graphes.

3.1.2 Définitions et propriétés

Cette section présente quelques définitions et propriétés des graphes.

Définition 3.1.1 (Graphe)**.**
Un graphe G est défini par un couple (N, A) avec :
- *N : un ensemble fini d'objets, appelés noeuds ou sommets du graphe ;*
- *$A \subseteq N \times N$: un ensemble de couples de sommets appelés arcs.*

Définition 3.1.2 (Noeuds adjacents)**.**
deux noeuds u et v d'un graphe $G = (N, A)$ sont dits adjacents lorsque $(i, j) \in A$.
- *i est appelé l'origine de l'arc ou prédécesseur de j ;*
- *j est appelé l'extrémité de l'arc ou successeur de i.*

Définition 3.1.3 (Chemin)**.**
Étant donné un graphe $G = (N, A)$ orienté ou non, $i \in N$ et $j \in N$,
Un chemin de i à j, noté $\pi(i, j)$, de longueur ℓ est une séquence ordonnée de noeuds $\langle n_1, \cdots, n_\ell \rangle$, tels que $\forall\ a < \ell\ (n_a, n_{a+1}) \in A$ avec $n_1 = i$ et $n_\ell = j$.

Définition 3.1.4 (Cycle)**.**
Un cycle est un chemin $(n_1, \cdots, n_n)$ tel que $n_1 = n_n$.

Pour mieux clarifier les principaux termes employés dans la suite de ce chapitre et le chapitre suivant, nous rappelons la définition d'une relation ainsi que certaines de ses propriétés.

Définition 3.1.5 (Produit et Relation)**.**
Soient $A = \{a_1, a_2, ..., a_n\}$ et $B = \{b_1, b_2, ..., b_n\}$ deux ensembles finis d'éléments,
- *Le produit cartésien de $A \times B$ est l'ensemble des paires ordonnées (a_i, b_j) où $a_i \in A$ et $b_j \in B$*
- *Une relation $\mathcal{R}$ sur un ensemble $E = A \times B$ est un sous-ensemble de $A \times B$. On note $a\mathcal{R}b$ si $(a, b) \in \mathcal{R}$.*

Une relation est caractérisée par :

1. Un ensemble de départ (A)
2. Un ensemble d'arrivée (B)
3. Un ensemble de couples vérifiant la relation (le sous-ensemble du produit cartésien).

Définition 3.1.6 (Domaine et Codomaine)**.**
Le domaine *d'une relation $\mathcal{R}$, noté $dom(\mathcal{R})$, est l'ensemble des éléments de l'ensemble de départ qui apparaissent comme première composante d'au moins une paire de la relation. Formellement,*

$$dom(\mathcal{R}) = \{a \mid \exists\ b : (a, b) \in \mathcal{R}\}$$

Le codomaine *d'une relation $\mathcal{R}$, noté $codom(\mathcal{R})$, est l'ensemble des éléments de l'ensemble d'arrivée qui apparaissent comme deuxième composante d'au moins une paire de la relation. Formellement,*

$$codom(\mathcal{R}) = \{b \mid \exists\ a : (a, b) \in \mathcal{R}\}$$

Une relation peut avoir plusieurs propriétés dont la réflexivité, l'anti-réflexivité, la symétrie et l'anti-symétrie, pour ne citer que celles-là.

Propriété 3.1 (Réflexivité)**.**
Une relation $\mathcal{R}$ sur un ensemble E est dite réflexive lorsque $\forall\, a \in E$, on a $a\mathcal{R}a$.

Propriété 3.2 (Anti-réflexivité)**.**
Une relation $\mathcal{R}$ sur un ensemble E est dite anti-réflexive lorsque $\forall\, a \in E$, $(a, a) \notin \mathcal{R}$.

Propriété 3.3 (Symétrie)**.**
Une relation $\mathcal{R}$ sur un ensemble E est dite symétrique lorsque
$$\forall\, a \in E \text{ et } \forall\, b \in E \text{ , si } a\mathcal{R}b \text{ alors } b\mathcal{R}a.$$

Propriété 3.4 (Anti-symétrie)**.**
Une relation $\mathcal{R}$ sur un ensemble E est dite anti-symétrique
$$\forall\, a \in E \text{ et } \forall\, b \in E \text{ , si } a\mathcal{R}b \text{ et } b\mathcal{R}a \text{ alors } a = b.$$

3.1.3 Types de graphes

Définition 3.1.7 (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*.*

Définition 3.1.8 (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.*

Définition 3.1.9 (Graphe étiqueté [6])**.**
Un graphe étiqueté est un graphe orienté auquel on a associé à chacun des sommets et chacun des arcs un ensemble non vide d'étiquettes.Formellement,
étant donné L_N un ensemble fini d'étiquettes de sommets et L_A un ensemble fini d'étiquettes d'arcs, un graphe étiqueté est défini par un tuple $G = (N, A, \alpha, \beta)$ avec :

- *(N, A) un graphe ;*
- *$\alpha : N \to L_N$ une application attribuant une étiquette à chaque sommet du graphe ;*
- *$\beta : A \to L_A$ une application attribuant une étiquette à chaque arc du graphe.*

Remarque : Tout graphe non-étiqueté peut être représenté par un graphe étiqueté. Il suffit d'associer la même étiquette de sommets à tous les sommets et la même étiquette d'arcs à tous les arcs.

Définition 3.1.10 (Graphe multi-étiqueté [7])**.**
Étant donné L_N un ensemble fini d'étiquettes de sommets et L_A un ensemble fini d'étiquettes d'arcs, un graphe multi-étiqueté est défini par un tuple $G = (N, r_N, r_A)$ où :

- *N est un ensemble de sommets ;*
- *$r_N \subseteq N \times L_N$ est une relation associant les sommets à leurs étiquettes,* i.e.*, r_N est l'ensemble de couples (υ, ℓ) tels que le sommet υ a pour étiquette ℓ ;*
- *$r_A \subseteq N \times N \times L_N$ est une relation associant les arcs à leurs étiquettes,* i.e.*, r_A est l'ensemble de triplets $(\upsilon, \upsilon', \ell)$ tels que l'arc (υ, υ') a pour étiquette ℓ. Si l'on suppose que chaque arc a au moins une étiquette, l'ensemble A des arcs est alors défini par $A = \{(\upsilon, \upsilon') | \exists \ell, (\upsilon, \upsilon', \ell) \in r_A\}$.*

3.1.4 Représentation des graphes

Les graphes sont des outils de modélisation de plusieurs problèmes. D'où la nécessité de les automatiser. Il existe trois techniques de représentation permettant d'encoder un graphe.

1. **Matrice d'adjacence** : Soit $G = (N, A)$ un graphe. La matrice d'adjacence de G est une matrice carrée $n \times n$ (où $n = |N|$) d'éléments binaires permettant d'avoir une représentation simple de G. Formellement, elle est définie par :

$$M_{i,j} = \begin{cases} 1 & Si\ (i,j) \in A \\ 0 & Sinon \end{cases}$$

Exemple 3.1.1. *Soit G le graphe de la Figure 3.1.*

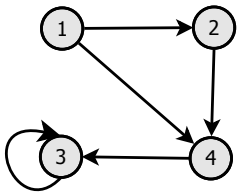


Figure 3.1 – Un exemple de graphe orienté

Sa matrice d'adjacence est :

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Les principaux avantages d'une telle représentation sont : la facilité de sa construction, la compacité de la représentation, la rapidité des recherches, la simplicité des algorithmes de calcul, etc.
Cette représentation à également des inconvénients : la redondance des informations pour les graphes non orientés. On stocke inutilement des zéros qui seront examinés quand la représentation ne convenant qu'aux graphes simples. Elle occupe n^2 cases alors que le nombre d'arcs peut être moindre. Ce qui constitue une occupation importante de mémoire.

2. **Matrice d'incidence** :
Un graphe de n sommets et m arcs peut être représenté par une matrice $(n \times m)$ d'entiers, appelée matrice d'incidence, définie comme suit :

$$M_{i,a} = \begin{cases} 1 & Si\ i\ est\ l'origine\ de\ l'arc\ a \\ -1 & Si\ i\ est\ l'extrémité\ de\ l'arc\ a \\ 0 & Sinon \end{cases}$$

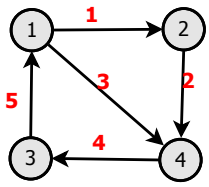


Figure 3.2 – Un exemple de graphe sans boucle

Exemple 3.1.2. *Soit G le graphe de la Figure 3.2.*
Sa matrice d'incidence est :

$$\begin{pmatrix} 1 & 0 & 1 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & -1 & -1 & 0 & 0 \end{pmatrix}$$

Avec cette représentation, les informations ne sont plus redondantes pour les graphes non orientés. Elle a également les autres avantages de la matrice d'adjacence. Cependant, il y a toujours un stockage et un examen inutile des zéros. Les calculs des matrices classiques ne s'appliquent pas non plus. Enfin, cette matrice n'est pas adaptée aux graphes contenant des boucles.

3. **Liste d'adjacence :**
Une liste d'adjacence d'un graphe de n sommets est un tableau de listes chaînées. Chaque sommet du tableau contient une liste chaînée des sommets qui lui sont adjacents.

Exemple 3.1.3. *Si on reprend le graphe de la Figure 3.1, sa liste d'adjacence est :*

1	*:*	*2 ;*	*4.*
2	*:*	*4.*	
3	*:*	*3.*	
4	*:*	*3.*	

L'avantage principal d'une liste d'adjacence est qu'elle ne représente que les arcs réellement présent dans le graphe. Cependant, elle est plus compliquée à mettre en oeuvre et le temps de recherche peut être long.

3.2 Algèbres de processus

3.2.1 Notions d'algèbre de processus

L'algèbre est la branche des mathématiques qui étudie les structures algébriques, indépendamment de la notion de limite (rattachée à l'analyse) et de la notion de représentation graphique (rattachée à la géométrie)[*Wikiversité*].
Une algèbre est constituée d'un domaine et d'un ensemble d'opérations qui traitent des données comme en arithmétique, mais sur un plan plus général en utilisant la théorie des ensembles.
Un processus fait référence au comportement d'un système, i.e. l'ensemble des événements ou actions qu'il peut réaliser, l'ordre dans lequel ils peuvent être exécutés et d'autres aspects de cette exécution telles que la durée ou les probabilités [8]. Une algèbre de processus exprime le fait d'utiliser une approche algébrique ou axiomatique pour modéliser le comportement d'un processus.
Les algèbres de processus ont été conçues dans le but de cerner les primitives de base, d'avoir une meilleure compréhension du parallélisme et d'élaborer des résultats théoriques dès la conception même d'un système informatique. Ces algèbres sont définies par un ensemble d'éléments de base ainsi qu'un ensemble d'opérateurs permettant de construire des composantes complexes à partir d'éléments basiques.

3.2.2 Opérateurs de base

La plupart des algèbres de processus ont en commun un ensemble d'opérateurs de bases dont les syntaxes peuvent varier d'une algèbre à une autre. Parmi ces opérateurs de l'algèbre classique, on retrouve :

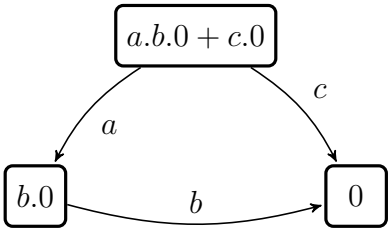
- **Le préfixage ($\cdot$) :** Une composante peut avoir un comportement purement séquentiel, entreprenant à plusieurs reprises une action à la suite d'une autre, retournant éventuellement par la suite au début de son comportement. Dans ce cas, l'opérateur de préfixage "$\cdot$" est utilisé pour indiquer l'ordre d'exécution des actions. Par exemple, $\alpha \cdot P$ effectuera l'action α puis se comportera comme la composante P.
- **Le choix ($+$) :** Un choix entre plusieurs composantes possibles est représenté par la somme des différentes possibilités. Par exemple $P + Q$ désigne un processus qui peut se comporter soit comme P soit comme Q.
- **Le parallélisme ($|$) :** La composition parallèle est utilisée pour représenter les cas où deux composantes d'un système coopèrent pour réaliser une certaine action. Par

exemple, $P|Q$ se compose de deux processus P et Q qui doivent exister simultanément.

- **La restriction ($P\backslash L$) :** Il est souvent commode de cacher quelques actions, les rendant privées aux composantes impliquées et invisibles de l'extérieur. Cela est représenté par le processus $P\backslash L$ qui se comporte comme P à l'exception qu'il ne peut exécuter les actions qui sont dans l'ensemble L.
- **Le renommage ($P[f]$) :** Cet opérateur permet de renommer les actions d'un processus pour en définir un nouveau. Par exemple, le processus $\alpha \cdot \beta \cdot P[\tau/\alpha]$ a le même comportement que le processus $\tau \cdot \beta \cdot P$.

Selon les écoles de pensées, la sémantique peut également différer d'une algèbre à une autre. On peut distinguer trois styles de sémantiques [8] :

- **Sémantique opérationnelle :** Dans une sémantique opérationnelle, un programme est modélisé comme l'exécution d'une machine abstraite en utilisant un système de transition. Un état d'une telle machine est une évaluation de variables, chaque transition entre les états est une instruction élémentaire du programme.
  McCarthy [9] est l'un des pionnier de ce type de formalisme.
  Exemple 3.2.1. *Le processus $a.b.0 + c.0$ peut être représenté par :*



  On peut par la suite, définir une règle opérationnelle pour chaque forme syntaxique en se référant aux transitions possibles des composantes du processus.

- **Sémantique dénotationnelle :** Plus abstraite que la sémantique opérationnelle, dans une sémantique dénotationnelle les programmes sont généralement modélisés par une fonction transformant chaque entrée en une sortie. Cette technique a été introduite par Scott et Strachey [10].
  Exemple 3.2.2. *Si nous considérons la syntaxe suivante pour les nombres binaires.*

$$\nu ::= 0 \mid 1 \mid \nu 0 \mid \nu 1$$

  Qui signifie qu'un binaire est soit l'un des chiffres 0 ou 1 soit le résultat de la suffixation d'un de ces chiffres à un binaire préalablement obtenu.

Pour déterminer la sémantique d'une expression, on définit la fonction suivante :

$$\mathcal{O} : Bin \to \mathbb{N}$$

Où Bin désigne l'ensemble des nombres binaires.
Ainsi, pour chaque $\nu \in Bin$, la fonction $\mathcal{O}[\![\nu]\!]$ retourne sa valeur entière. D'où la sémantique suivante :

$$\begin{aligned} \mathcal{O}[\![0]\!] &= 0 \\ \mathcal{O}[\![1]\!] &= 1 \\ \mathcal{O}[\![\nu 0]\!] &= 2 \cdot \mathcal{O}[\![\nu]\!] \\ \mathcal{O}[\![\nu 1]\!] &= 2 \cdot \mathcal{O}[\![\nu]\!] + 1 \end{aligned}$$

Par exemple : $\mathcal{O}[\![1001]\!] = 2 \cdot \mathcal{O}[\![001]\!] + 1 = 2 \cdot 2 \cdot \mathcal{O}[\![01]\!] + 1 = 2 \cdot 2 \cdot 2 \cdot \mathcal{O}[\![1]\!] + 1 = 9$

- **Sémantique axiomatique :** Dans ce type de sémantique, l'accent est mis sur les méthodes de preuve de la correction des programmes. Les notions centrales sont les assertions, les préconditions, les postconditions et les invariants. Les pionniers de cette approche sont Floyd [11] et Hoare [12].
  Exemple 3.2.3. *Considérons l'instruction d'affectation suivante :*

  $$x := f$$

  Avec
  – *x une variable.*
  – *f une expression d'un langage de programmation contenant x.*

  Si nous utilisons la notation

  $$P \ \{Q\} \ R$$

  pour formaliser l'expression « Si la précondition P est vraie avant l'initiation du programme Q, alors la postcondition R le sera à son achèvement »
  Nous pouvons définir l'axiome de l'affectation comme suit :

  $$\vdash \ P_0 \ \{x := f\} \ P$$

  où P est obtenue à partir de P_0 en substituant toutes les occurrences de x par f.

  Ainsi, si $P(f)$ doit être vraie après l'affectation, alors $P(x)$ doit l'être avant.

3.2.3 Quelques algèbres de processus

Dans le domaine de la spécification et la vérification des systèmes concurrents et des réseaux informatiques, plusieurs algèbres dont les suivantes ont été développées :

- CCS [13] (*Calculus Of Communicating Systems*) a été introduite par Robin Milner, un des personnages centrales dans l'histoire de l'algèbre des processus. CCS est souvent utilisée pour évaluer certaines propriétés d'un système telles que les blocages (*deadlock*).
- CSP [14] (*Communicating Sequential Processes*) a été proposée par Tony Hoare pour modéliser l'interaction entre les systèmes. CSP intègre un mécanisme de synchronisation basé sur le principe du rendez-vous permettant ainsi d'implémentation des paradigmes classiques de la programmation concurrente.
- ACP [15, 16] (*Algebra of Communicating Processes*) est une dérivée de CCS développée par Bergstra et Klop dans [15] pour établir un cadre (*framework*) pour la coopération équationnelle des processus asynchrones via une communication synchrone.
- PEPA [17, 18] (*Performance Evaluation Process Algebra*) est un formalisme développé par Hillston et Gilmore dans [17], qui étend l'algèbre de processus classique en associant à chaque action une variable aléatoire représentant l'aspect temporel.
- D'autres algèbres de processus telles que π-Calculs [19, 20], μ-Calcul [21], etc. sont également utilisées pour spécifier des systèmes concurrents.

Traditionnellement, les algèbres de processus se sont concentrées sur les aspects fonctionnels des systèmes telles que leurs comportements observables, le contrôle de flux et la synchronisation en tant que propriétés de temps relatif. À la fin des années 1980 l'intérêt a grandi dans l'extension des algèbres de processus pour intégrer des informations quantitatives comme le temps et la probabilité. Ainsi, on peut distinguer des algèbres de processus temporisées, probabilistes ou stochastiques [22].

1. **Algèbres de processus temporisées**
   Le principe des algèbres de processus temporisées est d'étendre l'algèbre classique avec un aspect temporelle. Ainsi, $(t).P$ indique que le processus P est atteinte après un délai d'unités de temps t. Une distinction principale entre les divers algèbres de processus temporisées porte sur l'interprétation du moment où les actions peuvent se produire. Dans certains cas, une action peut se produire après un certain délai, mais peut être soumis à un délai supplémentaire, par exemple, si elle doit se synchroniser avec son contexte.
2. **Algèbres de processus probabilistes**
   L'idée de base des algèbres de processus probabilistes est d'intégrer un opérateur de

choix probabiliste qui admet des termes tels que $P +_p Q$ (avec $p \in (0, 1)$) où P peut être sélectionné avec une probabilité p et Q avec une probabilité $1 - p$. Différents modèles de sémantiques peuvent être utilisés selon les hypothèses.

3. **Algèbres de processus stochastiques**
   La principale motivation derrière le développement des algèbres de processus stochastiques est de décrire avec plus de précision le comportement d'un système en combinant les aspects temporels et probabilistes. Ainsi, ces algèbres attachent une variable aléatoire à l'information temporelle afin d'évaluer des propriétés telles que la probabilité d'un délai d'attente, la durée des séquences d'un événement, etc.

Une description plus complète de ces différentes algèbres est présentée dans [22].

3.3 Synthèse

Les graphes constituent une méthode de pensée qui permet de modéliser une grande variété de problèmes en se ramenant à l'étude de sommets et d'arcs. Ainsi, ils capturent principalement la notion de relations binaires. Un réseau est un système de communication dans lequel des hôtes communiquent entre eux soit directement, soit par l'intermédiaire d'autres hôtes, en envoyant des messages qui circulent le long de lignes de communication. De tels systèmes se représentent facilement par des graphes, les hôtes et les lignes correspondant trivialement aux sommets et aux arcs. Cette modélisation est particulièrement fructueuse car elle permet de s'abstraire de beaucoup de détails et se restreindre sur un ensemble théorique très riche. Elle permet notamment de résoudre le problème de localisation. Cependant, une des difficultés pour travailler sur les graphes est de les représenter correctement sans utiliser trop de mémoire tout en ayant un temps d'accès courts.

L'utilisation des algèbres de processus pour la spécification et l'analyse des systèmes informatiques est motivée par les nombreux avantages qu'offre une telle approche. Parmi ces avantages, on peut citer :

– Le système est représenté comme un ensemble d'agents actifs qui coopèrent pour réaliser son comportement. Ce paradigme collaborateur est particulièrement approprié pour la modélisation de nombreux systèmes informatiques modernes. Un réseau n'échappe pas à cette règle.
– Le raisonnement de composition est une partie intégrante du langage de modélisation.
– La définition formelle clarifie la tâche de fournir des outils pour la manipulation, la simplification et l'analyse du modèle.

– L'algèbre de processus a une importance croissante en tant que méthodologie de conception car elle offre la possibilité d'intégrer l'analyse des performances dans le processus de conception des systèmes.
– La garantie mathématique de certaines propriétés telles que la cohérence, la complétude et la correction de la spécification.

Les algèbres de processus ou les méthodes formelles en général constituent donc de puissants outils permettant d'obtenir un niveau élevé de sécurité en développant des systèmes certifiés et sans erreurs.

Bien qu'elles garantissent autant de propriétés, elles ne sont pas suffisantes. En effet, certaines propriétés restent difficiles à exprimer avec les algèbres de processus. La confidentialité, par exemple, est souvent décrite à travers un moniteur de contrôle d'accès filtrant des requêtes, mais cela ne constitue qu'une vision partielle de la propriété recherchée. Il faut également améliorer les expressions afin de faciliter leur compréhension et leur utilisation.
Enfin, les méthodes formelles restent peu utilisées dans l'industrie en raison de leur réputation d'être complexes et couteuses à mettre en oeuvre. Leur exploitation est souvent limitée au processus de certification. Elles sont donc perçues comme un moyen de convaincre les autorités de certification et les évaluateurs et non comme un outils pour améliorer la spécification, l'analyse et la robustesse des systèmes.

Conclusion

Dans ce chapitre nous avons présenté un bref aperçu de la théorie des graphes ainsi que les avantages d'utilisation des graphes pour modéliser des systèmes complexes comme les réseaux informatiques. Nous avons ensuite présenté une famille de langages formels appelés algèbres de processus, permettant de modéliser les systèmes informatiques concurrents ou distribués. Enfin, nous avons effectué une étude comparative de ces deux approches pour cerner les avantages et limites de chacune d'elles. Les graphes offrent une bonne abstraction de certains détails d'un système et les méthodes formelles permettent d'accompagner un résultat de preuves mathématiques irréfutables. Ainsi, pour notre approche, nous allons définir un graphe étiqueté pour modéliser les réseaux et une logique propositionnelle pour spécifier une politique de sécurité afin de produire une représentation non ambigüe. Mais avant cela, dans le prochain chapitre, nous allons étudier l'état de l'art sur le déploiement des pare-feu pour renforcer une politique de sécurité.

Chapitre 4

Renforcement de politique de sécurité

Introduction

Les réseaux informatiques ont révolutionné le monde de l'entreprise et sont devenus quasiment indispensables au bon fonctionnement de toute institution. En effet, ils facilitent grandement leurs activités en fournissant de nouvelles façons de communiquer et de travailler. En retour, plusieurs risques liés à la sécurité ont augmenté. Chaque nouvelle pratique vient avec de nouvelles menaces d'où l'importance d'acquérir des outils pour protéger les systèmes informatiques. De nombreux travaux ont été réalisés dans ce sens et plusieurs outils et techniques ont été développés afin de forcer un réseau à se conformer à une politique donnée. Parmi ces outils, les pare-feu occupent une place de premier ordre et constituent une des pierres angulaires de la sécurité des réseaux informatiques.

Un pare-feu est un dispositif matériel ou logiciel qui contrôle le trafic entre les différentes entités d'un réseau afin de n'autoriser que les paquets respectant la politique de sécurité en vigueur. Ces pare-feu sont configurés en utilisant des listes de contrôles d'accès (ACL) constituées d'ensembles de règles définissant les actions à entreprendre selon des conditions spécifiques de filtrage. Une règle de filtrage est composée de plusieurs champs tels que les adresses IP source et destination, le protocole (UDP, TCP, etc.), les ports, ainsi qu'une action (*accepter*, *rejeter*, etc.). Par exemple, la règle suivante qu'on peut trouver dans un pare-feu Cisco.

```
access_list 101 permit TCP 20.9.17.8  121.11.127.20 range 23..27
```

Cette règle accepte tous les paquets provenant de la machine 20.9.17.8, ayant le serveur 121.11.127.20 comme destination et utilisant le protocole TCP et des ports source et destination entre 23 et 27.
Cependant, avoir des pare-feu sur les frontières d'un réseau est loin d'être suffisant pour le sécuriser. En effet, la configuration des pare-feu est une tâche subtile et sujette de nombreuses anomalies pouvant mener à une situation précaire. Ces anomalies peuvent varier d'une simple redondance entre deux règles à une violation totale de la politique de sécurité. La redondance affecte la performance du système tandis que la violation accroît ses vulnérabilités.
Dans ce chapitre, nous allons explorer les différents types d'anomalies qui entravent une bonne configuration des pare-feu ainsi que de nombreuses solutions pour les remédier.

Le reste du chapitre est organisé comme suit : Dans la première partie nous présentons les différents types d'anomalies rencontrées dans la configuration des pare-feu. Par la suite nous explorons quelques techniques de détection de ces anomalies. Enfin, nous passons en revue quelques solutions proposées pour résoudre ces problèmes avant d'en faire une brève synthèse comparative.

4.1 Classification des anomalies des pare-feu

Après l'acquisition d'un pare-feu, il est question de le configurer et de l'administrer selon une politique donnée. Une telle tâche est complexe et source d'erreurs pouvant atteindre l'intégrité même de la politique de sécurité.
En se focalisant sur les ensembles de règles des produits *Check Point FireWall-1*, A. Wool[23] a étudié 37 pare-feu provenant d'horizons différents (organisations de télécommunication, compagnies financières, laboratoires de recherche, institutions académiques, etc.) et a montré que ces derniers ont tous des vulnérabilités dues à de mauvaises configurations. Il a notamment identifié 12 erreurs de configuration fréquentes qui pourraient permettre un accès au-delà de la politique de sécurité en vigueur. Parmi ces erreurs, on peut citer l'absence d'une règle de furtivité pour cacher le pare-feu afin de le protéger lui-même contre tout accès non autorisé. Le fait d'autoriser le service NetBIOS est aussi une erreur pouvant exposer un réseau à des services très précaires.
Ainsi, environ 46% des pare-feu étudiés dans[23] ne sont pas configurés avec une règle de furtivité pour les cacher, plus de 70% sont ouverts à des protocoles de gestion insécurisés ou gérés par des machines externes.
Les travaux de Al-Shaer *et al.* dans [24] et [25] fournissent une description des différents types d'anomalies découvertes sur la configuration des pare-feu. Nous décrivons ces anomalies, classées [26] en fonction des failles de sécurité qu'elles peuvent constituer.

4.1.1 Violation de politique

Les administrateurs ont souvent une politique de haut niveau décrivant ce qui devrait être interdit (*liste noire*) ou autorisé (*liste blanche*) d'accéder au réseau. Il est donc crucial que les configurations de pare-feu reflètent exactement la politique de sécurité. L'autorisation d'un élément de la liste noire ou l'interdiction d'un élément de la liste blanche est appelée *violation de la politique*. Une telle violation peut entraîner un blocage non-désiré, un accès non autorisé, voire même la modification de certains paramètres de la configuration. Cette dernière doit donc être soigneusement vérifiée par rapport à la politique mise en oeuvre.

4.1.2 Incohérences

Un pare-feu ne fournit qu'une protection équivalente à la qualité de sa configuration. Les incohérences sont souvent de bons indicateurs d'erreurs de configuration. Contrairement aux violations de politiques pour lesquelles les références sont biens définies (listes noires et listes blanches), la vérification des incohérences est uniquement basée sur la configuration des fichiers et n'a pas besoin d'une entrée externe. Les incohérences les plus connues sont l'ombrage, la généralisation et la corrélation.

Définition 4.1.1 (Ombrage (*Shadowing*))**.**
Une anomalie est dite ombrage lorsqu'une règle R_2 est précédée par une autre R_1 ayant une action différente pour des conditions similaires ou plus génériques. Ce qui fait que la règle ombragée ne sera jamais exécutée.

Exemple 4.1.1. *Dans la spécification qui suit, la règle R_2 est ombragée par R_1.*

R_1 :	*accept*	*udp*	*any*	192.168.1.0/24
R_2 :	*deny*	*udp*	172.16.1.0/24	192.168.1.0/24

En effet, tous les paquets UDP de 172.16.1.0/24 à 192.168.1.0/24 sont acceptés par la règle R_1, qui correspond à tous les paquets UDP destinés à 192.168.1.0/24.

Définition 4.1.2 (Généralisation)**.**
On parle de généralisation lorsqu'une règle R_1 est un sous-ensemble d'une autre règle R_2 qui effectue une action différente. La généralisation est l'opposé de l'ombrage.

Exemple 4.1.2. *La règle R_3 ci-dessous est une généralisation de la règle R_2 car les paquets udp entre* 172.16.1.0/24 *et* 192.168.1.0/24 *forment un sous-ensemble des paquets*

udp de 172.16.1.0/24 *(règles 3), mais les deux règles n'ont pas la même décision.*

R_2 :	*deny*	*udp*	172.16.1.0/24	192.168.1.0/24
R_3 :	*accept*	*udp*	172.16.1.0/24	*any*

Définition 4.1.3 (Corrélation)**.**
Deux règles R_1 et R_2 sont dites corollaires lorsqu'elles couvrent les mêmes paquets alors que leurs actions sont différentes.

Exemple 4.1.3. *Les règles R_1 et R_4 ci-dessous sont corollaires.*

R_1 :	*accept*	*udp*	*any*	192.168.1.0/24
R_4 :	*deny*	*udp*	10.1.1.0/24	192.168.0.0/16

Les deux règles ont pour intersection : *udp* 10.1.1.0/24 192.168.0.0/24

4.1.3 Inefficacités

Une configuration est dite inefficace lorsqu'on peut supprimer une ou plusieurs règles sans changer le comportement global du système. Parmi ces inefficacités, on peut citer la redondance et la verbosité.

Définition 4.1.4 (Redondance)**.**
Une redondance se présente dans un pare-feu lorsque la suppression d'une règle ne change pas son action sur l'ensemble des paquets.
Une règle R_2 est dite redondante lorsqu'elle est précédée par une autre R_1 effectuant la même action et ayant une condition équivalente ou plus générique.

Exemple 4.1.4. *Les deux règles ci-dessous sont redondantes.*

r_1 :	*accept*	*tcp*	10.1.0.0/8	*any*
r_2 :	*accept*	*tcp*	10.2.1.0/24	*any*

Définition 4.1.5 (Verbosité)**.**
On parle de verbosité dans un pare-feu lorsque plusieurs règles peuvent se réduire en une seule.

Exemple 4.1.5. *Les quatre premières règles ci-dessous peuvent se résumer en une seule règle qui est la 5^e.*

1 :	*deny*	*udp*	10.1.0.0/26	*any*
2 :	*deny*	*udp*	10.1.1.64/26	*any*
3 :	*deny*	*udp*	10.1.1.128/26	*any*
4 :	*deny*	*udp*	10.2.1.192/26	*any*
5 :	*deny*	*udp*	10.2.1.0/24	*any*

Ces erreurs sont très courantes dans les configurations des pare-feu et peuvent affaiblir considérablement la sécurité envisagée. Par exemple, une redondance augmente le nombre de règles dans le pare-feu. Ce qui entraîne une croissance de la consommation en temps et en mémoire de l'analyse des paquets.

4.2 Techniques de détection d'anomalies

Dans cette section, nous présentons deux techniques de détection d'anomalies.

4.2.1 FIREMAN (*FIREwall Modeling and ANalysis*)

FIREMAN [26] est un outil de modélisation et d'analyse de pare-feu capable de détecter toutes les anomalies énumérées dans la section précédente. Le principe de FIREMAN consiste à faire une analyse statique permettant de déceler automatiquement les anomalies. Cet outil utilise un algorithme lui permettant de découvrir aussi bien les anomalies au sein d'un pare-feu que celles entre plusieurs pare-feu.
L'idée de base de cet algorithme est de parcourir toutes les ACLs des pare-feu et s'assurer que chaque nouvelle règle concerne des paquets non encore couverts.
L'analyse d'un pare-feu simple se fait différemment de celui d'un réseau de pare-feu :

4.2.1.1 Un seul pare-feu

Pour un seul pare-feu, les règles sont représentées par un graphe. Les variables ci-dessous sont utilisées pour la modélisation :

- A_j : Tous les paquets acceptés avant la j^{eme} règle ;
- D_j : Tous les paquets refusés avant la j^{eme} règle ;
- F_j : Tous les paquets détournés avant la j^{eme} règle ;
- R_j : Tous les paquets non-couverts avant la j^{eme} règle ;
- I : Entrée (*Input*) d'une ACL.

L'ensemble R_j peut se définir formellement par :

$$R_j = I \cap \neg(A_j \cup D_j \cup F_j) \tag{4.1}$$

Pour la première règle d'une ACL, l'état initial est : $A_1 = D_1 = F_1 = \emptyset$ et $R_1 = I$.
Par la suite, l'état du pare-feu est mis à jour après la lecture de chaque règle en utilisant la transformation 4.2.

$$\begin{cases} \langle A, D, F\rangle, & \langle P, accept\rangle \vdash \langle A \cup (R \cap P), D, F\rangle \\ \langle A, D, F\rangle & \langle P, deny\rangle \vdash \langle A, D \cup (R \cap P), F\rangle \\ \langle A, D, F\rangle & \langle P, pass\rangle \vdash \langle A, D, F \cup (R \cap \neg P)\rangle \end{cases} \tag{4.2}$$

La notation $S_i, r \vdash S_{i+1}$ signifie que si la règle r est lue à l'état S_i alors elle entraînera le système dans l'état S_{i+1}. R sera alors automatiquement mis à jour lorsque le triplet $\langle A, D, F\rangle$ change. À la fin d'un chemin π, nous pouvons déterminer les paquets acceptés et refusés, notés respectivement par A_π et D_π.
Puisque les paquets peuvent emprunter des chemins différents, l'ensemble des paquets acceptés par une ACL est l'union des paquets acceptés sur tous ses chemins. L'action par défaut d'une ACL rencontre tous les paquets, ils seront donc soit acceptés ou refusés. D'où :

$$\begin{cases} A_{ACL} = \bigcup\limits_{i \in \pi} A_{\pi_i} \\ D_{ACL} = \bigcup\limits_{i \in \pi} D_{\pi_i} \end{cases} \tag{4.3}$$

$$\begin{cases} A_{ACL} \cup D_{ACL} = I_{ACL} \\ R_{ACL} = \emptyset \end{cases} \tag{4.4}$$

En utilisant les variables préalablement déclarées, on peut définir une logique formelle ainsi qu'un ensemble de règles pour identifier les anomalies. Par exemple, pour les règles de la forme $\langle P, accept\rangle$:

- Si $P_j \subseteq R_j$, alors tout est correct ;
- Si $P_j \subseteq D_j$, alors il y a une ombrage ;
- Si $P_j \cap D_j \neq \emptyset$, alors il y a une correlation ;
- etc. (voir [26] pour plus de détails).

Et pour les règles de la forme $\langle P, deny\rangle$:

- Si $P_j \subseteq R_j$, alors tout est correct ;
- $P_j \cap R_j = \emptyset$, alors il y a une règle masquée et si $P_j \subseteq A_j$ c'est une ombrage.
- etc. (voir [26] pour plus de détails).

4.2.1.2 Un réseau de pare-feu

Dans les grands réseaux, plusieurs pare-feu sont souvent déployés pour mettre en oeuvre une politique de sécurité. Un paquet peut traverser une série d'ACLs pour atteindre sa destination. Dans ce cas, il a besoin de survivre aux filtrages de tous les ACLs sur son

chemin. Cependant, le nombre de chemins pouvant être multiple, un arbre est utilisé pour représenter la disposition des pare-feu (série, parallèle, etc.).
Pour un ensemble de n pare-feu en série (respectivement en parallèle), les paquets doivent survivre à tous les filtrages des ACLs sur leurs chemins. Par conséquent, l'ensemble des paquets acceptés est l'intersection (respectivement l'union) de ceux acceptés par chacun des pare-feu. D'où les expressions du Tableau 4.1 :

ACLs en série	ACLs en parallèle
$A = \bigcap_{acl \in n} A_{acl}$	$A = \bigcup_{acl \in n} A_{acl}$
$D = \bigcup_{acl \in n} D_{acl}$	$D = \bigcap_{acl \in n} D_{acl}$

Tableau 4.1 – *Équations pour ACL série et parallèle*

Exemple 4.2.1. *Soit l'arbre de la Figure 4.1 où W, X, X', Y et Z sont les pare-feu d'un réseau.*

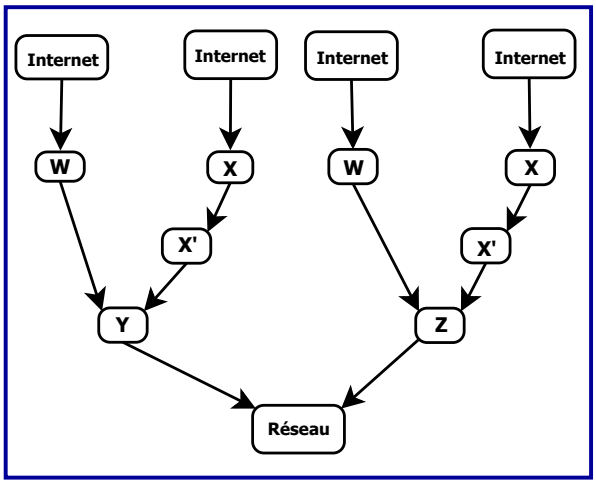


Figure 4.1 – Arbre d'ACLs

On peut montrer que I_Y (l'entrée de Y) est $I_Y = A_W \cup (A_X \cap A_{X'})$.
De même, l'ensemble des paquets qui peuvent atteindre le réseau interne depuis l'Internet est :

$$\begin{aligned} A = &\ A_W \cup (A_X \cap A_{X'}) \cap A_Y \cup \ A_W \cup (A_X \cap A_{X'}) \cap A_Z \\ = &\ A_W \cup (A_X \cap A_{X'}) \cap A_Y \cup A_Z \end{aligned}$$

Puisqu'il pourrait y avoir des arbres logiques distincts pour chaque pare-feu, il est donc primordial de s'assurer que les configurations sont compatibles à travers l'élaboration des politiques du réseau.
Pour un réseau de m pare-feu, l'entrée du $j^{ième}$ pare-feu doit correspondre à toutes les autres entrées. C'est-à-dire $\forall j \in m, I_j = I$. Ainsi, une violation de la politique sera détectée par les deux règles suivantes :

- $I \cap Liste\_noire \neq \emptyset \Rightarrow$ *violation de la politique* ;
- $Liste\_blanche \not\subseteq I \Rightarrow$ *violation de la politique* ;

où *Liste_noire* et *Liste_blanche* sont respectivement les listes des actions interdites et autorisées.

Pour la vérification des anomalies, on utilise les équations du Tableau 4.1. À partir de l'entrée I, on parcourt les règles des ACLs selon la transformation 4.2. Ainsi, pour les règles de la forme : $\langle P, accept \rangle$:

- Si $P \subseteq I$ alors tout est correct, puisqu'un paquet doit être accepté par tous les pare-feu sur son chemin pour atteindre sa destination.
- Si $P \subseteq \neg I$ alors il y a une ombrage. En effet, cette règle essaie d'accepter des paquets qui sont bloqués sur tous les chemins accessibles. Ce genre d'incohérence peut se manifester par des problèmes de connectivité qui sont difficiles à résoudre manuellement.

Quant aux règles de la forme : $\langle P, deny \rangle$:

- Si $P \subseteq I$; alors tout est correct. Cela révèle un niveau élevé de sécurité.
- Si $P \subseteq \neg I$. C'est probablement une redondance puisque les paquets refusés n'atteindront pas cette ACL.

4.2.2 Diagramme d'états

Un diagramme d'états est une autre technique simple et efficace proposée par Ehab *et al.* dans [27] pour découvrir toute sorte d'anomalie qui peut survenir entre deux ou plusieurs règles d'un pare-feu.

Étant donnée deux règles R_x et R_y d'un pare-feu tels que R_x précède R_y dans la disposition des règles de l'ACL, le principe consiste à comparer chaque domaine de R_y au domaine correspondant de R_x en commençant par le protocole, suivi des adresses et ports sources et destinations. On se sert ensuite des trois règles suivantes pour déterminer les anomalies entre R_x et R_y :

Règle 1 : Si tous les champs de R_y sont inclus ou égaux aux champs correspondants de R_x, alors :

– si les deux règles ont la même action donc R_y est redondante à R_x ;
– sinon R_y est ombragée par R_x.

Règle 2 : Si tous les champs de R_x sont inclus ou égaux aux champs correspondants de R_y, alors :
– si R_x a la même action que R_y, elle l'est éventuellement redondante ;
– sinon R_y est une généralisation de R_x.

Règle 3 : Si certains champs de R_x sont inclus ou égaux aux champs correspondants de R_y et que d'autres champs de R_x engendrent leurs correspondants dans R_y, alors si les deux règles ont des actions différentes donc elles sont en correlation.

La Figure 4.2 [1] illustre l'utilisation de ces règles pour la découverte d'anomalies de configuration dans un pare-feu. Ainsi, pour une ACL donnée, on représente la politique de

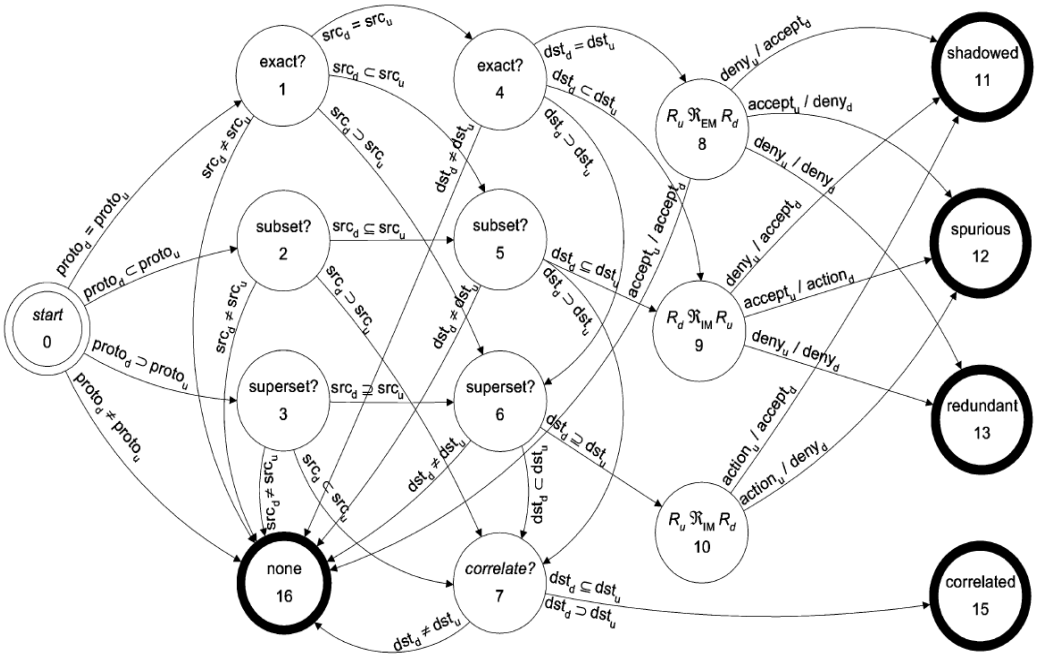


Figure 4.2 – *Détermination des anomalies avec un diagramme d'états*

sécurité par un arbre appelé "arbre de politique". Chaque noeud de cet arbre représente un champ d'une règle et chaque branche partant d'un noeud désigne une valeur possible du champ associé. Tout chemin allant de la racine à une feuille représente une règle de la politique. À partir d'un noeud spécifique, les règles de même champ se retrouvent sur la même branche. Ainsi, l'arbre de politique fournit une simple représentation des règles de filtrage et facilite la découverte d'éventuelles anomalies.

1. Source de la figure : [27]

Exemple 4.2.2.
Soit le pare-feu de la Figure 4.3[2] *ainsi que la politique qu'il met en oeuvre.*

		Source		Destination		
	Protocol	Address	Port	Address	Port	Action
1:	tcp,	140.192.37.20,	any,	*.*.*.*,	80,	deny
2:	tcp,	140.192.37.*,	any,	*.*.*.*,	80,	accept
3:	tcp,	*.*.*.*,	any,	161.120.33.40,	80,	accept
4:	tcp,	140.192.37.*,	any,	161.120.33.40,	80,	deny
5:	tcp,	140.192.37.30,	any,	*.*.*.*,	21,	deny
6:	tcp,	140.192.37.*,	any,	*.*.*.*,	21,	accept
7:	tcp,	140.192.37.*,	any,	161.120.33.40,	21,	accept
8:	tcp,	*.*.*.*,	any,	*.*.*.*,	any,	deny
9:	udp,	140.192.37.*,	any,	161.120.33.40,	53,	accept
10:	udp,	*.*.*.*,	any,	161.120.33.40,	53,	accept
11:	udp,	140.192.38.*,	any,	161.120.35.*,	any,	accept
12:	udp,	*.*.*.*,	any,	*.*.*.*,	any,	deny

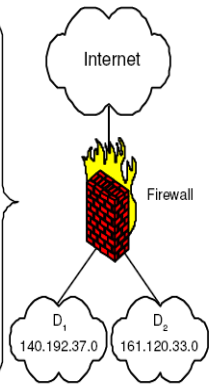


Figure 4.3 – *Exemple de pare-feu et sa politique de sécurité*

L'arbre de cette politique est représenté par la Figure 4.4[3].

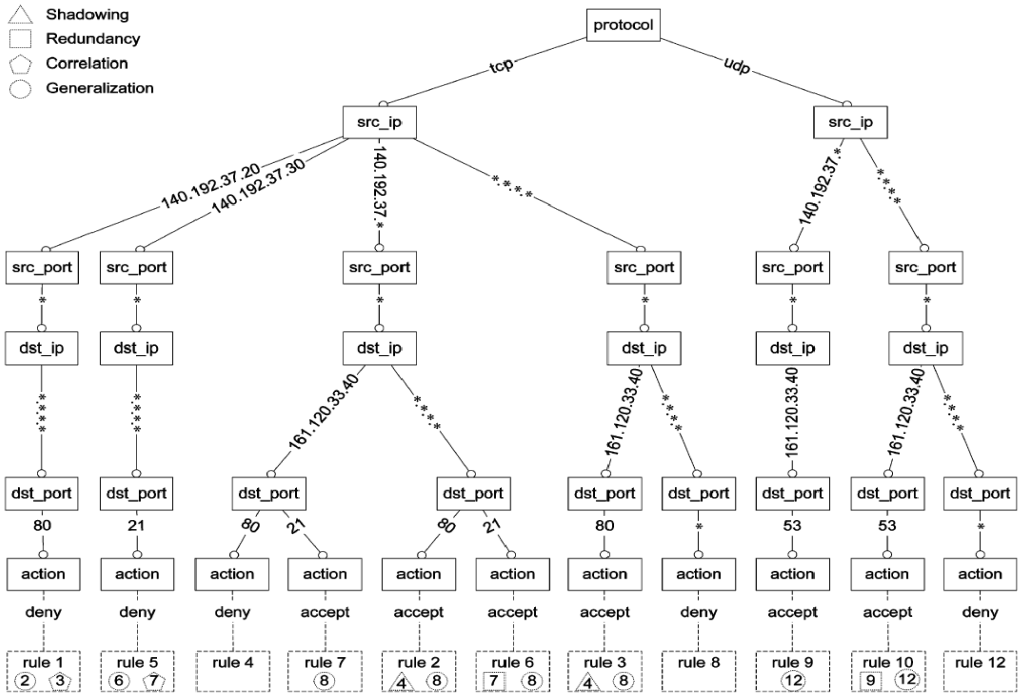


Figure 4.4 – *Arbre de politique du pare-feu de la Figure 4.3*

2. Source de la figure : [27]
3. Source de la figure : [27]

Dans cet arbre plus ou moins simple, car composé seulement de 2 protocoles, on voit facilement le lien entre les différentes règles ainsi que les anomalies qui peuvent en résulter. Il suffit de vérifier le protocole utilisé dans la règle (udp ou tcp dans notre cas) puis comparer les adresses et ports sources et destinations avant de voir les actions pour savoir s'il s'agit d'une règle en conflit avec une autre ou non en utilisant les règles précédemment définies .

4.3 Quelques solutions

Le renforcement de politique de sécurité fait l'objet de plusieurs recherches depuis quelques décennies et de nombreuses approches ont été proposées pour pallier à ces multiples anomalies qui entravent une mise en oeuvre efficace et optimale d'une politique de sécurité.
Dans la présente section, nous passons en revue quelques unes de ces approches. Nous établissons ensuite une synthèse comparative des différentes techniques utilisées.

4.3.1 Approche algébrique

La première approche que nous étudions est celle proposée par Touhami dans [28] : *Approche algébrique pour la sécurisation des réseaux informatiques.*
Cette approche consiste à développer une algèbre de processus (CMN) pour modéliser les réseaux informatiques ainsi qu'une logique propositionnelle pour spécifier une politique de sécurité. On définit ensuite un opérateur de renforcement qui génère une version du réseau respectant la politique.
Le principe peut donc se résumer comme suit :
Étant donné un réseau R et une politique de sécurité M, on se propose de définir un opérateur de renforcement $\otimes$ qui permet de générer une nouvelle version R' du réseau respectant la politique M. En d'autres termes, R' doit satisfaire les deux propriétés suivantes :

1. $R' \models M$: C'est à dire que R' ne doit pas violer la politique de sécurité M. Donc pour toute action a qui fait passer le réseau d'un état S à un état S', on a $a \models M$;
2. $R' \sqsubseteq_M R$: Toute action de la nouvelle version du réseau qui ne satisfait pas M sera bloquée (voir [28] pour plus de détails sur la définition de $\sqsubseteq_M$).

Cette approche est composée de trois étapes : D'abord, on modélise le réseau R avec une algèbre de processus, puis on spécifie formellement la politique de sécurité M avec une

logique propositionnelle et enfin on définit l'opérateur de renforcement $\otimes$ nous permettant de générer une version de R respectant M. Par la suite, on peut vérifier quelques propriétés telles que la correction et la complétude de l'opérateur .
La Figure 4.5 illustre ces différentes étapes de l'approche.

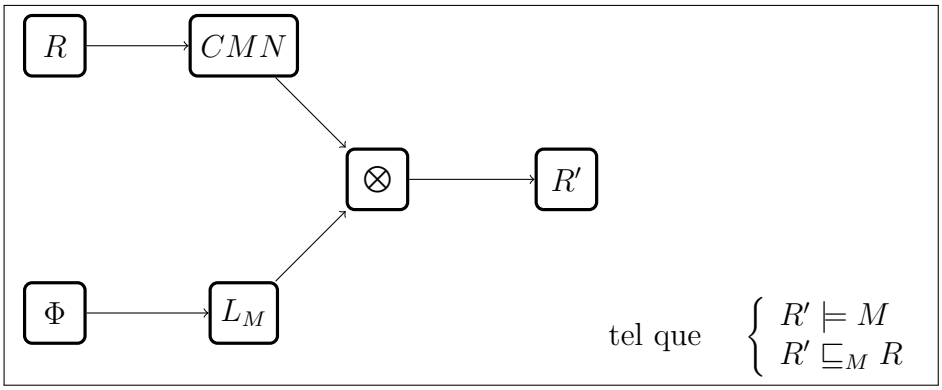


Figure 4.5 – *Approche algébrique pour la sécurisation des réseaux informatiques*

4.3.1.1 *CMN* (***Calculus for Monitored Network***)

CMN est une algèbre qui permet de décrire le comportement des processus réseaux où les actions de ces processus sont exprimées par des envois de messages ou des communications. Sa syntaxe est illustrée dans le tableau 4.2.
L'opérateur de surveillance, noté par $[\,]_M^N$, permet de contrôler toutes les actions externes d'un processus. $[P]_M^N$ signifie que le processus P, qui s'exécute dans l'emplacement N, est contrôlé par le moniteur M.
La sémantique opérationnelle des opérateurs de base est donnée dans [28], nous nous contentons alors de reprendre celle de l'opérateur de surveillance que nous présentons dans le Tableau 4.3.

Ces règles montrent l'évolution des différentes formes d'un processus monitoré. Nous donnons une idée du rôle de chacune d'elles :

$R^1_{[\,]}$ **et** $R^2_{[\,]}$ **:** Ces deux règles expriment respectivement le fait que le moniteur n'a aucun contrôle sur les communications internes d'un processus et sur les actions silencieuses. Donc si un processus P peut exécuter l'action τ et devenir P', alors le processus monitoré $[P]_M^N$ peut avancer en exécutant la même action τ et devenir $[P']_M^N$.

Tableau 4.2 – *Syntaxe de l'algèbre CMN*

Actions	$\alpha :=$	a	Envoi sur le canal a
		$\bar{a}$	Reception sur le canal a
		$\vec{a}$	Message en circulation
		τ	Action silencieuse
Processus	$P :=$	0	Processus vide
		$\alpha \cdot P$	Préfixage
		$X \stackrel{\text{def}}{=} P$	Définition
		$\sum_{i \in I} P_i$	Choix
		$P_1 \| P_2$	Parallélisme 1
		$\vec{a} \| P_2$	Parallélisme 2
		$P[f]$	Renommage
		$P \backslash L$	Restriction
		$[P]_M^N$	Surveillance

$R^3_{[\ ]}$ **:** Cette règle exprime le fait que si un processus peut avancer en exécutant une action d'envoi a, alors le processus monitoré avance en exécutant la même action. Le paquet qui circule dans le réseau ($\vec{a}$) reste dans la zone contrôlée par le moniteur M.

$R^4_{[\ ]}$ **et** $R^5_{[\ ]}$ **:** Ces deux règles permettent de contrôler les communications externes d'un processus. La première exprime le fait qu'un message en circulation dans le réseau $\vec{a}$, peut quitter la zone contrôlée par un moniteur M si et seulement s'il est autorisé par M. Par exemple, le processus monitoré $[\vec{a} \| P]_M^N$ peut exécuter l'action τ et devenir le processus $\vec{a} \| [P]_M^N$ si et seulement si $a \in O(N)$ et $a \models M$.
Contrairement à la règle $R^4_{[\ ]}$, $R^5_{[\ ]}$ montre comment le moniteur intercepte tout paquet qui viole la politique de sécurité.

$R^6_{[\ ]}$ **et** $R^7_{[\ ]}$ **:** Ces deux règles permettent de contrôler les entrées d'un processus de la même manière que celles pour les sorties.

Tableau 4.3 – *Sémantique de l'opérateur de surveillance*

$R^1_{[\,]}$:	$\dfrac{P \xrightarrow{c(a)} P'}{[P]^N_M \xrightarrow{c(a)} [P']^N_M}$	Communication
$R^2_{[\,]}$:	$\dfrac{P \xrightarrow{\tau} P'}{[P]^N_M \xrightarrow{\tau} [P']^N_M}$	Action silencieuse
$R^3_{[\,]}$:	$\dfrac{P \xrightarrow{a} \vec{a}\|P'}{[P]^N_M \xrightarrow{a} [\vec{a}\|P']^N_M}\ a \in \mathcal{A}$	Envoie
$R^4_{[\,]}$:	$\dfrac{\square}{[\vec{a}\|P]^N_M \xrightarrow{\tau} \vec{a}\|[P]^N_M}\ a \in O(N), a \models M$	Sortie externe
$R^5_{[\,]}$:	$\dfrac{\square}{[\vec{a}\|P]^N_M \xrightarrow{\tau} [P]^N_M}\ a \in O(N), a \not\models M$	Sortie bloquée
$R^6_{[\,]}$:	$\dfrac{\square}{\vec{a}\|[P]^N_M \xrightarrow{\tau} [\vec{a}\|P]^N_M}\ a \in I(N), a \models M$	Entrée
$R^7_{[\,]}$:	$\dfrac{\square}{\vec{a}\|[P]^N_M \xrightarrow{\tau} [P]^N_M}\ a \in I(N), a \not\models M$	Entrée bloquée

Exemple 4.3.1 (Modélisation d'un réseau avec CMN)**.**
Soit R_1 le réseau de la Figure 4.6. À partir de l'architecture de R_1, nous allons utiliser CMN pour trouver le processus réseau correspondant.
Rappelons d'abord les notations suivantes :

- *$a.P$ désigne $\sum_{m \in \mathcal{M}} a(m).P$, i.e. un processus qui peut envoyer n'importe quel message sur le canal a ;*
- *$\bar{a}.P$ désigne $\overline{a}(x).H$, i.e. un processus qui peut recevoir n'importe quel message sur le canal a ;*
- *$[P]^N$ désigne $[P]^N_M$ où M est un moniteur qui accepte n'importe quel message.*

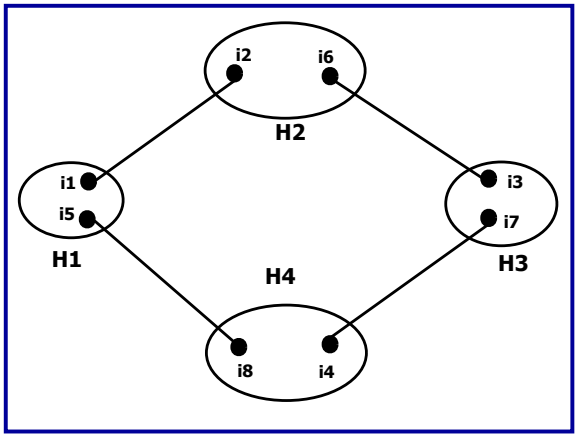


Figure 4.6 – Architecture simple d'un réseau

Le réseau R_1 est donc représenté par le processus suivant :

$$R_1 \stackrel{def}{=} [H_1]^{N_1} \| [H_2]^{N_2} \| [H_3]^{N_3} \| [H_4]^{N_4}$$

où :

$$H_1 \stackrel{def}{=} \left(\sum_{m \in \mathcal{M}} i_1(m).H_1 \right) + \overline{i_2}(x).H_1 + \left(\sum_{m \in \mathcal{M}} i_5(m).H_1 \right) + \overline{i_8}(y).H_1$$

$$H_2 \stackrel{def}{=} \left(\sum_{m \in \mathcal{M}} i_2(m).H_2 \right) + \overline{i_1}(x).H_2 + \left(\sum_{m \in \mathcal{M}} i_6(m).H_2 \right) + \overline{i_3}(y).H_2$$

$$H_3 \stackrel{def}{=} \left(\sum_{m \in \mathcal{M}} i_3(m).H_3 \right) + \overline{i_6}(x).H_3 + \left(\sum_{m \in \mathcal{M}} i_7(m).H_3 \right) + \overline{i_4}(y).H_3$$

$$H_4 \stackrel{def}{=} \left(\sum_{m \in \mathcal{M}} i_4(m).H_4 \right) + \overline{i_7}(x).H_4 + \left(\sum_{m \in \mathcal{M}} i_8(m).H_4 \right) + \overline{i_5}(y).H_4$$

On peut lire H_1 comme suit : c'est un processus qui peut envoyer des messages à travers les interfaces i_1 et i_5 et se comporter à nouveau comme H_1 (récursivement). Il peut également recevoir des messages en provenance des interfaces i_2 et i_8 et se comporter de nouveau comme H_1.

4.3.1.2 L_M

La logique utilisée pour spécifier une politique de sécurité est L_M. Sa syntaxe est présentée dans le Tableau 4.4. On appelle moniteur toute formule de L_M.

Tableau 4.4 – *Syntaxe de* L_M

$Moniteur\ M$	$::=$	$\top \mid p \mid a(p) \mid \bar{a}(p) \mid \neg M \mid M_1 \wedge M_2$
p	$::=$	$Champ \quad Op \quad Val$
$Champ$	$::=$	$ip\_src \mid ip\_dst \mid port\_src \mid port\_dst \mid prot$
Op	$::=$	$= \mid < \mid \in$
Val	$::=$	$Num \mid TCP \mid UDP \mid ICMP \mid Num.Num.Num.Num \mid *$
a	$::=$	nom d'un canal

La sémantique de L_M est illustrée dans le Tableau 4.5.

Tableau 4.5 – *Sémantique de* L_M

$a(m) \models \top \quad \bar{a}(m) \models \top$	$\tau \models M$
$a(m) \models Champ\ Op\ Val$	Si $m.Champ\ Op\ Val = true$
$\bar{a}(m) \models Champ\ Op\ Val$	Si $m.Champ\ Op\ Val = true$
$a(m) \models i(p)$	Si $a = i$ et $m.Champ\ Op\ Val = true$
$\bar{a}(m) \models \bar{i}(p)$	Si $a = i$ et $m.Champ\ Op\ Val = true$
$a(m) \models \neg M$	Si $a(m) \not\models M$
$\bar{a}(m) \models \neg M$	Si $\bar{a}(m) \not\models M$
$a(m) \models M_1 \wedge M_2$	Si $a(m) \models M_1$ et $a(m) \models M_2$
$\bar{a}(m) \models M_1 \wedge M_2$	Si $\bar{a}(m) \models M_1$ et $\bar{a}(m) \models M_2$
$c(a(m)) \models M$	Si $a(m) \models M$ et $\bar{a}(m) \models M$

Exemple 4.3.2 (Spécification d'une politique en L_M)**.** *Considérons à nouveau le réseau de l'exemple 4.3.1. Étant donné la politique de sécurité suivante :*

- *H_1 ne peut pas envoyer à la machine H_2 sur le port 25 par l'interface i_1 ;*
- *H_2 ne peut pas recevoir de la machine H_3 par l'interface i_6 ;*
- *Les paquets partant de H_3 sur le port 80 sont interceptés par le moniteur.*

Cette politique peut être spécifiée en L_M par :

$$M = M_1 \wedge M_2 \wedge M_3$$

avec :

- $M_1 = \neg i_1(ip\_src = ip\_src(H_1) \wedge ip\_dst = ip\_dst(H_2) \wedge port\_src = 25)$;
- $M_2 = \neg i_6(ip\_src = ip\_src(H_3) \wedge ip\_dst = ip\_dst(H_2)$;
- $M_3 = \neg(ip\_src = ip\_src(H_3) \wedge port\_src = 80)$.

4.3.1.3 Opérateur de renforcement ($\otimes$)

Le problème à résoudre admet plusieurs solutions. La plus évidente consiste à placer le moniteur M sur chaque composante du réseau. L'opérateur de renforcement $\otimes$ suivant est basé sur ce principe, donc satisfait les conditions de la section 4.3.1.

$$
\begin{aligned}
[P]^N_{M_0} \otimes M &= [P]^N_{M_0 \wedge M} \\
([P_1]^{N_1}_{M_1} \| \cdots \| [P_1]^{N_n}_{M_n}) \otimes M &= [P_1]^{N_1}_{M_1 \wedge M} \| \cdots \| [P_1]^{N_n}_{M_n \wedge M} \\
(\vec{a_1} \| \cdots \| \vec{a_m} \| [P_1]^{N_1}_{M_1} \| \cdots \| [P_1]^{N_n}_{M_n}) \otimes M &= \vec{a_1} \| \cdots \| \vec{a_m} \| [P_1]^{N_1}_{M_1 \wedge M} \| \cdots \| [P_1]^{N_n}_{M_n \wedge M}
\end{aligned}
$$

Exemple 4.3.3 (Application de l'opérateur $\otimes$)**.**
Si nous considérons le réseau R_1 de l'exemple 4.3.1 et la politique M spécifiée dans l'exemple 4.3.2, nous pouvons appliquer l'opérateur de renforcement. Ce qui donne le résultat suivant :

$$R_1 \otimes M = [H_1]^{N_1}_M \| [H_2]^{N_2}_M \| [H_3]^{N_3}_M \| [H_4]^{N_4}_M$$

Une version améliorée de cet opérateur est obtenue après application d'une fonction d'optimisation. Mais la solution reste peu pratique dans plusieurs situations.
Une solution plus intéressante est d'offrir à l'administrateur la possibilité de fournir certaines informations comme les emplacements des moniteurs : c'est le monitorage sélectif. Le problème du monitorage sélectif [28] consiste à trouver un nouvel opérateur $\otimes_S$, qui, à partir d'un réseau R, d'une politique de sécurité M et d'un ensemble de composantes réseau S, génère automatiquement un nouveau réseau R' respectant M dans lequel les moniteurs sont placés uniquement sur les éléments de S. La figure 4.7 illustre toute l'approche du monitorage sélectif.

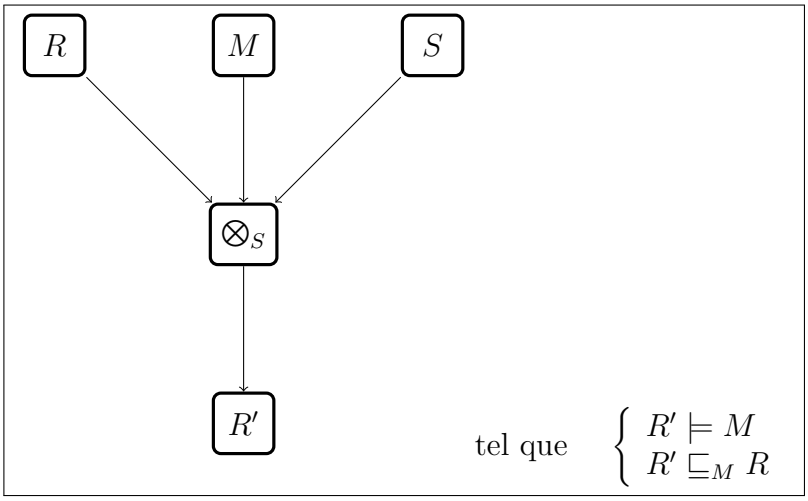


Figure 4.7 – *Opérateur de monitorage sélectif* $\otimes_S$

4.3.2 Filtrage de postures

Dans [29], Guttman décrit une approche pour générer des filtres basés sur une politique de sécurité et de vérifier que chaque filtre de paquets met en oeuvre correctement la politique en question. Il définit notamment un langage simple, dont la syntaxe est proche de LISP, permettant d'exprimer des politiques globales de contrôle d'accès où les routeurs sont capables d'assurer le filtrage. Il introduit ensuite un algorithme qui, à partir de la topologie d'un réseau, génère un ensemble de filtres pour les routeurs individuels garantissant une application correcte de la politique que l'on veut mettre en oeuvre.
Un deuxième algorithme compare un ensemble résultant de ces filtres à la politique globale du réseau pour déterminer toutes violations des politiques, ou pour signaler qu'il n'en existe pas.
Le principe est le suivant :
Deux filtres sont associés à chaque interface d'un routeur : l'un examine les paquets entrants (interface → routeur) et l'autre analyse les paquets sortants (routeur → interface). Un filtre est représenté par une contrainte ϕ qui ne laisse passer un paquet p que lors $p \in \phi$.
Les deux algorithmes peuvent s'expliquer comme suit :

1. **Vérification de posture :** Étant donnée une politique de sécurité P et un filtre F, la vérification de posture consiste à déterminer si F renforce correctement la politique P. Sinon, énumérer les contre-exemples.
2. **Génération de posture :** Étant donnée une politique P, la génération de posture consiste à construire un filtre F garantissant une mise en oeuvre correcte de P.

Pour mieux comprendre le fonctionnement de ces algorithmes, nous avons besoin de définir les notions suivantes :

Définition 4.3.1 (Posture de filtrage)**.**
Soit $\langle \pi = \ell_0, ..., \ell_n \rangle$ un chemin, avec ℓ_i une zone ou un routeur d'un graphe bipartite. La notation $\pi \frown \pi'$ est utilisée pour la concaténation de π et π'.
Une posture de filtrage est défini comme étant une fonction de la forme $f : R \times A \times D \longrightarrow B$, avec :

- ***R** : un ensemble de routeurs ;*
- ***A** : un ensemble de zones ;*
- ***D** : un ensemble contenant les directions* in *et* out *signifiant respectivement vers le routeur ou en provenance du routeur.*
- ***B** : une algèbre de Boole sur les ensembles de paquets.*

On choisit aussi par convention que $f(r, a, d) = \emptyset$, si r n'a pas d'interface sur a.

Définition 4.3.2 (Ensemble de faisabilité d'un chemin)**.**
Soit $\phi(\ell_i, \ell_j) = f(\ell_i, \ell_j, out)$. Si ℓ_i est un routeur et ℓ_j une zone réseau et $f(\ell_i, \ell_j, in)$ dans le cas inverse (lorsque ℓ_j est un routeur et ℓ_i une zone réseau).
L'ensemble de faisabilité du chemin π, noté $\Gamma(\pi)$, est défini par :

$$\begin{aligned} \Gamma(\langle\rangle) &= \top, \text{ l'élément au sommet de l'algèbre}; \\ \Gamma(\pi \frown \langle \ell_i, \ell_{i+1} \rangle) &= \Gamma(\pi) \cap \phi(\ell_i, \ell_{i+1}). \\ \Gamma(\pi \frown \pi') &= \Gamma(\pi) \cap \Gamma(\pi') \end{aligned}$$

Ainsi l'ensemble de faisabilité d'un chemin π est l'ensemble des paquets qui survivent à tous les filtres sur le chemin π.

4.3.2.1 Vérification de posture

Définition 4.3.3 (Correction d'une posture)**.**
Une posture F renforce correctement une politique $P : A \times A \longrightarrow B$ si et seulement si, tout paquet p allant d'un chemin $\pi = a_0 \cdots a_n$ et survivant à tous les filtres satisfait la contrainte $\Gamma(\pi) \subset P(a_0, a_n)$.

La vérification d'une posture consiste à parcourir tous les chemins π sans cycle allant d'une zone a à une zone a' et vérifier que $\Gamma(\pi) \subset P(a, a')$. Si cette condition n'est pas vérifiée alors l'ensemble des paquets $\{\Gamma(\pi) \backslash P(a, a')\}$ viole la politique de sécurité.

4.3.2.2 Génération de posture

Comme la vérification, pour générer une posture f, on calcule l'ensemble de faisabilité de tous les chemins π. Ensuite, à partir d'une posture arbitraire f_0, on corrige toute violation de la politique en faisant une mise de f_{i+1} avec la transformation suivante :

$$\begin{cases} f_{i+1} = f_i & Si\ \Gamma(\pi) \subset P(a, a') \\ f_{i+1}(r, a_n, out) = f_i(r, a_n, out) \backslash V & Sinon \end{cases}$$

avec $V = \Gamma(p) \backslash P(a_0, a_n)$

En itérant ce processus, on génère une posture de filtrage garantissant la mise en oeuvre de la politique P. Cette approche est utilisée pour implanter un prototype de vérification automatique d'une politique de sécurité.

4.3.3 Firmato (*Firewall Management Toolkit*)

Firmato [30], est un outil aux fonctionnalités similaires au filtrage de posture. Cependant, il est basé sur un modèle entité-relation et un langage (MDL) se servant de ce modèle, tous deux dédiés à la description des réseaux et des propriétés de sécurité. Firmato dispose également d'un compilateur générant automatiquement les règles de filtrage des pare-feu ainsi qu'un illustrateur pour leur visualisation.
Le principe de fonctionnement de Firmato peut se résumer comme suit :
Après la définition et la spécification de la politique de sécurité en MDL, cette version en MDL est ensuite utilisée par l'analyseur pour générer un modèle entité-relation. Ce dernier est transformé par le compilateur en fichiers de configuration de pare-feu garantissant une mise en oeuvre correcte de la politique. En effet, le principe de fonctionnement du compilateur est basé sur une notion de rôle. Un rôle est la description des opérations autorisées entres les hôtes selon leurs fonctions.

Les différentes composantes de Firmato peuvent être décrites comme suit :

1. **Le modèle entité-relation :** Cette entité contient sous une forme unifiée, une vue globale de la politique de sécurité ainsi que la topologie du réseau.
2. **Le langage de définition de modèle (MDL)** : MDL est un langage simple pour spécifier aussi bien une politique de sécurité que la topologie d'un réseau. Il permet ainsi de définir une instance du modèle entité-relation.

3. **Le Compilateur :** Le compilateur permet de transformer le modèle entité-relation en fichiers de configuration spécifiques aux pare-feu. Il ignore la structure du réseau et se focalise sur les définitions des rôles, des groupes de rôles et de leurs assignations aux groupes d'hôtes. À partir de ces informations, il déduit les groupes d'hôtes qui devraient avoir des règles autorisant certains services entre eux. La question de la passerelle qui pourra appliquer cette règle reste donc ignorée. Ainsi, le compilateur génère une base de règles centralisée contenant toutes les règles nécessaires pour mettre en oeuvre la politique.
4. **Le distributeur :** La base de règles générée par le compilateur doit être distribuée sur chacune des passerelles du réseau avec une personnalisation appropriée selon les interfaces. Pour s'assurer que la politique de sécurité est correctement renforcée, une règle concernant une paire d'hôtes doit être incluse dans chacune des passerelles se trouvant sur chaque chemin reliant les deux hôtes.
   Afin d'éviter de faire des hypothèses sur les protocoles de routage, la stratégie adoptée consiste à reproduire la base de règles sur toutes les interfaces de chaque passerelle. Toutefois, la direction du domaine des règles doit être définie car c'est important pour réduire les attaques d'usurpation. Pour ce faire, le pare-feu examine la direction dans laquelle le paquet s'introduit dans l'interface de la passerelle et la compare avec la direction du domaine de la règle. Si le paquet tente de quitter l'interface pour une zone adjacente, il ne sera autorisé que lorsque la direction de la règle est entrante ou double. De même, un paquet ne sera autorisé à entrer dans l'interface depuis une zone adjacente que lorsque la direction de la règle est sortante ou double. Le distributeur utilise un algorithme [30] pour déterminer les directions et permettre ainsi de diminuer le nombre de règles ayant une direction double afin d'être le plus précis possible.
5. **L'illustrateur :** L'illustrateur donne une représentation graphique des règles du pare-feu et permet ainsi de faire une rétro-ingénierie pour extraire ou retrouver la politique à partir de la configuration. Il prend en entrée le modèle généré par le compilateur et le transforme en un graphe visualisant la structure des groupes d'hôtes (*host-groups*) et des services autorisés. Il crée ainsi une visualisation de la politique comme on pourrait le voir sur une seule interface. Le débogage devient donc plus facile. On peut également retrouver la politique à partir de ce résultat.

4.3.4 Diagramme de décision d'un pare-feu (FDD)

Dans [31], Gouda *et al.* présentent une méthode pour la conception et l'ordonnancement des séquences de règles dans un pare-feu afin que ce dernier soit cohérent, complet et consistant. La cohérence indique que les règles sont correctement ordonnées, la complé-

tude signifie que chaque paquet doit satisfaire au moins une règle dans le pare-feu et la consistance est le fait que le pare-feu est dépourvu de toute redondance entre ses règles. Pour atteindre un tel objectif, Gouda *et al.* commencent par concevoir un diagramme de décision de pare-feu, FDD (*firewall decision diagram*) dont la cohérence et la complétude peuvent être vérifiées systématiquement (par un algorithme). On applique alors une série d'algorithmes à ce diagramme pour générer, réduire et simplifier les règles de pare-feu tout en maintenant la cohérence et la complétude du FDD original.

Définition 4.3.4 (**Diagramme de décision d'un pare-feu**). *Un diagramme de décision f d'un pare-feu est un arbre ayant les cinq propriétés suivantes :*

1. *Il n' y a qu'un noeud dans f n'ayant pas d'arêtes entrantes. C'est la racine de f. Les noeuds de f qui n'ont pas d'arêtes sortantes sont les noeuds terminaux de f.*
2. *Chaque noeud v a une étiquette $F(v)$ tel que :*

$$F(v) = \begin{cases} \{F_1, ...F_d\} & \text{si } v \text{ n'est pas un noeud terminal,} \\ DS, & \text{Si } v \text{ est un noeud terminal.} \end{cases}$$

où D est un ensemble de décision.

3. *Chaque arête e dans f a une étiquette $I(e)$ tel que si e est une arête sortante du noeud v, alors $I(e)$ est un sous-ensemble non vide de $D(F(v))$.*
4. *Un chemin direct de la racine à un noeud terminal dans f est appelé un chemin de décision de f. Deux noeuds sur un même chemin de décision ne peuvent avoir la même étiquette.*
5. *$E(v)$ désigne l'ensemble des arêtes sortantes d'un noeud v et satisfait les deux conditions suivantes :*

(a) ***La cohérence :*** $I(e) \cap I(e') = \emptyset$ *pour toutes arêtes e et e' distinctes dans $E(v)$,*

(b) ***la complétude :*** $\bigcup_{e \in E(v)} I(e) = D(F(v))$.

Exemple 4.3.4. *Soit f le pare-feu du Tableau 4.6.*

r_1 :	$S \in [4,7]$	$\wedge\ D \in [6,8]$	$\rightarrow a$
r_2 :	$S \in [4,7]$	$\wedge\ D \in [2,5] \cup [9,9]$	$\rightarrow d$
r_3 :	$S \in [4,7]$	$\wedge\ D \in [1,1] \cup [10,10]$	$\rightarrow a$
r_4 :	$S \in [3,3] \cup [8,8]$	$\wedge\ D \in [2,9]$	$\rightarrow d$
r_5 :	$S \in [3,3] \cup [8,8]$	$\wedge\ D \in [1,1] \cup [10,10]$	$\rightarrow a$
r_6 :	$S \in [1,2] \cup [9,10]$	$\wedge\ D \in [1,10]$	$\rightarrow a$

Tableau 4.6 – *Un pare-feu consistant f*

Le diagramme de décision équivalent est représenté par la Figure 4.8.

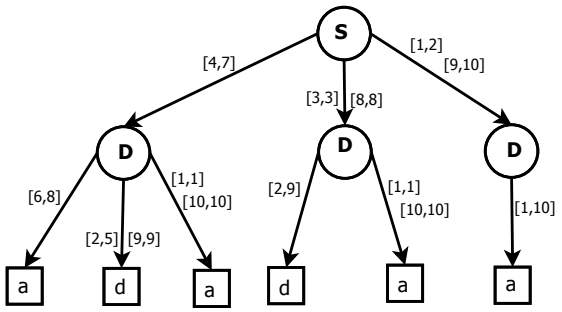


Figure 4.8 – *Diagramme de décision du pare-feu f*

4.3.5 Autres solutions

1. **Rétro-ingénierie :** Dans [32], Alex X. *et al.* présentent un langage structuré pour la spécification des requêtes dans un pare-feu : la SFQL (*Structured Firewall Query Language*).

   Une requête en SFQL, a le format suivant :

   select F_i
   from f
   where $(F_1 \in S_1) \wedge ... \wedge (F_d \in S_d) \wedge (\textbf{decision} = \langle dec \rangle)$

   – F_i désigne un des champs $F_1, ..., F_d$;
   – f est le pare-feu sur lequel s'effectue la requête ;
   – Chaque S_j est un sous-ensemble non vide du domaine $D(F_j)$ du champ F_j ;
   – $\langle dec \rangle$ est soit"*accepter*" ou "*rejeter*".

   Exemple 4.3.5 (Requêtes exprimées avec $SFQL$)**.**
   Question 1 :
   Quels hôtes du réseau privé protégé par le pare-feu f peuvent recevoir des paquets UDP provenant de l'Internet sur les ports 67 ou 68 ?

   Requêtes Q_1 **:**
   select D
   from f
   where $(I \in \{0\}) \wedge (S \in all) \wedge (D \in all) \wedge (N \in \{67, 68\})$
   $\wedge (P \in \{UDP\}) \wedge (\textbf{\textit{decision}} = accept)$

   0 *désignant l'interface du pare-feu f qui le relie à l'Internet.*

Question 2 :
Quels ports du serveur mail protégé par le pare-feu f sont accessibles ?

Requêtes Q_2 :
select N
from f
where $(I \in \{0,1\}) \wedge (S \in all) \wedge (D \in \{ServeurMail\}) \wedge (N \in all)$
$\wedge (P \in all) \wedge (\textbf{\textit{decision}} = accept)$

1 *désignant l'interface du pare-feu f qui le relie avec le réseau local.*

D'autres requêtes plus complexes peuvent être exprimées avec $SFQL$.

Le résultat d'une requête Q, noté $Q.result$, est défini par l'ensemble suivant :
$\{p_i | (p_1, ..., p_d) \in \Sigma$, et
$(p_1 \in S_1) \wedge ... \wedge (p_d \in S_d) \wedge (f.(p_1, ..., p_d) = \langle dec \rangle)\}$.
où Σ désigne un ensemble de paquets.

Pour obtenir ce $Q.result$, on commence par trouver tous les paquets $(p_1, ..., p_d)$ dans Σ satisfaisant la condition suivante :

$$(p_1 \in S_1) \wedge ... \wedge (p_d \in S_d) \wedge (f((p_1, ..., p_d)) = \langle dec \rangle).$$

Le théorème 4.1 permet de déterminer cet ensemble.

Théorème 4.1 ($Q.result$)**.** *Soit Q une requête de la forme :*
select F_i
from f
where $(F_1 \in S_1) \wedge ... \wedge (F_d \in S_d) \wedge (\textbf{\textit{decision}} = \langle dec \rangle)$
et f un pare-feu consistant, composé de n règles $r_1, ..., r_n$
où chaque r_j est de la forme $(F_1 \in S_1') \wedge ... \wedge (F_d \in S_d') \rightarrow \langle dec \rangle$.
$Q.result$ est l'union des ensembles $Q.r_j$.

$$Q.result = \bigcup_{j=1}^{n} Q.r_j$$

où chaque $Q.r_j$ est défini en utilisant la règle r_j comme suit :

$$Q.r_j = \begin{cases} S_i \cap S_i' & si\ (S_1 \cap S_1' \neq \emptyset) \wedge ... \wedge (S_d \cap S_d' \neq \emptyset) \wedge (\langle dec \rangle = \langle dec' \rangle) \\ \emptyset & sinon. \end{cases}$$

Ce théorème permet de trouver la réponse à une requête, mais peut nécessiter trop de répétitions dans certains cas. Pour améliorer sa performance, on fait recours aux FDD 4.3.4. L'idée consiste à convertir le pare-feu en un FDD équivalent et comme le FDD est une représentation consistante et compacte du pare-feu original, il sera utilisé comme structure de donnée principale pour le traitement des requêtes.

2. **FLIP** [33] est un autre langage de haut-niveau pour la configuration des listes de contrôle d'accès des pare-feu afin de renforcer correctement une politique de sécurité. FLIP est également compilable dans plusieurs langages de plus bas niveau. Cependant, une de ses limites est qu'elle ne prend pas en charge les chevauchements entre les sélecteurs des règles. Or, l'interdiction de ce chevauchement est un obstacle majeur, car il est impossible d'exprimer des exceptions. Ce qui pourrait emmener l'administrateur à écrire beaucoup de règles pour exprimer un chevauchement entre des sélecteurs. En outre, sa syntaxe est encore plus complexe que celle de Firmato. Toutefois, malgré ce manque d'expressivité, les ACLs générées sont indépendantes de toute ordre.

3. D'autres travaux portant sur le renforcement de politique ont été explorés durant ce travail. Dans [2], Stouls *et al.* ont utilisé la méthode du **B**-événementiel [34] pour développer formellement, par raffinements successifs, un moniteur qui surveille la conformité d'un réseau par rapport à une politique de sécurité donnée.
   Dans [35] et [36], Mayer *et al.* proposent deux outils passifs basés sur un moteur de requêtes pour analyser un pare-feu. De même, Verma *et al.* ont proposé *FACE* [37] (*A Firewall Analysis and Configuration Engine*), un outil permettant de générer et d'analyser automatiquement les configurations de tous les pare-feu d'un réseau en spécifiant des filtres.

4.4 Synthèse

Dans cette section, nous allons analyser les différentes solutions que nous venons d'étudier afin de mettre en évidence leurs forces ainsi que les améliorations possibles. Pour ce faire, nous commençons par les répartir en trois approches qui sont l'approche formelle, la spécification par un langage de haut niveau et la rétro-ingénierie.

L'approche formelle consiste à modéliser le réseau à sécuriser ainsi que la politique à

renforcer avec des méthodes formelles telles que les algèbres de processus et les logiques propositionnelles ou temporelles. On définit par la suite un opérateur permettant de générer une nouvelle version du réseau conforme à la politique. Cette approche est illustrée dans [28, 2, 38], etc. et est caractérisée par sa rigueur ainsi que le formalisme mathématique utilisé pour prouver la satisfaction des contraintes. Cependant, son plus grand handicape est sans doute les nombreux mythes sur les méthodes formelles ([39, 40, 41]) qui demeurent encrer dans les mémoires et restent un défi à relever.

La spécification par un langage de haut niveau quant à elle consiste à définir un langage pour la spécification des politiques ainsi que les différentes entités ou services du réseau. On conçoit par la suite un algorithme pour la vérification et la génération des règles de configuration de pare-feu. Cette approche est utilisée dans [29, 30, 33], etc. L'un de ses apports est la définition de langages de haut niveau facilitant la spécification des politiques de sécurité. Néanmoins, la plupart de ces langages ont des complexités similaires à celles des langages de bas niveau.

Enfin, **la rétro-ingénierie** vise à vérifier la correction et la conformité d'une configuration par rapport à une politique en effectuant des requêtes permettant de retrouver la politique mise en oeuvre. Cette approche permet de faire une meilleure analyse des pare-feu afin de maîtriser leurs modes de fonctionnement. Ce qui simplifie considérablement la tâche de l'administrateur en lui offrant la possibilité de savoir quel hôte serait vulnérable lors qu'un autre est sous attaques. Une des limitations de la rétro-ingénierie est son manque d'autonomie. Elle nécessite une technique de reconfiguration lorsqu'une faille est détectée. Elle n'est pas exhaustive non plus, il faut savoir poser la bonne question, ce qui peut nécessiter l'intervention d'un expert dans certains cas.

La plupart de ces approches sont basées sur une description de la topologie du réseau réalisée par l'administrateur. Elles nécessitent donc un administrateur compétent, ainsi que des réseaux topologiquement statiques (ou changeant suffisamment peu souvent pour que l'administrateur puisse prendre ces modifications en compte dans le temps imparti), ce qui nuit à leur utilisation dans le cadre de certains réseaux modernes dont la topologie peut être dynamique.
Tout de même, ces multiples et différentes approches démontrent à quel point la recherche sur le renforcement de politique de sécurité est actif durant ces dernières décennies. Néanmoins, nous prétendons qu'aucune d'entre elles n'est pleinement satisfaisante au regard des exigences actuelles. Dans le tableau 4.7 nous présentons une synthèse de quelques unes des solutions étudiées. Cette synthèse montre notamment des manquements ou limitations de chacune de ces approches.

Approche	Avantages	Limites
Approche algébrique	– Approche très rigoureuse ; – La correction de l'opérateur de renforcement est mathématiquement prouvée ; – L'introduction d'une fonction d'optimisation rend la solution beaucoup plus efficace ; – Cette solution permet de prendre en compte le contrôle interne car les différentes entités du réseau sont des hôtes et non des ensembles d'hôtes ;	– **Performance :** Le fait d'ajouter le moniteur dans toutes les composantes du réseau peut nuire à sa performance voire même son efficacité ; – **Intervention humaine :** l'utilisateur doit intervenir pour choisir les emplacements des moniteurs. Lesquels ne sont pas forcément les meilleurs ; – Cette solution ne prend pas en compte le fait que le réseau à configurer peut disposer d'une politique autre que celle à renforcer. Ce qui peut être à l'origine de certaines anomalies telles que les redondances ou les ombrages ; – **Segmentation** : les noeuds sont considérés comme des hôtes et non des zones (ensembles d'hôtes), ce qui ne correspond pas à la segmentation actuelle des réseaux en plusieurs domaines.

Filtrage de posture	– La rigueur de cette approche fait qu'elle est le point de départ de plusieurs autres ; – Cette solution permet de résoudre le problème de localisation évitant ainsi toute sorte d'usurpation d'identité ; – La politique est renforcée indépendamment de la table de routage qui peut changer d'un instant à l'autre, ce qui permet de couvrir l'ensemble des chemins possibles ; – Cette approche fournit une excellente technique pour la vérification de la correction d'une politique de sécurité.	– **Intervention humaine** : les configurations locales générées sont complexes et parfois inefficaces. Ce qui nécessite une intervention manuelle dans le but d'introduire des optimisations locales ; – Les filtres choisis pour commencer les itérations ne prennent pas en charge l'existence d'une politique ; – Cette approche ne fournit pas une séparation complète entre la politique de sécurité et la topologie du réseau ou la génération automatique des règles de pare-feu, ce qui rend difficile la modularisation de la politique ainsi que sa réutilisation ; – Elle ne prend pas en charge la protection des routeurs qui mettent en oeuvre la politique de sécurité.

Firmato	– Les règles sont toujours exprimées de façon cohérentes et indépendantes de toute ordre ; – Firmato fournit un mécanisme permettant de séparer la politique de sécurité de la topologie du réseau ; – Facilite la rétro-ingénierie et offre un niveau élevé de débogage des fichiers de configuration ; – Les fichiers de configuration des pare-feu sont générés automatiquement.	– La complexité du langage utilisé est similaire à celle de beaucoup de langages de bas niveau ; – Le langage utilisé ne peut représenter la connaissance que dans la logique positive (les règles autorisant des trafics) ce qui complique l'expression des exceptions ; – Explosion du nombre de rôles si l'on veut étendre le modèle. Par exemple, pour prendre en charge des réseaux tels que les VPNs, il faudra définir des rôles pour chaque service (l'encryption, l'authentification, etc.) – Pas de vérification formelle de l'algorithme. – Il faut maîtriser le langage MDL pour encoder la politique car aucun mécanisme ne permet de faire la conversion automatiquement.

Rétro-ingénierie	– Cette approche fournit une méthode simple et efficace pour analyser un pare-feu ; – Cette technique permet de mieux comprendre le fonctionnement de chaque pare-feu, ce qui facilite énormément la tâche d'un nouveau administrateur ; – Permet de détecter les anomalies des pare-feu et y apporter des corrections. Ce qui peut simplifier considérablement les activités quotidiennes d'un administrateur ; – Permet également de savoir quels hôtes seraient vulnérables lorsqu'un autre est sous attaque ; – L'utilisation d'un langage proche de SQL facilite la formulation des requêtes ; – La prise en charge des opérateurs algébriques tels que l'union et l'intersection facilite et simplifie la formulation des requêtes.	– Les requêtes ne sont jamais exhaustives, il faut savoir faire la bonne requête au bon moment ; – Il faut faire des requêtes quotidiennement pour accéder à l'information, ce qui nécessite une présence humaine ; – Pas de technique de reconfiguration lorsqu'une faille est détectée. – La notion de priorité ne résout pas le problème des anomalies car elle peut causer des chevauchements et des redondances ou des ombrages. Ce qui peut mener à des résultats différents pour une même requête. – La notion d'équivalence entre deux pare-feu étant définie indépendamment de leurs sémantiques, deux pare-feu peuvent être équivalents selon une fonction de mappage et ne pas l'être selon une autre, ce qui peut entraver la consistance des résultats.

Tableau 4.7 – Synthèse de quelques solutions étudiées

Conclusion

Avec la complexité et la forte croissance des réseaux d'entreprise, la configuration manuelle traditionnelle des dispositifs de sécurité s'est avérée insuffisante pour combler l'écart entre le haut niveau des exigences de sécurité et le bas niveau des dispositifs d'implémentation. Ainsi, les pare-feu sont devenus des outils presque indispensables pour le renforcement de la politique de sécurité d'un réseau. Cependant, un pare-feu ne fournit qu'une protection équivalente à la qualité de sa configuration. Malheureusement, un nombre important de pare-feu ne sont pas configurés correctement pour assurer une bonne protection. Par conséquent, une multitude d'anomalies continuent d'entraver une administration efficace d'une politique de contrôle d'accès.

Dans ce chapitre, nous avons exploré les principales anomalies de configuration dont souffre l'administration des pare-feu. Après une identification et définition formelle, il devient plus facile de remédier à ces anomalies. Nous avons passé en revue quelques techniques de détection d'anomalies ainsi que plusieurs solutions proposées dans la littérature pour les remédier et produire des langages et outils permettant un renforcement automatique d'une politique de sécurité dépourvue de toute incohérence dans les règles des pare-feu.

L'analyse de ces différentes solutions nous a permis de constater plusieurs limitations pouvant mener à une situation inattendue. La plupart de ces limitations sont liées à l'intervention humaine qui est souvent obligatoire. D'où la nécessité de proposer une solution automatisant la configuration d'un réseau selon une politique donnée tout en prenant en considération les paramètres de qualité de services.

Deuxième partie

Travail réalisé

Chapitre 5

Configuration formelle et optimisée d'un réseau

Introduction

Dans le chapitre 4 nous avons décrit sommairement quelques anomalies rencontrées dans la configuration des réseaux en général et des pare-feu en particulier. Nous y avons également présenté plusieurs techniques de sécurisation et de renforcement de politique de sécurité d'un réseau.
Le présent chapitre a pour objectif de définir une nouvelle technique formelle de sécurisation qui, à partir d'une configuration initiale d'un réseau et d'une politique de sécurité, génère une nouvelle configuration qui respecte la politique de sécurité donnée tout en offrant une meilleure qualité de service au réseau.
Pour ce faire, nous modélisons le réseau par un graphe étiqueté et définissons le coût d'une solution de sécurité afin de pouvoir calculer la configuration à coût minimal.
Puis, nous spécifions la politique de sécurité par une logique propositionnelle et définissons un langage pour la spécification des pare-feu. Enfin, nous définissons un opérateur de renforcement basé sur un algorithme permettant de définir et de placer les pare-feu contrôlant le trafic entre les différents noeuds du graphe, produisant ainsi une nouvelle version sécurisée et optimale du réseau.

Le reste du chapitre est organisé comme suit : D'abord nous allons formaliser la problématique, puis dans la deuxième section, nous présentons la syntaxe et la sémantique d'une logique propositionnelle permettant de spécifier formellement une politique de sécurité. La section 5.3 porte sur la spécification du réseau. Nous y définissons un graphe

étiqueté pour la modélisation ainsi qu'un langage de spécification de pare-feu. Nous introduisons ensuite la notion de coût attribué à une configuration dans la section 5.4. La section 5.5 définit l'opérateur de renforcement. Un exemple complet d'utilisation de toute l'approche est donné dans la section 5.6. Enfin, quelques propriétés telles que la correction et la complétude de cet opérateur font l'objet de la section 5.7 avant de conclure.

5.1 Formulation de la problématique

Étant donné un réseau R et une politique de sécurité Φ, nous voulons définir un opérateur de renforcement générant une nouvelle version R' du réseau R qui respecte la politique Φ tout en optimisant la performance du réseau, c'est-à-dire en conservant sa capacité maximale (voir Figure 5.1).

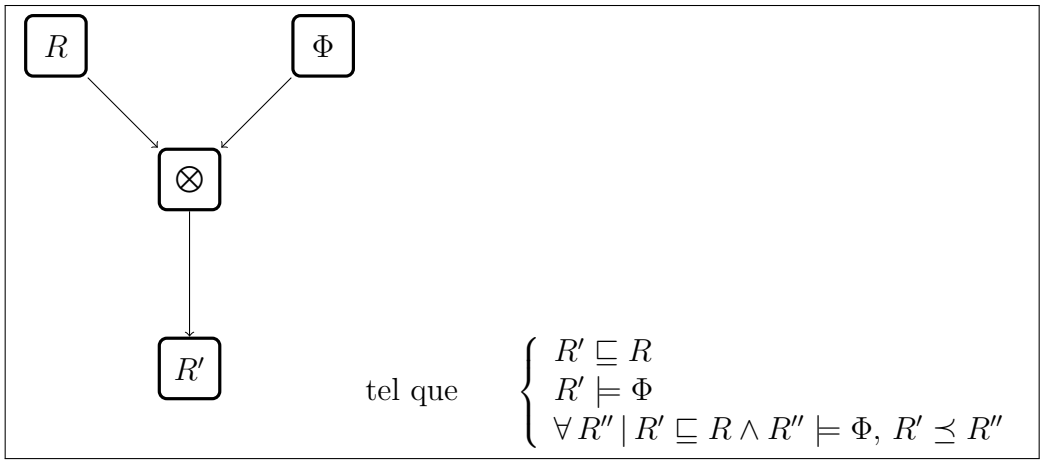


Figure 5.1 – *Problématique*

La nouvelle version de R doit donc respecter les trois propriétés suivantes :

1. $R' \sqsubseteq R$: R' conserve la même topologie que R, mais autorise moins de trafic. Une définition plus formelle de cette notion d'infériorité sera présentée dans la section 5.3.
2. $R' \models \Phi$: Le trafic dans R' doit se conformer à la politique de sécurité Φ. Autrement dit, si nous représentons le réseau R' par un graphe, on dit que R' satisfait Φ, noté par $R' \models \Phi$, lorsque pour tout paquet p :
   - si p transite d'un noeud $i \in R'$ vers un noeud $j \in R'$ alors $p \models \Phi$;
   - si p est bloqué dans R' alors $p \not\models \Phi$.

3. $(\forall\, R'' \mid R'' \sqsubseteq R \wedge R'' \models \Phi,\ R' \preceq R'')$: La nouvelle version R' doit être la meilleure des solutions qui respectent les deux premières propriétés. Une définition plus précise de cette notion d'optimalité est présentée dans la section 5.4.

5.2 Spécification d'une politique de sécurité

Cette section définit une logique propositionnelle simple, notée L_Φ [28], permettant de spécifier formellement une politique de sécurité. Une politique de sécurité est une formule de L_Φ.

5.2.1 Syntaxe de L_Φ

La syntaxe de la logique L_Φ est présentée dans le Tableau 5.1.

Tableau 5.1 – *Syntaxe de* L_Φ

Φ	$::=$	$\top \mid Predicat \mid \neg\Phi \mid \Phi_1 \wedge \Phi_2$
$Predicat$	$::=$	$Champ \quad Op \quad Val$
$Champ$	$::=$	$ip\_src \mid ip\_dst \mid port\_src \mid port\_dst \mid prot$
Op	$::=$	$= \mid < \mid \in$
Val	$::=$	$Num \mid TCP \mid UDP \mid ICMP \mid Num.Num.Num.Num \mid *$

Les valeurs de *Champ*, *Op* et *Val* sont juste à titre d'exemples, mais le langage L_Φ peut être facilement enrichi avec d'autres possibilités d'où l'utilisation du symbole $*$ pour indiquer les autres valeurs non spécifiées.

Les différentes constructions syntaxiques de L_Φ s'expliquent comme suit :

- $\top$: est utilisé pour exprimer la valeur booléenne "true".
- *Predicat* : est une proposition représentant une condition de la forme $\langle Champ\ Op\ Val \rangle$. Par exemple, la proposition $ip\_src = 132.203.114.23$ peut être utilisé pour indiquer qu'une adresse source est 132.203.114.23.
- $\neg\Phi$: est une formule qui représente une interdiction.
- $\Phi_1 \wedge \Phi_2$: est une formule représentant la conjonction de deux formules Φ_1 et Φ_2.

En plus de ces constructions syntaxiques, on peut utiliser les raccourcis suivants :

- $\bot = \neg\top$
- $\Phi_1 \vee \Phi_2 = \neg(\neg\Phi_1 \wedge \neg\Phi_2)$

5.2.2 Sémantique L_Φ

La sémantique d'une proposition Φ, notée par $[\![\Phi]\!]$ désigne l'ensemble des paquets qui respecte Φ. Formellement, si $\mathcal{P}$ est un ensemble de paquets, alors :

$$[\![\Phi]\!]_{\mathcal{P}} = \{p \in \mathcal{P} : p \models \Phi\}$$

où $p \models \Phi$ signifie que le paquet p respecte la formule Φ .

En supposant que $[\![\_]\!]_B$ évalue les conditions booléennes, nous définissons la sémantique de $p \models \Phi$ dans le Tableau 5.2 :

Tableau 5.2 – *Sémantique de* L_Φ

$p \models \top$	
$p \models \neg\varphi$	*si* $p \not\models \varphi$
$p \models Champ\ Op\ Val$	*si* $[\![Champ(p)\ Op\ Val]\!]_B = vrai$
$p \models \varphi_1 \wedge \varphi_2$	*si* $p \models \varphi_1$ et $p \models \varphi_2$

Nous pouvons également étendre cette logique avec des opérateurs temporels, mais pour des raisons de simplicité, nous nous limitons aux expressions ci-dessus.

5.3 Modélisation du réseau

Dans le cadre de ce travail, nous modélisons un réseau au moyen d'un graphe étiqueté défini comme suit :

Définition 5.3.1 (Graphe étiqueté)**.**
Un graphe étiqueté G est un triplet : $(\mathcal{N}, \mathcal{L}, \mathcal{A})$, avec :
- $\mathcal{N}$: *un ensemble de noeuds ;*
- $\mathcal{L}$: *un ensemble d'étiquettes spécifiées par le langage FL définies ci-après ;*
- $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N} \mapsto \mathcal{L}$ *: une relation partielle d'étiquetage.*

En d'autres termes, un réseau peut être représenté visuellement par un graphe orienté où chaque noeud est une zone et chaque arête est étiquetée par un pare-feu qui contrôle le trafic entre ses deux extrémités dans une direction spécifique. Nous assumons également que chaque noeud $i \in \mathcal{N}$ est une structure ayant un ensemble d'attributs **ip, ports, prots,** etc. Chacune de ces attributs prend sa valeur dans un domaine donné.

Exemple 5.3.1 (Spécification d'un réseau)**.** *Soit le graphe $G = (\mathcal{N}, \mathcal{L}, \mathcal{A})$, avec :*
- $\mathcal{N} = \{Z_0, Z_1, Z_2, Z_3, Z_4, Z_5\}$
- $\mathcal{L} = \{F_0, F_1, F_2, F_3, F_4, F_5\}$
- $\mathcal{A}$ *la relation telle que :*
  - $\mathcal{A}(Z_0, Z_1) = \mathcal{A}(Z_1, Z_0) = F_0$
  - $\mathcal{A}(Z_1, Z_2) = \mathcal{A}(Z_2, Z_1) = F_1$
  - $\mathcal{A}(Z_0, Z_3) = \mathcal{A}(Z_3, Z_0) = F_2$
  - $\mathcal{A}(Z_3, Z_4) = \mathcal{A}(Z_4, Z_3) = F_4$
  - $\mathcal{A}(Z_2, Z_5) = \mathcal{A}(Z_5, Z_2) = F_5$
  - $\mathcal{A}(Z_4, Z_1) = F_3$

La Figure 5.2 montre une représentation graphique de G.

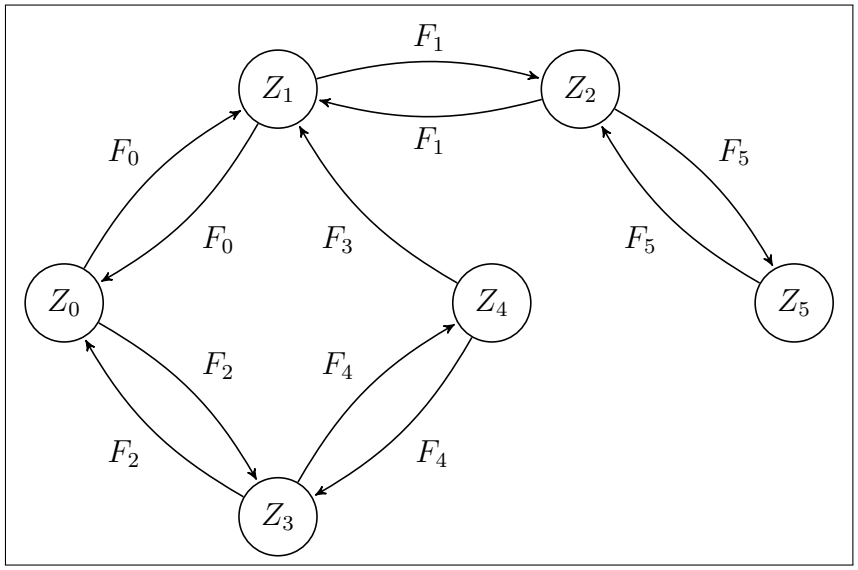


Figure 5.2 – Un exemple de graphe étiqueté

5.3.1 Spécification d'une étiquette

Pour spécifier une étiquette, qui n'est rien qu'un pare-feu, nous reprenons le language FL (*Firewall Language*), défini dans [42], auquel nous ajoutons quelques formalismes (Voir 5.3.1.1), pour accroître son expressivité.

5.3.1.1 Syntaxe de FL

La syntaxe de FL est donnée par la grammaire BNF suivante :

- Pare-feu :

$$F_1, F_2 \quad ::= \quad \varepsilon \,|\, 0 \,|\, 1 \,|\, R \,|\, \overline{F} \,|\, F_1 \cdot F_2 \,|\, F_1 \diamond F_2 \,|\, F_1 + F_2$$

  Ces définitions peuvent s'expliquer intuitivement comme suit :
  - ε est un pare-feu qui ne prend aucune décision ;
  - 0 représente un pare-feu qui rejette tous les paquets ;
  - 1 représente un pare-feu qui accepte tous les paquets ;
  - R est un pare-feu élémentaire qui contient une seule règle pouvant accepter (accept) ou rejeter (deny) une partie d'un trafic ;
  - $\overline{F}$ désigne le complémentaire d'un pare-feu F. Si F est un pare-feu, alors $\overline{F}$ est le pare-feu qui rejette ce qu'accepte F et accepte ce que rejette F.
  - $F_1 \cdot F_2$: Dans ce pare-feu, F_2 analyse uniquement le trafic pour lequel F_1 n'a pris aucune décision (accept ou deny).

- $F_1 \diamond F_2$: Dans ce pare-feu, F_2 analyse tout le trafic qui n'a pas été rejeté par F_1. Donc, contrairement à $F_1 \cdot F_2$, dans $F_1 \diamond F_2$, F_2 peut rejeter des paquets acceptés par F_1 .
- $F_1 + F_2$: Ce pare-feu accepte les paquets qui sont acceptés soit par F_1 ou par F_2 et rejette ceux qui sont rejetés par les deux pare-feu F_1 et F_2.

- Règle :

$$R ::= \varphi \rightarrow a \,|\, \varphi \rightarrow d$$

Où φ est une expression du langage L_Φ définissant les critères de filtrage. a et d sont les actions à entreprendre lorsque la condition φ est respectée.

Les différentes constructions syntaxiques de φ sont les mêmes que celles de Φ définies dans le tableau 5.1. Les actions a et d sont utilisées pour indiquer si on doit accepter (*accept*) ou refuser (*deny*) un paquet. Par exemple, la règle

$$ip\_src = * \wedge ip\_dst = 211.121.10.3 \wedge dst\_port = 80 \rightarrow d$$

est utilisée pour refuser tous les paquets à destination de l'hôte 211.121.10.3 via le port 80, quelque soit l'origine.

5.3.1.2 Sémantique dénotationnelle de FL

Un pare-feu est un filtre qui accepte certains paquets, en rejette d'autres et redirige ou délègue ceux pour lesquels il ne peut prendre aucune décision. Ainsi, lorsqu'un paquet traverse un pare-feu, trois cas de figures peuvent se présenter :

a) Le pare-feu dispose d'une règle qui accepte explicitement le paquet ;

b) Le pare-feu dispose d'une règle qui refuse explicitement le paquet ;

c) Le pare-feu délègue la décision à un autre si aucun des deux premiers cas ne s'applique.

Pour capter ces trois cas de figures, nous avons besoin de manipuler deux ensembles α et β désignant respectivement l'ensemble des paquets acceptés et l'ensemble de paquets refusés par un pare-feu. Ces deux ensembles étant disjoints et non complémentaires, le troisième n'a pas besoin d'être explicité car il est facilement déductible.
La sémantique d'un pare-feu F, notée $[\![F]\!]$, est alors définie dans le Tableau 5.3.

Tableau 5.3 – *Sémantique de FL*

$$
\begin{array}{rcl}
[\![\, \varepsilon \,]\!]_{\mathcal{P}} & = & (\emptyset, \emptyset) \\
[\![\, 0 \,]\!]_{\mathcal{P}} & = & (\emptyset, \mathcal{P}) \\
[\![\, 1 \,]\!]_{\mathcal{P}} & = & (\mathcal{P}, \emptyset) \\
[\![\, \varphi \to a \,]\!]_{\mathcal{P}} & = & ([\![\, \varphi \,]\!]_{\mathcal{P}}, \emptyset) \\
[\![\, \varphi \to d \,]\!]_{\mathcal{P}} & = & (\emptyset, [\![\, \varphi \,]\!]_{\mathcal{P}}) \\
[\![\, \overline{F} \,]\!]_{\mathcal{P}} & = & (\beta, \alpha) \ si \ [\![\, F \,]\!]_{\mathcal{P}} = (\alpha, \beta) \\
[\![\, F_1 \cdot F_2 \,]\!]_{\mathcal{P}} & = & [\![\, F_1 \,]\!]_{\mathcal{P}} \odot [\![\, F_2 \,]\!]_{\mathcal{P}} \\
[\![\, F_1 \diamond F_2 \,]\!]_{\mathcal{P}} & = & [\![\, F_1 \,]\!]_{\mathcal{P}} \otimes [\![\, F_2 \,]\!]_{\mathcal{P}} \\
[\![\, F_1 + F_2 \,]\!]_{\mathcal{P}} & = & [\![\, F_1 \,]\!]_{\mathcal{P}} \uplus [\![\, F_2 \,]\!]_{\mathcal{P}}
\end{array}
$$

avec

$$
\begin{array}{rcl}
(\alpha, \beta) \odot (\alpha', \beta') & = & (\alpha \cup (\alpha' \setminus \beta),\ \beta \cup \beta') \\
(\alpha, \beta) \otimes (\alpha', \beta') & = & (\alpha' \setminus \beta,\ \beta \cup \beta') \\
(\alpha, \beta) \uplus (\alpha', \beta') & = & (\alpha' \cup \alpha',\ \beta \cap \beta')
\end{array}
$$

5.3.1.3 Propriétés

Cette section présente quelques propriétés des pare-feu nécessaires pour une configuration sans anomalies.

Définition 5.3.2 (Équivalence)**.**
Deux pare-feu F_1 et F_2 sont équivalents, noté par $F_1 \sim F_2$, lors que

$$[\![\, F \,]\!] = [\![\, F' \,]\!].$$

Définition 5.3.3 (Infériorité)**.**
Soient F_1 et F_2 deux pare-feu. F_1 est inférieur à F_2, noté par $F_1 \sqsubseteq F_2$ lorsqu'on peut l'obtenir en éliminant une partie de F_2. Formellement,

$$F_1 \sqsubseteq F_2 \ \ si \ \ \left\{ \begin{array}{rcl} \alpha_1 & \subseteq & \alpha_2 \\ \beta_2 & \subseteq & \beta_1 \end{array} \right. \quad o\grave{u} \ (\alpha_n, \beta_n) = [\![F_n]\!], \quad n \in \{1, 2\}$$

Le Tableau 5.4 illustre quelques propriétés de cette relation.

Définition 5.3.4 (Intersection)**.** *L'intersection entre deux pare-feu F_1 et F_2, notée par $F_1 \sqcap F_2$, est le pare-feu défini comme suit :*

$$F_1 \sqcap F_2 = F \mid F \sqsubseteq F_1, F \sqsubseteq F_2 \wedge (\forall F' \mid F' \sqsubseteq F_1 \wedge F' \sqsubseteq F_2 : F' \sqsubseteq F).$$

Tableau 5.4 – *Propriétés d'un opérateur*

	1	$\sqsubseteq$	F
	F	$\sqsubseteq$	F
	$F\ op_1\ F_1$	$\sqsubseteq$	$F\ op_2\ F_2$ *si* $F_1 \sqsubseteq F_2$
	$F\ op_1\ F_1$	$\sqsubseteq$	$F'\ op_2\ F_2$ *si* $F_1 \sqsubseteq F'\ op_2\ F_2$
avec			
	$op_1 \in \{\cdot\ , \diamond\}$	et	$op_2 \in \{\cdot\ , \diamond\}$

5.3.2 Définitions et notations

Après avoir défini formellement les étiquettes du graphe, nous présentons quelques définitions et notations utiles pour comprendre leur utilisation.

Définition 5.3.5 (Chemin d'un graphe)**.**
Soit $G = (\mathcal{N}, \mathcal{L}, \mathcal{A})$ un graphe étiqueté. Un chemin dans G est une séquence finie de couples $(i_0, i_1), \cdots, (i_{n-1}, i_n)$ telle que $i_m \in \mathcal{N}$ et $(i_m, i_{m+1}) \in dom(\mathcal{A})$, $m \in [0, n[$.

L'ensemble des chemins entre deux noeuds i et j d'un graphe $G = (\mathcal{N}, \mathcal{L}, \mathcal{A})$, noté par $\Pi_G(i, j)$, peut être défini par le plus petit ensemble satisfaisant les propriétés suivantes :

- $(i, j) \in \Pi_N(i, j)$ si $(i, j) \in dom(\mathcal{A})$
- $\forall\ k \in \mathcal{V},\ \ \Pi_N(i, k) \times \Pi_N(k, j) \subseteq \Pi_N(i, j)$

Par souci de simplicité, un chemin (i, j) sera noté par $i.j$ lorsqu'il n'y a aucune ambiguïté. Ainsi, le chemin $(i, j).(j, k)$ sera simplement noté par $i.j.k$, etc.

Définition 5.3.6 (Pare-feu d'un chemin)**.**
Soient $G = (\mathcal{N}, \mathcal{L}, \mathcal{A})$ un graphe étiqueté et $(i, j) \in \mathcal{N} \times \mathcal{N}$ une paire de noeuds. Le pare-feu qui contrôle le trafic sur un chemin π de G allant de i à j, noté par $F_G(\pi)$, est défini comme suit :

- $$F_G((i, j)) = \begin{cases} \mathcal{A}(i, j) & si\ (i, j) \in dom(\mathcal{A}) \\ 0 & sinon \end{cases}$$

- $$F_G(\pi_1 \cdot \pi_2) = F_G(\pi_1) \diamond F_G(\pi_2)$$

Exemple 5.3.2 (Pare-feu d'un chemin)**.** *Si nous reprenons le graphe de l'exemple 5.2, le pare-feu du chemin $Z_0 \cdot Z_1 \cdot Z_2$ est donné par :*

$$\begin{aligned} F_G(Z_0 \cdot Z_1 \cdot Z_2) &= F_G(Z_0 \cdot Z_1) \diamond F_G(Z_1 \cdot Z_2) \\ &= \mathcal{A}(Z_0, Z_1) \diamond \mathcal{A}(Z_1, Z_2) \\ &= F_0 \diamond F_1 \end{aligned}$$

De même, le chemin $Z_0 \cdot Z_3 \cdot Z_4 \cdot Z_1 \cdot Z_2$ est contrôlé par le pare-feu $F_2 \diamond F_4 \diamond F_3 \diamond F_1$.

Nous pouvons étendre la définition $\mathcal{A}$ pour définir l'étiquette d'un chemin comme suit : $\mathcal{A}(\pi) = F_G(\pi)$. Nous étendons également la définition du pare-feu d'un chemin pour définir celui d'une paire de noeuds.

Définition 5.3.7 (Pare-feu d'une paire de noeuds)**.**
Soient $G = (\mathcal{N}, \mathcal{L}, \mathcal{A})$ un graphe étiqueté et $(i, j) \in \mathcal{N} \times \mathcal{N}$. Le pare-feu qui contrôle le trafic allant de i à j est défini par :

$$F_G(i, j) = \sum_{\pi \in \Pi_G(i,j)} F_G(\pi)$$

Exemple 5.3.3 (Pare-feu d'une paire de noeuds)**.** *Considérant toujours le graphe de l'exemple 5.2 et les résultats de l'exemple 5.3.2, le pare-feu de la paire (Z_0, Z_2) est donné par :*

$$\begin{aligned} F_G(Z_0, Z_2) &= F_G(Z_0 \cdot Z_1 \cdot Z_2) + F_G(Z_0 \cdot Z_3 \cdot Z_4 \cdot Z_1 \cdot Z_2) \\ &= F_0 \diamond F_1 + F_2 \diamond F_4 \diamond F_3 \diamond F_1 \end{aligned}$$

Ainsi, nous pouvons définir la notions d'infériorité entre deux graphes comme suit :

Définition 5.3.8 (Infériorité)**.**
Soient $G_1 = (\mathcal{N}, \mathcal{L}, \mathcal{A}_1)$ et $G_2 = (\mathcal{N}, \mathcal{L}, \mathcal{A}_2)$ deux graphes étiquetés. G_1 est inférieur à G_2, noté par $G_1 \sqsubseteq G_2$, lorsque pour toute paire $(i, j) \in \mathcal{N} \times \mathcal{N}$ on a :

$$F_{G_1}(i, j) \sqsubseteq F_{G_2}(i, j)$$

5.4 Fonction de coût

5.4.1 Notion de coût

Notre objectif principal est de définir un opérateur de renforcement ($\otimes$) qui, à partir d'un réseau R et d'une politique de sécurité Φ, génère une version de R qui respecte

Φ. Ce problème admet plusieurs solutions dont celle proposée par Mechri dans [28]. Cependant, les services et applications réseaux exigent un niveau de QoS (Qualité de service) de plus en plus élevé. Par conséquent, certaines solutions peuvent être privilégiées par l'administrateur réseau selon les qualités du service (vitesse, débit, coût, etc.) qu'elles offrent. Ainsi, nous avons besoin d'un moyen d'évaluer le coût des configurations possibles et adopter celle qui offre la meilleure performance.

Pour répondre à ce besoin, nous introduisons une notion de coût relatif aux paramètres de QoS. Ces derniers peuvent varier d'un environnement à un autre, ce qui rend difficile leur estimation par une formule exacte. Toutefois, étant donné que nous modélisons un réseau par un graphe étiqueté, nous pouvons simplement définir une fonction de coût en évaluant l'impact de placer une étiquette sur un noeud ou un arc du graphe.

5.4.2 Coût d'une configuration

Dans le cadre de ce travail, nous n'utilisons pas une fonction de coût spécifique. Nous la considérons comme une entrée de l'approche. Néanmoins, nous définissons sa signature et certaines propriétés qu'elle devrait avoir. Ainsi, nous supposons qu'il existe une fonction $\mathcal{C} : \mathcal{L} \times (\mathcal{N} \times \mathcal{N}) \rightarrow \mathbb{R}$ qui fournit le coût de placer une étiquette dans $\mathcal{L}$ sur un arc dans $\mathcal{N} \times \mathcal{N}$.

À partir du coût d'une étiquette, on calcule celui d'un chemin comme suit :

Définition 5.4.1 (Coût d'un chemin)**.**
Soit $G = (\mathcal{N}, \mathcal{L}, \mathcal{A})$ un graphe étiqueté. Le coût d'un chemin π dans G, noté par $\mathcal{C}(\pi)$, est défini par :

$$\begin{aligned} \mathcal{C}((i,j)) &= \mathcal{C}(\mathcal{A}(i,j),(i,j)) \\ \mathcal{C}(\pi_1 \cdot \pi_2) &= \mathcal{C}(\pi_1) + \mathcal{C}(\pi_2) \end{aligned}$$

Enfin, le coût d'une configuration est calculée comme indiqué ci-après.

Définition 5.4.2 (Coût d'un graphe)**.**
Soit $G = (\mathcal{N}, \mathcal{L}, \mathcal{A})$ un graphe étiqueté, le coût de G est déterminé par l'expression :

$$\mathcal{C}(G) = \sum_{(i,j) \in dom(\mathcal{A})} \mathcal{C}(\mathcal{A}((i,j)),(i,j))$$

Exemple 5.4.1. *Le coût du graphe G de la figure 5.2 est :*

$$\begin{aligned}\mathcal{C}(G) &= \mathcal{C}(F_0,(Z_0,Z_1)) + \mathcal{C}(F_0,(Z_1,Z_0)) + \mathcal{C}(F_1,(Z_1,Z_2)) + \mathcal{C}(F_1,(Z_2,Z_1)) \\ &+ \mathcal{C}(F_2,(Z_0,Z_3)) + \mathcal{C}(F_2,(Z_3,Z_0)) + \mathcal{C}(F_3,(Z_4,Z_1)) + \mathcal{C}(F_4,(Z_3,Z_4)) \\ &+ \mathcal{C}(F_4,(Z_4,Z_3)) + \mathcal{C}(F_5,(Z_2,Z_5)) + \mathcal{C}(F_5,(Z_5,Z_2))\end{aligned}$$

On peut donc utiliser cette fonction de coût pour comparer deux configurations et déterminer la plus optimale.

Définition 5.4.3 (Optimalité d'une configuration)**.**
Soient $G_1 = (\mathcal{N},\mathcal{L},\mathcal{A}_1)$ et $G_2 = (\mathcal{N},\mathcal{L},\mathcal{A}_2)$ deux graphes étiquetés. G_1 est plus optimal que G_2, noté par $G_1 \preceq G_2$, lorsque $\mathcal{C}(G_1) \leq \mathcal{C}(G_2)$.

5.5 Opérateur de renforcement

Dans cette section, nous définissons notre opérateur de renforcement. Nous commençons par déterminer la projection d'une politique sur une paire de noeuds, ensuite on définit l'opérateur en fonction de cette projection.

5.5.1 Projection d'un politique sur une paire de noeuds

La projection permet d'extraire d'une politique de sécurité la partie qui gouverne le trafic entre deux noeuds spécifiques et la transforme en un pare-feu. Ce dernier est défini en fonction des paquets rejetés afin de nous assurer qu'il est conforme à la politique. En effet, un paquet accepté par un pare-feu peut être réanalysé et rejeté par un autre. Ce qui conduit à des anomalies de configuration.

Définition 5.5.1 (Projection d'une politique sur une paire de noeuds)**.**
Soient $G = (\mathcal{N},\mathcal{L},\mathcal{A})$ un graphe étiqueté, Φ une politique de sécurité et $(i,j) \in \mathcal{N}\times\mathcal{N}$ une paire de noeuds dans G. La projection de Φ sur (i,j), notée $\Phi_{\downarrow(i,j)}$, est déterminée par une fonction $_{\downarrow} : L_\Phi \times \mathcal{N}\times\mathcal{N} \mapsto \mathcal{L}$, définie comme l'indique le Tableau 5.5.

Tableau 5.5 – *Projection d'une politique sur une paire de noeuds*

$$
\begin{array}{rcl}
\top_{\downarrow(i,j)} & = & 0 \\
(Champ\ Op\ Val)_{\downarrow(i,j)} & = & \begin{cases} \varphi \to d & \text{Si } (Champ = ip\_src \wedge [\![i.ip\ Op\ Val]\!]_B = faux) \\ & \quad \vee\ (Champ = ip\_dst \wedge [\![j.ip\ Op\ Val]\!]_B = faux) \\ & \quad \vee\ (Champ \neq ip\_src \wedge Champ \neq ip\_src) \\ \varepsilon & \text{Sinon} \end{cases} \\
\neg\Phi_{\downarrow(i,j)} & = & \overline{\Phi_{\downarrow(i,j)}} \\
(\Phi \wedge \Phi')_{\downarrow(i,j)} & = & \Phi_{\downarrow(i,j)} \diamond \Phi'_{\downarrow(i,j)}
\end{array}
$$

avec $\varphi = Champ\ \neg Op\ Val$

Exemple 5.5.1 (Projection d'une politique)**.** *Soient les politiques* Φ *et* Φ' *suivantes :* $\Phi = \neg(ip\_src = 192.168.1.3 \wedge port\_dst = 80)$ *et* $\Phi' = ip\_dst = 192.168.1.2 \wedge prot = SSL$*. Si nous considérons le graphe de l'exemple 5.2 avec* $Z_i.ip = 192.168.1.i$*,* $i \in \{0, 5\}$*, nous projetons* Φ *et* Φ' *sur* (Z_1, Z_2) *comme suit :*

- $\Phi_{\downarrow(Z_1,Z_2)} = (\neg(ip\_src = 192.168.1.3 \wedge port\_dst = 80))_{\downarrow(Z_1,Z_2)}$

  $=$ $\langle\ \neg\Phi_{\downarrow(i,j)} = \overline{\Phi_{\downarrow(i,j)}}\ \rangle$

  $\overline{(ip\_src = 192.168.1.3 \wedge port\_dst = 80)_{\downarrow(Z_1,Z_2)}}$

  $=$ $\langle\ (\Phi \wedge \Phi')_{\downarrow(i,j)} = \Phi_{\downarrow(i,j)} \diamond \Phi'_{\downarrow(i,j)}\ \rangle$

  $\overline{(ip\_src = 192.168.1.3)_{\downarrow(Z_1,Z_2)} \diamond (port\_dst = 80)_{\downarrow(Z_1,Z_2)}}$

  $=$ $\langle\ \overline{F_1 \diamond F_2} = \overline{F_1} \diamond \overline{F_2}\ \rangle$

  $\overline{(ip\_src = 192.168.1.3)_{\downarrow(Z_1,Z_2)}} \diamond \overline{(port\_dst = 80)_{\downarrow(Z_1,Z_2)}}$

  $=$ $\langle\ (Champ\ Op\ Val)_{\downarrow(i,j)}\ \rangle$

  $\overline{(ip\_src \neq 192.168.1.3 \to d)} \diamond \overline{(port\_dst \neq 80 \to d)}$

  $=$ $\langle\ Sémantique\ de\ \overline{F}\ \rangle$

  $(ip\_src = 192.168.1.3 \to d) \diamond (port\_dst = 80 \to d)$

- $\Phi_{\downarrow(Z_1,Z_2)} = (ip\_dst = 192.168.1.2 \wedge prot = SSL)_{\downarrow(Z_1,Z_2)}$

  $= \quad \langle\ (\Phi \wedge \Phi')_{\downarrow(i,j)} = \Phi_{\downarrow(i,j)} \diamond \Phi'_{\downarrow(i,j)}\ \rangle$

  $(ip\_dst = 192.168.1.2)_{\downarrow(Z_1,Z_2)} \diamond (prot = SSL)_{\downarrow(Z_1,Z_2)}$

  $= \quad \langle\ (Champ\ Op\ Val)_{\downarrow(i,j)}\ \rangle$

  $\varepsilon \diamond (prot \neq SSL \rightarrow d)$

  $= \quad \langle\ \varepsilon \diamond F \sim F\ \rangle$

  $(prot \neq SSL \rightarrow d)$

5.5.2 Définitions

Dans cette section, nous définissons l'opérateur de renforcement permettant de générer une version sécurisée du réseau initial. Mais auparavant, nous définissons le renforcement partiel d'une politique sur chemin.

Définition 5.5.2 (Renforcement partiel $\odot$)**.**
Soient F un pare-feu et π un chemin. Le renforcement de F sur π, noté par $F \odot_{\mathcal{C}} \pi$, est effectué par une fonction $\odot : \Pi \times F \times \mathcal{C} \mapsto F$, définie comme suit :

$$\begin{array}{rcl} F \odot_{\mathcal{C}} (i \cdot j) & = & \mathcal{A}(i,j) \cdot F \\ (F_1\ op\ F_2) \odot_{\mathcal{C}} \pi & = & ((F_1 \odot_{\mathcal{C}} \pi)\ op\ F_2) \odot_{\mathcal{C}} \pi \\ R \odot_{\mathcal{C}} (\pi_1 \cdot \pi_2) & = & \begin{cases} F_{\pi_1} \cdot R \diamond F_{\pi_2} & si\ \ \mathcal{C}_R(\pi_1) \leq \mathcal{C}_R(\pi_2) \\ F_{\pi_1} \diamond F_{\pi_2} \cdot R & sinon \end{cases} \end{array}$$

où $op \in \{\cdot, \diamond, +\}$ et $\mathcal{C}$ est la fonction de coût.

Définition 5.5.3 ($R \dagger [x \mapsto y]$)**.**
Soit $R : X \times Y$ une relation binaire. On note par « $R \dagger [x \mapsto y]$ » la relation définie comme suit :

$$\begin{cases} (R \dagger [x \mapsto y])(x) = y \\ (R \dagger [x \mapsto y])(z) = R(z) \quad if\ z \neq x \end{cases}$$

Définition 5.5.4 (Opérateur de renforcement $\bigotimes$)**.**
Soient $G = (\mathcal{N}, \mathcal{L}_F, \mathcal{A})$ un graphe étiqueté, Φ une politique de sécurité et $\mathcal{C}$ une fonction de coût. L'opérateur de renforcement $\bigotimes$: $G \times L_\Phi \times \Pi \to G'$ est défini par :

$$G \otimes_{\mathcal{C}} \Phi = ((R_1 \vee R_2 \vee R_3)^*)\langle G, \Phi, \mathcal{C}, \Pi_G \rangle$$

où R_1, R_2 et R_3 sont les règles définies dans le Tableau 5.6 et Π_G est l'ensemble des chemins de G.

La notation $(R_1 \vee R_2 \vee R_3)^*$ signifie qu'on applique une des règles $(R_1 \vee R_2 \vee R_3)$ selon la condition rencontrée ($\Phi_{\downarrow(i,j)} \sqsubseteq F_G(\pi)$, $F_G(\pi) \sqsubseteq \Phi_{\downarrow(i,j)}$, etc.) jusqu'à l'obtention d'un point fixe (jusqu'à ce qu'aucune règle ne s'applique).

Tableau 5.6 – *Règles de l'algorithme*

$$\mathbf{R}_1 : \quad \frac{\langle\ (\mathcal{N}, \mathcal{L}, \mathcal{A}),\ \Phi,\ \Pi \cup \{\pi\}\ \rangle}{\langle (\mathcal{N}, \mathcal{L} \cup \{\Phi_{\downarrow(i,j)}\}, \mathcal{A} \dagger [\pi \mapsto \Phi_{\downarrow(i,j)} \odot_{\mathcal{C}} \pi]),\ \Phi,\ \Pi\ \rangle} \ \Phi_{\downarrow(i,j)} \sqcap F_G(\pi) = 0$$

$$\mathbf{R}_2 : \quad \frac{\langle\ (\mathcal{N}, \mathcal{L}, \mathcal{A}),\ \Phi,\ \Pi \cup \{\pi\}\ \rangle}{\langle (\mathcal{N}, \mathcal{L} \cup \{F'\}, \mathcal{A} \dagger [\pi \mapsto F' \odot_{\mathcal{C}} \pi]),\ \Phi,\ \Pi\ \rangle} \ \exists F' \neq 0 \mid \Phi_{\downarrow(i,j)} \sqcap F_G(\pi) = F'$$

$$\mathbf{R}_3 : \quad \frac{\langle\ (\mathcal{N}, \mathcal{L}, \mathcal{A}),\ \Phi,\ \Pi \cup \{\pi\}\ \rangle}{\langle (\mathcal{N}, \mathcal{L}, \mathcal{A}),\ \Phi,\ \Pi\ \rangle} \ \Phi_{\downarrow(i,j)} \sqsubseteq F_G(\pi)$$

i et j sont respectivement les noeuds de départ et d'arrivée de π et $F' \sqcap \Phi_{\downarrow(i,j)} = 0$

Ces différentes règles de l'algorithme peuvent s'expliquer comme suit :

- R_1 renforce la politique régissant le trafic entre i et j, notée $\Phi_{\downarrow(i,j)}$, sur tous les chemins allant de i à j de façon optimale par l'entremise de la fonction de coût. Le chemin traité est ensuite supprimé de la liste initiale (on a juste Π dans le résultat). Ce qui nous donne une condition d'arrêt lorsque Π est vide. Cette règle s'applique lorsque $\Phi_{\downarrow(i,j)}$ est totalement différente de $F_G(\pi)$ (i.e. il n'y a aucune relation de supériorité de part et d'autre), donc toute la politique est à renforcer.

- R_2 effectue la même opération que R_1. Mais elle ne s'applique que lorsque la politique à renforcer est supérieure à celle du chemin ($\Phi_{\downarrow(i,j)} \sqsupseteq F_G(\pi)$).

Cette politique est donc partiellement existante d'où l'expression $\Phi_{\downarrow(i,j)} \sqcap F_G(\pi) = F'$ afin de renforcer uniquement la partie manquante, notée F'.

- Contrairement aux règles R_1 et R_2, R_3 n'effectue aucun renforcement car elle représente le cas où la politique à renforcer est incluse dans celle du chemin ($F_G(\pi) \sqsupseteq \Phi_{\downarrow(i,j)}$). La règle ne fera que supprimer le chemin en question de la liste Π pour faire évoluer l'algorithme. Ainsi, on évite les redondances ou toute autre anomalie pouvant affecter la correction ou l'optimalité de la configuration.

Pour implémenter l'opérateur ($\otimes$), nous pouvons l'écrire en pseudo-code, comme l'illustre l'algorithme 1.

Algorithm 1 Algorithme de renforcement

Entrées: un réseau $R = (\mathcal{N}, \mathcal{L}, \mathcal{A})$, une politique Φ et une fonction de coût $\mathcal{C}$.
  Pour tout $(i, j) \in \mathcal{N}$ **Faire**
    Pour tout $\pi \in \Pi_G(i, j)$ **Faire**
      Selon $F_G(\pi)$.
        Cas $F_G(\pi) \sqcap \Phi_{\downarrow(i,j)} = 0$:
          La politique à renforcer est totalement disjointe de celle du chemin π.
          On la renforce en intégralité.
        Cas $F_G(\pi) \sqcap \Phi_{\downarrow(i,j)} = F'$ avec F' différent de 0 et de $\Phi_{\downarrow(i,j)}$:
          Cette politique est partiellement incluse dans celle du chemin π.
          On renforce juste la partie manquante : F'.
        Sinon $(F_G(\pi) \sqcap \Phi_{\downarrow(i,j)} = \Phi_{\downarrow(i,j)})$:
          Cette politique est déjà incluse dans le chemin π.
          On passe au chemin suivant.
    Fin pour
  Fin pour
Sorties: R

On peut facilement voir que l'algorithme 1 est une implémentation règle par règle de l'opérateur de renforcement $\otimes$.

5.5.3 Complexité

L'ensemble des chemins Π à traiter peut être déterminé en utilisant l'algorithme proposé par Migliore *et al.* dans [43]. Cet algorithme parcourt la liste d'adjacence des noeuds du graphe et a une complexité de $n \times k \times \ell$ où :

- n désigne le nombre de noeuds du graphe ;
- k est le nombre maximal de voisins pour chaque noeud ;
- ℓ est la profondeur maximale à atteindre dans l'exploration du graphe.

D'autres algorithmes tels que celui proposé par Hura dans [44] peuvent être utilisés.
Le nombre maximal de chemins entre deux noeuds d'un graphe de n noeuds est de $(n-1)$, Si nous désignons par m le nombre d'arcs du graphe, alors dans le pire des cas, le temp d'exécution de $(R_1 \vee R_2 \vee R_3)$ est de l'ordre de $(n-1) \times m$. Donc la complexité en temps de l'algorithme est de l'ordre de $(n \times m)^2$.

5.6 Étude de cas

Dans cette section, nous exhibons un exemple complet illustrant toute l'approche.

Spécification du réseau : Le réseau à configurer est celui de la Figure 5.3.

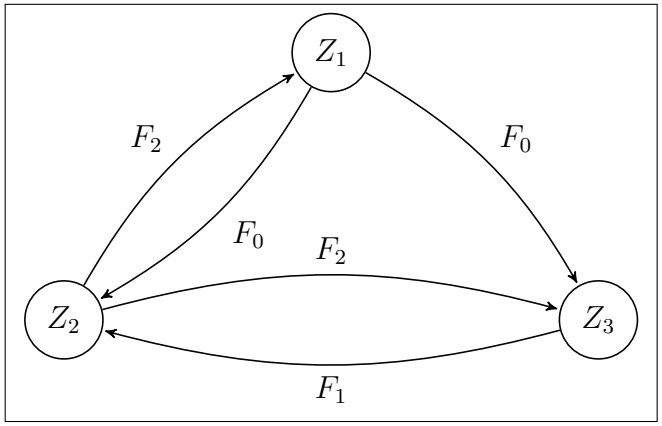


Figure 5.3 – Réseau à configurer

Ce réseau est spécifié par un graphe étiqueté $G = (\mathcal{N}, \mathcal{L}, \mathcal{A})$, avec :

- $\mathcal{N} = \{Z_1, Z_2, Z_3\}$
- $\mathcal{L} = \{F_0, F_1, F_2\}$
- $\mathcal{A}$ la relation telle que :

- $\mathcal{A}(Z_1, Z_2) = \mathcal{A}(Z_1, Z_3) = F_0$
- $\mathcal{A}(Z_2, Z_1) = \mathcal{A}(Z_2, Z_3) = F_2$
- $\mathcal{A}(Z_3, Z_2) = F_1$

Nous fixons les attributs suivants des noeuds :

- $Z_i.ports = \{21,\ 23,\ 25,\ 53,\ 80\}\ \ i \in \{1, 3\}$
- $Z_i.ip = @_i\ \ i \in \{1, 3\}$

Spécification de la politique de sécurité : La politique que nous voulons renforcer est la suivante :
« Toute communication dans G passe par la zone Z_2 et utilisent le protocole *tcp* ».
Cette politique peut être spécifiée en L_Φ par : $\Phi = (ip\_src = @_2 \wedge prot = tcp)$.

Fonction de coût : Nous supposons que le coût de placer un pare-feu F sur un arc (i, j) est proportionnel au poids de F et celui de (i, j). Nous définissons cette fonction par l'expression suivante :

$$\mathcal{C}(F, (i, j)) = \omega(F) \times (1 + w((i, j)))$$

avec $\omega(F)$ désigne la charge de F et $w(i, j)$ est le poids de (i, j).
où :

- le poids d'une règle R est égal à une constante k, $k \in \mathbb{N}$;
- la charge d'un pare-feu F composé de n est égal à $n \times k$;
- Le poids d'un arc est équivalent à la charge de son étiquette.

Pour cet exemple, nous fixons $k = 1$ et on suppose que F_0 est composé de deux règles, F_1 de trois et F_2 de cinq. D'où : $\omega(F_0) = 2,\ \ \omega(F_1) = 3$ and $\omega(F_2) = 5$.

Renforcement de la politique : Pour déterminer $G' = G \otimes_{\mathcal{C}} \Phi$, nous appliquons les règles R_1, R_2 et R_3 selon la méthode dictée par l'algorithme en commençant par la configuration initiale : $\begin{cases} \mathcal{N} &= \{Z_1, Z_2, Z_3\} \\ \mathcal{L} &= \{F_0, F_1, F_2\} \end{cases}$

Par souci de simplicité, nous ne présentons que les éléments qui ont subis des modifications. Nous supposons également que tous ces pare-feu existants sont disjoints de la projection $\Phi_{\downarrow(i,j)}$ *pour tout* (i, j) *dans* $\mathcal{N} \times \mathcal{N}$.

$\langle$ Application de $(R_1 \vee R_2 \vee R_3)^*$ $\rangle$

- Nous commençons par la paire (Z_1, Z_2),

$\Rightarrow \quad \langle$ Détermination $\Phi_{\downarrow(Z_1,Z_2)}$ $\rangle$

$$\begin{aligned}\Phi_{\downarrow(Z_1,Z_2)} &= (ip\_src \neq @_2 \wedge prot \neq tcp)_{\downarrow(Z_1,Z_2)} \\ &= (ip\_src \neq @_2)_{\downarrow(Z_1,Z_2)} \diamond (prot \neq tcp)_{\downarrow(Z_1,Z_2)} \\ &= (ip\_src \neq @_2 \rightarrow d) \diamond (prot \neq tcp \rightarrow d)\end{aligned}$$

$$\begin{cases} \Pi_G(Z_1, Z_2) &= \{(Z_1 \cdot Z_2), (Z_1 \cdot Z_3 \cdot Z_2)\} \\ \pi &= Z_1 \cdot Z_2 \\ F_G(\pi) &= F_0 \end{cases}$$

$\Rightarrow \quad \langle \Phi_{\downarrow(Z_1,Z_2)} \sqcap F_G(\pi) = 0 \Rightarrow R\grave{e}gle\ R_1 \rangle$

$\langle$ Détermination de $\Phi_{\downarrow(Z_1,Z_2)} \odot_{\mathcal{C}} \pi$ $\rangle$

L'unique pare-feu sur le chemin π est F_0, d'où $\Phi_{\downarrow(Z_1,Z_2)} \odot_{\mathcal{C}} \pi = F_0 \cdot \Phi_{\downarrow(Z_1,Z_2)}$.

$\Rightarrow \quad \langle$ Règles R_1 $\rangle$

$$\begin{cases} \mathcal{L} &= \mathcal{L} \cup \{(ip\_src \neq @_2 \rightarrow d) \diamond (prot \neq tcp \rightarrow d)\} \\ \mathcal{A} &= \mathcal{A} \dagger [(Z_1 \cdot Z_2) \mapsto F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \diamond (prot \neq tcp \rightarrow d)] \\ \Pi_G(Z_1, Z_2) &= \{(Z_1 \cdot Z_3 \cdot Z_2)\} \\ \pi &= Z_1 \cdot Z_3 \cdot Z_2 \\ F_G(\pi) &= F_0 \diamond F_1 \end{cases}$$

$\Rightarrow \quad \langle \Phi_{\downarrow(Z_1,Z_2)} \sqcap F_G(\pi) = 0 \Rightarrow R\grave{e}gle\ R_1 \rangle$

$\langle$ Détermination de $\Phi_{\downarrow(Z_1,Z_2)} \odot_{\mathcal{C}} \pi$ $\rangle$

Puisqu'il y a deux emplacements possibles, nous calculons d'abord le coût de placer un pare-feu F sur chacune des deux arcs formant le chemin π.

$$\begin{aligned}\mathcal{C}(F,(Z_1,Z_3)) &= w(F) \times (1 + w(F_0)) \\ &= k \times (1+2) \\ &= 3\end{aligned} \qquad \begin{aligned}\mathcal{C}(F,(Z_3,Z_2)) &= w(F) \times (1 + w(F_1)) \\ &= k \times (1+3) \\ &= 4\end{aligned}$$

La configuration la moins coûteuse est alors : $\Phi_{\downarrow(Z_1,Z_2)} \odot_{\mathcal{C}} \pi = F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \diamond F_1 \cdot (prot \neq tcp \rightarrow d)$

$\Rightarrow \quad \langle$ Règle R_1 $\rangle$

$$\begin{cases} \mathcal{A} & = & \mathcal{A} \dagger [(Z_1 \cdot Z_3 \cdot Z_2) \mapsto F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \diamond F_1 \cdot (prot \neq tcp \rightarrow d)] \\ \Pi_G(Z_1, Z_2) & = & \{\} \end{cases}$$

- Pour $(Z_2,\ Z_1)$, $\Phi_{\downarrow(Z_2,Z_1)} = \varepsilon \diamond (prot \neq tcp \rightarrow d) = prot \neq tcp \rightarrow d$

$$\begin{cases} \Pi_G(Z_2, Z_1) & = & \{(Z_2 \cdot Z_1)\} \\ \pi & = & Z_2 \cdot Z_1 \\ F_G(\pi) & = & F_2 \end{cases}$$

$$\Rightarrow \qquad \langle \Phi_{\downarrow(Z_1,Z_2)} \sqcap F_G(\pi) = 0 \Rightarrow R\grave{e}gle\ R_1 \rangle$$

$\langle$ Détermination de $\Phi_{\downarrow(Z_2,Z_1)} \odot_{\mathcal{C}} \pi$ $\rangle$

L'unique pare-feu sur le chemin π étant F_2, alors $\Phi_{\downarrow(Z_2,Z_1)} \odot_{\mathcal{C}} \pi = F_2 \cdot \Phi_{\downarrow(Z_2,Z_1)}$.

$$\Rightarrow \qquad \langle \text{ Règle } R_1 \rangle$$

$$\begin{cases} \mathcal{A} & = & \mathcal{A} \dagger [(Z_2 \cdot Z_1) \mapsto F_2 \cdot (prot \neq tcp \rightarrow d)] \\ \Pi_G(Z_2, Z_1) & = & \{\ \} \end{cases}$$

- Pour $(Z_1,\ Z_3)$, $\Phi_{\downarrow(Z_1,Z_3)} = (ip\_src \neq @_2 \rightarrow d) \diamond (prot \neq tcp \rightarrow d)$

$$\begin{cases} \Pi_G(Z_1, Z_3) & = & \{(Z_1 \cdot Z_2 \cdot Z_3), (Z_1 \cdot Z_3)\} \\ \pi & = & Z_1 \cdot Z_2 \cdot Z_3 \\ F_G(\pi) & = & F_0 \cdot \Phi_{\downarrow(Z_1,Z_2)} \diamond F_2 \end{cases}$$

$$\Rightarrow \qquad \langle \Phi_{\downarrow(Z_1,Z_3)} \sqsubseteq F_G(\pi) \Rightarrow R\grave{e}gle\ R_3 \rangle$$

$$\begin{cases} \Pi_G(Z_1, Z_3) & = & \{(Z_1 \cdot Z_3)\} \\ \pi & = & Z_1 \cdot Z_3 \\ F_G(\pi) & = & F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \end{cases}$$

$$\Rightarrow \quad \langle \Phi_{\downarrow(Z_1,Z_3)} \sqcap F_G(\pi) = (prot \neq tcp \rightarrow d) \Rightarrow R\grave{e}gle\ R_2 \rangle$$

$\langle$ Détermination de $(prot \neq tcp \rightarrow d) \odot_{\mathcal{C}} \pi$ $\rangle$

L'unique pare-feu sur le chemin π est $F_0 \cdot (ip\_src \neq @_2 \rightarrow d)$, donc
$(prot \neq tcp \rightarrow d) \odot_{\mathcal{C}} \pi = F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \cdot (prot \neq tcp \rightarrow d)$.

$\Rightarrow \quad \langle$ Règle R_2 $\rangle$

$$\begin{cases} \mathcal{A} & = & \mathcal{A} \dagger [(Z_1 \cdot Z_3) \mapsto F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \cdot (prot \neq tcp \rightarrow d)] \\ \Pi_G(Z_1, Z_3) & = & \{\ \} \end{cases}$$

- Pour $(Z_3,\ Z_1)$, $\Phi_{\downarrow(Z_3,Z_1)} \ = \ (ip\_src \neq @_2 \rightarrow d) \diamond (prot \neq tcp \rightarrow d)$

$$\begin{cases} \Pi_G(Z_3, Z_1) & = & \{(Z_3 \cdot Z_2 \cdot Z_1)\} \\ \pi & = & Z_3 \cdot Z_2 \cdot Z_1 \\ F_G(\pi) & = & F_1 \cdot (prot \neq tcp \rightarrow d) \diamond F_2 \cdot (prot \neq tcp \rightarrow d) \end{cases}$$

$$\Rightarrow \quad \langle \Phi_{\downarrow(Z_1,Z_3)} \sqcap F_G(\pi) = (ip\_src \neq @_2 \rightarrow d) \Rightarrow R\grave{e}gle\ R_2 \rangle$$

$\langle$ Détermination de $(ip\_src \neq @_2 \rightarrow d) \odot_{\mathcal{C}} \pi$ $\rangle$

Il y a deux emplacements possibles pour la partie manquante, nous commençons donc par calculer le coût de la placer sur chacun d'eux.

$$\begin{array}{rcl} \mathcal{C}(F, (Z_3, Z_2)) & = & \omega(F) \times (1 + \omega(F_1) + w(prot \neq tcp \rightarrow d)) \\ & = & k \times (1 + 2 + k) \\ & = & 4 \end{array}$$

$$\begin{array}{rcl} \mathcal{C}(F, (Z_2, Z_1)) & = & \omega(F) \times (1 + \omega(F_2) + w(prot \neq tcp \rightarrow d)) \\ & = & k \times (1 + 3 + k) \\ & = & 5 \end{array}$$

Par conséquent, $(ip\_src \neq @_2 \rightarrow d) \odot_{\mathcal{C}} \pi = F_1 \cdot \Phi_{\downarrow(Z_1,Z_3)} \diamond F_2 \cdot (prot \neq tcp \rightarrow d)$

$\Rightarrow \quad \langle$ Règle R_2 $\rangle$

$$\begin{cases} \mathcal{A} & = & \mathcal{A} \dagger [(Z_3 \cdot Z_2 \cdot Z_1) \mapsto F_1 \cdot \Phi_{\downarrow(Z_1,Z_3)} \diamond F_2 \cdot (prot \neq tcp \rightarrow d)] \\ \Pi_G(Z_3, Z_1) & = & \{\ \} \end{cases}$$

- Pour $(Z_2,\ Z_3)$, $\Phi_{\downarrow(Z_2,Z_3)} \ = \ \varepsilon \diamond (prot \neq tcp \rightarrow d) \ = \ (prot \neq tcp \rightarrow d)$

$$\begin{cases} \Pi_G(Z_2, Z_3) & = & \{(Z_2 \cdot Z_1 \cdot Z_3), (Z_2 \cdot Z_3)\} \\ \pi & = & Z_2 \cdot Z_1 \cdot Z_3 \\ F_G(\pi) & = & F_2 \cdot (prot \neq tcp \rightarrow d) \diamond F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \cdot (prot \neq tcp \rightarrow d) \end{cases}$$

$$\Rightarrow \quad \langle \Phi_{\downarrow(Z_1,Z_3)} \sqsubseteq F_G(\pi) \Rightarrow \textit{Règle } R_3 \rangle$$

$$\begin{cases} \Pi_G(Z_2, Z_3) & = & \{(Z_2 \cdot Z_3)\} \\ \pi & = & Z_2 \cdot Z_3 \\ F_G(\pi) & = & F_2 \end{cases}$$

$$\Rightarrow \quad \langle \Phi_{\downarrow(Z_1,Z_3)} \sqcap F_G(\pi) = 0 \Rightarrow \textit{Règle } R_1 \rangle$$

$$\langle \text{ Détermination de } \Phi_{\downarrow(Z_2,Z_3)} \odot_{\mathcal{C}} \pi \rangle$$

F_2 étant le seul pare-feu sur le chemin π, donc $\Phi_{\downarrow(Z_2,Z_3)} \odot_{\mathcal{C}} \pi = F_2 \cdot (prot \neq tcp \rightarrow d)$.

$$\Rightarrow \quad \langle \text{ Règle } R_1 \rangle$$

$$\begin{cases} \mathcal{A} & = & \mathcal{A} \dagger [(Z_2 \cdot Z_3) \mapsto F_2 \cdot (prot \neq tcp \rightarrow d)] \\ \Pi_G(Z_2, Z_3) & = & \{ \} \end{cases}$$

- Pour $(Z_3,\ Z_2)$, $\Phi_{\downarrow(Z_3,Z_2)} \quad = \quad (ip\_src \neq @_2 \rightarrow d) \diamond (prot \neq tcp \rightarrow d)$

$$\begin{cases} \Pi_G(Z_3, Z_2) & = & \{(Z_3 \cdot Z_2)\} \\ \pi & = & Z_3 \cdot Z_2 \\ F_G(\pi) & = & F_1 \cdot (prot \neq tcp \rightarrow d) \cdot (ip\_src \neq @_2 \rightarrow d) \end{cases}$$

$$\Rightarrow \quad \langle \Phi_{\downarrow(Z_1,Z_3)} \sqsubseteq F_G(\pi) \Rightarrow \textit{Règle } R_3 \rangle$$

$$\begin{cases} \mathcal{A} & = & \mathcal{A} \\ \Pi_G(Z_3, Z_2) & = & \{ \} \end{cases}$$

La configuration finale est alors :

$$\begin{cases} \mathcal{A}(Z_1, Z_2) & = & F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \diamond (prot \neq tcp \rightarrow d) \\ \mathcal{A}(Z_1, Z_3) & = & F_0 \cdot (ip\_src \neq @_2 \rightarrow d) \cdot (prot \neq tcp \rightarrow d) \\ \mathcal{A}(Z_2, Z_1) & = & F_2 \cdot (prot \neq tcp \rightarrow d) \\ \mathcal{A}(Z_2, Z_3) & = & F_2 \cdot (prot \neq tcp \rightarrow d) \\ \mathcal{A}(Z_3, Z_2) & = & F_1 \cdot (prot \neq tcp \rightarrow d) \cdot (ip\_src \neq @_2 \rightarrow d) \end{cases}$$

Cette configuration est illustrée par la figure 5.4 .

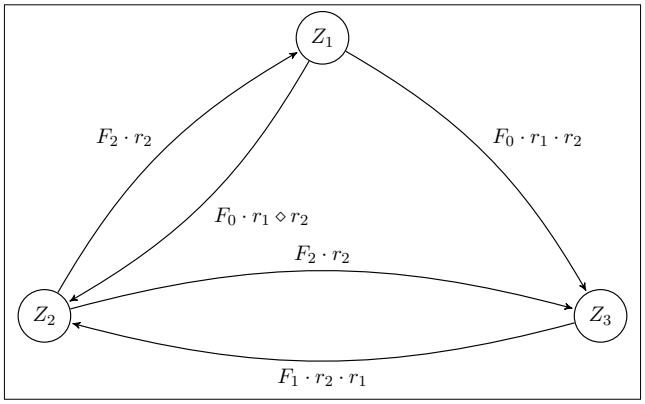


Figure 5.4 – Configuration finale

Dans cette nouvelle version, on peut voir que tous les chemins sont couverts par les politiques qui gouvernent le trafic entre les différents noeuds. Cela assure la conformité du renforcement. On peut également remarquer que la solution est optimale.

5.7 Correction de l'opérateur de renforcement

L'objectif de cette section est de définir et vérifier quelques propriétés de sécurité que la nouvelle version de notre configuration devrait respecter. Ces propriétés sont principalement la correction et la complétude. Intuitivement, la correction indique que tout ce qu'effectue un pare-feu est conforme à la politique globale de sécurité. La complétude prouve que tout ce qui est demandé dans la politique est mis en oeuvre[1] par le pare-feu. Ensemble, la correction et la complétude garantissent qu'un pare-feu fait exactement ce qui est demandé dans la politique (pas plus ni moins). Nous allons également apporter un résultat relatif à l'optimalité de notre opérateur de renforcement.

5.7.1 Définitions

Dans cette section, nous définissons plus formellement les propriétés que nous aimerions vérifier afin de satisfaire les conditions énoncées dans la section 5.1.

Définition 5.7.1 (Correction).
On dit que la configuration d'un réseau G a une configuration correcte par rapport à une politique de sécurité Φ si pour toute paire de noeuds (i, j) dans G, nous avons :

$$[\![F_G(i,j)]\!]_{\mathcal{P}_{ij}} \subseteq [\![\Phi]\!]_{\mathcal{P}_{ij}}$$

Où $\mathcal{P}_{ij}$ désigne l'ensemble des paquets ayant i comme source et j comme destination.

Définition 5.7.2 (Complétude).
On dit que la configuration d'un réseau G a une configuration correcte par rapport à une politique de sécurité Φ si pour toute paire de noeuds (i, j) dans G, nous avons :

$$[\![\Phi]\!]_{\mathcal{P}_{ij}} \subseteq [\![F_G(i,j)]\!]_{\mathcal{P}_{ij}}$$

Où $\mathcal{P}_{ij}$ désigne l'ensemble des paquets ayant i comme source et j comme destination.

5.7.2 Résultats

Cette section présente les résultats relatifs aux propriétés définies ci-haut. Pour ce faire, nous commençons par prouver la correction et la complétude de chaque étiquette.

Proposition 5.1 (Correction partielle).
Soient G un graphe étiqueté et Φ une politique de sécurité.
Si $(\mathcal{N}, \mathcal{L}, \mathcal{A}) = G \otimes_{\mathcal{C}} \Phi$ alors : $\forall\, (i,j) \in \mathcal{N} \times \mathcal{N}$ tels que $(i,j) \in dom(\mathcal{A})$, on a :

1. $[\![\mathcal{A}(i,j)]\!]_{\mathcal{P}_{ij}} \subseteq [\![\Phi]\!]_{\mathcal{P}_{ij}}$
2. $[\![\Phi]\!]_{\mathcal{P}_{ij}} \subseteq [\![\mathcal{A}(i,j)]\!]_{\mathcal{P}_{ij}}$

Preuve.

1.
$(i,j) \in dom(\mathcal{A})$
$\Rightarrow$ ⟨ Définitions 5.3.5 et 5.3.6 ⟩
$(i \cdot j) \in \Pi_G(i,j)$ et $F_G(i,j) = \mathcal{A}(i,j)$
$\Rightarrow$ ⟨ Règle $(R_1 \vee R_2 \vee R_3)$ et Définition 5.5.3 ⟩
$\mathcal{A}(i,j) \sqsubseteq \Phi_{\downarrow(i,j)} \odot_{\mathcal{C}} (i \cdot j)$
$\Rightarrow$ ⟨ Opération $F \odot_{\mathcal{C}\pi}$ (5.5.2) et définition de $F_1 \sqsubseteq F_2$ (5.3.3) ⟩
$\mathcal{A}(i,j) \sqsubseteq \Phi_{\downarrow(i,j)}$
$\Rightarrow$ ⟨ Projection d'une politique (5.5.1) ; $\Phi_{\downarrow(i,j)} \subseteq \Phi$ et Sémantique de $[\![\_]\!]$ ⟩
$[\![\mathcal{A}(i,j)]\!]_{\mathcal{P}_{ij}} \subseteq [\![\Phi]\!]_{\mathcal{P}_{ij}}$

2.

$(i,j) \in dom(\mathcal{A})$
$\Rightarrow$ ⟨ Définitions 5.3.5 et 5.3.6 ⟩
$(i \cdot j) \in \Pi_G(i,j)$ et $F_G(i,j) = \mathcal{A}(i,j)$
$\Rightarrow$ ⟨ Règle $(R_1 \vee R_2 \vee R_3)$ et Définition 5.5.3 ⟩
$\mathcal{A}(i \cdot j) = \Phi_{\downarrow(i,j)} \odot_{\mathcal{C}} (i \cdot j)$
$\Rightarrow$ ⟨ Opération $F \odot_{\mathcal{C}\pi}$ (5.5.2) ⟩
$\mathcal{A}(i \cdot j) = \Phi_{\downarrow(i,j)} \cdot \mathcal{A}(i,j)$
$\Rightarrow$ ⟨ Définition de $F_1 \sqsubseteq F_2$ (5.3.3) ⟩
$\Phi_{\downarrow(i,j)} \sqsubseteq \mathcal{A}(i,j)$
$\Rightarrow$ ⟨ Projection d'une politique (5.5.1) ; $\Phi_{\downarrow(i,j)} \subseteq \Phi$ et Sémantique de $[\![\_]\!]$ ⟩
$[\![\Phi]\!]_{\mathcal{P}_{ij}} \subseteq [\![\mathcal{A}(i,j)]\!]_{\mathcal{P}_{ij}}$

□

Cette proposition montre qu'après exécution de l'algorithme, l'étiquette $\mathcal{A}(i,j)$ renforce correctement la politique de sécurité entre i et j pour toute paire de noeuds adjacents (i,j). Ce qui est fondamental pour l'atteinte de nos objectifs. En effet, une telle paire constitue le premier chemin entre les deux noeuds (i,j), donc une des prémisses de $(R_1 \vee R_2 \vee R_3)$.

Proposition 5.2 (Correction de $\diamond$ et $+$)**.**
Soient F_i et F_j deux pare-feu, Φ une politique de sécurité et $\mathcal{P}$ un ensemble de paquets.
Si
- $[\![F_i]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}}$ *et*
- $[\![F_j]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}}$

Alors

1. $[\![F_i \diamond F_j]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}}$
2. $[\![F_i + F_j]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}}$

Preuve.

1.

$[\![F_i \diamond F_j]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}}$
$\Rightarrow$ ⟨ Sémantique de $[\![F_i \diamond F_j]\!]$ ⟩
$[\![F_j]\!] \otimes [\![F_i]\!] \subseteq [\![\Phi]\!]_{\mathcal{P}_{ij}}$
$\Rightarrow$ ⟨ Sémantique de $\otimes$ ⟩
$(\alpha([\![F_i]\!]_{\mathcal{P}}) \backslash \beta([\![F_j]\!]_{\mathcal{P}})) \subseteq \alpha([\![\Phi]\!]_{\mathcal{P}}) \ \wedge \ (\beta([\![F_i]\!]_{\mathcal{P}}) \cup \beta([\![F_j]\!]_{\mathcal{P}})) \subseteq \beta([\![\Phi]\!]_{\mathcal{P}})$
$\Rightarrow$ ⟨ Hypothèses de la proposition $[\![F_i]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}} \ \wedge \ [\![F_j]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}}$ ⟩
$Vrai.$

2.

$[\![F_i + F_j]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}}$

$\Rightarrow$ $\langle$ Sémantique de $[\![F_i + F_j]\!]$ $\rangle$

$[\![F_i]\!] \uplus [\![F_j]\!] \subseteq [\![\Phi]\!]_{\mathcal{P}_{ij}}$

$\Rightarrow$ $\langle$ Sémantique de $\uplus$ $\rangle$

$(\alpha([\![F_i]\!]_{\mathcal{P}}) \cup \alpha([\![F_j]\!]_{\mathcal{P}})) \subseteq \alpha([\![\Phi]\!]_{\mathcal{P}}) \ \wedge \ (\beta([\![F_i]\!]_{\mathcal{P}}) \cap \beta([\![F_j]\!]_{\mathcal{P}})) \subseteq \beta([\![\Phi]\!]_{\mathcal{P}})$

$\Rightarrow$ $\langle$ Hypothèses de la proposition $[\![F_i]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}} \ \wedge \ [\![F_j]\!]_{\mathcal{P}} \subseteq [\![\Phi]\!]_{\mathcal{P}}$ $\rangle$

$Vrai$.

□

Cette proposition démontre la correction des opérateurs utilisés pour définir la pare-feu d'une paire de noeuds, d'où les théorèmes 5.1 et 5.2.

Théorème 5.1 (Correction)**.**
Soient G un graphe étiqueté et Φ une politique de sécurité. Si $G' = (\mathcal{N}, \mathcal{L}, \mathcal{A}) = G \otimes_{\mathcal{C}} \Phi$ alors : $\forall\, (i,j) \in \mathcal{N} \times \mathcal{N}$, on a :

$$[\![F_{G'}(i,j)]\!]_{\mathcal{P}_{ij}} \subseteq [\![\Phi]\!]_{\mathcal{P}_{ij}} \tag{5.1}$$

Démonstration.

$G' = G \otimes_{\mathcal{C}} \Phi$

$\Rightarrow$ $\langle$ Définition de $F_G(i,j)$ (5.3.7) $\rangle$

$F_{G'}(i,j) = \sum_{\pi \in \Pi_{G'}(i,j)} F_{G'}(\pi)$

$\Rightarrow$ $\langle$ Sémantique de $[\![\_]\!]$ et Proposition 5.1 $\rangle$

$[\![\mathcal{A}(i,j)]\!]_{\mathcal{P}_{ij}} \subseteq [\![\Phi]\!]_{\mathcal{P}_{ij}}$

$\Rightarrow$ $\langle$ Définition 5.3.6 Proposition 5.2 $\rangle$

$[\![F_{G'}(i,j)]\!]_{\mathcal{P}_{ij}} \subseteq [\![\Phi]\!]_{\mathcal{P}_{ij}}$

□

Théorème 5.2 (Complétude)**.**
Soient G un graphe étiqueté et Φ une politique de sécurité. Si $G' = (\mathcal{N}, \mathcal{L}, \mathcal{A}) = G \otimes_{\mathcal{C}} \Phi$ alors : $\forall\, (i,j) \in \mathcal{N} \times \mathcal{N}$, on a :

$$[\![\Phi]\!]_{\mathcal{P}_{ij}} \subseteq [\![F_{G'}(i,j)]\!]_{\mathcal{P}_{ij}} \tag{5.2}$$

Démonstration.

$G' = G \otimes_{\mathcal{C}} \Phi$

$\Rightarrow$ $\langle$ Définition de $F_G(i,j)$ (5.3.7) $\rangle$

$F_{G'}(i,j) = \sum_{\pi \in \Pi_{G'}(i,j)} F_{G'}(\pi)$

$\Rightarrow$ $\langle$ Sémantique de $[\![\_]\!]$ et Proposition 5.1 $\rangle$

$[\![\Phi]\!]_{\mathcal{P}_{ij}} \subseteq [\![\mathcal{A}(i,j)]\!]_{\mathcal{P}_{ij}}$
$\Rightarrow$ $\langle$ Définition 5.3.6 Proposition 5.2 $\rangle$
$[\![\Phi]\!]_{\mathcal{P}_{ij}} \subseteq [\![F_{G'}(i,j)]\!]_{\mathcal{P}_{ij}}$

□

Le théorème 5.3 montre que le résultat obtenu satisfait également la relation d'infériorité.

Théorème 5.3 (Infériorité)**.**
Soient G un graphe étiqueté et Φ une politique de sécurité. $G \otimes_{\mathcal{C}} \Phi \sqsubseteq G$

Démonstration.
Soit $G' = (\mathcal{N}, \mathcal{L}, \mathcal{A}) = G \otimes_{\mathcal{C}} \Phi$
$\Rightarrow$ $\langle$ Règle $(R_1 \vee R_2 \vee R_3)$ $\rangle$
$\forall\ \pi \in \Pi_{G'}\ ,\quad F_{G'}(\pi) \sqsubseteq F_G(\pi) \odot_{\mathcal{C}} \pi$
$\Rightarrow$ $\langle$ Définition de $F_G(i,j)$ (5.3.7) $\rangle$
$\forall\ (i,j) \in \mathcal{N} \times \mathcal{N},\quad F_{G'}(i,j) \sqsubseteq F_G(i,j)$
$\Rightarrow$ $\langle$ Définition 5.3.8) $\rangle$
$G' \sqsubseteq G$

□

Enfin le théorème 5.4 énonce l'optimalité de notre opérateur de renforcement.

Théorème 5.4 (Optimalité)**.**
Soient G un graphe étiqueté et Φ une politique de sécurité.

$$(\forall\, G' \,|\, G' \models \Phi \wedge G' \sqsubseteq G, G \otimes_{\mathcal{C}} \Phi \preceq G')$$

Démonstration.
Soit $G' = G \otimes_{\mathcal{C}} \Phi$
$\Rightarrow$ $\langle$ Règle $(R_1 \vee R_2 \vee R_3)$ $\rangle$
$\forall\ \pi \in \Pi_{G'}\ ,\quad \mathcal{A}(\pi) = \Phi_{\downarrow ij} \odot_{\mathcal{C}} \pi$
$\Rightarrow$ $\langle$ Opération $F \odot_{\mathcal{C}\pi}$ (5.5.2) et Coût d'un chemin (5.4.1) $\rangle$
$\forall\ \pi \in \Pi_{G'}$, $\mathcal{C}(\pi)$ est minimal
$\Rightarrow$ $\langle$ Coût d'un graphe (5.4.2) et définition de $\preceq$ (5.4.3) $\rangle$
$(\forall\, G'' \,|\, G'' \models \Phi \wedge G'' \sqsubseteq G, G' \preceq G'')$

□

Conclusion

Dans ce chapitre, nous avons présenté une nouvelle approche formelle permettant de renforcer automatiquement une politique de sécurité sur un réseau en tenant compte des qualités de services. La nouvelle version du réseau est déterminée en utilisant un opérateur de renforcement qui prend en entrée une politique de sécurité, un réseau et une fonction de coût. La politique de sécurité est spécifiée en utilisant une logique propositionnelle simple. Le réseau est spécifié par un graphe étiqueté où chaque noeud abstrait une zone ou un simple ordinateur par un ensemble d'informations telles que les adresses IP, les ports ouverts, etc. Les étiquettes du graphe sont les pare-feu qui contrôlent le trafic entre les différents noeuds. Ces pare-feu sont spécifiés en utilisant une version étendue du langage FL présenté dans [42].
Toute l'approche est illustrée par un exemple. Nous avons également vérifié quelques propriétés de sécurité telles que la correction et la complétude de l'opérateur. Enfin, nous avons défini formellement les différentes types d'anomalies qui entravent la bonne configuration des pare-feu. Reste à définir une formulation concrète de la fonction de coût et l'utiliser pour implémenter un prototype mettant en oeuvre toute l'approche.

Chapitre 6

Conclusions et Perspectives

Résumé

Dans le présent mémoire, nous nous sommes intéressé à l'utilisation des méthodes formelles pour la sécurisation automatique des réseaux informatiques.
Un réseau informatique est un ensemble d'équipements reliés entre eux pour échanger des informations permettant ainsi la communication entre plusieurs personnes ou processus, le partage des ressources et tant d'autres services. Sécuriser un tel système consiste à assurer la confidentialité, l'intégrité et la disponibilité de ses données et ressources. De nombreux travaux ont été réalisés et plusieurs dispositifs de sécurité ont été proposés pour garantir ces propriétés. Parmi ces dispositifs les pare-feu occupent une place de premier ordre et sont des éléments cruciaux pour le renforcement d'une politique de sécurité. Ils sont largement déployés pour la sécurisation des réseaux informatiques, mais avec la forte croissance et l'hétérogénéité des réseaux actuels, leur configuration est de plus en plus complexe et souffre d'une multitude d'anomalies. Cette configuration fait l'objet de beaucoup de recherches ces dernières années. Ainsi, de nombreuses solutions ont été proposées pour pallier à ces anomalies et configurer correctement les pare-feu. Cependant, la plupart de ces solutions n'utilisent pas de méthodes formelles pour prouver une telle correction et ne prennent pas en considération la performance globale du réseau ou d'autres paramètres de qualité de services.

Dans ce mémoire nous introduisons une approche formelle pour configurer automatiquement un réseau de sorte qu'il respecte une politique de sécurité donnée tout en considérant ses paramètres de qualité de services.

La première partie de ce travail dresse un état de l'art sur la sécurité des réseaux. Nous y abordons notamment le fonctionnement des réseaux, la conception des architectures réseau, le renforcement d'une politique de sécurité ainsi que la spécification des réseaux.

Ainsi, dans le chapitre 2, nous avons défini la notion de politique de sécurité et exploré les techniques classiques de sécurisation d'un réseau telles que les pare-feu, les systèmes de detection et de prévention d'intrusions et la segmentation.
Un pare-feu permet de protéger un réseau en empêchant tout accès non autorisé. Ses fonctions sont principalement le renforcement d'une politique de sécurité et la journalisation. Un système de detection ou de prévention d'intrusions effectue une analyse, en temps réel, du trafic réseau afin de prédire toute anomalie en déclenchant une alerte ou en le stoppant dans certains cas. Enfin une segmentation physique ou logique des ressources permet d'éviter de concentrer la sécurité en un seul point et d'améliorer la performance et l'efficacité du réseau.

Dans le chapitre 3 nous étudions les deux approches les plus utilisées pour spécifier un réseau : Les structures de graphes et les algèbres de processus. Cette étude nous a permis de scinder les aspects positifs et négatifs de chacune de ces approches.

Le chapitre 4 passe en revue les différents travaux réalisés sur le renforcement d'une politique de sécurité dans un réseau. Nous y abordons particulièrement les différentes types d'anomalies qui entravent la bonne mise en oeuvre d'une politique de sécurité en utilisant un pare-feu. Nous avons défini et classifié ces anomalies et exploré quelques techniques pour les détecter. De nombreuses solutions ont été proposées durant les deux dernières décennies pour configurer, analyser et administrer un pare-feu. Ces solutions utilisent principalement trois approches qui sont l'approche formelle, la spécification par un langage de haut niveau et la rétro-ingénierie. L'approche formelle consiste à spécifier formellement le réseau et la politique à mettre en oeuvre et définir par la suite un opérateur permettant de renforcer la politique. La spécification par un langage de haut niveau quant à elle, définit un langage de haut niveau pour modéliser le réseau et la politique et concevoir un algorithme de configuration. Enfin, la rétro ingénierie vise à vérifier la correction et la conformité d'une configuration par rapport à une politique en effectuant des requêtes permettant de retrouver la politique mise en oeuvre.

Malgré ces multiples et rigoureuses solutions, nous avons constaté de nombreux manquements. Ainsi, nous prétendons qu'aucune de ces solutions n'est pleinement satisfaisante en regard des exigences actuelles. Parmi ces manquements, on cite l'absence de langages appropriés pour la spécification formelle et l'analyse des pare-feu.
Ainsi, nous avons opté pour une approche mixte en modélisant le réseau par un graphe étiqueté, la politique de sécurité par une logique propositionnelle et les pare-feu par un langage formel. Cette approche fait l'objet de la deuxième partie de ce mémoire.

Dans cette deuxième partie, nous avons défini un opérateur de renforcement qui, à partir d'un réseau et d'une politique de sécurité, génère une configuration sécurisée et optimisée du réseau initial. Pour ce faire, nous avons d'abord spécifié la politique de sécurité par une logique propositionnelle simple. Par la suite, nous avons spécifié le réseau par un graphe étiqueté où chaque noeud abstrait une zone ou un simple ordinateur par un ensemble d'informations telles que les adresses IP, les ports ouverts, etc. Les étiquettes du graphe sont les pare-feu qui contrôlent le trafic entre les différents noeuds. Ces pare-feu sont spécifiés formellement par un

langage algébrique. Nous avons ensuite introduit une notion de coût relatif à la performance d'un réseau et s'en servir pour définir un opérateur permettant de renforcer automatiquement une politique de sécurité sur un réseau en tenant compte des qualités de services. Un exemple complet illustre toute l'approche. Nous avons également prouvé la correction et la complétude de l'opérateur.

Sommaire des contributions

Dans ce mémoire nous avons apporté les contributions suivantes :

- Modélisation d'un réseau par un graphe étiqueté par des pare-feu contrôlant le trafic entre ses différentes entités.
- Extension du langage FL proposé par Mejri *and al.* dans [42], et son utilisation pour produire une configuration sécuritaire d'un réseau.
- Introduction d'une notion de coût relatif aux qualités de service d'un réseau afin de déterminer le coût d'une configuration.
- Définition de la projection d'une politique de sécurité sur une paire quelconque de noeuds d'un réseau.
- Définition d'un opérateur de renforcement permettant de générer une configuration sécuritaire et optimale d'un réseau.

Perspectives

Malgré les différentes contributions citées ci-haut, notre solution est loin d'être irréprochable. En effet, plusieurs points restent inexplorés tandis que d'autres nécessitent des améliorations ou approfondissements. Ainsi, nous proposons les perspectives suivantes :

- Étendre la logique utilisée pour exprimer les aspects temporels voire même probabilistes de certaines propriétés de sécurité.
- Définir une fonction de coût dans un environnement réelles en considérant des paramètres concrètes de qualité de service telles que le débit ou la latence (le délai de transit d'un paquet).
- Utiliser le langage FL pour spécifier formellement les anomalies des pare-feu.
- Définir des opérateurs algébriques afin d'utiliser FL pour effectuer une rétro-ingénierie.
- Implémenter un prototype mettant en oeuvre toute l'approche.

Bibliographie

[1] D. Valois C. Llorens, L. Levier. *Tableaux de bord de la sécurité réseau, 2e édition*. Eyrolles, octobre 2006.

[2] N. Stouls and M.-L. Potet. Security policy enforcement through refinement process. In *Proc. of Formal Specification and Development in B, 7th International Conference of B Users (B2007), LNCS 4355*, pages 216–231. Springer-Verlag, January 2007.

[3] B. Kenyon S. Andrés and E. P. Birkholz. *Security Sage's Guide to Hardening the Network Infrastructure*, chapter 10 and 11. Syngress Publishing, Inc., Rockland, MA, 2004.

[4] L. Mé, Z. Marrakchi, C. Michel, H. Debar, and F.Cuppens. La détection d'intrusions : les outils doivent coopérer. *evue de l'électricité et de l'électronique*, 6(5) :56–59, 2001.

[5] R. Diestel. *Graph Theory Electronic edition*. Speinger-Verlag, New York, NY, USA, 2005.

[6] S. Sorlin, P.-A. Champin, and C. Solnon. Mesurer la similarité de graphes étiquetés. In *9èmes Journées Nationales sur la résolution pratique de problèmes NP-Complets (JNPC 2003) - pages 325-339 Juin 2003*, pages 91–107. Hermes, juin 2003.

[7] S. Sorlin. *Mesurer la similarité de graphes*. PhD thesis, Université Claude Bernard, Lyon I, novembre 2006.

[8] J. C. M. Baeten. A brief history of process algebra. *Theoretical Computer Science*, 335(2-3) :131–146, 2005.

[9] J. McCarthy. A basis for a mathematical theory of computation, preliminary report. In *IRE-AIEE-ACM '61 (Western) : Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 225–238, New York, NY, USA, 1961. ACM.

[10] D.S Scott and C. Strachey. Toward a mathematical semantics for computer languages. In *Symposium on computers and automata*, pages 19–46, New York, NY, USA, 1971. Polytech. Press of Polytech. Inst. Brooklyn.

[11] R. W. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science, Proceedings of Symposia in Applied Mathematics 19*, pages 19–32, Providence, 1967. American Mathematical Society.

[12] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10) :576–580, 1969.

[13] R. Milner. A calculus of communicating systems. In *Number 92 in Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[14] C. A. R. Hoare. *Communicating Sequential Processes*. electronic version, June 21 2004.

[15] J. A. Bergstra and J. W. Klop. Process algebra for synchronous communication. *Inform. and Control*, pages 109–137, 1984.

[16] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theor. Comput. Sci.*, 37 :77–121, 1985.

[17] S. Gilmore and J. Hillston. The pepa workbench : a tool to support a process algebra-based approach to performance modelling. In *Proceedings of the 7th international conference on Computer performance evaluation : modelling techniques and tools*, pages 353–368, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[18] L. Kloul and A. Mokhtari. Algèbre des processus pour l'analyse des performances des noeuds actifs. *Technique et Science Informatiques*, 24(2-3) :279–309, 2005.

[19] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, i. *Inf. Comput.*, 100(1) :1–40, 1992.

[20] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, ii. *Inf. Comput.*, 100(1) :41–77, 1992.

[21] D. Janin and I. Walukiewicz. Automata for the modal mu-calculus and related results. In *MFCS '95 : Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science*, pages 552–562, London, UK, 1995. Springer-Verlag.

[22] H. Hermanns, U. Herzog, and J.-P. Katoen. Process algebra for performance evaluation. *Theor. Comput. Sci.*, 274(1-2) :43–87, 2002.

[23] A. Wool. A quantitative study of firewall configuration errors. *Computer*, 37(6) :62–67, 2004.

[24] E. S. Al-Shaer and H. H. Hamed. Firewall policy advisor for anomaly discovery and rule editing. In *Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on*, pages 17–30, March 2003.

[25] E. S. Al-Shaer and H. H. Hamed. Discovery of policy anomalies in distributed firewalls. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2605–2616 vol.4, March 2004.

[26] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra. FIREMAN : A toolkit for FIREwall Modeling and ANalysis. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006.

[27] E. S. Al-Shaer and H. H. Hamed. Modeling and management of firewall policies. *IEEE eTransactions on Network and Service Management*, 1(1), April 2004.

[28] T. Mechri. Approche algébrique pour la sécurisation des réseaux informatiques. Master's thesis, Faculté des sciences et de génie, Université Laval, Québec, Canada, 2007.

[29] J. D. Guttman. Filtering postures : local enforcement for global policies. *Security and Privacy, IEEE Symposium on*, 0 :120, 1997.

[30] Y.Bartal, A. J. Mayer, K. Nissim, and A. Wool. Firmato : A novel firewall management toolkit. *ACM Trans. Comput. Syst.*, 22(4) :381–420, 2004.

[31] M. G. Gouda and A. X. Liu. Firewall design : consistency, completeness and compactness. In *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems (ICDCS-04)*, pages 320–327, Tokyo, Japan, March 2004.

[32] A. X. Liu and M. G. Gouda. Firewall policy queries. *IEEE Transactions on Parallel and Distributed Systems*, 20(6) :766–777, 2009.

[33] B. Zhang, E. Al-shaer, R. Jagadeesan, J. Riely, and C. Pitcher. Specifications of a high-level conflict-free firewall policy language for multi-domain networks. In *In Proceedings of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT)*, 2007.

[34] J. R. Abrial, M. K. O. Lee, D. S. Neilson, P. N. Scharbach, and I. H. Sorensen. The b-method. In *VDM '91 Formal Software Development Methods*, pages 398–405, Berlin, Heidelberg, January 2006. Springer.

[35] A. J. Mayer, A. Wool, and E. Ziskind. Fang : A firewall analysis engine. *Security and Privacy, IEEE Symposium on*, 0 :0177, 2000.

[36] A. J. Mayer, A. Wool, and E. Ziskind. Offline firewall analysis. *International Journal of Information Security*, 5(3) :125–144, 2006.

[37] P. Verma and A. Prakash. Face : A firewall analysis and configuration engine. In *Proceedings of the 2005 Symposium on Applications and the Internet (SAINT)*, pages 74–81, Tokyo, Japan, 31 Jan-4 Feb 2005.

[38] T. Mechri, M. Langar, M. Mejri, H. Fujita, and Y. Funyu. Automatic enforcement of security in computer networks. In *Frontiers in Artificial Intelligence and Applications*, volume 161, pages 200–224, Feb 2007.

[39] A. Hall. Seven myths of formal methods. *IEEE Software*, 7 :11–19, 1990.

[40] J. P. Bowen and M. G. Hinchey. Seven more myths of formal methods. *IEEE Software*, 12 :34–41, 1995.

[41] J. P. Bowen and M. G. Hinchey. Ten commandments of formal methods. *Computer*, 28 :56–63, 1995.

[42] M. Mejri, K. Adi, and H. Fujita. Formal specification and analysis of firewalls. In *Proceeding of the 2009 conference on New Trends in Software Methodologies, Tools and Techniques*, pages 284–293, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press.

[43] M. Migliore, V. Martorana, and F. Sciortino. An algorithm to find all paths between two nodes in a graph. *J. Comput. Phys.*, 87(1) :231–236, 1990.

[44] G. S. Hura. Enumartion of all simple paths in a directed graph using petri net : A systematic approach. *Microelectronics and Reliability*, 23(1) :157–159, 1983.

[45] J. D. Guttman. Security goals : Packet trajectories and strand spaces. In *FOSAD*, pages 197–261, 2000.

[46] C. Baier and J-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[47] L. Bloch and C. Wolfhugel. *Sécurité informatique.(2e édition) : principes et méthode à l'usage des DSI, RSSI et administrateurs*. Eyrolles, 2009.

[48] CNRS. Unité réseau du CNRS . http://www.urec.cnrs.fr/accueil.php3, consulté en novembre 2009.

[49] S. Pozo, R. Ceballos, and R.M. Gasca. Model-based development of firewall rule sets : Diagnosing model inconsistencies. *Information and Software Technology*, 51(5) :894–915, 2009.

[50] A. Tanenbaum. *Réseaux, 3e édition*. DUNOD, 1996.

[51] A. El Kabbal K. Adi and M. Mejri. Un système de types pour l'analyse des pare-feux. In *4ème Conférence sur la Sécurité et Architectures Réseaux*, Batz sur Mer (France), 2005.

[52] E. S. Al-Shaer, H. H. Hamed, R. Boutaba, and M.Hasan. Conflict classification and analysis of distributed firewall policies. *IEEE Journal on Selected Areas in Communications*, 23(10) :2069–2084, October 2005.

[53] A. Recuero. Algorithms for path searching and for graph connectivity analysis. *Advances in Engineering Software*, 23 :27–35, May 1995.

[54] NetSPoC. a Network Security Policy Compiler . http://netspoc.berlios.de/.

[55] P. Mathon. *Windows Server 2003 - Les services réseaux TCP/IP*. Editions ENI, décembre 2003.

[56] R. C. Wilson and P. Zhu. A study of graph spectra for comparing graphs and trees. *Pattern Recogn.*, 41(9) :2833–2841, 2008.

[57] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Elsevier Science, New York, NY, USA, 1976.

[58] Cisco. *Architecture des réseaux et études de cas*. CampusPress France, 2000.

[59] M.-E. Triouillier G. Plouin, J. Soyer. *Sécurité des architectures web*. DUNOD, juin 2004.

[60] H. Bunke. Error correcting graph matching : On the influence of the underlying cost function. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(9) :917–922, 1999.

[61] E. W. Dijktra. The structure of the "the"-multiprogramming system. *Communications of the ACM*, 11(5) :341–346, 1968.

[62] P. Latu. Section réseau et sécurité de Linux France . http://www.linux-france.org/prj/inetdoc/, consulté de mars à novembre 2009.

[63] Securiteinfo. La sécurité informatique - la sécurité des informations . http://www.securiteinfo.com/, consulté en mars 2010.

[64] J. A. Bergstra and J. W. Klop. Real time process algebra. *Formal Aspects of Computing*, 3(2) :142–188, juin 1991.

[65] C. Stirling and D. Walker. Local model checking in the modal mu-calculus. In *TAPSOFT '89/CAAP '89 : Proceedings of the International Joint Conference on Theory and Practice of Software Development, Volume 1*, pages 369–383, London, UK, 1989. Springer-Verlag.

Annexe A

Glossaire

Dans ce qui suit, nous expliquons quelques termes techniques et acronymes anglais utilisés dans ce mémoire.

ACL **Access Control List**
Liste de contrôle d'accès permettant de configurer un pare-feu.

ACP **Algebra of Communicating Processes**
Algèbre de processus établissant un cadre de coopération des processus asynchrones via une communication synchrone.

ARP **Adress Resolution Protocol**
Protocole de recherche d'adresses MAC à partir d'adresses IP.

ATM **Asynchronous Transfer Mode**
Mode de transmission à haut débit basé sur la commutation des cellules.

BNF **Backus-Naur Form**
Notation permettant de décrire les règles syntaxiques des langages de programmation.

CCS **Calculus Of Communicating Systems**
Algèbre de processus permettant d'évaluer certaines propriétés de sécurités.

CMN **Calculus for Monitored Network**
Algèbre de processus pour décrire le comportement des processus réseaux.

CSP **Communicating Sequential Processes**
Algèbre de processus permettant de modéliser l'interaction entre des systèmes

DHCP	**Dynamic Host Configuration Protocol** Service qui permet de configurer les hôtes d'un réseau en les allouant des adresses IP, des masques, etc.
DMZ	**Demilitarized zone** Zone tampon entre un réseau privé et l'extérieur.
DNS	**Domain Name System** Service de traduction des noms de domaines en adresses IP.
Ethernet	Architecture de réseau local en bus. Elle est basée sur l'utilisation de la procédure CSMA/CD pour coorddonner l'accès des stations au médium.
FACE	**Firewall Analysis and Configuration Engine** Outil d'analyse et de configuration de pare-feu.
FDD	**Firewall decision diagram** Diagramme de décision d'un pare-feu.
FIREMAN	**FIREwall Modeling and ANalysis** Outil de modélisation et d'analyse de pare-feu.
Firmato	**Firewall Management Toolkit** Outil de gestion de pare-feu basé sur un modèle entité-relation.
FL	**Firewall Language** Langage de spécification et d'analyse des pare-feu.
FTP	**File Transfer Protocol** Protocole de transfert de fichier sur Internet.
HIDS	**Host Intrusion Detection System** IDS qui surveille une seule entité d'un réseau.
HTTP	**HyperText Transfer Protocol** Protocole utilisé sur Web pour échanger des données.
ICMP	**Internet Control Management Protocol** Protocole d'échange d'informations sur l'état du réseau Internet.
ICP	**Internet Cache Protocol** Protocole qui permet la communication entre les serveurs caches.
IDS	**Intrusion Detection System** Système de détection d'intrusion.
IP	**Internet Protocole** Protocole d'interconnexion et d'acheminement des données sur Internet.
IPS	**Intrusion Prevention System** Système de prévention d'intrusion.

LAN **Local Area Network**
Réseau privé d'entreprise dont la taille ne dépasse pas quelques kilomètres.

LSA **Link State Advertisements**
Messages contenants les états des liens. Il est utilisé dans le routage OSPF pour échanger les bases de données des liens

MAC **Medium Access Control**
Mécanisme de contrôle d'accès à un médium partagé sur un réseau local. Il désigne aussi une sous-couche de la couche liaison de données.

MAN **Metropolitan Area Network**
Grand réseau privé pouvant couvrir toute une ville.

MDL **Model Definition Language**
Langage de spécification de politique de sécurité et de topologies réseau.

NAT **Network Address Translation**
Traduction d'adresse réseau

NIDS **Network Intrusion Detection System**
IDS qui contrôle le trafic de tout un réseau.

OSI **Open Systems Interconnexion**
Standard qui définit les protocoles de communication et les interfaces d'interconnexion dans un réseau hétérogène.

OSPF **Open Shortest Path First**
Protocole de routage dynamique basé sur l'algorithme de recherche du plus court de chemin de Dijkstra

PAN **Personal Area Network**
Réseau personne constitué d'une simple interconnexion d'équipements personnels.

PEPA **Performance Evaluation Process Algebra**
extension de l'algèbre de processus classique en ajoutant un aspect temporel aux actions.

QoS **Quality of Service**
Qualité de service.

RARP **Reverse Address Resolution Protocol**
Protocole de recherche d'adresse MAC à partir d'une adresse IP.

RIP **Routing Information Protocol**
Protocole de routage basé sur les distances entres les routeurs.

SFQL **Structured Firewall Query Language**
Langage de spécification des requêtes dans un pare-feu.

SMTP	**Simple Mail Transfer Protocol** Protocole d'échange de courrier électronique.
SNMP	**Simple Network Management Protocol** Protocole de gestion de réseau sur Internet. Il permet de gérer les noeuds d'un réseau à distance
TCP	**Transmission Control Protocol** Protocole de transport d'Internet. Il est orienté connexion et offre un service de transmission fiable.
Telnet	Protocole d'émulation de terminaux sur Internet. Il permet la connexion à distance à travers un terminal virtuel.
Token Ring	**Anneau à jeton**. Architecture de réseau local en anneau. Basé sur l'utilisation d'un jeton pour coordonner l'accès des stations au médium.
UDP	**User Datagram Protocol** Protocole de transport d'Internet. Non orienté connexion et offre un service de transmission non fiable.
VLAN	**Virtual Local Area Network** Réseau local regroupant un ensemble de machines de façon logique et non physique
VPN	**Virtual Private Network** Réseau privé virtuel.
WAN	**Wide Area Network** Réseau étendu couvrant une zone géographique importante.