N° d'ordre : 22/2012-M/IN

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

Université des **S**ciences et de la **T**echnologie **H**ouari **B**oumediene

Faculté d'**E**lectronique et d'**I**nformatique

MEMOIRE

Présenté pour l'obtention du diplôme de **MAGISTER**

En**: Informatique**

Spécialité**: Systèmes Intelligents et Ingénierie du Logiciel**

Par: **Mme MERAR Naouel Née OUROUA**

THEME

UTILISATION DES MODÈLES FORMELS DANS LA SÉCURITÉ INFORMATIQUE : MODÉLISATION DE L'ATTAQUE SYN/TCP AVEC LES RÉSEAUX DE PETRI

Soutenu Publiquement le 29/05/2012, devant le jury composé de :

Mme	H. KHELLAF	Maître de Conférences /A, à l'USTHB	**Présidente**
Mme	M.BOUKALA	Professeur, à l'USTHB	**Directrice de mémoire**
M.	O. NOUALI	Maître de Recherche/A, au CERIST	**Examinateur**
M.	D. BAHLOUL	Maître de Conférences /A, à l'USTHB	**Examinateur**
Melle	N. SALMI	Maître de Conférences/A, à l'USTHB	**Invitée**

Dédicaces

Je Dédie Ce Travail A

Mes très chers parents, ma petite famille (mon mari et mon cher fils), ma sœur, mes frères et mon beau frère qui m'ont toujours encouragé, ainsi que ma belle famille, surtout à la mémoire de ma chère belle mère que dieu ai son âme.

Remerciements

Merci mon dieu pour toutes les faveurs.

Je tiens à exprimer ma reconnaissance et mes vifs remerciements à Madame le Professeur ***Malika BOUKALA-IOUALALEN****, ma directrice de mémoire, pour m'avoir proposé ce sujet et diriger mon travail et surtout pour la confiance qu'elle m'a toujours témoignée.*

Je souhaite aussi exprimer ma gratitude à ***Mme Haja Faiza HANED-KHELLAF*** *pour l'honneur qu'elle me fait en acceptant de présider le jury de mon mémoire. Je remercie aussi* ***Mr Djilalai BAHLOUL****,* ***Mr O. Nouali et Melle Nabila SALMI*** *pour avoir accepté de juger mon travail.*

Grand merci à mon mari ***Salem MERAR*** *pour son aide précieuse. Je remercie aussi mon petit* ***Abderrahmane*** *pour sa patience. Mes sincères reconnaissances vont à* ***mes parents ma sœur****,* ***son mari*** *et* ***mes frères*** *pour leur aide et soutien absolu.*

Je désire aussi remercier tous mes amis et collègues notamment ceux de ma promo, et tous ceux qui de prés ou de loin, ont aidé à l'aboutissement de ce modeste travail, plus particulièrement Melle ***Samira TALEB****.*

Table des Matières

Chapitre V : Le Modèle PetriDoS

Liste des figures et tableaux

Résumé

Les progrès technologiques, la disponibilité des ordinateurs et de réseaux informatiques puissants, ainsi que la nécessité d'augmenter l'efficacité des opérations, ont amené l'ensemble des organisations à privilégier les systèmes informatiques. Cependant, bien qu'ils intègrent la dimension sécuritaire, les systèmes informatiques ne sont pas dépourvus d'erreurs qui les exposent à divers risques de sécurité : accès non autorisés, vol ou perte d'informations confidentielles, sabotage, non disponibilité de données…et la liste ne cesse de s'allonger. L'information constitue l'une des principales richesses de plusieurs organisations. Il est donc essentiel de pouvoir protéger cette richesse. Ainsi une bonne maitrise de la conception des systèmes informatiques par des méthodes formelles, devient un impératif afin de prévenir ces risques lors des phases amont de la conception, avant même de procéder à la réalisation et à l'exploitation. C'est pourquoi, plusieurs modèles basés sur les méthodes formelles ont été proposés. Ces modèles dressent une représentation formelle des politiques de sécurité et de leur fonctionnement. Ils permettent de prouver des propriétés sur la sécurité du système.

Dans la littérature on trouve une diversité de modèles formels se rapportant à la sécurité informatique, il y a des modèles qui s'intéressent au *contrôle d'accès,* d'autres modèles s'intéressent aux *protocoles de sécurité.* On trouve aussi des modèles qui s'intéressent aux *attaques.*

Ces modèles utilisent des formalismes différents, à savoir, l'algèbre, l'algèbre de processus, la logique modale, les arbres, et le réseau de Petri et ses dérivés.

Le but de notre travail est de faire un état de l'art sur les modèles formels existants pour *la modélisation des propriétés liées à la sécurité informatique,* à travers la modélisation des différents aspects de celle-ci, à savoir les protocoles de sécurité, les contrôles d'accès et les attaques, *l'évaluation quantitative de la sécurité* a notamment été abordée dans les modèles d'attaque fournissant des mesures quantitatives. Nous avons fait une comparaison sur les différents formalismes utilisés pour chaque aspect de la sécurit*é.* Nous nous sommes intéressés aux modèles utilisant les réseaux de Petri comme formalisme de modélisation.

Comme le domaine de la sécurité est vaste, nous avons choisi d'intervenir dans le domaine des attaques. La disponibilité étant une des propriétés de la sécurité, cette dernière est violée par des attaques de type Déni de Service, nous avons alors proposé un modèle formel qui se base sur les réseaux de Petri stochastiques et déterministes colorés qui décrit le processus de l'attaque TCP/SYN en vue de calculer les performances du système sous ce type d'attaque.

Mots clés : *sécurité informatique, modèles formels, protocole de sécurité, contrôle d'accès, attaques informatiques, réseaux de Petri stochastiques et déterministes, attaques DOS, attaque TCP/SYN*

Summary

The technological advancements, the availability of the computers and powerful data-processing networks, as well as the need for increasing the effectiveness of the operations, led the whole of the organizations to privilege the computing systems. However, although they integrate safety dimension, the computing systems are not deprived of errors which expose them at the various risks of safety: access not authorized, stolen or loss of confidential information, sabotage, not availability of data… and lists it does not cease lengthening. Information constitutes one of the main wealth of several organizations. It is thus essential to be able to protect this richness. Thus a good control of the design of the computing systems by formal methods becomes a requirement in order to prevent these risks at the time of the phases upstream of the design, before even proceeding to the realization and the exploitation. This is why, several models based on the formal methods were proposed. These models draw up a formal representation of the security policies and their operation. They make it possible to prove properties on the safety of the system.

In the literature one finds a diversity of formal models referring to the computer security, there are models which are interested in the *access control*, other models are interested in the *cryptographic protocols.* we find also models which are interested in the *attacks.*

These models use different formalisms, namely, the algebra, the algebra of process, modal logic, the trees, and the Petri net and his derivatives.

The goal of our work is to make a state of the art on the existing formal models for the *modelling of the properties related to the computer security*, through the modelling of the various aspects from this one, namely the protocols of safety, the access controls and the attacks, the *quantitative evaluation of safety* was in particular approached in the models of attack providing of quantitative measurements. We made a comparison on the various formalisms used for each aspect of safety. We were interested in the models using the Petri nets like formalism of modelling.

As the field of safety is vast, we chose to intervene in the field of the attacks. The availability being one of the properties of safety, the latter is violated by attacks of the Denial-of-service type; we then proposed a formal model which is based on the Petri nets stochastic and deterministic coloured which describes the process of attack TCP/SYN in order to calculate the performances of the system under this type of attack.

Key words: *computer security, formal models, cryptographic protocol, access control, computer attacks, stochastic and deterministic Petri nets, DOS attacks , TCP/SYN attack.*

Introduction générale

INTRODUCTION GENERALE

Aujourd'hui, les progrès technologiques, la disponibilité des ordinateurs et des réseaux informatiques puissants, ainsi que la nécessité d'augmenter l'efficacité des opérations, ont amené l'ensemble des organisations à privilégier les systèmes informatiques. Cependant, bien qu'ils intègrent la dimension sécuritaire, les systèmes informatiques ne sont pas dépourvus d'erreurs qui les exposent à divers risques de sécurité : accès non autorisés, vol ou perte d'informations confidentielles, sabotage, …et la liste ne cesse de s'allonger. L'information constitue l'une des principales richesses de plusieurs organisations. Il est donc essentiel de pouvoir protéger cette richesse. En effet, une simple faille de sécurité dans le système peut entrainer une divulgation d'informations, une altération de données, un déni de service et une utilisation illicite des ressources. Les conséquences sont parfois très lourdes : des renseignements inexacts sur un dossier médical ou un fichier de crédit peuvent ruiner des réputations ou occasionner un mauvais diagnostic de maladie. Des infections par des virus peuvent causer des pertes en vie humaines, comme pour le crash du vol « Spanair JK-5022 », qui avait entraîné la mort de 154 passagers et membres d'équipage en août 2008 (selon le quotidien espagnol El Pais), à cause du système informatique censé surveiller les problèmes techniques de l'avion, et qui était infecté par un cheval de Troie qui a bloqué les alertes de sécurité. Les ordinateurs n'ont pas pu alerter l'équipage des problèmes par un signal sonore, comme c'est le cas habituellement.

Ainsi une bonne maitrise de la conception de ces systèmes à l'aide de modèles formels, devient un impératif afin de prévenir ces risques lors des phases amont de la conception, avant même de procéder à la réalisation et à l'exploitation. Ces modèles dressent une représentation formelle des politiques de sécurité et de leur fonctionnement. Ils permettent de prouver des propriétés sur la sécurité du système.

Pour avoir confiance en la sécurité du système informatique, les utilisateurs ont besoin d'éléments de comparaison leur permettant de choisir les systèmes qui répondent le mieux à leurs besoins. Dans ce but, des critères d'évaluation comme les TCSEC (Trusted Computer System Evaluation Criteria) et les ITSEC (Information Technology Security Evaluation Criteria), ont été développés, ce sont des normes de sécurité internationalement approuvées. Ces critères offrent une classification des niveaux de sécurité, pour les niveaux d'assurance d'évaluation élevés, ils imposent l'utilisation de méthodes formelles dans la spécification des systèmes ainsi que la démonstration par des preuves que la spécification formelle assure les

propriétés de sécurité. Ces contraintes ont favorisé les recherches dans le cadre de la formalisation du concept de sécurité.
Les mesures résultant de l'évaluation par les critères ont été définies comme des mesures qualitatives (Dacier, 1994), une mesure qualitative est une mesure dont les valeurs traduisent l'appartenance à une classe (Vache, 2008). Les mesures qualitatives ne permettent pas de comparer un même système à deux instants d'évaluation différents. Pour cela des études cherchant à étudier la quantification de la sécurité en mode opérationnel ont vu le jour.

Les activités en sécurité informatique se sont déroulées suivant plusieurs directions : celle de la *modélisation* et de l'expression formelle des propriétés, des règlements et des politiques de sécurité, celle du *développement* de systèmes ou sous-systèmes permettant de garantir la sécurité, et celle de la *vérification*, *validation* des propriétés de sécurité et de l'*évaluation* de systèmes existants.
Plusieurs modèles de sécurité ont été développés, se rapportant à divers politiques de sécurité (discrétionnaires, obligatoires et basées sur les rôles) et aux règles du contrôle d'accès de ces politiques. Une diversité de modèles et langages (langage logique, algèbre de processus, réseaux de Petri) ont été proposés pour formaliser des modèles de contrôle d'accès, dans le but de prouver que les propriétés ou les règles de la politique de sécurité sont bien renforcées dans le système, en bénéficiant des différentes techniques et outils offerts par le formalisme. Un autre aspect de la sécurité informatique a suscité l'intérêt des chercheurs, il s'agit des protocoles de sécurité ou protocoles cryptographiques, ces protocoles sont utilisés pour assurer des services liés à la protection de l'information, dont les principaux sont la confidentialité, l'authenticité et l'intégrité, là aussi une variété de modèles a été proposée, les principaux sont les modèles orientés algèbres, les modèles orientées logiques et les modèles qui se basent sur les réseaux de Petri. Le but de cette modélisation est de prouver la sécurité des protocoles cryptographiques contre les attaques. L'attaque informatique est le troisième aspect modélisé dans le domaine de la sécurité informatique. Différents formalismes (arbre des fautes, arbre des attaques, file d'attente, chaine de Markov, graphe d'attaque et réseaux de Petri) ont été utilisés pour décrire les scénarios possibles d'une attaque contre un système ou réseau informatique, dans le but d'analyser l'attaque et évaluer son coût ou son impact et développer ainsi des contremesures plus efficaces, ou l'exploiter pour la construction des systèmes de détection des intrusions.

Le but de ce travail est de faire un état de l'art sur les modèles formels existants et qui se

basent sur différents formalismes pour modéliser les propriétés de la sécurité informatique telles que : la confidentialité, l'intégrité et la disponibilité, et la modélisation de différents aspects de la sécurité à savoir les protocoles de sécurité, les contrôles d'accès et les attaques. Nous nous sommes intéressés particulièrement aux modèles utilisant les réseaux de Petri comme formalisme de modélisation. Nous avons abordé aussi des modèles formels relatifs à l'évaluation quantitative de la sécurité à travers les modèles d'attaque.

Comme le domaine de la sécurité est vaste, nous avons choisi d'intervenir dans le domaine des attaques. La disponibilité étant une des propriétés de la sécurité, elle définit le fait que le système est prêt à délivrer son service. Une attaque contre la disponibilité peut avoir deux origines. La première consiste à déjouer les politiques de sécurité et à exploiter une faute pour qu'elle produise une erreur affectant la délivrance du service. La seconde méthode consiste à engorger le système de demandes de service valides afin d'occuper le système et rendre sa disponibilité faible ou inexistante pour l'utilisateur légitime. Les attaques de type Déni de Service (DoS) appartiennent à la deuxième catégorie. Ces dernières années, plusieurs sites Internet principalement à caractère commercial, sont victimes d'attaques de DoS parmi celles-ci, on retrouve la célèbre attaque TCP/SYN sur le protocole TCP/IP qu'on appelle aussi l'attaque de SYN fooding, elle est caractérisée par sa simplicité et la disponibilité d'outils sur internet pour sa réalisation. Depuis 1996, cette attaque fut perpétrée à plusieurs reprises par des intrus ayant la capacité d'initier, avec un minimum d'effort, un grand nombre d'exécutions du protocole avec un même serveur. La possibilité d'utiliser le protocole TCP/IP afin de provoquer ce DoS est partiellement due à la facilité de forger une fausse identité et, conséquemment, la difficulté pour la victime de reconnaître un attaquant. Ce fléau a suscité l'intérêt des chercheurs et a donné naissance à des études qui se sont focalisées sur la détection de cette attaque, d'autres se sont orientées vers sa modélisation (Khan, et al., 2005)**,** (Wang, et al., 2007)**,** (Aissani, 2008)**.** Les travaux de modélisation sont basés sur le formalisme files d'attentes, dans le but de déterminer des paramètres de performances exploitables pour la détection de l'attaque tels que le taux d'arrivée, le temps de réponse et le taux de croissance de la file d'attente (Khan, et al., 2005), ou vers l'évaluation des performances du système informatique à travers le calcul des métriques de la sécurité telles que la probabilité de perte de connexion et le pourcentage d'occupation du buffer par les paquets d'attaques. Dans cette optique dans (Wang, et al., 2007), le processus d'attaque a été modélisé par une file d'attente à deux dimensions avec N serveurs, deux processus d'arrivée et deux types de temps de service avec différentes distributions, un algorithme efficace en

mémoire a été proposé pour calculer les probabilités à l'état stationnaire. Pour le calcul de ces probabilités, l'algorithme résout un système d'équations linéaires qui fait intervenir la matrice de probabilités de transition d'états, et qui nécessite de la mémoire si la matrice est de grande dimension. Dans (Aissani, 2008) le même modèle a été repris mais avec une distribution arbitraire des temps de service et une méthode différente pour le calcul des probabilités à l'état stationnaire, en effet les probabilités stationnaires sont obtenues par la résolution d'un système d'équations différentielles partielles.

Les files d'attentes sont souvent utilisées pour une évaluation quantitative des systèmes, cependant elles ne sont pas adaptées pour mener une étude qualitative du système modélisé, ni pour exprimer certains mécanismes des systèmes parallèles tels que la synchronisation, l'interblocage et l'allocation de ressources. Pour cela nous avons pensé à utiliser les réseaux de Petri pour modéliser le système sous ce type d'attaque. Nous avons opté pour la variante réseau de Petri stochastique et déterministe coloré(RdPSDC). Le choix est justifié par la puissance de modélisation de ce formalisme de haut niveau approprié pour décrire et analyser les performances des systèmes caractérisés par les propriétés de concurrence et de synchronisation. En effet le caractère aléatoire du processus de demande de connexion et d'attaque, nous a orientés vers les réseaux de Petri ***stochastiques***, la présence du time out (un délai fixé et constant) pour les paquets dans le buffer du serveur nous a dirigés vers les réseaux de Petri ***déterministes***, et enfin pour distinguer les paquets d'attaques, des paquets légitimes nous avons ajouté ***la couleur***. La disponibilité d'outils pour les analyses qualitative et quantitative est un autre facteur qui nous a orientés vers ce choix.

Notre contribution consiste donc en la proposition d'un modèle qui décrit le processus de l'attaque TCP/SYN de type DOS par le formalisme RdPSDC.

Le document est organisé comme suit :

Dans le chapitre 1, nous abordons les propriétés de la sécurité informatique, les problèmes qui lui sont liés et les mécanismes utilisés pour sécuriser les systèmes. Nous introduisons aussi les différentes évaluations de la sécurité.

Le chapitre 2 est consacré à l'un des aspects de la sécurité informatique entre autres, les protocoles de sécurité. Nous commençons par introduire l'aspect, montrer le besoin de sa modélisation, et ensuite nous présentons les différents formalismes adoptés pour la modélisation de l'aspect. Nous terminons le chapitre par un modèle qui se base sur les réseaux de Petri, et une comparaison entre les différents formalismes.

Le chapitre 3 est consacré à la modélisation des contrôles d'accès, nous présentons les

différents formalismes adoptés pour la modélisation de cet aspect et nous terminons par une synthèse sur les modèles cités et une comparaison entre les différents modèles.

Dans le chapitre 4, nous abordons la modélisation des attaques par différents formalismes, nous mettons l'accent sur les modèles d'attaques qui produisent des métriques, permettant ainsi une évaluation quantitative de la sécurité, une comparaison entre les différents formalismes utilisés est donnée vers la fin du chapitre.

Enfin le chapitre 5 est consacré à notre contribution. Le modèle que nous avons proposé, PetriDOS est une modélisation par un réseau de Petri stochastique et déterministe coloré de l'attaque SYN/TCP de type déni de service, en vue de calculer les performances du système sous ce type d'attaque. Ces mesures permettent de régler les paramètres de façon à garantir le degré de disponibilité de service sous différents scénarios.

Nous terminons par une conclusion et des perspectives sur notre travail.

Les problèmes de la sécurité informatique

I. Les problèmes de la sécurité informatique

I. Introduction

Etant donnée la croissance des systèmes informatiques, et le nombre de plus en plus grand de réseaux interconnectés, la sécurité des données informatiques est aujourd'hui un problème crucial. Assurer la sécurité informatique dans une organisation consiste à faire en sorte que les ressources matérielles ou logicielles de celles-ci soient utilisées uniquement dans le cadre où il est prévu qu'elles le soient.

Les problèmes de sécurité au sein des grandes sociétés causent des dommages se chiffrant en millions d'euros. Les virus et vers entraînent régulièrement d'importantes périodes de mise à l'arrêt des systèmes ainsi que des pertes commerciales et l'endommagement de données et d'ordinateurs. Nous allons illustrer ce fait par les chiffres (Boutiller, et al., 2008); en effet avec l'ouverture sur Internet, les ordinateurs ne peuvent plus être isolés, le réseau est devenu mondial ! Les entreprises sont devenues dépendantes de l'informatique, 67 % d'entre elles ont une dépendance forte à l'informatique, 20% des entreprises déclarent avoir déjà subi un sinistre informatique grave entraînant la paralysie de l'activité et des pertes importantes de données, 80% des entreprises ayant perdu leurs données font faillite dans les 12 mois qui suivent. Plus de 74 % des pertes financières sont dues à des attaques de virus, à des accès non autorisés aux réseaux, à des pertes et vols de portables, de périphériques mobiles, de données propriétaires ou d'éléments de propriété intellectuelle. 9 jours et 17 000€ sont le temps et le coût nécessaires à une entreprise pour ressaisir 20 Mo de données perdues. Ces chiffres mettent en exergue l'importance de la protection des ressources sensibles.

Dans ce chapitre, nous allons introduire des concepts définissant le domaine de la sécurité, les problèmes de la sécurité, les techniques et les contres mesures pour sécuriser un système ainsi que l'évaluation de la sécurité.

I.1. Définitions (Boutiller, et al., 2008)

I.1.1 Définition de la sécurité informatique

C'est un ensemble de mesures adaptées pour empêcher l'utilisation non autorisée, le mauvais usage, la modification ou le refus d'utilisation d'un ensemble de connaissances, de faits, de données ou de moyens.

Cette sécurité informatique s'adresse autant aux données qu'aux outils, mais elle ne peut être garantie.

I.1.2 Le vocabulaire de la sécurité informatique

- **La menace** en anglais « threat » c'est le type d'action susceptible de nuire dans l'absolu.
- **La vulnérabilité** en anglais « vulnerability », appelée parfois faille ou brèche, représente le niveau d'exposition face à la menace dans un contexte particulier.
- **La sûreté** : protection contre les actions non intentionnelles.
- **La sécurité** : protection contre les actions intentionnelles malveillantes.
- **Le risque** : prise en compte à la fois de la probabilité d'une menace et de sa gravité, si elle réussit. Le risque en terme de sécurité est généralement caractérisé par l'équation suivante :

$$\text{Risque} = \frac{\text{Menace x Vulnérabilité}}{\text{Contre-mesure}}$$

- **La contre-mesure** (solution) : est l'ensemble des actions mises en œuvre en prévention de la menace.

Les contre-mesures à mettre en œuvre ne sont pas uniquement des solutions techniques mais également des mesures de formation et de sensibilisation à l'intention des utilisateurs, ainsi qu'un ensemble de règles clairement définies. Sécuriser un système, c'est identifier les menaces potentielles, et donc connaître et prévoir la façon de procéder en cas de menace.

II. Les problèmes de la sécurité informatique

Le problème majeur de la sécurité informatique, est de garantir la sécurité des données et des systèmes. En effet un système informatique se doit d'assurer des propriétés liées à la sécurité à savoir la confidentialité, l'intégrité, l'authentification et la disponibilité. D'autres problèmes tels que les intrusions et les failles de sécurité sur les réseaux ne sont pas moins importants.

II.1 Les propriétés de la sécurité informatique (Florin, et al., 2007)

II.1.1 Confidentialité des données

C'est la propriété qui assure que seuls les utilisateurs habilités, dans des conditions prédéfinies, ont accès aux informations.

II.1.2 Intégrité des données

C'est la propriété qui assure qu'une information n'est modifiée que dans des conditions prédéfinies (selon des contraintes précises).

II.1.3 L 'authentification

C'est la propriété qui assure que seules les entités autorisées ont accès au système. L'authentification protège de l'usurpation d'identité. Ne pas confondre authentification avec confidentialité ou intégrité. L'authentification est un moyen clé de la sécurité pour assurer: ***la confidentialité***, ***l'intégrité.***

II.1.4 Non – répudiation

C'est la propriété qui assure que l'auteur d'un acte ne peut ensuite dénier l'avoir effectué. Il y a deux aspects spécifiques de non répudiation dans les transactions électroniques:

-la preuve d'origine : Un message (une transaction) ne peut être dénié par son émetteur.

-la preuve de réception : Un récepteur ne peut dénier avoir reçu un ordre s'il ne lui a pas plu de l'exécuter alors qu'il le devait.

II.1.5 La pérennité

C'est une terminologie de la sécurité informatique pour caractériser le bon fonctionnement du système informatique. En termes de la sûreté de fonctionnement on parle de:

- ***Disponibilité*****:** L'aptitude d'un système informatique à pouvoir être employé à un instant donné par les utilisateurs.

- ***Fiabilité*** **:** L'aptitude d'un système informatique à fonctionner correctement de manière continue pendant une période donnée.

II.2 Les failles de sécurité

II.2.1 Intrusions, Attaques, Vulnérabilités

Les fautes qui peuvent porter atteinte aux propriétés de sécurité peuvent être accidentelles, comme elles peuvent être intentionnelles, avec ou sans volonté de nuire.

Une **intrusion** est définie (Abou El Kalam, 2003) comme étant une faute opérationnelle, externe, intentionnellement nuisible, résultant de l'exploitation d'une vulnérabilité dans le système. L'usage courant du mot intrusion couvre le fait de pénétrer illégalement ou sans y être convié dans un lieu, une société, etc.

En outre, le système peut être attaqué sans succès. Dans ce cas, l'attaque existe, mais la protection est suffisamment efficace pour empêcher l'intrusion. Il existe deux causes sous-jacentes à une intrusion (figure 1) :

- une action malveillante ou **attaque** qui tente d'exploiter une faiblesse dans le système et de violer un ou plusieurs besoins de sécurité ;

- au moins une faiblesse, faille ou **vulnérabilité,** qui est une faute accidentelle ou intentionnelle (avec ou sans volonté de nuire), introduite dans la spécification, la conception ou la configuration du système.

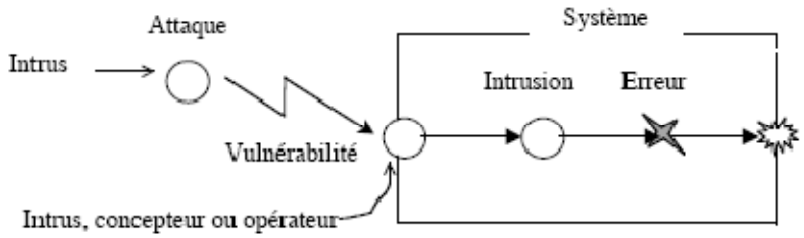


Figure 1 : Intrusion, attaque et vulnérabilité.

En définissant une **menace** (Abou El Kalam, 2003) comme une violation potentielle d'une propriété de sécurité, les couples (menace, vulnérabilité) permettent d'identifier les risques auxquels le système étudié peut être soumis. Une attaque contre la sécurité du système peut provenir de l'intérieur ou de l'extérieur. Un intrus interne peut être défini comme étant un utilisateur malveillant appartenant à l'organisation, tandis qu'un intrus externe est une personne n'ayant pas de privilèges. C'est donc un individu non enregistré comme utilisateur, mais qui tente de pénétrer dans le système en trompant ou en contournant les mécanismes d'authentification et d'autorisation.

II.2.1.1 Les failles de sécurité sur les réseaux

Les failles de sécurité présentes dans les réseaux informatiques et particulièrement dans Internet, sont les attaques et l'espionnage, nous citons ci après les principales attaques et les techniques d'espionnage les plus répandues, ainsi que les éléments des réseaux facilitant la réalisation des ces techniques.

Les principales attaques (LESCOP, 2002)

a- Les virus

Le virus est un code exécutable qui exécute des opérations plus ou moins destructrices sur une machine. Les virus existent depuis que l'informatique est née et se propageaient initialement par disquettes de jeux ou logiciels divers. Sur Internet, les virus peuvent contaminer une machine de plusieurs manières :

• Téléchargement de logiciel puis exécution de celui-ci sans précautions,

• Ouverture sans précautions de documents contenant des macros,

• Pièce jointe de courrier électronique,

• Exploitation d'un bug du logiciel de courrier électronique.

Les virus peuvent être très virulents, en plus la mise en place d'antivirus et la réparation des dégâts causés nécessitent beaucoup de temps.

b- Déni de service (DoS)

Le but d'une telle attaque n'est pas de dérober des informations sur une machine distante, mais de paralyser un service ou un réseau complet. Les utilisateurs ne peuvent plus alors accéder aux ressources.

c- Écoute du réseau (sniffer)

Il existe des logiciels qui permettent d'intercepter certaines informations qui transitent sur un réseau local, en retranscrivant les trames dans un format plus lisible *(Network packet sniffing).* C'est l'une des raisons qui font que la topologie en étoile autour d'un hub n'est pas la plus sécurisée, puisque les trames qui sont émises en « *broadcast* » sur le réseau local peuvent être interceptées. De plus, l'utilisateur n'a aucun moyen de savoir qu'un pirate a mis son réseau en écoute.

d- Intrusion

L'intrusion dans un système informatique a généralement pour but la réalisation d'une menace qui est donc une attaque. Les conséquences peuvent être catastrophiques : vol, fraude, incident diplomatique, chantage… .

e- Cheval de Troie

L'image retenue de la mythologie est parlante; le pirate, après avoir accédé au système, installe un logiciel qui va, à l'insu de l'utilisateur, lui transmettre par internet des informations de l'utilisateur. Un tel logiciel, aussi appelé troyen ou trojan, peut aussi être utilisé pour générer de nouvelles attaques sur d'autres serveurs.

f- Social engineering

En utilisant les moyens usuels (téléphone, email…) et en usurpant une identité, un pirate cherche à obtenir des renseignements confidentiels auprès du personnel de l'entreprise en vue d'une intrusion future.

L'espionnage : parmi les techniques d'espionnage les plus utilisées on trouve l'homme du milieu et les logiciels espions.

a- L'homme du milieu

Lorsqu'un pirate, prenant le contrôle d'un équipement du réseau, se place au milieu d'une communication, il peut écouter ou modifier celle-ci. On parle alors de « l'homme du milieu » en anglais « man in the middle ». Les éléments sensibles permettant cette technique sont :

• **DHCP** : le protocole DHCP n'est pas sécurisé, un pirate peut prendre le contrôle du réseau et fournir à une victime des paramètres réseau qu'il contrôle lui même.

• **ARP** : si le pirate est dans le même sous réseau que la victime et le serveur, il peut envoyer régulièrement des paquets ARP signalant un changement d'adresse MAC aux deux extrémités.

• **RIP** : le pirate envoie une table de routage à un routeur indiquant un chemin à moindre coût et passant par un routeur dont il a le contrôle.

• **DNS** : par « ID spoofing », un pirate peut répondre le premier à la requête de la victime. Et par « cache poisoning » il corrompt le cache d'un serveur DNS.

• **Proxy HTTP** : par définition un proxy est en situation d'homme du milieu.

• **Virus** : un virus, peut écrire dans le fichier « hosts ».

b- Espiogiciels

Ces logiciels espions sont aussi appelés ***« spyware ».*** Ils ne posent pas, à priori, de problème de sécurité mais plutôt celui du respect de la vie privée.

III. Techniques et mécanismes pour sécuriser un système

La sécurité informatique couvre généralement trois principaux objectifs (Abou El Kalam, 2003):

- **L'intégrité** : garantir que les données sont bien celles que l'on pense.
- **La confidentialité** : consiste à assurer que seules les personnes autorisées ont accès aux ressources.
- **La disponibilité** : permettant de maintenir le bon fonctionnement du système d'information. Son objectif est de garantir l'accès à un service ou à des ressources.

Cette liste n'est pas complète, on peut y ajouter des aspects qui restent importants :

- **Authenticité**: s'assurer de l'identité d'un utilisateur.
- **Responsabilité**: attacher une identité à une opération.
- **Fiabilité**: s'assurer du fonctionnement d'un système.

Afin d'éliminer les vulnérabilités, contrer les attaques, et garantir un niveau élevé de protection du réseau et du système d'information, on peut utiliser des services, des mécanismes, des outils et des procédures que l'on nomme, de façon générale, des solutions ou des mesures de sécurité. Par exemple, un service d'authentification aide à réduire le risque d'intrusion dans un système. Les politiques de sécurité sont des dispositifs nécessaires pour

renforcer la sécurité des systèmes. Nous citons également d'autres contre-mesures pour renforcer la sécurité comme les mécanismes cryptographiques, l'audit, la détection d'intrusions et la tolérance aux intrusions.

III.1 Politique de sécurité

Pour atteindre un niveau de protection satisfaisant, il convient de définir une politique de sécurité correspondant aux besoins. En effet, toute démarche de sécurité rigoureuse doit être inscrite dans une politique claire et documentée. Sa conception est donc une étape primordiale, qui consiste à identifier les objectifs de sécurité et à élaborer un ensemble de règles en fonction d'une analyse des risques.

Une politique de sécurité (Dacier, 1994) se développe selon trois axes : physique, administratif et logique. Le premier précise l'environnement physique du système à protéger, il s'agit des éléments critiques, et des mesures prises vis-à-vis du vol et des catastrophes. Le deuxième décrit les procédures organisationnelles à savoir la répartition des tâches, la séparation des pouvoirs. Le troisième a trait aux contrôles d'accès logiques (qui, quoi, quand, pourquoi, comment) et s'intéresse aux fonctions d'identification, d'authentification et d'autorisation mises en œuvre par le système informatique.

> ***La politique de sécurité est donc l'ensemble des orientations suivies par une organisation en termes de sécurité.***

III.2 Autres contre-mesures

Dans certains cas, la politique de sécurité peut être contournable ou mal implémentée, ou incomplète et ne couvre pas tous les accès possibles, conséquence d'une faute de conception ou d'un choix délibéré. Il convient donc de renforcer la sécurité par d'autres contre-mesures, tels que les mécanismes cryptographiques, la détection d'intrusions ou la tolérance aux intrusions.

III.2.1 Mécanismes cryptographiques

La *cryptologie* (Abou El Kalam, 2003)**,** (ADI, 2006) se compose de la *cryptographie*, l'art d'écrire des secrets pour les rendre inintelligibles à des tiers, et de la *cryptanalyse*, l'art de retrouver les secrets cachés dans des informations inintelligibles. Il ne sera ici question que des éléments de cryptographie, qui sont à la base de nombreux mécanismes de sécurité : le chiffrement, le hachage et la signature.

a- Chiffrement et déchiffrement

Les fonctions de base de la cryptographie sont le chiffrement et le déchiffrement. Le chiffrement vise à assurer la confidentialité d'informations ; il consiste à transformer un texte en clair en un *cryptogramme*, à l'aide d'un *chiffre* (ou algorithme de chiffrement) et d'une clé de chiffrement Kc. Le déchiffrement consiste à transformer le cryptogramme en un texte en clair identique à celui d'origine, à l'aide d'un algorithme de déchiffrement et d'une clé de déchiffrement Kd.

On parle de *chiffre symétrique* si Kd=Kc. Si, à l'inverse, Kc et Kd sont différents, et si, connaissant l'un, il est impossible de trouver l'autre, on parle de *chiffre à clé publique* (ou *asymétrique*). Le chiffre à clé publique le plus courant est le RSA, des noms de ses auteurs (Rivest, Shamir, Adleman). Quand on utilise un chiffre à clé publique pour la confidentialité, la clé de chiffrement *Kc* peut être connue publiquement, mais seul celui qui possède la clé de déchiffrement (secrète) *Kd* peut déchiffrer le cryptogramme.

Un chiffre est dit *hybride* s'il combine à la fois les chiffrements symétrique et asymétrique.

b- Fonctions de hachage

Une fonction de hachage à sens unique (*one-way hash function*) permet de générer une *empreinte* de taille fixe *n* à partir d'un message de taille quelconque. L'empreinte doit être une caractéristique du texte et il doit y avoir une très faible probabilité que deux messages différents aient la même empreinte. La fonction de hachage H doit donc être conçue de telle sorte que :

- connaissant M, il est facile de calculer l'empreinte H(*M*) ;
- connaissant H(*M*), il doit être "*impossible*" de trouver *M* ;
- connaissant H(*M*), il doit être "*impossible*" de trouver un texte *M*' différent de *M* et ayant la même empreinte : H(*M*') = H(*M*).

Les fonctions de hachage sont utilisées pour la génération de signature. MD5 et SHA-1 sont les deux fonctions actuellement les plus utilisées.

c- Signature et contrôles d'intégrité

La *signature* sert à garantir l'intégrité de l'information, elle est obtenue en appliquant à un texte une fonction de génération de signature utilisant une clé de signature *Ks*. La vérification de signature se fait à l'aide d'un algorithme et d'une clé de vérification *Kv* (figure 2).

L'intégrité du texte est garantie par le fait que si le texte ou la signature sont modifiés entre la génération et la vérification, l'algorithme de vérification doit rendre une réponse négative.

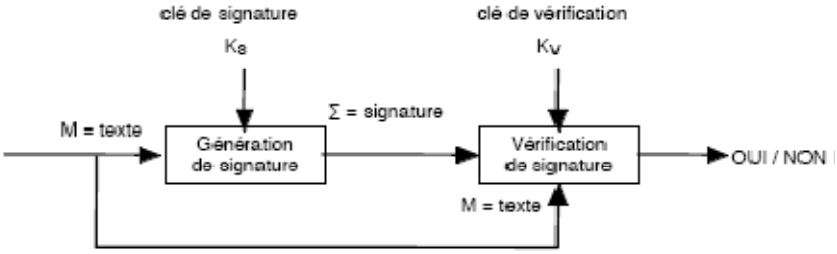


Figure 2 : Génération et vérification de signature.

Comme pour le chiffrement, on peut distinguer les signatures symétriques où *Ks=Kv* et les signatures à clé publique où *Kv* est une clé publique (n'importe qui peut vérifier la signature) alors que *Ks* est tenue secrète par le signataire. Dans ce cas, il doit être "impossible" de trouver *Ks* en connaissant *Kv*.

d- Certificats

Un *certificat* permet d'établir un environnement de confiance entre deux entités distantes ayant besoin de communiquer entre elles et de s'échanger des informations non-répudiables (nécessité de signature) ou confidentielles (application de chiffrement). En effet, un certificat est souvent destiné à remplir trois rôles : authentification de l'émetteur, garantie de l'intégrité des documents, et éventuellement un horodatage.

Un certificat électronique doit contenir : le nom de *l'autorité de certification*, le nom et le prénom de la personne, son entreprise, son adresse électronique, sa clé publique, les dates de validité du certificat ainsi qu'une signature électronique (figure 3). Cette signature, calculée sur la base des informations contenues dans le certificat, est l'empreinte de ces informations, chiffrée avec la clé privée de l'autorité de certification qui a délivré ce certificat.

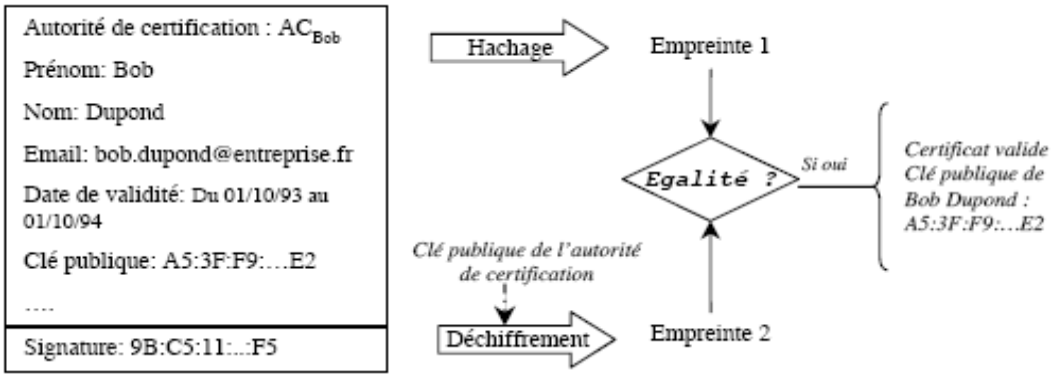


Figure 3 : Vérification de certificat et récupération de la clé publique.

III.2.2 Cloisonnement et pare-feux

Le cloisonnement est un principe de sécurité qui consiste à isoler tout ce qui n'a pas besoin de communiquer.

Les « pare-feux » (*firewalls* en anglais) permettent de surveiller et de restreindre les accès de l'extérieur, par exemple l'Internet, vers l'intérieur : une machine, un réseau local, les réseaux d'une entreprise, mais aussi les accès de l'intérieur vers l'extérieur. Un pare-feu est donc l'un des mécanismes de contrôle d'accès qui peut être mis en œuvre pour implémenter les règles de la politique de sécurité.

Un pare-feu comporte essentiellement une fonction de filtrage : il ne laisse passer que les paquets provenant de certaines adresses autorisées et à destination de certaines adresses autorisées. Mais il peut avoir d'autres fonctions complémentaires, comme la traduction d'adresses (NAT, pour *Network Address Translation*), ou jouer le rôle de mandataire d'application. La traduction d'adresses permet de gérer l'espace d'adressage du réseau interne indépendamment du réseau externe : les adresses internes ne sont pas connues de l'extérieur, elles sont traduites en adresses externes par le pare-feu. Un mandataire (*proxy* en anglais) d'application permet d'interpréter chacune des interactions d'une application (commandes, requêtes, réponses) pour vérifier que les échanges suivent bien un protocole autorisé.

III.2.3 Audit

L'audit sert à conserver des traces des opérations susceptibles de remettre en cause la sécurité, de façon à analyser, après coup ou en temps réel, si des malveillances ont lieu ou ont eu lieu et quels sont les moyens et les méthodes utilisés, de façon à punir les coupables et à corriger les vulnérabilités. Il faut donc enregistrer toutes les opérations liées à la sécurité, que ces opérations soient réussies (parce qu'autorisées) ou qu'elles aient échoué (empêchées par les mécanismes de contrôle d'accès). Les principales opérations à surveiller sont :

- la connexion et la déconnexion des utilisateurs ;
- la création, modification, destruction des informations de sécurité (droits d'accès, mots de passe, etc.) ;
- les changements de privilèges.

Les journaux d'audit doivent être indestructibles. Seuls les administrateurs de l'audit peuvent les détruire. Ils doivent porter sur tous les utilisateurs y compris les administrateurs et les responsables de la sécurité et contenir un maximum d'informations utiles (date et heure, identité de l'utilisateur, type d'opération, référence de l'information, etc.).

III.2.4 Détection d'intrusions

Les systèmes de détection d'intrusions (IDS pour Intrusion Detection System) ont pour objectif de révéler, généralement via des alertes, toute activité pouvant être considérée comme intrusive, depuis ou vers un système d'information, par analyse de données (BRIFFAUT, 2007). Il existe deux types d'IDS :

- Les IDS sur réseau (*Network Based IDS*), qui observent les paquets circulant sur le réseau ; ce sont des machines indépendantes dédiées à la détection d'intrusions ;
- Les IDS sur hôte (*Host Based IDS*), qui observent le comportement du système, en particulier les appels systèmes, ou qui analysent les informations d'audit ; il s'agit alors de fonctions intégrées au système qu'ils observent.

Les techniques de détection d'intrusions se répartissent en deux classes (figure 4) : détection d'anomalies, aussi appelée approche comportementale, et détection d'attaques, dite également approche par scénario.

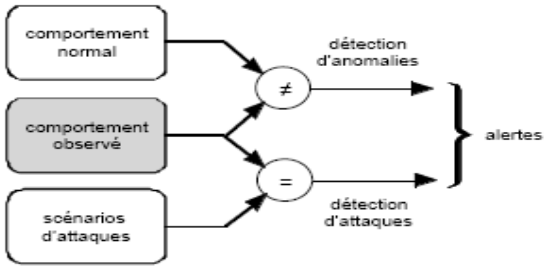


Figure 4 : Techniques de détection d'intrusions.

III.2.5 Tolérance aux intrusions

Il n'est pas possible d'éviter les attaques sur les systèmes de grandes tailles. De même, il est impossible d'éliminer toutes les vulnérabilités. Il faut donc s'attendre à ce que certaines attaques réussissent, c'est-à-dire produisent des intrusions. Puisqu'il est inévitable que des intrusions se produisent, il serait intéressant de tolérer les intrusions, c'est-à-dire de faire en sorte que l'intrusion dans une partie du système n'ait pas de conséquence sur sa sécurité.

Un système tolérant aux intrusions doit pouvoir s'auto diagnostiquer, se réparer et se reconfigurer tout en continuant à fournir un service acceptable aux utilisateurs légitimes pendant les attaques. Parmi les techniques classiques existantes pour tolérer ces intrusions dans le domaine de la sécurité des systèmes, on trouve le chiffrement, la réplication, et le brouillage des données. Une technique a été développée pour préserver la confidentialité tout en permettant de tolérer les fautes accidentelles et les intrusions, y compris par des utilisateurs privilégiés. Cette technique (Vache-Marconato, 2009) utilise les concepts de fragmentation,

redondance et dissémination. L'information est découpée en fragments ; ces fragments sont dupliqués et disséminés sur différents sites. La reconstitution de l'information par une personne malveillante devient une tâche bien plus difficile. Concernant la redondance avec diversification, le principe repose sur le constat suivant : une attaque qui cible une vulnérabilité d'un système fonctionnant sur une architecture matérielle particulière a peu de chance d'être réalisée avec succès sur un autre système d'architecture différente. Le principe est d'appliquer le principe de redondance en fournissant le service depuis plusieurs sous-systèmes fonctionnant sur des architectures matérielles et logicielles différentes. Le système global fournit le service en se basant sur un vote majoritaire entre les sous systèmes.

IV. Les différents aspects de la sécurité informatique

Le domaine de la sécurité informatique est vaste, les protocoles de sécurité, le contrôle d'accès, et les attaques sont les aspects les plus étudiés dans ce domaine. Nous allons parler brièvement de ces aspects en vue de les détailler dans les prochains chapitres.

IV.1 Les protocoles de sécurité

Les mécanismes de sécurité que nous avons présentés, sont utilisés à travers les procédures d'authentification ou d'échange de messages décrites par des protocoles de sécurité en vue de sécuriser les communications contre les attaques qui tentent d'exploiter d'éventuelles failles présentes dans les protocoles.

Les protocoles de sécurité peuvent donc être utilisés pour de nombreux buts : échanger des messages confidentiels, authentifier des individus, se brancher sur un serveur web, etc.

Il est possible de classer les différents types de protocoles de sécurité selon les services qu'ils offrent (Lafrance, 2005). Par exemple, les protocoles cryptographiques sont caractérisés par l'utilisation du cryptage de données afin d'assurer la confidentialité de certaines données échangées entre les usagers. Les services offerts par ces protocoles incluent, notamment, l'authentification des usagers et l'échanges de clés et de messages confidentiels, ainsi que la création de connexions fiables pour les flots de données. Par exemple, un protocole d'authentification offre un service d'authentification entre deux participants par l'entremise d'une certaine procédure d'authentification. L'authentification est donc la pierre angulaire de la sécurité des protocoles de sécurité.

Afin d'assurer la confidentialité et l'authenticité des messages échangés, les protocoles de sécurité nécessitent régulièrement l'utilisation de stratégies de cryptage, de signature et de hachage. Cependant, malgré l'hypothèse de cryptage parfait (c'est-à-dire un intrus ne peut

décrypter un message que s'il possède la clé correspondante), les protocoles de sécurité peuvent tout de même contenir des failles au niveau de leur conception, ce qui les rend vulnérables à des attaques perpétrées par des intrus ayant accès aux réseaux publics sur lesquels les données sont échangées.

IV.2 Le contrôle d'accès

Dans les systèmes et réseaux informatiques, les mécanismes de sécurité sont utilisés en vue d'assurer un accès sécurisé.

Le contrôle d'accès est indispensable pour la sécurité dans les systèmes informatiques. Il permet de limiter les actions ou les opérations qu'un utilisateur ou un système informatique peut réaliser. Il contraint donc ce que l'utilisateur peut faire directement, mais aussi les programmes exécutés sur demande de l'utilisateur, afin que seules les entités autorisées puissent accéder aux ressources du système d'information. C'est ainsi, un mécanisme qui assure la confidentialité et l'intégrité des données car tout sujet qui souhaite faire une opération sur un objet, doit être autorisé à accéder à l'objet, et selon la politique de sécurité en vigueur, l'accès à l'objet va être accordé.

IV.3 Les attaques

Aujourd'hui les attaques sérieuses sont complexes, ce sont des scénarios à plusieurs étapes et peuvent s'impliquer en déviant de multiples mécanismes de sécurité et utiliser de nombreux systèmes d'ordinateurs. Les hôtes qui sont contrôlées par les attaquants peuvent devenir point de lancement pour des intrusions.

Sur internet, des attaques ont lieu en permanence, à raison de plusieurs attaques par minute sur chaque machine connectée. Ces attaques sont pour la plupart lancées automatiquement à partir de machines infectées par des virus, chevaux de Troie, vers, etc., à l'insu de leur propriétaire. Les motivations des attaquants sont différentes, il peut s'agir d'un accès non autorisé au système ou de récupérer des données confidentielles ou de troubler le bon fonctionnement d'un service etc., elles se résument toutes en la violation d'une des propriétés de la sécurité.

V. Modélisation des différents aspects de la sécurité informatique

Lorsqu' un mécanisme de sécurité est implémenté dans un protocole de sécurité, ou pour contrôler l'accès à un système informatique, il est difficile de valider formellement les propriétés de sécurité. La preuve des propriétés de sécurité nécessite de disposer de méthodes pour abstraire le comportement d'un système en modèle simplifié, qui peut ensuite être

analysé, ainsi il devient possible d'utiliser des outils mathématiques pour prouver la validité des propriétés.

Les protocoles de sécurité, comme nous l'avons déjà mentionné, il peut exister des failles de conception qui rendent ces protocoles vulnérables à des attaques perpétrées par des intrus ayant accès aux réseaux publics sur lesquels les données sont échangées. Il n'y a pas si longtemps, des protocoles étaient considérés sécuritaires simplement si personne n'y avait trouvé de faille. Ainsi, certains protocoles furent utilisés pendant plusieurs années avant d'être prouvés non sécuritaires (Lafrance, 2005), d'où la nécessité de vérifier et de valider les protocoles de sécurité. La première étape vers la vérification d'un protocole cryptographique est son expression dans une syntaxe suffisamment générale pour permettre d'exprimer le protocole de façon précise. Il convient ensuite de choisir une syntaxe pour les propriétés à vérifier. Enfin, il reste à choisir le modèle permettant de définir la sémantique des protocoles et des propriétés. La validation de protocoles de sécurité est une tâche qui requiert typiquement le développement de méthodes formelles permettant la détection de toute attaque possible sur un protocole donné.

Le contrôle d'accès, dans le domaine de la vérification des politiques de sécurité, au cours des dernières années, les démarches de vérification formelle ont connu un essor croissant. La vérification d'une politique de sécurité se fait généralement en deux étapes qui sont complémentaires : d'une part, la formalisation de la politique sous la forme d'un langage ou d'un modèle formel qui exprime les exigences de sécurité (les règles d'accès) propres au système auquel la politique est destinée, et d'autre part, la vérification au moyen de techniques et outils appropriés permettant de vérifier que ces exigences seront effectivement satisfaites après une implémentation dans le système cible.

Les attaques, la recherche dans le domaine des cybers attaques a gagné une immense emphase ces dernières années. Un des efforts de recherche dans un tel domaine est la modélisation des attaques. Des formalismes ont été développés pour décrire le processus par lequel un utilisateur malicieux tente d'exploiter ou arrêter des systèmes et des réseaux informatiques, dans le but de décomposer et de visualiser l'attaque afin de pouvoir l'analyser et développer ainsi des contremesures plus efficaces ou mesurer le coût de l'attaque.

Dans les chapitres suivants un état de l'art sur les modèles formels des différents aspects de la sécurité sera présenté.

VI. Evaluation de la sécurité informatique

Il est important d'évaluer le niveau de sécurité des systèmes informatiques, pour savoir si on a obtenu un niveau de sécurité satisfaisant, et identifier les points les plus critiques à surveiller ou à corriger, et enfin pour estimer s'il est rentable de mettre en œuvre telle ou telle défense supplémentaire. L'évaluation quantitative de la sécurité vis-à-vis d'une vulnérabilité donnée se présente comme un besoin grandissant pour permettre la prévention et la prévision.

Mesurer la sécurité des systèmes a commencé avec le besoin d'assurer des objectifs de sécurité des systèmes d'information militaires. Des travaux de recherche ont conduit à plusieurs approches ayant toutes pour but de pouvoir évaluer la sécurité. Trois grandes méthodes d'évaluation de la sécurité peuvent être distinguées : l'utilisation des critères d'évaluation, l'analyse des risques et les méthodes d'évaluation quantitative de la sécurité. (Dacier, 1994),(Abou El Kalam, 2003), (Vache-Marconato, 2009).

VI.1 Critères d'évaluation qualitative

Les approches d'évaluation de la sécurité par critères permettent une évaluation de la sécurité dès la phase de conception et d'implémentation du système. Elles permettent d'inclure des politiques de sécurité en donnant au concepteur des directives pour assurer certaines propriétés de sécurité.

Les premiers critères d'évaluation de la sécurité ont été définis aux Etats-Unis (U.S, 1985), dans le Livre Orange ou TCSEC (Trusted Computer System Evaluation Criteria), ou dans les différentes interprétations associées à des livres de diverses couleurs qui l'accompagnent, comme le Livre Rouge ou T N I (Trusted Network Interpretation of the TCSEC*)*. Ces critères, fondés sur les listes de fonctions de sécurité à remplir et sur les techniques employées pour la vérification, ont conduit à une classification des systèmes en sept catégories (D, C1, C2, B1, B2, B3, A1). Pour chaque niveau, quatre familles de critères sont définies, traitant de la politique d'autorisation, de l'audit, de l'assurance et de la documentation :

- **La politique d'autorisation** stipule une politique précise à suivre discrétionnaire ou obligatoire en fonction des différents niveaux de certifications visés. La politique obligatoire imposée est celle définie par Bell-LaPadula.
- **Les critères d'audit** précisent les fonctions requises en matière d'identification, d'authentification et d'audit des actions.

- **Les critères d'assurance** fixent des recommandations concernant des méthodes de conception et de vérification utilisées afin d'augmenter la confiance de l'évaluateur, il s'agit de garantir que le système implémente bien la fonctionnalité qu'il prétend avoir.
- **Les critères de documentation** spécifient les documents qui doivent être fournis avec le produit lors de l'évaluation.

Les caractéristiques principales des différents niveaux définis par le livre orange sont ainsi :

- un système classé au niveau **D** est un système qui n'a pas été évalué ;
- jusqu'au niveau **C1** et **C2**, le système peut utiliser une politique discrétionnaire ;
- pour les niveaux **B1**, **B2**, et **B3** le système utilise une politique obligatoire ;
- un système classé **A1** est fonctionnellement équivalent à un système classé **B3**, sauf qu'il est caractérisé par l'utilisation de méthodes formelles de vérification pour prouver que les contrôles utilisés permettent bien d'assurer la protection des informations sensibles.

Les TCSEC visent d'abord à satisfaire les besoins du DoD (Department of Defense) des États-Unis, privilégiant ainsi la confidentialité des données militaires. Par ailleurs, le manque de souplesse et la difficulté de leur mise en œuvre, ont conduit au développement de nouvelles générations de critères.

Les ITSEC (Information Technology Security Evaluation Criteria) sont le résultat d'harmonisation de travaux réalisés au sein de quatre pays européens : l'Allemagne, la France, les Pays-Bas et le Royaume-Uni. La différence essentielle entre les TCSEC et les ITSEC réside dans la distinction entre fonctionnalité et assurance. Une classe de fonctionnalité décrit les fonctions que doit mettre en œuvre un système tandis qu'une classe d'assurance décrit l'ensemble des preuves qu'un système doit apporter pour montrer qu'il implémente les fonctions qu'il prétend fournir. Les classes d'assurance sont au nombre de six (E1 à E6). Parmi les classes de fonctionnalité, on retrouve les classes (F-C1, F-C2, F-B1, F-B2, F-B3) correspondant aux classes C1 à B3 définies dans les TCSEC.

VI.2 Analyse des risques

L'analyse des risques est une discipline destinée à mettre en œuvre une politique de sécurité en tenant compte des vulnérabilités résiduelles (Dacier, 1994), en évaluant leurs impacts sur la sécurité, et en calculant les rapports coûts/bénéfices apparaissant dans une organisation.

On qualifie de risque, un évènement redouté auquel seront attribuées une fréquence de réalisation et une conséquence (Dacier, 1994). Les risques sont donc évalués à partir d'une estimation de la fréquence de réalisation des menaces et des conséquences.

La démarche générale d'une analyse de risques (Vache-Marconato, 2009) peut se résumer dans les trois étapes suivantes :

• Identifier les menaces auxquelles devra faire face le système ou l'organisation (quels types d'attaquants, mais aussi quels phénomènes physiques : incendie, inondation, intrusion…) ;

• Identifier les vulnérabilités du système ou de l'organisation face à ces menaces ;

• Estimer les conséquences qui résulteraient de la réalisation des menaces et de l'exploitation des vulnérabilités.

Il existe plusieurs méthodes d'analyse de risques dédiées aux systèmes d'information. Parmi les plus anciennes on trouve MARION développée en 1980, et parmi les plus récentes, on trouve MEHARI (www.clusif.asso.fr) développée en 1995 et mise à jour en 2007.

VI.3 Evaluation quantitative

Les évaluations fournies par les critères et les méthodes d'analyse de risques ne permettent pas une évaluation régulière et un suivi de l'évolution du niveau de sécurité du système. De plus, les mesures résultant de ces évaluations ne permettent pas de comparer des systèmes entre eux ou le même système à deux instants d'évaluation différents. Ces mesures ont été définies comme des mesures qualitatives. Une mesure qualitative ne permet pas de quantifier une distance à un objectif (Vache-Marconato, 2009). Pour ces raisons, des études cherchant à étudier la quantification de la sécurité en mode opérationnel ont vu le jour. Une partie du chapitre 4 sera consacrée aux modèles d'évaluation quantitative de la sécurité à travers des modèles d'attaques fournissant des métriques.

VII. Conclusion

Ce chapitre nous a permis de mettre le point sur les problèmes de la sécurité informatique tout en présentant les concepts définissant le domaine. La fin de ce chapitre est marquée par la nécessité d'une modélisation formelle de la sécurité à travers la modélisation des différents aspects, et la nécessité de l'évaluation quantitative de la sécurité pour permettre la comparaison entre les systèmes. Dans les chapitres suivants une description de plusieurs modèles formels sera présentée.

Les Modèles Formels pour les protocoles de sécurité

2. Les Modèles Formels pour les protocoles de sécurité

Introduction

Plusieurs formalismes ont été proposés pour modéliser les protocoles de sécurité et les contrôles d'accès, afin d'exprimer les besoins de la sécurité informatique et de fournir des moyens pour montrer que ces besoins sont satisfaits. Les attaques ont aussi été modélisées, pour décrire et analyser le processus par lequel un utilisateur malicieux tente d'exploiter ou arrêter des systèmes et/ou des réseaux informatiques.

Dans la littérature, nous avons rencontré une diversité de modèles relevant de la sécurité et adoptant des formalismes différents, que nous classifions en trois catégories : les formalismes algébriques, les formalismes logiques et enfin les formalismes basés sur les réseaux de Petri (Lee, et al., 1997) (Bidan, et al., 1995). Ce chapitre est consacré aux protocoles de sécurité, nous allons commencer par définir l'aspect modélisé, ensuite les formalismes les plus utilisés dans la modélisation de cet aspect.

I. Les protocoles de sécurité

Les protocoles de sécurité sont une spécification des modèles de communications qui visent à permettre aux agents de partager les secrets à travers le réseau public. On requiert aux protocoles de s'exécuter correctement (Abdul Sahid, et al., 2005), surtout en présence d'intrus malicieux, qui écoutent aux échanges de messages dans les réseaux, et manipulent le système en bloquant ou en falsifiant des messages. La correction requise inclut ***le secret (la confidentialité)*** : l'intrus ne peut pas lire le contenu du message destiné à un autre, et ***l'authentification*** : si B reçoit un message qui apparaît être de la part de l'agent A et destiné pour B, alors A a en effet envoyé le même message destiné à B dans le passé récent (il y a eu envoi puis réception du même message). La présence d'utilisateurs malveillants nécessite l'utilisation d'une communication cryptée. Mais malgré l'utilisation des outils de cryptographie les plus parfaits (c'est-à-dire un intrus ne peut décrypter un message que s'il possède la clé correspondante), les buts de sécurité désirés ne sont pas toujours garantis. Cette situation se pose essentiellement à cause des défauts logiques dans la conception des protocoles.

Un exemple basé sur le protocole de Needham et Schroeder montre bien quelques problèmes importants.Ce protocole cryptographique utilise le cryptage à clé publique dans le but d'établir une

authentification mutuelle entre deux participants (désignés par A et B), un protocole est généralement défini par une suite de messages échangés entre deux ou plusieurs participants :

$$\text{Msg 1. } A \rightarrow B : \{x, A\}_{pubkB}$$

$$\text{Msg 2. } B \rightarrow A : \{x, y\}_{pubkA}$$

$$\text{Msg 3. } A \rightarrow B : \{y\}_{pubkB}$$

Afin de compléter la procédure d'authentification, ce protocole exige que chaque participant possède la clé publique de son homologue, ces clés publiques sont respectivement désignées par pubkA et pubkB. Les agents **A** et **B** partagent des nonces secrets ***x*** et ***y*** (un nonce désigne un nombre aléatoire). Il a été démontré que ***x*** et ***y*** suivant ce protocole ne sont pas nécessairement connus que par **A** et **B.** En effet, Lowe (Lowe, 1996) a découvert une faille dans ce protocole pouvant mener à une attaque de type « man-in-the-middle ». L'attaque nommée « Low's attack » est présentée par le contre exemple :

$$\text{Msg } \alpha.1. A \rightarrow I: \{x, A\}_{pubkI}$$

$$\text{Msg } \beta.1. (I)A \rightarrow B : \{x,A\}_{pubkB}$$

$$\text{Msg } \beta.2. B \rightarrow (I)\ A: \{x, y\}_{pubkA}$$

$$\text{Msg } \alpha.2. I \rightarrow A: \{x, y\}_{pubkA}$$

$$\text{Msg } \alpha.3. A \rightarrow I: \{y\}_{pubkI}$$

$$\text{Msg } \beta.3. (I)A \rightarrow B: \{y\}_{pubkB}$$

Cette fameuse attaque est seulement réalisable lorsque le participant A initie le protocole d'authentification avec un participant malveillant *I* (souvent appelé intrus ou participant ennemi). Dans l'attaque ci-dessus. $(I)A \rightarrow B : m$ signifie que l'intrus *I* envoie le message *m* à *B* au nom de *A*, tandis que $A \rightarrow (I)\ B : m$ signifie que l'intrus *I* bloque le message envoyé par *A* à l'intention de *B*, l'attaque est réalisée en deux sessions de protocole (deux exécutions parallèles du protocole de Needham-Schröder), l'une des deux sessions dont les messages sont libellés par α qui implique *A* comme initiateur et *I* comme répondeur, et l'autre dont les messages sont libellés par β qui implique *I* (au nom de *A*) comme initiateur et *B* comme répondeur. Après le cinquième message, l'intrus arrive à connaître ***y*** qui est le secret généré par *B* dans une session avec quelqu'un que *B* croit que c'est *A*. Cela montre que le protocole ne satisfait pas la propriété : *quand un agent B s'engage dans une session de protocole comme répondeur et B croit que l'initiateur est A, alors le*

secret généré par B n'est connu que par A et B .Cette attaque assez simple sur le protocole de Needham et Schroeder a été découverte dix-sept ans après que le protocole original a été proposé. Cet exemple montre bien que les protocoles de sécurité peuvent contenir des failles au niveau de leur conception, ce qui les rend vulnérables à des attaques perpétrées par des intrus ayant accès aux réseaux publics sur lesquels les données sont échangées.

I. 1 Définition d'un Protocole (ADI, 2006)

Un protocole est défini comme une séquence de phases de communication et de calcul. Une phase de communication est le transfert de messages d'une entité qu'on appelle émetteur vers une autre qu'on appelle récepteur, alors qu'une phase de calcul met à jour l'état interne de l'entité.

I. 1.1 Définition d'un protocole cryptographique (Bidan, et al., 1995)

C'est un protocole qui se base sur la cryptographie pour assurer certains objectifs de sécurité. Les phases de calcul des protocoles cryptographiques consistent soit en la génération d'aléas et/ou de clés de chiffrement, soit au chiffrement/déchiffrement de messages, soit en la vérification des signatures des messages.

Les objectifs des protocoles cryptographiques ont évolué dans le temps. Les premiers protocoles consistaient simplement à authentifier un utilisateur d'un système lors de sa connexion. Par la suite est apparue la nécessité de construire des protocoles cryptographiques permettant à chaque entité d'authentifier l'entité homologue. Ainsi il ne suffit plus au serveur d'authentifier l'utilisateur, il doit également s'authentifier auprès de cet utilisateur.

Pour que deux entités puissent s'authentifier, il est nécessaire que, soit les deux entités partagent une clé secrète commune dans le cas de l'utilisation d'un cryptosystème à clé secrète, soit chaque entité connaisse la clé publique de l'autre entité dans le cas de l'utilisation d'un cryptosystème à clé publique. Ces propriétés n'étant pas garanties, la notion de serveur d'authentification a été introduite, dans le cas d'un cryptosystème à clé secrète, chaque entité possède une clé secrète commune avec le serveur d'authentification ; alors que dans le cas d'un cryptosystème à clé publique, le serveur d'authentification connait la clé publique de chaque entité. Le serveur d'authentification doit être digne de confiance pour les entités l'utilisant comme serveur.

I.1.2 Système cryptographique (ADI, 2006)

Un système cryptographique ou cryptosystème est composé d'un algorithme de cryptage (chiffrement) et d'un algorithme de décryptage (déchiffrement).

Définition formelle d'un cryptosystème (ADI, 2006)

C'est un quintuplé (P, C, K , E,D)

P: ensemble de textes clairs.

C : ensemble de textes chiffrés.

K : ensemble de clefs.

E : un ensemble de fonctions $\{E_k| k\in K\}$ de cryptage (chiffrement) de P vers C.

D : un ensemble de fonctions $\{D_k| k\in K\}$ de décryptage (déchiffrement) de C vers P. satisfaisant: $\forall k\in K, \exists k^{-1}\in K |\forall m \in P: D_k^{-1}(E_k(m))=m$.

I.1.3 Notations (ADI, 2006)

1. Principal : c'est un agent qui participe dans une session de l'exécution du protocole.

2. Les Messages : ils sont constitués de **:**

L'identité d'un principal : A, B, C, etc.

L'identité d'un serveur : S.

Les Nonces : Na, Nb, etc.

Le Message m crypté avec une clé k : $\{m\}_{k.}$

Le Message composé : m, m'.

3. Les étapes de communication

Une communication est représentée par un numéro de l'étape, l'émetteur, le récepteur et enfin le message : i $A\longrightarrow B : m$

Exemple :

$$\begin{array}{llll} 1 & A & \longrightarrow B & : N_a \\ 2 & B & \longrightarrow A & : \{N_a\}_{k_b^{-1}} \end{array}$$

4. Rôle: c'est une abstraction du protocole où l'emphase est mise sur un seul agent.

Exemple:

Le protocole : $1\ A\rightarrow B :\ A$ ***Rôle de A :*** $\alpha.1\ A\ \rightarrow B\ :\ A$

$2\ B\rightarrow A:\ N_b$ $\alpha.2\ B\ \rightarrow A\ :\ N_b$

$3\ A\rightarrow B:\ \{N_b\}_{kas}$ $\alpha.3\ A\ \rightarrow B\ :\{N_b\}_{kas}$

$4\ B\rightarrow S:\ \{A,\{Nb\}_{kas}\}_{kbs}$

$5\ S\rightarrow B : \{Nb\}_{kas}$

5. Nonce, Timestamp : ce sont des entiers permettant d'assurer la fraîcheur d'un message.

Exemples : Message sans fraîcheur :

$$\begin{array}{llll} 1 & A & \longrightarrow B & : \text{ es-tu là} \\ 2 & B & \longrightarrow A & : \{\text{ oui, je suis là}\}_{k_b^{-1}} \end{array}$$

Message avec fraîcheur :

$$1 \quad A \longrightarrow B : \text{ es-tu là, } N_a$$
$$2 \quad B \longrightarrow A : \{ \text{ oui, je suis là, } N_a \}_{k_b^{-1}}$$

I. 1.4 Les types de protocoles cryptographiques (ADI, 2006)

Les protocoles cryptographiques diffèrent selon leurs objectifs.

A. Les protocoles d'authentification

On trouve les protocoles d'*Authentification de l'identité* et les protocoles d'*Authentification de messages (intégrité).* Le premier type permet à un principal de prouver son identité à un autre. La sécurité ici consiste à ne pas permettre à un principal *X* de prouver que son identité est *Y* ($X \neq Y$). Alors que le deuxième type, s'assurer que le contenu d'un message n'a pas été modifié. La sécurité dans ce type de protocole consiste en la capacité du récepteur à détecter une modification malicieuse ou accidentelle survenue en cours de route d'un message.

B. Les protocoles de distribution de clés

L'objectif de ce type de protocole est de distribuer de nouvelles clés (symétriques ou publiques) aux principaux pour qu'ils s'en servent pendant leurs communications futures, la sécurité ici, est d'assurer la *confidentialité*, c'est-à-dire la non divulgation de la clé à un principal qui n'est pas supposé la connaître ; et l'*intégrité* c'est-à-dire toute modification de la clé devrait être détectée par ses récepteurs.

C. Protocole de commerce électronique

C'est un protocole cryptographique qui permet d'acheter ou de vendre des biens ou des services sur Internet ; en plus de la confidentialité et de l'authentification, l'analyse de protocoles de commerce électronique nécessite la définition de propriétés de sécurité spécifiques telles l'anonymat, la non répudiation, la garantie de livraison, l'atomicité de l'argent, des biens et services. L'anonymat des participants est une application particulière de la propriété de confidentialité qui exige que l'identité (nom, courriel, adresse, emplacement, etc.) d'un certain participant à un protocole demeure cachée aux autres participants.

I.1.5 Les Failles dans les protocoles cryptographiques

Un protocole cryptographique contient une faille, s'il ne remplit pas les exigences sécuritaires pour lesquelles il a été conçu (ADI, 2006)**.** Une faille peut contenir un nombre important d'étapes de communication, elle est difficile à trouver manuellement, d'où le besoin d'un outil automatique qui aide à trouver des failles. Les principales failles des protocoles cryptographique sont les : failles de fraîcheur, failles d'oracle, failles d'association, failles de type, failles d'implantation, failles de répudiation.

A. ***Failles de fraîcheur***

Cas où un intrus attaque un protocole en utilisant des informations récupérées des sessions précédentes. (ADI, 2006)

B. Failles d'oracle

C'est le cas où un intrus parvient à utiliser le protocole afin de connaître des informations secrètes ou des messages utiles pour attaquer le protocole. (ADI, 2006)

C. Failles d'association

C'est le cas où un intrus parvient à corrompre la correspondance entre la clé publique d'un principal et son identité. (ADI, 2006)

D. Failles de type

C'est le cas ou un intrus base son attaque sur la substitution d'un champ (ou une composante) d'un message par un champ de type différent. (ADI, 2006)

E. Failles d'implantation

C'est le cas où un intrus profite des faiblesses dues à la combinaison de l'algorithme de chiffrement et du protocole cryptographique. (ADI, 2006)

F. Failles de répudiation

Une faille de répudiation survient lorsqu'un principal peut nier sa participation à une session du protocole. (ADI, 2006)

I.2 Propriétés de sécurité

La propriété de sécurité est toute propriété qu'un protocole cherche à assurer, elle dépend de ses objectifs. Les propriétés les plus courantes sont le secret et l'authentification. On dira en général qu'un protocole assure le secret d'une donnée si l'intrus ne peut pas connaître cette donnée. Les propriétés d'authentification sont des propriétés de la forme : si l'agent B reçoit un message de la forme m1 alors l'agent A a envoyé un message de la forme m2 (qui est en correspondance avec le premier). Ainsi, dans le cas du protocole de Needham-Schroeder, on peut formuler la propriété suivante : si l'agent A émet le message $\{Nb\}_{pub(B)}$, alors le nonce Nb a bien été engendré par l'agent B (Lafrance, 2005). La propriété la plus étudiée est le secret : un terme donné reste-t-il un secret gardé entre les différents acteurs honnêtes du protocole, même en présence d'un intrus? (Bernat, 2006).

De par le nombre important d'attaques possibles sur les protocoles et la difficulté de déceler les attaques dues aux failles de conception des protocoles, il est nécessaire d'analyser les protocoles de sécurité afin de prouver leurs validité, l'analyse commence nécessairement par une phase de

modélisation pour une spécification rigoureuse du protocole de sécurité. La validation des protocoles requiert donc le développement de méthodes formelles permettant la détection de toute attaque possible sur un protocole donné.

II Modélisation des protocoles de sécurité

Une variété de modèles qui se base sur des formalismes différents est utilisée pour la spécification et l'analyse des protocoles cryptographiques, les principaux sont : les modèles orientés algèbres(ou recherches), les modèles orientées logiques et les réseaux de Petri.

II. 1 Modèles algébriques des protocoles cryptographiques

Dans les modèles orientés algèbres ou orientés recherche**,** le langage de spécification des protocoles cryptographiques utilisé est un langage algébrique (Bidan, et al., 1995), il sert à décrire la connaissance de l'attaquant sous forme ensembliste en général, ces modèles considèrent le protocole cryptographique du point de vue de l'attaquant. Les règles de réécriture sont également décrites dans le même langage algébrique et sont des techniques de réécriture d'ensembles, union et/ou intersection d'ensembles, à partir des données que l'intrus peut écouter, intercepter ou modifier. Des hypothèses sont également faites concernant la capacité de l'attaquant à écouter, à intercepter et à faire passer des messages sur le réseau.

Une recherche des scénarios d'attaque est réalisée en examinant quelle information l'adversaire peut obtenir ou apprendre. L'objectif est de trouver une séquence de messages où au moins un acteur aboutit à un état local qui est interdit par les spécifications du protocole cryptographique c'est-à-dire, il existe au moins un axiome qui n'est plus vérifié.

II.1.1 Modèle de Dolev et Yao

La littérature concernant les méthodes orientées algèbres est dense. Dolev et Yao en 1983 (Dolev, et al., 1983) ont proposé un modèle de spécification basé sur les règles de réécriture de termes qui sont des règles de réduction algébrique. Le modèle de Dolev et Yao suppose que les messages envoyés et reçus ne sont ni des nombres ni des suites de bits, mais des éléments d'une algèbre de termes. Dans ce modèle le réseau informatique est supposé être sous le contrôle de l'intrus qui peut, altérer et détruire les messages, créer des messages et exécuter toute opération disponible aux utilisateurs légitimes du système telle que l'opération de cryptage. Toutefois, il est supposé que l'intrus ne connait aucune information qui doit être maintenue secrète, telles que les clés de cryptage appartenant aux utilisateurs honnêtes du système. Les mots sont obtenus suivant des règles de réductions, comme exemple de règle de réduction, la règle d'annulation de terme

d(e(m,k),k)→m (le décryptage et le cryptage avec la même clé). Ainsi l'intrus manipule un système de réécriture de termes. Si le but de l'intrus est de découvrir un message secret, alors le problème qui consiste à prouver que le protocole est sécurisé, est équivalent à prouver que certains mots ne pourront pas être générés par le système de réécriture de terme.

La limite de ce modèle (Lee, et al., 1997) est la spécification uniquement du secret des messages initiaux sans spécifier l'authentification des messages et des objets, l'approche n'est pas automatisée, et aussi le modèle ne peut s'appliquer que sur une classe de protocoles, les protocoles « ping-pong » (protocoles de distribution des clés publiques, qui se base sur l'échange de messages entre deux entités du protocole). Le modèle n'est utilisé que pour prouver la défaillance de la sécurité du protocole. Néanmoins le modèle de Dolev et Yao a servi de support à de nombreux modèles et méthodes de vérifications. En effet dans la plupart de ces modèles, les actions de l'intrus sont décrites par un ensemble succinct de règles où apparaissent la capacité de l'intrus à déchiffrer des messages lorsqu'il a l'inverse de la clef ou sa capacité à former de nouveaux messages. De tels modèles sont souvent appelés « modèles à la Dolev-Yao ».

II.1.2 Modèle de Woo et Lam

Woo et Lam dans (Y.C.Woo, et al., 1993) ont proposé un modèle algébrique et des spécifications sémantiques pour une définition formelle de la correction du protocole en considérant les propriétés du secret et de l'authentification, dans les protocoles de distribution de clés. Ils ont identifié deux propriétés de correction concernant les protocoles d'authentification, le secret et la correspondance. La propriété du secret est déjà connue, alors que la correspondance traite l'authentification et souligne que certains évènements ne se produisent que si d'autres évènements se sont déjà produits. Des assertions pour spécifier ces propriétés ont été définies, ainsi qu'une sémantique formelle pour la satisfaction de ces assertions dans un modèle sémantique. La sémantique de la spécification du protocole est donnée par l'ensemble de ses exécutions possibles et l'approche adoptée pour caractériser ces exécutions est basée sur les transitions d'états.

Exemple d'application de ce modèle sur le protocole d'Otway-Rees : C'est un protocole d'authentification mutuelle entre deux principaux et un troisième principal de confiance, le but du protocole est d'assurer l'authentification mutuelle et la distribution de la clé secrète de session K.

P et ***Q*** sont les principaux qui cherchent à s'authentifier, ***S*** est le serveur d'authentification qui a la confiance de ***P*** et ***Q,*** et il est aussi responsable de la génération de la nouvelle clé de session K.

$$
\begin{array}{lll}
(1) & P \rightarrow Q: & n, P, Q, \{n_P, n, P, Q\}_{k_{PS}} \\
(2) & Q \rightarrow S: & n, P, Q, \{n_P, n, P, Q\}_{k_{PS}}, \\
 & : & \{n_Q, n, P, Q\}_{k_{QS}} \\
(3) & S \rightarrow Q: & n, \{n_P, k\}_{k_{PS}}, \{n_Q, k\}_{k_{QS}} \\
(4) & Q \rightarrow P: & n, \{n_P, k\}_{k_{PS}}
\end{array}
$$

La représentation formelle du protocole Otway-Rees dans le modèle de Woo et Lam est :

({(l) (2)}, {Initiator(i), Responder(r), Server(S)})

Donc le protocole Otway-Rees dans le modèle de Woo et Lam devient :

Initial Conditions

(1) $\forall x, y : x$ **has** y
(2) $\forall x, y : y$ **has** $key(x, \mathrm{S}) \Leftrightarrow [x = y \vee y = \mathrm{S}]$

Initiator (i) Protocol

(I1): **BeginInit** (r)
(I2): **NewNonce** (n)
(I3): **NewNonce** (n')
(I4): **Send** $(r, [n, i, r, \{n', n, i, r\}_{key(i,\mathrm{S})}])$
(I5): **Receive** $(r, [n, \{n', \mathrm{K}\}_{key(i,\mathrm{S})}])$
(I6): **Accept** (K)
(I7): **EndInit** (r)

Responder (r) Protocol

(R1): **BeginRespond** (i)
(R2): **Receive** $(i, [\mathrm{N}, i, r, \mathrm{X}])$
(R3): **NewNonce** (n)
(R4): **Send** $(\mathrm{S}, [\mathrm{N}, i, r, \mathrm{X}, \{n, \mathrm{N}, i, r\}_{key(r,\mathrm{S})}])$
(R5): **Receive** $(\mathrm{S}, [\mathrm{N}, \mathrm{Y}, \{n, \mathrm{K}\}_{key(r,\mathrm{S})}])$
(R6): **Send** $(i, [\mathrm{N}, \mathrm{Y}])$
(R7): **Accept** (K)
(R8): **EndRespond** (i)

Server (S) Protocol

(S1): **Receive** $(r, [\mathrm{N}, i, r, \{\mathrm{X}, \mathrm{N}, i, r\}_{key(i,\mathrm{S})}, \{\mathrm{Y}, \mathrm{N}, i, r\}_{key(r,\mathrm{S})}])$
(S2): **NewSecret** $(\{i, r\}, k)$
(S3): **Send** $(r, [\mathrm{N}, \{\mathrm{X}, k\}_{key(i,\mathrm{S})}, \{\mathrm{Y}, k\}_{key(r,\mathrm{S})}])$

Les assertions de correspondance de ce protocole correspondent à :

$$
\begin{array}{ll}
\textbf{(P.(I7), EndInit(Q))} \hookrightarrow & \\
\quad \textbf{(Q.(R1), BeginRespond(P))} & (1) \\
\textbf{(Q.(R8), EndRespond(P))} \hookrightarrow & \\
\quad \textbf{(P.(I1), BeginInit(Q))} & (2)
\end{array}
$$

L'assertion **(1)** Spécifie que chaque fois que **P** applique la transition **Endlnit(Q),** Q doit appliquer la transition **BeginResponse(P)** en premier. Chaque fois que **P** termine l'exécution de l'authentification échangée avec **Q, Q** doit avoir appliqué au moins sa première étape dans un tel échange. Il y a autant de transitions **BeginResponse (P)** par **Q** que **Endlnit (Q)** par **P.**

L'assertion **(2)** spécifie une propriété semblable, mais de point de vue de **Q**.

(1) et **(2)** garantissent ensemble que les échanges d'authentification entre **P** et **Q** considèrent les soucis d'authentification.

L'assertion de secret correspond à :

$$\forall x, p : x \text{ has } key(p, \mathsf{S}) \Leftrightarrow [x = p \vee x = \mathsf{S}]$$

Elle est identique à la condition initiale (2), qui stipule que la clé sécrète de chaque principal doit être partagée uniquement entre le principal même et **S.**

Le protocole est correct s'il satisfait toutes les assertions de correspondance et de secret.

II.1.3 L'analyseur NRL

Des outils qui se basent sur une modélisation algébrique du protocole ont été développés, parmi ces outils, l'analyseur de protocole NRL (Meadow, 1992), implémenté en Prolog. Il se base sur le modèle de Dolev et Yao, mais il est plus général, puisque il permet de vérifier une variété de propriétés de sécurité sur une diversité de protocoles. Pour vérifier le protocole de sécurité en utilisant l'analyseur NRL, on doit d'abord spécifier le protocole par un ensemble de règles de transitions et spécifier les besoins de sécurité par des états non sécurisés, et spécifier aussi les connaissances initiales de l'intrus et de chaque principal honnête. Les règles de transition décrivent les actions exécutées par les principaux honnêtes, envoi ou réception de messages, et les actions exécutées par l'intrus, chiffrement, déchiffrement etc. dans le but de produire de nouveaux messages.

L'outil est utilisé pour trouver les failles du protocole de sécurité en spécifiant les états non sécurisés, et essayer de trouver un chemin vers ces états à partir de l'état initial. Cet outil a été utilisé pour trouver des failles non découvertes auparavant et pour démontrer des failles déjà connues. Cependant cet outil de spécification et d'analyse présente des inconvénients, parmi ses points faibles, il y a le problème d'arrêt, à cause de l'espace d'état infini, dû à la recherche exhaustive. Ce problème est commun à toutes les approches algébriques (de recherches). Une autre limite concerne le manque de lisibilité dû au fait que la spécification est textuelle.

II.2 Modèles basés sur les Algèbres de processus

II.2.1 Algèbre de processus

Une algèbre de processus ou calcul de processus est un modèle mathématique défini par une syntaxe et une sémantique. La syntaxe comprend un nombre réduit d'opérateurs algébriques primitifs (composition séquentielle, parallèle, choix non déterministe, …) qui par assemblage permettent de décrire des comportements complexes. La sémantique est définie formellement de

manière axiomatique et opérationnelle. Une sémantique axiomatique consiste en un ensemble de lois algébriques (commutativité, associativité, distributivité, …) permettant de démontrer l'équivalence des termes. La sémantique opérationnelle consiste en une relation de transition $B \xrightarrow{a} B$ exprimant le fait que le terme *B* peut effectuer l'action *a*, puis évoluer et se transformer en un terme *B'*.

Parmi les premiers calculs de processus, on peut citer *CCS* (Calculus of communicating systems) introduit par Milner (Milner, 1989) et qui apparaît comme un paradigme de calcul destiné à servir de base théorique à la programmation concurrente. Il décrit les interactions entre processus comme des communications binaires, synchronisées à travers des canaux nommés (*x, y,…).* Une communication se produit dans un système lorsque deux processus effectuent une action complémentaire (écouter et émettre sur un même canal) au même instant. Cette algèbre de processus modélise la communication par message à l'exclusion d'un éventuel contenu de cette communication. Autrement dit, elle se contente seulement d'étudier la synchronisation entre les différents composants d'un système. Le *Pi-calcul* est une émanation directe de CCS qui introduit la notion de mobilité pour la spécification de processus mobiles concurrents.

II.2.2 Spi-calcul

Le Spi-calcul est un langage de processus appliqué aux protocoles cryptographiques. Il s'agit d'une extension du Pi-calcul introduite par Abadi & Gordon (Abadi, et al., 1999) avec des primitives cryptographiques. Il permet en particulier d'exprimer l'envoi et la réception de messages, la création de nonces, la réplication et la mise en parallèle de processus. L'intérêt principal du Spi-calcul est de permettre une définition extrêmement précise des propriétés de secret et d'authentification. Sa syntaxe est composée d'un opérateur *d'extraction de couple* (let (x, y) = m in P), d'un opérateur de *décryptage* (case m of $\{x\}_k$ in P) et d'un opérateur *d'énumération d'entiers* (case m of 0 : P succ(x) : Q).

SPI est équipé de mécanismes pour la communication et la synchronisation basés sur l'interaction entre processus. Durant cette interaction, un message est transmis via un canal d'un processus à un autre. L'idée d'utiliser Spi calcul pour la vérification des protocoles cryptographiques consiste à spécifier le protocole à vérifier en termes de processus parallèles, et spécifier la propriété de sécurité en termes d'équivalence entre des processus. L'intrus n'est pas modélisé explicitement.

Abadi a aussi développé des principes et des règles de typage pour le spi calcul utiles pour la vérification de propriétés de sécurité dans les protocoles de sécurité. Ces règles garantissent qu'un protocole ne divulguera pas ses données secrètes lorsque son typage est correct (Lafrance, 2005).

Syntaxe de Spi-calcul

Un processus peut avoir une des formes suivantes :

P, Q ::=		processus	
$\overline{M}\langle N\rangle$.P		output	
M(*x*) .P	$x \in X$	input	
P | Q		composition	
(*vn*) P		restriction	
!P		replication	
[*M is N*] P		match	si M et N sont les mêmes alors l'exécution continue sinon l'exécution s'arrête.
0		nil	
let (x, y)=M in P	x ,y$\in$X	pair splitting	si M est égal à (N,L) alors l'exécution est *P[N/x][L/y]*.
case M of 0: P suc(x) : Q	$x \in X$	integer case	si M es 0 alors l'exécution continue en P, si M est suc(N) alors l'exécution est *Q[N/x]*,sinon l'exécution s'arrête.
case L of $\{x\}_K$ *in P*	$x \in X$	description de clé partagée	Le processus essaye de décrypter L avec la clé K : si L est $\{M\}_K$ alors on calcule *P* [*M*/*x*], sinon le processus s'arrête.

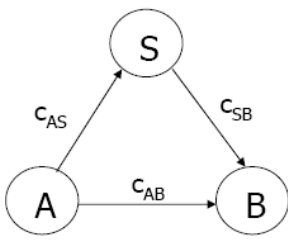
Tableau 1 : Syntaxe de Spi-calcul.

Exemple d'application de Spi calcul sur une version simplifiée du protocole Wide Mouthed Frog : C'est un protocole d'échange de clé. Les principaux A et B partagent la clé K_{AS} et K_{SB} respectivement, avec le serveur S, quand A et B voudront communiquer secrètement, A crée une nouvelle clé K_{AB}, l'envoie au serveur sous K_{AS}, et le serveur l'expédie à B sous K_{SB}. Puisque toute la communication est protégée par le chiffrement, elle peut se faire à travers des canaux publique c_{AS} , c_{SB} , et c_{AB} :

Message 1 $A \rightarrow S: \{K_{AB}\}_{K_{AS}}$ on c_{AS}

Message 2 $S \rightarrow B: \{K_{AB}\}_{K_{SB}}$ on c_{SB}

Message 3 $A \rightarrow B: \{M\}_{K_{AB}}$ on c_{AB}



Avec le spi calcul, cette séquence de messages est exprimée par :

$$A(M) \triangleq (\nu K_{AB})(\overline{c_{AS}}\langle \{K_{AB}\}_{K_{AS}}\rangle . \overline{c_{AB}}\langle \{M\}_{K_{AB}}\rangle)$$
$$S \triangleq c_{AS}(x).case\ x\ of\ \{y\}_{K_{AS}}\ in\ \overline{c_{SB}}\langle \{y\}_{K_{SB}}\rangle$$
$$B \triangleq c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in$$
$$c_{AB}(z).case\ z\ of\ \{w\}_{y}\ in\ F(w)$$
$$Inst(M) \triangleq (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B)$$

Où F (w) est une abstraction du comportement du processus B après la réception de w.
Inst(M) décrit une instance du protocole. *A(M)* , *S* et *B* décrivent les principaux du protocole.
Pour vérifier l'authentification, A est maintenue, B est remplacé par une variante B_{spec}(M) qui se comporte comme B lorsque il reçoit le message de A. La spécification devient :

$$A(M) \triangleq (\nu K_{AB})(\overline{c_{AS}}\langle \{K_{AB}\}_{K_{AS}}\rangle . \overline{c_{AB}}\langle \{M\}_{K_{AB}}\rangle)$$
$$S \triangleq c_{AS}(x).case\ x\ of\ \{y\}_{K_{AS}}\ in\ \overline{c_{SB}}\langle \{y\}_{K_{SB}}\rangle$$
$$B_{spec}(M) \triangleq c_{SB}(x).case\ x\ of\ \{y\}_{K_{SB}}\ in$$
$$c_{AB}(z).case\ z\ of\ \{w\}_{y}\ in\ F(M)$$
$$Inst_{spec}(M) \triangleq (\nu K_{AS})(\nu K_{SB})(A(M) \mid S \mid B_{spec}(M))$$

Les propriétés (propriétés d'équivalence observationnelle) de l'authentification et du secret sont définies par les spécifications suivantes :
Authentification: $Inst(M) \simeq Inst_{spec}(M)$, *for all M*
Secret: $Inst(M) \simeq Inst(M')$ *if* $F(M) \simeq F(M')$ *for all M et M'*
Prouver ces propriétés revient à prouver que : le processus *Inst(M)* est équivalent au processus
$Inst_{spec}(M)$ pour tout message M.
Et que *Inst(M)* est équivalent à *Inst(M')* si *F(M)* est équivalent à *F(M')* pour tout M et M'.

II.2.3 Autres Modèles

D'autres modèles de spécification et d'analyse des protocoles cryptographiques qui se basent sur l'algèbre des processus ont été développés, parmi ces modèles on trouve le modèle Crypographic Security Process Algebra (CSPA ou CryptoSPA) développé par Focardi et Gorrieri (Focardi, et al., 1994/1995), Security Protocols Process Algebra (SPPA) (Lafrance, et al., 2003).
Cependant on note que la spécification avec les algèbres de processus est compliquée, d'autant plus qu'il n'existe actuellement aucun outil automatique manipulant des processus en CSPA ou en spi-calcul ou en SPPA.

II.3 Les Modèles de protocoles basés sur la logique

Les méthodes orientées logiques ne considèrent pas le protocole cryptographique du point de vue de l'attaquant. La plupart de ces logiques soulignent l'évolution de la connaissance pour chaque acteur du protocole y compris l'attaquant en prenant en compte les messages transmis (Bidan, et al., 1995). Le langage de spécification des protocoles est une logique spécialement construite pour modéliser ces protocoles. Le langage est formel et inclut des axiomes et/ou des règles d'inférences. Plus précisément, les logiques utilisées sont des logiques modales qui permettent de modéliser des concepts naturels tels que la connaissance ou la croyance. La preuve de la sécurité ou non des protocoles cryptographiques consiste ici à prouver des théorèmes qui modélisent les propriétés de sécurité de ces protocoles.

II.3.1 La logique BAN

La première logique modale introduite pour analyser les protocoles cryptographiques est la logique de **B**urrows **A**badi **N**eedham, la logique BAN (Burrows, 1989). C'est une logique de croyance introduite en 1990, qui modélise les croyances des principaux impliqués dans un protocole et l'évolution de ces croyances après échange de messages durant l'exécution du protocole. L'analyse d'un protocole se fait sur des étapes, la première étape consiste à idéaliser le protocole (enlever les messages non cryptés et les messages cryptés qui ne contribuent pas à l'évolution des connaissances des agents), ensuite vient l'étape d'écriture des hypothèses initiales et l'ajout des assertions après chaque étape de communication *les* assertions sont de la form*e* $P \equiv X$ ou de la forme Q $\supset$X, la premier assertion contient les hypothèses initiales et la dernière représente la conclusion. [Hypothèses]S^1 [assertion 1] S2 [assertion 2]….S^{n-1}[assertion n] [conclusions], enfin l'utilisation des postulats (axiomes) de la logique pour déduire d'autres croyances.

Syntaxe de BAN

$P \mid\equiv X$	P croit X : le principal peut agir comme si X est vraie	$\langle X \rangle_Y$	Message X combiné avec un message Y :Y prouve l'origine du message.
$P \triangleleft X$	P voit X : la formule est vraie si le principal reçoit le message X lors de l'exécution courant du protocole	$\sharp(X)$	La formule X est fraîche : x n'a pas été envoyé avant l'exécution courante du protocole.
$P \mid\sim X$	P a dit X : la formule est vraie si le principal envoie le message X lors de l'exécution courante du protocole	$P \overset{K}{\leftrightarrow} Q$	La clé K est partagée entre P et Q.
$P \Rightarrow X$	P a une juridiction sur X	$\overset{K}{\mapsto} P$	P a une clé publique K.
$P \overset{X}{\rightleftharpoons} Q$	Secret X partagé entre P et Q	$\{X\}_K$	Message X crypté par une clé K.

Tableau 2 : La syntaxe de la logique BAN.

Les principales règles d'inférences de la logique BAN (Postulats)

$\frac{P \models Q \overset{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \models Q \sim X}$	**R1** :Si P croit qu'il partage la clé K avec Q et voit un message X crypté avec K alors on peut inférer que P croit que Q a dit X.
$\frac{P \models \overset{K}{\mapsto} Q, \quad P \triangleleft \{X\}_{K^{-1}}}{P \models Q \sim X}$	**R2** :La même que la précédente mais pour la clé publique.
$\frac{P \models \sharp(X), \quad P \models Q \sim X}{P \models Q \models X}$	**R3** : Clarifie la différence entre « dire » et « croire ». En fait le principal P croit que Q croit X si le principal P croit que Q a dit X et P croit que X est frais.
$\frac{P \models Q \Rightarrow X, \quad P \models Q \models X}{P \models X}$	**R4** :Montre comment les principaux peuvent ajouter de nouveaux messages à leurs croyances.Si Q croit X, il ne peut transmettre ces croyances à un autre principal P que si ce principal croit que Q a une juridiction sur X .
$\frac{P \models Q \overset{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \triangleleft X}$	**R5** :Si P croit qu'il partage la clé K avec Q et il voit le message X crypté avec K alors on conclue que P voit le message X.

Tableau 3 : Les principales règles d'inférences de la logique BAN.

La proprieté d'authentification

Pour quelques protocoles, la propriété d'authentification est spécifiée par la formule de croyance de premier niveau suivante :

$$P \models P \overset{k}{\rightarrow} Q \quad \text{et} \quad Q \models P \overset{k}{\rightarrow} Q$$

Cependant pour d'autres protocoles une deuxième formule est nécessaire :

$$P \models Q \models P \overset{k}{\rightarrow} Q \quad \text{et} \quad Q \models P \models P \overset{k}{\rightarrow} Q$$

Exemple sur le protocole de Otway-Res

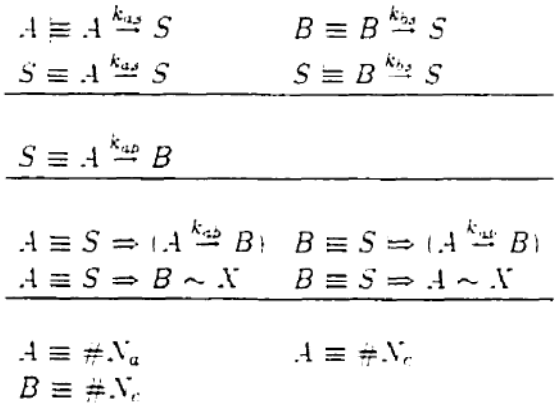
$$\begin{array}{lll} Message\ 1 & A \longrightarrow B : & M.A.B.\{N_a.M.A.B\}_{k_{as}} \\ Message\ 2 & B \longrightarrow S : & M.A.B.\{N_a.M.A.B\}_{k_{as}}.\{N_b.M.A.B\}_{k_{bs}} \\ Message\ 3 & S \longrightarrow B : & M.\{N_a.k_{ab}\}_{k_{as}}.\{N_b.k_{ab}\}_{k_{bs}} \\ Message\ 4 & B \longrightarrow A : & M.\{N_a.k_{ab}\}_{k_{as}} \end{array}$$

La phase d'idéalisation : consiste à spécifier les messages, transmis à l'exécution du protocole, par des formules de logique Ban.

$$\begin{array}{lll} Message\ 1 & A \longrightarrow B : & \{N_a.N_c\}_{k_{as}} \\ Message\ 2 & B \longrightarrow S : & \{N_a.N_c\}_{k_{as}}\{N_b.N_c\}_{k_{bs}} \\ Message\ 3 & S \longrightarrow B : & \{N_a.(A \overset{k_{ab}}{\rightarrow} B).B \sim N_c\}_{k_{as}}. \\ & & \{N_b.(A \overset{k_{ab}}{\rightarrow} B).A \sim N_c\}_{k_{bs}} \\ Message\ 4 & B \longrightarrow A : & \{N_a.(A \overset{k_{ab}}{\rightarrow} B).B \sim N_c\}_{k_{as}} \end{array}$$

Avec N_c dénote le message *M.A.B*

é***criture des hypothèses initiales :*** toutes les hypothèses initiales adoptées dans le protocole devraient être spécifiées en formule de logique BAN.



Le premier groupe d'hypothèses concerne les croyances des principaux sur les clés partagées initiales. Le deuxième groupe concerne les clés nouvellement générées. Finalement le troisième et quatrième groupe concernent respectivement les hypothèses de juridiction et de fraîcheur.

Ajout des assertions apres chaque étape de communication

Après l'exécution du message 1 du protocole , l'assertion 1 devient $[A_0.B \lhd \{N_a,N_c\}]$ avec A_0 dénote les hypothèses initiales.

Utilisation des postulats de la logique pour déduire d'autres

Si on applique les règles R1, R3 et R4 sur l'assertion S^3 on découvre qu'à l'étape 3 le principal B croit A $\xrightarrow{Kab} B$ et A$|\sim N_c$. Si on applique ces règles à l'étape S^4 on découvre que A est capable de croire A $\xrightarrow{Kab} B$ et B $|\sim N_c$ à l'étape 4.

Limite de la logique

Le modèle BAN se focalise sur l'authentification alors que le secret est laissé implicite. La phase d'idéalisation a été critiquée dans plusieurs travaux parmi ces travaux, citons (Y.C.Woo, et al., 1993), les auteurs soulignent que le protocole d'authentification est specifié par une séquence de formules logiques, cette spécification est obtenue par un processus informel d'idéalisation qui requiert une parfaite compréhension du protocole, ils trouvent que l'idealisation n'est pas souhaitable à cause de l'écart sémantique potentiellement large, entre le protocole d'origine et la version idéalisée. Mais cette logique reste encore aujourd'hui l'une des plus importantes et la plus étudiée.

II.3.2 Autres modèles

On peut citer d'autres approches formelles qui se sont basées sur la logique, il s'agit de la logique CKT5 qui fut introduite par Bieber en 1990, mais elle n'a traité que le secret des objets et l'authentification des messages, mais pas l'authentification de l'identité (Lee, et al., 1997). Bien que de nombreuses méthodes orientées logiques existent dans la littérature, aucune d'entre elles ne peut se prévaloir de détecter toutes les failles des protocoles.

II.4 Les modèles basés sur les Réseaux de Petri

II.4.1 Les Réseaux de Petri

Les réseaux de Petri (RdP) constituent des modèles graphiques et mathématiques permettant de modéliser le comportement dynamique des systèmes à évènements discrets comme les systèmes de télécommunications et les réseaux de transport. Leur représentation graphique permet de visualiser d'une manière naturelle le parallélisme, la synchronisation, le partage de ressources, les choix (conflits). Leur représentation mathématique permet d'établir les équations d'états, à partir desquelles il est possible d'apprécier les propriétés du modèle et de les comparer avec le comportement du système modélisé (Stephenson, 2004).

Depuis la création des réseaux de Petri par Carl Adam Petri en 1962 de la faculté de mathématiques et de physique de l'université technique de Darmstadt à l'ouest de l'Allemagne, de nombreuses extensions ont été apportées, soit pour améliorer la concession du modèle (réseaux de Petri Colorés), soit pour augmenter la puissance de spécification de cet outil (les réseaux de Petri temporisés, les réseaux de Petri temporels, les réseaux de Petri stochastiques,…).

En général, la notion de couleur est utilisée pour contrôler les conditions de franchissement de transition. Dans ces cas, les places génèrent les jetons colorés, alors les transitions peuvent nécessiter des jetons de couleurs spécifiques pour leurs franchissements. De tels réseaux sont appelés les réseaux de Petri colorés (CPN). D'autres variantes des réseaux de Petri, nommées les réseaux de Petri stochastiques temporels sont utilisés pour la modélisation des systèmes. Dans cette famille de réseaux, on trouve les réseaux de Petri stochastiques (SPNs) (Murata, 1989) et les réseaux de Petri stochastiques généralisés (GSPNs), qui supposent que le temps de franchissement des transitions est une distribution exponentielle.

II.4.2 Définition de base

Un réseau de Petri est un graphe biparti constitué de places, de transitions et d'arcs orientés qui relient les transitions aux places et les places aux transitions. Il est représenté par un quadruplet $N=<P, T, Pré, Post>$ où :

P : ensemble fini de places P= (p1, p2,…, pn),

T : ensemble fini de transitions T= (t1,t2,…,tn),

Pré : application de PxT→N (ensemble des entiers naturels) correspondante aux arcs orientés reliant les places aux transitions,

Post : application de PxT→N correspondante aux arcs orientés liant les transitions aux places.

Le marquage

On appelle marquage d'un réseau de Petri *N= (P, T, Pré, Post)* la fonction *M : P* →N.

Le marquage peut être représenté par un vecteur M = (*M(p1),M(p2),….,M(pn)*). On appelle réseau marqué tout couple (*N, M_0*) où *N* est un réseau de Petri et M_0 un marquage appelé *marquage initial.*

Pour une place *p*, *M(p)* représente le nombre de marques ou de jetons dans la place *p*, les jetons sont représentés par des points noirs à l'intérieur de la place.

Une place contient n jetons, cela signifie par exemple la disponibilité de n ressources, n processus attendent d'entrer en section critique ...

Exemple d'un réseau de Petri

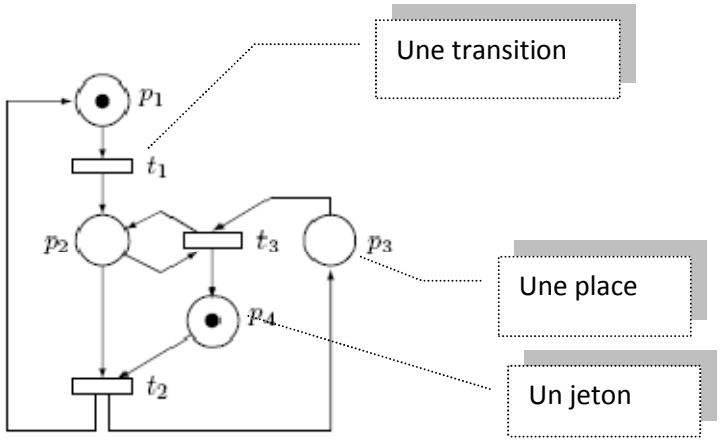


Figure 5 : Exemple d'un réseau de Petri.

Sensibilisation d'une transition

Soit un réseau de Petri N, t une transition de N et M un marquage. La transition t est sensibilisée ou franchissable ou tirable pour le marquage M si et seulement si :∀p ∈P, M(p) >Pre(t, p) On note par M[t> que la transition t est sensibilisée par le marquage M.

Tir d'une transition

Dans un réseau de Petri N, toute transition sensibilisée par un marquage M peut être tirée (franchie) et son tir conduit à un nouveau marquage M' défini, pour tout p, par :

M'(p) = M(p) - Pre(t, p) + Post(t, p)

Le franchissement de la transition t est noté : M[t>M'.

Modélisation des protocoles cryptographiques avec les réseaux de Petri

Les méthodes de spécification et d'analyse formelle présentées auparavant sont complexes et manque de représentation graphique. Les réseaux de Petri sont connus comme étant des modèles appropriés pour les protocoles caractérisés par des propriétés de parallélisme, de concurrence et d'asynchronisation et offrent une flexibilité qu'on ne trouve pas dans les autres méthodes. En 1990 Varadhrajan (Varadharajan, 1990) a trouvé que le réseau de Petri est un modèle utile pour les flux d'informations et la donnée secrète dans la sécurité des systèmes. Les recherches de Nieh et Tavares (Nieh, et al., 1992) et de Stal (Stal, et al., 1994) ont abouti à une spécification graphique de certains protocoles qui présentaient des failles en utilisant les réseaux de Petri colorés. L'arbre d'accessibilité et les scenarios d'attaques ont été générés pour trouver les états non sécurisés en insérant des modèles d'intrus dans les spécifications résultantes.

II.4.3 Le modèle CTPN (Cryptographique Timed Petri Net)

Dans (Lee, et al., 1997), les auteurs ont présenté un modèle de spécification et d'analyse des protocoles cryptographiques basé sur les réseaux de Petri qu'ils ont appelé CTPN (cryptographic Timed Petri Net) et qu'ils considèrent comme une extension des réseaux de Petri, utile dans le domaine de la sécurité informatique et des réseaux. Ce modèle permet la spécification graphique du protocole et des actions de l'intrus sur le protocole, où les places représentent les acteurs et les variables du protocole et les transitions temporelles représentent les différentes primitives (de communication et de cryptographie) du protocole avec le temps nécessaire à leurs exécutions. Le module CTPN-Langage, traduit cette spécification en texte, une analyse est appliquée sur cette spécification textuelle par un module appelé CTPN-Analyzer (figure 6) et les différents scénarios d'attaques sont générés ainsi que la durée de l'attaque.

CTPN définit des classes de places, des classes de transitions, d'arcs, et des transitions temporisées (qui sont annotées par des délais temporels constants). Des règles de spécifications pour chaque élément du CTPN et du protocole ont été définies.

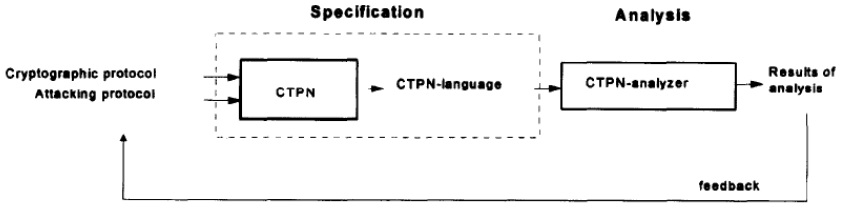


Figure 6 : La structure du CTPN.

CTP	Le protocole cryptographique
Normale Transition	Les fonctions ou algorithmes cryptographiques partiellement ordonnés ; fonctions d'envoi/réception, jointures de données élémentaires dans le message envoyé ; scinder le message reçu en donnée élémentaire, les fonctions locales et indépendantes de chaque principal.
Crypto Transition	Fonction ou algorithme cryptographique (cryptage, décryptage).
Timed Transition	Délai d'envoi /réception de message et le temps de calcul pour les algorithmes crypto.
Module Transition	Module et sous système.
Place Normale	Variable (e.g., nom du principal, nonce, clé), flot de contrôle, buffer du canal de communication.
initiale et finale Place	Les états initiaux et finaux du protocole.
Ciphertext Place	Canal contenant une donnée cryptée.

Tableau 4 : Exemple de règles de spécification pour les classes de transitions et de places.

Le Modèle CTPN est défini comme suit :

CTPN = (P, T, I, O, M)

P : est une collection de classes de places ; zéro ou plus places appartenant à la classe « *normal-places* »pour spécifier les variables du protocole telles que la clé, les principaux, les nonces etc..;

Places appartenant à la classe « *ciphertext-places* » représentées par double cercle pour spécifier que le jeton dans la place représente une donnée ou un message crypté ;

Et une ou plusieurs places appartenant à la classe « *initial –places* » et « *final-places* » pour spécifier le marquage initial et final,

T : est une collection de classes de transitions. zéro ou plusieurs transitions appartenant à la classe : « *normal-transitions* » ; «*crypto-transitions* » qui sont utilisées pour spécifier la fonction de cryptage, décryptage dans le protocole, elles sont représentées par une barre épaisse ; des transitions de la classe « *module-transitions* », représentées par un carré et qui sont utilisées pour réduire les parties complexes du CTPN ; des transitions de la classe « *sink-transitions* » qui n'ont pas de places de sortie pour exprimer la perte du jeton, et une ou plusieurs transitions appartenant à la classe « *timed-transitions* » annotant le délai temporel d'une fonction élémentaire d'envoi/réception ou de cryptographie,

I⊂P x T : correspondant aux arcs orientés reliant les places aux transitions, y compris les *arcs inhibiteurs* pour exprimer la logique négative dans les protocoles, les *key-arcs* qui jouent le rôle de la clé pour les fonctions de cryptographie.

O⊂ T x P : correspondant aux arcs orientés reliant les transitions aux places,

M⊂P x NI : est un ensemble de marquages avec NI l'ensemble des entiers non négatifs.

Notons que la ciphertext -place, la crypto- transition, la transition module et le key-arc ont été définis pour accroître la lisibilité de la spécification CTPN du protocole.

La méthodologie de spécification est en cinq étapes :

Etape1. Identification des fonctions : énumérer toutes les fonctions et les objets indépendants du protocole concerné par l'analyse. L'ensemble des fonctions élémentaires qui spécifie les transitions dans le CPTN consiste en des fonctions de cryptage, décryptage, envoi, réception, jointure des messages, fragmentation du message, perte de jeton. Les objets sont, les noms de principaux, les clés, les adresses des principaux. Les données ou messages sont spécifiés par les jetons.

Etape2. Identification des relations partielles : identifier les relations partielles (séquentielle, parallèle, conflit, join, fork…) entre les fonctions élémentaires (Figure 7).

Exemple de fonction séquentielle $t_i \rightarrow t_j$ i.e. : t_i est exécuté avant t_j.

Exemple de fonction Parallèle $t_i \parallel t_j$ i.e. : not $(t_i \rightarrow t_j) \wedge$ not $(t_j \rightarrow t_{i)}$

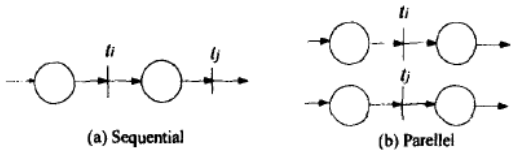


Figure 7 : Exemple de spécification des relations partielles.

Etape3. Raffinement progressif du CTPN : construire progressivement le graphe CTPN à partir des relations partielles entre les fonctions élémentaires spécifiées par les transitions et les caractéristiques de réseaux de Petri (un arc ne relie pas deux nœuds de même type par exemple). Si le CTPN est très complexe, on remplace un ensemble de transitions en corrélation par une transition de la classe « *Module-transition* », pour réduire la complexité du CTPN.

Etape4.Spécification du protocole d'attaque : le CTPN construit représente la fonction normale du protocole comme le secret ou l'authentification. Le protocole de l'attaquant CPTN' est construit sur la base du CPTN en lui incorporant des suppositions sur les attaques possibles.

Etape5.Transformation du CTPN en CTPN-language : transformer le CTPN et le CTPN' en langage CTPN au format textuel afin de lancer l'analyseur CTPN.

La sécurité

L'attaquant en général, connait le protocole conventionnel, les clés publiques et les adresses des principaux. Cependant, il ne connait pas les clés secrètes et les algorithmes cryptographiques des crypto systèmes. En utilisant les informations qu'il connait, l'attaquant peut accéder illégalement à l'information secrète en modifiant, ou escroquant, mettant sous écoute, rejouant, insérant supprimant, différant, ou changeant l'ordre des messages dans les canaux de communication. La procédure d'attaque du crypto-système est le protocole d'attaque qui est spécifié par le CTPN', il est utilisé pour analyser les attaques sur le protocole normal. Dans le modèle d'attaque si le jeton atteint un état non sécurisé alors le protocole présente un point faible, le chemin de l'état initial vers l'état non sécurisé représente le scenario d'attaque.

L'analyse

L'analyseur du CTPN qui est un outil d'aide à l'analyse, implémenté sous DOS en langage C, est composé d'un module d'analyse syntaxique (parseur) qui génère la matrice de structure du protocole, du module statique, du module dynamique et du module de performance. Grâce à ces modules, cette approche réalise différents types d'analyse.

Les propriétés structurelles : ce sont les propriétés statiques du protocole avant son exécution, par exemple : la vivacité structurelle, T-invariant, P-invariant, cohérence, etc.…elles sont indépendantes du marquage initial du CTPN.

L'analyse dynamique : les propriétés dynamiques (scenario d'exécutions, accessibilité, vivacité, blocage,..) du protocole sont analysées en utilisant le graphe d'accessibilité temporel du CTPN. Ce graphe fournit l'information sur les états accessibles, les scenarios d'exécution et les délais temporels des transitions, la conservation de données et de messages, le degré de parallélisme des fonctions.

L'analyse de performance : en utilisant CTPN ou CTPN', on peut avoir le délai total de l'exécution d'un protocole ou d'une attaque grâce aux transitions temporisées et au graphe d'accessibilité temporel obtenu en exécutant le CTPN.

L'analyse de l'attaque : grâce au CTPN', le protocole d'attaque, les points faibles du protocole ciblé par l'attaque sont détectés; et grâce au graphe d'accessibilité du CTPN' on obtient le scenario d'attaque, l'accessibilité aux états non sécurisés et le délai total de l'attaque.

Exemple de mise en œuvre **: Le Protocole TMN**

C'est un protocole de distribution de clé, par lequel deux stations au sol, dans un système de communication mobile, obtiennent une clé commune de session par l'intermédiaire d'un serveur de confiance. Les messages vers le serveur sont cryptés par la clé publique du serveur, connue par

toutes les stations au sol. Les stations au sol génèrent des clés, qu'elles utilisent comme clés de session (Figure 8). L'exécution avec succès du protocole (le CTPN sans la partie attaquant) correspond au marquage de p27 et p28. (Figure 9 sans la partie attaquant)

Spécification de l'attaque : on suppose que l'attaquant connait des informations qui sont exprimées dans le marquage initial du CTPN', il connait la clé publique du serveur et la procédure locale de B, et peut modifier, insérer et supprimer les messages dans les canaux de communication (p16, p25, p20). L'attaquant intervient à la place de B (Figure 9), la place p16 contient le message envoyé par A au serveur, l'attaquant le récupère (au lieu de B), crypte son propre message et l'envoie au serveur, le serveur crypte le message de A avec celui de l'attaquant et l'envoi à A, A récupère le message de l'attaquant (A croit qu'il vient de B) et l'utilise comme clé de session pour crypter les données. Le succès de l'attaque correspond au marquage des places p34 et p35.

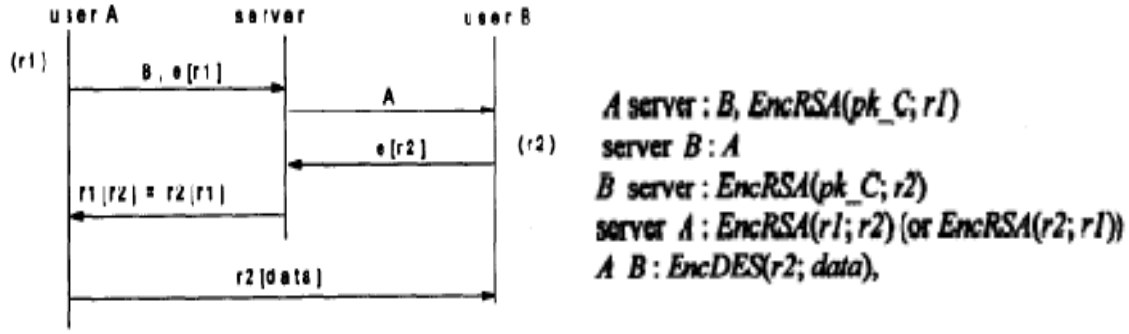


Figure 8 : Notation conventionnelle de protocole de TMN**.**

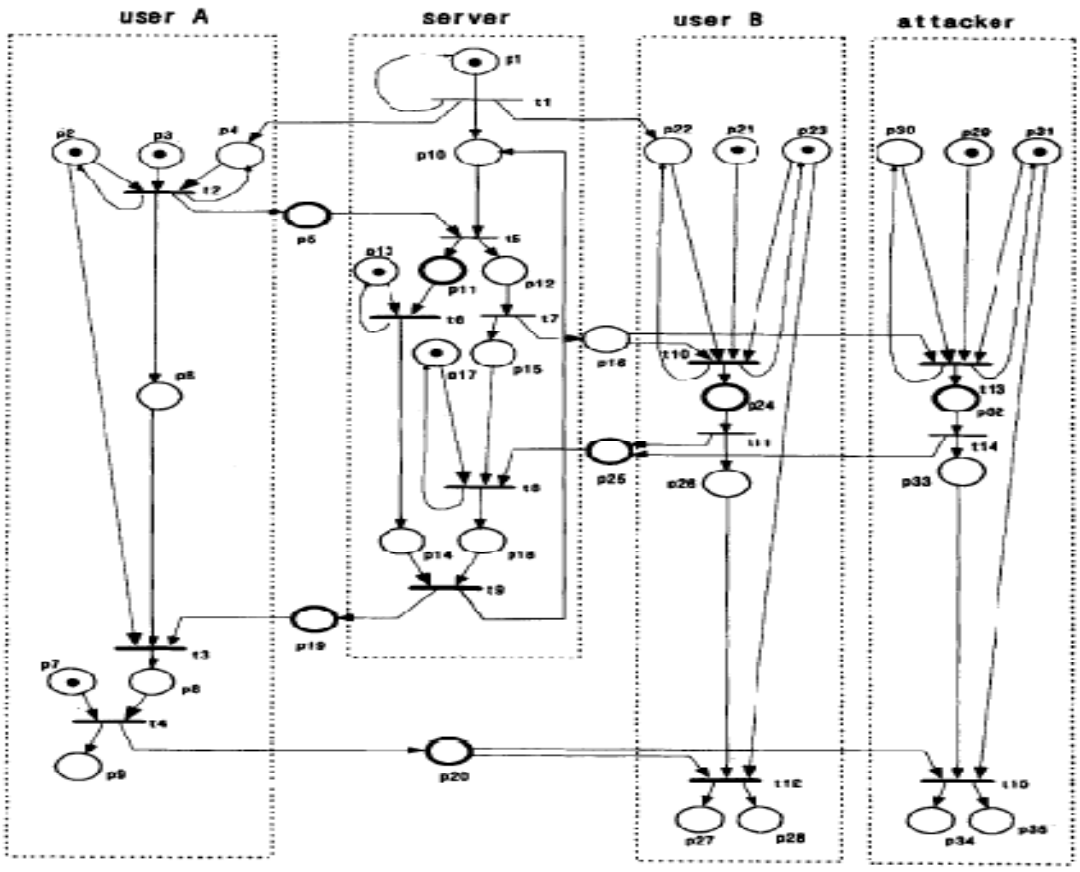


Figure 9 : Spécifications du CTPN' du protocole de TMN avec l'attaquant déguisant la partie ***B.***

La matrice de marquage est générée automatiquement par le module dynamique de l'analyseur et le graphe de marquage est dessiné sur la base de cette matrice, les scénarios d'attaque sont visibles sur le graphe de marquage.

Avantages et inconvénients du modèle

Ce modèle présente plusieurs avantages, l'avantage major est la puissance d'expression, en effet toutes les relations partielles entre les fonctions du protocole peuvent être représentées dans ce modèle. La lisibilité fournie par les classes de transitions et de places est le deuxième point fort du modèle, ainsi que l'existence de l'outil d'analyse développé dans le cadre du modèle. Néanmoins il n'est pas dépourvu d'inconvénients. Parmi les inconvénients de ce modèle, on peut citer :

-La traduction du protocole de sa forme graphique vers le langage textuel se fait manuellement.

-Le problème commun à tous les modèles qui se basent sur les réseaux de Petri, y compris CTPN, est le problème de l'explosion combinatoire du nombre de nœuds importants dans le graphe d'accessibilité.

II.4.4 Autres modèles

Dans (Al-Azzoni, et al., 2005) les auteurs ont proposé une modélisation des protocoles cryptographiques par des réseaux de Petri colorés en utilisant l'outil *Design/CPN* pour l'analyse. Initialement le protocole est modélisé sans l'intrus, puis un modèle d'intrus générique est introduit dans la spécification, la technique d'analyse du réseau de Petri coloré résultant, est l'exploitation du graphe d'occurrence, les auteurs ont présenté plusieurs mécanismes pour réduire la taille du graphe d'occurrences. Un graphe d'occurrences plus petit est facilement exploitable pour déduire si des buts particuliers de sécurité sont atteints.

Dans ce modèle hiérarchique les entités du protocole sont représentées par des transitions de substitutions (elles sont développées par la suite par des sous réseaux de Petri colorés, pour décrire le comportement de l'entité), les jetons représentent les messages qui circulent entre les entités, les couleurs concernent les différentes parties du message. Le protocole présente une faille si un état non sécurisé est atteint. Leur modèle est caractérisé par l'utilisation d'une place centrale qu'ils ont appelé BD-place, qui contient tout les jetons interceptés par l'intrus, et qui modélise les connaissances cumulées par l'intrus, ce qui facilite la construction du modèle d'intrus en se basant sur la couleur du message intercepté, chaque sous-processus d'intrus correspond à la partie du message intercepté. Pour avoir un graphe d'occurrences réduit, au plus un sous processus d'intrus est déclenché.

III. Synthèse sur les modèles formels des protocoles cryptographiques

La preuve des protocoles cryptographiques est un problème très difficile. A l'heure actuelle, les solutions proposées ne permettent pas de dire avec certitude qu'un protocole donné est sécurisé ou non. Les recherches ont aujourd'hui tendance à se diriger vers des preuves en utilisant des méthodes orientées logiques ou les algèbres de processus. En parallèle, des recherches ont lieu afin de trouver de nouvelles failles au sein des protocoles existants. Malgré l'inexistence de protocoles prouvés sécurisé, certains protocoles sont aujourd'hui considérés comme suffisamment sûrs pour être utilisés puisqu'ils résistent à l'ensemble des failles aujourd'hui connues.

Nous récapitulons dans le tableau ci-dessous les caractéristiques des différents modèles des protocoles de sécurité que nous avons présentés.

Caractéristiques	Modèle algébrique	Modèle algèbre de processus	Modèle logique	Modèle graphique (Réseau de Petri)
Point de vue considéré	-L'intrus.	- Chaque rôle du protocole est représenté par un processus.	-Les connaissances de chaque acteur du protocole. -Prise en compte du message transmis.	-Les acteurs du protocole. -L'intrus.
Langage de spécification	-Algébrique : les ensembles. -Axiomes et règles de réécriture.	- Processus avec Primitive cryptographique ou opérateur de déduction.	-Logique modale. -Axiomes et règles d'inférences.	-Graphique. -Places, transitions et jetons.
Objectif	-Trouver un axiome qui n'est pas vérifié.	Modélisation de la propriété du secret et prouver qu'elle est assurée par le protocole.	Prouver les théorèmes qui modélisent les propriétés de sécurité du protocole.	-Atteindre un état non sécurisé. -Générer les scénarios d'attaques s'il y a lieu.
Inconvénient	-Chaque modèle spécifie une seule propriété soit le secret soit l'authentification -L'analyse ne permet pas de détecter toutes les failles. -Il y a des modèles qui ne s'appliquent qu'à des protocoles bien spécifiques.	Spi calcul : l'équivalence observationnelle de deux processus est difficile à prouver. Il n'existe actuellement aucun outil automatique de preuve pour les algèbres de processus.	-Aucune logique ne peut détecter toutes les failles. -Les modèles logiques sont difficiles à automatiser car elles utilisent des preuves non triviales.	- Le Problème d'explosion du nombre d'états du graphe des marquages. - Les attaques non incluse dans le modèle ne sont pas détectées.

Tableau 5 : Synthèse sur les modèles formels des protocoles cryptographiques.

critères → ***Modèles*** ↓	Puissance d'expression	Représentation graphique	Utilisation d'outil automatisé	Capacité de preuve
Modèles Algébriques	-	-	+	+
Modèles logiques	-	-	-	+
Modèles d'algèbres de processus	+	-	-	+
Modèles de réseaux de Petri	+	+	+	+

Tableau 5.1 : Comparaison entre les différents formalismes.

La conclusion qu'on peut tirer de ces tableaux est que chaque formalisme a des points forts et des limites, les formalismes algébriques et logiques, se caractérisent par leur capacité de preuve, mais

on leur reproche la faiblesse d'expression. En effet, les modèles algébriques et logiques ne permettent pas la représentation de caractéristiques importantes des protocoles de sécurité telles que le parallélisme, l'asynchronisme et le non déterminisme. Le manque de lisibilité dû au fait que la spécification est textuelle, l'absence d'outil d'analyse pour les modèles qui se basent sur l'algèbre de processus et sur la logique, ce qui rend la preuve difficile à effectuer. On constate aussi la complexité de spécification pour les modèles qui se basent sur l'algèbre de processus. Par contre, les modèles qui se basent sur les réseaux de Petri pallient à la plupart de ces inconvénients, pour la capacité de preuve, ce modèle s'oriente vers la recherche d'un état non sécurisé pour conclure que le protocole présente une faille, l'absence de cet état signifie que le protocole est sécurisé contre cette faille .

Conclusion

Dans ce chapitre nous avons fait un tour d'horizon sur les différentes modélisations des protocoles cryptographiques, nous avons commencé par des définitions nécessaires à la compréhension, ensuite nous avons présenté les différents modèles en introduisant le formalisme utilisé, nous avons conclu le chapitre par une synthèse et une comparaison entre les différents modèles, qui est en faveur des modèles qui adoptent le formalisme réseaux de Petri.

Les Modèles Formels pour les contrôles d’accès

3.Les Modèles Formels pour les contrôles d'accès

Introduction

Une politique de sécurité est exprimée en langage naturel, mais son application nécessite une formalisation au moyen d'un modèle de sécurité pour lever les ambiguïtés sur la spécification et implémenter les mécanismes de sécurité nécessaires.

L'un des mécanismes les plus utilisés est le contrôle d'accès. Le contrôle d'accès définit l'ensemble des opérations qu'une entité du système peut réaliser ou non sur un ensemble de ressources.

Le problème de formalisation des modèles de contrôle d'accès n'est pas nouveau, il remonte aux années soixante-dix environ où une large partie de la recherche a été faite dans le contexte des systèmes militaires. Il en est résulté un nombre de modèles formels.

I. Modélisation des contrôles d'accès

Les politiques d'autorisation les plus citées dans la littérature sont généralement associées à un modèle de sécurité. Un modèle de sécurité peut être défini comme un formalisme souvent mathématique permettant de représenter, de façon claire et non-ambiguë, la politique de sécurité. Il aide à abstraire la politique de sécurité afin de réduire sa complexité, et à faciliter sa compréhension, comme il peut servir à vérifier que cette politique est complète et cohérente (pas de règles contradictoires), et que la mise en œuvre par le système de protection est conforme aux propriétés attendues du système.

La plupart des politiques de sécurité reposent sur les notions de *sujets*, d'*objets* et de *droits d'accès*. Un sujet est une entité active, correspondant à un processus qui s'exécute pour le compte d'un utilisateur. Dans ce contexte, un *utilisateur* est une personne physique connue du système informatique et enregistrée comme utilisateur ; ou un serveur, sorte de personne morale représentant des fonctions de service automatiques, tel que le serveur d'impression, le serveur de base de données, le serveur de messagerie, etc. Un objet est une entité considérée comme "passive" qui contient ou reçoit des informations. A un instant donné, un sujet à un droit d'accès sur un objet si et seulement si le processus correspondant au sujet est autorisé à exécuter l'opération correspondant à ce type d'accès sur cet objet.

Les politiques du contrôle d'accès peuvent être groupées en trois principales classes, les politiques discrétionnaires (DAC), les politiques obligatoires (MAC) et les politiques basées sur les rôles (RBAC).

I.1 Politiques et modèles d'autorisation discrétionnaires (*DAC*)

Dans le cas d'une politique discrétionnaire (Abou El Kalam, 2003), les droits d'accès à chaque information sont manipulés librement par le responsable de l'information généralement le propriétaire, à sa *discrétion.* Les droits peuvent être accordés par ce responsable à chaque utilisateur, à des groupes d'utilisateurs, ou bien aux deux. Ceci peut parfois amener le système dans un état d'insécurité.

Une politique discrétionnaire n'est applicable que dans la mesure où il est possible de faire totalement confiance aux utilisateurs et aux sujets qui s'exécutent pour leur compte. Une telle politique est par là même vulnérable aux abus de pouvoir provoqués par maladresse ou par malveillance. Ainsi, s'il est possible à un utilisateur d'accéder à certains objets ou d'en modifier les droits d'accès, il est possible qu'un cheval de Troie s'exécutant pour le compte de cet utilisateur (à son insu) en fasse de même. De plus, si un utilisateur a le droit de lire une information, il a en général le droit de la transmettre à n'importe qui. Parmi les modèles associés aux DAC, on trouve le *Modèle de Lampson* **,** le *Modèle Harrison-Ruzzo-Ullman(HRU),* le *Modèle Take-Grant* ,le *Modèle TAM* ***etc.…***

I.2 Politiques et modèles d'autorisation obligatoires (*MAC*)

Les politiques discrétionnaires présentent plusieurs inconvénients, notamment les fuites d'informations et la vulnérabilité aux chevaux de Troie. Pour pallier à ce type de problèmes, les politiques obligatoires décrètent des règles incontournables destinées à forcer le respect des exigences de sécurité. Dans ces politiques mandataires (MAC), le contrôle d'accès est effectué sous l'égide du système plutôt que des utilisateurs (Rakkay, 2009). Le sujet n'a accès à une information que si le système l'y autorise. Les règles d'une politique mandataire diffèrent selon qu'il s'agisse de maintenir des propriétés de confidentialité ou d'intégrité. Les politiques mandataires les plus souvent utilisées sont des politiques multi-niveaux, qui reposent sur la notion de classes de sécurité, ainsi les politiques multi-niveaux affectent aux objets et aux sujets des niveaux non-modifiables par les usagers, et donc qui limitent leur pouvoir de gérer les accès aux données qu'ils possèdent. Le modèle de Bell-LaPadula (Méludovi, 2007), qui a été utilisé pendant longtemps dans les systèmes militaires, est l'un des plus influents pour la confidentialité, le modèle de la Muraille de Chine de Brewer et Nash (Brewer, et al., 1989) est un autre exemple de modèle de confidentialité. Pour l'intégrité, nous citons le modèle de Biba et le modèle de Clark et Wilson.

I.2.1 Les politiques multi-niveaux

Modèle de Bell-LaPadula (**BLP**)

Le modèle **de Bell-LaPadula** (Méludovi, 2007) a été développé par David Elliott Bell et Leonard J. LaPadula en 1973 pour formaliser la politique de sécurité multi-niveaux au département de la défense des États-Unis. Destiné au domaine militaire, la propriété à assurer est la confidentialité.

Le modèle se base sur la matrice de contrôle d'accès pour représenter les permissions d'accès du système, et définit en plus des sujets S et des objets O, un treillis de sécurité, noté (SC,<=) où SC est un ensemble de classes de sécurité et <= une relation de dominance. Chaque objet a ainsi un niveau de sécurité qui représente le danger que peut constituer la divulgation de l'information contenue dans cet objet, et chaque sujet a une habilitation qui désigne la confiance qui lui est accordée.

Le modèle peut être représenté par une machine à états où chaque état est défini (Abou El Kalam, 2003) par un triplet (*S,O,M*), où :

- *S* : un ensemble de sujets ;
- *O* : un ensemble d'objets ;
- *A* : un ensemble d'opérations d'accès ; *A* = {*exécuter*, *lire*, *ajouter*, *écrire*} ;
- un ensemble de niveaux de sécurité muni d'une relation d'ordre partiel ;
- *B* : l'ensemble de tous les états possibles ;
- *M* : l'ensemble des matrices (M_{so})$s \in S$; $o \in O$,
- $F \subset Ls$ x *Lc* x *Lo* : l'ensemble des niveaux de sécurité ; dans un état donné, *fs* est le plus haut niveau de sécurité qu'un sujet peut avoir ; *fc* est le niveau courant de chaque sujet ; *fo* est la classification de l'objet.

Les deux règles de contrôle d'accès sont :

- *La propriété simple* : un état est *sûr* si et seulement si, pour chaque sujet *s* qui observe un objet *o*, le niveau de sécurité maximal du sujet domine le niveau de sécurité de l'objet.

Formellement, un état satisfait la propriété simple si :

$$\forall (s, o, a) \in \bar{b} \quad \bar{a} \in \{lire, \bar{e}crire\} \Rightarrow f_o(o) \leq f_s(s).$$

- *La propriété étoile* : un état est *sûr* si et seulement si, pour chaque sujet *s* qui modifie un objet *o*, le niveau de sécurité de *o* domine le niveau courant de sécurité de *s*. Formellement, un état est sûr si :

$$\forall (s, o, a) \in \mathrm{b}, \quad \mathrm{a} \in \{ajouter, écrire\} \Rightarrow f_c(s) \leq f_o(o)$$

D'une manière plus précise :

- si l’opération est un ajout, elle ne sera possible que si le niveau de sécurité de l’objet domine le niveau courant de sécurité du sujet ;
- si l’opération consiste à créer un objet et écrire dedans, le niveau de sécurité de l’objet est égal au niveau courant de sécurité du sujet ;
- si l’opération est une lecture, le niveau de sécurité de l’objet est dominé par le niveau courant du sujet.

Mais la politique de Bell-Lapadula présente plusieurs inconvénients. Le plus important est la dégradation du service provoquée par la surclassification des informations (Rakkay, 2009). En effet, au cours de sa vie, le niveau d’une information ne peut que croître : si une information non classifiée est utilisée par un sujet habilité au secret, tout objet modifié par ce sujet avec cette information sera classifié secret. Petit à petit, les niveaux de classification des informations croissent de façon automatique, et il faut les « déclassifier » manuellement.

Politique d’intégrité de Biba

Le modèle de Bell-LaPadula ne répond qu’à des besoins de confidentialité. C’est pour cette raison que d’autres politiques obligatoires ont été développées pour le maintien de l’intégrité. C’est le cas pour le modèle d'intégrité de Biba, développé par Kenneth J. Biba en 1977. Le modèle est conçu de manière à ce que les intervenants ne soient pas en mesure de corrompre des données placées dans un niveau qui leur est supérieur, ou être corrompus par des données d'un niveau inférieur à celui de l'intervenant.

En général, garantir l'intégrité des données vise à:

- empêcher des modifications par des tiers non-autorisés.
- empêcher des modifications sur des données non-autorisées par des tiers autorisés.

Tout comme le modèle de Bell-LaPadula, le modèle de Biba (Méludovi, 2007) définit une propriété de sécurité simple et une propriété étoile, mais leur signification est opposée entre les deux modèles. Dans le cas de Biba, ces propriétés sont :

- propriété de sécurité simple : un intervenant à un niveau donné d'intégrité ne peut pas lire un objet situé à un niveau d'intégrité inférieur.
- propriété étoile : un intervenant à un niveau donné d'intégrité ne peut pas écrire un objet à un niveau d'intégrité supérieur.

Le modèle Biba présente l’inconvénient dual du modèle Bell et Lapadula, c’est celui de la sous-classification ou la dégradation des niveaux d’intégrité des sujets et des objets. Un sujet peut être

empêché d'exécuter une procédure à cause des opérations de lecture qu'il a effectué précédemment et qui ont abouti à une dégradation de son niveau d'intégrité.

I.3 Politiques et modèles de sécurité par rôles (*RBAC*)

Les politiques obligatoires, en particulier les politiques multi-niveaux, imposent des contraintes fortes sur les organisations. Les relations d'ordre partiel qu'elles utilisent sont mal adaptées à la réalité des entreprises: peut-on dire qu'une information du service commercial est plus secrète ou plus critique que celle qui provient du bureau d'étude ? Les politiques discrétionnaires sont elles aussi mal adaptées et permettent difficilement de garantir des propriétés de sécurité. D'autre part, il faut constamment redéfinir les règles, chaque fois qu'un nouvel utilisateur ou un nouvel objet est introduit dans le système. Les politiques discrétionnaires sont donc très difficiles à administrer.

Les politiques basées sur les rôles RBAC (R. Sandhu, 1996), pour *Role-Based Access Control* forment la plus grande famille des modèles de contrôle d'accès et sont les plus étudiées du domaine. Elles visent à faciliter l'administration de la sécurité. Un rôle représente de façon abstraite une fonction identifiée dans l'organisation par exemple, chef de service, ingénieur d'étude, etc. A chaque rôle, on associe des permissions ou privilèges, ensemble de droits correspondant aux tâches qui peuvent être réalisées par chaque rôle. Les permissions ne sont plus associées d'une façon directe aux sujets, mais à travers des rôles. Les deux relations de la figure 10 « *Détient* (*Rôle, Permission*) » et« *Joue* (*Sujet*, *Rôle*) » définissent précisément les permissions accordées à chaque sujet. Un rôle peut avoir plusieurs permissions et une permission peut être associée à plusieurs rôles. De même qu'un sujet peut être membre de plusieurs rôles et inversement, un rôle peut être exécuté par plusieurs sujets. Les sujets ayant reçu l'autorisation de jouer un rôle, héritent alors des permissions associées à ce rôle.

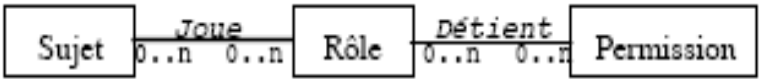


Figure 10: Attribution des permissions aux sujets à travers des rôles.

Chaque organisation peut avoir sa propre politique interne basée sur les rôles des divers sujets qui lui appartiennent. Initialement, les règles de la politique sont définies en fonction de tous les rôles existant au sein de l'organisation. Par la suite, lorsqu'un nouvel employé est recruté, il suffira de lui assigner les rôles qu'il est censé avoir au sein de l'organisation pour que le système puisse automatiquement cantonner ce nouveau sujet dans les limites autorisées par la politique.

Des variantes de RBAC ont été proposées telles que les modèles TBAC (Task Based Authorization Controls), TMAC (Team-based Access Control), ORBAC (Organization Role Based Access Control), TRBAC (Temporal Role Based Access Control), GTRBAC (Generalized Temporal Role Based Access Control). Ces modèles raffinent le modèle RBAC en introduisant respectivement les notions de tâches, d'équipes, d'organisation et de temps.

La vérification d'une politique de sécurité se fait au moyen de techniques et outils appropriés permettant de vérifier que ses exigences seront effectivement satisfaites après une implémentation dans le système cible. Plusieurs modèles et langages ont été proposés pour formaliser les structures, les principes et les propriétés des modèles de contrôle d'accès.

I.3.1 Modélisation en algèbre de processus des contrôles d'accès RBAC

Des approches se sont orientées vers l'intégration des algèbres de processus dans l'expression des modèles de contrôle d'accès à base de rôles. L'intérêt de ces approches est motivé par la capture des comportements dynamiques des modèles RBAC. Ces approches sont recensées dans la thèse de doctorat de (Rakkay, 2009). Nous allons nous contenter de donner un aperçu sur l'approche de (Braghin, et al., 2004), les auteurs ont proposé une extension du π-calcul pour décrire les modèles RBAC.

Extension de π-calcul

L'extension consiste à intégrer la notion d'utilisateurs représentés par des processus, deux réductions pour exprimer l'activation et la désactivation des rôles et de nouvelles façons pour exprimer l'assignation des permissions aux rôles. Le terme $r|P|_\rho$ représente une session de l'utilisateur appelé ***r***, s'exécutant via un processus *P* et ayant l'ensemble de rôles actifs ρ. Les nouvelles réductions sont exprimées comme suit:

$$r|roleR.P|_\rho\, I \longmapsto r|P|_{\rho \cup R}$$

$$r|yield\ R.P|_\rho \longmapsto r|P|_{\rho - R}$$

Quand un utilisateur *r* active un rôle *R* pendant une session, *R* doit être ajouté à l'ensemble de rôles ρ activé, et le restant de la session *P* sera exécuté avec l'ensemble ρ mis à jour. Vice-versa pour la désactivation du *R*.

Soit l'exemple du système suivant :

Client | **role** auth_ client.*port_ 80<index.html>.P* |ρ || *server* | *port_ 80*(*x*).*Q*|ρ'

Ce système modélise l'interaction entre le client et le serveur HTTP. Le système contient deux utilisateurs *client et server*, s'exécutant en parallèle (||). Il peut évoluer comme suit : d'abord, l'utilisateur *client* active le rôle auth_client par l'exécution de l'action **role** (**role** auth_ client), qui dans la pratique impliquerait de s'authentifier au moyen d'un certificat de sécurité. Puis, elle envoie la demande au serveur HTTP au port habituel 80(*port_ 80<index.html>*), c.-à-d., exécute une action d'output sur le *port 80.*

Dans l'exemple de client/serveur ci-dessus, le client non authentifié (c.-à-d. l'interaction avec le serveur sans avoir exécuté précédemment le **rôle** auth_client) va être rejeté, si la politiques RBAC permet seulement aux utilisateurs autorisés d'exécuter des requêtes HTTP.

La syntaxe de l'extension du pi-calcul que les auteurs ont défini est la suivante :

$$\text{PROCESSES:}\quad P,Q ::= \mathbf{nil} \mid P \mid Q \mid !P \mid [u = v]P \mid (\nu a : R)P \mid a(x).P \mid u\langle v\rangle.P \mid \mathbf{role}\, R.P \mid \mathbf{yield}\, R.P$$

$$\text{SYSTEMS:}\quad A,B ::= \mathbf{0} \mid r[\![P]\!]_\rho \mid A \parallel B \mid (\nu a^r : R)A$$

Les contraintes de sécurité sont exprimées sous forme de règles sur le système de transitions, et la vérification est réalisée au moyen de la bisimulation et la congruence barbue (Braghin, et al., 2004).

Un système de transitions est une structure définie par un triplet (Q, q_0,→) où Q est un ensemble d'états, $q_0 \in$Q est l'´etat initial, et→ est une relation de transitions entre les états de Q. La notation q →q' est utilisée pour spécifier que (q, q')∈→.

La bisimulation est une relation binaire entre systèmes de transition d'états, associant les systèmes qui se comportent de la même façon au sens qu'un des systèmes simule l'autre et vice-versa. Intuitivement deux systèmes sont bisimilaires s'ils sont capables de s'imiter l'un l'autre. Dans cette optique, chaque système ne peut être distingué de l'autre par un observateur. La congruence barbue est une équivalence observationnelle typée.

I.3.2 Modélisation logique des contrôles d'accès

Le modèle ORBAC (Organisation Based Access Control) (SERES, 2008) étend le modèle RBAC. Il est centré sur le concept d'organisation, il permet également de prendre en compte des informations de contexte dans l'expression des règles, afin de spécifier un contrôle d'accès fin et adapté. ORBAC attribut des permissions, des obligations, des recommandations et des interdictions sur un groupe d'objets, à un rôle, pour réaliser des activités dans une organisation, dans un contexte

donné. Une permission correspond à un fait ayant la forme ***permission (org,r,a,v,c)*** qui signifie que l'organisation ***org,*** dans le contexte ***c,*** permet au rôle ***r*** de réaliser l'activité ***a*** sur la vue ***v*** **.**

Logique déontique

EL Kalam (Abou El Kalam, 2003) à défini un langage basé sur la logique déontique pour représenter la politique de sécurité ORBAC, ce qui permet d'exprimer des permissions, des interdictions et des obligations, mais aussi des recommandations, concept fort utile dans le domaine de la santé. La motivation était de sécuriser les systèmes d'information et de communication dans le domaine de la santé.

La logique déontique comporte une unique modalité qui dénote l'obligation, à partir de cette modalité on définit trois modalités, le permis, l'interdit et l'optionnel.

La logique modale est appelée *logique déontique* lorsque les formules " A" et "◊A", sont lues, « il est obligatoire que A » et « il est permis que A » respectivement. La formule " ¬A" exprime l'interdiction de "A". Plus précisément, si Φ est un ensemble de propositions atomiques *a*, *b*, *c*, …, ¬,∧,∨,⇒, désignent les connecteurs booléens habituels, et si **O**, **P** et **F** désignent les trois opérateurs modaux de la logique déontique (obligation, permission, interdiction), le langage de la logique déontique noté $L_O(\Phi)$ est l'ensemble des formules (ou expressions) construit par les règles suivantes :

- si $p \in \Phi$, p est une formule,
- si *p* et *q* sont des formules, $\neg p$, $p \vee q$, $p \wedge q$, $p \Rightarrow q$ sont des formules,
- si p est une formule, **O***p*, **P***p* et **F***p* sont des formules.

La sémantique utilisée est définie par le modèle $M = < W, R, V, D >$ où :

- ***W*** est un ensemble de mondes possibles *w* ;
- **R** est une relation d'accessibilité (relation binaire sur *W*) ;
- ***D*** est un domaine. Un domaine est un ensemble non-vide de valeurs ;
- ***V*** est une fonction qui donne les valeurs de vérité des éléments du langage :

V(*Formule Atomique d'arité n*)$\in D^n$, V(*variable*) $\in D$, V(*constante*) $\in D$.

Intuitivement, « (*Bob, médecin, Hôpital-Rangueil*) $\in V(w, RdO)$ » signifie que dans le monde *w*, *Bob* joue le rôle médecin au sein de l'organisation "*Hôpital-Rangueil*". De la même manière, « (*Sam, f*5.*doc*) $\in V(w,LIRE)$ » signifie que dans le monde w, *Sam* exécute l'action *LIRE* sur le fichier *f*5.*doc*.

Organisation, *Sujet*, *Objet*, *Action*, *Rôle*, *Vue*, *Activité* et *Contexte* représentent les types des constantes du langage défini par el kalam.

Les objectifs de sécurité sont exprimés par l'utilisation d'opérateurs modaux, par exemple $F[RdO(u, Pharmacien, \_) \wedge CREER(u, ordonnance)]$ est un objectif de sécurité relié aux propriétés de confidentialité et d'intégrité. Il correspond du point de vue sécurité au souci d'interdire une certaine formule. Plus précisément, il indique que, dans aucun des mondes accessibles, on ne peut trouver un utilisateur qui joue le rôle pharmacien et qui crée une ordonnance.

Une règle de sécurité est une formule modale dont les différentes clauses ne sont pas toutes des formules modales (par exemple : $r \rightarrow \mathbf{P}q$). Elle reflète la manière dont l'état de sécurité est en relation avec les différentes permissions, obligations, interdictions et recommandations qui existent dans le système.

Dans le langage proposé, les règles de sécurité prennent essentiellement la forme suivante :

$\forall s \forall a \forall o \forall r \forall a \forall v \forall c$

$\mathbf{P}(org, r, a, v, c) \wedge$

$RdO(org, s, r) \wedge$

$VdO(org, o, v) \wedge$

$AdO(org, a, a) \wedge CdO(org, s, a, o, c) \rightarrow Est\_permis(s, a, o)$:

Si l'organisation *org*, dans le contexte *c*, accorde la permission au rôle *r* de réaliser l'activité *a* sur la vue *v*, si *org* habilite le sujet *s* dans le rôle *r*, si *org* utilise l'objet *o* dans la vue *v*, si *org* considère l'action a comme faisant partie de l'activité *a* et si au sein *org* le contexte *c* est vrai entre *s*, a et *o*, alors le sujet *s* a la permission de réaliser l'action a sur l'objet *o*.

Pour la vérification de la politique de sécurité (si partant d'un état sûr, qui satisfait les objectifs de sécurité, et en appliquant les règles de sécurité, on retrouve un état ou certains objectifs de sécurité sont violés, état non sûr), et pour la détection et résolution des conflits (un utilisateur a simultanément la permission et l'interdiction d'effectuer une action sur un objet), l'auteur propose l'utilisation d'autres formalismes tels que, la méthode des tableaux (M. Fitting, 1993) pour la vérification et la logique possibiliste pour la résolution des conflits.

Afin de faciliter la conception et la manipulation de politiques fondées sur ORBAC, l'auteur souligne qu'il serait souhaitable d'utiliser une représentation graphique afin de faciliter la conception et la manipulation de politiques fondées sur ORBAC, dont l'intérêt pratique peut s'avérer significatif si cette représentation est supportée par un outil d'édition.

Logique de description avec défauts et exceptions

Dans (Boustia, et al., 2012) les auteurs, ont proposé un modèle de sécurité ORBAC à l'aide d'une logique de description avec défauts (δ) et où le changement de contexte est exprimé explicitement par une exception (ε).

Les logiques de description forment une famille de langages de représentation de connaissances utilisées pour représenter la connaissance terminologique d'un domaine d'application d'une façon structurée et formelle. Ces logiques ont été conçues à partir des réseaux sémantiques qui sont des graphes orientés étiquetés auxquels on associe des concepts aux nœuds et des relations aux arcs, et de la sémantique des frames (cadres) où l'on a des concepts représentés par des cadres qui sont caractérisés par un certain nombre d'attributs qui contiennent de l'information sur leur contenu.

Les auteurs ont développé un prototype pour administrer (en respectant les objectifs de la sécurité) la politique de sécurité, qu'ils ont baptisé DL_Orbacδε. Ce prototype est capable d'assigner des autorisations à des utilisateurs au sein d'une organisation donnée dans un contexte donné, conformément aux règles de sécurité introduites. Leur modèle permet de considérer le contexte dynamiquement, et d'inférer une autorisation. L'idée consiste en la conception et la réalisation d'une base de connaissances à l'aide d'une logique de description, qui inclut les concepts atomiques du RBAC à savoir : organisation, sujet, objet, rôle, vue, action et activité et de permettre le raisonnement sur ces connaissances par des mécanismes d'inférence qui traitent les variations de contexte entre autres le cas de défaut et le cas d'exception. La base de connaissance est constituée d'une TBox, qui comprend la définition des concepts et des rôles, et de plusieurs ABox , l'ABox décrit les individus en les nommant et en spécifiant en termes de concepts et de rôles, les assertions qui portent sur ces individus nommés. La TBox dans ce modèle inclut les axiomes d'attribution de rôle, de définition de vue, axiomes de définition d'activité et de hiérarchie, les axiomes de permission concrète, et les axiomes de définition de règles de sécurité, alors que l'ABox inclut des axiomes d'assertion d'organisation, de sujet, d'objet, de vue, de rôle, d'action, et d'activité. Les inférences dans les deux cas (défaut et exception) sont faites sur la base des TBox et ABox. Cependant le modèle ne prend pas en charge les recommandations et les obligations.

I.4 Modélisation des contrôles d'accès avec les Réseaux de Petri

Le but d'une approche basée sur les réseaux de Petri est de prouver que les propriétés ou les règles de la politique de sécurité sont bien renforcées dans le système en bénéficiant des différentes techniques et outils offerts par les réseaux de Petri.

I.4.1 Les Réseaux de Petri pour les modèles mandataires MAC

Les réseaux de Petri ont connu une large diffusion dans le domaine de la vérification de la sécurité, notamment dans le contexte des modèles mandataires.

Information Flow Secure net (IFS) avec RdP classique

Varadharajan dans (Varadharajan, 1990) a proposé une nouvelle extension des réseaux de Petri classiques, appelée IFS "Information Flow Secure net", qui introduit un treillis de sécurité de plusieurs niveaux. Ce modèle sert à spécifier, formellement, un système d'information. Les places représentent les canaux pour transférer des messages entre les processus atomiques (sujets), les messages sont représentés par des jetons, les transitions représentent les processus atomiques, un niveau de sécurité est associé a chaque message, un sous ensemble de sécurité est associé à chaque place. Lorsqu'une transition est franchie, elle émet un jeton dont la donnée et le niveau de sécurité sont produits en fonction des jetons reçus en entrée. Dans le modèle de Varadharajan la notion de sécurité repose sur deux concepts, un marquage sûr (les jetons ont des niveaux de sécurité inférieur ou égaux à ceux associés à la place où ils résident, i.e une transition ne peut lire ou envoyer un jeton confidentiel dans un canal publique. Ceci permet de préserver la confidentialité des données, dans la mesure où l'information est inaccessible à un utilisateur non autorisé à accéder le canal) et un flot d'informations sûr (le flot se produit via des transitions, une transition est dite sûre si son franchissement, pour tout marquage accessible à partir du marquage initial, respecte la politique de sécurité établie et mène vers un marquage sûr). La vérification de la sécurité repose sur une analyse de tous les marquages accessibles ainsi que les flots d'information afin de repérer toutes les exécutions qui sont jugées inacceptables. Toutefois, le modèle présente des limites (Bouzegza, 2007), car étant basé sur une classe de réseau de Petri de bas niveau, il est peu adaptable pour vérifier automatiquement des propriétés de sécurité par les outils standards ainsi que pour représenter des problèmes de grande taille, en plus ce modèle présente des ambiguités qui peuvent affecter son interprétation, ils ont été soulignées dans le travail de Juopperi.

IFS avec RdP Prédicats/Transition

Juopperi (Juopperi, 1995) a donc repris le modèle IFS en utilisant les réseaux Prédicats/Transition, son modèle augmente la puissance de description en considérant les transitions comme des règles d'une logique du premier ordre, ce qui permet de décrire des systèmes complexes de manière compacte. Mais de point de vue vérification, on note l'indisponibilité d'outil d'analyse.

IFS avec RdP coloré

Quelques années plus tard, Juszczyszyn (Juszczyszyn, 2003) proposa un autre modèle de sécurité appelé SCPN (Secure Colored Petri Net) issu de l'IFS, pour une représentation plus compacte et plus concise au moyen des réseaux de Petri Colorés, l'auteur propose d'utiliser les outils des CPN, tel que, l'outil *Design/CPN* pour la description du système et la génération automatique du graphe des marquages utilisé pour l'analyse du système .

Le modèle de Bell-LaPadula avec RdP coloré

Dans (Jiang, et al., 2004) les auteurs proposèrent l'équivalent du modèle de Bell-LaPadula en un réseau de Petri coloré. Les notions du modèle de Bella-LaPadula (entités actives, passives, le niveau de sécurité d'un objet, le niveau de sécurité courant et maximal d'un sujet, etc.) sont exprimés au travers d'un réseau de Petri coloré. L'analyse est basée sur l'exploration du graphe d'accessibilité pour vérifier les objectifs de sécurité suivants :

Le contrôle des accès aux objets, les problèmes de transfert caché des objets (par la règle de No-Write down) et les flots d'information implicites qui se produisent entre les objets.

Les auteurs définissent des relations d'accès temporelles qui sont évaluées sur le graphe des marquages pour vérifier si une propriété est vérifiée ou non. L'analyse (Bouzegza, 2007) nécessite la construction des graphes de marquages pour chaque sujet, par la suite les relations d'accès sont appliquées sur le graphe du sujet en vue de produire la trace des différents accès qu'il a effectués. Le but est de pouvoir situer dans le temps les accès aux objets et de détecter les possibilités de flots implicites qu'un sujet risque de produire à travers les objets. Donc pour avoir tous les états possibles du modèle, il sera nécessaire de représenter les graphes de marquages de tous les sujets du système, et par conséquence, l'analyse du flot d'information doit se faire sur tous ces graphes ensembles, ce qui n'est pas une tâche facile.

La politique de la muraille de Chine et le modèle d'intégrité de Biba ont été aussi modélisés en utilisant les réseaux de Petri Colorés (Zhang, et al., 2006) (Zhang, et al., 2006a).

Timed Secure Colored Petri Net (TSCPN)

Rakkay et Boucheneb (Rakkay, 2006), ont proposé un nouveau modèle de sécurité à base d'une variante temporisée des réseaux de Petri colorés, nommée TSCPN (*Timed Secure Colored Petri Net*) qui intègre à la fois le temps et les notions de sécurité. La notion de sécurité se base sur le principe des modèles mandataires, Rakkay et Boucheneb ont attribué des classes de sécurité aux informations et des niveaux d'autorisation aux utilisateurs. Cela est dans le but de contrôler à la fois l'accès aux informations et les flux qui peuvent se produire dans tout le système. L'intégration

du temps consiste à associer aux données des intervalles qui expriment leur disponibilité et validité, et ce dans le but de permettre des contrôles d'accès et de flux temporels et dynamiques. En effet, comme mesure de sécurité supplémentaire, les informations ne peuvent être utilisées qu'à l'intérieur de leurs intervalles de temps pour garantir leur validité. En dehors de cet intervalle, soit que l'information n'est pas encore disponible, soit qu'elle n'est plus valide. L'objectif est de s'assurer de la bonne conception du système, en accord avec les politiques de sécurité et par rapport aux propriétés énoncées. Néanmoins le problème de l'explosion des états construits pour l'étape de l'analyse constitue la limite principale du modèle.

I.4.2 Les réseaux de Petri pour les politiques à base de rôles RBAC

Les réseaux de Petri en tant que support de modélisation offrent aussi un support pour mieux gérer la spécification d'une politique de sécurité RBAC, sa mise à jour, en plus de la capture des propriétés dynamiques du modèle, notamment les flots d'autorisation. Suite à de nombreux travaux, il s'est avéré que les réseaux de Petri présentent un niveau d'abstraction suffisant pour permettre la gestion des autorisations et les propriétés s'y rattachant. Il est possible d'y exprimer l'ensemble des éléments constituant une politique de sécurité, à savoir : les sujets, les objets, les actions, les rôles, etc. Plusieurs travaux (Shafiq, et al., 2005), (Joshi, 2003), (Rakkay, 2009), proposent des approches de modélisation et de vérification pour les modèles RBAC à base des réseaux de Petri, principalement pour deux raisons :

- expliciter les flots d'information dans un modèle de contrôle d'accès,
- faciliter l'administration d'une politique RBAC dans les systèmes d'informations.

Nous n'aborderons que l'approche de (Rakkay, 2009) pour les deux autres nous orientons les intéressés vers la thèse de Rakkay.

RBAC avec les réseaux de Petri colorés hiérarchiques

En se basant sur ces arguments, l'auteur dans (Rakkay, 2009) a présenté une approche qui consiste en la modélisation du modèle RBAC avec les réseaux de Petri colorés hiérarchiques, en utilisant l'environnement logiciel CPNtools. Cet outil permet l'édition et la simulation des réseaux de Petri colorés ainsi que la génération et l'analyse de l'espace d'´etats.

La spécification se base sur des contraintes de cohérence (Une incohérence se produit si deux ou plusieurs règles d'un ensemble de règles de la politique se contredisent) ce qui permet de représenter diverses contraintes de RBAC notamment, les contraintes de cardinalité (par exemple le nombre d'usagers autorisés à un rôle ne doit pas excéder la cardinalité fixé à ce rôle), d'héritage (par exemple : si le rôle r1 hérite du rôle r2 avec r1 et r2 qui sont distincts, alors r2 ne peut pas

hériter r1) et de séparation des tâches (par exemple : si deux utilisateurs u1 et u2 sont en conflit d'activation pour un rôle r, alors celui-ci ne peut pas être activé simultanément par u1 et u2).

La structure du réseau CPN se compose de sept modules, au plus haut niveau, on retrouve le modèle RBAC, avec les six principales fonctions (activation désactivation ; assignation déassignation ; et enabled disabled du rôle) sont représentées par des transitions de substitution. Les transitions représentent les différentes actions (évènements). Les ensembles de sujets, de rôles et de sessions sont représentés par les couleurs cUser, cRole et cSession, les autres couleurs sont des combinaisons de ces couleurs atomiques. Les places capturent les états des différents éléments du modèle RBAC, à savoir les différents états des rôles, les différentes commandes à exécuter et les différentes cardinalités. Les jetons sont définis à partir de l'ensemble de domaine des couleurs, par exemple : <p, u> est le jeton sujet où p est la place où il réside et u $\in$ cUser. Les arcs, les expressions sur les arcs et les gardes (c'est un prédicat booléen qui valide le franchissement de la transition) modélisent les différentes contraintes de cardinalité, d'héritage et de séparation de tâches à satisfaire pour assurer la cohérence du modèle.

Fonctionnement du modèle

Le modèle passe d'un état à un autre par franchissement de transitions. A partir du moment où une transition est sensibilisée, elle peut être franchie à n'importe quel instant.

Une transition représente un évènement. Ainsi, lorsqu'une transition est sensibilisée, cela veut dire que les contraintes associées à cet évènement sont remplies. Les conditions de sensibilisation et de franchissement sont différentes selon les évènements. Les conditions de franchissement sont exprimées par des gardes propres à chaque transition, alors que les conditions de sensibilisation consistent en la présence de jetons dans les places d'entrées de la transition.

Vérification du modèle et de ses contraintes

Le modèle est défini par son marquage. Ainsi, la vérification de la cohérence du modèle RBAC est effectuée à partir d'une analyse de chaque état accessible du réseau de Petri coloré. L'état est dit cohérent s'il satisfait à toutes les contraintes de cardinalité, des séparations de tâches et d'héritage. Il s'agit ainsi d'effectuer une analyse d'accessibilité qui repose sur le graphe des marquages correspondant au réseau de Petri. En s'assurant de la cohérence du réseau de Petri, on s'assure également de la garantie des différentes propriétés de sécurité liées à la politique de sécurité. L'outil CPNtools offre un module de génération de graphes d'occurrences sur des modèles de RdP colorés. Un rapport est généré, il contient beaucoup d'informations utiles relatives au comportement du modèle CPN. Il permet de repérer les erreurs et constitue une source de référence nécessaire pour

exprimer correctement des formules CTL et vérifier les propriétés souhaitées avec le Model-checking.

Le Model-checking est une technique de vérification de propriétés temporelles par exploration des graphes d'accessibilité (systèmes de transitions). Il permet via un algorithme appelé model-checker, capable de dire à partir du modèle si oui ou non le système vérifie la propriété énoncée.

Pour les propriétés exprimables en CTL, le Model-checking se base sur le marquage des états du système de transitions. Un système de transitions satisfait une propriété *p* si et seulement si son état initial satisfait *p*. Un algorithme construit par induction l'ensemble de tous les états qui satisfont la propriété *p*. Chaque état est marqué par *p* s'il satisfait la propriété ou sa négation si non.

I.5 Synthèse sur les modèles formels pour les contrôles d'accès

La formalisation du contrôle d'accès est incontournable pour une gestion sûre et efficace des droits d'accès. Plusieurs modèles ont été proposés pour formaliser les principes et les propriétés des modèles de contrôle d'accès. Le tableau ci-dessous récapitule les caractéristiques des différents modèles que nous avons présentés.

	Modèles DAC	**Modèles MAC**
Modèle existants	Lampson,HRU, Take-Grant,TAM….	Braw et Nash Multi niveaux : Bell-la Padula, Biba,..
Inconvénients	-Fuites d'informations -Vulnérabilité aux Chevaux de Troie -Difficile à administrer	-Relations d'ordre partiel mal adaptées à la réalité. -Surclassification des informations et sous classification des niveaux d'intégrité des sujets et des objets. -Une seule propriété est considérée par modèle.

	Modèle algèbre de processus	**Modèle logique**	**Modèle graphique (Réseau de Petri)**
Modèle considéré	**RBAC** - Pi-calcul	**OR-BAC** : -Logique déontique -Logique des descriptions	**MAC** : - IFS avec RdP classique -IFS avec RdP predicats/transitions. -SCPN avec RdP coloré. - TSCPN avec RdP coloré temporel. **Muraille de chine, BIBA , Bell- Lapadula** :-RdP coloré. **RBAC :-**RdP coloré.
Inconvénients	Complexité de la spécification et de la preuve. Absence d'outils d'analyse.	-Absence de représentation graphique et d'outil d'édition	Explosion combinatoire des espaces d'états dans le graphe d'accessibilité.

Tableau 6 : Synthèse sur les modèles formels pour les contrôles d'accès.

Critères → / *Modèles* ↓	Puissance d'expression	Représentation graphique	Utilisation d'outil automatisé	Capacité de preuve
Modèles logiques	-	-	-	+
Modèles d'algèbres de processus	+	-	-	+
Modèles de réseaux de Petri	+	+	+	+

Tableau 6.1 : Comparaison entre les différents formalismes.

La conclusion qu'on peut tirer de ces tableaux est que chaque formalisme a des points forts et des points faibles, dans le cas général aucune formalisation n'est meilleure qu'une autre, cependant les formalismes algébriques et logiques, se caractérisent par leur capacité de preuve, mais on leur reproche, le manque de lisibilité dû au fait que la spécification est textuelle, l'absence d'outil d'analyse est un autre inconvénient pour ces modèles, surtout pour les modèles d'algèbre de processus car la preuve est difficile et elle est faite manuellement. Les modèles qui se basent sur les réseaux de Petri pallient à la plupart de ces inconvénients, en effet, les réseaux de Petri en tant que support de modélisation offre une puissance d'expression qui permet de représenter l'ensemble des éléments constituant une politique de sécurité, à savoir, les sujets, les objets, les actions, les rôles, etc, en plus de l'existence d'outil d'édition et d'analyse. Pour la vérification, on note la possibilité d'utilisation des modèles checking et la modélisation des propriétés de sécurité par une logique temporelle (CTL).

Conclusion

Nous avons consacré ce chapitre à la modélisation des contrôles d'accès et les différents formalismes utilisés, nous avons terminé le chapitre par une synthèse sur ces modèles et un comparatif sur les formalismes de modélisation. Nous avons conclu que le formalisme réseau de Petri présente des avantages qui le favorisent par rapport aux autres formalismes.

Les Modèles Formels pour les attaques

4.Les Modèles Formels pour les attaques

Introduction

Une « **attaque** » est l'exploitation d'une faille d'un système informatique à des fins non connues par l'exploitant des systèmes et généralement préjudiciables. Sur internet des attaques ont lieu en permanence, à raison de plusieurs attaques par minute sur chaque machine connectée. Ces attaques sont pour la plupart lancées automatiquement à partir de machines infectées par des virus, chevaux de Troie, vers, etc., à l'insu de leur propriétaire. Plus rarement, il s'agit de l'action de pirates informatiques. Afin de contrer ces attaques, il est indispensable de connaître les principaux types d'attaques afin de mettre en œuvre des dispositions préventives.

Les motivations des attaques peuvent être de différentes sortes :

- 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 ;
- s'informer sur l'organisation (entreprise de l'utilisateur, etc.) ;
- 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 (Stephenson, 2004).

Les attaques ont aussi été modélisées, pour décrire et analyser le processus par lequel un utilisateur malicieux tente d'exploiter ou arrêter des systèmes et/ou des réseaux informatiques.

I. Modélisation des Attaques

Aujourd'hui, les attaques sont complexes, et les scénarios d'attaque sont de plus en plus compliqués, l'attaquant peut utiliser plusieurs machines dans le réseau sur plusieurs étapes, pour atteindre son but, déviant ainsi les multiples mécanismes de sécurité. Les hôtes contrôlés par les attaquants peuvent devenir un point de lancement pour les intrusions.

Par conséquence, l'analyse des interactions entre les composants d'un réseau et la possibilité d'exploiter les vulnérabilités dans les hôtes est cruciale pour la sécurité des réseaux. La façon la plus directe et effective pour analyser les interactions entre les composants, est le paradigme de

modèle de l'exploitation des vulnérabilités. Ce paradigme de modèle peut explicitement dépister toutes les situations possibles des attaques impliquant plusieurs hots et réalisables sur plusieurs étapes. Les avantages de ce paradigme sont nombreux. Cela peut aider le gestionnaire du réseau à mieux comprendre les menaces présentes dans le réseau et prendre les mesures effectives contre ces menaces. C'est aussi utile pour les tests de pénétration qui sont utilisés pour simuler des attaques connues en vue d'évaluer la résistance du système contre ces attaques. Cela facilite aussi la construction des systèmes de détection des intrusions IDS.

Les approches de modélisation des attaques peuvent être divisées en approches combinatoires (les arbres de fautes, les arbres d'attaques), les approches stochastiques (les files d'attentes et les chaines de Markov) et les approches graphiques (les réseaux de Petri, graphe des attaques). Parmi ces approches, on trouve celles qui en plus de la modélisation du processus d'attaque, fournissent une ou plusieurs métriques pour l'évaluation quantitative de la sécurité du système modélisé.

I.1 L'arbre des fautes

Les arbres des fautes sont couramment utilisés dans le domaine de la sûreté de fonctionnement pour représenter comment les défaillances de composants élémentaires d'un système peuvent causer une défaillance globale du système.

Le but dans ce contexte est d'analyser la conception du système afin d'identifier les champs de risque. FTA (Fault Tree Analyse) est une technique déductive qui se focalise sur un évènement particulier indésirable. Par exemple un système associé à un réacteur ou arme nucléaire, ou les systèmes de vol des avions. Dans ce cas l'évènement indésirable peut être « la valve de sécurité ne s'ouvre pas », ou « l'avion s'écrase ». L'arbre des fautes illustre qualitativement comment l'évènement indésirable résulte à partir d'évènements primaires à travers, zéro ou plusieurs, évènements intermédiaires. Cette technique peut être appliquée pour la conception et l'analyse des systèmes « sécurité critique » par exemple dans le système de détection d'intrusions. Peu de travaux se sont focalisés sur l'application des techniques de sûreté pour la sécurité. Dans (Brook, et al., 2003), les auteurs proposent d'appliquer cette technique à la sécurité qu'ils illustrent avec un exemple inspiré d'un crypto système à clé publique.

Définition de l'arbre des fautes (Nai Fovino, et al., 2009)

L'arbre des fautes (FT) est un graphe orienté acyclique défini par le tuple <E, P, D, TOP>. P représente l'ensemble des portes logiques, E représente les évènements, P∪E représente les nœuds du graphe ; D est l'ensemble des arcs orientés qui relient un évènement à l'entrée d'une porte

logique, ou une sortie d'une porte logique à un évènement, tel qu'une connexion directe entre portes logiques ou entre évènements est interdite. TOP est l'évènement de l'FT qui n'est l'entrée d'aucune porte logique, il n'y a pas d'arc qui sort de l'évènement TOP, l'évènement de base d'un arbre de fautes n'est une sortie pour aucune porte logique, il n'y a pas d'arc qui entre vers cet évènement.

I.1.1 Les concepts de FTA (Brook, et al., 2003)

FTA se base sur les symboles d'évènements, les symboles de portes inspirés des circuits logiques et les symboles de transferts qui permettent de partager les arbres larges sur plusieurs pages. Les figures (Figures : 11, 12,13) montrent les différents symboles utilisés dans le FTA.

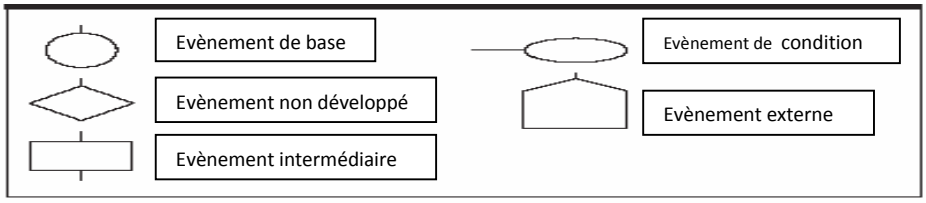


Figure 11 : Les symboles des évènements.

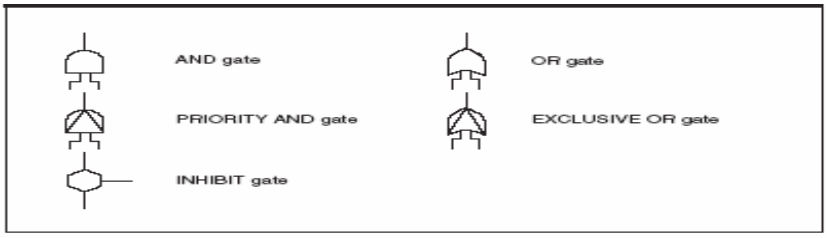


Figure 12 : Les symboles des portes

Figure 13 : Les symboles de transfert

I.1.2 La sécurité avec FTA

Le concepteur d'un système peut utiliser l'FTA pour analyser les besoins du système à un haut niveau d'abstraction dans la phase analyse du risque.

La méthode se résume par :

A-L'identification des évènements indésirables (racine ou TOP) pouvant se réaliser.

B- Pour chaque évènement, construire l'arbre de faute correspondant.

C-Analyse des arbres de faute.

A- L'identification d'évènements indésirables

En se basant sur le document des besoins du système, on tire l'évènement non désiré. Par exemple soit la condition "seuls les utilisateurs légitimes peuvent lire les documents du patient ", alors on crée l'évènement "les utilisateurs non autorisés lisent les documents du patient " : et c'est un évènement indésirable qui contredit les exigences du système.

B -La construction de l'arbre des fautes

A partir d'un évènement non désiré, construire l'arbre en respectant les règles de construction (ces règles sont prédéfinies et regroupées dans le guide des arbres de fautes), on s'arrête lorsque toutes les feuilles sont des évènements primaires.

C- L'analyse :

Les évènements primaires de l'arbre permettent à l'analyste de connaitre les causes de l'échec du système.

L'analyse de l'arbre des fautes fournit de l'information sur les interactions qui causent l'échec du système, dans ce sens FTA est un manuel sur les attaques, il identifie les chemins pour l'obtention de l'information qui est à l'encontre des objectifs de sécurité du système, et c'est à l'analyste de décider si ces chemins sont plausibles ou non réalisables.

Les évènements élémentaires peuvent être dotés de probabilités sur la réalisation de l'évènement, ainsi les probabilités des évènements intermédiaires jusqu'à la racine peuvent être calculées. Mais ces valeurs ne sont pas faciles à attribuer dans les systèmes « à sécurité critique », vue la nature discrète et non linéaire des systèmes.

I.1.3 Exemple d'illustration

L'exemple de la figure 14 concerne la signature des paquets logiciels : on suppose qu'il y a une méthode de signature de paquets logiciels, le paquet est constitué d'un fichier archive (ex. archive tar) et d'une signature numérique. La signature est calculée à partir du hachage de l'archive et de la clé secrète : cette signature peut être vérifiée par la clé publique. La sécurité du système est assurée par les deux faits :

- Il est informatiquement difficile de choisir l'entrée de l'algorithme de hachage qui produit une sortie particulière,
- Il est informatiquement difficile de dériver la clé secrète à partir de la clé publique.

On peut voir que chacun des trois évènements primaires (les feuilles) peut conduire à l'évènement de la racine, i.e. que l'attaquant peut contrefaire les paquets, ce qui conduit à compromettre le système. En

considérant les feuilles de la gauche vers la droite, la deuxième et la troisième sont des cas « informatiquement difficiles », ces deux cas sont étendus dans les Figure 15 et Figure 16 :

En considérant un algorithme de hachage puissant qui est difficile à attaquer, l'attaquant tente d'avoir la clé secrète directement à partir de l'ordinateur de l'utilisateur.

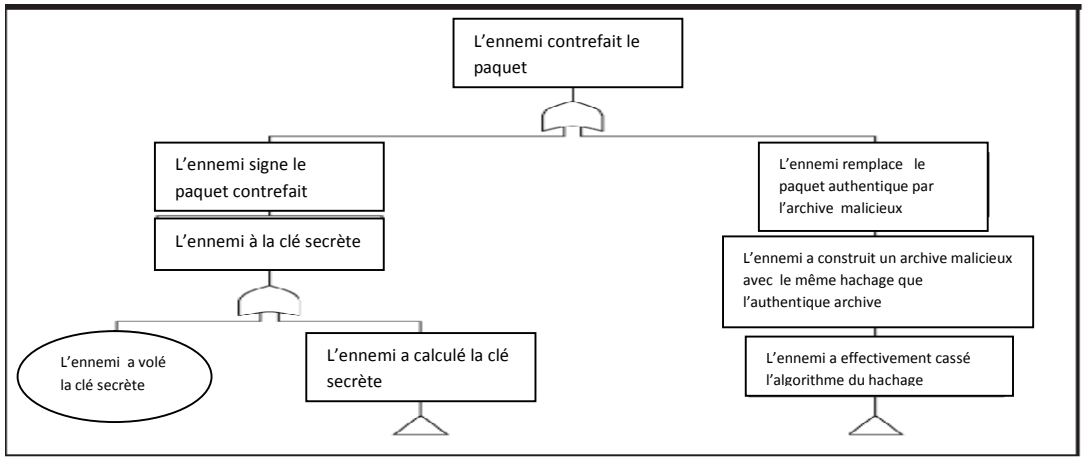


Figure 14 : Exemple FTA pour la signature des paquets logiciels

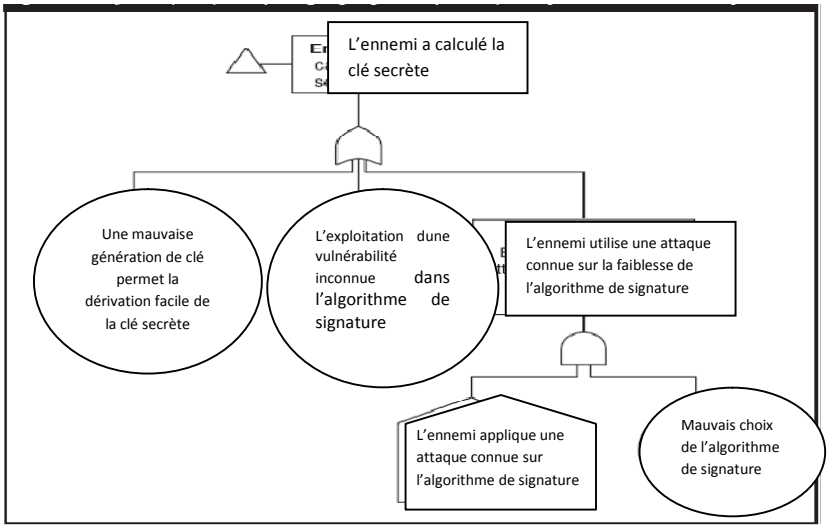


Figure 15: Développement de l'évènement « L'ennemi a calculé la clé secrète ».

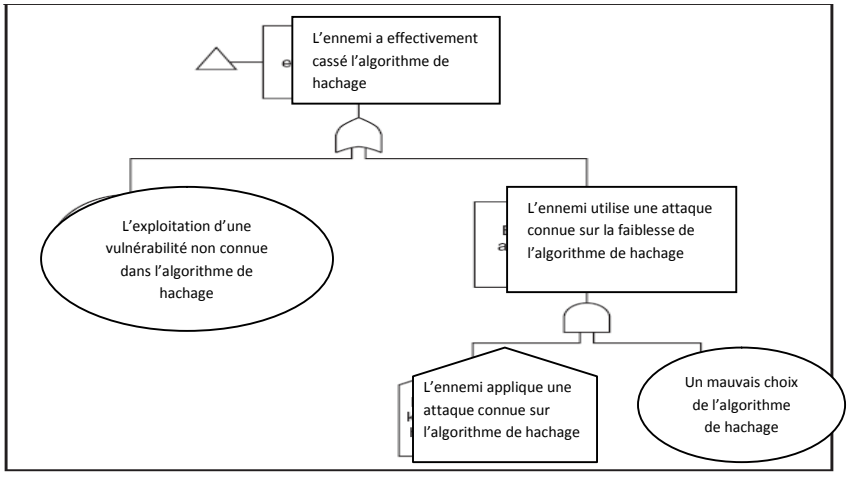


Figure 16 : Développement de l'évènement « L'ennemi a effectivement cassé l'algorithme du hachage ».

Dans cet exemple, en supposant que les algorithmes de hachage et de signature choisis sont solides, on peut conclure que le risque major provient de l'utilisateur, si l'attaquant peut avoir la clé directement de l'ordinateur de l'utilisateur, c'est que ce dernier n'a pas protégé correctement sa clé.

I.1.4 Les faiblesses de l'arbre des fautes

L'inconvénient de l'arbre des fautes est sa taille, un très petit système peut engendrer un arbre très large. Deuxième inconvénient, c'est la difficulté d'assigner aux évènements de base des probabilités, à cause de la nature discrète non linéaire des systèmes informatiques (Brook, et al., 2003).

I.2 L'arbre des attaques

L'arbre des attaques a été conçu pour simplifier l'évaluation du système de sécurité. Il offre un moyen méthodologique pour analyser la sécurité du système, et déterminer ses points forts et ses faiblesses en considérant les tentatives d'intrusions (Pudar, et al., 2009). Le principe est de mettre en évidence les différents scénarios qui pourraient être suivis par l'attaquant pour atteindre son objectif. Les arbres d'attaques sont utilisés pour capturer les étapes de l'attaque et l'interdépendance entre ces étapes.

L'arbre d'attaque défini par Scheiner dans (Bruce, 1999) est constitué de trois parties : la racine, les feuilles et les composants sous arbres. Le nœud racine dénote le comportement désiré du système compromis. Les feuilles décrivent les tentatives d'attaque comme des actions simples indivisibles. Les composants sous arbres peuvent être des évènements AND ou OR. Les nœuds AND décrivent la dépendance requise pour compromettre tous les fils afin de compromettre le composant. Les nœuds OR signifient qu'il est suffisant qu'un fils soit compromis pour

compromettre le sous arbre. L'arbre d'attaques modélise le comportement de l'attaquant par l'énumération de tous les scénarios d'attaques possibles en vu de compromettre le but représenté par la racine. Une illustration simple de l'arbre des attaques est faite par l'exemple ci-dessous (Pudar, et al., 2009).

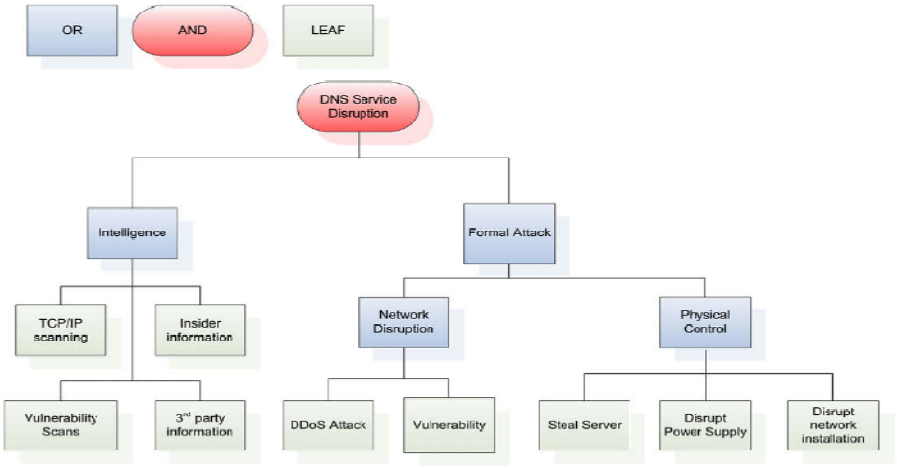


Figure 17 : L'arbre des attaques représentant une attaque DNS simple

Une fois l'arbre d'attaque créé, le concepteur peut assigner des valeurs pour chaque feuille. Les valeurs sont de nature qualitative ou quantitative. Le choix de la nature des valeurs dépend de l'habilité de définir des valeurs raisonnables. En général les valeurs utilisées sont les probabilités ou le coût des attaques. Ces valeurs peuvent être déterminées en cherchant des attributs pour les attaques particulières des feuilles. Prenant l'exemple d'un des nœuds feuille de l'exemple de la Figure 17, qui décrit l'attaquant utilisant une vulnérabilité présente dans le serveur DNS interne de la victime. Cet évènement peut être décrit par sa probabilité statistique évaluée en pourcentage. Dans cet exemple le pourcentage est déterminé en examinant les vulnérabilités de la machine qui permet au serveur d'être atteint par l'attaquant. De même, on peut associer le coût nécessaire à l'exploitation de cette vulnérabilité.

Les méthodes d'analyse de l'arbre d'attaque tentent en général de calculer la valeur, souvent la probabilité ou le coût, du nœud racine et analyser les scénarios d'attaque.

Les scénarios d'attaques sont des ensembles de combinaisons des nœuds feuilles qui permettent d'accéder à la racine. Pour l'arbre d'attaque de la Figure 17, il y a 20 scénarios d'attaques, chacun consiste en une feuille du sous arbre de la gauche et une feuille du sous arbre de la droite, ces scénarios d'attaques peuvent être utilisés pour trouver de l'information additionnelle telle que l'attaque à moindre coût et l'attaque avec une grande chance de succès.

I.2.1 Les modèles se basant sur l'arbre des attaques

Des travaux, se sont focalisés sur la représentation des processus d'attaques par des arbres d'attaques, ces travaux sont recensés dans (Vache, 2008). Parmi ces travaux, on trouve l'approche de G. Rice et J. Davis (Rice, et al., 2002.), les arbres d'attaques ont pour racine le but de l'attaquant, représenté par un losange, l'arbre dresse ensuite les différentes façons d'arriver à ce but. Les rectangles de ce graphe sont les sous objectifs à atteindre avant d'arriver au but final de l'attaquant. L'arbre d'attaques met en relief les différents moyens de l'attaquant pour atteindre ses objectifs.

Dans l'approche (Evans, et al., 2004), les auteurs ont utilisé l'arbre des attaques dans l'analyse du risque, ils ont développé une méthodologie en plusieurs étapes qui utilise l'arbre d'attaques, ils commencent par identifier les éléments importants du système, l'arbre est construit sur la base de ces informations. L'arbre des attaques définit l'espace des attaques contre lequel le système doit se défendre. Enfin l'analyse de l'arbre permet de définir des stratégies de défense.

I.2.2 Les modèles se basant sur l'arbre des attaques & fournissant des métriques

Dans (Dalton II, et al., 2006) les auteurs ont converti les arbres d'attaques en réseaux de Petri stochastiques généralisés (GSPN), (des valeurs sont assignées aux feuilles de l'arbre, elles représentent soit des probabilités de réalisation des attaques représentées par ces feuilles, soit des coûts de réalisation de ces attaques). Le but de l'utilisation des GSPN étant d'automatiser l'analyse en utilisant des outils de simulation, dans ce cadre les auteurs ont utilisé un outil appelé PIPE. Ils stipulent que les résultats de ces simulations et analyses pourront être utilisés pour raffiner l'arbre des attaques ou pour développer des contremesures.

L'approche présentée par D. Balzarotti (Balzarotti, et al., 2006) utilise le formalisme d'arbre des attaques. Elle produit une mesure de risque en utilisant une méthodologie en deux étapes, la première consiste en l'élaboration de l'arbre d'attaques, chaque nœud de l'arbre représentant une vulnérabilité potentielle que l'attaquant serait tenté d'exploiter. Un index numérique E (*Exploitability*) est estimé pour chaque vulnérabilité, représentant la probabilité d'une exploitation avec succès; la seconde étape met en évidence les dépendances entre les vulnérabilités qui apparaissent dans l'arbre d'attaque : la vulnérabilité A dépend de la vulnérabilité B si et seulement si, quand B a été exploitée, A devient plus facilement exploitable. Chaque index E est réévalué en tenant compte des dépendances mises à jour dans la seconde étape. Cette étape est réitérée jusqu'à convergence. Cette mesure représente pour chaque vulnérabilité la probabilité qu'un attaquant l'exploite, en tenant compte des dépendances entre les vulnérabilités. L'inconvénient est que l'arbre

d’attaques n’est pas simple à élaborer, car il n’y a pas de technique automatisée pour produire ce formalisme, ce qui rend difficile la mise au point de la mesure.

Une approche introduite dans (Nai Fovino, et al., 2009) consiste à combiner les deux arbres d’attaques et de fautes pour l’évaluation quantitative du risque pour la sécurité des systèmes complexes. Cette approche permet de considérer l’interaction entre les actes délibérés malicieux et les défaillances aléatoires. Le principe est d’intégrer l’arbre d’attaques dans l’arbre des fautes en remplaçant l’évènement de base de l’arbre de faute dont l’origine est une attaque par son arbre d’attaques, des adaptations ont été introduites pour pouvoir analyser quantitativement l’arbre de fautes ainsi obtenu, en se basant sur les probabilités d’occurrence des différents évènements de base de l’arbre des fautes (voir en annexe l’exemple détaillé) .

I.2.3 Avantages et faiblesses de l’arbre des attaques

L’arbre des attaques a prouvé sa simplicité, facilité d’utilisation et facilité pour l’analyse du résultat. En plus de la modélisation du comportement de l’attaquant, l’arbre des attaques est utile pour la modélisation des vulnérabilités des systèmes. Cependant les capacités de l’arbre des attaques sont limitées, selon (Pudar, et al., 2009) pour une analyse fondée, assigner des valeurs aux feuilles est difficile. Généralement c’est une estimation. Cette subjectivité et incapacité de trouver des valeurs précises est l’une des grandes faiblesses de l’arbre des attaques.

A cause de la simplicité de la modélisation, l’arbre des attaques ne peut représenter que le modèle statique du système de sécurité et du comportement de l’attaquant. En plus, les constructeurs limités de l’arbre d’attaques échouent dans la modélisation des dépendances entre les évènements et les évènements séquentiels. Si le système change avec le temps, ou de nouvelles vulnérabilités apparaissent, l’arbre des attaques initial devient non valide. Donc le modèle statique est valide seulement pour un temps limité. Aussi dans l’arbre des attaques il y a une ambiguïté dans l’interprétation des nœuds car il n y a qu’un type de nœud pour représenter l'état actuel du réseau et l'action d'une attaque.

I.3 Le graphe des attaques

Le graphe des attaques (Gupta, et al., 2007) est une aide visuelle pour dresser les risques de sécurité identifiés dans un système après sa conception. Il montre les chemins que les attaquants pourront exploiter pour atteindre leurs buts.

Le réseau informatique est représenté par un graphe orienté connecté dans lequel les nœuds représentent l’état du réseau, les arcs orientés représentent les actions de l’exploitation de

vulnérabilité, les nœuds de départ pour les arcs orientés représentent les conditions d'activation de l'action, et les nœuds d'arrivée des arcs orientés représentent la conséquence de l'action. L'enchaînement des nœuds et des arcs représente la procédure de transition d'état, qui est aussi le processus d'exploitation de la vulnérabilité.

Les graphes d'attaques sont utilisés pour déterminer si les états désignant les buts peuvent être atteints par l'attaquant en tentant de pénétrer les réseaux informatiques depuis les états initiaux, le nœud de départ représente l'attaquant dans une localisation dans le réseau spécifié. Les nœuds et les arcs représentent les actions que l'attaquant entreprend et les changements sur l'état du réseau que ces actions provoquent. Les actions impliquent typiquement les étapes d'exploitation des vulnérabilités dans le système. Le graphe d'attaques complet montre les séquences possibles des actions des attaquants qui aboutissent éventuellement au but (Wu, et al., 2008).

Mais l'inconvénient majeur de cette représentation est que le nombre d'états croit de façon exponentielle, puisqu'on représente tout les états du réseau, ce qui rend le graphe illisible et incompréhensible par l'exploitant.

I.3.1 Les modèles se basant sur le graphe des attaques

Dans l'approche *Scenario graphs and attack graphs* (Sheyner., 2004), chaque état du graphe représente l'ensemble des privilèges de l'attaquant et l'ensemble de ses connaissances ainsi que l'état de son environnement. Ainsi, il y aura changement d'état dans le graphe, lorsqu'une action a lieu, même si celle-ci n'apporte pas à l'attaquant de privilèges supplémentaires. Par exemple, un balayage des accès (*port scan* en anglais) peut apporter à l'attaquant de nouvelles connaissances sans lui donner de nouveaux privilèges. Le graphe d'attaque peut s'obtenir directement à partir de l'analyse du réseau, mais il doit dans ce cas-là être réduit pour pouvoir être exploité.

I.3.2 Les modèles se basant sur le graphe des attaques fournissant des métriques

Pour représenter les vulnérabilités présentes dans un système informatique et en évaluer les conséquences, une méthode générale d'évaluation quantitative de la sécurité basée sur le graphe des privilèges a été développée par Dacier (Dacier, 1994). A partir des vulnérabilités que l'on sait présentes sur le système, le graphe met en évidence les différentes combinaisons d'exploitation de vulnérabilités qu'un attaquant pourrait utiliser afin de mettre en défaut l'objectif de sécurité choisi. La pondération des arcs du graphe des privilèges par un effort à fournir pour réussir l'exploitation aboutit à une mesure : le *Mean Time To Failure*. Cette approche se décompose en trois étapes :

• Identification des vulnérabilités du système, représentation de ces vulnérabilités sous la forme d'un *graphe des privilèges*, et définition des objectifs d'évaluation de la sécurité à partir de ce graphe ;

• Génération automatique de scénarios d'attaque en minimisant les hypothèses concernant les stratégies mises en œuvre pour exploiter le graphe des privilèges et mettre en défaut des objectifs de sécurité ;

• Calcul de mesures quantitatives caractérisant la capacité des systèmes à résister à des attaques.

I.4 Les modèles stochastiques

Les approches utilisant les modèles stochastiques convertissent ces modèles en chaines de Markov, et les analysent en utilisant la matrice des transitions des états stables. Dans (Madan, et al., 2002), les auteurs ont utilisé le diagramme d'états dynamique pour décrire le comportement du système tolérant l'intrusion (Figure 18). Ce diagramme d'état générique est un modèle de processus semi-markovien, du point de vue quantification de sécurité, des fonctions de distribution de temps de séjour dans un état peuvent être non-exponentielles, donc le modèle stochastique a besoin d'être modélisé en termes de processus semi markovien (SMP), qui est ensuite résolu en utilisant la Chaine de Markov Temporelle Discrète (DTMC). L'avantage de ce travail est qu'il présente un modèle générique qui peut être utilisé pour les attaques de sécurité spécialisées. L'analyse quantitative du modèle produit deux métriques utiles, l'existence de l'état stationnaire et le temps moyen pour la défaillance de la sécurité (MTTSF).

Les modèles stochastiques assurent une modélisation puissante qui n'est pas présente dans l'arbre des attaques. Malheureusement, le fait que le comportement de l'attaquant ne suit aucune des fonctions de distribution connues et utilisées dans les modèles stochastiques, les distributions de probabilités sont supposées des hypothèses. Pour développer un modèle d'attaque précis, il n'est pas rigoureux d'utiliser les modèles stochastiques avec une distribution de probabilités définie (Pudar, et al., 2009).

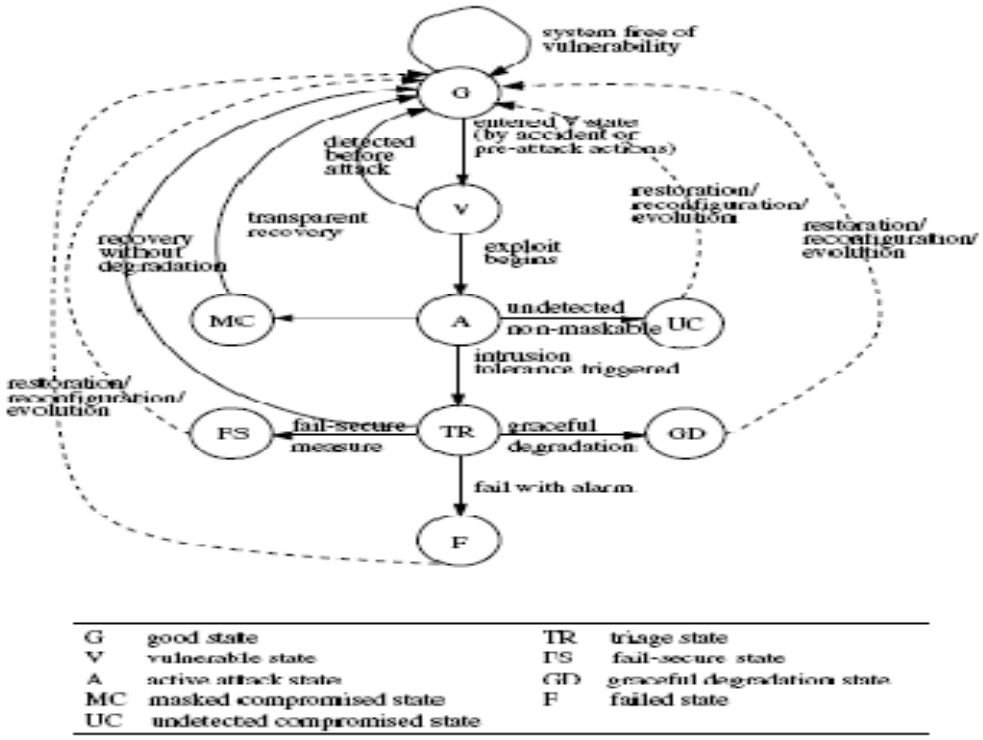


Figure 18 : Diagramme de transition d'états pour les systèmes tolérant les intrusions.

I.4.1 Les modèles de files d'attente

Les modèles de files d'attente traitent dans leurs majorités les attaques de type déni de service DOS, c'est un type d'attaque visant à rendre indisponible pendant un temps indéterminé les services ou ressources d'une organisation. Ces travaux se sont orientés vers l'étude analytique de l'impact de l'attaque DOS en utilisant les files d'attentes, motivés par le fait que si l'attaque à un impact sur un paramètre du système informatique, alors le paramètre peut être utilisé comme une mesure de détection d'attaque. Dans cette optique, les auteurs dans (Khan, et al., 2005) se sont orientés vers l'analyse de l'impact de l'attaque DOS sur trois paramètres : le taux d'arrivée, le temps de réponse et le taux de croissance de la file d'attente. Ces paramètres pourront être utilisés comme mesure pour détecter l'attaque. Dans leur modélisation, ils considèrent la ressource CPU et son buffer, qu'ils modélisent en file d'attente simple M/M/1/k avec une politique de service cyclique round robin.

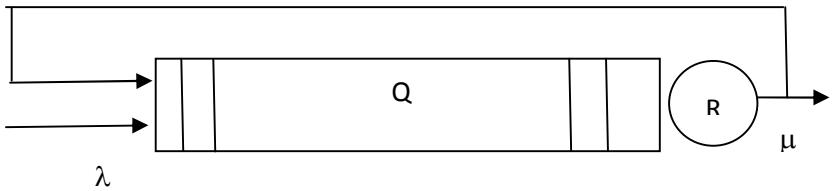


Figure 18.1 : Le modèle de ressource.

Q représente le buffer, R représente la ressource(CPU), λ et μ représentent les taux d'arrivée et de départ des requêtes pour le CPU.

Dans la même direction de recherche, et afin d'étudier le système sous une attaque DOS, les auteurs dans (Wang, et al., 2007) ont modélisé le réseau informatique sous l'attaque Syn flooding, par une file d'attente à deux dimensions avec N serveurs, deux processus d'arrivée et deux types de temps de service avec différentes distributions. Ce modèle est décrit par une chaine de Markov à deux dimensions. La distribution de la probabilité stationnaire relative à cette chaine est obtenue à l'aide d'un algorithme de censure « A level-eliminating algorithm » qui permet de résoudre un système d'équations linéaires pour l'obtention des probabilités à l'état stationnaire. Cette distribution est utilisée pour calculer des mesures de performances telle que la probabilité de perte de connexion.

Dans le but de quantifier le dommage qu'une attaque peut avoir sur les performances d'un réseau informatique, l'auteur dans (Aissani, 2008) a repris le même modèle mais avec une distribution arbitraire des temps de services, et une méthode différente pour le calcul des probabilités à l'état stationnaire, la distribution des probabilités stationnaires est obtenue par la résolution d'un système d'équations différentielles partielles.

.I.5 Modèles avec les Réseaux de Petri

Le réseau de Petri pour les modèles d'attaques, contient deux types de nœuds. L'un correspond aux places qui décrivent les états du système. L'autre aux transitions qui décrivent les actions. Les places et les transitions sont connectées par des arcs orientés.

Chaque place contient un ensemble de jetons. Les jetons peuvent se déplacer d'une place à une autre le long des arcs. Les expressions d'arcs décrivent le changement d'état du réseau de Petri, quand une transition s'exécute, l'expression de l'arc spécifie le nombre de jetons à retirer des places d'entrés et à ajouter aux places de sorties d'une transition. Le type de distribution des jetons représente un certain état du système. Si le jeton est dans une place, alors l'attaquant a gagné le contrôle de l'entité correspondante, dans l'état représenté par la place.

I.5.1 Les modèles basés sur les réseaux de Petri

L'approche appelée réseaux d'attaque qui utilise les réseaux de Petri pour modéliser l'attaque a été proposée en premier par (McDermott, 2001). McDermott a appliqué l'idée qui combine réseaux de Petri avec hypothèses de failles dans le test de pénétration. Cependant McDermott n'a considéré que la modélisation du processus de test de pénétration, mais il n'a pas pris avantage des caractéristiques de la hiérarchisation des réseaux de Petri. Les auteurs de (Wu, et al., 2008) dans leur

approche qui est basée sur l'idée de McDermott, utilisent le réseau de Petri coloré hiérarchisé pour décrire la situation d'attaques par deux niveaux, le premier niveau représente le chemin d'attaque et le deuxième niveau représente les spécifications de certaines actions d'attaque. Dans les réseaux de Petri colorés, chaque jeton véhicule une valeur de donnée qui appartient à un type donné. Cette expansion améliore largement la capacité d'expression du modèle.

Le premier niveau modélise les réseaux, les états des hôtes telle que la relation de confiance, le privilège d'accès, les connections de réseaux etc… et les actions d'attaque dues à l'exploitation de vulnérabilités. Chaque chemin représente une séquence de vulnérabilités exploitée qui peut aboutir à un état imprévu du réseau. Chaque vulnérabilité exploitée représente une attaque atomique. Le détail de l'activité d'attaque est décrit dans le second niveau. Cette approche permet la construction d'un modèle large en utilisant de petits CP-nets qui représentent les détails des transitions de substitution utilisées dans le premier niveau, ce qui facilite l'exploitation du graphe.

I.5.2. Approche basée sur les Réseaux de Petri colorés et fournissant des métriques

G.Vache (Vache, 2008) propose une approche dont l'objectif est d'obtenir une mesure qui traduit la probabilité que le système soit compromis par l'exploitation d'une vulnérabilité. Cette approche se base sur l'étude d'un phénomène influençant un processus d'attaque, le cycle de vie de la vulnérabilité :

1) la **découverte** de la vulnérabilité. Cette découverte peut être faite par une personne malveillante ou bien intentionnée.

2) la **publication** de la vulnérabilité ;

3) la **publication du correctif** de la vulnérabilité.

Ce modèle prend en compte plusieurs facteurs importants :

1) les principaux évènements du cycle de vie de la vulnérabilité et l'existence potentielle d'une attaque ;

2) l'action et la rigueur de l'utilisateur ;

3) l'état du système.

L'approche est modélisée par un réseau de Petri car ce formalisme permet de mettre facilement en évidence le comportement du système et de son environnement. Cette modélisation a pour but de générer une mesure de la probabilité d'intrusion d'un système vis-à-vis d'une vulnérabilité. Il faut déterminer les distributions de probabilités pour chaque évènement, et déterminer si une hypothèse markovienne est possible. Un logiciel de modélisation Mobïus permet à la fois la résolution analytique d'un modèle dans le cadre Markovien et la résolution par simulation.

Description du modèle

Le réseau obtenu est représenté sur la Figure 19.

Les phases du cycle de vie de la vulnérabilité identifiées sont représentées par l'ensemble de places {*V*EXISTE, *V*DECOUVERTE, *V*PUBLIEE, *V*CORRIGEE}. Elles sont séparées par les évènements représentés par les transitions {*découverteV, publicationV,correctionV*}. La place ¬*V*EXISTE traduit l'état où la vulnérabilité (et le composant qui la contient) n'existe pas encore. Le passage de la transition *créationV* indique que la vulnérabilité existe.

Dans l'état ¬*Install,* il est considèré que le composant vulnérable n'a pas encore été installé et que le système n'est pas en danger. Le passage de la transition *InstallationV* indique que la vulnérabilité a été installée sur le système : le système devient vulnérable. Tant que la vulnérabilité n'a pas été découverte, elle peut être installée sur le système sans risquer d'être exploitée. Dès l'instant où la vulnérabilité est découverte, un moyen d'exploiter cette vulnérabilité peut être mis au point par un attaquant – modélisé par le passage de la place ¬*Exploit* à la place *Exploit* par tirage de la transition *ExploitationV*. Le système devient alors exploitable. Trois conditions sont donc nécessaires pour rendre le système exploitable vis-à-vis d'une vulnérabilité : 1) la vulnérabilité existe et a été découverte ; 2) il existe un moyen d'exploiter cette vulnérabilité ; 3) la vulnérabilité est présente sur le système. Dès que ces trois évènements ont eu lieu, le système est considéré comme étant exploitable (passage d'une des transitions instantanées).

En présence d'une vulnérabilité, le système peut être : 1) **vulnérable**, représenté par l'état *Install*, la vulnérabilité est présente sur le système ; 2) **exploitable**, représenté par l'ensemble d'états {*VVE, VVP, VVC*} : la vulnérabilité est présente sur le système mais elle n'a pas encore été exploitée avec succès ; 3) **compromis**, représenté par l'ensemble d'états {*IVE, IVP, IVC*} : la vulnérabilité a été exploitée avec succès ; 4) **corrigé**, représenté par l'état *C* : le correctif a été appliqué mais les dégâts liés à une intrusion ne sont pas réparés ; 5) **sain**, représenté par l'état *S* : le correctif a été appliqué et les dégâts éventuels dus à une intrusion ont été réparés. Les états du système vulnérable et introduit sont chacun représentés par trois places dans le réseau de Petri. Les trois états {*VVE, VVP, VVC*} (respectivement {*IVE, IVP, IVC*}) correspondent à l'état exploitable (respectivement compromis) sachant une phase du cycle de vie : la vulnérabilité a été découverte, découverte et publiée ou découverte, publiée et corrigée.

L'état introduit est le résultat de l'exploitation réussie de la vulnérabilité sur le système. Cette exploitation est symbolisée par l'ensemble des transitions {*attaque/VE, attaque/VP, attaque/VC*}. C'est la probabilité d'être dans cet état que l'auteur essaye de quantifier. Les états corrigé et sain ne

sont accessibles que si le correctif a été publié et installé. L'état corrigé suit l'application du correctif en état d'intrusion par tir de la transition *installationC/I*. Il met en relief que la vulnérabilité a bien été corrigée mais que les dégâts causés ne sont pas encore réparés. L'état sain peut suivre l'installation du correctif sans que le système ait subi d'attaques par tir des transitions *installationC/*$\bar{E}$ ou *installationC/V*. Il peut encore suivre l'état corrigé par tir de la transition *nettoyage* après réparation des dégâts causés par l'intrusion.

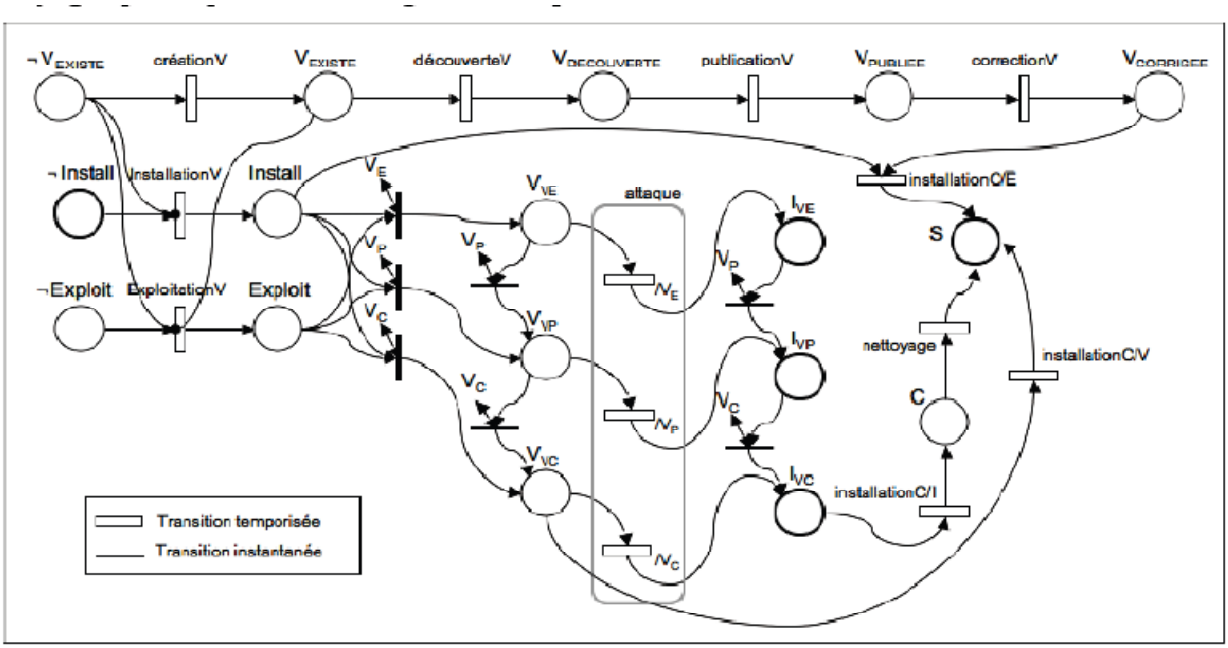


Figure 19 : Modélisation du système et de son environnement par un réseau de Petri

Cette approche décrit un processus d'attaques via une vulnérabilité. Le phénomène d'installation du correctif est un évènement probabiliste car aléatoire. Les autres évènements décrits dans cette modélisation sont de nature déterministe : ils se produiront obligatoirement. Néanmoins, le contexte de ce modèle permet de représenter ces phénomènes de manière probabiliste. Considérons par exemple le phénomène d'attaque. Si une vulnérabilité est découverte et qu'il existe une attaque disponible, cette vulnérabilité sera exploitée : il s'agit d'un évènement déterministe. Mais puisque on ne sait pas quand l'attaque aura lieu contre le système considéré. Nous obtenons donc une représentation probabiliste du système et de son environnement.

I.6 Synthèse sur les modèles formels pour les attaques

La modélisation des attaques est devenue un point chaud pour la sécurité des réseaux. Plusieurs modèles ont été proposés pour formaliser les attaques. Cependant chaque modèle a des avantages et

des inconvénients. Le tableau ci-dessous récapitule les caractéristiques des différents modèles que nous avons présentés.

La sécurité informatique a été évaluée en premier temps qualitativement, mais cette évaluation ne permet pas une évaluation régulière et un suivi de l'évolution du niveau de sécurité du système. De plus, les mesures résultantes de ces évaluations ne permettent pas de comparer des systèmes entre eux ou le même système à deux instants d'évaluation différents, de là des méthodes de quantification de la sécurité ont vu le jour. Mais que mesure t'on ?

A travers les approches que nous avons présentées, la majorité des modèles qui produisent des métriques sont des modèles qui représentent un processus d'attaque, ou d'exploitation de vulnérabilité. Les métriques sont soit des probabilités d'exploiter des vulnérabilités par l'attaquant, soit le temps ou l'effort moyen pour mettre en défaut la politique de sécurité, soit le coût du moyen de protection à utiliser, soit mesurer le temps pour compromettre, ou la probabilité d'intrusion d'un système face à une vulnérabilité. Ce panorama des mesures permet d'avoir une idée sur les attributs à considérer pour des mesures représentatives pour la sécurité.

	Approche combinatoire		Approche stochastique	Approche Graphiques	
Modèle	Arbre des fautes	Arbre des attaques	Chaine de Markov et file d'attente	Réseaux de Petri	Graphe des attaques
Inconvénients	-La taille de l'arbre. -Absence d'une méthode pour construire l'arbre -Nécessite la connaissance du domaine pour pouvoir développer l'arbre.	-Capacité de modélisation limitée (ensemble de constructeurs limités). -Le modèle est statique. -Risque d'oublier un scénario d'attaque. -Un type de nœud pour représenter l'évènement (action d'attaque) et l'état du système.	- Le comportement de l'attaquant ne suit aucune des fonctions de distribution connues et utilisées dans les modèles stochastiques. - La file d'attente ne permet pas d'exprimer la synchronisation.	-Les modèles qui n'utilisent pas les caractéristiques de compactage des PN (les couleurs, la hiérarchisation) engendrent une difficulté de perception pour l'exploitant.	Nombre d'états exponentiel.

Tableau 7 : Synthèse sur les modèles formels pour les attaques.

Critères → / *Modèles* ↓	Puissance d'expression	Représentation graphique	Utilisation d'outil automatisé
Modèles combinatoires	-	+	-
Modèles stochastiques	+/-(synchronisa,…)	+ /-	-
Modèles de réseaux de Petri	+	+	+

Tableau 7.1 : Comparaison entre les différents formalismes

La conclusion qu'on peut tirer de ces tableaux, est que chaque formalisme a des avantages et des inconvénients. Les modèles combinatoires, se caractérisent par la facilité de représenter les scénarios d'attaques, mais cette facilité engendre une puissance d'expression limitée, on note aussi l'absence d'outils pour l'analyse. Les modèles stochastiques sont plus expressifs, mais l'exactitude du modèle d'attaque développé est remise en cause par le fait que le comportement de l'attaquant ne suit aucune loi de distribution connue et utilisée dans les modèles stochastiques. Les graphes des attaques sont intéressants, mais on leur reproche la croissance exponentielle du nombre d'états, ce qui se répercute sur la clarté du graphe et complique son exploitation. Enfin le réseau de Petri, c'est le modèle qui présente le plus d'avantages et pallie à la plupart de ces inconvénients. En effet, les réseaux de Petri offrent une représentation graphique attrayante, ce qui accroît la lisibilité et facilite la compréhension ; une puissance d'expression qui permet la description des comportements complexes, et concurrents. Ils fournissent une étude comportementale et structurelle du système. La vérification peut s'effectuer directement sur le modèle sans passer par des modèles auxiliaires.

II. Conclusion

Dans ce chapitre, nous avons présenté une diversité de modèles formels concernant les attaques, nous avons remarqué que différents formalismes sont utilisés pour modéliser les attaques. Nous avons aussi abordé l'évaluation quantitative de la sécurité à travers les modèles d'attaques qui produisent des métriques, nous avons vu que différentes mesures sont produites, elles se basent toutes sur le processus d'attaque ou les vulnérabilités présentes dans le système.

Parmi les modèles d'attaques que nous avons abordé, nous citons les modèles de files d'attentes (Khan, et al., 2005), (Wang, et al., 2007) , (Aissani, 2008), pour la modélisation des attaques de type DOS. Les files d'attentes représentent un outil de modélisation utilisé pour l'évaluation quantitative des systèmes, l'avantage de ce type de modèle est qu'il possède des techniques

d’analyse exactes ou approchées et permet d’obtenir la solution du système de taille importante. Cependant on ne peut pas l’utiliser pour mener une étude qualitative, ni pour exprimer certains mécanismes telle que la synchronisation. En revanche les RdP permettent de mener l’étude qualitative et l’étude quantitative, et permettent d’exprimer de manière simple des propriétés comportementales liées au parallélisme tel que le blocage, la vivacité, l’équité. Ils supportent une théorie complète qui débouche sur de nombreux outils de preuve, telle que la construction du graphe de marquages accessible(GMA), la méthode des invariants (calcul des flots, de semi-flots), le graphe de couverture etc...Nous notons aussi la disponibilité d’outils d’édition et d’analyse (*CPNtools, Design/CPN, GreatSPN, TINA, Time-NET…),* ce qui permet de faciliter la tâche de construction du modèle, et de réaliser l’évaluation des performances d’une manière automatique. En voulant tirer profit de tous ces avantages, nous avons choisi d’utiliser ce formalisme pour la modélisation d’un type d’attaques de type DOS que nous allons présenter dans la chapitre suivant.

Le Modèle PetriDoS

5. Le Modèle PetriDoS

Introduction

Dans les chapitres précédents, nous avons fait un tour d'horizon des modèles formels utilisés dans le domaine de la sécurité informatique. Le but de tous ces modèles quelque soit le formalisme utilisé est de garantir une ou plusieurs propriétés de la sécurité. L'état de l'art que nous avons présenté nous a permis de percevoir l'importance de l'outil réseaux de Petri (RdP) avec ses caractéristiques graphiques et mathématiques, dans la modélisation des systèmes, en particulier des systèmes informatiques.

Nous avons vu également que parmi les problèmes majeurs de la sécurité, il y a les attaques, en particulier les attaques de dénis de service dont le nombre ne cesse d'augmenter chaque année et occasionne beaucoup de problèmes.

La disponibilité étant une des propriétés de la sécurité, elle assure que les demandes faites par des sujets autorisés sont toujours traitées, on dit qu'il n'y a pas de dénis de service(DoS). C'est cet objectif de la sécurité qui est remis en cause par des attaques de type Déni de Service**.** En effet, les sociétés dont la principale activité est basée sur un flux d'information internet, sont menacées. Aujourd'hui les pirates utilisent les attaques par déni de service comme nouvelle arme pour exercer un chantage contre ces sociétés. Le message est clair « la bourse ou l'exploitation de votre site ». Toute société accessible par Internet est potentiellement en danger.

Une attaque contre la disponibilité peut avoir deux origines. La première consiste à déjouer les politiques de sécurité et à exploiter une faute pour qu'elle produise une erreur affectant la délivrance du service. La seconde méthode consiste à engorger le système de demandes de services valides afin d'occuper le système et rendre sa disponibilité faible ou inexistante pour l'utilisateur légitime, les attaques de type Déni de Service appartiennent à la deuxième catégorie. Ces dernières années, plusieurs sites Internet principalement à caractère commercial, sont victimes d'attaques de DoS parmi celles-ci, on retrouve la célèbre attaque TCP/SYN sur le protocole TCP/IP qu'on appelle aussi l'attaque SYN flooding, elle est caractérisée par sa simplicité et la disponibilité d'outils sur internet pour sa réalisation. Depuis 1996, cette attaque fut perpétrée à plusieurs reprises par des intrus ayant la capacité d'initier, avec un minimum d'effort, un grand nombre d'exécutions du protocole avec un même serveur. La possibilité d'utiliser le protocole TCP/IP afin de provoquer ce DoS est partiellement due à la facilité de forger une fausse identité et, conséquemment, la difficulté pour la victime de

reconnaître un attaquant. Ce fléau a suscité l'intérêt des chercheurs et a donné naissance a plusieurs études: des études se sont orientées vers la détection de ce type d'attaques en proposant des mécanismes de détection exploitables dans les pares feu, les routeurs ou la construction des systèmes de détection d'intrusions (IDS) (Wang, et al., 2004).

D'autres se sont orientés vers la modélisation de ce type d'attaques par les arbres d'attaques ou par les réseaux de Petri dans le but d'énumérer tous les scenarios possibles pour la réalisation de l'attaque, afin de détecter les anomalies du réseau et couper la route à l'attaque ou renforcer les mécanismes de sécurité (Pudar, et al., 2009).

Des travaux de modélisation par files d'attente (citées dans le chapitre précédent) se sont orientées vers la détermination des paramètres exploitables pour la détection de l'attaque tels que le taux d'arrivée, le temps de réponse et le taux de croissance de la file d'attente (Khan, et al., 2005), ou vers l'évaluation des performances du système informatique à travers le calcul des métriques de la sécurité telles que la probabilité de perte de connexion et le pourcentage d'occupation du buffer par les paquets d'attaques. Dans cette optique, les auteurs de (Wang, et al., 2007) ont modélisé le processus d'attaque par une file d'attente à deux dimensions avec N serveurs, deux processus d'arrivée et deux type de temps de service avec différentes distributions, ce modèle est décrit par une chaine de Markov à deux dimensions. La distribution de la probabilité stationnaire relative à cette chaine est obtenue par la résolution d'un système d'équations linéaires à l'aide d'un algorithme de censure « A level-eliminating algorithm », basé sur la structure de la matrice des probabilités de transitions et qui permet de faire le calcul sur des matrices de grandes dimensions par la décomposition de celles –ci. Cette distribution est utilisée pour calculer des mesures de performance. Dans (Aissani, 2008), l'auteur a repris le même modèle mais avec une distribution arbitraire des temps de services et une méthode différente, de celle utilisée dans le modèle de Wang, pour le calcul des probabilités à l'état stationnaire, qui sont obtenues par la résolution d'un système d'équations différentielles partielles. Le but de l'auteur est de quantifier le dommage qu'une attaque réussie peut avoir sur un réseau informatique, à travers le calcul des paramètres de performances tels que, la probabilité de perte de connexion, et le taux d'occupation de buffer par les connexions semi-ouvertes.

Les files d'attente sont souvent utilisées pour une évaluation quantitative des systèmes, cependant elles ne sont pas adaptées pour mener une étude qualitative du système modélisé, ni d'exprimer certains mécanismes des systèmes parallèles tels que la synchronisation, l'interblocage et l'allocation de ressources. Pour cela nous avons pensé à utiliser les réseaux

de Petri pour modéliser un système informatique sous ce type d'attaques, nous nous sommes basés sur le modèle de Wang. Nous avons opté pour la variante réseau de Petri stochastique et déterministe coloré (RdPSD). Le choix est justifié par la puissance de modélisation de ce formalisme de haut niveau approprié pour décrire et analyser les performances des systèmes caractérisés par les propriétés de concurrence et de synchronisation. En plus de la disponibilité d'outils pour les analyses qualitative et quantitative. En effet le caractère aléatoire du processus de demande de connexion et d'attaque, nous a orientés vers les réseaux de Petri ***stochastiques***, la présence du time out (un délai fixé et constant) pour les paquets dans le buffer du serveur nous a dirigé vers les réseaux de Petri ***déterministes***, et enfin pour distinguer les paquets d'attaques, des paquets légitimes nous avons ajouté ***la couleur***. Nous avons appelé ce modèle PetriDos.

I. Les attaques par Déni de Service

Une attaque par déni de service en anglais **Denial of Service(DoS)** est un type d'attaques visant à rendre indisponible pendant un temps indéterminé les services ou ressources d'une organisation. Il s'agit la plupart du temps d'attaques à l'encontre des serveurs d'une entreprise, afin qu'ils ne puissent être utilisés et consultés.

Les attaques par déni de service sont un fléau pouvant toucher tout serveur d'entreprise ou tout particulier relié à internet. Le but d'une telle attaque n'est pas de récupérer ou d'altérer des données, mais de nuire à la réputation de sociétés ayant une présence sur internet et éventuellement de nuire à leur fonctionnement si leur activité repose sur un système d'information.

On distingue habituellement deux types de dénis de service :

- Les dénis de service par saturation, consistant à submerger une machine de requêtes, afin qu'elle ne soit plus capable de répondre aux requêtes réelles ;
- Les dénis de service par exploitation de vulnérabilités, consistant à exploiter une faille du système distant afin de le rendre inutilisable.

I.1 Quelques attaques DoS

I.1.1 Attaque par réflexion (Smurf)

La technique dite attaque par réflexion en anglais *smurf* (www.Commentcamarche.net) est basée sur l'utilisation de serveurs de diffusion broadcast pour paralyser un 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. Le scénario d'une telle attaque est le suivant :

- la machine attaquante envoie une requête *ping* à un ou plusieurs serveurs de diffusion en falsifiant l'adresse IP source (adresse à laquelle le serveur doit théoriquement répondre) et en fournissant l'adresse IP d'une machine cible ;
- le serveur de diffusion répercute la requête sur l'ensemble du réseau ;
- toutes les machines du réseau envoient une réponse au serveur de diffusion,
- le serveur broadcast redirige les réponses vers la machine cible.

Ainsi, lorsque la machine attaquante adresse une requête à plusieurs serveurs de diffusion situés sur des réseaux différents, l'ensemble des réponses des ordinateurs des différents réseaux vont être routées sur la machine cible.

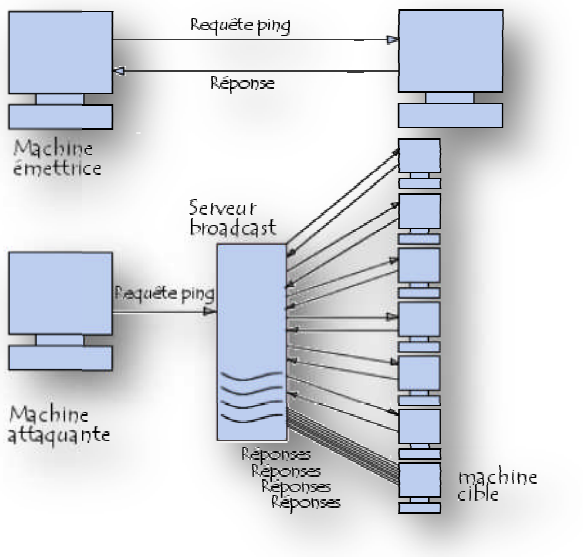


Figure 20 : L'attaque Smurf.

De cette façon l'essentiel du travail de l'attaquant consiste à trouver une liste de serveurs de diffusion et à falsifier l'adresse de réponse afin de les diriger vers la machine cible.

I.1.2 Attaques Teardrop

Les attaques Teardrop (www.Commentcamarche.net) utilisent une faille dans le réassemblage des paquets IP fragmentés en insérant des données de décalage erronées dans l'en-tête des paquets. Une machine recevant ces paquets IP risque de redémarrer si elle est sensible à cette attaque.

I.1.3 Attaque Unreachable Host

Cette attaque envoie des messages ICMP de type « Host unreachable » (www.Commentcamarche.net) à une cible, provoquant la déconnexion des sessions et

paralyse ainsi la victime. La simplicité de cette attaque est qu'elle demande qu'un faible débit du fait que les envois de datagramme ICMP peuvent être sur une faible cadence.

I.1.4 Attaque UDP Flood

Le concept de cette attaque (www.Commentcamarche.net) est de saturer les ressources de la cible en termes de débit, processeur, mémoire à l'aide de datagramme UDP.

I.1.5 Attaque Mail Bombing

Cette attaque consiste à envoyer plusieurs milliers d'emails à destination d'une entreprise ou d'un utilisateur cible. L'impact est évidemment avec un remplissage massif de la boite à lettres utilisateur, mais surtout de saturer le débit internet de l'entreprise ciblée.

I.1.6 Attaque SYN/TCP

L'attaque **SYN** appelée également « *SYN Flooding* » (www.Commentcamarche.net) est une attaque réseau par saturation exploitant le mécanisme de poignée de main en trois temps du protocole TCP. Le mécanisme de poignée de main en trois temps est la manière selon laquelle toute connexion « fiable » à internet utilisant le protocole TCP s'effectue (figure 21).

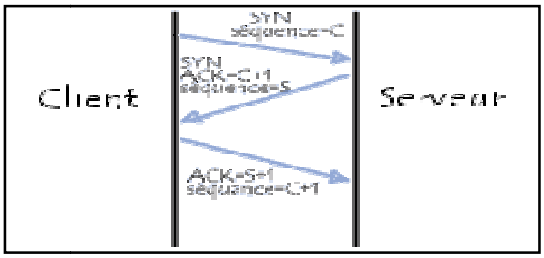


Figure 21 : L'attaque SYN/TCP.

Lorsqu'un client établit une connexion à un serveur, le client envoie une requête *SYN*, le serveur répond alors par un paquet *SYN/ACK* et enfin le client valide la connexion par un paquet *ACK*.

Une connexion TCP ne peut s'établir que lorsque ces 3 étapes ont été franchies. L'attaque SYN consiste à envoyer un grand nombre de requêtes SYN à un hôte avec une adresse IP source inexistante ou invalide. Ainsi, il est impossible que la machine cible reçoive un paquet *ACK*.

Les machines vulnérables aux attaques SYN mettent en file d'attente, dans une structure de données en mémoire, les connexions ainsi ouvertes (ce qu'on appelle connexions semi ouvertes), et attendent de recevoir un paquet *ACK*. Il existe un mécanisme d'expiration permettant de rejeter les paquets au bout d'un certain délai. Néanmoins, avec un nombre de

paquets SYN très important, si les ressources utilisées par la machine cible pour stocker les requêtes en attente sont épuisées, elle risque d'entrer dans un état instable pouvant conduire à un blocage ou un redémarrage.

De part la nature simple de cette attaque et la disponibilité d'outils nécessaires à sa réalisation, a fait que ce type d'attaques soit le plus utilisé des attaques de type Dos.

II. Le modèle PetriDos

II.1 Modèle de base

Dans le modèle de Wang (Wang, et al., 2007), les arrivées et les services sont des processus stochastiques. Les arrivées des paquets légitimes et des paquets d'attaques des requêtes de connexion SYN sont toutes les deux, des processus de poisson de taux λ_1 et λ_2 respectivement. Il est supposé que chaque connexion semi-ouverte dans le buffer est maintenue au plus tard une période déterminée B (un time out), qui est l'intervalle de temps depuis que la connexion semi-ouverte a débuté (arrivé de SYN) jusqu'à ce qu'elle soit abandonnée (pas d'ACK, si c'est une attaque). La victime a un buffer de connexion pour la file des requêtes non traitées dans lequel N connexions semi-ouvertes sont permises simultanément, il y a donc N serveurs. Les paquets arrivés lorsque le buffer est plein sont rejetés. Il est supposé que les connexions semi-ouvertes pour les paquets légitimes sont maintenues pour une durée aléatoire qui est distribuée exponentiellement avec un paramètre μ (arrivée des ACK). Les deux processus d'arrivée sont indépendants l'un de l'autre et sont indépendants des temps de maintien des connexions semi-ouvertes. Le processus d'attaque est modélisé par une file d'attente à deux dimensions, à N serveurs, à source infinie et à nombre d'états finis.

Nous allons reprendre le modèle de Wang (Wang, et al., 2007) avec les mêmes hypothèses, que nous modélisons avec les réseaux de Petri stochastiques et déterministes colorés (RdPSDC) en vue d'évaluer les performances du système pendant l'attaque, à travers la quantification d'un des attributs de sécurité. Nous nous focalisons sur la propriété de disponibilité (la probabilité de perte de connexion et le taux d'occupation du buffer par les paquets d'attaques). Ces mesures de performances permettront de dimensionner les paramètres pour garantir la disponibilité des services pendant les attaques. Le choix du RdPSDC est justifié par la puissance de modélisation de ce formalisme de haut niveau, qui est plus compact et plus concis que les réseaux de Petri ordinaires, et par le caractère aléatoire du processus d'attaque et de demande de connexion, ce qui justifie le réseau de Petri stochastique. La présence du time out B justifie le choix des réseaux de Petri déterministes. La

diversité des paquets de demande de service justifie les couleurs. En effet en plus de la différence dans les taux d'arrivée, les couleurs sont utilisées dans notre modèle pour distinguer les paquets légitimes des paquets d'attaques. Dans la littérature, nous n'avons rencontré aucun travail de modélisation avec des réseaux de Petri stochastiques et déterministes, qui utilise le coloré. Donc notre contribution consiste en la modélisation par le formalisme réseaux de Petri du processus de l'attaque TCP/SYN, ainsi que l'introduction de la couleur dans les réseaux de Petri stochastique et déterministe.

II.2 Les réseaux de Petri stochastiques et déterministe RdPSD

Les réseaux de Petri stochastiques *RdPS*

Les réseaux de Petri stochastiques (Molloy, 1982) sont une extension des RdP, définis dans le but d'augmenter la puissance de spécification en prenant en compte la notion de temps. Ils sont caractérisés par des transitions temporisées, à chaque transition est associée une variable aléatoire modélisant le délai de franchissement de cette dernière, ces transitions décrivent les opérations nécessitant un temps aléatoire pour s'exécuter. Mais des fois dans un même système à modéliser, on éprouve le besoin de représenter au même titre des opérations à caractère aléatoire et des actions de synchronisation ou des opérations d'allocation des ressources, les transitions associées à ce genre d'actions sont immédiates.

Les modèles RdPS qui incluent des transitions temporisées avec un délai de franchissement de distribution exponentielle et des transitions immédiates avec un temps de franchissement nul sont appelés réseaux de Petri stochastiques généralisés (RdPSG).

Les réseaux de Petri stochastiques et déterministe RdPSD

Les modèles RdPS qui incluent des transitions temporisées avec un délai de franchissement constant ou de distribution exponentielle, et des transitions immédiates avec un temps de franchissement nul sont appelés réseaux de Petri stochastiques déterministes (RdPSD) (M. Ajmone, et al., 1987).

Les réseaux de Petri stochastiques et déterministes colorés RdPSDC

Au réseau de Petri stochastique déterministe, nous ajoutons la notion de couleur, qui permet de modéliser les comportements similaires au moyen d'une seule représentation condensée.

Dans le modèle PetriDos que nous avons proposé, les paquets de demandes de connexion et les paquets d'attaques ont le même comportement, nous allons exprimer un seul comportement et utiliser les couleurs des jetons pour différencier les deux types de paquets.

II.3 Le modèle PetriDos

PetriDos (figure 22) est une modélisation par réseaux de Petri stochastiques et déterministes colorés de l'attaque de type DOS intitulée SYN/TCP ou SYN Flooding. Dans notre modélisation, une place représentée par un cercle peut contenir des jetons de différentes couleurs, elle représente l'état du système. La transition représente l'évènement qui provoque le changement d'état du système, dans notre modèle il y a trois type de transitions, les transitions avec un délai de franchissement de distribution exponentielle, et qui sont représentées par un rectangle, on leur associé des variables aléatoires modélisant le délai de franchissement de ces transitions, i.e. l'intervalle de temps qui sépare l'instant de déclenchement, de l'instant de la fin de l'évènement. Les transitions pour lesquelles le temps de franchissement est constant sont représentées par des rectangles noirs pleins, celles pour lesquelles le temps de franchissement est nul sont représentées par un trait noir épais, les transitions immédiates sont plus prioritaires que les transitions temporisées. Nous ajoutons au modèle de Wang un processus d'arrivée, ce processus représente le déclenchement de plusieurs attaques SYN, avec un taux λ_0 représentant l'intervalle entre le déclenchement d'une attaque et l'attaque suivante.

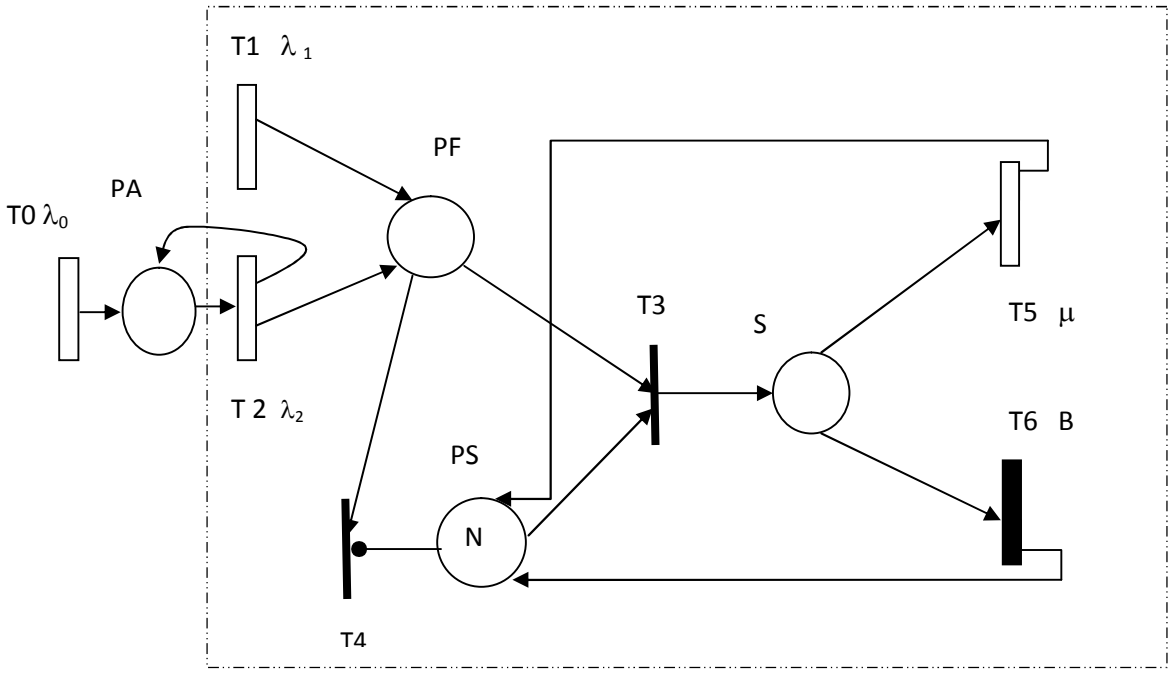


Figure 22 : Le modèle PetriDos.

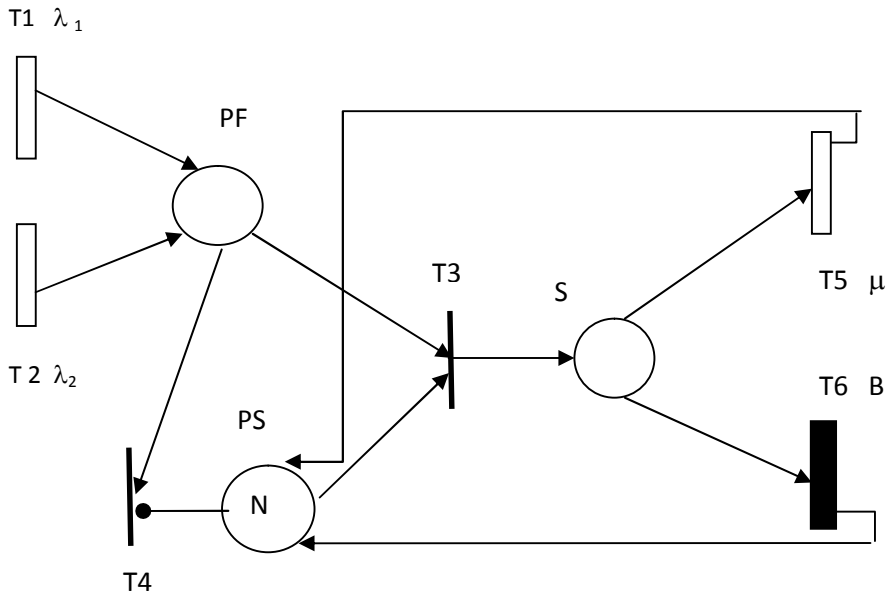


Figure 23: Le Modèle PetriDos étudié.

Nous allons considérer dans un premier temps le processus d'une seule attaque, c'est le modèle encadré par des pointillés dans la figure 22, et repris dans la figure23.

II.3.1 Description du modèle

Les places

PF : contient les paquets (légitimes et d'attaques) prêts pour le service.

S : contient les paquets (légitimes et d'attaques) en service.

PS : les serveurs disponibles (le nombre de cases disponibles dans le buffer).

Les Transitions

T1 : l'arrivée des paquets légitimes avec un taux $\lambda_{1.}$

T2 : l'arrivée des paquets d'attaque avec un taux λ_{2}.

T3 : transition immédiate pour la synchronisation entre les paquets prêts pour le service et la disponibilité des serveurs.

T4 : transition immédiate pour le rejet des paquets (légitimes et d'attaques) lorsque tous les serveurs sont indisponibles (zéro jeton dans PS).

T5 : fin de service pour les paquets légitimes (arrivée d'ACK) avec un taux μ.

T6 : fin de service pour les paquets d'attaques (pas d'arrivée d'ACK) au bout d'un délai déterminé B.

II.3.2 Définition formelle

PetriDOS = un tuple $<\Delta, P, T, Pré, Post, Arc, Col, E, Pri, Temp, M_0>$

Δ : Domaine de couleurs. Δ= {PQ, S}, PQ pour les paquets et S pour les serveurs,

PQ= {L, A},

L : pour les jetons représentant les paquets légitimes,

A : pour les jetons représentant les paquets d'attaques.

S= {SN},

SN : pour les jetons représentant les serveurs.

P : ensemble non vide de places.

T : ensemble non vide de transitions partitionné en trois ensembles disjoints T^I, T^E et T^D de transitions immédiates, exponentielles, et déterministes respectivement . Avec $P \cap T = \varnothing$.

Post : P x T$\rightarrow$N, est la fonction d'incidence avant.

Pré : P x T$\rightarrow$N, est la fonction d'incidence arrière.

Arc : ensemble fini d'arc= {$\rightarrow$, $\multimap$}, deux types d'arcs, arc simple, et arc inhibiteur pour exprimer le test à zéro.

Col : P$\rightarrow\Delta$. Col(p) est la fonction couleur qui associe à chaque place un domaine de couleur.

E :est une fonction d'expression d'arcs, qui associe aux arcs une expression d'entrée pour déterminer quel jeton retirer de la place en amont et une expression de sortie pour spécifier le type de jeton à produire.

Pri : T $\rightarrow${0,1}, est la fonction de priorité, elle associe aux transitions temporisées la valeur 1 et aux transitions immédiates la valeur 0 en sachant que 0 est plus prioritaire.

Temp : $T \rightarrow R^+$, la fonction qui associe à chaque transition temporisée un délai de franchissement.

M_0 : $P \rightarrow N$ c'est le marquage initial du réseau.

Le marquage initial du réseau est M_0= {M(PF), M(S), M(PS), M(PR)}= {(0L,0A), (0L,0A), n, (0L,0A)}.

Selon le domaine de couleur de la place, chaque place peut contenir un multi ensemble de couleur. Dans le marquage initial, par exemple (0L,0A) veut dire que la place correspondante contient zéro jeton de couleur L, et zéro jeton de couleur A.

II.3.3 Fonctionnement du modèle

Le franchissement de T1 ouT2 (toutes les deux sont des transitions sources) indique l'arrivée de paquets légitimes ou d'attaque dans la place PF, si la place PS contient au moins un serveur libre, la transition immédiate T3 est franchie, ce qui indique le début de service pour le paquet arrivé.

La transition immédiate T4 est franchie à l'arrivée d'un paquet (légitime ou d'attaque) ne trouvant pas de serveur libre (M(PS)=0), le paquet est alors rejeté.

Le franchissement de la transition T5 signifie que l'ACK attendu est arrivé et par conséquence c'est la fin de service pour le paquet légitime, et le serveur devient libre et il est prêt à servir un autre paquet. La fonction associée à l'arc (S, T5) retire un jeton de couleur L de la place d'entrée et produit un jeton de couleur SN pour la place de sortie.

Le franchissement de la transition T6 signifie que l'ACK attendu n'est pas arrivé et par conséquence c'est la fin de service pour le paquet d'attaque au bout d'une période B déterminé, et le serveur devient libre et prêt à servir un autre paquet. La fonction associée à l'arc (S, T6) retire un jeton de couleur A de la place d'entrée et produit un jeton de type SN pour la place de sortie.

L'attaque réussit lorsqu'avec un nombre très important de paquets d'attaques SYN, les ressources utilisées par la victime pour stocker les requêtes en attente sont épuisées, le serveur risque d'entrer dans un état instable pouvant conduire à un blocage ou un redémarrage. Autrement dit, dans notre modèle, cela est exprimé par le franchissement de la transition T4, et le nombre de fois où le marquage de PS est à zéro, et aussi le nombre de fois où le marquage de la place S est égal à N (la somme des jetons des deux couleurs), car c'est à partir de là que les paquets sont rejetés et le service n'est plus disponible.

II.3.4 L'analyse de performance

L'analyse des performances est basée sur le calcul de paramètres quantitatifs. Ces paramètres sont calculés à base de formules en fonction des probabilités stationnaires. Donc on a besoin de connaitre les probabilités stationnaires du modèle.

Un modèle RdPSD est sous jacent à un processus semi-markovien, à condition qu'au plus, une seule transition déterministe concurrente est sensibilisée pour chaque marquage (M. Ajmone, et al., 1987).

Les marquages du graphe de marquage représentent les différents états du système à étudier. Le passage d'un marquage à un autre se fait par le franchissement de transitions.

Dans un modèle RdPSD, on trouve deux types de marquages, le marquage évanescent et le marquage tangible, le marquage évanescent est le marquage pour lequel une transition immédiate est franchissable, et le marquage tangible, c'est le marquage pour lequel une transition temporisée (déterministe ou exponentielle) est franchissable. Un graphe de marquage réduit aux marquages tangibles, est obtenu à partir du graphe de marquage du modèle RdPSD en fusionnant les marquages évanescents avec les marquages tangibles les succédant.

Dans un marquage M_i, une transition T_j est dite ***exclusive,*** si T_j est la seule transition franchissable dans ce marquage. Si T_j est franchissable avec une autre transition T_k, T_j est ***une transition compétitive*** dans un marquage M_i si le franchissement de T_k désensibilise la transition T_j. Si le franchissement de T_k ne désensibilise pas T_j alors T_j est appelée ***transition concurrente*** dans le marquage M_i.

Dans les réseaux de Petri stochastiques la politique d'exécution est composée de deux éléments (GALLON, 1997) :

La politique de sélection de la transition à tirer quand plusieurs transitions sont sensibilisées par un marquage du réseau de Petri.

La mémoire temporelle qui autorise la prise en compte, dés l'instant d'arrivée dans un marquage, du passé du système.

On distingue deux types *de politiques de sélection* :

- La présélection « preselection Policy »: qui permet de définir, de manière statique, des priorités de tir entres les transitions.
- La compétition « race policy » qui autorise le tir de transition dont la variable aléatoire associée à statistiquement la valeur la plus petite.

On distingue trois types *de politique temporelle :*

1-La réinitialisation « resampling » qui ne prend pas en compte la passé du système ; chaque variable aléatoire est réinitialisée après le tir d'une transition ;

2-La mémoire de la dernière sensibilisation « enabling memory » où seule la dernière période de sensibilisation, qui débute au dernier instant de sensibilisation, est prise en compte ;

3-La mémoire de toutes les sensibilisations « Age Memory » où toutes les périodes de sensibilisation de la transition sont prises en compte ;

La mémoire de la dernière sensibilisation modélise naturellement des processus comme les temporisations « time out », qui sont réinitialisées dès qu'elles ne sont plus actives. Le processus débute à l'instant de sensibilisation et est réinitialisé dès que la transition est désensibilisée. Dans le cas de la mémoire de toutes les sensibilisations, ce processus débute lors de la première sensibilisation de la transition, et n'est pas réinitialisé jusqu'au tir de la transition.

Pour les RdPSD, la politique de franchissement adoptée pour choisir la prochaine transition à franchir, si plusieurs transitions temporisées sont simultanément sensibilisées, est la politique de compétition avec mémorisation de la dernière période de sensibilisation. Cette politique de compétition traduit le fait que la transition, dont l'activité associée est la première à arriver à terme, est celle qui provoque le changement d'état. La mémoire temporelle associée à une transition ne peut agir que si celle-ci est sensibilisée, dans le cas contraire la mémoire est remise à zéro. La politique de présélection est appliquée au cas où des transitions immédiates sont simultanément sensibilisées. Si des transitions immédiates et temporisées sont en concurrence, les transitions immédiates sont prioritaires.

L'analyse en régime stationnaire d'un RdPSD

La condition qu'au plus, une seule transition déterministe concurrente est sensibilisée pour chaque marquage, est vérifiée dans notre modèle, puisqu'il n'existe qu'une seule transition déterministe. Donc l'analyse à l'état stationnaire existe.

Dans un modèle RdPSD (M. Ajmone, et al., 1987), pour calculer les probabilités stationnaires on considère trois processus stochastiques différents :

- Le processus RdPSD, qui n'est pas markovien, obtenu à partir du graphe de marquage réduit au marquage tangible du modèle RdPSD, en fusionnant les marquages évanescents avec les marquages tangibles les succédant. Pour le modèle PetriDos, c'est le processus issus du graphe du marquage de PetriDos pour lequel les transitions T_3 et T_4 ont été fusionnées aux transitions tangibles qui les succèdent. Les états du processus sont les différents marquages tangibles accessibles.

- Le processus chaine de Markov à temps discret incluse CMTDI construit en considèrent des instants temporels appropriés. Pour notre modèle le processus CMTDI, correspond au processus SMP à temps continu considéré aux instants de franchissement de la transition déterministe T_6 *si* elle est sensibilisée, et aux instants de franchissement de chaque transition exponentielle (T_1,T_2,T_5) *sinon*.
- Le processus semi markovien SMP qui est un processus à temps continu correspondant au CMTDI.

La procédure pour calculer les probabilités stationnaire est résumée comme suit

1. Générer le graphe des marquages et le graphe des marquages réduit du RdPSD
2. Former la CMTDI en considérant des instants temporels appropriés du processus RdPSD (les instants de franchissement de la transition déterministe, si elle est sensibilisée, et les instants de franchissement de chaque transition exponentielle sinon).
3. Construire la matrice P de probabilité de transition d'une-étape, on considère l'instant où la transition déterministe est sensibilisée compétitivement ou en concurrence avec des transitions exponentielles.
4. Construire le vecteur des probabilités stationnaires v de la CMTDI.
5. Calculer E[S], le temps moyen de séjour dans les états du SMP.
6. Calculer le vecteur w des probabilités stationnaires du SMP en fonction de v et de E[S].
7. Construire la matrice de conversion C qui contient la différence entre le temps de séjour moyen E [M_i] dans les états du marquage du RdPSD, et le temps de séjour moyen E [S_i] dans les états du marquage du SMP.
8. Calculer le vecteur π des probabilités stationnaires du modèle RdPSD avec π=w C.

Chaque étape nécessite un calcul compliqué.

Notons l'existence d'un outil d'analyse des RdPSD appelé TimeNet, Pour l'exploiter il faut définir les transitions exclusives, compétitives et concurrentes, et respecter la condition structurelle qui stipule qu'au plus, une seule transition déterministe concurrente est sensibilisée pour chaque marquage. On doit aussi définir les paramètres de performance à calculer en respectant une grammaire défini dans l'outil. Cet outil ne fait pas l'analyse des RdSDC, nous n'avons pas rencontré d'outils qui traitent les RdPSDC.

Pour analyser le modèle PetriDos, nous allons le déplier en un modèle de réseau de Petri stochastique et déterministe, c'est-à-dire nous allons éliminer la couleur des jetons, nous allons considérer deux places pour les deux types de paquets et deux transitions de synchronisation pour l'accès aux services et deux transitions pour le rejet des deux types de paquets en cas d'indisponibilité de service.

Cette analyse et les calculs ne seront pas réalisés dans le cadre de ce mémoire, mais laissés comme perspectives pour la suite.

III Conclusion

Dans ce chapitre, nous avons présenté notre modèle PetriDos, qui consiste en la modélisation de l'attaque SYN/TCP de type DOS. C'est un type d'attaque qui augmente chaque année et qui coûte très cher puisque il interrompt le cours normal des transactions d'une organisation. L'utilisation du réseau de Petri stochastique et déterministe coloré, est justifié par les caractéristiques mathématiques et graphiques du formalisme, la puissance d'expression et aussi par le caractère stochastique du processus d'attaque et la diversité de paquets. Nous avons laissé l'analyse des performances du système sous l'attaque en perspective pour la suite.

Conclusion générale

Conclusion Générale

Le domaine de la sécurité informatique est actuellement un champ d'intérêt pour la recherche. En effet les problèmes de la sécurité informatique sont aujourd'hui relativement bien connus. Ces problèmes ont plus ou mois d'importance selon le domaine d'application du système à sécuriser, par exemple le domaine de l'industrie de la défense s'intéresse en premier lieu à la confidentialité, alors que celui de la banque donne beaucoup d'importance à l'intégrité, contrairement au domaine de la téléphonie qui privilégie la disponibilité. Des contremesures ont été définies, pour pallier à ces problèmes, mais comment garantir que ces mesures sécurisent réellement le système et que le système assure les propriétés de sécurité. La preuve des propriétés de sécurité nécessite de disposer de méthodes pour abstraire le comportement d'un système en modèle simplifié, qui peut ensuite être analysé. En exprimant une politique de sécurité à l'aide d'un modèle formel, il devient possible d'utiliser des outils mathématiques pour la vérification et la validation de la politique.

Dans ce travail nous avons fait un état de l'art sur les modèles formels utilisés dans la sécurité informatique, plusieurs aspects de la sécurité informatique y sont modélisés, entre autres les protocoles de sécurité, le contrôle d'accès et enfin les attaques. Différents formalismes sont adoptés, chaque formalisme présente des avantages et des inconvénients, les formalismes algébriques, logiques et les réseaux de Petri sont plus utilisés dans le domaine de la sécurité informatique. Nous avons présenté une récapitulation sur les modèles utilisés pour un même aspect et une comparaison entre les formalismes utilisés pour chaque aspect. Nous nous sommes intéressés aux modèles graphiques se basant sur le réseau de Petri et ses variantes, aussi nous avons présenté des exemples de modèles existants en littérature. Nous avons abordé l'évaluation quantitative de la sécurité informatique à travers les modèles des attaques, nous avons constaté la variété des mesures produites ; des approches ont produit des mesures d'effort, d'autres ont tenté de générer une mesure de la probabilité d'intrusion d'un système vis-à-vis d'une vulnérabilité. Nous avons ensuite proposé un modèle que nous avons nommé PetriDos, c'est une modélisation de l'attaque SYN flooding de type DOS par le formalisme réseau de Petri stochastique et déterministe coloré. Notre modèle peut être utilisé pour le calcul des performances du système sous l'attaque.

Perspectives

Nous avons proposé une modélisation par réseau de Petri stochastique et déterministe coloré d'une attaque de SYN flooding de type Dos, nous envisageons dans le futur :

- l'analyse du modèle ainsi que le calcul des performances du système sous l'attaque.
- comparaison des résultats avec ceux des modèles existants (le modèle de Wang et de Aissani).
- l'analyse du système sous plusieurs attaques SYN /TCP.
- Nous espérons aussi analyser le système sous l'attaque, dans les cas ou les taux d'arrivées ou de service ne sont pas exponentiels.
- Nous envisageons d'étendre notre modèle à d'autre type d'attaques de type DOS qui se basent sur la saturation des ressources (Bande passante, CPU,I/O …..).
- Prendre en considération les autres types d'attaques DOS.
- Reprendre par le formalisme Réseau de Petri, le modèle de file d'attente de (Khan, et al., 2005) pour l'analyse de l'impact de l'attaque DOS sur trois paramètres : le taux d'arrivée, le temps de réponse et le taux de croissance de la file d'attente.

Outre ces perspectives, nous nous intéressons à l'étude de toutes les propriétés de sécurité d'un système informatique, à savoir l'authentification, la confidentialité et l'intégrité.

Bibliographie

Abadi, Martin and Gordon, Andrew D. 1999. A Calculus for Cryptographic Protocols:The Spi Calculus. 1999.

Abdul Sahid, Khan, Mukund, Madhavan and P., S. 2005. Generic Verification of Security Protocols. *SPIN pp. 221–235.* 2005.

Abou El Kalam, Anas. 2003. Modèles et politiques de sécurité pour les domaines de la sante et des affaires sociales : . *THÈSE de doctorat.* 2003.

Adi, KAMEL. 2006. Protocoles cryptographiques Groupe LRSI Université du Québec en Outaouais Québec, Canada. . *Kamel Adi .* [Online] 2006. http://w3.uqo.ca/lrsi.

Aissani, Amar. 2008. Queueing Analysis for Networks Under DoS Attack University of Science and Technology Houari Boumediene (USTHB), Berlin Heidelberg . *Springer-Verlag.* 2008.

Al-Azzoni, Issam, G. Down, Douglas and Khedri, Ridha. 2005. Modeling And Verification Of Cryptographic Protocols Using Coloured Petri Nets And Design/CPN. *Nordic Journal of Computing.* 2005.

Behrmann, G., David, A. and Larsen, K. G. 2004. A tutorial on UPPAAL. . *4th International School on Formal Methods for the Design of Computer,.* 2004.

Bernat, Vincent. 2006. Théories de l'intrus pour la vérification des protocoles cryptographiques. *THÈSE pour obtenir le grade de Docteur de l'École Normale Supérieure de Cachan présentée à l'École Normale Supérieure de Cachan.* 2006.

Bidan, Christophe and Issarny, Valérie. 1995. Un aperçu des problèmes de sécurité dans les systèmes informatiques. *Publication interne n° 959 Programme1 - Architectures parallèles, bases de données, réseaux et systèmes distribués Projet Solidor .* Octobre 1995.

Bird, R, et al. 1991. Advances in Cryptology ,CRYPTO'91 Proceeding. ,*Springer Verlag.* 1991.

Boustia, N. and Mokhtari, A. 2012. A dynamic access control model. *In Applied Intelligence Journal.* 2012. Vol. 36, Number 1 PP-190-207.

Boutiller, Philippe, et al. 2008. Cahier de synthèse La sécurité informatique Un enjeu stratégique pour votre entreprise Sessions d'ateliers de l'année 2008 La Rochelle, Châtellerault, Montmorillon, Saintes, Rochefort, Cognac, Bressuire, Parthenay. 2008.

Bouzegza, Wassila. 2007. Modélisation formelle des politiques de sécurité à rôles RBAC mémoire de magister . 2007.

Braghin, C, Gorla, D. and Sassone, V. 2004. A distributed calculus for role-based access control. *Computer Security Foundations Workshop. Proceedings. 17th IEEE, pages 48–60.* 2004.

Brewer, D. and M. Nash. 1989. "The Chinese Wall Security Policy", IEEE. *Symposium on Security and Privacy, Oakland, Californie, 1-3 mai 1989, IEEE Computer.* 1989.

Briffaut, Jérémy. 2007. Formalisation et garantie de propriétés de sécurité système :application à la détection d'intrusions. *Thèse pour obtenir le grade de docteur de l'université d'ORLEANS.* 2007.

Brook, Philip j and Paige, Richard F. 2003. Fault trees for security system design and analysis.and .Computer & security. *Elseiver vol 22 N° 3 pp 256-264.* 2003.

Bruce, Schneier. 1999. Attack trees: modeling security threats. . *Dr.Dobb's Journal.* December 1999.

Burrows, Michael et Abadi, Martín et Needham, Roger. 1989. A logic of authentication. , 426(1871):233-271,. *Proceedings of the Royal Society.* 1989.

Cuppens, Frédéric. 2002. Protection des systèmes d'informations: Menaces et solutions actuelles BDA02 . Octobre 2002.

Dacier, Marc. 1994. Vers une evaluation quantitative de la securité informatique . *These de doctorat .* 1994.

Dalton II, GC, et al. 2006. Analyzing attack trees using generalized stochastic Petri nets. *IEEE information assurance workshop . pp. 116–23.* 2006.

Dolev, D. and Yao, A. 1983. On the security of public-key protocols,. *IEEE Trans. on Information Theory 29,198-203.* 1983.

Evans, Shelby, et al. 2004. Risk-based systems security engineering:stopping attacks with intention. *IEEE Security and Privacy with intention. IEEE Security and Privacy;02(6):59–62.* 2004.

Florin, G and Natkin, S. 2007. Le livre sur la sécurité informatique. 2007.

Focardi, R. and Gorrieri, R. 1994/1995. A classification of security properties for process algebras. . *Journal of Computer Security, 3(1) :5-33.* 1994/1995.

G. Chiola, D. Dutheillet, G. Franceschinis, S. Haddad,. 1993. Stochastic Well-Formed Colored nets and symmetric modeling applications. *IEEE Transactions on computers 42.* 1993.

Gallon, Laurant. 1997. Le modèle réseaux de petri temporisés stochastiques: extensions et applications. *Thèse de doctorat université Paul Sabatier de TOULOUSE.* 1997.

Gupta, Suvajit and Winstead, Joel. 2007. Using Attack Graphs to Design Systems. *Published By The IEEE Computer Society .* 2007.

Hoon, Choi, Varsha, Mainkar and S.Trivedi, Kishor. Sensitivity Analysis of Detreministic and Stochastic Petri Nets.

Jensen, K. 1997. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use, . *Springer-Verlag, Berlin,Heidelberg, New York, .* 1997.

Jiang, Y., et al. 2004. Security analysis of mandatory access control mode. *2004 IEEE International Conference.* 2004.

Joshi, J. 2003. A Generalized Temporal Role Based Access Control Model for Developing Secure Systems. *PhD thesis, Purdue Univ.* 2003.

Juopperi, J. 1995. PrT-net based analysis of information flow security nets. Research. *Helsinki University of Technology, Department of Computer Science andEngineering, Digital Systems Laboratory, Espoo, Finland.* 1995.

Juszczyszyn, K. 2001. "Modelling Information Flow Security with Coloured Petri Nets",. *35th Int. Conference on Modelling and Simulation of Systems, Ostrava, Czech Republic, vol.1, pp.287-295.* 2001.

Juszczyszyn, Krzysztof. 2003. Verifying Enterprise's Mandatory Access Control Policies with Coloured Petri Nets. Wroclaw University of Technology, Wroclaw, Poland rzysiek@ists.pwr.wroc.pl. *IEEE.* 2003.

Khan, S and Traore, I. 2005. Queue-based Analysis of DoS Attacks . In: . *Proceedings IEEE Workshop. on Information Assurance and Security, United States Military Academy, West Point,pp. 266–273. IEEE Press, New York .* 2005.

Lafrance, S. and Mullins, J. 2003. An information flow method to detect denial of service vulnerabilities. *Journal of Universal Computer Science, November.* 2003.

Lafrance, Stéphane. 2005. Spécification et validation de protocoles de sécurité . *thèse en vue de l'obtention du diplôme de philosophiae doctor (phd) (génie informatique).* avril 2005.

Lee, Gang-So and Lee, Jin-Seok. 1997. Petri net based models for specification and analysis of cryptographic protocols. 1997.

Lescop, Yves. 2002 . La Sécurité Informatique : Lescop Yves [V1.6] , Post BTS R2i . *ylescop.free.fr/mrim/cours/securite LESCOP Yves.* [Online] 2002 .

Lowe, G. 1996. Breaking and fixing the needham-schröder public-key protocol using FDR. 1996.

Lu, Y., et al. 2006. Using pi-calculus to formalize domain administration of RBAC. *In ISPEC, pages 278–289.* 2006.

M. Ajmone, Marsan and Chiola, G. 1987. On Petri Nets With Deterministic and Exponentially Distributed Firing Times. *in Advances in Petri Nets 1987, Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Verlag, , vol. 266, pp. 132–145.* 1987.

M. Fitting. 1993. Basic Modal Logic. *Handbook of Logic in Artificial Intelligence and Logic Programming Logic Foundations,.* 1993.

Madan, Bharat B., et al. 2002. Modeling and Quantification of Security Attributes of Software Systems. *IEEE 2002.*.

McDermott, J.P. 2001. Attack Net Penetration Testing. *New Security Paradigm Workshop 9•00 Ballycotton, Co. Cork, ireland.* 2001.

Meadow, C. 1992. Applying Formal Methods to the Analysis ofa Key Management Protocol,. *Journal of Computer Security.* 1992.

Méludovi, Ludovic. 2007 . Politiques et modèles de sécurité ENST-Bretagne . [Online] janvier 2007 . http://www.rennes.supelec.fr/rennes/si/equipe/lme..

Milner, R. 1989. Communication and Concurrency. *Prentice-Hall International. .* 1989.

Molloy, M. 1982. Performance Analysis using Stochastic Petri Nets . *IEEE Transaction on Computers 31,9 , 913-917.* 1982.

Murata, T. 1989. Petri nets: properties, analysis and applications. *Proceedings of the IEEE77(4),541–580.* 1989.

Nai Fovino, Igor and Masera, Marcelo. 2009. Integrating cyber attacks within fault trees. *Elseiver.* 2009.

Nieh, B and Tavares, S. 1992. Modeling and Analyzing Cryptographic Protocols using Petri Nets. *Proc. Of AUSCRYPT'92,275-295.* 1992.

Pudar, Srdjan, Manimaran, G. and Liu, Chen-Ching. 2009. PENET: a practical method and tool for integrated modeling of security attacks and countermeasures. *USA a r t i c l e i n f o Article history.* May 5, 2009.

R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. 1996. Role- Based Access Control Models. *IEEE Computer, 29(2):38–47,.* 1996.

Rakkay, H. and Boucheneb, H. 2006. Timed secure colored petri net based analysis of information flow. *Annals of Telecommunications, pages 1314–1346.* 2006.

Rakkay, Hind. 2009. Approches formelles pour la modélisation et la vérification du contrôle d'accès et des contraintes temporelles dans les systèmes d'information. *Thèse de doctorat.* avril 2009.

Rice, G. and J. Davis. 2002.. a genealogical approach to analysing post-mortem. 2002.

SERES, equipe. 2008. http://orbac.org. [Online] 2008.

Shafiq, B., et al. 2005. A role-based access control policy verification framework for real-time systems. *Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, Washington, DC, USA.* 2005.

Sheyner., O. M. 2004. Scenario graphs and attack graphs, . *Thesis of School of Computer Science, Computer Science department, Carnegie Mellon University, Pittsburgh, PA,.* 2004.

Stal, D., Tavares, S. and Meijer, H. 1994. State Analysis of Cryptographic Protocol using Colored Petri Nets. *Backward Proc. of workshop on selected areas in Cryptography,275-295.* 1994.

Stephenson, Peter. 2004. Modeling a virus or worm attack. 2004.

Tjaden, Brett C. 1997. A Method For Examining Cryptographic Protocols. *In Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy (Computer Science)University of Virginia.* 1997.

Toussaint, M,J. 1991. Verification of Cryptographic Protocols. *Thèse de PhD Université de Liége.* 1991.

U.S, department. 1985. U.S. Department of Defence Trusted Computer Security Evaluation Criteria, 5200-28-STD. 1985.

Vache, Géraldine. 2008. *Vers des mesures de la sécurité informatique Directeur(s) de thèse: Jean-Claude LAPRIE Laboratoire d'accueil: Laboratoire d'Analyse et d'Architecture des Systèmes Toulouse.* 2008.

Vache-Marconato, Géraldine. 2009. Evaluation quantitative de la sécurité informatique : approche par les vulnérabilités. *THESE DOCTORAT.* décembre 8, 2009.

Varadharajan, v. 1990. Petri net based modelling of information flow security requierements. *In The Computer Security Foundation Workshop III,p51-61, Franconia,USA, IEEE*. 1990.

Wang, Haining, Zhang, Danlu and Kang, G. 2004. Change-Point monitoring for the Detection of DOS Attacks. 2004. p. . VOL.1 N°4.

Wang, Yang, Lin, Chuang and Li, Quan-Lin. 2007. A queueing analysis for the denial of service (DoS) attacks in computer networks. 2007.

Wu, Ruoyu, Li, Weiguo and Huang, He. 2008. An Attack Modeling Based on Hierarchical Colored Petri Nets:. *IEEE.* 2008.

www.clusif.asso.fr. [Online]

www.Commentcamarche.net. Commentcamarche. *www.commentcamarche.net.* [Online]

Y.C.Woo, Thomas and Lam, Simon. 1993. A Semantic Model for Authentication Protocols. *IEEE.* 1993.

Zhang, Z.-L., Hong, F. and Liao, J.-G. 2006. Modeling chinese wall policy using colored petri nets. *In CIT '06: Proceedings of the Sixth IEEE International Conferenceon Computer and Information Technology, Washington.* 2006.

Zhang, Z.-L., Hong, F. and Xiao, H.-J. 2006a. Verification of strict integrity policy via petri nets. *In ICSNC '06: Proceedings of the International Conference on Systems and Networks Communication,Washington IEEE.* 2006a.

Annexe

Dans cette annexe nous allons présenter un exemple de combinaison d'un arbre des fautes, utilisé traditionnellement pour l'analyse de fiabilité des systèmes, et d'un arbre d'attaques, utilisé pour étudier les modèles d'attaques malveillants. Le but est de calculer la probabilité d'occurrence de l'évènement Top de l'arbre des fautes en prenant en considération les deux arbres et par conséquence deux type d'évènements, l'évènement d'échec aléatoire et l'évènement résultant d'actes délibérés malveillants (Nai Fovino, et al., 2009).

Le modèle de l'arbre d'attaque adopté dans cet exemple, considère trois types de nœuds : le nœud représentant les vulnérabilités que l'attaquant peut exploiter et qui est représenté par une éclipse, le nœud représentant les opérations que l'attaquant peut effectuer et qui est représenté par un hexagone et le nœud représentant les assertions logiques représenté par un rectangle.

Considérons l'arbre de Fautes de la figure 1 qui illustre en racine un événement non désiré « Top », logiquement connecté aux évènements plus bas (E1-E7). L'événement supérieur « Top » considéré dans l'exemple est la libération d'une substance toxique dans l'environnement par une usine chimique ; cet événement peut se produire si un tuyau (pipe) se casse en raison d'une forte pression et de l'échec du système de protection automatique et des commandes d'arrêt à distance, en même temps que l'échec du système des réservoirs centenaires. Les étiquettes soulignées dénoteront les probabilités qui sont connues au début du procédé, tandis que les étiquettes non soulignées représentent des probabilités à calculer

Les probabilités des évènements E4, E5, E6 et E7 sont initialement connues; l'évènement E3 est un résultat possible de l'attaque A, dans le but de faire les calculs d'analyses de l'arbre des fautes il est nécessaire de calculer ou d'estimer la probabilité d'occurrence de l'évènement E3. Ce but est atteint par le moyen de l'analyse probabiliste de l'arbre des attaques.

Supposons que l'arbre d'attaque de la Fig. 2. décrit le modèle des attaques possibles qui causent l'évènement E3. L'évènement E3 est lié à l'échec du processus d'arrêt à distance. Cet évènement peut être provoqué par un attaquant de plusieurs manières.

La plus évidente est qu'un attaquant essaye de couper la connexion réseau du serveur à distance, pour l'empêcher de fournir la commande d'arrêt appropriée. Par exemple, si nous supposons que le serveur est basé sur le système d'exploitation linux avec Kerne2.6 et avec une table d'IP active, un attaquant - en envoyant simplement quelques paquets mal formés ad hoc - pourrait empêcher n'importe quelle connexion réseau (en exploitant un bug bien connu contenu dans la table IP sur les systèmes kernel 2 .6).

Une autre attaque possible est l'attaque DDoS, si nous supposons que le serveur est accessible à partir des réseaux avec des ‘‘sous réseaux zombies’’, l'attaquant contrôlant les zombis peut arrêter la connexion du serveur en consommant la bande passante, simplement en exécutants une attaque DDoS.

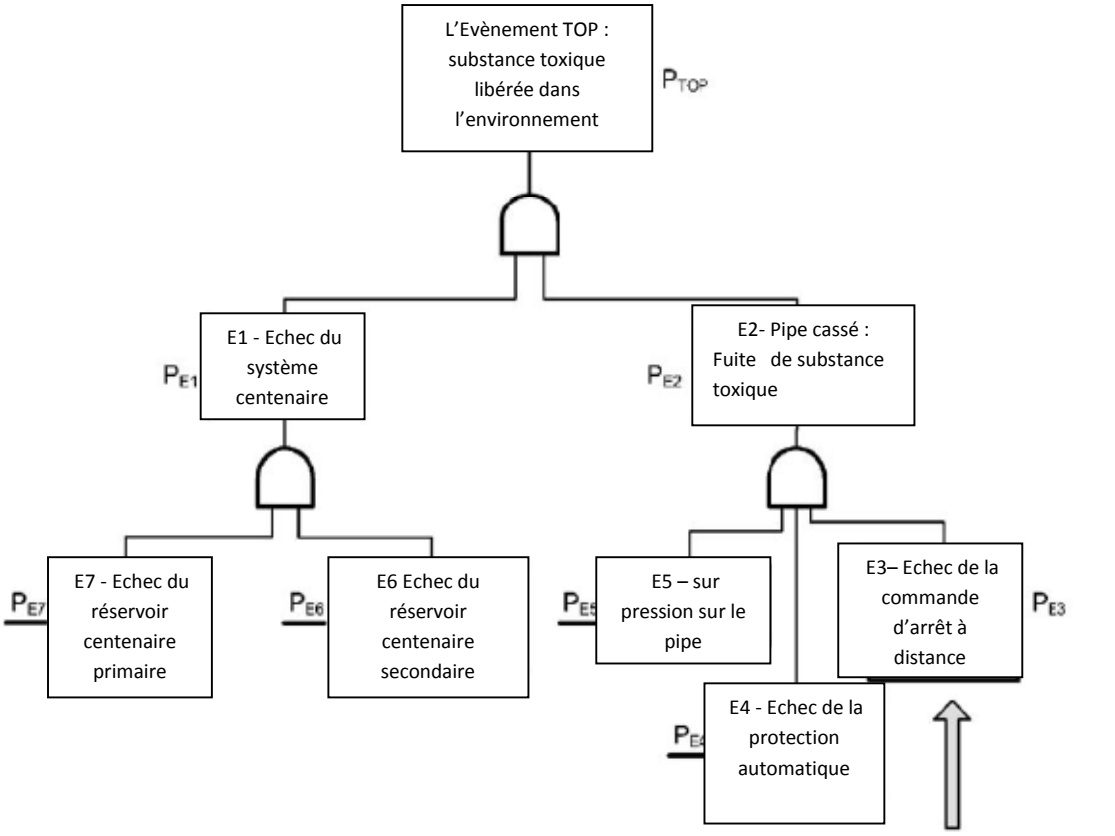


Fig. 1. Arbre des fautes

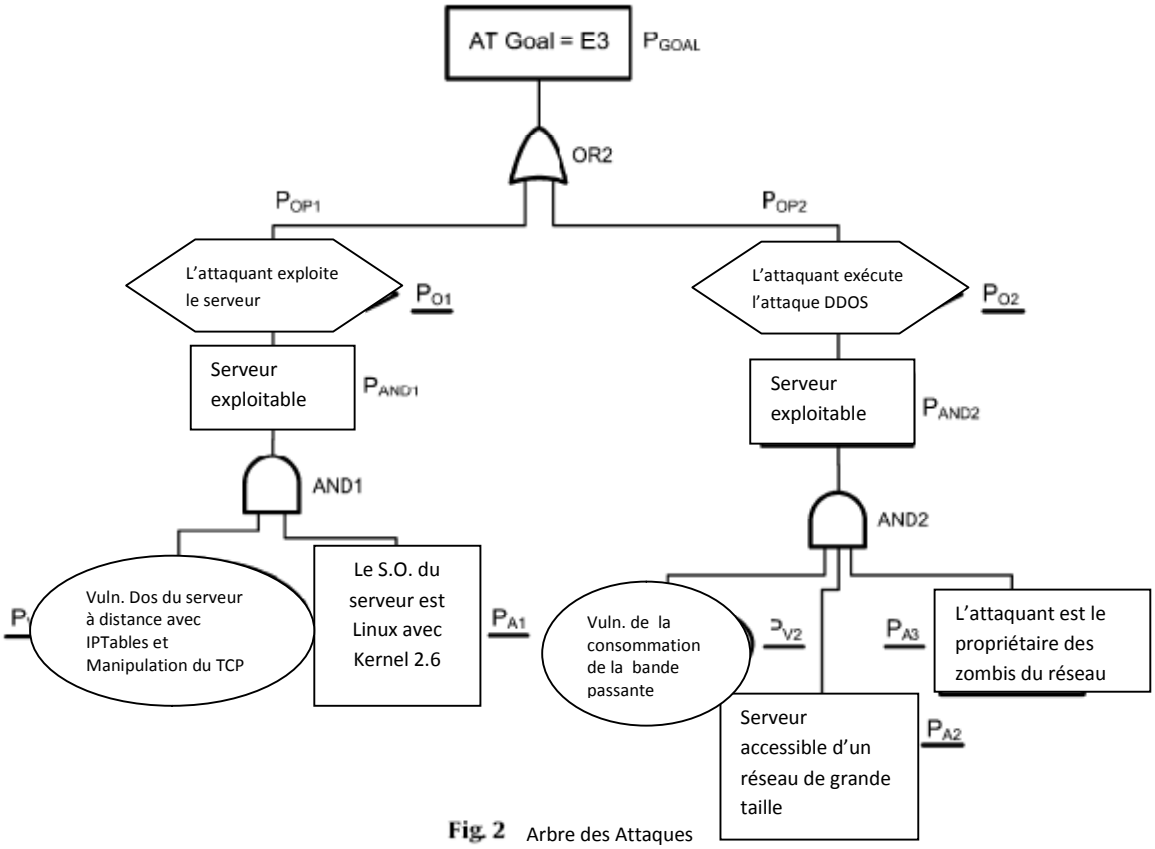


Fig. 2 Arbre des Attaques

Il est possible de calculer la probabilité associée à la racine de l'arbre des attaques exactement de la même façon que pour l'arbre des fautes. La différence entre les deux arbres c'est les nœuds d'opération P_{op1} et $P_{op2.}$ En effet dans l'arbre des fautes, on ne connait initialement que les probabilités d'occurrence des nœuds feuilles, alors que dans l'arbre des attaques on prend en considération aussi l'information sur les probabilités P_{op1} et P_{op2}, probabilités que l'attaquant va réellement réaliser ces opérations, les nœuds correspondants ne sont pas des nœuds feuilles. Comme dans l'arbre des fautes les nœuds intermédiaires ne sont pas considérés, les nœuds opérations de l'arbre des attaques sont transformés en nœuds feuilles, la figure 3 illustre la procédure de transformation.

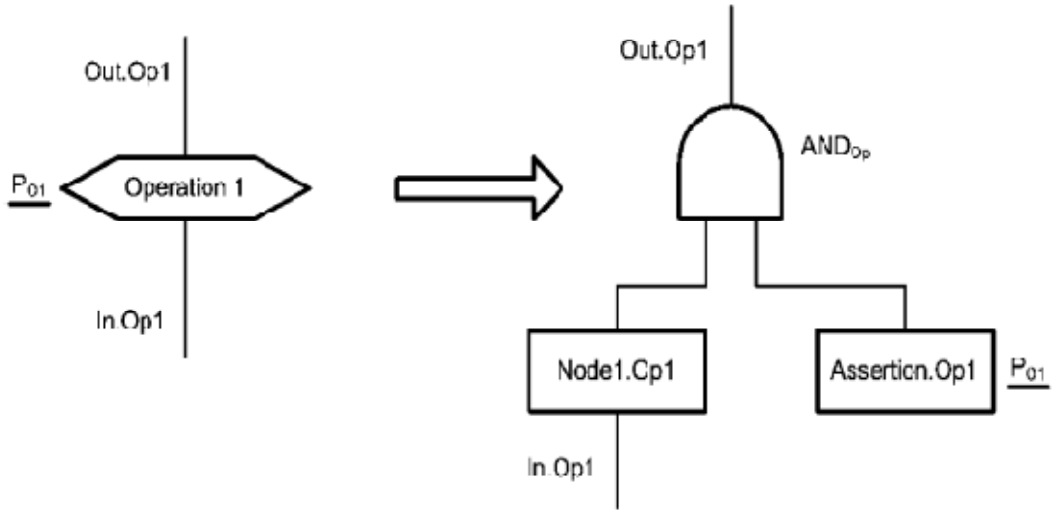


Fig. 3. Transformation des nœuds opérations

Le résultat de la transformation de l'arbre des attaques est montré dans la figure 4. Maintenant que les deux arbres ont une structure identique, nous pouvons intégrer l'arbre des attaques, qui est la cause potentielle de l'évènement E3, à l'arbre des fautes. Le résultat de cette intégration est illustré par la figure 5.

L'évaluation de la probabilité de l'occurrence de l'évènement TOP de l'arbre des fautes prendra en considération ainsi un évènement malicieux délibéré, en plus des évènements aléatoires d'échecs.

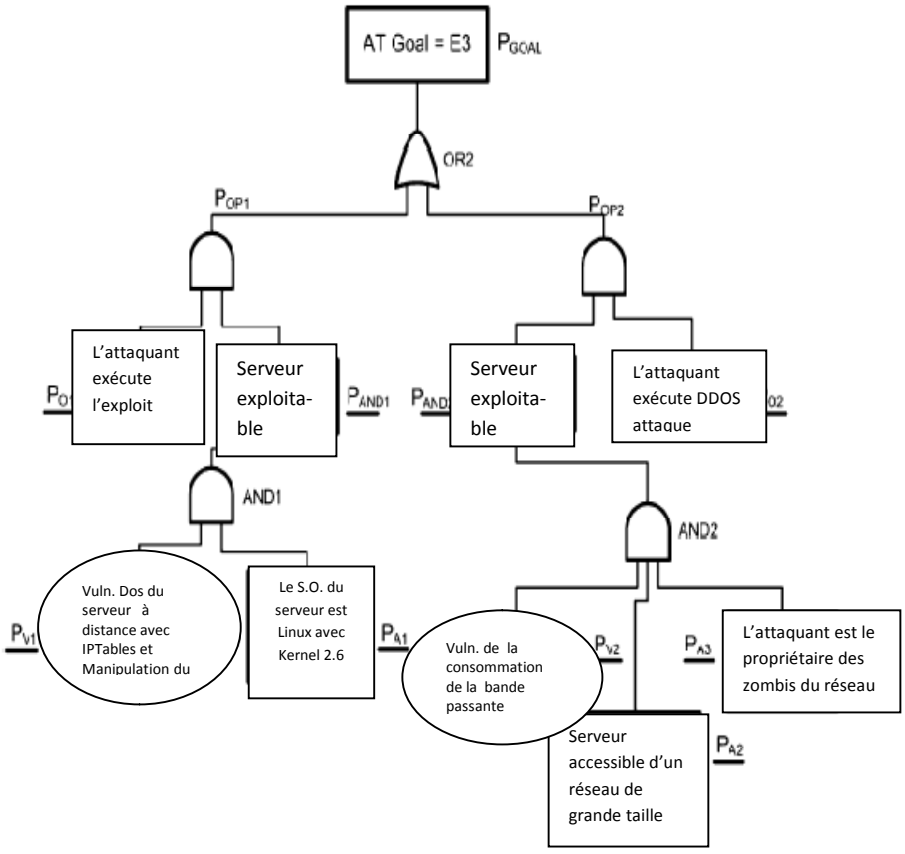


Fig 4. Arbre des attaques

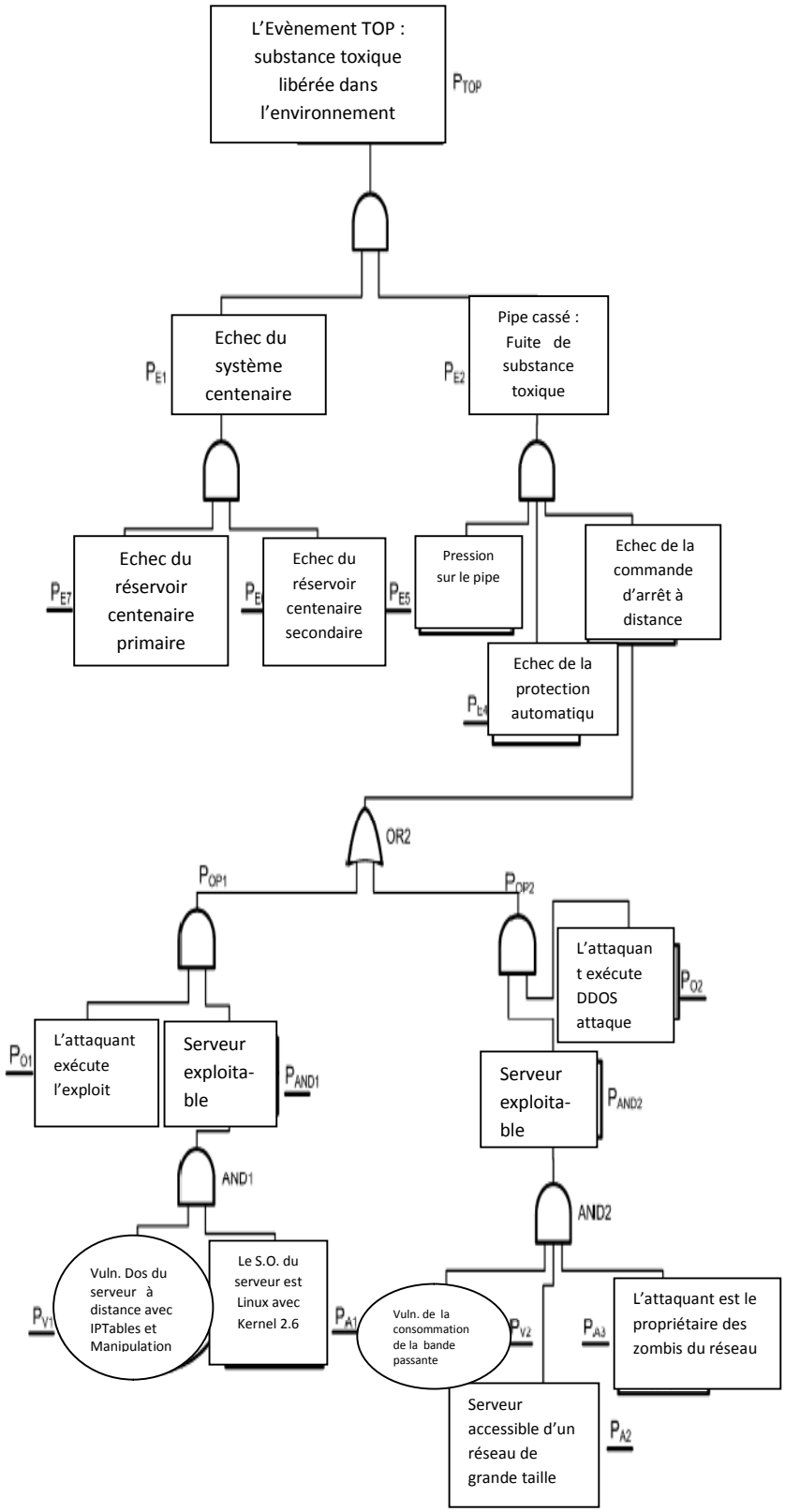


Fig5 : Intégration de l'arbre des fautes et des attaques