Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département Informatique

MEMOIRE

En vue de l'obtention du diplôme de

MASTER EN INFORMATIQUE

Option : Génie Logiciel

L'INTITULE DU MEMOIRE

Les attaques par déni de service distribué dans les systèmes informatiques

Préparé par :

SADAOUI Idir

Dirigé par :

HAMZA Lamia

Date de soutenance : Septembre 2017

Jury :

Président : HOUHA Amel

Examinateur : TIAB Amal

Examinateur : OUIAZINE Kahina

Année universitaire : 2016-2017

Dédicaces

Je dédie ce modeste travail

A la mémoire de mes parents,

A ma famille,

Mes proches et mes amis.

[Le véritable maître d'une chose n'est pas celui qui la possède mais celui qui peut la détruire] Frank Herbert

[Interfaces emerge when the system is used] Kari Kuutti

Remerciements

Je tiens à exprimer ma profonde gratitude et mes vifs remerciements à mon encadreur M^{me} HAMZA Lamia, qui n'a ménagé aucun effort pour m'apporter toute son aide et assistance tout le long de la réalisation de ce mémoire.

Je lui suis également reconnaissant pour la patience qu'elle a portée à mon égard et pour m'avoir encouragé dans mon travail.

Je tiens aussi à remercier chaleureusement ma famille pour leur compréhension, leur soutien moral et leurs encouragements continus et soutenus.

Enfin, je tiens particulièrement à remercier sincèrement les membres du jury qui m'ont fait l'honneur d'accepter de juger ce travail.

Liste des abréviations

ACL :	Access Lists
ACK :	Aknowledgment
ARP :	Address Resolution Protocol
CCS :	Calculus of Communicating System
DMZ :	DeMilitarized Zone
DNS :	Domain Name Server
DDOS :	Distributed Denial Of Service
DOS :	Denial Of Service
FTP :	File Transfer Protocol
GenSpec :	Génération de Spécification
HTTP :	HyperText Transfer Protocol
HTTPS :	HyperText Transfer Protocol Secure
ICMP :	Internet Control Message Protocol
IGMP :	Internet Group Management Protocol
IDS :	Intrusion Detection system
IMAP :	Internet Message Access Protocol
IP :	Internet Protocol
IPS :	Intrusion Prevention System
MVC :	Model, View , Controller
SMTP :	Simple Mail Transfer Protocol
TCP :	Transmission Control Protocol
TFN :	Tribal Flood Network
TFN2K :	Tribal Flood Network 2000
UDP :	User Datagram Protocol
UML :	Unified Modeling Language

Sommaire

Sommaire

Introduction générale

Introduction générale

L'avènement des systèmes informatiques, des réseaux et d'Internet a un impact mondial sur la croissance économique et la prospérité ainsi que sur la vie quotidienne des individus et des entreprises. Bien que la vie actuelle dans un monde dépendant d'Internet ait apporté des avantages sans précédent aux entreprises et aux particuliers, la menace sous-jacente d'attaques, ne peut être ignorée.

Aujourd'hui, le nombre de réseaux reliés à l'Internet ne cesse de croître, il devient donc facile pour un attaquant d'exploiter l'ouverture des réseaux dans un but malveillant. La sécurité est donc tout naturellement devenue primordiale.

La sécurité informatique, ou plus globalement la sécurité des systèmes d'information, représente l'ensemble des moyens et des techniques mises en œuvre pour assurer l'intégrité et la non-diffusion involontaire des données transitant dans le système d'information [2].

La sécurité informatique est un enjeu important et majeur à prendre sérieusement et prioritairement en considération dans tout système informatique (infrastructure réseau, machines, etc.).

Que ce soit à l'échelle d'une entreprise, d'une école, d'une université, d'un particulier ou même d'un pays; la sécurité d'un système d'information prend son importance dans la valeur des informations qu'il contient.

Avec le déploiement du réseau, on est face à une augmentation de la quantité, mais aussi et surtout à l'importance des données qui y transitent.

L'accès non autorisé à un réseau peut donner lieu à la divulgation d'information, à la modification de données, au déni de service et à l'utilisation illicite de ressources.

Sécuriser un réseau consiste à prendre en compte et à réduire autant que possible tous les risques possibles tels que : les attaques volontaires, les accidents, les défauts logiciels ou matériels et les erreurs humaines.

Une attaque dite de « déni de service distribué » (DDoS) si elle est effectuée simultanément par une multitude de sources.

Les attaques DDoS sont généralement destinées à perturber, voire immobiliser totalement un serveur, un service ou une infrastructure réseau.

Les attaques DDoS trouvent leur origine dans des botnets ou des réseaux vulnérables. Généralement considérées comme des attaques volumétriques de par leur énorme consommation de bande passante et leur envergure planétaire, les attaques DDoS ont pour objectif de provoquer la saturation des routeurs, pare-feu, serveurs et autres périphériques ou comme arme de diversion pour voler des données ou la propriété intellectuelle [18].

Les attaques par déni de service distribué (DDoS) sont aujourd'hui fréquentes, notamment du fait qu'elles sont relativement simples à mettre en œuvre, efficaces contre une cible non préparée. Ces attaques peuvent engendrer des conséquences non négligeables sur les pertes financières, par l'interruption de service ou encore indirectement par l'atteinte portée à l'image de la cible [19].

Une attaque distribuée (de déni de service distribué - DDoS) est une attaque évoluée invoquant plusieurs stations ou provenant de plusieurs sources. Les attaques distribuées sont plus dangereuses et difficiles à détecter puisqu'elles utilisent plusieurs stations intermédiaires, ce qui a pour effet la difficulté de déterminer la source d'une telle attaque.

Il est donc essentiel de mettre au point des techniques de détection et de répondre aux attaques rapidement. Le développement de ces techniques exige une bonne connaissance des attaques et d'identifier leurs propriétés de base et leur mode opératoire.

Aujourd'hui les chercheurs basent leurs recherches sur l'aspect mathématique et le résultat étant de définir des méthodes formelles et qui sont applicables.

En effet, pour notre travail, nous étudions une approche permettant de protéger un réseau contre les attaques DDoS.

Organisation du mémoire :

Ce mémoire est structuré en quatre chapitres :
Le premier chapitre présente une description des attaques par déni de service distribué (DDoS) les plus fréquentes dans les systèmes informatiques,

Le deuxième chapitre dresse un état de l'art des méthodes de modélisation et spécification formelles (approche algébrique de processus) de système informatique,

Le troisième chapitre est consacré à une étude de cas qui consiste en la modélisation et spécification du comportement d'une attaque par déni de service distribué (DDoS) à l'aide d'une méthode formelle, suivie par une application d'une approche de solution de sécurité contrant cette attaque,

Le quatrième chapitre est consacré au volet analyse (méthodologie) et application d'une implémentation sur la solution proposée,

Enfin, nous terminons le mémoire par une conclusion générale et quelques perspectives.

Chapitre 1

Les attaques par déni de service distribué dans les systèmes informatiques

1.1. Introduction

Ce chapitre définit tout d'abord ce qu'est l'attaque et les motivations des attaques, présente les principaux types d'attaques d'un système informatique, décrit ensuite l'attaque par déni de service (DoS attack) et les principales catégories d'attaques DoS couramment utilisées pour pénétrer les systèmes informatiques, définit en détail l'attaque par déni de service distribué (DDoS attack) illustrée schématiquement par des exemples, expose les principales catégories d'attaques DDoS, passe en revue l'impact et la protection des attaques DDOS puis présente les catégories d'attaques par exploitation de vulnérabilités et enfin termine par une conclusion.

1.2. Les attaques et leurs motivations

Tout ordinateur connecté à un réseau informatique est potentiellement vulnérable à une attaque informatique.

Une attaque informatique est l'exploitation d'une faille d'un système informatique (système d'exploitation, logiciel, etc.) à des fins non connues par l'exploitant du système et généralement préjudiciables.

Les motivations des attaques peuvent être de différentes sortes [24], [26] :

- Obtenir un accès au système
- Voler des informations, tels que des secrets industriels ou des propriétés intellectuelles
- Glaner des informations personnelles sur un utilisateur
- Récupérer des données bancaires
- Troubler le bon fonctionnement d'un service
- Utiliser le système de l'utilisateur comme « rebond » pour une attaque
- Utiliser les ressources du système de l'utilisateur, notamment lorsque le réseau sur lequel il est situé, possède une bande passante élevée

1.3. Les attaques par déni de service

1.3.1. Les attaques par déni de service (DoS Attacks)

Une attaque par déni de service consiste à saturer les ressources d'un système de façon à empêcher le bon fonctionnement. Les conséquences pour le système attaqué sont : l'instabilité, l'indisponibilité partielle ou totale du système.

Une attaque par déni de service se traduit par une interruption de service de réseau sur les utilisateurs, les périphériques ou les applications

On distingue deux méthodes majeures pour perpétrer une attaque DoS :

- Quantité encombrante de trafic : elle se produit lorsqu'un réseau, un hôte ou une application reçoit une énorme quantité de données à un rythme qui ne peut pas être géré
- Paquets formatés de manière malveillante : envoyés au destinataire (hôte ou application) et que ce dernier est incapable de les traiter [17].

1.3.2. Attaques par déni de service distribué (DDoS Attacks)

Il s'agit d'inonder une cible avec de nombreuses requêtes destinées à écrouler le système. Les attaques sont coordonnées à partir de N sources distinctes vers une cible unique (N pouvant être de l'ordre de quelques centaines, voire quelques milliers).[1]

Une attaque par déni de service distribuée (attaque DDoS) est similaire à une attaque DoS, mais elle provient de sources multiples et coordonnées.

Les attaques DDoS qui ne cessent d'accroître en complexité et en ampleur avec de nouveaux vecteurs d'attaques, émanent souvent de nombreux hôtes dits zombies (machines hostiles aux adresses différentes) contrôlés par l'attaquant. Ce qui rend difficile leur classification et les solutions pour les contrer ou les éviter, s'avèrent inefficaces. Et le principe même de l'attaque par déni de service distribuée est de diminuer les possibilités de stopper l'attaque [8].

Les attaques DDoS sont conçues pour saturer les connexions réseau de données illégitimes. Ces données peuvent submerger une liaison Internet au point d'empêcher tout trafic légitime. Les attaques DDoS s'appuient sur des méthodes similaires aux attaques DoS, mais elles se déploient toutefois à une plus grande échelle. En général, des centaines ou des milliers de points d'attaques tentent de submerger une cible.

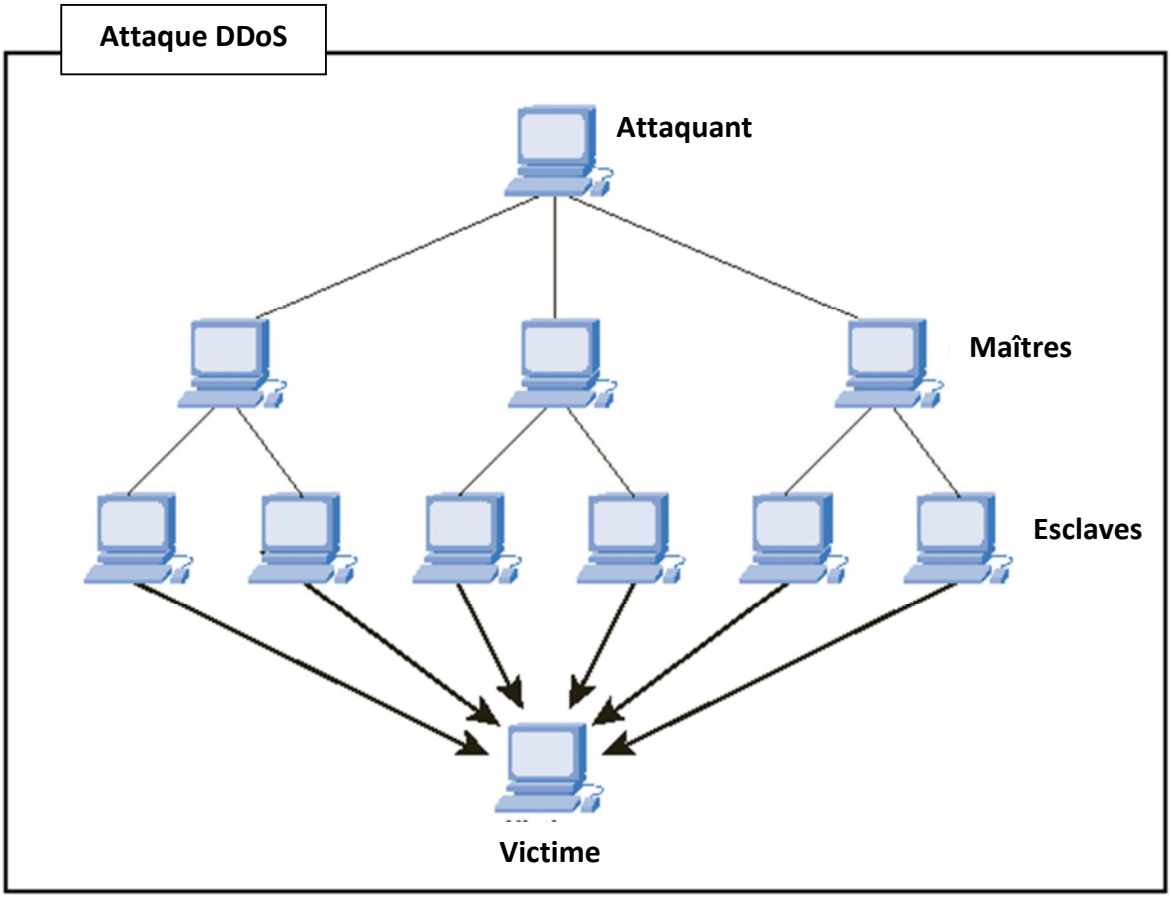

Figure 1.1 Exemple d'architecture d'une attaque DDoS [61]

1.3.2.1. Principe de l'attaque DDoS

Un attaquant établit via des techniques de compromission diverses, un réseau d'hôtes infectés (par des virus, chevaux de Troie ou vers), appelé réseau de zombies. Ces derniers sont contrôlés à distance par l'attaquant.

Les ordinateurs zombies analysent et infectent constamment plus d'hôtes et créent ainsi plus de zombies. L'attaquant communique avec les hôtes (agents) via des connexions sur des ports spécifiques ou en utilisant les mécanismes de Covert Channeling tel que LOK. Cela lui permet de coordonner et synchroniser les agents (zombies) de manière à ce que l'attaque soit initiée de manière simultanée et dirigée par le biais du réseau de zombies vers la cible (hôte, serveur, réseau), ce qui augmente les chances de réussite (pour l'attaquant) [1],[17].

1.3.2.2. Catégories de cibles d'attaques DDoS

Bande passante : consiste à saturer la capacité réseau d'un serveur, le rendant ainsi injoignable

Ressources : catégorie d'attaque consistant à épuiser les ressources système de la machine l'empêchant ainsi de répondre aux requêtes légitimes

Exploitation de la faille logicielle « Exploit » : ciblant une faille logicielle particulière afin de rendre la machine indisponible, ou d'en prendre le contrôle.

1.3.2.3. Les principaux types d'attaques DoS et DDoS

Nom de l'attaque	Niveau OSI	Type d'attaque	Explications du principe de l'attaque
ICMP Echo Request Flood	L3	Ressources	Aussi appelé Ping Flood, envoi massif de paquets (ping) impliquant la réponse de la victime (pong) ayant le même contenu que le paquet d'origine.
IP Packet Fragment Attack	L3	Ressources	Envoi de paquets IP référençant volontairement d'autres paquets qui ne seront jamais envoyés, saturant ainsi la mémoire de la victime.
SMURF	L3	Bande Passante	Attaque ICMP en broadcast usurpant l'adresse source pour rediriger les multiples réponses vers la victime
IGMP Flood	L3	Ressources	Envoi massif de paquets IGMP (protocole de gestion du multicast)
Ping of Death	L3	Exploit	Envoi de paquets ICMP exploitant un bogue d'implémentation dans certains systèmes d'exploitation
TCP SYN Flood	L4	Ressources	Envoi massif de demandes de connexion TCP
TCP Spoofed SYN Flood	L4	Ressources	Envoi massif de demandes de connexion TCP en usurpant l'adresse source
TCP SYN ACK Reflection Flood	L4	Bande passante	Envoi massif de demandes de connexion TCP vers un grand nombre de machines, en usurpant l'adresse source par l'adresse de la victime. La bande passante de la victime sera saturée par les réponses à ces requêtes.
TCP ACK Flood	L4	Ressources	Envoi massif d'accusés de réception de segments TCP
TCP Fragmented Attack	L4	Ressources	Envoi de segments TCP référençant volontairement d'autres segments qui ne seront

			jamais envoyés, saturant ainsi la mémoire de la victime
UDP Flood	L4	Bande Passante	Envoi massif de paquets UDP (ne nécessitant pas d'établissement de connexion préalable)
UDP Fragment Flood	L4	Ressources	Envoi de datagrammes UDP référençant volontairement d'autres datagrammes qui ne seront jamais envoyés, saturant ainsi la mémoire de la victime
Distributed DNS Amplification Attack	L7	Bande Passante	Envoi massif de requêtes DNS usurpant l'adresse source de la victime, vers un grand nombre de serveurs DNS légitimes. La réponse étant plus volumineuse que la question, s'ensuit une amplification de l'attaque
DNS Flood	L7	Ressources	Attaque d'un serveur DNS par l'envoi massif de requêtes
HTTP(S) GET/POST Flood	L7	Ressources	Attaque d'un serveur web par l'envoi massif de requêtes
DDoS DNS	L7	Ressources	Attaque d'un serveur DNS par l'envoi massif de requêtes depuis un grand ensemble de machines contrôlées par l'attaquant

Le **tableau 1.1** : Principales catégories d'attaques DoS et DDoS [20]

1.3.2.4. Exemples d'attaques DoS et DDoS

a. L'attaque TCP SYN Flooding

Etablissement d'une session TCP :

TCP est un protocole orienté connexion. L'établissement d'une session permet de s'assurer que les applications sont prêtes à recevoir les données et que toutes les données sont bien reçues par le destinataire [2].

Une connexion TCP s'établie en trois phases selon le mécanisme de poignée de main en trois temps (Three-ways handshake). Ces trois étapes sont l'envoi d'un SYN, la réception d'un SYN-ACK et l'envoi d'un ACK.

SYN Flooding :

L'attaque SYN Flood est l'une des attaques les plus répandues qui exploite le mécanisme de poignée de main en trois temps (Three-ways handshake) du protocole TCP. Elle consiste en l'envoi d'un grand nombre de demandes de connexions au serveur cible (SYN) à partir de plusieurs machines et ne pas y répondre. Lors d'une demande de connexion, le serveur est en attente et bloque pendant un certain temps une partie de ses ressources pour cette nouvelle connexion. Le but est d'envoyer plus de demandes qu'il ne peut en traiter dans un temps donné. Ainsi, le serveur gaspille toutes ses ressources réseau à répondre à des requêtes qui ne mènent nulle part et il ne pourra plus subvenir aux besoins de vrais clients [2][18].

b. L'attaque Smurf

L'attaque Smurf s'appuie sur le ping (**ping** est un outil exploitant le protocole ICMP, permettant de tester les connexions sur un réseau en envoyant un paquet et en attendant la réponse) et les serveurs de diffusion (broadcast) pour paralyser le réseau (Un serveur broadcast est un serveur capable de dupliquer un message et de l'envoyer à toutes les machines présentes sur le même réseau).

On falsifie d'abord l'adresse IP source pour se faire passer pour la machine cible. On envoie alors un ping sur un serveur de broadcast. Il le fera suivre à toutes les machines qui sont connectées qui renverront chacune un pong au serveur qui fera suivre à la machine cible [24].

c. L'attaque UDP Flood

L'attaque UDP Flood consiste en l'envoi d'une grande quantité de paquets UDP sur des ports aléatoires de la machine victime. Ainsi, celle-ci sera obligée de répondre par l'envoi de nombreux paquets ICMP, la rendant inaccessible par d'autres clients. Ce qui conduit, au bout d'un certain temps à une congestion du réseau ainsi qu'à une saturation des ressources de la machine cible. Cette congestion est généralement plus importante qu'avec le TCP Flood car UDP ne possède pas de mécanisme de contrôle de congestion (time out), donc la totalité de la bande passante peut être saturée (effondrement de la totalité du réseau).
Comme solution pour contrer cette attaque : bloquer les paquets d'attaque en utilisant des règles de filtrage (Firewall) [58].

d. L'attaque ARP

L'attaque ARP est l'une des attaques les plus célèbres, elle consiste à exploiter une faiblesse du protocole ARP. Son principe consiste à s'interposer entre deux machines du réseau et de transmettre à chacune un paquet ARP falsifié indiquant que l'adresse ARP (adresse MAC) de l'autre machine a changé, l'adresse fournie étant celle de l'attaquant.
Les deux machines cibles vont ainsi mettre à jour leurs tables dynamiques appelées cache ARP. On parle ainsi « ARP cache poisoning » ou « ARP spoofing » ou encore « ARP redirect » pour désigner ce type d'attaque. De cette manière, à chaque fois qu'une des deux machines souhaitera communiquer avec la machine distante, les paquets seront envoyés à l'attaquant qui les transmettra de manière transparente à la machine destinatrice [58].

1.3.2.5. Les outils d'attaques DDoS

Le tableau **1.2** décrit les outils les plus utilisés pour déclencher (lancer) des attaques DDoS

Outil	But
TFN	Architecture client/agent Protocole de communication : ICMP Attaque sur les protocoles : IP/TCP/UDP/ICMP Les agents TFN sous le contrôle du TFN client, réalisent l'attaque distribuée sur la victime/cible et avec le type d'attaque désiré par le client. Les adresses du client et des agents sont usurpées (falsifiées)
TFN2K	Version évoluée de TFN Communications entre le client et ses agents sont chiffrées : ce qui le rend plus dur à détecter Protocoles de communication : TCP/UDP/ICMP
Trin00	Architecture à trois couches : un client qui envoie des commandes à des serveurs maîtres qui se chargent chacun d'un sous réseau d'agents. Attaque sur le protocole : UDP

Tableau 1.2 : Exemple d'outils d'attaques DDoS [21]

1.3.2.6. Impact potentiel des attaques DDoS

L'impact potentiel des attaques DDoS se résume en des points suivants :

- Perturber voire immobiliser totalement les services
- Perte de visibilité sur les ressources
- Ces attaques peuvent aussi bien servir des intentions criminelles que d'un simple acte de malveillance destiné à causer du tort à la cible
- Leur rayon d'action s'élargit, ciblant tous types d'organisations et tous secteurs d'activités en proie à des attaques DDoS
- Risque de perte de compétitivité (cas d'indisponibilité de service)
- Impact sur la confiance des investisseurs, des clients et des collaborateurs [18].

1.3.2.7. Prévention des attaques DDoS

Dans le cas d'une attaque DDoS cela se complique énormément puisque les adresses IP sources sont nombreuses et variées.

En effet, les attaques DDoS sont relativement compliquées à arrêter une fois que les hôtes (agents) ont commencé à attaquer la cible.

Le meilleur moyen de les prévenir consiste à les détecter avec les IDS pendant la phase préliminaire, lorsque l'attaquant communique avec les agents (hôtes).

1.4. Conclusion

Dans ce premier chapitre, nous avons présenté une description des principales attaques DoS et DDoS illustrées par quelques exemples;

Le chapitre suivant présentera l'état de l'art de quelques méthodes formelles qui pourront être utilisées pour spécifier et vérifier des réseaux informatiques à sécuriser.

Chapitre 2

Les méthodes formelles et l’algèbre de processus

2.1. Introduction

L'utilisation des méthodes formelles est devenue incontournable pour la spécification et l'analyse des systèmes informatiques dans le but d'augmenter leurs fiabilités.

En effet, ces méthodes ont des fondements mathématiques consistants, leur permettant d'accompagner par des preuves irréfutables les conclusions qu'elles produisent.

Principalement, les méthodes formelles sont des langages non ambigus pour la spécification des systèmes et de leurs propriétés, des techniques permettant la vérification des propriétés de ces systèmes et des outils permettant d'automatiser l'utilisation de ces techniques.

Dans ce chapitre, nous présentons les méthodes de spécification formelle, l'algèbre de processus et les langages les plus utilisés pour spécifier les systèmes concurrents d'une manière générale et les réseaux informatiques en particulier ainsi que leurs propriétés.

2.2. Les méthodes formelles et spécification formelle

2.2.1. Définitions :

Méthodes formelles :

Selon Wing (1990), les *méthodes formelles* :

- Sont des techniques basées sur les mathématiques pour décrire les propriétés d'un système,

- Fournissent un cadre systématique pour :
  - développer le système
  - valider et vérifier le système [31].

« Les *méthodes formelles* peuvent être utilisées pour spécifier et modéliser le comportement d'un système et pour vérifier de façon mathématique que la conception et la mise en œuvre du système satisfont bien les fonctionnalités ainsi que diverses propriétés de sûreté. » Holloway (1996) [31].

Spécification formelle :

Selon Gaudel, Marre, Bernet et Schlienger [36], une *spécification* est dite *formelle* si :

- « Elle est écrite en suivant une syntaxe bien définie, comme celle d'un langage de programmation »,

- « La syntaxe est accompagnée d'une sémantique rigoureuse qui définit des modèles mathématiques représentant les réalisations acceptables de chaque spécification syntaxiquement correcte ».

Validation et vérification:

Processus consistant à s'assurer qu'un système satisfait une spécification.

2.2.2. Avantages des méthodes formelles :

- Utilisation de concepts de la logique et de la technique mathématique
- La précision et la clarté (non ambiguïté)
- Les notions d'ensemble, de relation, de fonction et leurs différentes propriétés et opérations avec les quantifications universelles et existentielles, nous permettent d'établir une spécification d'une manière simple et claire et de démontrer mathématiquement les propriétés de la spécification
- Les méthodes formelles fournissent des spécifications qui peuvent être rigoureusement vérifiées, analysées et testées dés les premières étapes du cycle de développement et améliore la qualité du logiciel
- Les méthodes formelles nous permettent de spécifier ce qui est nécessaire à un niveau d'abstraction particulier. [22]

2.2.3. Objectifs des méthodes de spécification formelle :

- Pour spécifier clairement et précisément le comportement attendu d'un système

2.2.4. Langage formel de spécification: [31]

Le *langage formel* est un langage :

- Ayant la syntaxe et la sémantique bien définies : *pas de flou possible*
- Permettant une description abstraite du comportement d'un système : *concision*

Syntaxe d'un langage :

La *syntaxe d'un langage* décrit les mots, les phrases et les énoncés acceptables et bien formés.

Sémantique d'un langage :

La *sémantique* définit le sens et la signification des énoncés du langage

Parmi les *langages formels* qui ont été développés dans le but de décrire mathématiquement et sans ambiguïté des systèmes informatiques; on en trouve les *Algèbres de processus*.

2.3. Algèbres de processus

L'*algèbre* est la branche des mathématiques qui étudie les structures algébriques, indépendamment de la notion de limite (rattachée à l'analyse) et de la notion de représentation graphique (rattachée à la géométrie) [*Wikiversité*]

Une *algèbre* est constituée d'un domaine et d'un ensemble d'opérations qui traitent des données comme en arithmétique, mais sur un plan plus général en utilisant la théorie des ensembles.

Un *processus* fait référence au comportement d'un système, c'est-à-dire à l'ensemble d'événements ou actions qu'il peut réaliser, l'ordre dans lequel ils peuvent être exécutés et d'autres aspects de cette exécution telle que la durée ou les probabilités [44].

Une *algèbre de processus* exprime le fait d'utiliser une approche algébrique ou axiomatique pour modéliser le comportement d'un processus.

L'*algèbre de processus* est une technique de description formelle pour les systèmes complexes, plus particulièrement ceux qui communiquent et s'exécutent en parallèle.
Elle offre : - une modélisation compositionnelle
- une sémantique opérationnelle
- un raisonnement sur les comportements : relations d'équivalence, notions de raffinement [38].

Les *algèbres de processus* ont été conçues dans le but de cerner les primitives de base, d'avoir une meilleure compréhension du parallélisme et d'élaborer des résultats théoriques dés la conception même d'un système informatique [27].

Avantages de l'algèbre de processus :

- Les *algèbres de processus* se prêtent bien à la modélisation puisqu'elles constituent des langages formels, qu'elles ont des syntaxes simples et expressives et qu'elles permettent la vérification de propriétés de sécurité.
- Les *algèbres de processus* constituent une base importante pour l'ingénierie des systèmes en fournissant un support à la spécification, à la vérification, à l'implantation, aux tests ainsi qu'aux autres étapes du cycle de vie des activités de ces systèmes.
- Dispose d'un seul langage
- Les processus associés peuvent être substitués les uns aux autres

- Les processus peuvent être minimisés [38].

Parmi les algèbres de processus les plus connus dans le domaine de la spécification et la vérification des systèmes concurrents et les réseaux informatiques, nous trouvons : le *CCS* [39] et le *π-Calcul* [40] développées par Milner, le *CSP* [41] proposée par Hoare, l'*ACP* [42] introduite par Brookes et Roscoe et le *calcul ambient* [43] développé par Cardelli et Gordon.

Dans ce qui suit, nous présentons les trois algèbres de processus suivantes : *CCS, π-Calcul* et le *calcul ambient :* ce sont les trois algèbres qui peuvent être utilisées pour modéliser le comportement des composants réseau.

2.3.1. CCS

Le *CCS* est une algèbre de processus pure, qui se concentre uniquement sur la synchronisation des processus.

CCS permet de modéliser les systèmes et leurs interactions.

Tout processus CCS est exprimé selon les règles syntaxiques et évolue selon des règles de la sémantique opérationnelle.

CCS : algèbre développé par Milner au cours des années 70, est l'objet d'une première publication 1980 intitulée *a Calculus of Communicating System*. Cette algèbre est la première sur le sujet et aura une influence marquante sur le développement des algèbres de processus pour les décennies à venir [38].

CCS est souvent utilisé pour évaluer certaines propriétés d'un système telles que les blocages (deadlock) [27].

2.3.2. π-Calcul

Le *π-Calcul* : version évoluée de CCS, porte sur la nature et les vecteurs de communication inter-processus.

Le *π-Calcul* est un modèle mathématique de processus qui, lors de l'interaction des processus, permet de modifier la topologie des canaux de communication. La singularité de ce calcul est le transfert des canaux de communication entre processus. Le processus recevant un canal peut utiliser ce dernier pour poursuivre l'interaction avec l'environnement et nouveaux processus.

Le passage de canaux s'apparente aux notions de contrôle d'accès et à l'accès de ressources ; il procure au *π-Calcul* toute sa puissance expressive : il offre une expressivité accrue en considérant la migration de la portée de variables échangées [38].

Le *π-Calcul* a été introduit par Milner, Parrow, et Walker dans [40] comme une extension de CCS, car ce dernier permet de modéliser les systèmes distribués mais ne supporte pas la capacité d'exporter des noms de canaux d'un processus à un autre. Dans le *π-Calcul*, il devient possible de faire connaitre des noms de canaux en les transmettant à travers d'autres canaux connus [25].

2.3.3. Le Calcul Ambient

Le *Calcul Ambient* a été introduit par Cardelli et Gordon en 1998 [43]. Son but est de fournir un langage formel simple et expressif pour l'étude de la programmation mobile dans les réseaux à large échelle.

Le calcul Ambient provient originalement du *π-Calcul* et constitue une évolution majeure grâce à l'ajout d'une notion additionnelle de représentation spatiale des processus : l'Ambient. Absente dans les autres calculs, cette notion (d'ambient) est très utile pour la spécification et vérification de certains systèmes réactifs et en particulier les réseaux informatiques.

Un ambient est un espace (un réseau, une machine, une page web, etc.) ayant un intérieur et un extérieur dans lequel peuvent se dérouler des calculs et il possède un nom unique non modifiable au cours du temps. 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 d'autres ambients. Par ailleurs, un ambient peut migrer (notion de mobilité) avec son contenu d'un espace à un autre.

Nous développons ci-après la syntaxe du calcul ambient et sa sémantique.

2.3.3.1. Syntaxe du Calcul Ambient

La syntaxe du calcul ambient est indiquée dans le tableau 2.1

- Le processus "0" représente le processus inactif
- La restriction de n à P, notée (vn)P
- Le symbole " | " désigne la composition parallèle
- 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 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 puis il continue comme P
- 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 ambient parent N, tandis que *open* N élimine les frontières de l'ambient fils N.

(a)Processus

P	::=		Processus
	|	0	Inactivité
	|	(νn)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
|	Є	Capacité nulle
|	M.M'	Chemin d'accès
(c)Noms		
|	N	Nom
|	≪n≫	Sortie d'un nom
|	((n)).P	Entrée d'un nom

Tableau 2.1 : Syntaxe du calcul ambient [43]

2.3.3.2. Sémantique opérationnelle du calcul ambient

La sémantique opérationnelle du calcul ambient est définie via deux relations; la congruence entre processus ≡ et des règles de réduction → comme le montrent le **tableau 2.2** et le **tableau 2.3** respectivement.

Dans le calcul ambient, l'opérateur ≡ représente un processus ayant la même structure interne, les règles régissant ces évolutions sont appelées règles de congruence structurelle. Quant à l'opérateur →, 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 →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 2.2**. 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.

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

Tableau 2.2: Règles de congruence structurelle du calcul ambient [43]

La relation de réduction est définie par les axiomes et les règles du tableau 2.3, 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.

$n[in\ m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R]$	$P \equiv P', P' \rightarrow Q', Q' \equiv Q \Rightarrow P \equiv Q$
$m[n[out\ m.P \mid Q] \mid R \rightarrow n[P \mid Q] \mid m[R]$	$P \rightarrow Q \Rightarrow (vn)(P) \rightarrow (vn)(Q)$
$open\ n.P \mid n[Q] \rightarrow P \mid Q$	$P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]$
$\langle M \rangle \mid (x).P \rightarrow P\{M \rightarrow x\}$	$P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R$

Tableau 2.3: Règles de réduction du calcul ambient [43]

2.4. Approche algébrique pour sécuriser un système informatique

Dans cette section, nous allons présenter le travail de M. Adi et ses collaborateurs [46], définissant la syntaxe et la sémantique d'un calcul approprié pour la spécification de la sécurité des systèmes informatiques avec les comportements des composants du système incluant sa protection à travers le comportement de l'intrus. Les auteurs modélisent les composants du système en termes de processus et décrivent les interactions de processus et l'interaction avec un intrus potentiel.

2.4.1. Syntaxe

La syntaxe de l'approche définit les modules requis pour un renforcement de sécurité dans un système informatique via des clefs de sécurité et elle est présentée dans un tableau 2.4. 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 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 ce calcul [46] . L'opérateur de concaténation″.″ décrit le comportement séquentiel d'un processus. Le terme ACTION est utilisé pour décrire l'exécution des capacités de processus.

Nous présentons ci-dessous l'ensemble de processus 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 :** $^{k,k'}_{n}[P]$ 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 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 :** $\boldsymbol{req}^{x}_{n,t}$ 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 :** $\boldsymbol{pub}^{x}_{n,t}$ se rapporte à la publication de clefs d'accès ou de sortie, selon la valeur du paramètre t.

• **Mouvement :** $\boldsymbol{mov}^{k}_{n}$ 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.

• **Choix compositionnel:** a ⊕ 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** Π **b** permet à l'´evolution d'un processus d'être définie comme choix entre deux processus composants, mais ne permet pas à l'environnement n'importe quel 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'. 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 de processus qui n'ont pas un comportement prévisibles. L'opérateur **exp** est utilisé pour imiter le comportement dynamique d'un intrus. Les divers aspects du comportement de processus sont présentés dans la section 2.4.2.

n	□	N	Nom
x	□	V	Variable
k,k'	□	K	Clefs de sécurité
t	□	{e, s}	Types de clefs
P, Q	::=		Processus
	|	0	Inactivité
	|	P | Q	Composition parallèle
	|	!P	Réplication
	|	$^{k,k'}_{n}[a.\,\mathrm{P}]$	Séquence d'ambient
a, b	::=		Capacités de processus
	|	$req^{x}_{n,t}$	Demande de clefs
	|	$pub^{x}_{n,t}$	Publication des clefs
	|	mov^{k}_{n}	Mouvement d'un processus
	|	*Exp*	Exploration

|	$a \oplus b$	Choix compositionnel
|	$a \Pi b$	Choix non-déterministe

Tableau 2.4 : Syntaxe de l'approche [46]

2.4.2. Sémantique

Les deux composantes de la sémantique opérationnelle du calcul sont : la congruence structurelle notée par $\equiv$ et la relation de réduction notée par $\rightarrow$ Le tableau 2.5 détaille la congruence structurelle en tant qu'équivalence de processus dans le contexte du calcul proposé dans [46] :

- La congruence séquentielle d'ambient (Séquence 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, $a \oplus b$ 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.

$P \equiv P$	Réflexivité
$P \equiv Q \Rightarrow Q \equiv P$	Symétrie
$P \equiv Q \wedge Q \equiv R \Rightarrow Q \equiv R$	Transitivité
$P \equiv Q \Rightarrow Q \mid R \equiv P \mid R$	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
$a \oplus b \equiv a \prod b \prod a.b \prod b.a$	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
$a \prod b \equiv b \prod a$	Commutativité choix non déterministe
$(a \prod b) \prod c \equiv a \prod (b \prod c)$	Associativité choix non déterministe
$(a \prod b).P \equiv a.P \prod b.P$	Choix d'exécution

Tableau 2.5: Règles de congruence structurelle de l'approche [46]

La relation de réduction définie dans le tableau 2.6 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).

Une demande de la clef appropriée du service approprié de publication a comme conséquence une communication de la valeur de la clef, règle (5).
La règle (6) 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.

P' → Q'	if P'≡ P, P → Q, Q ≡Q'	(1)
P | R → Q | R	if P → Q	(2)
mov_n^k.P | ${}^{k,k\prime}_{n}$[Q] → ${}^{k,k\prime}_{n}$[P | Q]		(3)
${}^{k,k\prime}_{n}$[$mov_n^{k\prime}$. P | Q] → P | ${}^{k,k\prime}_{n}$[Q]		(4)
$req_{n,t}^{x}$. P | ($pub_{n,t}^{k}$. Q) → P [$x \leftarrow$ k] | ($pub_{n,t}^{k}$. Q)		(5)
exp. P | ($pub_{n,t}^{k}$. Q) → ($mov_n^k \oplus exp$) . P | ($pub_{n,t}^{k}$. Q)		(6)

Tableau 2.6: Relations de réduction de l'approche [46]

L'approche de calcul proposée dans [46] permet d'identifier des menaces potentielles en combinant le $\oplus$ de mov_n^k et de l'expression *exp*.P de la règle (6) dans le tableau 2.6 avec le choix compositionnel et le choix d'exécution de la relation de congruence structurelle présentée dans le tableau 2.5.

Nous obtenons :

$$(mov_n^k \oplus exp).\mathrm{P} \equiv (mov_n^k.\mathrm{P}) \prod (exp.\mathrm{P}) \prod (mov_n^k.\, exp.\mathrm{P}) \prod (exp.\, mov_n^k.\, \mathrm{P})$$

a) exp.P \| ($pub_{n,t}^{k}$. Q) \| ${}^{k,k\prime}_{n}$[R] → ($pub_{n,t}^{k}$. Q) \| ${}^{k,k\prime}_{n}$[R \| exp. P]

b) ${}^{k,k\prime}_{n}$[exp. P] \| ($pub_{n,t}^{k}$. Q) \| [R] → exp.P \| ${}^{k,k'}_{n}\left[\left(pub_{n,t}^{k} . \mathrm{Q}\right)\middle| R\right]$

Figure 2.1 : Capacités d'exploration de l'intrus dans le réseau [46]

Si l'intrus choisit, par exemple, la troisième option ($mov_n^k.\ exp$.P), alors le résultat est un mouvement intérieur ou extérieur d'un ambient n, dépendant de la valeur de t : e ou s. Ce sont les scénarios illustrés dans la figure 2.1.

De plus, si la clé publique est employée pour la protection d'ambient, le processus d'intrus peut entrer ou sortir comme il veut (aucune restriction sur l'utilisation des clés) : les scénarios a) et b) sur la figure 2.1 illustrent ce fait.

2.5. Conclusion

Dans ce chapitre, nous avons présenté un bref aperçu des méthodes de spécification formelle, les langages les plus utilisés et les trois algèbres de processus suivantes : *CCS, π-Calcul* et le *calcul ambient* avec plus de détails pour ce dernier. Ces trois algèbres peuvent être utilisées pour modéliser le comportement des composants réseau.

Nous avons clos ce chapitre par présenter l'approche [46] basée sur le calcul ambient pour la spécification de la sécurité des systèmes, qui consiste en la modélisation des composants du système en termes de processus et la définition des interactions de processus et l'interaction avec un intrus potentiel.

Dans le prochain chapitre, nous présenterons la modélisation d'une des attaques DDoS présentée précédemment dans le premier chapitre et en proposer des solutions pour sécuriser le réseau.

Chapitre 3

Etude de cas

3.1. Introduction

L'objectif du présent chapitre est de modéliser une attaque DDoS, dont le principe est cité dans la section 1.4.

Se basant sur des articles de travaux de recherches menés par des chercheurs dans les domaines liés à l'algèbre de processus, plus précisément au calcul ambient (Ambient Calculus) par l'introduction de concepts nouveaux, à savoir :

L'ambient, sous-ambients, capacités, opérateurs logiques, l'interaction avec l'environnement extérieur, propriétés, clés d'accès et de protection du réseau, syntaxe du calcul ambient, la sémantique opérationnelle du calcul ambient définie par deux relations : congruence entre processus (règles de congruence), relations de réduction (règles de réduction); et ce dans le but de spécification de sécurité du système et réseaux informatique et du comportement de l'intrus.

Ce présent chapitre présente notre cas d'étude et est organisé comme suit :
La section 3.2 donne la spécification et la modélisation du système, la section 3.3 définit le processus de l'intrus et enfin la section 3.4 présente quelques solutions pour sécuriser le système.

3.2. Spécification et modélisation du système

Notre étude de cas (exemple d'application), illustré dans la figure **3.1**, est un système simple représenté par deux zones logiques : Internet et un réseau appliquant un protocole de routage vulnérable à une attaque DDoS. Le réseau comprend trois serveurs broadcast S1, S2, S3 reliés à un ensemble de machines M, représentés respectivement par des ambients protégés s1, s2, s3 et m. Chaque ambient comporte un processus portant le même nom que la machine (S1, S2, S3 et M) et représentant les ressources de cette dernière.

Le réseau est lui aussi représenté par un ambient protégé nommé « r » et comporte un processus « R »

La hiérarchie des ambients et des sous-ambients reflète la topologie du réseau. Ainsi, l'ambient r est l'ambient conteneur et les ambients s1, s2, s3 et m sont ses sous-ambients.

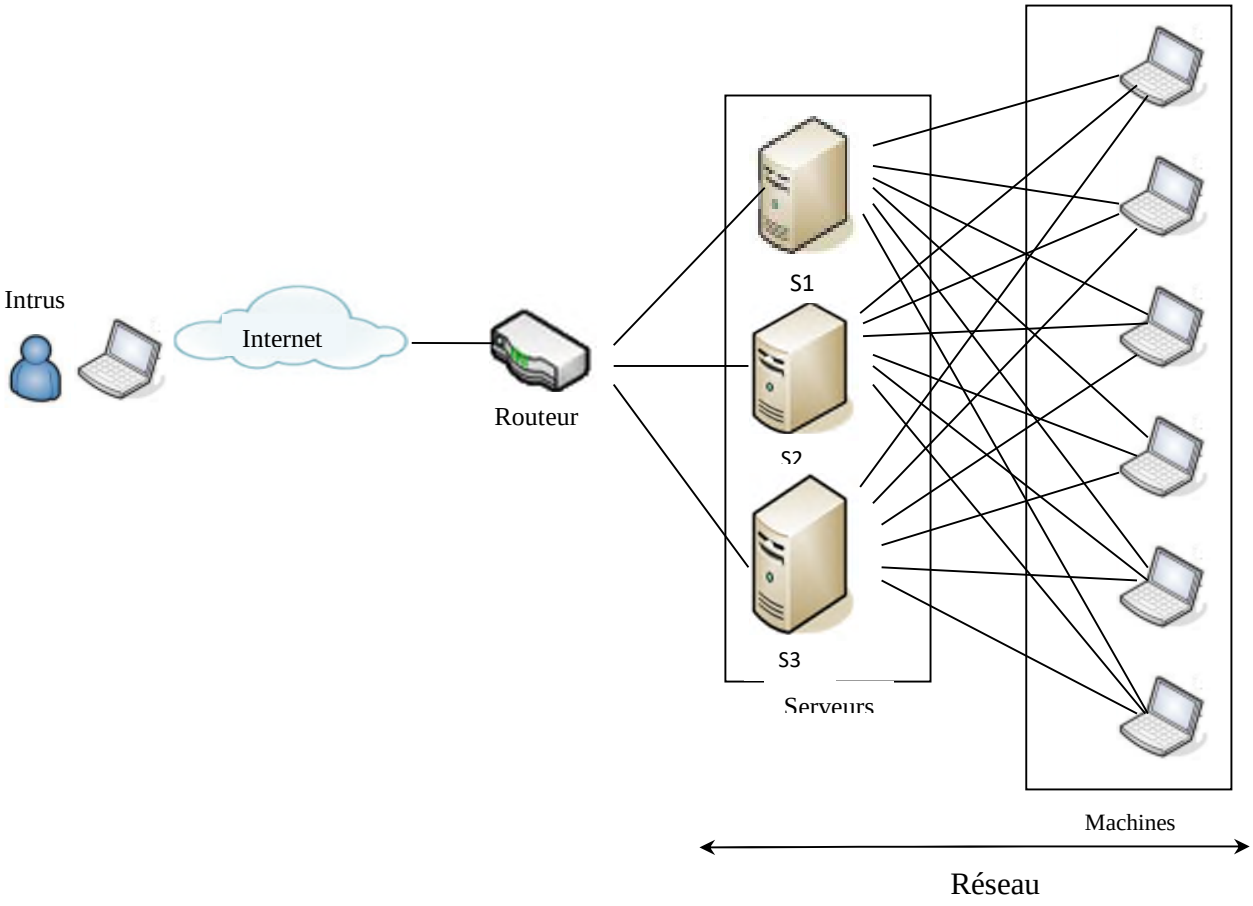

Figure 3.1: Spécification du système

La politique de sécurité impose l'implémentation d'une paire de clés de sécurité par machine : une clé d'accès (d'entrée) k et une clé de sortie k' et qu'il n'y a pas de restriction d'accès ou de sortie.

La spécification du système qui représente le réseau est décrite par le processus « S », comme suit :

$S = {}^{k,k\prime}_{r}[R \mid {}^{k,k\prime}_{s1}[S1 \mid !\,(pub^{k1}_{m,e}.0)] \mid {}^{k,k'}_{s2}[S2 \mid !\,(pub^{k1}_{m,e}.0)] \mid {}^{k,k\prime}_{s3}[S3 \mid !\,(pub^{k1}_{m,e}.0) \mid {}^{k1}_{m,e}[M]\,]$

3.3. Le processus de l'intrus

L'intrus représenté par son processus I se connecte au réseau pour explorer le système dans le but d'effectuer une attaque. L'opérateur *exp* modélise la manière dont l'intrus explore le système et gagne des capacités en termes d'action *mov*. Cet opérateur est le constructeur syntaxique qui imite le comportement non déterministe de l'intrus. Le processus de l'intrus connait la clef d'accès au réseau représenté par l'ambient *r* (qui est la clef publique *k*) et la clef d'accès k_1 à l'ambient *m* et il donne à l'intrus le choix d'employer ces clefs et de lancer son exploration. Il gagnera d'autres capacités de mouvement et de publication de clefs (sans faire de requêtes, contrairement au processus régulier), puisque l'intrus connait initialement les clefs *k*, *k'* et k_1

Le processus de l'intrus est modélisé comme suit :

I = $mov_r^k \oplus mov_{s1}^k \oplus mov_{s2}^k \oplus mov_{s3}^k \oplus exp.\mathrm{I}'$

Les étapes de l'exploration de l'intrus sont montrées dans le tableau suivant (Tableau 3.1) :

$$
\begin{aligned}
S \mid I \rightarrow\ & {}^{k,k\prime}_{r}[R \mid {}^{k,k\prime}_{s1}[S1 \mid !\,(pub^{k1}_{m,e}.0)] \mid {}^{k,k'}_{s2}[S2 \mid !\,(pub^{k1}_{m,e}.0)] \mid {}^{k,k\prime}_{s3}[S3 \mid !\,(pub^{k1}_{m,e}.0) \mid {}^{k1}_{m,e}[M]\,] \\
& \mid exp.I' \\
\rightarrow\ & {}^{k,k\prime}_{r}[R \mid {}^{k,k'}_{s1}[S1 \mid !\,(pub^{k1}_{m,e}.0) \mid exp.I'] \mid {}^{k,k'}_{s2}[S2 \mid !\,(pub^{k1}_{m,e}.0) \mid exp.I'] \mid \\
& {}^{k,k\prime}_{s3}[S3 \mid !\,(pub^{k1}_{m,e}.0) \mid exp.I'] \mid {}^{k1}_{m,e}[M]\,] \\
\rightarrow\ & {}^{k,k\prime}_{r}[R \mid {}^{k,k\prime}_{s1}[S1 \mid !\,(pub^{k1}_{m,e}.0) \mid (mov^{k}_{r} \oplus mov^{k}_{s1} \oplus exp).I'] \mid {}^{k,k'}_{s2}[S2 \mid !\,(pub^{k1}_{m,e}.0) \mid \\
& (mov^{k}_{r} \oplus mov^{k}_{s2} \oplus exp).I'] \mid {}^{k,k'}_{s3}[S3 \mid !\,(pub^{k1}_{m,e}.0) \mid (mov^{k}_{r} \oplus mov^{k}_{s3} \oplus exp).I'] \mid \\
& {}^{k1}_{m,e}[M]\,] \\
\rightarrow\ & {}^{k,k\prime}_{r}[R \mid {}^{k,k\prime}_{s1}[S1 \mid !\,(pub^{k1}_{m,e}.0) \mid (mov^{k}_{r}.mov^{k}_{s1}.exp).I'] \mid {}^{k,k'}_{s2}[S2 \mid !\,(pub^{k1}_{m,e}.0) \mid \\
& (mov^{k}_{r}.mov^{k}_{s2}.exp).I'] \mid {}^{k,k'}_{s3}[S3 \mid !\,(pub^{k1}_{m,e}.0) \mid (mov^{k}_{r}.mov^{k}_{s3}.exp).I'] \mid {}^{k1}_{m,e}[M]\,] \\
\rightarrow\ & {}^{k,k\prime}_{r}[R \mid {}^{k,k\prime}_{s1}[S1 \mid !\,(pub^{k1}_{m,e}.0)] \mid (mov^{k}_{s1}.exp).I' \mid {}^{k,k'}_{s2}[S2 \mid !\,(pub^{k1}_{m,e}.0)] \mid \\
& (mov^{k}_{s2}.exp).I' \mid {}^{k,k'}_{s3}[S3 \mid !\,(pub^{k1}_{m,e}.0)\,] \mid (mov^{k}_{s3}.exp).I' \mid {}^{k1}_{m,e}[M]\,] \\
\rightarrow\ & {}^{k,k\prime}_{r}[R \mid {}^{k,k\prime}_{s1}[S1 \mid !\,(pub^{k1}_{m,e}.0)] \mid {}^{k,k'}_{s2}[S2 \mid !\,(pub^{k1}_{m,e}.0)] \mid {}^{k,k\prime}_{s3}[S3 \mid !\,(pub^{k1}_{m,e}.0)\,] \mid \\
& {}^{k1}_{m,e}[exp.I' \mid M]\,]
\end{aligned}
$$

Tableau 3.1: Etapes d'exploration du processus de l'intrus

3.4. Sécurisation du système

Pour parer toute attaque DDoS sur le système, effectuée par un intrus ; il y lieu de l'empêcher d'intercepter les échanges de paquets et bloquer ou limiter l'exploration du système.

La problématique est comment transformer le système initial S en un système sécurisé S'.

Pour ce faire, plusieurs approches de solutions ont été proposées pour la sécurisation du système, toutes se basent sur la disponibilité de clefs d'accès et de sortie [46], [53], [59] dont la possession de celles-ci permet à une entité dans un système d'entrer (/de sortir) à (/de) l'ambient.

La première solution consiste à éviter la publication de clés qui étaient disponibles dans le réseau initial.

La deuxième solution consiste à ajouter de nouvelles paires de clefs privées aux ambients non protégés

Parmi les solutions proposées pour limiter l'effet de l'attaque, la configuration d'un Firewall (pare-feu) pour contrôler les paquets qui s'échangent dans le réseau. Nous avons exploité la solution qui consiste à créer un nouvel ambient f afin de pouvoir modéliser le nouveau composant (Firewall).

3.4.1. Nouvelles spécifications du système

Le nouveau composant ajouté (Firewall) est placé juste à la frontière du réseau avec l'extérieur, son rôle est de filtrer les requêtes arrivant de l'extérieur pour entrer dans le réseau. Le firewall est représenté par un ambient *f* comportant un processus F dont les clés d'accès et de sortie (*k, k'*) sont celles publiées au niveau des serveurs, comme l'illustre la figure 3.2

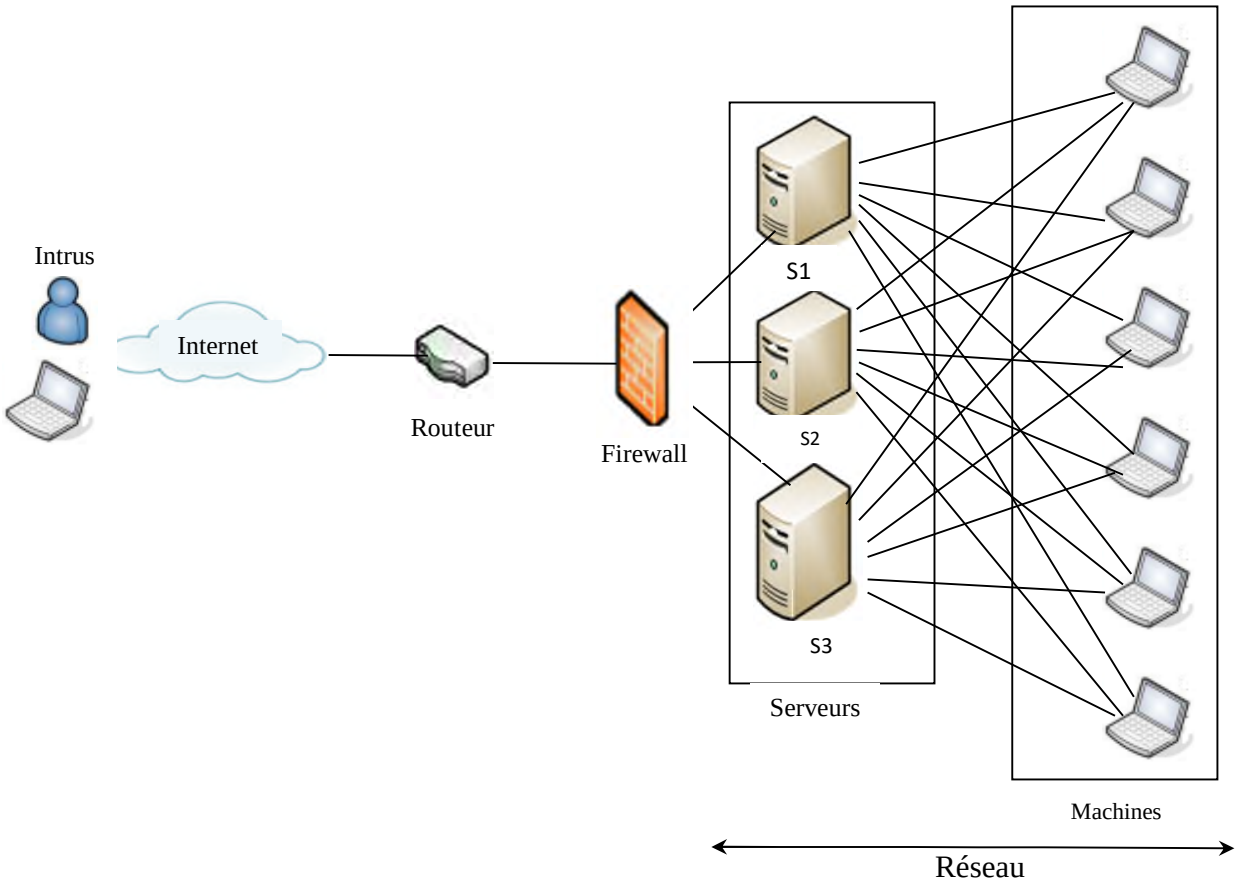

Figure 3.2: Sécurisation du réseau par un pare feu externe

Cette solution rend le réseau (serveurs et machines) plus sécurisé en bloquant l'accès à celui-ci et la nouvelle spécification S' du système représentant le réseau est définie comme suit :

$$S' = {}^{k,k\prime}_{f}[F] \mid {}^{ke,ks}_{r}[R \mid {}^{k,k\prime}_{s1}[S1 \mid !\,(pub^{ke}_{f,e}.0)] \mid {}^{k,k'}_{s2}[S2 \mid !\,(pub^{ke}_{f,e}.0)] \mid {}^{k,k\prime}_{s3}[S3 \mid !\,(pub^{ke}_{f,e}.0)\,] \mid$$

$${}^{k1}_{m,e}[exp.I' \mid M]\,]$$

3.5. Conclusion

Dans ce chapitre, nous avons spécifié le système représenté par un réseau et modélisé une attaque sur le réseau par un intrus et cela à l'aide de la méthode de calcul ambient [46] qui utilise plusieurs concepts de spécification tels que le contrôle d'accès pour les domaines (publication des clefs, mouvement d'un processus et l'exploration de l'intrus). Aussi, nous avons proposé une solution pour sécuriser le système en bloquant ou limitant l'accès à l'intrus de s'introduire dans le réseau : une solution basée sur l'utilisation de pare-feu qui est en mesure de distinguer les paquets d'attaque (requêtes) envoyés par des utilisateurs illégitimes et les filtre avant d'atteindre la victime.

Dans le chapitre qui suit, nous présenterons l'implémentation du système de spécification.

Chapitre 4

Implémentation

4.1. Introduction

L'objectif du présent chapitre est d'expliquer le déroulement pratique de l'implémentation du système de spécification présenté dans le chapitre 3.

Pour cela nous nous sommes inspirés de l'approche de l'application graphique de génération de spécification **GenSpec** réalisée par Marc Saint-Laurent [47]. Cette approche permet de décrire le comportement des divers composants d'un réseau avec leurs interactions. Elle est à même, de trouver comment un intrus pourrait exploiter ou contourner l'implémentation défectueuse de certaines politiques de sécurité.

Pour spécifier correctement les attaques, l'approche requiert que le réseau soit modélisé formellement en tenant compte de la topologie particulière d'un réseau donné.

L'application est ainsi répartie en trois étapes :

- La représentation graphique d'une topologie de réseau
- L'affectation des processus aux divers nœuds du réseau et la définition des liens entre eux
- La génération automatique de la spécification de la topologie

Dans ce présent chapitre, nous allons présenter la structure de l'application, puis nous expliquerons le déroulement du système de spécification (son implémentation).

4.2. Présentation de la structure de l'application

4.2.1. Objectifs

- ✓ Représentation graphique d'une topologie de réseau (modélisation)
- ✓ Définition des liens (interactions) entre les nœuds (composants) du réseau
- ✓ Définition des propriétés de tous les composants du réseau (nom, identifiants, processus, ambient, clés)
- ✓ Génération automatique de la spécification de la topologie du système.

4.2.2. Spécification des besoins

4.2.2.1. Génération de la spécification

Le but ultime de cette application est de générer l'expression algébrique qui décrit le comportement du réseau créé par l'utilisateur. Cette section expliquera comment cette génération doit être faite, à l'aide de l'exemple suivant :

$$^{k,k\prime}_{r}[\mathrm{R} \mid {}^{k,k\prime}_{s}[\mathrm{S} \mid !\,(pub^{k1}_{m,e}\,.0)] \mid {}^{k1}_{m,e}[\mathrm{M}]\,]$$

Figure 4.1: Exemple de spécification

Un ambient possède un nom de domaine, r par exemple. Il est protégé par les clés d'entrée (ou d'accès) k et de sortie k'. Ces clés suivent toujours le domaine et sont en exposants. L'intérieur de l'ambient est représenté par les grandes accolades, dans lesquelles on peut trouver soit des processus ou des sous-ambients. Il y a trois ambients dans cette expression r, s et m. Le premier est le parent des deux derniers, puisque s et m se trouvent à l'intérieur de r.

Les processus sont représentés par une lettre majuscule comme R , S ou M. Le 0 indique l'inactivité de processus. Le processus neutre 0 est précédé par une action *pub* de publication de clés d'accès à l'ambient *m*. Le « ! » dénote une réplication de ce processus. Une barre horizontale entre deux processus ou ambients permet d'exprimer qu'ils sont en parallèle.

On distingue quatre types d'actions : mov, pub, req et exp. Elles sont toujours appliquées à un processus, et sont placées devant lui. Ces actions ainsi que les processus sont séparés entre eux par un point. Leur format d'affichage varie, puisqu'ils prennent des arguments différents.

Pour faciliter son implémentation, l'application est en mesure de générer une spécification en mode texte, pour utilisation par d'autres applications. Le tableau qui suit explique la génération de la spécification affichée en mode interface graphique et en mode texte, représentant les éléments de la spécification avec leurs arguments.

Génération de la spécification			
Élément de la spécification	**Arguments**	**Spécification en mode interface graphique**	**Spécification en mode texte**
Ambient $m^{k_1,k_2}[\ ]$	m, le nom de l'ambient k_1, la clé d'entrée k_2, la clé de sortie	$m^{k_1,k_2}[\ ]$	m(k1, k2)[]
Process P	--	P	P
Actions pub et req	m, le nom d'un ambient $e\ ou\ s$, le type de clé (entrée ou sortie) k, la clé	$pub^{k}_{m,e}$ $req^{k}_{m,s}$	pub(m,e,k) req(m,s,k)
Action mov	m, le nom d'un ambient k, la clé	mov^{k}_{m}	mov(m,k)
Action exp	--	exp	Exp
Composition parallèle	Processes et/ou ambients	|	|
Inactivité		0	0

Tableau 4.1 : Génération de la spécification

Ainsi, la représentation textuelle de l'exemple précédent est comme suit :

r(k, k')[R | s(k, k') [S | !(pub(m,e,k1).0)] | m(e,k1)[M]]

Figure 4.2: Exemple de spécification en mode texte

4.2.2.2. Structure graphique de l'application

La fenêtre principale de l'application est divisée en 2 régions distinctes : la première contient la zone de création de la topologie, et le bas affiche la spécification générée (Figure 4.3).

Dans la zone de création de la topologie, l'utilisateur pourra manipuler deux types d'objets : les nœuds, et les liens entre eux. Neuf différents types de nœuds sont pris en

charge : pont, ordinateur, pare-feu, concentrateur, appareil mobile, routeur, sous-réseau, commutateur et point d'accès sans fil.

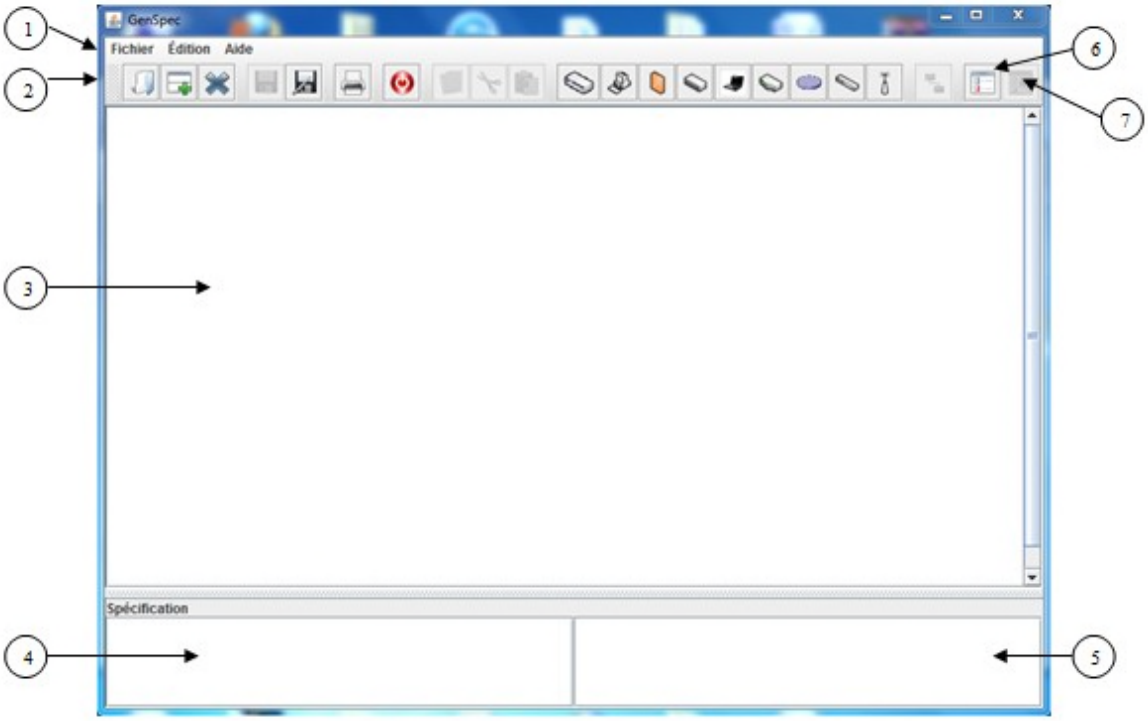

Figure 4.3: La fenêtre principale [47]

(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.

(3) – La zone qui représente graphiquement la topologie du réseau (Fig. 4.4).

(4) – La zone qui affiche la spécification de la topologie dans son état actuel.

(5) - La zone qui permet d'obtenir la spécification en mode texte.

(6) - Le bouton qui permet de voir et modifier la spécification.

(7) - Le bouton qui permet de lancer la réduction

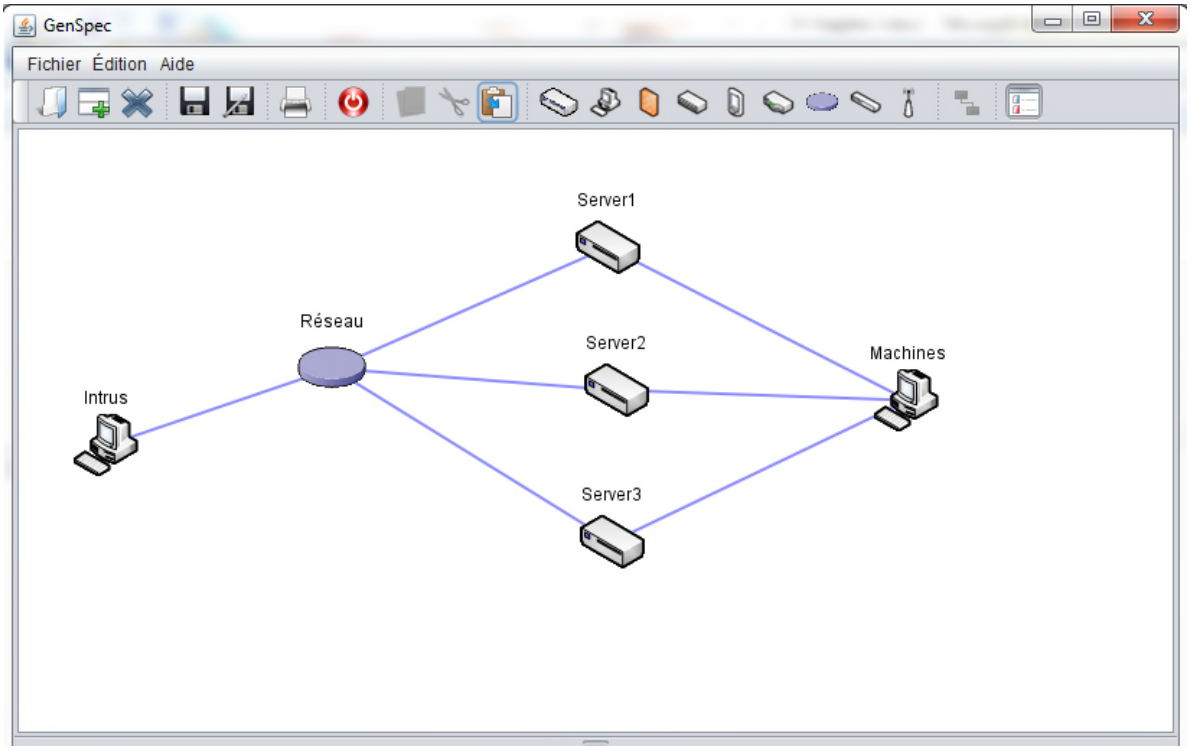

Figure 4.4 : Exemple d'une zone de création de topologie [47]

En plus, l'utilisateur pourra modifier les propriétés des objets existants. L'application devra donc fournir une vue qui contient toutes les informations de l'ambient et du processus, et qui laisse l'utilisateur en mesure de faire ses changements.

Dans la zone en bas, il y a affichage de la spécification de la topologie du réseau dans son état actuel.

4.2.2.3. Fonctions de l'application

La partie la plus visible de l'application sera la zone de création de topologie. La première fonction qu'elle devra assurer est la création de nouveaux nœuds (Voir l'exemple de la figure 4.4).

4.2.3. Environnement et outils de développement (Artifices de programmation)

Son but est de préciser les moyens retenus pour répondre aux besoins énoncés dans l'analyse fonctionnelle (spécification des besoins). Elle comprend les outils utilisés pour développer l'application, la description de l'architecture conceptuelle du logiciel et sa traduction en code source et enfin les moyens pour la mise en production de ce logiciel.

- ✓ **JAVA :** le langage de programmation
- ✓ **JDK (Java Development Kit) :** l'environnement dans lequel le code Java est compilé pour être transformé en ByteCode afin que la JVM (Java Virtual Machine) de Java puisse l'interpréter
- ✓ **Environnement de développement Eclipse IDE**

4.3. Conclusion

Dans ce chapitre, nous avons présenté la structure de l'application GenSpec qui se base sur le calcul ambient et permettant la modélisation du système et la génération de la spécification qui décrit le comportement du réseau créé par l'utilisateur.

Conclusion générale

Conclusion générale

Nous avons présenté dans ce mémoire dans le premier chapitre une description des attaques par déni de service DOS et les attaques par déni de service distribué (DDoS) les plus fréquentes dans les systèmes informatiques,

Nous avons dressé, dans le deuxième chapitre, un état de l'art des méthodes de modélisation et spécification formelles (approche algébrique de processus) de système informatique,

Par la suite, nous avons présenté, au troisième chapitre, une étude de cas qui consistait en la modélisation et la spécification du comportement d'une attaque par déni de service distribué (DDoS) à l'aide de la méthode formelle [46], suivie par application d'une solution de sécurité contrant cette attaque ,

Le quatrième chapitre est consacré à l'application d'un logiciel développé [47] sur l'exemple étudié.

Comme perspectives :

- ✓ Essayer de mette en œuvre l'approche dans un système réel donné
- ✓ Tester l'approche avec d'autres sources de données pour éviter les problèmes de similitudes de quelques attaques
- ✓ Améliorer l'approche et l'adapter pour modéliser les différentes attaques sur différents réseaux existants.

Références

Références

1. Thierry Evangelista, *Les Systèmes de détection d'intrusions informatiques (IDS),* DUNOD Paris 2004,

2. ACISSI, Collection EPSILON *Sécurité informatique Ethical Hacking «Apprendre l'attaque pour mieux se défendre »* (4ème édition) Editions ENI France 2015,

3. Marie-Claude GAUDEL, Bruno MARRE, Françoise SCHLIENGER, Gilles BERNOT *Précis de* génie logiciel MASSON Paris 1996,

4. YAHIAOUI Soraya, OUATAH Saida *Approche algébrique pour la prévention d'intrusions* Mémoire de Master, Université de Béjaia 2009,

5. ABDOUNE Mohamed Oulhadj, HAMIDOUCHE Ahcène *Approche algébrique pour la prévention d'intrusions (attaque Blackhole)* Mémoire de Master, Université de Béjaia 2014,

6. KEMICHE Mokrane *Détection de nouvelles attaques dans un système informatique et contre-mesure* Mémoire de Magister, Université de Béjaia 2014,

7. H. SOUILAH *Sécurité contre les attaques DOS et DDOS dans les réseaux IEEE 802.11* Mémoire de Magister, Université de Béjaia 2014,

8. N.BOUREZANE, A. TOUNANI *Cloud-based web application firewall* Mémoire de Master, Université de Béjaia 2013,

9. D. Barbara, N. Wu, S. Jajodia *Detecting novel network intrusions using bayes estimators.* In first SIAM international conference on Data mining (SDM 2001), 2001

10. J.P Anderson *Computer security threat monitoring and surveillance Technical report,* James P. Anderson Co, Fort Washington, Pennsylvania 1980,

11. C. Langini & S. Rahimi *Soft computing intrusion detection : the stat of threat – 1(2), 133-145* , 2010,

12. Debar, Hervé, Dacier, Marc, Wespi, Adreas *A revised taxonomy intrusion-dtection system.* In Annales des Télécommunications, Springer-Verlag (P.361_378-3), 2000,

13. Site www.rivalhost.com, consulté Fév. 2017,

14. ST. Zargar, J. Joshi, D. Tipper *A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks,* Article de recherche, IEEE Communications Society, 2013

15. SM. Specht, RB. Lee *Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasure,* Article de recherche, ISCA PDCS, 2004

16. Site www.homputersecurity.com , *Comment fonctionnent les attaques DDoS ?*, publié le 15/09/2016 par mdestroy

17. Intro to CyberSecurity-FR 1216 de Cisco Networking Academy (NetAcad), 2017

18. Site www.nttcomsecurity.com, *Déni de service distribute (DDoS)*, publié 2014

19. Site www.ssi.gouv.fr, ANSSI, *Comprendre et anticiper les attaques DDoS* , Mars 2015

20. Site www.OVH.com *Qu'est ce qu'une attaque DDoS et comment s'en protéger*, 2017

21. C. JABOU, M.SCHILLINGS, A.HANTAS *Détection d'anomalies sur le réseau*, TER-Université Paris Descartes, 2009

22. H. DIAB *Evaluation de méthodes formelles de spécification*, Mémoire (M. Sc), Univ Sherbrooke (Canada), 1999

23. F. BENALI *Modélisation et classification automatique des informations de sécurité*, Thèse Doctorat, INSA Lyon, 2009

24. S. AMOURI, Y. AKIK *Approche algébrique pour la prévention d'intrusions*, Univ Béjaia, 2013

25. T. MECHRI *Approche algébrique pour la sécurisation des réseaux informatiques*, Mémoire, Université Laval, Québec, 2007

26. A. CHARLIER *Introduction à la sécurité informatique*, Paris academy of computer science of Supinfo projects, 30/05/2006 Supinfo Paris

27. Marfall N'Diaga Fall, *Sécurisation formelle et optimisée de réseaux informatiques*, Mémoire (M. Sc) Univ Laval (Québec), 2010

28. S. LAFRANE *Spécification et validation de protocoles de sécurité*, thèse Ph. D Génie Info, Montréal, 2005

29. BOURKACHE *un IDS réparti basé sur une société d'agents intelligents*, Magister , Univ. Boumerdés, 2007

30. F. Lone Sang *Protection des systèmes informatiques contre les attaques par entrées-Sorties*, thèse Doctorat, Univ. Toulouse, 2012

31. A. TERRASA *Modélisation et spécification formelles des logiciels*, INF 3143, (UQAM), 2017

32. Site searchsecurity.techtarget.com *What is distributed denial of service (DDoS) attack*

33. Site www.verisign.com, *Types d'attaques DDoS*, 2017

34. D. VALOIS, C. LORENS, L. LEVIER *Tableaux de bord de la sécurité réseaux*, 2ème édition Eyrolles, Oct. 2006

35. ISO/AFNOR, Dictionnaire de Génie Logiciel AFNOR, 1997

36. B. Marre, F. Schlienger, M.C Gaudel et G. Bernot *Précis de Génie Logiciel*, 1996

37. A. LACASSE *Approche algébrique pour la prévention d'intrusions, Mémoire (M. Sc), Univ. Laval, Québec, 2006*

38. A. CORMIER, *Modélisation et sécurité des réseaux*, Mémoire (M.Sc) Univ. Laval, Québec, 2007

39. R. Milne *Communication and concurrency*, Prentice Hall International (UK) Ltd, Hcrtfordshirc, UK, 1995

40. R. Milner *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge University Press, June 1999

41. 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*

42. *J.A. Bergstra and J.W. Klop Process algebra for synchronous communication, Technical Report 60*, 1984

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

44. J.CM. Baeten *A brief history of process algebra*, Theoritical Computer Science, 335 (2-3): 131-146, 2005

45. M. LANGAR *Cadre algébrique pour le renforcement de politique de sécurité sur des systèmes concurrents par réécriture automatique de programmes*, Thèse Doctorat, Faculté des Sciences et Génie, Univ. Laval, Québec, 2010

46. K. ADI, L. HAMZA, L. PENE *Formal modeling for security behavior analysis of computer systems*, In: Proceedings of the 2008 International MCETECH Conference on e-Technologies, IEEE Computer Society pp 49-59, 2008

47. M.Saint-Laurent, *GenSpec : Rapport final « Environnement graphique pour la spécification de propriétés des systèmes informatiques »*, Univ. du Québec en Ouataouais, Québec, 2009

48. L. Hamza, K. Adi et L. Pene *Approche algébrique pour le renforcement de la politique de sécurité dans les systèmes informatiques*, Rapport de recherche, JDI'10, Univ. Béjaia, Oct. 2010

49. L. Pene and K. Adi *Calculus for distributed firewall specification and verification.* In Proceedings of the 5th International Conference on Software Methodologies. SoMeT'06, pages 301–315. IOS Press, 2006

50. G. Ferrari, E. Moggi, and R. Pugliese *Guardians for ambient-based monitoring* FWAN: Foundations of Wide Area Network Computing, number 66, Elsevier Science, 2002

51. D. Hirschkoff, E. Lozes and D. Sangiorgi *On the expressiveness of the ambient logic*. In Logical Methods in Computer Science, 2006

52. A. Lacasse, M. Mejri and B. Ktari. Formal *implementation of network security policies*. In The Second Annual Conference on Privacy, Security and Trust (PST04). New Brunswick Canada, 2004

53. F. Nielson, H.R. Nielson, R.R. Hansen and J.G. Jensen *Validating firewalls in mobile ambients*. In International Conference on Concurrency Theory, pages 463–477, 1999

54. D. Teller, P. Zimmer and D. Hirschkoff. *Using ambients to control resources*. In: In Proceedings of Concur'02, LNCS 2421, Springer, 2002

55. Horstmann, Cay s. et Gary Cornell. *Core Java 2: Volume 1, Fundementals, Sixth Edition*. Coll. The Sun Microsystems Press Java Series, s.l.: Prentice Hall, 752 p, 2002

56. Larman Craig. *UML 2 et les design patterns. 3e édition*, Paris : Pearson Education France, 655 p, 2005.

57. G. Mathieu. *In Décorateur*. En ligne: http://www.design-patterns.fr/Decorateur.html, Modifié Oct. 2011

58. F. SAAOUI, N. ADRAR *Approche algébrique pour la modélisation d'attaques déni de service (DDoS)*, Mémoire de Master, U. Béjaia, 2009.

59. A. El Kabbal *« Un système de types pour l'analyse des pare-feux »*, Mémoire, Univ. du Québec en Ouataouais, Québec, 2005

60. S. Graine *« UML 2 pour une modélisation orientée objet »*, Edition l'Abeille, Tizi-Ouzou, 2009

61. Site www.cisco.com , *Distributed Denial of Service Attacks - The Internet Protocol Journal - Volume 7, Number 4*, Charalampos Patrikakis, Michalis Masikos, and Olga Zouraraki, National Technical University of Athens, Juin 2004

Table des matières

Table des matières

Chapitre 3 : Etude de cas

Chapitre 4 : Implémentation

RESUME

De plus en plus de systèmes subissent des attaques par déni de service distribué (DDoS), et est visée par ces attaques toute entité disposant d'une infrastructure réseau connectée à Internet. Les attaques DDoS se muent constamment pour prendre de nouvelles formes ; c'est pourquoi les réseaux informatiques souffrent toujours de nouvelles attaques et intrusions alors que les recherches dans ce domaine sont multiples.

Aujourd'hui, les chercheurs basent leurs recherches sur l'aspect mathématique et le résultat est de définir des méthodes formelles et qui sont applicables.

La fréquence et sévérité croissantes des attaques DDoS modifient rapidement l'aspect de la sécurité du réseau et des systèmes informatiques. C'est pourquoi, il fallait en amont, identifier les risques d'attaques DDoS et définir une solution, des installations et une configuration adaptées.

Les entreprises doivent donc intégrer la variable de mutation des attaques à leur stratégie de sécurité de l'information, leurs politiques, leurs niveaux de service et leur exposition au risque. Elles seront ainsi en mesure de faire évoluer leur solution pour assurer une prévention et une réponse adaptées.

MOTS-CLEFS : Attaque DoS, attaque DDoS, Spécification du système

ABSTRACT

More and more systems are undergoing Distributed Denial of Service (DDoS) attacks, targeting any entity that has a network infrastructure connected to the Internet. DDoS attacks are constantly changing into new forms; which is why computer networks are still suffering from new attacks and intrusions while research in this domain is multiple.

Today, researchers base their research on the mathematical aspect and the result is to define formal methods that are applicable.

The increasing frequency and severity of DDoS attacks quickly changes the security aspect of network and computer systems. Therefore, it was necessary upstream, identify risks of DDoS attacks and define a suitable solution, installations and configuration.

Therefore, companies must integrate the mutation of attack variable into their security information strategy, policies, service levels and risk exposure. They will thus be able to evolve their solution to ensure adapted prevention and response.

KEYWORDS: DoS Attack, DDoS Attack, System Specification