République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira - Béjaïa

Faculté des sciences exactes

Département d'informatique

Mémoire de fin d'étude

Pour l'obtention du Diplôme de Master en Informatique

Option : Réseaux et Systèmes Distribués

Thème

Approche algébrique pour la prévention d'intrusions (attaque Blackhole)

Présenté par :

ABDOUNE Mohamed Oulhadj

HAMIDOUCHE Ahcène

Devant le jury composé de :

Président : M^{me} BATTAT Nadia

Examinateur : M^{me} OUYAHIA Samira

Examinateur : M^{me} YAHIAOUI Soraya

Encadreur : M^{me} HAMZA Lamia

Année universitaire 2013-2014

Remerciements

Nos vifs remerciements vont d'embler à dieu tout puissant de nous avoir donné le courage, la force, et le savoir nécessaire pour mener à bien ce modeste travail.

Nous tenons à remercier notre encadreur M^{me} HAMZA pour nous avoir fait profiter de sa grande experience, pour sa disponibilité, sa patience, son suivi et ces précieux conseils.

Nous remercions tout particulièrement les membres de jury, qui ont accepté de juger notre travail.

Nous ne pourrons clôturer ces remerciements sans nous retourner vers les êtres qui nos sont les plus chers, qui ont eu un rôle essentiel et continu pendant notre réussite. Nous adressons de tout cœur nos remerciements à nos familles pour leur préoccupation et le souci qu'ils se sont fait pour nous, leur encouragement et leur suivi. Enfin, nous remercions tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

Dédicaces

Nos dédicaces vont à nos familles

Nos frères et soeurs, nos amis

Et à tous nos chèrs.

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Les systèmes informatiques et les réseaux sont devenus des outils indispensables pour la société actuelle. Ils sont aujourd'hui déployés dans tous les secteurs professionnels. Ce développement phénoménal est accompagné également par la croissance du nombre d'utilisateurs, qui ne sont pas forcément pleins de bonnes intentions vis-à-vis de ces systèmes informatiques, en exploitant ses vulnérabilités.

Cependant, l'un des grands problèmes de ces systèmes est la sécurité, les travaux de recherche indiquent que les réseaux sans fil sont plus vulnérables que les réseaux filaires en raison de leurs caractéristiques tels que le milieu ouvert, la topologie dynamique, l'absence d'administration centrale, la coopération distribuée, et la capacité restreinte (en termes de puissance et de calcul). L'utilisation de liaisons sans fil rend ces réseaux plus sujets à des menaces de sécurité physiques que les réseaux câblés, allant de l'écoute passive à l'interférence active. Sans aucune sécurité adéquate, les hôtes mobiles sont facilement capturés, compromis et détournés par des nœuds malveillants. L'adversaire peut écouter et modifier les messages dans le canal de communication, injecter des messages erronés, supprimer des messages, et même passer par d'autres nœuds. Par conséquent, les mécanismes de sécurité dans de tels réseaux sont essentiels pour protéger les données émises par les utilisateurs.

Ces problèmes de sécurité ne cessent de troubler davantage les réseaux AD HOC, ce qui fait appel à établir des règles et des méthodes de vérification et de restrictions qui assurent la sécurité et la sureté dans ces réseaux, cette procédure constitue ce qu'on appelle une politique de sécurité. Des travaux plus récents tentent de sécuriser

un réseau AD HOC selon des méthodes formelles utilisant des calculs mathématiques et des preuves de validation formelles. Parmi ces calculs mathématiques, on retrouve : le CCS (Calculus of Communicating System), le Π-calcul, et le calcul ambient. Le but de ce travail est de modéliser une attaque de déni de service (blackhole) avec un modèle amélioré et inspiré du calcul ambient et de déterminer une solution appropriée à cette attaque. Ce mémoire est organisé en quatre chapitres, chaque chapitre aborde des points spécifiques. Il est structuré comme suit : Le premier chapitre présente des concepts de base sur la sécurité des réseaux AD HOC. Dans le deuxième chapitre, nous avons dressé un état de l'art sur le renforcement des politiques de sécurité, où on a analysé quelques travaux antérieurs. Nous avons ensuite modélisé notre proposition dans le troisième chapitre et on l'a réalisé dans le quatrième. Enfin nous terminons le mémoire par une conclusion générale dans laquelle nous résumons notre travail.

Chapitre 1

Concepts de base sur la sécurité des réseaux AD HOC

1.1 Introduction

De nos jours, la possibilité d'accéder rapidement et facilement aux réseaux informatiques est devenue une nécessité. Les solutions sans fil actuelles offrent ces services, mais ces réseaux sans fil sont beaucoup plus vulnérables aux menaces externes, il est donc nécessaire de prévoir une architecture de gestion, et une meilleure politique de sécurité pour sécuriser ces réseaux, et les rendre plus résistants aux attaques. Dans ce chapitre, nous allons présenter quelques concepts de base sur les réseau sans-fil et exposer quelques attaques connues sur ce type de réseau.

1.2 Les réseaux sans-fil

Un réseau sans-fil ou un réseau Wi-Fi (Wireless Fidelity), est un réseau dans lequel on relie différentes machines (ordinateurs, tablettes, téléphones. . .) entre elles sans aucun intermédiaire physique. La transmission des données se fait via les ondes hertziennes. Ceci permet aux utilisateurs de se déplacer dans un périmètre de couverture pouvant aller d'une dizaine de mètres à quelques kilomètres [1].

1.2.1 Les architectures des réseaux sans fil

Les architectures les plus courantes dans les réseaux sans fil sont les architectures avec infrastructure et les architectures sans infrastructure appelées aussi architectures ad hoc.

a- Mode avec infrastructure

Ce mode de fonctionnement est très semblable au protocole Ethernet des réseaux filaires. Les machines se connectent à un point d'accès AP (*AccesPoint*), appelé aussi station de base, qui partage la bande passante disponible. Les stations de base sont munies d'une interface de communication sans fil avec les sites mobiles qui se trouvent dans sa zone géographique ou sa couverture radio [2]. Ce mode de

fonctionnement est illustré dans la figure 1.1 :

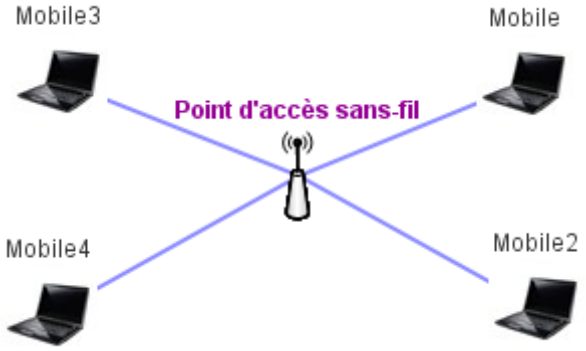

FIGURE 1.1: Réseau sans fil avec infrastructure

b- Mode sans infrastructure

Ce mode n'a pas besoin de point d'accès pour fonctionner, ce sont les stations elles-mêmes qui entrent en communication sans s'appuyer sur un équipement extérieur. Tous les nœuds d'un réseau de ce type se comportent comme des routeurs et prennent part à la découverte et à la maintenance des chemins de communication entre les différentes machines. Ce type de réseau s'organise lui-même [2]. La figure 1.2 montre le mode sans infrastructure.

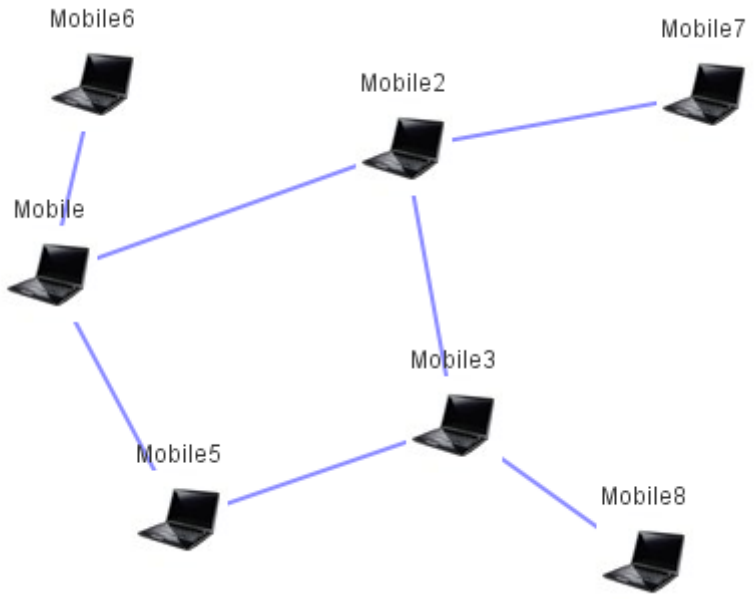

FIGURE 1.2: Réseau sans fil sans infrastructure

1.3 La sécurité dans les réseaux ad hoc

La sécurité d'un système informatique a pour objectif la protection des informations contre toutes divulgations, altération ou destruction.

1.3.1 Objectifs

Quelle que soit la nature du réseau, sa politique de sécurité vise à satisfaire les propriétés suivantes [3] :

- **Intégrité** : L'intégrité peut être vue comme un ensemble de propriétés garantissant la protection des données contre les modifications et altérations non autorisées. On peut distinguer les altérations accidentelles dues à l'environnement dur de communication, par exemple une mauvaise couverture des ondes, et les altérations volontaires d'un attaquant. Cela concerne aussi la protection contre l'injection ou la modification des paquets.

- **La confidentialité** : La confidentialité des données est une exigence importante dans la sécurité du réseau. Elle permet d'assurer qu'une communication de données reste privée entre un émetteur et un destinataire. La cryptographie ou le chiffrement des données est la seule solution fiable pour sécuriser le transfert des données.
- **La disponibilité** : La disponibilité est un service réseau qui donne une assurance aux entités autorisées d'accéder aux ressources réseaux avec une qualité de service adéquate.
- **L'authentification** : L'authentification peut être définie comme le processus de prouver une identité revendiquée. La confidentialité, l'intégrité des données, et la non-répudiation dépendent tous de l'authentification appropriée. Un système sans cette fonctionnalité ne peut pas fournir les objectifs de sécurité mentionnés de manière satisfaisante.
- **La non-répudiation** : Mécanisme permettant de garantir qu'un message a bien été envoyé par un émetteur et reçu par un destinataire, c'est-à-dire aucun des correspondants ne pourra nier l'envoi ou la réception du message.

1.3.2 Menaces

Les systèmes informatiques sont tout le temps confrontés à diverses menaces pouvant porter atteinte aux composants matériels, logiciels ou informationnels. On distingue principalement deux types de menaces [4] :

a- Menaces non-intentionnelles (ou mauvais comportement)

Ne supposant aucune préméditation, par exemple : les bugs logiciels, les pannes matérielles, les sinistres (incendies, séismes, vols, ...) ou également d'un utilisateur du système : l'énorme majorité des problèmes liés à la sécurité d'un système d'information a pour origine un utilisateur, généralement insouciant. Il n'a pas le désir de porter atteinte à l'intégrité du système sur lequel il travaille, mais son comportement

favorise le danger.

b- Menaces intentionnelles

Reposent sur l'intervention d'une tiers personne désirant nuire à la sécurité du système. Ces menaces intentionnelles (attaques) peuvent se produire de différentes manières, la classification de ces menaces dépend de plusieurs paramètres :

- **Menace passive / active** : Une attaque passive obtient les données échangées dans le réseau sans perturber le fonctionnement de la communication, tandis qu'une attaque active implique l'interruption d'information, la modification, ou la fabrication, ce qui perturbe le fonctionnement normal du réseau sans fil [4].
- **Menace interne/ externe** : Les attaques peuvent aussi être classées en deux catégories, à savoir les attaques externes et les attaques internes, selon le domaine de l'attaque. Les attaques externes sont effectuées par des noeuds qui n'appartiennent pas au domaine du réseau. Les attaques internes sont entreprises par des noeuds compromis, qui font partie du réseau. Les attaques internes sont plus graves par rapport aux attaques externes car l'attaquant connaît des informations précieuses et secrètes, et possède un accès privilégié au réseau [4].
- **Menace individuelle/ distribuée** : Les attaques peuvent enfin être classées en attaques individuelles ou attaques distribuées. Les attaques individuelles sont simples et elles sont issues d'une seule source et par un chemin simple sans utiliser des stations intermédiaires. Par contre, une attaque distribuée est une attaque évoluée invoquant plusieurs stations ou provenant de plusieurs sources. Les attaques distribuées sont plus dangereuses et difficiles à détecter puisqu'ils utilisent plusieurs stations intermédiaires, ce qui a pour effet la difficulté de déterminer la source d'une telle attaque [4].

La menace peut parvenir de :

- **Une Personne Malveillante** : une personne parvient à s'introduire sur le système, légitimement ou non, et à accéder ensuite à des données ou à des programmes auxquels elle n'est pas censée avoir accès. Le cas fréquent est de passer par des logiciels utilisés au sein du système, mais mal sécurisés.
- **Un programme malveillant** : un logiciel destiné à nuire ou à abuser des ressources du système est installé (par mégarde ou par malveillance) sur le système, ouvrant la porte à des intrusions ou modifiant les données ; des données confidentielles peuvent être collectées à l'insu de l'utilisateur et être réutilisées à des fins malveillantes.

1.3.3 Attaques existantes

Ci-dessous quelques attaques connues :

- **Man in the middle** : L'attaquant peut personnifier le récepteur pour l'expéditeur, et vice versa, sans que l'un ou l'autre se rendent compte qu'ils ont été attaqués. De cette façon, l'attaquant se positionne entre le récepteur et l'émetteur et en conséquence, il peut mener facilement son attaque dans le réseau [5].
- **Spoofing attack** : C'est une attaque qui consiste à usurper l'identité d'un autre nœud pour envoyer des messages à sa place. Dans le cas des protocoles de routage réactifs, elles consistent souvent à répondre avant que la vraie destination ne le fasse et avec un nombre de saut plus court. Les conséquences possibles sont : dégradation des communications, et nœuds injoignables [6].
- **Attaque sybil** : L'attaquant pratique ici une usurpation des identités de plusieurs nœuds, autrement dit, l'attaquant ne se contente pas d'usurper une identité mais se fait plutôt passer pour plusieurs nœuds différents. Dès lors, il est possible pour l'attaquant d'intercepter l'ensemble du trafic même s'il est réparti sur plusieurs routes (l'attaquant aura juste à être présent sur l'ensemble des routes) [7].

- **Wormhole** : Dans l'attaque Wormhole, deux nœuds malicieux coopèrent pour lancer une telle attaque. Le premier nœud malicieux reçoit les messages dans une zone dans le réseau et les envoie, via un tunnel, à son coopérant (deuxième nœud malicieux) qui se trouve dans une autre zone dans le réseau. La conséquence d'une telle attaque peut être la falsification de l'information du voisinage [8].
- **Eavesdropping** : L'eavesdropping consiste à écouter les messages circulant sur le réseau. Le but classique est souvent de récupérer différentes informations non chiffrées (par exemple dans le cas de protocoles non sécurisés) ou encore d'obtenir un volume de données chiffrées assez important pour effectuer des analyses afin de retrouver les clés de chiffrement. Cette technique est notamment employée pour casser le chiffrement WEP ($WiredEquivalentPrivacy$), lequel est généralement utilisé pour chiffrer les connexions WiFi.
- **Denial Of Service – DoS** : Une attaque par déni de service ($denialofserviceattack$) est une attaque qui vise à limiter ou stopper complètement le bon fonctionnement d'une ressource réseau, afin qu'elle ne puisse plus ou mal fournir son service. Il existe de nombreuses formes ou méthodes pour perpétrer une attaque DOS, la forme la plus commune est d'inonder le réseau par des paquets invalides, ainsi, empêcher le trafic légitime du réseau, Une autre méthode consiste à noyer la victime dans un calcul fastidieux de sorte qu'elle soit trop occupée pour répondre à toutes les requêtes [9]. Quand plusieurs noeuds sont impliqués dans l'attaque, on parle alors d'une attaque DDOS ($DistributedDenialOfService$), dans laquelle l'attaquant exploite un grand nombre d'hôtes infectés par des virus, chevaux de Troie ou vers, il peut alors contrôler à distance ces machines (qualifiés de zombies ou des esclaves) et diriger l'attaque à tout autre hôte ou réseau [10]. Dans le déni de service on retrouve plusieurs attaques comme le traffic jamming, blackhole, etc.
- **Traffic Jamming** : Le jamming est une attaque très connue qui s'en prend à

la communication sans fil. En effet, vu la sensibilité du média sans fil au bruit, un noeud peut provoquer un déni de service en émettant des signaux à une certaine fréquence pour interférer avec les fréquences radio employées par les noeuds du réseau [11].

- **Blackhole** : L'attaque de blackhole signifie qu'un noeud malveillant utilise le protocole de cheminement, prétendant ainsi avoir le plus court chemin vers le noeud destinaire. Mais ce noeud malveillant n'achemine pas les paquets à ses voisins, il les détruit ou les éxpédie vers une autre destination, ce qui fait en sorte l'existence d'un trou noir dans le réseau. Une attaque simple de trou noir est facilement produite dans les réseaux ad hoc mobiles [12]. Un exemple est illustré dans la figure 1.3. Le noeud 1 se tient comme le noeud source et le noeud 4 comme le noeud destinataire. Le noeud 3 est un noeud malveillant qui répond au message $RREQ$(Route-Request) envoyé du noeud source avec $RREP$(route-replay), et fait une fausse réponse prétendant avoir l'itinéraire le plus rapide au noeud destinaire. Par conséquent le noeud 1 juge incorrectement le procédé de découverte d'itinéraire avec l'accomplissement, et commence à envoyer des paquets de données au noeud 3. Comme il est mentionné ci-dessus, un noeud malveillant détruit ou consomme les paquets. Ce noeud malicieux peut être considéré comme un problème de trou noir dans le réseau. En conséquence, le noeud 3 peut dérouter les paquets facilement.

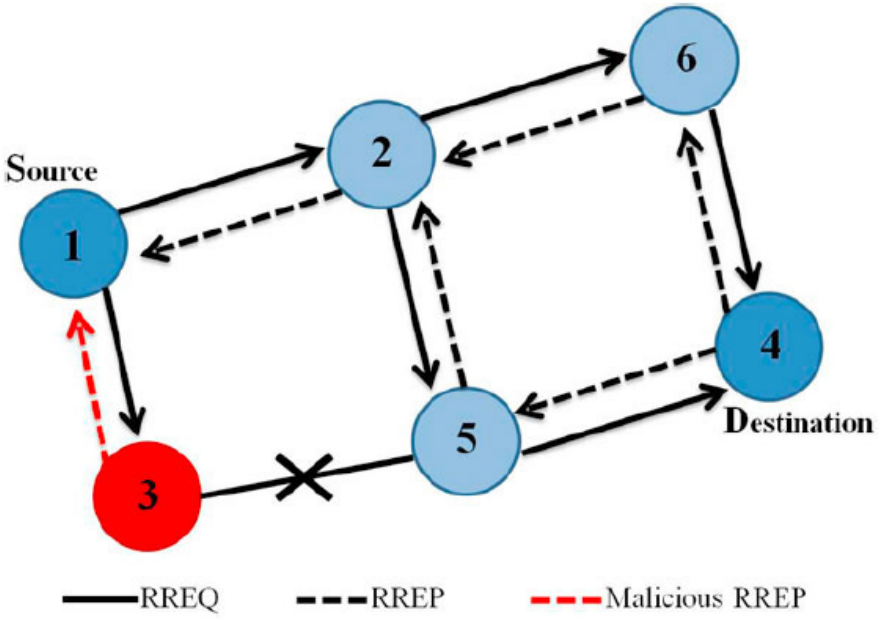

FIGURE 1.3: Attaque Blackhole sur un réseau AD HOC

1.4 Conclusion

Dans ce premier chapitre, nous avons présenté le réseaux sans-fil, et ses modes d'architecture. Nous avons aussi présenté une vue générale sur la sécurité informatique et les attaques menaçant cette dernière, ce chapitre n'a pas la prétention de faire une liste exhaustive des attaques mais de couvrir les avenues possible. La sécurité des réseaux sans-fil est vitale à son bon fonctionnement. Il est donc nécessaire d'assurer sa protection, nous allons décrire dans ce qui suit les techniques utilisées pour renforcer une politique de sécurité.

Chapitre 2

Renforcement des politiques de sécurité

2.1 Introduction

Les politiques de sécurité ont été toujours associées avec les systèmes informatiques, de manière plus ou moins évidente. Elles assurent la sécurité et la sureté des systèmes, et contrôlent l'accès à certaines informations en établissant des règles et des méthodes de vérification et de restrictions.

Les restrictions peuvent être imposées sur des ressources physiques (imprimante, ports USB, etc.) ou logiques (des fichiers, utilisation de processeur, etc.). Elles peuvent aussi se référer aux manipulations des données et différents niveaux d'accès. Vu la diversité des systèmes et les besoins, c'est difficile de définir une procédure ou une façon universelle de restriction sous la forme de renforcement de politique de sécurité. Ce chapitre présente un état de l'art sur le renforcement de la politique de sécurité.

2.2 Définition de politique de sécurité

La division des comportements possibles d'un système en comportements acceptables et inacceptables constitue ce qu'on appelle une politique de sécurité. Autrement dit, une politique de sécurité définie un ensemble de contraintes qui doivent être respectées par les exécutions effectuées par un système [13].

Une récente définition plus pratique des politiques de sécurité est donnée par Schneider [14] : Une politique de sécurité définit une exécution qui, pour une raison ou une autre, a été jugé inacceptable. Par exemple, une politique de sécurité pourrait concerner :

- Contrôle d'accès : c'est l'ensemble des moyens qui garantissent que seules les entités autorisées peuvent accéder aux ressources d'un système informatique, et seulement d'une manière autorisée.
- Flot de l'information : restreindre ce que l'utilisateur peut insinuer à partir de l'observation du comportement d'un système donné.

– Disponibilité : elle assure l'accessibilité du système d'information en temps voulu et de la manière requise aux personnes autorisées. Ainsi cette politique détermine les règles qui permettent de prévenir les attaques de dénis de service.

2.3 Historique

Le terme de politique de sécurité a été introduit en 1982 par Goguen et Meseguer [15] comme une réponse à la nécessité de restreindre les droits de lire, d'ajouter, de modifier ou de supprimer des informations dans des contextes spécifiques. Leur approche innovante pour définir et vérifier la sécurité concerne les politiques de sécurité statiques et dynamiques, leur approche a déclaré des flux d'information qui doivent être refusée. Le système d'information est modélisé comme un automate appelé un système de capacité. Les auteurs ont également préconisé l'utilité d'un langage de définition de la politique de sécurité fondée sur de simples assertions de $non - interférence$ [1]. Un croquis de modèles de vérification possibles est également présenté. Leur court article a une densité et une vision étonnante qui était à peine né à l'époque.

2.4 Technique de vérification

La simple énonciation du contenu d'une politique est inutile en l'absence d'un mécanisme d'exécution, souvent appelé « technique de vérification ». Une technique de vérification est un mécanisme de renforcement de certaine politique de sécurité dans un système donné, quel que soit un programme, machine, ou bien un réseau. Le renforcement peut être appliqué à différents niveaux de l'infrastructure informatique, et peut soutenir des politiques allant d'une propriété unique de sécurité (authentification, confidentialité, intégrité) à un ensemble complet de contraintes du

1. La non-interférence est un modèle strict de politique de sécurité à multi-niveaux, décrit pour la première fois par Goguen et Meseguer en 1982, et amplifié ensuite en 1984

système. L'objectif est de s'assurer que le système se comporte conformément à la politique de sécurité. Ce qui veut dire qu'aucun comportement violant la politique ne peut être effectué par le système renforcé.

Le renforcement de politique de sécurité peut s'appliquer sur les programmes ainsi que sur les systèmes, peu de travaux ont été réalisés sur le renforcement de politique de sécurité sur les réseaux. Dans ce qui suit, nous allons présenter d'abord quelques travaux liés au renforcement des programmes, ensuite, nous allons terminer par ceux liés aux réseaux.

2.4.1 Renforcement des politiques de sécurité sur les programmes

Il existe trois principales classes de technique de vérification sur la sécurité des programmes, à savoir l'analyse statique, l'analyse dynamique ou monitorage, et la réécriture de programmes, comme le montre la figure 2.1.

a- Politiques de sécurité statiquement renforçables

Les politiques de sécurité qui peuvent être statiquement renforcées sont appelées $statiquement - v\acute{e}rifiables$. La vérification statique est une technique qui permet d'analyser un programme avant de l'exécuter. Alors le résultat de l'analyse soit l'acceptation ou le rejet du programme, le programme sera accepté si toutes ses exécutions possibles satisfont la politique de sécurité. Ainsi il suffit de trouver une seule exécution qui viole la politique et ça sera le rejet de tout le programme. Par exemple, la vérification de type effectuée par la machine virtuelle Java est une technique d'analyse statique qui rejette les programmes Java mal typé [13].

b- Politiques de sécurité renforçables par monitorage

Les politiques de sécurité qui peuvent être renforcées par un moniteur d'exécution sont intitulées $EM - renforçables$. Un moniteur d'exécution EM (Execution

Monitoring) est un mécanisme de sécurité qui s'exécute en même temps que le programme surveillé. Il intervient en lançant une procédure d'intervention lorsque la cible est au point d'exécuter une action qui engendre la violation de la politique de sécurité. La procédure d'intervention consiste généralement à arrêter l'exécution de la cible. Les moniteurs d'exécution qui appliquent cette méthode sont appelés « moniteurs d'exécution conventionnels » CEM (Conventional Execution Monitoring). Toutefois, les EMs peuvent avoir des procédures d'intervention plus puissantes qui leur permettent d'insérer des actions dans le programme surveillé ou de supprimer les actions dangereuses de la cible [16]. Les EMs ayant cette force sont appelés « moniteurs d'exécution basé sur la réécriture » RWEM (Rewritable Execution Monitoring).

Schneider [14] a initié la recherche sur la classe des politiques de sécurité runtime renforçable. Il a caractérisé précisément les mécanismes de renforcement de classe de l'EM et il l'a précisé avec des automates de sécurité. Un automate de sécurité est défini par :

- Un ensemble dénombrable Q des états automates.
- Un ensemble dénombrable $Q_0 \in Q$ états initiaux d'automate.
- Un ensemble dénombrable I des symboles d'entrée.
- Une fonction de transition $\delta : (Q \times I) \rightarrow 2^Q$.

Un automate de sécurité est utilisé pour la modélisation d'une politique particulière. L'automate comporte deux sections ; l'une pour la déclaration de variable d'état et l'autre pour la définition des transitions permises. Les renforcements de la politique de sécurité dans les EM sont accomplis par des simulations d'automates de sécurité, qui sont exécutées en tandem avec la cible. Les simulations sont alimentées avec des symboles d'entrée générés par chaque étape que la cible est sur le point de prendre. L'état de l'automate change s'il peut faire la transition sur les symboles d'entrée, sinon la simulation se termine et la cible est interrompue. En ce qui concerne l'application de la politique de sécurité, l'automate de sécurité accorde

à la cible le droit d'effectuer l'étape si la transition est possible, ou bien elle nie l'exécution en raison d'une violation de la politique.

Le renforcement des politiques par le biais d'automates jouit d'une popularité parmi beaucoup d'autres recherches. Bauer, Ligatti et Walker [18, 16, 19] émissent une série d'études sur le sujet. Leur technique d'application repose sur les automates de sécurité travaillant comme moniteurs de programmes qui examinent et ; si nécessaire ; transforment la séquence des actions du programme. Les automates respectent une certaine classe de la politique de sécurité de Schneider discutée en [20] en insérant ou supprimant les actions. Un nouveau langage et un système qui prend en charge le développement de « facile à maintenir », les politiques de sécurité d'exécution complexes pour des applications Java est également introduites. Les politiques sont des objets de première classe et se composent de deux types de méthodes : l'une pour les actions sensibles à la sécurité et l'autre pour les mises à jour d'état. Les systèmes complexes peuvent être modélisés par des politiques Meta paramétrées [19] ou des politiques d'ordre supérieur [16]. Une grande bibliothèque de politique combinatoire est fournie, ce qui démontre l'utilisation des fonctions de programmation qui permettent la composition des politiques. Une sémantique formelle a été définie pour un sous-ensemble du langage composé des principales caractéristiques seulement.

c- Politiques de sécurité renforçables par réécriture de programmes

Les politiques de sécurité qui peuvent être renforcées par réécriture de programmes sont intitulées $RW - renforçables$. Ce mécanisme se base sur la réécriture du programme surveillé. Afin de renforcer une politique, cette technique n'a pas besoin de vérifier si la cible satisfait ou non la politique de sécurité, le programme est automatiquement réécrit pour générer une nouvelle version qui satisfait la politique de sécurité. Mais le nouveau programme doit être équivalent à l'original excepté les exécutions qui violent la politique de sécurité. Cela peut être fait, par exemple,

par injection d'automates, comme indiqué par Ould-Slimane [21] ou par transformation de formules et de procédés, comme l'a démontré Langar et Al en [22, 23, 24]. Schneider et Al ont [14] ont défini une nouvelle catégorie de politiques qui peuvent être renforcées par le programme de réécriture. Dans ce contexte, le programme de réécriture se réfère à un mécanisme de renforcement qui, en un temps fini, modifie un programme non approuvé avant l'exécution. La classe RW permet à certains politiques qui ne sont pas renforçables par le suivi du programme. L'utilisation de réécriture de programme pour renforcer les politiques de sécurité a été initiée en fin des années 60. Mais elle a été reprise dans des études plus récentes sur l'isolement de faille basé sur le software (SFI ; Software-based Fault Isolation). Schneider et Erlingsson [17] ont développé SASI (Security Automata SFI Implementation). SASI renforce les politiques de sécurité en modifiant le code objet pour un système cible avant que le système soit exécuté. Cette approche a été le prototype pour deux architectures : x86 d'Intel et JVM de Sun (Java Virtual Machine). Les prototypes ont exploré l'utilisation d'une approche SFI-comme pour les politiques EM-renforçables. La figure 2.1 présente les différentes techniques de renforcement des programmes qui sont mentionnées en dessus ; l'analyse statique, par monitorage, et par la réécriture de programmes.

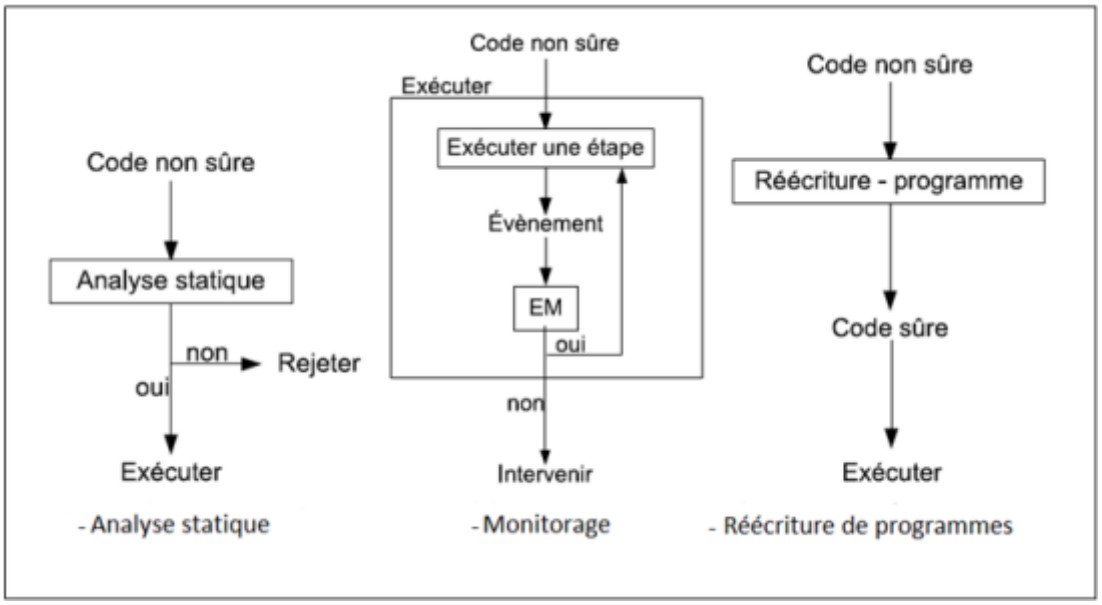

FIGURE 2.1: Les différentes techniques de renforcement des programmes

Caractérisation des politiques de sécurité selon les classes de calculabilité

Hamlen et al.[25] ont proposé une caractérisation des politiques de sécurité selon les classes de calculabilité. Ce travail se base sur la classification des propriétés renforçables en considérant les programmes comme une *machine de Turing*[2] déterministe, appelée (PM), manipulant trois rubans infinis :

- Un ruban d'entrée contenant les informations d'entrée du programme.
- Un ruban de travail modélisant l'espace de travail fourni au programme durant son exécution. Il est à noter que cet espace n'est pas accessible au mécanisme de renforcement.
- Un ruban de trace stockant les événements de sécurité exécutés par PM. Le contenu de ce ruban est accessible au mécanisme de renforcement. Ce ruban permet principalement de distinguer entre les actions observables de l'exécution et celles qui ne le sont pas.

2. machine de Turing est un modèle abstrait du fonctionnement des appareils mécaniques de calcul créé par Alan Turing.

En utilisant ces trois types de rubans, le modèle proposé permet de comprendre les raisons derrières l'impossibilité pour un mécanisme de renforcement de renforcer un certain type de politiques de sécurité, à savoir :

- Le mécanisme de renforcement ne peut pas observer les événements critiques pertinents relatifs à la politique renforcée. Il est à noter que dans ce cas, indépendamment du mécanisme de renforcement utilisé, l'ensemble des événements observables est inadéquats pour le renforcement de la politique en question.
- Le mécanisme de renforcement ne possède pas assez de puissance de calcul pour prévenir la violation de la politique de sécurité. A partir de cette constatation, les auteurs ont conclu que lorsqu'un mécanisme de renforcement échoue, il se peut que d'autres réussissent.

Par ailleurs, selon ce modèle formel, Hamlen et al.[25] se sont attardé sur la classification des trois mécanismes de renforcements (Figure 2.2) :

- **Renforcement statique** : Un mécanisme renforçant statiquement une politique de sécurité P est modélisé par une machine de Turing (PM) M_P qui prend en entrée une PM M représentant le programme renforcé. Si M satisfait P, alors M_P accepte M dans un temps fini, sinon M_P rejette M dans un temps fini. Intuitivement, il s'agit de l'ensemble des politiques de sécurité pour lesquelles il existe une machine de Turing qui, dans un temps fini, accepte un programme donné si ce dernier satisfait la politique de sécurité ou le rejette s'il viole la politique. Cette définition correspond exactement à la définition de la classe des propriétés décidables.
- **Monitorage** : Un CEM renforçant une politique de sécurité P est caractérisé par une PM M_P qui prend en entrée une PM M représentant le programme renforcé. Si M ne satisfait pas P, alors M_P rejette M dans un temps fini, sinon M_P boucle infiniment. Cette caractérisation correspond à la définition de la classe de propriétés co-récursivement énumérables (co-RE), c'est-à-dire que le CEM rejette le programme dans un temps fini s'il viole la politique de

sécurité ou boucle infiniment. Il est à noter que puisque co-RE est un sur-ensemble des propriétés décidables, toute propriété renforçable statiquement est aussi renforçable dynamiquement. En effet, Hamlen et al.[25] supposent que les moniteurs peuvent effectuer une analyse statique après avoir chargé le programme en mémoire avant que ce dernier commence à s'exécuter.

- **Réécriture de programmes** : Un mécanisme renforçant une politique de sécurité P par réécriture est modélisé par une relation de réécriture, liant les programmes originaux à leurs alternatifs réécrits. $R : PM \rightarrow PM$ tel que :

$$P(R(M)) \ (RW1)$$
$$P(M) \Rightarrow M \cong R(M) \ (RW2)$$

Où $\cong$ est une relation d'équivalence entre les machines représentant les programmes. Ainsi, une politique de sécurité P est renforçable par réécriture de programmes s'il existe une relation R permettant de réécrire tout programme $PM\ M$ en un programme $R(M)$ qui satisfait la politique de sécurité $(RW1)$ et si le programme original $PM\ M$ satisfait déjà la politique de sécurité, alors le programme modifié $R(M)$ lui est équivalent $(RW2)$. Hamlen et al. [25] ont démontré que la classe de politiques de sécurité renforçables par réécritures de programmes ne correspond à aucune classe de la hiérarchie arithmétique.

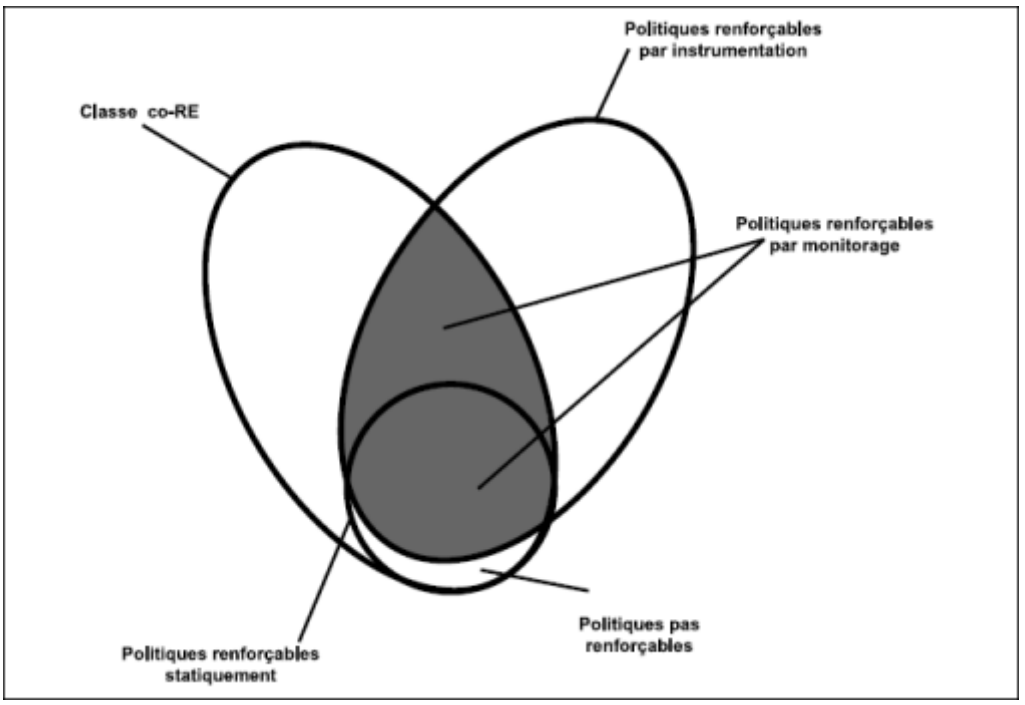

FIGURE 2.2: La taxonomie de politiques de sécurité

2.4.2 Renforcement des politiques sur les réseaux informatiques

Lacasse [27] a défini une nouvelle algèbre de processus basée sur CCS. Cette algèbre qui est destinée à la sécurisation formelle des réseaux, est munie d'un opérateur de surveillance qui permet de contrôler les entrées et les sorties d'un processus, ou d'un sous-processus, à l'image d'un pare-feu dans un réseau informatique. L'algèbre permet donc de modéliser des réseaux munis de moniteurs, et également, n'importe quel système communicant devant être contrôlé par des moniteurs.

L. Pene et K.Adi [33] ont présenté une nouvelle approche pour spécifier et vérifier les politiques de sécurité distribuées mises en application dans les pare-feu distribués. Ils ont démontré par une étude de cas comment leur nouveau calcul qui est inspiré du calcul ambient peut être employé pour aborder le problème de la détection de conflit dans les pare-feu distribués. Le but de leur contribution est d'identifier comment les

paquets légitimes peuvent être arrêtés d'atteindre leur destination et les paquets potentiellement nocifs peuvent passer.

Nous constatons que Lacasse[27] et L.pene [33] ont focalisé leur travaux sur les politiques de sécurité mises en application sur les pare-feu hors que les politiques de sécurités peuvent être appliquées sur des domaines plus vastes.

T.Mechri [24] a proposé une technique formelle permettant de configurer automatiquement un réseau donné afin qu'il respecte une politique de sécurité donnée. En d'autres termes, étant donné un réseau informatique N, et une politique de sécurité Φ, il a introduit une technique formelle qui produit automatiquement un autre réseau N' tel que $N' \vDash \Phi$, N, et N' se comportent d'une façon équivalente (par rapport à une définition d'équivalence donnée). On effet, il a défini une nouvelle algèbre de processus permettant de mieux préciser et analyser un réseau surveillé. Il a défini aussi un opérateur $\otimes$ qui produit à partir d'un réseau initial N et une politique de sécurité Φ une autre version du réseau, notée $N \otimes \Phi$, configurée d'une manière que la politique de sécurité soit toujours respectée.

Un autre système de renforcement de politique de sécurité a été présenté par Gloria et Pugliese [28]. Leur système traite des questions liées à la mobilité de code dans les systèmes distribués, tels que le commerce électronique et les services bancaires en ligne. Le type est expressément conçu pour le processus de calcul $\mu - Klaim$ [29], un calcul de processus distribués et mobiles. Il permet l'attribution de différents privilèges aux utilisateurs sur différentes ressources, qui imite le comportement d'un système réel. Cela rend le système pratique, comme en témoigne leur modèle de système de gestion de compte bancaire.

Martins [30] a introduit un système de type différent applicable aux réseaux complexes qui ont besoin d'une meilleure sécurité. Il a utilisé Dπ [31, 32] pour contrôler la migration de code entre les nœuds de grands systèmes distribués. La topologie du réseau se reflète par des politiques de sécurité partagés dans les sites web. Le résultat est un système qui est libre de toute violation des règles lors de

l'exécution.

2.5 Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur le renforcement de la politique de sécurité sur les programmes et sur les réseaux, dans lequel nous avons expliqué les différentes approches pour remédier aux problèmes dues aux intrusions sur les réseaux. A l'issue de cette étude bibliographique, nous pouvons conclure qu'une sécurité parfaite est loin d'être évidente et de nouveaux mécanismes de sécurité doivent être mise en œuvre pour la renforcer. Dans le chapitre suivant, nous allons présenter une étude d'une algèbre de processus pour la spécification des systèmes informatiques.

Chapitre 3

Modélisation algébrique d'une attaque blackhole

3.1 Introduction

Aujourd'hui, la tâche de la détection des problèmes dans les systèmes informatiques est devenue de plus en plus coûteuse et délicate à cause de la complexité augmentée de ces derniers, en effet l'utilisation des méthodes formelles est incontournable pour la spécification et l'analyse de tels systèmes dans le but d'augmenter leur fiabilité. Pour cela, plusieurs langages formels ont été développés dans le but de décrire mathématiquement et sans ambiguïté des systèmes informatiques.

Le non acheminement du trafic de contrôle ou de données par un nœud intermédiaire constitue une déviation comportementale dont la conséquence est la violation de l'objectif pour lequel le réseau est déployé. Un tel comportement malicieux s'appelle l'attaque de suppression des paquets (attaque blackhole). Grossièrement, dans ce chapitre, nous allons donner un bref aperçu sur le calcul ambient, ensuite, nous allons définir l'approche RSC ainsi que sa syntaxe et sa sémantique. Par la suite, nous allons mettre l'accent sur l'attaque blackhole et présenter une solution proposée en appliquant l'approche RSC sur un réseau AD HOC, en spécifiant un système vulnérable à l'attaque blackhole.

3.2 Calcul et algèbre de processus

Ces vingt dernières années, une dizaine d'algèbres de processus ont été proposées, la majorité d'entre elles considèrent un système complexe et cela en fonction des actions élémentaires qu'il peut exécuter.

Une algèbre est définie par un ensemble d'éléments de base ainsi qu'un ensemble d'opérateurs permettant de construire des composantes complexes à partir des éléments basics. Parmi les algèbres de processus les plus connues dans le domaine de la spécification et la vérification des systèmes concurrents et les réseaux informatiques, nous trouvons : le CCS [34] et le π-Calcul [35] développées par Milner, le CSP [36] proposé par Hoare, l'ACP [37] introduite par Brookes et Roscoe et le calcul ambient

[38] développé par Cardelli et Gordon.

3.3 Le Calcul Ambient

Le calcul ambient a été développé par Cardelli et Gordon [39] au cours des années 90. L'objectif de leurs travaux était d'offrir un calcul qui permettrait d'exprimer tous les aspects de la mobilité, en visant principalement les agents mobiles et le code mobile. Un ambient est un espace (un réseau, une machine, une page web, etc.) borné ayant un intérieur et un extérieur, dans lequel peuvent se dérouler des calculs, il possède un nom unique non modifiable. Ce nom est utilisé pour contrôler les accès à l'ambient, A l'intérieur d'un ambient, on peut trouver des processus concurrents, ou bien une collection d'ambients. Ainsi chaque ambient possède une collection d'agents locaux, ces derniers, plus simple que les ambients, en définissent le comportement, ils peuvent par exemple permettre à l'ambient de pénétrer à l'intérieur d'un autre ambient. Nous décrivons maintenant la syntaxe du calcul ambient et sa sémantique opérationnelle.

3.3.1 Syntaxe du calcul ambient

La syntaxe du calcul ambient est comme indiquée dans le tableau 3.1.

- Le processus "0" représente le processus inactif.
- La restriction de n à P, notée(vn)P, et la composition parallèle notée par le symbole "|".
- La notation n [P] dénote un processus P exécuté dans un ambient nommé n.
- L'opérateur de réplication " ! " permet d'obtenir autant de duplications parallèles souhaitées d'un processus donné. Le processus !P est donc équivalent à P| !P.
- Le symbole M.P représente un processus qui commence par exécuter la capacité M puis il continue comme P.

	(a) Processus		
P	:: =	0	Inactivité
	|	(vn)P	Restriction
	|	P|P'	Composition parallèle
	|	!P	Réplication
	|	n[p]	Ambient
	|	M.P	Action
	|	⟨M⟩	Sortie d'une capacité
	|	(x).P	Entrée d'une capacité
	(b) Capacités		
	|	in N	Entrer dans N
	|	out N	Sortir de N
	|	open N	Ouvrir N
	|	ε	Capacité nulle
	|	M.M'	Chemin d'accès
	(c) Noms		
	|	n	Nom
	|	≪n≫	Sortie d'un nom
	|	((n)).P	Entrée d'un nom

TABLE 3.1: Syntaxe du calcul ambient

- Les symboles ⟨M⟩ et (x)P expriment une communication qui permet de passer d'une capacité M d'un processus émetteur ⟨M⟩ à un processus récepteur (x)P parallèle et voisin (localisé dans le même ambient).
- La capacité ε est la capacité vide.
- La capacité M.M' signifie la concaténation de deux capacités M et M'.

Les capacités *in* N, *out* N et *open* N : *in* N fait entrer l'ambient conteneur dans un ambient parallèle N, *out* N fait sortir l'ambient conteneur de son ambient parent N, tandis que *open* N élimine les frontières de l'ambient fils N.

3.3.2 Sémantique opérationnelle du calcul ambient

La sémantique opérationnelle du calcul ambient est définie via deux relations ; la congruence entre processus $\equiv$ et des règles de réduction $\rightarrow$ comme le montre le tableau 3.2 et le tableau 3.3 respectivement.

Dans le calcul ambient, l'opérateur $\equiv$ représente un processus ayant la même structure interne, les règles régissant ces évolutions sont appelées règles de congruence structurelle. Quant à l'opérateur $\rightarrow$ il est utilisé pour marquer l'évolution d'un processus en un autre processus, les règles régissant ces évolutions sont appelées règles de réduction. Ainsi, lorsque l'on note P $\rightarrow$Q, cela indique que le processus P évolue pour en devenir le processus Q. La relation de congruence est définie par les axiomes et les règles du tableau 3.2. Ils assurent que la relation de congruence est une relation d'équivalence, que la composition parallèle est commutative et associative avec le processus inactif, ils décrivent aussi le comportement de la réplication.

$P \equiv P$	$P \mid Q \equiv Q \mid P$
$P \equiv Q \Rightarrow Q \equiv P$	$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$
$P \equiv Q, Q \equiv R \Rightarrow P \equiv R$	$!\, P \equiv P \mid !\, P$
$P \equiv Q \Rightarrow (vn)Q \equiv (vn)P$	$P \mid 0 \equiv P$
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	$(vn).0 \equiv 0$
$P \equiv Q \Rightarrow !\, P \equiv !\, Q$	$!\, 0 \equiv 0$
$P \equiv Q \Rightarrow n[Q] \equiv n[P]$	$\varepsilon.P \equiv P$
$P \equiv Q \Rightarrow (x).Q \equiv (x).P$	$(M.M').P \equiv M.M'.P$
$P \equiv Q \Rightarrow M[Q] \equiv M[P]$	$P \equiv Q \Rightarrow M.P \equiv M.Q$

TABLE 3.2: Règles de congruence structurelle du calcul ambient

La relation de réduction est définie par les axiomes et les règles du tableau 3.3, ces règles illustrent d'une part l'emploi des capacités, et spécifient d'autre part que la congruence structurelle peut être utilisée pour réarranger les expressions des processus ambients, les restrictions, les processus ambients ainsi que les compositions parallèles.

n[in m.P | Q] | m[R] → m[n[P | Q] | R]	P ≡ P',P' → Q',Q' ≡ Q ⇒ P ≡ Q
m[n[out m.P | Q] | R → n[P | Q] | m[R]	P → Q ⇒ (vn)(P) → (vn)(Q)
open n.P | n[Q] → P | Q	P → Q ⇒ n[P] → n[Q]
⟨M⟩ | (x).P → P{M →x}	P → Q ⇒ P | R → Q | R

TABLE 3.3: Règles de réduction du calcul ambient

3.4 Approche RSC (Renforcement Security Calculus)

Dans cette section, nous allons présenter la syntaxe et la sémantique du calcul de [40, 41], ce dernier est adapté pour décrire l'interaction avec un intrus potentiel, les auteurs modélisent les composants du système en termes de processus et interactions de processus.

3.4.1 Syntaxe

La syntaxe de l'approche RSC définit les modules requis pour un renforcement de sécurité dans un système informatique via des clefs de sécurité et elle est présentée dans le tableau 3.4. Soit N l'ensemble des noms, K l'ensemble des clefs, et V l'ensemble des variables. Dans cette approche chaque processus est emboité à l'intérieur d'un ambient qui est protégé via deux clefs de sécurités k et k' ce qui dénote l'accès contrôlé aux ressources. Les clefs sont saisies comme clefs d'entrée et de sortie (e et s, respectivement). Par conséquent, des processus doivent connaitre la clef appropriée afin d'entrer dans un ambient. Les ambients nouvellement créés contiennent par défaut deux clefs de même valeur δ connue par tous les autres processus ; le symbole δ définit une clef publique. Dans la suite nous dénotons P l'ensemble de tous les processus qui peuvent être exprimés par le calcul RSC [40, 41]. L'opérateur de concaténation "." décrit le comportement séquentiel d'un processus. Le terme ACTION est employé pour décrire l'exécution des capacités de processus, et ci-dessous

nous présentons l'ensemble P :

- **Inactivité** : 0 est un processus qui n'exécute rien.
- **Composition parallèle** :P | Q se rapporte à l'exécution parallèle des processus P et Q.
- **Réplication** : !P est un processus qui dénote la réplication illimitée du processus P. c'est un opérateur commutatif et associatif.
- **Ambient protégé** : ${}^{k,k'}_{n}[\mathrm{P}]$ dénote un ambient n qui contient une ressource P protégée par deux clefs k, et k'.
- **Action** : a.P présente le comportement séquentiel d'un processus comme séquence : il emploie d'abord la capacité « a » puis se comporte comme P.

Les auteurs ont combiné les notions d'ambient protégé et d'action dans la séquence d'ambient, ce qui dénote le fait qu'une action peut se produire à l'intérieur d'un ambient protégé. Les sous-catégories suivantes se rapportent à des capacités de processus :

- **Demande de clef** : $req^{x}_{n,t}$ permet à un processus de faire une demande de clef d'accès k ou de sortie k', selon la valeur du paramètre t.
- **Publication de clef** : $pub^{x}_{n,t}$ se rapporte à la publication de clefs.
- **Mouvement** : mov^{k}_{n} se rapporte aux capacités d'un processus pour se déplacer en utilisant la clef d'ambient appropriée.
- **Exploration** : exp permet la présentation des processus dynamiques qui peuvent choisir d'une manière non-déterministe ce qu'est leur prochaine action.
- **Protection** : $prot^{k,k'}_{n}$ permet de renforcer la protection des ressources à l'intérieur d'un ambient n avec les clefs k et k'.
- **Unprotection** : $unprot^{k,k'}_{n}$ permet d'enlever la protection de l'ambient n et de la rendre par défaut δ, δ.
- **Choix compositionnel** : a $\oplus$ b permet à l'évolution d'un processus d'être définie comme choix entre toute combinaison possible des actions a et b
- **Choix non déterministe** : a $\sqcap$ b permet à l'évolution d'un processus d'être

définie comme choix entre deux processus composants, mais ne permet pas à l'environnement n'importe qu'elle contrôle sur des processus composants.

Il existe deux types de mouvement selon le paramètre k : mouvement d'accès et mouvement de sortie. Le mouvement d'accès permet à un processus d'entrer dans un ambient n protégé par la clef d'accès k. Le mouvement de sortie permet à un processus de sortir d'un ambient n protégé par la clef de sortie k_0. La représentation séquentielle des processus (actions suivies d'autres actions) exprime le fait que la toute première action doit se produire avant que la prochaine soit exécutée. Un ordre prédéfini des actions prêtera au processus un comportement prévisible. Ces processus sont considérés comme des processus réguliers. Cependant, il n y'a pas des processus qui n'ont pas un comportement prévisible. L'opérateur *exp* est utilisé pour imiter le comportement dynamique d'un intrus. Les divers aspects du comportement de processus seront présentés par la suite.

n	$\in$	N	Nom
x	$\in$	V	Variable
k, k'	$\in$	K$\cup\{\delta\}$	Clefs de sécurité
t	$\in$	e,s	types de clefs
P,Q	::=		Processus
	|	0	Inactivité
	|	P | Q	Composition parallèle
	|	!P	Réplication
	|	$^{k,k'}_{n}$[a.P]	Séquence d'ambient
a,b	::=		Capacités de processus
	|	$req^{x}_{n,t}$	Demande de clefs
	|	$pub^{x}_{n,t}$	Publication de clefs
	|	mov^{k}_{n}	Mouvement d'un processus
	|	exp	Exploration
	|	$prot^{k,k'}_{n}$	Protection
	|	$unprot^{\delta,\delta}_{n}$	Unprotection
	|	a $\oplus$ b	Choix compositionnel
	|	a $\sqcap$ b	Choix non-déterministe

TABLE 3.4: La syntaxe de l'approche RSC

3.4.2 Sémantique

Les deux composants de la sémantique opérationnelle du calcul sont : la congruence structurelle notée par $\equiv$ et la relation de réduction notée par $\rightarrow$. Le tableau 3.5 détaille la congruence structurelle en tant qu'équivalence de processus dans le contexte du calcul. La congruence séquencielle d'ambient est une extension de la séquence d'équivalence de processus aux processus protégés. Le zéro parallélisme énonce l'élément neutre du calcul. Les itérations sont dénotées par la réplication, qui peut être employée pour modéliser un service (tel que la publication de clefs) ou une tentative répétée d'accéder à un domaine protégé. Le choix compositionnel décrit le comportement potentiel de l'intrus : soit a et b deux capacités, a$\bigoplus$b signifie que le

processus peut continuer soit comme une action a ou comme b, ou en tant que séquence de deux capacités a.b ou b.a. Le choix d'exécution illustre comment le choix non-déterministe des actions est effectué pour l'exécution de processus. La relation de réduction définie dans le tableau 3.6 capture le raccordement entre le comportement des processus, les capacités des processus et la protection d'ambient. Les règles (1) et (2) sont triviales. La règle (3) décrit le fait qu'un processus peut entrer dans un ambient protégé. La capacité de sortir d'un ambient protégé est présentée par la règle (4). Les règles (5) et (6) dénotent le niveau de la protection offert par la clef publique δ, les tentatives d'accéder ou de sortir d'un ambient protégé par δ comme clef d'accès ou de sortie, respectivement, sont réussies indépendamment de la clef utilisée.

$P \equiv P$	Réflexivité
$P \equiv Q \Rightarrow Q \equiv P$	Symétrie
$P \equiv Q \bigwedge Q \equiv R \Rightarrow P \equiv R$	Transitivité
$P \equiv Q \Rightarrow P \mid R \equiv Q \mid R$	Parallélisme
$P \equiv Q \Rightarrow {}^{k,k'}_{n}[a.P] \equiv {}^{k,k'}_{n}[a.Q]$	Séquence d'ambient
$P \equiv 0 \equiv P$	Zéro parallélisme
$(P \mid Q) \mid R \equiv P \mid (Q \mid R)$	Associativité
$P \mid Q \equiv Q \mid P$	Commutativité
$!P \equiv P \mid !P$	Réplication parallèle
$P \equiv Q \Rightarrow !P \equiv !Q$	Réplication
$!0 \equiv 0$	Zéro réplication
$a \oplus b \equiv a \sqcap b \sqcap a.b \sqcap b.a$	Choix compositionnel
$a \oplus b \equiv b \oplus a$	Commutativité choix compositionnel
$(a \oplus b) \oplus c \equiv a \oplus (b \oplus c)$	Associativité choix compositionnel
$a \sqcap b \equiv b \sqcap a$	Commutativité chois non déterministe
$(a \sqcap b) \sqcap c \equiv a \sqcap (b \sqcap c)$	Associativité choix non déterministe
$(a \sqcap b) .P \equiv a.P \sqcap b.P$	Choix d'exécution

TABLE 3.5: Les règles de congruence structurelle de l'approche RSC

Le symbole (-) désigne n'importe quelle valeur de la clef à employer par les capacités mov. Une demande de la clef appropriée du service approprié de publication a comme conséquence une communication de la valeur de la clef, règle (7). La règle (8) décrit le renforcement des clefs de sécurités. L'action $prot_n^{k,k'}$ peut seulement être exécutée sur des ambients protégés avec une paire de clefs publiques. Afin de changer les clefs, l'action $unprot_n^{k,k'}$ est nécessaire, comme illustré par la règle (9). La paire de clefs indiquées par le $unprot_n^{k,k'}$ devrait assortir la paire réelle de protection de l'ambient. Le résultat d'un *unprot* réussi est un ambient protégé avec une paire de clefs publiques. La règle (10) décrit le comportement de l'intrus. Elle modélise la manière dont le processus d'intrus peut employer un mouvement d'accès ou de sortie s'il aboutit à n'importe quel privilège pour un certain scénario d'attaque. Cette relation s'ajoute à la connaissance de l'intrus et lui donne de nouvelles capacités de mouvement. La compréhension de ce qui peuvent être prévus de l'intrus est cruciale pour assurer la protection proportionnée à un système. Le calcul de [40, 41] permet d'identifier des menaces potentielles en combinant le $\oplus$ de mov_n^k et de l'expression exp.P de la règle (10) dans le tableau 3.6 avec le choix compositionnel et le choix d'exécution de la relation de congruence structurelle présentée dans le tableau 3.5. Nous obtenons :
$(mov_n^k \oplus exp).\mathrm{P} \equiv (mov_n^k.\mathrm{P}) \sqcap (exp.\mathrm{P}) \sqcap (mov_n^k.exp.\mathrm{P}) \sqcap (exp.mov_n^k.\mathrm{P})$ Par conséquent, l'intrus a un choix pour faire une des quatre options suivantes :

- Utiliser la clef et cesser d'explorer.
- Ignorer la clef et continuer l'exploration.
- Utiliser la clef puis continuer d'explorer.
- Continuer d'explorer et utiliser la clef à un futur moment.

Si l'intrus choisit, par exemple, la troisième option ($mov_n^k.exp.\mathrm{P}$), alors le résultat est un mouvement intérieur ou extérieur d'un ambient n, dépendant si t prend la valeur e ou s. C'est les scénarios illustrés dans a) et b) du tableau 3.7. Les scénarios c) et d) illustrent le fait que si la clef publique est employée pour la protection

d'ambient, le processus d'intrus peut entrer ou sortir comme il veut.

P' $\rightarrow$ Q' if P' $\equiv$ P,P $\rightarrow$ Q , Q $\equiv$ Q'	(1)
P | R $\rightarrow$ Q | R if P $\rightarrow$ Q	(2)
mov_n^k.P | ${}_n^{k,k'}$[Q] $\rightarrow$ ${}_n^{k,k'}$[P | Q]	(3)
${}_n^{k,k'}$[$mov_n^{k'}$.P | Q] $\rightarrow$ P | ${}_n^{k,k'}$[Q]	(4)
$mov_{\overline{n}}^{-}$.P | ${}_n^{\delta,k'}$[Q] $\rightarrow$ ${}_n^{\delta,k'}$[P | Q]	(5)
${}_n^{k,\delta}$[$mov_{\overline{n}}^{-}$.P | Q] $\rightarrow$ P | ${}_n^{k,\delta}$[Q]	(6)
$req_{n,t}^x$.P | ($pub_{n,t}^k$.Q) $\rightarrow$ P[$x \leftarrow k$] | ($pub_{n,t}^k$.Q)	(7)
$prot_n^{k,k'}$.P | ${}_n^{\delta,\delta}$[Q] $\rightarrow$ P | ${}_n^{k,k'}$[Q]	(8)
$unprot_n^{k,k'}$.P | ${}_n^{k,k'}$[Q] $\rightarrow$ P | ${}_n^{\delta,\delta}$[Q]	(9)
exp.P | ($pub_{n,t}^k$.Q) $\rightarrow$ ($mov_n^k \oplus exp$).P | ($pub_{n,t}^k$.Q)	(10)

TABLE 3.6: Les règles de réduction de l'approche RSC

a) exp.P | ($pub_{n,t}^k$.Q) | ${}_n^{k,k'}$[R] $\rightarrow$ ($pub_{n,e}^k$.Q) | ${}_n^{k,k'}$[R | exp.P]
b) ${}_n^{k,k'}$[exp.P | ($pub_{n,s}^{k'}$.Q) | R] $\rightarrow$ exp.P | ${}_n^{k,k'}$[($pub_{n,s}^{k'}$.Q) | R]
c) exp.P | ${}_n^{\delta,k'}$[R] $\rightarrow$ ${}_n^{\delta,k'}$[R|exp.P]
d) ${}_n^{k,\delta}$[exp.P|R] $\rightarrow$ exp.P | ${}_n^{k,\delta}$[R]

TABLE 3.7: Capacités d'exploitation de l'intrus

3.5 Discussion

Une amélioration de cette approche a été déjà réalisé par S.Amouri et Y.Akik [42], ils ont utilisé une seule clé qui garantit l'accès et la sortie d'un ambient au lieu de deux clés, ce qui a simplifié l'approche. Ils ont aussi enlevé la notion "unprot" qui a pour rôle d'enlever la protection d'un ambient en remplaçant la clé privée k par une clé par défaut δ, en précisant qu'on peut utiliser juste l'action "prot" mais en indiquant la clé à mettre à l'ambient.

$unprot_n^k = prot_n^\delta$.

Les auteurs ont aussi amélioré l'approche RSC pour simuler la mobilité des ambients dans les réseaux AD HOC, pour cela ils ont ajouté deux capacités de mouvement d'ambient *in* et *out* qu'on voit ci-dessous :

- in_n^k Pour faire entrer l'ambient conteneur dans un ambient parallèle n en utilisant sa clef d'accès appropriée k.
- out_n^k Pour faire sortir l'ambient conteneur dans un ambient parallèle n en utilisant sa clef d'accès appropriée k.

Cependant cette amélioration reste insuffisante, du fait qu'elle modélise pas la mobilité des paquets dans le réseau AD HOC, pour palier à ce problème, nous avons ajouté quelques capacités qui réalisent ce qui est demandé. Ces capacités sont définis comme suit :

- req_c permet à un processus de faire une demande pour commencer à émettre les paquets.
- rep_c il est renvoyé au noeud source qui a demandé req_c pour lui confirmer la possibilité de l'émission .
- $send_P^c$ permet d'envoyer les paquet P vers un noeud destinataire.
- $drop_P^c$ action faite par l'intrus, qui permet de détruire les paquets reçus.

L'addition de ces nouvelles capacités nous a amené à ajouter à l'approche RSC dix autres règles de réduction présentées dans le tableau 3.9. Les règles (10) et (11) décrivent les capacités d'un ambient d'entrer et de sortir respectivement d'un autre ambient protégé. Les règles (12) et (13) dénotent le niveau de la protection offert par la clef publique δ, les tentatives d'un ambient pour accéder ou sortir d'un ambient protégé par δ sont réussies indépendamment de la clef utilisée. Le symbole "-" désigne n'importe qu'elle valeur de clef à employer par les capacités *in* et *out*. Les règles (14) et (15) montrent l'éxecusion des capacités d'exploration (exp.P) en parallèle avec la capacité de publication de clef (*pub*), ce qui génère un *in* ou un *out*, tout dépend de type de clef publié. Les règles (16)..(19) expliquent le processus d'exploitation de l'intrus pour intercepter les paquets, qui commence par envoyer une requête *req*

par le noeud qui veut émettre, suivie d'une réponse *rep* de l'intrus, après un *send* pour émettre les paquets, et enfin un *drop* qui détruit les paquets par l'intrus. Une requête *req* qui exécute en parallèle avec l'exploration de l'intrus (*exp*.I) génère une réponse *rep*. Et une requête *req* avec une réponse *rep* génère un *send*. Un *send* qui est exécuté avec une exploration de l'intrus (*exp*.I) permet de créer la capacité *drop* qui détruit les paquets.

Le calcul de [1,2] permet d'identifier des menaces potentielles en combinant le $\oplus$, de in_n^k ou de out_n^k avec l'expression (*exp*).P des règles (14) et (15) dans le tableau 3.9 avec le choix compositionnel et le choix d'exécution de la relation de congruence structurelle présentée dans le tableau 3.5. Nous obtenons :
$(in_n^k \oplus exp).\mathrm{P} \equiv (in_n^k.\mathrm{P}) \sqcap (exp.\mathrm{P}) \sqcap (in_n^k.exp.\mathrm{P}) \sqcap (exp.in_n^k.\mathrm{P})$.....(a)
$(out_n^k \oplus exp).\mathrm{P} \equiv (out_n^k.\mathrm{P}) \sqcap (exp.\mathrm{P}) \sqcap (out_n^k.exp.\mathrm{P}) \sqcap (exp.out_n^k.\mathrm{P})$.....(b)

Par conséquent, la nouvelle syntaxe et les nouvelles règles de réduction de l'approche RSC sont présentées dans les tableaux 3.8, et 3.9 respectivement :

- **Inactivité** : 0 est un processus qui n'exécute rien.
- **Composition parallèle** :P | Q se rapporte à l'exécution parallèle des processus P et Q.
- **Réplication** : !P est un processus qui dénote la réplication illimitée du processus P. c'est un opérateur commutatif et associatif.
- **Ambient protégé** : ${}_n^k[\mathrm{P}]$ dénote un ambient n qui contient une ressource P protégée par la clef k.
- **Action** : a.P présente le comportement séquentiel d'un processus comme séquence : il emploie d'abord la capacité « a » puis se comporte comme P.

Les sous-catégories suivantes se rapportent à des capacités de processus :

- **Publication de clef** : pub_n^x se rapporte à la publication de clefs.
- **Mouvement** : mov_n^k se rapporte aux capacités d'un processus pour se déplacer en utilisant la clef d'ambient appropriée.

- **Exploration** : exp permet la présentation des processus dynamiques qui peuvent choisir d'une manière non-déterministe ce qu'est leur prochaine action.
- **Protection** : $prot_n^k$ permet de renforcer la protection des ressources à l'intérieur d'un ambient n avec la clef k.
- **Choix compositionnel** : a $\oplus$ b permet à l'évolution d'un processus d'être définie comme choix entre toute combinaison possible des actions a et b
- **Choix non déterministe** : a $\sqcap$ b permet à l'évolution d'un processus d'être définie comme choix entre deux processus composants, mais ne permet pas à l'environnement n'importe qu'elle contrôle sur des processus composants.
- **Mouvement à l'intérieur** : in_n^k fait entrer l'ambient conteneur dans un ambient parallèle n en utilisant sa clef d'accès appropriée k.
- **Mouvement à l'extérieur** : out_n^k fait sortir l'ambient conteneur de son ambient parent n en utilisant sa clef d'accès appropriée k.
- **Trouver le chemin** : req_c permet à un processus de faire une demande pour commencer à émettre les paquets.
- **Confirmer le chemin** : rep_c il est renvoyé au noeud source qui a demandé req_c pour lui confirmer la possibilité de l'émission .
- **Envoyer les paquets** : $send_P^c$ permet d'envoyer les paquet P vers un noeud destinataire.
- **Détruire les paquets** : $drop_P^c$ action faite par l'intrus, qui permet de détruire les paquets reçus.

n	$\in$	N	Nom
x	$\in$	V	Variable
k	$\in$	K$\cup\{\delta\}$	Clefs de sécurité
c	::=		Entête de paquet
	|	adress = adr val	Adresses IP
adress	::=	src IP | dest IP	Adresses
adr val	::=	val.val.val.val	Valeurs de l'adresse
val	::=	num | *	Valeurs
P,Q	::=		Processus
	|	0	Inactivité
	|	P | Q	Composition parallèle
	|	!P	Réplication
	|	${}^{k}_{n}$[a.P]	Séquence d'ambient
a,b	::=		Capacités de processus
	|	pub^{x}_{n}	Publication de clefs
	|	mov^{k}_{n}	Mouvement d'un processus
	|	in^{k}_{n}	Mouvement d'entrée d'un ambient
	|	out^{k}_{n}	Mouvement de sortie d'un ambient
	|	exp	Exploration
	|	$prot^{k}_{n}$	Protection
	|	a $\oplus$ b	Choix compositionnel
	|	a $\sqcap$ b	Choix non-déterministe
	|	req_{c}	Demande la présence de chemin
	|	rep_{c}	Confirme la présence de chemin
	|	$send^{c}_{P}$	Envoyer les paquets
	|	$drop^{c}_{P}$	Détruire les paquets

TABLE 3.8: La nouvelle syntaxe de l'approche RSC

P' $\rightarrow$ Q' if P' $\equiv$ P,P $\rightarrow$ Q , Q $\equiv$ Q'	(1)
P | R $\rightarrow$ Q | R if P $\rightarrow$ Q	(2)
mov_n^k.P | ${}_n^k$[Q] $\rightarrow$ ${}_n^k$[P | Q]	(3)
${}_n^k$[mov_n^k.P | Q] $\rightarrow$ P | ${}_n^k$[Q]	(4)
$mov_{\overline{n}}$.P | ${}_n^\delta$[Q] $\rightarrow$ ${}_n^\delta$[P | Q]	(5)
${}_n^\delta$[$mov_{\overline{n}}$.P | Q] $\rightarrow$ P | ${}_n^\delta$[Q]	(6)
req_n^x.P | (pub_n^k.Q) $\rightarrow$ P[$x \leftarrow k$] | (pub_n^k.Q)	(7)
$prot_n^k$.P | ${}_n^\delta$[Q] $\rightarrow$ P | ${}_n^k$[Q]	(8)
exp.P | (pub_n^k.Q) $\rightarrow$ ($mov_n^k \oplus exp$).P | (pub_n^k.Q)	(9)
${}_n^\delta$[in_m^k.P | Q] | ${}_m^k$[R] $\rightarrow$ ${}_m^k$[${}_n^\delta$[P | Q]| R]	(10)
${}_m^k$[${}_n^\delta$[out_m^k.P | Q] | R] $\rightarrow$ ${}_n^\delta$[P | Q]| ${}_m^k$[R]	(11)
${}_m^\delta$[$in_{\overline{n}}$.P | Q] | ${}_n^\delta$[R] $\rightarrow$ ${}_n^\delta$[${}_m^\delta$[P | Q] | R]	(12)
${}_m^\delta$[${}_n^\delta$[$out_{\overline{n}}$.P | Q] | R] $\rightarrow$ ${}_n^\delta$[P | Q]| ${}_m^\delta$[R]	(13)
exp.P | (pub_n^k.Q) $\rightarrow$ ($in_n^k \oplus exp$).P | (pub_n^k.Q)	(14)
exp.P | (pub_n^k.Q) $\rightarrow$ ($out_n^k \oplus exp$).P | (pub_n^k.Q)	(15)
${}_n^\delta$ [req_c.P| exp.I] $\rightarrow$ ${}_n^\delta$ [req_c.P|$rep_{c'}$.I]	(16)
${}_n^\delta$ [req_c.P|$rep_{c'}$.I] $\rightarrow$ ${}_n^\delta$ [$send_P^{c'}$.P|I]	(17)
${}_n^\delta$ [$send_P^c$.P|exp.I] $\rightarrow$ ${}_n^\delta$ [$send_P^c$.P| $drop_P^c$.I]	(18)
${}_n^\delta$ [$send_P^c$.P| $drop_P^c$.I] $\rightarrow$ ${}_n^\delta$ [P|I]	(19)

TABLE 3.9: Les nouvelles règles de réduction de l'approche RSC

3.6 Modélisation de l'attaque Blackhole

3.6.1 Présentation du réseau

Nous avons choisi d'appliquer l'attaque blackhole sur un système simple qui contient deux zones : un établissement et un réseau AD HOC. Cet exemple est illustré dans la figure 3.1. On suppose que le niveau de confiance dans le réseau AD HOC est suffisamment haut pour que les machines n'utilisent que des clefs par défaut, ce qui rend ce réseau vulnérable à une attaque blackhole. Nous avons minimisé le nombre de machines utilisés dans cet exemple pour mieux le simplifier. Le réseau AD HOC contient 3 machines mobiles (A, B, et C). L'établissement inclut ce réseau AD HOC plus une autre machine I qui représente l'intrus. Chaque machine est représenté par un ambient qui comporte un processus présentant les ressources de cette machine et portant le même nom que cette machine, cet ambient porte le nom de la machine en minuscule. Le réseau AD HOC est aussi représenté par un ambient qui est nommé "r" et qui comporte un processus "R". On peut dire que l'ambient "r" est l'ambient conteneur et les ambient "a", "b", et "c" sont ses sous-ambients.

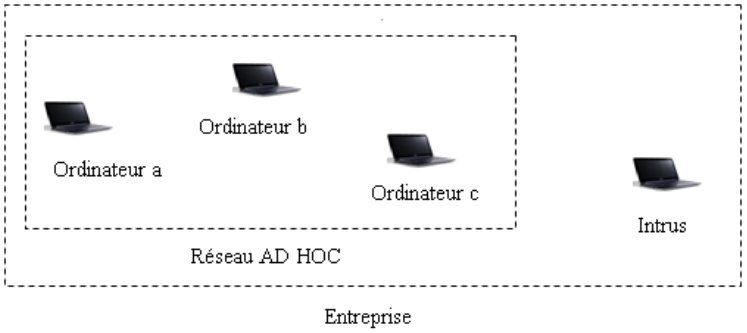

FIGURE 3.1: Représentation d'un réseau AD HOC

Chaque sous-ambient est protégé par une clef δ par défaut ; vue le niveau de confiance, donc il y a pas de restriction d'entrée ou de sortie. L'intrus une fois entré dans ce réseau, il peut choisir sa cible arbitrairement, dans cet exemple, on suppose

que sa cible sera la machine (A). Donc l'intrus entre dans le sous-ambient "a" et intercepte le "route-request" (req) et répond par un "route-reply" (rep) en le faisant croire qu'il a le chemin direct vers le destinataire de (A) quoi que ce soit, ce qui donne le feu vert à la machine (A) pour commencer à émettre les paquets vers la destination de l'intrus, et ce dernier pourra détruire ces paquets, ce qui fait un trou noir dans ce réseau ou les paquets seront perdus.

On dénote le système qui représente le réseau AD HOC par "S", qui est défini comme suit :
$\mathrm{S} = {}^{k}_{r}[\mathrm{R}|^{\delta}_{a}[req_c.\mathrm{A}]|^{\delta}_{b}[\mathrm{B}] \ |^{\delta}_{c}[\mathrm{C}]]$

3.6.2 Processus de l'intrus

Le processus de l'intrus est composé de processus "I" et de suite d'action qui le fait entrer jusqu'en ambient désiré dont le but et d'effectuer une attaque blackhole. L'intrus se connecte au réseau AD HOC pour explorer le système. L'opérateur exp modélise la manière dont l'intrus explore le système et gagne des capacités en termes d'action mov, in , ou out. Cet opérateur est le constructeur syntaxique qui imite le comportement non déterministe de l'intrus et il donne à l'intrus le choix d'employer ces clefs et de lancer son exploration. Une fois il est dans l'ambient désiré il génère des "route-reply" (rep)et des $drop$ grâce à sa capacité d'exploration.
Le processus de l'intrus est modélisé comme suit :

$$\mathrm{I} = in^{k}_{r} \oplus mov^{\delta}_{i} \oplus mov^{\delta}_{a} \oplus !(exp.\mathrm{I'})$$

Il y a soixante-quatre comportements possibles de l'intrus issues de la combinaison des trois opérateurs $\oplus$ dans l'expression ci-dessus. Considérant le cas simplifié de a, b, c et d, les choix sont :

$a \oplus b \oplus c \oplus d = a \sqcap b \sqcap c \sqcap d \sqcap a.b \sqcap a.c \sqcap a.d \sqcap b.a \sqcap b.c \sqcap b.d \sqcap c.a \sqcap c.b \sqcap c.d \sqcap d.a \sqcap d.b \sqcap d.c \sqcap a.b.c \sqcap b.a.c \sqcap a.c.b \sqcap b.c.a \sqcap c.a.b \sqcap c.b.a \sqcap b.c.d \sqcap c.b.d \sqcap b.d.c \sqcap c.d.b \sqcap d.b.c \sqcap d.c.b \sqcap a.b.d \sqcap b.a.d \sqcap a.d.b \sqcap b.d.a \sqcap d.a.b \sqcap d.b.a \sqcap c.d.a \sqcap d.c.a \sqcap c.a.d \sqcap d.a.c \sqcap a.c.d \sqcap a.d.c \sqcap a.b.c.d \sqcap a.b.d.c \sqcap a.c.b.d \sqcap a.c.d.b \sqcap a.d.b.c \sqcap a.d.c.b \sqcap b.a.c.d \sqcap b.a.d.c \sqcap b.c.a.d \sqcap b.c.d.a \sqcap b.d.a.c \sqcap b.d.c.a \sqcap c.a.b.d \sqcap c.a.d.b \sqcap c.b.a.d \sqcap c.b.d.a \sqcap c.d.a.b \sqcap c.d.b.a \sqcap d.a.b.c \sqcap d.a.c.b \sqcap d.b.a.c \sqcap d.b.c.a \sqcap d.c.a.b \sqcap d.c.b.a$. Le meilleur comportement que l'intrus peut choisir est :

I = $in_r^k.mov_i^\delta.mov_a^\delta.!(exp.\text{I'})$

Les autres comportements sont ignorés soit parce qu'ils ne produisent aucun résultat ou bien ils paralysent l'attaque. L'intrus entre dans l'ambient r pour se connecter au réseau AD HOC puisqu'il connait au commencement la clef d'accès à cet ambient ; ensuite, le processus d'intrus sort de l'ambient i par l'exécution de mov_i^δ pour entrer dans l'ambient a en exécutant mov_a^δ, puis lance l'exploration. Le tableau 3.10 montre les étapes de l'exploration .

$\text{S}\|_i^\delta[\text{I}]$	$\rightarrow$	${}_r^k[\text{R}\|_a^\delta[req_c.\text{A}]\|_b^\delta[\text{B}]\ \|_c^\delta[\text{C}]]\|_i^\delta[in_r^k.mov_i^\delta.mov_a^\delta.!(exp.\text{I})]$
	$\rightarrow$	${}_r^k[\text{R}\|_a^\delta[req_c.\text{A}]\|_b^\delta[\text{B}]\ \|_c^\delta[\text{C}]\|_i^\delta[mov_i^\delta.mov_a^\delta.!(exp.\text{I})]]$
	$\rightarrow$	${}_r^k[\text{R}\|mov_a^\delta.!(exp.\text{I})\|_a^\delta[req_c.\text{A}]\|_b^\delta[\text{B}]\ \|_c^\delta[\text{C}]]$
	$\rightarrow$	${}_r^k[\text{R}\|_a^\delta[req_c.\text{A}\|!(exp.\text{I})]\|_b^\delta[\text{B}]\ \|_c^\delta[\text{C}]]$
	$\rightarrow$	${}_r^k[\text{R}\|_a^\delta[req_c.\text{A}\|rep_{c'}.!(exp.\text{I})]\|_b^\delta[\text{B}]\ \|_c^\delta[\text{C}]]$
	$\rightarrow$	${}_r^k[\text{R}\|_a^\delta[send_P^c.\text{A}\|!(exp.\text{I})]\|_b^\delta[\text{B}]\ \|_c^\delta[\text{C}]]$
	$\rightarrow$	${}_r^k[\text{R}\|_a^\delta[\text{A}\|drop_P^c.!(exp.\text{I})]\|_b^\delta[\text{B}]\ \|_c^\delta[\text{C}]]$
	$\rightarrow$	${}_r^k[\text{R}\|_a^\delta[\text{A}\|!(exp.\text{I})]\|_b^\delta[\text{B}]\ \|_c^\delta[\text{C}]]$

TABLE 3.10: Modélisation de l'attaque blackhole avec l'approche RSC

3.6.3 Sécurisation du système

Après la modélisation de l'attaque blackhole sur un réseau AD HOC avec la nouvelle approche de RSC, nous avons constaté que l'intrus intercepte la requête

ROUTE_REQUEST émise par l' « ordinateur a » et répond par sa réponse falsifiée ROUTE_REPLY afin d'intercepter d'une manière discrète tous les paquets échangés entre la machine "A" et son destinataire et par la suite effectuer un déni de service (DoS). Le problème qui se pose est, comment modifier le système initial et le rendre sécurisé ?

Plusieurs approches et solutions ont été proposées pour sécuriser le routage dans les réseaux mobiles AD HOC, parmi celles-ci l'utilisation d'un IDS distributif et coopératif proposée par Zhang, Lee et Huang [43, 44], la vérification de l'authenticité des nœuds en utilisant la redondance du réseau ou l'utilisation des numéros de séquences pour les paquets proposées par Mohammad Al-Shurman, Seong-Moo Yoo et Seungjin Park dans [45].

Nous avons exploité une solution qui consiste à protéger chaque ordinateur avec une clef privée, de plus, pour s'assurer qu'une ancienne clef, possiblement compromise, n'est pas utilisée par un intrus, les clefs des ordinateurs a, b et c seront mis-à-jour (changement de clef) quand un nouveau système va joindre le réseau AD HOC.
Le renforcement du système se réalise grâce à l'utilisation de l'action *prot* par chaque ordinateur. Le tableau 3.11 montre les étapes de renforcement du système.

S'	$\rightarrow$	${}^{k}_{r}[prot^{ka}_{a}.prot^{kb}_{b}.prot^{kc}_{c}.\mathrm{R}|^{\delta}_{a}[req_c.\mathrm{A}]|^{\delta}_{b}[\mathrm{B}]\ |^{\delta}_{c}[\mathrm{C}]]$
S'	$\rightarrow$	${}^{k}_{r}[prot^{kb}_{b}.prot^{kc}_{c}.\mathrm{R}|^{ka}_{a}[req_c.\mathrm{A}]|^{\delta}_{b}[\mathrm{B}]\ |^{\delta}_{c}[\mathrm{C}]]$
S'	$\rightarrow$	${}^{k}_{r}[prot^{kc}_{c}.\mathrm{R}|^{ka}_{a}[req_c.\mathrm{A}]|^{kb}_{b}[\mathrm{B}]\ |^{\delta}_{c}[\mathrm{C}]]$
S'	$\rightarrow$	${}^{k}_{r}[\mathrm{R}|^{ka}_{a}[req_c.\mathrm{A}]|^{kb}_{b}[\mathrm{B}]\ |^{kc}_{c}[\mathrm{C}]]$

TABLE 3.11: Renforcement du système

Le processus d'intrus dans le nouveau système S' est modélisé comme suite :

I = $in_r^k \oplus mov_i^\delta \oplus !(exp.\text{I'})$

Il y a quinze comportements possibles de l'intrus présentés par les deux opérateurs $\oplus$ dans l'expression ci-dessus. Considérant le cas simplifié de a, b et c, les choix sont :
$a \oplus b \oplus c = a \sqcap b \sqcap c \sqcap a.b \sqcap a.c \sqcap b.a \sqcap b.c \sqcap c.a \sqcap c.b \sqcap a.b.c \sqcap b.a.c \sqcap a.c.b \sqcap b.c.a \sqcap c.a.b \sqcap c.b.a.$
Le meilleur comportement que l'intrus peut choisir est :
I = $in_r^k.mov_i^\delta.!(exp.\text{I'})$

Les autres comportements sont ignorés pour les raisons suivantes : ceux qui ne produisent aucun résultat ou ils paralysent l'attaque. L'exploration de l'intrus dans le nouveau système S' est similaire à celle du système S, lorsqu'il sort de l'ambient i il sera bloqué et n'entre pas dans l'ambient a, puisqu'il ne connait pas la clef d'accès vers cet ambient (ka). Les étapes de l'exploration de l'intrus sont présentées dans le tableau 3.12

$\text{S'}\|_i^\delta[\text{I}]$	$\rightarrow$	$_r^k[\text{R}\|_a^{ka}[req_c.\text{A}]\|_b^{kb}[\text{B}] \ \|_c^{kc}[\text{C}]]\|_i^\delta[in_r^k.mov_i^\delta.!(exp.\text{I'})]$
$\text{S'}\|_i^\delta[\text{I}]$	$\rightarrow$	$_r^k[\text{R}\|_i^\delta[mov_i^\delta.!(exp.\text{I'})]\|_a^{ka}[req_c.\text{A}]\|_b^{kb}[\text{B}] \ \|_c^{kc}[\text{C}]]$
$\text{S'}\|_i^\delta[\text{I}]$	$\rightarrow$	$_r^k[\text{R}\| !(exp.\text{I'})\|_a^{ka}[req_c.\text{A}]\|_b^{kb}[\text{B}] \ \|_c^{kc}[\text{C}]]$

TABLE 3.12: Exploration de l'intrus dans le système renforcé

3.7 Conclusion

Dans ce chapitre, nous avons amélioré l'approche RSC pour modéliser une attaque blackhole sur un réseau AD HOC, cette modélisation nous a montré la vulnérabilité du réseau qui utilise des clefs par défaut. Pour palier à ce problème de sécurité et parvenir à protéger le réseau, nous avons proposé une autre modélisation de système en utilisant des clefs privées, ce qui défend l'intrus de faire ces attaques.

Chapitre 4

Réalisation

4.1 Introduction

Dans ce chapitre, nous allons détailler le fonctionnement de l'application de Génération de spécification. Celle-ci a été réalisée par Marc Saint-Laurent [46] et améliorée par Y.Akik, et S.Amouri [42], dont le but est la spécification de propriétés de systèmes informatiques. Afin d'appliquer l'approche RSC sur notre système de spécification présenté dans le chapitre 3, nous avons implémenté les règles de réduction sur cette application. Dans la première partie, nous allons présenter le langage de programmation utilisé, ainsi que son environnement et ces outils nécessaires. Par la suite, nous allons présenter les règles de réduction, puis, nous expliquerons le déroulement de notre système de spécification.

4.2 Langage de programmation

Marc Saint-Laurent[46] a utilisé JAVA pour développer l'application de « Genspec », en effet, Java est un langage de programmation objet dont les premières versions datent de 1995 et il a été mis au point par la firme Sun Microsystems. Le langage Java a une syntaxe très proche du langage C++ et répond aux trois principes fondamentaux de la programmation orientée objet (POO) : l'encapsulation, le polymorphisme et l'héritage. Ce langage est caractérisé par sa facilité d'utilisation.

4.2.1 Environnement et outils de développement

Les outils suivants ont été utilisés pour créer cette application :

- **JDK (Java Development Kit)** :

  L'environnement dans lequel le code Java est compilé pour être transformé en ByteCode afin que la JVM (Java Virtuel Machine) de Java puisse l'interpréter.

- **L'environnement Eclipse IDE** :

  Eclipse IDE est un environnement de développement intégré libre, extensible, universel et polyvalent, permettant potentiellement de créer des projets de

développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java à l'aide de la bibliothèque graphique SWT d'IBM et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plugin (en conformité avec la norme OSGi) : toutes les fonctionnalités de cet atelier sont développées en tant que plug-in[47].

Cette application est développée principalement avec l'environnement de développement Eclipse, version 3.4.2. NetBeans est utilisé juste pour créer quelques interfaces graphiques.

4.3 Structure de l'application

La fenêtre principale (figure 4.1) se compose d'une barre de menus, une barre d'outils, le reste de la fenêtre est occupé par un espace de travail divisé en deux sections :

- La première section qui représente graphiquement la topologie du réseau.
- La deuxième section qui représente formellement le réseau.

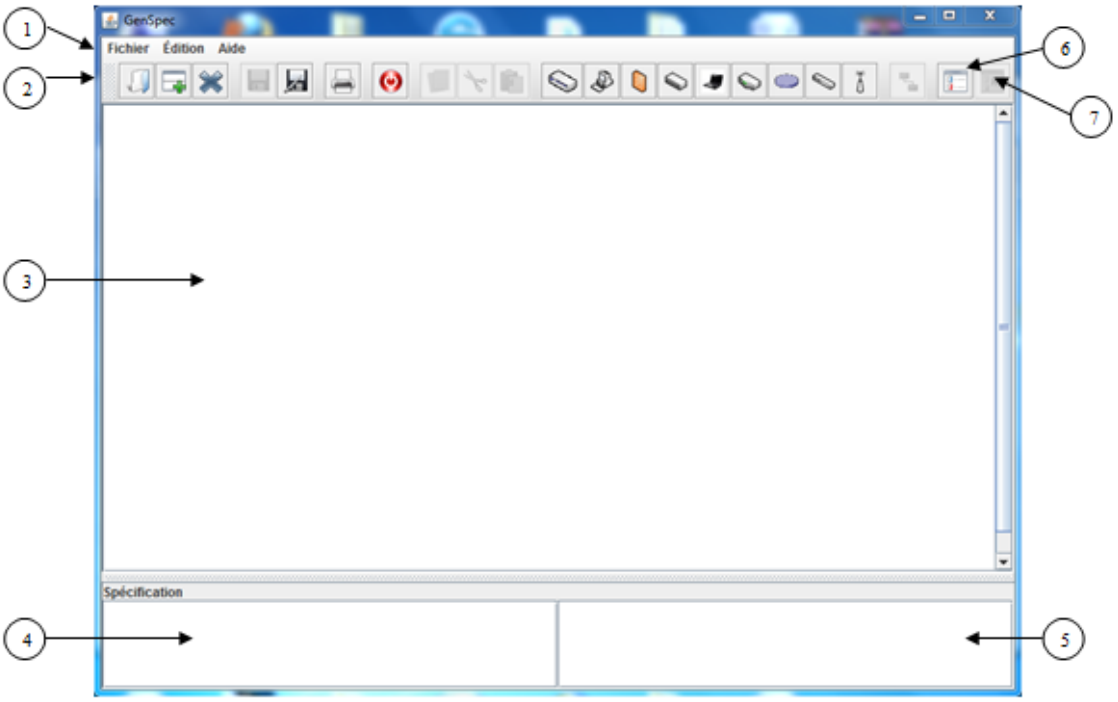

FIGURE 4.1: La fenêtre principale.

(1) : La barre de menu qui permet d'accéder aux fonctions de l'application.

(2) : La barre d'outils regroupe plusieurs boutons, et des icônes qui peuvent être retirées ou ajoutées de l'interface graphique.(figure 4.1)

(3) : Le composant qui représente graphiquement la topologie du réseau.

(4) : Le composant qui affiche la spécification de la topologie dans son état actuel.

(5) : Le composant qui permet d'obtenir la spécification en mode texte.

(6) : Le bouton qui permet de voir et modifier la spécification. (figure 4.2 et figure 4.3)

(7) : Le bouton qui permet de lancer la réduction. (figure 4.4)

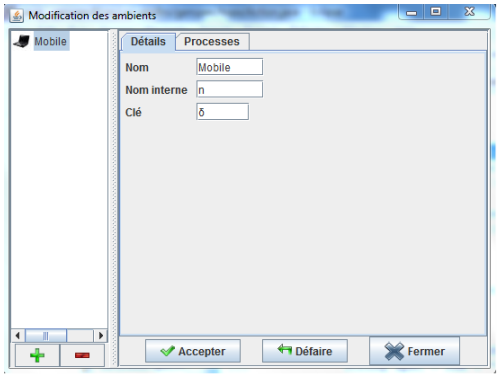

FIGURE 4.2: Détails ambients

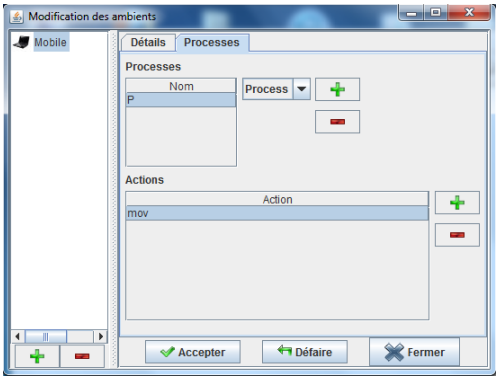

FIGURE 4.3: Détails Processes

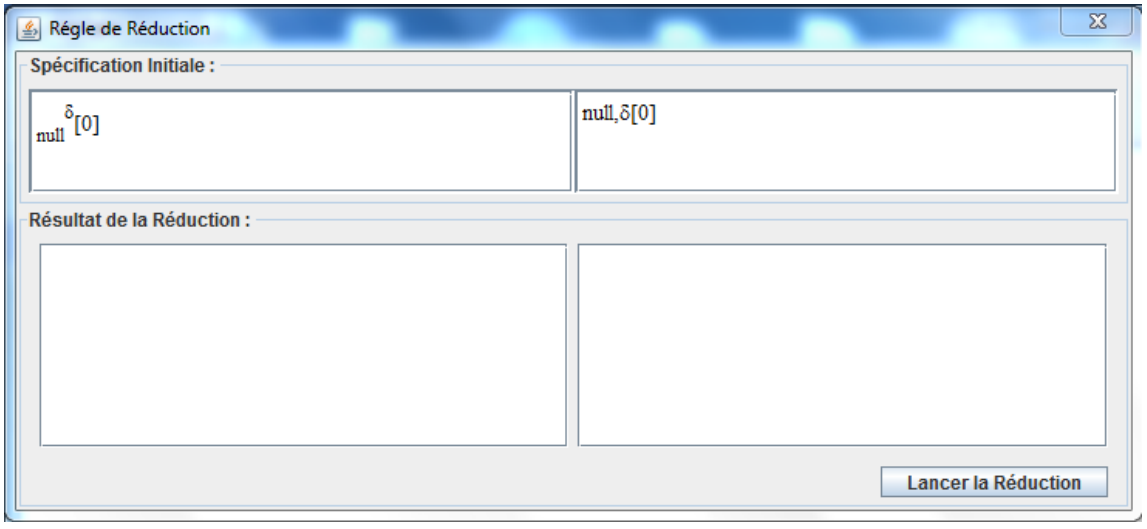

FIGURE 4.4: La fenêtre de la réduction.

4.4 Déroulement et résultats de l'exécution de l'application

4.4.1 Les règles de réduction

a. La règle numéro 8 :

Dans la figure 4.5, nous présentons le résultat de l'exécution de la règle numéro 8, celle-ci décrit le renforcement des clefs de sécurité.

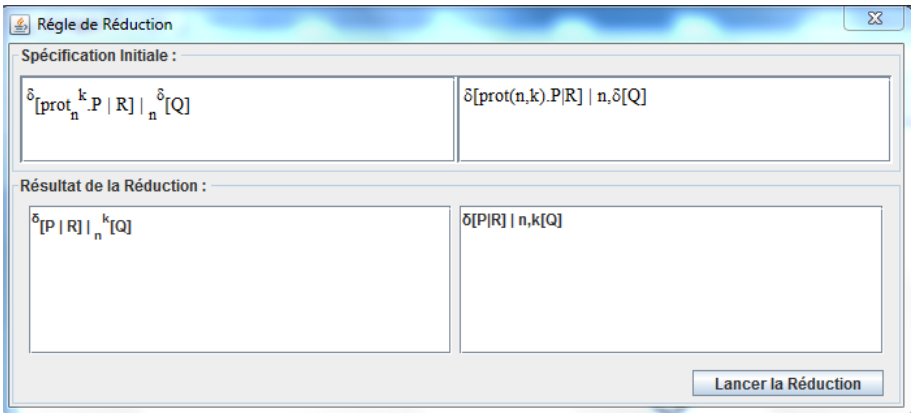

FIGURE 4.5: Résultat de la règle 8

b. Les règles numéro 9, 14, et 15 :

Dans la figure 4.6, nous présentons le résultat de l'exécution des règles numéro 9,14 et 15, celles-ci décrivent le comportement de l'intrus. Elles modélisent la manière dont le processus d'intrus peut employer un mouvement d'ambient ou de processus, d'accès ou de sortie s'il aboutie à n'importe quel privilège pour un certain scénario d'attaque. Cette relation s'ajoute à la connaissance de l'intrus et lui donne de nouvelles capacités de mouvement.

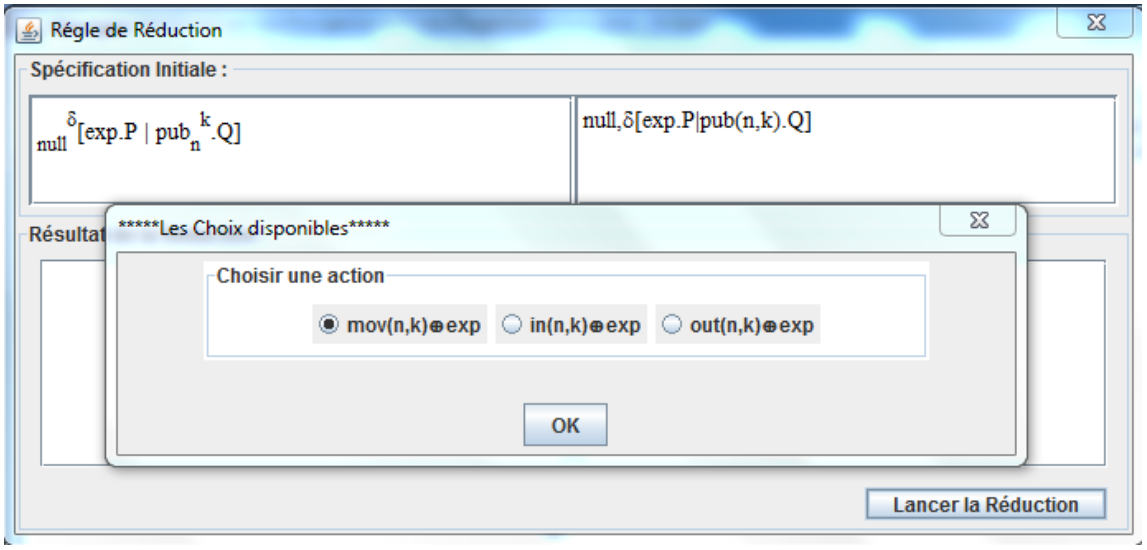

FIGURE 4.6: Résultat des règles 9,14,et 15.

d. La règle numéro 16 :

Dans la figure 4.7, nous présentons le résultat de l'exécution de la règle numéro 16, celle-ci décrit la capacité d'envoyer une requête par un noeud source.

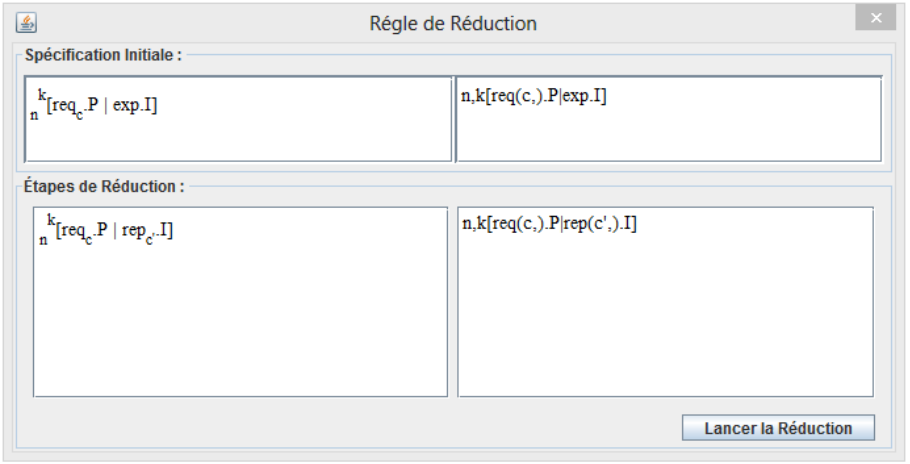

FIGURE 4.7: Résultat de la règle 16.

d. La règle numéro 17 :

Dans la figure 4.8, nous présentons le résultat de l'exécution de la règle numéro

17, celle-ci décrit la capacité de l'intrus à répondre à une reqûete.

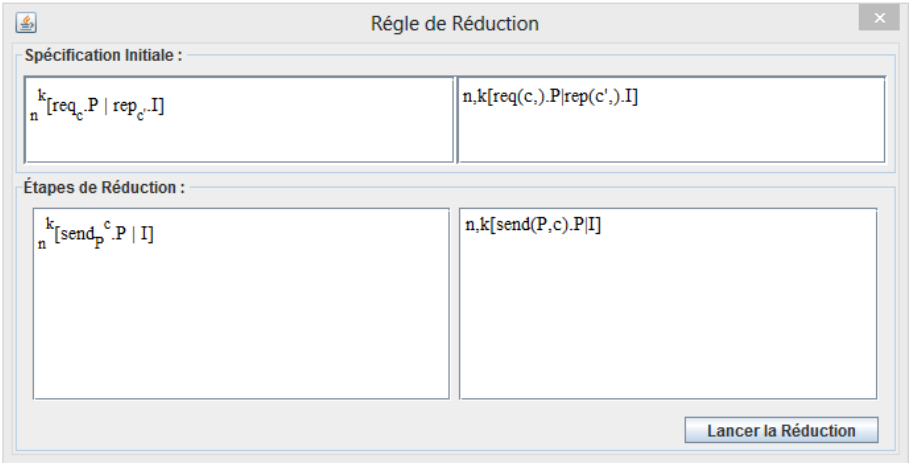

FIGURE 4.8: Résultat de la règle 17.

d. La règle numéro 18 :

Dans la figure 4.9, nous présentons le résultat de l'exécution de la règle numéro 18, celle-ci décrit la capacité d'envoyer les paquets à un noeud voisin.

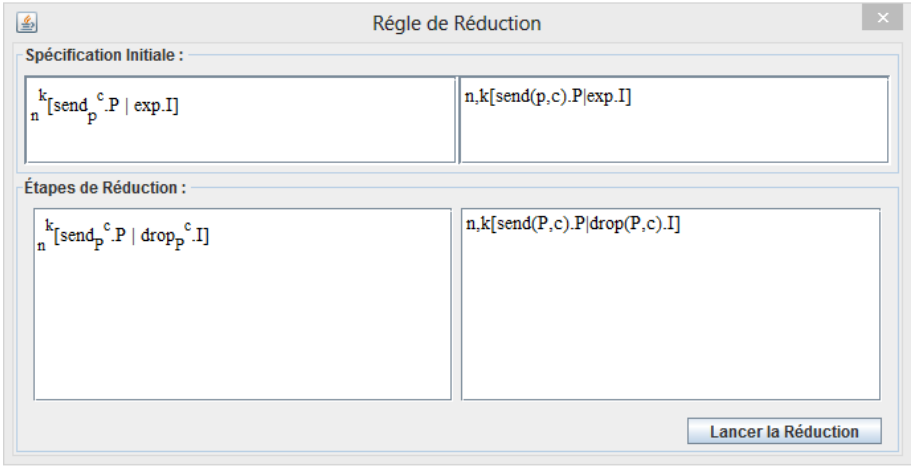

FIGURE 4.9: Résultat de la règle 18.

d. La règle numéro 19 :

Dans la figure 4.10, nous présentons le résultat de l'exécution de la règle numéro 19, celle-ci décrit la capacité de l'intrus à détriure les paquets.

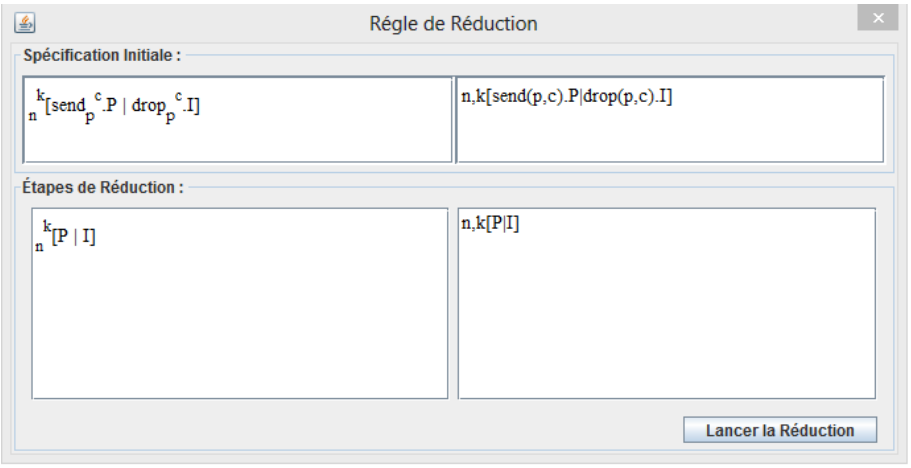

FIGURE 4.10: Résultat de la règle 19.

4.4.2 L'attaque blackhole

La figure 4.11 montre la topologie d'un réseau vulnérable à l'attaque blackhole présentée dans le chapitre 3, et la figure 4.14 présente le résultat finale de l'exploration de l'intrus. La figure 4.14 montre le résultat de l'exploration de l'intrus après la sécurisation du système en utlisant la capacité prot illustré dans la figure 4.13.

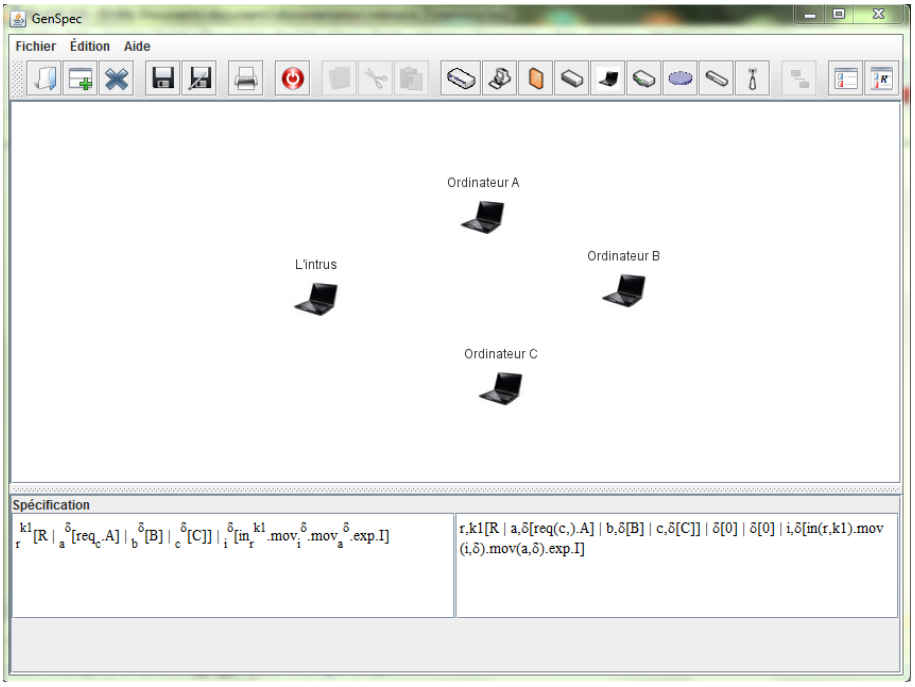

FIGURE 4.11: Attaque blackhole sur un réseau AD HOC

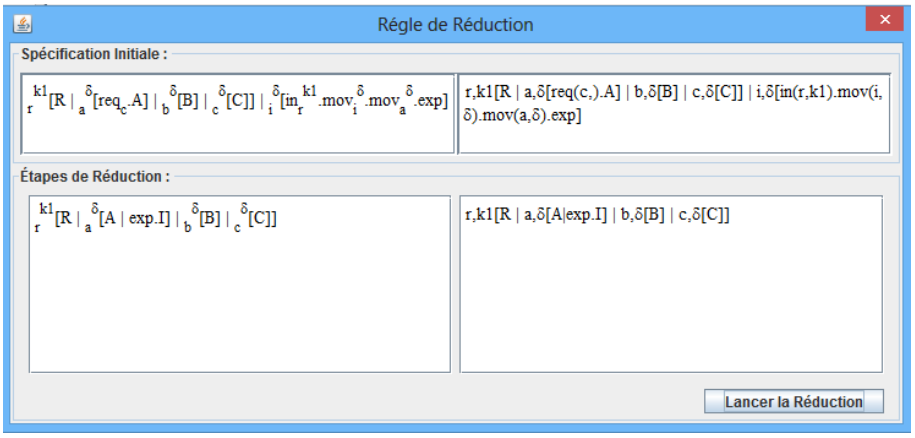

FIGURE 4.12: Résultat de l'attaque blackhole

On remarque sur la figure 4.12, que l'intrus a réalisé son attaque sur l'ambient "a", en entrant et en se servant des règles de réduction il a réussi à détruire les paquets envoyés par "a".

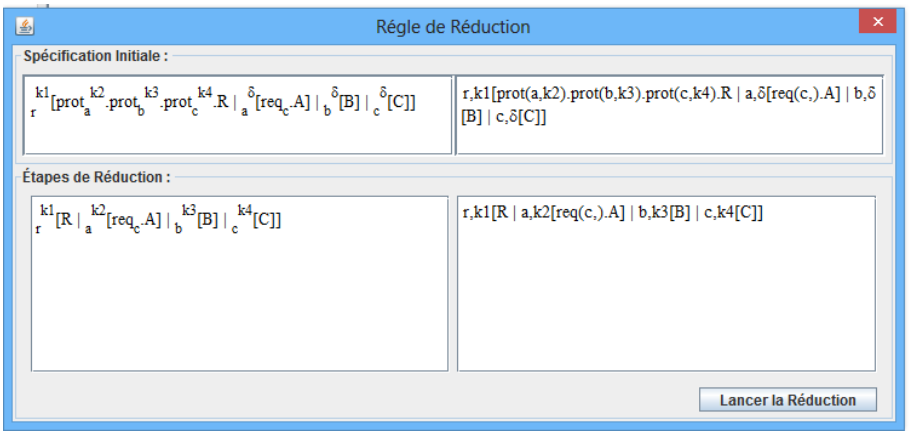

FIGURE 4.13: sécurisation de systèmes

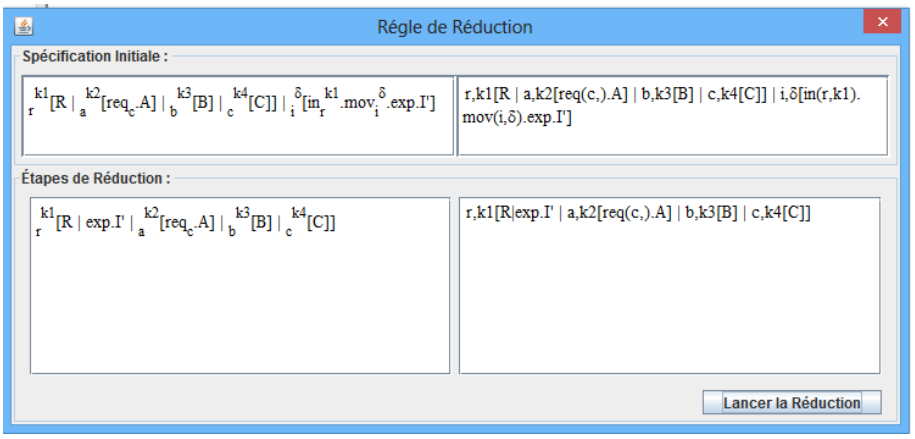

FIGURE 4.14: L'exploration de l'intrus sur le système sécurisé

4.5 Conclusion

Dans ce chapitre, nous avons, au premier lieu, présenté le langage que nous avons utilisé pour implémenter l'application. Par la suite, nous avons présenté quelques interfaces de l'application. Puis, nous avons exposé toutes les règles de réduction de l'approche RSC implémentées sur l'application Genspec, ainsi que la modélisation de l'attaque Blackhole sur un réseau AD HOC.

Conclusion générale et Perspectives

La sécurisation du routage dans les réseaux Ad hoc reste un problème majeur. Elle se heurte souvent à la difficulté de proposer des mécanismes relativement robustes face aux différentes attaques possibles, causées par les intrusions externes et les nœuds compromis sans pour autant affecter les performances globales du réseau ad hoc.

Dans ce mémoire, nous nous sommes intéressés à l'analyse de l'attaque Blackhole, nous avons présenté l'approche RSC basé sur le calcul ambient, ensuite nous lui avons apporté des améliorations afin de modéliser cette attaque. Ensuite, nous avons proposé une solution pour prévenir et éviter cette attaque, cette solution consiste à sécuriser les ambients avec des clefs privées. Nous avons implémenté les règles de réduction de l'approche étudiée sur l'application de Marc-Saint Laurent, et la modélisation de l'attaque, ainsi nous avons pu observer le comportement de l'intrus dans les deux systèmes l'initial et le renforcé. En guise de perspectives nous envisageons d'adapter l'approche RSC qu'on a amélioré pour modéliser différentes attaques connues, sur différents réseaux existants. Il faut aussi penser à perfectionner l'application pour répondre aux exigences de l'approche.

Bibliographie

[1] P. Mühlethaler, 802.11 et les réseaux sans fil, Eyrolles, 2002.

[2] K. Ayad, Sécurité du routage dans les réseaux ad hoc mobile, Thèse de doctorat, Ecole doctorale STIC, Ecole nationale Supérieure en Informatique (ESI), Alger, 2012.

[3] C. Liorens et L. Levier, Tableaux de bord de la sécurité réseau, Eyrolles, Paris-France, 2003.

[4] B. Wu et al, A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks, Department of Computer Science and Engineering, Florida Atlantic University, springer, 2006.

[5] P.Burkholder, Man in the Middle Attacks, Copyright SANS institute Author Retains Full Rights, 2002.

[6] A.Berg, AD HOC Networks specific attacks, Seminar Paper, Seminar AD HOC, Networking : concept, applications and security, 2003.

[7] N.B.Salem, L.Buttyan, G-P.Hubaux et M.Jakobsson, A charging and rewarding scheme for packet forwarding in multihop cellular networks. Proceedings of the 4^{th} ACM international symposium on mobile AD HOC networking and computing, Annapolis, Maryland, USA, 2003.

[8] Y.C.Hu, A.Pering et D.B.Johnson, A defense Againt Wormhole Attacks in Wireless Ad Hoc Networks, conference, San Francisko, 2003.

[9] B. Prêtre, "Attacks on Peer to peer networks", thesis, Dept. of Computer Science Swiss Federal Institute of Technology (ETH) Zurich, 2005.

[10] P. Kohli and U. Ganugula, 'DDoS Attacks using P2P Networks', Centre for Security Theory and Algorithmic Research, International Institute of Information Technology, India, April 25, 2007.

[11] Yingshu Li, My T. Thai, Weili Wu, «Wireless Sensor Networks and Applications », Springer Science+Business Media LLC, 2008

[12] Deng H, Li W, Agrawal DP (2002) Routing Security in Wireless Ad-hoc Networks. IEEE Communications Magazine 40(10) :70–75. doi : 10.1109/MCOM.2002.

[13] M. Langar, Cadre algébrique pour le renforcement de politique de sécurité sur des systèmes concurrents par réécriture automatique de programme,thèse de doctorat, Faculté des sciences et génie, université Laval, Québec, 2010.

[14] Fred B.Schneider. Enforceable security policies. ACM Trans. Inf. Syst.Secur., 3(1) :30-50, 2000.

[15] J. A. Goguen and J. Meseguer.Security policies and security models. IEEE Symposium on Security and Privacy, 00 :11, 1982.

[16] L. Bauer. Composing security policies with polymer. In Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation, pages 305-314. ACM, 2005.

[17] Ù. Erlingsson and F. B. Shneider. Sasi enforcement of security policies : a retrospective. In NSPW '99 :Proceedings of the 1999 workshop on New security paradigms, pages 87-95, New York, NY, USA, 2000. ACM.

[18] L. Bauer, J. Ligatti, and D. Walker.More enforceable security policies, survey, 2002.

[19] L. Bauer, J. Ligatti, and D. Walker.A language and system for composing security policies.Technical report, 2004.

[20] K. W. Hmalen, G. Morrisett, and F. B. Schneider.Computability classes for enforcement mechanisms.ACM Trans. Program. Lang. Syst., 28(1) :175-205,2006.

[21] H. Ould-Slimane, M. Mejri and K. Adi. Using edit automata for rewriting-based security enforcement. In DBSec, pages 175-190, 2009.

[22] M. Langar and M. Mejri. Formal and efficient enforcement of security policies. In FCS, pages 143-149, 2005.

[23] M. Langar, M. Mejri, and K. Adi. A formal approach for security policy enforcement in concurrent programs. In Security and Management, pages 165-171, 2007.

[24] T. Mechri, M. Langar, M. Mejri, H. Fujita, and Y. Funyu. Automatic enforcement of security in computer networks. In SoMeT, pages 200-222, 2007.

[25] K. W. Hamlen, G. Mortisett, and F. B. Schneider, Computability classes of enforcement mechanisms, ACMTOPLAS : ACM Transaction On Programming Language And Systems, 28, 2006.

[26] K. W. Hamlen, G. Mortisett, and F. B.Schneider, Computability classes of enforcement mechanisms, Technical report, Cornell University, August 26, 2003.

[27] A. Lacasse, Approche algébrique pour la prévention de l'intrusion, Faculté des sciences et génie, université Laval, Québec, Fevrier 2006.

[28] D. Gorla and R. Pugliese. Enforcing security policies via types, 2003.

[29] D. Gorla and R. Pugliese. Resource access and mobility control with dynamic privileges acquisition. In In Proc. Of ICALPS03, volume 2719 of LINCS, pages 119-132. Springer-Verlag, 2003.

[30] F. Martins and V. Vasconcelos. Controlling security policies in a distributed environment, 2004.

[31] M. Hannessy, M. Merro, and J. Rathke.Towards a behavioral theory of access and mobility control in distributed systems. In Theoretical Computer Science, pages 282-299. Springer-Verlag, 2003.

[32] M. Hennessy and J. Riely. Ressource access control in systems of mobile agents. Information and Computation, 173 :2002, 1998.

[33] L. Pene and K. Adi, A Calculus for distributed firewall specification and verification, Computer Science and Engineering Department, Université de Québec.

[34] R. Milner. "Communication and concurrency". Prentice Hall International (UK)Ltd, Hcrtfordshirc, UK, UK, 1995.

[35] R. Milner. Communicating and Mobile Systems : the Pi-Calculus. Cambridge University Press, June 1999.

[36] C. A. R. Hoare, "A model for communicating sequential processes" On the Construction of Programs, R. M. McKeag and A. M. McNaughton, Eds. London, England : Cambridge University Press, 1980, pp. 229-243.

[37] J. A. Bergstra and J. W. Klop. "Process algebra for synchronous communication". Technical Report 60, 1984.

[38] K.I.Consulting K.Ingham and S.Forrest. "A history and survey of network firewalls". The University of New Mexico Computer Science Department Technical Report, 2002.

[39] L. Cardelli and A. D. Gordon. Mobile ambients. In Foundations of Software Science and Computation Structures : First International Conference, FOSSACS 98. Springer-Verlag, Berlin Germany, 1998.

[40] K.ADI, L.HAMZA, L.PENE, Formal modeling for security behavior analysis of computer systems, IEEE Montreal conferenece on E technologies.23 25 janvier 2008. Canada.

[41] K.ADI, L.HAMZA, L.PENE, Approche algébrique pour le renforcement de la politique de sécurité dans les systèmes informatique , doctorial , JDI'10, BEJAIA. Octobre 2010.

[42] Y.Akik, S,Ammouri, Un Cadre Formel Pour la Prévention d'Intrusions, Mémoire, Université de Béjaia.

[43] Y.Zhang, W.Lee :Intrusion detection in wireless ad hoc networks. *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking,(MobiCom'00)2000.*

[44] Y.Zhang, W.Lee et Y.Huang :Inrusion detection techniques for mobile wirelessnetworks. *ACM/Kluwer Mobile Networks and Applications(MONET),to appear,2002.*

[45] M. Al-Shurman, Seong-Moo Yoo et S. Park "Black Hole Attack in Mobile Ad Hoc Networks". In Proceedings of the 42nd Annual Southeast Regional Conference, 2004, Huntsville, Alabama, USA, April 2-3,2004.

[46] Saint-Laurent, M. : Genspec : Rapport final. environnement graphique pour la spécification de propriétés des systèmes informatiques. Technical report, Université du Québec en Outaouais, Gatineau, Québec (2009).

[47] http ://www.e-citz.com/eclipse.html

Résumé

Le développement des systèmes informatiques facilite l'accès à l'information, et contribue également à l'essor de nouveaux services, ce développement s'est accompagné d'une panoplie de problèmes. Parmi ceux-ci, les plus inquiétants ont trait à la sécurité puisqu'elles mettent en péril le bon fonctionnement de ces systèmes. De ce fait il est important de mettre en place des méthodes formelles permettant de sécuriser les systèmes informatiques. Le but de ce papier est de modéliser une attaque blackhole dans un réseau AD HOC sous forme d'un langage formel en utilisant l'approche RSC, ce qui nous a permis de développer un système plus sécurisé.

Mots clés *: Algèbre de prcessus, Sécurité informatique, Calcul ambient, Méthode formelle, Blackhole.*

Abstract

The development of the computing systems facilitates the access to the information, and also contributes to the rise of new services, this development is accompanied by a panoply of problems. Among those, most worrying are related with security since they put in danger the good performance of these systems. Therefore it is important to develop formal methods in order to make safe the computing systems. The goal of this paper is to model an attack blackhole in an Ad hoc network in the form of a formal language by using the RSC approach, which enabled us to develop a protected system.

Key words *: Processus algevra, Computer security, Calculus ambient, Formal methods Blackhole.*